
Web-Mapping mit Open Source-GIS-Tools

Web-Mapping mit Open Source-GIS-Tools

Tyler Mitchell

*Überarbeitung von
Astrid Emde & Arnulf Christl*

*Deutsche Übersetzung von
Jørgen W. Lang*

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen.

Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im Wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:

O'Reilly Verlag

Balthasarstr. 81

50670 Köln

Tel.: 0221/9731600

Fax: 0221/9731608

E-Mail: kommentar@oreilly.de

Copyright der deutschen Ausgabe:

© 2008 by O'Reilly Verlag GmbH & Co. KG

1. Auflage 2008

Die Originalausgabe erschien 2005 unter dem Titel

Web Mapping Illustrated bei O'Reilly Media, Inc.

Die Darstellung einer Schnepfe im Zusammenhang mit dem

Thema Web-Mapping ist ein Warenzeichen von O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der

Deutschen Nationalbibliografie; detaillierte bibliografische Daten

sind im Internet über <http://dnb.ddb.de> abrufbar.

Deutsche Überarbeitung von Astrid Emde, Köln & Arnulf Christl, Bonn

Deutsche Übersetzung: Jørgen W. Lang, Oberlauterbach

Fachliche Unterstützung: Armin Retterath, Mendig & Nicol Hermann, Mainz

Lektorat: Alexandra Follenius, Köln

Korrektur: Sibylle Feldmann, Düsseldorf

Satz: Tim Mergemeier, Reemers Publishing Services GmbH, Krefeld; www.reemers.de

Umschlaggestaltung: Ellie Volckhausen, Bosten & Michael Oreal, Köln

Produktion: Karin Driesen, Köln

Belichtung, Druck und buchbinderische Verarbeitung:

Druckerei Kösel, Krugzell; www.koeselbuch.de

ISBN-13 978-3-89721-723-2

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Vorwort	IX
Einleitung	XIII
1 Einführung in die digitale Kartografie	1
Die Macht digitaler Karten	2
Die Herausforderungen beim Erstellen von Karten	2
Verschiedene Arten der Web-Kartografie	8
2 Aufgaben und Werkzeuge der digitalen Kartografie	17
Welche Aufgaben erwarten Sie beim Erstellen von Karten?	18
Was ist beim Umgang mit Geodaten zu beachten?	19
Ermittlung der Arbeitsschritte eines Projekts	23
3 Daten konvertieren und betrachten	27
Rasterdaten und Vektordaten	28
Quantum GIS	28
OpenEV	28
UMN MapServer	30
Die GDAL – Geospatial Data Abstraction Library	32
Die OGR Simple Features Library	37
PostGIS	40
Zusammenfassung der Anwendungen	42
4 MapServer installieren	43
So funktionieren MapServer-Anwendungen	44
Ein Blick auf die wichtigsten Bestandteile von UMN MapServer	45
Installation von MapServer	49
MapServer-Support	69

5	Bezugsquellen für Geodaten	71
	Bewertung des Datenbedarfs	71
	Beschaffung geeigneter Daten	76
6	Geodaten analysieren	85
	Download der Demodaten	85
	Installation von GDAL und FWTools	85
	Dateninhalte untersuchen	86
	Werkzeuge zum Zusammenfassen der Informationen	97
7	Geodaten konvertieren	109
	Geodaten konvertieren	109
	Vektordaten konvertieren	110
	Rasterdaten konvertieren	115
8	Geodaten in einem Desktop-GIS visualisieren	121
	Desktop-GIS-Programme	121
	Einführung in Quantum GIS (QGIS)	125
	Quantum GIS zur Visualisierung von Geodaten verwenden	128
	Erzeugen einer 3-D-Ansicht	141
9	Eigene Geodaten erzeugen und bearbeiten	149
	Planen Sie Ihr eigenes Projekt	149
	Daten mithilfe von Quantum GIS aufbereiten und erfassen	152
10	Erstellen statischer Karten – erster Einstieg in die MapServer-Mapdatei	167
	MapServer-Hilfsprogramme	168
	Die Kommandozeilenprogramme im Einsatz	169
	Das Format des Ausgabebilds bestimmen	190
11	Interaktive Karten im Web veröffentlichen	193
	MapServer einrichten und testen	194
	Anpassen des Startbereichs	204
	Weiterführende Informationen	229
12	Kartenzugriff über standardisierte Webdienste	231
	Kartendienste	232
	Nutzung von OGC-Webdiensten mit UMN MapServer	237
	OGC Web Map Service (WMS)	238
	OGC Web Feature Service (WFS)	253
	Mapdateien für den Einsatz als WMS und WFS	263

13	Kartendienste nutzen und verwalten	271
	Verschiedene Clients für Kartendienste	271
	OpenLayers – Karten nach Maß	274
	Mapbender – Kartenanwendungen nach Maß	281
	Mapbender-Anwendungen entwickeln	297
14	PostgreSQL/PostGIS – Einsatz von räumlichen Datenbanken	311
	Einführung in PostgreSQL/PostGIS	311
	Was ist eine räumliche Datenbank?	312
	Installation von PostgreSQL und PostGIS	315
	PostGIS einrichten Schritt für Schritt	318
	createdb – Anlegen einer räumlichen Datenbank	327
	Geodatenhaltung in PostGIS	328
	Geodaten selbst erzeugen	329
	Laden von Geodaten	332
	Räumliche Daten abfragen	336
	Nutzen der PostGIS-Daten in anderen Applikationen	349
15	Programmieren mit MapServer MapScript	357
	Was ist MapScript?	357
	Die MapScript-Objekte	359
	Beispiele für die Verwendung von MapScript	361
A	Eine kurze Einführung in die Kartenprojektionen	377
	Die Gestalt der Erde	377
	Geografische Koordinatensysteme	380
	Beispiele für Kartenprojektionen mit MapServer	396
	Vorstellung ausgewählter Kartenprojektionen	398
	Einsatz von Projektionen in anderen Anwendungen	406
	Weiterführende Informationen	407
B	MapServer-Referenz für den Zugriff auf Vektordaten	409
Index		441

Vorwort

Die Erstausgabe dieses Buchs kam im Jahr 2005 zeitgleich mit Google Earth auf den Markt. Die (GIS-)Welt hielt den Atem an – war das Ende der Welt jetzt abzusehen? Zehntausende von Geografen und GIS-Spezialisten arbeitslos, über Nacht degradiert zu vorsintflutlichen Software-Abenteurern, die noch wussten, wie 3 MByte große Dateien auf vier biegsame Datenträger (Floppys) verteilt werden mussten, um sie dem wissenschaftlichen Mitarbeiter zum Ausdrucken im Rechenzentrum mitzugeben. Lebende Anachronismen – und das ist noch gar nicht so lange her!

Was ist geschehen? Das Internet ist über uns gekommen, so langsam, dass wir es fast nicht bemerkt hätten, und gleichzeitig so schnell und weltumspannend wie keine Kulturtechnik je zuvor. Das erste Mal in der bekannten Menschheitsgeschichte ist es möglich, von jedem Ort auf der Welt zu jedem anderen Ort Kontakt aufzunehmen, zeitgleich und sofort. Eine unglaubliche Gleichschaltung ist im Gange, alles wird virtuell, Grenzen fallen, Konzerne operieren global, ehemals enorme Strecken werden durch Glasfaserleitungen bei knapp Lichtgeschwindigkeit zum Nichts. Der Raum, wie wir Alten ihn noch kannten, hat sich aufgelöst.

Was uns geblieben ist aus der Pionierzeit der Computerei, ist ein Symbol in der Werkzeugleiste der Software, das uns mahnt, wie schnell die Zeit vergeht. Sie wissen, wovon ich spreche? Oben genannter Datenträger, auch bekannt als »Diskette«, zierte noch immer als Ikone der Vergänglichkeit den Knopf, mit dem wir Dateien speichern. Denn dafür gibt es kein Symbol. Die Möglichkeit, etwas verlustfrei zu speichern, es für die (zumindest eingebildete) Ewigkeit unveränderlich aufzubewahren, ist ein Konzept, das unsere eigene Vergänglichkeit vergessen macht. Ebenso wie wir im Raum gefangen sind und täglich im Stau verhöhnt werden von den Informationsfetzen, die uns mit nahezu Lichtgeschwindigkeit überholen. Von wegen, es gäbe keinen Raum mehr und Distanzen seien geschrumpft ...

Karten biegen die Virtualität zurück auf den realen Raum; durch die viel beschworene digitale Autobahn sind sie zu sprichwörtlichem neuem Leben erwacht und mehr denn je ein Werkzeug des Alltags geworden. Sei es als animierte Weltkugel, die über unsere Computer-Arbeitsflächen rollt oder als Navigations-Helferlein in unserem heimlichen

Wohnzimmer, dem Automobil. Karten ermöglichen es, uns virtuell aus der Enge unserer eigenen physischen Trägheit zu erheben. Blitzschnell zaubern wir andere Länder auf unseren Bildschirm und fragen Daten ferner Orte ab.

Was hat das alles mit dem Buch zu tun, das Sie gerade in Ihren Händen halten? In dem vorliegenden Werk finden Sie eine gelungene Mixtur aus Einsteigerlektüre bis hin zu Extremistenspezialistenwissen rund um digitale Karten. Egal ob Sie dieses Wissen für Ihre Arbeit brauchen oder fürs Studium, ob Sie eine Karte in die eigene Webseite integrieren möchten oder ob Sie einfach mal gern jenseits der ausgetrampelten Pfade schnöder Straßen- und Luftbildkarten durch die Welt surfen möchten – hier werden Sie fündig. Das Buch aber will mehr, es sieht Sie nicht als reinen Konsumenten, sondern will Sie befähigen, selbst zum Kartenautor zu werden. Die vorgestellten Technologien sind komplex und werden täglich im professionellen Kontext genutzt, sind aber durchaus beherrschbar geblieben. Viel wichtiger ist jedoch, dass Sie alle Open Source-Komponenten vollkommen legal und ohne doppelten Boden verwenden können, weil sie als Freie Software lizenziert sind.

Ich traf Tyler Mitchell zum ersten Mal persönlich auf der OSGIS-Konferenz 2004 in Ottawa, Kanada (<http://omsug.ca/osgis2004/program.html>). Zu der Zeit nahm dieses Buch erste Formen an. Im Erscheinungsjahr 2005 war ich auch wieder auf der Map-Server-Konferenz, und wir überlegten gemeinsam, ob es wohl einen deutschsprachigen Markt gäbe. Google Earth, VirtualEarth, WorldWind und all die anderen haben es zur gleichen Zeit geschafft, geografische Daten und Karten populär und massentauglich zu machen.

Anfang 2006 wurde die Non-Profit Open Source Geospatial Foundation (<http://www.osgeo.org>) gegründet. OSGeo unterstützt und fördert Open Source-Projekte in der räumlichen Datenverarbeitung auf organisatorischer, legaler, technischer und finanzieller Ebene. Ende 2006 wurde Tyler Mitchell zum Executive Director der OSGeo bestellt und hat seitdem die Organisation maßgeblich mit konsolidiert und auf ein solides Fundament gesetzt. Inzwischen verlassen sich Hunderte von Entwicklern auf die technischen Infrastrukturen der OSGeo (im Oktober 2007 genau 185), um qualitativ hochwertige Software zu implementieren und zu pflegen. Tausende von Anwendern profitieren davon und verbreiten die Technologien auf allen Ebenen der Industrie, Verwaltung, Forschung und Bildung.

In der Zwischenzeit ist also viel passiert, und es wurde Zeit, das Buch zu überarbeiten. Die deutsche Übersetzung war Anlass, das zu tun. Astrid Emde hat mit unendlich viel Detailwissen die Inhalte auf den neuesten Stand gebracht, erweitert und ergänzt. Gemeinsam haben wir den Fokus mehr in Richtung offener und standardisierter Schnittstellen verschoben, ein weiteres Tribut an die immer bessere Vernetzung. Zwei Projekte haben wir ganz neu hinzugenommen, das Desktop-GIS Quantum GIS und die webbasierte Entwicklungsplattform Mapbender, die vor allem im deutschsprachigen Raum weit verbreitet ist.

Viele tausend Menschen verdienen inzwischen ihr täglich Brot mit Dienstleistungen rund um Freie und Open Source-Software, es ist ein stetig wachsender Markt. Aber noch immer gibt es auch große Unsicherheit, ob es denn bei Open Source mit rechten Dingen zugehe – und das, obwohl Urgesteine wie die Firma IBM längst die Vorteile erkannt haben und für sich wirtschaftlich nutzen. Wenn dieses Buch in der Lage ist, die Vielfalt, Zuverlässigkeit und Leistungsfähigkeit der Open Source-Software im Geodatenbereich zu dokumentieren, dann hat sich die Arbeit gelohnt.

Viel Spaß bei der Lektüre wünscht

Arnulf Christl

Was macht Karten so besonders? Einige von uns sind von Karten fasziniert, egal wo wir uns gerade befinden. Ich selbst habe schon Stunden damit verbracht, Karten zu studieren und dadurch fremde Orte zu entdecken und kennenzulernen. Dabei haben Karten immer noch etwas Geheimnisvolles. Sie enthalten Informationen, die sich nur durch ihre Erforschung erschließen lassen.

Digitale Karten gehen hierbei sogar noch einen Schritt weiter, indem sie die *Interaktion* mit dem Benutzer ermöglichen. Traditionelle statische Karten besaßen diese Fähigkeit nicht. Mittlerweile können Benutzer die angezeigten Informationen selbst steuern und die Karte so an ihre eigenen Bedürfnisse anpassen.

Entdeckungen in der Kindheit

Meine erste Erfahrung mit einer Karte machte ich schon als kleiner Junge. Ich erinnere mich noch gut an die Aufregung, als ich unser Pfadfinderlager auf einer topografischen Karte entdeckte. Einen Teil dieser Karte sehen Sie in Abbildung 0-1. Ich fand meine Heimatstadt, Straßen und Wege und sogar Gewächshäuser. Es fiel mir schwer, zu glauben, dass sich jemand die Mühe machte, die Straßen, auf denen ich spielte, und die Tabakfelder, in denen ich während der Sommerferien arbeitete, auf einer Karte einzzeichnen. Und trotzdem waren sie da, eingezeichnet auf der Karte, die nun, 20 Jahre später, an der Wand meines Büros hängt.

Diese Karten waren der Schlüssel für die Planung von Wanderungen und Fahrradausflügen. Ich mache Karten dafür verantwortlich, dass ich mich weiter und weiter von zu Hause entfernte, nur um zu sehen, wie ein Symbol auf der Karte oder eine bestimmte Stadt in Wirklichkeit aussah.

Wenn ich nicht gerade auf Entdeckungsreise war, saß ich oft am Computer und lernte Neues über die *digitale* Welt. Dieses kombinierte Interesse an Computern und Entdeckungen führte mich direkt in die Welt computerbasierter Kartografie und geographischer Informationssysteme (GIS). Ich hätte nie geahnt, dass meine Faszination von Karten und Computern jemals zu meinem Beruf werden könnte.

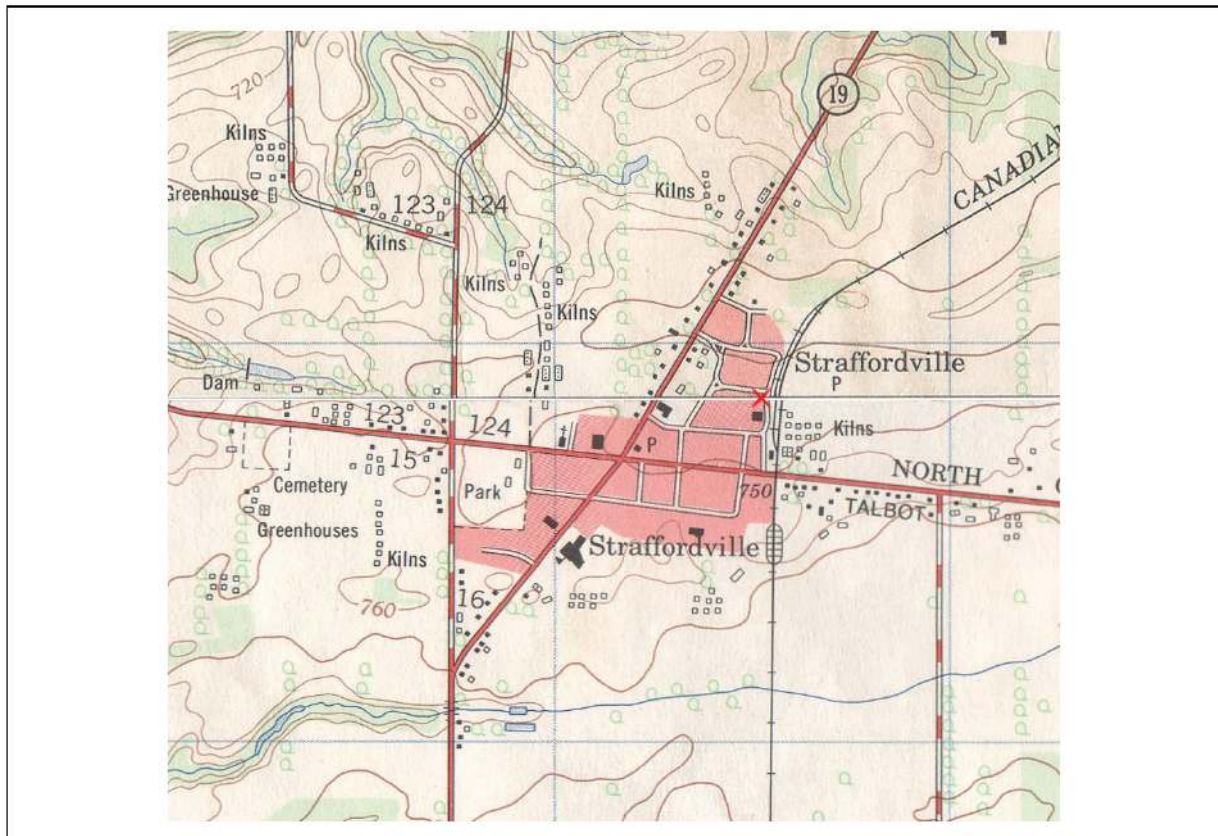


Abbildung 0-1: Teil einer topografischen Karte meiner Heimatstadt in Süd-Ontario, Kanada. Mein Haus stand dort, wo sich das rote X befindet. (Ausschnitt der NTS-Karten 40I/15c und 10c/15c ©2005. Produced under licence from Her Majesty the Queen in Right of Canada, with permission of Natural Resources Canada.)

Karten spielen im Leben eine wichtige Rolle – egal ob es nun darum geht, einem Freund zu zeigen, wo Sie wohnen, oder die Zugroute eines Hurrikans nachzuvollziehen. Die Werkzeuge und Möglichkeiten, Ihre ganze Welt in eine Karte umzuwandeln, verleihen große Macht. Hierzu ein Zitat aus dem Mayan-Atlas:

Karten stehen für Macht. Entweder Sie kartieren, oder Sie werden kartiert. Wenn Sie von denjenigen kartiert werden, die Kontrolle über Ihr Land und Ihre Ressourcen erlangen wollen, wird die Karte die Rechtfertigungen für deren Ansprüche zeigen – und nicht für Ihre.

Der offene Zugang zu GIS-(Geoinformationssystem-) und Mapping-Werkzeugen bedeutet eine zusätzliche Steigerung Ihrer kartografischen Möglichkeiten. Die Aussicht darauf, dass Sie Ihre Karten mit der Welt teilen können, macht den Aufwand noch lohnenswerter.

Werkzeuge in diesem Buch

Die in diesem Buch vorgestellten und verwendeten Werkzeuge werden als Open Source-Projekte entwickelt und sind als Freie Software lizenziert. Sie sind mein digitaler »Werkzeugkasten« für professionelles Datenmanagement. Ob es nun um die Visualisie-

rung von Geodaten oder die Konvertierung zwischen verschiedenen Formaten geht – viele dieser Werkzeuge kommen bei mir täglich zum Einsatz. Gleichzeitig sind sie Teil meiner Zukunftspläne in Bezug auf Kartografie und Datenverwaltung.

Diese Werkzeuge bilden eine kleine Auswahl der derzeit verfügbaren Programme. Die Open Source-Werkzeuge können Sie frei verwenden und bei Bedarf an Ihre Bedürfnisse anpassen. Viele dieser Programme übertreffen die Fähigkeiten vergleichbarer proprietärer Produkte.

Kurz nach meiner Studienzeit begann ich, viele dieser Werkzeuge in Gebrauch zu nehmen. Meine damalige Arbeit bestand in der Erstellung von Karten und der Analyse kartografischer Daten. Ich hatte das Glück, dabei die neuesten kommerziellen Werkzeuge einsetzen zu können. Wenn ich aber in meiner Freizeit zu Hause an einem Projekt weiterarbeiten wollte, standen die kommerziellen Werkzeuge einfach nicht zur Verfügung. Die Lizenzbeschränkungen und Kosten zwangen mich, Alternativen zu finden. Dadurch nahmen die Open Source-Werkzeuge im Laufe der Zeit immer mehr Platz ein. Sollten in meinem Werkzeugkasten noch Teile fehlen, werden diese sicherlich in naher Zukunft über Open Source-Projekte realisiert.

Die Entwicklergemeinschaft für Open Source-basierte Mapping- und GIS-Projekte ist äußerst aktiv. Viele dieser Projekte verwenden die neuesten Standards für Interoperabilität und implementieren diese tendenziell schneller als kommerzielle Produkte.

Der ursprüngliche Zweck dieses Buchs war, die Lücken in der existierenden Dokumentation zu schließen und die am häufigsten vorkommenden Einsteigerfragen zu beantworten. Dieses Buch geht nun sogar noch darüber hinaus. Ich hoffe, Sie können meine Begeisterung für die Werkzeuge, die ich Ihnen vorstellen werde, teilen. Durch die jahrelange Programmierarbeit an den einzelnen Open Source-Projekten steht uns nun eine mächtige Palette an Werkzeugen zur Datenerfassung, Visualisierung, Datenverwaltung und sogar für kindliche Entdeckungen zur Verfügung.

Was erwartet Sie in diesem Buch?

Dieses Buch stellt eine Reihe von Konzepten und Werkzeugen vor. Diese lassen sich in die folgenden fünf Bereiche unterteilen:

- Konzepte für die Kartenerstellung und Datenverwaltung
- Kommandozeilenbasierte Werkzeuge zur Datenverwaltung
- Kommandozeilen- und webbasierte Werkzeuge zur Erzeugung von Karten
- Programme, die standardisierte Dienste unterstützen (OGC-Dienste)
- Verwaltung von Geodaten in räumlichen Datenbanken

Sie werden in diesem Buch die folgenden Werkzeuge kennenlernen:

Geospatial Data Abstraction Library (GDAL) mit OGR

Dieses Werkzeug enthält Programmierschnittstellen (APIs) und kommandozeilenbasierte Hilfsprogramme für die Arbeit mit Raster- und Vektordaten. Die GDAL-Website finden Sie unter <http://www.gdal.org>.

Quantum GIS (QGIS)

Für einfache Desktop-GIS-Funktionalitäten. *Quantum GIS* ermöglicht die Visualisierung von Geodaten und deren Bearbeitung.

OpenEV

Für einfache Desktop-GIS und Bildanalyse. *OpenEV* enthält Werkzeuge zur Erstellung neuer Karten, die in anderen Programmen verwendet werden können.

UMN MapServer

Zu diesem Werkzeug gehören Hilfsprogramme zur Erstellung CGI-basierter Webanwendungen. Mithilfe der MapServer-API mit dem Namen MapScript lassen sich auf einfache Weise eigene Kartenanwendungen erstellen.

Mapbender

Mapbender ist eine Client-Suite, mit der Sie Dienste, die nach den OGC-Standards aufgebaut sind, ansprechen können. Mapbender unterstützt dabei WMS, WFS und WMC. Mit Mapbender können Sie leicht individuelle Oberflächen erstellen und darüber WMS-Dienste ansprechen.

OpenLayers

Mit OpenLayers können Sie leicht dynamische Karten in eine Webseite einbinden. OpenLayers ist ein ThinClient, der neben den OGC-Standards WMS und WFS auch noch weitere Formate wie GoogleMaps und ka-Map unterstützt.

PostGIS

PostGIS ist die räumliche Erweiterung der PostgreSQL-Datenbank. Die Kombination PostgreSQL/PostGIS ermöglicht Ihnen, räumliche Daten und alphanumerische Daten gemeinsam speichern und bearbeiten zu können.

Wie ist dieses Buch aufgebaut?

Dieses Buch enthält 15 Kapitel und zwei Anhänge:

Kapitel 1, Einführung in die digitale Kartografie

Dieses Kapitel gibt einen Überblick über die digitale Kartografie, inklusive der webbasierten, und zeigt einige der Hürden bei der Verwendung dieser Technologie auf. Es enthält außerdem eine Liste mit Links zu einer Reihe von Kartenanwendungen und gibt eine kurze Übersicht über die eingesetzte Technologie.

Kapitel 2, Aufgaben und Werkzeuge der digitalen Kartografie

Dieses Kapitel umreißt die Ziele digitaler Kartografie und die typischerweise damit verbundenen Aufgaben, wie die Betrachtung, Analyse, Erstellung, Manipulation, Konvertierung und Weitergabe digitaler Daten.

Kapitel 3, Daten konvertieren und betrachten

Dieses Kapitel stellt die Konzepte vektor- und rasterbasierter Datentypen vor. Für den Umgang mit diesen Formaten lernen Sie Programme des GDAL/OGR-Projekts kennen.

Kapitel 4, MapServer installieren

Dieses Kapitel stellt die Hauptbestandteile einer UMN MapServer-Anwendung vor. Sie finden hier detaillierte Anweisungen für die Installation von MapServer anhand vorkonfigurierter Pakete und zur Kompilierung von MapServer aus dem Quellcode. Außerdem finden Sie einige Links, die Ihnen weiterhelfen können, wenn Sie Support benötigen.

Kapitel 5, Bezugsquellen für Geodaten

In diesem Kapitel geht es darum festzustellen, welche Daten Sie für Ihre Aufgabe benötigen und wie diese beschaffen sein müssen. Außerdem finden Sie hier Hinweise zu freien Geodaten und eine Liste, über die Sie frei verfügbare Geodaten bekommen können.

Kapitel 6, Geodaten analysieren

Dieses Kapitel beschäftigt sich mit der Einrichtung des FWTools-Pakets und der Verwendung der GDAL/OGR-Hilfsprogramme zur Untersuchung von Raster- und Vektordaten. Außerdem finden Sie hier Beispiele, in denen diese Werkzeuge mit kommandozeilenbasierten Textverarbeitungsprogrammen kombiniert werden, um individuelle angepasste Ausgaben in Form von Berichten und Zusammenfassungen zu erstellen.

Kapitel 7, Geodaten konvertieren

In diesem Kapitel geht es um die Konvertierung von Raster- und Vektordaten in verschiedene Formate mithilfe der GDAL/OGR-Werkzeuge. Sie werden lernen, wie Sie zwischen verschiedenen Formaten wie ESRI Shape, GML, DGN und PostGIS konvertieren können.

Kapitel 8, Geodaten in einem Desktop-GIS visualisieren

Dieses Kapitel stellt eine Reihe von Open Source-Desktop-GIS-Programmen kurz vor. Das Open Source-Desktop-GIS Quantum GIS wird dabei genauer vorgestellt. Mithilfe von Quantum GIS lernen Sie den Umgang mit verschiedenen Datenformaten kennen. Anhand des Desktop-GIS OpenEV wird aufgezeigt, wie 3-D-Ansichten erstellt werden können.

Kapitel 9, Eigene Geodaten erzeugen und bearbeiten

Dieses Kapitel zeigt, wie Sie eigene Daten erstellen können und was Sie bei der Erzeugung beachten müssen. Unter Verwendung von Quantum GIS werden neue Geodaten digitalisiert und in einer Shape-Datei gespeichert. Außerdem wird die Georeferenzierung von Rasterdaten vorgestellt.

Kapitel 10, Erstellen statischer Karten – erster Einstieg in die MapServer-Mapdatei

In diesem Kapitel lernen Sie die UMN MapServer-Mapdatei (*.map) kennen. Hier zeigen wir die Möglichkeiten, die die kommandozeilenbasierten MapServer-Werk-

zeuge bei der Erstellung von Kartenbildern, Maßstabsleisten und Legenden bieten. Sie werden die Mapdatei verwenden, um farbige Karten mit den passenden Beschriftungen zu erstellen.

Kapitel 11, Interaktive Karten im Web veröffentlichen

In diesem Kapitel geht es um die Verwendung von MapServer zusammen mit einem Webserver. Aufbauend auf Kapitel 10, wird die zuvor erstellte Anwendung über eine Webseite zugänglich gemacht. Sie lernen, wie Sie einen einfachen HTML-Client aufbauen können. Außerdem werden Ihnen weitere Konfigurationsmöglichkeiten, die die Mapdatei bietet, auf Grundlage der derzeit aktuellen MapServer-Version 5.0 vorgestellt.

Kapitel 12, Kartenzugriff über standardisierte Webdienste

In diesem Kapitel wird das Konzept der Webdienste und die Spezifikationen des Open Geospatial Consortium (OGC) vorgestellt. Das Hauptaugenmerk liegt auf dem Web Map Service (WMS) und dem Web Feature Service (WFS). Außerdem wird aufgezeigt, wie die MapServer-Mapdatei erweitert wird, damit UMN MapServer als WMS/WFS Server bzw. Client verwendet werden kann.

Kapitel 13, Kartendienste nutzen und verwalten

In diesem Kapitel werden verschiedene WMS-Clients vorgestellt. Es wird aufgezeigt, welche Möglichkeiten der Einsatz von OGC-Standards bietet. Am Beispiel der OSGeo-Projekte OpenLayers und Mapbender wird der Umgang mit WMS und WFS aus zwei Perspektiven erklärt.

Kapitel 14, PostgreSQL/PostGIS – Einsatz von räumlichen Datenbanken

Dieses Kapitel stellt die PostGIS-Erweiterung für die PostgreSQL-Datenbank vor. Sie finden hier eine Installationsanleitung. Es wird beschrieben, wie Sie Ihre Geodaten in die Datenbank überführen können, wie Sie SQL-Abfragen erstellen, wie PostGIS-Funktionen eingesetzt werden können und wie Sie PostGIS-Datenquellen in MapServer-Anwendungen ansprechen können.

Kapitel 15, Programmieren mit MapServer MapScript

Dieses Kapitel befasst sich mit MapScript. Hierbei wird die Installation bzw. die Kompilierung von MapScript für verschiedene Programmiersprachen behandelt. Anschließend werden die wichtigsten MapScript-Objekte vorgestellt. Anhand einiger Skriptbeispiele und der passenden Mapdatei wird gezeigt, wie Sie MapScript einsetzen können.

Anhang A, Eine kurze Einführung in die Kartenprojektionen

In diesem Anhang geht es um Koordinatenreferenzsysteme und Projektionen. Es werden die wichtigsten Projektionssysteme und ihre Verwendung vorgestellt. Eine Reihe von Abbildungen und Beispielen für die notwendigen Einstellungen in der Mapdatei ergänzen die Informationen. Außerdem wird die Verwendung der EPSG-Codes erklärt.

Anhang B, MapServer-Referenz für den Zugriff auf Vektordaten

Dieser Anhang beschreibt verschiedene Vektordatenformate umfassend. Für jedes Format wird ein Beispiel aufgeführt, über das deutlich wird, wie das Format über die MapServer-Mapdatei angesprochen werden kann.

Informationen zur deutschen Ausgabe

Dieses Buch ist eine umfassende Überarbeitung der englischen Ausgabe des Buchs *Web Mapping Illustrated* von 2005. Die Kapitel 8 und 9 wurden neu geschrieben und stellen Desktop-GIS anhand von Quantum GIS unter Verwendung von Daten aus Deutschland vor. Das Kapitel 13 zu standardisierten Kartenanwendungen wurde neu hinzugefügt und soll die Bedeutung von Anwendungen, die OGC-Standards verwenden, hervorheben. Der Trend geht eindeutig in Richtung Standardisierung und damit hin zu OGC WMS und OGC WFS. Das Buch behandelt weiterhin den Aufbau von UMN MapServer-HTML-Clients und den Einsatz von MapServer MapScript, obwohl deren Bedeutung im Vergleich WMS/WFS mehr und mehr in den Hintergrund tritt.

Beachten Sie, dass in dem Buch oft von MapServer die Rede ist. Hiermit ist der UMN MapServer gemeint und nicht der Begriff Mapserver, der im deutschen Sprachgebrauch für Kartenserver steht. Im Englischen wird der UMN MapServer, der aus einem Projekt der University of Minnesota (UMN) entstanden ist, lediglich als MapServer angesprochen.

Ein großer Dank gilt Armin Retterath für die Überarbeitung des Anhangs A zum Thema Projektionen. Dank an Nicol Hermann für die Überarbeitungen und Beiträge zum MapScript-Kapitel. Einen Dank an die Kollegen der WhereGroup und an Marco Hugentobler vom Quantum GIS-Projekt. Und auch ein großes Dankeschön an Tyler Mitchell für die gute Zusammenarbeit.

CD-ROM und Codebeispiele

Dem Buch liegt die Live-CD WebGIS.rlp bei, die im Rahmen des Aufbaus der Geodateninfrastruktur Rheinland-Pfalz erstellt wurde. Die WebGIS.rlp-CD beinhaltet Quantum GIS, GRASS, Mapbender, UMN MapServer, MapStorer, GeoServer und PostgreSQL mit PostGIS sowie PgAdmin3. Neben der Software liegen Testdaten auf der CD vor, die in den Kapiteln 8 und 9 verwendet werden. Mit der Live-CD können Sie die beschriebenen Programme und die Übungen direkt ausprobieren. Diese CD wurde freundlicherweise von der Initiative zum Aufbau der Geodateninfrastruktur Rheinland-Pfalz (GDI-RP) zur Verfügung gestellt.

Zu diesem Buch gibt es eine Website von O'Reilly, auf der Sie Beispiele und weitere Informationen finden können:

<http://www.oreilly.de/catalog/webmappingger/>

Der Beispiel-Link auf dieser Katalogseite führt Sie zu der Website der Autoren, auf der die in diesem Buch abgedruckten Codebeispiele und Beispieldaten zum Herunterladen zur Verfügung stehen. Sie können die Website der Autoren aber auch direkt unter dieser Adresse erreichen:

<http://www.webmappingillustrated.com/beispiele>

Auf dieser Website finden Sie außerdem zahlreiche hilfreiche Links rund um das Web-Mapping und Open Source-Software.

Typografische Konventionen

Kursivschrift wird verwendet für:

- neu eingeführte Begriffe und Hervorhebungen
- Pfad-, Datei- und Programmnamen
- sowie für Rechner- und Domainnamen (z.B. *<http://www.maptools.org>*).

Nichtproportionalschrift wird verwendet für:

- Codebeispiele und -fragmente
- alle Dinge, die in einem XML-Dokument vorkommen, inklusive Elementnamen, Tags, Attributwerten, Entity-Referenzen und Verarbeitungsanweisungen sowie
- alle Dinge, die in einem Programm vorkommen, inklusive Schlüsselwörtern, Operatoren, Methodennamen, Klassennamen, Hilfsprogrammen und Literalen

Nichtproportionalschrift fett wird verwendet für:

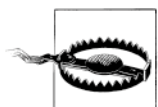
- Benutzereingaben
- Hervorhebungen in Codebeispielen und -fragmenten

Nichtproportionalschrift kursiv wird benutzt für:

- ersetzbare Elemente im Code



Dieses Icon steht für einen Tipp, einen Vorschlag oder eine allgemeine Anmerkung.



Dieses Icon kennzeichnet eine Warnung.

Dateinamen und Befehle, bei denen die Groß-/Kleinschreibung berücksichtigt werden muss, machen es teilweise nötig, dass von der üblichen Rechtschreibung abgewichen wird. In der Regel ist es möglich, den Satz so zu formulieren, dass es keine Konflikte gibt. Wo dies möglich war, habe ich mich bemüht, entsprechend vorzugehen. In den seltenen Fällen, in denen das nicht möglich war, habe ich zugunsten der korrekten Schreibweise des Befehls entschieden.

Die meisten Codebeispiele sind so angelegt, dass Sie sie direkt verwenden können. Es ist natürlich möglich, dass einige Beispiele nicht sofort funktionieren, weil die Umgebung anders ist (ein anderes Betriebssystem, eine neuere Version der Software usw.). Sie können sich dann immer an die entsprechende Mailingliste wenden oder Foren nutzen, um herauszufinden, wo das Problem liegt. Sie können die Beispiele in Teilen oder ganz in Ihren eigenen Anwendungen benutzen, ohne dass hierfür eine besondere Erlaubnis notwendig wäre. Soweit es mich persönlich betrifft, sind die Codebeispiele keinen lizenzrechtlichen Beschränkungen unterworfen. Dies gilt allerdings nicht für den erläuternden Text.

Danksagung zur englischen Ausgabe

Einige Passagiere der Achterbahnfahrt, die das Schreiben dieses Buchs für mich bedeutete, verdienen besondere Erwähnung. Dieses Projekt wäre ohne die Unterstützung und Geduld meines Lektors, Simon St. Laurent, niemals möglich gewesen. Er half mir, aus vielen Vorschlägen die richtigen Dinge auszufiltern und sich auf die wichtigen zu konzentrieren.

Dank gilt meiner Frau, ohne die ich dieses Buch niemals angefangen hätte. Ohne ihre Ermutigungen, ihre Geduld und ihren Enthusiasmus hätte ich dieses Buch nicht zu Ende gebracht.

Die technischen Fachgutachter haben mir geholfen, meine Fehler auszubügeln und den Inhalt dieses Buchs beträchtlich zu verbessern. Vielen Dank an euch alle: Bart van den Eijnden, Darren Redfern, Jeff McKenna, Paul Ramsey und Tom Kralidis. Eine Handvoll weiterer Leute half mir, den schier endlosen Strom meiner Fragen zu beantworten. Vielen Dank auch an diejenigen, die einfach nur mit Ja oder Nein geantwortet haben.

Ich bin dankbar für die Unterstützung und die Ermutigungen von Dave McIlhagga und der DM Solutions Group. Sie haben mich immer wieder daran erinnert, dass dieses Buch gebraucht wird.

Besondere Anerkennung verdienen die Entwickler der in diesem Buch vorgestellten Werkzeuge für ihre Open Source-Entwicklungen. Ohne sie gäbe es nicht viel, worüber ich zu schreiben hätte! Ich möchte an dieser Stelle auch der University of Minnesota für die andauernde Unterstützung und ihre Bereitschaft danken, die MapServer-Gemeinschaft weit über ihre ursprünglichen Grenzen hinaus wachsen zu lassen.

Für MapServer existiert bereits eine umfangreiche Dokumentation. Ohne die Anstrengungen vieler Firmen und Freiwilliger hätte ich niemals so viel über diese großartigen Werkzeuge gelernt. Vielen Dank an alle Autoren, die das möglich gemacht haben. Ich hoffe, dass dieses Buch eine gute Ergänzung dargestellt und vielleicht sogar die eine oder andere Lücke zwischen technischer Dokumentation und inhaltlicher Anwendung zu schließen hilft.

Viele Freunde und Kollegen haben mich über die Jahre ermutigt. Ohne diese Aufmunterungen, die Anregung, ruhig auch einmal quer zu denken, und ohne ihre tatkräftige Unterstützung hätte ich dieses Buch vermutlich nicht geschrieben.

Viel zu viele Stunden habe ich damit verbracht, die Hirne der anderen Chatlings¹ anzuzapfen. Wenn E-Mail einfach zu langsam war, hat mir eure Hilfe besonders viel gebracht und ermöglichte die Unterstützung durch die weitere Gemeinschaft in Echtzeit. Danke!

1 Der Begriff »Chatling« ist eine Wortschöpfung, die es noch nicht einmal bis in die Wikipedia geschafft hat. Ein »Chat« ist eine Unterhaltung oder auch ein Geplauder und bezieht sich hier auf eine offene Gemeinschaft von Geo- und Open Source-Enthusiasten, die seit Jahren über Intern Relay Chat (IRC) miteinander kommunizieren, teilweise ohne sich jemals persönlich getroffen zu haben. Der Chat, auf dem man Tyler Mitchell eigentlich immer trifft, ist der OSGeo-Kanal unter *irc://irc.freenode.net#osgeo*.

Einführung in die digitale Kartografie

Karten funktionieren wie ein Mikroskop, nur andersherum, es sind sozusagen Makroskope. Sie geben uns die Möglichkeit, Zusammenhänge zu erkennen, die wir sonst nicht sehen könnten. Selbst wenn wir in einem Flugzeug sitzen, können wir z.B. nicht ohne Weiteres die Geologie unter der Erdoberfläche sehen, die Verteilung der Einwohner in einer Stadt oder die Lärmbelastung durch einen Flughafen. Das alles sind Dinge, die erst durch Karten begreifbar werden.

Vor gar nicht so langer Zeit haben die Menschen diese Karten noch von Hand gezeichnet und koloriert. Daten zu analysieren und daraus Karten zu erstellen war eine langsame und schwierige Arbeit. Dank immer weiter sinkender Kosten für die Datenverarbeitung und -speicherung eröffnen digitale Karten eine Unmenge neuer Möglichkeiten. Mit einem Mausklick und ein paar Zeilen Code analysiert Ihr Computer kartografische Informationen, zeichnet die Karte und klassifiziert dabei die Daten noch farblich. Vom GPS-Navigationssystem in Ihrem Auto bis hin zur Website mit den örtlichen Buslinien ist deutlich erkennbar, dass die digitale Kartografie mittlerweile Teil des täglichen Lebens ist.

Die Erstellung digitaler Karten zu erlernen ist natürlich mit einem gewissen Aufwand verbunden. Wenn Sie beispielsweise die zugrunde liegenden Daten falsch verwenden, erhalten Sie auch fehlerhafte Karten oder solche mit missverständlichem Inhalt. Wie die traditionelle, so bietet auch die digitale Kartografie keine Garantie für Qualität und schon gar nicht für Objektivität. Dieses Buch soll helfen, die Macht der Karten etwas transparenter zu machen und Ihnen die Möglichkeiten geben, selbst den Weg von den rohen Daten bis zur fertigen Karte zu gehen.



Wenn Sie mehr über die Bedeutung der Karten erfahren möchten, sind die Bücher *Die Macht der Karten* von Ute Schneider und *Kartenwelten. Der Raum und seine Repräsentation in der Neuzeit* von Christof Dipper und Ute Schneider (beide Primus Verlag, 2006) als Lektüre sehr zu empfehlen.

Die Macht digitaler Karten

Beim Vergleich traditioneller und digitaler Kartografie werden die Vorteile der digitalen Kartografie schnell deutlich. Bei der traditionellen Kartografie werden Beobachtungen der echten Welt auf Papier übertragen. Ändert oder verschiebt sich ein Merkmal oder wurde es falsch eingezeichnet, muss eine neue Karte erstellt werden, um diese Änderungen zu berücksichtigen. Wächst eine Stadt, muss der Stadtplan komplett neu erstellt werden.

Bei der digitalen Kartografie sind diese Probleme längst nicht so groß. Da die Objekte in Form einzelner Ebenen – wir sprechen hier auch von Layern – gespeichert werden, können Sie Änderungen vornehmen, ohne komplett neu beginnen zu müssen. Sobald sich ein Objekt ändert, kann dies bei einer computerbasierten Karte sofort angepasst werden, und die Karte wird bei der nächsten Betrachtung automatisch korrekt dargestellt. Bei interaktiven Karten können Benutzer genau den für sie interessanten Ausschnitt betrachten, ohne dabei durch die Ausmaße einer gedruckten Seite beschränkt zu werden. Der Benutzer kann die Darstellung auch auf bestimmte Inhalte beschränken. Der GIS-Administrator, der die Karten veröffentlicht, muss nicht mehr raten, welche Informationen der Betrachter möglicherweise sehen möchte, sondern kann eine Auswahl verschiedener Daten anbieten.

Statt die unveränderliche Darstellung einer bestimmten Region in einem bestimmten Kartentyp festzulegen, können Informationen so bereitgestellt werden, dass der Betrachter selbst die Möglichkeit hat, die Darstellung, den Ausschnitt und vielleicht sogar die Auswahl der Inhalte oder die Farbgebung zu verändern. Der Unterschied ist vergleichbar mit dem zwischen einem Autor und einem Webdesigner. In der digitalen Welt liegt der Schwerpunkt eher darauf, anderen beim Auffinden der Informationen zu helfen, anstatt Informationen statisch wie auf einer gedruckten Seite darzustellen. Der Kartograf von heute hat oftmals auch die Rolle eines Webentwicklers, Programmierers oder eines Analytikers für geografische Informationen inne. Sein Augenmerk liegt auf der Verwaltung und Präsentation von Informationen für ein bestimmtes Publikum. Dies können zum Beispiel Energieversorger, die Forstwissenschaft, die öffentliche Verwaltung oder der Bürger sein.

Die Herausforderungen beim Erstellen von Karten

Wenn Sie schon einmal mit digitalen oder auch traditionellen Karten gearbeitet haben, werden Sie trotz meiner Begeisterung wissen, dass die Kartografie nicht immer einfach ist. Warum ist es manchmal so schwierig, die Welt um uns herum auf Karten abzubilden? Wie gut könnten Sie Ihren normalen Weg zum Supermarkt in Form einer Karte darstellen? Oftmals ist es viel einfacher, die Route zu *beschreiben*, als eine Karte davon zu zeichnen. Vielleicht haben wir eine Ahnung davon, wie eine Karte aussehen muss, haben aber Angst, unsere eigene Karte könnte im Vergleich zu einem professionellen Plan lächerlich wirken. Trotz dieser Zweifel ist die Karte, die uns ein Freund auf eine Serviette gekritzelt hat, vermutlich wesentlich nützlicher als ein professioneller Stadtplan es jemals sein kann.

Persönliche Karten

Durch den Einsatz von *persönlicher* Erfahrung anstelle *allgemeinen* Wissens kann eine einigermaßen hilfreiche Karte zu genau dem werden, was wir gerade brauchen. Lässt sich die Position von etwas, das nicht Teil des Allgemeinwissens ist, nicht mit Worten beschreiben, kann eine Karte möglicherweise die Lücke füllen. Karten können eine Beschreibung mit Worten ergänzen; da Sie eine solche Karte oft einfach aus dem Gedächtnis erstellen, kann Ihnen eine solche Zeichnung möglicherweise leicht peinlich sein. Nach unserer Auffassung werden Karten nicht von einfachen Leuten erstellt, sondern von *Experten*. Trotzdem kann eine Karte wie die in Abbildung 1-1 gezeigte wesentlich mehr Aussagekraft besitzen als eine professionelle Karte der gleichen Gegend. Was fehlt den professionellen Karten also? Meist enthalten sie nur allgemeine Informationen anstelle von persönlichen Fakten, die die Karte für Sie wesentlich nutzbringender machen würde.

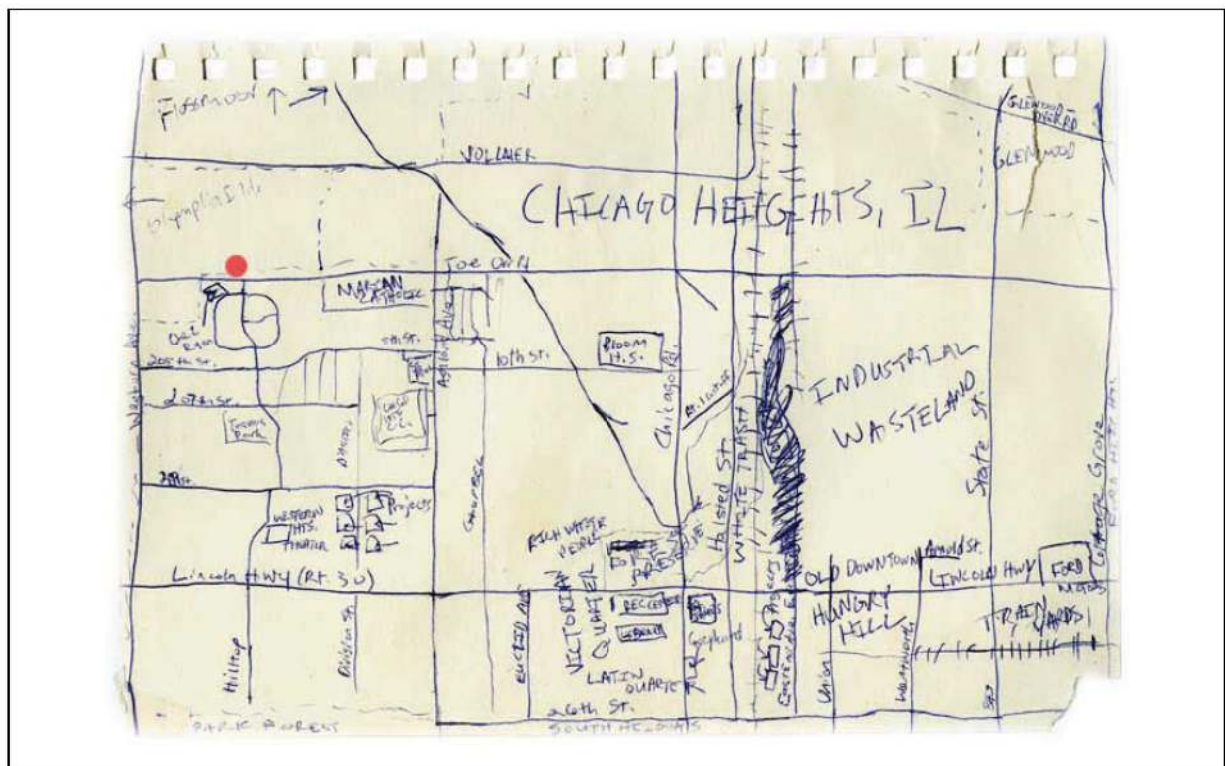


Abbildung 1-1: Eine persönliche Karte, die von Ryan Mendenhall gezeichnet wurde. Sie zeigt die Chicago Heights in Illinois, USA. Diese Karte wurde freundlicherweise vom Lori Napoleons Kartenprojekt, <http://www.subk.net/maps.html>, zur Verfügung gestellt.

Technologische Grenzen

Eigentlich ist die digitale Kartografie gar nicht so neu, wie es manchmal scheint. Seitdem Computer in der Lage sind, grafische Darstellungen der Erde zu erstellen, wurden sie auch für die Kartenerstellung verwendet. Zu Beginn wurden die Karten noch über ASCII-basierte Textdateien dargestellt. (Ich erinnere mich noch, dass ich früher ASCII-

Karten auf einem Tandy Color Computer für ein Rollenspiel erstellt habe.) Allerdings waren diese ASCII-basierten Karten nicht gerade schön. Glücklicherweise hat sich die Computertechnologie weiterentwickelt, sodass Sie heute selbst auf Ihrem eigenen PC hochqualitative Karten erstellen können.

Vielleicht haben Sie bereits eigene Karten erstellt, sind aber mit den Werkzeugen nicht zufrieden gewesen. Proprietäre Programme können sehr teuer sein, besonders wenn Sie erst einmal nur herumspielen und ein Gefühl für die Arbeit bekommen möchten. Wenn Sie Open Source-Software verwenden, müssen Sie nicht schon zu Beginn große Geldbeträge in Software investieren.

Für andere sind die Möglichkeiten der Software wichtiger als die Kosten. Wie bei proprietären Produkten können sich auch die Open Source-Produkte in ihren Leistungsmerkmalen stark unterscheiden. Verbesserte Eigenschaften könnten beispielsweise eine leichtere Benutzung oder eine höhere Ausgabequalität sein. Große Unterschiede gibt es auch in der Art, wie bestimmte Programme untereinander zusammenarbeiten. Dies nennt man auch *Interoperabilität* – die Fähigkeit eines Programms, mit einem anderen Daten auszutauschen. Hierbei kommen oftmals sogenannte *offene Standards* zur Anwendung. Dies sind Protokolle für die Kommunikation zwischen Anwendungen. Der Grundgedanke besteht darin, Standards zu definieren, die nicht von einem bestimmten Softwarepaket abhängen. Stattdessen hängen sie von dem Kommunikationsprozess ab, den der Entwickler implementieren wollte. Ein Beispiel für den Einsatz dieser Standards wäre die Fähigkeit Ihres Programms, Karten von einem entfernten Dienst (OGC WMS) über das Internet abzufragen. Die ganze Tragweite offener Standards wird klar, wenn Ihr Programm in der Lage ist, mit einer Anwendung zu kommunizieren, die von einem anderen Entwicklerteam oder einem anderen Hersteller stammt. Dies ist besonders bei großen Organisationen – insbesondere in der öffentlichen Verwaltung – äußerst wichtig, bei denen die Möglichkeit, Daten auszutauschen, für den Erfolg des Projekts entscheidend sein kann. Produkte, die offene Standards implementieren, tragen dazu bei, dass eine langfristige Benutzbarkeit gewährleistet wird. Doch obwohl von manchen Produkten behauptet wird, sie seien interoperabel, implementieren sie nicht immer die kompletten Standards. Sie als Benutzer sollten also eine gewisse Vorsicht walten lassen. Einige Firmen ändern die Standards ihrer Produkte ab und widersprechen so dem Sinn der Standards. Auf der anderen Seite werden die Standards mit jeder Version weiterentwickelt und verändern sich.

Vermutlich sind Kosten und Ihre Fähigkeiten das größte Hindernis. Vielleicht wollen Sie Ihre eigenen Karten erstellen, wissen aber noch nicht, wie. Möglicherweise wissen Sie nicht, welche Werkzeuge zur Verfügung stehen. Dieses Buch beschreibt einige der frei verfügbaren Werkzeuge, die Sie bei der Erstellung eigener Karten unterstützen können.

Eventuell fehlt Ihnen aber auch einfach das notwendige technische Fachwissen. Während die traditionelle Kartografie den meisten Menschen nicht vertraut ist, gibt es bei den digitalen Verfahren Hürden für Anwender, die technisch nicht besonders bewandert sind. Das liegt daran, dass die Installation und Modifikation von Software oftmals über

den Horizont vieler Computernutzer hinausgeht. Andere kennen sich dagegen sehr gut mit der digitalen Kartografie aus und können einfach zu benutzende Werkzeuge für andere entwickeln. Dies bedeutet einen großen Vorteil für beide Seiten, dabei besteht ein enger Kontakt zwischen Anwendern und Entwicklern. Neue Benutzer können räumliche Informationen mit minimalem Aufwand mithilfe existierender Anwendungen betrachten. Ein gutes Beispiel für die Zusammenarbeit von Entwicklern und Anwendern ist die Open Source Geospatial Foundation (siehe Kasten), über die die Entwicklung, Verbreitung und Nutzung von Freier und Open Source-Software gefördert wird.

OSGeo– Open Source Geospatial Foundation

Die *Open Source Geospatial Foundation* fördert und unterstützt die Entwicklung, Verbreitung und Nutzung von Freier und Open Source-Software in der räumlichen Datenverarbeitung. Die Förderung der freien Verfügbarkeit öffentlicher Geodaten und die Erstellung ganz freier Geodaten ist ein weiterer Schwerpunkt, denn ohne Daten ist die beste Software sinnlos. Die Bereitstellung und Pflege von frei zugänglichem Lehrmaterial, Dokumentationen und die Unterstützung von Bildungsträgern fördern die Wissensvermittlung. Daraus ergeben sich drei sich ergänzende Schwerpunkte:

- Softwareprojekte
- Bildung und Lehre
- öffentliche Geodaten

Die meisten Arbeiten der OSGeo-Mitglieder konzentrieren sich auf die Entwicklung und Pflege der Softwareprojekte, von denen einige bereits seit Jahren entwickelt werden. Bevor eine Software offizielles OSGeo-Projekt wird, muss es den sogenannten Inkubationsprozess durchlaufen und erfolgreich graduieren – also alle Bedingungen der OSGeo erfüllen (z.B. Quellcodeanalyse, gesunde Entwicklergemeinschaft).

Ein weiteres Ziel der OSGeo ist die Kooperation mit Universitäten und Bildungsstätten, um die Erstellung und Pflege von Lehrplänen zu unterstützen. Die effiziente Nutzung von Software setzt voraus, dass sie nicht nur gut ist, sondern auch kompetent eingesetzt wird. Die OSGeo betreibt eine Bildungsplattform auf ihrem Portal, die Lehrgänge in verschiedenen Sprachen zu unterschiedlichen Themen wie WebGIS, Desktop-GIS und viele mehr zur Verfügung stellt.

Die OSGeo betreibt ein Onlineportal, das von Entwicklern und Anwendern der Software als Drehscheibe zur Wissensvermittlung genutzt und weiterentwickelt wird. Die OSGeo ist eine offene Organisation, in die Sie Ihre eigenen Ideen, Erfahrungen und Entwicklungen einbringen können. Ein erster Einstieg ist das OSGeo-Wiki: <http://wiki.osgeo.org/>.

→

Die formale Organisationsstruktur der OSGeo ist überschaubar und besteht aus einem neunköpfigen Board of Directors, das von derzeit 45 Satzungsmitgliedern gewählt wird. Alle Mitglieder, Direktoren und Funktionäre arbeiten auf freiwilliger Basis und werden nicht bezahlt. Der einzige bezahlte Mitarbeiter ist Tyler Mitchell (der Autor dieses Buchs), der als Executive Director formale Funktionen übernimmt. Dadurch ist das Budget der OSGeo sehr überschaubar, und das soll auch so bleiben. Die Finanzierung der OSGeo erfolgt über Sponsorengelder und Einnahmeüberschüsse der jährlichen OSGeo-Konferenz FOSS4G.

Werden Sie Mitglied und profitieren Sie von den Vorteilen, die eine kollaborative Entwicklungsplattform und eine große Anwendergemeinschaft bieten können. Es erwarten Sie aktuelle Informationen zu GIS-Technologien, Tutorials, zahlreiche Mailinglisten, Issue Tracker und Code Repositories: <http://www.osgeo.org>

Sicher gibt es technologische Schranken. Für diejenigen, die einen Computer benutzen können, um damit kartografische Informationen zu verarbeiten, sind die Möglichkeiten nahezu endlos. Die hier beschriebenen Werkzeuge sind nicht unbedingt einfach zu bedienen. Sie erfordern eine gewisse Portion an technischem Verständnis, weil Geodaten bereits für sich genommen ein recht komplexes Thema sind. Manchmal liegt keine Installationsroutine vor, und gelegentlich müssen eigene Anpassungen vorgenommen werden. Aber wenn die Werkzeuge einmal eingerichtet sind, benötigen sie im Allgemeinen nur noch minimale Wartung.

Hintergrund: Freie und Open Source-Software

Der Begriff *Open Source* bezieht sich auf ein Entwicklungsmodell, bei dem der Quelltext (engl. Source Code) einer Software frei verfügbar ist. Dieser Quelltext wird in eine Maschinsprache übersetzt, bevor er auf einem Computer ausgeführt werden kann (engl. Object Code oder Binary). Um einen Fehler in der Software beheben zu können oder um neue Funktionalität hinzuzufügen, muss der Entwickler Zugriff auf den Quelltext haben.

Wenn Sie nicht selbst Software entwickeln können oder wollen, scheint dies zunächst ein unwichtiges Detail zu sein. Die Definition von Open Source geht allerdings noch weiter und beinhaltet ein Lizenzierungsmodell, das auch als »Freie Software« (engl. Free Software) bekannt ist. Bei dieser Art der Lizenzierung wird die Software zunächst vom Entwickler unter das Copyright gestellt und dadurch geschützt. Dann werden Vertriebsbedingungen hinzugefügt, die allen Menschen das Recht geben, diese Software zu verwenden, zu verändern und weiterzugeben. Dafür ist Open Source natürlich eine Voraussetzung.

Die meisten international operierenden Firmen der sogenannten Software-Industrie mit mehreren tausend Beschäftigten verfolgen dagegen ein proprietäres Geschäftsmodell, bei dem der Quelltext der Software nicht veröffentlicht wird. Das birgt Gefahren. Zum

Beispiel enthält die Software möglicherweise schwerwiegende Fehler, die nicht bekannt gemacht werden, weil dem Hersteller sein Image wichtiger ist und keine unabhängige Stelle den Quelltext überprüfen kann. Außerdem kann der Hersteller jederzeit etwas Grundsätzliches in der Software ändern und damit seine Kunden dazu zwingen, eine neue Version zu kaufen. Das passiert ständig und wird von vielen einfach hingenommen, weil sie es nicht anders kennen. Freie und Open Source-Software verursachen diese Probleme nicht.

Es gibt viele weitere Gründe dafür, dass Freie und Open Source-Software gute Konzepte sind, manche sind eher pragmatischer Art, andere von mehr philosophischer oder ethischer Natur. Eine sehr vollständige und gute Zusammenstellung der Hintergründe bietet das Buch *Freie Software – Zwischen Privat- und Gemeineigentum* von Volker Grassmuck. Es wird vom deutschen Bundesamt für Politische Bildung herausgegeben und kann entweder aus dem Netz heruntergeladen (<http://freie-software.bpb.de/>) oder als Buch bestellt werden.

Es gibt weltweit Hunderttausende von Open Source-Projekten, die von Millionen von Menschen benutzt, entwickelt und verbessert werden. Zu den bekanntesten gehören GNU/Linux, der Webbrowser Firefox und die Bürosoftware OpenOffice. Die Projekte existieren aus sehr unterschiedlichen Motivationen. Um bei den drei genannten Beispielen zu bleiben: Linux ist ein typisches Graswurzel-Projekt, an dem weltweit Hunderttausende gleichzeitig arbeiten. Deshalb wird vieles auf unterschiedliche Weise und mehrfach entwickelt und tausendfach getestet. Diese Diversität ist nachweislich jeder Monokultur überlegen und ermöglicht die Entwicklung hochspezialisierter und leistungsfähiger Software. Das zweite Beispiel ist der Browser Firefox. Er wird von Mozilla.org entwickelt, einer Non-Profit-Organisation, die es sich zum Ziel gesetzt hat, Open Source-Software zu entwickeln und Standards im Internet zu unterstützen. Das dritte Beispiel ist die Bürosoftware OpenOffice.org, die von dem Softwarehersteller Sun Microsystems hergestellt wird, einem Konzern mit einem Jahresumsatz in 2007¹ von 13,9 Milliarden und 34.600 Mitarbeitern. Freie und Open Source-Software wird also von verschiedensten Organisationen, Hochschulen, der privaten Wirtschaft und der öffentlichen Verwaltung sowohl entwickelt als auch genutzt.

Die in diesem Buch behandelten Werkzeuge werden zum größten Teil als Open Source-Software entwickelt und werden durch eine Freie Software-Lizenz geschützt. Das bedeutet, dass Sie die Software in den meisten Fällen kostenfrei herunterladen können und jederzeit sowohl zu Ihrem eigenen Spaß und Vergnügen als auch zu kommerziellen und professionellen Zwecken nutzen können. Das Schlüsselwort der Open Source-Software ist Freiheit – die Freiheit, ein Programm nach den eigenen Bedürfnissen zu verwenden und zu modifizieren.

1 http://www.sun.com/aboutsun/investor/earnings_releases/pr/2007-q4.html

Weitere allgemeine Informationen zu Open Source-Software finden Sie unter den folgenden Adressen:

- Open Source Initiative: <http://opensource.org/>
- Free Software Foundation: <http://www.fsf.org/>
- das O'Reilly-Buch *Open Sources*: <http://www.oreilly.de/catalog/opensources/>

Deutschsprachige Informationen zur Geodatenverarbeitung mit Open Source-Software finden Sie hier:

- Open Source Geospatial Foundation: <http://www.osgeo.org/>
- FreeGIS: <http://www.freegis.org/>
- GAV e.V (GRASS-Anwender-Vereinigung e.V.): <http://www.grass-verein.de/>

Beachten Sie auch die deutschsprachige Konferenz FOSSGIS (Freie und Open Source Software für Geoinformationssysteme), die jährlich stattfindet. Die Konferenz wird zusammen mit der OSGeo Foundation, der GRASS-Anwender-Vereinigung e.V. und mit Unterstützung von FreeGIS.org veranstaltet. Die Konferenz bietet einen Überblick über Open Source-Projekte aus dem GIS-Bereich. Verschiedene Software-Projekte stellen sich in Fachvorträgen und Workshops vor. Es gibt zahlreiche Präsentationen von Anwendern und Berichte aus der Praxis. Informationen zur Konferenz finden Sie unter <http://www.fossgis.de/>.

Verschiedene Arten der Web-Kartografie

Eine sehr effiziente Methode, Endbenutzern kartografische Informationen zur Verfügung zu stellen, besteht in der Verwendung einer Webseite. Webbasierte Kartenanwendungen werden immer beliebter. In der Web-Kartografie wird zwischen statischen und interaktiven Anwendungen unterschieden.

Durch *Google Maps* und *Google Earth* sind webbasierte Kartenanwendungen weithin bekannt geworden und werden umfassend genutzt. Sie haben sicherlich auch schon einmal eine Adresse über Google Maps ermittelt oder sich Ihren letzten Urlaubsort im Luftbild angesehen. Und sicherlich waren Sie begeistert von dem Ergebnis. Wenn nicht, sollten Sie es direkt mal ausprobieren: <http://maps.google.com/maps>.

Immer seltener begegnen uns statische Karten, die als Bilddatei auf der betreffenden Seite eingebunden wurden. Besitzen Sie bereits eine digitale Karte (beispielsweise indem Sie ein Dokument eingescannt haben), können Sie diese sehr leicht in eine Webseite einbetten. Hierfür sind nur sehr geringe Webdesign-Fertigkeiten notwendig, weil Sie eigentlich nur ein Bild in eine Webseite integrieren.



In diesem Buch geht es nicht um Webdesign. Dazu gibt es bei O'Reilly eine Reihe anderer Bücher für Anfänger und Fortgeschrittene, wie beispielsweise *Webdesign mit (X)HTML und CSS*, *HTML mit CSS & XHTML von Kopf bis Fuß*, *HTML & XHTML – Das umfassende Handbuch* und viele andere.

Interaktive Karten werden immer populärer. Es sind aber besondere Fähigkeiten notwendig, um solche Sites einzurichten und zu betreiben. Der Begriff *interaktiv* bedeutet, dass der Benutzer mit der Karte interagieren kann. Dies kann beispielsweise die Auswahl bestimmter Kartenebenen bedeuten oder das Hineinzoomen in einen besonders interessanten Bereich der Karte. Nach jeder Interaktion wird die Karte aktualisiert.

Interaktive Karten, die über eine Webseite zugänglich sind, werden als *webbasierte Karten (web maps)* bezeichnet. Im Unterschied zu statischen Karten handelt es sich bei interaktiven Karten um webbasierte Anwendungen. In Abbildung 1-2 sehen Sie eine Darstellung des Informationsflusses und der Vorgänge, die hinter den Kulissen ablaufen, wenn ein Benutzer eine Karte von einer Website anfordert. Der Benutzer fordert eine Karte über einen Aufruf vom Webserver an, und der Webserver reicht die Anfrage an den Kartenserver (Mapserver) weiter, der die benötigten Daten zusammenträgt. Am Ende wird die fertige Karte im Webbrowser angezeigt.

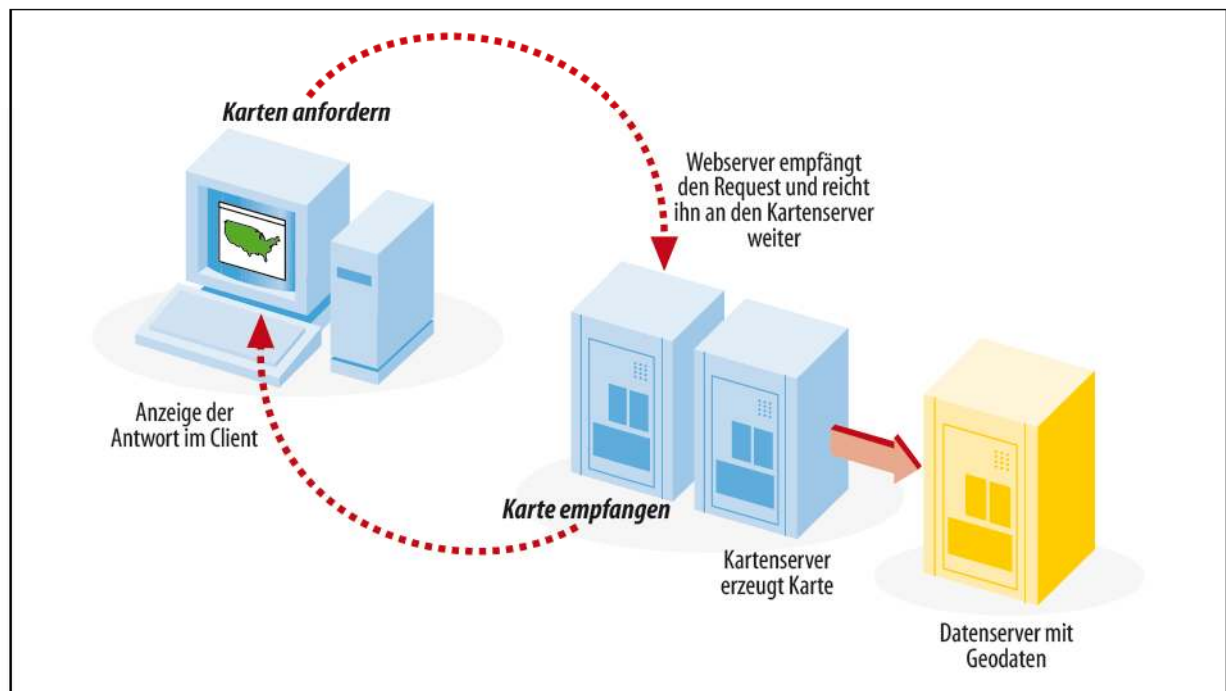


Abbildung 1-2: Das Diagramm zeigt, wie ein Kartenserver mit dem Endbenutzer und den Backend-Programmen auf dem Server kommuniziert

Wer sind die Benutzer von webbasierten Karten?

Einfach gesagt, werden webbasierte Karten von zwei verschiedenen Gruppen benutzt: Diensteanbietern und Nutzern.

Die Bedürfnisse der Endbenutzer können sehr unterschiedlich sein. So spricht eine Karte eine ganz bestimmte Nutzergruppe an. Vielleicht suchen Sie eine Website, über die Sie routen können. Jemand anderer sucht möglicherweise nach einer Luftaufnahme und einer topografischen Karte für die nächste Wanderung. Für manche Nutzergruppen gibt es Karten-Websites, die genau auf ihre Bedürfnisse zugeschnitten sind.

Diensteanbieter stellen webbasierte Karten aus unterschiedlichen Gründen zur Verfügung. Vielleicht sollen bestimmte Prozesse, die sich im Büro ständig wiederholen, automatisiert werden. Der Einsatz einer Karten-Website kann hervorragend dazu geeignet sein, vorher unzugängliche Daten für eine breite Nutzergruppe zu öffnen. Anstatt mehrere Mitarbeiter einer Firma in traditioneller GIS-Software zu schulen, ist es oft sinnvoller und günstiger, wenn Sie eine webbasierte Kartenanwendung aufsetzen, die von allen Mitarbeitern leicht zu bedienen ist.

Sollen bestimmte Informationen einem breiten Publikum zugänglich gemacht werden, ohne den Kunden mit den Rohdaten zu belasten, ist eine WebGIS-Anwendung möglicherweise eine gute Lösung. Ein gutes Beispiel ist der Stadtplandienst der Stadt Rostock (<http://www.rostock.de/>). Hier werden unter anderem Luftbilder, ein Stadtplan und detaillierte Grundkarten angezeigt. WebGIS-Anwendungen können auch zum Datawarehouse ausgebaut werden, wobei Daten über die Anwendung bestellt werden können.

Beispiele für webasierte Kartenanwendungen

Es gibt mittlerweile eine ganze Reihe von Sites, die webbasierte Karten enthalten. In Tabelle 1-1 finden Sie eine Liste mit Verweisen zu Anwendungen, die auf der in diesem Buch beschriebenen Software oder auf ähnliche Prinzipien aufbauen. Beachten Sie bei der Liste vor allem die ersten vier Links, die wiederum auf weitere WebGIS-Anwendungen verweisen.

Tabelle 1-1: Beispielanwendungen, die UMN MapServer-basierte interaktive Karten bereitstellen

Link zu Demoanwendung	Beschreibung
http://mapserver.gis.umn.edu/gallery	MapServer-Galerie mit vielen Links zu laufenden Anwendungen
http://www.mapbender.org/index.php/Gallery	Mapbender-Galerie mit vielen Links zu laufenden Anwendungen
http://openlayers.org/gallery/	OpenLayers-Galerie mit Links zu Demoanwendungen
http://cartoweb.org/demo.html	Cartoweb-Demoanwendungen
http://www.geoportal.rlp.de/	GeoPortal-Rheinland-Pfalz
http://portalu.de/	PortalU – Umweltportal Deutschland
http://www.naturparkscout.de	NaturparkScout
http://www.flo.rlp.de/	FLOrlp – Flächeninformationen Online

Tabelle 1-1: Beispielanwendungen, die UMN MapServer-basierte interaktive Karten bereitstellen (Fortsetzung)

Link zu Demoanwendung	Beschreibung
http://www.muenchen.de/Rathaus/rgu/mobilitaet_freizeit/verkehr/radfahren/101633/	Radroutenplaner der Stadt München
http://www.so.ch	Interaktive Karten des Kantons Solothurn
http://www.dnr.state.mn.us/maps/	verschiedene Freizeit- und Naturkarten für den Staat Minnesota, USA
http://mesonet.tamu.edu/	Echtzeitwetterkarten der USA

Die Abbildungen 1-3, 1-4 und 1-5 zeigen drei Anwendungen, die den UMN MapServer als Kartenserver einsetzen. Wie Sie sehen, lassen sich mit MapServer unterschiedlichste Daten visualisieren. In den Beispielen kommen drei verschiedene Clientanwendungen zum Einsatz: die WMS-Clients *OpenLayers* und *Mapbender* sowie der Client *p.mapper*, der MapScript verwendet.

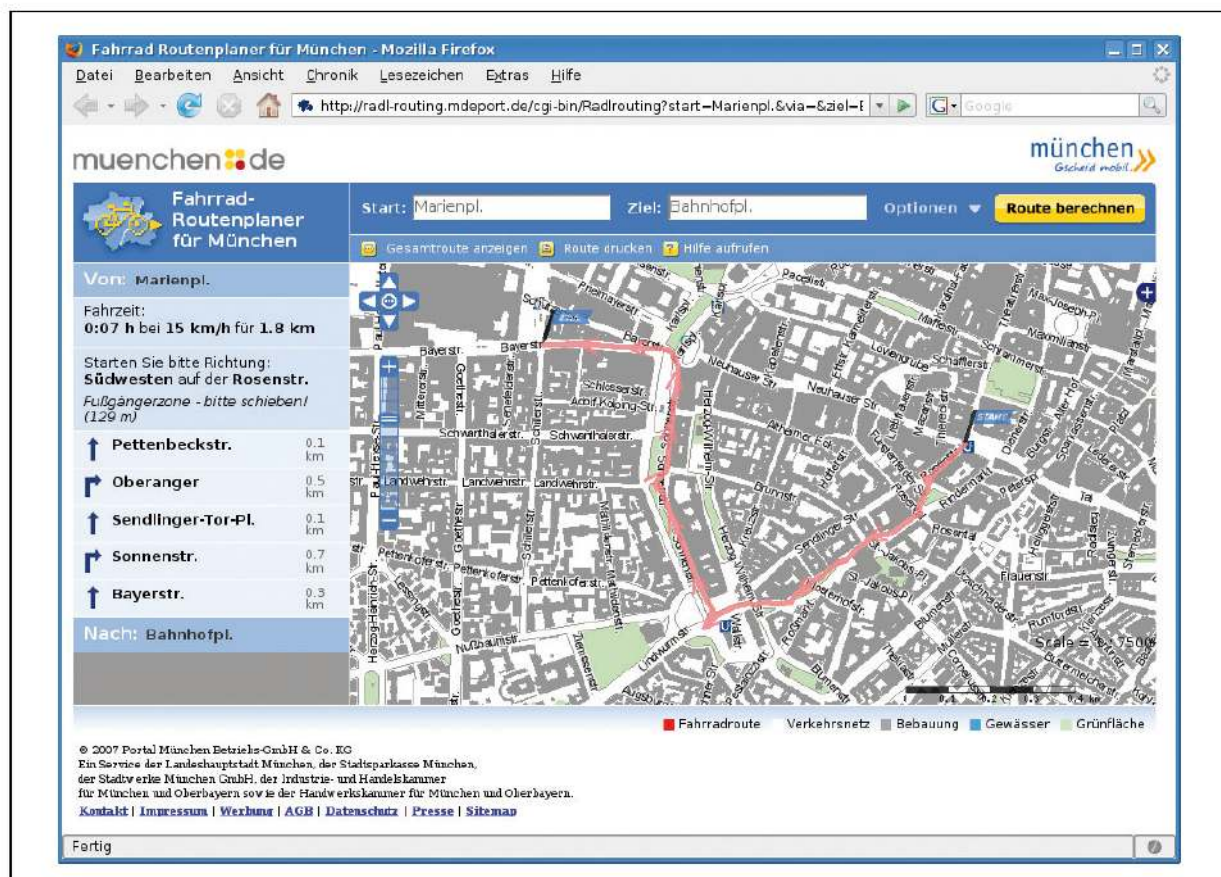


Abbildung 1-3: Radroutenplaner der Stadt München, der OGC-Kartendienste nutzt und OpenLayers als WMS-Client verwendet.

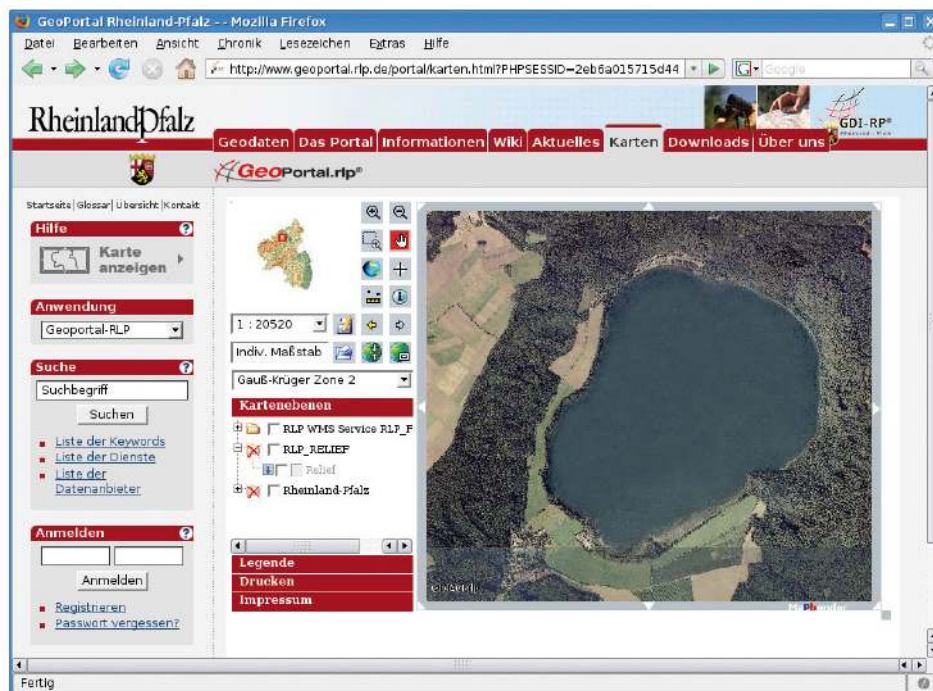


Abbildung 1-4: GeoPortal.rlp – Informations- und Kommunikationsplattform für Geodaten, Geoinformationen und Geodienste in Rheinland-Pfalz (Deutschland) unter Verwendung von Mapbender als OGC WMS-Client

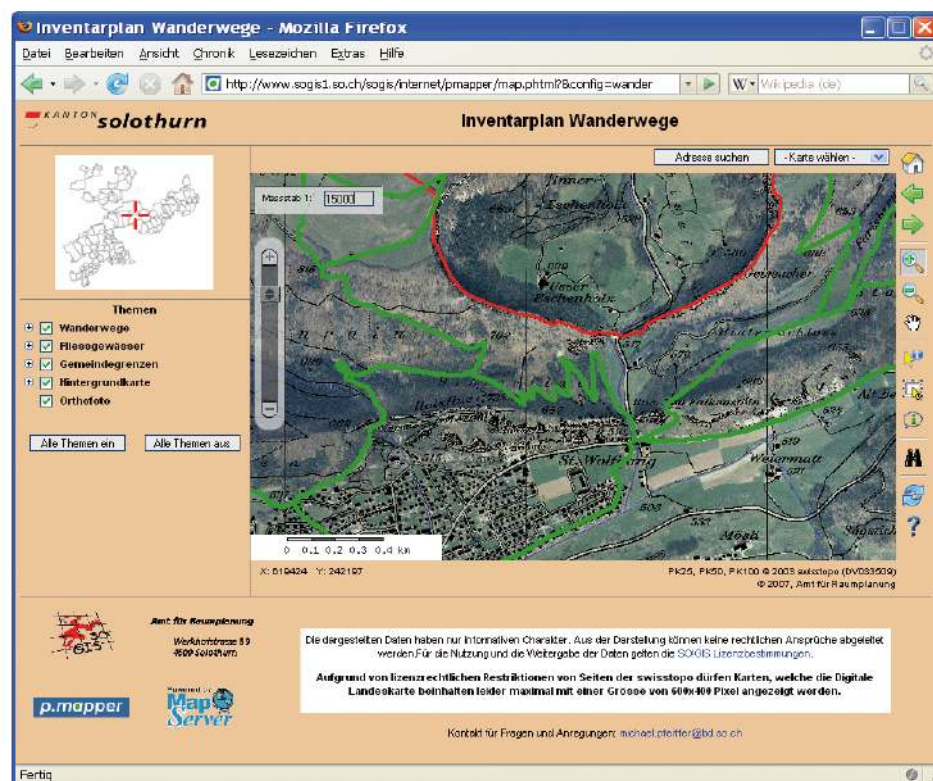


Abbildung 1-5: Wanderwegeauskunft des Kantons Solothurn mit p.mapper und MapServer

Was passiert im Hintergrund?

WebGIS-Anwendungen sind keine Zauberei, jedoch benötigt man entsprechendes Hintergrundwissen, damit sie verständlich sind. Auch wenn im Hintergrund unterschiedliche Prozesse ablaufen können, basieren die meisten Anwendungen auf den gleichen Grundkonzepten:

- Der Webserver ist für die Beantwortung von Browseranfragen zuständig. Er schickt das Ergebnis der Anfrage inklusive der eingebetteten Bilder an den Browser des Benutzers zurück.
- Der Kartenserver (Mapserver) nimmt Anfragen entgegen, die ihm vom Webserver übergeben wurden. Die Anfrage bezieht sich in der Regel auf eine Karte mit bestimmtem Inhalt für einen bestimmten Bereich. Die Ergebnisse einer Abfrage oder Datenanalyse können auch in tabellarischer Form zurückgegeben werden. Der Kartenserver erzeugt auf Basis der Anfrage die entsprechende Bilder- oder Informationsausgabe und übergibt diese dem Webserver, der sie schließlich an den Browser des Benutzers zurückschickt.
- Wie in Abbildung 1-2 gezeigt, benötigt der Kartenserver Zugriff auf die Datenquellen, um die Anfragen beantworten zu können. Die notwendigen Dateien können sich entweder auf dem gleichen Computer oder auf entfernten Rechnern befinden. Hält sich der Kartenserver an die Standards, können die Daten auch von anderen Kartenservern über WMS/WFS in Echtzeit abgefragt werden.

Weitere Informationen zu Kartenservern im Web finden Sie in den Kapiteln 4, 11 und 12 sowie 13, die sich detaillierter mit MapServer und standardisierten Kartenanfragen auseinandersetzen.

Erstellen Sie Ihre eigene Web-Mapping-Anwendung

Dieses Buch behandelt die verschiedenen Bestandteile, die notwendig sind, um Ihre eigenen Web-Mapping-Projekte zu erstellen, sowie die notwendigen Dinge, die Sie zur Verwaltung von räumlichen Daten wissen müssen. Um Ihnen einen Überblick über die verwendeten Technologien zu geben, finden Sie hier eine Liste der wichtigsten Voraussetzungen, um eine eigene Kartenanwendung zu betreiben. Von der nun folgenden Liste werden lediglich der Web-Mapping-Client, der Kartenserver und die Geodaten in diesem Buch ausführlich behandelt.

Ein Computer

Das versteht sich eigentlich von selbst. Allerdings sollten Sie bedenken, dass Sie für größere Kartografie-Projekte auch leistungstärkere Rechner benötigen werden. Große und komplexe Karten erfordern natürlich eine höhere Rechenleistung. Ein schnellerer Prozessor kann Anfragen auch schneller beantworten. Die üblichen Webhosting-Angebote reichen häufig nicht aus, weil Sie Zugriff auf das verwendete Betriebssystem und den Webserver benötigen. Mittlerweile gibt es auch spezielle Hosting-Lösung für Web-Mapping-Projekte.

Eine Internetverbindung

Es ist zwar vorstellbar, dass Sie eine Web-Mapping-Anwendung nur für sich selbst oder ein internes Netzwerk (z.B. innerhalb Ihrer Firma oder im städtischen Intranet) betreiben wollen. Wollen Sie Ihre Karten jedoch veröffentlichen, benötigen Sie auch eine öffentlich zugängliche Internetverbindung. Einige Internetangebote beschränken die verfügbaren Dienste. Benötigen Sie zusätzliche Funktionen, müssen Sie oft auf ein teureres Angebot ausweichen. Die Leistungsfähigkeit einer Website hängt oft von der Bandbreite der Internetverbindung ab. Wollen Sie beispielsweise größere Bilder (und damit auch größere Dateien) übertragen, kann es unter Umständen einige Sekunden dauern, bis sie beim Benutzer ankommen, auch wenn sie lokal sofort angezeigt werden.

Ein Webserver

Ein Webserver wird gebraucht, um die Verständigung zwischen dem Endbenutzer (der mit einem Browser auf Ihre Site zugreift) und dem Kartenserver auf Ihrem Computer zu ermöglichen. Der Webserver zeigt dem Benutzer eine Webseite an, die die Karte und die entsprechenden Werkzeuge enthält. Der beliebteste Webserver ist sicherlich der Apache HTTP Server (<http://httpd.apache.org/>), der ebenfalls ein Open Source-Projekt ist. Außerdem ist Microsoft Internet Information Services (IIS) (<http://www.microsoft.com/WindowsServer2003/iis/default.msp>) für den Betrieb unter Windows als Webserver einsetzbar. Bei einigen Webhosting-Angeboten ist es unter Umständen nicht möglich, auf die zugrunde liegenden Einstellungen des Servers zuzugreifen.

Ein Kartenserver

Der Kartenserver ist sozusagen der »Motor« hinter den Karten, die auf der Webseite angezeigt werden. Der Kartenserver bzw. das Kartografieprogramm muss so eingerichtet werden, dass es mit dem Webserver kommunizieren und die notwendigen Datenebenen zu einem passenden Bild zusammenfügen kann. Auch wenn es eine Reihe von Alternativen gibt, werden wir uns in diesem Buch hauptsächlich mit dem UMN MapServer als Kartenserver beschäftigen.

Ein Client

Je nach Projektaufbau benötigen Sie einen eigenen Web-Mapping-Client, der die Anfragen zusammenstellt und die Karten vom Kartenserver anfordert. In diesem Buch werden wir OpenLayers und Mapbender als WMS-Clients genauer kennenlernen.

Geodaten

Ohne Geodaten können Sie keine Karten erstellen. Geodaten können Satellitenfotos sein, Daten aus räumlichen Datenbanken, dateibasierte Geodaten, Textdateien, die Koordinaten enthalten, oder Daten, die von entfernten Rechnern angefordert werden. Geodaten werden oft auch als räumliche Daten bezeichnet. Sie können über verschiedene Clients visualisiert werden – über Desktop-GIS-Programme oder über Kartenserver wie MapServer.

Metadaten

Diese Informationen sind zwar nicht unbedingt notwendig, ich erwähne sie hier trotzdem, weil sie in Zukunft immer wichtiger werden. Bei Metadaten handelt es sich um Daten über Daten. Dies können zum Beispiel Informationen darüber sein, um was für Daten es sich handelt, woher die Daten stammen, wann sie erfasst wurden, mögliche Verwendungsformen und an wen Sie sich bei Fragen wenden können. Je mehr räumliche Daten im Internet verfügbar sind, desto wichtiger wird es, diese Informationen zu katalogisieren. Bereits jetzt existieren Dienste, die Onlinedatenquellen durchsuchen und katalogisieren, damit andere die gesuchten Informationen leichter finden können. Als Beispiel für einen Katalogdienst kann das Projekt PortalU (<http://www.portalu.de/>) genannt werden, das einen Umweltdatenkatalog bereitstellt.

Im Verlauf dieses Buchs werden Sie lernen, wie Sie die einzelnen Bestandteile zu einer eigenen, interaktiven Kartenanwendung zusammenstellen können.

Aufgaben und Werkzeuge der digitalen Kartografie

Karten können sehr schön sein. Einige antike Karten, die heute als Drucke, Schreibpapier oder sogar Grußkarten Verwendung finden, werden eher aus ästhetischen als aus kartografischen Gründen geschätzt. Diese Meisterstücke der Kunst und Wissenschaft können gerade auf Kartografieneulinge einschüchternd wirken. Glücklicherweise ist die Erstellung einer Karte heutzutage kein Hexenwerk mehr.

Bevor Sie beginnen, sollten Sie wissen, dass alle Karten für einen bestimmten Zweck erstellt werden. Wenn Sie diese Regel verstanden haben, halten Sie den Schlüssel für ein erfolgreiches Projekt bereits in der Hand. Das gilt vollkommen unabhängig davon, wie die Karte erstellt wird. Die traditionellen Werkzeuge waren Stift und Papier – nicht etwa Magie. Bei digitalen Karten erfolgt das Zeichnen der Karten, indem Punkte zu Linien und Formen oder zu einem Mosaik aus farbigen Quadraten zusammengefügt werden.

Der Zweck und die Grundlage digitaler Karten unterscheidet sich kaum und ist auch nicht komplexer als die Erstellung traditioneller Karten. In der Vergangenheit hätte sich der Kartograf hingesetzt, ein Stück Papier genommen und erst einmal eine Skizze angefertigt. Beim Einsatz digitaler Werkzeuge ist der Computer die Leinwand, die Software übernimmt das Zeichnen, indem sie geografische Daten als Grundlage verwendet. Digitale Werkzeuge bieten kartografische Möglichkeiten, die den Erstellungsprozess entscheidend vereinfachen.

Dieses Kapitel beschäftigt sich mit den Aufgaben und Schwierigkeiten, die bei der digitalen Erstellung von Karten auftreten. Dies beinhaltet einen Überblick über die notwendigen Arbeitsschritte. Die Ziele *digitaler Kartografie* unterscheiden sich nicht von denen der traditionellen: die Darstellung räumlicher Informationen für ein bestimmtes Publikum zu einem ganz bestimmten Zweck. Vielleicht müssen Sie für Ihre Firma den Ort für eine geplante Filiale visualisieren, oder Sie wollen zeigen, wo sich im See bei Ihrer Datscha die besten Angelstellen befinden. Verschiedene Gründe führen zum gleichen Ergebnis: einer Karte.

In den meisten Fällen können die Begriffe *räumliche Informationen* und *Karten* synonym verwendet werden. Normalerweise bezieht sich eine *Karte* allerdings auf die gedruckte oder digitale Ausgabe am Ende des Erstellungsprozesses. *Räumliche Informa-*

tionen beziehen sich auf Daten, die auf einem Computer gespeichert sind und für eine Reihe verschiedener Zwecke eingesetzt werden können.

Wenn das Endprodukt einer Feldforschung eine ausgedruckte Karte ist, ist der gesamte Prozess mit der Ausgabe der Papierkarte eigentlich abgeschlossen. Zwar kann diese Karte verändert und erweitert werden, wenn zusätzliche Informationen verfügbar werden, aber die ausgedruckte Karte ist das Endprodukt, das für einen ganz bestimmten Zweck erstellt wurde.

Die digitale Kartografie geht darüber hinaus. Computerbasierte Werkzeuge helfen dabei, räumliche Daten zu sammeln und mit ihnen zu interagieren. Die Daten werden verwendet, um Karten zu erstellen, sie können aber auch dazu benutzt werden, neue Daten zu generieren oder statistische Zusammenfassungen zu erstellen. Die gleichen Geodaten können in unterschiedlichen Projekten eingesetzt werden, ohne dass neue Forschungsdaten erhoben werden müssen oder eine Papierkarte kopiert werden muss. Diese Eigenschaft macht die digitale Kartografie effizienter und ermöglicht einen anwenderfreundlichen Umgang.

Die Arbeitsabläufe, die normalerweise einen großen manuellen Arbeitsaufwand bedeuteten, werden in der digitalen Kartografie durch computergestützte Techniken vereinfacht. Die Aufgaben sind die gleichen wie die eines Kartografen, allerdings gibt es große Unterschiede bei den Lösungsansätzen und den Werkzeugen. Abbildung 2-1 zeigt den Arbeitsablauf beim Erstellen einer digitalen Karte.

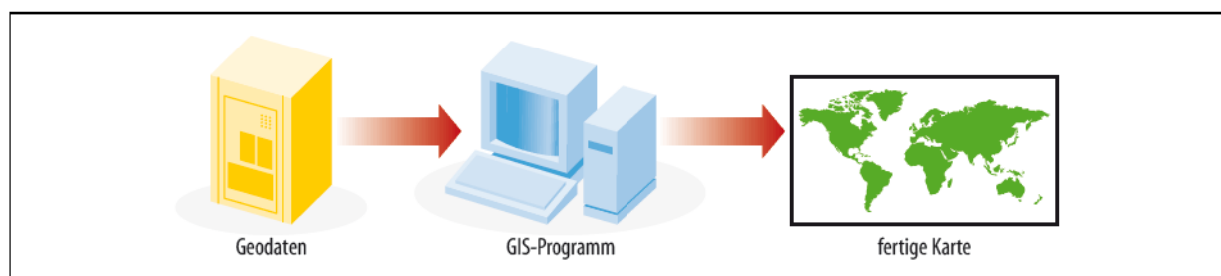


Abbildung 2-1: Digitale Karten werden mithilfe eines GIS-Programms erstellt, das auf die Geodaten zugreift und die Karte an den Nutzer zurückgibt

Welche Aufgaben erwarten Sie beim Erstellen von Karten?

Um eine Karte zu erstellen, sind drei Grundschrte notwendig: Datenerhebung, Positionierung der Daten und die Visualisierung der Daten in einer Karte. Digitale Werkzeuge haben diese Aufgaben effizienter und genauer gemacht.

Datenerhebung

Messinstrumente wie Lasermessgeräte oder bilderzeugende Satelliten erstellen eigenständige Daten, die weniger von persönlichen Interpretationen abhängig sind als traditionelle Beobachtungen wie das Interpretieren von Fotos oder das Zeichnen der Landschaftsmerkmale von Hand, die leicht eine objektive Betrachtung verzerren können. Geografische Referenzwerkzeuge wie GPS-Empfänger können präzise Daten liefern, sogar von in Bewegung befindlichen Objekten.

Positionierung der Daten

Geografische Referenzsysteme bilden tatsächliche Orte auf der Erde ab, z.B. über geografische Koordinatensysteme wie Breite und Länge. Bilddaten, die von Satelliten oder Flugzeugen erfasst werden, müssen ebenfalls georeferenziert werden, also einen Bezug zu diesen Koordinatensystemen bekommen, um mit anderen Geodaten dargestellt werden zu können. Oft werden diese Positionen heute mithilfe von Signalen berechnet, die Satelliten aussenden. Da die Positionen der Satelliten wiederum genau bekannt sind, kann der Standort relativ zu den Satelliten ermittelt werden, eine moderne Form der traditionellen Triangulation mit digitalen Bezugspunkten. Astronomische Messungen und hochgenaue bodenbasierte Messverfahren ermöglichen die exakte Positionierung der Satelliten und helfen, die Bewegungen der Erdoberfläche auszugleichen (bei Erdbeben kann sich die Erdoberfläche z.B. ohne Weiteres auf einen Schlag um mehrere Meter verschieben).

Visualisierung der Daten in einer Karte

Mithilfe von GIS-Programmen können Benutzer neu erhobene Geodaten wie z.B. die Positionen von Orten oder Objekten als Karte mit Punkten darstellen oder viele Satellitenbilder zu einem Bildteppich zusammenfügen, sodass die Karte ein großes Ganzes ergibt. Diese Kartenebenen können dann auch mit anderen Inhalten im gleichen Referenzsystem kombiniert werden. Traditionelle handgezeichnete Karten können mit der Geschwindigkeit und Flexibilität dieser digitalen Kartenmanipulation nicht mehr mithalten. Natürlich müssen hierfür erst einmal die nötigen Daten vorliegen. Danach sind im Prinzip jedoch unendlich viele verschiedene Darstellungen möglich, die alle auf den gleichen Grunddaten basieren.

Alle Fehler, die in einem dieser Grundschrte gemacht wurden, beeinflussen das Gesamtergebnis. Wenn Karten auf Grundlage von ungenauen Daten erstellt wurden, ist der gewonnene Aussagewert dieser Karten vielleicht sogar bei technisch anspruchsvollen und schön wirkenden Ergebnissen fraglich. Ziel ist es, die durch die Beobachtungen gewonnenen Informationen möglichst präzise und transparent zu präsentieren. Deshalb müssen die drei Grundschrte besonders gut aufeinander abgestimmt sind.

Was ist beim Umgang mit Geodaten zu beachten?

Karten können heute zu Hause oder im Büro erstellt werden, ohne dafür extra auf eine gefährliche Expedition zur Vermessung neuer Territorien gehen zu müssen. Die traditionellen, nicht digitalen Methoden der Kartenerstellung hatten ihre eigenen Gefahren, die allerdings nicht mit den heutigen Problemen vergleichbar sind. Die aktuellen Stolpersteine haben nichts mehr mit körperlicher Gefahr zu tun, sie führen dagegen zu schlechter Qualität und ungenauen Daten.

Auffinden guter Datenquellen

Eine schlechte Karte kann mehr Schaden anrichten, wenn sie falsche Informationen vermittelt, als wenn es gar keine Karte gegeben hätte. Deshalb beschreibt Kapitel 5 dieses Buchs Bezugsquellen, Art und Qualität der Quelldaten. Es ist sehr wichtig, die Karten auf Grundlage von möglichst geeigneten Quelldaten aufzubauen. Liegen Quelldaten schlechter Qualität vor, können auch nur schlechte Karten und ungenaue Analysen dabei herauskommen. Jede Analyse und die darauf aufbauende Karte benötigen eigene Daten von unterschiedlicher Art. Es gibt nicht »die eine Karte«, sondern eine ganze Fülle unterschiedlicher Inhalte und Darstellungsformen.

Abhängigkeit von digitalen Werkzeugen

Mit der Verbreitung digitaler Karten treten viele Fertigkeiten der traditionellen Kartografie in Vergessenheit. Digitale Mittel reichen zwar aus, um Karten zu erstellen, die traditionelle Kartografie bietet jedoch eine Vielfalt weiterer hilfreicher Methoden, um noch viel bessere Ergebnisse zu erzielen. Man könnte denken, dass eine Ausbildung in digitaler Kartografie auch traditionelle Methoden der Kartenerstellung behandelt. Das ist aber nur selten der Fall. Häufig geht es nur darum, den Umgang mit bestimmten Programmen zu erlernen. Nehmen Sie der Person diese Software weg oder konfrontieren Sie sie mit einem umfangreichen theoretischen Problem, ist sie oft schon überfordert. Die Wahl der richtigen Projektion ist ein gutes Beispiel. Wenn Ihnen der theoretische Hintergrund zum Umgang mit Projektionen fehlt, kann sich die Qualität Ihrer Daten bei einer Umprojektion oder beim Zusammenführen verschiedener Datensätze ebenfalls erheblich verschlechtern (mehr zum Umgang mit Projektionen finden Sie in Kapitel 8 und Anhang A).

Ein weiteres Beispiel ist die Unwissenheit über geometrische Berechnungen. Es war ziemlich beschämend festzustellen, dass ich nicht wusste, wie man den Flächeninhalt eines unregelmäßigen Polygons berechnet. Selbst meine hochgeschätzten GIS-Kollegen mussten hier passen. In Abbildung 2-2 sehen Sie eine Reihe häufig vorkommender geometrischer Formen und die Formeln, die zur Berechnung ihres Flächeninhalts nötig sind.

Viele dieser Berechnungen kennen Sie vielleicht noch aus dem Mathematikunterricht. Die Formel für die Berechnung der Fläche unregelmäßiger Polygone kennen Sie aber vermutlich nicht. Hierfür gibt es keine einfache Formel, und der Flächeninhalt kann auf verschiedene Arten berechnet werden. Eine Methode besteht darin, aus sämtlichen Eckpunkten des Polygons Dreiecke zu erzeugen und danach die Flächeninhalte der Dreiecke zu addieren. Sie sehen schon, warum wir diese Berechnung gern einem Computer überlassen.

Es war recht amüsant festzustellen, dass wir ständig digitale Werkzeuge benutzen, um Flächeninhalte von Polygonen zu berechnen, aber keine Ahnung hatten, wie die Berechnung eigentlich erfolgt. Andererseits war es auch beunruhigend zu sehen, wie viele Aufgaben wir mittlerweile an den Computer weitergegeben haben. Die meisten von uns wären auch nicht einmal in der Lage, die vom Computer gelieferten Antworten zu verifizieren.

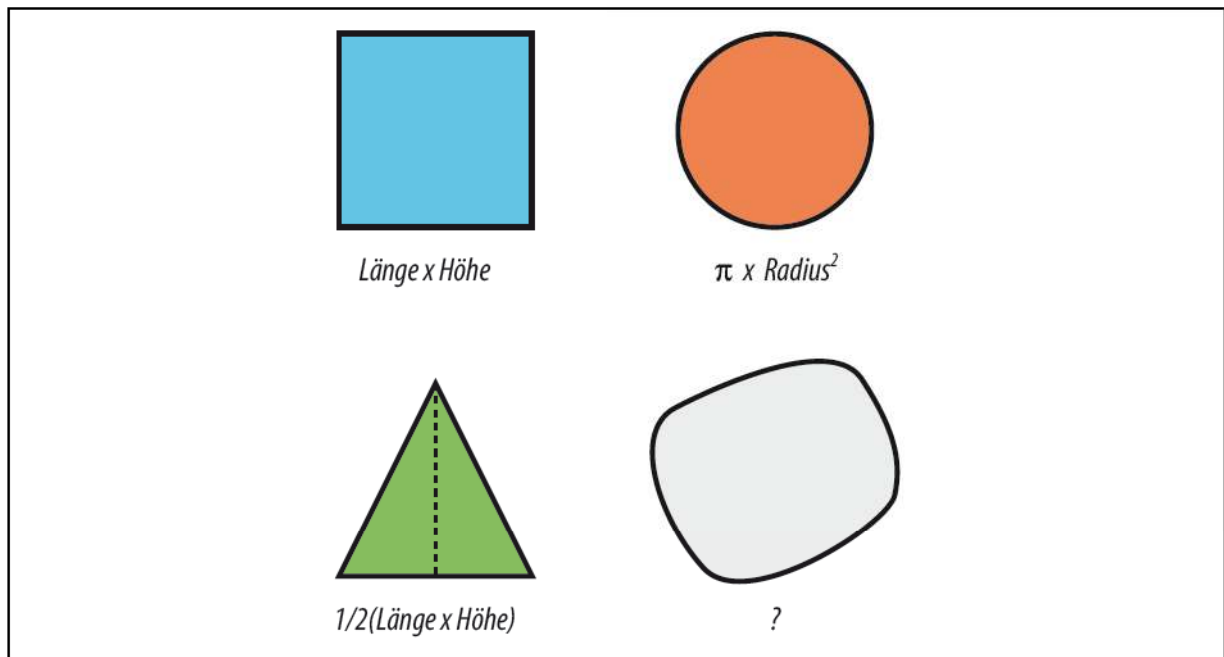


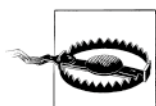
Abbildung 2-2: Berechnung des Flächeninhalts von Quadrat, Kreis, Dreieck und dem berücksichtigten unregelmäßigen Polygon

Digitale Kartografie und GIS sollten mehr sein als nur Methoden, um eine bestimmte Aufgabe zu erfüllen. Mehr theoretisches und praktisches Wissen über die manuellen Hintergründe würden den Erzeuger und Analytiker der Karten in die Lage versetzen, digitale Werkzeuge sinnvoller und effizienter einzusetzen. Die Abhängigkeit von digitalen Werkzeugen macht uns nicht unbedingt schlauer. Dieses Buch behandelt keine kartografischen Grundlagen. Es gibt aber viele gute Texte und Bücher zu diesem Thema, unter anderem die folgenden:

- M.J. Kraak and F.J. Ormeling. *Cartography: Visualization of Spatial Data*. London: Longman. 1996.
- Alan M. MacEachren. *Some Truth with Maps*. Washington: Penn State. 1994.

Datenformate verstehen

Es gibt unterschiedliche Datenformate für alle Typen von Geodaten. Um die Daten effektiv nutzen zu können, müssen Sie wissen, welches Format Ihre Daten haben und welches Format Sie ausgeben möchten. Wenn das aktuelle Format ungeeignet ist, müssen Sie wissen, wie es konvertiert oder anders abgespeichert werden kann.



Die Vielzahl an Programmen und Datenformaten kann schnell verwirren. Viele Hersteller haben ihre eigenen, proprietären Formate entwickelt. Dies bindet Benutzer an Formate, solange sie ein bestimmtes Produkt verwenden. Mittlerweile gibt es aber auch von Seiten der Hersteller Ansätze, unabhängige und übergreifende Standards festzulegen, wodurch verschiedene Anwendungen die Daten verwenden können und der Datenaustausch erleichtert wird. Aber nicht nur das – es wird sogar

möglich, die Daten beim Urheber zu belassen, der sich regelmäßig um die Pflege kümmert. Wenn Sie z.B. einen Straßendatenbestand von Nordrhein-Westfalen in Ihre Karte einbinden möchten, greifen Sie auf den Landesserver zu oder binden den Stadtplandienst des Amts für Geoinformation ein. Sie müssen diese Daten gar nicht kopieren und selbst auf einem Server bereitstellen. Die Technik dafür können Sie jederzeit herunterladen und mithilfe dieses Buchs und etwas Lernaufwand installieren und betreiben. Aber die regelmäßige Pflege allein beschäftigt in einem Land der Größe Deutschlands mehrere hundert, wenn nicht sogar Tausende von Menschen, von der Vermessung bis zum Stadtplaner, der neue Hausnummern vergibt.

Sie sollten sich möglichst genau im Klaren darüber sein, für welche Aufgabe Sie die Quelldaten verwenden wollen. Möchten Sie eine spezielle Karte, die Daten mit anderen Informationen kombinieren, oder soll einfach nur eine Bildschirmausgabe erzeugt werden? Die Antworten auf diese Fragen werden Ihnen helfen, die richtigen Formate für Ihre Aufgabe zu wählen.

Verwendung der richtigen Werkzeuge

Es gibt zahlreiche Werkzeuge für den Umgang mit Geodaten, was die Wahl natürlich nicht gerade vereinfacht. Zum Teil wird die Meinung vertreten, dass nur die größten und teuersten Werkzeuge wirklich gut sind und dass diese Produkte stabiler sind und besser unterstützt werden, obwohl diese Meinung vermutlich nur durch gutes Marketing hervorgerufen wird. Die Auswahl des richtigen Werkzeugs für Ihre Aufgabe kann schwierig sein, wenn das Angebot sehr groß erscheint. Manchmal wird beispielsweise ein Desktop-GIS benutzt, obwohl ein schlichtes Zeichenprogramm ausgereicht hätte. Das wird leicht übersehen, weil wir es hier ja schließlich mit Kartografie und Geografie zu tun haben und daher auch ein entsprechendes GIS-Programm benötigen – oder?

Sie finden sehr einfache und extrem komplexe Programme im Bereich der GIS-Software. Ein gutes Beispiel hierfür ist der Vergleich von CAD-(Computer Aided Design-) und GIS-Anwendern. GIS-Anwender verwenden oftmals sehr modulare, einfache Datenstrukturen. Um daraus ein visuell ansprechendes Produkt zu machen, ist ein ganzes Stück Arbeit nötig. CAD-Programme bieten dagegen eine Vielzahl an Optionen für die grafische Ausgestaltung. Dafür fehlen ihnen häufig einfache Analysewerkzeuge. Die vielen Möglichkeiten in CAD-Programmen, Linien zu gestalten, zu verschieben oder zu verändern, gehen oftmals auf Kosten der Bedienbarkeit und der Verständlichkeit solcher Programme. Andererseits sind die überaus vielfältigen Möglichkeiten zur Datenverwaltung und -analyse einiger GIS-Programme für viele Fragestellungen vollkommen überflüssig.

Ermittlung der Arbeitsschritte eines Projekts

Ähnlich wie Handwerker, wissen auch Kartografen die Wahl des richtigen Werkzeugs für eine bestimmte Aufgabe zu schätzen. Kartografen, die mit digitalen Werkzeugen arbeiten, steht hierbei eine Vielzahl von Werkzeugen zur Auswahl, wobei viele für eine oder zwei Aufgaben besonders gut geeignet sind, während sie andere Aufgaben nur mit größerem Aufwand oder gar nicht bewältigen können. In der digitalen Kartografie und den verwandten Disziplinen gibt es fünf Kategorien von Werkzeugen, deren Anwendungsgebiete sich oftmals überschneiden.

Anzeige von Geodaten in Karten

Die Anzeige von Geodaten ist der erste Schritt und kann auch Vorbereitung für die Erstellung von Karten sein. Einige Programme sind nur für die Visualisierung von Daten vorgesehen, während andere auf die Erstellung von Karten ausgerichtet sind. Die Kartenherstellung zielt hierbei eher auf hochwertige gedruckte Karten ab. In diesem Buch verwenden wir den Begriff *Betrachtungswerkzeuge*, wenn wir von Programmen sprechen, mit denen Geodaten visuell dargestellt werden können – wie sind die Daten angeordnet, welchen räumlichen Bereich decken sie ab, wie verhalten sie sich zu anderen Daten usw.

Andere Werkzeuge (*Web-Mapping-Tools*) werden als Software für die Veröffentlichung von Geodaten im Internet genutzt, zum Beispiel Kartendienste. Diese Werkzeuge können zwar auch eingesetzt werden, um eine Karte auszudrucken, die Möglichkeiten der kartografischen Feinarbeit sind aber begrenzt. Die Konzepte der Anzeige und der Kartengenerierung lassen sich dennoch thematisch zusammenfassen, weil beide eine grafische Ausgabe erzeugen. Oftmals ist die grafische Ausgabe das Endprodukt der Arbeitsschritte der folgenden Kategorien.

Geodaten analysieren

Selten ist die reine Betrachtung von Karten das alleinige Ziel eines Projekts. Damit die Anzeige verständlicher oder repräsentativer wird, sind oftmals verschiedene Datenanalysen notwendig. Hierzu gehören die Klassifizierung der Daten (Gruppierung ähnlicher Objekte in Kategorien), Entfernungsberechnungen (Objekte mit einem bestimmten räumlichen Abstand zueinander) und statistische Zusammenfassungen (Gruppierung von Daten mittels statistischer Funktionen wie Durchschnitt oder Summe). Analysen fassen Daten temporär zusammen, während eine Manipulation die Daten dauerhaft verändert und neue Daten erzeugen kann.

Geodaten erzeugen und bearbeiten

Die Erzeugung von Geometrien wird als Digitalisierung bezeichnet. Die Geometrien können beispielsweise durch eine Analyse erzeugt worden sein, bei der Daten eines bestimmten Gebiets herausgefiltert wurden.

Für die Be- und Verarbeitung der Daten gibt es eine Reihe von Programmen, diese reichen von kommandozeilenbasierten Anwendungen bis hin zu Drag-and-Drop-fähigen Desktop-Programmen. Viele Visualisierungsprogramme können die Daten nur lesen und nicht bearbeiten. Manche Digitalisierungsprogramme erzeugen neue Daten durch Zeichnen auf dem Bildschirm (digitalisieren). Einige Produkte haben weitergehende Fähigkeiten. Sie können Geometrien konstruieren, Daten puffern, Schnittmengen berechnen, Geometrien zusammenfassen und vieles mehr. Im Open Source-Bereich finden Sie eine Reihe leistungsstarker Desktop-GIS-Programme mit verschiedenen Schwerpunkten. Durch die Integration von GRASS GIS in Quantum GIS stehen Ihnen komplexe Digitalisierungsfunktionen und Berechnungsmöglichkeiten zur Verfügung. In den Kapiteln 8 und 9 wird Quantum GIS vorgestellt.

Datenkonvertierung

Für bestimmte Anwendungen müssen die Daten in einem ganz bestimmten Datei- oder Datenbankformat vorliegen. Dies trifft oft bei proprietären Herstellern zu, deren Anwendungen nur die eigenen Formate unterstützen und keine oder nur minimale Unterstützung für andere Formate bieten. Der Einsatz proprietärer Formate führt zu einer langfristigen Abhängigkeit von einem bestimmten Produkt. Inzwischen haben die Anwender von ihren Herstellern eingefordert, dass auch Datenformate der Mitbewerber unterstützt werden, um den Datenaustausch zwischen Programmen zu erleichtern. Dies wiederum führte zur Entwicklung von herstellerunabhängigen Standards durch Organisationen wie dem Open Geospatial Consortium (OGC), deren Ziel die Interoperabilität zwischen Applikationen ist. Mehr Details zur Rolle des OGC und der durch das OGC entwickelten Standards finden Sie in Kapitel 12.

Quelldaten liegen nicht immer in dem Format vor, das von Ihrem GIS-Programm benötigt wird. Wenn Sie Daten erhalten, die mit einem anderen GIS-Programm erzeugt wurden, ist es sehr wahrscheinlich, dass die Daten konvertiert werden müssen. Daten, die durch Manipulationsprozesse erzeugt wurden, haben ebenfalls nicht immer das Format, das benötigt wird. Hier kommen die Konvertierungswerkzeuge zum Einsatz.

Konvertierungsprogramme helfen dabei, Daten in das benötigte Format zu überführen. Für diese Aufgabe steht eine Reihe hervorragender Werkzeuge zur Verfügung. Es gibt Bibliotheken, über die Programme direkt auf die Quelldaten zugreifen können, ohne die Daten vorher konvertieren zu müssen. Einige dieser Bibliotheken werden wir in diesem Buch noch kennenlernen (GDAL/OGR). Ein hervorragendes, wenn auch proprietäres Konvertierungswerkzeug ist die Feature Manipulation Engine (FME) der Firma Safe Software (siehe <http://safe.com>).

Datenaustausch

Vielleicht lesen Sie dieses Buch, weil Sie Karten und Geodaten bereitstellen und austauschen möchten. Es kann Spaß machen, eigene Karten zu erstellen und zu veröffentlichen. Da es mittlerweile eine große Auswahl an frei verfügbaren Werkzeugen und freien Datenquellen gibt, ist dies kein Traum mehr.

In diesem Buch werden zwei Arten des Datenaustauschs aufgeführt: zum einen der Austausch von Karten (statisch oder interaktiv) mithilfe von Webapplikationen, zum anderen die Verwendung von standardisierten Diensten (z.B. OGC WMS und WFS) für den Datenaustausch.

Der Begriff des *Web-Mapping* umfasst ein weites Feld von Anwendungen und Prozessen. Eine einfache Webseite mit einem Satellitenbild kann genauso gemeint sein wie etwa eine Flash-basierte Anwendung mit Interaktionen, Animationen und vielleicht sogar Toneffekten. Meistens geht es aber um eine Webseite, in der eine interaktive Karte enthalten ist. Oft kann der Benutzer aus einer Liste von Ebenen (Layers) auswählen und diese Ebenen ein- und ausblenden. Mithilfe von Navigationswerkzeugen kann der Benutzer die Karte verschieben und in die Karte hineinzoomen, um mehr Details anzuzeigen.

Das Open Geospatial Consortium (OGC) hat eine Reihe von Standards für Webservices (OWS) entwickelt. Diese Standards ermöglichen es Web-Mapping-Anwendungen, Daten untereinander oder mit anderen Programmen, die ebenfalls die Standards befolgen, auszutauschen. Eine Anwendung kann dabei webfähig sein und selbst nicht über eine grafische Oberfläche verfügen. Stattdessen können andere Anwendungen mithilfe der Standards Daten von einem entfernten Webservice anfordern. Diese Interoperabilität ermöglicht es Programmen, miteinander zu kommunizieren, auch wenn sie auf verschiedenen Systemen basieren. Der Einsatz standardisierter Dienste setzt sich mehr und mehr durch. Standardisierte Dienste ermöglichen es, direkt auf die Quelldaten zuzugreifen, und machen große Datensammlungen nicht mehr notwendig. Außerdem haben Sie beim Einsatz von standardisierten Diensten mehr Freiheit bei der Auswahl der verwendeten Programme und deren Möglichkeiten. OWS ist ein offener Standard zum Austausch und für den Zugriff auf Informationen. Anstatt vom Datenformat eines bestimmten Herstellers abhängig zu sein, muss ein Programm nur noch OWS unterstützen. Abbildung 2-3 zeigt beispielsweise eine Karte, die aus unterschiedlichen Datenquellen erstellt wurde. Einige Ebenen sind Kopien von Daten, auf die das Mapping-Programm direkt zugreift. Die anderen beiden Ebenen stammen von einer OWS-Datenquelle: eine von einer Firma aus den Niederlanden, die andere (Wetter-Radar-Daten) von einer Universität in den Vereinigten Staaten.

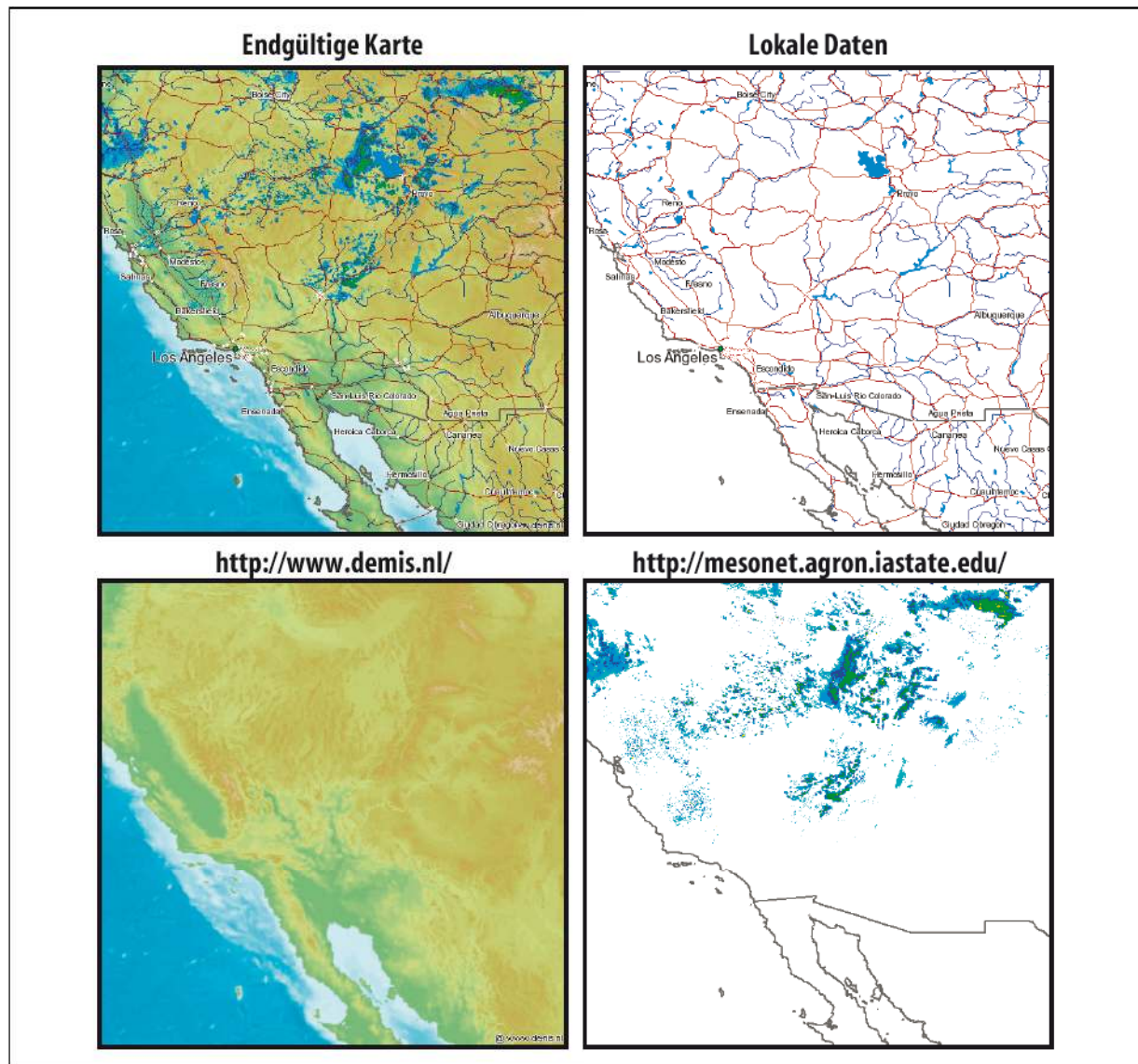


Abbildung 2-3: Karte, die sich aus verschiedenen OWS-Datenquellen entfernter Server aufbaut

Daten konvertieren und betrachten

Karten im Web zu präsentieren ist eine feine Sache, doch müssen die Daten für diese Karten auch irgendwoher kommen. Um Karten im Web zu präsentieren, müssen zunächst die Geodaten aufbereitet werden. Hierfür benötigen Sie Werkzeuge, mit deren Hilfe Sie die Geodaten erstellen, gegebenenfalls konvertieren und nach Ihren Anforderungen anpassen können. Vor allem wenn Sie Ihre Anwendung in einer Umgebung entwickeln, die noch nicht GIS-orientiert ist, haben Sie hier freies Feld und sind nicht auf eine bestehende Infrastruktur festgelegt. Dieses Kapitel stellt eine Übersicht aus der große Auswahl an frei verfügbaren Open Source-Programmen bereit, mit denen Sie Daten visualisieren und gemeinsam nutzen können. Des Weiteren werden Sie hier einige einfache Werkzeuge kennenlernen, über die Sie auf Daten zugreifen und Daten konvertieren können.

Die Auswahl der Open Source-GIS- und Mapping-Werkzeuge in diesem Kapitel beschränkt sich auf die Programme, die auch im weiteren Verlauf dieses Buchs zum Einsatz kommen. Diese hier beschriebenen Werkzeuge eignen sich für den professionellen Einsatz und sind bereits in vielen Bereichen erfolgreich im Gebrauch.

Zusätzliche Informationen und Links zu weiteren Werkzeugen finden Sie auf den folgenden Websites:

<http://osgeo.org>

<http://freegis.org>

<http://opensourcegis.org>

<http://maptools.org>

Sie können selbstverständlich auch proprietäre Software verwenden, um die Daten zu erzeugen, die Sie hier mithilfe von Open Source-Programmen bearbeiten. Einer der größten Vorteile der Open Source-Werkzeuge besteht darin, dass viele in der Lage sind, auch mit Daten zu arbeiten, die von proprietären Programmen stammen oder in proprietären Formaten vorliegen.

Rasterdaten und Vektordaten

Die Begriffe Raster und Vektor werden Ihnen im Zusammenhang mit Geodaten immer wieder begegnen. Sie beziehen sich auf zwei unterschiedliche Arten von Daten.

Rasterdaten sind in Form einer Matrix oder eines Gitters aus Spalten und Zeilen organisiert. Aus den Spalten und Zeilen ergeben sich Pixel bzw. Zellen. Jede Zelle enthält die gleiche Art von Information, z.B. einen Farbwert, eine Temperaturmessung, die Höhe oder auch die Kaufkraft der Bewohner des entsprechenden Gebiets. Luftbilder und digitale Höhenmodelle sind Rasterdaten, die durch Bildaufnahmen oder Laserscanning erhoben werden. Sie sind eine bestimmte Anzahl von Pixeln hoch und breit, wobei ein Pixel einen bestimmten Bereich auf der Erde widerspiegelt. Landsat-Bilder sind beispielsweise ca. 6,166 Pixel hoch und breit und decken damit eine Fläche von 185 x 185 Kilometern ab. Daraus ergibt sich, dass jedes einzelne Pixel eine Fläche von 30 x 30 Metern repräsentiert.

Vektordaten werden dagegen aus Koordinatenlisten aufgebaut. Jeder Punkt hat eine Koordinate, aus mehreren verbundenen Stützpunkten können sich Linien und Polygone (Flächen) ergeben. Vektordaten sind oft auch mit Sachinformationen verknüpft. Dabei enthält jedes Geometrieobjekt (Punkt, Linie, Polygon) eigene Informationen.

Je nach Datentyp können auf den Geodaten unterschiedliche Operationen ausgeführt werden. Früher wurde deshalb zwischen Raster- und Vektor-GIS unterschieden, heute können viele Systeme beide Arten von Daten verarbeiten.

Quantum GIS

Quantum GIS (QGIS) ist ein Open Source-Desktop-GIS, das zahlreiche Vektor- und Rasterdatenformate unterstützt. Die GDAL/OGR-Bibliothek wird von Quantum GIS verwendet, sodass die Formate, die GDAL/OGR unterstützt, von Quantum GIS angesprochen werden können. Quantum GIS wird in C++ entwickelt und besitzt eine Plugin-Architektur, über die weitere Funktionalitäten einfach ergänzt werden können. Das Programm wird in den Kapiteln 8 und 9 genauer vorgestellt.

OpenEV

OpenEV ist ein Desktop-GIS, das die Geospatial Data Abstraction Library (GDAL/OGR) einbindet. Dadurch unterstützt OpenEV automatisch alle in GDAL/OGR definierten Raster- und Vektorformate. OpenEV kann sowohl für Bildanalysen als auch für die Erstellung einfacher Karten und die Erzeugung neuer Daten eingesetzt werden. OpenEV ist Teil des FWTools-Pakets, das Sie sich unter der Adresse <http://fwtools.maptools.org> herunterladen können.

OpenEV kann auch 3-D-Visualisierungen in Echtzeit erzeugen, was es von anderen GIS abhebt. In Abbildung 3-1 sehen Sie ein Beispiel für die Darstellung eines solchen Bilds in OpenEV. Abbildung 3-2 zeigt das gleiche Bild, nachdem es mit OpenEV aufbereitet wurde.

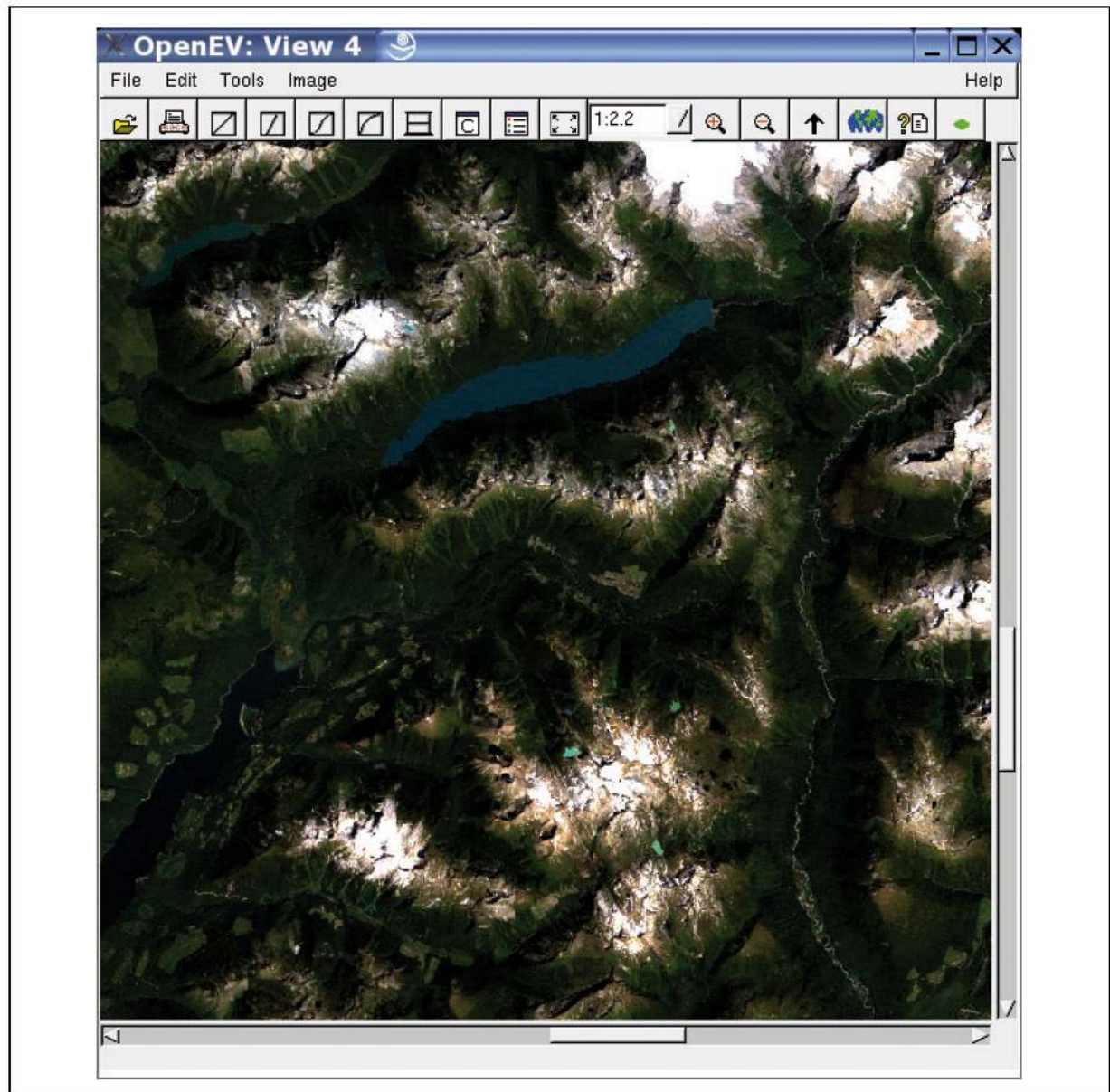


Abbildung 3-1: Ein unbearbeitetes Landsat-Bild in der Darstellung durch OpenEV

OpenEV ist ein gutes Beispiel dafür, wie Open Source funktioniert. Die GDAL/OGR-Bibliothek wird von anderen Entwicklern programmiert und kann trotzdem auch von weiteren Programmen genutzt werden (z.B. von Quantum GIS, siehe oben). OpenEV wurde in der Programmiersprache Python implementiert und ist deshalb einfach erweiterbar, weil die Möglichkeiten von Python intern nutzbar sind. Dies erlaubt den Zugriff auf die GDAL/OGR-Funktionen, aber auch auf jedes andere Python-Modul.

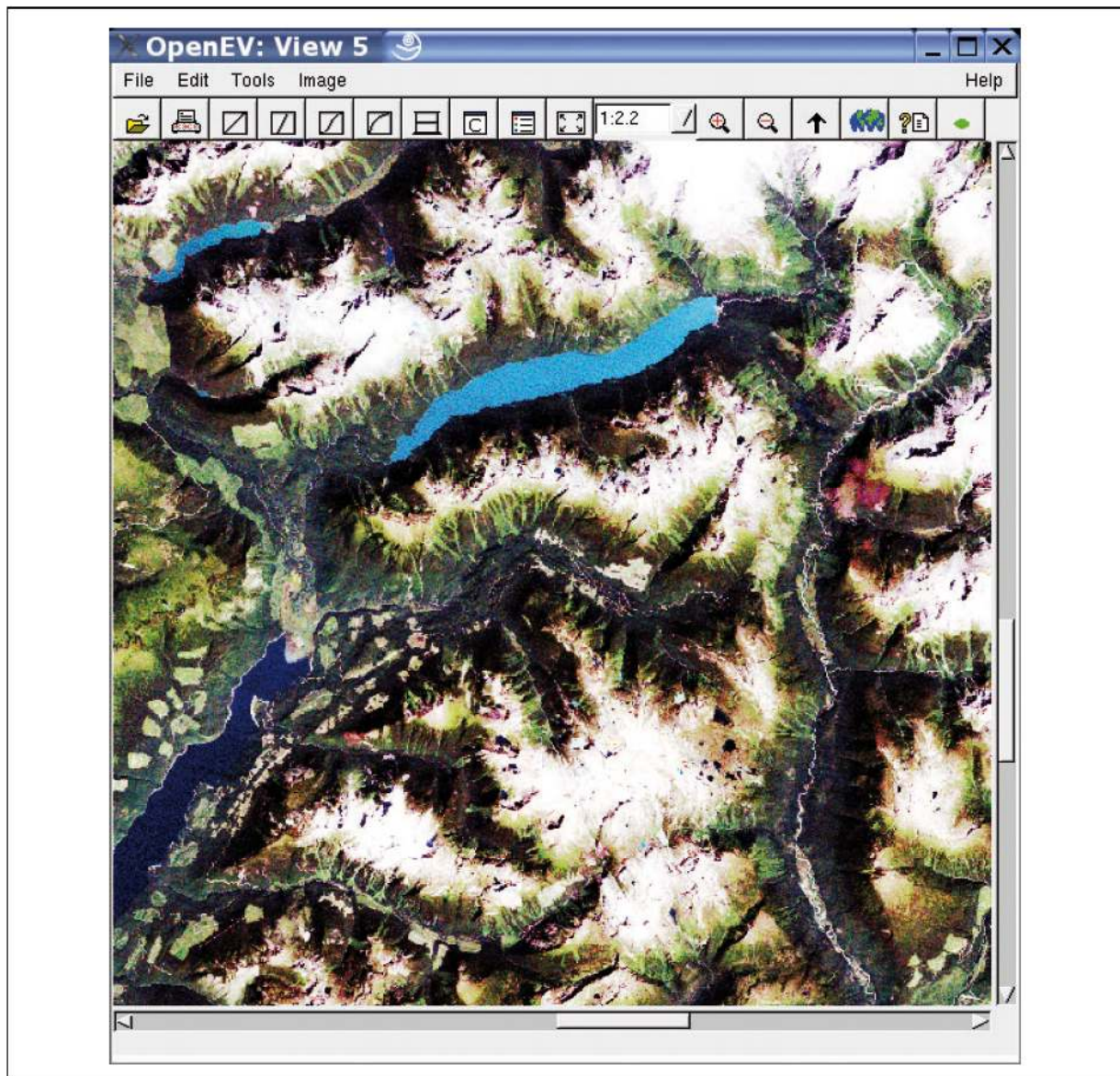


Abbildung 3-2: Das Landsat-Satellitenbild in der Darstellung durch OpenEV, nachdem es mit einem Equalizer-Werkzeug bearbeitet wurde

UMN MapServer

MapServer, auch als UMN MapServer bekannt, ist das vorrangig in diesem Buch verwendete Open Source-Werkzeug zur Generierung von Karten. Die offizielle MapServer-Website finden Sie unter <http://mapserver.osgeo.org>.

Es gibt viele Gründe, sich für MapServer zu entscheiden, dazu zählen die hohe Leistung, Stabilität und nicht zuletzt die große Anwendergemeinschaft. Ein weiterer Grund ist, dass MapServer die Möglichkeit bietet, eigene räumliche Daten leicht über das Web zugänglich zu machen. Wenn MapServer als Kartendienst betrieben wird, können handelsübliche Webbrowser zur Anzeige und Analyse der Karten genutzt werden. Der Betrachter muss selbst keine spezielle Software mehr installieren.

Ein anderer Grund, MapServer zu wählen, ist die vielfältige Unterstützung unterschiedlicher Formate. Durch die Verwendung der Bibliotheken GDAL/OGR kann MapServer auf viele Datenformate direkt zugreifen, ohne dass diese zuvor konvertiert werden müssen.

Stellen Sie sich vor, Sie haben zehn verschiedene Datenformate vorliegen, die in einer gemeinsamen Karte dargestellt werden sollen, ohne dass die Daten aus ihrem ursprünglichen Format konvertiert werden müssen. Die Formate können auch von verschiedenen proprietären Anbietern stammen. Mit MapServer können Sie beispielsweise ESRI Shape, Intergraph Microstation Design-Dateien (DGN), MapInfo TAB-Dateien und Daten aus einer Oracle-Spatial-Datenbank gemeinsam verwenden, ohne dass die Formate vorher umgewandelt werden müssen. Selbstverständlich werden alle offenen Formate unterstützt. MapServer kann auf Standarddaten im OGC-Format SFS in Datenbanken zugreifen, wie sie von PostgreSQL/PostGIS, inzwischen aber auch von Oracle Spatial unterstützt werden. Die OGC-Standards Geography Markup Language (GML), Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS) und viele weitere werden ebenfalls unterstützt.

Datenzugriff und Performance

Viele Datenformate werden vom MapServer-Executable direkt unterstützt. Auf andere Formate kann mittels der GDAL/OGR-Bibliothek zugegriffen werden. Allerdings wird für den Datenzugriff über Erweiterungsbibliotheken eine zusätzliche Kommunikations-ebene benötigt (was in einigen Fällen zu einem Performanceverlust führen kann).

Normalerweise ist der Zugriff auf direkt von MapServer unterstützte Formate schneller als auf solche, die über die Bibliothek GDAL/OGR angesprochen werden. Beispielsweise können proprietäre Shape-Dateien und GeoTiff direkt von MapServer verarbeitet werden. Die Unterstützung für das TIGER-Format benötigt dagegen die OGR-Bibliothek. Der Geschwindigkeitsunterschied beim Zugriff auf TIGER-Dateien (ein US-amerikanisches Format für Straßendaten) und Shape-Dateien kann signifikant sein. Der eigentliche Flaschenhals liegt hierbei in dem TIGER-Datenformat, das zwar vielfältige Informationen enthält, aber so strukturiert ist, dass der Zugriff erst über mehrere Interpretationsebenen laufen muss, bevor er dargestellt werden kann. Solche Eigenschaften können deutlichen Einfluss auf die Geschwindigkeit und Leistung einer Anwendung haben.

Als Faustregel können Sie sich merken, dass Daten im ESRI Shape-Format und GeoTiff-Dateien am schnellsten verarbeitet werden. Dank der Hilfsprogramme `gdal_translate` und `ogr2ogr` können die Daten zur Performancesteigerung in diese Formate konvertiert werden. Wenn Sie über ein Netzwerk auf die Daten zugreifen, ist die Speicherung in einer PostgreSQL/PostGIS-Datenbank oftmals die beste Wahl. Da PostgreSQL/PostGIS Ihre Datenanfragen direkt auf dem Server verarbeitet, müssen nur die benötigten Ergebnisse über das Netzwerk übertragen werden. Werden die Daten dagegen in einfachen Dateien abgelegt, müssen mehr Informationen im Netzwerk übertragen werden, bevor MapServer entscheidet, was tatsächlich gebraucht wird. Daher kann die serverseitige Verarbeitung in einer PostgreSQL/PostGIS-Datenbank die Leistungsfähigkeit einer MapServer-Applikation deutlich steigern.

Die Konvertierung aller Daten ist nicht immer möglich und auch nicht erforderlich, dennoch können diese Hinweise zur Performancesteigerung hilfreich sein.

Portabilität

MapServer und die dazugehörigen Hilfsprogramme werden von vielen Plattformen und Betriebssystemen unterstützt. Darüber hinaus kann auf die Funktionalität von MapServer über eine Reihe von Programmierschnittstellen zugegriffen werden. Auf diese Weise können die Fähigkeiten von MapServer auch in eigenen Programmen genutzt werden.

Da MapServer Open Source-Software ist, können Entwickler den Programmiercode direkt anpassen, Probleme beheben und die Software auf neue Betriebssysteme und Plattformen portieren. Benötigen Sie eine bestimmte Funktionalität, können Sie einen Programmierer engagieren, der die Funktionalität einbaut, und die gesamte Nutzergemeinschaft kann anschließend von der Erweiterung profitieren.

In erster Linie ist MapServer ein Programm zum Erstellen und Betrachten von Karten. Benutzer können mithilfe des Browsers auf Karten zugreifen. Daten können so von verschiedenen Anwendern gemeinsam genutzt werden. MapServer ist in der Lage, eine Pseudokonvertierung von Daten durchzuführen, indem verschiedene Formate eingelesen und standardisierte Formate ausgegeben werden, die wiederum anderen Anwendungen zur Verfügung stehen. MapServer ist kein Analysewerkzeug, kann aber zur Visualisierung von Berechnungsergebnissen eingesetzt werden.

Die GDAL – Geospatial Data Abstraction Library

GDAL ist Teil des FWTools-Pakets, das unter der Adresse <http://fwtools.maptools.org> heruntergeladen werden kann. Auf der GDAL-Homepage (<http://www.gdal.org>) wird das Projekt wie folgt beschrieben:

... eine Übersetzungsbibliothek für räumliche Rasterdatenformate ... Die Bibliothek bietet den aufrufenden Programmen ein einfaches abstraktes Datenmodell für alle unterstützten Formate.

GDAL besitzt drei wichtige Eigenschaften. Erstens unterstützt die Bibliothek über 60 verschiedene Rasterformate. Zweitens kann sie von anderen Programmen eingebunden werden. Programme, die die GDAL-Bibliotheken verwenden, können ohne zusätzlichen Programmieraufwand auf alle unterstützten Formate zugreifen. Drittens helfen Ihnen die beiliegenden Hilfsprogramme, die GDAL-Programmierbibliotheken zu benutzen, ohne dass Sie eigene Programme schreiben müssen.

Diese drei Eigenschaften machen GDAL sehr leistungstark. Sie brauchen sich keine Gedanken mehr darüber zu machen, in welchem Format ein Bild vorliegt. Da GDAL bereits Dutzende von Formaten beherrscht, ist es sehr wahrscheinlich, dass auch die von Ihnen benötigten unterstützt werden. Ob Sie nun Daten umwandeln wollen, Bilder in eigenen Programmen darstellen möchten oder einen neuen Treiber für ein eigenes

Bildformat entwickeln – GDAL besitzt Programmierschnittstellen und Hilfsprogramme, die Ihnen bei Ihrer Aufgabe zur Seite stehen.

Von GDAL unterstützte Rasterformate

GDAL unterstützt mehrere Dutzend Rasterformate. Die im Folgenden abgedruckte Liste stammt von der GDAL-Formatliste, die Sie auch im Web unter der Adresse http://www.gdal.org/formats_list.html finden können. Die Liste bietet eine Auswahl der von GDAL unterstützten Formate. Sie kann sämtliche hier aufgelisteten Formate lesen, jedoch nicht alle schreiben. Die Formatliste auf der GDAL-Website gibt ebenfalls Auskunft darüber, für welche Formate lesender Zugriff möglich ist und welche Formate geschrieben werden können.

- Arc/Info Binary Grid (.adf)
- Microsoft Windows Device Independent Bitmap (.bmp)
- BSB Nautical Chart Format (.kap)
- VTP Binary Terrain Format (.bt)
- CEOS (zum Beispiel Spot)
- First Generation USGS DOQ (.doq)
- New Labelled USGS DOQ (.doq)
- Military Elevation Data (.dt0, .dt1)
- ERMapper Compressed Wavelets (.ecw)
- ESRI .hdr labeled
- ENVI .hdr labeled Raster
- Envisat Image Product (.n1)
- EOSAT FAST Format
- FITS (.fits)
- Graphics Interchange Format (.gif)
- Arc/Info Binary Grid (.adf)
- GRASS Rasters
- TIFF/GeoTIFF (.tif)
- Hierarchical Data Format Release 4 (HDF4)
- Erdas Imagine (.img)
- Atlantis MFF2e
- Japanese DEM (.mem)
- JPEG, JFIF (.jpg)
- JPEG2000 (.jp2, .j2k)
- NOAA Polar Orbiter Level 1b Data Set (AVHRR)
- Erdas 7.x .LAN und .GIS
- In Memory Raster
- Atlantis MFF
- Multi-resolution Seamless Image database
- NITF
- NetCDF

OGDI Bridge
PCI *.aux* labeled
PCI Geomatics database file
Portable Network Graphics (*.png*)
Netpbm (*.ppm*, *.pgm*)
USGS SDTS DEM (**CATD.DDF*)
SAR CEOS
USGS ASCII DEM (*.dem*)
X11 Pixmap (*.xpm*)
OGC WCS
OGC WMS

Diese Liste ist umfangreich, aber sicher nicht statisch. Wird ein von Ihnen benötigtes Format noch nicht unterstützt, wenden Sie sich einfach an die Entwickler. Manchmal ist nur ein geringer Programmieraufwand notwendig, um das von Ihnen gewünschte Format ebenfalls einzubinden.

GDAL-Bibliotheken

Eine der wichtigsten Eigenschaften von GDAL ist die Bereitstellung von Bibliotheken. Diese können von verschiedenen Programmiersprachen genutzt werden. Um die von GDAL unterstützten Formate nutzen zu können, ist keine eigene Programmierung notwendig. Wiederverwendbarkeit ist eine der großen Stärken dieser Bibliothek.

Die Dokumentation der Programmierschnittstelle (API) von GDAL enthält Beispiele für den Zugriff auf Rasterdaten für die Sprachen C, C++ und Python. Für weitere Sprachen wie Perl, Java, C# usw. können Sie den Simplified Wrapper and Interface Generator (SWIG) verwenden. Weitere Informationen zu SWIG finden Sie unter <http://www.swig.org/>.

Die Möglichkeit, die GDAL-Bibliotheken direkt einzubinden, hat dazu beigetragen, dass viele GIS-Programme (sowohl proprietäre und als auch Open Source-Programme) um neue Fähigkeiten erweitert wurden. Auf der GDAL-Website finden Sie über 30 Projekte, die GDAL einsetzen, unter anderem FME, MapServer, GRASS, Quantum GIS, gvSIG, Google Earth und ArcGIS 9.2.x.

GDAL-Hilfsprogramme

Das Projekt GDAL/OGR liefert die Bibliothek zusammen mit ebenso mächtigen Werkzeugen aus. Hierzu gehören kommandozeilenbasierte Programme für Datenzugriff und Datenmanipulation. Die folgenden Abschnitte sind eher technischer Natur und erläutern, wie diese Programme aufgerufen werden, welche Parameter sie benötigen und welche Ergebnisse sie liefern.

gdalinfo

Untersucht eine Rasterdatei und gibt Informationen über die Datei zurück. Dem Befehl wird der Name der Datei oder der Zugang zu einer Datenquelle übergeben. Als Ergebnis gibt gdalinfo eine Liste mit verschiedenen statistischen Informationen über die enthaltenen Daten zurück.

```
# gdalinfo vancouver.tif
Driver: GTiff/GeoTIFF
Size is 1236, 1028
Coordinate System is:
PROJCS["NAD83 / UTM zone 10N",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS 1980",6378137,298.2572221010042,
        AUTHORITY["EPSG","7019"]],
      AUTHORITY["EPSG","6269"]],
    PRIMEM["Greenwich",0],
    UNIT["degree (supplier to define representation)",0.01745329251994328],
    AUTHORITY["EPSG","4269"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",-123],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AUTHORITY["EPSG","26910"]]
Origin = (480223.000000,5462627.000000)
Pixel Size = (15.00000000,-15.00000000)
Corner Coordinates:
Upper Left  ( 480223.000, 5462627.000) (123d16'19.62"W, 49d18'57.81"N)
Lower Left  ( 480223.000, 5447207.000) (123d16'16.88"W, 49d10'38.47"N)
Upper Right ( 498763.000, 5462627.000) (123d 1'1.27"W, 49d18'58.96"N)
Lower Right ( 498763.000, 5447207.000) (123d 1'1.10"W, 49d10'39.61"N)
Center      ( 489493.000, 5454917.000) (123d 8'39.72"W, 49d14'48.97"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
```

Verschiedene wichtige Informationen werden ausgegeben: Bildformat, Größe, verwendete Projektion (sofern angegeben), Darstellungsbereich, Anzahl der Farben sowie Pixelgröße. All diese Informationen können bei der Arbeit mit Daten sehr hilfreich sein. Dies gilt besonders für die Arbeit mit Daten externer Quellen, über die nur wenige Informationen vorliegen.

gdal_translate

Überführt eine Rasterdatei von einem Format in ein anderes. Das Werkzeug besitzt eine Reihe von Funktionalitäten, z.B. für die Änderung der Bildgröße, die Georeferenzierung und das Erzeugen von Bildausschnitten. gdal_translate ist besonders nützlich, wenn die Bilder sehr groß sind oder mit anderer Software nur schwer bearbeitet werden können.

```
# gdal_translate -of "JPEG" canada_mosaic_lcc_1000m.tif canada.jpeg
Input file size is 5700, 4800
0...10...20...30...40...50...60...70...80...90...100 - done.
```

gdaladdo

Fügt einer Rasterdatei interne Übersichtsebenen hinzu. Die Berechnung dieser Übersichten kann über verschiedene Algorithmen erfolgen. Die internen Übersichten steigern die Performance bei der Anzeige der Daten. Programme, die diese Übersichtsebenen lesen können, können entsprechend nur die geeignete Auflösung für die Darstellung in einem bestimmten Maßstab anfordern, ohne dass das Bild in der vollen Auflösung in den Speicher geladen werden muss.

```
# gdaladdo -r average canada_mosaic_lcc_1000m.tif 2 4 8 16
0...10...20...30...40...50...60...70...80...90...100 - done.
```

gdalwarp

Aus einer Quelldatei wird ein Bild erzeugt, das in einem anderen Projektionssystem vorliegt. Dies kann hilfreich sein, wenn die Quelldaten in einem anderen Projektionssystem vorliegen als das benötigte. Zum Beispiel kann ein georeferenziertes Satellitenbild in UTM-Projektion vorliegen, es wird jedoch in WGS84 benötigt. Mithilfe von `gdalwarp` können Sie diese Umwandlung leicht durchführen.

```
# gdalwarp -t_srs EPSG:4326 canada.tif canada_4326.tif
Creating output file that is 10452P x 3133L.
Processing input file canada.tif.
:0...10...20...30...40...50...60...70...80...90...
```

gdal_merge.py

Ein sehr mächtiges Werkzeug, mit dem mehrere Einzelbilder zu einem Ausgabebild zusammengefügt werden können. Hierbei handelt es sich um ein Python-Skript, für das die Python-Interpretersoftware installiert sein muss. Python liegt allen gängigen Linux-Distributionen bei, ist aber auch für Windows erhältlich. Weitere Informationen zur Programmiersprache Python finden Sie unter <http://python.org/>. Im aktuellen Download-Paket der FWTools sind GDAL und Python bereits enthalten. Siehe <http://fwtools.maptools.org/>.

```
# python gdal_merge.py -o ausgabebild.tif bild1.tif bild2.tif
```

gdaltindex

Fügt einer vom Programm neu erzeugten Index-Shape-Datei oder einer bestehenden Index-Shape-Datei den Darstellungsbereich eines Rasterbilds als Polygon hinzu. Diese Index-Shape-Dateien werden auch als Bildkataloge bezeichnet. Als Parameter übergeben Sie `gdaltindex` die Namen der Bilddateien. GDAL überprüft den Darstellungsbereich der Bilder und erzeugt aus diesen Informationen ein Polygon in der Shape-Datei. Zusätzlich wird zu jedem Polygon der Dateiname des Bilds oder der Dateipfad gespeichert. Dieser Bildkatalog kann von MapServer verwendet werden, um eine Ebene zu definieren, die mehrere Bilder über den Bildkatalog anspricht. Der Bildkatalog kann als virtuelle Bild-Datenquelle angesehen werden, die es MapServer ermöglicht, schnell das richtige Bild für die Anzeige zu finden.

```
# gdaltindex index_dgk.shp *tif
Creating new index file...
```

GDAL hilft beim Zugriff, der Konvertierung und der Manipulation von Rasterdaten. Bei weiterer Programmierung, z.B. mit Python, kann GDAL auch sehr gut zur Analyse verwendet werden.



Wenn Sie die GDAL-Hilfsprogramme als Teil der FWTools für Windows installieren, finden Sie auf dem Desktop ein Icon mit dem Namen FWTools Shell. Dieses startet ein Befehlszeilenfenster, in dem die Hilfsprogramme direkt ausgeführt werden können.

Die OGR Simple Features Library

OGR ist ebenfalls Teil des FWTools-Pakets. Die Homepage des OGR-Projekts (siehe <http://www.gdal.org/ogr>) beschreibt OGR so:

OGR ist eine C++-Open Source-Bibliothek (und beinhaltet Kommandozeilenwerkzeuge), die lesenden (und zum Teil auch schreibenden) Zugriff für eine Reihe von Vektorformaten ermöglicht, einschließlich ESRI Shape, S-57, SDTS, PostGIS, Oracle Spatial, Mapinfo mid/mif sowie einige TAB-basierte Formate.



OGR steht ursprünglich für OpenGIS Simple Features Reference Implementation. Diese Bedeutung trifft mittlerweile nicht mehr zu, der Name OGR wird aber weiterhin verwendet.

OGR unterstützt über 25 verschiedene Vektorformate und enthält ebenfalls Hilfsprogramme, die den GDAL-Werkzeugen ähneln.

Von OGR unterstützte Vektorformate

Die folgende Liste der von OGR unterstützten Vektordatenformate stammt von der OGR-Website unter http://www.gdal.org/ogr/ogr_formats.html. Die Liste auf der OGR-Website führt zusätzlich auf, welche der Formate nicht nur gelesen, sondern auch geschrieben werden können.

- Arc/Info Binary Coverage
- ATLAS BNA
- Comma Separated Values (.csv)
- DWG
- DXF
- ESRI Personal GeoDatabase
- ESRI ArcSDE
- ESRI shapefile
- DODS/OPeNDAP
- FMEObjects Gateway
- GML
- GMT
- GRASS

Informix DataBlade
INTERLIS
KML
IHO S-57 (ENC)
Mapinfo file
Microstation DGN
OGDI vectors
ODBC
Oracle Spatial
PostgreSQL
SDTS
SQLite
UK .NTF
U.S. Census TIGER/Line
VRT: Virtual Datasource

OGR ist Teil des GDAL/OGR-Projekts und liegt GDAL bei. Während GDAL für die Verarbeitung von Rasterdaten entwickelt wurde, ist OGR für Vektordaten zuständig.

OGR-Hilfsprogramme und Beispiele

Wie GDAL, besteht auch OGR aus einer Reihe von Bibliotheken, die direkt in Programme eingebunden werden können. Auch OGR verfügt über einige nützliche Hilfsprogramme.

`ogrinfo`

Untersucht Vektordaten und gibt Informationen über die Objekte aus. Diese Informationen können für alle unterstützten Formate ausgegeben werden. Der folgende Code zeigt die Ausgabe von `ogrinfo` für ein Shape-Datei:

```
# ogrinfo -summary placept.shp placept
Had to open data source read-only.
INFO: Open of `placept.shp'
using driver `ESRI Shapefile' successful.

Layer name: placept
Geometry: Point
Feature Count: 497
Extent: (-140.873489, 42.053455) - (-52.808067, 82.431976)
Layer SRS WKT:
(unknown)
AREA: Real (12.3)
PERIMETER: Real (12.3)
PACEL_: Integer (10.0)
PACEL_ID: Integer (10.0)
UNIQUE_KEY: String (5.0)
NAME: String (50.0)
NAME_E: String (50.0)
NAME_F: String (50.0)
```



```

UNIQUE_KEY: String (5.0)
UNIQUE_KEY: String (5.0)
REG_CODE: Integer (2.0)
NTS50: String (7.0)
POP91: Integer (7.0)
SGC_CODE: Integer (7.0)
CAPITAL: Integer (3.0)
POP_RANGE: Integer (3.0)

```

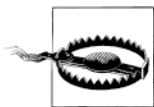
Das Beispiel zeigt die von ogrinfo ausgegebenen Informationen. Diese beinhalten den Darstellungsbereich der Daten, eine Liste der Attribute, den Attributtyp sowie die Anzahl der in der Datei enthaltenen Objekte. Das Programm kann mit weiteren Parametern aufgerufen werden, um detaillierte Informationen über die Daten zu erhalten. Eine ausführliche Beschreibung von ogrinfo finden Sie in Kapitel 6, *Geodaten analysieren*.

ogr2ogr

Konvertiert Geodaten in alle von OGR unterstützten Formate. Dieses Werkzeug kann verwendet werden, um Daten bei der Konvertierung in ein anderes Projektionssystem zu überführen. Weitere Möglichkeiten sind die Filterung, das Löschen von Objekten und die Auswahl gewünschter Attributspalten. Der folgende Code zeigt eine einfache Konvertierung einer Shape-Datei in das GML-Format:

```
# ogr2ogr -f "GML" places.gml placept.shp
```

Die Konvertierung kann in alle Formate erfolgen, die OGR schreiben kann. Nur wenige proprietäre Programme, insbesondere die Feature Manipulation Engine (FME) von SAFE Software, übertreffen die Möglichkeiten, die ogr2ogr bietet.



Beachten Sie, dass beim Aufruf von ogr2ogr der Name der Ziel-/Ausgabe-datei zuerst angegeben werden muss und erst danach der Name der Quelldatei. Diese Reihenfolge kann verwirren, wenn Sie das Programm noch nicht kennen, da bei vielen Kommandozeilenwerkzeugen erst die Quelldatei und danach die Zielfeldatei angegeben wird.

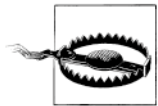
Wie GDAL hilft OGR beim Zugriff, der Konvertierung und der Manipulation von Daten. OGR kann auch mit Skriptsprachen zur Bearbeitung von Daten verwendet werden. Den GDAL/OGR-Paketen liegen bereits einige Python-Module bei. Weitere Bindings liegen für Perl, C#, Ruby, Java, R und Visual Basic 6 vor.



Zwar existiert auch eine PHP-Erweiterung für OGR, diese wird aber nicht offiziell unterstützt oder aktiv weiterentwickelt. Die Erweiterung wurde unabhängig vom GDAL/OGR-Projekt entwickelt und kann unter der Adresse http://dl.maptools.org/dl/php_ogr/ heruntergeladen werden.

PostGIS

PostgreSQL ist eine leistungsstarke relationale Datenbank, die als Open Source-Projekt entwickelt wird. PostgreSQL wird in zahlreichen Projekten erfolgreich eingesetzt. Die Firma Refrations Research (<http://www.refrations.net>) hat PostGIS entwickelt, das die PostgreSQL-Datenbank um die Fähigkeiten, räumliche Daten zu speichern und zu manipulieren, erweitert. Durch die PostGIS-Erweiterung wird PostgreSQL zu einer *räumlichen Datenbank (spatial database)*, in der Geometrien wie andere Daten in Tabellen abgelegt werden können.



PostgreSQL unterstützt bereits nativ geometrische Datentypen wie Punkte, Linien und Flächen. Die unterstützten Eigenschaften und Methoden entsprechen aber nicht den Ansprüchen, die ein GIS stellt. Deshalb wurde für PostgreSQL die Erweiterung PostGIS entwickelt, die die Geometrietypen der räumlichen Funktionen des OGC-Standards *Simple Features for SQL* implementiert.

PostGIS ist mehr als eine räumliche Erweiterung zur Speicherung von Daten. PostGIS bietet die Möglichkeit, räumliche Daten direkt in der Datenbank zu manipulieren. Die Manipulation der Daten erfolgt über einfache SQL-Anweisungen und zeichnet PostgreSQL/PostGIS dadurch gegenüber anderen Datenbanken aus, die lediglich als proprietärer Datenspeicher fungieren und einen Zugriff auch nur über proprietäre Werkzeuge ermöglichen.



Die PostGIS-Funktionen greifen auf zusätzliche Bibliotheken zu. Diese gehören zum GEOS-(Geometry Engine Open Source-)Projekt. GEOS ist eine C++-Bibliothek, die der »OGC specification for Simple Features for SQL« entspricht. Es werden die gesamten Funktionen der JTS (Java Topology Suite) abgebildet. Die GEOS-Bibliothek wurde nicht nur für PostGIS entwickelt, sondern kann auch von anderen Applikationen genutzt werden. Weitere Informationen zu GEOS finden Sie unter <http://geos.refrations.net/>.

Räumliche Analysen mithilfe von SQL

Mit PostGIS können Sie räumliche Daten über SQL-Anweisungen erzeugen und manipulieren. Darüber hinaus kann PostGIS als eigenständiges GIS eingesetzt werden, wobei gleichzeitig die gesamte Funktionsvielfalt von PostgreSQL zur Verfügung steht. Über die PostGIS-Funktionen können Sie beispielsweise Verschneidungen, Berechnungen von Nachbarschaftsbeziehungen, Flächenberechnungen, Umprojektionen und vieles mehr durchführen. Sämtliche Funktionen stehen hier in einer serverseitigen Datenbanklösung zur Verfügung. Beispiel 3-1 verdeutlicht diese Kombination aus SQL- und GIS-Funktionalität. In Beispiel 3-1 werden die Punkte ermittelt, die sich in einem bestimmten Polygon befinden. Eine Einführung in PostGIS finden Sie in Kapitel 14, *PostgreSQL/PostGIS – Einsatz von räumlichen Datenbanken*.

Beispiel 3-1: SQL-Anweisung, über die alle Punktoobjekte ermittelt werden, die sich in einem Polygon einer anderen Tabelle befinden

```
> SELECT town_name
   FROM towns, ontario
   WHERE ST_Contains(ontario_polygon,town_points);
```

```
town_name
-----
Port Hope Simpson
Hopedale
Makkovik
Churchill Falls
North West River
Rigolet
Cartwright
Tignish
Cheticamp
Sheet Harbour
(10 rows)
```

PostGIS wird immer häufiger als Werkzeug zum Speichern und Bearbeiten von Geodaten eingesetzt. Es ist ideal für den gleichzeitigen Zugriff durch mehrere Programme. So können die Daten bearbeitet werden und gleichzeitig über eine Kartenanwendung zur Verfügung stehen.

Andere GIS-Programme sind in der Lage, mit den PostGIS-Daten zu interagieren. Beispielsweise kann OGR PostGIS-Datenquellen lesen und schreiben, und auch MapServer kann auf PostGIS-Daten zugreifen. GeoServer (<http://www.geoserver.org>) kann über WFS-T (OGC Transactional Web Feature Server) lesend und schreibend auf PostGIS-Daten zugreifen. Über eine PostGIS-Funktion können Geometrieobjekte als Scalable Vector Graphics (SVG) ausgegeben werden. Auch einige Desktop-GIS-Programme können PostGIS einbinden und teilweise schreiben, z.B. Quantum GIS (QGIS, siehe <http://www.qgis.org>). Mehr dazu erfahren Sie in Kapitel 14.

Eine PostGIS-Datenbank besitzt viele Funktionen, die Sie auch aus anderen Datenbanken kennen. So werden SQL-Anweisungen für den Zugriff auf die Daten verwendet. Über die SQL-Anweisungen können ebenfalls Geometrien ausgewählt werden. Über PostGIS-Funktionen können die Daten manipuliert und zusammengefasst werden, und es können neue Geometriedaten erzeugt werden.



Der Hauptunterschied zwischen den räumlichen Komponenten von MySQL und PostGIS besteht darin, dass die meisten Funktionen, die in PostGIS vorliegen, in MySQL noch nicht implementiert sind. Außerdem bietet MySQL keine Transaktionsintegrität für Geometrien.

Zusammenfassung der Anwendungen

Tabelle 3-1 fasst die Funktionen der in diesem Kapitel angesprochenen Applikationen zusammen. Dabei wird durch ein oder zwei Häkchen gekennzeichnet, für welche Bereiche die entsprechende Software am besten eingesetzt werden kann.

Tabelle 3-1: Zusammenfassung der Bereiche, in denen die Programme in der Regel eingesetzt werden

	GDAL	OGR	PostGIS	OpenEV	Quantum GIS	MapServer
Visualisierung	✓			✓✓	✓✓	✓✓
Analyse	✓		✓✓	✓	✓	✓
Manipulation	✓	✓	✓✓	✓	✓	
Konvertierung	✓✓	✓✓				
Datenbereitstellung			✓			✓✓

Diese Tabelle zeigt lediglich die häufigsten und wichtigsten Einsatzgebiete der Programme, die in diesem Kapitel vorgestellt wurden. Für manche Aufgaben sind mehrere Programme geeignet, Überschneidungen bieten die Möglichkeit der Wahl aufgrund anderer Parameter (z.B. Programmiersprache oder Lizenz). Letztendlich hängt die Wahl des Werkzeugs immer davon ab, welche Aufgabe Sie bewältigen wollen.

MapServer installieren

MapServer eignet sich nicht nur zum Erzeugen statischer Karten, sondern auch besonders für die Veröffentlichung interaktiver Karten in Websites. Interaktive Karten ermöglichen es den Benutzern, in bestimmte Bereiche zu zoomen und verschiedene Ebenen ein- und ausblenden. Durch einen Klick auf Objekte in der Karte können weitere Informationen aufgerufen werden. Auch wenn es zahlreiche Möglichkeiten gibt, ein Web-Mapping-Projekt zu realisieren, ist das Endprodukt meistens eine interaktive Applikation, die der Benutzer per Mausklick und Tastatur bedienen kann. Hierbei wählt der Benutzer einen Bereich aus, der betrachtet werden soll, oder eine Ebene, die angezeigt werden soll, und die Seite wird mit dem angeforderten Inhalt aktualisiert. Benötigen Sie eine statische Karte, können Sie die Grafik zur späteren Benutzung lokal speichern.

MapServer kann auch von der Kommandozeile aus oder – für Programmierer besonders interessant – über die API angesprochen werden. Außerdem lässt sich MapServer als Common Gateway Interface (CGI) betreiben oder über eine Programmiersprache wie PHP, Perl, Python und Java steuern. Zur Performancesteigerung können Sie MapServer ebenfalls mit FastCGI einsetzen.

Unabhängig davon, auf welche Weise Sie MapServer einsetzen, wird das Programm durch Konfigurationsdateien gesteuert, die während der Laufzeit übergeben werden. Diese Konfigurationsdateien steuern z.B. die angezeigten Ebenen und legen ihre Darstellung (Symbolik, Beschriftung) fest. Geodaten können leicht zu einer bestehenden MapServer-Anwendung hinzugefügt werden. Es müssen lediglich die entsprechenden Verbindungsinformationen und Darstellungsangaben in der Konfigurationsdatei angegeben werden.

Dieses Kapitel stellt die allgemeine Funktionsweise und die Einsatzmöglichkeiten von MapServer vor. Außerdem werden die Programme behandelt, auf die MapServer aufsetzt, um spezielle Funktionalität zur Verfügung zu stellen. Beispiele für die Entwicklung von MapServer-Anwendungen finden Sie in den Kapiteln 10 und 11.

So funktionieren MapServer-Anwendungen

Normalerweise arbeitet MapServer hinter einem Webserver. Erhält der Webserver eine Kartenanfrage, reicht er die Anfrage an MapServer weiter. MapServer erzeugt daraufhin das angeforderte Kartenbild und gibt es an den Webserver zurück. Dieser übergibt die Karte danach an den anfragenden Browser. In Abbildung 4-1 sehen Sie, wie der Benutzer mit dem Webserver interagiert, der seinerseits die Anfragen an das MapServer-Programm weiterleitet.

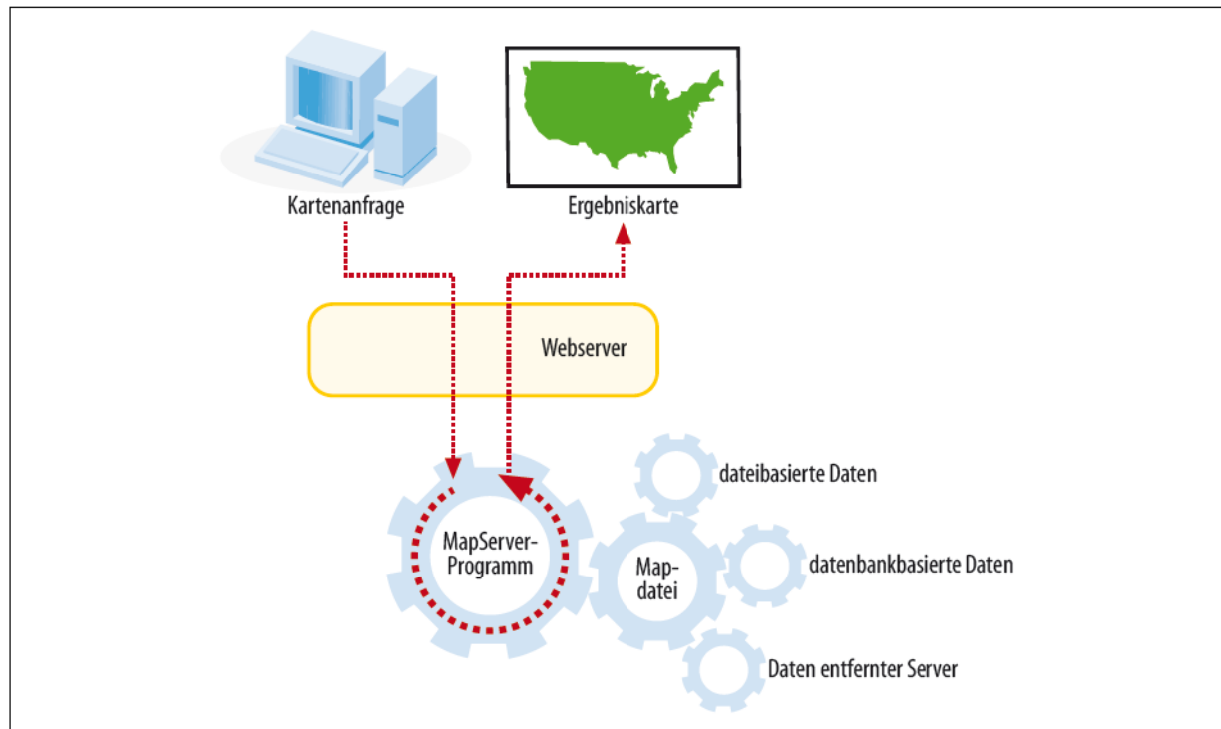


Abbildung 4-1: Grundfunktionen einer MapServer-Anwendung

Die Hauptaufgabe von MapServer besteht darin, Daten aus verschiedenen Datenquellen zu lesen und diese in Form von Ebenen in einer Grafikdatei, dem sogenannten *Kartenbild*, zusammenzuführen. Eine Ebene könnte beispielsweise ein Satellitenbild sein, eine weitere der Umriss Ihres Landes oder Punkte, die Städte markieren. Die Ebenen werden wie Folien übereinandergelegt und schließlich in Form eines Bilds an den Betrachter zurückgegeben. Ein gutes Beispiel für die Verwendung von Ebenen zum Überlagern verschiedener Informationen sehen Sie in Abbildung 4-2. Dort sehen Sie ein Satellitenbild (von einem entfernten Server), Straßenverläufe und die Position verschiedener Städte. Die Beschriftungen der Städte wurden von MapServer dynamisch erzeugt.

Dieser Prozess (auch *Rendering* genannt) wird bei jeder Kartenanfrage durchgeführt, beispielsweise wenn der Benutzer für eine Detailansicht in eine Karte hineinzoomt. Das Rendering findet aber auch statt, wenn der Benutzer manuell ein Neuzeichnen der Karte auslöst, um etwa bei der Änderung des Dateninhalts einer Ebene die Änderung in der Karte zu sehen.

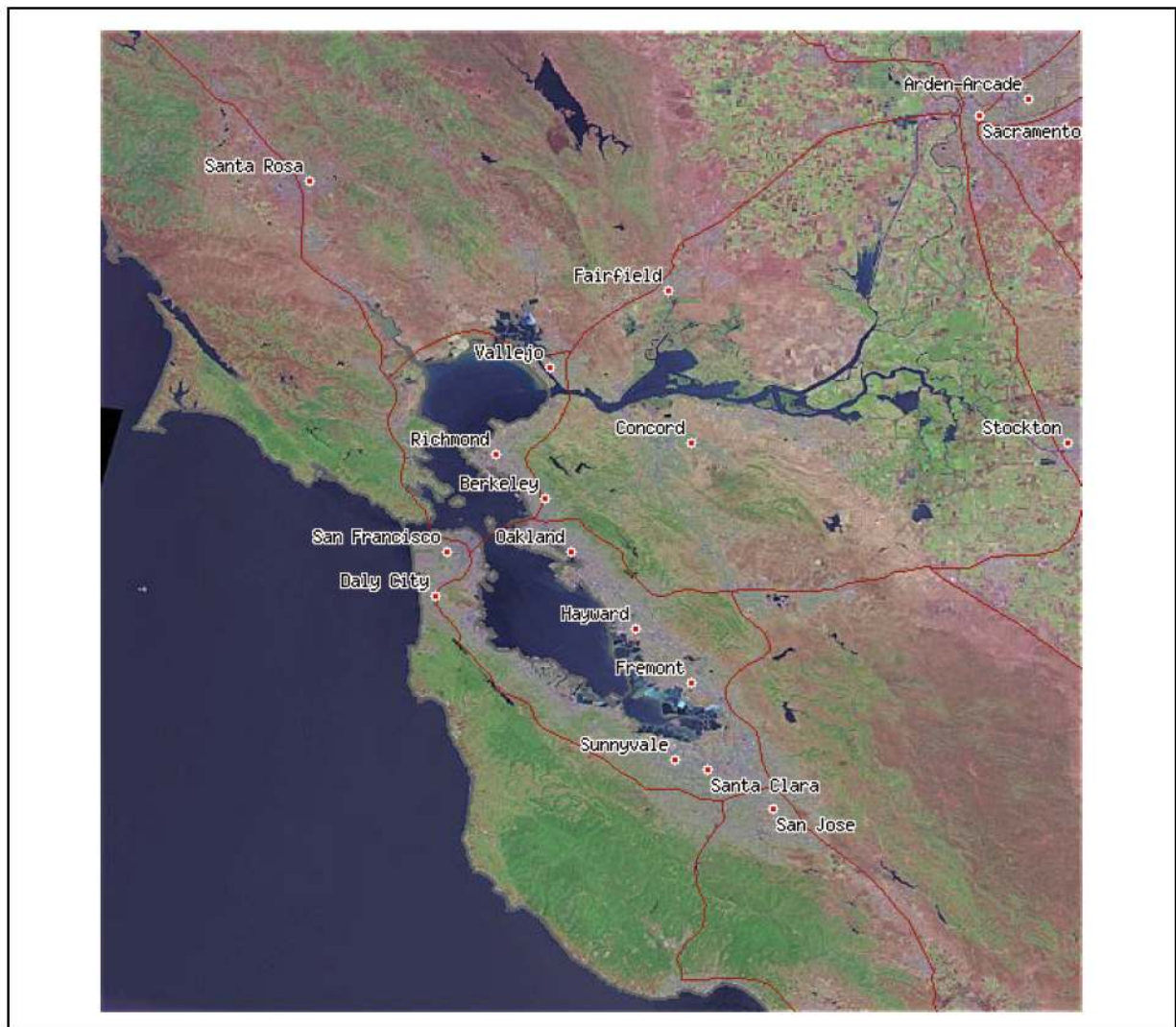


Abbildung 4-2: Eine Karte mit verschiedenen Informationsebenen

Ein Blick auf die wichtigsten Bestandteile von UMN MapServer

MapServer erstellt anhand der eingehenden Benutzeranfragen und den Konfigurationsangaben in der Mapdatei die Bilder. Die wichtigsten Bestandteile einer typischen MapServer-Anwendung sind die MapServer-Executable bzw. das CGI-Programm, eine Mapdatei, Datenquellen und die ausgegebenen Bilder. In Abbildung 4-3 sehen Sie, wie die einzelnen Teile zusammenarbeiten: Nach einer Anfrage greift das MapServer-CGI-Programm auf eine Mapdatei zu, zeichnet die Informationen aus den verschiedenen Datenquellen und gibt ein Bild der Karte zurück.

Das MapServer-Executable

Die einfachste Form von MapServer läuft als ausführbares CGI-Programm auf einem Webserver. Technisch gesehen ist MapServer ein zustandsloser, HTTP-basierter Prozess. Zustandslos bedeutet dabei, dass der Prozess eine Anfrage bearbeitet und dann

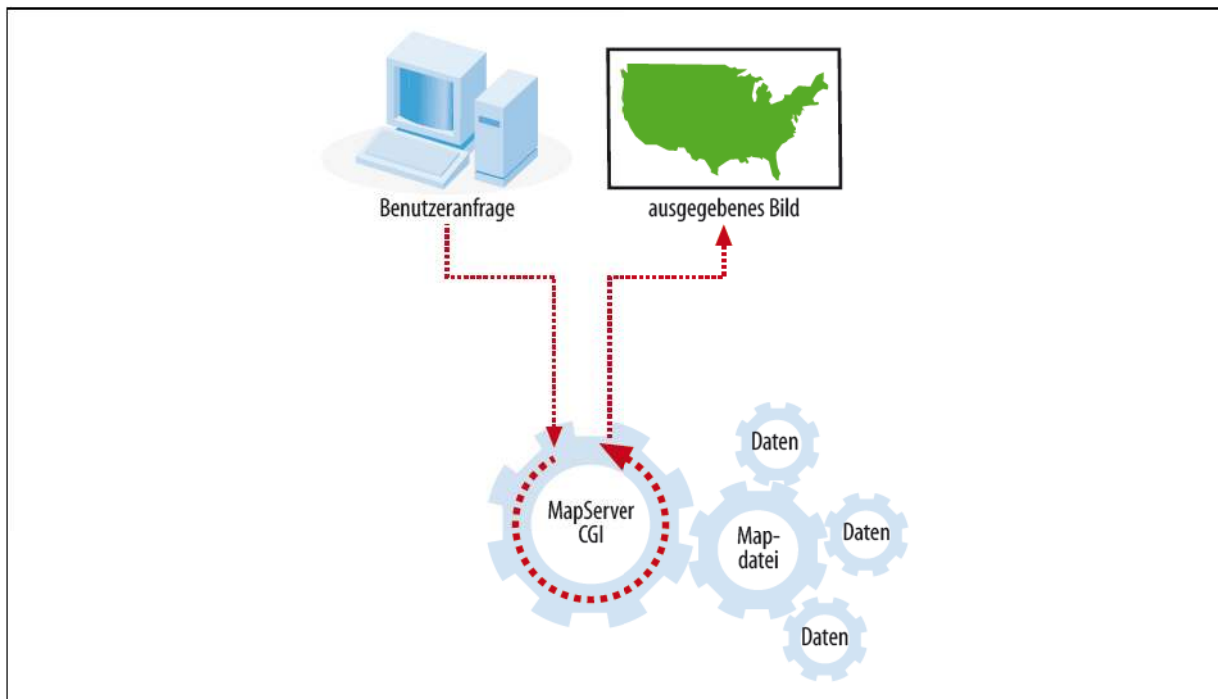


Abbildung 4-3: Die Hauptbestandteile einer MapServer-Applikation

wieder beendet wird. Ein CGI-Programm erhält Anfragen von einem Webserver, bearbeitet sie und gibt eine Antwort oder bestimmte Daten an den Webserver zurück. Wegen seiner Einfachheit ist die Verwendung von MapServer als CGI besonders beliebt. Es ist keine zusätzliche Programmierung notwendig, um MapServer zum Laufen zu bringen. Sie müssen nur eine textbasierte Konfigurationsdatei erzeugen, eine Webseite erstellen und beides für die Bereitstellung durch den Webserver einrichten.



Als Programmierer können Sie auch eigene Programme schreiben, die auf die MapServer-API zugreifen. Dieses Vorgehen gilt allerdings schon als fortgeschrittene Praxis. Eine Übersicht finden Sie in Kapitel 15.

Das MapServer-CGI-Programm funktioniert als Vermittlungsstelle zwischen den Daten und dem Webserver, der die Karte anfordert. Die Anfragen werden in Form von CGI-Parametern vom Webserver an MapServer weitergereicht. Die erzeugten Bilder werden dann an den Webserver zurückgeliefert, der sie schließlich an den Browser des Benutzers überträgt. Weitere Informationen zur mapserv-Executable finden Sie weiter unten in diesem Kapitel.

Das Augenmerk dieses Kapitels liegt darauf, Kartenbilder mit MapServer zu erzeugen. Darüber hinaus kann MapServer aber auch Maßstabsleisten, Legenden und Referenzkarten erstellen. Näheres hierzu finden Sie in den Kapiteln 10 und 11.

Der Webserver

Um MapServer einsetzen zu können, brauchen Sie einen Webserver. Da dessen Rolle nicht unerheblich ist, sollten Entwickler von MapServer-Anwendungen wissen, wie sich Änderungen in der Entwicklung von Webservern auf ihre eigenen Programme auswirken können. In diesem Buch wird die Installation und Verwaltung von Webservern nicht weiter besprochen, allerdings gibt es eine Reihe anderer guter Bücher zu diesem Thema von O'Reilly. Zwei der wichtigsten Themen sind wohl der Betrieb unter hoher Serverlast (viele gleichzeitige Anfragen) und die Sicherheit. Beide Themen sind für den Betrieb von MapServer-Anwendungen in gleicher Weise wichtig. Bücher wie *Web Security, Privacy, and Commerce*, *High Performance Websites* oder *Apache – Das umfassende Handbuch* (alle O'Reilly) können Ihnen dabei helfen.

Die MapServer-Mapdatei

Der MapServer verarbeitet Geodaten zu Karten. Zunächst muss das Programm darüber in Kenntnis gesetzt werden, wo die Geodaten liegen, die verwendet werden sollen. Um zu wissen, wie diese Geodaten dargestellt werden sollen (z.B. ob eine Linie blau oder rot gezeichnet werden soll) benötigt die Software Anweisungen.

Sowohl der Speicherort der Geodaten als auch die Anweisung dazu, wie die Geodaten zu zeichnen sind, werden in der sogenannten Mapdatei hinterlegt. Der Name leitet sich aus der Dateierweiterung **.map* dieses Formats ab. In diesem Buch wird dieser Typ von Datei einfach als »Mapdatei« bezeichnet. Die Mapdatei ist eine textbasierte Konfigurationsdatei, in der die Einstellungen für die Darstellung der Karte und die Interaktion mit ihr festgelegt werden. Sie enthält Informationen darüber, wie viele Ebenen es gibt und welchen Bereich die Daten abdecken, welches Projektionssystem (Koordinatensystem) benutzt wird und welches Bildformat (z.B. PNG, JPEG oder GIF) für die Ausgabe verwendet werden soll. Nur mithilfe dieser Konfigurationsdatei kann der MapServer Kartenbilder erstellen.

Erweiternd kann hier definiert werden, wie die Maßstabsleiste und die Legende dargestellt werden sollen. Eine sehr einfache Version einer Mapdatei sehen Sie in Beispiel 4-1.

Beispiel 4-1: Einfache Mapdatei, die eine Ebene enthält

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  LAYER
    NAME countries
    TYPE POLYGON
    STATUS ON
    DATA countries.shp
    CLASS
```

Beispiel 4-1: Einfache Mapdatei, die eine Ebene enthält (Fortsetzung)

```
STYLE
  OUTLINECOLOR 100 100 100
END
END
END
END
```

In Anfragen an MapServer muss angegeben werden, welche Mapdatei verwendet werden soll. Danach erzeugt MapServer die Karte basierend auf den Einstellungen in der Datei. Dies macht die Mapdatei zu einem wesentlichen Bestandteil. Eine detaillierte Betrachtung der Mapdatei finden Sie in Kapitel 10.

Datenquellen

MapServer ist in der Lage, auf zahlreiche verschiedene Datenquellen zuzugreifen. Die meisten gängigen Formate werden von dem Standard-MapServer bereits unterstützt. Mithilfe von Erweiterungen kann die Zahl um Dutzende von Vektor- und Rasterformaten erhöht werden (z.B. mithilfe der GDAL- und OGR-Bibliotheken). Als Datenquelle kommen dateibasierte Geodaten genauso in Frage wie datenbankbasierte Daten, und sogar einfache Textdateien, in denen die Daten durch Kommata getrennt vorliegen (CSV-Dateien – comma-separated values), werden unterstützt.

MapServer ist außerdem in der Lage, durch die Unterstützung einiger OGC-Standards Daten über das Netz bereitzustellen und auf entfernte Daten zuzugreifen. So können Ebenen von anderen OGC-fähigen Servern im Netz angefordert werden. Weitere Informationen zu MapServer-Datenquellen finden Sie in Kapitel 5. Näheres zum OGC und den OGC-Standards sowie standardisierten Webservices erhalten Sie in den Kapiteln 12 und 13.

Kartenausgabe

Während die Mapdatei das zentrale Objekt jeder MapServer-Anwendung darstellt, ist der Benutzer eher an dem erzeugten Kartenbild interessiert. Nachdem alle Ebenen verarbeitet und in eine Bilddatei geschrieben wurden, wird der Browser des Benutzers angewiesen, das erzeugte Bild in der Webseite anzuzeigen. Oftmals haben Anfänger bei MapServer Probleme damit, dass kein Ausgabebild zurückkommt. In Kapitel 11 wird genauer auf dieses Problem und die Einrichtung von MapServer eingegangen.

Neben Karten können auch noch andere Grafiken erstellt werden. Maßstabsleisten, grafische Legenden und Referenzkarten sind ebenfalls möglich. Die Arbeit mit diesen Bildern erfolgt auf ähnliche Weise wie bei den Karten selbst. Beispiele hierfür finden Sie in den Kapiteln 10 und 11.

Installation von MapServer

MapServer läuft auf einer Reihe von Betriebssystemen. Die Details der Installation können sich daher in Abhängigkeit vom verwendeten Server und der Integration ins restliche System deutlich unterscheiden.

Plattformen

MapServer läuft erfolgreich auf vielen verschiedenen Plattformen und Prozessortypen. Eine Studie von MapServer-Entwicklern zeigt Dutzende verschiedener Implementierungen mit sehr unterschiedlichen Prozessorgeschwindigkeiten und -typen. Folgende Systeme wurden getestet:

- Red Hat/SuSE 9 und 10.x/Debian 3.1 und 4.0/Mandrake Linux 6 bis 9/Ubuntu
- Solaris
- Mac OS X
- Vmware mit Windows und Linux
- FreeBSD
- SCO Open Server
- SGI IRIX
- Windows 2000, XP, 95

Die angegebenen Leistungen lagen bei einem Minimum von 120 MHz Taktfrequenz und 64 MByte Arbeitsspeicher. Andere verwendeten die maximal verfügbaren Ressourcen an Prozessorleistung und Arbeitsspeicher. Dementsprechend existieren in den Mailinglisten Berichte vom derzeit oberen Ende des Leistungsspektrums, wonach die Kombination GDAL/MapServer auch auf gängigen aktuellen 64-Bit-Architekturen (Intel, Itanium, AMD) erfolgreich betrieben wird. In aller Regel ist dabei von Linux-Systemen die Rede.

Viele Entwickler sind der Meinung, dass nur MapServer in der Lage ist, diese hohen Ansprüche an Vielseitigkeit und Flexibilität bei der Bereitstellung webbasierter Karten zu erfüllen.

Download der MapServer-Binärdateien

Viele Dokumente und How-tos beschreiben die Kompilierung von MapServer. Man könnte daher glauben, dies sei prinzipiell notwendig. Für die meisten Anwender ist eine Kompilierung jedoch nicht nötig. Beim Download von *Binaries* werden bereits kompilierte ausführbare Dateien nebst den notwendigen Bibliotheken heruntergeladen und können sofort auf dem Betriebssystem der Wahl ohne Kompilierung eingesetzt werden.

MapServer für Windows

Wollen Sie MapServer unter Windows einsetzen, reicht es in der Regel, das Standard-Windows-Paket herunterzuladen. Es enthält meistens alle notwendigen MapServer-Programme in einer gemeinsamen ZIP-Datei. Die Pakete sind auf der MapServer-Website (<http://mapserver.gis.umn.edu/win32binaries.html>) und der MapTools.org-Site (<http://dl.maptools.org/dl/> bzw. <http://maptools.org/ms4w/>) erhältlich. Durch die Diskussionsforen und Mailinglisten werden auch über die MapServer-Nutzer kompilierte Versionen bereitgestellt.

Die MapTools-Website bietet das MS4W-Paket an. Das Paket heißt MapServer For Windows (kurz: MS4W) und ist unter <http://maptools.org/ms4w/> zu finden. Dieses Paket ist für den Einstieg sehr zu empfehlen und setzt keinen installierten Webserver auf Ihrem System voraus.

Das MS4W-Paket ist eine Entwicklung der DM Solutions Group (Ottawa, Kanada). Diese Firma bietet rund um MapServer Unterstützung in den Bereichen Entwicklung, Implementierung und Support. Über das MS4W-Paket soll dem Benutzer ein möglichst leichter Einstieg in die Arbeit mit MapServer ermöglicht werden.

Das MS4W-Paket ist eine ZIP-Datei, die alles enthält, was Sie für die Arbeit mit MapServer brauchen – sogar der Apache-Webserver liegt bereits vorkonfiguriert bei. Wenn Sie sich bisher an die Arbeit mit Webservern nicht so recht herangetraut haben, ist dieses Paket genau richtig für Sie! Um MS4W zu installieren, brauchen Sie nur die Paketdatei auf einem Laufwerk Ihrer Wahl im Wurzelverzeichnis (z.B. C:\) zu entzippen. Automatisch wird eine Reihe an Unterverzeichnissen im Verzeichnis C:\MS4W angelegt, unter anderem auch das Verzeichnis *Apache*, in dem der Webserver zu finden ist.



Sie können das Paket auch an eine andere Stelle in Ihrem Dateisystem verschieben. Sie müssen dann allerdings einige Einstellungen in der Webserver-Konfiguration anpassen.

Nach dem Entpacken müssen Sie die ausführbare Datei *apache-install.bat* aufrufen. Dies kann über einen Doppelklick auf die Datei erfolgen. Über diese Datei wird Apache installiert, als Dienst eingerichtet und gestartet. Dieser Schritt entfällt, falls Sie einen bereits installierten Webserver (z.B. Apache) verwenden möchten. In diesem Fall können Sie alle Dateien aus dem Verzeichnis C:\MS4W\Apache\cgi-bin in das cgi-bin-Verzeichnis Ihres Webserverns legen. Nun können Sie in Ihren Webbrowser die Adresse <http://localhost> eingeben. Wird die MS4W-Startseite angezeigt, zeigt dies, dass der Apache-Webserver läuft. Über diese wenigen Schritte ist alles eingerichtet, und Sie können mit dem Aufsetzen eines MapServer-Projekts beginnen. Das MS4W-Paket enthält alle notwendigen Bibliotheken, Programme und Konfigurationsdateien. Eine Zusammenfassung der in MS4W enthaltenen Dateien sehen Sie in Tabelle 4-1. Die Verwendung dieser Dateien zum Erstellen von Applikationen wird genauer in Kapitel 10 besprochen.

Tabelle 4-1: Zusammenfassung der im Paket MapServer für Windows (MS4W) enthaltenen Komponenten

Komponente	Beschreibung
Apache	Webserver
PHP	Skriptsprache
<i>mapserv.exe</i>	MapServer-CGI-Programm
<i>php_mapscript.dll</i>	PHP/MapScript-Bibliotheken; siehe Kapitel 15 für weitere Details zu MapScript
<i>proj.exe/cs2cs.exe</i> usw.	PROJ.4-Projektions-Hilfsprogramme und -Bibliotheken
<i>shp2img.exe/shptree.exe/scalebar.exe/legend.exe</i> usw.	MapServer-Kommandozeilen-Hilfsprogramme
<i>gdalinfo.exe/gdaladdo.exe/gdaltindex.exe/ogr2ogr.exe/ogrinfo.exe/ogrtindex.exe</i> usw.	GDAL/OGR-Kommandozeilen-Hilfsprogramme

Es können zusätzliche Pakete zum MS4W-Paket hinzugefügt werden (u.a. ka-Map, Chameleon, OpenLayers, Mapbender). Diese erhalten Sie unter <http://www.maptools.org/ms4w/index.phtml?page=downloads.html>.

Linux-Binärdateien

Unter <http://maptools.org/fgs/> finden Sie ein Projekt mit dem Namen FGS Linux Installer. Hierbei handelt es sich um eine selbst extrahierbare Datei, die MapServer mit PHP/MapScript und allen Abhängigkeiten auf Ihrem Server installiert. Es wird eine unabhängige (stand-alone) Umgebung mit der vollständigen Software (Apache, PHP usw.) erzeugt. Das Paket basiert auf der Free Open Source Software (FOSS) GIS Suite (FGS), hat aber einen anderen Fokus bekommen.

GIS-Distributionen für Linux

Es gibt mindestens vier Linux-Distributionen, die MapServer und verwandete Applikationen beinhalten. Der leichteste Weg zu einer lauffähigen MapServer-Installation führt über die Paketverwaltung der Betriebssysteme. Dabei muss man allerdings in Kauf nehmen, dass die installierten Komponenten nicht dem neuesten Stand entsprechen.

Die Installation vorgefertigter Pakete erfolgt bei Debian/Ubuntu über die Paketverwaltungen APT (Advanced Packaging Tool), bei SuSE über RPM (Red Hat Package Manager). Debians APT bedienen Sie entweder über eine der verfügbaren grafischen Benutzerschnittstellen (häufig Adept unter KDE und Synaptic unter Gnome) oder direkt über die APT-Tools der Kommandozeile. Unter SuSE erfolgt die Installation grafisch unterstützt per YaST oder auf der Kommandozeile mit dem Befehl `rpm`.

Die Installation von Paketen muss in aller Regel mit root-Rechten erfolgen. Das wird bei den grafischen Tools automatisch per Passwortabfrage geregelt. Auf der Kommandozeile muss man dagegen als root eingeloggt sein oder andernfalls den `sudo`-Befehl dem eigentlichen Aufruf voranstellen.

Live-CD WebGIS.rlp

Die diesem Buch beigelegte Live-CD WebGIS.rlp ist eine Zusammenstellung von Software zur Erzeugung, zur Verwaltung und zum Publizieren von Geodaten. Es handelt sich um ein Projekt im Rahmen des Aufbaus der Geodateninfrastruktur Rheinland-Pfalz. Federführend bei dieser landesweiten Aufgabe ist der Interministerielle Ausschuss für Geoinformation Rheinland-Pfalz (IMAGI-RP).

WebGIS.rlp ist vor allem dazu gedacht, Institutionen den Einstieg in die Bereitstellung und Nutzung standardkonformer Geo-Webservices zu erleichtern. Die WebGIS.rlp-CD beinhaltet Quantum GIS (inkl. Georeferenzierungstool), GRASS, Mapbender, UMN MapServer, Mapstorer, GeoServer und PostgreSQL mit PostGIS sowie PgAdmin3.

Um den Einstieg so leicht wie möglich zu gestalten, sind neben der o.g. Software auch einige Testdaten auf der CD vorhanden. Es handelt sich dabei um ein Orthofoto, Shape-Dateien mit ausgewählten Objekten des digitalen Landschaftsmodells, Flurkarten, Bebauungspläne sowie ein DEM für ein 100km großes Gebiet des Landes Rheinland-Pfalz (Deutschland). Als Betriebssystem wurde Knoppix in der Version 5.1.1 verwendet. Hier gibt es die Möglichkeit, die Live-CD als Debian-System auf dem Rechner zu installieren. Für ein intensives Testen (größere Datenmengen) empfiehlt es sich, den Inhalt der CD mit einem mitgelieferten Programm auf einen 1-GByte-USB-Stick zu übertragen und anschließend vom Stick zu booten.

Download der Image-Datei unter:

<http://www.gdi-rp-dienste3.rlp.de/webgis/>

Eine ausführliche Beschreibung des WebGIS.rlp-Projekts finden Sie unter:

<http://www.geoportal.rlp.de/mediawiki/index.php/WebGIS.rlp>

Debian/Ubuntu:

```
apt-get install <PaketNAME 1> [... <PaketNAME n>]
```

Anhand des Paketnamens wird bei der Verwendung von APT das Paket aus einer Internetquelle heruntergeladen und installiert. Abhängigkeiten von anderen Paketen werden hierbei automatisch aufgelöst. Das ist die übliche Vorgehensweise. Auf die wichtigsten Paketnamen wird im weiteren Verlauf hingewiesen. Für die alternative Installation einer heruntergeladenen *deb*-Datei verwenden Sie den *dpkg*-Befehl:

```
dpkg -i <PaketDATEI 1> [... <PaketDATEI n>]
```

SuSE:

```
rpm -i <PaketDATEI>
```

Weitere Optionen entnehmen Sie bitte der jeweiligen Dokumentation.

DebianGIS

Diese Debian-basierte Distribution enthält verschiedene GIS-Applikationen. Siehe <http://wiki.debian.org/DebianGis/PackageList>.

HostGIS

Zu dieser Distribution gehören MapServer, PostGIS, PHP, Python, Perl und viele andere Standard-GNU/Linux-Applikationen. Dieses Paket basiert auf Slackware Linux. Siehe <http://www.hostgis.com/linux/>.

Live-CD WebGIS.rlp

Die diesem Buch beigelegte Live-CD WebGIS.rlp ist eine Zusammenstellung aus Software zur Erzeugung, zur Verwaltung und zum Publizieren von Geodaten. Als Betriebssystem wurde Knoppix verwendet. Download der Image-Datei: <http://www.gdi-rp-dienste3.rlp.de/mediawiki/images/testdaten/>.

Eine ausführliche Beschreibung des WebGIS.rlp-Projekts finden Sie unter <http://www.geoportal.rlp.de/mediawiki/index.php/WebGIS.rlp>.

ominiverdi Live-CDs

Ominiverdi bietet Distributionen unter <http://livecd.ominiverdi.org/> an. Hier finden Sie Ominiverdi-Desktop-Versionen mit einer Sammlung von Desktop-GIS, GDAL/OGR und PostgreSQL/PostGIS sowie anderen Programmen. In Kürze soll eine Ominiverdi-Serverversion erscheinen, die MapServer beinhalten wird.

GIS-Knoppix

Dies ist ebenfalls eine bootfähige CD-Distribution. Sie enthält die meisten wichtigen Pakete, wie MapServer, PostGIS, Terraview, JUMP, QGIS, Thuban, Maplab usw. Siehe <http://www.sourcepole.com/sources/software/gis-knoppix/>.

RPM-Versionen für Linux

Sie können auch Red Hat Package Manager-(RPM-)Dateien für Linux herunterladen. Es kann sein, dass Sie Hilfe bei der Auflösung der Abhängigkeiten benötigen. Selbst wenn Sie alle Pakete von der gleichen Site wählen, kann es sein, dass nicht alles dabei ist, was Sie brauchen, auch wenn einige Versionen die meisten benötigten Dinge schon enthalten. Suchen Sie einfach im Netz nach RPM-Dateien.

Mapping Hacks GIS RPMs for Fedora Core

Die Firma Locative Technologies (<http://locative.us>) hält speziell für Fedora Core zusammengestellte Pakete bereit (leider nur für ältere MapServer-Versionen), zu finden unter <http://mappinghacks.com/rpm/>.

Debian und Ubuntu Linux-Versionen

Für Debian und Ubuntu Linux gibt es eine Reihe von Paketen, die MapServer enthalten. Allerdings gibt es bei den Leistungsmerkmalen und Abhängigkeiten gewisse Unterschiede. Eine Übersicht über die verfügbaren Debian-Pakete erhalten Sie über

http://packages.debian.org/cgi-bin/search_packages.pl?keywords=mapserv&searchon=names&subword=1&version=all&release=all. Eine Installationsanleitung und weitere Informationen bietet der Link <http://mapserver.gis.umn.edu/docs/howto/debian-install>.

Informationen zu den Ubuntu-Paketquellen finden Sie unter <http://packages.ubuntu.com/>. Auch zu Ubuntu finden Sie eine ausführliche Beschreibung auf der MapServer-Website unter <http://mapserver.gis.umn.edu/docs/howto/Ubuntu%20Compilation>.

Mac OS X

Ein aktuelles MapServer-Installationspaket gibt es unter <http://www.kyngchaos.com/software/unixport/mapserver>. Eine detaillierte Beschreibung der Installation finden Sie hier: <http://www.kyngchaos.com/macosex/mapservinstall>.

Kompilieren des MapServer-Source-Codes

Jedes Projekt, das den Anspruch hat, Open Source-Software zu sein, muss den Quellcode zur Verfügung stellen. Wenn Sie den Quellcode von MapServer vorliegen haben, können Sie die Quellen nach Bedarf anpassen. Es kann aber auch sein, dass Sie keine Änderungen im Code durchführen möchten, MapServer jedoch selbst kompilieren wollen, um bestimmte Bibliotheken einzubinden oder auszuschließen. In den Binärversionen von MapServer wird normalerweise eine Reihe von Datenformaten unterstützt. Oftmals stehen mehr Formate zur Verfügung, als Sie in Ihrer Applikation tatsächlich benötigen. Wenn Sie MapServer selbst kompilieren, können Sie die nicht benötigten Formate einfach ausschließen. Wird andererseits ein bestimmtes Format nicht standardmäßig unterstützt, können Sie es über den Kompilierungsvorgang aktivieren.

Die Kompilierbeispiele in diesem Kapitel verwenden Linux und Open Source-Kompilierwerkzeuge des GNU-Projekts (siehe <http://gnu.org>). Die Werkzeuge und Befehle werden von der Kommandozeile oder einer Shell-Konsole ausgeführt. Die Kompilierung für andere Systeme kann sich je nach den verfügbaren Werkzeugen unterscheiden. Dieser Abschnitt geht davon aus, dass Sie sich mit dem Kompilieren von Software auskennen und Ihnen die notwendigen Werkzeuge inklusive eines C-Compilers wie gcc und Konfigurations-/Build-Werkzeuge, wie autoconf und automake zur Verfügung stehen.

Quellcode herunterladen

Die aktuellen Quellcodepakete finden Sie auf der MapServer-Downloadseite unter <http://mapserver.gis.umn.edu/download>. Um die brandaktuellen Neuerungen vom MapServer zu testen, können Sie auch einen der Nightly-Builds herunterladen. Hierbei handelt es sich um eine Kopie des aktuellen Code-Repository.



Wenn Sie absolut auf dem neuesten Stand sein möchten, steht der Quellcode auch über ein Versionskontrollsystem (Subversion SVN) zur Verfügung. Zugangsinformationen zum SVN finden Sie unter <http://mapserver.gis.umn.edu/development/cvs>. Das SVN protokolliert die Änderungen im Quellcode und wird für die Verwaltung und Entwicklung vieler Softwareprojekte eingesetzt. Nähere Informationen zu SVN finden Sie unter <http://subversion.tigris.org/>.

Sind die Quellen heruntergeladen, müssen Sie sie entpacken. Dies geht in einem Schritt mit dem GNU-Befehl `tar`:

```
# tar -xvzf mapserver-5.0.0.tar.gz
```

Dieser Befehl entzippt die Datei in ein neues Verzeichnis und erzeugt eine Unterverzeichnisstruktur, die den Quellcode enthält. Nach dem Entpacken können Sie in das neue Verzeichnis wechseln und mit der Konfiguration Ihres Builds beginnen.

Konfiguration der Quellen

Vor dem Kompilieren des MapServer-Programms müssen Sie eine Reihe von Optionen konfigurieren. Dieser Schritt ist notwendig, um den Quellcode für die Kompilierung vorzubereiten. Die Programmdatei `configure` finden Sie in der obersten Verzeichnisebene der Quellen, zu der Sie nach dem Entpacken gewechselt sind.

In diesem Verzeichnis befindet sich auch eine Datei `README.CONFIGURE` mit einer ausführlichen Beschreibung. Außerdem finden Sie Erläuterungen im MapServer-Wiki unter http://mapserver.gis.umn.edu/docs/howto/compiling_on_unix/. Wenn Sie `configure` ohne weitere Optionen ausführen, werden Standardwerte als Einstellungen verwendet. Das ist vielleicht auch schon ausreichend, um MapServer für Ihre Zwecke zu nutzen. Auf dem Bildschirm wird bei der Ausführung eine Reihe von Statusmeldungen und Abhängigkeitstests ausgegeben. Beachten Sie, dass der Aufruf des Skripts mit vorangestellter Pfadangabe erfolgen muss. Da wir uns im selben Verzeichnis wie `configure` befinden, ist das ein Punkt gefolgt von einem Slash, wie in Beispiel 4-2 zu sehen ist. Das Beispiel enthält außerdem einen Auszug aus den Ausgaben des über `configure` gestarteten Konfigurationsprozesses.

Beispiel 4-2: Ausführung von `configure` und die resultierenden Ausgaben des Konfigurationsprozesses

```
# ./configure
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
....
MapServer Version from mapserver.h: '5.0.0'
configure: checking where FreeType 2.x is installed...
checking for FT_Init_FreeType in -lfreetype... yes
        using libfreetype -lfreetype from system libs.
configure: checking where Zlib is installed...
```

Beispiel 4-2: Ausführung von `configure` und die resultierenden Ausgaben des Konfigurationsprozesses (Fortsetzung)

```
checking for zlibVersion in -lz... yes
    using libz from system libs (-DUSE_ZLIB).
configure: checking where PNG is installed...
checking for png_init_io in -lpng... yes
    using libpng from system libs.
....
configure: checking for GD 2.0.16 or higher...
....
checking if AGG support requested... no
configure: checking whether we should include EPPL7 support...
    including EPPL7 support.
configure: checking whether we should include PROJ.4 support...
    PROJ.4 support not requested.
configure: checking whether we should include thread safe support...
    thread safe support disabled.
configure: checking whether we should include ESRI SDE support...
    ESRI SDE support not requested.
checking if GEOS support requested... no
configure: checking whether we should include OGR support...
    OGR support not requested.
checking if GDAL support requested... no
checking checking ms JPEG output... yes, GDAL is not available but libjpeg is.
configure: checking whether we should include TIFF support...
checking for TIFFOpen in -ltiff... no
    libtiff not found... TIFF support not included.
checking if PostGIS support requested... no
checking if MyGIS support requested... no
checking if OracleSpatial support requested... no
checking if MING/Flash support requested... no
configure: checking whether we should include WMS Server support...
    OGC WMS Compatibility not enabled (PROJ.4 is required for WMS).
configure: checking whether we should include WFS Server support...
    OGC WFS Server support not requested.
configure: checking whether we should include WCS Server support...
    OGC WCS Server support not requested.
configure: checking whether we should include WMS Client Connections support...
    OGC WMS Client Connections not enabled (PROJ.4 and libcurl required).
configure: checking whether we should include WFS Client Connections support...
    OGC WFS Client Connections not enabled (PROJ.4, libcurl and OGR required).
configure: checking whether we should include OGC SOS Server support...
    OGC SOS Support not enabled (libxml2 and PROJ4 and OGR required).
....
configure: creating ./config.status
config.status: creating Makefile
....
```

In den in Beispiel 4-2 gezeigten Ausgaben von `configure` gibt es ein paar Dinge zu beachten. Zuerst wird überprüft, ob alle für den Kompiliervorgang benötigten Werkzeuge auch vorhanden sind. Fehlt beispielsweise der C-Compiler, bricht der Prozess mit einem Warnhinweis ab, wie in Beispiel 4-3 gezeigt.

Beispiel 4-3: Fehlt der C-Compiler, bricht der Konfigurationsprozess ab

```
checking for gcc... (cached) gcc
checking whether the C compiler (gcc ) works... no
configure: error: installation or configuration problem:
        C compiler cannot create executables.
```

Das ist gut zu wissen. Schließlich ist es sinnlos weiterzumachen, wenn klar ist, dass Sie später nicht weiterkommen.

configure überprüft Ihr System auf die verschiedenen Grafikbibliotheken wie PNG und GD und führt eine Reihe von Tests mit ihnen durch. Stehen die Bibliotheken zur Verfügung, legt configure fest, dass sie beim Kompilieren verwendet werden.

Hier sehen Sie die typische Fehlermeldung, wenn bestimmte Bibliotheken nicht gefunden werden konnten:

```
PNG (libpng) library cannot be found, possibly needed for GD
```

Diese Information ist besonders dann wichtig, wenn Sie anstelle von GIF lieber PNG als Ausgabeformat für Ihre Karten verwenden möchten. Aus diesem Beispiel geht nicht hervor, dass configure auch auf GIF- und JPEG-Dateien – ähnlich wie für PNG – prüft.

Im nächsten Schritt überprüft configure, ob die Unterstützung für verschiedene Datenformate und Hilfsprogramme einkompiliert werden soll. Hierbei hilft es natürlich, die verschiedenen Formate zu kennen. Kann configure diese nicht finden, wird Ihr fertiges MapServer-Programm diese auch nicht unterstützen. Einige scheinen vielleicht nicht so wichtig, es sei denn, Sie wissen, worum es sich handelt. Das folgende Beispiel zeigt einen solchen Hinweis:

```
PROJ.4 support not requested.
```

Dies ist keine gute Nachricht, wenn die von Ihnen verwendeten Daten in verschiedenen Projektionen vorliegen. MapServer kann Daten aus verschiedenen Kartenprojektionen einlesen und diese für die fertige Karte in das gewünschte System umprojizieren. Es ist daher sehr wahrscheinlich, dass Sie die Unterstützung für PROJ.4 irgendwann brauchen werden. Hierfür müssen die PROJ.4-Dateien allerdings auf Ihrem System vorliegen. Wie Sie optionale Bibliotheken wie PROJ.4 einbinden, wird später in diesem Kapitel angesprochen.

Die Meldungen zu anderen Formaten sind etwas deutlicher, wie Sie in Beispiel 4-4 sehen können.

Beispiel 4-4: Ausgabe der Überprüfung der Unterstützung verschiedener Datenbankformate durch configure

```
configure: checking whether we should include ESRI SDE support...
ESRI SDE support not requested.
checking if PostGIS support requested... no
checking if MyGIS support requested... no
checking if OracleSpatial support requested... no
```


Im Beispiel wird überprüft, ob die Unterstützung für bestimmte Datenbankformate angefordert wurde. Standardmäßig werden diese Optionen nur eingebunden, wenn die Optionen ausdrücklich ausgewählt wurden.

Weitere wichtige Optionen, die nicht standardmäßig aktiviert sind, sind GDAL, OGR, AGG (Anti-Grain Geometry) sowie die OGC-Spezifikationen. Der durchschnittliche Benutzer wird sich über die Unterstützung von GDAL und OGR freuen. AGG steht ab der MapServer-Version 5.0 als neues Rendering-Backend zur Verfügung. Wenn Sie AGG zum Generieren der Grafiken einsetzen, können Sie eine höhere Ausgabequalität ohne merkliche Performanceverluste erreichen. Die Unterstützung der OGC WMS/WFS-Client- und -Serveroptionen wird benötigt, wenn Sie auf standardisierte Dienste aufsetzen bzw. externe standardisierte Dienste ansprechen möchten. Diese Technologie verbreitet sich mehr und mehr. Bei der Kompilierung müssen Sie beachten, dass eine Reihe von Abhängigkeiten bestehen (z.B. PROJ.4, OGR, CURL).

GDAL

In den allermeisten Fällen empfiehlt es sich, die in Kapitel 10 eingehend beschriebene GDAL/OGR-Bibliothek für die Nutzung durch MapServer zu verwenden. Sie ermöglicht dem MapServer den Zugriff auf eine Vielzahl unterschiedlicher Datenformate. Sollten Sie den MapServer aus einer aktuellen Linux-Distribution (SuSE, Debian etc.) oder als Windows-Paket (MS4W) installieren, geschieht dies im Normalfall automatisch.

Kompilieren Sie den MapServer selbst, können Sie unterschiedlich vorgehen: Entweder Sie verwenden ein vorhandenes Binärpaket aus einer Distribution, oder Sie kompilieren auch GDAL/OGR selbst. Im ersten Fall müssen Sie darauf achten, neben der eigentlichen Programmbibliothek auch das entsprechende Entwicklerpaket mitzuinstallieren (also z.B. bei Debian *libgdal1-1.3.2* und *libgdal1-1.3.2-dev* oder bei SuSE *gdal-1.3.2-5.17.i586.rpm* und *gdal-devel-1.3.2-5.17.i586.rpm*). Sollten Sie besondere Anforderungen an die bereitgestellten Datenformate haben (z.B. PostGIS-Unterstützung), vergewissern Sie sich in der entsprechenden Dokumentation, dass das Paket entsprechend konfiguriert wurde.

Wenn Sie die Konfiguration der unterstützten Formate selber in der Hand behalten wollen, kompilieren Sie GDAL aus den Quellen (erhältlich und beschrieben unter <http://www.gdal.org>). Im Prinzip funktioniert das Erstellen von GDAL ähnlich wie in diesem Kapitel für den MapServer beschrieben (bis auf die Installationsroutine). Lesen Sie deshalb im Zweifelsfall zunächst die folgenden Abschnitte über das Kompilieren des MapServers durch.

Wurde `configure` erfolgreich ausgeführt, sehen Sie eine Meldung wie die in Beispiel 4-5.

Beispiel 4-5: Der Konfigurationsprozess wurde erfolgreich abgeschlossen

```
updating cache ./config.cache
creating ./config.status
creating Makefile
```


Am Ende der Konfiguration wird ein Makefile erzeugt. Diese Datei enthält alle Informationen, die der Befehl `make` benötigt, um den Quellcode zu kompilieren (und die notwendigen Bibliotheken einzubinden usw.). Wenn Sie das Programm selbst kompilieren, haben Sie sicherlich auch keine Hemmungen, `configure` zu verwenden. Dem durchschnittlichen Benutzer erscheinen die Meldungen vermutlich ein wenig kryptisch. Wenn Schwierigkeiten auftreten, mit denen `configure` selbst nicht umgehen kann, bricht der Prozess normalerweise mit einer entsprechenden Fehlermeldung ab. Daher ist es wichtig, die Ausgaben zu verstehen, um das Problem zu lösen.

In Beispiel 4-3 kann `configure` den C-Compiler `gcc` (die GNU Compiler Collection, wird typischerweise unter Linux zum Kompilieren von Quellcode verwendet) nicht finden und bricht sofort mit einer Meldung auf der Konsole ab. Selbst wenn Sie die Meldung nicht verstehen, ist dies noch kein Grund aufzugeben. Oftmals reichen bereits die letzten Zeilen der `configure`-Ausgaben, um Hilfe aus den Mailinglisten zu erhalten.

configure – mögliche Optionen

Nachdem wir ein paar Beispiele für die Entschlüsselung der Ausgaben von `configure` gesehen haben, ist es an der Zeit, die möglichen Optionen für den `configure`-Befehl kennenzulernen. Optionale Parameter werden durch zwei Bindestriche eingeleitet, auf die der Name und der Wert der Option folgen, wie beispielsweise `--with-sde`. Um festzustellen, welche Optionen zur Verfügung stehen, können Sie `configure` mit der Option `--help` aufrufen, wie hier gezeigt:

```
# ./configure --help | more
```

Der Hilfetext für `configure` besteht aus fünf Abschnitten:

- Konfiguration (Configuration)
- Installationsverzeichnisse (Installation Directories)
- Optionale Funktionen (Optional Features)
- Optionale Pakete (Optional Packages)
- Einige wichtige Umgebungsvariablen (Some Influential Environment Variables)

Den normalen Anwender dürften hauptsächlich die optionalen Funktionen und Pakete interessieren, die wir in Beispiel 4-6 für Sie aufgelistet haben. Am Anfang einer Zeile sehen Sie jeweils die verwendete Syntax gefolgt von den für MapServer spezifischen Optionen.

Beispiel 4-6: Liste der Konfigurationsoptionen für die optionalen Features und optionalen Pakete

Optional Features:

```
--enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
--enable-ignore-missing-data  Ignore missing data file errors at runtime
--enable-point-z-m          Use point Z and M parameters (disabled by default).
--enable-debug              Include "-g" in CFLAGS for debugging.
--enable-gcov               Enable source code coverage testing using gcov
--enable-perlv-ld-detect    (applies to --with-php only)
```

Beispiel 4-6: Liste der Konfigurationsoptionen für die optionalen Features und optionalen Pakete (Fortsetzung)

	Use perl -V output to figure the command to use to link php_mapscript.so. Try this only if the default internal macro didn't work.
--enable-runpath	Include runtime library path flags (-Wl,-R) in link. Required on *BSD systems.

Optional Packages:

--with-PACKAGE[=ARG]	use PACKAGE [ARG=yes]
--with-freetype=DIR	GD: Specify where FreeType 2.x is installed (DIR is path to freetype-config program or install dir).
--with-zlib=DIR	GD,SVG: Specify where zlib is installed (DIR is path to zlib install dir).
--with-png=DIR	GD: Specify where PNG is installed (DIR is path to PNG install dir).
--with-jpeg[=DIR]	Include JPEG support (DIR is LibJPEG's install dir).
--with-xpm=DIR	GD: Specify where libXpm is installed (DIR is the libXpm install prefix).
--with-libiconv=DIR	Specify where libiconv is installed (DIR is path to libiconv install dir (prefix)).
--with-gd[=[static,]DIR]	Specify which version of GD to use (DIR is GD's install dir).
--with-pdf[=DIR]	Include PDF support (DIR is PDFlib's install dir).
--with-agg[=DIR]	Include AGG support (DIR is AGG's install dir).
--with-eppl	Include EPPL7 support.
--with-proj[=DIR]	Include PROJ.4 support (DIR is PROJ.4's install dir).
--with-threads[=linkopt]	Include thread safe support
--with-sde[=DIR]	Include ESRI SDE support (DIR is SDE's install dir).
--with-sde-version[=VERSION NUMBER]	Set ESRI SDE version number (Default is 91).
--with-geos[=PATH]	Include GEOS support (PATH is path to geos-config)
--with-ogr[=PATH]	Include OGR support (PATH is path to gdal-config).
--with-gdal[=PATH]	Include GDAL support (PATH is path to gdal-config)
--without-tiff	Disable TIFF support.
--with-tiff[=DIR]	Include TIFF support (DIR is LibTIFF's install dir).
--with-postgis[=ARG]	Include PostGIS Support (ARG=yes/path to pg_config)
--with-mygis[=ARG]	Include MyGIS Support (ARG=yes/path to my_config)
--with-oraclespatial[=ARG]	Include OracleSpatial Support (ARG=yes/path to Oracle home)
--with-ming[=DIR]	Include MING/Flash Support (DIR=path to Ming directory)
--with-wfs	Enable OGC WFS Server Support (OGR+PROJ4 required).
--with-wcs	Enable OGC WCS Server Support (GDAL+PROJ4 required).
--with-wmsclient	Enable OGC WMS Client Connections (PROJ4 and libcurl required).
--with-wfsclient	Enable OGC WFS Client Connections (PROJ4, libcurl and OGR required).
--with-sos	Enable OGC SOS Server Support (libxml2 and PROJ4 and OGR required).
--with-curl-config=PATH	Specify path to curl-config.
--with-xml2-config=PATH	Specify path to xml2-config.
--with-fastcgi=path	Enable FastCGI, point to installed tree.

Beispiel 4-6: Liste der Konfigurationsoptionen für die optionalen Features und optionalen Pakete (Fortsetzung)

```
--with-httpd          Specify path to 'httpd' executable.
--with-warnings[=flags] Enable strict warnings (or user defined warnings)
--with-php=DIR         Specify directory where PHP4's include files are
                        installed (or a pointer to the full source tree)
                        Required in order to compile the PHP/MapScript
                        module.
--with-apxs[=FILE]     (CURRENTLY DISABLED)
                        Use this option only if building the PHP MapScript
                        on a system where PHP was built as a shared Apache
                        module. FILE is the optional pathname to the
                        Apache apxs tool; defaults to apxs.
--with-apache=DIR      (CURRENTLY DISABLED)
                        Use this option only if building the PHP MapScript
                        on a system where PHP was built as an Apache
                        module statically linked into the httpd executable.
                        DIR is the path to the apache include files.
--with-java-include-os-name=dirname
                        (AUTODETECTED)
                        Use this option only if building of Java Mapscript
                        fails because os-dependent headers are not found.
                        dirname is the name of subdirectory of the Java
                        installation where os-dependent include files are
                        found (for instance linux or solaris).
                        Only used by Java mapscript.
```

Wie Sie sehen, ist die Zahl der Optionen ziemlich hoch. Lassen Sie sich davon aber nicht verwirren. Die für Sie wichtigen werden Sie schnell erkennen – die restlichen können Sie vermutlich ignorieren und deaktivieren. Jede Option bezieht sich auf ein bestimmtes Daten- oder Ausgabeformat. Wenn Sie wissen, welche Ausgabeformate Sie benötigen, sollten Sie dazu eine Angabe in der Liste finden. Ist Ihnen das Ausgabeformat der Bilder nicht so wichtig, reichen die MapServer-Standardereinstellungen vermutlich schon aus.

Die meisten Optionen lassen sich über die Parameter `--enable` bzw. `--disable` oder äquivalent dazu `--with` oder `--without` ein- und ausschalten. Um die obige Liste einfach zu halten, wurden die Optionen zum Deaktivieren (`--disable`/`--without`) nicht aufgeführt.

Die meisten Parameter sind nach einer schlichten Einführung leicht zu verstehen. Es werden hier nur die absolut notwendigen Anforderungen, die für die Konfiguration und Kompilierung von MapServer notwendig sind, angesprochen. Wollen Sie mehr über Techniken zum Kompilieren von Quellcode erfahren, gibt es eine Reihe guter Fachbücher zu diesem Thema, zum Beispiel die folgenden:

- *GNU Autoconf, Automake and Libtool* (New Riders); online unter <http://sources.redhat.com/autobook/>
- *Programming with GNU Tools* (O'Reilly)
- *GNU make* (O'Reilly Verlag)

- *GNU Make: A Program for Directing Compilation* (GNU Press); online unter: <http://www.gnupress.org/book8.html>

Die Parameter können auf zwei verschiedene Arten angegeben werden:

- durch die Angabe von `--enable-x` bzw. `--with-x`
- durch das Hinzufügen eines Werts am Ende `--enable-x=y` oder `--with-x=y`

Die erste Form ist recht einfach: Wollen Sie ein bestimmtes Merkmal aktivieren, fügen Sie einfach den entsprechenden Parameter hinzu. Hierbei steht `x` für den Namen der Option, wie in der Hilfeausgabe gezeigt.

Das folgende Beispiel zeigt, wie `configure` angewiesen werden kann, die für die Unterstützung des JPEG-Bildformats notwendigen Bibliotheken oder Programme einzubinden. Können die notwendigen Dateien nicht gefunden werden, gibt `configure` eine entsprechende Fehlermeldung aus:

```
./configure --with-jpeg
```

Können die Bibliotheken (in unserem Beispiel die für die JPEG-Unterstützung) nicht automatisch gefunden werden, hilft es meistens, die zweite Methode für die Angabe eines zusätzlichen Parameters zu verwenden. In der Hilfemeldung (siehe Beispiel 4-6) ist zu lesen:

```
--with-jpeg[=DIR]
```

Hierbei steht `[=DIR]` für einen zusätzlichen Parameter. Steht der Parameter in eckigen Klammern, muss er nicht prinzipiell angegeben werden. Offenbar braucht `configure` ein wenig Hilfe beim Auffinden der JPEG-Dateien. In diesem Fall wird der Installationspfad benötigt. Wenn Sie sich nicht sicher sind, welche Angaben `configure` erwartet, sehen Sie sich einfach den Hilfetext noch einmal an:

```
Include JPEG support (DIR is LibJPEG's install dir)
```

Wenn Sie nun wissen, wo sich die `LibJPEG`-Dateien befinden, können Sie `configure` explizit anweisen, dort nachzusehen. Das Auffinden von Abhängigkeiten wird im nächsten Abschnitt behandelt.

In manchen Fällen braucht `configure` die *include*- und *lib*-Dateien, die zur betreffenden Bibliothek gehören, wie beispielsweise bei der Konfiguration für `LibJPEG`. Sie müssen hier das Wurzelverzeichnis angeben, in dem die Bibliotheken installiert wurden. Dieses enthält die Verzeichnisse *include* und *lib* und auch die `LibJPEG`-Dateien. Alternativ kann dies das Verzeichnis sein, in dem Sie `LibJPEG` vor der Installation kompiliert haben. Ich kompiliere alle meine Quellen zum Beispiel im Verzeichnis `/home/tyler/src/`. Wenn ich für `LibJPEG` also den Ordner `/home/tyler/src/jpeg` benutze, kann ich `configure` auch diesen Pfad übergeben.

In meinem Fall wurde `LibJPEG` unter `/usr/lib` installiert. Also führe ich `configure` wie im folgenden Beispiel aus:

```
./configure --with-jpeg=/usr
```


Da dieses Verzeichnis sehr häufig zum Speichern dieser Bibliotheken verwendet wird, hätte `configure` dort vermutlich sowieso nachgesehen. Wurden die Bibliotheken dagegen an einem ungewöhnlichen Ort installiert (etwa in einem eigenen Verzeichnis zum Testen von MapServer), kann das Verständnis dieser Funktion wichtig sein.

Nicht alle Optionen benötigen die Angabe eines Verzeichnispfads. Einige brauchen als Wert einen Pfad (`[=PATH]`), ein Verzeichnis (`[=DIR]`) oder ein Argument (`[=ARG]`). Ein Beispiel finden Sie bei der Angabe der Option für die Unterstützung von GDAL (Beispiel 4-6):

```
--with-gdal[=PATH] Include GDAL support (PATH is path to gdal-config)
```

Das Lesen der Beschreibung auf der rechten Seite ist sehr wichtig, um den Parameter richtig anzugeben. Für GDAL wird nicht das Installationsverzeichnis angegeben, sondern der vollständige Pfad zur Datei `gdal-config`. Da `configure` von sich aus an ein paar Standardspeicherorten nachsieht, ist dieser Parameter optional. Sobald Sie die Datei aber an einem ungewöhnlichen Ort gespeichert haben, müssen Sie auch hier den kompletten Pfad angeben, wie im folgenden Beispiel. Sind Sie sich nun darüber klar geworden, welche Optionen benutzt werden sollen, fügen Sie diese zu einem langen Befehl zusammen und lassen `configure` die Arbeit erledigen.

```
./configure --with-jpeg --with-png --with-ogr --with-gdal=/usr/local/bin/gdal-config  
--with-postgis
```

Wie erwähnt, müssen die Argumente `DIR`, `PATH` und `ARG` nur in bestimmten Fällen angegeben werden. Wenn Sie `configure` nach der Auswahl der Optionen ausführen, sollten Sie die Ausgabemeldungen im Auge behalten und hierbei besonders auf die Zeilen achten, in denen es um die von Ihnen ausgewählten Optionen geht. Schlägt – wie im folgenden Beispiel – die Überprüfung fehl, haben Sie offenbar irgendwo einen Fehler gemacht.

```
checking whether we should include OGR support...  
./configure: line 1: /usr/local/bin/gdal-config: No such file or directory
```

In diesem Fall wurde zwar ein Speicherort für `gdal-config` angegeben, die Datei selbst konnte aber nicht gefunden werden, weil GDAL gar nicht installiert ist (die Installation von GDAL ist ein separater Prozess, der in Kapitel 6 beschrieben wird). Manchmal sucht `configure` auch an nicht durch `PATH` angegebenen Orten und findet die Datei. Erhalten Sie Fehlermeldungen, können Sie den `configure`-Befehl einfach ändern und erneut aufrufen, bis alles funktioniert.

Abhängigkeiten

Eine der vermutlich größten Stärken von MapServer besteht darin, dass MapServer auf bereits existierenden Bibliotheken aufbauen kann. Die Entwickler haben sich darauf konzentriert, einen kleinen zentralen Kern an Anforderungen zu unterstützen. Statt weitere Abhängigkeiten in das zentrale Projekt einzubauen, werden zusätzliche Leistungsmerkmale durch die Einbindung externer Bibliotheken hinzugefügt. Das Ergebnis ist ein mächtiges Produkt, das leicht um neue Funktionen erweitert werden kann, ohne hierfür größere Änderungen am Kern vornehmen zu müssen.

Wenn Ihnen die schlanke Basisversion ausreicht, brauchen Sie sich nur wenige Gedanken um externe Abhängigkeiten zu machen. Auf vielen Systemen sind die benötigten Bibliotheken oder externen Programme bereits installiert.

Mögliche Abhängigkeiten beinhalten:

- Bibliotheken für Bildformate: JPEG, PNG, XPM, LIBICONV
- Bibliotheken zur grafischen Formatierung/Zusatzbibliotheken: GD, AGG, FreeType, PDF, MING
- Bibliotheken für Datenformate und die Unterstützung externer Programme: TIFF, EPPL, SDE, MPATROL, OGR, GDAL, POSTGIS, MYGIS, ORACLESPIATIAL
- Projektionsbibliothek: PROJ.4
- Optionen für OGC-Webservices: WMS, WFS, WCS, SOS, CURL, HTTPD, PHP

Auch wenn Sie nicht wissen, welche externen Bibliotheken Sie benötigen, werden die Programme `configure` und `make` Sie informieren, falls etwas fehlt.

Durch Ihr Wissen darüber, wofür die wichtigsten Bibliotheken verwendet werden, verstehen Sie auch die Meldungen (von `configure` und `make`) besser. Außerdem hilft es vor auszuplanen und die benötigten Bibliotheken, die Ihnen noch fehlen könnten, bereits zur Hand zu haben.

Für die einfachste Konfiguration von MapServer wird nur die Bilderstellungsbibliothek GD benötigt (von <http://www.boutell.com/gd/>). Die bereitgestellten Funktionen werden gebraucht, um die Bilder der Karten zu erzeugen.



Als ich MapServer zum ersten Mal in Aktion sah, war ich davon überwältigt, wie schnell aus ziemlich komplexen Vektor- und Rasterdaten ein Bild erstellt wurde. Der Schlüssel hierfür ist die GD-Bibliothek. GD ist eine Umgebung zur Bildverarbeitung und -erstellung, ohne die MapServer nicht funktioniert.

GD ist so wichtig für MapServer, dass der `configure`-Prozess abbricht, sollten Sie versuchen, die Bibliothek zu deaktivieren, wie in Beispiel 4-7 gezeigt.

Beispiel 4-7: Versuch, die GD-Unterstützung bei der Konfiguration zu deaktivieren

```
# ./configure --without-gd
...
checking for GD 2.0.16 or higher...
configure: error: GD library cannot be disabled
```

Mit Ausnahme von GD gibt es keine grundsätzlichen Abhängigkeiten für MapServer. Allerdings braucht GD selbst die Unterstützung weiterer Bildverarbeitungsbibliotheken, um brauchbare Bilder für MapServer zu erstellen. Damit GD richtig funktioniert, wird mindestens eine PNG- oder JPEG-Bibliothek benötigt. Vermutlich ist beides bereits auf Ihrem System installiert.

MapServer kann die Bilder in einer Vielzahl verschiedener Formate ausgeben. Am bekanntesten sind hierbei wohl JPEG, GIF und PNG. Wollen Sie eines oder mehrere dieser Formate einsetzen, müssen die jeweiligen Bibliotheken GD zur Verfügung stehen.

Auf den meisten modernen Linux-Plattformen ist GD in der Version 2 installiert. Der Name lautet GD2. MapServer benötigt ab der Version 4.4 die GD2-Bibliotheken und die entsprechenden Entwicklungsdateien für GD2. Diese Entwicklungsdateien zu einem Programm oder einer Bibliothek enthalten Informationen, die es anderen Programmen ermöglichen, auf seine Funktionalität zuzugreifen. Es handelt sich meist um sogenannte Header-Dateien (mit der Endung **.h*). Bei Linux-Distributionen sind sie häufig als gesonderte Pakete zu den eigentlichen Programmdateien erhältlich. So ist die übliche Konvention unter Debian/Ubuntu (deb-Pakete) `<paketname>-dev` und unter SuSE (RPM-Pakete) `<paketname>-devel`.

Wenn Sie das RPM-System für die Installation von Software nutzen, brauchen Sie das Paket `gd2-devel`. Mit folgendem Befehl können Sie feststellen, ob bereits ein Paket installiert ist:

```
rpm -qa | grep <Name des Pakets>
```

Unter Debian/Ubuntu erreichen Sie dasselbe mittels:

```
dpkg -l | grep <Name des Pakets>
```

Zur Prüfung, ob `gd` installiert ist, können Sie schreiben:

```
rpm -qa | grep gd
```

Daraufhin werden die installierten Pakete angezeigt. Keine Antwort bedeutet, dass keine RPM-Pakete installiert wurden. Bei älteren Linux-Versionen und anderen Betriebssystemen ist GD2 vermutlich noch nicht vorhanden. Trotzdem brauchen Sie die Entwicklungsbibliotheken für die von Ihnen eingesetzte GD-Version.



Für die Unterstützung des PNG-Formats werden auch die ZLIB-Bibliotheken benötigt. Für die Installation von GD werden Sie sehr wahrscheinlich darüber hinaus die PNG-/ZLIB-, JPEG- und FreeType-Bibliotheken brauchen.

Die einfachsten MapServer-Konfigurationen verwenden Bibliotheken für PNG, JPEG, WBMP (Windows Bitmap) und FreeType (Unterstützung für TrueType-Zeichensätze), die bereits auf Ihrem System vorliegen, und bauen diese automatisch ein.

Selbst wenn Sie bei der Konfiguration explizit alle Optionen mittels `--without` zu deaktivieren versuchen, wird `configure` (trotz Ihrer Anweisung) ein paar Bibliotheken einbinden. In dieser einfachsten Form kann MapServer nur zwei Dateiformate lesen: JPEG-Bilder und ESRI Shape-Dateien. Diese minimalistische Konfiguration kann Bilder in den Formaten PNG, JPEG und WBMP ausgeben und optional TrueType-Fonts (TTF) für die Beschriftung der Bilder verwenden.

Diese Minimalversion ist zwar äußerst schlicht, reicht aber aus, um die Grundfunktionen von MapServer besser kennenlernen zu können. Viele Anwender führen `./configure` auch einfach ohne besondere Optionen aus, um zu sehen, ob es irgendwelche offensichtlichen Probleme gibt.

Kompilieren mit make

Ist die Konfiguration abgeschlossen, können Sie die Applikation kompilieren. Je nachdem, welche Optionen Sie gewählt haben, kann dieser Prozess auch etwas länger dauern. Die Kompilierung beginnt mit dem Aufruf des Befehls `make`, wie in Beispiel 4-8 zu sehen.

Nach dem Aufruf von `make` werden Sie ein Vielzahl von Befehlen und Informationen auf dem Bildschirm sehen. Ist die Kompilierung abgeschlossen, gibt es keine weitere Meldung. Es erscheint einfach wieder die Eingabeaufforderung. Beispiel 4-8 zeigt den Beginn und das Ende des `make`-Prozesses.

Beispiel 4-8: Auszug aus den Ausgaben des Befehls `make` (Beginn und Ende der Kompilierung)

```
# make
touch mapscriptvars
pwd > mapscriptvars
echo -DHAVE_VSNPRINTF -DNEED_STRLCAT -DUSE_EPPL -DUSE_JPEG -DUSE_GD_GIF -DUSE_GD_PNG
      -DUSE_GD_JPEG -DUSE_GD_WBMP -DUSE_GD_FT -DGD_HAS_FTEX_XSHOW -DGD_HAS_GDIMAGEGIFPTR
      -DGD_HAS_GETBITMAPFONTS >> mapscriptvars
echo -I. -I/usr/include >> mapscriptvars
echo -L. -lmapserver -lgd -L/usr/lib -lgd -ljpeg -lfreetype -lpng -lz -lXpm -lX11
      -ljpeg -lfreetype -lpng -lz -lXpm -lX11 -ljpeg
      -lc -lz -lm -lstdc++ >> mapscriptvars
echo >> mapscriptvars
grep '#define MS_VERSION' mapserver.h >> mapscriptvars
rm -f mapserver-config
echo '#!/bin/sh' > mapserver-config
echo 'CONFIG_LIBS="-lgd -L/usr/lib -lgd -ljpeg -lfreetype -lpng -lz -lXpm -lX11
      -ljpeg -lfreetype -lpng -lz -lXpm -lX11 -ljpeg
      -lc -lz -lm -lstdc++"' >> mapserver-config
echo 'CONFIG_DEP_LIBS=""' >> mapserver-config
echo 'CONFIG_CFLAGS="-O2 -fPIC -Wall -DHAVE_VSNPRINTF -DNEED_STRLCAT -DUSE_EPPL
      -DUSE_JPEG -DUSE_GD_GIF -DUSE_GD_PNG -DUSE_GD_JPEG -DUSE_GD_WBMP
      -DUSE_GD_FT -DGD_HAS_FTEX_XSHOW -DGD_HAS_GDIMAGEGIFPTR
      -DGD_HAS_GETBITMAPFONTS -DUSE_ICONV -DUSE_ZLIB
      -I/usr/include "' >> mapserver-config
```

Es kann sein, dass die Ausgabe einige Warnungen enthält. Solange der Prozess weiterläuft, können Sie die Meldungen jedoch ignorieren.

Wenn Sie bereits Erfahrungen mit der Kompilierung haben, werden Sie versucht sein, als Nächstes `make install` aufzurufen. Das ist zwar in Ordnung, aber nicht notwendig. Für MapServer gibt es keine `install`-Regeln. Stattdessen verschieben Sie die erzeugte `mapserv`-Datei einfach in das `cgi-bin`-Verzeichnis Ihres Webservers. Rufen Sie `make`

install trotzdem auf, erhalten Sie weitere Anweisungen zu den nächsten Arbeitsschritten, wie Beispiel 4-9 zeigt:

Beispiel 4-9: Ausgabe nachdem Aufruf von make install

```
# make install
***** MapServer Installation *****
To install MapServer, copy the 'mapserv' file to your web server's cgi-bin
directory.
If you use MapScript then see the documentation for your specific MapScript
version for installation instructions.
```

Durch die Ausführung des Befehls `make` erhalten Sie die ausführbare `mapserv`-Datei, auf die sich die obige Meldung von Beispiel 4-8 bezieht. Haben Sie nach dem `make`-Prozess keine Datei mit dem Namen `mapserv` vorliegen, wurde der Befehl offenbar nicht richtig abgeschlossen. Die Ausgabe von `make install` geht auch auf MapScript ein, eine erweiterte Scripting-Option für MapServer, die in Kapitel 15 näher besprochen wird.

Zum Testen des `mapserv`-CGI-Programms rufen Sie es mit der Option `-v` auf (siehe Beispiel 4-10). Hierdurch werden die Versionsnummer von MapServer und die einkompilierten Optionen angezeigt.

Beispiel 4-10: Ausgabe der Versionsnummer und der unterstützten Formate des MapServer-CGI-Programms

```
# ./mapserv -v
MapServer version 5.0.0 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP
OUTPUT=SVG SUPPORTS=FREETYPE INPUT=EPPL7 INPUT=JPEG INPUT=SHAPEFILE
```

Dieses Beispiel zeigt, welche Optionen für Ein- und Ausgabeformate während der Konfiguration gewählt wurden. Rufen Sie `mapserv` ohne die Option `-v` auf, erhalten Sie eine Warnung, die Sie daran erinnert, dass es nicht als Standalone-Programm ausgeführt werden kann (Beispiel 4-11).

Beispiel 4-11: Ausgabe beim Ausführen von mapserv auf der Kommandozeile ohne weitere Optionen

```
# ./mapserv
This script can only be used to decode form results and
should be initiated as a CGI process via a httpd server.
```

Rufen Sie `mapserv` ohne weitere Parameter im Browser auf, erhalten Sie eine andere Nachricht:

```
http://localhost/cgi-bin/mapserv
No query information to decode. QUERY_STRING is set, but empty.
```

Über diese Schritte können Sie prüfen, ob `mapserv` korrekt installiert wurde. Mehr Details hierzu finden Sie in Kapitel 11.

Zusammen mit `mapserv` werden noch eine Reihe weiterer Programme erzeugt. Hierzu gehören `scalebar`, `shp2img`, `shptree`, `tile4ms` und andere. Weitere Informationen dazu finden Sie in Kapitel 10.



Auf der MapServer-Website finden Sie unter <http://mapserver.gis.umn.edu/docs/howto> eine Reihe von Anleitungen zur Kompilierung von MapServer.

Installation

Eine weitere Stärke von MapServer besteht darin, dass mehrere Programmversionen mit verschiedenen Fähigkeiten gleichzeitig auf einem Computer laufen können. So kann eine stabile Version neben einer aktuellen Testversion betrieben werden, ohne dass diese sich gegenseitig stören. Diese Flexibilität ist möglich, weil `mapserv` als CGI-Programm über einen Webserver läuft. Jede MapServer-basierte Webapplikation kann eine eigene `mapserv`-Version aufrufen. Dieses Vorgehen wird in Kapitel 11 genauer behandelt.

Typischerweise wird MapServer installiert, indem Sie die Datei `mapserv` in ein Verzeichnis legen, aus dem Ihr Webserver die Datei ausführen kann (standardmäßig `cgi-bin`). Mehr ist nicht notwendig, um die CGI-Version von MapServer zu installieren. Spezielle Konfigurationen haben möglicherweise weitere Abhängigkeiten. Dennoch besteht der wichtigste Installationsschritt darin, die Datei `mapserv` in das `cgi-bin` (oder ein anderes Verzeichnis mit den notwendigen Eigenschaften) zu kopieren.

Wenn Sie MapServer über RPM-Pakete installieren, nehmen Ihnen diese die Arbeit ab. Allerdings müssen Sie sicherstellen, dass die `mapserv`-Datei auch am richtigen Ort installiert wird. Standardmäßig versucht RPM, die Datei unter `/var/www/cgi-bin` abzulegen, auch wenn Sie sie an anderer Stelle brauchen. Für den Apache-Webserver unter SuSE Linux wäre das korrekte Verzeichnis beispielsweise `/srv/www/cgi-bin`. Entsprechendes gilt für Debian-basierte Distributionen. Hier befindet sich das `cgi-bin`-Verzeichnis üblicherweise unter `/usr/lib/cgi-bin`.

Planen Sie, MapServer per RPM zu installieren, ist es sinnvoll, sich eine Liste der von RPM angelegten Dateien und deren Speicherorten ausgeben zu lassen. Hierfür hält RPM eigene Abfrage- und Listenfunktionen bereit, wie in Beispiel 4-12 gezeigt.

Beispiel 4-12: Auflistung des Inhalts des MapServer-RPM-Pakets

```
# rpm -qpl mapserver-4.0.1-1.i386.rpm
/usr/bin/shp2img
/usr/bin/shptree
/usr/bin/sortshp
/usr/bin/tile4ms
/var/www/cgi-bin/mapserv
```

Nach Abschluss der Installation kann `mapserv` CGI-Anfragen bearbeiten und Karten ausgeben.

Testen der Installation

Sie können `mapserv` bzw. `mapserv.exe` auf einfache Weise testen, indem Sie das Programm direkt aufrufen. Öffnen Sie hierfür eine Eingabeaufforderung und wechseln Sie in den Ordner, der das `mapserv`-Programm enthält. Der Name kann `mapserv`, `mapserv.exe` oder auch `mapserv_xx` lauten, wobei `xx` für die Versionsnummer von MapServer steht. 50 steht beispielsweise für MapServer in der Version 5.0. Bei der Verwendung von MS4W finden Sie die `mapserv.exe` im Verzeichnis `c:\ms4w\apache\cgi-bin`.

Wenn Sie MapServer selbst kompilieren, befindet sich `mapserv` in dem Quellcodeverzeichnis, in das das Programm kompiliert wurde. Bei einer Installation per RPM befindet sich `mapserv` im Standardverzeichnis des Apache-Webservers. Unter SuSE liegt die Datei in `/srv/www/cgi-bin`, bei Fedora dagegen in `/var/www/cgi-bin`. Bei Debian finden Sie die Datei unter `/usr/lib/cgi-bin/`.

Um das `mapserv`-Programm von der Kommandozeile auszuführen, benutzen Sie unter SuSE Linux beispielsweise den Befehl:

```
/srv/www/cgi-bin/mapserv
```

Befinden Sie sich schon in dem entsprechenden Verzeichnis, erfolgt der Aufruf über:

```
./mapserv
```

Mit MS4W verwenden Sie:

```
c:\ms4w\apache\cgi-bin\mapserv.exe
```

In beiden Fällen sollte die Ausgabe lauten:

```
This script can only be used to decode form results and  
should be initiated as a CGI process via a httpd server.
```

Das ist eine Fehlermeldung, die aber bestätigt, dass Ihre Version des MapServer-Programms korrekt funktioniert. Herzlichen Glückwunsch – jetzt können Sie Ihre Arbeit mit MapServer beginnen. Näheres hierzu finden Sie in den Kapiteln 10 und 11.

MapServer-Support

Es gibt verschiedene Möglichkeiten, während der Einrichtung und Installation von MapServer Hilfe zu bekommen. Hierzu gehören ein kommerzieller Support, Mailinglisten und ein Internet Relay Chat (IRC).

MapServer-Website

- Die Haupt-Website mit der Dokumentation finden Sie unter <http://mapserver.gis.umn.edu>.
- Ausführliche How-tos finden sich unter <http://mapserver.gis.umn.edu/docs/howto>.
- Eine FAQ-Liste gibt es unter <http://mapserver.gis.umn.edu/doc/mapserver-FAQ.html>.

- Eine deutschsprachige MapServer-Dokumentation finden Sie unter http://umn.mapserver.ch/index_de.php sowie unter <http://www.umn-mapserver.de/>.
- Bugs (Programmierfehler) werden im Trac gemeldet und protokolliert. Den Link finden Sie unter <http://mapserver.gis.umn.edu/development/bugs>.

MapServer-Mailinglisten

- Eine Auflistung der verschiedenen Mailinglisten finden Sie unter <http://mapserver.gis.umn.edu/community/maillinglists/>.
- Die englischsprachige MapServer-User-Liste finden Sie unter <http://mapserver.gis.umn.edu/community/maillinglists/mapserver-users/>.
- Das Archiv gibt es hier: <http://lists.umn.edu/archives/mapserver-users.html>.
- Ein durchsuchbares GMANE-Archiv finden Sie unter <http://dir.gmane.org/gmane.comp.gis.mapserver.user>.
- Eine Mailingliste für MapServer-Entwickler gibt es auch: <http://mapserver.gis.umn.edu/community/maillinglists/mapserver-dev>.
- Die deutschsprachige Mailingliste finden Sie unter <https://freegis.org/mailman/listinfo/mapserver-de>.

MapServer-IRC-Kanal

- Suchen Sie nach `#mapserver` auf dem Server irc.freenode.net.

Kommerzieller Support

- Auf der OSGeo-Website können Sie über http://www.osgeo.org/search_profile den für Ihre Fragestellung passenden Dienstleister suchen.

Weitere Referenzen

- Referenzliste mit MapServer-Lösungen finden Sie unter <http://mapserver.gis.umn.edu/gallery>.
- Eine Liste verfügbarer Bücher, die auf MapServer eingehen, erhalten Sie hier: <http://mapserver.gis.umn.edu/docs/faq/books>.
- Verfolgen Sie die Diskussionen der OSGeo Foundation über die Mailinglisten. Weitere Informationen finden Sie unter http://www.osgeo.org/content/faq/mailling_lists.html.



Nutzen Sie die Möglichkeiten, die Ihnen die OSGeo Foundation bietet. Sie erhalten über die OSGeo eine Dienstleisterübersicht unter http://www.osgeo.org/search_profile mit weltweit vertretenen Firmen und Einzelpersonen, die sicherlich qualifizierte Hilfe zu Ihrer Fragestellung anbieten können.

Bezugsquellen für Geodaten

Vermutlich wollen Sie möglichst schnell mit einem Projekt beginnen und möchten die vorgestellten neuen Werkzeuge endlich einmal ausprobieren. Vorher müssen allerdings noch ein paar Hausaufgaben erledigt werden. Der erste Schritt bei jedem Mapping-Projekt besteht immer darin zu ermitteln, welche Daten gebraucht werden. Der nächste Schritt ist dann, die gewünschten Informationen zu beschaffen.

Bewertung des Datenbedarfs

Die Anforderungen an Ihre Daten hängen sehr stark von Ihrem Ziel ab. Daher sollten Sie sich vorher auch sehr genau über das Endprodukt Ihrer Arbeit im Klaren sein. Sie müssen definieren, welche räumlichen Informationen Sie darstellen wollen und wie detailgenau diese Darstellung sein soll. Die folgenden Abschnitte sollen als Orientierungshilfe zur Definition Ihrer Anforderungen vor Beginn eines Projekts dienen.

Vektor- oder Rasterdaten?

Wollen Sie eigene Karten erstellen, müssen Sie herausfinden, welche Daten Sie dafür benötigen. Es gibt zwei Arten von Karten, die jeweils auf unterschiedlichen Datenformaten basieren:

Bei einer *Vektorkarte* werden die Objekte in Form von Punkten, Linien oder Polygonen dargestellt, wie in Abbildung 5-1 gezeigt wird. Dies kann beispielsweise eine örtliche Straßenkarte oder auch eine Weltkarte sein, auf der alle Kraftwerke und die zugehörigen Länder angezeigt werden.

Bei einer *Rasterkarte* dienen beispielsweise Luftbilder, Satellitenfotos oder computergenerierte Bilder der Erdoberfläche als Datenquellen. In Abbildung 5-2 sehen Sie ein Beispiel für eine Rasterkarte, deren Quelldaten von einem Wettersatelliten stammen.

Eine spezielle Form der Rasterkarte ist die eingescannte Karte. Wird eine Papierkarte eingescannt und in eine digitale Rasterkarte umgewandelt, kann sie als Ebene für andere Karten verwendet werden. Auf diese Weise können Daten leicht bereitgestellt werden, ohne dass die Karten erst digital rekonstruiert werden müssten.

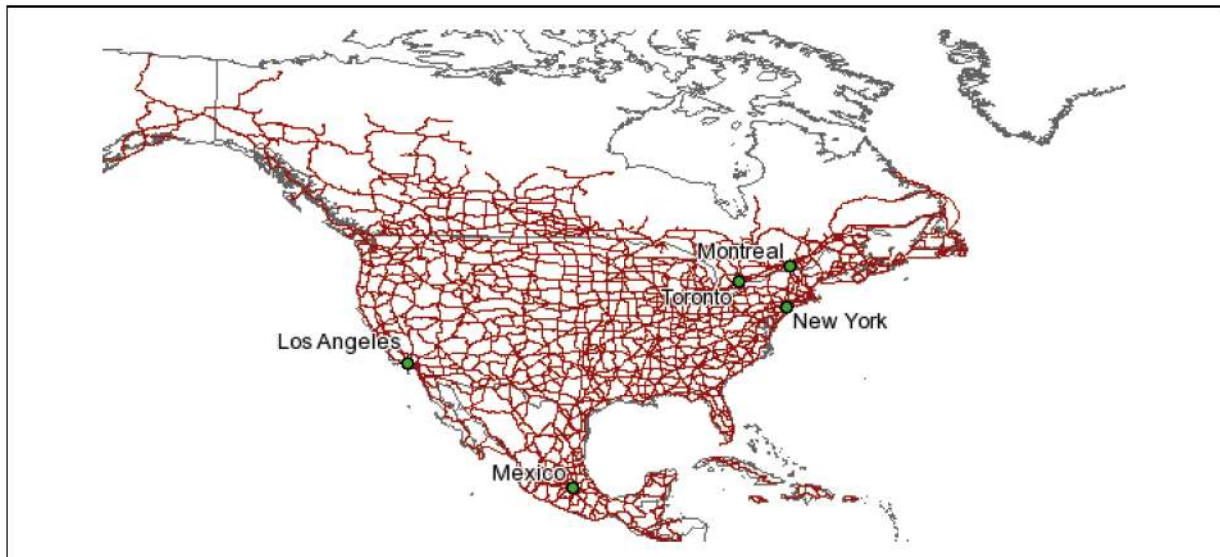


Abbildung 5-1: Vektorkarte von Nordamerika, die Straßen, Landesgrenzen und einige große Städte anzeigt



Abbildung 5-2: Rasterkarte, die das Wolkenbild über Nordamerika anzeigt

Oftmals werden Vektor- und Rasterdaten in einer Karte kombiniert. Auf diese Weise sind aussagekräftigere Darstellungen als bei der Verwendung nur eines Datentyps möglich (siehe Abbildung 5-3).

Welche Arten von Vektordaten werden benötigt?

Es gibt drei Arten von Vektordaten: Punkte, Linien und Polygone. Alle drei Arten können benutzt werden, um bestimmte Informationen abzubilden. Punkte können beispielsweise die Position von Städten oder Sendemasten anzeigen. Linien eignen sich für die Darstellung von Reiserouten oder Grenzen. Polygone können eingefärbt werden, um Seen anzuzeigen oder um ein Land auf einer Weltkarte zu markieren.

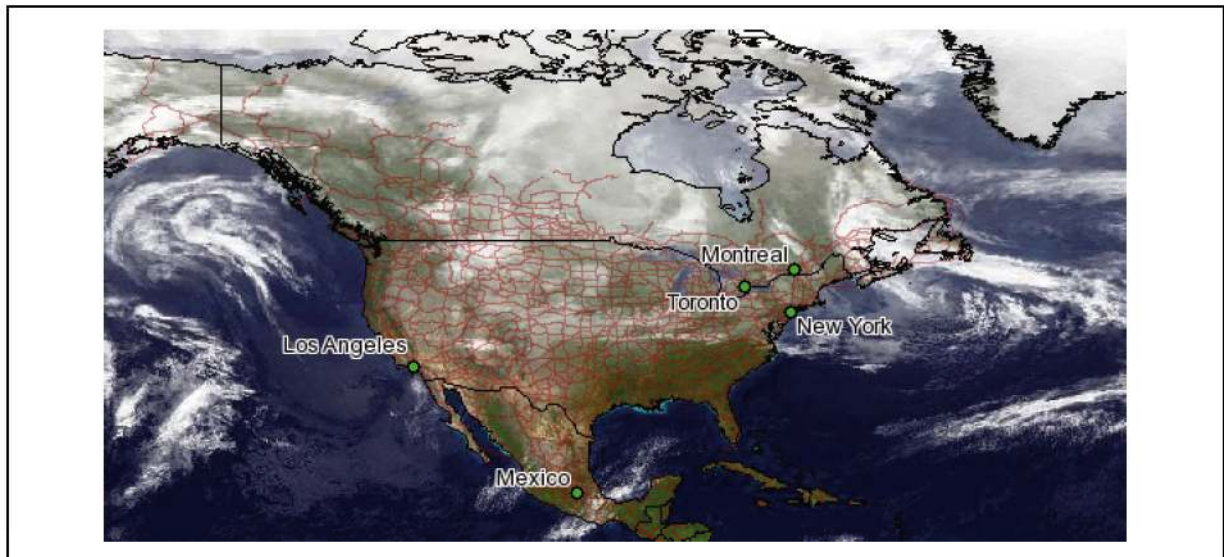


Abbildung 5-3: Eine aus Vektor- und Rasterdaten kombinierte Karte von Nordamerika

Für die Entscheidung, welche Arten von Vektordaten Sie für Ihre Karte brauchen, sollten Sie überlegen, was auf Ihrer Karte dargestellt werden soll. Sollen linienförmige Objekte mit Beschriftungen versehen werden? Planen Sie die Darstellung über eine einfache Koordinatenliste (z.B. eine CSV-Datei), um die Positionen bestimmter Punktoobjekte anzuzeigen, oder benötigen Sie komplexere Daten, um ein bestimmtes Gebiet darzustellen?

Welche Art von Rasterdaten werden benötigt?

Digitale Bilder, wie Luftbilder oder Satellitenfotos, gehören zu den rasterbasierten Daten. Ein Raster steht hierbei für ein Gitternetz aus Daten, das aus kleinen Quadraten – Zellen oder Pixeln – besteht. Jedem Quadrat kann ein oder mehrere Werte zugeordnet werden. Bei farbigen Bildern, z.B. Fotografien, werden die Anteile von Rot, Grün und Blau (RGB-Werte) pro Zelle angegeben. In digitalen Höhenmodellen (DHM, englisch: digital elevation model, DEM), wird auch die Höhe jeder Zelle gespeichert.

Überlegen Sie vor Ihrer Arbeit genau, welche Arten von Rasterdaten Sie verwenden wollen. Wenn Sie die Daten gefunden haben, können Sie entscheiden, ob sie in ihrer ursprünglichen Form nutzbar sind oder ob Sie sie anpassen oder verbessern müssen.

Welcher Maßstab wird benötigt?

Wenn Sie jemals versucht haben, einen kleinen See in der Nähe Ihres Hauses auf einer Weltkarte zu finden, werden Sie verstehen, dass Karten nicht immer jedes kleine Detail zeigen können. Erst wenn Sie die Weltkarte um ein Vielfaches vergrößern, werden Sie in der Lage sein, auch kleinere Details zu erkennen. Eine detailliertere, vergrößerte Ansicht hat einen größeren Maßstab, da die Dinge für den Betrachter vergrößert dargestellt werden. Eine Karte mit einem kleinen Maßstab verkleinert die Dinge, kann dafür aber einen größeren Bereich abdecken.



Die Verwendung von Maßstäben erscheint im ersten Moment nicht besonders intuitiv, die Schreibweise ergibt aber durchaus Sinn. Der Betrachter einer Karte könnte denken, dass der Maßstab 1:1000 größer ist als etwa 1:500, weil 1000 schließlich größer als 500 ist. Dies ist aber nicht der Fall. Der Maßstab wird als Verhältnis der gemessenen Distanz in der Karte zur wirklichen Distanz im Gelände angegeben. Die Angabe erfolgt als Bruch. 1:500 bedeutet daher, dass eine gemessene Einheit der Karte 500 Einheiten derselben Maßeinheit im Gelände entspricht. 1:500 steht für 1 geteilt durch 500 bzw. 0,001. Da 1 geteilt durch 500 einen größeren Wert ergibt als 1:1000, spricht man hier von einem *größeren* Maßstab.

Der Maßstab 1:500 zeigt mehr Details an, weil die Merkmale größer dargestellt werden als beim Maßstab 1:1000. Dies ist der zweite Grund, warum 1:500 als *größerer* Maßstab bezeichnet wird als 1:1000.

Das Verständnis von Größe und Maßstab Ihrer Karte ist sehr wichtig, denn hierdurch wird bestimmt, wie detailliert Ihre Ausgangsdaten sein müssen. Wenn Sie eine Karte Ihrer Nachbarschaft erstellen möchten, sind Weltkarten vermutlich nicht detailliert genug, um Ihre Straße anzuzeigen. Wollen Sie umgekehrt das weltweite Wetter abbilden, sind genaue Straßendaten mit ziemlicher Sicherheit unwichtig.

Die meisten Daten werden für die Betrachtung in einem bestimmten Maßstab erstellt. Versuchen Sie, die Daten in einem anderen Maßstab zu betrachten, sieht die Darstellung möglicherweise nicht richtig aus oder wird nicht korrekt zu anderen Objekten abgebildet. Entscheiden Sie daher zuerst, wie groß Ihre Karte sein soll – besonders wenn Sie die Karte ausdrucken möchten. Entscheiden Sie dann, was auf der Karte abgebildet werden soll: Ihr Land oder Ihre Stadt? Wie groß sollen die Objekte angezeigt werden? Handelt es sich um eine digitale Karte, müssen Sie definieren, welcher Bereich der Karte auf dem Bildschirm zu sehen sein soll. Die Antworten auf diese Fragen werden Ihnen dabei helfen herauszufinden, welchen Maßstab die benötigten Informationen haben müssen. Bei einer digitalen Karte haben Sie dabei natürlich eine wesentlich höhere Flexibilität als bei einem gedruckten Endprodukt.

Dritte Dimension gefällig?

Vielleicht wollen Sie Ihre Daten auch in einer dreidimensionalen (3-D-)Ansicht präsentieren. Diese Art der Analyse kann sehr komplex sein.

Dreidimensionale Darstellungen simulieren die Ansicht einer Landschaft aus der Luft oder vom Boden aus gesehen. Diese Karten werden in Bereichen eingesetzt, in denen ausgeprägtes Relief vorliegt, beispielsweise bei der Planung einer Bergtour oder eines Staudamms. Die üblichen zweidimensionalen Karten sind weiterhin hilfreich, doch kann der Informationswert durch eine dreidimensionale Darstellung noch gesteigert werden.

Wenn Sie 3-D-Karten erzeugen wollen, sollten Sie ein paar Dinge beachten. Vor allem müssen Sie entscheiden, wie Sie die Höheninformationen darstellen möchten. Ein Höhenmodell wird oft durch ein zweidimensionales Raster abgebildet. Dieses enthält helle und dunkle Pixel, die für verschiedene Höhenwerte stehen. Jedes Pixel besitzt also seine eigene Höhenangabe. In Abbildung 5-4 sehen Sie ein solches Rasterbild. Hierbei steht Schwarz für die niedrigsten und Weiß für die höchsten Bereiche.

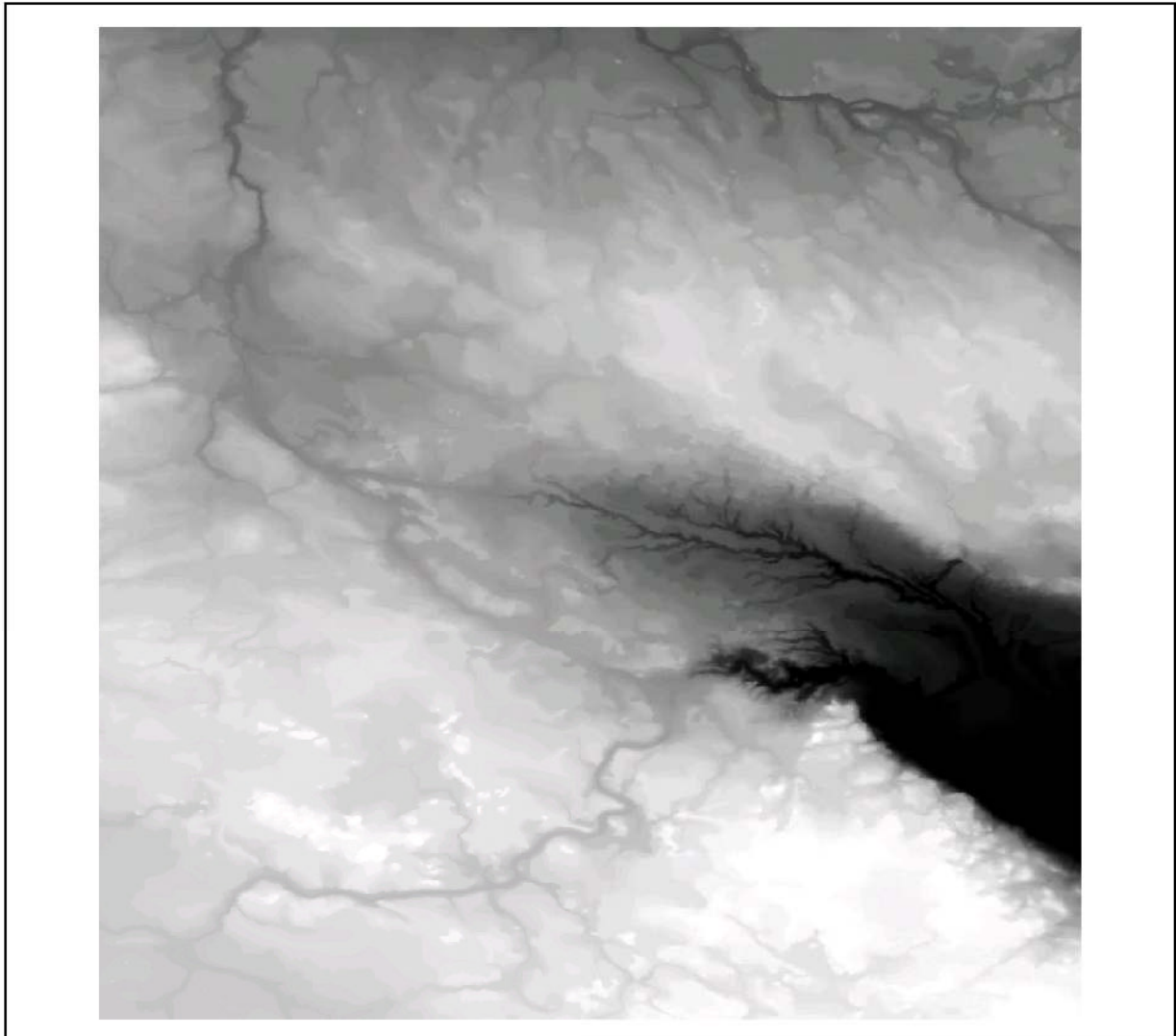


Abbildung 5-4: Beispiel für ein Höhenmodell von Nord-Minnesota, USA

Das zweite wichtige Element im Umgang mit dreidimensionalen Karten ist die Art der Informationen, mit denen Sie das Höhenmodell überlagern wollen. Raster- oder Vektordaten können mit dem sogenannten *Draping*-Verfahren auf das Höhenmodell gelegt werden. Hierbei werden die Informationen wie ein Stück dünner Stoff über eine unebene Oberfläche gelegt – oder wie Christos Verpackung um den Reichstag. Die »drapierte« Ebene nimmt die Form des darunter liegenden Höhenmodells an und gibt den

Informationen so eine eindeutige Perspektive. Abbildung 5-5 zeigt das gleiche Bild wie Abbildung 5-4, dieses Mal wurde es allerdings mithilfe von OpenEV in eine dreidimensionale Darstellung verwandelt.

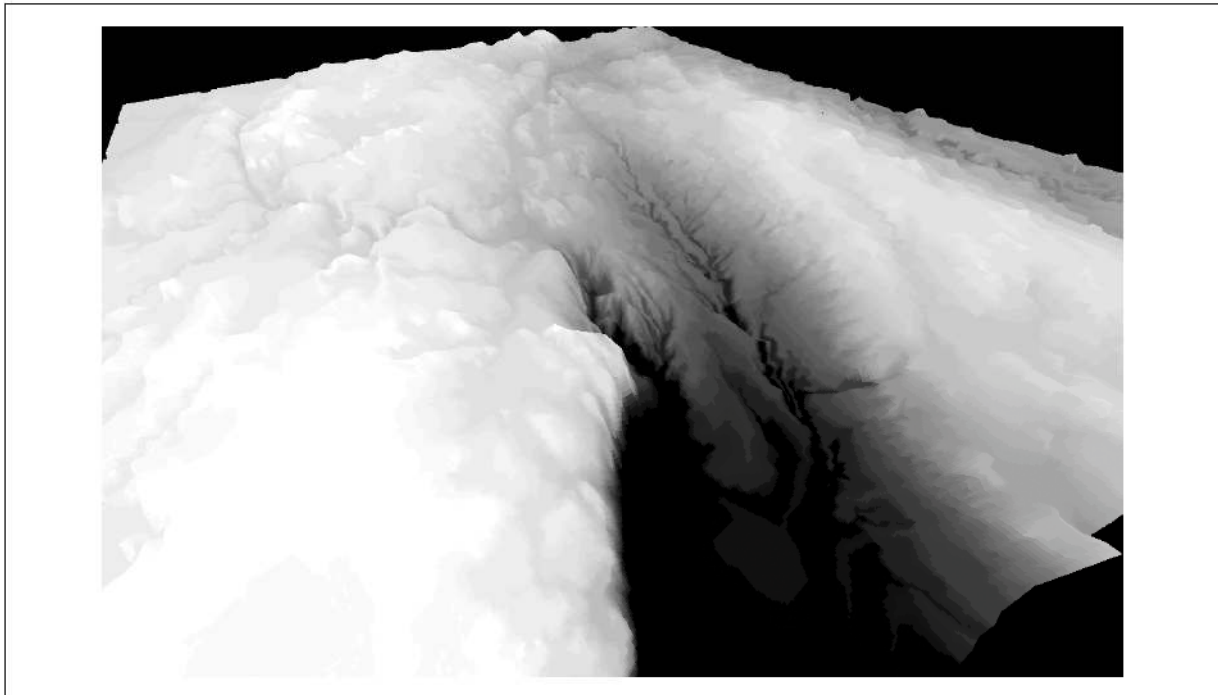


Abbildung 5-5: Beispiel für eine 3-D-Ansicht des Höhenmodells

Wenn Sie keine Daten für ein Höhenmodell finden können oder nicht wissen, welche Informationen Sie über Ihr Höhenmodell legen können, wird es schwierig, eine 3-D-Karte zu erstellen. Ein Beispiel für die Darstellung einer 3-D-Karte finden Sie in Kapitel 8.

Beschaffung geeigneter Daten

Sobald Sie wissen, was Sie für Ihr Kartenprojekt benötigen, müssen Sie die geeigneten Daten beschaffen. Diese finden Sie vor allem im Internet, z.B. auf den Seiten der öffentlichen Verwaltung. Im Internet gibt es sowohl fertige Karten als auch vollständige Datenbestände, die Sie herunterladen können. Dieses Kapitel enthält eine Reihe von Hinweisen, die Ihnen die Suche nach den Daten erleichtern soll.

Wo finden Sie vorgefertigte Karten?

Manchmal ist eine einfache Welt- oder Landkarte bereits ausreichend. Eine gute Quelle ist sicher das Internet, beispielsweise die Site CIA World Factbook unter <https://www.cia.gov/library/publications/the-world-factbook/index.html>. Hier finden Sie einfache politische Karten für jedes Land sowie detaillierte demografische, kulturelle und politische Zusammenfassungen. Großformatige Referenzkarten in Druckqualität liegen als PDF (Portable Document Format) vor.

Eine weitere gute Quelle ist die University of Texas Online Map Library unter <http://www.lib.utexas.edu/maps/>. Schon beim Überfliegen der vielen Links können Sie die verschiedensten Karten ausfindig machen, beispielsweise Stadtpläne, Länderübersichten oder auch eine Reliefkarte Großbritanniens, in der besonders die schottischen Highlands gut zur Geltung kommen. Die Bibliothek hat viele dieser Karten eingescannt und stellt sie der Öffentlichkeit kostenlos zur Nutzung und Weiterverarbeitung zur Verfügung.

Wo finden Sie Satellitenbilder oder Höhenmodelle?

Fertige Karten, die aus Bildern oder Höhenmodellen erstellt wurden, sind ebenso zu finden. Das CIA Factbook (<https://www.cia.gov/library/publications/the-world-factbook/index.html>) enthält kleine politische Karten für jedes Land. Weitere Referenzkarten zeigen physikalische Daten (z.B. Wassertiefen und Höhenangaben). Hier gibt es sogar fotografische Abbildungen der Erde. Auf den Seiten des U.S. National Geophysical Data Center (NGDC) sind ein paar schöne Galeriebilder zu sehen, die aus sehr detaillierten Quelldaten erstellt wurden. Zu finden ist die Galerie unter <http://www.ngdc.noaa.gov/mgg/topo/global.html>.

Wo finden Sie Geodaten?

Das Internet ist voll von räumlichen Informationen, die Sie für Ihre Projekte nutzen können. Daher bietet es sich an, hier mit der Suche nach den nötigen Daten zu beginnen. Einige staatliche Institutionen stellen ihre Daten im Internet frei zur Verfügung. Es gibt sogar spezielle Organisationen, die es sich zum Ziel gemacht haben, Geodaten kostenlos für andere bereitzustellen. Außerdem finden Sie einige durchsuchbare Kataloge für Geodaten. Einige Adressen hierzu erhalten Sie am Ende dieses Kapitels.

Es werden immer mehr räumliche Informationen zur freien Verfügung für alle bereitgestellt, vor allem über standardisierte OGC-Dienste. Mithilfe von standardisierten Diensten können Daten weitergegeben werden, ohne dass die Quelle kopiert werden muss. So wurde das Satellitenbild in Abbildung 5-6 beispielsweise von einem freien Web Map Service (WMS) angefordert. Weitere Informationen zu OGC und freien Diensten finden Sie in Kapitel 12.

Für die Vereinigten Staaten gibt es sehr viele Datenquellen im Internet. Auch für den Rest von Nordamerika sieht es recht positiv aus, wobei Kanada hier besonders gut abschneidet.

In Europa, Österreich, der Schweiz und Deutschland ist die Situation sehr heterogen. Leider gibt es durch die föderale Struktur der öffentlichen Verwaltung keinen zentralen Ansprechpartner. Neben diversen unterschiedlichen Interessenskonflikten führt der zunehmende finanzielle Druck auf die Vermessungs- und Katasterämter dazu, dass der Verkauf von Nutzungslizenzen für digitales Kartenmaterial zunehmend als Einnahmequelle genutzt werden soll. Das ist leider sowohl strategisch als auch politisch kurzfristig gedacht, da der Mehrwert von Geodaten in der Nachnutzung liegt. Anwendungen,

die durch die Nutzung der Geodaten zu besseren Planungsergebnissen führen oder neue Geschäftsfelder eröffnen, haben einen wesentlich höheren volkswirtschaftlichen Wert, als durch den Verkauf von Nutzungslizenzen eingenommen werden kann. Deshalb ist die restriktive Vorgehensweise vieler staatlicher Stellen wirtschaftlich und politisch gesehen ineffizient.

Das Auffinden globaler Daten bedeutet allerdings eine gewisse Herausforderung. Internationale Organisationen wie die Vereinten Nationen oder auch amerikanische Organisationen, die globale Ereignisse beobachten, wie z.B. die National Oceanic and Atmospheric Administration (NOAA, <http://www.noaa.gov/>), verfügen über recht gute globale Daten.

Im Internet können Sie aus einer Reihe von Datensammlungen auswählen. Meistens wurden die Daten für einen bestimmten Zielmaßstab zusammengetragen, der berücksichtigt werden muss. Ein Satellitenbild kann beispielsweise eine Ansicht Ihres Wohnorts enthalten. Abbildung 5-6 verwendet beispielsweise unterschiedliche Farben, um Straßen, Felder und Wasserstraßen abzubilden. Allerdings eignet sich diese Ansicht nur bis zu einem bestimmten Maßstab. Wenn Sie gehofft haben, die Karten verwenden zu können, um den nächsten Laden zu finden, werden Sie enttäuscht sein. Möchten Sie dagegen nur eine Übersicht Ihrer Stadt darstellen, sind vermutlich schon genügend Details vorhanden. In Abbildung 5-7 sehen Sie eine vergrößerte Ansicht des gleichen Bilds wie in Abbildung 5-6. Hier wurde die maximale Auflösung (ein Bildpixel wird von mehreren Bildschirmpixeln dargestellt) überschritten, und das Bild sieht unscharf aus. Dies war der Versuch, ein Bild außerhalb seines geeigneten Maßstabsbereichs anzuzeigen.



Abbildung 5-6: Ein Landsat-Satellitenbild der Region um Toronto



Abbildung 5-7: Eine (unscharfe) vergrößerte Ansicht des in Abbildung 5-6 verwendeten Landsat-Bilds

Es gibt viele Möglichkeiten, an gute Daten zu kommen. Sie können Ihre Suche in Mailinglisten bekannt geben oder auf Websites nachschauen, die frei verfügbare Datenquellen auflisten. Die Webseiten der Landesvermessungsämter, Kommunen und Kreise sind ebenfalls gute Startpunkte für die Geodaten-Suche.

Einige Adressen zu frei verfügbare Geodaten erhalten Sie am Ende dieses Kapitels in Tabelle 5-1.

Datensuche und -zugriff sind aber nicht die einzigen Probleme. Einige Daten sind einfach nicht verwendbar oder für die anstehende Aufgabe ungeeignet. Hierbei geht es weniger um Maßstab und Auflösung als um Genauigkeit und Informationswert der Daten. Ein Beispiel ist die Forstinitiative der Regierung von British Columbia (Kanada), bei der es darum ging, den Schädlingsbefall von Bäumen zu kartieren. Hierfür wurde das betreffende Gebiet von einem Flugzeug überflogen, das mit einem Aufklärungs-werkzeug ausgestattet war, das die kranken Bäume ermittelte und ihren Standort auf einer Papierkarte einzeichnete. Im Büro wurden die Papierkarten dann digitalisiert und auf eine Grundkarte im Computer übertragen. Diese Karten sind zwar ausreichend, um landschaftliche Analysen durchzuführen, allerdings sollten Sie nicht erwarten, mithilfe dieser Daten einen bestimmten Baum finden zu können. Das Verständnis für den sinnvollen Einsatz Ihrer Daten wird Ihnen dabei helfen, beim Beginn eines neuen Projekts realistische Erwartungen zu entwickeln.

Das Internet ist übrigens nicht die einzige Datenquelle. In Kapitel 9 sehen Sie, wie Sie Ihre eigenen Daten mithilfe von Desktop-GIS selbst erstellen können.

OSGeo und öffentliche Geodaten

Neben der Software und Bildung und Lehre fördert die OSGeo als dritten Schwerpunkt die freie Verfügbarkeit von Geodaten. Geodaten entstehen auf allen Ebenen und in verschiedensten Formen, ein großer Bereich kommt aus der öffentlichen Verwaltung und ist damit auch ein öffentliches Gut. Dieses Gut nutzbar zu machen und zu fördern, ist eine langfristige und nachhaltige Aufgabe, die durch die Verwendung von Freier und Open Source-Software erheblich erleichtert wird. Die OSGeo wird zunehmend auch als Ansprechpartner bei Lizenzierungsfragen genutzt, sei es zur kommerziellen Nutzung öffentlicher oder komplett frei verfügbarer Geodaten oder zur Kooperation mit der privaten Wirtschaft. Anfragen kommen sowohl vom kleinen Ingenieurbüro als auch von Google. Dabei bietet die OSGeo als unabhängige, offene und kollaborative Gemeinschaft ein einzigartiges und umfangreiches Forum für Diskussionen mit allen Beteiligten.

Weitere Informationen finden Sie unter <http://www.osgeo.org/geodata>.

Freie Geodaten

Neben diesen hochqualitativen, aber leider immer noch sehr heterogen strukturierten öffentlichen Geodaten gibt es auch ganz freie Geodaten. Diese Entwicklung wurde noch vor wenigen Jahren als vollkommen abstrus betrachtet, hat aber inzwischen ganz beträchtliche Dimensionen angenommen. Es gibt mehrere Kristallisationspunkte, der interessanteste ist derzeit sicherlich das Projekt *OpenStreetMap*, das als »Wikipedia der Geodaten« per GPS gesammelte Geodaten aufbereitet und bereitstellt. Die Webseite des Projekts lautet <http://www.openstreetmap.org>.

Im OpenStreetMap-Wiki steht auf der deutschsprachigen Hauptseite zu lesen:

OpenStreetMap hat es sich zum Ziel gesetzt, freie geographische Daten wie zum Beispiel Straßenkarten zu erstellen und allen zur Verfügung zu stellen, die sie brauchen können. Dieses Projekt wurde begonnen, weil so ziemlich alle Karten, von denen man glaubt, dass sie von jedem für alle Zwecke und kostenlos verwendbar sind, tatsächlich aus technischen und rechtlichen Gründen leider nur eingeschränkt nutzbar sind. Dadurch wird verhindert, dass Leute diese Karten in kreativer und produktiver Weise einsetzen können. OpenStreetMap möchte diesem unglücklichen Zustand abhelfen. (Quelle: <http://wiki.openstreetmap.org/index.php/Hauptseite>)

Öffentliche Geodaten in Europa

Freie Geodaten sind derzeit im europäischen Raum kaum und in den DACH-Ländern (Deutschland, Österreich, Schweiz) nur schwer zu bekommen. Das ist schade, vor allem vor dem Hintergrund, dass gerade diese Länder mit mehreren hundert Jahren Erfahrung in der Geodäsie und bereits vermessenen Daten auf der Weltrangliste digitalisierter Geodaten ganz vorne stehen. Vor allem in den USA wurden bereits in den Achtzigern politi-

sche Entscheidungen getroffen, die zumindest einen flächigen Grunddatenbestand an Geodaten bereitstellen. Allerdings ist die Qualität dieser Daten in keiner Weise vergleichbar mit den Daten der öffentlichen Verwaltung in den DACH-Ländern.

Auf europäischer Ebene ist durch die Verabschiedung der INSPIRE-Richtlinie ein erster Schritt in die bessere Verfügbarmachung öffentlicher Geodaten gegangen worden: <http://www.ec-gis.org/inspire/>.

Deutschland

Qualitativ hochwertige Basisgeometrien werden in Deutschland als Katastergrundkarte geführt. Diese Daten werden nicht zentral gehalten, sondern sind je nach Bundesland entweder dezentral organisiert (z.B. in Nordrhein-Westfalen) oder landesweit zentral (Bayern). Beide Varianten haben Vor- und Nachteile: So ist z.B. die flächige Verfügbarkeit von Daten in Bayern besser, die Aktualität und vor allem die Vielfalt unterschiedlicher Daten ist dagegen in NRW unschlagbar. Beide Angebote werden über Geodatendienste (z.B. WMS, WFS) bereitgestellt, die entweder bereits produktiv laufen, in Vorbereitung oder zumindest geplant sind. Auf Bundesebene werden diese Dienste über die GDI-DE zusammengeführt. Die Geodateninfrastruktur Deutschland versteht sich dabei als Broker (Vermittler) zwischen Datennutzer und Anbieter. Diese Architektur lässt sich auch auf die europäische Ebene erweitern, hier wurde im Jahr 2007 die INSPIRE-Richtlinie verabschiedet, um eine erste Harmonisierung der Anbieter und Nutzerrechte zu beginnen. Die Seiten der GDI-DE sind in ein Portal eingebunden, das unter anderem auch Open Source-Komponenten einsetzt (<http://geoportal.bkg.bund.de/>).

Schweiz

Die Schweiz hat mit dem Bundesamt für Landestopografie Swisstopo (<http://www.swisstopo.ch/>) einen zentral organisierten Ansprechpartner und qualitativ hochwertige Geodaten. Diese stehen zwar nicht zentral ganz frei zum Herunterladen bereit, auf Nachfrage ist es aber möglich, Datensätze zu erwerben oder zur weiteren Nutzung als Dienst zu bekommen. Einige Kantone bieten bereits OGC-standardisierte Dienste mit Geodaten an, die über die Portalseiten der Verwaltungen genutzt werden können.

Vorbildlich organisiert und gleichzeitig auch ein Vorreiter in der Nutzung des UMN Map-Server ist der Kanton Solothurn (siehe auch Abbildung 1-5 in Kapitel 1), der seine Dienste hier bereitstellt: <http://www.so.ch/departemente/bau-justiz/so-gis/web-map-services-wms.html>. Der Kanton ist dafür bekannt, im GIS-Bereich ganz auf Open Source zu setzen, im WebGIS-Bereich zum Beispiel auf den UMN MapServer. Das WMS-Angebot des Kantons ist vorbildlich dokumentiert und listet eine große Zahl verfügbarer Ebenen auf. Bei jeder Nutzung ist immer auf das Copyright zu achten und an welche Bedingungen die Verwendung der Daten gebunden sind.

Im Kanton Thurgau (<http://www.geoinformation.tg.ch/>) wird die rechtliche Seite in den Lizenzbedingungen einfach und verständlich dokumentiert, der Datenbezug ist kostenlos. Leider ist die Schweiz noch nicht flächendeckend so gut ausgestattet, da die Kantone sehr unterschiedliche Vorgehensweisen bei Geodaten verfolgen.

Österreich

In Österreich gibt es Zuständigkeiten sowohl auf Landes- als auch auf Bundesebene. Wien als größte Stadt geht teilweise eigene Wege, was zu einer ähnlich verwirrenden Vielfalt führt, wie es in Deutschland der Fall ist. Zunehmend bieten Kommunen eigene Daten so wie in Deutschland und der Schweiz über Portale und Dienste an. Auch hier ist die Frage der rechtlichen Bedingungen oft ungeklärt. Vor einer professionellen Nutzung dieser Daten ist deshalb unbedingt sorgfältig das jeweilige Nutzungsrecht zu prüfen; die Nutzung sollte immer mit einem schriftlichen Vertrag abgesichert werden. Die Länder stellen Geodaten über Dienste bereit und organisieren sich im Geoportal AT auf den Seiten <http://www.geoportal.at>. Zu den kommunalen Beispielen, die eigene Daten als Dienste anbieten, gehört unter anderen die Gemeinde Zwischenwasser <http://www.zwischenwasser.at/>.

Tabelle 5-1: Links zu Diensten, Geoportalen und verfügbaren Geodaten

Organisation	Website
Canada GeoBase Portal	http://www.geobase.ca/
Kanadische Regierung	http://geogratis.cgdi.gc.ca/download/
Earth Observation Portal	http://maps.eoportal.org/
FreeGIS – Softwareübersicht, Datensammlung und Mailingliste	http://freegis.org
GeoConnections Discovery Portal	http://geodiscover.cgdi.ca/
GeoCommunity	http://data.geocomm.com/catalog/
Global Change Master Directory	http://gcmd.nasa.gov/
Global Land Cover	http://glcf.umiacs.umd.edu/data/
GTOPO-Höhendaten	http://edcdaac.usgs.gov/gtopo30/gtopo30.asp
OSGeo Public Geospatial Data Project	http://wiki.osgeo.org/index.php/Public_Geospatial_Data_Project
Linkliste zu Geodaten	http://de.wikipedia.org/wiki/Wikipedia:Karten
Links zu freien Geodaten	http://de.wikipedia.org/wiki/Geodaten#Freie_Geodaten
MapServer	http://mapserver.gis.umn.edu
OpenStreetMap	http://www.openstreetmap.org/
INSPIRE	http://www.ec-gis.org/inspire/b
GISCO – Dienst des Europäischen Statistischen Amtes (Eurostat)	http://epp.eurostat.ec.europa.eu/portal/page?_pageid=2254,62148876,2254_62153840&_dad=portal&_schema=PORTAL
GDI-DE (Geodateninfrastruktur Deutschland)	http://www.gdi-de.org/

Tabelle 5-1: Links zu Diensten, Geoportalen und verfügbaren Geodaten (Fortsetzung)

Organisation	Website
AdV – Koordination des Amtlichen deutschen Vermessungswesens	http://www.adv-online.de/
Landesvermessungsamt Nordrhein-Westfalen	http://www.lverma.nrw.de/
Tim-Online	http://www.tim-online.nrw.de/
GeoPortal.rlp	http://www.geoportal.rlp.de/
PortalU – Umweltportal Deutschland	http://www.portalu.de/
Umweltdatenkatalog Niedersachsen	http://www.umweltdatenkatalog.niedersachsen.de/
Umweltdatenkatalog Baden-Württemberg	http://www2.lfu.baden-wuerttemberg.de/wwwudk/UDKServlet
Schweizerischer Umweltdatenkatalog	http://www.envirocat.ch/
SO!GIS Kanton Solothurn	http://www.so.ch/departemente/bau-justiz/so-gis/
Amt für Geoinformation Thurgau	http://www.geoinformation.tg.ch/
GEOLAND.AT – Geodatenportal der österreichischen Länder	http://www.geoland.at/
Topografische Daten der NASA	http://www2.jpl.nasa.gov/srtm/dataproduct.htm
U.S. Geologic Survey	http://edcscns17.cr.usgs.gov/EarthExplorer/
U.S. Maps and Data	http://geodata.gov
U.S. National Atlas	http://www.nationalatlas.gov/
U.S. National Geospatial Data Clearinghouse	http://clearinghouse1.fgdc.gov/

Geodaten analysieren

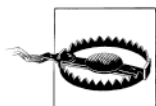
Wenn Sie geeignete Geodaten vorliegen haben, ist es oft wichtig, die Daten zu untersuchen oder zu bearbeiten, bevor diese in einer Karte genutzt werden können. Dieses Kapitel stellt einige Werkzeuge vor und zeigt deren Einsatz anhand von Beispielen mit Demodaten. Einige der Beispieldaten werden auch in anderen Kapiteln dieses Buchs verwendet. Die hier gezeigten Techniken produzieren zwar noch keine außergewöhnlichen Karten, es soll aber vermittelt werden, wie einfach es ist, eigene Karten zu erstellen.

Download der Demodaten

In diesem Abschnitt zeigen wir Ihnen, wie Sie ein paar recht einfache Daten über den US-Staat Minnesota herunterladen und einsetzen können. Um die Dinge unkompliziert zu halten, verwenden wir einen bekannten MapServer-Beispieldatensatz.

Die vollständige URL zu den Demodaten lautet http://maps.dnr.state.mn.us/mapserver_demos/workshop.zip. Laden Sie die Datei herunter und entpacken Sie den Inhalt. Sie brauchen nur das Verzeichnis *workshop/data*.

Auf der MapServer-Website finden Sie neben den aktuellen Versionen von MapServer selbst auch eine Reihe von Beispieldatensätzen und Anwendungen unter <http://mapserver.gis.umn.edu/download/current/>.



Es ist nicht wichtig, wo Sie die im Archiv enthaltenen Ordner entpacken, sondern nur, dass die Verzeichnisnamen erhalten bleiben. Bei den meisten Archivprogrammen ist das der Standard.

Installation von GDAL und FWTools

Wenn Sie sich den Inhalt des Verzeichnisses *data* genauer ansehen, werden Sie hier Daten verschiedenster Formate vorfinden. Sie werden nun Werkzeuge des GDAL-Projekts kennenlernen, um mehr über die Daten zu erfahren. GDAL liegt im FWTools-

Paket vor. FWTools ist eine Sammlung von Open Source-GIS-Bibliotheken. Zum FWTools-Paket gehört auch das Programm OpenEV, mit dem Geodaten visualisiert werden können.

Die aktuellste Version von FWTools erhalten Sie unter <http://fwtools.maptools.org/>. Die Beispiele in diesem Buch beziehen sich auf die Linux-Systeme.

Windows-Benutzer

Für Windows liegt FWTools als Setup-Datei vor. Laden Sie die Datei herunter und führen Sie das Installationsprogramm aus. Bei der Installation wird ein Icon für die FWTools-Shell auf Ihrem Desktop angelegt. Ein Doppelklick auf dieses Icon startet die FWTools-Eingabeaufforderung, in die Sie Befehle eingeben können.

Linux-Benutzer

Laden Sie zuerst den aktuellsten Tarball von FWTools herunter (z.B. *FWTools-linux-1.4.2.tar.gz*) und entpacken Sie diesen in ein Verzeichnis Ihrer Wahl. Linux-Benutzer werden hier auf der Kommandozeile (Shell) arbeiten und die nachfolgenden Befehle eingeben.

Öffnen Sie eine Shell und wechseln Sie mittels des Befehls `cd` gefolgt von dem entsprechenden Pfad in das Verzeichnis, in dem Sie die *tar*-Datei gespeichert haben. Mit dem folgenden Befehl können Sie die Datei in einem Schritt entpacken:

```
> tar -xzf FWTools-linux-1.4.2.tar.gz
```

Wechseln Sie in das Verzeichnis *FWTools-linux-1.4.2*. Um das Installationsprogramm auszuführen, benutzen Sie den Befehl `./install.sh`. Hiermit wird die Installation vorbereitet und ein Verzeichnis mit dem Namen *bin_safe* angelegt. Dieser Verzeichnispfad zu *bin_safe* muss nun der Variablen `$PATH` hinzugefügt werden:

```
> PATH = $PATH:/demo/FWTools-linux-1.4.2/bin_safe
> export PATH
```

Nun können Sie die FWTools-Programme nutzen. Genauere Informationen zur Installation unter Linux liefert die im Paket enthaltene Datei *README.1ST*.

Sie werden im Folgenden einige Programme zur Bearbeitung, Visualisierung, Analyse und Konvertierung kennenlernen. In den Beispielen ist den auszuführenden Befehlen jeweils ein `>` vorangestellt. Das `>`-Zeichen ist dabei kein Teil des Befehls, dieser beginnt erst nach dem `>`-Zeichen. Das Kommandozeilensymbol kann je nach System unterschiedlich sein.

Dateninhalte untersuchen

Die Untersuchung des Aufbaus und Inhalts der Daten ist ein wichtiger Teil jedes Projekts. Das Wissen über den Aufbau und die Struktur Ihrer Daten hilft Ihnen, die Daten sinnvoll und effektiv zu nutzen. Geodaten verfügen über räumliche Informationen sowie über weitere Sachinformationen, die *Attribute*. Attribute enthalten Informationen über die Geometrieobjekte, wie beispielsweise Einwohnerzahl, Straßename, die Farbe

eines Sees usw. Oftmals werden die Koordinaten als räumliche Daten bezeichnet, während die Informationen der Daten Attribute oder auch tabellarische Daten genannt werden. Sobald ein Datensatz räumliche Daten enthält, kann von Geodaten gesprochen werden.

ogrinfo – Ausgabe von Basisinformationen zu Vektordaten

Die MapServer-Beispieldaten enthalten eine Reihe von Vektordateien. Mithilfe des Werkzeug `ogrinfo` können Sie sich Informationen über den Inhalt dieser Dateien ausgeben lassen. Wechseln Sie auf der Kommandozeile in das Verzeichnis `workshop` und führen Sie den Befehl `ogrinfo` unter Angabe des Verzeichnisses `data` als Übergabeparameter aus. Daraufhin listet `ogrinfo` Ihnen sämtliche Datensätze auf, die sich im Verzeichnis `data` befinden. Die Ausgabe des Befehls sehen Sie in Beispiel 6-1.

Beispiel 6-1: Liste der verfügbaren Layer (Shape-Dateien) im Verzeichnis data

```
> ogrinfo data
INFO: Open of 'data'
using driver 'ESRI Shapefile' successful.
1: twprgpy3 (Polygon)
2: rmpdln3 (Line String)
3: lakespy2 (Polygon)
4: stprkpy3 (Polygon)
5: ctyrdln3 (Line String)
6: dlgstln2 (Line String)
7: mcd90py2 (Polygon)
8: twprdln3 (Line String)
9: plsscpy3 (Polygon)
10: mcdrdln3 (Line String)
11: majrdln3 (Line String)
12: drgidx (Polygon)
13: airports (Point)
14: ctybdpy2 (Polygon)
```

Die Ausgabe zeigt, dass das `data`-Verzeichnis 14 Layer enthält. Außerdem wird angezeigt, dass die Daten im Verzeichnis im ESRI Shape-Format vorliegen. Jede Shape-Datei entspricht einem Layer in der Liste. Wenn Sie sich die Dateien im Verzeichnis ansehen, werden Sie allerdings feststellen, dass es wesentlich mehr als 14 Dateien gibt. Eine Shape-Datei besteht immer aus mehreren Dateien: Eine enthält die Geodaten, eine weitere die Informationen usw.

Wenn Sie dem Befehl `ogrinfo` den Namen eines Layers und den Parameter `-summary` hinzufügen, erhalten Sie weitere Informationen zur jeweiligen Shape-Datei (Beispiel 6-2).

Beispiel 6-2: Nutzung von ogrinfo zur Ausgabe von Informationen über den Aufbau eines Layers

```
> ogrinfo -summary data airports
INFO: Open of 'data'
using driver 'ESRI Shapefile' successful.
```


Beispiel 6-2: Nutzung von ogrinfo zur Ausgabe von Informationen über den Aufbau eines Layers (Fortsetzung)

```
Layer name: airports
Geometry: Point
Feature Count: 12
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
Layer SRS WKT:
(unknown)
NAME: String (64.0)
LAT: Real (12.4)
LON: Real (12.4)
ELEVATION: Real (12.4)
QUADNAME: String (32.0)
```

Dieses Beispiel zeigt Informationen zum Layer airports (Flughäfen).

```
Geometry: Point
```

Eine Shape-Datei kann jeweils nur einen Geometrietyp enthalten. Die Shape-Datei *airports* enthält Punkte. Im nächsten Beispiel werden Sie sehen, dass jeder Flughafen seine Position über ein Koordinatenpaar erhält.

```
Feature Count: 12
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
```

Über Feature Count wird die Anzahl der Geometrieobjekte des Layers ausgegeben. Der Extent gibt die linke untere und rechte obere Koordinate des Umgebungsrechtecks an, mit dem die Punkte umschlossen werden können. Die Koordinaten werden in Metern angegeben und verwenden die Universal Transverse Mercator-(UTM-)Projektion.

```
Layer SRS WKT:
(unknown)
```

Layer SRS WKT gibt die Projektion, in der die Daten vorliegen, aus. SRS steht für Spatial Reference System und WKT für das Well-known-Textformat. Diese Begriffe wurden beide vom OGC entwickelt. Hierbei enthält SRS unter anderem Informationen über die Projektion, Datumsangaben und die Einheiten der Daten. WKT ist ein Format, über das räumliche Informationen als Text definiert werden können (im Gegensatz zu binären Formaten). Weitere Informationen zu Projektionen, SRS und EPSG finden Sie in Anhang A. Mehr zum OGC und der Rolle des OGC bei der Erstellung von Standards finden Sie in Kapitel 12.

Die folgende Zeile, unknown, zeigt, dass bei der Erstellung der Daten keine Informationen über die Projektion angegeben wurden. Wenn Sie nicht wissen, wo sich diese Daten befinden, ist diese Angabe nicht besonders hilfreich. Kennen Sie sich allerdings ein wenig mit Geodaten aus, werden Sie sehen, dass es sich um UTM-Daten handelt.

```
NAME: String (64.0)
LAT: Real (12.4)
LON: Real (12.4)
ELEVATION: Real (12.4)
QUADNAME: String (32.0)
```

Diese fünf Zeilen geben nicht räumliche Informationen aus, über die jede Geometrie verfügt. Diese Zusatzinformationen werden oftmals auch als Attribute, Eigenschaften, Spalten oder Sachdateninformationen bezeichnet. Jedes Attribut hat einen Namen und kann einen bestimmten Datentyp enthalten. Bei der Ausgabe durch *ogrinfo* steht vor dem Doppelpunkt der Name des Attributs. Lassen Sie sich nicht davon verwirren, dass die erste Zeile ein Attribut enthält, das selbst NAME heißt. Das Wort nach dem Doppelpunkt zeigt an, welchen Datentyp ein Attribut enthalten kann – String (Zeichenkette) oder Real (Zahlen). Die Zahlen in Klammern machen genauere Angaben über die Größe des Felds. So bedeutet NAME: String (64.0), dass das Attribut NAME maximal 64 Buchstaben oder Ziffern enthalten kann. Entsprechend steht ELEVATION: Real (12.4) für das Attribut ELEVATION, dessen Wert aus einer höchstens zwölfstelligen Zahl mit maximal vier Nachkommastellen bestehen darf.

Vielleicht fragen Sie sich jetzt, warum diese Informationen so wichtig sind. Die meisten Fehler bei der Verwendung räumlicher Daten entstehen durch Unwissenheit über den Aufbau der Daten. Daher kann die Untersuchung der Daten mit Werkzeugen wie *ogrinfo* vor der Kartenerstellung hilfreich sein. Ein fundamentales Verständnis Ihrer Daten verhindert Fehler während der Kartenerstellung oder bei Ihren Analysen. Wenn Ihre Analyse sich auf Zahlen bestimmter Genauigkeit oder genau definierte Textlängen verlässt, sollten Sie sicherstellen, dass die verwendeten Daten diese Vorgaben auch erfüllen. Ansonsten werden Sie schnell falsche Ergebnisse bekommen.

ogrinfo zur Ausgabe von Detailinformationen zu Vektordaten

Die Zusammenfassung über die Daten kann natürlich nur eine Übersicht über diese geben. Mit *ogrinfo* lassen sich auch detaillierte Informationen über die räumlichen Daten und ihre Attribute ausgeben. Um sich anstelle der Zusammenfassung die vollständigen Daten anzeigen zu lassen, rufen Sie *ogrinfo* unter Angabe des Verzeichnisses und des gewünschten Layers auf, wie in Beispiel 6-3 gezeigt. Allerdings verzichten Sie diesmal auf die Option *-summary*.

Beispiel 6-3: ogrinfo zur Anzeige der Informationen und der Datensätze einer Shape-Datei

```
> ogrinfo data airports
INFO: Open of 'data'
using driver 'ESRI Shapefile' successful.

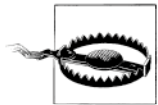
Layer name: airports
Geometry: Point
Feature Count: 12
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
Layer SRS WKT:
(unknown)
NAME: String (64.0)
LAT: Real (12.4)
LON: Real (12.4)
ELEVATION: Real (12.4)
QUADNAME: String (32.0)
```

Beispiel 6-3: ogrinfo zur Anzeige der Informationen und der Datensätze einer Shape-Datei (Fortsetzung)

```
OGRFeature(airports):0
  NAME (String) = Bigfork Municipal Airport
  LAT (Real) =    47.7789
  LON (Real) =   -93.6500
  ELEVATION (Real) = 1343.0000
  QUADNAME (String) = Effie
  POINT (451306 5291930)

OGRFeature(airports):1
  NAME (String) = Bolduc Seaplane Base
  LAT (Real) =    47.5975
  LON (Real) =   -93.4106
  ELEVATION (Real) = 1325.0000
  QUADNAME (String) = Balsam Lake
  POINT (469137 5271647)
```

Diese Detailansicht zeigt die Attributwerte für die Objekte. Wie Sie sehen, wird auch am Anfang der Ausgabe die Zusammenfassung angezeigt. Darauf folgen jedoch die detaillierten Abschnitte für jedes einzelne Objekt. In diesem Fall werden sieben Zeilen oder Attribute für jedes Objekt ausgegeben. So sehen Sie beispielsweise den Namen sowie die UTM-Koordinaten (POINT).



In der Shape-Datei *airport* liegen die numerischen Felder LAT und LON vor. Diese haben nichts mit der Darstellung der Geometrie zu tun. Nicht alle Arten von Punktdaten besitzen diese zwei Attribute. Es handelt sich einfach nur um zwei Attributspalten. Die eigentlichen UTM-Koordinaten finden sich im letzten Attribut POINT.

In diesem Beispiel werden nur zwei Objekte ausgegeben, von denen das erste mit `OGRFeature(airports):0` beginnt. Die komplette Ausgabe endet mit `OGRFeature(airports):11`. Es liegen also zwölf Flughäfen als Datensätze vor.

`ogrinfo` ist hervorragend dazu geeignet, mehr über Daten zu erfahren. Für den Aufruf gibt es noch eine Reihe weiterer Optionen. Hierzu gehört auch eine Art Datenbankabfrage. Über den Aufruf von `ogrinfo --help` erhalten Sie eine Zusammenfassung der Optionen. Sofern Ihr System die Manpages (Hilfe- und Dokumentationsseiten unter Unix) unterstützt, erstellt Ihnen der Befehl `man ogrinfo` eine vollständige Liste zur Verwendung der einzelnen Parameter. Zusätzlich finden Sie eine Beschreibung der Programme auf der OGR-Website unter http://gdal.org/ogr/ogr_utilities.html.

Beispiel 6-4 zeigt, wie die Ausgabe mithilfe der Option `-where` auf die gewünschten Objekte eingeschränkt werden kann.

Beispiel 6-4: ogrinfo unter Verwendung von -where zur abfrageabhängigen Ausgabe von Objekten

```
> ogrinfo data airports -where "name='Bolduc Seaplane Base'"
INFO: Open of 'data'
using driver 'ESRI Shapefile' successful.
```

Beispiel 6-4: ogrinfo unter Verwendung von -where zur abfrageabhängigen Ausgabe von Objekten (Fortsetzung)

```
Layer name: airports
Geometry: Point
Feature Count: 1
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
Layer SRS WKT:
(unknown)
NAME: String (64.0)
LAT: Real (12.4)
LON: Real (12.4)
ELEVATION: Real (12.4)
QUADNAME: String (32.0)
OGRFeature(airports):1
  NAME (String) = Bolduc Seaplane Base
  LAT (Real) = 47.5975
  LON (Real) = -93.4106
  ELEVATION (Real) = 1325.0000
  QUADNAME (String) = Balsam Lake
  POINT (469137 5271647)
```

Dieses Beispiel zeigt nur die Flughäfen, deren Name Bolduc Seaplane Base ist. Wie Sie sich vorstellen können, gibt es lediglich einen solchen Flughafen. Es werden daher nur die Zusammenfassung und die Attribute des einen Objekts ausgegeben, auf das die Abfrage in Beispiel 6-4 zutrifft. Mithilfe der Option `-sql` können Sie zusätzlich angeben, welche Attribute in der Ausgabe von `ogrinfo` berücksichtigt werden sollen (Beispiel 6-5).



Bei der Verwendung der Option `-sql` können Sie eine SQL-Abfrage angeben. Wenn Sie sich mit SQL noch nicht auskennen, erhalten Sie über die folgenden Quellen einen Einstieg:

- *Einführung in SQL* (O'Reilly)
- *SQL von Kopf bis Fuß* (O'Reilly)

Die SQL-Implementierung in `ogrinfo` ist nicht vollständig und unterstützt nur `SELECT`-Abfragen.

Beispiel 6-5: ogrinfo unter Verwendung von -sql. Auswahl bestimmter Objekte unter Angabe der gewünschten Attribute für die Ausgabe im Ergebnis

```
> ogrinfo data airports -sql "select name from airports where quadname='Side Lake'"
INFO: Open of 'data'
using driver 'ESRI Shapefile' successful.
layer names ignored in combination with -sql.
```

```
Layer name: airports
Geometry: Point
Feature Count: 2
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
Layer SRS WKT:
(unknown)
name: String (64.0)
```


Beispiel 6-5: ogrinfo unter Verwendung von -sql. Auswahl bestimmter Objekte unter Angabe der gewünschten Attribute für die Ausgabe im Ergebnis (Fortsetzung)

```
OGRFeature(airports):4
  name (String) = Christenson Point Seaplane Base
  POINT (495913 5279532)

OGRFeature(airports):10
  name (String) = Sixberrys Landing Seaplane Base
  POINT (496393 5280458)
```

In der SQL-Abfrage wird angegeben, dass für die Ergebnismenge nur das Attribut NAME angezeigt werden soll. Standardmäßig werden zusätzlich die Koordinaten ausgegeben. In der WHERE-Bedingung der SQL-Abfrage wird definiert, dass nur die Objekte ausgegeben werden sollen, deren Attribut QUADNAME einen bestimmten Wert (»Side Lake«) hat. In Beispiel 6-6 sehen Sie, wie ogrinfo unter Angabe der Option -spat Geometrien, die sich in einem bestimmten Bereich befinden, ausgeben kann.

Beispiel 6-6: Ausgabe von Objekten, die sich in einem bestimmten Bereich befinden

```
> ogrinfo data airports -spat 451869 5225734 465726 5242150
Layer name: airports
Geometry: Point
Feature Count: 2
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
Layer SRS WKT:
(unknown)
NAME: String (64.0)
LAT: Real (12.4)
LON: Real (12.4)
ELEVATION: Real (12.4)
QUADNAME: String (32.0)
OGRFeature(airports):7
  NAME (String) = Grand Rapids-Itasca County/Gordon Newstrom Field
  LAT (Real) = 47.2108
  LON (Real) = -93.5097
  ELEVATION (Real) = 1355.0000
  QUADNAME (String) = Grand Rapids
  POINT (461401 5228719)

OGRFeature(airports):8
  NAME (String) = Richter Ranch Airport
  LAT (Real) = 47.3161
  LON (Real) = -93.5914
  ELEVATION (Real) = 1340.0000
  QUADNAME (String) = Cohasset East
  POINT (455305 5240463)
```

Um Objekte basierend auf ihrer räumlichen Lage zu selektieren, wird der Parameter -spat gefolgt von zwei Koordinatenpaaren verwendet. Das erste Paar, 451869 5225734, bezeichnet die südwestliche Ecke der Fläche, während das zweite Koordinatenpaar

(465726 5242150) sich auf die nordöstliche Ecke bezieht. Auf diese Weise lässt sich über die Angabe von zwei Stützpunkten ein rechteckiges Abfragegebiet definieren.



Ein so definierter Bereich wird als *Bounding Box* (Umgebungsrechteck) bezeichnet, wobei das erste Koordinatenpaar die linke untere Ecke und das zweite Paar die rechte oberere Ecke angibt.

ogrinfo gibt nur die Objekte aus, die sich innerhalb der Bounding Box befinden. Da die Daten in UTM vorliegen, müssen die über -spat angegebenen Koordinaten ebenfalls im UTM-Format spezifiziert werden. Die Koordinaten müssen immer in den Einheiten und der Projektion der Quelldaten angegeben werden, ansonsten erhalten Sie falsche Ergebnisse.

In Beispiel 6-7 wird über -sql eine komplexe SQL-Abfrage übergeben.

Beispiel 6-7: Einsatz von ogrinfo mit komplexer SQL-Abfrage

```
> ogrinfo data airports -  
sql "select * from airports where elevation > 1350 and quadname like '%Lake'" -summary  
layer names ignored  
INFO: Open of 'data'  
using driver 'ESRI Shapefile' successful.  
layer names ignored in combination with -sql.
```

```
Layer name: airports  
Geometry: Point  
Feature Count: 5  
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
```

Wird beim Aufruf die Option -summary verwendet, besteht die Ausgabe lediglich aus der Zusammenfassung. Über Feature Count wird die Anzahl der Ergebnisse ausgegeben. Dies ist sehr praktisch, wenn Sie lediglich wissen möchten, wie viele Objekte bestimmte Bedingungen erfüllen, während Sie die übrigen Details nicht interessieren.

gdalinfo zur Ausgabe von Informationen zu Rasterdaten

Ein Beispiel für ein Satellitenbild finden Sie unter http://geogratis.cgdi.gc.ca/download/RADARSAT/mosaic/canada_mosaic_lcc_1000m.zip. Nach dem Entpacken erhalten Sie eine Datei namens *canada_mosaic_lcc_1000m.tif*. Diese Datei enthält ein vom Satelliten RADARSAT erstelltes Bild. Mehr Informationen zu RADARSAT finden Sie unter der Adresse http://ccrs.nrcan.gc.ca/radar/spaceborne/radarsat1/specs/index_e.php.

Mithilfe des Tools gdalinfo können die Daten genauer untersucht werden. Wie ogrinfo für Vektordaten kann gdalinfo bestimmte Informationen über eine Rasterdatei ausgeben. An Beispiel 6-8 können Sie sehen, dass die Ausgaben von gdalinfo und ogrinfo recht ähnlich sind. Vor dem Ausführen von gdalinfo sollten Sie in das Verzeichnis wechseln, in dem sich die Rasterdatei befindet.

Beispiel 6-8: gdalinfo zur Anzeige von Informationen zu einer Rasterdatei

```
> gdalinfo canada_mosaic_lcc_1000m.tif
Driver: GTiff/GeoTIFF
Size is 5700, 4800
Coordinate System is:
PROJCS["LCC",
    GEOGCS["NAD83",
        DATUM["North_American_Datum_1983",
            SPHEROID["GRS_1980",6378137,298.2572221010042,
                AUTHORITY["EPSG","7019"]],
            AUTHORITY["EPSG","6269"]],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433],
        AUTHORITY["EPSG","4269"]],
    PROJECTION["Lambert_Conformal_Conic_2SP"],
    PARAMETER["standard_parallel_1",49],
    PARAMETER["standard_parallel_2",77],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",-95],
    PARAMETER["false_easting",0],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]]]]
Origin = (-2600000.000000,10500000.000000)
Pixel Size = (1000.00000000,-1000.00000000)
Corner Coordinates:
Upper Left  (-2600000.000,10500000.000) (177d17'32.31"W, 66d54'22.82"N)
Lower Left  (-2600000.000, 5700000.000) (122d54'49.00"W, 36d12'53.87"N)
Upper Right  ( 3100000.000,10500000.000) ( 9d58'39.57"W, 62d25'50.45"N)
Lower Right  ( 3100000.000, 5700000.000) ( 62d32'49.65"W, 34d18'5.61"N)
Center       ( 250000.000, 8100000.000) ( 89d56'43.00"W, 62d46'47.18"N)
Band 1 Block=5700x1 Type=Byte, ColorInterp=Gray
```

Dieser Bericht ist in fünf Abschnitte unterteilt. Anders als bei ogrinfo gibt es nur wenige Optionen für den Aufruf von gdalinfo.

Die erste Zeile gibt das Bildformat der Datei aus.

```
Driver: GTiff/GeoTIFF
```

In unserem Fall handelt es sich um eine Rasterdatei im GeoTIFF-Format. TIFF-Bilder werden häufig in der digitalen Fotografie und im Druck verwendet. Das GeoTIFF erweitert das eigentliche TIFF-Format um zusätzliche räumliche Informationen. Wenn Sie gdalinfo mit der Option --formats ausführen, erhalten Sie eine Liste sämtlicher Rasterformate, die gdalinfo lesen kann und die möglicherweise auch durch die GDAL-Bibliothek geschrieben werden können. Die FWTools beiliegende GDAL-Version unterstützt über 60 Formate. Hierzu gehören auch mehrere proprietäre sowie eine Reihe besonderer Formate, die für das Speichern von Satellitendaten verwendet werden.

Über Size wird die Bildgröße in Pixeln ausgegeben:

```
Size is 5700, 4800.
```

Die Bildgröße ergibt sich aus der Anzahl der Zeilen und Spalten. Daher wird hier auch von Rasterdaten gesprochen. Das Raster baut sich aus einer Vielzahl von Zeilen auf, die aus zusammenhängenden Quadraten bestehen, auch bezeichnet als Zellen oder Pixel. Unser Bild verfügt über 5.700 Spalten und 4.800 Zeilen. Der erste Wert in der Größenangabe gibt in der Regel die Breite, also die Anzahl der Spalten an, der zweite Werte die Höhe, also die Anzahl der Zeilen. Die Aufzählung der Zeilen und Spalten beginnt normalerweise in der linken oberen Ecke des Bilds und endet in der rechten unteren Ecke. Die Angabe 0, 0 bezieht sich demnach auf die Zelle in der linken oberen Ecke, und die Zelle 5700, 4800 steht für die Zelle der rechten unteren Ecke.

Die Bilder können in verschiedene Koordinatenreferenzsystemen projiziert werden (mehr zu den verschiedenen Projektionen finden Sie in Anhang A):

```
Coordinate System is:
PROJCS["LCC", E008",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS_1980",6378137,298.2572221010042,
        AUTHORITY["EPSG","7019"]],
      AUTHORITY["EPSG","6269"]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4269"]],
  PROJECTION["Lambert_Conformal_Conic_2SP"],
  PARAMETER["standard_parallel_1",49],
  PARAMETER["standard_parallel_2",77],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",-95],
  PARAMETER["false_easting",0],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]]]]
```

Diese Angaben weisen einer Zelle eine Koordinate zu. Oftmals müssen diese Koordinaten angepasst werden, um das Erscheinungsbild in einer bestimmten Anwendung zu verbessern oder um sie mit anderen Daten in Deckung zu bringen. Nach den obigen Angaben liegt unser Beispielbild in einer Projektion namens Lambert Conformal Conic (LCC) vor. Wenn Sie das Bild zusammen mit anderen Daten nutzen möchten, müssen Sie wissen, welche Projektion vorliegt. Stimmt die Projektion verschiedener Daten nicht überein, so müssen Sie die Daten in eine gemeinsame Projektion umwandeln oder ein Programm verwenden, das on-the-fly Projektionen durchführen kann.



MapServer ist in der Lage, Dateien/Layer on-the-fly, d.h. während der Laufzeit, zu projizieren. Das bedeutet, eine Konvertierung der Quelldaten ist nur notwendig, wenn Sie eine höhere Performance erzielen möchten.

Die Angaben zur Bezugsbreite (`latitude_of_origin`) und zum Hauptmeridian (`central_meridian`) wurden in geografischen Koordinaten in Grad (°) angegeben. Diese

Angaben beschreiben, wo sich die Koordinate 0,0 befindet. Latitude 0° entspricht dem Äquator. Der Hauptmeridian (*central_meridian*) wird über eine Angabe zur geographischen Länge (*Longitude*) definiert. Longitude -95° oder auch 95° westlicher Länge verläuft beispielsweise mitten durch Kanada.

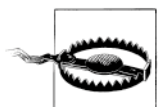
```
PARAMETER["latitude_of_origin",0],
PARAMETER["central_meridian",-95],
```

Beachten Sie bitte, dass bei den vorangegangenen Beispielen die Einheit (UNIT) der Projektionen in Metern angegeben wurde. Wenn wir im nächsten Abschnitt zum Eintrag *Pixel Size* kommen, werden Sie nur eine Zahl, aber keine Einheiten finden. Hierfür wird die weiter oben definierte Einheit (Meter) benutzt.

Den Zellen werden zum einen Zeilen- und Spaltennummern zugewiesen, gleichzeitig erhalten sie aber auch geographischen Koordinaten. *Origin* gibt die Koordinate der Zelle in Zeile 0 in der Spalte 0 an. In unserem Beispiel liegt der Wert für den Ursprung in der gleichen Projektion und den gleichen Einheiten vor wie der Wert der Projektion des gesamten Bilds. Die Ost/West-Koordinate lautet -2600000, liegt also 2.600.000 Meter westlich vom Hauptmeridian. Die Nord/Süd-Koordinate liegt 10.500.000 Meter nördlich vom Äquator:

```
Origin = (-2600000.000000,10500000.000000)
Pixel Size = (1000.00000000,-1000.00000000)
```

Die Zellen werden als Pixel bezeichnet, und jedes Pixel hat eine definierte Größe. In unserem Beispiel entspricht ein Pixel einer Fläche von 1.000×1.000 Metern. -1000 ist nur eine Notation. Das negative Vorzeichen können Sie momentan einfach ignorieren. In den meisten Fällen haben die Pixel eine quadratische Form, obwohl es auch Rasterdaten mit nicht quadratischen Pixeln geben kann. Die Einheit der Pixelgröße ist, nach der Definition des Parameters *UNIT*, Meter. Jedes Pixel hat also eine Höhe und eine Breite von einem Kilometer.



Jedes Pixel besitzt einen Koordinatenwert. Die Koordinate bezieht sich auf die linke obere Ecke des Pixels. Je nachdem, welches Gebiet ein Pixel abdeckt, kann es daher schwierig werden, seine Position genau zu bestimmen. Ein Pixel ist ein Quadrat und kein festgelegter Punkt. Aus diesem Grund bezieht sich die linke obere Ecke des Pixels auf einen anderen tatsächlichen Ort als sein Zentrum. Trotzdem haben beide die gleiche Koordinate. Die Genauigkeit von Rasterdaten hängt also von der Größe der Pixel ab.

Vergleichbar mit den Angaben zum Ursprung geben die Eckkoordinaten (*Corner Coordinates*) an, welche Koordinaten die Eckpixel und die Bildmitte haben:

```
Corner Coordinates:
Upper Left  (-2600000.000,10500000.000) (177d17'32.31"W, 66d54'22.82"N)
Lower Left  (-2600000.000, 5700000.000) (122d54'49.00"W, 36d12'53.87"N)
Upper Right ( 3100000.000,10500000.000) ( 9d58'39.57"W, 62d25'50.45"N)
Lower Right ( 3100000.000, 5700000.000) ( 62d32'49.65"W, 34d18'5.61"N)
Center      ( 250000.000, 8100000.000) ( 89d56'43.00"W, 62d46'47.18"N)
```

Die Koordinaten werden zuerst mit ihren projizierten Werten und dann mit den unprojizierten geografischen Koordinaten (Breite und Länge) angegeben. Mit diesen Informationen können Sie leicht feststellen, welchen Teil der Erde das Bild darstellt. Sollten Sie gedacht haben, das Bild zeige Griechenland, liegen Sie allerdings falsch. Die geografischen Koordinaten zeigen eindeutig, dass das Bild irgendwo im Westen liegen muss: 177d17'32.31"W, also 177 Grad, 17 Minuten und 32,31 Sekunden westlich des Nullmeridians.

Die Daten enthalten verschiedene Farbkanäle. Jeder Farbkanal speichert Daten einer bestimmten Wellenlänge des Lichts, die von einem Sensor aufgenommen wurde. In unserem Beispiel liegt nur ein Farbkanal vor: Band 1. Die Angabe ColorInterp=Gray besagt, dass es sich hier um ein *Graustufenbild* handelt. Type=Byte gibt an, dass das Bild eine Farbtiefe von 8 Bit hat (8 Bit = 1 Byte). Da 8 Bit insgesamt 256 Werte speichern können, enthält das Bild maximal 256 verschiedene Grautöne.

Band 1 Block=5700x1 Type=Byte, ColorInterp=Gray



Enthält das Bild mehr als einen Farbkanal, lassen sich durch eine Kombination der Werte Farbbilder erstellen, z.B. aus den Farbkanälen für Rot, Grün und Blau (RGB). Die meisten digitalen Fotografien sind entsprechend aufgebaut. Jeder Farbkanal kann 256 verschiedene Werte einer bestimmten Farbe annehmen. Durch ihre Kombination können ihnen, beispielsweise auf Ihrem Monitor, bestimmte RGB-Werte zugewiesen werden. Ein solches Bild wird als 24-Bit-Bild bezeichnet (8 Bit pro Farbkanal x 3 Farbkanäle).

Rufen Sie `gdalinfo` mit dem Parameter `-mm` auf, wie in Beispiel 6-9, erhalten Sie eine Zusammenfassung der minimalen und maximalen Farbwerte für die im Bild enthaltenen Farbkanäle.

Beispiel 6-9: gdalinfo unter Verwendung des Parameters -mm zur Ausgabe der minimalen und maximalen Farbwerte der Farbkanäle

```
> gdalinfo canada_mosaic_lcc_1000m.tif -mm
...
Band 1 Block=5700x1 Type=Byte, ColorInterp=Gray
  Computed Min/Max=0.000,255.000
```

Wie Sie sehen, enthält das Bild 256 verschiedene Werte (wobei 0 dem kleinsten Wert entspricht).

Werkzeuge zum Zusammenfassen der Informationen

Die Werkzeuge der GDAL/OGR-Bibliothek sind für sich genommen bereits sehr mächtig. Kombiniert mit Textverarbeitungswerkzeugen wie `sort`, `uniq`, `wc` und `sed`, erhalten sie aber noch einmal deutlich mehr Flexibilität. Die hier genannten Werkzeuge sollten auf jedem Unix-artigen Betriebssystem standardmäßig zur Verfügung stehen. Eventuell müssen Sie sich für Ihr System aber auch ein passendes Paket herunterladen (z.B. von <http://gnu.org>).

Jeder dieser Befehle ist in der Lage, die Textausgaben anderer Programme weiterzuverarbeiten. In unserem Fall verwenden wir die von `ogrinfo` auf dem Bildschirm angezeigten Informationen als Eingabe. Die Befehle erfassen die Zeilen und ermöglichen es Ihnen, beispielsweise bestimmte Teile auszufiltern, sie neu zu formatieren, Such- und Ersetzungsfunktionen auszuführen oder bestimmte Dinge zu zählen. Viele Funktionen lassen sich bereits per `ogrinfo -sql` realisieren. Die endgültige Formatierung entspricht aber nicht immer Ihren Erwartungen. In den folgenden Beispielen zeigen wir daher einige der am häufigsten angewandten Methoden, um Informationen auszufiltern und besser an die eigenen Bedürfnisse angepasste Statistiken zu erstellen.

Installation der Werkzeuge auf Nicht-GNU-Plattformen

Auch wenn diese Textverarbeitungswerkzeuge manchmal in gemeinsamen Paketen vorliegen, handelt es sich um eigenständige Projekte. Die meisten von ihnen wurden als Teil des GNU/Free Software Foundation-Projekts entwickelt und sind daher im GNU-Verzeichnis für freie Software unter <http://www.gnu.org/directory/> eingetragen. Die Zielgruppe der GNU-Werkzeuge sind die freien Betriebssysteme. Dies kann gelegentlich zu Problemen führen, wenn Sie von einem bestimmten System, z.B. Microsoft Windows, abhängig sind. Anderen Systemen liegen die Werkzeuge zwar nicht bei, die Quellen stehen aber im Internet zum Download.

Eine umfassende Sammlung dieser Werkzeuge für Windows-Plattformen sind unter <http://unxutils.sourceforge.net/> zu finden und können als ZIP-Datei heruntergeladen werden. Nach dem Entzippen und Speichern der ausführbaren Dateien (.exe) im Windows-Systemverzeichnis (z.B. `C:\Windows\System32` oder `C:\winnt\System32`) können die Programme über die Eingabeaufforderung ausgeführt werden.

Ist das gewünschte Werkzeug nicht dabei, werden Sie vielleicht im GNU-Verzeichnis (<http://gnu.org>) fündig. Dies ist ein guter Ausgangspunkt, wenn Sie nach einem bestimmten Werkzeug suchen. Hier finden Sie auch eine Homepage und weitere Informationen zu dem Programm. Schauen Sie auf der Download-Seite zuerst nach einer Binärversion für Ihr Betriebssystem. Gibt es keine, müssen Sie möglicherweise den Quellcode für das Werkzeug herunterladen und es selbst kompilieren.

Eine weitere gute Seite ist die Freshmeat-Website (<http://freshmeat.net>). Diese hilft Benutzern, bestimmte Programme oder Projekte zu finden, und erstellt außerdem tägliche Berichte dazu, welche Dinge aktualisiert wurden oder neu hinzugekommen sind. Viele der bei Freshmeat aufgelisteten Projekte werden auf Sourceforge unter <http://sourceforge.net> gehostet.

Ein häufig unter Windows verwendetes Tool ist Cygwin, das Sie unter der Adresse <http://www.cygwin.com> finden können. Cygwin ist ein Programm, mit dem auf Windows eine Linux-artige Umgebung geschaffen werden kann. Es enthält viele der zuvor erwähnten Textverarbeitungswerkzeuge. Darüber hinaus sind auch Kompilierwerkzeuge wie der GCC-Compiler enthalten.

Mac OS X verfügt bereits über viele der Textverarbeitungswerkzeuge. Auch wenn sich die Versionen leicht von den üblichen GNU-Werkzeugen unterscheiden, sind ähnliche Alternativen im Darwin-Kern (der Mac OS X zugrunde liegt) schon enthalten. Werkzeuge, die dem System noch nicht beiliegen, können aus dem GNU-Quellcode selbst kompiliert oder mit einem Paketmanager wie beispielsweise Fink nachinstalliert werden.

ogrinfo zur Ausgabe der Daten einer Shape-Datei

Die standardmäßige Ausgabe der ogrinfo-Berichte besteht aus mehreren Zeilen, die Informationen über die einzelnen Geometrien enthalten. Wie bereits gesehen, können diese Ausgaben sehr ausführlich sein. Am Anfang steht eine Zusammenfassung, gefolgt von den Abschnitten für die einzelnen Merkmale. Im Fall der Flughafendaten enthält jeder Abschnitt sieben Zeilen. In Beispiel 6-10 sehen Sie die Informationen zu zwei der zwölf Objekte.

Beispiel 6-10: ogrinfo – einfache Ausgaben der Informationen der Flughafen-Shape

```
> ogrinfo data airports
INFO: Open of 'data'
using driver 'ESRI Shapefile' successful.

Layer name: airports
Geometry: Point
Feature Count: 12
Extent: (434634.000000, 5228719.000000) - (496393.000000, 5291930.000000)
Layer SRS WKT:
(unknown)
NAME: String (64.0)
LAT: Real (12.4)
LON: Real (12.4)
ELEVATION: Real (12.4)
QUADNAME: String (32.0)
OGRFeature(airports):0
  NAME (String) = Bigfork Municipal Airport
  LAT (Real) = 47.7789
  LON (Real) = -93.6500
  ELEVATION (Real) = 1343.0000
  QUADNAME (String) = Effie
  POINT (451306 5291930)

OGRFeature(airports):1
  NAME (String) = Bolduc Seaplane Base
  LAT (Real) = 47.5975
  LON (Real) = -93.4106
  ELEVATION (Real) = 1325.0000
  QUADNAME (String) = Balsam Lake
  POINT (469137 5271647)
```


Was ist aber zu tun, wenn Sie sich gar nicht für alle Informationen interessieren? Zwar können Sie ogrinfo mit den Optionen `-sql` und `-where` verwenden, trotzdem wird immer noch die Zusammenfassung angezeigt, und die Informationen haben vielleicht nicht das benötigte Format. Hier kommen die erwähnten Textverarbeitungswerkzeuge ins Spiel, mit denen Sie die Ausgaben an Ihre eigenen Bedürfnisse anpassen können.

Verwendung von grep, um lediglich die Namen der Flughäfen auszugeben

Mithilfe des Befehls `grep` können Sie bestimmte Zeilen der Bildschirmausgabe ausfiltern, um beispielsweise eine ganz bestimmte Zeile in einer Textdatei zu finden. In unserem Fall leiten wir die Textausgabe von `ogrinfo` an `grep` weiter. `grep` analysiert die ankommenden Daten und gibt nur die gewünschten Informationen aus. In unserem Beispiel weisen wir `grep` an, nur Zeilen anzuzeigen, die mit zwei Leerzeichen beginnen und das Wort `NAME` enthalten. Die übrigen Zeilen werden nicht ausgegeben. Die Befehle werden mit dem Pipe-Zeichen (`|`) verkettet. Hierdurch wird der Kommandozeilen-Interpreter angewiesen, die Ausgaben von `ogrinfo` zur weiteren Verarbeitung an `grep` weiterzuleiten. Als Option erhält `grep` ein Muster, mit dem Sie definieren, welche Textbestandteile Sie behalten möchten, wie in Beispiel 6-11 zu sehen.

Beispiel 6-11: Nutzung von grep, um die Ausgaben von ogrinfo zu filtern

```
> ogrinfo data airports | grep '  NAME'
NAME (String) = Bigfork Municipal Airport
NAME (String) = Bolduc Seaplane Base
NAME (String) = Bowstring Municipal Airport
NAME (String) = Burns Lake Seaplane Base
NAME (String) = Christenson Point Seaplane Base
NAME (String) = Deer River Municipal Airport
NAME (String) = Gospel Ranch Airport
NAME (String) = Grand Rapids-Itasca County/Gordon Newstrom Field
NAME (String) = Richter Ranch Airport
NAME (String) = Shaughnessy Seaplane Base
NAME (String) = Sixberrys Landing Seaplane Base
NAME (String) = Snells Seaplane Base
```

Möchten Sie andere Informationen ausfiltern, ändern Sie einfach `' NAME'` (inklusive der zwei vorangestellten Leerzeichen) in `'abc'` oder andere Buchstaben und Zahlen, die Sie interessieren. Schreiben Sie beispielsweise `grep 'LAT'`, werden nur die Zeilen ausgegeben, die die Zeichen `'LAT'` enthalten. Lassen Sie bei der Angabe von `'NAME'` die Leerzeichen weg, werden übrigens auch Zeilen gefunden, die das `QUADNAME`-Attribut enthalten.

Verwendung von wc, um die Namen der Flughäfen zu zählen

Der Befehl `wc` kann eine Reihe von Analysefunktionen durchführen. Der Name `wc` steht für »word count« – Wortzähler. Das Werkzeug ermittelt die Anzahl von Buchstaben, Wörtern oder Zeilen im übergebenen Text und gibt diese zurück. Die Ausgabe von `grep` oder `ogrinfo` kann zur weiteren Analyse an `wc` weitergegeben werden.

In unserem Beispiel verwenden wir `wc` zusammen mit der Option `-l`, um die Anzahl der Zeilen zu bestimmen. Kombiniert mit dem bereits bekannten `grep`-Befehl, wird Ihnen so die Anzahl der in der Datei enthaltenen Flughäfen auf dem Bildschirm ausgegeben.

```
> ogrinfo data airports | grep ' NAME' | wc -l  
12
```

Verwendung von `sed`, um bestimmte Suchmuster im Namen der Flughäfen zu finden

Ein weiteres mächtiges Werkzeug ist das Tool `sed`. Mit `sed` können Sie den Eingabetext filtern, Such- und Ersetzungsfunktionen daran ausführen, bestimmte Teile finden oder löschen usw. Wenn Sie sich bereits mit der Syntax von regulären Ausdrücken auskennen, werden Sie mit `sed` leicht zurechtkommen. `sed` verwendet reguläre Ausdrücke zur Definition der Filter.

Im folgenden Beispiel arbeiten wir wieder mit den kompletten Ausgaben von `ogrinfo` und suchen darin nach dem Vorkommen der Wörter `Seaplane Base`. `sed` verwendet das Dollarzeichen (`$`), um das Suchmuster am Ende einer Zeile zu suchen. Dieser Filter gibt also nur solche Zeilen aus, die auf `Seaplane Base` enden. Kommen die Wörter an anderer Stelle in der Zeile vor, wird die Zeile ignoriert. Wie Sie in Beispiel 6-12 sehen können, wird der Flughafen mit dem Namen `Joese Seaplane Base and Cafe` nicht mit ausgegeben, weil der Suchtext nicht am Ende der Zeile steht.

Beispiel 6-12: Verwendung von `sed` zur einfachen Filterung der Ergebnisse

```
> ogrinfo data airports | sed -n '/Seaplane Base$/p'  
NAME (String) = Bolduc Seaplane Base  
NAME (String) = Burns Lake Seaplane Base  
NAME (String) = Christenson Point Seaplane Base  
NAME (String) = Shaughnessy Seaplane Base  
NAME (String) = Sixberrys Landing Seaplane Base  
NAME (String) = Snells Seaplane Base
```



Eine Textverarbeitung kann auch mit dem Werkzeug `awk` vorgenommen werden. Um aus den Ausgabezeilen beispielsweise den Text `NAME (String)` zu entfernen, wenden Sie `awk` wie folgt auf das Ergebnis an:

```
| awk -F= '{print $2}'
```

Formatierung der Ausgaben mit `sed`

Für einen kurzen Überblick über die Daten sind die Ausgaben der letzten Beispiele vielleicht ausreichend. Wenn dagegen ein richtiger Bericht erstellt werden soll oder bestimmte Teile gesucht und ersetzt werden müssen, ist es oft am besten, die Ergebnisse neu zu formatieren. In Beispiel 6-13 verwenden wir `grep`, um die Zeilen auszufiltern, die keine Namen von Flughäfen enthalten. Danach setzen wir zwei hintereinandergeschaltete `sed`-Filter, um den Attributnamen und schließlich sämtliche Flughäfen, die mit

einem B beginnen, aus der Ausgabe zu entfernen. Wie Sie sehen, werden die Ausgaben von ogrinfo durch drei Filter geleitet, um am Ende eine leicht lesbare Liste aller Flughäfen zu erhalten, die unseren Kriterien entsprechen.

Beispiel 6-13: sed zum Entfernen von Ergebnissen, die mit einer bestimmten Zeichenkette beginnen

```
> ogrinfo data airports | grep ' NAME' | sed 's/ NAME (String) = //' | sed '/^B/d'
Christenson Point Seaplane Base
Deer River Municipal Airport
Gospel Ranch Airport
Grand Rapids-Itasca County/Gordon Newstrom Field
Richter Ranch Airport
Shaughnessy Seaplane Base
Sixberrys Landing Seaplane Base
Snells Seaplane Base
```

Der letzte sed-Filter nutzt das Caret-Zeichen (^). Dieses Zeichen steht für den Zeilenanfang. In unserem Fall wird also nach Zeilen gesucht, die mit einem B beginnen. Der Rest der Zeile wird nicht berücksichtigt. Das abschließende /d löscht die Zeilen, die mit B beginnen.

Beispiel 6-14 verwendet einen ähnlichen Ansatz. Allerdings muss der gesuchte Text hier nicht am Zeilenanfang stehen. Alle Flughäfen, deren Namen das Wort Municipal enthalten, werden ausgefiltert.

Beispiel 6-14: sed verwenden, um Ergebnisse mit einem bestimmten Schlüsselwort zu entfernen

```
> ogrinfo data airports | grep ' NAME' | sed 's/ NAME (String) = //' | sed '/Municipal/d'
Bolduc Seaplane Base
Burns Lake Seaplane Base
Christenson Point Seaplane Base
Gospel Ranch Airport
Grand Rapids-Itasca County/Gordon Newstrom Field
Richter Ranch Airport
Shaughnessy Seaplane Base
Sixberrys Landing Seaplane Base
Snells Seaplane Base
```

Nutzung von sed zum Entfernen von Zeilen und zur Bearbeitung der Zeilenanfänge

sed besitzt eine Vielzahl verschiedener Optionen, die besonders bei der Kombination von mehreren Filtern schnell unübersichtlich werden können. In Beispiel 6-15 sehen Sie, wie mehrere Befehle und sed-Filter in einem Arbeitsschritt ausgeführt werden können.

Beispiel 6-15: Die Kombination von mehreren sed-Befehlen zum Ausfiltern von Zeilen mit bestimmten Kriterien

```
> ogrinfo data airports | sed -
n '/^ NAME/,/^ ELEVATION/p' | sed '/LAT/d' | sed '/LON/d' | sed 's/.....//'
Bigfork Municipal Airport
```

Beispiel 6-15: Die Kombination von mehreren sed-Befehlen zum Ausfiltern von Zeilen mit bestimmten Kriterien (Fortsetzung)

```
= 1343.0000
Bolduc Seaplane Base
= 1325.0000
Bowstring Municipal Airport
= 1372.0000
Burns Lake Seaplane Base
= 1357.0000
Christenson Point Seaplane Base
= 1372.0000
Deer River Municipal Airport
= 1311.0000
Gospel Ranch Airport
= 1394.0000
Grand Rapids-Itasca County/Gordon Newstrom Field
= 1355.0000
Richter Ranch Airport
= 1340.0000
Shaughnessy Seaplane Base
= 1300.0000
Sixberrys Landing Seaplane Base
= 1372.0000
Snells Seaplane Base
= 1351.0000
```

In diesem Beispiel werden vier sed-Filter angewendet. Der erste Filter ist vermutlich auch der komplexeste. Er hat zwei durch Komma getrennte Optionen:

```
'/^ NAME/,/^ ELEVATION/p'
```

Auch hier wird das Caret-Zeichen verwendet, das den Filter am Zeilenanfang verankert. Wir suchen nach Zeilen, die mit NAME (und ein paar Leerzeichen, die ogrinfo standardmäßig ausgibt) beginnen. Gleichzeitig wurde aber auch noch ELEVATION angegeben. Das Komma weist sed an, nach mehreren aufeinanderfolgenden Zeilen zu suchen. Gesucht werden die Zeilen zwischen der Zeile, die mit NAME beginnt, und der Zeile, die mit ELEVATION beginnt. NAME bezeichnet den Anfang, ELEVATION das Ende. Auf diese Weise können Sie mehrere Zeilen auf einmal betrachten, ohne sie einzeln auswählen zu müssen, und der Zusammenhang der Zeilen bleibt erhalten. Dies kann gerade bei Textausgaben wie der von ogrinfo wichtig sein, bei denen die Attribute bestimmter Objekte auf mehrere Zeilen verteilt sind.

```
sed '/LAT/d' | sed '/LON/d'
```

Der zweite und der dritte Filter löschen alle Zeilen, die LAT oder LON enthalten. Beachten Sie, dass diese Zeilen in der ursprünglichen Liste eigentlich zwischen NAME und ELEVATION liegen. Der Filter entfernt also einfach wieder einen Großteil der Zeilen, die vom vorigen Filter ausgegeben wurden.

```
sed 's/.....//'
```


Bei dem vierten Filter handelt es sich um einen Ersetzungsfiler, wie am vorangestellten `s/` zu erkennen ist. Jeder Punkt steht für ein beliebiges Zeichen, das `sed` vom Anfang einer Zeile entfernen soll.

Das Endergebnis dieser vier Filter ist eine wesentlich besser lesbare Liste der Flughäfen, gefolgt von ihre Höhe über dem Meeresspiegel.

sort zur Erstellung einer nach Höhenangaben geordneten Liste

Ein weiteres praktisches Kommandozeilenwerkzeug ist `sort`. `sort` macht genau das, was sein Name verspricht: Es bringt Text und Zahlen in eine bestimmte Reihenfolge. Standardmäßig sortiert es in aufsteigender Reihenfolge vom niedrigsten (möglichst nah an »a«) zum höchsten (möglichst nah an »z«) Wert oder Buchstaben.

In Beispiel 6-16 betrachten wir nur die Zeilen, die das Wort `ELEVATION` enthalten. Unnötige Zeichen am Anfang der Zeile werden ausgefiltert. Die Ausgabe wird schließlich mit `sort` sortiert, wodurch die Zeilen mit aufsteigenden Höhenangaben ausgegeben werden.

Beispiel 6-16: Die Verwendung von `sort` zum Ordnen der Ergebnisse

```
> ogrinfo data airports | grep 'ELEVATION' | sed -n 's/ ELEVATION (Real) = //p' | sort
1300.0000
1311.0000
1325.0000
1340.0000
1343.0000
1351.0000
1355.0000
1357.0000
1372.0000
1372.0000
1372.0000
1394.0000
```

uniq zum Entfernen doppelter Zeilen

Bei der Ausgabe von `sort` entstehen einige doppelte Werte. Offenbar gibt es mehrere Flughäfen, die sich auf einer Höhe von 1.372 Fuß über dem Meer befinden. Wenn diese Ausgaben in einem Bericht verwendet werden sollen, sind doppelte Werte unter Umständen unerwünscht.

In Beispiel 6-17 werden die Ergebnisse der verketteten Befehle `grep`, `sed` und `sort` an den Befehl `uniq` weitergereicht. `uniq` entfernt die Duplikate aus der Liste, wodurch der Wert 1372 in den Ergebnissen nur noch einmal auftaucht.

Beispiel 6-17: Die Verwendung von `uniq` zum Löschen doppelter Werte

```
> ogrinfo data airports | grep 'ELEVATION' | sed -
n 's/ ELEVATION (Real) = //p' | sort | uniq
1300.0000
```

Beispiel 6-17: Die Verwendung von uniq zum Löschen doppelter Werte (Fortsetzung)

```
1311.0000
1325.0000
1340.0000
1343.0000
1351.0000
1355.0000
1357.0000
1372.0000
1394.0000
```

uniq besitzt eine Reihe von Optionen. Wie Sie in Beispiel 6-18 sehen können, weist die Option -c uniq an, die Anzahl der Vorkommen einer Zeile mit auszugeben. Hierdurch lässt sich leicht feststellen, dass die Höhenangabe 1372 mehr als einmal in der ursprünglichen Liste vorkommt.

Beispiel 6-18: Ermittlung der Häufigkeit bestimmter Werte mit uniq -c

```
> ogrinfo data airports | grep 'ELEVATION' | sed -
n 's/ ELEVATION (Real) = //p' | sort | uniq -c
  1 1300.0000
  1 1311.0000
  1 1325.0000
  1 1340.0000
  1 1343.0000
  1 1351.0000
  1 1355.0000
  1 1357.0000
  3 1372.0000
  1 1394.0000
```

Die Option -d weist uniq an, nur Einträge auszugeben, die mehr als einmal vorkommen. Sie können mehrere Optionen kombinieren, um genau die gewünschten Ergebnisse zu erhalten. In Beispiel 6-19 suchen wir nur nach Flughäfen, die auf der gleichen Höhe liegen. Um anzuzeigen, wie viele dies jeweils sind, geben wir uniq die Option -d mit.

Beispiel 6-19: Verwendung von uniq mit dem Parameter -d zur Ausgabe von mehrfach vorkommenden Werten

```
> ogrinfo data airports | grep 'ELEVATION' | sed -
n 's/ ELEVATION (Real) = //p' | sort | uniq -cd
  3 1372.0000
```

Wollen Sie die Ausgabezeilen mit fortlaufenden Nummern versehen, können Sie den Text an den Befehl nl weitergeben (Beispiel 6-20):

Beispiel 6-20: nl zur Ausgabe des Ergebnisses mit fortlaufender Nummerierung

```
> ogrinfo data airports | grep 'ELEVATION' | sed -
n 's/ ELEVATION (Real) = //p' | sort | uniq -c | nl
  1      1 1300.0000
  2      1 1311.0000
```

Beispiel 6-20: *nl* zur Ausgabe des Ergebnisses mit fortlaufender Nummerierung (Fortsetzung)

3	1	1325.0000
4	1	1340.0000
5	1	1343.0000
6	1	1351.0000
7	1	1355.0000
8	1	1357.0000
9	3	1372.0000
10	1	1394.0000

Bei der Verwendung von `uniq` müssen Sie bedenken, dass nur die umgebenden beiden Zeilen für den Vergleich herangezogen werden. Die Sortierung sorgt dafür, dass gleiche Ergebnisse tatsächlich hintereinanderstehen. Ansonsten gibt es keine Garantie, dass `uniq` die gewünschten Ergebnisse erzielt. Wenn keine Sortierung möglich ist, sind andere Textverarbeitungswerkzeuge möglicherweise besser geeignet, beispielsweise das im folgenden Abschnitt aufgeführte `tsort`.

Weitere Textverarbeitungswerkzeuge

Bei den meisten Unix-Implementierungen, inklusive Linux, stehen noch wesentlich mehr Textverarbeitungswerkzeuge zur Verfügung. Die unten stehende Liste enthält eine Auswahl an Werkzeugen, die für Sie möglicherweise nützlich sein könnten. Wenn Sie sich über deren Einsatzmöglichkeiten nicht im Klaren sind, hilft es meistens, den Befehl mit der Option `--help` aufzurufen, um sich eine Liste der Optionen und weitere Benutzungshinweise anzeigen zu lassen. Zusätzliche Informationen finden Sie außerdem in der entsprechenden Manpage. Geben Sie hierfür auf der Kommandozeile einfach `man <Name des Befehls>` ein, und die verfügbare Dokumentation wird angezeigt.

<code>sort</code>	Textzeilen sortieren.
<code>paste</code>	Textzeilen zusammenfügen.
<code>sed</code>	Einfache Veränderungen am Texteingabestrom durchführen.
<code>tsort</code>	Topologische Sortierung.
<code>join</code>	Fügt Zeilen aus zwei Dateien zusammen.
<code>awk</code>	Programmiersprache zur Bearbeitung und Auswertung einfacher Textdaten.
<code>uniq</code>	Entfernt doppelte Zeilen aus einer sortierten Datei.
<code>head/tail</code>	Gibt den Anfang/das Ende einer Datei aus.

wc

Gibt die Anzahl von Zeilen, Wörtern und Bytes einer Datei aus.

expand/unexpand

Wandelt Tabulatorzeichen und Leerzeichen um.

grep

Gibt Zeilen aus, die auf ein bestimmtes Muster passen.

column

Liste in Spaltendarstellung ausgeben.

cut

Entfernt bestimmte Teile aus jeder Zeile.

look

Gibt Zeilen aus, die mit einer bestimmten Zeichenkette beginnen.

colrm

Entfernt bestimmte Spalten aus einer Datei.

nl

Gibt die Zeilen des Textstroms nummeriert aus.

Eine umfassende Liste mit Beispielen für die Verwendung dieser Befehle finden Sie auf der Website des Linux Documentation Project. Die Adresse lautet <http://www.tldp.org/>. Suchen Sie hier nach dem »Text Processing«-HOWTO-Dokument. Weitere Informationen finden Sie auch in der GNU-Onlinehilfe unter <http://www.gnu.org/manual/>. Hilfestellungen zu Unix finden Sie in dem O'Reilly-Buch *Unix Power Tools*.

Geodaten konvertieren

In Kapitel 2 wurden die verschiedenen Aufgaben rund um die digitale Kartografie vorgestellt. Eine dieser Aufgaben ist die Konvertierung von Geodaten in verschiedene Formate. In einer perfekten Welt gäbe es nur ein oder zwei Formate – tatsächlich existieren aber Dutzende verschiedener Möglichkeiten, geografische Informationen zu speichern. Wenn Sie dazu noch bedenken, dass dies sowohl für Vektordatenformate als auch für Rasterdatenformate gilt, kommen Sie schnell auf eine ansehnliche Zahl von Optionen. Wie Kapitel 6 wird auch dieses Kapitel Ihnen nicht automatisch zu schönen Karten verhelfen. Dennoch spielen die hier gezeigten Schritte eine Schlüsselrolle beim Erstellen sinnvoller Karten.

Die Beispiele in diesem Kapitel verwenden die in Kapitel 6 vorgestellten GDAL/OGR-Werkzeuge, die Sie über das FWTools-Paket erhalten.

Geodaten konvertieren

Durch unterschiedliche Leistungsanforderungen und den Wunsch nach individuellen Lösungen sind im Laufe der Zeit diverse verschiedene Datenformate entstanden. Es kann notwendig sein, ein neues Format zu entwickeln, das mit komplexeren Informationen umgehen oder besser und performanter verarbeitet werden kann. Aber es scheint wenig produktiv, zahlreiche Datenformate zu nutzen, nur weil jeder Anbieter ein eigenes Format für sein Programm entwickelt hat. Glücklicherweise gibt es eine Reihe ausgezeichneter Werkzeuge für die Umwandlung von Raster- und Vektordaten sowie für das Umprojizieren. Aufgrund ihrer Eigenschaften müssen Raster- und Vektordaten allerdings unterschiedlich behandelt werden. Das grundsätzliche Konzept ist aber gleich: Wandle ein Format in ein anderes um.

Für die Notwendigkeit, Daten zu konvertieren, kann es eine Reihe von Gründen geben:

- Die Quelldaten liegen in einem Format vor, mit dem Sie sich nicht auskennen.
- Ihr Programm kann nur mit einem bestimmten Format umgehen.
- Sie brauchen nur schnell ein Bild in einem gängigen Rasterdatenformat.

Vektordaten konvertieren

Nur selten haben die Daten ein Format, das gleichzeitig von Menschen gelesen und von der GIS-Software verarbeitet werden kann. Viele Datenformate verwenden binäre Formate, um die räumlichen Daten zu speichern. Binärformate können nur von Computern gelesen und von Software verarbeitet werden. Einige Geodaten liegen auch als einfache Textdateien vor, die sich daher leichter untersuchen lassen.

Geography Markup Language (GML)

Ein Beispiel für ein textbasiertes Vektorformat ist die Geography Markup Language (GML). GML verwendet ein XML-Schema, in dem die Geodaten abgebildet werden. GML ist leicht zu lesen und kann auch ohne spezielle Software bearbeitet werden – ein einfacher Texteditor reicht aus. Die manuelle Erstellung von GML-Daten ist allerdings kein Vergnügen. Glücklicherweise gibt es auch hierfür ein OGR-Werkzeug, das andere OGR-lesbare Formate nach GML umwandeln kann.

GML gibt es in drei verschiedenen Versionen: GML1, GML2 und GML3. Daneben existieren noch eine Reihe von Unterversionen. Durch die Unterschiede zwischen den Versionen kann es zu Problemen kommen, da sich bestimmte Eigenschaften möglicherweise nicht direkt übertragen lassen. Die in diesem Kapitel vorgestellten Werkzeuge verwenden `ogr2ogr`, das die Daten im GML2-Format ausgibt.

GML wurde vom OGC entwickelt, um den Austausch von Geodaten in einem gemeinsamen Format zu ermöglichen. Durch dieses Schnittstellenformat wurden viele Webservices erst möglich, die auf unterschiedlicher Software basieren und nun mithilfe dieses Standardformats untereinander kommunizieren können. Nähere Informationen dazu finden Sie in Kapitel 12.

Der Nachteil von GML ist die Dateigröße. Um die gleiche Menge an Informationen zu speichern, benötigen Textdateien mehr Platz als binäre Dateien. Dadurch sind GML-Dateien meist größer als andere Datenformate. Für eine dauerhafte Speicherung Ihrer gesamten Geodaten scheint GML daher wenig geeignet. Es ist dagegen ideal für den Datenaustausch. Da GML ein textbasiertes Format ist, lässt es sich allerdings sehr gut mit Werkzeugen wie `gzip` und `WinZip` komprimieren.

XML (Extensible Markup Language) ist ein beliebter Standard für den allgemeinen Datenaustausch, besonders über das Internet. XML-basierte Datenformate (wie z.B. GML) enthalten in der Regel einen Verweis auf ein entsprechendes Schemadokument. Im Schema wird beschrieben, wie die Daten in der XML-Datei strukturiert sind, welche Attribute sie besitzen, welche XML-Version verwendet wird, welche Informationen als Attribute definiert sind usw. Die Dateiendung für Schemadateien lautet `.xsd`. Zu einer GML-Datei mit dem Namen `airports.gml` gehört daher auch eine passende Datei `airports.xsd`, um den Inhalt von `airports.gml` zu beschreiben.

Konvertieren einer Shape-Datei nach GML

Beispiel 7-1 zeigt, wie Sie die Flughafendaten mithilfe des ogr2ogr-Befehls von Shape nach GML konvertieren können.

Beispiel 7-1: Verwendung von ogr2ogr, um eine Shape-Datei ins GML-Format zu konvertieren

```
> ogr2ogr -f "GML" airports.gml data/airports.shp airports
> more airports.gml
<?xml version="1.0" encoding="utf-8" ?>
<ogr:FeatureCollection
  xmlns:xsi="http://www.w3c.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ogr.maptools.org/airports.xsd"
  xmlns:ogr="http://ogr.maptools.org/"
  xmlns:gml="http://www.opengis.net/gml">
  <gml:boundedBy>
    <gml:Box>
      <gml:coord><gml:X>434634</gml:X><gml:Y>5228719</gml:Y></gml:coord>
      <gml:coord><gml:X>496393</gml:X><gml:Y>5291930</gml:Y></gml:coord>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <ogr:airports fid="F0">
      <ogr:geometryProperty><gml:Point>
        <gml:coordinates>451306,5291930</gml:coordinates>
      </gml:Point></ogr:geometryProperty>
      <ogr:NAME>Bigfork Municipal Airport</ogr:NAME>
      <ogr:LAT>47.7789</ogr:LAT>
      <ogr:LON>-93.6500</ogr:LON>
      <ogr:ELEVATION>1343.0000</ogr:ELEVATION>
      <ogr:QUADNAME>Effie</ogr:QUADNAME>
    </ogr:airports>
  </gml:featureMember>
  <gml:featureMember>
    <ogr:airports fid="F1">
      <ogr:geometryProperty><gml:Point>
        <gml:coordinates>469137,5271647</gml:coordinates>
      </gml:Point></ogr:geometryProperty>
      <ogr:NAME>Bolduc Seaplane Base</ogr:NAME>
      <ogr:LAT>47.5975</ogr:LAT>
      <ogr:LON>-93.4106</ogr:LON>
      <ogr:ELEVATION>1325.0000</ogr:ELEVATION>
      <ogr:QUADNAME>Balsam Lake</ogr:QUADNAME>
    </ogr:airports>
  </gml:featureMember>
  ....
```

Der oben gezeigte Befehl wandelt die Shape-Datei mit den Flughafendaten in das GML-Format um. Das GML kann verändert oder erweitert und anschließend wieder zurückkonvertiert werden. Wollen Sie die Position eines Flughafens verändern, brauchen Sie die GML-Datei nur mit einem Texteditor wie Notepad (unter Windows) oder vim (unter Unix) zu öffnen und anzupassen. Programme wie Word oder Wordpad können

zwar auch als Texteditoren verwendet werden, allerdings müssen Sie darauf achten, die Daten als einfachen Text zu speichern.



In Beispiel 7-1 wird der Befehl `more` verwendet, der den Inhalt einer Datei seitenweise auf dem Bildschirm ausgibt. So wird verhindert, dass die Ausgabe zu schnell zum Lesen über den Bildschirm scrollt. Sie sollten wissen, dass es `more` auch für viele Windows-Versionen gibt. Auf anderen Systemen wurde er mittlerweile gegen `less` ausgetauscht. In unserem Fall funktionieren beide Programme auf die gleiche Weise: Durch Drücken der Leertaste wird die nächste Bildschirmseite angezeigt. Um zur nächsten Zeile zu springen, drücken Sie die Enter-Taste. Mit Strg-B blättern Sie seitenweise zurück.

Wenn Sie sich die Datei *airports.gml* in einem Texteditor ansehen, werden Sie bemerken, dass der Inhalt hierarchisch strukturiert ist. Für jeden Flughafen gibt es einen Abschnitt, der mit `<gml:featureMember>` beginnt und mit `</gml:featureMember>` endet. In dem Attribut `<ogr:geometryProperty>` befinden sich die räumlichen Informationen. Um die Position eines Flughafens anzupassen, brauchen Sie lediglich die Koordinaten zwischen den `<gml:coordinates>`-Tags zu ändern:

```
<gml:coordinates>444049,5277360</gml:coordinates>
```

wird beispielsweise geändert in:

```
<gml:coordinates>450000,5260000</gml:coordinates>
```

Danach können Sie sich die Informationen über die GML-Datei mit `ogrinfo` anzeigen lassen oder die Daten per `ogr2ogr` wieder ins Shape-Format zurückkonvertieren. Die Konvertierung nach Shape sehen Sie im folgenden Beispiel:

```
> ogr2ogr -f "ESRI Shapefile" airports_gml.shp airports.gml airports
```

Dieser Befehl übernimmt eine GML-Datei und wandelt sie ins Shape-Format um. Die Ergebnisse können Sie mit `ogrinfo` überprüfen.

Gefilterte Ausgabe von Objekten

Gelegentlich kommt es vor, dass Sie nur einen bestimmten Teil der Projektdaten benötigen. Wenn ich beispielsweise eine Karte meines Landkreises erstellen möchte, benötige ich nur den Ort des nächstgelegenen Flughafens und nicht alle zwölf, die zur Verfügung stehen. `ogr2ogr` ist in der Lage, eine Teilmenge der Objekte auszugeben. Die Syntax ähnelt dabei sehr der von `ogrinfo`.

Speichern eines Flughafens in einer neuen Shape-Datei

Sie können `ogr2ogr` auch benutzen, um bestimmte Objekte aus einer Datei zu extrahieren. Wollen Sie beispielsweise eine Karte für den Flughafen Bigfork erstellen, rufen Sie `ogr2ogr` mit der Option `-where` auf, um eine neue Shape-Datei zu erzeugen, die nur die Daten für diesen Flugplatz enthält, wie im folgenden Beispiel zu sehen:

```
> ogr2ogr -f "ESRI Shapefile" bigfork data/airports.shp -  
where "name='Bigfork Municipal Airport'"
```

Der erste Parameter weist ogr2ogr an, die extrahierten Daten in einer neuen Shape-Datei zu speichern. Der Befehl legt daraufhin ein neues Verzeichnis mit dem Namen *bigfork* an und erzeugt darin die neue Shape-Datei namens *airports*. Rufen Sie, wie in Beispiel 7-2 gezeigt, für die neue Datei den Befehl ogrinfo auf, können Sie sehen, dass diese tatsächlich nur ein Objekt enthält: Bigfork Municipal Airport.

Beispiel 7-2: Verwendung von ogrinfo zum Anzeigen der zuvor mit ogr2ogr erzeugten Daten

```
> ogrinfo bigfork airports  
INFO: Open of 'bigfork'  
using driver 'ESRI Shapefile' successful.  
  
Layer name: airports  
Geometry: Point  
Feature Count: 1  
Extent: (451306.000000, 5291930.000000) - (451306.000000, 5291930.000000)  
Layer SRS WKT:  
(unknown)  
NAME: String (64.0)  
LAT: Real (12.4)  
LON: Real (12.4)  
ELEVATION: Real (12.4)  
QUADNAME: String (32.0)  
OGRFeature(airports):0  
  NAME (String) = Bigfork Municipal Airport  
  LAT (Real) = 47.7789  
  LON (Real) = -93.6500  
  ELEVATION (Real) = 1343.0000  
  QUADNAME (String) = Effie  
  POINT (451306 5291930)
```

Konvertieren von Shape in andere Datenformate

Die Stärke von ogr2ogr ist die Fähigkeit, eine große Anzahl von Vektorformaten lesen und schreiben zu können. Eine Liste der möglichen Formate erhalten Sie, wenn Sie ogr2ogr ohne zusätzliche Parameter aufrufen, wie in Beispiel 7-3 gezeigt. Neben einem Hilfetext werden auch die möglichen Werte für die Formatnamenoption -f ausgegeben.

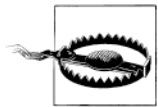
Beispiel 7-3: Eine Liste der von ogr2ogr unterstützten Ausgabeformate

```
-f format_name: output file format name, possible values are:  
-f "ESRI Shapefile"  
-f "MapInfo File"  
-f "TIGER"  
-f "S57"  
-f "DGN"  
-f "Memory"  
-f "CSV"  
-f "GML"
```

Beispiel 7-3: Eine Liste der von ogr2ogr unterstützten Ausgabeformate (Fortsetzung)

```
-f "KML"  
-f "SQLite"  
-f "ODBC"  
-f "PostgreSQL"  
-f "MySQL"
```

Das Konvertieren der Daten in andere Formate ist einfach. Sie müssen lediglich die Option `-f` für das benötigte Format anpassen und einen gültigen Namen für das Ausgabeformat vergeben. Der `-where`-Parameter braucht hierbei nicht verändert zu werden.



Welche Formate ogr2ogr unterstützt hängt davon ab, welche Formate bei der Kompilierung von ogr2ogr eingebunden wurden. Taucht das benötigte Format in der Liste nicht auf, können Sie sich eventuell eine andere Version von ogr2ogr besorgen. Versuchen Sie es beispielsweise über eine der Mailinglisten. Es gibt einige Formate, die ogr2ogr nicht schreiben kann, weil die OGR-Bibliothek manche Formate nur lesen kann. Eine vollständige Liste der Formate, die OGR lesen und schreiben kann, finden Sie unter http://www.gdal.org/ogr/ogr_formats.html.

Die Konvertierung zwischen verschiedenen Dateiformaten ist besonders einfach. Im folgenden Beispiel sehen Sie, wie eine GML-Datei angelegt wird:

```
> ogr2ogr -f "GML" bigfork_airport.gml data/airports.shp  
-where "name='Bigfork Municipal Airport'"
```

Und hier sehen Sie die Konvertierung ins DGN-Format:

```
> ogr2ogr -f "DGN" -select "" bigfork_airport.dgn data/airports.shp  
-where "name='Bigfork Municipal Airport'"
```

Das DGN-Format unterstützt allerdings keine Attributfelder. Versuchen Sie, eine Datei mit Attributen ins DGN-Format umzuwandeln, erhalten Sie die folgende Fehlermeldung:

```
ERROR 6: CreateField() not supported by this layer.
```

Die Datei wird zwar angelegt, allerdings enthält sie keine Attribute. Um die Fehlermeldung zu umgehen, können Sie die Option `-select` benutzen. Bei anderen Formaten geben Sie mithilfe von `-select` an, welche Attribute in das Zielformat überführt werden sollen. Da ogr2ogr nicht in der Lage ist, die Attribute für die DGN-Datei zu konvertieren, legen Sie fest, dass keine Felder konvertiert werden sollen, indem Sie einen leeren Wert in doppelten Anführungszeichen angeben.

Konvertieren in eine PostgreSQL-Datenbank

ogr2ogr kann Daten auch in verschiedene Datenbankformate umwandeln, wie etwa PostgreSQL (unter Verwendung der PostGIS-Erweiterung). Die Syntax für den Befehl ist komplizierter als bei einfachen dateibasierten Formaten, da zusätzliche Angaben für die

Verbindung mit der Zieldatenbank notwendig sind. Auf die einzelnen Optionen werden wir hier nicht näher eingehen. Eine detaillierte Betrachtung finden Sie in Kapitel 14.

Dieses Beispiel zeigt die Konvertierung von einer Shape-Datei in eine PostgreSQL-Datenbank:

```
> ogr2ogr -  
f "PostgreSQL" "PG:dbname=myairports host=myhost.com user=pgusername" data/airports.shp  
-where "name='Bigfork Municipal Airport'"
```

ogr2ogr baut eine Verbindung zur Datenbank myairports auf dem Server myhost.com auf. Hierbei wird in dem Beispiel der Datenbankbenutzer pgusername verwendet. Zusätzlich könnte noch der Port und das Passwort angegeben werden. In der Datenbank wird eine neue Tabelle mit dem Namen airports angelegt. Standardmäßig wird der gleiche Name wie der der Shape-Datei verwendet. Eine Abfrage der Datenbank mit dem PostgreSQL-Client psql zeigt, dass der Export erfolgreich war (siehe Beispiel 7-4).

Beispiel 7-4: Überprüfung der Ergebnisse in PostgreSQL mit dem Client psql

```
psql> select * from airports;  
-[ RECORD 1 ]+-----  
ogc_fid      | 1  
wkb_geometry | SRID=-1;POINT(451306 5291930)  
name         | Bigfork Municipal Airport  
lat          | 47.7789  
lon          | -93.6500  
elevation    | 1343.0000  
quadname     | Effie
```



Auch andere Werkzeuge sind in der Lage, Shape-Dateien in eine PostgreSQL/PostGIS-Datenbank zu überführen. Das Kommandozeilenprogramm shp2pgsql exportiert eine Shape-Datei in SQL-Anweisungen, die entweder in einer Textdatei gespeichert oder direkt zur Ausführung an eine Datenbank geschickt werden können. Dieses Hilfsprogramm ist Teil der PostGIS-Suite.

Auch das Desktop-GIS Quantum GIS (QGIS) besitzt ein Plugin (SPIT) zur Konvertierung von Shape-Dateien nach PostgreSQL/PostGIS. Weitere Informationen zu QGIS finden Sie unter <http://qgis.org> und in den Kapiteln 8 und 9.

In Kapitel 14 werden wir uns genauer mit PostgreSQL/PostGIS befassen. Dort wird der Import und Export räumlicher Daten beschrieben.

Rasterdaten konvertieren

Wie Vektordaten können auch Rasterdaten in einer Vielzahl unterschiedlicher Formate gespeichert werden. Einige Formate ergeben sich aus bestimmten Softwareanforderungen oder sind Formate für Satellitenbilder. Manche Formate sind für allgemeine Aufgaben geeignet, während andere für komplexere Anforderungen gedacht sind. In diesem

Abschnitt befassen wir uns mit `gdal_translate`, einem Werkzeug zur Konvertierung von Rasterdaten. Das Programm verfügt über eine große Zahl an möglichen Parametern, die Ihnen helfen sollen, den größtmöglichen Nutzen aus Ihren Daten zu ziehen.

Konvertieren eines Bilds in ein anderes Format

Wie `ogr2ogr` bietet auch `gdal_translate` eine Reihe von Umwandlungsoptionen. Die Syntax unterscheidet sich etwas, ist vom Konzept her aber ähnlich. Hier die Schreibweise:

```
gdal_translate <Optionen> <Eingabebild> <Ausgabebild>
```

Eine Auflistung der möglichen Optionen erhalten Sie, wenn Sie den Befehl ohne Parameter aufrufen:

Beispiel 7-5: Auflistung der Optionen von `gdal_translate`

```
> gdal_translate
Usage: gdal_translate [--help-general]
       [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/
           CInt16/CInt32/CFloat32/CFloat64}] [-not_strict]
       [-of format] [-b band] [-outsize xsize[%] ysize[%]]
       [-scale [src_min src_max [dst_min dst_max]]]
       [-srcwin xoff yoff xsize ysize] [-projwin ulx uly lrx lry]
       [-a_srs srs_def] [-a_ullr ulx uly lrx lry] [-a_nodata value]*
       [-gcp pixel line easting northing [elevation]]*
       [-mo "META-TAG=VALUE"]* [-quiet] [-sds]
       [-co "NAME=VALUE"]*
       src_dataset dst_dataset
```

GDAL 1.4.2.0, released 2007/06/27

Im ersten Teil der Ausgabe sehen Sie die verschiedenen Optionen. Alle Einträge in eckigen Klammern ([]) sind optional. Ohne weitere Angaben konvertiert `gdal_translate` Eingabebilder standardmäßig ins GeoTIFF-Format. Einige Optionen werden Sie in den Beispielen der folgenden Abschnitte oder in anderen Kapiteln dieses Buchs wiederfinden. Neben den Optionen sehen Sie in Beispiel 7-5 auch die Versionsnummer und das offizielle Release-Datum des GDAL-Projekts.

In Beispiel 7-6 erhalten Sie eine Liste der unterstützten Ausgabeformate. Je nach Betriebssystem und den bei der Kompilierung angegebenen Optionen (nur sofern Sie `gdal_translate` selbst kompiliert haben) können die Formate leicht voneinander abweichen. Diese Liste zeigt nur die möglichen Ausgabeformate, die über die Option `-of` angegeben werden können. Daneben kann das Werkzeug weitere Formate lesen, aber nicht schreiben. Um das Ausgabeformat zu wählen, geben Sie die Option `-of` gefolgt von der Kurzform des Formats (siehe Beispiel 7-6) an. Eine aktuelle Liste aller unterstützten Formate finden Sie auf der GDAL-Supportseite unter http://www.gdal.org/formats_list.html.

Beispiel 7-6: Liste der unterstützten Ausgabeformate von `gdal_translate`

The following format drivers are configured and support output:

VRT: Virtual Raster
GTiff: GeoTIFF
NITF: National Imagery Transmission Format
HFA: Erdas Imagine Images (.img)
ELAS: ELAS
AAIGrid: Arc/Info ASCII Grid
DTED: DTED Elevation Raster
PNG: Portable Network Graphics
JPEG: JPEG JFIF
MEM: In Memory Raster
GIF: Graphics Interchange Format (.gif)
FITS: Flexible Image Transport System
XPM: X11 PixMap Format
BMP: MS Windows Device Independent Bitmap
PCIDSK: PCIDSK Database File
PCRaster: PCRaster Raster File
ILWIS: ILWIS Raster Map
GMT: GMT NetCDF Grid Format
netCDF: Network Common Data Format
PNM: Portable Pixmap Format (netpbm)
ENVI: ENVI .hdr Labelled
EHdr: ESRI .hdr Labelled
PAux: PCI .aux Labelled
MFF: Atlantis MFF Raster
MFF2: Atlantis MFF2 (HKV) Raster
BT: VTP .bt (Binary Terrain) 1.3 Format
IDA: Image Data and Analysis
ECW: ERMapper Compressed Wavelets
JP2ECW: ERMapper JPEG2000
FIT: FIT Image
RMF: Raster Matrix Format
RST: Idrisi Raster A.1
USGSDem: USGS Optional ASCII DEM (and CDED)

Das einfachste Beispiel übernimmt ein Bild in einem Format und gibt es in einem anderen Format wieder aus:

```
> gdal_translate image1.png image1.tif  
> gdal_translate -of "PNG" image1.tif image1.png
```



Auch wenn Sie das Ausgabeformat festlegen, vergibt `gdal_translate` nicht automatisch eine passende Dateiendung für die Ausgabedatei. Daran müssen Sie selbst denken.

Ein Bild kann auch wieder in sein Ursprungsformat überführt werden, sofern es von GDAL als Ausgabeformat unterstützt wird.

Erzeugen des JPEG-Vorschaubilds eines Satellitenfotos mit `gdal_translate`

In Kapitel 6 haben wir `gdalinfo` verwendet, um die Detailinformationen eines RADAR-SAT-Bilds von Kanada auszugeben. Das Bild ist mit 5.700×4.800 Pixeln und etwas über 25 MByte relativ groß. Nur spezielle Bildbearbeitungswerkzeuge und Programme wie die `gdal`-Werkzeuge sind in der Lage, mit solchen und noch wesentlich größeren Dateien zu arbeiten. Wenn Sie Ihre Freunde behalten wollen, schicken Sie Ihnen diese Bilder besser nicht als E-Mail-Anhang. Davon abgesehen ist nicht nur die Dateigröße ein Problem, sondern möglicherweise auch das Format.

Das Hilfsprogramm `gdal_translate` kann für viele Aufgaben eingesetzt werden, beispielsweise um die Dateigröße zu verringern und die Daten in einem für die Ausgabe zweckmäßigeren Format auszugeben. In Beispiel 7-7 verringert `gdal_translate` die Bildgröße auf 5% der ursprünglichen Größe und gibt das Ergebnis in einer JPEG-Datei aus. Eine Überprüfung mit `gdalinfo` ergibt, dass das Ausgabe-JPEG gerade noch 285×240 Pixel groß ist. Die Dateigröße ist nun kleiner als 20 KByte. So eine Datei lässt sich selbst mit dem einfachsten Bildbetrachtungsprogramm darstellen.

Beispiel 7-7: Konvertierung und Größenänderung einer GeoTIFF-Datei mit `gdal_translate`

```
> gdal_translate -of "JPEG" -outsize 5% 5% canada_mosaic_lcc_1000m.tif can_radar.jpg
> gdalinfo can_radar.jpg
Driver: JPEG/JPEG JFIF
Size is 285, 240
```

Abbildung 7-1 zeigt, wie das Bild `can_radar.jpg` aus Beispiel 7-7 aussieht.

Bereiche aus einem Satellitenbild ausschneiden und als JPEG sichern

Mit `gdal_translate` können Sie auch einen bestimmten Teil des Ursprungsbilds angeben, der ausgewählt werden soll. Das Ergebnis ist ein Ausschnitt des Originals, der in einer neuen Datei gesichert wird. Hierfür verwenden Sie die Option `-projwin`, um die Koordinaten des Ausschnitts anzugeben, oder `-srcwin`, wenn Sie wissen, welche Pixel in die neue Datei übernommen werden sollen. Mithilfe von Quantum GIS können Sie sich eine Vorschau von dem Bild anzeigen lassen, um über die linke obere und die rechte untere Koordinate den gewünschten Ausschnitt zu ermitteln. Danach führen Sie `gdal_translate` wie bisher aus. Sie geben allerdings zusätzlich die Koordinaten der Bounding Box über den Parameter `-projwin` an. Beispiel 7-8 zeigt, wie Sie das Ausschneiden eines Bereichs und eine Konvertierung in ein anderes Bildformat mit nur einem Befehl durchführen können.



Abbildung 7-1: Das ursprüngliche Satellitenbild vor dem Ausschneiden

Beispiel 7-8: Auswahl eines Bildausschnitts und Konvertierung in ein anderes Format mit `gdal_translate`

```
> gdal_translate -of "JPEG" -projwin -2281300 7464800 -
1812300 7001800 canada_mosaic_lcc_1000m.tif can_radar_clip.jpg
Input file size is 5700, 4800
Computed -srcwin 318 3035 469 463 from projected window.
```



Abhängig vom Bildbearbeitungsprogramm und der Ausgangsdatei kann es Unterschiede in der Angabe der Koordinaten geben. Bei einigen Programmen geben Sie das Umgebungsrechteck an, indem Sie zuerst die Koordinaten für die linke untere Ecke und dann für die rechte obere Ecke angeben. Im vorigen Beispiel wird dagegen zuerst die linke obere und dann die rechte untere Ecke definiert, da sich das Ursprungspixel links oben befindet. Wenn Sie die Koordinaten verwechseln, werden Sie sehr seltsame Fehler erhalten. Weitere Informationen zum Problem der Achsenreihenfolge finden Sie im OSGeo-Wiki unter http://wiki.osgeo.org/index.php/Axis_Order_Confusion.

Abbildung 7-2 zeigt den generierten Ausschnitt des Satellitenbilds.

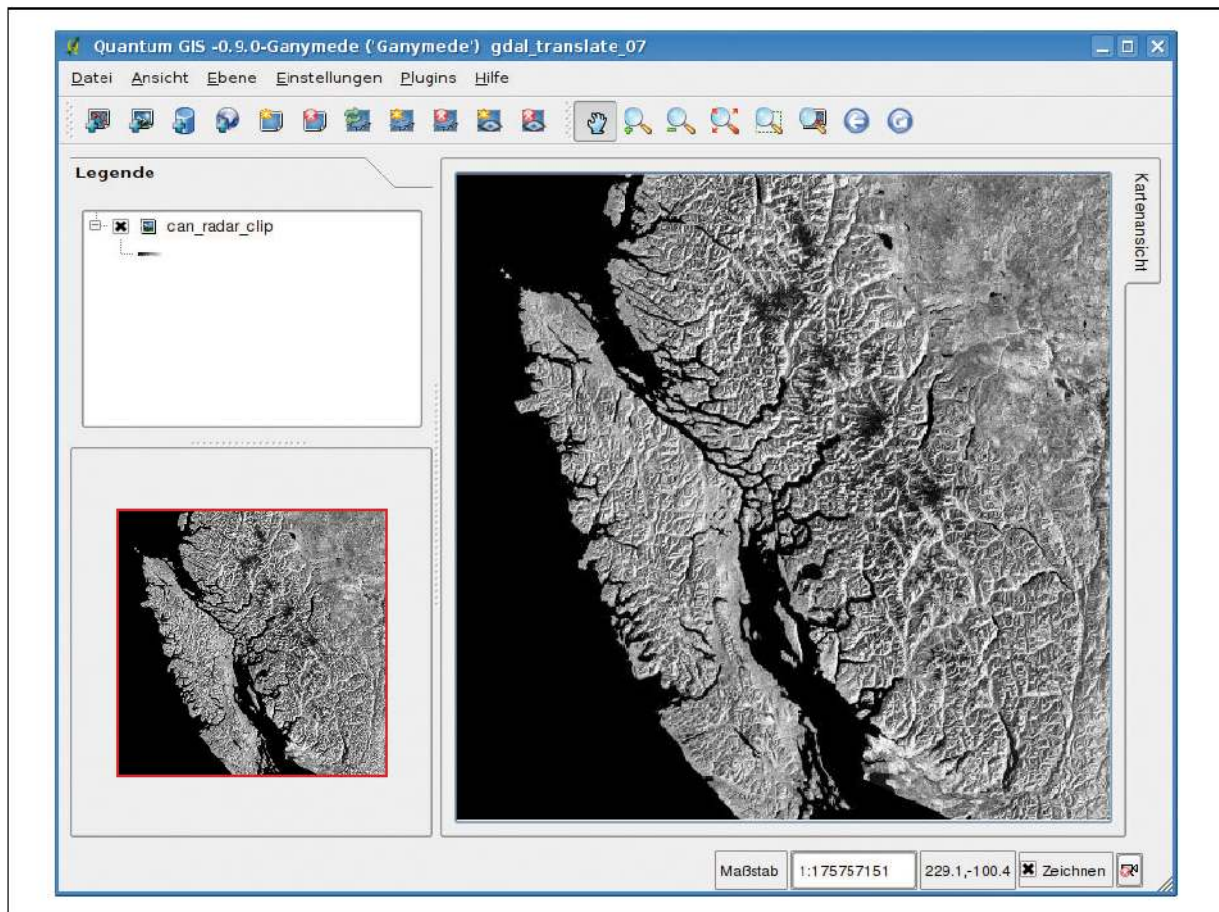


Abbildung 7-2: Bildausschnitt, der mit `gdal_translate` unter Verwendung der Option `-projwin` erzeugt wurde, angezeigt mithilfe von Quantum GIS

Geodaten in einem Desktop-GIS visualisieren

Desktop-GIS bieten einen benutzerfreundlichen Einstieg in den Umgang mit Geodaten. In diesem Kapitel lernen Sie verschiedene Desktop-GIS kennen. Am Beispiel der Software Quantum GIS werden die ersten Schritte im Umgang mit einem Desktop-GIS ausführlich erläutert. Außerdem wird die Möglichkeit aufgezeigt, mit der Software OpenEV eine einfache 3-D-Ansicht zu erstellen.

Desktop-GIS ermöglichen es einer breiten Anwenderschaft, Geodatenverarbeitung zu betreiben. Sie können auf einem PC installiert werden und sind nach einer kurzen Einarbeitung einfach zu bedienen. Die Anforderungen, die an ein Desktop-GIS gestellt werden, sind von Aufgabe zu Aufgabe unterschiedlich. Manche Anwender benötigen lediglich einen Viewer, um die Geodaten zu visualisieren und abzufragen. Andere brauchen ein Programm, mit dem sie digitalisieren und eine Geoprocessing-Funktionalität nutzen können.

Mit der Einführung von dienstbasierten Architekturen (WebGIS) ist der Kreis der GIS-Anwender explosionsartig gestiegen. Damit verliert das Desktop-GIS in seiner Rolle als reiner Viewer an Bedeutung, da hierfür webbasierte Anwendungen zur Verfügung stehen. Der Administrator eines Kartenservers benutzt aber in aller Regel weiterhin ein Desktop-GIS, um die Daten anzuschauen und aufzubereiten, bevor er sie über einen Webserver publiziert.

Viele Desktop-GIS unterstützen mittlerweile ebenfalls die Möglichkeit, serverseitige Dienste als Datenquellen zu laden, und verbinden so die Vorteile einer lokalen Installation mit der Nutzung verteilter Architekturen.

Desktop-GIS-Programme

Der Fokus dieses Kapitels liegt auf der Vorstellung von Quantum GIS. Anhand dieses Programms soll eine Einführung in die Arbeit mit einem Desktop-GIS erfolgen.

Daneben gibt es aber noch andere leistungsstarke Open Source-Desktop-GIS-Programme, die alle mehr oder weniger stark für bestimmte Einsatzzwecke optimiert wurden.

Weitere Informationen zu freien Desktop-GIS finden Sie unter:

<http://www.freegis.org>

<http://www.osgeo.org>

Open Source-Desktop-GIS-Programme

Wie finden Sie nun das Desktop-GIS, das für Ihre Fragestellung am geeignetsten ist? Die nun folgende Übersicht soll einige Open Source-Desktop-GIS kurz vorstellen, hat aber keinen Anspruch auf Vollständigkeit.

- Quantum GIS (QGIS): <http://www.qgis.org>
- JUMP Unified Mapping Platform (JUMP): <http://jump-project.org>
- User-friendly Desktop Internet GIS (uDig): <http://udig.refrations.net>
- gvSIG: <http://www.gvsig.gva.es>
- OpenEV: <http://fwtools.maptools.org>
- GRASS GIS (Geographic Resources Analysis Support System): <http://grass.itc.it>
- SAGA: <http://www.saga-gis.uni-goettingen.de/>
- Open Source Software Image Map (OSSIM) Info Sheet: <http://www.ossim.org/>
- Thuban: <http://thuban.intevation.org>
- OpenMap: <http://openmap.bbn.com>

JUMP

Die JUMP Unified Mapping Platform (JUMP) ist eine GUI-basierte Applikation für das Betrachten und Bearbeiten von räumlichen Daten. JUMP beinhaltet viele der üblichen GIS-Funktionalitäten. JUMP wurde in Java entwickelt und kann über Plugins erweitert werden. Es bestehen diverse Unterprojekte von JUMP. Dabei ist JUMP das Ursprungsprojekt, das von VividSolutions entwickelt wurde, mittlerweile aber nur noch sporadisch erweitert wird. OpenJUMP (<http://openjump.org>) hat die JUMP-Funktionalität übernommen und wird nun weiterentwickelt.

OpenJUMP verfügt über eine GIS-API und unterstützt GML, WMS sowie WFS. Weiterhin besteht die Möglichkeit, Daten aus einer PostgreSQL/PostGIS-Datenbank zu laden. OpenJUMP bietet außerdem eine Digitalisierungsfunktionalität. Als weitere Projekte der JUMP-Familie sind unter anderem deeJUMP (Schwerpunkt auf OGC-Standards) und PirolJUMP zu nennen. Eine Übersicht über die OpenJUMP-Familie finden Sie unter <http://openjump.org/wiki/show/OpenJUMPs+Family>.

uDig

Das User-friendly Desktop Internet GIS (uDig) ist eine GIS-Applikation, die als Betrachter und Editor für räumliche Daten unter spezieller Berücksichtigung der OGC-Standards WMS und WFS eingesetzt wird. Darüber hinaus unterstützt uDig SLD (OGC

Styled Layer Descriptor). uDig ist in Java implementiert und setzt auf der Bibliothek GeoTools auf.

uDig wird nicht nur als fertiges Endprodukt eingesetzt, sondern ist die Basis für die Entwicklung spezialisierter Fachapplikationen. Die Software wird von der Firma Refractions Inc. in Kanada als Open Source-Projekt entwickelt und ist unter der GNU Lesser General Public License (LGPL) verfügbar.

gvSIG

gvSIG ist ein leistungsstarkes, auf Java basierendes Open Source-Desktop-GIS. Das Programm wurde zudem als Client für GDIs konzipiert. Es können OGC-Standards (WMS, WFS und WCS), Catalog Services und Gazeteer oder anderen Remote-Dienste wie ECWP oder ArcIMS Feature und Image Services genutzt werden. gvSIG ermöglicht sowohl die herkömmliche Analyse und das Editieren von Geodaten als auch das Plotten und den Export von Karten in verschiedenen Größen und Formaten. Darüber hinaus verfügt es über die gebräuchlichsten Geoprocessing-Funktionen. Zahlreiche Vektor- und Rasterformate werden unterstützt. Georeferenzierung, Berechnung von Flächen, Arbeiten mit Tabellen sowie Beschriftungen sind mit dem Programm möglich. Außerdem kann auf Daten aus Geodatenbanken (PostgreSQL, MySQL, Oracle Spatial) lesend und schreibend zugegriffen werden. gvSIG ist unter der GNU General Public License (GPL) verfügbar

OpenEV

OpenEV ist eine Bibliothek und eine Anwendung zum Anzeigen und Analysieren von Raster- und Vektordaten. OpenEV ermöglicht die 2-D- und 3-D-Visualisierung. Es verfügt über eine Bibliothek, die in anderen Anwendungen verwendet werden kann, und wird plattformunabhängig in C und Python entwickelt. Es baut auf OpenGL, GTK+, GDAL und PROJ.4 auf. Das Projekt wurde unter der GNU LGPL License veröffentlicht. Weitere Informationen über die Möglichkeiten von OpenEV finden Sie unter <http://openev.sourceforge.net/index.php?page=features>.

GRASS

GRASS ist ein geografisches Informationssystem (GIS) mit Raster- und topologischer Vektordatenfunktionalität, Bildverarbeitung und Visualisierungsmöglichkeiten, das auf verschiedenen Betriebssystemen mit einer grafischen Benutzeroberfläche sowie optionaler Kommandozeile läuft. Das Programm umfasst über 350 Module aus allen GIS-Bereichen. GRASS bietet die Möglichkeit der 3-D-Rasterdatenanalyse. GRASS 6 ermöglicht 3-D-Vektoranalysen sowie Voxelanalyse (z.B. 3-D-Grundwasserfluss). GRASS ist unter der GNU General Public License (GPL) verfügbar.

Quantum GIS ist eine optimale Ergänzung zu GRASS, da es dessen Hilfsprogramme über eine Oberfläche auch weniger versierten Anwendern zugänglich macht. Die Kom-

bination aus Quantum GIS, GDAL/OGR und GRASS deckt die meisten Fragestellungen in der räumlichen Datenverarbeitung ab.

SAGA

SAGA – System für Automatisierte Geowissenschaftliche Analysen – ist eine hybride GIS-Software, die unter der GNU General Public License (GPL) lizenziert ist. SAGA wurde in C++ geschrieben, objektorientiert aufgebaut und enthält eine API, über die geowissenschaftliche Methoden leicht implementiert werden können. SAGA ist modular aufgebaut und verfügt derzeit über mehr als 120 Module. Das Programm unterstützt die gängigen Raster- und Vektordateiformate, es bietet die Möglichkeit, Simulationen von dynamischen Prozessen durchzuführen, Interpolation von Vektordaten, die Erstellung von Geostatistiken, Reliefanalysen und vieles mehr.

OSSIM

OSSIM ist ein Projekt der OSGeo und unter der GNU General Public License (GPL) verfügbar. Das Programm eignet sich zur Fernerkundung, Bildverarbeitung (Image Processing), zur Fotogrammetrie und 3-D-Visualisierung. OSSIM wurde in C++ implementiert und verfügt über kommandozeilenbasierte Werkzeuge und eine grafische Benutzeroberfläche.

Weitere kostenlos zur Verfügung stehende Desktop-GIS

Viele große Dienstleister stellen eigene Desktop-GIS oder Viewer kostenlos zur freien Nutzung zur Verfügung. Diese Programme können aber nicht als Freeware angesehen werden, weil Sie sie nicht frei verteilen dürfen. Ihnen steht auch nicht der Quellcode zur Verfügung, und Sie dürfen die Programme nicht modifizieren.

- ArcExplorer: <http://www.esri.com/software/arcexplorer/>
- GeoMedia Viewer: <http://imgs.intergraph.com/gviewer/>
- Geomatica FreeView: http://www.pcigeomatics.com/product_ind/freeview.html
- Spatial Commander: <http://gdv.com/download/scommander.php>

Programme, die Freewareanwendungen beinhalten:

- Gaia: <http://www.thecarbonproject.com/products/gaia.html>
- Spring: <http://www.dpi.inpe.br/spring/>
- Map Maker: <http://www.mapmaker.com/products.htm>
- TatukGIS Viewer: <http://www.tatukgis.com/Products/viewer/viewer.aspx>

Einführung in Quantum GIS (QGIS)

Quantum GIS ist ein Open Source GIS, das unter der GNU Public License (GPL) herausgegeben wird. Das Projekt wurde 2002 ins Leben gerufen und hatte das Ziel, GIS-Software kostenfrei für jeden auf einem PC über einen Datenviewer zugänglich zu machen. Dies wurde erfolgreich realisiert, und herausgekommen ist ein benutzerfreundliches Desktop-GIS mit einer einfach zu bedienenden und intuitiven grafischen Benutzeroberfläche, die zahlreiche Funktionalitäten aufweist. Das Projekt verfügt über eine große Anwender- und Entwicklergemeinschaft und macht große Schritte in der Entwicklung.

QGIS läuft unter GNU/Linux, Unix, Mac OS X und Windows. Es wurde mithilfe des Qt Toolkit (<http://www.trolltech.com>) und C++ entwickelt. QGIS unterstützt über die GDAL/OGR eine Vielzahl von Raster- und Vektorformaten. Dabei werden aber nicht alle von GDAL/OGR unterstützten Formate auch von Quantum GIS unterstützt. Quantum GIS kann Daten aus einer PostgreSQL/PostGIS-Datenbank laden sowie WMS- und WFS-Dienste einbinden. Mithilfe einer Plugin-Architektur können weitere Funktionalitäten einfach ergänzt werden. Hier ist beispielsweise GRASS GIS zu nennen, das über ein Plugin angesprochen werden kann. Des Weiteren können über Quantum GIS Rasterdaten georeferenziert werden.

Für die Weiterverarbeitung durch UMN MapServer bietet Quantum GIS die Möglichkeit, die Projekteinstellungen in eine UMN MapServer-Konfigurationsdatei (Mapdatei) zu übertragen. Diese kann direkt von UMN MapServer verwendet und das vorkonfigurierte Projekt kann im Web als Dienst veröffentlicht werden.

An dieser Beschreibung von Quantum GIS können Sie sehen, dass mehrere Open Source-Projekte (PROJ.4, GDAL/OGR, UMN MapServer, GRASS, PostgreSQL/PostGIS) zum Einsatz kommen und so eine leistungsstarke, praxisorientierte Architektur aufbauen.

Die Beschreibung der Funktionalität von QGIS in diesem Kapitel orientiert sich am Benutzerhandbuch von Quantum GIS, das Sie in deutscher Sprache über die QGIS-Website unter *Documentation* herunterladen können. In den nächsten Abschnitten werden Sie Quantum GIS und seine Funktionen kennenlernen. Die Oberfläche wird vorgestellt, und Sie werden den Umgang mit Vektor- und Rasterdaten im Desktop-GIS erlernen. Außerdem wird gezeigt, wie Daten aus PostgreSQL/PostGIS sowie über einen WMS geladen werden können.

Installation von Quantum GIS

Für Windows und Mac OS X steht QGIS als Standard-Installationspaket bereit. Binärpakete und aktuelle Informationen für verschiedene GNU/Linux-Distributionen gibt es auch auf der QGIS-Webseite unter <http://qgis.org>.

Für GNU/Linux stehen verschiedene Binärpakete für Debian, Ubuntu, Fedora, Mandriva und SuSE bereit. Diese .rpm- oder .deb-Pakete können einfach mit dem entsprechenden Paketmanager installiert werden.

Auf GNU/Linux- und Unix-Systemen können Sie Quantum GIS auch aus dem Quellcode kompilieren und installieren. Dies erfolgt über die folgenden drei Schritte:

```
cmake
make
sudo make install
```

Beachten Sie beim Kompilieren, dass Abhängigkeiten von Quantum GIS zu anderen Paketen bestehen (z.B. PROJ.4, SQLite3). Diese Pakete müssen vorliegen, bevor Sie Quantum GIS kompilieren können. Eine detaillierte Beschreibung des Kompilierens und aktuelle Informationen zur Installation finden Sie auf der QGIS-Website oder im Quantum GIS-Handbuch im Abschnitt *Installation*.

Beispieldaten und Start von Quantum GIS

In diesem Kapitel werden Sie mithilfe der Testdaten der WebGIS.rlp-Live-CD der Umgang mit Desktop-GIS kennenlernen. Bei den Testdaten handelt es sich um ein Orthophoto, Shape-Dateien mit ausgewählten Objekten des digitalen Landschaftsmodells, Flurkarten, Bebauungspläne sowie ein DEM für ein 100 km großes Gebiet des Landes Rheinland-Pfalz. Die Daten liegen in Gauß-Krüger-Zone 2 vor und verwenden daher den EPSG-Code 31466.

Sie können die Daten über den folgenden Link herunterladen:

<http://www.webmappingillustrated.com/beispiele>

Oder schauen Sie sich die Daten auf der CD, die dem Buch beiliegt, direkt an.

Aufbau der grafischen Benutzeroberfläche von Quantum GIS

Sie können Quantum GIS entweder über die Kommandozeile über den Befehl `qgis` oder unter Windows über das Desktop-Icon aufrufen. Unter Mac OS X befindet sich das Icon im Programmordner.

Die grafische Benutzeroberfläche setzt sich aus sechs Bereichen zusammen. Diese wurden in Abbildung 8-1 durch Nummern gekennzeichnet. Die Bereiche werden im Folgenden kurz vorgestellt.

❶ Menüleiste

Über eine hierarchische Menüleiste können verschiedene Funktionalitäten angesprochen werden. Dabei finden Sie im Untermenü *Datei* Funktionalitäten zur Projektverwaltung. Das Untermenü *Ansicht* beinhaltet Navigationsmöglichkeiten. Über das Untermenü *Ebenen* können Daten aus verschiedenen Datenquellen geladen und verwaltet werden.

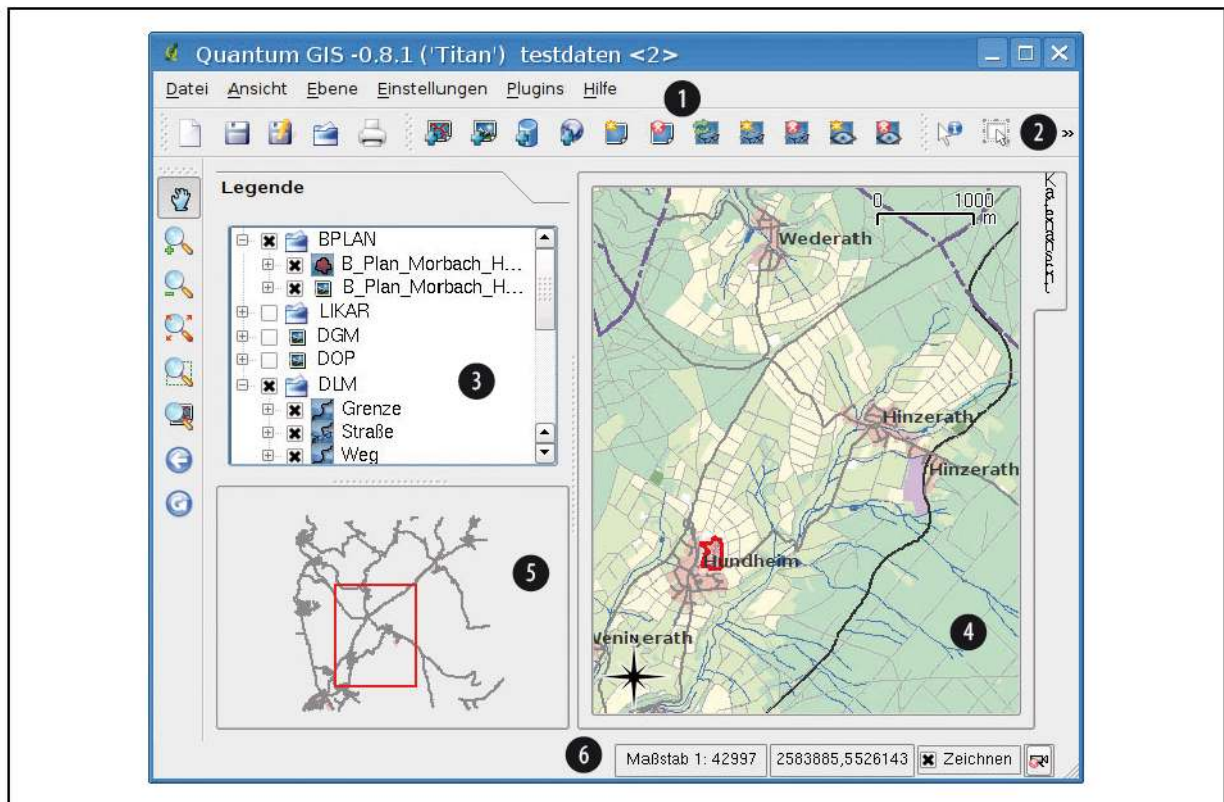


Abbildung 8-1: Die Benutzeroberfläche von Quantum GIS

tet werden. Das Untermenü *Einstellungen* stellt Möglichkeiten bereit, Quantum GIS zu konfigurieren, Projekteinstellungen vorzunehmen und eigene Koordinatenreferenzsysteme zu definieren. Über die Menüleiste *Plugins* können Sie Plugins laden und verwenden. Das letzte Untermenü *Hilfe* informiert rund um Quantum GIS und bringt Sie zur Dokumentation.

② Werkzeugleiste

Über die Werkzeugleiste können Sie die Funktionen der Menüleiste direkt ansteuern. Sie erhalten eine kurze Beschreibung der Funktion jedes Buttons, wenn Sie mit dem Mauszeiger über den Button fahren. Die Werkzeugleiste besteht aus einzelnen Blöcken, diese können nach Ihren Wünschen verschoben werden. Sollte eine Leiste nicht angezeigt werden, können Sie diese über den Menüpunkt *Ansicht* → *Werkzeugleisten* wieder einblenden.

③ Kartenlegende

In der Kartenlegende werden die Kartenebenen angezeigt und verwaltet. Eine Ebene, die in der Legende weiter oben steht, wird über einer weiter unten angeordneten Ebene im Kartenfenster angezeigt. Über die Kartenlegende können Kartenebenen aktiviert werden. Ein Klick mit der rechten Maustaste auf einen Legendeneintrag öffnet das Kontextmenü, über das die Kartenebene verwaltet und angepasst werden kann. Über das

Kontextmenü kann die Kartenebene in die Übersichtskarte überführt werden, und es kann im Kartenfenster auf die Ausdehnung der Ebene gezoomt werden.

④ Kartenfenster

Das Kartenfenster visualisiert die Daten, die Sie über die Kartenlegende ausgewählt haben. Im Kartenfenster können Sie die Navigationsmöglichkeiten der Werkzeugleiste nutzen (z.B. zoomen, Karte verschieben).

⑤ Kartenübersicht

Quantum GIS verfügt über ein Übersichtsfenster (Overview), in dem der aktuelle Darstellungsbereich des Kartenfensters über ein Rechteck angezeigt wird. Sie können das Rechteck im Übersichtsfenster mit der Maus verschieben und so den Darstellungsbereich des Kartenfensters verändern. Über die Ebenenverwaltung können Sie einzelne Ebenen der Kartenübersicht zuweisen. Dies erfolgt über das Kontextmenü der jeweiligen Ebene.

⑥ Statusleiste

Die Statusleiste gibt kurze Informationen zu den Funktionen der Buttons aus. Es wird die Koordinate des Mauszeigers angezeigt sowie der Maßstab und die Koordinaten des aktuellen Darstellungsbereichs der Karte.

Quantum GIS zur Visualisierung von Geodaten verwenden

Die folgenden Abschnitte sollen zeigen, welche Möglichkeiten Ihnen in Quantum GIS zur Visualisierung zur Verfügung stehen. Sie erfahren, welche Daten Sie laden können und welche Gestaltungsmöglichkeiten Sie haben.

Laden von Vektordaten

Wenn Sie Quantum GIS starten, öffnet sich die Benutzeroberfläche, und es wird ein leeres Kartenprojekt erzeugt, in das nun Geodaten geladen werden können.

Wir wollen mit dem Laden und der Konfiguration von Vektordaten beginnen. Vektordaten können Sie über verschiedene Wege laden, entweder über den Menüeintrag *Ebenen* → *Füge Vektorebene hinzu*, über einen Button oder über die Taste V. Es öffnet sich ein Dateiauswahlfenster, über das Sie die gewünschte Vektordatei selektieren können.

Zu Beginn wollen wir Shape-Dateien laden. Im Umgang mit Shape müssen Sie beachten, dass immer mindestens drei Dateien benötigt werden.

- *.shp enthält die Geometrien.
- *.shx enthält die Zuordnung zwischen Geometriedaten und Sachdaten.
- *.dbf enthält die Attribute im dBase-Format.

Achten Sie darauf, dass sich alle drei Dateien in Ihrem Datenverzeichnis befinden. Darüber hinaus kann noch eine Datei mit der Endung .proj vorliegen, die Hinweise zur Projektion enthält.

Navigieren Sie in das Verzeichnis /testdaten/test_dlm der Beispieldaten und wählen Sie die Datei 25805520_g_2101_p_polygon.shp aus. Diese Datei enthält Ortslagen. Über das Auswahlfeld *Dateityp* können unterschiedliche Dateiformate ausgewählt werden. An der Angabe in dieser Auswahl ([OGR] ESRI Shapefiles ...) sehen Sie, dass Quantum GIS die uns bereits bekannte OGR-Bibliothek für den Zugriff nutzt. Neben Shape werden noch weitere Vektorformate angeboten. Theoretisch könnten alle Formate, die von OGR unterstützt werden, geladen werden, Tests liegen aber nur für eine Auswahl der Formate vor. Das Laden von Vektordaten wird in Abbildung 8-2 verdeutlicht.

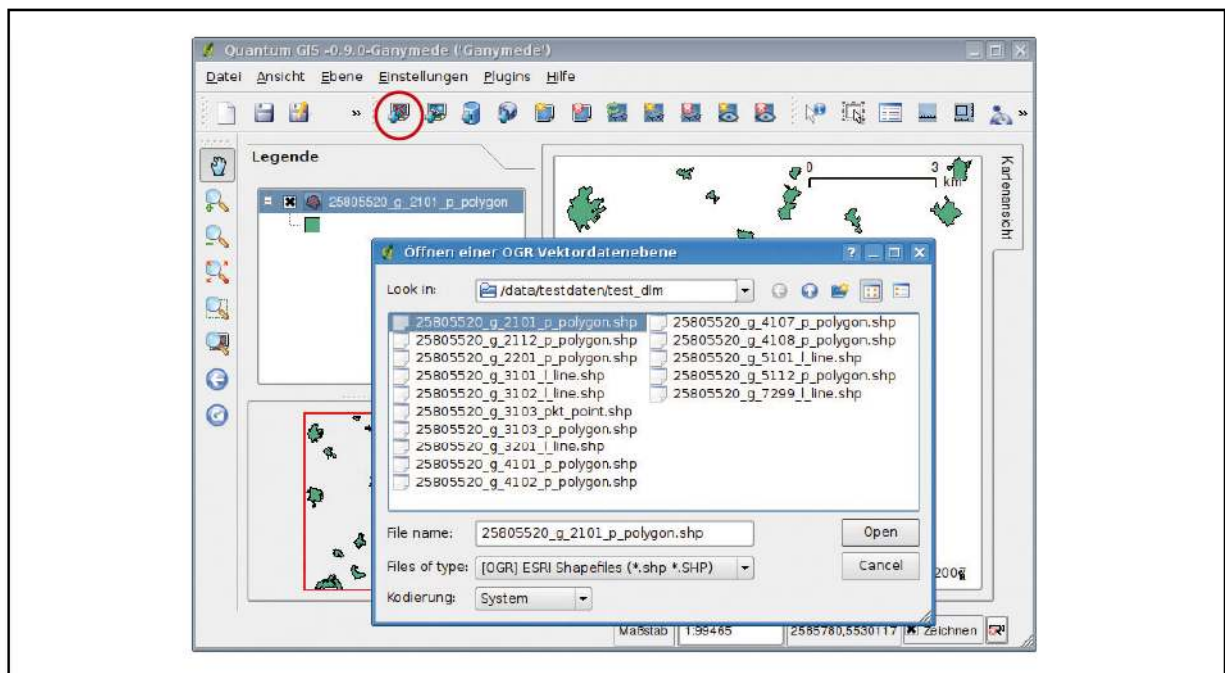


Abbildung 8-2: Laden von Vektordaten am Beispiel einer Shape-Datei

Nach der Selektion einer oder mehrerer Vektordaten lädt Quantum GIS diese und zeigt sie in der Legende und im Kartenfenster an. Dabei werden den einzelnen Ebenen zufällige Farben zugewiesen.

Beachten Sie, dass die Ebene, die in der Baumstruktur unten steht, in der Karte zuerst gezeichnet wird und von den darüber befindlichen Karten verdeckt werden kann. Die Zeichenreihenfolge der Ebenen können Sie verändern, indem Sie die Ebenen in der Legende mit der Maus anklicken und verschieben

Anpassung der Ebeneneigenschaften

Sie können die Darstellung jeder Kartenebene nun verändern. Dazu müssen Sie zu jeder Ebene das Fenster *Ebeneneigenschaften* öffnen. Dieses Fenster sehen Sie auch in Abbildung 8-3. Hier stehen über verschiedene Register Konfigurationsmöglichkeiten zur Ver-

fügung. Das Eigenschaften-Fenster öffnen Sie über einen Doppelklick auf den Ebenennamen oder über das Kontextmenü der Ebene. Es folgt eine kurze Vorstellung der einzelnen Register.

Allgemeine Ebeneinstellungen

Nicht immer gefällt Ihnen der Name der Datei, die Sie geladen haben. Über das Register *Allgemein* können Sie den Ebenennamen Ihren Wünschen anpassen (siehe Abbildung 8-3).

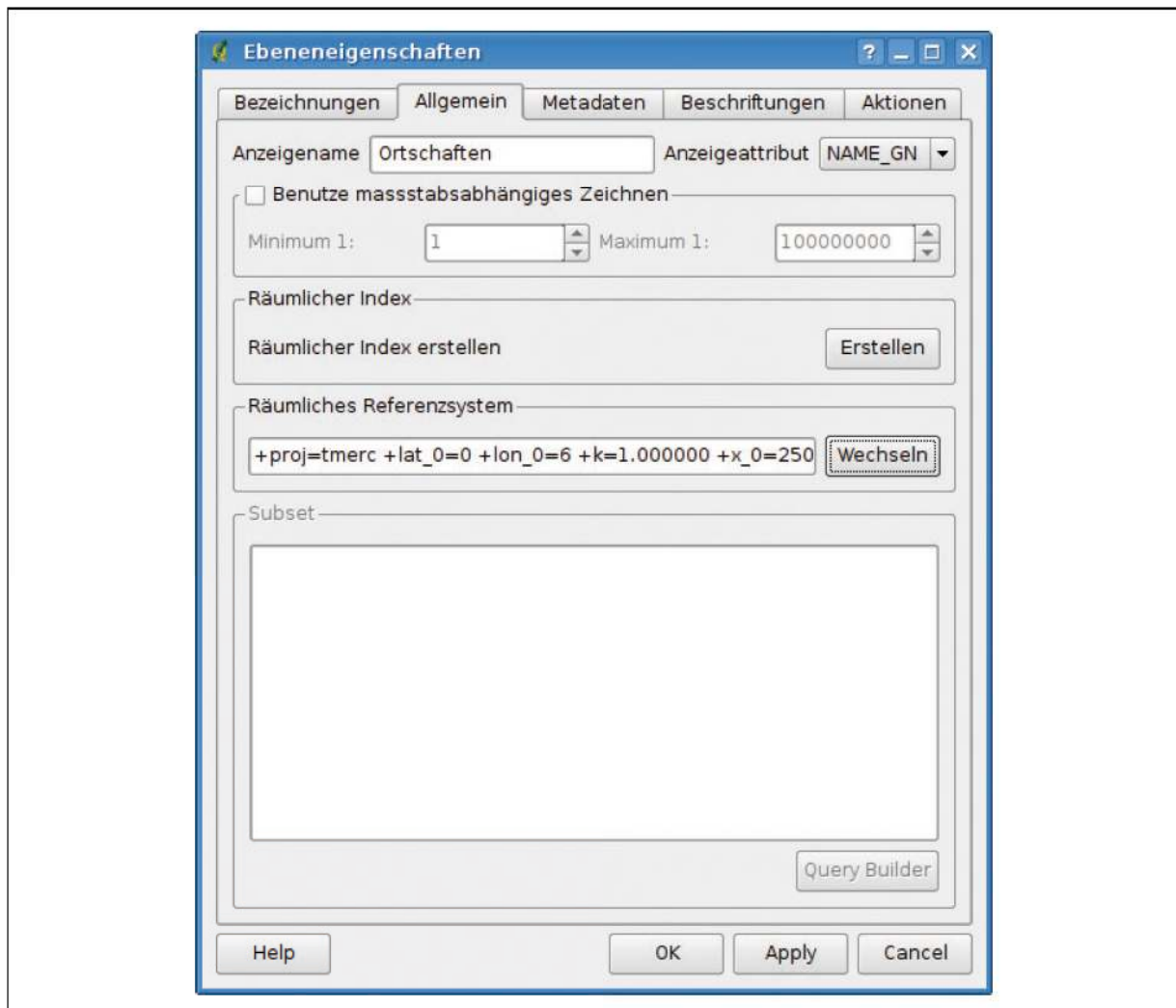


Abbildung 8-3: Anpassen der Ebeneneigenschaften

Maßstabsbereich festlegen: Wenn eine Ebene nur in einem bestimmten Maßstabsbereich angezeigt werden soll, können Sie diesen Bereich über die Auswahl *Benutze maßstabsabhängiges Zeichnen* festlegen. Hierbei müssen Sie den minimalen und maximalen Maßstab angeben, bei dem die Ebene gezeichnet werden soll.

Räumlicher Index: Für Shape-Dateien können Sie einen räumlichen Index erzeugen. Dieser Index bewirkt eine Performancesteigerung beim Zugriff auf die Daten.

Auswahl des räumlichen Referenzsystems: Der Kartenebene wird ein räumliches Referenzsystem zugewiesen. Dies ist normalerweise das Referenzsystem, das für das Projekt eingestellt wurde (zu den Projekteinstellungen später mehr). Weicht das Koordinatenreferenzsystem der Ebene von dem des Projekts ab, kann über einen Dialog das richtige Referenzsystem ausgewählt werden. Öffnen Sie dazu über den Button *Wechseln* das Fenster *Projektionsselektion*. Die Auswahl der Projektion wird in Abbildung 8-4 verdeutlicht.

Hier können Sie nun die gewünschte Projektion auf verschiedenen Wegen ermitteln. Sie können den Namen des Referenzsystems oder den EPSG-Code eingeben.

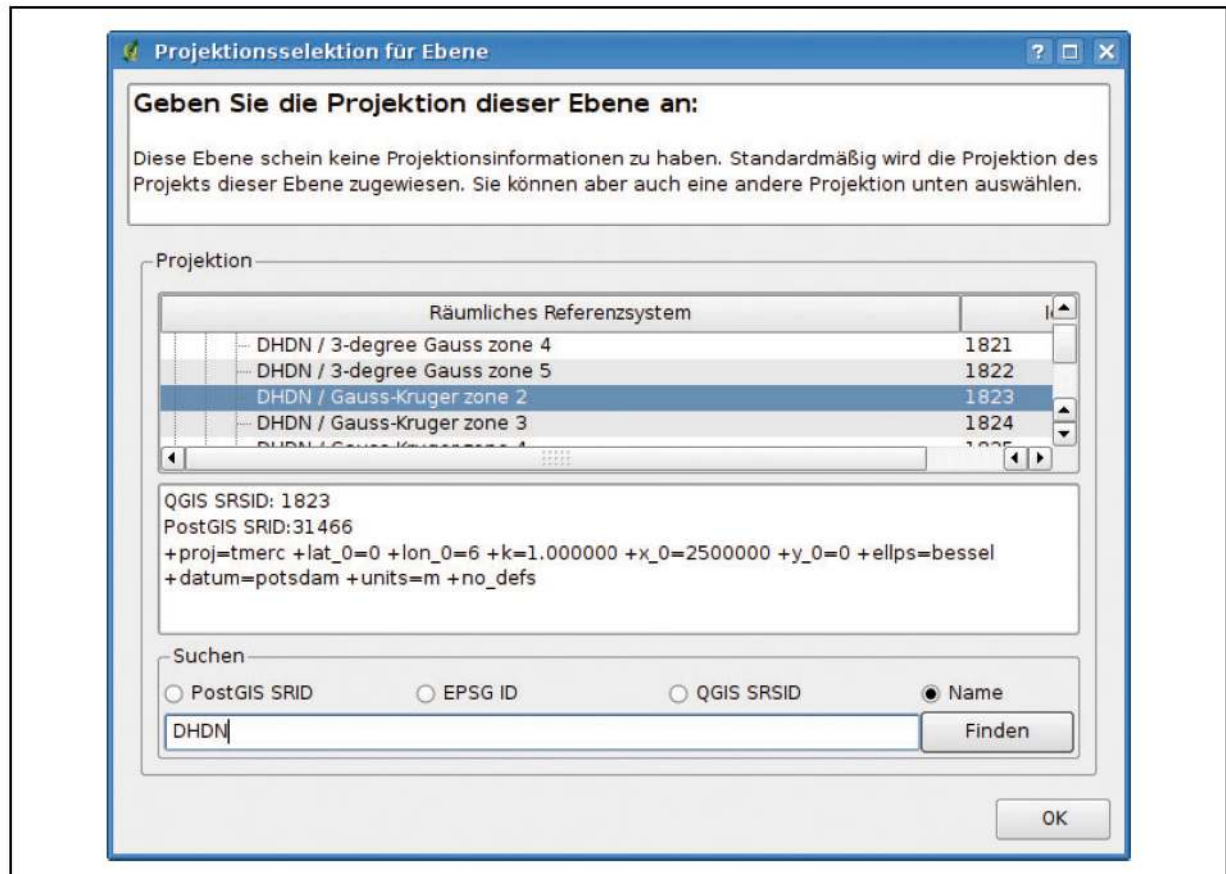


Abbildung 8-4: Auswahl eines Referenzsystems für eine Kartenebene

Metadaten: Über das Register *Metadaten* werden Informationen über die Daten ausgegeben. Diese Informationen werden Ihnen bekannt vorkommen. Sie wurden bei der Vorstellung von *ogrinfo* in Kapitel 6 eingehend behandelt.

Anpassen der Darstellung der Ebene über das Register Bezeichnungen: Über das Register *Bezeichnungen*, das Sie in Abbildung 8-5 sehen, kann die Darstellung der Geometrien angepasst werden. Hierbei können Farben, Symbole, Linienstile sowie verschiedene Füllmuster ausgewählt werden. Dabei können Sie entscheiden, ob alle Geometrien einheitlich (einfaches Symbol) dargestellt werden sollen oder ob eine Klassifizierung nach einer Attributspalte (abgestuftes Symbol) erfolgen soll.

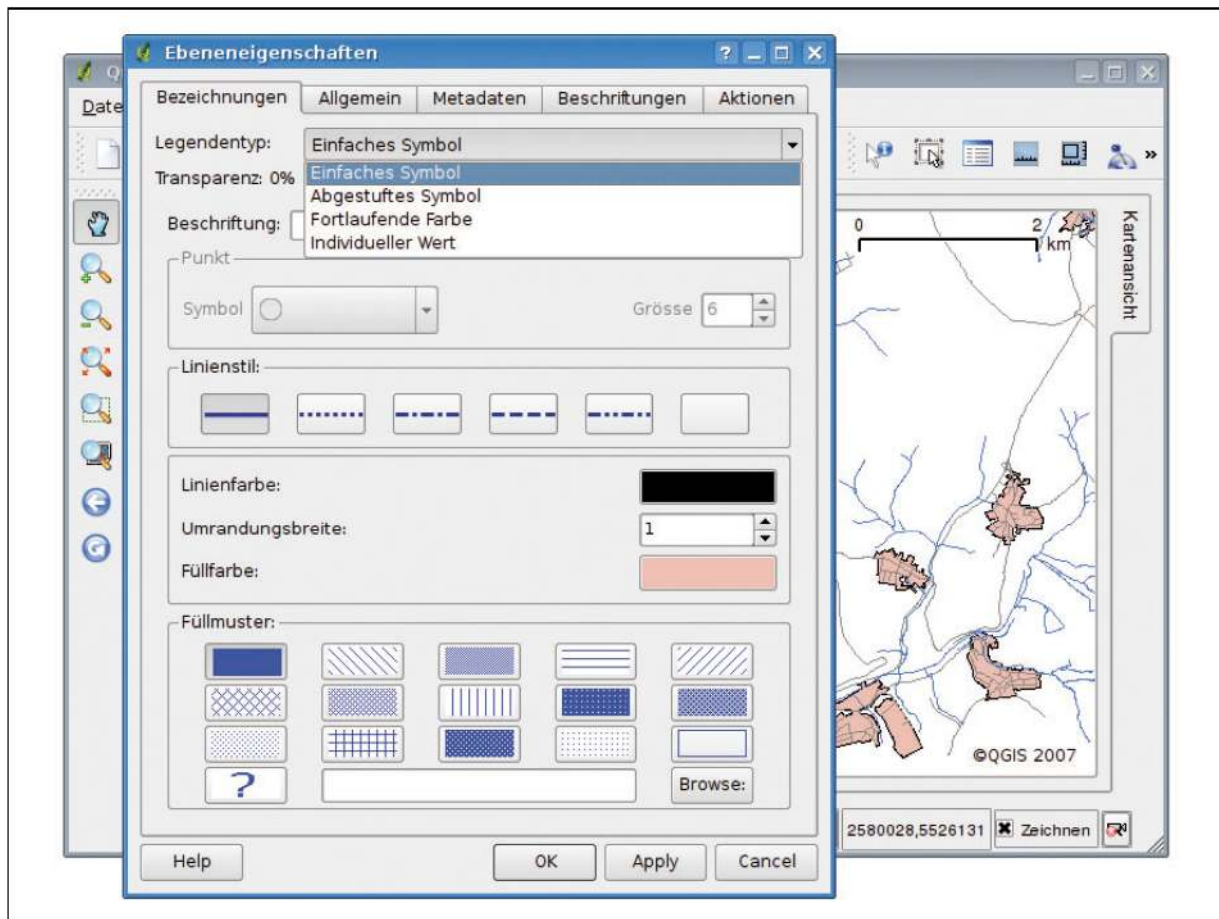


Abbildung 8-5: Anpassen der Darstellung der Objekte

Beschriftungen: Die Einstellung einer Beschriftung für die Geometrien einer Ebene erfolgt über das Register *Beschriftungen*. Hier kann entweder ein Attribut für die Beschriftung ausgewählt werden, oder Sie definieren einen festen Text, der dann an jedem Geometrieobjekt erscheint. Über die Optionen *Schriftstil*, *Schriftausrichtung*, *Puffer* und *Position* können Sie die Darstellung der Beschriftung genauer definieren. Schauen Sie sich dazu Abbildung 8-6 an.

Die Ortslagen (25805520_g_2101_p_polygon.shp) sollen mit ihren Namen in der Karte angezeigt werden. Über das Fenster *Ebeneneigenschaften* wird das Register *Beschriftungen* ausgewählt. Aktivieren Sie hier zunächst das Feld *Zeige Beschriftung* an. Nun muss das Attribut für die Beschriftung unter *Quelle* ausgewählt werden. Der Ortsname befindet sich in der Attributspalte *NAME_GN*. Wählen Sie diese aus. Unter *Schriftstil* wählen Sie eine Schriftfarbe und -art aus, außerdem kann die Schrift positioniert und gedreht werden. Damit sich die Schrift vom Hintergrund abhebt, kann es zusätzlich sinnvoll sein, eine Umrandung um den Text zu erzeugen. Dies erfolgt über *Puffer* unter Angabe einer Farbe und Puffergröße. Durch Anklicken des Button *Apply* können Sie sich die Änderungen direkt in der Karte anzeigen lassen.

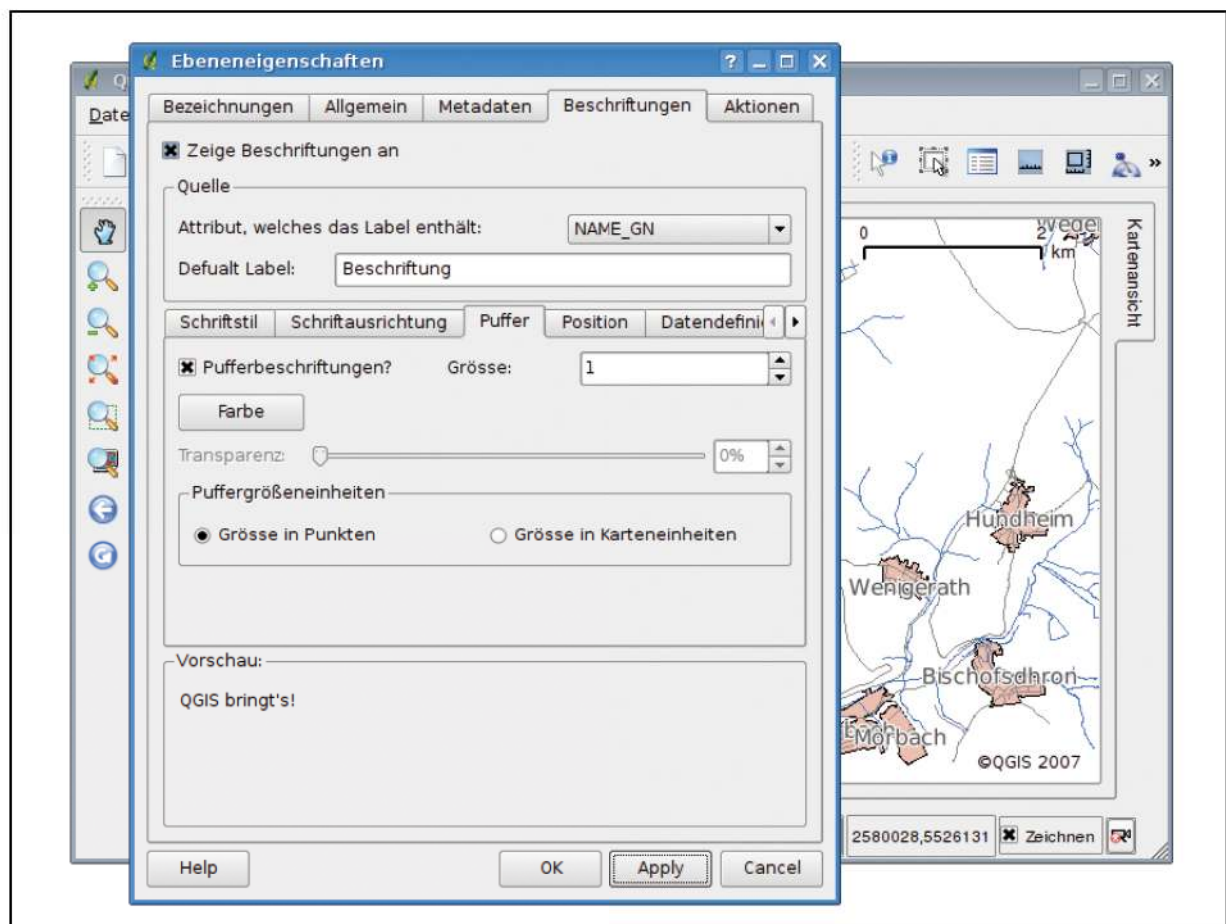


Abbildung 8-6: Beschriftung der Kartenebene Ortslagen über eine Attributspalte

Navigationsmöglichkeiten in Quantum GIS

Sie haben verschiedene Möglichkeiten, in der Karte zu navigieren. Das Untermenü *Ansicht* listet diese auf und zeigt die Kurzbefehle an, die Sie zur Navigation nutzen können. Außerdem können Sie mithilfe der Buttons in der Werkzeugleiste navigieren. Diese Buttons sehen Sie in Abbildung 8-7. Testen Sie die Möglichkeiten einfach aus.

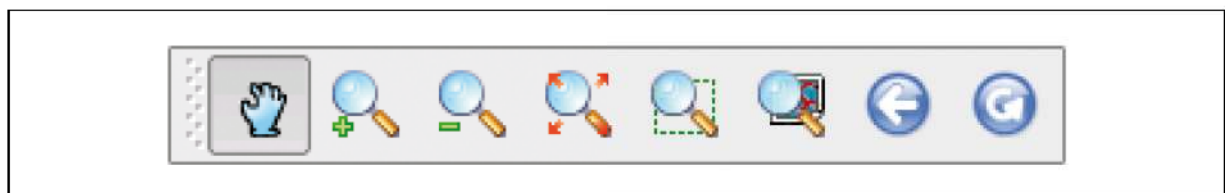


Abbildung 8-7: Navigationsleiste von Quantum GIS

Über das Kontextmenü der Ebene haben Sie zusätzlich die Möglichkeit, über den Eintrag *Auf Ebenenausschnitt zoomen* auf den Darstellungsbereich der Kartenebene zu zoomen. Dies kann z.B. hilfreich sein, wenn Sie Daten vorliegen haben, von denen Sie nicht wissen, welchen Bereich sie abdecken.

Attributinformationen ausgeben

Über eine weitere Werkzeugleiste können Sie die Attribute der einzelnen Geometrien abfragen. Nutzen Sie dazu den Button *Abfragen* und klicken Sie auf das gewünschte Objekt. In einem zusätzlichen Fenster werden die Informationen zum Objekt ausgegeben (Abbildung 8-8).

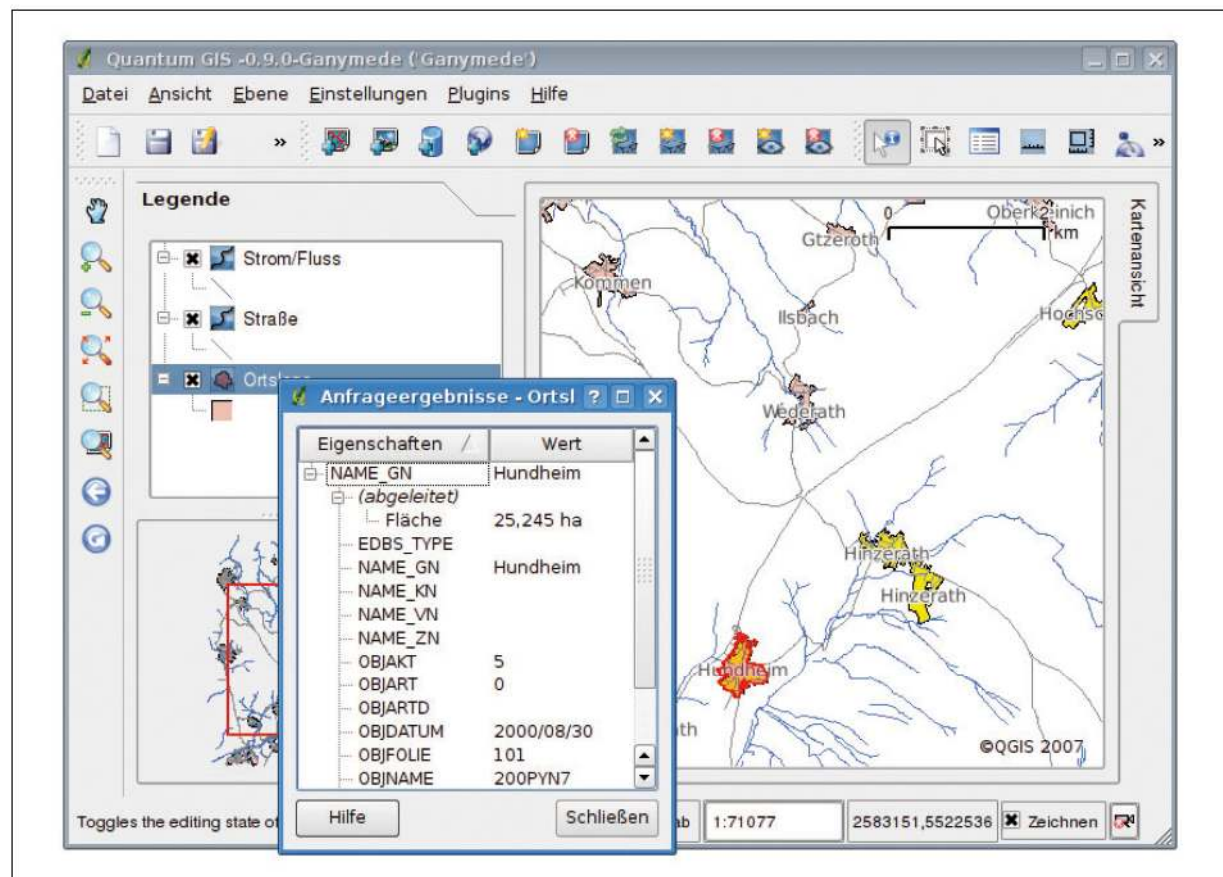


Abbildung 8-8: Ausgabe der Objektinformationen

Über den Button *Öffne Attributtabelle* erhalten Sie eine Übersicht der Attributwerte der Geometrien. In dieser Ansicht können Sie Filter anwenden und so nach Objekten suchen. Das Abfragen von Objekten wird in Abbildung 8-8 verdeutlicht. Abbildung 8-9 zeigt die Attributtabelle, über die Attribute der Ebene angezeigt und gefiltert werden können.

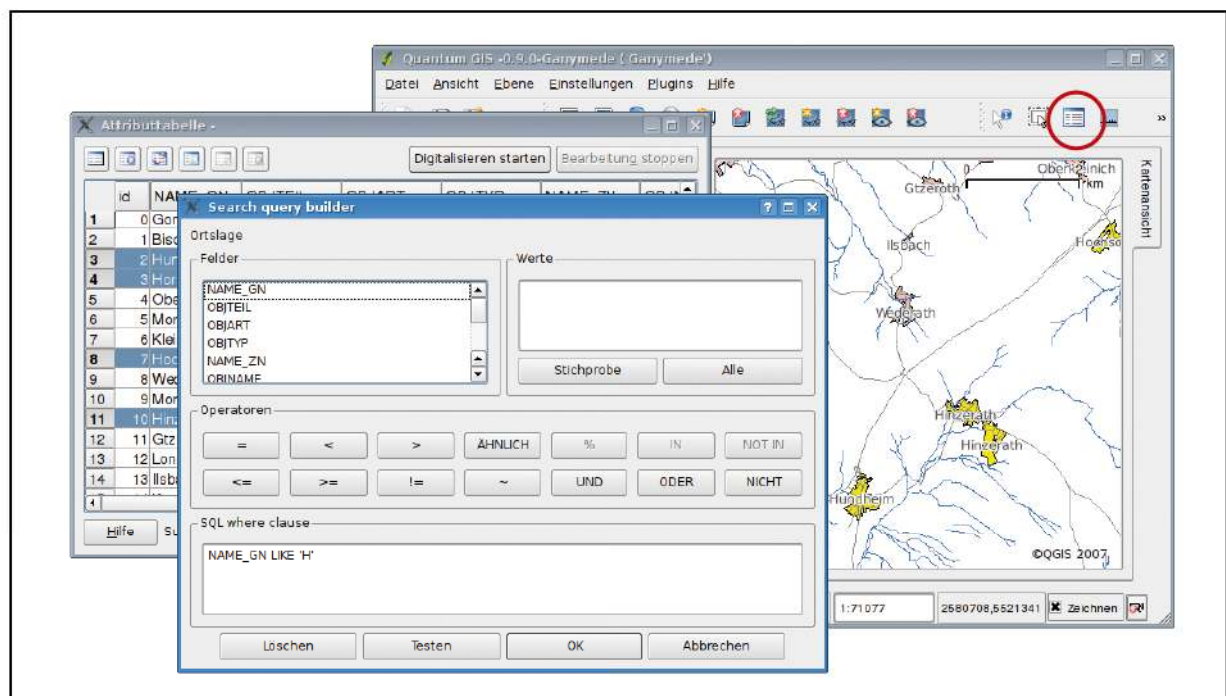


Abbildung 8-9: Filtern der Objekte über die Attributtabelle

Einbinden und Konfigurieren von Rasterdaten

Rasterdaten können Sie entweder über den Menüeintrag *Ebenen → Füge Rasterebene hinzu*, über einen Button oder über die Taste R laden. Anhand der Formatliste des Dialogfensters sehen Sie, dass diverse Rasterdatenformate unterstützt werden. Das Laden von Rasterdaten wird in Abbildung 8-10 verdeutlicht.

Als Erstes wollen wir ein Orthophoto (dop_50cm.tif) im TIF-Format laden. Dieses befindet sich im Verzeichnis /testdaten/test_dop auf der CD oder in den Beispieldaten. Nach dem Laden können Sie auch für Rasterdaten die Ebeneneigenschaften anpassen. Öffnen Sie dazu das Fenster *Rasterebeneigenschaften* über einen Doppelklick auf die Ebene oder über das Kontextmenü.

Das Register *Metadaten* gibt Informationen zur Ebene aus. Diese Informationen kommen Ihnen sicherlich bekannt vor. Wir haben sie uns in den Kapiteln 3 und 6 über die Kommandozeile mit *gdalinfo* ausgegeben lassen.

Pyramiden erzeugen

Mithilfe des Registers *Pyramiden* der Ebeneneigenschaften können Sie die Datei mit internen Übersichten versehen und so einen schnelleren Zugriff auf die Daten ermöglichen. Auch diese Funktionalität haben wir schon in Kapitel 3 beim Einsatz des GDAL-Hilfsprogramms *gdaladdo* kennengelernt.

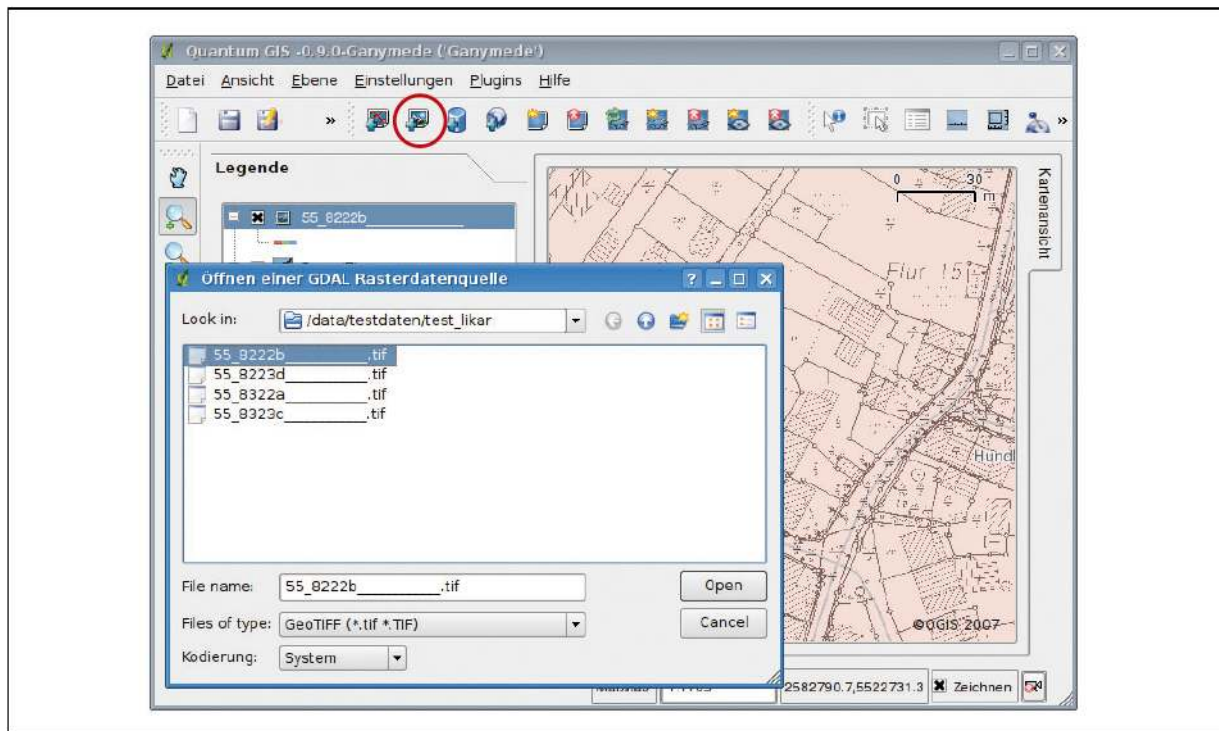


Abbildung 8-10: Einbindung von Rasterdaten in Quantum GIS

Bezeichnungen

Über das Register *Bezeichnungen* können Sie die Darstellung anpassen. Sie können den Grad der Transparenz definieren, das Bild als Graustufenbild anzeigen lassen und den Farbkanälen Farben zuweisen.

Als Nächstes wollen wir Liegenschaftskarten laden. Diese befinden sich im Verzeichnis /testdaten/testdaten_likar. Diese Daten haben eine eingeschränkte Farbpalette, jedem Pixel steht nur ein Bit für die Definition der Farbe zur Verfügung – also 0 oder 1. Damit das darunter liegende Orthophoto weiterhin sichtbar ist, sollten Sie eine Transparenz einstellen.

Daten aus der PostgreSQL/PostGIS-Datenbank laden

Quantum GIS kann Daten aus einer PostgreSQL/PostGIS-Datenbank laden und auch editieren. Das Laden der Daten aus der Datenbank wird in Abbildung 8-11 verdeutlicht. In PostGIS werden die Geodaten in Feature Tables gespeichert. Um auf diese zugreifen zu können, muss eine Verbindung zur Datenbank hergestellt werden. Hierzu bietet Quantum GIS ein Dialogfenster, über das die Verbindungsparameter eingetragen, getestet und gespeichert werden können. Ist eine Verbindung eingerichtet, können Sie sich über den Button *Verbinden* mit der Datenbank verbinden. Sie erhalten eine Übersicht der Feature Tables und können eine oder mehrere davon auswählen. Es steht Ihnen dabei der volle Zugriff auf die Attributspalten zur Verfügung. PostgreSQL/PostGIS werden Sie erst in Kapitel 14 genauer kennenlernen.

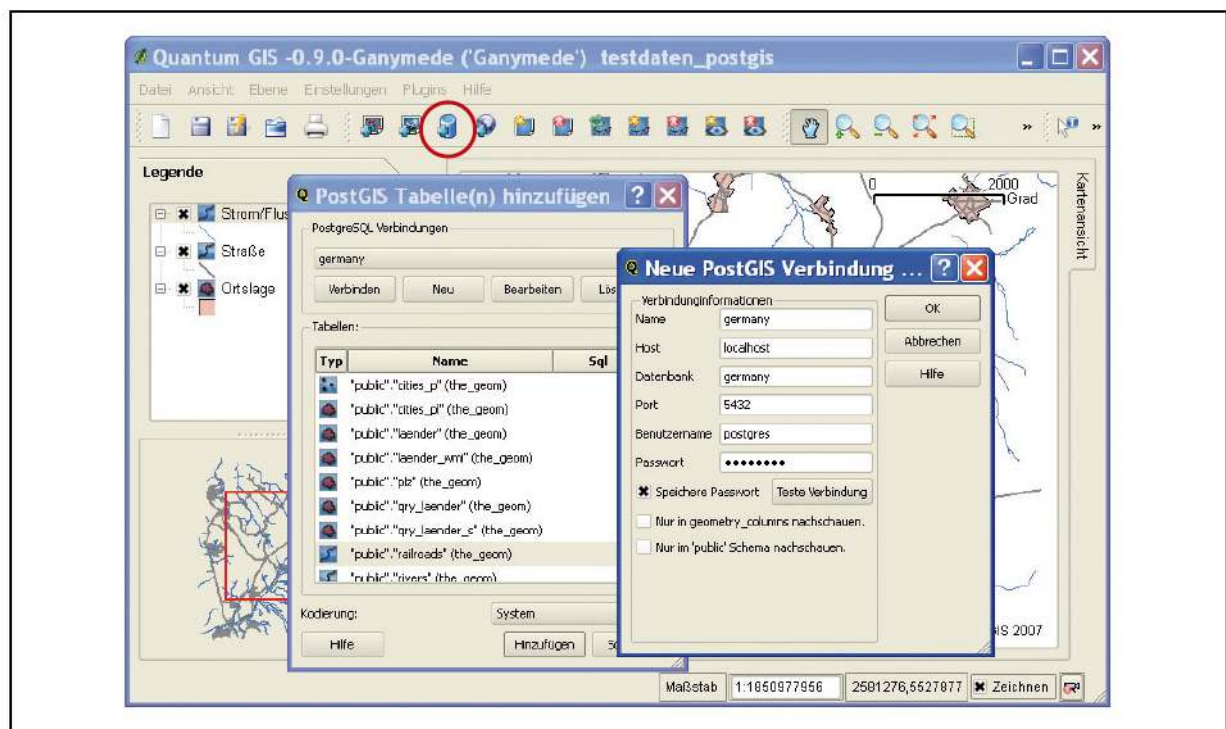


Abbildung 8-11: Laden von Geodaten aus einer PostgreSQL/PostGIS-Datenbank

OGC Web Map Service (WMS) eines entfernten Servers als Datenquelle nutzen

Mit Quantum GIS können freie OGC WMS-Dienste geladen und beispielsweise als Hintergrundkarte genutzt werden. Hierzu benötigen Sie den `getCapabilities`-Aufruf eines WMS-Diensts. Was sich dahinter verbirgt, werden Sie in den Kapiteln 12 und 13 noch erfahren.

An dieser Stelle soll das Laden eines WMS kurz an einem Beispiel veranschaulicht werden. Hierzu verwenden wir einen freien Dienst, der für das GeoPortal Rheinland-Pfalz erstellt wurde und ATKIS-RP-Daten sowie Daten aus den DLM250 und DLM1000 des Bundesamts für Kartografie und Geodäsie enthält. Diesen Dienst können Sie über die folgende URL ansprechen:

http://www.gdi-rp-dienste2.rlp.de/cgi-bin/atkis_ldi?VERSION=1.1.1&REQUEST=GetCapabilities&SERVICE=WMS

Um den WMS in Quantum GIS zu laden, wählen Sie den Button *WMS Ebene hinzufügen*, die Taste *W* oder den entsprechenden Button (siehe Abbildung 8-12) aus. Nun müssen Sie den Dienst unter *Serververbindungen* veröffentlichen und über den Button *Neu* auf den Dienst verweisen. In einem neuen Fenster müssen Sie die Verbindungsinformationen angeben. Hier müssen Sie unter *URL* den `GetCapabilities`-Request des Diensts eintragen. Sollten Sie über eine Proxyverbindung ins Internet gehen, müssen Sie auch diese angeben. Wenn Sie sich anschließend mit dem Dienst verbinden, erhalten Sie eine Liste der Layer dieses WMS. Mit der Maus können Sie die Layer markieren, die

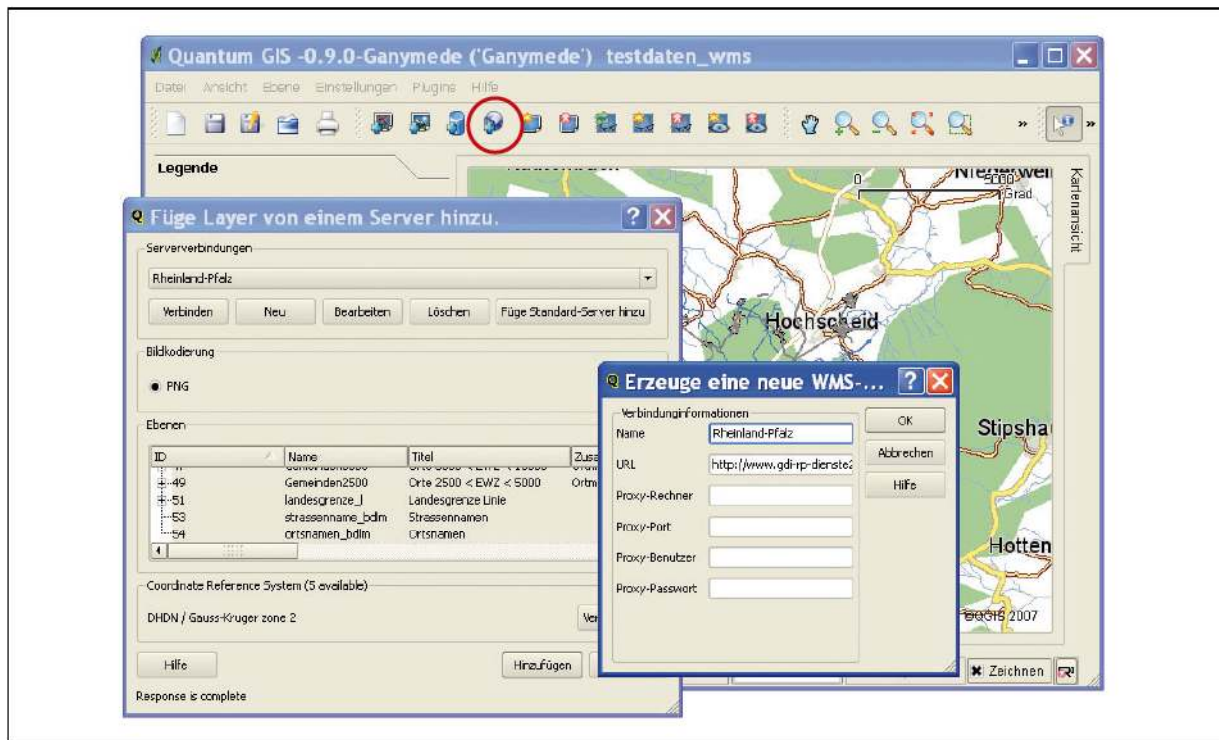


Abbildung 8-12: Laden eines OGC WMS-Diensts als Kartenebene

Sie laden möchten. Sie müssen nun noch das Koordinatenreferenzsystem auswählen, in dem die Daten angefordert werden sollen. Schauen Sie sich hierzu Abbildung 8-12 an, die das Laden eines WMS verdeutlicht.

In Abbildung 8-13 sehen Sie das Ergebnis. Ihr Projekt wurde um Informationen erweitert, die von einem entfernten Server stammen. Schauen Sie über die Eigenschaften der Kartenebene die Metadaten des Diensts an. Diese liefern eine Beschreibung zu diesem Dienst. WMS bietet den großen Vorteil, dass Daten plattformunabhängig und netzübergreifend anderen Nutzern zur Verfügung stehen. Sie werden in Kapitel 12 noch lernen, was sich genau hinter WMS verbirgt und wie Sie selbst einen WMS-Dienst aufsetzen können.

Das Projekt im Druck ausgeben

Quantum GIS bietet die Kartenausgabe als SVG (Scalable Vector Graphic), Bilddatei oder PDF-Dokument. Dabei können Sie sich das Drucklayout über den PrintComposer selbst zusammenstellen. Hier können Sie Karten, Legende, Maßstabsleiste, Texte und eigene Bilder in den Druck einbinden. Die Konfiguration des Drucklayouts sehen Sie in Abbildung 8-14.

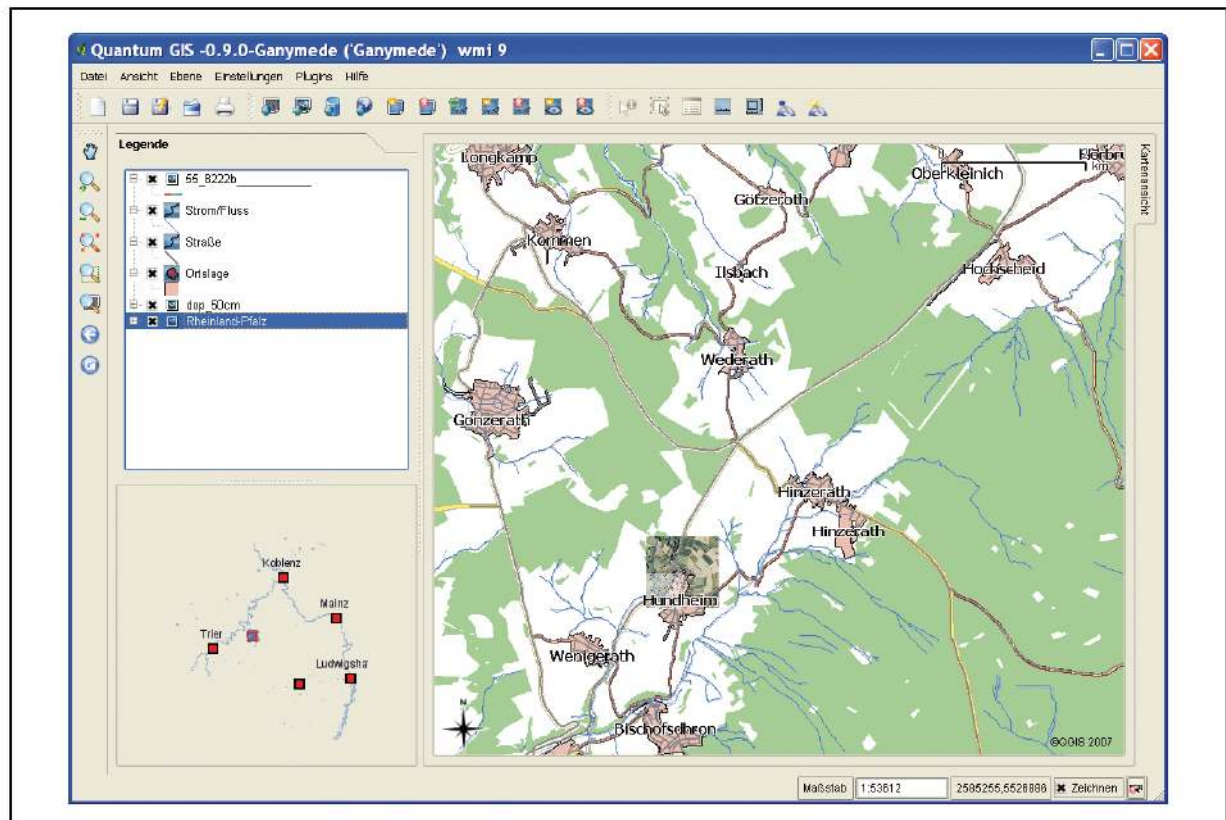


Abbildung 8-13: Der WMS liefert in diesem Projekt die Hintergrundkarte

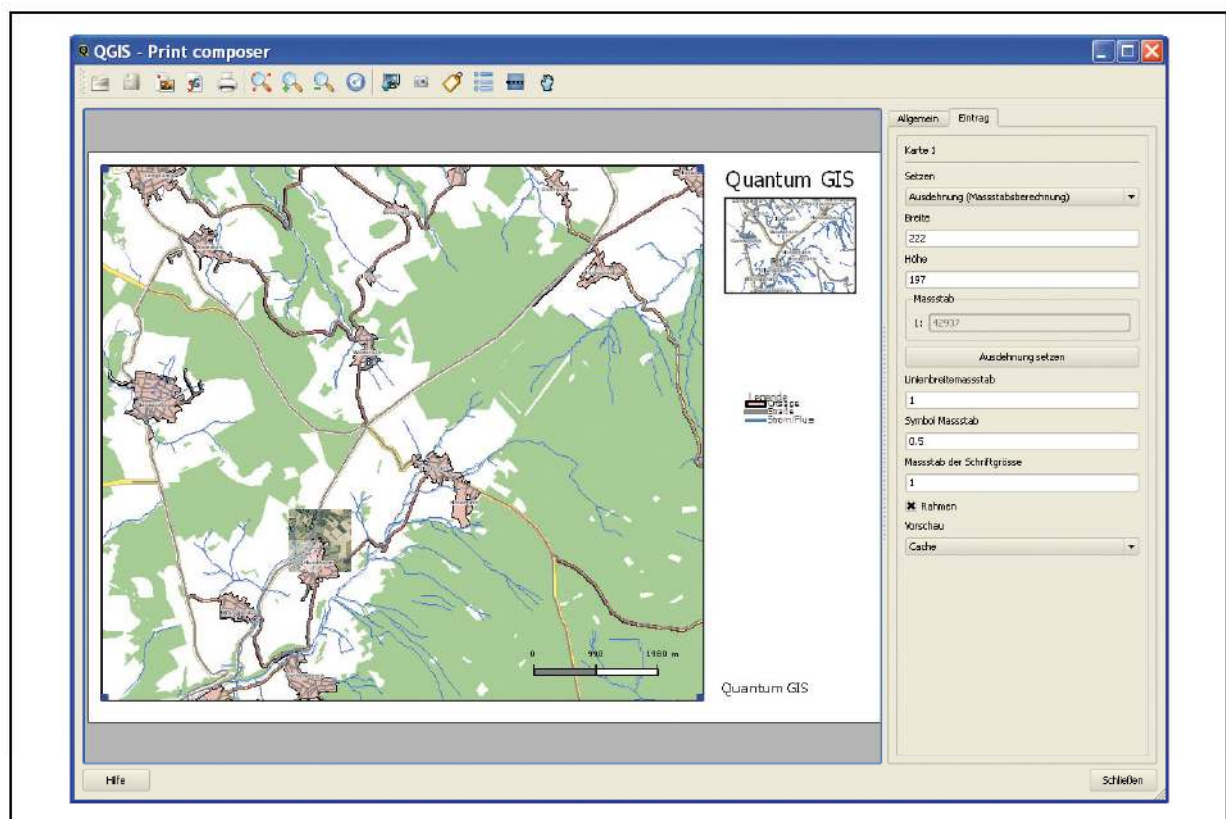


Abbildung 8-14: Konfiguration des Layouts für die Druckausgabe

Export des QGIS-Projekts in eine MapServer-Mapdatei

Quantum GIS bietet die Möglichkeit, ein Projekt in eine UMN MapServer-Mapdatei zu überführen (Abbildung 8-15). Die Mapdatei ist die Konfigurationsdatei für UMN MapServer, die alle wichtigen Informationen über Datenquellen, Darstellungsbereich und Ausgestaltung enthält. Über die Mapdatei können die Daten leicht in einem WebGIS-Projekt veröffentlicht werden. Wir werden den Aufbau der UMN MapServer-Mapdatei und die Einsatzmöglichkeiten von UMN MapServer in den folgenden Kapiteln noch näher behandeln, lernen mit Quantum GIS aber an dieser Stelle schon ein Werkzeug kennen, das uns auf dem Weg ins Web unterstützt.

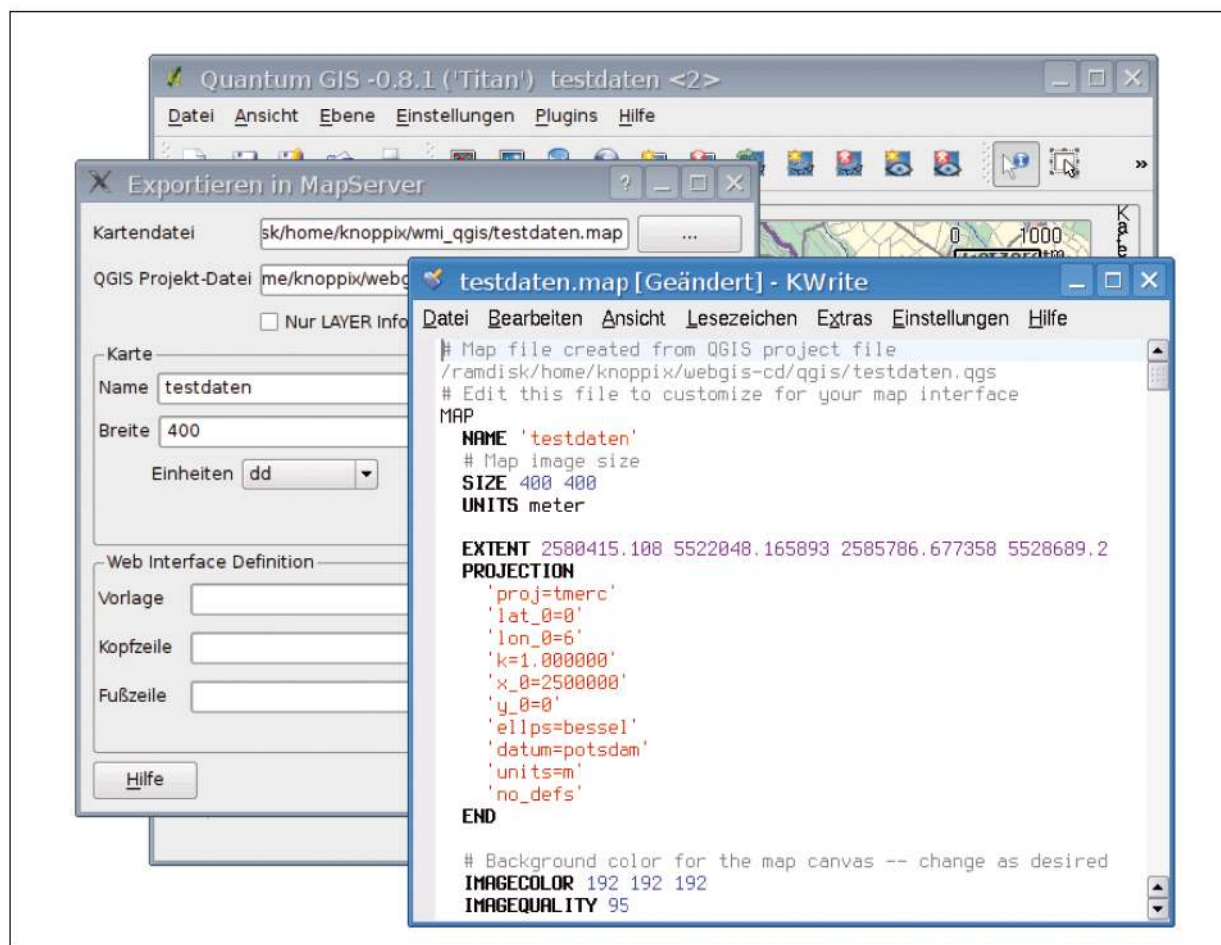


Abbildung 8-15: Ausgabe einer UMN MapServer-Mapdatei; die Projekteinstellungen werden in UMN MapServer-Syntax übertragen.

Speichern des Projekts

Wenn Sie ein Projekt aufgebaut und konfiguriert haben, sollten Sie dieses sichern, um es zu einem späteren Zeitpunkt wiederverwenden zu können. Vorher sollten Sie noch die Projekteigenschaften einstellen. Zu den Projekteigenschaften gelangen Sie über den Menüeintrag *Einstellungen* → *Projekteinstellungen*. Hier können Sie dem Projekt einen Titel geben und die Karteneinheiten sowie die verwendete Projektion festlegen.

Das Speichern des Projekts erfolgt über den Menüpunkt *Datei* → *Speichere Projekt*. Sie müssen einen Speicherort bestimmen und einen Dateinamen vergeben. Es wird eine QGS-Datei erzeugt, die nach einem XML-Schema aufgebaut ist.

Ein fertiges Projekt könnte so aussehen wie das Testprojekt auf der WebGIS.rlp Live-CD. Schauen Sie es sich einmal an.

Erzeugen einer 3-D-Ansicht

Abschließend soll in diesem Kapitel nun vorgestellt werden, wie 3-D-Ansichten erstellt werden können. Dieses soll mithilfe der Software OpenEV demonstriert werden. Aus dem Open Source-Bereich können Sie beispielsweise auch das Werkzeug OSSIM (<http://www.ossim.org>) zum Erzeugen von 3-D-Ansichten einsetzen.

Um eine 3-D-Ansicht zu erzeugen, benötigen Sie ein Digitales Geländemodell (DGM) oder auch Digitales Höhenmodell (DHM). Ein Digitales Geländemodell ist ein digitales, numerisches Modell der Geländehöhen und -formen. Im Englischen werden die Begriffe Digital Terrain Model (DTM) und Digital Elevation Model (DEM) verwendet. Im DGM wird die Geländeoberfläche durch unregelmäßig oder regelmäßig angeordnete Stützpunkte angenähert.

OpenEV erzeugt einfache 3-D-Ansichten Ihrer Daten. Dies erfolgt über die Angabe von zwei Ebenen in OpenEV. Die eine Datei zeigt das Digitale Geländemodell (DGM), die zweite Datei, die geladen wird, enthält die Grafik, die über das DEM gelegt werden soll.

Aufbereiten der DEM-Datei

Sie benötigen ein Höhenmodell. Für die USA liegen viele DGM-Daten vor. Diese können in OpenEV ohne Bearbeitung genutzt werden. Schwarze Bereiche stellen dabei die tiefer gelegenen Regionen und weiße die höheren Lagen dar.

Unter den folgenden Links finden Sie DGM-Datenquellen. Schauen Sie sich auch auf Wikipedia die Seite http://de.wikipedia.org/wiki/Digitales_Gel%C3%A4ndemodell an. Hier erhalten Sie weiterführende Informationen und Links.

- GTOPO: Global digital elevation model: <http://edcdaac.usgs.gov/gtopo30/gtopo30.html>
- SRTM: Global coverage from Shuttle Radar Topography Mission: <http://srtm.csi.cgiar.org/Index.asp>
- USGS DEM: Digital elevation models for the United States: <http://edc.usgs.gov/products/elevation.html>
- U.S. National Geophysical Data Center: Various types of data: <http://www.ngdc.noaa.gov/mgg/bathymetry/relief.html>

Das Digitale Geländemodell, das in diesem Beispiel verwendet werden soll, heißt *hibbing-w.htm* und wurde im USGS DEM-Format von der Seite <http://edc.usgs.gov/>

products/elevation.html heruntergeladen. Sie finden die Datei aber auch auf der Beispiel-seite zum Download unter:

<http://www.webmappingillustrated.com/beispiele>

Starten Sie OpenEV und laden Sie die DEM-Daten in einen *View*. *View* ist die Bezeichnung für ein Projekt in OpenEV. Die Daten laden Sie über den Menüeintrag *File* → *Import*. Sie sehen ein Graustufenbild mit hellen und dunklen Bereichen. Wenn Sie in den 3-D-Modus wechseln, werden Sie sehen, dass Schwarz die niedrigen Bereiche und Weiß die höheren Bereiche in diesem Modell widerspiegeln.

Schauen Sie sich die wichtigsten Hinweise zu OpenEV auf der Projektseite (*<http://openev.sourceforge.net/>*) an. Sie können in OpenEV viele Prozesse über die Tastatur oder die Maus anstoßen. Schauen Sie sich daher den folgenden Link mit Hinweisen zur Verwendung der Buttons, der Maus sowie der Tastaturbefehle an:

http://openev.sourceforge.net/app/viewarea_keys.html



Abbildung 8-16: DEM eines Teils von Minnesota (USA) (Ansicht im 2-D-Modus)

Die Datei *hibbing-w.htm* ist ein gutes Beispiel für den leichten Übergang der Formen von Schwarz nach Weiß, in Abbildung 8-16 gut zu erkennen über eine Ansicht im 2-D-Modus. Dieser sanfte Farbübergang ist sehr gut für die 3-D-Darstellung geeignet. Ein DGM, das über einen eingeschränkten Farbgradienten verfügt und scharfe Kontraste aufweist, erzeugt eine unrealistisch wirkende Oberfläche.

Vorbereitung der Daten

Die zweite Datei, die Sie benötigen, ist ein Bild, das auf das DGM gelegt werden soll. Dabei müssen Sie ein paar Dinge beachten. Das DGM und das überlagernde Bild müssen den gleichen Bereich abdecken. Sie können ein hochaufgelöstes DGM und ein Bild mit einer niedrigen Auflösung verwenden. Dies stellt kein Problem dar. Die Dateien müssen aber über den gleichen Extent verfügen.

Wie erzeugen Sie nun zwei Dateien mit gleichem Extent? Um sicherzugehen, können Sie die Dateien jeweils in eine neue Datei überführen. Dies wird über die Schnappschussfunktion (Menüpunkt *File* → *Print*) von OpenEV realisiert. In diesem Beispiel werden die neuen Dateien über einen Schnappschuss erzeugt.



Das Aufziehen eines Rechtecks zum Zoomen erfolgt in OpenEV mit der Maus bei gedrückter Strg-Taste.

Beginnen Sie, indem Sie das DEM in eine Ansicht von OpenEV laden, und zwar über den Menüpunkt *File* → *Import*. Laden Sie dazu zusätzlich die Datei *landcover.tif*. Die Daten finden Sie unter dem folgenden Link:

<http://www.webmappingillustrated.com/beispiele>

Nun soll der Schnappschuss erstellt werden. Achten Sie dabei darauf, dass Sie in den Bereich zoomen, in dem sich die DGM-Daten befinden. Stellen Sie nun alle Ebenen mit Ausnahme der DEM-Ansicht aus und zoomen Sie auf die Ansicht des Bilds. Nun erzeugen Sie einen Schnappschuss der DGM-Ansicht. Nutzen Sie dazu die Druckfunktion über den Menüeintrag *File* → *Print*. Machen Sie im Druckdialog die folgenden Angaben und speichern Sie die Ansicht als TIFF:

Driver: TIFF
Device: File
File: <path to a local folder>/dem.tif
Output Type: Greyscale
Resolution: 1

Abbildung 8-17 zeigt den Druckdialog von OpenEV und veranschaulicht diesen Vorgang.

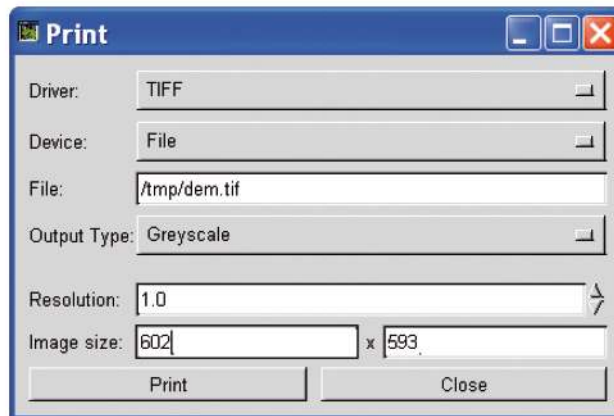


Abbildung 8-17: OpenEV-Druckdialog zum Erstellen eines Schnappschusses

Im nächsten Schritt schalten Sie das DGM zur Ansicht aus und stattdessen alle anderen Ebenen an. Sie erzeugen nun ein Bild, das das DGM überlagern soll. Denken Sie daran, dass Sie, während Sie die Bilder erzeugen, nicht im Client navigieren, da ja das Ziel der Aktion darin besteht, zwei Dateien, die denselben Bereich abdecken, zu erstellen. In Abbildung 8-18 sehen Sie eine Ansicht, die Sie für die Überlagerung verwenden können.

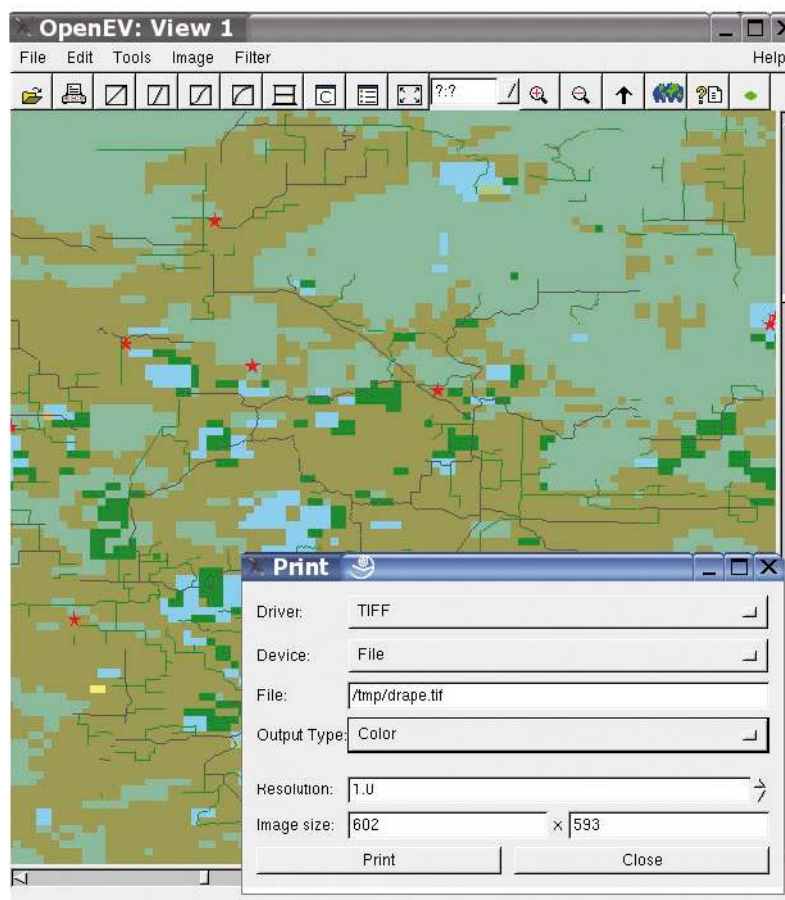


Abbildung 8-18: Ausgabe einer Datei, die das DEM überlagern soll

Initialisierung der 3-D-Ansicht

Nun kann die 3-D-Ansicht erstellt werden. Wählen Sie den Menüpunkt *FILE* → *New*, um eine neue Ansicht zu erzeugen. Wählen Sie anschließend den Menüpunkt *File* → *Open 3D*. Es öffnet sich ein Auswahlfenster, über das Sie zunächst die DGM-Datei und anschließend die Überlagerungsdatei angeben. Wählen Sie hier die neu erstellten Dateien aus. Belassen Sie ansonsten die Standardeinstellungen. Schauen Sie sich dazu Abbildung 8-19 an.

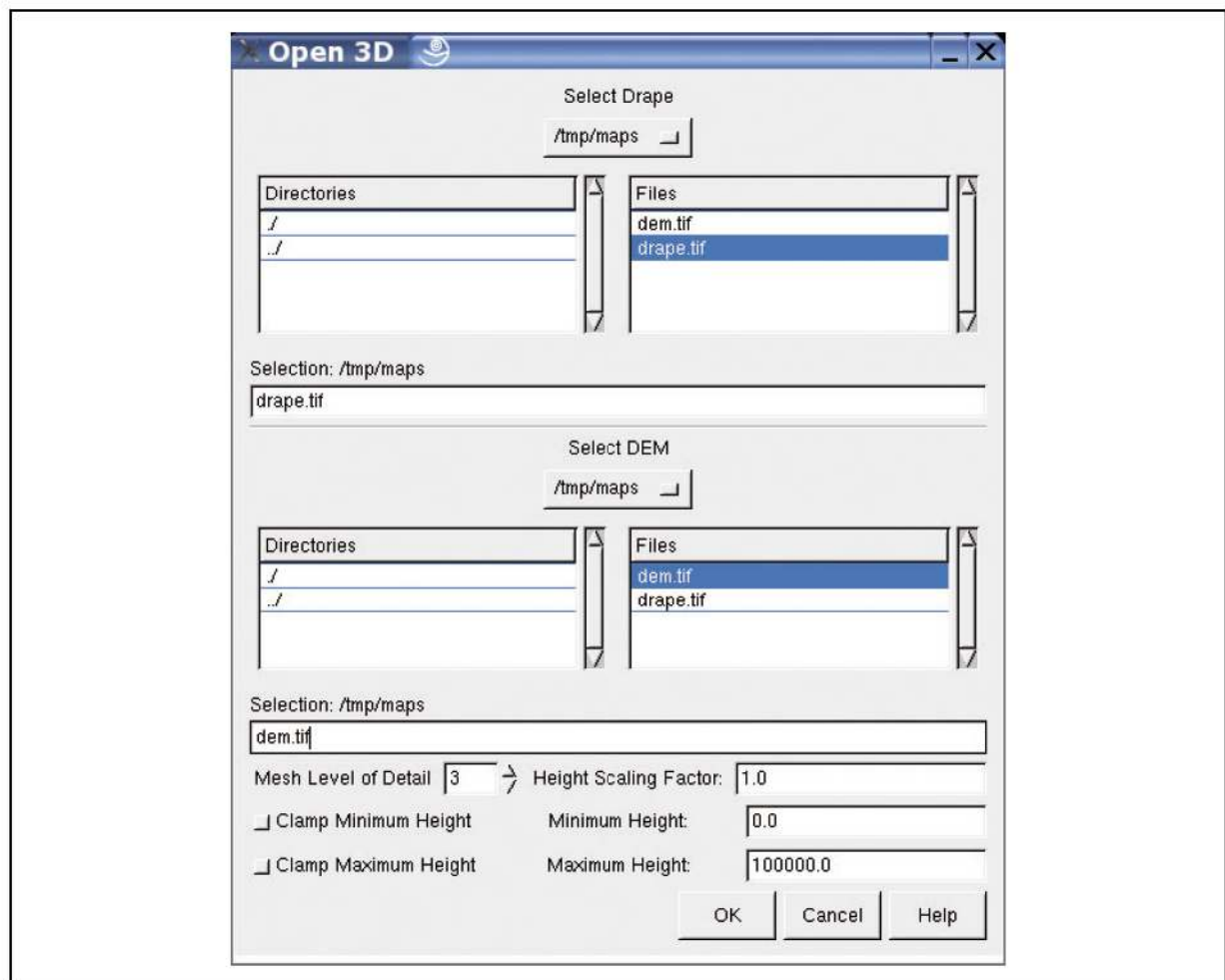


Abbildung 8-19: Laden der Dateien für die 3-D-Ansicht

Der Hintergrund wird schwarz angezeigt, und das farbige Überlagerungsbild wird entsprechend den Angaben der darunter befindlichen DGM-Datei als Relief angezeigt. Sie können nun durch das Bild navigieren und das Bild erhöhen oder abflachen (Abbildung 8-20).

Navigation in der 3-D-Ansicht

Die Navigation in der 3-D-Ansicht ist nicht unbedingt intuitiv und auch nicht einfach zu nutzen, aber sie reicht aus, um einen schnellen Eindruck von Ihren Daten zu bekommen. Behalten Sie im Hinterkopf, dass OpenEV verschiedene Informationen über das 3-D-Modell und Ihre Position vorhält. Die 3-D-Ansicht ist dabei mit einer Kamera ver-

gleichbar, bei der Sie durch einen Bildsucher schauen. Die Kamera schaut immer aus der Luft auf das Objekt, in unserem Fall das 3-D-Modell. Standardmäßig wird die Kamera oberhalb des Modells, aber nicht direkt darüber positioniert. Der Bereich des Modells, der im Zentrum der Ansicht erscheinen soll, kann angegeben werden, ebenso der Neigungswinkel der Kamera.

Wie geschieht nun die Navigation in OpenEV? Das Zoomen erfolgt über einen Klick der linken Maustaste in die Karte. Dies gilt sowohl für 2-D- als auch für 3-D-Ansichten. Beim Zoomen ist es dabei nicht relevant, wo sich der Mauszeiger befindet. Es wird immer in das Zentrum der Ansicht gezoomt. Wenn Sie nun die Maus bei gedrückter linker Maustaste bewegen, wird die Kamera in horizontaler und vertikaler Richtung bewegt. Diese Rotation ist ein bisschen knifflig, denn wenn Sie die Maus einen Moment nicht bewegen, während Sie die Kamera schwenken, zoomt OpenEV direkt in die Karte.

Um die Kamera auf und ab bzw. von links nach rechts zu bewegen, müssen Sie die Strg-Taste drücken, während Sie die Maus mit gedrückter linker Maustaste bewegen.

Es gibt ein paar weitere Tastaturbefehle, die die gleichen Aufgaben wie die Mausaktionen erfüllen. Über die Tasten *Bild auf* und *Bild ab* können Sie zoomen. Mithilfe der Pfeiltasten bewegen sich horizontal und vertikal. Die Taste *Pos1* bringt Sie zur Ursprungsansicht zurück.

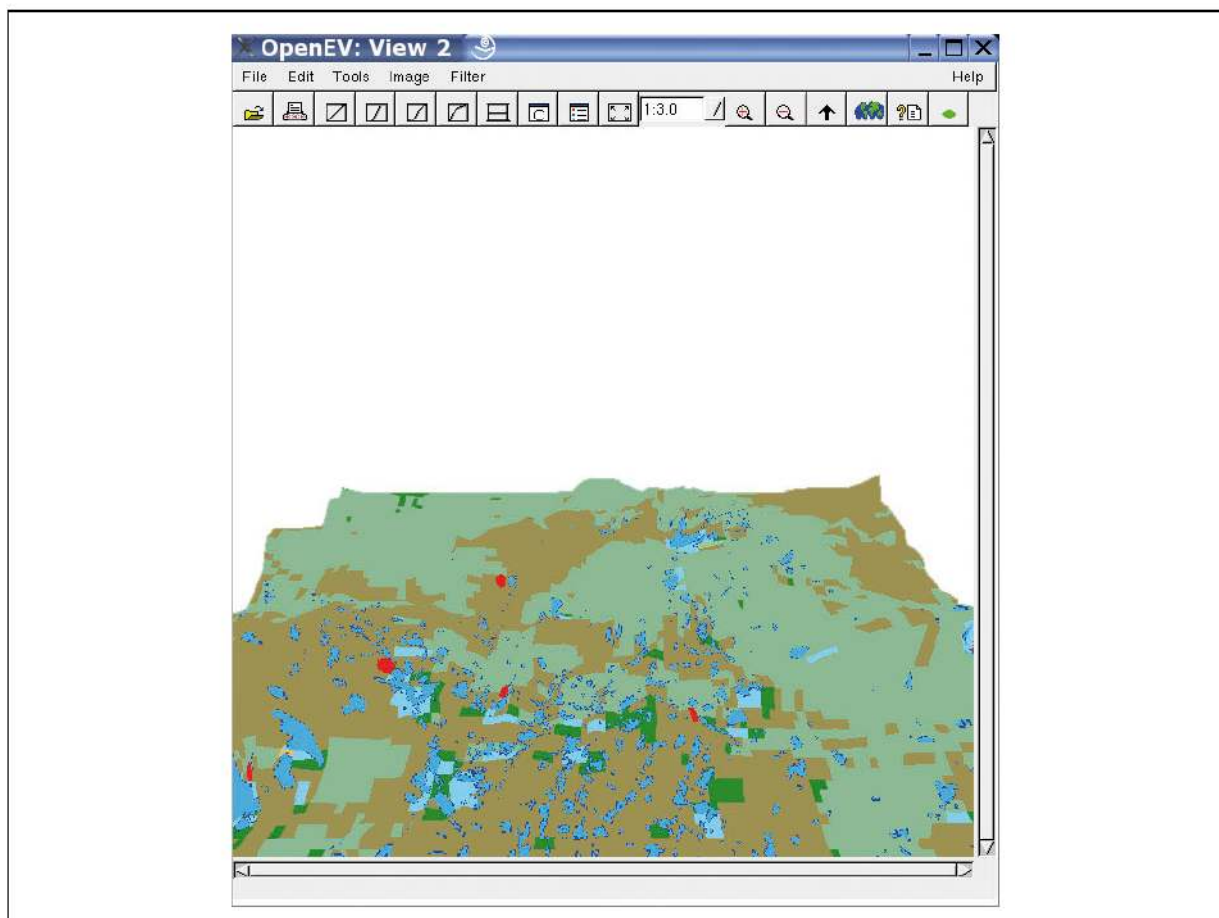


Abbildung 8-20: Standardansicht des 3-D-Modells

Sie können die Kamera auch manuell über die Angabe der Koordinaten positionieren. Dies ist hilfreich, wenn Sie eine Ansicht zu einem späteren Zeitpunkt wiederherstellen möchten. Um die Koordinaten einzugeben, öffnen Sie über den Menüpunkt *Edit* → *3D Position* ein Fenster. Wie Sie in Abbildung 8-21 sehen können, stehen Ihnen drei Felder zur Verfügung, in die Sie die x-, y- und z-Position Ihrer Kamera eingeben können. Zusätzlich können Sie die Koordinaten angeben, auf die die Kamera gerichtet sein soll. Diese Angaben verweisen auf Pixel, Zeilen und Spalten. Wenn Sie die Angaben ändern, wird die Ansicht aktualisiert, sobald Sie die Enter-Taste drücken oder mit der Tabulatortaste in das nächste Feld wechseln.

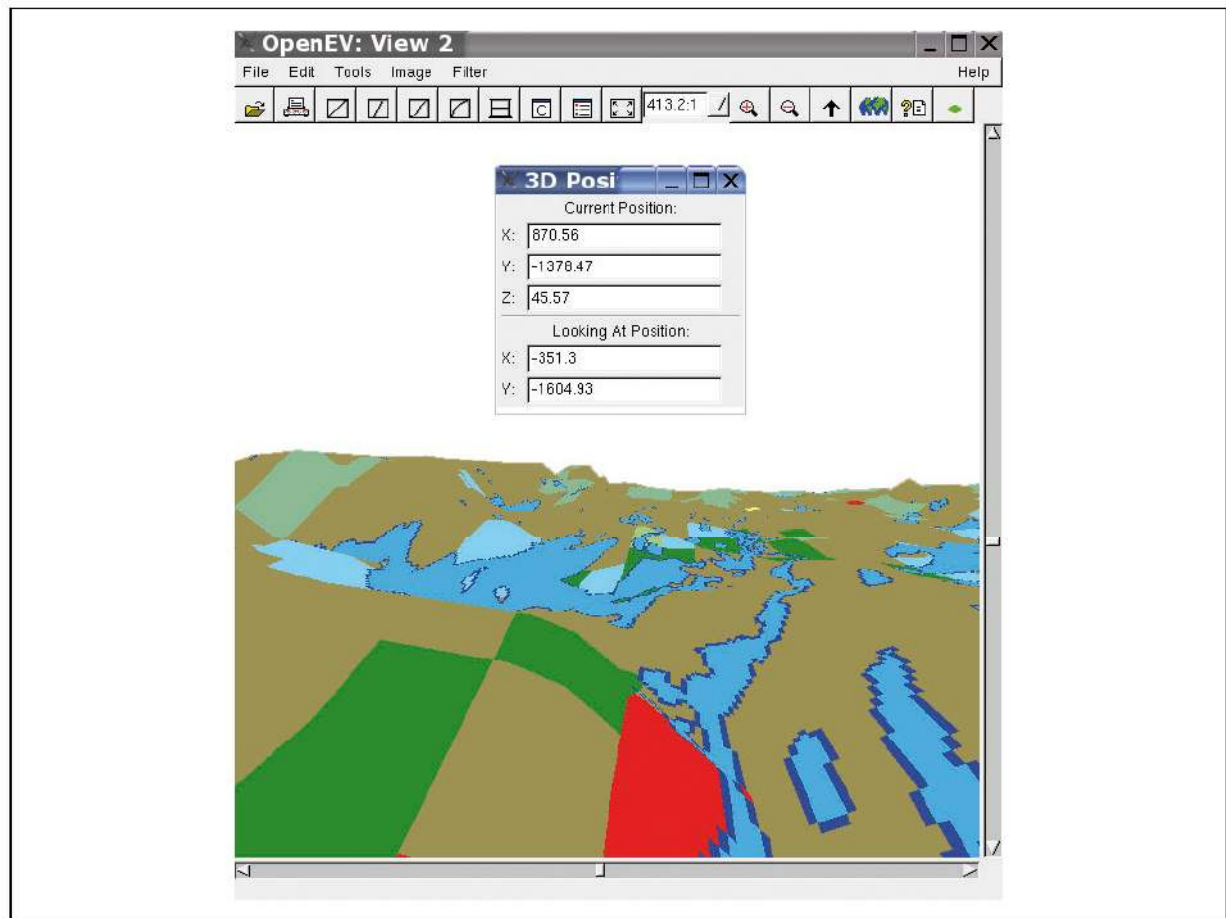


Abbildung 8-21: Positionsfenster, über das die Kameraposition und der Fokus angegeben werden können

Jedes 3-D-Modell verfügt über Angaben zur Höhe (Z-Skalierungsfaktor). Dies ist der Faktor, um den das DGM vertikal gedehnt wird, was der Ansicht den 3-D-Effekt verleiht. Dieser Wert gibt in der Regel eine Zahl größer oder gleich 1 an und wird auch als VE (vertical exaggeration) bezeichnet. Der Standardwert ist 1 und wird in OpenEV im geöffneten 3-D-Fenster vor der Initialisierung der 3-D-Ansicht gesetzt. Ist der Wert zu klein, erscheint die Ansicht flach, wird der Wert zu hoch angegeben, wirkt sie unrealistisch. Die Einstellung kann während der Ansicht geändert werden. Nutzen Sie dazu die Plus- und Minustasten in Kombination mit der Umschalt-Taste.

Dieses einfache Beispiel sollte lediglich einen kleinen Einblick in die Möglichkeit der 3-D-Ansicht geben. Wenn Sie Google Earth kennen, haben Sie sicherlich die 3-D-Funktionalität schon einmal bewundert. Dieses Beispiel sollte Ihnen einen kleinen Einblick in die Hintergründe und die Abläufe geben.

Eigene Geodaten erzeugen und bearbeiten

Dieses Kapitel soll einen Einblick geben, wie Sie ein eigenes Kartenprojekt aufbauen können und was es dabei zu beachten gilt. Ein Projekt fängt schon mit der Planung an, daher erhalten Sie in diesem Kapitel zunächst Hinweise dazu. Je nach Projekt liegen die Geodaten schon vor oder müssen erst erstellt werden. Es wird an Beispielen gezeigt, wie Sie eigene Geodaten erheben können. Die Erhebung der Daten wird mithilfe von Quantum GIS demonstriert, ein Tool, das Sie schon im vorangegangenen Kapitel kennengelernt haben. Dabei sollen eigene Linienobjekte erzeugt und mit Attributen versehen werden. Außerdem wird das Importieren von GPS-Daten demonstriert. Sie erfahren darüber hinaus, wie Daten aus einer Textdatei geladen werden können. Außerdem wird anhand eines Bauleitplans die Georeferenzierung von Rasterdaten demonstriert.

Planen Sie Ihr eigenes Projekt

Wenn Sie ein Kartenprojekt aufsetzen wollen, ist es sinnvoll, dieses im Vorhinein gut zu planen. Erstellen Sie eine Liste, in der Sie die Arbeitsschritte definieren. Führen Sie die Daten auf, die Sie benötigen, und definieren Sie den Darstellungsbereich des Projekts. Ein gutes Konzept kann viel Zeit ersparen und Ihnen die Arbeitsschritte erleichtern.

Wenn Sie ein Projekt beginnen, sollten Sie erst einmal folgende Fragen klären:

- Welcher Bereich soll in dem Projekt dargestellt werden?
- Welches Koordinatenreferenzsystem und welche Karteneinheiten sollen verwendet werden?
- Welche Kartengröße wird benötigt?
- Soll eine interaktive Karte erstellt werden oder eine statische?
- Soll ein standardisierter Kartendienst aufgesetzt werden (WMS)?
- Welche Daten benötigen Sie, und wie müssen diese aufgebaut sein?
- Greifen Sie auf externe Kartendienste zu? Müssen Sie hierzu eine Genehmigung einholen?
- Müssen Sie eigene Daten erheben?

Bestehende Daten zur Orientierung nutzen

Wenn Sie ein neues Projekt aufsetzen und eigene Daten erheben müssen, ist es gut, eine Hintergrundkarte des Gebiets zur Orientierung vorliegen zu haben. Hierbei ist es aber auch wichtig, dass die Daten, die Sie zur Orientierung nutzen, in einem geeigneten Maßstab vorliegen. Auf Grundlage dieser Daten können Sie dann im nächsten Schritt neue Daten erheben. Als Hintergrundkarte eignen sich beispielsweise Orthophotos, die Sie als Dateien vorliegen haben. Sie können aber auch einen Kartendienst (WMS) zur Orientierung laden. Im Geoportal.rlp (<http://www.geoportal.rlp.de>) gibt es beispielsweise einen WMS mit Luftbildern, den Sie als Hintergrund laden können. Suchen Sie nach dem Schlagwort *Luftbild* und lassen Sie sich die Informationen zu dem Dienst ausgeben. In den Informationen finden Sie auch den GetCapabilities-Aufruf, über den Sie den Dienst laden können. Beachten Sie aber dabei die Nutzungsbedingungen des jeweiligen Diensts. In Kapitel 5 finden Sie eine Liste mit freien Geodaten und Links zu standardisierten Diensten. Schauen Sie sich einmal um, vielleicht finden Sie geeignete Daten.

Darstellungsbereich und Maßstab auswählen

Achten Sie darauf, dass Sie in Ihrem Kartenprojekt das richtige Koordinatenreferenzsystem verwenden. Dies ist wichtig, da die Daten, die Sie erheben werden, in dem Koordinatenreferenzsystem gespeichert werden, das in dem Projekt ausgewählt ist. Wenn Sie Daten in Gauß-Krüger-Zone 2 erheben wollen, sollte die Kartenprojektion auch auf EPSG:31466 gesetzt werden und beispielsweise nicht WGS84 sein. Außerdem müssen Sie die Karteneinheiten angeben. Gauß-Krüger ist ein metrisches Koordinatenreferenzsystem, daher geben Sie Meter als Karteneinheiten an. Navigieren Sie in den Bereich, in dem Sie Daten erheben wollen. Wählen Sie einen geeigneten Maßstab aus, damit Sie die neuen Objekte in einem sinnvollen Maßstab erzeugen können. Wenn Sie beispielsweise Flurstücke erfassen möchten, ist es nicht sinnvoll, diese in einem Maßstab zu erfassen, in dem der ganze Landkreis dargestellt wird. Sicherlich ist es günstiger, dazu in einen Maßstab zu zoomen, bei dem nur einige Flussstücke angezeigt werden und daher der Verlauf der Objekte genauer erfasst werden kann.

Welche Datenquellen sollen genutzt werden?

Nachdem Sie die Karten, die zur Orientierung dienen sollen, gefunden haben, brauchen Sie eigene Daten. Diese Daten können Sie auf unterschiedliche Weise erstellen.

GPS

Sie können Daten über GPS-Geräte erheben. GPS (Global Positioning System) ist ein satellitengestütztes System, über das Sie mit einem GPS-Empfänger Ihre exakte Position anhand von Koordinaten bestimmen können. GPS wird in vielen Bereichen als Hilfsmittel verwendet (z.B. in Navigationsgeräten in Fahrzeugen oder zur Orientierung bei

Wanderungen). Der GPS-Empfänger benutzt dabei Signale der Satelliten, um Längen- und Breitengrad und gegebenenfalls auch die aktuelle Höhe zu berechnen.

GPS-Geräte sind ein geeignetes Werkzeug, mit dem Sie Ihre Daten erheben können. Die mit dem GPS-Gerät erhobenen Daten können entweder direkt mit einem Client geladen werden oder müssen exportiert und in ein geeignetes Format (PostgreSQL/PostGIS, Shape) überführt werden.



Wenn Sie aus dem Gerät beispielsweise eine CSV-Datei ausspielen, kann diese leicht über `ogr2ogr` in ein anderes Datenformat überführt werden. `ogr2ogr` kann auch das Format GPX (GPS Exchange Format), in dem GPS-Daten gespeichert werden können, lesen und schreiben. In diesem Kapitel wird das Quantum GIS-Plugin für den Import von GPS-Daten vorgestellt.

Luftbildinterpretation

Luftbilder eignen sich gut zur Orientierung. Das Luftbild, das Sie verwenden, muss georeferenziert sein, d.h., dem Bild müssen Informationen über seine räumliche Lage zugewiesen werden. Den Prozess der Georeferenzierung werden Sie in diesem Kapitel anhand von Quantum GIS kennenlernen.

Sie können neue Daten auf Grundlage von Luftbildern erheben. Sie ermitteln Objekte auf dem Luftbild, zeichnen diese als Punkte, Linien oder Polygone nach und fügen der Geometrie beschreibende Informationen hinzu. Durch die Angabe von Attributen werden die Geodaten wertvoll und können für Auswertungen und Abfragen verwendet werden.

Geodaten aufbereiten

Manchmal liegen geeignete Daten vor, müssen aber noch aufbereitet werden, um in Ihrem Projekt genutzt werden zu können. Vielleicht haben Sie die Länder der ganzen Welt vorliegen, benötigen aber nur die Länder Europas. Das heißt, Sie müssen Ihre Daten anpassen. Dies können Sie mithilfe von Quantum GIS durchführen. Sie haben in Kapitel 7 auch schon die Programme `gdal_translate` und `ogr2ogr` kennengelernt, mit denen Sie über einen Befehl auf der Kommandozeile Teilmengen der Daten in eine neue Datei ausgeben können.

Manchmal bedarf es einer komplexeren Datenaufbereitung. Vielleicht müssen Sie Daten puffern, Verschneidungen durchführen oder Distanzen berechnen. Hierzu benötigen Sie leistungsstarke Programme, die ein Geoprocessing durchführen können. Im Open Source-Bereich ist hier vor allem GRASS zu nennen. Es empfiehlt sich ebenso eine Datenhaltung in PostgreSQL/PostGIS. Über die PostGIS-Funktionen können Sie leicht Berechnungen durchführen.

Persönliche Daten

Vielleicht möchten Sie auch persönliche Informationen in einer Karte darstellen. Dies sind Informationen, die nur Sie erfassen können oder wollen. Für andere sind diese Daten vielleicht nicht von Interesse. Um eigene Daten zu erheben, brauchen Sie ein Werkzeug und ein Datenformat, in dem Sie die Daten speichern. Das Werkzeug kann ein Desktop-GIS sein oder auch eine webbasierte Anwendung. Als Datenformat stehen Ihnen zahlreiche Möglichkeiten zur Verfügung. Im einfachsten Fall können Sie Ihre Daten auch in einer einfachen Textdatei ablegen.

Daten mithilfe von Quantum GIS aufbereiten und erfassen

In diesem Kapitel sollen Sie weitere Funktionalitäten von Quantum GIS kennenlernen. Eine ausführliche Beschreibung des Projekts finden Sie auf der Projektseite unter <http://qgis.org/> in der Dokumentation. Quantum GIS soll nur beispielhaft vorgestellt werden. Andere Programme weisen ähnliche Funktionalitäten auf. Schauen Sie sich dazu noch mal die Übersicht der verschiedenen Desktop-GIS in Kapitel 8 an. Natürlich ist diese Liste nur eine Momentaufnahme. Aktuelle Informationen finden Sie auf den jeweiligen Projektseiten und auf der Seite der OSGeo unter <http://osgeo.org>.

Die Daten, die in diesem Kapitel verwendet werden, finden Sie über den folgenden Link: <http://www.webmappingillustrated.com/beispiele>. Sie werden teilweise die Testdaten der WebGIS.rlp-CD verwenden. Diese finden Sie auch auf der CD, die dem Buch beiliegt.

Im ersten Abschnitt sollen Sie eine Wanderung von Hundheim über Hinzerath und Emmeroth nach Longkamp planen. Mit den Testdaten bauen Sie ein eigenes Projekt auf, in dem Sie eine Wanderroute in mehreren Etappen digitalisieren. Zusätzlich können Sie die Orte, an denen Sie eine Pause eingeplant haben, als Punkt verorten. Ihre neuen Geometrien werden Sie mit Attributen versehen.

Quantum GIS bietet als Funktionalität derzeit das Neuerstellen von Shape-Dateien und das Editieren nach Shape und PostgreSQL/PostGIS. Über das GRASS-Plugin stehen Ihnen komplexere Editierfunktionalitäten zur Verfügung, die aber an dieser Stelle nicht näher behandelt werden sollen. Informationen zum GRASS-Plugin finden Sie in der Dokumentation von Quantum GIS.

Einrichten des Projekts

Bevor Sie mit der Erfassung Ihrer Daten beginnen, müssen Sie Ihr Projekt einrichten, d.h., Sie müssen das richtige Koordinatenreferenzsystem auswählen, in einen geeigneten Maßstab wechseln und Daten laden, an denen Sie sich orientieren können. Die Testdaten liegen in Gauß-Krüger-Zone 2 vor. Daher müssen Sie 31466 als EPSG-Code auswählen und als Karteneinheit Meter angeben.

Laden Sie zunächst das Orthophoto (*test_dop/dop_50cm.tif*) als Hintergrundkarte zur Orientierung.

Da das Orthophoto nur einen kleinen Teil des Gebiets abdeckt, sollten Sie zusätzlich die folgenden Vektordaten laden: Ortslagen (*test_dlm/25805520_g_2101_p_polygon.shp*), Straßen (*test_dlm/25805520_g_3101_l_line.shp*) und Flüsse (*test_dlm/25805520_g_5101_l_line.shp*).

Sie können auch noch zusätzlich einen WMS zur Orientierung laden. In Kapitel 8 haben Sie schon den folgenden WMS kennengelernt:

http://www.gdi-rp-dienste2.rlp.de/cgi-bin/atkis_ldi?VERSION=1.1.1&REQUEST=GetCapabilities&SERVICE=WMS

Stellen Sie nun das Projektionssystem Ihres Projekts auf EPSG:31466 und setzen Sie die Karteneinheiten auf Meter. Dies erfolgt in den Projekteinstellungen, die Sie über den Menüeintrag *Einstellungen* → *Projekteinstellungen* anpassen können. Wechseln Sie in den Bereich, in dem Sie digitalisieren wollen, und wählen Sie einen geeigneten Maßstab aus. Speichern Sie anschließend Ihr Projekt. Sie haben nun die Grundlagen geschaffen und können mit der Erfassung neuer Daten beginnen.

Eigene Daten erheben

Um in Quantum GIS digitalisieren zu können, müssen Sie die Werkzeugleiste *Digitalisierung* aktivieren. Sollte diese nicht sichtbar sein, können Sie die Leiste über einen Klick mit der rechten Maustaste auf die Menüleiste und anschließender Auswahl des Eintrags *Digitalisierung* aktivieren.

Anlegen einer neuen Vektorebene

Um eigene Daten erfassen zu können, benötigen Sie nun eine neue Vektorebene. In diesem Beispiel sollen die Daten in einer Shape-Datei gespeichert werden. Das Anlegen einer Shape-Datei erfolgt in Quantum GIS über den Menüeintrag *Ebene* → *Neue Vektorebene*. Hier können Sie zwischen verschiedenen Geometrietypen (Punkt, Linie, Polygon) auswählen. Als Datenformat kann derzeit nur Shape ausgewählt werden. Sie können Attributfelder vom Datentyp string, real und integer definieren. Sobald Sie eine neue Vektorebene definiert haben, wird diese in das bestehende Projekt eingefügt.

Erstellen Sie eine neue Vektorebene vom Typ *Linie* und fügen Sie die Attributfelder *etappe* (integer), *info* (string) und *dauer* (real) hinzu. Das Anlegen einer Vektorebene in Quantum GIS wird in Abbildung 9-1 verdeutlicht.

Erzeugen von Geometrien

Nun können Sie mit der Datenerhebung beginnen. Dazu müssen Sie die neue Ebene erst zum Editieren aktivieren. Dies erfolgt über das Kontextmenü der Ebene und den Eintrag *Editieren erlauben* oder über den Button *Editieren erlauben* in der Werkzeugleiste. Beachten Sie, dass die Ebene ausgewählt sein muss, um zum Editieren aktiviert werden zu können.

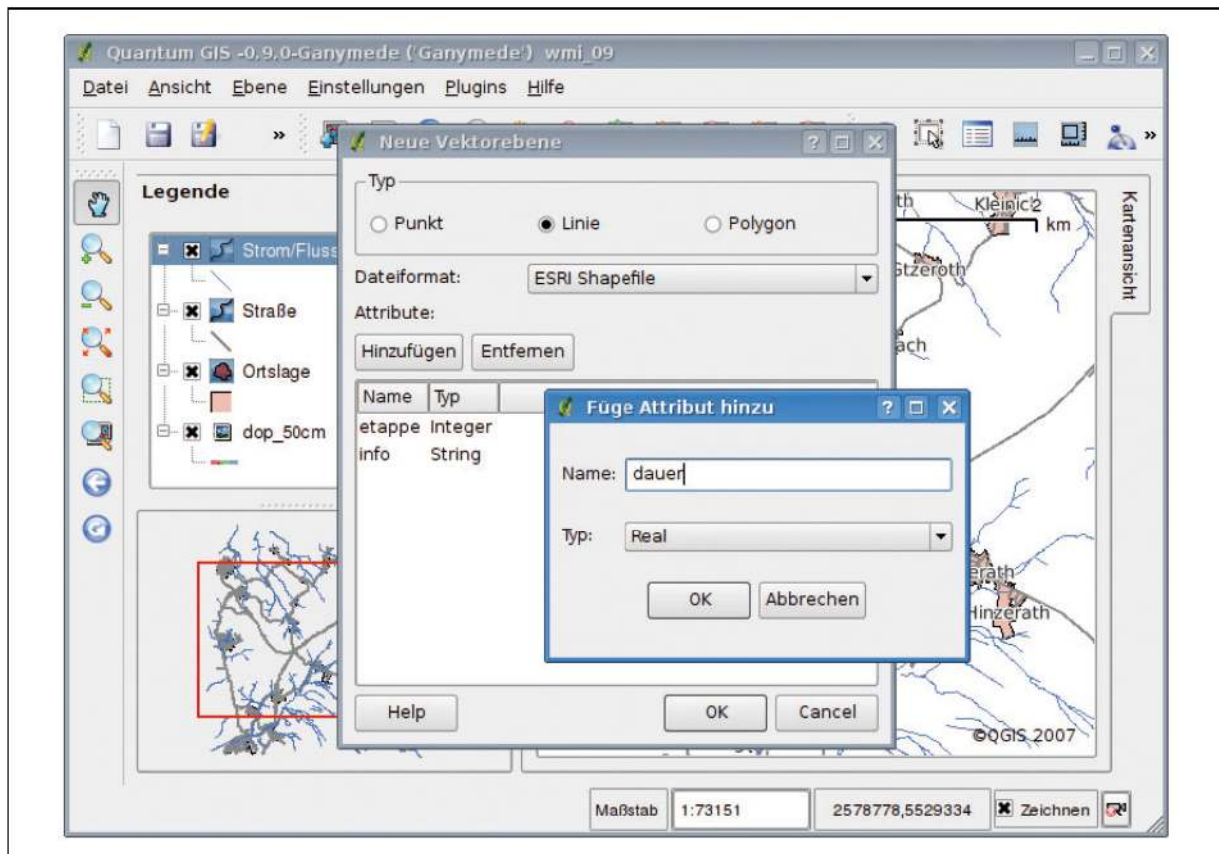


Abbildung 9-1: Anlegen einer neuen Vektorebene für Linienobjekte in Quantum GIS

Legen Sie nun die erste Etappe als Linie an. Aktivieren Sie die Ebene und wählen Sie über die Werkzeugleiste den Button *Linie digitalisieren* aus. Nun können Sie über Klicks mit der linken Maustaste in die Karte die Linie digitalisieren. Sie beenden die Linie über einen Klick mit der rechten Maustaste in die Karte. Es öffnet sich das Attributfenster, in dem Sie die Eigenschaften des Objekts eintragen können. Nach dem Bestätigen der Einträge in die Attributfelder wird das neue Objekt in der Karte angezeigt. Beachten Sie, dass die Stützpunkte der Linien als Kreuze (x) angezeigt werden. Abbildung 9-2 zeigt die neu erstellten Geometrien.

Beachten Sie, dass die neu erzeugten Geometrien im Speicher von Quantum GIS abgelegt werden. Das Speichern der Objekte erfolgt erst, wenn Sie den Editiermodus beenden und an dieser Stelle bestätigen, dass die Objekte gespeichert werden sollen.

Geometrien verändern

Sie können Vektorobjekte, die bereits vorliegen, auch verändern. Hierzu müssen Sie in den Editiermodus wechseln. Die mit x gekennzeichneten Knotenpunkte können nun verändert werden. Sie können Knoten hinzufügen, verschieben und löschen.

In Quantum GIS wird die Linie, die dem Mauszeiger am nächsten ist, als Zielobjekt verwendet. Wollen Sie einen Knotenpunkt verschieben oder löschen, bezieht sich diese Aktion auf den nächsten Knoten.

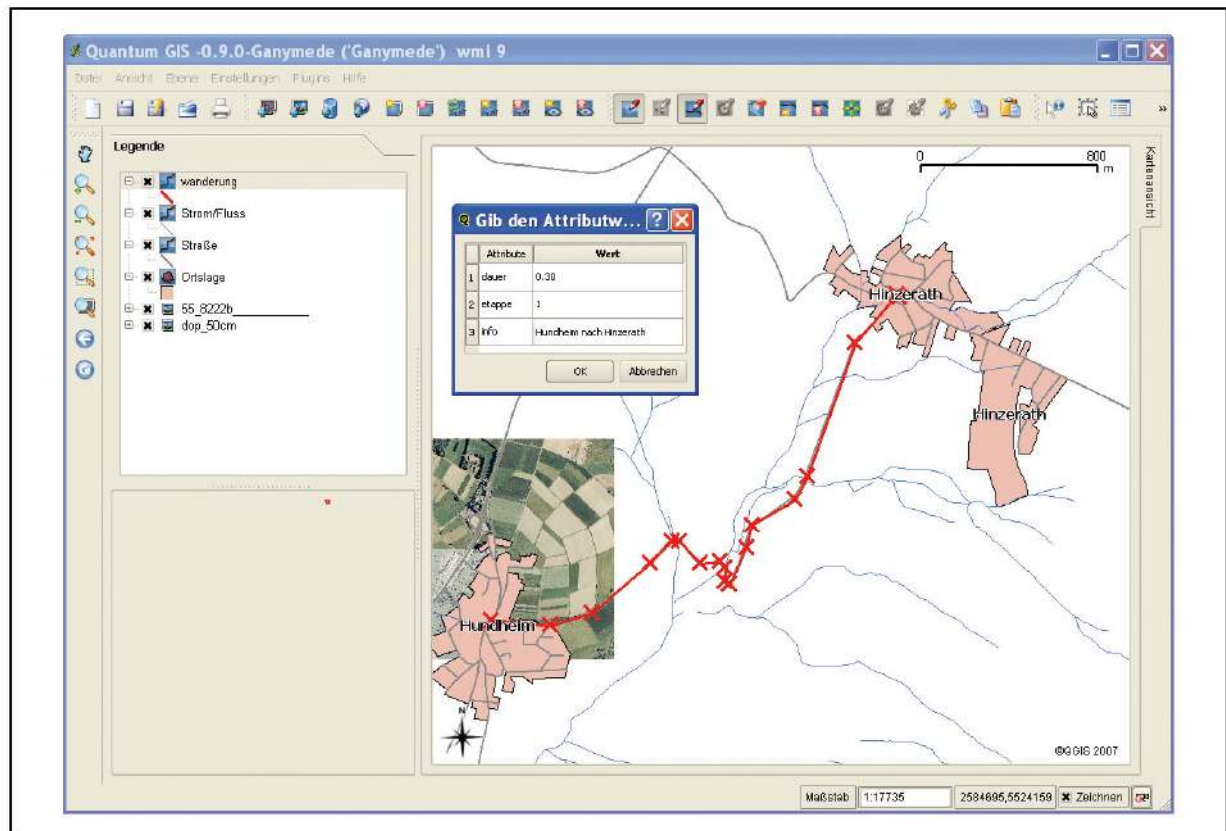


Abbildung 9-2: Mit Quantum GIS erzeugte Geometrien. Im Editiermodus werden die Knotenpunkte mit x gekennzeichnet.



In Quantum GIS können Sie einen Suchradius definieren, über den die Knoten in einer bestimmten Entfernung ausgewählt werden. Dies erfolgt über *Einstellungen* → *Projekteinstellungen* im Bereich *Digitalisierung*. Hier können Sie die Snapping Tolerance in Karteneinheiten angeben. In unserem Beispiel ist die Karteneinheit Meter. Geben Sie hier beispielsweise 10m an.

Um einen Stützpunkt zu verschieben, wählen Sie den Button *Verschiebe Knotenpunkt* in der Werkzeugleiste aus, gehen dann mit der Maus auf den Knoten, den Sie verschieben möchten, und verschieben ihn mit gedrückter rechter Maustaste. Sollten Sie den Knoten nicht fangen können, müssen Sie gegebenenfalls den Suchradius erhöhen.

Sie können auch einen Knotenpunkt zu einem bestehenden Objekt hinzufügen. Dazu wählen Sie den Button *Knotenpunkt hinzufügen* in der Werkzeugleiste aus. Klicken Sie anschließend auf die Stelle der Linie, an der der Stützpunkt eingefügt werden soll.

Knotenpunkte können auch gelöscht werden. Hierzu wählen Sie den Button *Knotenpunkt löschen* aus und klicken mit der Maus auf die Knotenpunkte, die entfernt werden sollen.



Beachten Sie, dass alle diese Änderungen erst gespeichert werden, wenn Sie den Editiermodus verlassen und das Speichern bestätigen.

Sie können auch ganze Objekte aus einer Ebene ausschneiden oder kopieren und in eine andere Ebene überführen. Dazu muss sich die Zielebene ebenfalls im Editiermodus befinden.

Abbildung 9-3 zeigt die neu erstellten Linien. Dabei wurde die Darstellung über die Eigenschaften angepasst. Es wurde eine Beschriftung über das Attribut *info* eingerichtet.

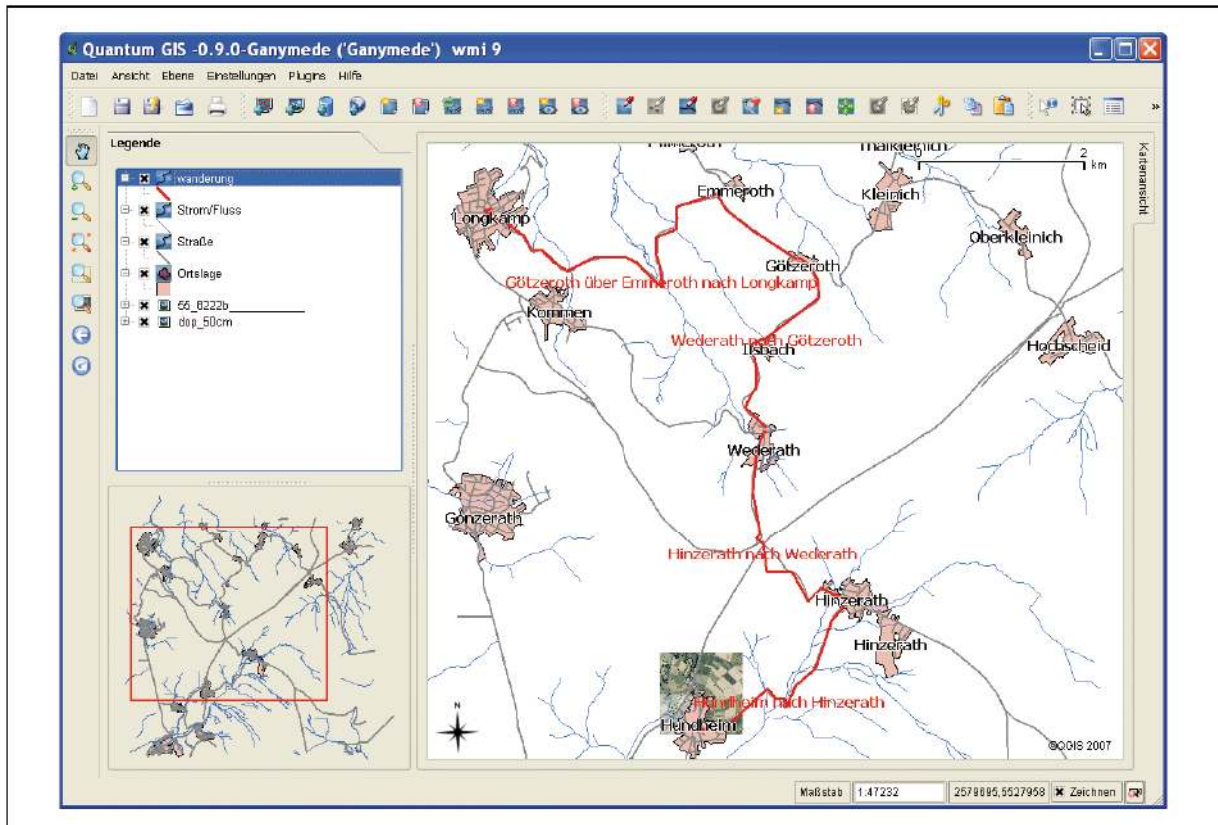


Abbildung 9-3: Über Quantum GIS erzeugte Liniengeometrien zur Darstellung der Wanderroute

Nachdem Sie Ihre Wanderroute digitalisiert haben, können Sie diese auf verschiedene Weise veröffentlichen: Sie speichern die Karte als Grafik (Menüpunkt *Datei* → *Speichere Bild als*), Sie können über den Print Composer (Menüpunkt *Datei* → *Drucken*) ein Drucklayout erstellen und als PDF ausgeben, oder Sie exportieren Ihr Projekt als Map-Server-Mapdatei (Menüpunkt *Datei* → *Exportiere als MapServer Datei*) und veröffentlichen die Daten im Netz über einen Kartendienst.

GPS-Daten importieren

Quantum GIS verfügt über ein Plugin, mit dem Sie GPS-Daten aus verschiedenen Formaten importieren können. Außerdem können Sie Daten im GPX-Format in ein GPS-Gerät hochladen.

GPX steht für GPS Exchange Format und ist ein Datenformat zur Speicherung von GPS-Daten, das von der Firma TopoGrafix entwickelt wurde. Es basiert auf dem allgemeinen XML-Standard. Als Dateiendung wird die Abkürzung *gpx* verwendet.

Die meisten GPS-Empfänger können die Positionen auch in Form von Wegpunkten, Routen oder Spuren speichern. Quantum GIS stellt Wegpunkte als Punkte und Routen sowie Spuren als Vektorlinien dar.

Um den Umgang mit GPX-Dateien zu verdeutlichen, werden Sie über die Webanwendung Naturpark-Scout eine Wanderroute definieren und als GPX-Datei exportieren. Diese Datei soll dann anschließend in Quantum GIS geladen und zusammen mit einem WMS dargestellt werden.

Wanderroute definieren und als GPX-Datei exportieren

Der Naturpark-Scout ist ein Pilotprojekt des Naturparks Schwarzwald Mitte/Nord (Baden-Württemberg, Deutschland). Sie erreichen den Naturpark-Scout über den Link <http://www.naturparkscout.de/>. Es handelt sich um eine Anwendung, die über MapServer WMS dynamische Karten bereitstellt. Die Vektordaten liegen dabei in einer PostgreSQL/PostGIS-Datenbank. Der verwendete Client ist Mapbender, den Sie in Kapitel 13 näher kennenlernen werden.

Über die Oberfläche können Sie Ihre Wander- oder Mountainbike-Route planen. Dazu müssen Sie Start- und Endpunkt Ihrer Route und gegebenenfalls Zwischenpunkte in der Karte setzen. Ihre Route wird dann berechnet und in der Karte dargestellt. Die Berechnung erfolgt über *pgRouting* in der PostgreSQL/PostGIS-Datenbank. In Abbildung 9-4 sehen Sie eine Route von Bad Wildbad zum Sommerberg. Die Route wird in der Karte dargestellt, und es werden zusätzliche Tourinformationen ausgegeben.

Diese Route können Sie nun über den Link *Export* in das GPX-Format überführen. Sie können die Route auch als *KML* speichern und direkt über Google Earth anschauen. KML steht für Keyhole Markup Language und ist ein Austauschformat für die Clientkomponente von Google Earth. KML soll als OGC-Standard aufgenommen werden und wird mittlerweile von vielen Anwendungen unterstützt (z.B. PostGIS, GeoServer, Mapbender).

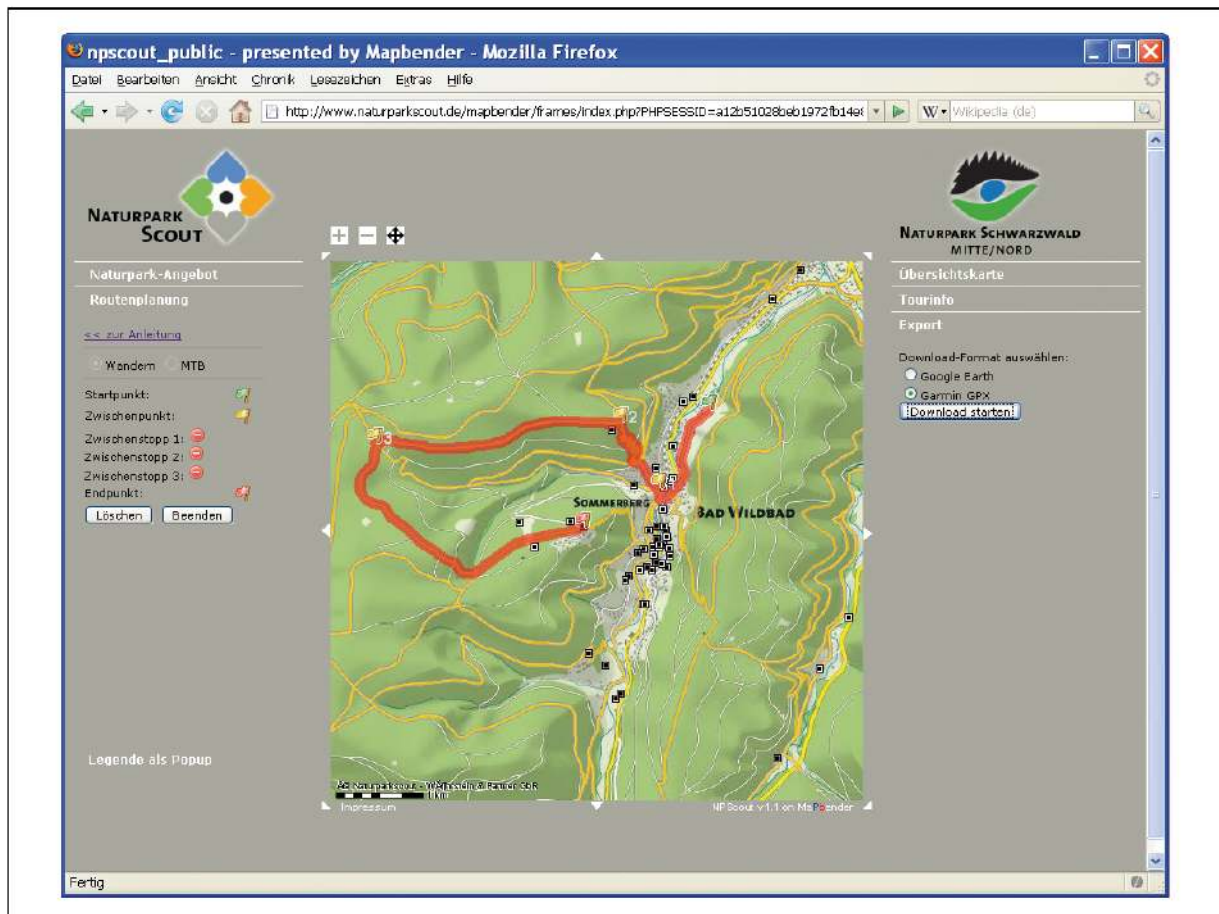


Abbildung 9-4: Routenplanung über den Naturpark-Scout

Laden der GPX-Datei in Quantum GIS

Im nächsten Schritt soll die GPX-Datei in Quantum GIS geladen werden. Dazu müssen Sie das GPS-Plugin über den Plugin-Manager (Eintrag *GPS Werkzeuge*) aktivieren. Anschließend können Sie über den Menüeintrag *Plugins → Gps → GPS Werkzeuge* ein Dialogfenster öffnen. Über dieses Fenster laden Sie nun die GPX-Datei, die Sie durch den Export erhalten haben. Abbildung 9-5 verdeutlicht das Laden einer GPX-Datei in Quantum GIS. Nach dem Laden der GPX-Datei sehen Sie bis zu drei neue Kartenebenen in Ihrem Projekt (waypoints, routes und tracks). Schauen Sie sich die Attribut-tabelle der Ebenen an. Auch die Attribute wurden über die GPX-Datei importiert.

Öffnen Sie die GPX-Datei in einem Texteditor.

```
....
<wpt lat="48.742893214" lon="8.547448665">
  <ele>460.000000</ele>
  <name>Tennisplaetze im Kurpark</name>
  <cmt><br>
    <p>3 Tennisplätze im herrlichen Kurpark..</p>
    <p>Kostenlose Nutzung für Gästekarteneinhaber...</p>
  </cmt>
  <desc>
```



```

.....
</desc>
</wpt>
.....

```

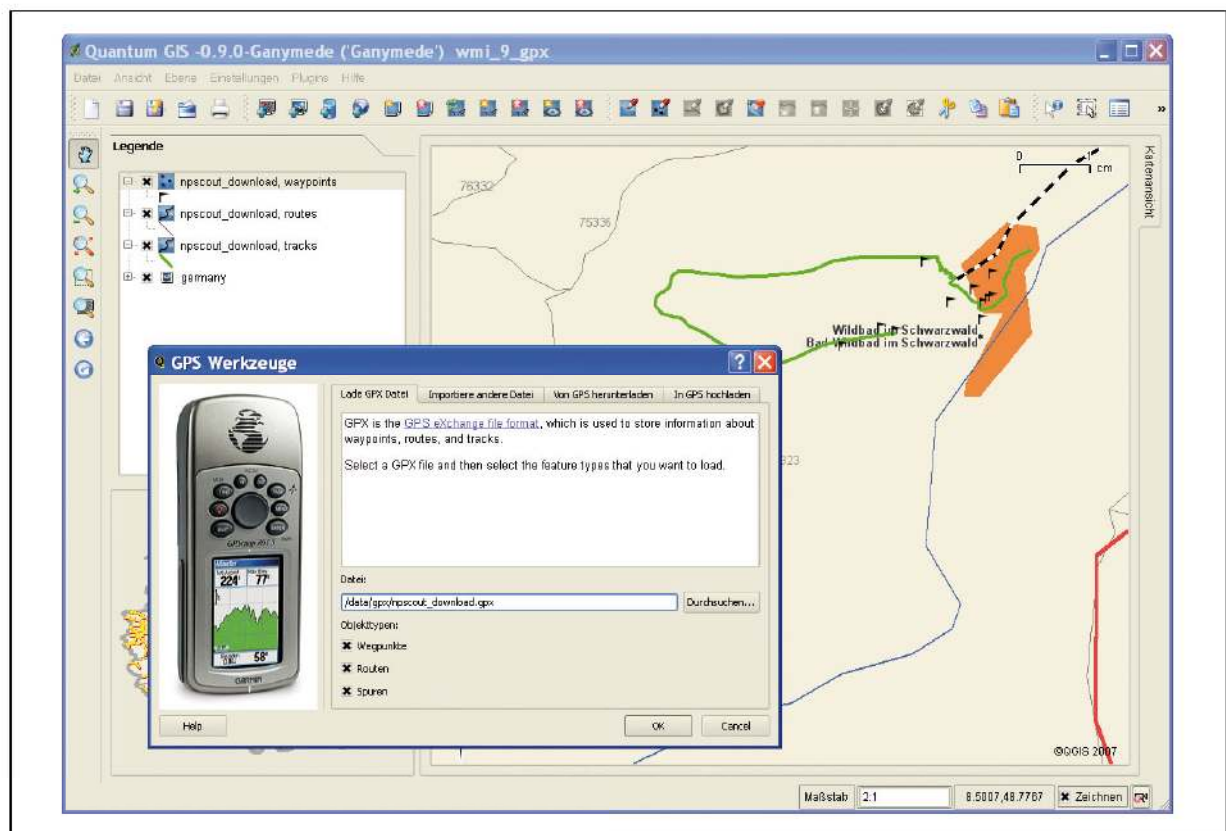


Abbildung 9-5: Laden einer GPX-Datei in Quantum GIS über das GPS-Werkzeug

Sie können auch Daten von anderen GPS-Empfängern in Quantum GIS visualisieren. Dabei müssen die Daten über die Freie Software *GPSTabel* (<http://www.gpsbabel.org>) erst in das Austauschformat GPX überführt werden.

Damit Sie Ihre Route im Zusammenhang mit anderen Daten anschauen können, laden Sie den folgenden WMS in Ihr Projekt:

<http://wms.wherogroup.com/cgi-bin/germany.xml?Request=GetCapabilities&Version=1.1.1&Service=WMS>

Dies erfolgt über den Button *WMS Ebene hinzufügen*, die Taste W oder den entsprechenden Button. Nun müssen Sie den Dienst unter *Serververbindungen* veröffentlichen und über den Button *Neu* auf den Dienst verweisen. Geben Sie die Verbindungsinformationen über den GetCapabilities-Request des Diensts an. Verbinden Sie sich mit dem Dienst und wählen Sie die gewünschten Ebenen aus der Liste der Layer aus. Sie müssen nun noch das Koordinatenreferenzsystem auswählen, in dem die Karte angefordert werden soll. Schauen Sie sich dazu auch die Beschreibung aus Kapitel 8 an.

Wenn Sie die GPX-Daten in Quantum GIS bearbeitet haben, können Sie diese über das GPS-Werkzeug auch wieder in ein GPS-Gerät laden, und zwar im GPS-Werkzeug im Register *In GPS* hochladen.

Laden von Punkten aus Textdateien

In Quantum GIS können auch Punkte aus einer Textdatei geladen werden. Dabei sind einige Details zu beachten, und die Textdatei muss einen bestimmten Aufbau haben:

- In der ersten Zeile müssen die Spaltennamen vorliegen.
- Die Textdatei muss über x- und y-Koordinaten verfügen. Die Bezeichnungen der Spalten kann dabei beliebig sein.
- Die x- und y-Koordinaten müssen als Zahlen angegeben werden. Das Koordinatensystem kann dabei beliebig sein. Das Dezimaltrennzeichen kann ein Punkt oder ein Komma sein.
- Als Trennzeichen kann ein beliebiges Zeichen mit Ausnahme von Anführungszeichen (") verwendet werden.

Ein Beispiel hierfür ist die Textdatei *ortschaften.csv*:

```
gid|name|x|y
3|Sylt|8.3098525939834|54.9165916474873
5|Flensburg|9.4312429380207|54.7838744815123
22|Schleswig|9.55522822724063|54.5264511335158
33|Fehmarn|11.195258113133|54.4376640036691
47|Kiel|10.1207542669195|54.3253593665031
51|Stralsund|13.0751771783027|54.3112259071443
52|Rendsburg|9.66611290035299|54.3105506767494
68|Heide|9.08711241211079|54.195926636856
83|Rostock|12.0867728938417|54.0945892499222
84|Greifswald|13.3912439538919|54.0897445402343
```

Laden Sie die Datei *ortschaften.csv* der Beispieldaten. Die Datei enthält Ortschaften von Deutschland. Zum Laden der Datei muss im Plugin-Manager die Erweiterung *Lade kommaseparierte (CSV) Textebene* aktiviert sein. Sie können den Dialog zum Laden einer Textdatei über *Plugin → Getrennter Text* öffnen. In dem Fenster wählen Sie die Textdatei aus. Geben Sie anschließend das Trennzeichen im Textfeld *Trennzeichen* an und klicken Sie auf den Button *Parse*. Sie können nun die x- und y-Spalte auswählen und der Ebene einen Namen geben. Die Daten werden über einen Klick auf den OK-Button in das Projekt geladen. Diesen Vorgang sehen Sie in Abbildung 9-6.

Wählen Sie ein geeignetes Symbol für die Darstellung der Ortschaften aus und versehen Sie diese mit einer Beschriftung. Diese Konfiguration können Sie über die Ebeneneinstellungen vornehmen. Das Dialogfenster öffnen Sie über einen Doppelklick auf die Ebene.

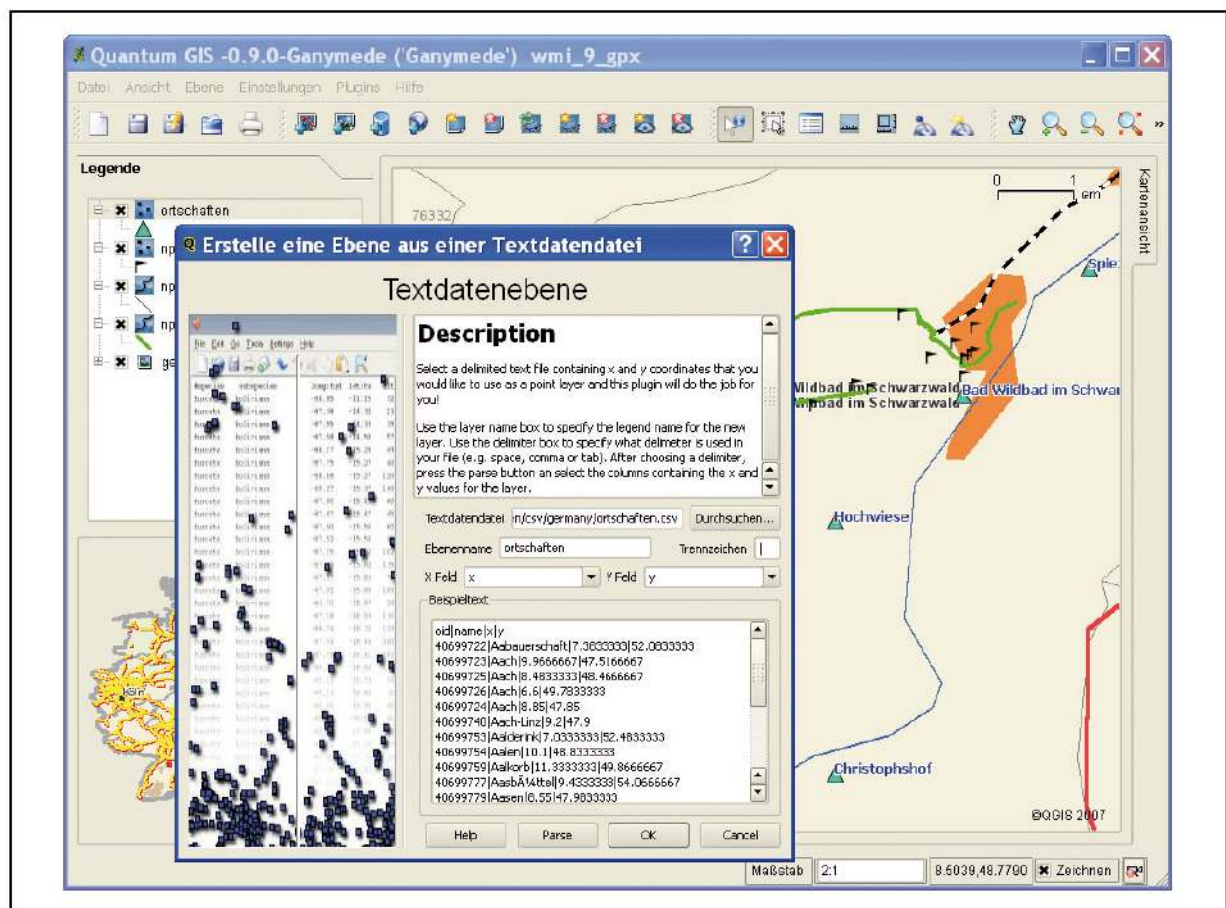


Abbildung 9-6: Laden von Punktdaten aus einer Textdatei

Georeferenzieren von Rasterdaten

In diesem Abschnitt lernen Sie zum Abschluss nun noch das Quantum GIS-Plugin *Georeferenzieren* kennen. Georeferenzierung bezeichnet die Zuweisung raumbezogener Referenzinformationen zu einem Geodatensatz. Wenn Sie beispielsweise eine Karte einscannen, erhalten Sie eine Rasterdatei, die aber noch nicht über einen räumlichen Bezug verfügt. Diesen können Sie über eine Georeferenzierung herstellen und somit Ihre Karte an der richtigen Stelle positionieren.

Um Ihre Rasterdaten zu georeferenzieren, können Sie das Quantum GIS-Plugin nutzen. Dieses Plugin erzeugt wld-Dateien für die georeferenzierte Datei. Die wld-Datei trägt dabei den gleichen Namen wie die Rasterdatei, verfügt aber lediglich über die Dateiendung *wld*. Die Georeferenzierung kann über zwei Wege erfolgen:

- Sie wählen Punkte in der Rasterdatei aus, zu denen Sie Koordinaten aus der Hauptkarte übernehmen.
- Sie geben die Koordinaten der Punkte manuell ein.

Das Plugin berechnet anschließend Parameter und schreibt diese in eine wld-Datei. Beachten Sie, dass die Berechnung umso genauer wird, je mehr Koordinaten Sie eingeben.

Die wld-Datei muss sich im selben Verzeichnis befinden wie die Rasterdatei. Die Endung der Datei kann je nach Rasterdatenformat beispielsweise auch *jgw* oder *tfw* sein. Schauen Sie sich die Datei *test_dop/dop_50cm.tfw* einmal im Editor an. Es handelt sich um eine einfache Textdatei, die folgende Zeilen enthält:

```
0.500000
0.000000
0.000000
-0.500000
2582500.250000
5523499.750000
```

Die Datei enthält die folgenden Informationen:

- 1. Zeile: Pixelbreite in Karteneinheiten; diese berechnet sich aus:
 $(\text{maxx} - \text{minx}) / (\text{Kartenbreite in Pixeln})$
- 2. Zeile: Rotationsfaktor
- 3. Zeile: ein weiterer Rotationsfaktor
- 4. Zeile: Pixelhöhe in Karteneinheiten (die Pixelhöhe ist meist als negativer Wert anzugeben, da sich das Bild in der Regel von oben nach unten aufbaut, die Software aber von der unteren linken Ecke ausgeht)
- 5. Zeile: x-Koordinate des Zentrums des oberen linken Pixels des Bilds
- 6. Zeile: y-Koordinate des Zentrums des oberen linken Pixels des Bilds

Im GeoPortal.rlp-Wiki liegt eine Beschreibung der WebGIS.rlp-CD vor. Hier finden Sie unter *Georeferenzierung* ein Video, das die Georeferenzierung mit Quantum GIS anschaulich vermittelt. Den Link finden Sie unter:

<http://www.geoportal.rlp.de/mediawiki/index.php/WebGIS.rlp>

Öffnen Sie Ihr bestehendes Projekt, in dem die Testdaten der WebGIS.rlp-CD visualisiert werden. Laden Sie das Orthophoto (*test_dop/dop_50cm.tif*) und die Datei *test_bplan/B_Plan_Morbach_Hundheim_Auf_der_Noh_Geltungsbereich.shp* und zoomen Sie in den Bereich von Hundheim. Geben Sie in den Projekteinstellungen EPSG:31466 und Meter als Einheiten an.

Anhand eines Bauleitplans soll die Georeferenzierung veranschaulicht werden. Öffnen Sie über *Plugin* → *Georeferenzierer* das Dialogfenster. Wählen Sie nun zunächst den Bauleitplan *test_bplan/B_Plan_Morbach_Hundheim_Auf_der_Noh_Plan.jpg* aus, der georeferenziert werden soll. Es handelt sich dabei um einen Bauleitplan für Hundheim. Ordnen Sie die Fenster für eine bessere Übersicht nebeneinander an (Abbildung 9-7).

Zoomen Sie im Georeferenzierer näher in den Bauleitplan hinein. Wählen Sie einen Bereich aus, den Sie zunächst georeferenzieren wollen. Zoomen Sie im Kartenfenster von Quantum GIS in den entsprechenden Bereich. Aktivieren Sie im Georeferenzierer den Button *Addiere Punkt*. Wählen Sie anschließend einen markanten Punkt im Bauleitplan aus. Es öffnet sich das Dialogfenster *Kartenkoordinaten eingeben*. Hier können Sie entweder eine Koordinate manuell über die Textfelder eingeben, oder Sie klicken auf

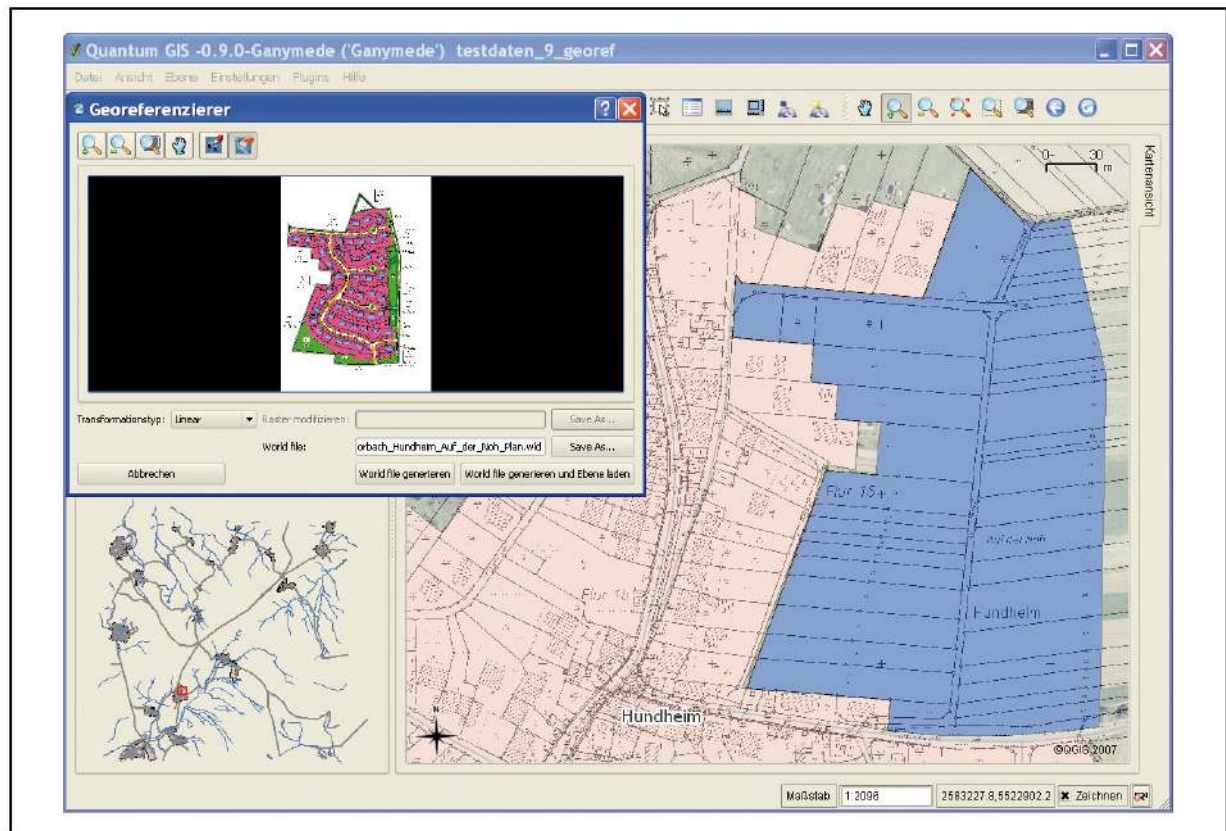


Abbildung 9-7: Laden des Bauleitplans in Quantum GIS über den Georeferenzierer

den Button *Aus Karte* und wählen anschließend per Klick in der Referenzkarte den korrespondierenden Punkt aus.

Markieren Sie auf die Weise ausreichend viele Punkte, die sich am besten über den gesamten Plan verteilen. Je mehr Punkte Sie markieren, desto genauer wird die Georeferenzierung (Abbildung 9-8).

Nachdem Sie den letzten Punkt verortet haben, klicken Sie auf den Button *Auf Ebene zoomen*. Wählen Sie als Transformationstyp *Helmert* aus. Klicken Sie anschließend auf den Button *World file generieren und Ebene laden*, um die Georeferenzierung abzuschließen (Abbildung 9-9).

Eine neue Rasterdatei mit einer passenden wld-Datei wird angelegt. Das Ergebnis sehen Sie in Abbildung 9-10.

In wenigen Schritten haben Sie ein bzw. zwei eigene Projekte aufgebaut. Sie haben dabei gesehen, dass Sie auf einfachem Weg eigene Daten erheben können. Oft bietet es sich aber an, bestehende Dienste, die frei zur Verfügung stehen, zu nutzen. Hier haben Sie die Möglichkeiten kennengelernt, die über WMS zur Verfügung stehen. Liegen Ihnen Rasterdaten vor, die noch keinen räumlichen Bezug haben, können Sie diese einfach über Quantum GIS positionieren. Sicherlich stehen Ihnen noch viele weitere Funktionalitäten im Umgang mit Geodaten zur Verfügung, sodass dieses Kapitel nur als Einstieg verstanden werden kann.

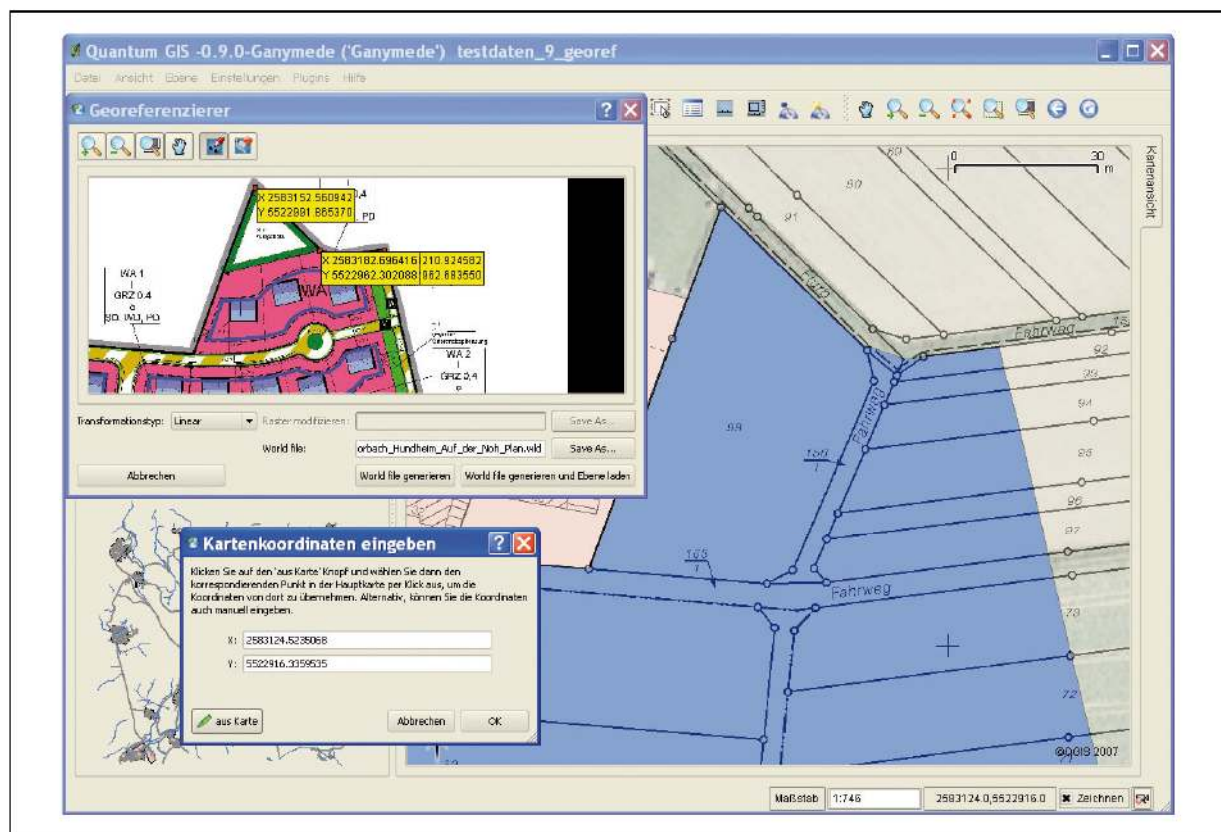


Abbildung 9-8: Zuweisung von Punkten des Bauleitplans zu Koordinaten in der Referenzkarte

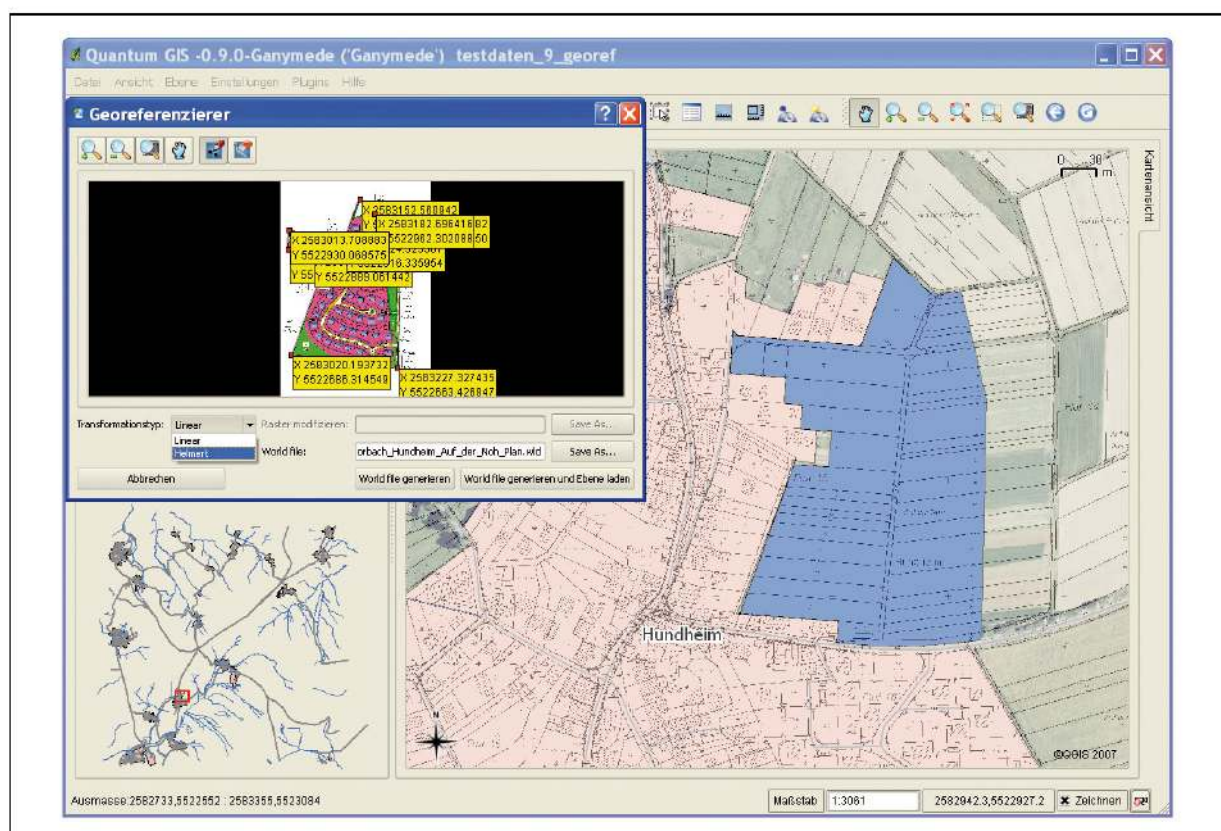


Abbildung 9-9: Auswahl des Transformationstyps Helmert

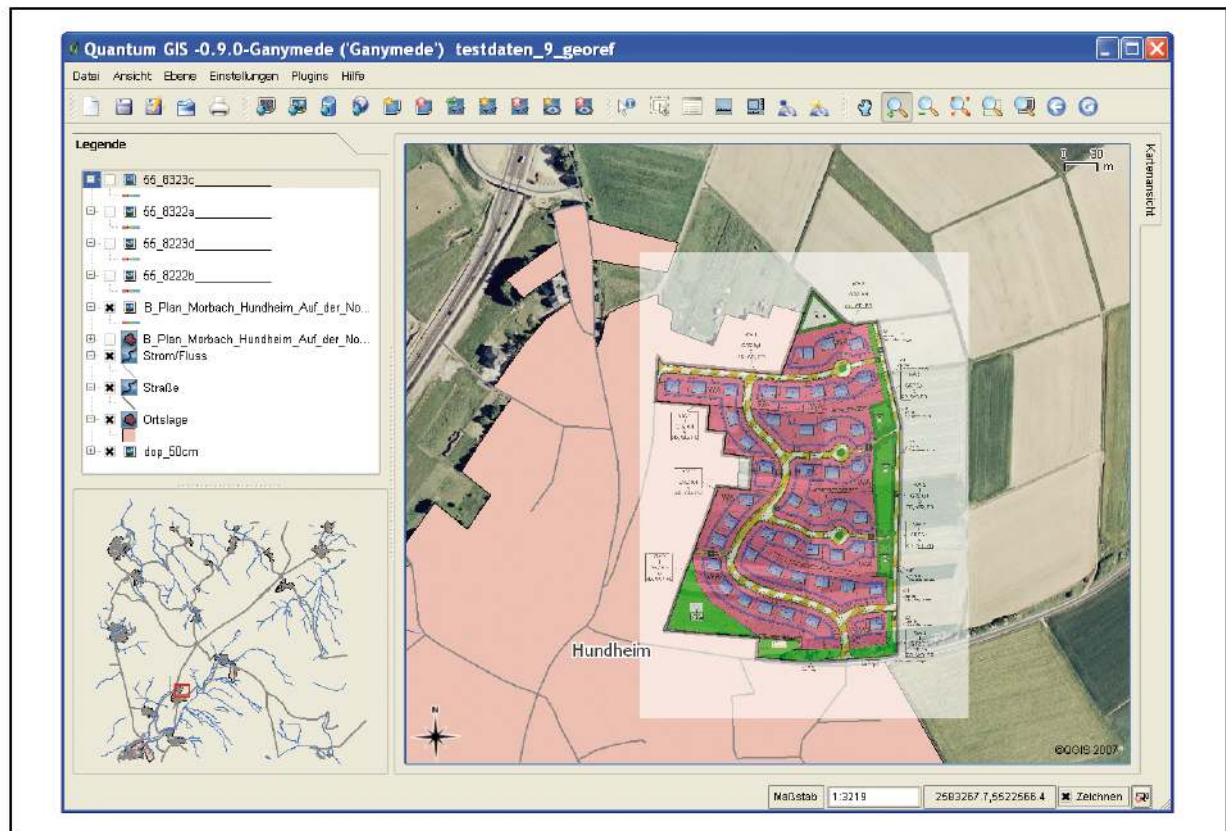


Abbildung 9-10: Ansicht des georeferenzierten Bauleitplans im fertigen Projekt

Erstellen statischer Karten – erster Einstieg in die MapServer-Mapdatei

Die Integration von Karten in eine Internetseite oder eine interaktive Website ist einfacher, als Sie denken. Oft reicht MapServer für die Umsetzung schon aus. Dieses Kapitel zeigt Ihnen, wie Sie MapServer einsetzen können, um Ihre Daten so schnell wie möglich auf eine Website zu bekommen. Selbst wenn Sie eigentlich interaktive Karten erstellen möchten, kann die Verwendung von MapServer zur Generierung statischer Karten gelegentlich hilfreich sein. Außerdem ist es eine gute Möglichkeit, das Programm besser kennenzulernen.

MapServer ist vor allem für seine Web-Mapping-Fähigkeiten bekannt. Dem Programm liegen aber auch noch einige eigenständige kommandozeilenbasierte Werkzeuge bei, mit denen Sie statische Karten für Websites, für Dokumente oder zum Verschicken per E-Mail erstellen können. Zu Beginn dieses Kapitels werden die einzelnen Werkzeuge kurz vorgestellt. Danach werden die Aufgaben, die am häufigsten beim Erstellen einer Karte auftreten, mithilfe einer einfachen MapServer-Konfigurationsdatei (Mapdatei) erläutert.

Im Folgenden wird von Grund auf beschrieben, wie Sie eine Mapdatei auf klassische Weise von Hand mit einem Texteditor erstellen. Dies ist der bevorzugte Weg, um ein angemessenes Verständnis für die Funktionsweise einer Mapdatei zu entwickeln. Sollten Sie im Arbeitsalltag mit einer Vielzahl von Mapdateien umgehen müssen, empfiehlt sich ein Blick auf die GPL-lizenzierte Software *MapStorer*, mit der Sie über eine komfortable Weboberfläche Ihre MapServer-Projekte verwalten können, inklusive Import bereits vorhandener Mapdateien (siehe <http://www.mapstorer.org>).

In Kapitel 4 ging es bereits um das Kompilieren und die Installation von MapServer. Kapitel 11 wird sich auf die Verwendung von MapServer als dynamischen Kartendienst konzentrieren, und in Kapitel 12 werden Sie MapServer als OGC WMS- bzw. WFS-Client und -Server kennenlernen.

MapServer-Hilfsprogramme

Zu MapServer gehört wesentlich mehr als nur das *mapserv*-Programm. Die folgenden Programme werden gemeinsam mit *mapserv* kompiliert und gehören zum MapServer-Installationspaket. MapServer ist mehr als ein Web-Mapping-Programm. Die beiliegenden Kommandozeilenwerkzeuge können als eigenständige Programme zum Erstellen von Karten verwendet werden, ohne dass hierfür ein Webserver gebraucht wird. Die Details zu den einzelnen Programmen finden Sie im jeweiligen Abschnitt.

legend

Erzeugt eine Legendengrafik. Diese basiert auf den Angaben, die zum Zeichnen der Ebenen in einer Mapdatei definiert wurden.

scalebar

Erzeugt das Bild einer Maßstabsleiste. Die Darstellung der Maßstabsleiste basiert auf den Angaben in der Mapdatei (Projektion, Karteneinheiten).

shp2img

Erzeugt, basierend auf einer Mapdatei, eine Karte. Der Darstellungsbereich und die gewünschten Ebenen (Layer) können gewählt werden. Die Optionen sind so vielseitig, dass ein einfaches Desktop-GIS-Programm nur mit diesem Werkzeug in der Lage wäre, Karten zu erzeugen. *shp2img* ist ähnlich wie *shp2pdf*, das die Ausgabe der Karte im PDF-Format ermöglicht.

shp2pdf

Erzeugt als Ausgabedatei ein PDF, das eine auf der angegebenen Mapdatei basierende Karte enthält. Hierfür muss MapServer allerdings mit PDF-Unterstützung kompiliert worden sein. *shp2pdf* hat eine ähnliche Syntax wie *shp2img*. Voraussetzung ist, dass der Mapserver mit PDF-Unterstützung (`--with-pdf=DIR`) kompiliert wurde.

shptree

Erweitert eine Shape-Datei um einen (Quadtree-basierten) Index, der beim Erstellen von Karten aus großen Shape-Dateien eine Performancesteigerung bewirkt.

tile4ms

Erzeugt eine Übersichts-Shape-Datei, die Verweise auf die tatsächlichen Dateien und deren Umgebungsrechteck enthält. Die ausgegebene Shape-Datei kann von MapServer verwendet werden, um eine große Anzahl weiterer Shape-Dateien (oder andere unterstützte Datenformate) zu referenzieren. So müssen die Daten nicht in einer Datei zusammengefügt werden, und es muss auch nicht für jede Datei eine eigene Ebene in der Mapdatei angelegt werden. Stattdessen enthält der Parameter `TILEINDEX` die Informationen darüber, welche Daten verfügbar sind und wo diese sich räumlich befinden. Dieser Weg ist sehr effizient, weil die Umgebungsrechtecke als Indizes verwendet werden, um nur die Kacheln zu finden, die sich im derzeitigen Kartenausschnitt befinden. Zusätzlich ist es empfehlenswert, *shptree* für die erzeugte Übersichts-Shape-Datei selbst noch einmal auszuführen. Die Werkzeuge *ogrindex* und *gdaltindex*, die zum GDAL/OGR-Paket gehören, sind ähn-

lich. Diese Werkzeuge sind sehr zu empfehlen, weil sie nicht nur Shape-Dateien indizieren können (weitere Informationen unter <http://www.gdal.org/>).

sortshp

Sortiert eine Shape-Datei nach einem bestimmten Feld der Tabelle und ordnet die physikalische Reihenfolge der Objekte in der Datei neu. Dies kann hilfreich sein, wenn die Reihenfolge für die Darstellung wichtig ist, oder zur Abfrageoptimierung.

Aus historischen Gründen haben manche der Hilfsprogramme das Präfix shp. Ursprünglich waren diese Programme für den Umgang mit Shape-Dateien gedacht. Inzwischen arbeiten sie mit einer MapServer-Mapdatei, greifen auf verschiedene Datenformate zu und geben ein Bild aus. Daher ist das Präfix shp eher irreführend, weil Shape-Dateien nicht unbedingt beteiligt sein müssen. MapServer kann mittlerweile auf Dutzende verschiedener Datenformate zugreifen.

Die Kommandozeilenprogramme im Einsatz

Wenn MapServer als Web-Mapping-Programm eingesetzt wird, werden zahlreiche Bilddateien dynamisch erzeugt. Das Kartenbild ist nur eine dieser Dateien. Daneben kann MapServer ein Bild einer Maßstabsleiste, Referenzkarte und Legende erzeugen. Die Kommandozeilenprogramme helfen Ihnen dabei, diese Grafiken manuell (oder über die Skriptsprache Ihrer Wahl) zu erstellen. Die folgenden Abschnitte sollen Ihnen anhand einiger einfacher Beispiele ein Gefühl für die Verwendung einiger Werkzeuge geben. Wir bedienen uns hier der Hilfsprogramme einer typischen MapServer-Installation, wie in Kapitel 4 beschrieben.

Bereitstellung von Basisdaten

In diesem Beispiel wird eine einfache Weltkarte mit den Polygonen der einzelnen Länder verwendet. Die Daten stehen auf der FreeGIS-Website (<http://www.freegis.org>) unter der Adresse http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.1_simpl.tar.gz zum Download.



Die hier verwendete Version enthält vereinfachte Ländergrenzen und hat etwa eine Größe von 3,7 MByte. Die Ländergrenzen liegen auch als komplexere Daten vor. Dabei ist die Datei etwa 30 MByte groß.

Sie können für die Übungen in diesem Kapitel auch die komplexeren Daten verwenden. Der Aufwand, die Daten darzustellen, ist für MapServer dabei aber wesentlich höher. Um effektiv mit den Daten arbeiten zu können, müssen allerdings ein paar Optimierungen vorgenommen werden. Mit den komplexeren Daten werden die Übungen etwas länger dauern als mit den vereinfachten Polygonen. Wenn Sie die Möglichkeiten von MapServer genauer untersuchen möchten, sind die komplexeren Daten natürlich ein guter Ausgangspunkt. Sie können unter der Adresse http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.1.tar.gz heruntergeladen werden.

Als Erstes sollten Sie die Daten überprüfen. Hierfür benutzen wir das in Kapitel 6 vorgestellte Werkzeug ogrinfo. Dieses Programm gehört zwar eigentlich nicht zu MapServer, ist aber für den Anfang ein wertvolles Hilfsmittel:

Beispiel 10-1: Untersuchung der Shape-Datei mithilfe von ogrinfo

```
> ogrinfo countries_simpl.shp -al -summary
INFO: Open of 'countries_simpl.shp'
using driver 'ESRI Shapefile' successful.
Layer name: countries_simpl
Geometry: Polygon
Feature Count: 3901
Extent: (-179.999900, -89.999900) - (179.999900, 83.627357)
Layer SRS WKT:
(unknown)
gid: Integer (11.0)
cat: Integer (11.0)
fibs: String (2.0)
name: String (255.0)
f_code: String (255.0)
total: Integer (11.0)
male: Integer (11.0)
female: Integer (11.0)
ratio: Real (24.15)
```

Dieses Beispiel zeigt Schlüsselinformationen, die vor der Erstellung der MapServer-Mapdatei wichtig sind.

Geometry: Polygon

Die vorliegende Shape-Datei enthält Polygone. MapServer kann die Polygone mit einer Flächenfüllung darstellen oder nur den Umring zeichnen. Würden in der Shape-Datei Linien vorliegen, wäre eine Flächenfüllung nicht möglich.

Feature Count: 3901

Feature Count gibt aus, wie viele Polygone in dieser Datei enthalten sind (genau 3.901). Steht hier, dass keine Objekte vorliegen, ist sicherlich etwas mit der Datei nicht in Ordnung.

Extent: (-179.999900, -89.999900) - (179.999900, 83.627357)

EXTENT zeigt, dass die Daten nahezu den gesamten Globus abdecken. Das erste Zahlenpaar bezieht sich auf die südwestliche Ausdehnung, während das zweite Zahlenpaar die nordöstliche Ausdehnung der Daten angibt. Die Mapdatei sollte so konfiguriert werden, dass die Karte den gesamten Globus anzeigt. Hierfür verwenden Sie die Gradangaben -180, -90 und 180, 90.

Layer SRS WKT:
(unknown)

Die Shape-Datei enthält keine Angaben zum Koordinatenreferenzsystem (Spatial Reference System – SRS). Wenn Sie allerdings bedenken, dass die Ausdehnung in Dezimal-

grad angegeben wurde, können Sie davon ausgehen, dass die Datei unprojizierte geografische Koordinaten enthält.

```
gid: Integer (11.0)
cat: Integer (11.0)
fibs: String (2.0)
name: String (255.0)
f_code: String (255.0)
total: Integer (11.0)
male: Integer (11.0)
female: Integer (11.0)
```

Dies ist eine Liste der *Attribute* (oder Felder), die ebenfalls zur Shape-Datei gehören. Diese Angaben sind hilfreich, wenn Sie die Karte mit Beschriftungen versehen oder die Daten klassifizieren wollen.

Mit diesen Informationen können Sie nun eine einfache Mapdatei erstellen.



Vielleicht wollen Sie einen kurzen Blick darauf werfen, welche Länder in der Shape-Datei vorliegen. Mit ein paar einfachen Textverarbeitungsbeehlen können Sie die Ausgaben von ogrinfo so filtern, dass die wichtigen Informationen übrig bleiben. Genauer hierzu finden Sie in Kapitel 6. Hier ein kurzes Beispiel:

```
> ogrinfo countries_simpl.shp -al | grep name | sort | uniq
Layer name: countries_simpl
name: String (255.0)
name (String) = Afghanistan
name (String) = Albania
name (String) = Algeria
name (String) = American Samoa
continued...
```

Erstellen einer einfachen Mapdatei für den UMN MapServer

Die Mapdatei ist das zentrale Objekt einer MapServer-Anwendung. Sie legt fest, wie die Karte aussehen soll, welche Größe sie haben soll und welche Ebenen angezeigt werden sollen. Es existiert eine Vielzahl von Einstellungsmöglichkeiten. Um den Aufbau der Mapdatei kennenzulernen, sollten Sie unbedingt die Referenzdokumente auf der MapServer-Website (<http://mapserver.gis.umn.edu/doc/mapfile-reference.html>) lesen. Diese Dokumente geben einen sehr guten Überblick über den Aufbau der Mapdatei.

Eine deutschsprachige Einführung in den UMN MapServer bieten die folgenden Links:

<http://umn.mapserver.ch/>

<http://www.umn-mapserver.de/>

http://www.mapmedia.de/dokumente/umn_signaturen_howto/index.html

Bei den Mapdateien handelt es sich um einfache Textdateien, die eine hierarchische (verschachtelte) Objektstruktur verwenden, um die Einstellungen für die Karte zu definieren. Die Mapdatei besteht aus verschiedenen Objekten (Blöcken). Jedes Objekt

beginnt mit einem bestimmten Schlüsselwort und endet mit dem Wort END. Objekte können ihrerseits wiederum Objekte enthalten und über Parameter verfügen. Beispielsweise kann die Definition einer Ebene Objekte für die Darstellung oder die Beschriftungen dieser Ebene enthalten. Beispiel 10-2 zeigt die einfachsten Definitionen einer Mapdatei mit nur 15 Zeilen Text.

Beispiel 10-2: Eine einfache Mapdatei mit einer Ebene, in der der Beginn und das Ende der Objekte zu sehen ist

```
MAP                # Beginn des MAP-Objekts
  SIZE 600 300      # Größe des Kartenbilds
  EXTENT -180 -90 180 90 # Darstellungsbereich
  LAYER             # Beginn des LAYER-Objekts
    NAME countries  # Name des Layer-Objekts
    TYPE POLYGON    # Typ, der verwendet werden soll
    STATUS DEFAULT  # Status DEFAULT - Layer wird immer gezeichnet
    DATA countries_simpl # Datenquelle
    CLASS           # Beginn des CLASS-Objekts
      STYLE         # Beginn des STYLE-Objekts
        OUTLINECOLOR 100 100 100 # Umrangungsfarbe, Angabe als RGB-Wert
      END           # Ende des STYLE-Objekts
    END            # Ende des CLASS-Objekts
  END              # Ende des LAYER-Objekts
END                # Ende des MAP-Objektes und der Mapdatei
```

Diese Mapdatei enthält nur vier Objekte:

- Ein MAP-Objekt, das projektweit geltende Einstellungen enthält.
- Ein LAYER-Objekt, das eine Ebene (Layer) definiert, die in der Karte angezeigt werden soll.
- Ein CLASS-Objekt, das eine Klasse definiert, über die bestimmte Objekte dargestellt werden.
- Ein STYLE-Objekt, das festlegt, wie die Objekte, die dieser Klasse angehören, dargestellt werden sollen.

Abbildung 10-1 zeigt eine vereinfachte grafische Darstellung der Objekthierarchie mit Angabe einiger Objekteigenschaften. In Wirklichkeit ist alles etwas komplexer. Schauen Sie sich die Mapfile Reference-Website (<http://mapserver.gis.umn.edu/docs/reference/mapfile>) an. Hier finden Sie alle Objekte, und die möglichen Eigenschaften der Objekte werden kurz beschrieben.

Die Objekte sind ineinander verschachtelt. Jedes Objekt besitzt dabei eigene Eigenschaften. Die Einstellungen SIZE (Größe) und EXTENT (Darstellungsbereich der Karte) sind Eigenschaften des MAP-Objekts. Das Objekt LAYER besitzt die Parameter NAME, TYPE, STATUS und DATA. Für das Objekt CLASS wurden in diesem Beispiel keine eigenen Einstellungen angegeben, es enthält aber das Objekt STYLE, das die Darstellung in der Karte festlegt. In unserem Beispiel wird über OUTLINECOLOR die Umrangungsfarbe der Polygone als RGB-Wert definiert. RGB ist die Abkürzung für die Grundfarben Rot, Grün und

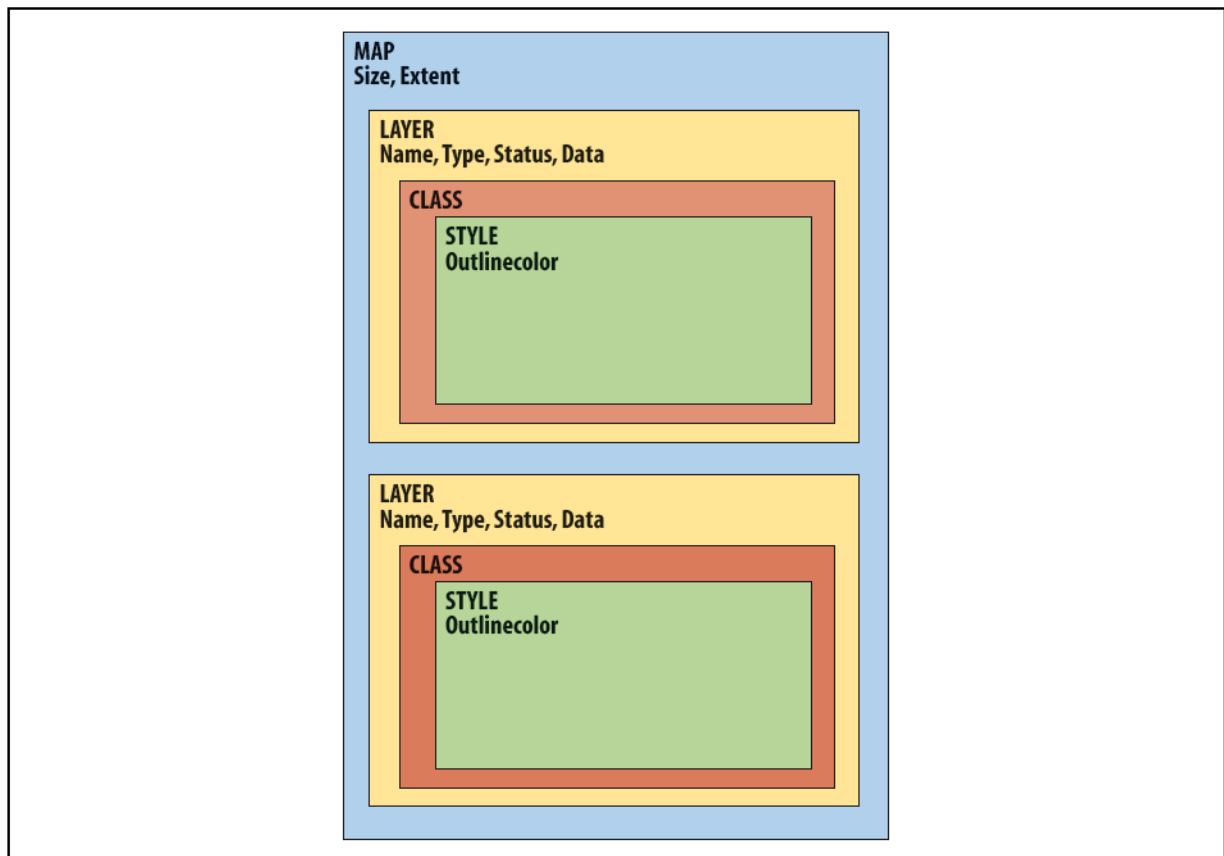


Abbildung 10-1: Einige der wichtigsten in einer MapServer-Mapdatei verwendeten Objekte

Blau (englisch red, green und blue). Die Farbwerte der Grundfarben werden zwischen 0 und 255 angegeben. Durch die Kombination der drei Farbwerte entstehen neue Farben. Hier ein paar Beispiele:

```

0 0 0      (Schwarz)
255 255 255 (Weiß)
255 0 0    (Rot)
124 124 124 (Grau)
  
```

Im Objekt LAYER gibt es den Parameter DATA, über den die Daten angegeben werden, die über diese Ebene dargestellt werden sollen. In den folgenden Beispielen stammen die Daten aus einer Shape-Datei mit dem Namen *countries_simpl*. In den Beispielen wird davon ausgegangen, dass die folgenden Befehle im gleichen Verzeichnis ausgeführt werden, in dem sich auch die Mapdatei befindet. Weitere Informationen zu anderen Datenformaten und ihre Verwendung in MapServer finden Sie in Anhang B.

Regeln und Tipps für den Aufbau der Mapdatei

Sie können Mapdateien mit Kommentaren versehen, indem Sie dem Kommentar eine Raute (#) voranstellen. Alles was nach diesem Symbol auf der gleichen Zeile steht, wird von MapServer ignoriert. Beispiel 10-2 verwendet beispielsweise Kommentare, um den Anfang und das Ende der Objekte zu kennzeichnen. Gehen Sie aber sparsam mit Kommentaren um, da MapServer die Mapdatei bei jedem Aufruf erneut ausliest.

Rücken Sie die Zeilen entsprechend der Objekthierarchie ein. Das Einrücken der Zeilen ist nicht vorgeschrieben, aber es macht die Mapdatei leichter lesbar. Diese Struktur wird hierarchisch behandelt. Der Beginn eines neuen Objekts wird eingerückt, und nach dem Ende des Objekts wird wieder ausgerückt.

Die Parameter und Objektnamen in der Mapdatei sind nicht case-sensitive, d.h. zwischen Groß- und Kleinschreibung wird nicht unterschieden. Allerdings wird empfohlen, Objektnamen (wie MAP) und andere Parameter (DATA, COLOR usw.) in Großbuchstaben anzugeben. Die Werte der Parameter (Datei-, Ebenennamen) sollten dagegen in Kleinbuchstaben angegeben werden. Auch wenn dies teilweise meine persönlichen Vorlieben sind, kann diese Konvention die Lesbarkeit der Mapdatei deutlich erhöhen.

Die Attribute, die sich auf die Felder der Datenquelle beziehen, sind dagegen case-sensitive. Geben Sie hier die Werte entsprechend den Einstellungen der Datenquelle an. Auch Pfadangaben sollten case-sensitive angegeben werden.

Die Mapdatei liegt als Datei in Ihrem Dateisystem vor. Mapdateien müssen mit der Dateiendung `.map` versehen werden. Die Mapdatei aus Beispiel 10-2 hat beispielsweise den Namen *global.map*.

Pfadangaben in der Mapdatei können absolut oder relativ angegeben werden. Relativ bedeutet dabei relativ zur Mapdatei

Manche Werte müssen mit Anführungszeichen umgeben werden. Hierbei können einfache und doppelte Anführungszeichen verwendet werden. In vielen Fällen ist das Quoting aber optional. Ein gutes Beispiel hierfür sind Dateinamen: Sie können entweder in einfachen, doppelten oder auch ohne Anführungszeichen angegeben werden. Der Wert des Parameters SYMBOL muss dagegen mit Anführungszeichen versehen werden.

Schauen Sie sich dazu auch die Hinweise der MapServer-Dokumentation an:

<http://mapserver.gis.umn.edu/docs/reference/mapfile/introduction>

Erzeugen Sie Ihr erstes Kartenbild

Der MapServer-Befehl `shp2img` übernimmt die Einstellungen aus einer Mapdatei und erzeugt daraus ein Bild, das die Karte enthält. Beispiel 10-3 und Beispiel 10-4 zeigen die Syntax sowie ein Beispiel dafür, wie der Befehl zusammen mit der Mapdatei *global.map* benutzt wird. Öffnen Sie ein Befehlsfenster oder eine Shell und führen Sie den Befehl `shp2img` aus. Beachten Sie dabei, dass zuvor der Verzeichnispfad zu dem Verzeichnis, in dem sich das Programm befindet, in der Umgebungsvariablen `$PATH` eingetragen werden muss. Ist dies der Fall, können Sie das Programm ohne Pfadangabe aufrufen. Im MS4W-Paket können Sie über die Datei `setenv.bat` die Variable `PATH` setzen.

Beispiel 10-3: Syntax für die Benutzung von shp2img

```
> shp2img
Syntax: shp2img -m [mapfile] [-o image] [-e minx miny maxx maxy]
        [-s sizex sizey] [-l layer1 "[layers2...] "]" [-i format] [-p n]
        [-all_debug n] [-map_debug n] [-layer_debug n]
```

Beispiel 10-3: Syntax für die Benutzung von *shp2img* (Fortsetzung)

- m mapfile: Map file to operate on - required.
- i format: Override the IMAGETYPE value to pick output format.
- o image: output filename (stdout if not provided)
- e minx miny maxx maxy: extents to render
- s sizex sizey: output image size
- l layers: layers to enable - make sure they are quoted and space separated if more than one listed.
- all_debug n: Set debug level for map and all layers.
- map_debug n: Set map debug level.
- layer_debug layer_name n: Set layer debug level.
- p n: pause for n seconds after reading the map

Im folgenden Beispiel werden lediglich zwei Optionen (-m und -o) verwendet:

```
> shp2img -m global.map -o mymap.png
```



In der Datei *global.map* wird davon ausgegangen, dass sich die Shape-Datei *countries_simpl* im gleichen Verzeichnis wie die Mapdatei befindet.

Dieses Beispiel erzeugt eine neue Datei mit dem Namen *mymap.png* (siehe Abbildung 10-2).



Abbildung 10-2: Unser erstes Kartenbild zeigt die Umringe der Länder der Welt

Nachdem das Bild angelegt wurde, können Sie es mit einem beliebigen Bildbetrachtungsprogramm ansehen. Webbrowser sind normalerweise in der Lage, die Formate PNG, JPEG, GIF und einige andere darzustellen.



PNG steht für das Grafikformat Portable Network Graphics. Beachten Sie, dass Sie bei der Verwendung von PNG oder GIF (Graphics Interchange Format) als Grafikformat transparente Bilder erzeugen können. Dies ermöglicht Ihnen, mehrere Kartenbilder zu überlagern.

Möglicherweise werden die standardmäßig von MapServer erzeugten Bilder nicht im PNG-Format erstellt. Das hängt davon ab, woher Ihre MapServer-Version stammt und mit welchen Optionen sie kompiliert wurde. Wenn Sie explizit festlegen wollen, dass die Ausgabedateien im PNG-Format gespeichert werden sollen, verwenden Sie die Option `-i`, wie hier:

```
> shp2img -m global.map -o mymap.png -i PNG
```

Sollten Sie die Fehlermeldung erhalten, dass ein bestimmtes Bildformat nicht verfügbar ist, oder es wird ein nicht lesbares Bild erzeugt, kann es sein, dass MapServer keine Bilder in diesem Format ausgeben kann. Sollte es also mit PNG nicht klappen, probieren Sie es einmal mit JPEG oder GIF als Ausgabeformat. Vergessen Sie dabei nicht, auch die Dateiendung anzupassen, zum Beispiel:

```
> shp2img -m global.map -o mymap.jpg -i JPEG
```

Weitere Tipps zur Wahl des richtigen Ausgabeformats finden Sie im letzten Abschnitt am Ende dieses Kapitels.

Die Karte um Beschriftungen erweitern

Mit nur wenigen Schritten können Sie Länder in der Karte zusätzlich mit Beschriftungen versehen. In Beispiel 10-4 sehen Sie, um welche Zeilen die Mapdatei hierfür erweitert werden muss.

Beispiel 10-4: Definition einer Beschriftung in der Mapdatei

```
MAP
SIZE 600 300
EXTENT -180 -90 180 90
LAYER
  NAME countries
  TYPE POLYGON
  STATUS DEFAULT
  DATA countries_simpl
  LABELITEM 'NAME'          # Beschriftungsspalte
  CLASS
    STYLE
      OUTLINECOLOR 100 100 100
    END
    LABEL                  # Start des Beschriftungsobjekts
      MINFEATURESIZE 40    # Mindestgroesse des Objekts in Pixeln
    END                    # Ende des Beschriftungsobjekts
  END
END
END
```


Das Objekt LAYER wurde um vier Zeilen erweitert.

```
LABELITEM 'NAME'
```

Diese Einstellung teilt MapServer mit, welches Attribut der Daten für die Beschriftung der Polygone verwendet werden soll. Jedes Polygon besitzt eine Reihe von Attributen, die für die Erzeugung von Beschriftungen genutzt werden könnten. Mit der Einstellung LABELITEM legen Sie fest, welches Attribut das sein soll. Wie Sie aus Kapitel 7 wissen, kann ogrinfo eine Liste der in der Datei verfügbaren Attribute ausgeben.

Sie können die Shape-Datei auch in Quantum GIS laden und über die Attributtabelle eine geeignete Beschriftungsspalte auswählen.

Es wird ein neues Objekt für die Beschriftung erzeugt:

```
LABEL  
  MINFEATURESIZE 40  
END
```

Die zweite Zeile ist zwar nicht unbedingt notwendig, macht die Karte aber besser lesbar. Es gibt viele Möglichkeiten, die Darstellung der Beschriftung durch MapServer zu beeinflussen. Mit der Option MINFEATURESIZE können Sie festlegen, wie groß ein Land mindestens sein muss (Breite in Pixeln), damit es eine Beschriftung erhält. Der Name kleinerer Länder wird also nicht angezeigt, weil er sowieso nicht lesbar wäre. Je nachdem, welchen Wert Sie für MINFEATURESIZE angeben, erhalten Sie recht unterschiedliche Ergebnisse. Probieren Sie einmal statt 40 Pixeln 20 oder sogar 10 als Grenzwert aus.

Ändern Sie die Zeile EXTENT, sodass ein kleinerer Ausschnitt der Welt dargestellt wird. Nun werden weitere Beschriftungen eingefügt, weil die einzelnen Länder größer angezeigt werden.

Führen Sie nach diesen Änderungen (siehe Beispiel 10-4) den Befehl shp2img erneut aus, wird eine Weltkarte mit Beschriftungen erzeugt, wie in Abbildung 10-3 gezeigt.

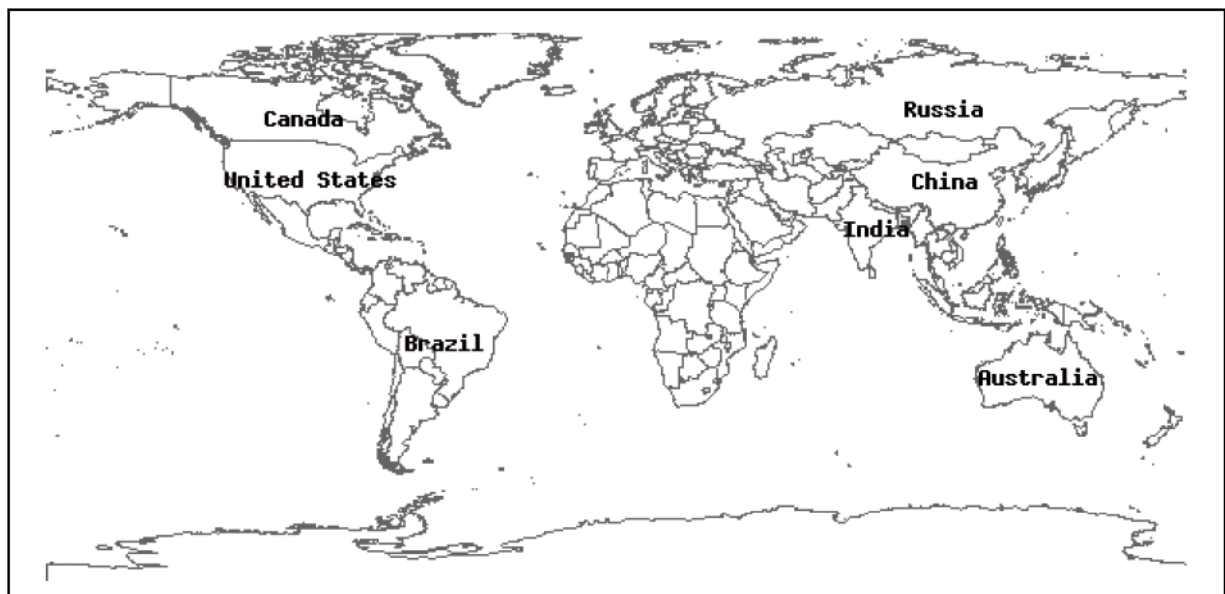


Abbildung 10-3: Eine Weltkarte, in der die Ländernamen als Beschriftungen hinzugefügt wurden

Ausgabe eines anderen Darstellungsbereichs

Das Werkzeug `shp2img` besitzt noch weitere Optionen, zum Beispiel können Sie über einen Parameter, den Sie dem Programm übergeben, einen eigenen Darstellungsbereich bestimmen. Das kann sehr praktisch sein, weil Sie hierfür keine Änderungen in der Mapdatei selbst vornehmen müssen. Stattdessen können Sie beim Aufruf von `shp2img` den gewünschten Bereich über die Option `-e` angeben.

Vor der Verwendung von `-e` sollten Sie jedoch festlegen, welcher Bereich der Karte dargestellt werden soll. Hierzu können Sie sich die Shape-Datei mit dem Desktop-GIS *Quantum GIS* anschauen, das Sie bereits in den Kapiteln 8 und 9 kennengelernt haben, und den gewünschten Bereich aus der Karte ablesen. Sie können `ogrinfo` auch zusammen mit dem Konvertierungswerkzeug `ogr2ogr` benutzen.

`ogrinfo` gibt die Gesamtausdehnung einer Ebene zurück, kennt aber nicht die Ausdehnungen eines bestimmten Objekts. Daher ist es sinnvoll, die Objekte, die Sie interessieren, zu extrahieren und in einer temporären Shape-Datei zu speichern. Sobald die gewünschten Informationen mit `ogrinfo` ausgelesen sind, können Sie die Datei wieder löschen.



Alternativ stehen Ihnen noch diverse weitere Programme zur Verfügung, über die Sie den gewünschten Darstellungsbereich ermitteln können. Hier sind beispielsweise *Quantum GIS*, *uDig*, *gvSIG*, *OpenJump* und *OpenEV* zu nennen. Laden Sie die Daten und ermitteln Sie die Koordinaten des gewünschten Ausschnitts über das jeweilige Programm.

In Beispiel 10-5 sehen Sie, wie beide Programme benutzt werden, um die Ausdehnungen eines bestimmten Landes zu ermitteln.

Beispiel 10-5: Auswahl des Objekts Bulgarien und Speichern in einer eigenen Shape-Datei

```
> ogr2ogr -where "name='Bulgaria'" bulgaria.shp countries_simpl.shp
> ogrinfo bulgaria.shp -al -summary
```

```
INFO: Open of 'bulgaria.shp'
using driver 'ESRI Shapefile' successful.
```

```
Layer name: bulgaria
Geometry: Polygon
Feature Count: 1
Extent: (22.371639, 41.242084) - (28.609278, 44.217640)
continued...
```

Mit der Option `-where` können Sie `ogr2ogr` anweisen, eine neue Shape-Datei zu erzeugen, die nur bestimmte Datensätze der ursprünglichen Datei enthält, z.B. Datensätze, bei denen `name='Bulgaria'` ist. Rufen Sie anschließend `ogrinfo` mit der neuen Datei auf, so können Sie anhand der Ausgaben bestimmen, welche Angaben Sie für das Attribut `EXTENT` in Ihrer Mapdatei machen müssen, um dieses Land heranzuzoomen. Mit diesen Informationen rufen Sie nun `shp2img` auf und geben den ermittelten `EXTENT` über die Option `-e` an. So erhalten Sie die Karte aus Abbildung 10-4.

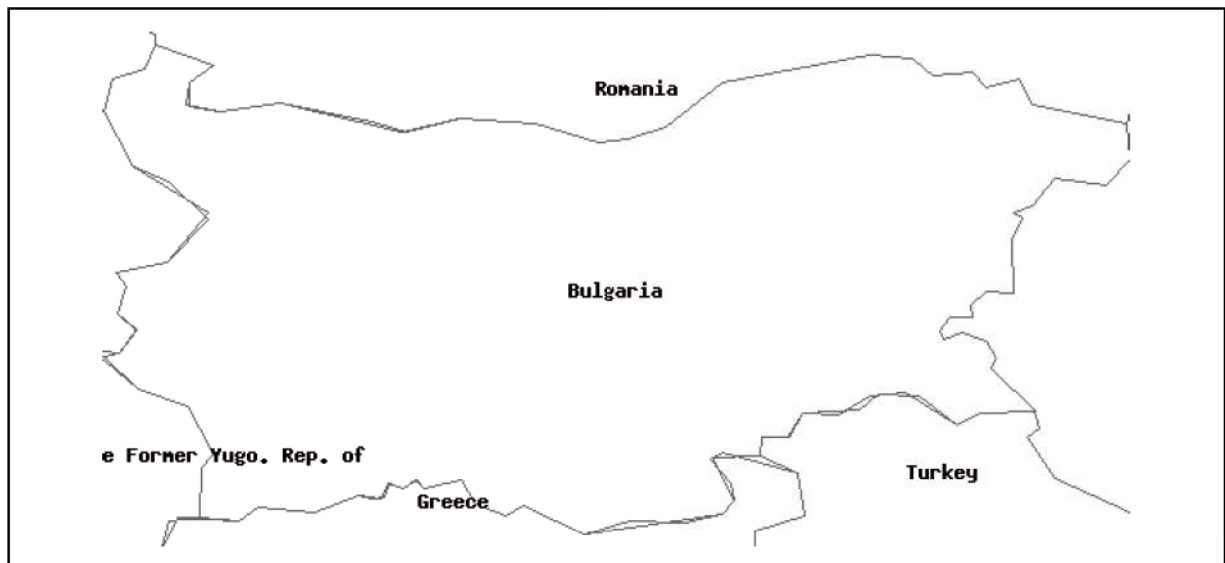


Abbildung 10-4: Zoom auf Bulgarien und Anzeige von Bulgarien im Kartenausschnitt



Bei der Verwendung dieser Methode sollten Sie allerdings vorsichtig sein. Nicht alle Länder haben ein so einfaches Polygon wie Bulgarien. Zu Großbritannien gehören beispielsweise auch Territorien außerhalb der Britischen Inseln. Wenn Sie also eine Datei anlegen wollen, die nur die Informationen zu Großbritannien enthalten soll, werden Sie sich möglicherweise wundern, warum die Ausdehnung weit über Europa hinausgeht.

Hier ein Beispiel, das die über ogrinfo gewonnenen Informationen für die Angaben von `-e` verwendet:

```
> shp2img -m global.map -o mymap.png -e 22.371639 41.242084 28.609278 44.217640
```

Diese Karte zeigt Bulgarien und die umgebenden Länder.

Wollen Sie wieder herauszoomen, verringern Sie die minimalen Koordinaten (das erste Zahlenpaar) und vergrößern die maximalen Koordinaten (das zweite Paar) um ein oder zwei Grad, wie in diesem Beispiel:

```
> shp2img -m global.map -o mymap.png -e 19 39 31 46
```

Auf der Karte in Abbildung 10-5 wird Bulgarien im Zusammenhang mit den angrenzenden Ländern dargestellt, weil ein größerer Ausschnitt gewählt wurde.



Abbildung 10-5: Durch die Änderung der Ausdehnung (EXTENT) wurde mit dem Befehl `shp2img` ein größerer Kartenausschnitt erstellt

Erzeugen klassifizierter Karten

Mit ein paar zusätzlichen Zeilen können Sie die Karte ein wenig verschönern. In Beispiel 10-6 sehen Sie die angepasste Mapdatei mit weiteren Zeilen und geänderten Parametern.

Beispiel 10-6: Mithilfe mehrerer CLASS-Objekte können Sie die Objekte klassifizieren und für jede Klasse eine eigene Farbgebung definieren

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250          # Hintergrundfarbe der Karte
  LAYER
    NAME countries
    TYPE POLYGON
    STATUS DEFAULT
    DATA countries_simpl
    LABELITEM 'NAME'
    CLASSITEM 'NAME'              # Klassifizierungsspalte
    CLASS
      EXPRESSION 'Bulgaria'       # zeichne diese Objekte in der Klasse
      STYLE
        OUTLINECOLOR 100 100 100 # Umrandungs-farbe
        COLOR 255 255 150        # Füllfarbe bei Polygonen
      END
    LABEL
      SIZE LARGE                  # Schriftgröße für den BITMAP-Schrifttyp
      MINFEATURESIZE 40
    END
  END
```

Beispiel 10-6: Mithilfe mehrerer CLASS-Objekte können Sie die Objekte klassifizieren und für jede Klasse eine eigene Farbgebung definieren (Fortsetzung)

```
CLASS
  EXPRESSION ('[NAME]' ne 'Bulgaria')
  STYLE
    OUTLINECOLOR 100 100 100
    COLOR 200 200 200
  END
END
END
END
```

Der Befehl `shp2img` ist der gleiche wie im Beispiel zuvor. Allerdings wird die Karte nun farbig dargestellt, wobei Bulgarien deutlich hervorgehoben wird. Der Aufruf zur Erzeugung des Bilds lautet:

```
> shp2img -m global.map -o example6b.png -e 19 39 31 46
```

Der mit `shp2img` festgelegte Ausschnitt zeigt immer noch Bulgarien. Denken Sie daran, dass die Option `-e` Vorrang vor den Einstellungen für `EXTENT` in der Mapdatei besitzt.

Das Ergebnis der neuen Mapdatei sehen Sie in Abbildung 10-6.



Abbildung 10-6: Durch eine einfache Klassifizierung kann ein Polygon farblich hervorgehoben werden. Außerdem wurde eine Hintergrundfarbe definiert

In der Mapdatei liegen nun drei Farbdefinitionen (`COLOR`, `IMAGECOLOR`, `OUTLINECOLOR`) vor:

```
IMAGECOLOR 180 180 250
```

Über den Parameter `IMAGECOLOR` wird die Hintergrundfarbe der Karte festgelegt (Standard ist Weiß). In diesem Fall bedeutet das, dass alle Bereiche, die nicht von Objekten, in unserem Fall den Länderpolygonen, bedeckt sind, blau dargestellt werden. Hier eignet sich diese Hintergrundfarbe recht gut dazu, das Wasser der Ozeane darzustellen.

Vermutlich ist Ihnen schon aufgefallen, dass es in der Mapdatei nun zwei CLASS-Objekte gibt. Das eine legt fest, welche Farben für die Fläche und die Beschriftung für Bulgarien verwendet werden sollen. Die zweite Definition bezieht sich auf alle übrigen Länder.

```
CLASSITEM 'NAME'
```

Erst durch die Angabe des Parameters CLASSITEM können mehrere Klassen innerhalb eines Layers definiert werden. In unserem Fall wird festgelegt, dass bestimmte Objekte anhand des Attributs NAME gefiltert werden sollen. Die Filterung an sich erfolgt durch den Parameter EXPRESSION. Mehr hierzu im nächsten Abschnitt dieses Kapitels.

```
CLASS
  EXPRESSION 'Bulgaria'
  STYLE
    OUTLINECOLOR 100 100 100
    COLOR 255 255 150
  END
  LABEL
    SIZE LARGE
  END
END
```

Beim Zeichnen dieser Klasse berücksichtigt MapServer nur Polygone, deren Attribut NAME den Wert 'Bulgaria' haben. Alle anderen Polygone oder Länder werden von dieser Klasse nicht dargestellt. Die Einstellung COLOR im STYLE-Objekt teilt MapServer mit, dass alle Objekte dieser Klasse eine Flächenfüllung erhalten sollen. Zusätzlich wird auch noch der Umring des Objekts in einem Grauton gezeichnet. Die zweite Änderung in dieser Klasse besteht darin, dass für die Beschriftung eine größere als die Standardschrift verwendet werden soll.



MapServer verfügt über eigene interne Bitmap-Zeichensätze. Diese werden standardmäßig als Beschriftung verwendet. Es können dabei verschiedene Größen angegeben werden: TINY (winzig), SMALL (klein), MEDIUM (mittel), LARGE (groß) oder GIANT (riesig). Es ist aber auch möglich, TrueType-Fonts zu verwenden, die Beschriftungen mit einer höheren Qualität erzeugen. Hierfür müssen allerdings ein paar zusätzliche Parameter angegeben werden, und die entsprechenden Zeichensatzdateien (.ttf) müssen vorhanden sein.

Verwenden Sie TrueType-Zeichensätze so müssen Sie anstelle der Schlüsselwörter (TINY oder LARGE) eine tatsächliche Schriftgröße (z.B. 12) angeben. Vermutlich kennen Sie TrueType-Zeichensätze bereits aus Ihrem Textverarbeitungsprogramm (Arial, Verdana, Times New Roman).

In der zweiten Klasse wird festgelegt, wie die übrigen Länder (außer Bulgarien) dargestellt werden sollen:

```
CLASS
  EXPRESSION ('[NAME]' ne 'Bulgaria')
  STYLE
    OUTLINECOLOR 100 100 100
```



```

        COLOR 200 200 200
    END
END

```

Wie Sie sehen, haben wir für diese Klasse eine komplexere EXPRESSION-Klausel verwendet. Diese sorgt dafür, dass diese Klasse nur Länder darstellt, deren Name nicht »Bulgaria« lautet. Beachten Sie, dass jedes Objekt der angesprochenen Daten nur einmal gezeichnet wird. Für die Darstellung der restlichen Objekte können Sie statt der Angabe '[NAME]' ne 'Bulgaria' den Parameter EXPRESSION ganz weglassen oder auch die folgende Zeile angeben:

```
EXPRESSION /. /
```

Über diese Definition sprechen Sie alle Objekte an. Da Bulgarien schon über die erste Klasse gezeichnet wird, ist das Ergebnis das gleiche. Es wird aber deutlich, dass die Reihenfolge, in der die Klassen in der Mapdatei definiert werden, von Bedeutung sein kann. Im folgenden Abschnitt werden wir genauer auf die Verwendung der EXPRESSION-Klauseln eingehen.

Was bedeuten die einzelnen Operatoren?

Für die Schlüsselwörter FILTER und EXPRESSION können einfache oder komplexe Ausdrücke verwendet werden, die definieren, welche Objekte dargestellt werden sollen. Beispiel 10-6 zeigte die einfachste Möglichkeit, über EXPRESSION festzulegen, welche Objekte über ein CLASS-Objekt dargestellt werden sollen. Zur Unterscheidung wurde eine einfache Zeichenkette angegeben. MapServer sucht daraufhin in der über CLASSITEM definierten Spalte nach dieser Zeichenkette.

Logische Ausdrücke können eingesetzt werden, wenn mehr als eine Zeichenfolge ausgewertet werden muss. Es können auch bestimmte Wertebereiche angegeben werden oder eine Kombination verschiedener Attribute. In Tabelle 10-1 finden Sie eine Zusammenfassung der möglichen Ausdrücke.

Tabelle 10-1: Logische Operatoren und reguläre Ausdrücke

Operator	Beschreibung	Beispiele
=	ist gleich	('[NAME]' eq 'Canada')
eq		([TOTAL] = 32507874)
ne	ist nicht gleich	('[NAME]' ne 'Canada')
>	größer als	([TOTAL] gt 1000000)
gt		([TOTAL] > 1000000)
<	kleiner als	([TOTAL] lt 1000000)
lt		([TOTAL] < 1000000)
>=	größer oder gleich	([TOTAL] ge 1000000)
ge		([TOTAL] >= 1000000)
<=	kleiner oder gleich	([TOTAL] le 1000000)
le		([TOTAL] <= 1000000)

Tabelle 10-1: Logische Operatoren und reguläre Ausdrücke (Fortsetzung)

Operator	Beschreibung	Beispiele
/./	beliebiges Zeichen	/./ regulärer Ausdruck
^	beginnt mit der Zeichenfolge	/^Bulg/ regulärer Ausdruck
\$	endet auf der Zeichenfolge	/garia\$/ regulärer Ausdruck
AND	beide Aussagen müssen erfüllt sein	(('[NAME]' ne 'Canada') AND ([TOTAL] > 1000000))
OR	eine der beiden Aussagen muss erfüllt sein	(('[NAME]' eq 'Canada') OR ('[NAME]' eq 'Brazil'))

Im LAYER-Objekt können Sie den Parameter FILTER angeben, der die Daten vor der Verarbeitung durch die Klassen einschränkt.

Das Erstellen des Filters ist nicht immer einfach. Hier ein paar Faustregeln:

- Umgeben Sie nicht den gesamten Ausdruck mit Anführungszeichen, wie hier: "('[COUNTRY]' = 'Canada')". Dieses Beispiel ist falsch!
- Bei Feldern, deren Werte aus Zeichenketten bestehen, müssen sowohl der Feldname als auch der Wert mit Anführungszeichen umgeben werden, z.B.: ('[COUNTRY]' = 'Canada').
- Bei Feldern, die numerische Werte enthalten, darf weder Feldname noch Wert mit Anführungszeichen angegeben werden, z.B. ([TOTAL] = 1000).

Werden Regeln mit der Option FILTER verwendet, gibt es ein paar Einschränkungen bezüglich der Datenquellen. Dies gilt besonders für Datenbankverbindungen wie PostGIS:

- Die Operatoren eq, ne, gt, lt, ge und le stehen nicht zur Verfügung.
- Attributnamen müssen nicht mit Anführungszeichen oder eckigen Klammern umgeben werden. Die Ausdrücke (TOTAL > 1000) und (NAME = 'Canada') sind beide gültig.

Eine Legende für die Karte erstellen

Ein wichtiger Teil der Karten ist die Legende, die beschreibt, welche Bedeutung bestimmte Farben und Symbole haben. MapServer ist in der Lage, Legenden in Echtzeit bei der Erstellung der Karte oder als separates Bild zu erzeugen.

Wollen Sie eine Legende in Ihre Karte einbetten, muss in der Mapdatei im MAP-Objekt ein LEGEND-Objekt angegeben werden. Die Legende ist übrigens nicht Teil eines LAYER-Objekts, sondern Teil der Einstellungen für die gesamte Karte. Beispiel 10-7 zeigt die ersten Zeilen einer Mapdatei, bei der die Einstellungen für die Legende hervorgehoben wurden.

Beispiel 10-7: Definition des LEGEND-Objekts in der Mapdatei

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250
```

Beispiel 10-7: Definition des LEGEND-Objekts in der Mapdatei (Fortsetzung)

```
LEGEND
  STATUS EMBED          # Status EMBED, ON oder OFF
  POSITION LR           # Position in der Karte unten rechts(lower right)
  TRANSPARENT TRUE     # transparenter Hintergrund
END
...
```

Die möglichen Werte für das Attribut STATUS lauten ON, OFF oder EMBED. Der Wert EMBED bietet die Möglichkeit, die Legende direkt in die Karte einzubetten. Wählen Sie ON, müssen Sie die Legende als separate Datei erzeugen. (Wie das funktioniert, werden Sie später noch erfahren.) Sie können der Legende einen transparenten Hintergrund zuweisen. Wollen Sie das nicht, entfernen Sie die Zeile TRANSPARENT TRUE einfach. Standardmäßig wird die Legende dann mit einem weißen Hintergrund versehen.

Schauen Sie sich den folgenden Link an, um weitere Parameter kennenzulernen, die Sie für die Gestaltung der Legende angeben können:

<http://mapserver.gis.umn.edu/docs/reference/mapfile/legend>

Damit die Legende angezeigt wird, ist noch eine weitere Änderung notwendig: Für jede Klasse muss der Parameter NAME definiert werden. Der angegebene Name entspricht dem Text, der neben den Symbolen in der Legende erscheint. Hierfür können Sie die erste Klasse um die Zeile NAME 'Bulgaria' und die zweite Klasse um NAME 'All Countries' (alle übrigen Länder) ergänzen, wie in Beispiel 10-8 gezeigt. Ohne diese Angaben wird keine Legende erzeugt. MapServer generiert aufgrund der Angaben im STYLE-Block Legendenbilder. Sollten Ihnen diese Bilder nicht gefallen, können Sie auch über den Parameter KEYIMAGE auf Grafiken verweisen, die in der Legende angezeigt werden sollen. Dabei müssen Sie den Pfad zu der Datei relativ zum Pfad der Mapdatei angeben.

Beispiel 10-8: Damit die Legende erzeugt wird, muss im CLASS-Objekt der Parameter NAME angegeben werden

```
...
CLASS
  NAME 'Bulgaria'        # Text, der in der Legende erscheint
  # KEYIMAGE './keyimages/bulgaria.png' # Bild für die Legende
  EXPRESSION 'Bulgaria'
  STYLE
    OUTLINECOLOR 100 100 100
    COLOR 255 255 150
  END
  LABEL
    SIZE LARGE
    MINFEATURESIZE 40
  END
END
CLASS
  NAME 'All Countries'
  EXPRESSION ./          # Angabe über einen regulären Ausdruck
  STYLE
    OUTLINECOLOR 100 100 100
```

Beispiel 10-8: Damit die Legende erzeugt wird, muss im CLASS-Objekt der Parameter NAME angegeben werden (Fortsetzung)

```
COLOR 200 200 200
END
END
...
```

Wenn Sie `shp2img` erneut ausführen, erhalten Sie das in Abbildung 10-7 gezeigte Bild.



Abbildung 10-7: In die Karte wurde eine Legende eingebettet

Anhand der Einstellung `POSITION` können Sie MapServer mitteilen, wo die Legende auf der Karte platziert werden soll. Für die relative Position gibt es einen Zwei-Buchstaben-Code: unten rechts (lower right, LR), oben Mitte (upper center, UC) usw.

Bis jetzt haben wir die Legende direkt in die Karte eingebettet. Sie können sie aber auch als eigene Datei anlegen, die nur die Legende enthält. Hierfür gibt es den Befehl `legend`:

```
> legend global.map legend.png
```

Der Befehl `legend` erzeugt eine Datei, die nur die Legende enthält, wie in Abbildung 10-8 zu sehen ist. Auf den Namen des Befehls folgen zwei Parameter: zuerst der Name der Mapdatei und dann der Name für das Ausgabebild.

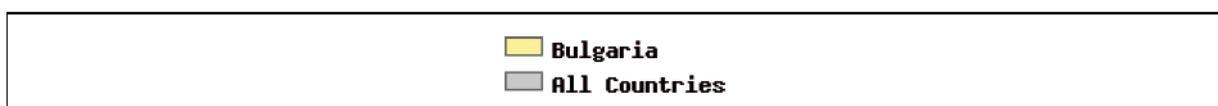


Abbildung 10-8: Mit dem Befehl `legend` erstelltes Bild der Legende zur Karte

Die Legende wird auch diesmal entsprechend den Einstellungen im `LEGEND`-Objekt der Mapdatei erstellt. Zwar können Sie ein paar Einstellungen ändern, die Werte der Optionen `POSITION` und `EMBED` werden bei der Verwendung des Hilfsprogramms `legend`

jedoch nicht berücksichtigt. Diese Optionen sind für die Verwendung mit shp2img oder einer mapserv-Webapplikation gedacht.

Erzeugen einer Maßstabsleiste über das SCALEBAR-Objekt

Als Maßstab wird das Verhältnis zwischen der abgebildeten Größe und der entsprechenden Größe in der Wirklichkeit bezeichnet. Über die Maßstabsleiste erhalten Sie einen Anhaltspunkt, welche Entfernungen in der Karte abgebildet werden. Die Leiste besteht aus Linien oder Balken und Textangaben, die mitteilen, für welche Entfernung ein Abschnitt der Leiste steht. Die Mapdatei lässt sich leicht um eine Maßstabsleiste erweitern. Alle Optionen für die Maßstabseinstellungen finden Sie in der MapServer-Dokumentation unter <http://mapserver.gis.umn.edu/docs/reference/mapfile/scalebar>.

In Beispiel 10-9 wurden die zusätzlichen Zeilen für die Definition der Maßstabsleiste in der bereits bekannten Mapdatei hervorgehoben.

Beispiel 10-9: Kartenbild mit einer eingebetteten Maßstabsleiste

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250

  UNITS DD                                # Karteneinheiten innerhalb des Projekts

  SCALEBAR                                # Beginn des SCALEBAR-Objekts
    STATUS EMBED                          # Status EMBED, ON, OFF
    STYLE 0                              # Stil der Leiste 0 oder 1
    UNITS KILOMETERS                      # Einheiten der Maßstabsleiste
    INTERVALS 3                          # Anzahl der Abschnitte
    TRANSPARENT TRUE
    OUTLINECOLOR 0 0 0
  END                                    # Ende des SCALEBAR-Objekts

  LEGEND
    STATUS EMBED
  ...
```

In der linken unteren Ecke von Abbildung 10-9 sehen Sie nun die Maßstabsleiste. Über die Leiste können Entfernungen auf der Karte (Zentimeter, Pixel, Zoll) auf Entfernungen auf der Erde übertragen werden.

Auch dieses Mal verwenden wir den bekannten shp2img-Befehl, um eine Karte mit Maßstabsleiste zu erzeugen:

```
> shp2img -m global.map -o mymap.png -e 19 39 31 46
```

In der ersten zusätzlichen Zeile finden Sie den Parameter UNITS. Der Wert DD legt fest, dass die Koordinaten in Dezimalgrad angegeben werden. Diese Angabe wirkt sich nicht nur auf die Maßstabsleiste, sondern auf die gesamte Mapdatei aus. Daher gehört dieser Parameter nicht zum SCALEBAR-Objekt.



Abbildung 10-9: Ein Kartenbild mit eingebettetem Maßstab



Die ausführliche Behandlung von Einheiten, Koordinatensystemen und Projektionen geht über den Rahmen dieses Kapitels hinaus. Dennoch ist das Wissen darum wichtig, wenn Sie mit Daten aus unterschiedlichen Quellen arbeiten. Weitere Informationen zu Projektionsmodellen finden Sie in Anhang A.

Für das Objekt SCALEBAR in der Mapdatei gibt es eine Reihe von Optionen, von denen Sie einige (aber nicht alle) in Beispiel 10-9 finden können. Die hier verwendeten Möglichkeiten reichen aus, um eine ansprechende, einfache Maßstabsleiste zu erzeugen. Die Option STATUS EMBED kennen Sie schon vom LEGEND-Objekt. Die Maßstabsleiste wird hierdurch direkt in die Karte eingebettet. Über den Parameter STYLE können Sie die Maßstabsleiste in verschiedenen Stilen darstellen. Hier stehen 0 und 1 zur Verfügung. Die Angabe UNITS KILOMETERS besagt, welche Maßeinheit für den Maßstab verwendet werden soll. In unserem Beispiel wird der Maßstab in Kilometern angegeben. Es sind aber auch andere Maßeinheiten möglich (z.B. Inch, Meter usw.).

Die Angabe INTERVALS 3 legt fest, dass die Maßstabsleiste in drei Abschnitte unterteilt werden soll. Standardmäßig werden vier Abschnitte verwendet. Mit dieser praktischen Option können Sie direkt die Lesbarkeit Ihrer Maßstabsleiste beeinflussen. Gibt es in einem kleinen Bereich zu viele Abschnitte, sind die Beschriftungen möglicherweise nicht mehr richtig zu erkennen.

Eine wichtige Option für die Maßstabsleiste wurde in diesem Beispiel nicht gezeigt. Mit der Option SIZE $x \times y$ können Sie explizit festlegen, wie breit (x) und wie hoch (y) die Maßstabsleiste sein soll. Um die Einstellungen einfach zu halten, können Sie hier den Standardwert übernehmen. Geben Sie über INTERVALS die Anzahl der Abschnitte an, wird die Breite übrigens automatisch entsprechend geteilt. Wählen Sie für INTERVALS

eine hohe Zahl, müssen Sie vermutlich über `SIZE` die Größe der Maßstabsleiste erhöhen, damit die Angaben gut lesbar sind.

Die letzten beiden Optionen in Beispiel 10-9 sind rein ästhetischer Natur. `TRANSPARENT` sorgt dafür, dass die Maßstabsleiste vor einem transparenten Hintergrund erscheint, während die Angabe von `OUTLINECOLOR` die einzelnen Abschnitte in diesem Fall mit einem dünnen schwarzen Rand versieht.

Soll die Maßstabsleiste nicht in die Karte eingebettet werden, ändern Sie `STATUS EMBED` in `STATUS ON` und legen dann mittels des Hilfsprogramms `scalebar` ein separates Bild für der Maßstabsleiste an. Ähnlich wie das Programm `legend` wird `scalebar` wie folgt aufgerufen:

```
> scalebar global.map scalebar.png
```

Auch diesem Befehl werden über zwei Parameter der Name der Mapdatei und der Name der Ausgabedatei übergeben. Beachten Sie, dass `scalebar` nicht über die gleichen Optionen verfügt wie `shp2img`.

Abbildung 10-10 zeigt das von `scalebar` erzeugte Bild.

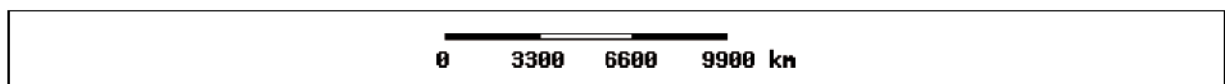


Abbildung 10-10: Eine einfache, mit dem Werkzeug `scalebar` erzeugte Maßstabsleiste

Möchten Sie die Maßstabsleiste zusammen mit Kartenbildern manuell für eine Publikation – z.B. für eine Webseite – zusammenstellen, sollten Sie darauf achten, die Grafiken nicht zu skalieren. Wenn Sie die Größe der Maßstabsleiste verändern, nicht jedoch entsprechend auch die Größe des Kartenbilds, zeigt die Maßstabsleiste nicht mehr den korrekten Maßstab an.

Die fertige Mapdatei

Die fertige Mapdatei dieses Beispiels finden Sie in Beispiel 10-10. Der Wert für `EXTENT` wurde an die in den früheren Beispielen mit dem Befehl `shp2img` verwendeten Ausdehnungen angepasst.

Beispiel 10-10: Die fertige Mapdatei mit allen behandelten Änderungen und Aktualisierungen

```
MAP                                # Beginn des MAP-Objekts
SIZE 600 300                      # Bildgröße des Kartenbilds 600 x 300 Pixel
#EXTENT -180 -90 180 90           # auskommentierter Original-Extent
EXTENT 19 39 31 46                # Darstellungsbereich
IMAGECOLOR 180 180 250            # Hintergrundfarbe
UNITS DD                           # Karteneinheiten
SCALEBAR                           # Beginn des SCALEBAR-Objekts
STATUS EMBED                       # in das Kartenbild eingebundene Maßstabsleiste
UNITS KILOMETERS                  # Einheiten der Maßstabsleiste (km)
INTERVALS 3                       # Anzahl der Abschnitte der Leiste
TRANSPARENT TRUE                  # transparenter Hintergrund
OUTLINECOLOR 0 0 0                # schwarze Umrandung
```

Beispiel 10-10: Die fertige Mapdatei mit allen behandelten Änderungen und Aktualisierungen

```
END                                # Ende des SCALEBAR-Objekts
LEGEND                             # Beginn des LEGEND-Objekts
  STATUS EMBED                     # in das Kartenbild eingebettete Legende
  POSITION LR                       # Position der Legende unten rechts
  TRANSPARENT TRUE                 # transparenter Hintergrund
END                                # Ende des LEGEND-Objekts
LAYER                              # Beginn des LAYER-Objekts
  NAME countries                   # Name der Ebene
  TYPE POLYGON                    # Typ der Objekte im LAYER
  STATUS DEFAULT                   # Status ON, OFF, DEFAULT
  DATA countries_simpl           # Datenquelle
  LABELITEM 'NAME'                # Beschriftungsfeld
  CLASSITEM 'NAME'                # Klassifizierungsfeld
  CLASS                           # Beginn des CLASS-Objekts
    NAME 'Bulgaria'                # Name der Klasse
    EXPRESSION 'Bulgaria'          # diese Objekte in der Klasse zeichnen
    STYLE                         # Beginn des STYLE-Objekts
      OUTLINECOLOR 100 100 100     # Umrangungsfarbe
      COLOR 255 255 150           # Flächenfüllung der Polygone
    END                            # Ende des STYLE-Objekts
    LABEL                         # Beginn des LABEL-Objekts
      SIZE LARGE                   # Schriftgröße, Angabe für Bitmap-Schrift
      MINFEATURESIZE 40           # minimale Größe, bei der eine Beschriftung erfolgt
    END                            # Ende des LABEL-Objekts
  END                              # Ende des CLASS-Objekts
  CLASS                           # Beginn des zweiten CLASS-Objekts
    NAME 'All Countries'           # Name der Klasse
    EXPRESSION ('[NAME]' ne 'Bulgaria')
    STYLE                         # Beginn des STYLE-Objekts
      OUTLINECOLOR 100 100 100     # Umrangungsfarbe
      COLOR 200 200 200           # Farbe der Flächenfüllung
    END                            # Ende des STYLE-Objekts
  END                              # Ende des CLASS-Objekts
END                                # Ende des LAYER-Objekts
END                                # Ende des MAP-Objektes und Ende der Mapdatei
```

Das Format des Ausgabebilds bestimmen

MapServer kann die Kartenbilder, Legenden und Maßstabsleisten in verschiedenen Ausgabeformaten speichern. Neben unterschiedlichen Bildformaten wie PNG, GIF, JPEG oder TIFF ist u.a. auch die Ausgabe von SWF, PDF oder ImageMaps möglich. In diesem Kapitel haben wir das Standardformat PNG verwendet. Unterschiedliche MapServer-Pakete verwenden möglicherweise ein anderes Bildformat als Standard. Normalerweise unterstützen alle Versionen von MapServer PNG, manchmal jedoch nicht oder zumindest nicht als Standard.



MapServer kann mit einer Vielzahl verschiedener Optionen kompiliert werden. Der Begriff »MapServer-Paket« steht hier für MapServer selbst sowie die dazugehörigen Programme, z.B. das bereits bekannte shp2img. Die Fähigkeiten dieser Programme hängen davon ab, wie sie kompiliert wurden.

Das Format des Ausgabebilds kann explizit in der Mapdatei festgelegt werden. Hierfür müssen Sie die Datei um eine Zeile erweitern. Beispiel 10-11 zeigt die zusätzliche Zeile in der Mapdatei aus den vorigen Beispielen.

Beispiel 10-11: Festlegen des Formats der Ausgabebilder als JPEG

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250
  IMAGETYPE JPEG

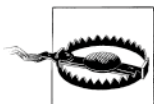
  UNITS DD
  ...
```

Diese Einstellung kann bei Verwendung von shp2img mit der Option -i überschrieben werden. Über die Programme scalebar und legend ist ein Überschreiben des Standardformats dagegen nicht möglich.

Das mit IMAGETYPE angegebene Ausgabeformat muss vom verwendeten MapServer-Paket unterstützt werden. Um zu erfahren, welche Formate Ihre MapServer-Version unterstützt, rufen Sie ./mapserv bzw. mapserv.exe mit der Option -v auf. Neben der Versionsnummer werden alle unterstützten Datenformate und Ausgabeformate (hier hervorgehoben) angezeigt:

```
> ./mapserv -v
MapServer version 5.0.0 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP OUTPUT=PDF OUTPUT=SWF OUTPUT=SVG
SUPPORTS=PROJ SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT
SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER SUPPORTS=FASTCGI
SUPPORTS=THREADS SUPPORTS=GEOS INPUT=JPEG INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

Die Liste zeigt, dass diese MapServer-Installation Dateien in den Formaten GIF, PNG, JPEG, WBMP, PDF, SWF, SVG ausgeben kann.



Die von MapServer erstellten WBMP-Bilder scheinen manchmal nicht lesbar zu sein. Dies ist aber nur auf Nicht-Windows-Systemen ein Problem.

Das PNG-Format reicht für eine große Zahl an Anwendungen als Ausgabeformat aus. In vielen Fällen ist es das Standardformat. Haben Sie aber eine Rasterebene mit mehr als 256 Farben, werden die Bilder im PNG-Standardformat möglicherweise nicht korrekt

dargestellt. Meistens reicht es jedoch aus, die von MapServer für das Ausgabebild verwendete Farbtiefe anzupassen, indem Sie beispielsweise die 24-Bit-Version des PNG-Formats festlegen:

```
IMAGETYPE PNG24
```

Aber Vorsicht! Eine Erhöhung der Farbtiefe sorgt auch für größere Bilddateien. Dies kann zu starken Leistungseinbußen führen, wenn die Bilder über das Web verschickt werden sollen.

Neben fest definierten Ausgabeformaten wie PNG oder JPEG kann mit dem Schlüsselwort `IMAGETYPE` auch auf zusätzlich definierte Böcke mit Ausgabeformaten verwiesen werden.

Diese feinere Einstellung für das Ausgabeformat können Sie in der Mapdatei mit dem Objekt `OUTPUTFORMAT` vornehmen. Jedes definierte Ausgabeformat beginnt dabei mit dem Schlüsselwort `OUTPUTFORMAT` und endet mit dem `END`-Tag. Welches der Ausgabeformate für die Erstellung der Karte dann zum Einsatz kommt, definiert der Parameter `IMAGETYPE`. Dessen Wert verweist auf die Benennung der verschiedenen Ausgabeformate im `Keyword-Tag NAME` des `OUTPUTFORMAT`-Objekts. Im folgenden Beispiel sehen Sie drei `OUTPUTFORMAT`-Objekte, die jeweils unterschiedliche Treiber für die Ausgabe des PNG-Formats verwenden (GD, AGG, GDAL).

Beispiel 10-12: Drei OUTPUTFORMAT-Objekte, die unterschiedliche Treiber verwenden

```
OUTPUTFORMAT
  NAME png
  DRIVER "GD/PNG"      # Treiber, über den das Ausgabebild generiert werden soll
  MIMETYPE "image/png" # MIME-Typ, der verwendet werden soll
  IMAGEMODE PC256      # Ausgabemodus PC256 - 8 Bit Farbtiefe
  EXTENSION "png"      # Dateiendung, die verwendet werden soll
END
```

```
OUTPUTFORMAT
  NAME png
  DRIVER "GDAL/PNG"
  MIMETYPE "image/png"
  IMAGEMODE PC256
  EXTENSION "png"
END
```

```
OUTPUTFORMAT
  NAME png
  DRIVER "AGG/PNG"
  MIMETYPE "image/png"
  IMAGEMODE RGB      # Ausgabemodus RGB - 24 Bit Farbtiefe ohne Transparenz
  EXTENSION "png"
END
```

Weitere Informationen zum `OUTPUTFORMAT`-Objekt finden Sie in der MapServer-Dokumentation unter <http://mapserver.gis.umn.edu/docs/reference/mapfile/outputformat>.

Interaktive Karten im Web veröffentlichen

Typischerweise wird MapServer über einen Webserver wie Apache oder IIS eingesetzt, prinzipiell kann aber jeder Webserver verwendet werden, der CGI-Programme unterstützt. Der Webserver leitet die Anfrage an MapServer weiter und gibt das Ergebnis – in der Regel eine Karte – wieder an den Browser zurück. In Kapitel 4 wurde beschrieben, wie Sie das MapServer-CGI-Programm kompilieren bzw. herunterladen und einrichten können. In diesem Kapitel befassen wir uns mit `mapserv`, dem Hauptprogramm von UMN MapServer. Das Executable `mapserv` wird im Gegensatz zu den Kommandozeilenwerkzeugen (Kapitel 10) für Webserver-Applikationen eingesetzt. Wenn wir in diesem Buch von der »Integration von MapServer mit einem Webserver« sprechen, beziehen wir uns auf den Einsatz des `mapserv`-Executables in Verbindung mit einem Webserver.

MapServer besitzt außerdem eine eigene Skriptsprache: MapScript. Sie bietet Programmierschnittstellen zu vielen Programmiersprachen. Nähere Informationen zu MapScript finden Sie in Kapitel 15. Wenn Sie als Programmierer die Funktionen von MapServer in eigene Programme einbinden möchten, ist MapScript möglicherweise genau das Richtige für Sie.

Die in diesem Kapitel besprochene CGI-Version von MapServer ist für diejenigen gedacht, die MapServer ohne Programmieraufwand einsetzen möchten.

In Kapitel 12 werden Sie dann erfahren, wie Sie MapServer konfigurieren müssen, um standardisierte Dienste (OGC WMS, OGC WFS) aufzusetzen. Einen standardisierten Dienst können Sie dann mit jedem Client, der diesen Standard unterstützt, aufrufen, d.h., Sie konfigurieren die Karte, nutzen aber bestehende Clientanwendungen zur Visualisierung und sparen so den Aufwand, einen eigenen Client zu programmieren. In Kapitel 13 werden Sie dann verschiedene WMS-Clients kennenlernen. Das Desktop-GIS Quantum GIS als WMS-Client kennen Sie schon aus Kapitel 8.

MapServer einrichten und testen

Bevor Sie MapServer benutzen können, müssen aber noch ein paar Dinge eingerichtet werden. Hierbei gibt es zwischen den Betriebssystemen einige Unterschiede zu beachten – beispielsweise den Namen des MapServer-Executables, die Speicherorte für Dateien usw.

MapServer für Windows (MS4W)

In Kapitel 4 wurde beschrieben, wie Sie das MS4W-Paket herunterladen und installieren können; vielleicht wollen Sie sich diesen Teil noch einmal ansehen, bevor Sie den Webserver starten. Später in diesem Kapitel werden Sie eine eigene Anwendung erstellen. Allerdings müssen Sie vorher wissen, wie MapServer in der MS4W-Umgebung funktioniert.

Das MS4W-Paket beinhaltet bereits den Apache-Webserver. Das MapServer-Programm liegt im Verzeichnis *cgi-bin*. Dies ist bei Apache das Verzeichnis, in dem Skripten ausgeführt werden können. Haben Sie eine eigene *mapserv.exe* kompiliert, können Sie diese installieren, indem Sie die Datei nach *cgi-bin* kopieren. Bei der MS4W-Standardinstallation finden Sie dieses Verzeichnis unter dem gleichen Laufwerksbuchstaben wie MS4W selbst (z.B. C: oder D:) unter dem Pfad *\ms4w\Apache\cgi-bin*. Möglicherweise liegt dort mehr als eine MapServer-Version. *mapserv_36.exe* steht beispielsweise für MapServer in der Version 3.6 (diese Version ist schon ziemlich alt), während *mapserv_5.exe* die aktuelle Version 5.0 bezeichnet. Normalerweise trägt die aktuellste Version allerdings nur den Namen *mapserv.exe*.

MapServer unter Linux mit Apache

Eine Beschreibung der Einrichtung von MapServer für den Einsatz unter Linux finden Sie ebenfalls in Kapitel 4. Die Installation des MapServer-CGI-Programms erfolgt dabei ganz ähnlich wie mit MS4W. Auch hier wird das *mapserv*-Executable einfach in das *cgi-bin*-Verzeichnis von Apache kopiert (oder über einen symbolischen Link verbunden). Unter SuSE Linux befindet sich das *cgi-bin*-Verzeichnis z.B. unter dem Pfad */srv/www/cgi-bin*. *mapserv* muss in diesem Verzeichnis liegen, damit der Webserver darauf zugreifen kann. Dies gilt auch für andere nicht MS4W-basierte Installationen, bei denen Apache verwendet wird. Je nachdem, welche Distribution (Ubuntu, Debian, Red Hat o.Ä.) Sie verwenden, kann der Pfad zum *cgi-bin* unterschiedlich ausfallen.

Apache konfigurieren

Die Einrichtung von Apache für die Verwendung mit MapServer muss nicht zwangsläufig kompliziert sein. Es reicht vermutlich schon aus, *mapserv* bzw. *mapserv.exe* im Standard-*cgi-bin*-Verzeichnis des Webserver abzulegen. Im nächsten Abschnitt finden Sie ein paar Tests, mit denen Sie feststellen können, ob alles richtig funktioniert.

Bevor Sie auf die von MapServer erstellten Karten zugreifen können, werden allerdings noch ein paar Dinge benötigt. MapServer braucht einen temporären Speicherort für

Kartenbilder, Legenden, Maßstabsleisten usw. Auf dieses Verzeichnis muss auch der Webserver zugreifen können, damit die Benutzer die erzeugten Grafiken betrachten können. In dem temporären Verzeichnis speichert MapServer laufend neue Dateien, daher müssen Sie dafür sorgen, dass dieses Verzeichnis regelmäßig geleert wird. Ansonsten würden sich hier Unmengen an Dateien ansammeln, die nicht mehr benötigt werden. Zum Leeren des Verzeichnisses können Sie z.B. einen *cronjob* einrichten.

Beachten Sie, dass Sie diese temporären Verzeichnisse nicht unbedingt benötigen, wenn Sie MapServer als WMS-Server einsetzen. In diesem Fall werden die Bilder direkt im Client angezeigt und nicht erst als Datei gespeichert.

Das temporäre Verzeichnis unter Windows

Bei der Verwendung von MS4W ist das temporäre Verzeichnis bereits eingerichtet. Sie finden das Verzeichnis unter *c:\ms4w\tmp\ms_tmp*. In der Apache-Konfigurationsdatei unter *c:\ms4w\Apache\conf\httpd.conf* wurde für dieses Verzeichnis ein *Alias* auf */ms_tmp* angelegt. Wenn Apache also eine Anfrage einer Webapplikation nach einer Datei in */ms_tmp* bekommt, sucht er diese im Verzeichnis *c:\ms4w\tmp\ms_tmp*. Hat MapServer eine Datei unter dem Pfad *c:\ms4w\tmp\ms_tmp\mymap.png* angelegt, können Sie diese dann über die URL *http://localhost/ms_tmp/mymap.png* aufrufen.

Der temporäre Verzeichnis unter Linux

Benutzen Sie eine eigene Apache-Installation, müssen Sie das Verzeichnis für die temporären Dateien und den Alias in der Konfigurationsdatei von Apache (*httpd.conf*) selbst einstellen. Wir wollen dieses Vorgehen am Beispiel von Apache 2 unter SuSE Linux einmal erläutern. Prinzipiell kann sich das Verzeichnis für temporäre Dateien an einem beliebigen Ort Ihres Dateisystems befinden. Um die Sache zu vereinfachen, können Sie dieses Verzeichnis auch direkt im *DocumentRoot*-Verzeichnis des Webserver, dies ist standardmäßig das Verzeichnis *htdocs*, anlegen. In diesem Fall müssen Sie kein Alias erstellen, weil Apache direkt auf die Dateien zugreifen kann.

Unter SuSE Linux finden Sie die Einstellungen für das *DocumentRoot* in der Konfigurationsdatei unter */etc/apache2/httpd.conf*. Der entsprechende Eintrag lautet hierbei standardmäßig *DocumentRoot /srv/www/htdocs*.

Wenn Sie das temporäre Verzeichnis als */srv/www/htdocs/ms_tmp* anlegen, kann MapServer es über die URL *http://localhost/ms_tmp* finden. Für einen schnellen Test können Sie die bereits erstellte Datei *mymap.png* ins Verzeichnis */srv/www/htdocs/ms_tmp* kopieren. Wenn Sie nun die Adresse *http://localhost/ms_tmp/mymap.png* in Ihren Browser eingeben, sollte das Bild angezeigt werden.

Um Dateien im *ms_tmp*-Verzeichnis speichern zu können, müssen ausreichende Benutzerrechte für das Verzeichnis eingestellt sein. Der Trick besteht darin, sowohl den Webserver-Benutzer als auch dessen Gruppe als Besitzer des Verzeichnisses zu definieren. Hierfür können Sie den folgenden Befehl verwenden:

```
> chown wwwrun:www /srv/www/htdocs/ms_tmp
```

Jetzt sollte der Webserver und damit auch MapServer darauf zugreifen können, wobei es je nach Plattform und verwendetem Webserver gewisse Unterschiede geben kann. Das Konzept bleibt dabei aber gleich.

Bei der Erstellung von MapServer-Anwendungen ist es äußerst wichtig, den Pfad zum Verzeichnis *ms_tmp* im Dateisystem (das Verzeichnis muss übrigens nicht unbedingt *ms_tmp* heißen, eigentlich können Sie einen beliebigen Namen verwenden) und die URL nicht zu verwechseln.

MapServer testen

Mit ein paar Tests können Sie ermitteln, ob MapServer richtig eingerichtet ist und funktioniert. Die Tests unter MS4W und Linux sind recht ähnlich. Allerdings gibt es ein paar Unterschiede, die erwähnt werden sollten.

mapserv.exe unter Windows testen

Bei unserem ersten Test für MS4W gehen wir davon aus, dass der Webserver bereits wie in Kapitel 4 gestartet wurde. Rufen Sie die *mapserv.exe* über die URL *http://localhost/cgi-bin/mapserv.exe* ohne weitere Parameter im Browser auf, um festzustellen, ob dass MapServer-Executable aufgerufen werden kann. Daraufhin sollten Sie eine Nachricht wie diese erhalten:

```
No query information to decode. QUERY_STRING is set, but empty.
```

Das ist eine gute Nachricht, denn sie kommt direkt von MapServer. Sie haben im Aufruf keine weiteren Parameter angegeben, die MapServer normalerweise benötigt. Aber die Nachricht bestätigt, dass die Anfrage angekommen ist und darauf reagiert wurde. Erhalten Sie stattdessen eine Fehlermeldung, liegt dies wahrscheinlich an falsch gesetzten Dateirechten, und Apache kann möglicherweise nicht auf die angeforderte Datei zugreifen. Vielleicht läuft auch der Webserver noch nicht richtig.

MapServer (mapserv) mit Apache unter Linux testen

Unter Linux können Sie MapServer auf die gleiche Weise testen wie unter Windows. Der einzige Unterschied besteht darin, dass unter Linux der Name des Programms etwas kürzer ist.

Unter Linux gibt es keine standardisierten Dateiendungen wie *.exe*, die besagen, ob eine Datei ausführbar ist oder nicht. Stattdessen gibt es Dateirechte. Unter Linux heißt das MapServer-Hauptprogramm daher einfach *mapserv*.

Sie können das *mapserv*-Executable unter Linux durch die Eingabe der URL *http://localhost/cgi-bin/mapserv* aufrufen. Es fehlt hier das Suffix *.exe*. Im Browser sollte nun die gleiche Meldung angezeigt werden wie beim Test mit MS4W:

```
No query information to decode. QUERY_STRING is set, but empty.
```

MapServer-Einstellungen mithilfe einer Mapdatei testen

Um zu überprüfen, ob MapServer auch wirklich in der Lage ist, Karten zu erstellen, verwenden wir die Mapdatei, die wir bereits in Kapitel 10 mit den Kommandozeilenwerkzeugen genutzt haben.

Die verwendete Shape-Datei *countries_simpl* können Sie unter der Adresse http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.1_simpl.tar.gz als komprimiertes Archiv herunterladen.

Wir werden die Mapdatei aus Kapitel 10 im Laufe dieses Kapitels anpassen und in eine Webanwendung einbinden. In diesem Kapitel soll mit MapServer ein eigener HTML-Client aufgebaut werden.

Hierfür werden drei Dateien benötigt:

Mapdatei

Speichert alle für die Erstellung der Karte notwendigen Einstellungen.

Startseite

Enthält die Starteinstellungen und ruft die erste Karte auf.

HTML-Template

Das Template ist eine Vorlage, die Platzhalter für Karte, Legende usw. enthält. Diese Platzhalter werden von MapServer mit den aktuellen Informationen gefüllt.

In Abbildung 11-1 sehen Sie, wie die Anwendung aufgerufen wird. Über ein Formular in der Startseite werden die Einstellungen an unser CGI-Programm geschickt, um aufgrund der Angaben die Karte zu erstellen.

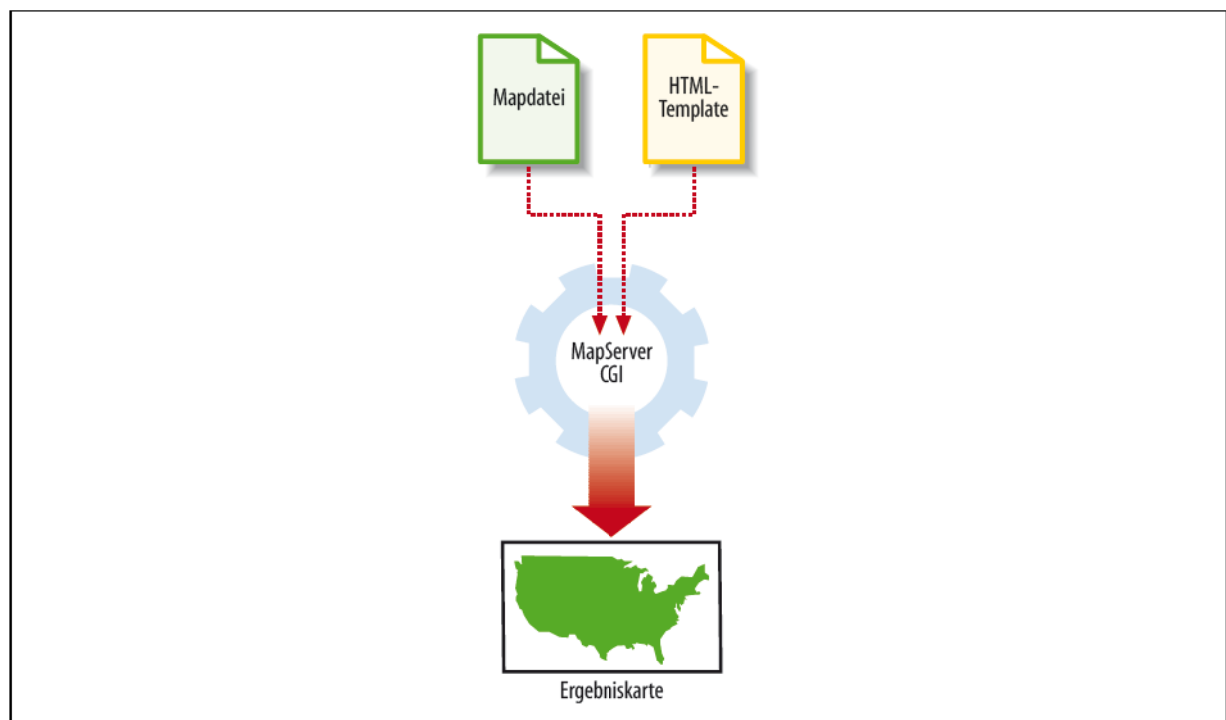


Abbildung 11-1: Die Startseite initialisiert MapServer durch die Übergabe einer Mapdatei und eines HTML-Templates

In Beispiel 11-1 wird die bereits aus den vorherigen Kapiteln bekannte Mapdatei *global.map* verwendet. Durch die Einbindung in das Template wird daraus eine echte, webbasierte Karte. Die zusätzlich erforderlichen Einstellungen in der Mapdatei werden hervorgehoben.

Beispiel 11-1: Beispiel einer Mapdatei, die von einem HTML-Template genutzt wird

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250

  UNITS DD

  SCALEBAR
    STATUS EMBED
    UNITS KILOMETERS
    INTERVALS 3
    TRANSPARENT TRUE
    OUTLINECOLOR 0 0 0
  END

  LEGEND
    STATUS EMBED
    POSITION LR
    TRANSPARENT TRUE
  END

  WEB
    TEMPLATE global.html
    IMAGEPATH "/srv/www/htdocs/tmp/"
    IMAGEURL "/tmp/"
  END
  # Beginn des WEB-Objekts
  # verwendetes HTML-Template
  # Verzeichnis für temp-Dateien
  # URL zu den temporären Dateien
  # Ende des WEB-Objekts

  LAYER
    NAME countries
    TYPE POLYGON
    STATUS DEFAULT
    DATA countries_simpl
    LABELITEM 'NAME'
    LABELMAXSCALE 50000000
    CLASSITEM 'NAME'
    CLASS
      NAME 'Bulgaria'
      EXPRESSION 'Bulgaria'
      STYLE
        OUTLINECOLOR 100 100 100
        COLOR 255 255 150
      END
    LABEL
      SIZE LARGE
      OUTLINECOLOR 255 255 255
      MINFEATURESIZE 40
    # Maßstabsgrenze für Beschriftungen
```


Beispiel 11-1: Beispiel einer Mapdatei, die von einem HTML-Template genutzt wird (Fortsetzung)

```
END
END
CLASS
  NAME 'All Countries'
  EXPRESSION ('[NAME]' ne 'Bulgaria')
  STYLE
    OUTLINECOLOR 100 100 100
    COLOR 200 200 200
  END
END
END
END
```

Wie Sie sehen, waren nur ein paar zusätzliche Zeilen erforderlich, um die Mapdatei statt mit den Kommandozeilenwerkzeugen mit einem Webserver nutzen zu können.



Bei der Verwendung von MS4W sehen die Angaben im WEB-Objekt ein wenig anders aus:

```
WEB
  TEMPLATE global.html
  IMAGEPATH "/ms4w/tmp/ms_tmp/"
  IMAGEURL "/ms_tmp/"
END
```

Es wurde ein komplett neues WEB-Objekt hinzugefügt. Das Objekt enthält drei Parameter:

TEMPLATE global.html

Im WEB-Objekt wird über den Parameter TEMPLATE eine HTML-Seite angegeben. Diese Seite enthält eine Reihe von Platzhaltern, die durch die von MapServer erzeugten Daten ersetzt werden. Hierdurch entsteht schließlich eine HTML-Seite, die eingebettete Karten enthält. Die HTML-Seite muss für die jeweilige Anwendung entsprechend angepasst werden. Die in diesem Kapitel gezeigten Beispiele sind dabei bewusst einfach gehalten. Die Möglichkeiten sind jedoch fast endlos – die Seite kann so gut oder schlecht aussehen, wie Sie wollen.

IMAGEPATH "/srv/www/htdocs/tmp/"

Der Parameter IMAGEPATH gibt an, wo MapServer die generierten Bilder im lokalen Dateisystem speichern soll. Der MapServer muss die Berechtigung haben, Dateien in dieses Verzeichnis zu schreiben.

IMAGEURL "/tmp/"

Mit dem Parameter IMAGEURL wird dem Webbrowser des Benutzers mitgeteilt, unter welcher URL das Ausgabebild zu finden ist. Bei dieser Adresse handelt es sich nicht um den Speicherort im Dateisystem (IMAGEPATH), sondern um die Webadresse des Verzeichnisses. In diesem Beispiel wird /tmp/ dem DocumentRoot von Apache angehängt. Die URL zur Anzeige des Bilds lautet *http://localhost/tmp/bulgaria.png*.

Zusätzlich wurde für die Ebene `countries` die Anweisung `LABELMAXSCALE 50000000` definiert. Auf diese Weise werden die Beschriftungen erst ab einem bestimmten Maßstab angezeigt. Der Maßstab der anfänglichen Karte ist so gering, dass Bulgarien kaum sichtbar ist. Wenn Sie dagegen auf einen größeren Maßstab heranzoomen, werden auch die Beschriftungen angezeigt.

Sobald die notwendigen Einstellungen in der Mapdatei vorgenommen sind, können Sie eine Startseite zur Initialisierung der Applikation einrichten. (Bei uns heißt sie *index.html*. Sie können den Namen aber frei wählen.) In Beispiel 11-2 sehen Sie, welche Teile diese Seite enthalten muss.



Um die Beispiele nicht schwerer zu machen als notwendig, können Sie alle hier vorgestellten Dateien im DocumentRoot-Verzeichnis Ihres Web-servers ablegen. Später wollen Sie die Website vermutlich woanders ablegen, aber in unserem Beispiel verwenden wir folgende Speicherorte:

- Linux: `/srv/www/htdocs/`
- Windows mit MS4W: `\ms4w\apache\htdocs\`

Beispiel 11-2: Der Quellcode der MapServer-Initialisierungsdatei *index.html*

```
<html>
<head><title>MapServer Test</title></head>
<body>
<center><h2>MapServer Test</h2>

<form method="GET" action="/cgi-bin/mapserv">
  <input type="hidden" name="map" value="/srv/www/htdocs/global.map" />
  <input type="hidden" name="layer" value="countries" />

  <input type="hidden" name="zoomdir" value="1" />
  <input type="hidden" name="zoomsize" value="2" />

  <input type="hidden" name="program" value="/cgi-bin/mapserv" />

  <input type="submit" value="Start MapServer" />
</form></center></body></html>
```

Dieses Beispiel enthält nur die absolut notwendigen Bestandteile. Es gibt einen Titel und eine Überschrift. Der restliche HTML-Code befindet sich in einem Formular und wird benötigt, um MapServer zu starten. Diese HTML-Seite kann komplett an die eigenen Bedürfnisse angepasst werden, solange das Formular und die enthaltenen Variablen nicht verändert werden.

Unsere Beispielseite enthält nur einen Button mit der Beschriftung »Start MapServer«. Sie beinhaltet ein paar versteckte Elemente, deren Einstellungen beim Abschicken des Formulars an MapServer übergeben werden. Diese Elemente können Sie sich anzeigen lassen, indem Sie `type="hidden"` auf `type="text"` umstellen. Nach dem Start der Anwendung wird die Seite *index.html* nicht mehr benötigt.



CGI-Programme funktionieren so ähnlich wie Börsenmakler. Diese erhalten Anfragen, Informationen und eine Provision. Im Austausch erhält der Kunde eine Reihe von Aktien.

MapServer-basierte CGI-Programme erhalten Kartenanfragen. Hierzu gehören Informationen darüber, welcher Bereich der Karte und welche Ebenen angezeigt werden sollen. Basierend auf diesen Daten werden dann die gewünschten Karteninhalte zurückgegeben.

Obwohl die Seite *index.html* für die Applikation nicht unbedingt notwendig ist, erleichtert sie doch den Einstieg. Über die *index.html*-Seite werden die Einstellungen über GET an MapServer übergeben. Sie können die Einstellungen auch im Aufruf übergeben und so MapServer direkt ansprechen. Nach der anfänglichen Anfrage werden sämtliche Einstellungen und Request-Informationen in der URL kodiert und zur weiteren Verarbeitung an das CGI-Programm geschickt.

Nach dem Klick auf den Start-Button sendet das MapServer-CGI-Programm eine neue Webseite an den Benutzer. Die URL für diese Seite enthält eine Reihe von Variablen (ähnlich den in der *index.html* definierten). Fordert der Benutzer eine Änderung der Karte an, werden diese Variablen zur erneuten Verarbeitung wieder an das CGI-Programm geschickt. Der Zweck der *index.html* besteht darin, diesen Prozess anzustoßen, ohne hierfür eine komplexe URL zusammenstellen zu müssen.

Wie Sie aus Beispiel 11-1 wissen, wurde in der Mapdatei die Datei *global.html* als Template definiert. Hierbei handelt es sich ebenfalls um ein HTML-Dokument. Bevor MapServer diese Seite an den Browser zurückschickt, tauscht das Programm bestimmte Platzhalter im Dokument gegen eigene Angaben aus. Den Quellcode zu *global.html* sehen Sie in Beispiel 11-3. Schauen Sie sich die HTML-Seite einfach mal im Browser an.

Beispiel 11-3: Der Quellcode des MapServer-Templates *global.html*

```
<html>
<head><title>MapServer Test</title></head>
<body>
  <center><h2>MapServer Test</h2>
  <hr>

  <form method="GET" action="/cgi-bin/mapserv">
    <input name="img" type="image" src="[img]" width=600 height=300 border=0
      alt="Map Image">

    <input type="hidden" name="zoomdir" value=1 [zoomdir_1_check] />
    <input type="hidden" name="zoomsize" size=4 value=[zoomsize] />

    <input type="hidden" name="imgxy" value="300 150" />
    <input type="hidden" name="imgext" value="[mapext]" />
    <input type="hidden" name="map" value="[map]" />
    <input type="hidden" name="savequery" value="true" />
    <input type="hidden" name="mapext" value="shapes" />

  </form></center>
</body></html>
```

Wenn Sie sich mit HTML-Formularen und -Tags auskennen, wird Ihnen aufgefallen sein, dass auf der Seite nur eine Überschrift, der Titel und ein Bild zu sehen sind. Die Bildquelle (src) wird von MapServer beim Erzeugen der Seite eingefügt.

MapServer überprüft hierbei die Datei *global.html* auf MapServer-Variablen wie [img]. In unserem Beispiel ersetzt MapServer diesen mit der URL für das zuvor erstellte Kartenbild.



MapServer-Variablen werden in eckigen Klammern angegeben. Neben [img] gibt es noch eine ganze Reihe weiterer Variablen, die im Template verwendet werden können. Die Platzhalter [legend] und [scalebar] werden wir später in diesem Kapitel noch genauer betrachten.

Abbildung 11-2 zeigt die über das Template *global.html* erstellte fertige Webseite.

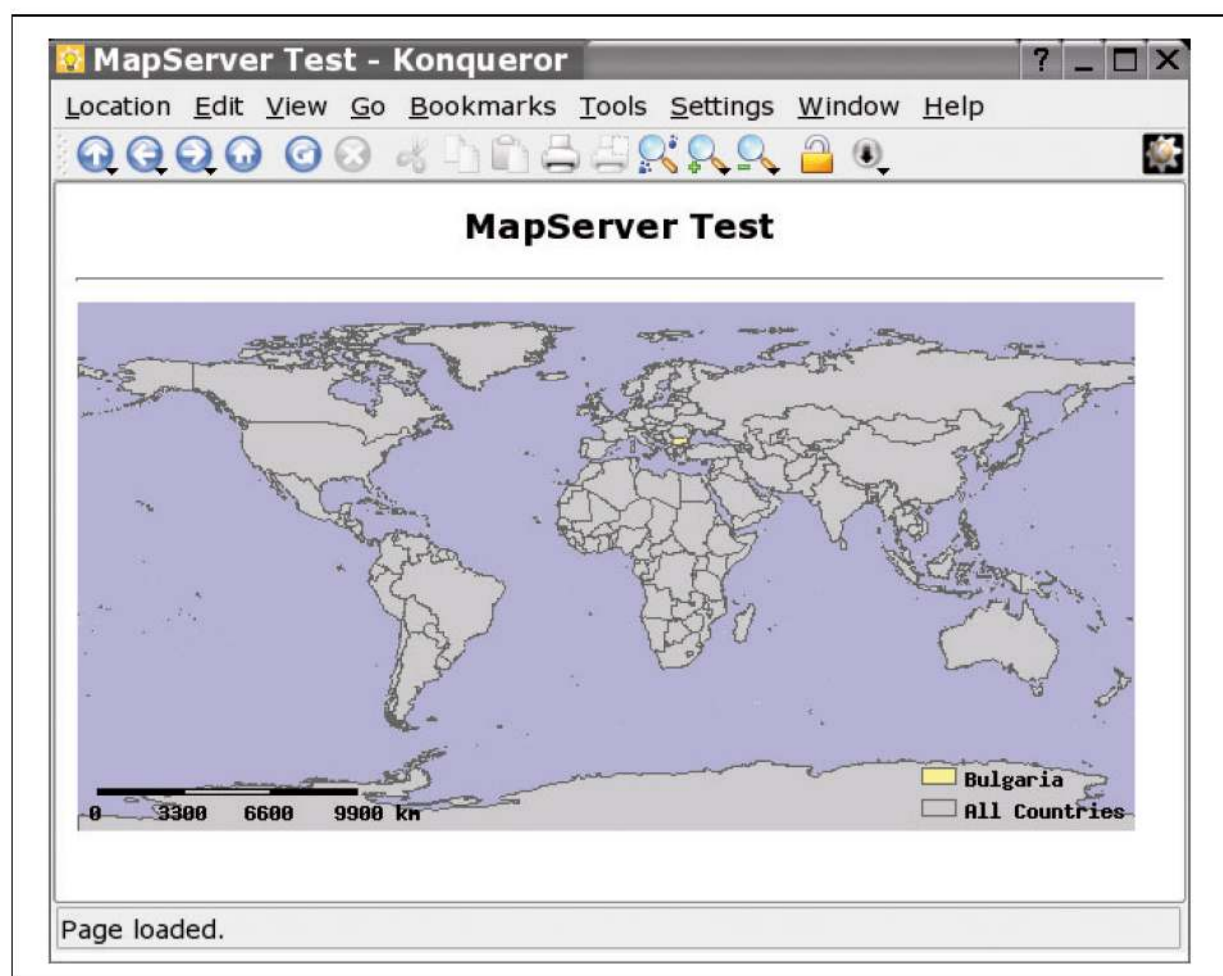


Abbildung 11-2: Beispiel einer einfachen webbasierten Anwendung

Die Karte sieht genau so aus wie die, die Sie in Kapitel 10 über das Kommandozeilenwerkzeug *shp2img* erstellt haben. Der Unterschied besteht aber darin, dass diese Karte auf eine Benutzeranfrage von MapServer erstellt und in eine Webseite eingebettet wurde.

Das einfache Template *global.html* enthält keinerlei Steuerungselemente – jedenfalls keine sichtbaren. Mit *Steuerungselementen* meinen wir Buttons und Auswahlfelder, die Teil vieler webbasierter Anwendungen sind. Wir haben diese Elemente aus Gründen der Einfachheit erst einmal weggelassen, werden sie aber in den folgenden Beispielen noch kennenlernen.

Um in der Karte zu interagieren, brauchen Sie nur in die Karte zu klicken, und die Darstellung wird vergrößert. Klicken Sie einmal in die Karte und warten Sie, bis die Karte aktualisiert wurde. Klicken Sie noch einmal. Mit jedem Klicken zoomen Sie weiter in die Karte hinein. Je weiter Sie hereinzoomen, desto deutlicher wird die Beschriftung für Bulgarien. In Abbildung 11-3 sehen Sie, wie die Karte nach zweimaligem Klicken dargestellt wird.

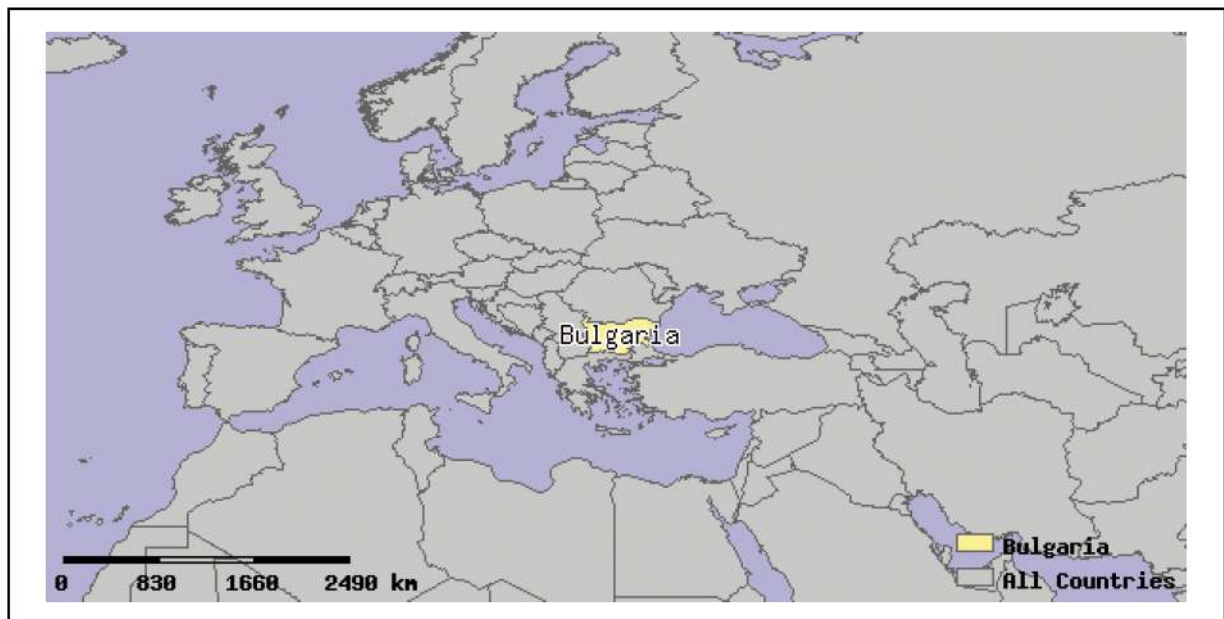


Abbildung 11-3: Durch zweimaliges Klicken auf die Karte wurde die Ansicht von Bulgarien herangezoomt

Da wir in diesem Template absichtlich auf Steuerungselemente verzichtet haben, gibt es keinen Button zum Herauszoomen. In dieser einfachen Anwendung benutzen Sie dazu einfach den Zurück-Button des Browsers. Dies funktioniert sogar recht intuitiv, weil eigentlich jeder Anwender das Zurückgehen zu vorherigen Seiten in dieser Form bereits gewohnt ist.

Wenn Sie es mit Ihrer Anwendung bis hierher geschafft haben, ist MapServer eingerichtet und funktioniert korrekt. Natürlich gibt es eine Vielzahl weiterer Optionen und Einstellungsmöglichkeiten. Einige finden Sie in den folgenden Abschnitten, in denen wir unser Beispiel weiter ausbauen werden.

Anpassen des Startbereichs

Je weiter Sie Ihre Kenntnisse im Umgang mit MapServer vertiefen, desto mehr unterschiedliche Einstellungsmöglichkeiten werden Sie kennenlernen. In den Beispielen dieses Kapitels werden wir allerdings nur einen sehr kleinen Teil dieser Optionen zeigen können. Eine vollständige Liste der Einstellungen für die Mapdatei finden Sie im entsprechenden Referenzdokument unter <http://mapserver.gis.umn.edu/docs/reference>.

Beachten Sie die vielen in diesem Kapitel verwendeten MapServer-CGI-Variablen. Nähere Informationen hierzu finden Sie im Template-Referenzdokument unter <http://mapserver.gis.umn.edu/docs/reference/templatereference/templatereference>.

Anpassen des anfänglichen Darstellungsbereichs der Karte

In Kapitel 10 haben Sie gesehen, wie Sie den Darstellungsbereich der Karte mit dem Kommandozeilenwerkzeug `shp2img` beeinflussen können. Bei interaktiven Karten kann der Darstellungsbereich durch einfaches Heranzoomen verändert werden. In dem vorangegangenen Beispiel wird als Startbereich die ganze Welt angezeigt. Über die Anpassung des Parameters `EXTENT` können Sie den Startbereich ändern.

Die folgenden Erläuterungen basieren auf den bisherigen Beispielen in diesem Kapitel, die Datei *global.map* eingeschlossen. Die ursprünglichen Einstellungen lauteten:

```
EXTENT -180 -90 180 90
```

Nun wollen wir eine Karte von Kanada erstellen. Hierfür benutzen wir weiterhin die Shape-Datei *countries_simpl.shp* und verwenden die Werkzeuge `ogr2ogr` sowie `ogrinfo`, um zu ermitteln, welche Angaben Sie für den Darstellungsbereich von Kanada machen müssen. Die Programme `ogr2ogr` und `ogrinfo` haben Sie schon in Kapitel 10 genutzt. Schauen Sie sich dazu nochmals das Beispiel 10-5 an, in dem die Ausdehnung von Bulgarien ermittelt wurde.

Die genannten Kommandozeilenwerkzeuge sind nicht Teil von MapServer, sondern des in Kapitel 3 beschriebenen GDAL/OGR-Pakets.

Mit `ogr2ogr` extrahieren Sie die Umrisse von Kanada und speichern sie in einer eigenen Datei. Danach benutzen Sie `ogrinfo`, um zu ermitteln, welche Koordinaten für die Darstellung von Kanada benötigt werden.

Die neu erstellte Shape-Datei wird nur zwischenzeitlich benutzt, um die Koordinaten des für uns interessanten Bereichs zu ermitteln. Diese verwenden Sie anschließend als Werte für `EXTENT` in der Mapdatei *global.map*, wodurch MapServer nun zu Beginn einen Kartenausschnitt ausgibt, der Kanada zeigt.

Die von `ogrinfo` für die Ausdehnung von Kanada ausgegebenen Koordinaten lauten:

```
Extent: (-141.000000, 41.675980) - (-52.636291, 83.110458)
```

Ersetzen Sie nun die Angaben des Parameters `EXTENT` in der Mapdatei. Die ersten Zeilen der aktualisierten Mapdatei sehen Sie in Beispiel 11-4.

Beispiel 11-4: Die angepasste Mapdatei für die Darstellung von Kanada

```
MAP
  SIZE 600 300
  EXTENT -141 42 -52 83
  ...
  CLASS
    NAME 'Canada'
    EXPRESSION 'Canada'
    OUTLINECOLOR 100 100 100
    COLOR 255 255 150
  END
  ...
```

Es müssen keine exakten Koordinaten angegeben werden. Sie können die Zahlen runden oder die Dezimalstellen weglassen. Beachten Sie die Änderungen im LAYER-Objekt. Die Klasse, in der Bulgarien hervorgehoben wurde, wird nun für die spezielle Darstellung von Kanada verwendet. Außerdem wurde das LABEL-Objekt aus dem CLASS-Objekt entfernt. Es wird also keine Beschriftung erzeugt. In Abbildung 11-4 sehen Sie die Karte, die mit diesen neuen Einstellungen erzeugt wurde.

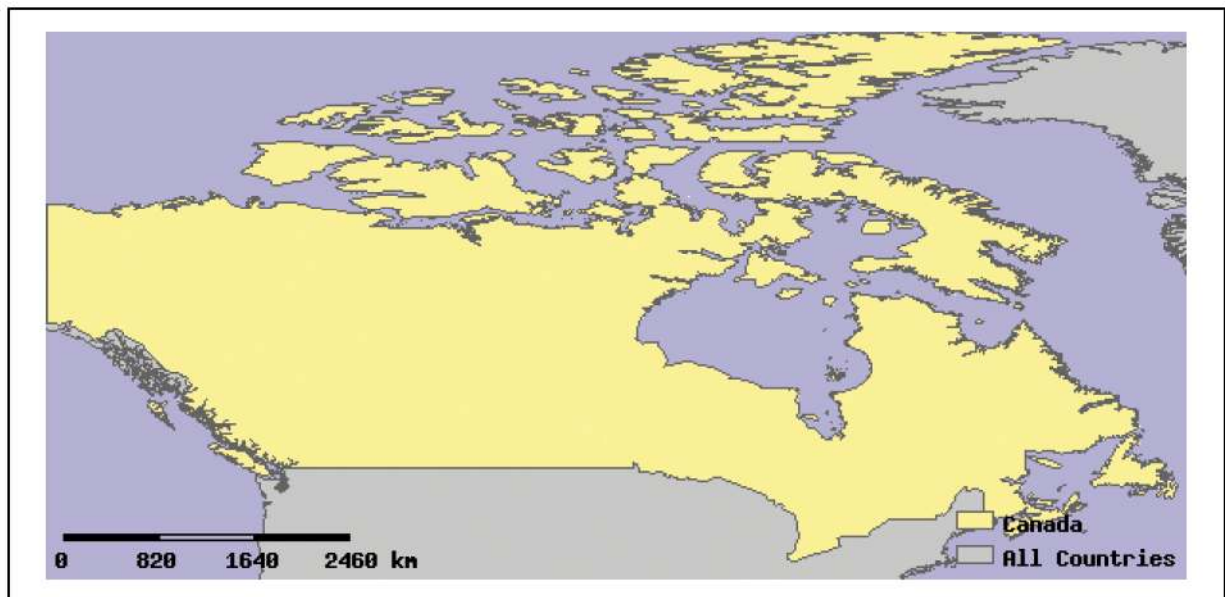


Abbildung 11-4: Die neue Karte zeigt eine Ausschnittvergrößerung von Kanada und der unmittelbaren Umgebung



MapServer war bei der Anzeige des geforderten Darstellungsbereichs sehr genau. Das ist eigentlich eine gute Sache. Andere Programme erweitern diesen Bereich dagegen von sich aus – für die schnelle Erstellung einer Grafik sicherlich hilfreich. Soll die Darstellung allerdings möglichst genau sein, ist diese Eigenschaft eher hinderlich.

Damit Kanada im Zusammenhang mit den angrenzenden Ländern dargestellt wird, erweitern Sie den Darstellungsbereich um jeweils fünf Grad:

Ursprünglicher Darstellungsbereich

EXTENT -141 42 -52 83

Angepasster Darstellungsbereich

EXTENT -146 37 -47 88

Die ersten zwei Zahlen wurden reduziert, um mehr vom Süden und Westen der Karte zu zeigen. Die letzten beiden Zahlen wurden vergrößert, damit ein größerer Bereich nordöstlich von Kanada zu sehen ist. Abbildung 11-5 zeigt die beiden Darstellungsbereiche, Abbildung 11-6 zeigt die Karte, die aus dem angepassten Darstellungsbereich resultiert.

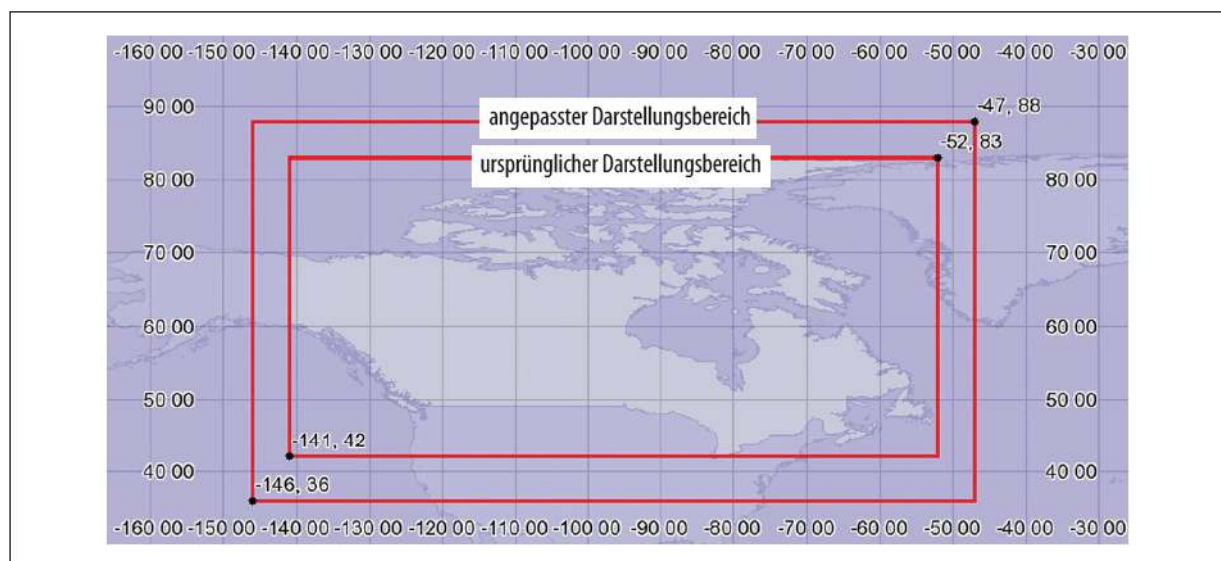


Abbildung 11-5: Der ursprüngliche und der angepasste Darstellungsbereich für die Karte von Kanada



Um die Änderungen der Projektion im folgenden Abschnitt besser veranschaulichen zu können, haben wir die Karte mit einem Gitternetz überlagert. Beachten Sie, dass sich das Gitter aus Rechtecken aufbaut und sämtliche Linien im rechten Winkel zueinander stehen.

Ändern der Projektion der Karte

Über `PROJECTION`-Objekte können Sie die Projektion für die Karte oder die einzelnen Ebenen angeben. Wurden in der Mapdatei keine `PROJECTION`-Objekte definiert, geht MapServer davon aus, dass für alle Ebenen das gleiche Koordinatenreferenzsystem verwendet wird und keine Umprojektion notwendig ist. Für Daten, die in Dezimalgraden angegeben werden, verwendet MapServer eine Standardprojektion (Plate Carée bzw. die äquidistante zylindrische Projektion). Diese Projektion ist nicht sonderlich geeignet, um die Karte von Kanada darzustellen, da Teile der Karte verzerrt werden. Ein Großteil von Kanada befindet sich in Polnähe, die über diese Projektionsmethode besonders stark verzerrt wird.

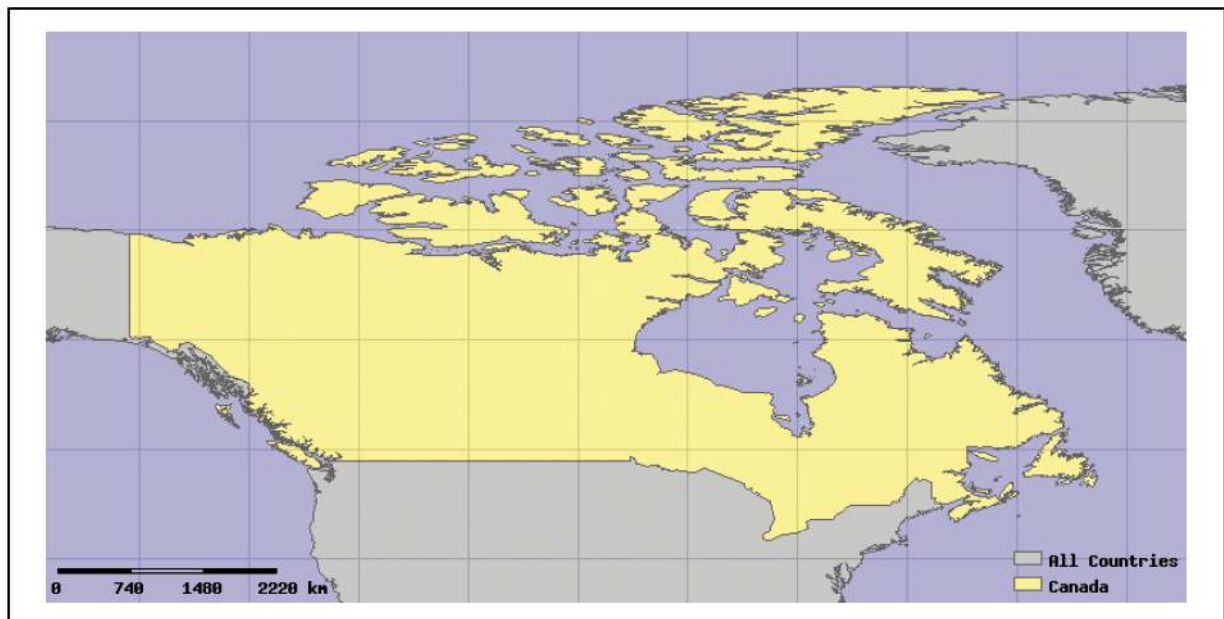


Abbildung 11-6: Durch eine leichte Anpassung des Darstellungsbereichs wird die Darstellung von Kanada deutlicher



In diesem Kapitel finden Sie einige Beispiele für die Anpassung der Projektion. Eine ausführliche Einführung in Kartenprojektionen finden Sie in Anhang A.

Angabe der Karten- und Ebenenprojektion

In der Mapdatei kann jede Ebene, aber auch die Karte selbst eine eigene Projektion besitzen. Wenn die Projektion der Layer nicht der Projektion der Ausgabekarte entspricht, muss MapServer die Quelldaten für die Erstellung der Karte umprojizieren. In diesem Fall muss die Projektion für jede Ebene angegeben werden, damit MapServer weiß, wie die Quelldaten für die gewünschte Ausgabeprojektion umgewandelt werden müssen. Liegt in den Ebenen keine Angabe zur Projektion vor, geht MapServer davon aus, dass die gleiche Projektion wie die des MAP-Objekts vorliegt.

Das erste PROJECTION-Objekt in der Mapdatei gibt die Ausgabeprojektion an und ist Teil des MAP-Objekts. Weitere Projektionen werden innerhalb der LAYER-Objekte definiert. Hiermit wird MapServer mitgeteilt, welche Projektion für die Daten dieser Ebene verwendet wird. Unterscheiden sich diese Angaben von denen für das MAP-Objekt, muss MapServer die Quelldaten der Ebene für die Ausgabeprojektion umprojizieren.

Da sowohl die Ausgabekarte wie auch die Quelldaten in Dezimalgrad angegeben sind, wurden in *global.map* bisher keine weiteren Angaben zur verwendeten Projektion benötigt. Erweitern Sie die Karte jedoch um eine Ebene, die ein anderes Koordinatenreferenzsystem verwendet, wird diese ohne weitere Informationen nicht korrekt angezeigt. MapServer benötigt die Angaben zur Projektion, um die Objekte von einer in die andere Projektion zu überführen. In Beispiel 11-5 wird das PROJECTION-Objekt sowohl im MAP-Objekt als auch für die einzelnen Ebenen explizit definiert.

Beispiel 11-5: Angabe der Projektion für die Ausgabekarte und die Ebenen

```
...
UNITS DD

PROJECTION
  "proj=latlong"
  "ellps=WGS84"
END

SCALEBAR

...
LAYER
  NAME countries
  TYPE POLYGON
  ...
  CLASS
    NAME 'All Countries'
    OUTLINECOLOR 100 100 100
    COLOR 200 200 200
  END
END
END
```

Wie Sie sehen, sind in Beispiel 11-5 die Projektionen für Karte und Ebenen identisch. Auch wenn eine Definition der Projektion in diesem Fall nicht notwendig wäre, dient die Angabe der Übersichtlichkeit. Auf diese Weise wird die spätere Erweiterung um weitere Ebenen mit unterschiedlichen Referenzsystemen erleichtert.

Wahl der richtigen Kartenprojektion für die Ausgabe

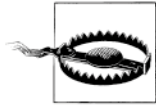
Wenn für die Ausgabe der Karte eine andere Kartenprojektion verwendet werden soll, gibt es bei der Angabe zwei Dinge zu beachten. Zu Beginn müssen Sie die Definitionen für die Projektion des MAP-Objekts anpassen (das erste PROJECTION-Objekt in Beispiel 11-5). Die Ermittlung der zu verwendenden Projektion würde den Umfang dieses Kapitels sprengen. Sie sollten allerdings überlegen, welchen Bereich die Karte darstellen soll. Handelt es sich um eine Weltkarte, eine Landkarte oder um eine Karte, die nur eine kleine Region zeigen soll?



Auf der Website des Atlas of Canada gibt es einige gute Seiten über kartografische Konzepte, Projektionen und ihre Anwendung in der Web-Kartografie. Die Adresse lautet http://atlas.gc.ca/site/english/learningresources/carto_corner/index.html.

In diesem Beispiel wollen wir eine Karte von Kanada erstellen. Wenn Sie eine speziell auf Kanada ausgelegte Projektion verwenden, sollten Sie sicherstellen, dass die Benutzer nicht versuchen, Gegenden außerhalb von Kanada anzuzeigen, da die Projektion für andere Gegenden der Welt nicht geeignet ist.

Außerdem müssen Sie bei der Wahl Ihrer Kartenprojektion die Maßeinheit für das verwendete Koordinatenreferenzsystem beachten. In den bisherigen Beispielen haben wir Dezimalgrade als Maßeinheit verwendet. Andere Projektionen benutzen stattdessen möglicherweise Meter oder Fuß. In diesem Beispiel legen wir Meter als Maßeinheit fest.



Wurden die Projektionseinstellungen für die Ebenen einmal korrekt definiert, werden sich diese nur noch ändern, wenn sich die Daten selbst ändern. In der Regel werden Sie die Änderungen an der Ausgabeprojektion der Karte vornehmen, nicht jedoch die Projektion der Quelldaten ändern.

Im folgenden Beispiel werden wir die *LCC-Projektion* (*Lambert Conformal Conic Projection*) einsetzen – ein Projektionsmodell, das häufig im *Atlas of Canada* verwendet wird. Für diese Projektion müssen Sie bestimmte Angaben machen, die die Darstellung auf Kanada zentrieren und die Bereiche dieses Kartenausschnitts sinnvoll anzeigen. Diese Projektion eignet sich speziell für Kanada und ist für andere Bereiche nicht geeignet.

EPSG-Codes oder Projektionsparameter

In der Mapdatei können Projektionen auf zwei verschiedene Arten angegeben werden. Einerseits können Sie den EPSG-Code angeben, der auf die Details zur Projektion verweist, oder Sie geben die Projektionsparameter direkt in der Mapdatei an.

Die EPSG-Codes werden vom Surveying & Positioning Committee der International Organisation of Oil and Gas Producers (OGP) verwaltet. Der EPSG-Code ist eine Zahl, die auf weitere Parameter zur Beschreibung der jeweiligen Projektion verweist. Die Liste der EPSG-Codes und der Detailinformationen können Sie über die Website der EPSG herunterladen. Bei MapServer befinden sich die Codes in einer Textdatei namens *epsg*, auf die die PROJ.4-Bibliothek zugreift. Die PROJ.4-Bibliothek ermöglicht es MapServer, Koordinaten zur Laufzeit umzuprojizieren. Mehr zu den EPSG-Codes und PROJ.4 finden Sie in Anhang A.

Die zweite mögliche Angabe der Projektion besteht darin, die Parameter zur Projektion direkt in der Mapdatei anzugeben. Beispiel 11-6 zeigt die beiden Arten, über die die Projektion angegeben werden kann – einmal über den EPSG-Code und anschließend anhand der detaillierten Beschreibung. Beide Methoden können verwendet und auch parallel eingesetzt werden.

Beispiel 11-6: Die beiden möglichen Angaben von Projektionsinformationen mittels des EPSG-Codes oder über die direkte Auflistung der Parameter zur Projektion

```
PROJECTION
  "init=epsg:42304"
END
```

```
PROJECTION
  "proj=lcc"
  "ellps=GRS80"
```


Beispiel 11-6: Die beiden möglichen Angaben von Projektionsinformationen mittels des EPSG-Codes oder über die direkte Auflistung der Parameter zur Projektion (Fortsetzung)

```
"lat_0=49"  
"lon_0=-95"  
"lat_1=49"  
"lat_2=77"  
"datum=NAD83"  
"units=m"  
"no_defs"  
END
```

Es ist offensichtlich, dass die EPSG-Methode viel einfacher anzugeben ist. Hierbei müssen Sie darauf achten, dass der verwendete EPSG-Code auch in der *epsg*-Datei vorliegt.

Die Ermittlung der korrekten Projektion und des richtigen EPSG-Codes ist möglicherweise ein wenig aufwendig. Bei der Wahl der Projektion für Kanada kann es helfen, sich daran zu orientieren, was andere Leute benutzen. Auf der Website des Atlas of Canada ist beispielsweise beschrieben, welche Projektionsmethoden zum Einsatz kommen. Dort finden Sie Informationen über die Einstellungen Ihres Webservice, inklusive der verwendeten EPSG-Codes (siehe http://atlas.gc.ca/site/english/dataservices/web_map_service.html).

PROJ.4 liegt eine Liste der wichtigsten EPSG-Codes bei. Diese kann um eigene Codes erweitert werden. Einige der für Kanada spezifischen Codes, wie beispielsweise 42304, sind keine offiziellen von der EPSG entwickelten Codes. Stattdessen haben andere Stellen ihre eigenen Erweiterungen entwickelt. Benötigen Sie speziell die Codes der 42xxx-Reihe, können Sie die entsprechende EPSG-Definitionsdatei unter der Adresse <http://maptools.org/dl/proj4-epsg-with-42xxx.zip> herunterladen.

Öffnen Sie die EPSG-Datei mit einem Texteditor, können Sie die detaillierten Projektionseinstellungen für die verschiedenen EPSG-Codes sehen. Die Definition für den Code 42304 sehen Sie im folgenden Beispiel. Die *epsg*-Datei liegt im Verzeichnis *proj* vor. Bei MS4W finden Sie die Datei unter dem Pfad */ms4w/proj/nad/epsg*; unter Linux lautet der Standardpfad */usr/local/share/proj/epsg*.

```
<42304> +proj=lcc +ellps=GRS80 +lat_0=49 +lon_0=-  
95 +lat_1=49 +lat_2=77 +datum=NAD83 +units=m no_defs <>
```

Die Projektionsinformationen stehen alle in einer Zeile. Aus Platzgründen wurde ein Umbruch eingefügt. Beachten Sie die *<42304>*. Dies ist unsere Codenummer. Der Rest der Zeile enthält die Parameter.

Wenn Sie nochmals einen Blick auf Beispiel 11-6 werfen, sehen Sie, wie die Projektionsdetails in ein für MapServer passendes Format gebracht werden. Entfernen Sie die Pluszeichen und umgeben Sie die jeweilige Angabe mit doppelten Anführungszeichen. Indem Sie die Angaben jeweils auf eine eigene Zeile in der Mapdatei stellen, können Sie die Lesbarkeit deutlich erhöhen.

Die verschiedenen Einstellungen übergeben MapServer die notwendigen Informationen für die Projektion der Daten:

- Projektion: `proj=lcc`
- Ellipsoid: `ellps=GRS80`
- Bezugssbreite: `lat_0=49`
- Hauptmeridian: `lon_0=-95`
- Erste/zweite Standardparallele: `lat_1=49/lat_2=77`
- Datum: `datum=NAD83`
- Maßeinheit, hier Meter: `units=m`

Weitere Informationen zu Kartenprojektionen und den möglichen Einstellungen finden Sie in Anhang A.

Anpassung des Darstellungsbereich der Karte

Damit die Karte korrekt dargestellt wird, reicht es nicht aus, die Projektion der Karte zu ändern. Sie müssen auch den Darstellungsbereich der Karte in den Einheiten der verwendeten Projektion angeben (`units=m`). In unserem Fall werden nun statt Dezimalgrad Meter angegeben. Um zu ermitteln, welcher Darstellungsbereich für Ihre Karte der richtige ist, kann es helfen, sich ein wenig mit den verschiedenen Projektionsmodellen zu befassen.

Das folgende Beispiel kann Ihnen helfen, den richtigen Darstellungsbereich für Ihre Karte zu finden. Der Darstellungsbereich der Karte wird in Metern angegeben. Die Angaben beziehen sich dabei auf die Bezugssbreite und den Hauptmeridian der Projektion. Wie Sie in Beispiel 11-6 sehen können, liegt der Hauptmeridian in unserem Fall bei einer Länge von -95° , und die Bezugssbreite liegt bei 49° Nord. Um den Darstellungsbereich zu definieren, geben Sie die Entfernung der südwestlichen und der nordöstlichen Ecke der Karte in Metern vom Punkt $-95^\circ, 49^\circ$ an. Zum Beispiel:

```
EXTENT -3000000 -1000000 3000000 1000000
```

Durch diese Angaben wird die Ausdehnung der Karte folgendermaßen definiert: Die südwestliche Ecke liegt 3.000 km westlich und 1.000 km südlich vom Kartenmittelpunkt entfernt, die nordöstliche Ecke liegt 3.000 km östlich und 1.000 km nördlich davon. Mithilfe des Gitters in Abbildung 11-5 können Sie sehen, wo sich in etwa der Kartenmittelpunkt befindet. Obwohl er sich in der Mitte der Abbildung befindet, liegt er geografisch eher südlich in der Nähe der Grenze zu den USA. Mit diesen Informationen können Sie leicht selbst entscheiden, wie groß der Darstellungsbereich Ihrer Karte werden soll. Abbildung 11-6 gibt Ihnen einen Eindruck der Entfernungen. Mithilfe des Maßstabs können Sie leicht die Gesamtbreite des Landes einschätzen. Selbst eine Gesamtbreite von 6.000 km scheint nicht übertrieben. Abbildung 11-7 zeigt die Ergebniskarte bei Angabe der obigen Ausdehnung.

Unsere ursprüngliche Schätzung war also gar nicht so schlecht, obwohl die Karte zu weit nach Süden und nicht weit genug nach Osten reicht. Mit ein paar Änderungen und Anpassungen bekommen Sie schnell die optimalen Angaben heraus. Wir reduzieren

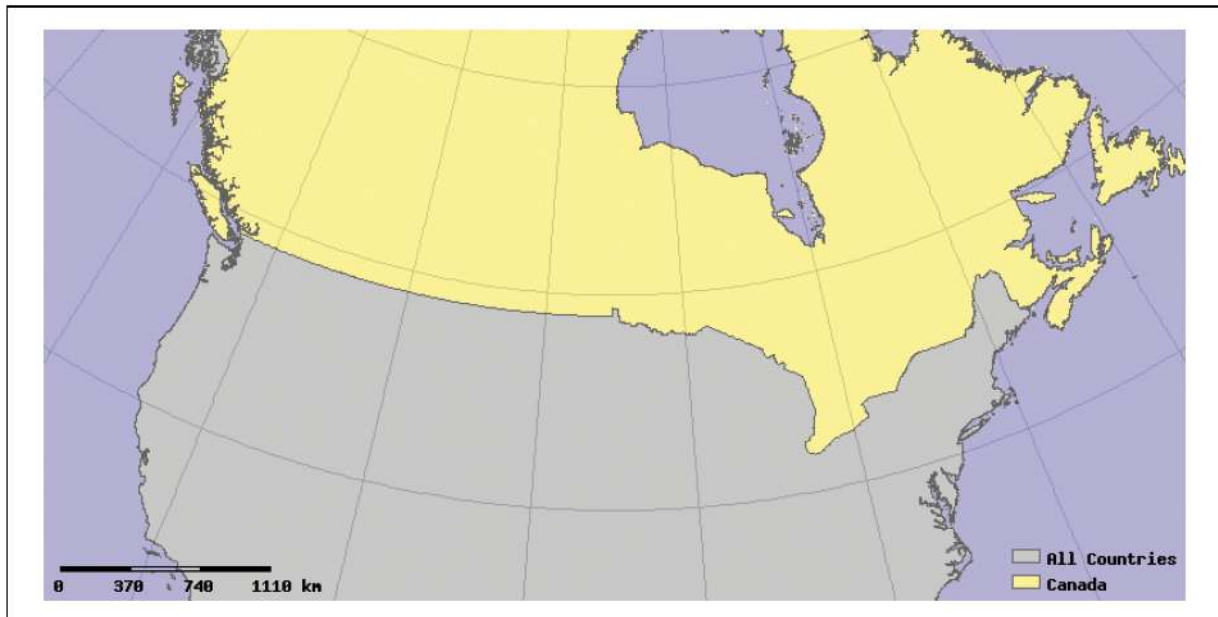


Abbildung 11-7: Eine erste Schätzung für die Einstellung des Darstellungsbereichs der Karte von Kanada in der LCC-Projektion

den minimalen y-Wert ein wenig, gleichzeitig wird der maximale y-Wert etwas erhöht. Probieren Sie ruhig einmal ein paar Varianten aus. Hier eine EXTENT-Angabe, die schon recht gut aussieht:

```
EXTENT -2400000 -900000 3100000 4000000
```

Anhand dieser Angaben hat MapServer die Karte in Abbildung 11-8 erzeugt. Auch hier haben wir die Karte wieder mit einem Gitternetz überlagert, um die Änderung der Form der Objekte zu verdeutlichen, die mit einer Projektionsänderung einhergehen. Sehen Sie im Vergleich dazu die gleichmäßig rechteckige Darstellung in Abbildung 11-6.

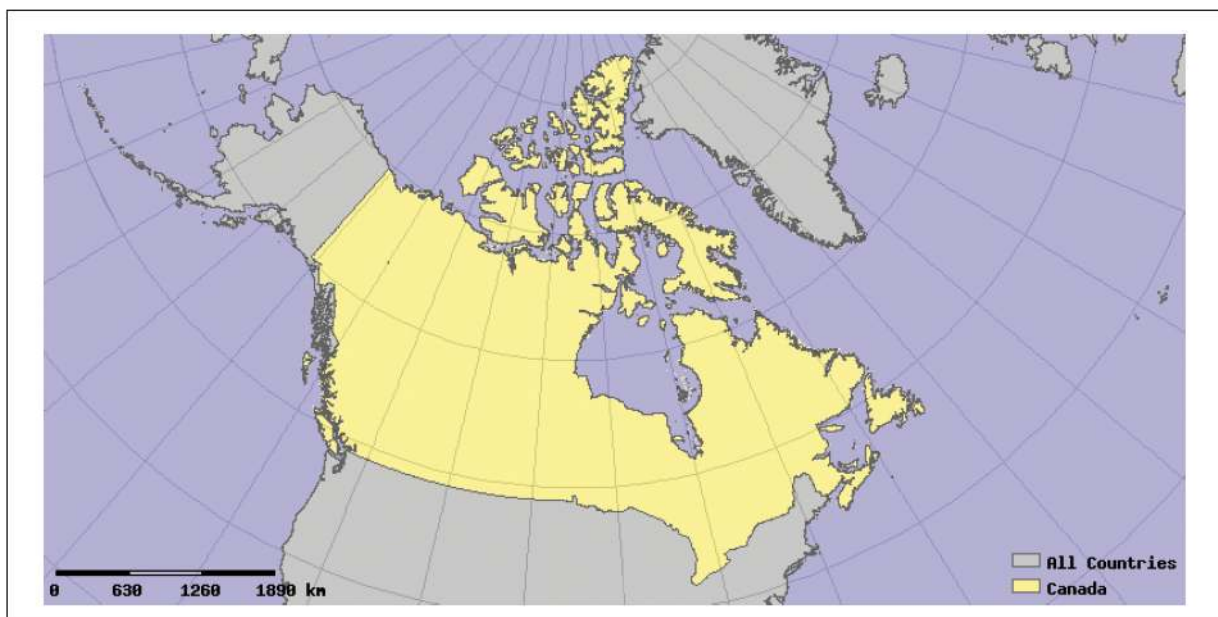


Abbildung 11-8: Der korrekt definierte Darstellungsbereich für die Karte von Kanada

Anpassen der Bildgröße an den Darstellungsbereich

Links und rechts der Karte aus Abbildung 11-8 gibt es jetzt etwas zu viel Platz. Das lässt sich aber leicht dadurch beheben, dass wir die Größe der Karte anpassen.

Unsere ursprüngliche Kartenanwendung sollte die gesamte Erde anzeigen. Hierbei haben wir eine Projektion verwendet, bei der die Weltkarte etwa doppelt so breit ist wie hoch. Während der letzten Beispiele haben wir den Parameter `SIZE` mit 600×300 Pixeln angegeben, was einem Verhältnis von 2:1 von Breite zu Höhe entspricht. Die jetzige Karte von Kanada in der LCC-Projektion ist dagegen fast quadratisch. Wenn Sie das Verhältnis auf 1:1 ändern, sollte die Kartengröße besser passen. Ändern Sie die Größe einmal auf 400×400 oder 600×600 . Hierfür müssen Sie im Map-Objekt die Angaben für den Parameter `SIZE` anpassen: `SIZE 600 600`.

Wenn Sie sich das Ergebnis in Abbildung 11-9 anschauen, werden Sie feststellen, dass diese Angaben für die Darstellung von Kanada wesentlich besser geeignet sind.

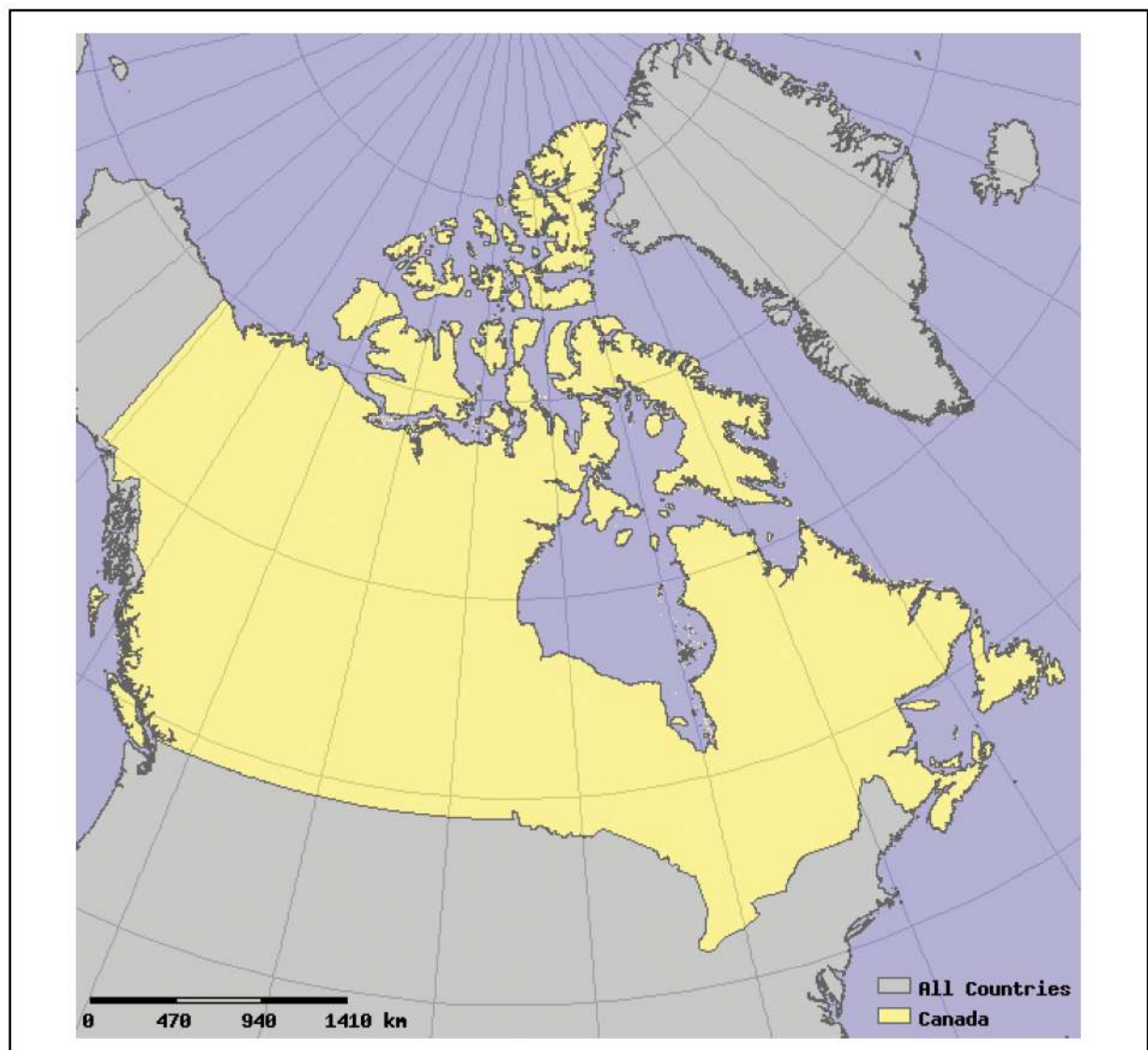


Abbildung 11-9: Karte von Kanada mit der Darstellung von Höhe und Breite im Verhältnis 1:1

Bevor Sie die Karte im Browser betrachten, müssen Sie die Bildgröße auch im HTML-Template *global.html* anpassen. Bisher haben wir diese Angaben manuell in die Datei eingetragen. Um die Karte flexibler zu gestalten, können Sie diese Werte aber durch MapServer-Variablen ersetzen. Auf diese Weise kann die Bildgröße im HTML-Template automatisch an die Angaben der Mapdatei angepasst werden.

Für die Verwendung der MapServer-Variablen müssen Sie Zeile 7 in der Datei *global.html* ändern. Verwenden Sie anstelle der Pixelwerte für Breite und Höhe die Variablen `[mapwidth]` bzw. `[mapheight]`. Die Zeile sollte nun so aussehen:

```
<input name="img" type="image" src="[img]" width="[mapwidth]" height="[mapheight]"
border=0 alt="Map Image" />
```

Auf die gleiche Weise passen Sie nun den mit dem Formular übergebenen Parameter `imgxy` an, damit die Karte automatisch anhand der Werte in der Mapdatei zentriert wird und Sie keine hart kodierten Pixelwerte mehr benutzen müssen:

```
<input type="hidden" name="imgxy" value="[center_x] [center_y]" />
```



Sollten Sie die Größe des Bilds doch einmal im HTML-Template fest angeben, müssen Sie auf jedem Fall darauf achten, dass Sie hier das gleiche Verhältnis verwenden wie auch in der Mapdatei. Ansonsten kann es passieren, dass die Darstellung der Karte in der Webseite verzerrt erscheint.

Erweitern des HTML-Clients um Werkzeuge zum Zoomen und Zentrieren

Bis jetzt verfügte unser Client lediglich um eine Navigationsmöglichkeit, nämlich das Hineinzoomen in die Karte über einen Klick. Ein Herauszoomen war nicht möglich (außer durch die Betätigung des Zurück-Buttons des Browsers). Auch konnte der Kartenausschnitt bisher nicht neu zentriert werden. Im folgenden Beispiel werden wir die Karte um drei Optionen erweitern, mit denen genau diese Funktionalitäten zur Verfügung gestellt werden: eine Option zum Zoomen in die Karte hinein, eine zum Herauszoomen und eine, um die Karte neu zu zentrieren.

Wenn Sie sich bereits mit HTML-Formularen auskennen, ist die nächste Übung ein Kinderspiel für Sie. Das Formular wird einfach um einige Kontrollelemente erweitert, die zusätzliche Variablen an das MapServer-Programm übergeben. Bisher war die Möglichkeit zum Zoomen fest eingestellt und konnte nicht durch den Benutzer geändert werden. Dieses Problem beheben wir nun, indem wir die Karte mit drei Radiobuttons und einem Textfeld, über das der Zoomfaktor bestimmt werden kann, versehen.



Radiobuttons unterscheiden sich von Checkboxes und normalen Buttons. Besitzen mehrere Radiobuttons den gleichen Namen, kann jeweils nur einer ausgewählt werden. Dies bietet sich an, wenn immer nur ein Werkzeug oder eine Option verwendet werden soll.

In Beispiel 11-7 sehen Sie das erweiterte HTML-Template *global.html*. Abbildung 11-10 zeigt das resultierende Ergebnis.

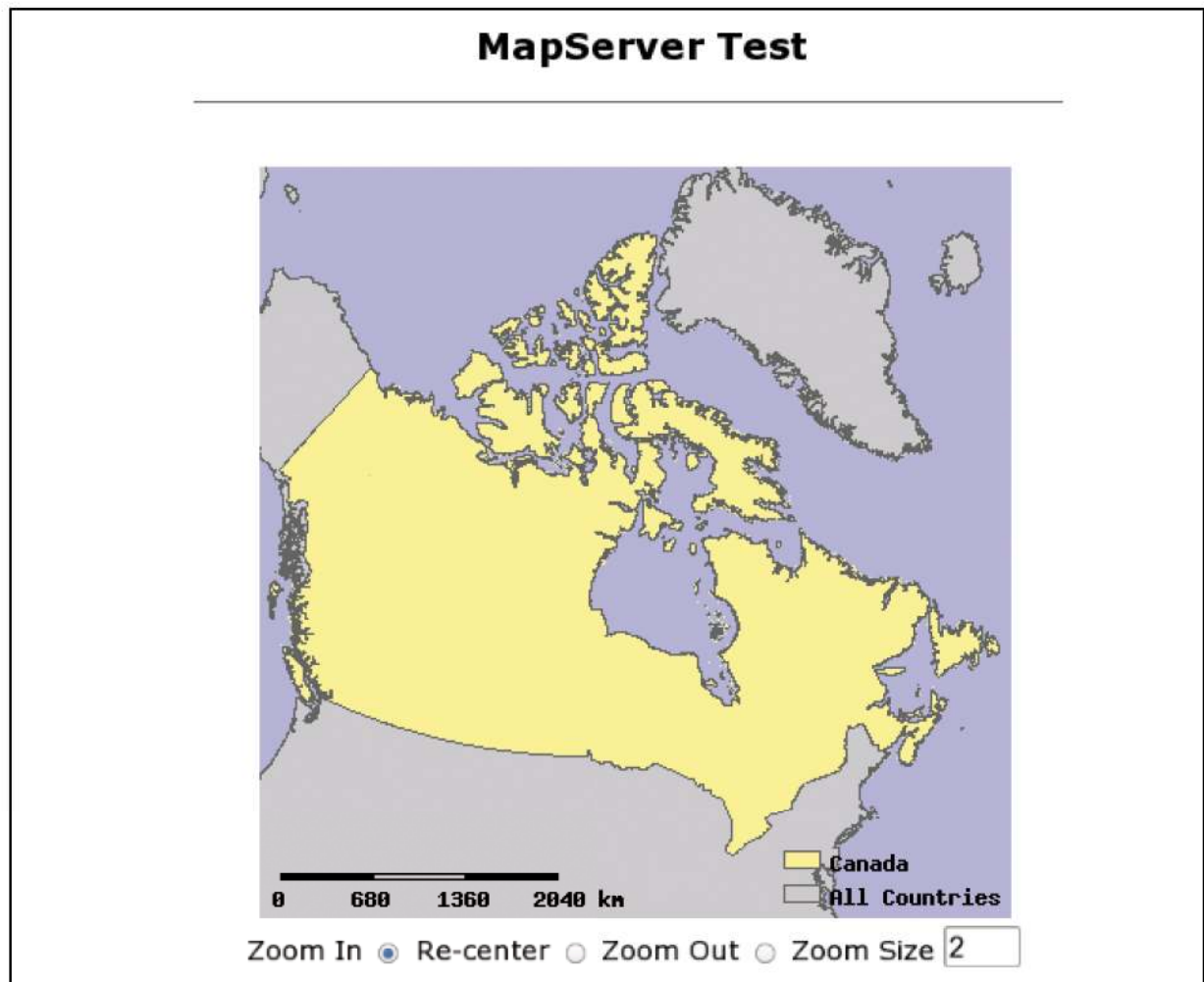


Abbildung 11-10: Die Webseite wurde um ein paar einfache Steuerungselemente erweitert

Beispiel 11-7: Erweiterung der Anwendung um Zoom- und Zentrierwerkzeuge

```
<html>
<head><title>MapServer Test</title></head><body>
<center><h2>MapServer Test</h2>
<hr>

<form method="GET" action="/cgi-bin/mapserv">
  <input name="img" type="image" src="[img]" width="[mapwidth]"
        height="[mapheight]" border=0 alt="Map Image" />
  <br />

  Zoom In
  <input type="radio" name="zoomdir" value=1 [zoomdir_1_check] />

  Re-center
  <input type="radio" name="zoomdir" value=0 [zoomdir_0_check] />
```

Beispiel 11-7: Erweiterung der Anwendung um Zoom- und Zentrierwerkzeuge (Fortsetzung)

Zoom Out

```
<input type="radio" name="zoomdir" value=-1 [zoomdir_1_check] />
```

Zoom Size

```
<input type="text" name="zoomsize" size=4 value=[zoomsize] />
```

```
<!-- Entfernen Sie die folgenden zwei Zeilen
```

```
<input type="hidden" name="zoomdir" value=1 [zoomdir_1_check] />
```

```
<input type="hidden" name="zoomsize" size=4 value=[zoomsize] />
```

```
-->
```

```
<input type="hidden" name="imgxy" value="[center_x] [center_y]" />
```

```
<input type="hidden" name="imgext" value="[mapext]" />
```

```
<input type="hidden" name="map" value="[map]" />
```

```
<input type="hidden" name="savequery" value="true" />
```

```
<input type="hidden" name="mapext" value="shapes" />
```

```
</form></center></body></html>
```

Als Erstes haben wir das Formular um die Radiobuttons erweitert. Um sie hervorzuheben, haben wir sie hier ein wenig eingerückt. Alle Radiobuttons sind sich ähnlich. Neben ihnen wird eine erläuternde Beschriftung angezeigt. Der Formularcode definiert, wofür die einzelnen Buttons zuständig sind.

```
<br />
```

Zoom In

Das `<br />`-Tag sorgt für einen Zeilenumbruch, sodass die Werkzeuge unterhalb der Karte dargestellt werden. Die Wörter Zoom In stehen in einer neuen Zeile und befinden sich links neben dem Button. Der Radiobutton selbst wird durch den folgenden Code definiert:

```
<input type="radio" name="zoomdir" value=1 [zoomdir_1_check] />
```

Das `input`-Tag ist Teil des CGI-Formulars und besitzt drei Attribute:

`type="radio"`

Hiermit wird definiert, welche Art Kontrollelement in das Formular integriert werden soll. Bisher wurde in den Beispielen nur `type="hidden"` verwendet, mit dem kein Kontrollelement angezeigt wird, sondern die angegebenen Werte quasi »hinter den Kulissen« übergeben werden. Der Wert `radio` legt fest, dass der Browser an dieser Stelle einen Radiobutton darstellen soll. Insgesamt enthält die Seite drei Radiobuttons, die alle zur gleichen Gruppe gehören, weil sie einen gemeinsamen Namen haben. Wie wir gleich sehen werden, kann durch die Vergabe eines gemeinsamen Namens immer nur ein Button ausgewählt werden.

`name="zoomdir"`

Mit diesem Attribut wird der Name der Variablen angegeben. Diese enthält den über `value` definierten Wert. Die Angaben für `name` (und `value`) werden an das MapServer-Programm übergeben und verwendet, um anhand deren die nächste Karte zu erstellen.

value=1

Dies ist der Wert, der für die einzelnen Buttons vergeben wird. Jeder Button besitzt einen anderen Wert (-1, 0, 1). Zusammen mit dem Wert von name wird diese Angabe zur Verarbeitung an MapServer übergeben. Lautet die Angabe beispielsweise zoomdir=1, wird in die Karte hineingezoomt. Bei einem Wert von -1 wird aus der Karte herausgezoomt. Ist der Wert 0, wird die Karte an der Stelle zentriert, auf die der Benutzer geklickt hat.

Der letzte Teil des input-Tags lautet: [zoomdir_1_check]. Hierbei handelt es sich wieder um einen Platzhalter (ähnlich [img]), der von MapServer beim Erzeugen der HTML-Seite aus dem Template durch einen entsprechenden Wert ersetzt wird. In unserem Beispiel verwendet MapServer diese Variable, um dem zuletzt gewählten Button den Wert checked hinzuzufügen. Wenn Sie den »Zoom In«-Button anklicken, um in die Karte hineinzuzoomen, »merkt« sich MapServer, welcher Button zuletzt ausgewählt war. In der neuen Darstellung ist dieser Button automatisch aktiv, weil er von MapServer als checked markiert wurde. Schauen Sie sich dazu den HTML-Code der Anwendung an:

```
<input type="radio" name="zoomdir" value=1 checked />
```

Wählen Sie den Button zum Zentrieren der Karte, erhält der zweite Button den Wert checked:

```
<input type="radio" name="zoomdir" value=0 checked />
```

Auf diese Weise kann der Browser die zuvor von Ihnen gewählten Optionen wieder vorgeben, wenn eine neue Darstellung angezeigt oder die Seite aktualisiert wird.

Außerdem erhält das Formular noch ein Textfeld, in das der Zoomfaktor eingegeben werden kann:

Zoom Size

```
<input type="text" name="zoomsize" size=4 value=[zoomsize] />
```

Mit dieser Einstellung wird MapServer mitgeteilt, mit welchem Faktor gezoomt werden soll, wenn der Benutzer auf die Karte klickt. Die Zoomgröße 2 verdoppelt beispielsweise den Maßstab. Die Objekte der Karte erscheinen doppelt so groß. Bisher wurde diese Anweisung mithilfe eines versteckten Formularfelds fest angegeben. Diese Zeilen können Sie nun aus dem HTML-Template löschen. Auch die Variable zoomdir war mit der Einstellung value=1 explizit festgelegt, sodass bei einem Klick in die Karte prinzipiell hineingezoomt wurde. Mit den neuen Steuerungselementen ist dies nicht mehr notwendig. Die folgenden beiden Zeilen können daher gelöscht werden

```
<input type="hidden" name="zoomdir" value=1 [zoomdir_1_check] />
<input type="hidden" name="zoomsize" size=4 value=[zoomsize] />
```

Es muss auch eine Änderung in der *index.html* vorgenommen werden. In den früheren Beispielen wurde standardmäßig in die Karte hineingezoomt. Auch hier wurde über ein verstecktes Formularfeld für die Variable zoomdir die Einstellung value=1 festgelegt. Damit bei einem Klick auf die anfängliche Karte nicht mehr automatisch gezoomt wird, ändern wir den Wert. Durch die Einstellung value=0 wird die Karte stattdessen neu zen-

triert. Das wird gegen Ende dieses Kapitels wichtiger, wenn wir einen Button erzeugen, über den die Karte neu gezeichnet wird. Wäre standardmäßig der Wert 1 ausgewählt, würde die Karte beim Neuzeichnen gleichzeitig herangezoomt. Hier sehen Sie die geänderte Zeile in der *index.html*:

```
<input type="hidden" name="zoomdir" value=0 />
```

Liste zur Auswahl der Ebenen einfügen

Die Möglichkeiten, innerhalb der Karte zu zoomen und die Karte neu zu zentrieren, sind für die meisten GIS-Anwendungen unabdingbar. Außerdem wird die Fähigkeit benötigt, bestimmte Ebenen ein- und ausblenden zu können. Wenn Sie diese Funktionalität einbauen, sollten Sie versuchen, bestimmte Fehler von vornherein auszuschließen. Würden Sie dem Benutzer nämlich gestatten, alle Ebenen auszublenden, wird möglicherweise nur noch eine leere Karte angezeigt. Das entspricht zwar genau dem, was der Benutzer ausgewählt hat, macht aber den Eindruck, als läge ein Fehler in der Anwendung vor. Diesen Fall könnten Sie abfangen

Um die Möglichkeit der Ebenenauswahl zu haben, müssen Sie erst eine zweite Ebene anlegen. Diese neue Ebene können Sie direkt in der Mapdatei definieren. Vielleicht wollen Sie ein paar Bilder oder eine weitere Shape-Datei integrieren. Um die Sache einfach zu halten, wird im folgenden Beispiel eine Ebene erstellt, die die Textbeschriftungen für die angezeigten Länder bereitstellt. Hierfür können Sie die gleichen Daten verwenden. Beispiel 11-8 zeigt die neu in *global.map* angelegte Ebene mit dem Namen *country_labels*. Durch die Angabe `TYPE ANNOTATION` wird eine Beschriftungsebene erzeugt.

Beispiel 11-8: Die Karte wird um eine Beschriftungsebene erweitert

```
LAYER
  NAME country_labels
  TYPE ANNOTATION
  STATUS DEFAULT
  DATA countries_simpl
  PROJECTION
    "proj=latlong"
    "ellps=WGS84"
  END
  LABELITEM 'NAME'
  CLASS
    LABEL
      COLOR 255 255 255
      OUTLINECOLOR 0 0 0
      POSITION CC
      MINFEATURESIZE 100
    END
  END
END
```

Sie könnten die neue Ebene auch in einer separaten Datei (z.B. *layer_country_labels.map*) definieren und über `INCLUDE` in die Mapdatei einbinden. Die Angabe würde dann wie folgt aussehen und steht vor dem `END`, das die Mapdatei abschließt:

```
INCLUDE "layer_country_annotation.map"
```

MapServer kann die Ebenen auf zwei Arten zeichnen. Die erste besteht darin festzulegen, dass die Ebene immer gezeichnet werden soll. Dies geschieht, indem Sie für die betreffende Ebene den `STATUS DEFAULT` festlegen. Die zweite Möglichkeit ist, die Ebene nur auf Anfrage darzustellen. Die Einstellung hierfür lautet `STATUS OFF`. Wurde eine Ebene in der Mapdatei als `OFF` definiert, heißt das nicht, dass die Ebene prinzipiell nicht angezeigt werden kann. Wird die Ebene über die Webanwendung angefordert, wird sie auch gezeichnet. Beachten Sie, dass `STATUS` immer `ON` sein muss, wenn Sie MapServer als WMS-Server einsetzen möchten. Aber mehr dazu in Kapitel 12.

Damit die Umringe der Länder immer gezeichnet werden, setzen wir den Parameter `STATUS` für die Ebene `countries` auf den Wert `DEFAULT`. Für `country_labels` definieren wir stattdessen `STATUS OFF`. Auf diese Weise kann der Benutzer die Beschriftungen nach Bedarf ein- und ausschalten.

Als Nächstes müssen Sie in der Datei *index.html* ein paar Änderungen vornehmen, um MapServer mitzuteilen, welche Ebenen beim Start der Anwendung angezeigt werden sollen. Im Unterschied zu den Einstellungen für das Zeichnen der Ebenen in der Mapdatei wird hier eine Liste der gewünschten Ebenen über *index.html* an MapServer übergeben. Wurden die gewünschten Ebenen in der Mapdatei definiert, wird MapServer angewiesen, sie darzustellen. Die Namen der Ebenen werden in der URL als CGI-Parameter (z.B. `&layers=countries`) übergeben und von MapServer ausgewertet, um die darzustellenden Ebenen zu ermitteln.

In Beispiel 11-2 sahen Sie den ursprünglichen Aufbau der *index.html*. Uns interessiert besonders die folgende Zeile:

```
<input type="hidden" name="layer" value="countries" />
```

Die Angabe dieser Zeile bei der Initialisierung weist MapServer an, beim Start die Ebene `countries` darzustellen. Da diese Ebene durch die Einstellungen in der Mapdatei standardmäßig angezeigt wird, ohne dass der Benutzer sie ausblenden könnte, ist diese Zeile eigentlich nicht mehr nötig. Stattdessen verwenden wir sie nun für unsere neue Ebene, `country_labels`:

```
<input type="hidden" name="layer" value="country_labels" >
```

Jetzt werden bei der ersten Anzeige der Karte beide Ebenen dargestellt. Sollen lediglich die Ebenen angezeigt werden, die in der Mapdatei den `STATUS DEFAULT` haben, brauchen Sie diese Zeile nur aus der *index.html* zu entfernen.

Der dritte Ort, an dem Ebenenoptionen definiert werden, ist das HTML-Template (*global.html*). Zusammen mit der Karte selbst zeigt MapServer alle von Ihnen definierten Steuerelemente an. Hierzu gehört auch eine Auswahlliste für die Ebenen. Über diese

bekommt der Benutzer die auswählbaren Ebenen tatsächlich angezeigt. Die Einstellungen in *global.map* und *index.html* bleiben dem Benutzer dagegen verborgen (wenn man einmal von der Anzeige in der URL absieht).

Im HTML-Template wurde bereits ein Formular definiert, das Sie nun um eine Auswahlliste erweitern können. In Beispiel 11-9 sehen Sie die ersten Zeilen des HTML-Templates mit der Definition des select-Tags, das die Auswahlliste erzeugt.

Beispiel 11-9: Beginn der Definition einer Auswahlliste zum Ein- und Ausschalten der Ebenen

```
...
<form method="GET" action="/cgi-bin/mapserv">
  <br /> <input name="img" type="image" src="[img]" width="[mapwidth]" height="[mapheight]"
        border=0 alt="Map Image" />
  <select multiple name="layer" size=3>
    <!--Hier steht die Liste der Ebenen, die ausgewählt werden können -->
  </select>
  <br />
  Zoom In <input type="radio" name="zoomdir" value=1 [zoomdir_1_check] />
...
```



Die Angabe `size=3` für die Auswahlliste legt fest, wie viele Zeilen der Liste angezeigt werden sollen. Wenn Ihre Karte nur eine geringe Zahl an Ebenen enthält, können Sie die Darstellung so anpassen, dass die Zeilen der Anzahl der Ebenen entspricht. Gibt es mehr Ebenen, als sich bequem anzeigen lassen, sind diese zwar weiterhin in der Liste enthalten, Sie müssen aber scrollen, um sie auszuwählen.

Bis jetzt hat die Liste noch keine konkreten Inhalte. Das ändert sich, sobald Sie einen Eintrag für die Ebene `country_labels` hinzufügen. Jede Ebene, die Sie über die Oberfläche steuern wollen, benötigt ein `option`-Tag:

```
<option value="country_labels" [country_labels_select]> Country Names</option>
```



Sie können in der Mapdatei auch Ebenen zu Gruppen zusammenfügen und als Gruppe steuern. Dazu müssen Sie im `LAYER`-Objekt neben dem Parameter `NAME` auch einen Wert für den Parameter `GROUP` angeben. Wenn Sie Gruppen nutzen, können Sie anstelle eines Ebenennamens auch den Namen einer Gruppe als Auswahlelement verwenden.

Die zweite Definition (`[country_labels_select]`) ist notwendig, damit die zuvor ausgewählte Ebene als selektiert angezeigt werden kann. Dies ist vergleichbar mit der Definition von `checked` für die Radiobuttons zum Zoomen bzw. neu Zentrieren. Diese Einstellung entspricht dem Wert des Attributs `value` (`country_labels`), dem die Zeichen `_select` nachgestellt wurden.



Tippfehler in diesen zwei Einstellungen können leicht für Verwirrung und Frustration sorgen. Sie sollten daher besonders genau darauf achten, dass der Text dem Ebenennamen in der Mapdatei entspricht.

Als Drittes geben wir den Text `Country Names` an. Im Gegensatz zu den anderen beiden Definitionen wird dieser nicht als Attribut des `option`-Tags definiert, sondern er steht zwischen dem öffnenden und dem schließenden Tag. Sie können hier einen Text einfügen, der als Beschreibung der Ebene in der Auswahlliste erscheinen soll. Sind nun alle Änderungen vollzogen, sieht die Webseite aus wie in Abbildung 11-11.

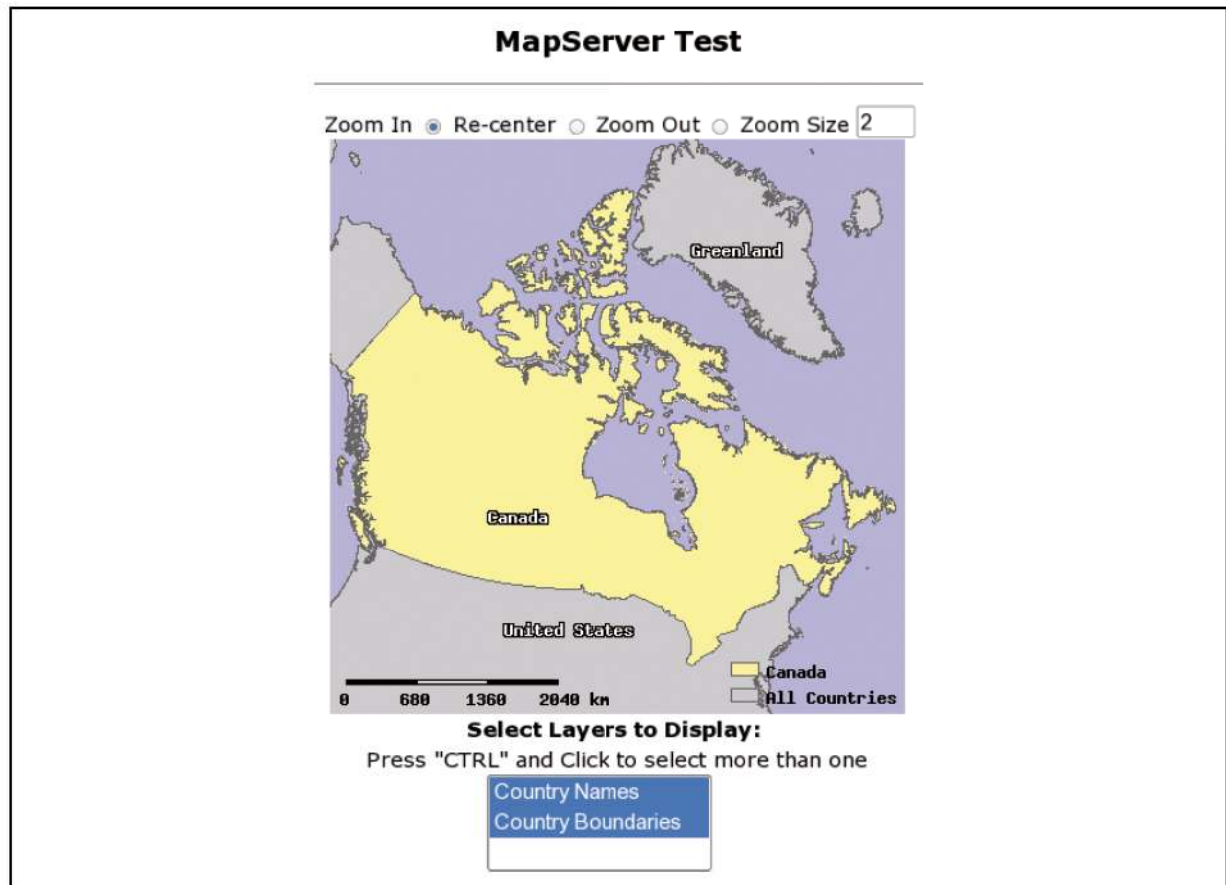


Abbildung 11-11: Die Seite wurde um eine Auswahlliste für die Ebenenauswahl erweitert

Den kompletten Quellcode sehen Sie in `global.html` in Beispiel 11-10. Es wurde außerdem ein kurzer Text für die Benutzung der Liste eingefügt. Einige Tags wurden verschoben, um der Seite eine bessere Struktur zu geben.

Beispiel 11-10: Neue Auswahlliste mit den Einträgen für die zwei Ebenen

```
<html>
<head><title>MapServer Test</title></head>
<body>
<center><h2>MapServer Test</h2>
<hr>
```

Beispiel 11-10: Neue Auswahlliste mit den Einträgen für die zwei Ebenen (Fortsetzung)

```
<form method="GET" action="/cgi-bin/mapserv">
  Zoom In <input type="radio" name="zoomdir" value=1 [zoomdir_1_check] />
  Re-center <input type="radio" name="zoomdir" value=0 [zoomdir_0_check] />
  Zoom Out <input type="radio" name="zoomdir" value=-1 [zoomdir_-1_check] />
  Zoom Size <input type="text" name="zoomsize" size=4 value=[zoomsize] /><br />
  <input name="img" type="image" src="[img]" width="[mapwidth]" height="[mapheight]"
  border=0 alt="Map Image" /><br />
  <b>Select Layers to Display:</b><br />
  Press "CTRL" and Click to select more than one<br />
  <select multiple name="layer" size=3>
    <option value="country_labels" [country_labels_select]> Country Names</option>
    <option value="countries" [countries_select]> Country Boundaries</option>
  </select>

  <input type="hidden" name="imgxy" value="[center_x] [center_y]" />
  <input type="hidden" name="imgext" value="[mapext]" />
  <input type="hidden" name="map" value="[map]" />
  <input type="hidden" name="savequery" value="true" />
  <input type="hidden" name="mapext" value="shapes" />

</form></center></body></html>
```

Button zum Neuzeichnen der Karte hinzufügen

Als Nächstes wollen wir einen Button zum Neuzeichnen der Karte einbauen. Hierfür können Sie beispielsweise einen Submit-Button für das Formular verwenden. Setzen Sie diesen Code direkt hinter das schließende select-Tag der Auswahlliste:

```
<br /><input type="submit" value="Redraw/Update" />
```

Dieses erzeugt sofort unterhalb der Liste einen Button, wie in Abbildung 11-12 zu sehen ist. Jetzt können Sie bestimmte Ebenen auswählen, den Button anklicken, und die Änderungen werden in der neu gezeichneten Karte berücksichtigt.

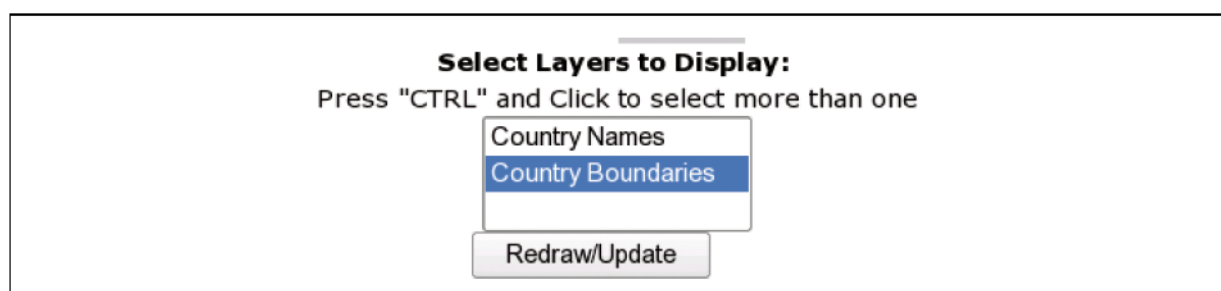


Abbildung 11-12: Die Webseite wird um einen Button für die Aktualisierung der Karte erweitert

Herzlichen Glückwunsch! Sie haben soeben Ihre erste interaktive Webanwendung fertiggestellt. In dem Beispiel wurden aber nur die Grundlagen vermittelt, darüber hinaus gibt es noch zahlreiche Möglichkeiten. Die nächsten Seiten dieses Kapitels werden ein paar weitere Anpassungen aufzeigen, mit denen Sie Ihre Anwendung ausbauen können.

Einbinden der Legende als separates Bild

In unseren bisherigen Beispielen wird eine Legende in die Karte eingebettet. Nun werden wir die Legende als eigenständige Grafik in die Webseite einbauen. Hierfür müssen zwei Anpassungen vorgenommen werden: Zum einen muss die Mapdatei *global.map* so angepasst werden, dass die Legende nicht mehr in die Karte eingebettet wird, zum anderen muss das Template *global.html* erweitert werden, damit die Legende als eigenständige Grafik angezeigt wird.

Die in der Karte dargestellte Legende entfernen

Damit die Legende nicht mehr in die Karte eingebettet wird, müssen Sie den Parameter STATUS im LEGEND-Objekt EMBED auf STATUS ON stellen. Die Parameter POSITION und TRANSPARENT haben in diesem Fall keine Relevanz mehr, da sie sich nur auf eingebettete Legenden beziehen. In Beispiel 11-11 sehen Sie zuerst die ursprüngliche Version des LEGEND-Objekts in *global.map*. Im Anschluss folgt die angepasste Version.

Beispiel 11-11: Auslagern der Legende aus den Karten durch Änderung der STATUS-Angabe

```
# ursprüngliche Legende (in Karte eingebettet)
LEGEND
  STATUS EMBED
  POSITION LR
  TRANSPARENT TRUE
END

# modifizierte Legende
LEGEND
  STATUS ON
END
```

Hinter den Kulissen sorgen diese Einstellungen dafür, dass MapServer die Legende nun als eigenständiges Bild erzeugt.

Einbau der separaten Legende in das HTML-Template

Mit wenigen zusätzlichen Zeilen Code im HTML-Template können Sie das von MapServer erzeugte Legendenbild in die Webseite integrieren:

Beispiel 11-12: Einbinden der Legende in das HTML-Template global.html

```
<br /><hr>
<b>Legend</b><br />
<hr>
```



Diese Webseite ist vielleicht nicht besonders schön, aber sie funktioniert. Selbstverständlich können Sie das Design der Seite komplett an Ihre Bedürfnisse anpassen.

Der oben stehende Code bindet das Bild der Legende in Ihre Webseite ein. Für ein besseres Verständnis können Sie sich die Webseite in der Quellcodeansicht (meistens zu finden unter *Anzeige* → *Seitenquelltext*) anzeigen lassen. Auf diese Weise können Sie den HTML-Code der Seite betrachten.



Über das Attribut `alt` wird ein Text angegeben, der erscheint, wenn das Bild nicht angezeigt werden kann.

In anderen Webseiten definiert das Attribut `src` die Adresse des Bilds, das angezeigt werden soll. Über `src="[legend]"` wird auf die Legendengrafik als Bildquelle verwiesen. MapServer fügt den Wert automatisch ein. Bei der Interpretation des HTML-Templates ersetzt MapServer `[legend]` durch den Pfad zur Legendengrafik. Die fertige Datei wird schließlich an Ihren Browser geschickt, der nun, wie in Abbildung 11-13 gezeigt, die Legende als eigene Grafik dargestellt.

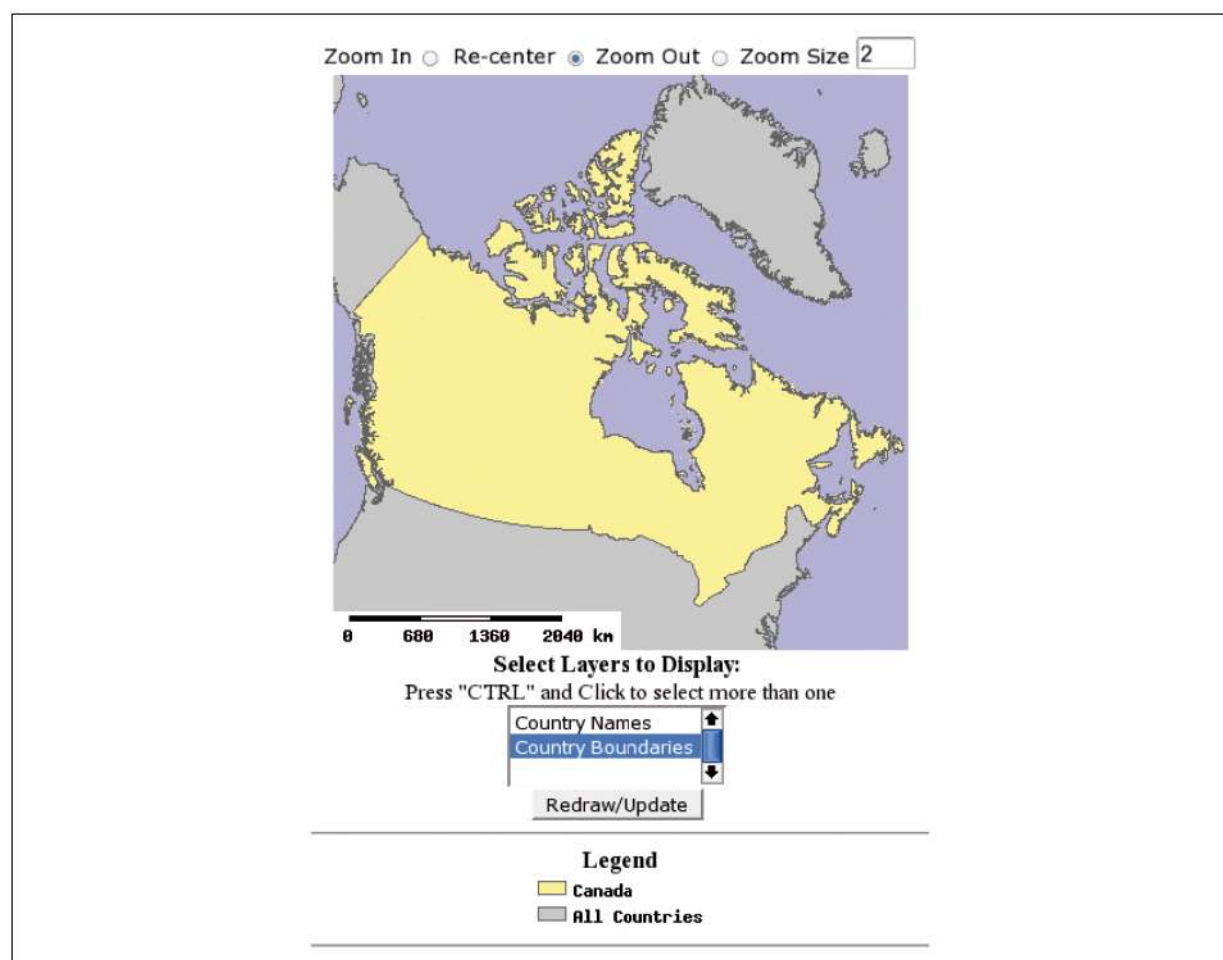


Abbildung 11-13: Die Legende wurde als separate Grafik in die Webseite integriert

Einbinden der Maßstabsleiste als separate Datei

Die Grafik für die Maßstabsleiste wird auf die gleiche Weise erzeugt wie die Grafik der Legende. Wenn der Maßstab nicht in die Karte eingebettet wird, wird ebenfalls eine eigene Datei erzeugt. Um die Maßstabsleiste in das Template einzubinden, verwenden Sie die Template-Variable [scalebar].

Das Template *global.html* muss – wie bei der Legende – nur um eine Zeile erweitert werden. Den notwendigen HTML-Code sehen Sie in Beispiel 11-13.

Beispiel 11-13: HTML-Code zur Anzeige der Maßstabsleiste

```
<input name="img" type="image" src="[img]" width="[mapwidth]"
      height="[mapheight]" border=0 alt="Map Image" /><br />
<br />
<b>Select Layers to Display: </b><br />
```

In der Mapdatei müssen Sie für das SCALEBAR-Objekt die Angabe STATUS EMBED in STATUS ON umwandeln. Die fertige HTML-Seite sehen Sie in Abbildung 11-14.

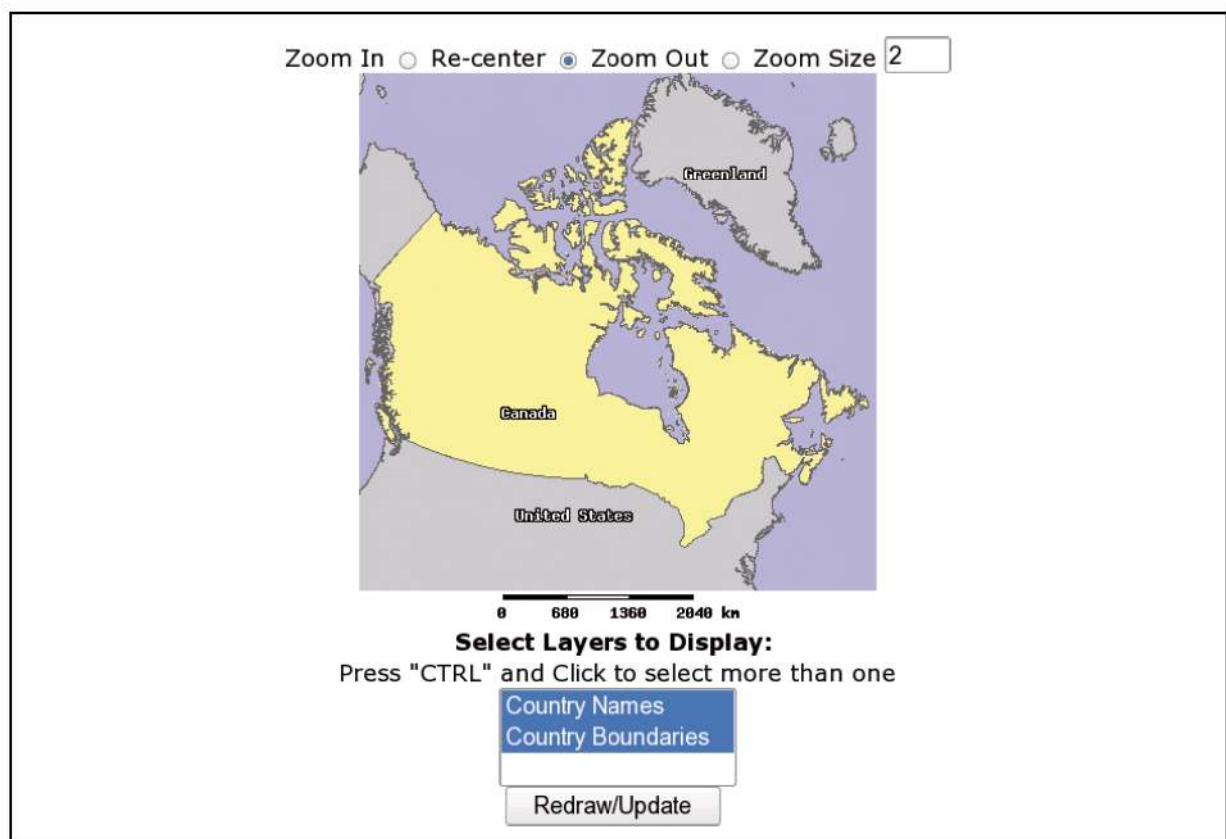


Abbildung 11-14: Die Webseite wurde um eine separate Maßstabsangabe erweitert

Hinzufügen einer Übersichtskarte

Sie können auch eine Übersichtskarte in Ihre Anwendung integrieren. Hierfür sind allerdings ein paar Überlegungen und Vorbereitungen notwendig. Übersichtskarten werden aus zwei Gründen benötigt. Sie zeigen an, welcher Bereich in der Hauptkarte angezeigt wird, und erleichtern darüber hinaus die Orientierung in der Karte. In Abbildung 11-15 sehen Sie beispielsweise eine MapServer-basierte Website, die eine Referenzkarte verwendet, um den Benutzern die Orientierung auf der aktuellen Karte zu erleichtern.

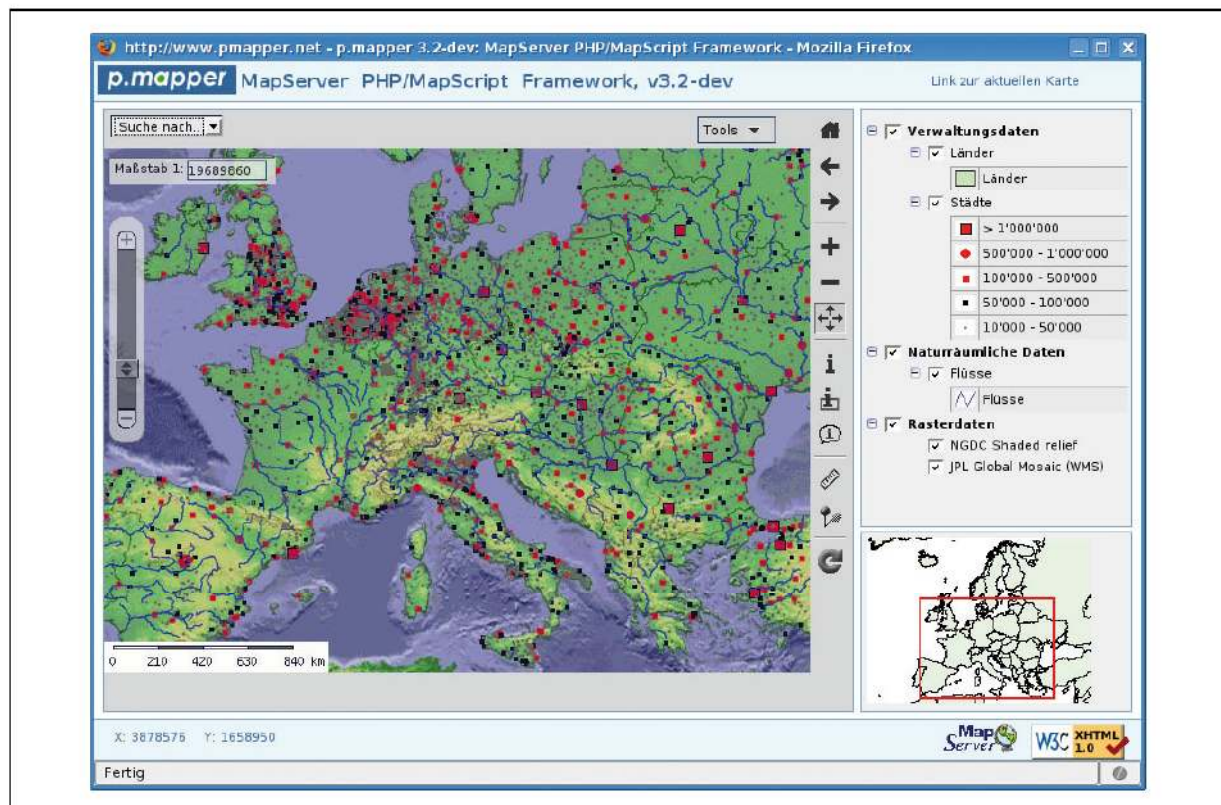


Abbildung 11-15: p.mapper-Anwendung mit integrierter Referenzkarte

Übersichtskarten können auch dazu verwendet werden, die Hauptkarte neu zu zentrieren. Dies erfolgt über einen Klick in die Übersichtskarte. Diese Möglichkeit ist besonders hilfreich, wenn Sie bei unverändertem Maßstab einen anderen Bereich der Karte betrachten wollen.

Wenn Sie eine Übersichtskarte einbauen wollen, muss in der Mapdatei ein REFERENCE-Objekt angegeben werden. Außerdem muss das HTML-Template (z.B. *global.html*) um ein weiteres `img`-Tag für die Übersichtskarte erweitert werden. Dazu benötigen Sie ein Bild, das in der Übersicht angezeigt werden soll. Dieses Bild wird nicht zusammen mit der eigentlichen Karte und ihren Ebenen erzeugt, sondern ist ein statisches Hintergrundbild.

Eine Weltansicht können Sie als PNG oder GIF von der Website zu diesem Buch herunterladen. Die Adresse lautet:

<http://www.webmappingillustrated.com/beispiele>

Das Bild sehen Sie in Abbildung 11-16.

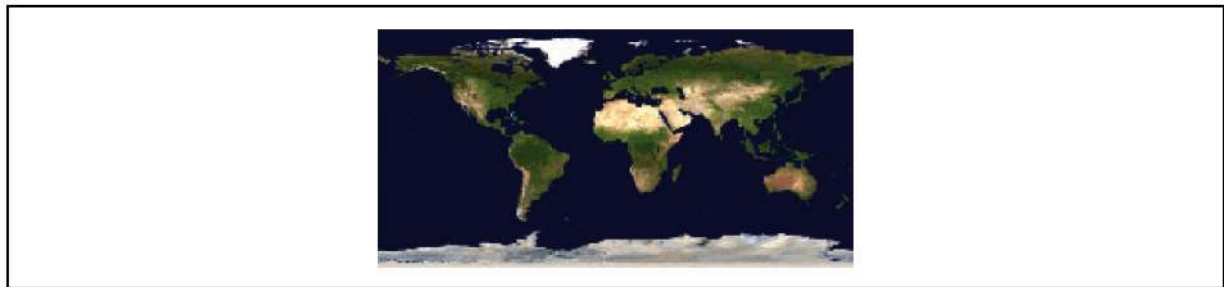


Abbildung 11-16: Das Bild für die Übersichtskarte

Speichern Sie dieses Bild nun im gleichen Verzeichnis wie die Datei *global.map* und erweitern Sie die Mapdatei um die in Beispiel 11-14 hervorgehobenen Zeilen für das REFERENCE-Objekt. Fügen Sie die Angabe, die für die gesamte Mapdatei gültig ist, möglichst am Anfang der Mapdatei ein.

Beispiel 11-14: Die Mapdatei wird um ein REFERENCE-Objekt ergänzt

```
...  
LEGEND  
  STATUS EMBED  
  POSITION LR  
  TRANSPARENT TRUE  
END  
  
REFERENCE  
  STATUS ON  
  IMAGE keymap.png          # Bild für die Übersichtskarte  
  EXTENT -180 -90 180 90    # Darstellungsbereich der Übersichtskarte  
  SIZE 241 121              # Bildgröße in Pixeln  
  COLOR -1 -1 -1  
  OUTLINECOLOR 255 0 0  
END  
....
```

Anhand des EXTENT-Parameters innerhalb des REFERENCE-Objekts geben Sie die Koordinaten für den Darstellungsbereich der Übersichtskarte an. Die Definition in unserem Beispiel teilt MapServer mit, dass die Referenzkarte *keymap.png* den gesamten Globus abdeckt. Die Angaben für SIZE übermitteln dem Programm die Ausmaße der Datei (in Pixeln). Mit diesen beiden Einstellungen ist MapServer in der Lage, bestimmte Koordinaten auf der Referenzkarte zu finden. Wenn die Referenzkarte einmal nicht korrekt reagiert, sollten Sie zuerst die oben genannten Einstellungen überprüfen.

Wird in der Applikation die Referenzkarte verwendet, überlagert MapServer die statische Datei (in unserem Beispiel *keymap.png*) mit den Umrissen des in der Hauptkarte dargestellten Bereichs. Anhand der Einstellungen für den Parameter COLOR können Sie festlegen, in welcher Farbe das Rechteck gezeichnet werden soll. Standardmäßig wird als Farbe Rot verwendet. Die Definition COLOR -1 -1 -1 teilt MapServer mit, dass das

Rechteck transparent dargestellt werden soll, sodass nur die Umrisse zu sehen sind. Mit den Angaben zu `OUTLINECOLOR` kann die Farbe des Umrings festgelegt werden (in unserem Beispiel Rot).

Für das `REFERENCE`-Objekt gibt es keine eigenen Projektionseinstellungen. Das bedeutet, dass für die `EXTENT`-Angabe die gleichen Einheiten verwendet werden müssen wie für die Projektion der Hauptkarte. Dieses Beispiel kann nur mit einer Karte eingesetzt werden, die die Längen- und Breitenangaben vom Anfang dieses Kapitels verwendet. Als Einheit muss `UNITS DD` angegeben werden, damit das Beispiel funktioniert. Die Angaben für `EXTENT` müssen also in Dezimalgrad vorgenommen werden.

Das HTML-Template *global.html* muss ebenfalls angepasst werden. Hierfür gibt es zwei Möglichkeiten. Entweder Sie verwenden ein weiteres `img`-Tag, z.B. ``. Oder aber Sie sorgen dafür, dass die Hauptkarte bei einem Klick auf die Referenzkarte neu zentriert werden kann. Hierfür verwenden Sie anstelle des obigen den folgenden Code:

```
<input name="ref" type="image" src="[ref]" width="241" height="121" alt="Reference map">
```

Bei dieser Herangehensweise wird das Bild als Button verwendet. Klicken Sie das Bild an, wird die Hauptkarte geändert. Wenn Sie die Referenzkarte direkt nach der Hauptkarte in die Seite einbauen, sollte Ihre Darstellung in etwa so aussehen, wie in Abbildung 11-17 gezeigt. Hier wurde die Ebenenauswahl deaktiviert, während Legende und Maßstab wieder als Teil der Hauptkarte dargestellt werden.

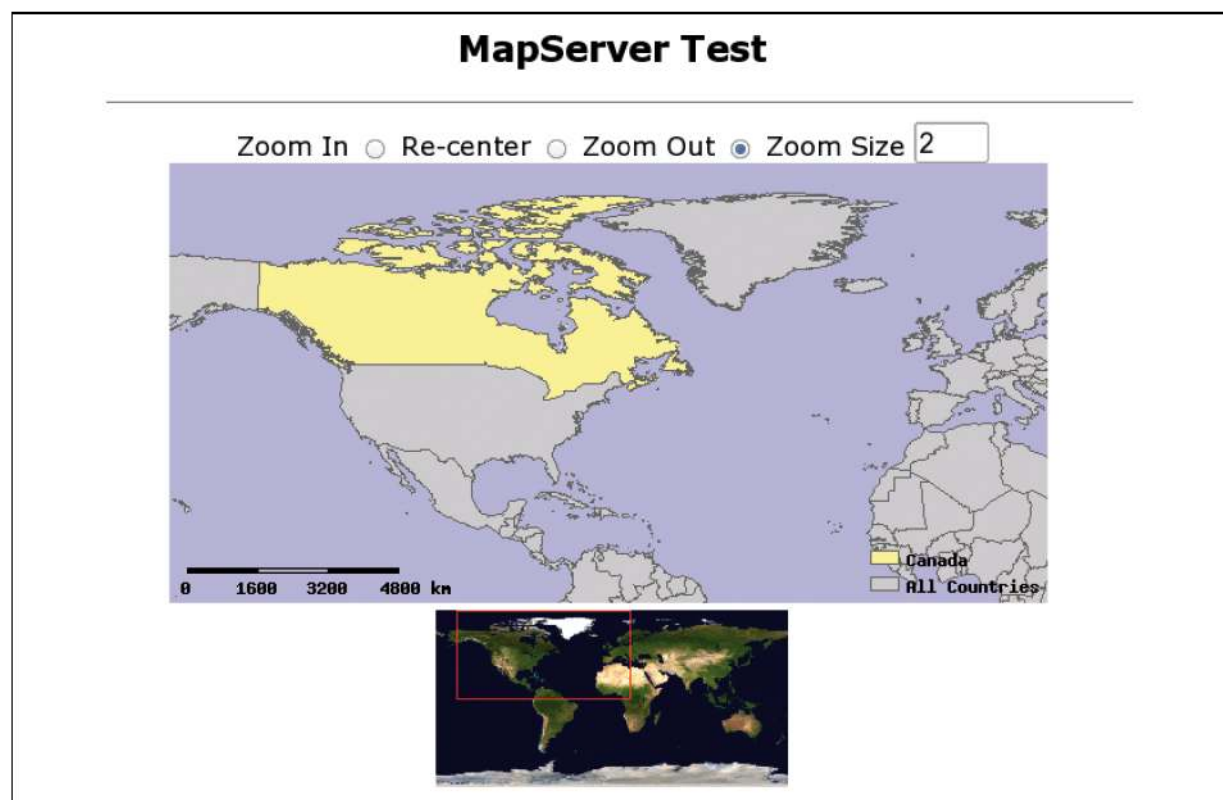


Abbildung 11-17: Die fertige Webseite inklusive der interaktiven Referenzkarte

Weiterführende Informationen

Die Beispiele in diesem Kapitel behandeln gerade einmal die elementarsten Möglichkeiten von MapServer. Um eine Idee davon zu bekommen, was an Möglichkeiten zur Verfügung stehen, werfen Sie einfach einen Blick in die MapServer-Galerie. Hier finden Sie eine Vielzahl von verschiedenen MapServer-basierten Anwendungen (siehe <http://mapserver.gis.umn.edu/gallery>).

Außerdem gibt es eine Reihe von Initiativen, die die Entwicklung von MapServer-Anwendungen erleichtern sollen. Das Stichwort lautet *application development frameworks* (Frameworks zur Applikationsentwicklung). Aufgrund ihrer Abhängigkeit von Web-Programmiersprachen (z.B. PHP) müssen diese Frameworks erst konfiguriert werden. In der folgenden Liste finden Sie ein paar dieser Werkzeuge:

- Chameleon: <http://www.maptools.org>
- CartoWeb: <http://www.cartoweb.org/>
- ka-map: <http://ka-map.maptools.org/>
- p.mapper: <http://www.pmapper.net/>
- Flexible Internet Spatial Template (FIST): <http://datashare.gis.unbc.ca>

Vermehrt wird MapServer-Funktionalität auch in Content-Management-Systeme (CMS), Groupware-Anwendungen oder Wikis integriert. Ein Beispiel hierfür ist Tiki-wiki (<http://tikiwiki.org/tiki-index.php?page=CommunityWorldMap>).

Kartenzugriff über standardisierte Webdienste

Webdienste (Webservices) werden dann interessant, wenn Geodaten anderen Anwendungen zur Verfügung gestellt werden sollen, auf die nur über ein Netzwerk zugegriffen werden kann. Dabei können Software und Daten unter Umständen Tausende von Kilometern entfernt sein oder auch nur zwischen dem ersten Stock und dem Keller eines Rechenzentrums liegen. Auf jeden Fall besteht kein direkter Zugriff des Anwenders auf die Datenräger. Für die Bereitstellung von Daten über ein Netzwerk sollten Sie Webdienste verwenden, am besten solche, die Standards nutzen, die von einem internationalen, unabhängigen Gremium verbindlich definiert sind.

Ein solcher Standard ist das Internetprotokoll, das Sie täglich nutzen, wenn Sie in einen Webbrowser *http://* oder *https://* eingeben. Das gleiche Protokoll kann auch für Geodaten genutzt werden, allerdings müssen diese aufgrund ihrer Komplexität mit etwas mehr Informationen verpackt werden, als es für einen normalen Text oder ein Bild erforderlich ist. Diese Informationen stecken in modernen Netzwerken meist in XML-Dateien. In der Einleitung zu seinem Buch *Web Services Essentials* (O'Reilly) gibt Ethan Cerami eine gute Definition von Webdiensten:

Als Webservice wird ein Dienst bezeichnet, der über das Internet verfügbar ist, der ein standardisiertes XML-Kommunikationssystem verwendet und unabhängig von Betriebssystemen oder Programmiersprachen ist.

Auf ähnliche Weise definiert Venu Vasudevan in seinem Artikel *A Web Services Primer* einen Webdienst einfach als »XML plus HTTP«. Den kompletten Artikel finden Sie unter <http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html>.

Der Begriff *Webservice* (*Webdienst*) steht normalerweise für ein Internetkommunikationsprotokoll, mit dem Anwendungen Informationen austauschen können. Webdienste bauen dabei auf das HTTP-Protokoll auf, das für die Kommunikation im Netz verwendet wird. HTTP ist das Transferprotokoll, während XML als Sprache verwendet wird. Mithilfe dieser Standards können Anwendungen das Internet als Kommunikationsnetzwerk nutzen.

Standardisierte Webdienste ermöglichen die Interoperabilität von Anwendungen, die unabhängig voneinander entwickelt wurden. Programme können nur auf Webdienste zugreifen, wenn diese den gleichen, klar definierten Standard verwenden. Deshalb dürfen diese Standards auch nicht von einer bestimmten Programmiersprache oder einem Betriebssystem abhängig sein. Die standardisierte, plattformunabhängige Kommunikation ermöglicht es Web-Mapping-Anwendungen, Informationen und Daten über das Web auszutauschen.

Wenn Sie Webdienste nutzen, müssen Sie nicht alle Einzelheiten und Hintergründe kennen. Um zum Beispiel Karten von einem OGC WMS (Web Map Service) nutzen zu können, reicht es zu wissen, unter welcher Adresse der Dienst im Internet zu finden ist – und die von Ihnen verwendete Clientsoftware muss natürlich ebenfalls den OGC WMS-Standard unterstützen. Wenn Sie hingegen eigene Geodaten bereitstellen wollen, müssen Sie wissen, wie Sie Ihre Anwendung konfigurieren müssen, damit andere von außen auf Ihren Dienst zugreifen können.

Wollen Sie Kartendienste lediglich in Ihre eigene Webseite einbinden oder eigene Anwendungen mit bestehenden Diensten erstellen, finden Sie in Kapitel 13 alle erforderlichen Informationen. Dieses Kapitel geht mehr ins Detail und ermöglicht es Ihnen, selbst Kartendienste aufzusetzen.

Kartendienste

Die internen Details von Webdiensten, wie XML-RPC, UDDI, WSDL oder SOAP, die in dem Buch *Web Services Essentials* (O'Reilly) beschrieben werden, sind für Programmierer interessant, sollen aber an dieser Stelle nicht weiter behandelt werden. Die zugrunde liegende Architektur kann RESTful¹ sein oder dem herkömmlichen RPC-Style folgen. Doch diese Details sind ebenfalls nur für Programmierer interessant und daher an dieser Stelle nicht relevant. Für den Umgang mit räumlichen Daten sind spezielle, standardisierte Schnittstellen entwickelt worden, die sich inzwischen weit verbreitet haben. Obwohl sie über das gleiche Netz kommunizieren, weisen sie Merkmale auf, die sich deutlich von allgemeinen Webdiensten, mit denen ein Webentwickler es sonst zu tun hat, unterscheiden.

Dieses Kapitel soll einen Einblick in die vom OGC (Open Geospatial Consortium) entwickelten Webdienste geben. Dabei wird der Schwerpunkt auf der Vorstellung des Web Map Service (WMS) und des Web Feature Service (WFS) liegen. Sie werden lernen, welche Webdienste von MapServer unterstützt werden und wie Sie diese einsetzen können.

¹ *Web Services mit REST* von Leonard Richardson & Sam Ruby, O'Reilly Verlag 2007.

Wie werden OGC-Webdienste in der digitalen Kartografie eingesetzt?

Die OGC-Webdienste wurden zu einer Zeit entwickelt, als das Internet noch aus einer Vielzahl unterschiedlicher Protokolle bestand (Gopher, Usenet, FTP usw.). Glücklicherweise wurden bereits ganz zu Anfang darauf geachtet, die Schnittstellen unabhängig von Protokollen zu entwickeln, sodass sie von Softwarepaketen ganz unterschiedlich implementiert werden konnten. Das Internet hat sich inzwischen als kleinster gemeinsamer Nenner durchgesetzt, und die damit verbundenen Protokolle und die Anwendungen, die diese Protokolle unterstützen, sind die mit großem Abstand erfolgreichsten.

Diese speziellen Webdienste erhalten eine immer größere Bedeutung in der digitalen Kartografie und sind die Zukunft des Web-Mappings. Die Entwicklung geht immer mehr dahin, dass die Daten auf einem Server bereitstehen und von den Nutzern über OGC-Webdienste angesprochen werden können. Die Daten liegen also nicht mehr bei dem Nutzer auf dem Rechner, sondern werden mit einem Dienst angefordert. Dies ist nur möglich, weil Computer- und Netzwerkressourcen mittlerweile eine Leistungsfähigkeit erreicht haben, die vor wenigen Jahren noch nicht zur Verfügung stand. Je mehr Daten und Datenquellen es gibt, desto größer wird das Bedürfnis, diese auch zu nutzen. Dies ist nur mithilfe von OGC-Webdiensten möglich.

Hierbei erfüllen OGC-Webdienste hauptsächlich zwei Aufgaben: den Zugriff auf bereitgestellte Informationen als Nutzer oder die Bereitstellung von Daten als Anbieter für die Nutzung durch andere. Es geht dabei also immer um den Austausch von Informationen.

Um auf externe Datenquellen zugreifen zu können, werden verschiedene Zugriffsinformationen über die Daten benötigt. Hierzu gehört beispielsweise die Adresse des Servers, zu dem eine Verbindung aufgebaut werden soll, sowie die Information darüber, welche Daten zur Verfügung stehen oder in welchen Formaten Daten angefordert werden können. Sind Ihnen diese und weitere Informationen über den Dienst bekannt, können Sie Ihre Anwendung so einrichten, dass genau die gewünschten Daten abgefragt werden.

Die Veröffentlichung eigener Daten über den UMN MapServer als Karten- oder Datendienst ist recht einfach. Ein paar kleine Änderungen und Erweiterungen in der Mapdatei reichen aus, und schon können andere Anwendungen auf Ihre Daten zugreifen.

Um selbst OGC-Webdienste ansprechen zu können, brauchen Sie kein großes Wissen darüber, wie die Anfragen aufgebaut sind oder verschickt werden. Diese Aufgabe übernimmt Ihr Web-Mapping-Client für Sie: Er erstellt die HTTP-Requests oder XML-basierten Requests und nimmt die Antworten und Daten entgegen. Normalerweise müssen Sie, wenn Sie OGC-Webdienste verwenden, selbst keinen XML-Code lesen oder Requests formulieren. Es ist aber sicherlich hilfreich, die Kommunikation zwischen Client und Server zu verstehen. Gelegentlich sollten Sie sich beispielsweise das XML-basierte Capabilities-Dokument anschauen, um mehr über OGC-Webdienste zu erfahren.



Dieses Kapitel soll Ihnen den Einsatz von OGC-Webdiensten Schritt für Schritt näherbringen. Dabei werden Sie die Requests, die für den Dienst definiert sind, kennenlernen und manuell absetzen, um das Konzept, das hinter den Webdiensten steht, zu verstehen. Dieses Wissen können Sie anschließend verwenden, um eigene OGC-Webdienste aufzusetzen und um bestehende Dienste anzusprechen.

Das Open Geospatial Consortium (OGC)

Das Open Geospatial Consortium (früher bekannt als Open GIS Consortium) hat eine Reihe von Standards für den Austausch räumlicher Daten entwickelt. Das Ziel des OGC ist es, die Interoperabilität zwischen verschiedenen Anwendungen mithilfe einer gemeinsamen Sprache zum Datenaustausch über die entsprechenden Standards herzustellen. Die Liste der Mitglieder ist genauso beeindruckend wie die formulierten Ziele. Hier eine kurze Beschreibung von <http://opengeospatial.org/about/> (Stand 11/2007):

Das Open Geospatial Consortium, Inc. (OGC) ist ein internationales Konsortium aus 344 Firmen, Regierungsinstitutionen und Universitäten, die in einem Konsensverfahren öffentlich verfügbare Schnittstellenspezifikationen entwickeln. OpenGIS®-Spezifikationen ermöglichen interoperable Lösungen, die das Internet geodatentauglich machen. Umgekehrt formuliert, werden durch OGC-Standards Geodaten internetfähig. Der Begriff Internet bezieht sich hierbei auf die Webtechnologie an sich, also nicht nur auf das öffentliche Netz, sondern auch auf firmeninterne Netze oder die der öffentlichen Verwaltung, auf drahtlose (location-based Services) wie ortsgebundene Dienste. Nicht zuletzt profitiert davon auch die traditionelle Informationstechnologie. OGC-Standards werden inzwischen in allen möglichen Bereichen genutzt, bis hinunter zur technischen Ebene von Netzwerkprotokollen, die GML als Format nutzen, um Serverstandorte bei Übertragungen zu übermitteln.

Das Hauptaugenmerk liegt hierbei nicht auf der Entwicklung eigener Applikationen, sondern auf der Entwicklung von Spezifikationen für den Datenaustausch. Deshalb finden sich in diesem Konsortium alle wichtigen proprietären GIS-Anbieter wie z.B. Autodesk, ESRI, MapInfo, SmallWorld und Intergraph wieder. Da Geodaten und Karten zunehmend in allen Bereichen der IT an Bedeutung gewinnen, finden sich inzwischen aber auch viele andere Firmen im OGC wieder, z.B. der Datenbankhersteller Oracle oder Microsoft, aber auch die Suchmaschinenbetreiber Google und Yahoo!.

In diesem Zusammenhang ist es besonders interessant, dass viele Open Source-Projekte wie der UMN MapServer, Mapbender und OpenLayers die Standards des OGC unterstützen oder sogar der GeoServer, deegree oder GeoNetwork Open Source-Referenzimplementierungen sind. Hierdurch wird die gemeinsame Nutzung von professioneller Open Source-Software mit proprietären Formaten möglich. Anbieter und Anwender sind nicht mehr auf eine bestimmte Plattform festgelegt und können wesentlich freier die Software auswählen, die ihren Zwecken am besten genügt.

Was spricht für den Einsatz standardisierter Dienste?

Es gibt viele Gründe für den Einsatz von standardisierten Diensten. Beispielsweise werden in Deutschland von den Vermessungsämtern mittlerweile viele räumliche Informationen öffentlich über Web Map Services (WMS), die dynamische Karten und Objektinformationen liefern, bereitgestellt. Weitere Informationen hierzu finden Sie in Kapitel 5. Diese Informationen können über standardisierte Aufrufe abgefragt werden. Der Benutzer muss die Daten nicht selbst lokal auf dem Rechner vorliegen haben, sondern fordert die Informationen über Anfragen an den Server an. So müssen beispielsweise Gemeinden nicht alle Daten selbst vorhalten und pflegen, sondern greifen direkt auf Dienste zu, die als WMS zur Verfügung stehen. Durch die zentrale Datenbereitstellung kann es zu einer erheblichen Kostenersparnis kommen, Arbeitsabläufe können vereinfacht und die Aktualität und Qualität der Daten kann verbessert werden.

Es kann sein, dass der Anbieter der Dienste direkt einen Client zur Visualisierung der Informationen bereitstellt. Dieser kann dann über den Browser direkt genutzt werden. Es ist also keine Installation auf dem Rechner erforderlich. Sie können einen OGC-Webdienst aber auch in Ihren eigenen Client einbinden, sofern Ihr Client den Standard unterstützt.

Vielleicht möchte jemand seine Daten mit Ihnen austauschen, kann oder will Ihnen aber keine vollständige Kopie der Daten geben. Ein OGC-Webdienst kann so eingerichtet werden, dass nur bestimmte Informationen zugänglich gemacht werden.

Einige Daten ändern sich ständig, beispielsweise die Positionen von Wirbelstürmen. Das Herunterladen dieser Informationen ist vielleicht nicht möglich oder müsste in kurzen Abständen erfolgen, um auf dem aktuellen Stand zu sein. Die Lösung dieses Problems ist die Integration eines WMS-Layers in Ihrer Anwendung. Über diesen Layer können in Echtzeit Bilder und Informationen von dem Dienst angefordert werden. Andere Datensammlungen sind zu groß, sodass es unrealistisch und unsinnig wäre, sie komplett lokal zu speichern. OGC-Webdienste sind in der Lage, Ihnen zur gewünschten Zeit nur die gerade benötigten Daten zur Verfügung zu stellen. Auf diese Weise sinkt der Bedarf nach ständig wachsender Speicherkapazität. Abbildung 12-1 zeigt die Funktionsweise einer UMN MapServer-Anwendung, die über eine standardisierte Anfrage an einen entfernten OGC WMS auf Daten zugreift.

Sowohl bei der Weitergabe wie auch beim Zugriff auf Daten geht es hauptsächlich darum, Zeit und Ressourcen möglichst sinnvoll zu nutzen. Wenn Sie eine Anwendung ständig für den Zugriff auf neue Datenquellen einrichten müssen, kann dies schnell sehr aufwendig werden. Auch die lokale Speicherung großer Datenmengen kommt nicht immer als Option in Frage und ist auch nicht unbedingt sinnvoll.

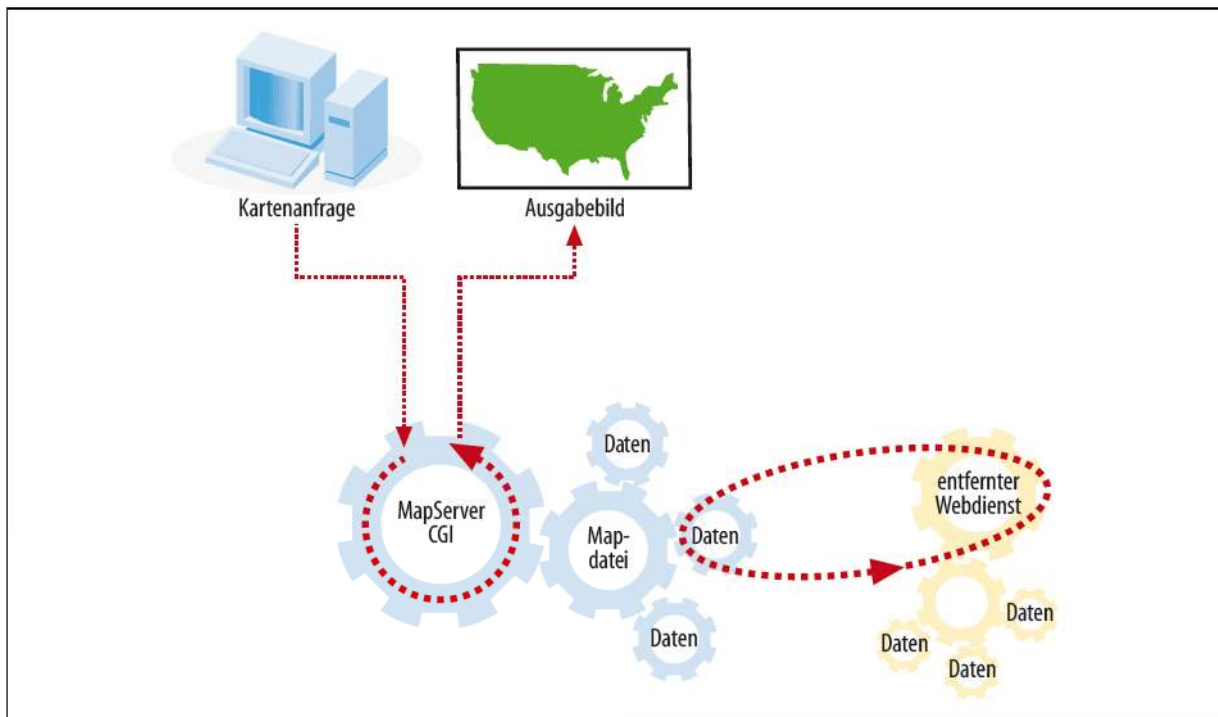


Abbildung 12-1: Anfordern von Daten von einem entfernten WMS durch den UMN MapServer

Verschiedene Anforderungen an Kartendienste

Einige Anbieter, die Daten bereitstellen, nutzen diese Informationen intern möglicherweise gar nicht. Nehmen Sie beispielsweise öffentliche Einrichtungen, die räumliche Informationen bereitstellen. Hier profitiert vor allem die Öffentlichkeit von den Diensten, weil sie Zugriff auf die Informationen erhält. Die öffentliche Einrichtung profitiert ebenfalls, weil sie ihrem Auftrag erfolgreich nachgeht.

Auf der anderen Seite kann es sein, dass eine Firma die Daten nur intern nutzen möchte, ohne dass die Öffentlichkeit Zugriff darauf erhält. OGC-Webdienste müssen nicht notwendigerweise öffentlich zugänglich sein. Die Dienste können genau wie jede andere interne Webanwendung abgesichert werden. Selbst große Unternehmen können auf diese Weise räumliche Informationen innerhalb der Firma zugänglich machen, ohne in den einzelnen Abteilungen redundante Hardwareinfrastrukturen unterhalten zu müssen. Neben der möglichen Kostenersparnis kann dieses Vorgehen auch verstärkten Informationsfluss ermöglichen. Die Benutzer benötigen vor allem zwei Dinge: Zugang zu Informationen und Zugang zu Informationen in einem geeigneten Format. Die Vorteile von OGC-Webdiensten bestehen darin, dass die Daten leicht zugänglich gemacht werden können und dass die Art der Bereitstellung ebenfalls gut steuerbar ist.

Mit UMN MapServer lassen sich interne wie externe Lösungen umsetzen. Dabei kann UMN MapServer sowohl als Server Daten bereitstellen als auch als Client Daten von entfernten Diensten anfordern.

Nutzung von OGC-Webdiensten mit UMN MapServer

Die Verwendung von OGC-Webdiensten bietet viele Vorteile und setzt sich immer mehr durch. Für den Einsatz von Webdiensten ist es nicht notwendig, eigene Anwendungen zu programmieren. Es stehen viele Clientanwendungen zur Verfügung, die Kartendienste unterstützen. Und auf der anderen Seite gibt es viele Programme, über die Sie Kartendienste konfigurieren können.

MapServer kann OGC-Webdienste sowohl als Datenquelle einbinden als auch selbst Karten als Dienst bereitstellen. Deshalb wird der UMN MapServer sowohl als Client als auch als Server verwendet. Um mit MapServer selbst Karten anzubieten, müssen Sie lediglich Ihre Mapdatei entsprechend aufbauen. Dieses Kapitel zeigt Ihnen, wie Sie MapServer für die Verwendung von Webdiensten einrichten müssen. Sie werden lernen, wie Sie eine Ebene definieren, die ihre Daten aus einer entfernten Quelle bezieht, und wie Sie Ihre eigenen Daten per Webdienst bereitstellen können. Bei der zweiten Variante sprechen wir auch von einem kaskadierenden Kartenserver.

Von UMN MapServer unterstützte OGC-Standards

MapServer implementiert viele der Standards des OGC oder zumindest Teile davon. Eine Übersicht der von MapServer unterstützten Standards finden Sie in Tabelle 12-1. Beachten Sie, dass sich sowohl die Standards als auch die Software unabhängig voneinander weiterentwickeln. Es ist also gut möglich, dass die Tabelle nicht mehr aktuell ist. Die neuesten Informationen und aktuellen Versionen der Standards finden Sie immer auf den Webseiten des OGC bzw. des UMN MapServer.

Tabelle 12-1: OGC-Standards, die UMN MapServer unterstützt

Abkürzung	Name	Einsatz
WMS	Web Map Service	WMS dient zum Bereitstellen und Anfordern dynamischer Karten, wobei die Datengrundlage aus Vektor- und Rasterdaten besteht.
WFS	Web Feature Service	WFS dient zum Bereitstellen und Anfordern von vektorbasierten Daten im GML-Format.
WCS	Web Coverage Service	WCS dient zum Bereitstellen von Rasterdaten.
WMC	Web Map Context	WMC dient um Speichern und Laden einer Kartenansicht, die sich aus Layern verschiedener WMS zusammensetzt, unter Verwendung von XML.
SLD	Styled Layer Descriptors	SLD ermöglicht die Definition von Stilvorgaben im XML-Format, die an einen WMS übergeben werden können.
GML	Geography Markup Language	GML ist ein XML-basiertes Format für den Austausch raumbezogener Objekte.
Filter	Filter Encoding	Über ein XML-basiertes Format können Filter definiert werden. Diese kommen bei Verwendung von WFS sowie SLD zum Einsatz.
SOS	Sensor Observation Service	SOS wird eingesetzt, um Informationen von Sensoren bereitstellen und abfragen zu können.
WMS time support	WMS mit Unterstützung für zeitliche Dimensionen	Über den WMS können Daten für einen bestimmten Zeitraum angezeigt werden.

Von den in Tabelle 12-1 aufgelisteten Standards sind zwei besonders oft im Einsatz: Web Map Service (WMS) und Web Feature Service (WFS). Diese zwei Standards werden wir in diesem Kapitel näher betrachten.



Unter der Adresse <http://ms-ogc-workshop.maptools.org/> können Sie das Paket »MapServer OGC Web Services Workshop« herunterladen. Es veranschaulicht, wie die verschiedenen Dienste genutzt werden können.

Referenzdokumente und Anleitungen zum Einsatz von WMS und WFS finden Sie im Abschnitt »OGC« in der Liste der MapServer-How-tos. Dort finden Sie die aktuellsten Informationen und ausführliche Beschreibungen zur Verwendung von MapServer als WMS und WFS sowie zu anderen OGC Webdiensten (<http://mapserver.gis.umn.edu/docs/howto>).

Schauen Sie sich auch die Beispiele unter dem folgenden Link an:

<http://www.webmappingillustrated.com/beispiele>

Hier erhalten Sie verschiedene Beispiele für die Verwendung von OGC-Standards mit MapServer. Beachten Sie, dass die abgedruckten Beispiele in diesem Buch nur die notwendigen Parameter enthalten und nicht alle Möglichkeiten, die MapServer bietet, vorstellen können.

Wenn Sie UMN MapServer als WMS-Client verwenden wollen, können Ihnen die folgenden Links dabei helfen, öffentlich zugängliche Webdienste zu finden. Schauen Sie sich dazu auch noch die Links aus Tabelle 5-1 an. Hier finden Sie weitere Verweise zu öffentlich zugänglichen Webdiensten:

- GeoConnections Discovery Portal: <http://geodiscover.cgdi.ca/gdp/search?action=searchForm&entryType=webService>
- Refrations Research: OGC Services Survey: <http://www.refrations.net/ogcsurvey/>
- National Geophysical Data Center: Interactive Map Services: <http://www.ngdc.noaa.gov/maps/interactivemaps.html>

OGC Web Map Service (WMS)

Der OGC WMS-Standard definiert, wie Geodaten über einen Kartenserver bereitgestellt und abgerufen werden. Solchermaßen standardisierte Kartendienste können über das Internet (und andere Protokolle) angesprochen werden, um dynamisch erzeugte Karten anzufordern.

Die Kommunikation zwischen Client und Server erfolgt über Requests, die einen definierten Aufbau haben. Für den WMS sind die folgenden Requests definiert (WMS Version 1.1.1):

GetCapabilities

Liefert eine Beschreibung des Diensts als XML zurück.

GetMap

Liefert dynamisch erzeugte Karten zurück.

GetFeatureInfo

Gibt Informationen zu Objekten aufgrund eines Klickereignisses in der Karte zurück (optional, wird aber von den meisten Servern unterstützt).

GetLegendGraphic

Liefert eine Legende zur Kartenebene als Bild zurück (optional).

DescribeLayer, GetStyles, PutStyles

Dies sind optionale Request, über die nicht jeder WMS verfügt. Die Request dienen dem Einsatz von WMS und SLD.

Über MapServer können dynamische Karten von externen Kartendiensten, die WMS unterstützen, über einen GetMap-Request angefordert werden. Es gibt zahlreiche WMS-Quellen, die Sie nutzen und in Ihre Projekte beispielsweise als Hintergrundkarte einbinden können.

Für das Anfordern einer Karte per WMS müssen definierte REQUEST-Parameter angegeben werden. Diese Parameter geben beispielsweise die Projektion, den Darstellungsbe-
reich, die Bildgröße und die gewünschten Ebenen an. Die Parameter des GetMap-Aufrufs sind in Tabelle 12-2 aufgelistet und werden kurz beschrieben.

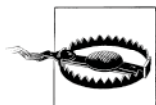
Tabelle 12-2: Benötigte WMS-Parameter für eine GetMap-Anfrage

Request-Parameter	Funktion	Beschreibung
SERVICE=WMS	Art des Diensts	Es werden verschiedene OGC-Services unterstützt. Über WMS wird angegeben, dass Sie eine WMS-Anfrage vornehmen möchten.
REQUEST=GetMap	Karte anfordern	Über GetMap wird eine dynamische Karte von einem WMS angefordert.
VERSION=1.1.1	WMS-Version	Angabe der WMS-Version, die verwendet werden soll.
SRS=EPSG:4326	SRS Spatial Reference System	Die Projektion, in der das Kartenbild angefordert werden soll, wird über den EPSG-Code angegeben. Entsprechend muss der Parameter BBOX in dieser Projektion angegeben werden.
FORMAT=image/jpeg	Ausgabeformat	Welches Bildformat soll für die Ausgabekarte verwendet werden?
LAYERS=countries, roads, water, cities	Liste der Layernamen	Über eine kommaseparierte Liste werden die Namen der Layer angegeben, die angefordert werden sollen.
WIDTH=800 HEIGHT=600	Bildgröße	Angabe zur Höhe und Breite des Ausgabebilds in Pixeln. Beachten Sie, dass viele Server die mögliche Bildgröße einschränken.

Tabelle 12-2: Benötigte WMS-Parameter für eine GetMap-Anfrage (Fortsetzung)

Request-Parameter	Funktion	Beschreibung
BBOX=10,0,50,30	Bounding Box	Der Bereich, der angefordert werden soll, wird über die linke untere Ecke und die rechte obere Ecke in den Einheiten des SRS angegeben. Die Angabe erfolgt durch Kommata getrennt.
STYLES=gray,red oder STYLES=,,,	Darstellungsoptionen für die einzelnen Ebenen	Sofern der Dienst STYLES unterstützt, können die Layer in unterschiedlicher Darstellung angefordert werden. Ohne Angabe wird der Default-Style verwendet.

Wenn Sie über MapServer von einem entfernten Dienst Karten anfordern wollen, reicht es in vielen Fällen schon aus, die Angabe zu SRS, die gewünschten Layer und den Dateityp in der MapServer-Mapdatei anzugeben. Den Rest erledigt MapServer während der Kartenanfrage. MapServer bestimmt die Bildgröße der Karte und fordert den gewünschten Ausschnitt vom Kartendienst entsprechend an. Sie können den GetMap-Aufruf aber auch manuell zusammenstellen und beispielsweise im Browser abschicken, um ein Bild anzufordern. Hierbei müssen sämtliche Request-Parameter angegeben werden, damit der Server die Anfrage korrekt auswerten und das entsprechende Bild dynamisch erstellen kann. Wenn Sie WMS nutzen möchten, ist es sicherlich gut, sich den Aufbau des GetMap-Requests einmal genauer anzuschauen, um die Kommunikation zwischen Client und Server zu verstehen.



Bei der Angabe der Parameternamen wird nicht zwischen Groß- und Kleinschreibung unterschieden – bei den Werten dagegen schon. Möglicherweise müssen Sie die Werte an den verwendeten Server anpassen, obwohl vom Server eine gewisse Transferleistung erwartet werden könnte, sodass beispielsweise sowohl die Angabe GetMap als auch getMap möglich ist. Eine genaue Beschreibung der WMS-Request-Parameter erhalten Sie über die OGC WMS-Spezifikation (<http://www.opengeospatial.org/standards/wms>). Beachten Sie bei der Angabe im Request auch die korrekte Schreibweise der Ebenennamen.

Anfordern des WMS Capabilities-Dokuments

Wenn Sie einen WMS gefunden haben, haben Sie vermutlich eine URL bekommen, über die Sie den Dienst ansprechen können. Einen WMS sprechen Sie über einen GetCapabilities-Aufruf an. Der Aufruf ist nach einem bestimmten Schema aufgebaut und liefert eine Beschreibung des Diensts als XML-Dokument zurück. Dieses Dokument wird als Capabilities-Dokument bezeichnet.

Auf der Seite <http://www.refrations.net/ogcsurvey/> finden Sie eine Liste mit öffentlich zugänglichen WMS und auch anderen OGC-Diensten. Die dort aufgeführten Links weisen nicht auf normale Webseiten, sondern auf WMS Capabilities-Dokumente, die den jeweiligen Dienst beschreiben. Diese Informationen müssen Sie als Anfänger nicht alle interpretieren können. Das Capabilities-Dokument ist der Standard, über den die Informationen über den Dienst ausgetauscht werden. Ein Client, der WMS unterstützt,

weiß über diese Informationen, wie der Dienst aufgebaut ist, und kann entsprechend einen Aufruf zusammenstellen, um beispielsweise eine Karte von dem Dienst anzufordern.

In den folgenden Beispielen wird ein beliebiger WMS verwendet, der Satellitenbilder der Erde und verschiedene Ebenen für die USA bereitstellt. Dieser Dienst wird über den folgenden getCapabilities-Request angesprochen:

```
http://wms.jpl.nasa.gov/wms.cgi?REQUEST=GetCapabilities&SERVICE=
WMS&VERSION=1.1.1
```

Was hat es nun mit diesem Request auf sich? Je nachdem, welche WMS-Version Sie verwenden, wird nach dem Abschicken des Requests direkt ein XML-Dokument in Ihrem Browser angezeigt. Es kann aber auch sein, dass Ihnen nach dem Abschicken des Requests eine Datei zum Speichern angeboten wird. Diese können Sie speichern und in einem Editor oder im Browser aufrufen.

Es wird ein XML-Dokument zurückgegeben, das Informationen über den abgefragten WMS enthält. Sie können das Capabilities-Dokument auch mit dem Kommandozeilenwerkzeug `wget` anfordern und direkt als Datei speichern. Der Befehl lautet beispielsweise:

```
> wget 0 jplwms.xml "http://wms.jpl.nasa.gov/wms.cgi?request=GetCapabilities
&service=WMS&version=1.1.1"
```

Das ist schon alles. Die Option `-O` speichert den Download unter dem nach der Option angegebenen Dateinamen. Ohne diese Option wird der Inhalt auf dem Bildschirm ausgegeben. Verwenden Sie beim Speichern die Dateierweiterung `.xml`.

In Beispiel 12-1 sehen Sie einige der wichtigsten Abschnitte im Capabilities-Dokument. Darüber hinaus enthält das Dokument viele weitere Informationen, wie beispielsweise Kontaktinformationen, eine Beschreibung des Diensts sowie mögliche Nutzungsbeschränkungen. Am wichtigsten sind für Sie allerdings die Angaben zu den Ebenen, zur Projektion und den unterstützten Bildformaten.

Beispiel 12-1: Auszug aus einem WMS Capabilities-Dokument

```
...
<Title>OnEarth Web Map Server</Title>
  <SRS>EPSG:4326</SRS>
  <SRS>AUTO:42003</SRS>
...
<Layer queryable="0" opaque="0" noSubsets="0">
  <Name>global_mosaic</Name>
  <Title>WMS Global Mosaic, pan sharpened</Title>
  ...
  <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90"/>
...
</Layer>
...
```


Das Capabilities-Dokument

Das Capabilities-Dokument liefert Metadaten zum Dienst im XML-Format und ist nach einem definierten Schema aufgebaut. Dieses Schema wurde vom OGC erarbeitet und wird mit jeder WMS-Version verändert und erweitert.

Das Capabilities-Dokument baut sich aus verschiedenen Bereichen auf:

Service

Enthält allgemeine Informationen zum WMS.

Capabilities

Enthält die Bereiche Request und Layer.

Request

Enthält eine Beschreibung der zur Verfügung stehenden Requests sowie deren unterstützten Formate und gibt die Adresse (OnlineResource) an, über die diese Requests angesprochen werden können.

Layer

Hier können Teile der Karte als Ebenen angegeben werden. Der Layer-Block enthält unter anderem Informationen über den Namen und den Titel der Ebenen und über den Maßstabbereich, in dem sie angezeigt werden. Falls die Anzeige bestimmter Ebenen dieses Diensts auf bestimmte Projektionen beschränkt ist, wird das ebenfalls hier vermerkt. Optional können unterschiedliche kartografische Darstellungstypen angeboten werden. Es wird angegeben, ob Ebenen Informationen zu den Objekten bereitstellen und ob Legendengrafiken vorliegen. Ebenen können hierarchisch verschachtelt werden. Die erste Ebene ist dabei immer die hierarchisch oberste und wird deshalb als Wurzelebene (root layer) bezeichnet. Sie enthält allgemeine Informationen und umschließt logisch alle weiteren Ebenen, die der WMS definiert.

Um einen Dienst anzusprechen, also z.B. ein Bild anzufordern, sind vor allem die Online-Resource und die Namen der Layer von Interesse.

Das Capabilities-Dokument wird benötigt, um einen Dienst korrekt ansprechen zu können. Das Dokument sollte zusätzlich über Metadaten verfügen, die die Qualität, das Format, den Ursprung, Kontaktdaten und weitere Informationen enthalten. Suchmaschinen und Katalogdienste benötigen diese Informationen, um den Kartendienst in ihre Anfragekategorien und Kataloge einbinden zu können.

Im folgenden Abschnitt gehen wir nur auf die Bereiche des Capabilities-Dokuments ein, die für Sie wichtig sind.

Bei der Verwendung von WMS-Ebenen mit MapServer ist die Projektion von besonderer Bedeutung. Der entfernte Dienst stellt die Karten möglicherweise in verschiedenen Projektionen zur Verfügung. Wenn Sie einen Layer eines Diensts ansprechen möchten, müssen Sie daher nachschauen, in welcher Projektion dieser Dienst angefordert werden kann. Es wäre gut, wenn der Dienst die von Ihnen verwendete Projektion unterstützt. Wo finden Sie nun die Informationen im Capabilities-Dokument? Die drei ersten Zeilen

aus Beispiel 12-1 zeigen den Titel des Dienstes (Title) und zwei SRS-Angaben, über die die zur Verfügung stehenden Koordinatenreferenzsysteme definiert werden. Wenn Sie MapServer dazu einsetzen, einen entfernten WMS anzusprechen, können Sie den Dienst auch dann nutzen, wenn die von Ihnen verwendete Projektion nicht aufgeführt ist. Allerdings müssen Sie MapServer dann anweisen, das gelieferte Bild umzuprojizieren, damit es zu den übrigen Ebenen Ihres Projekts passt. In unserem Beispiel verwenden wir den EPSG-Code 4326. 4326 sollte von jedem WMS unterstützt werden und steht für das globale Koordinatenreferenzsystem WGS84:

```
<SRS>EPSG:4326</SRS>
```

Sie können die Karte direkt vom WMS anfordern, ohne dass MapServer zusätzlich umprojizieren muss.

Zusätzlich wird 42003 vom WMS unterstützt:

```
<SRS>AUTO:42003</SRS>
```

Die offizielle Liste des EPSG geht derzeit bis ca. 32000. Die Zahlen ab 40000 sind für eigene Projektionen reserviert, die nicht über EPSG definiert sind. Deshalb darf es auch nicht EPSG:42003 heißen, sondern AUTO:42003.

Der folgende Abschnitt aus Beispiel 12-1 gibt die Informationen zu einem Layer aus und ist der wichtigste Abschnitt, sofern Sie bereits wissen, dass der Dienst für Sie nützlich ist:

```
<Layer queryable="0" opaque="0" noSubsets="0">
  <Name>global_mosaic</Name>
  <Title>WMS Global Mosaic, pan sharpened</Title>
  ...
  <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90"/>
</Layer>
```

In den Abschnitten der einzelnen Layer finden Sie die nötigen Informationen, um eine Ebene anzufordern. Die wichtigste Angabe ist hierbei das Tag Name:

```
<Name>global_mosaic</Name>
```

Dies ist der eindeutige Name der Ebene, über den Sie den Layer des WMS aus anderen Anwendungen heraus ansprechen können. In unserem Beispiel lautet der Name global_mosaic. Die Layernamen sind oftmals recht kryptisch. Die Angabe unter Title sowie die Beschreibung des Layers im Tag Abstract liefern möglicherweise genauere Hinweise auf die Daten.

Der nächste Abschnitt in Beispiel 12-1 bestimmt den Darstellungsbereich der Daten (BoundingBox). Die Angabe erfolgt über die LatLonBoundingBox:

```
<LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90"/>
```

Diese Information kann sehr hilfreich sein, wenn Sie nicht wissen, welchen Bereich der WMS abdeckt. Die Angabe in diesem Beispiel umfasst den gesamten Globus. Sie können die Daten also in einer beliebigen Anwendung benutzen, unabhängig davon, welchen Bereich Sie darstellen wollen.

Im Capabilities-Dokument finden sich aber auch noch andere nützliche Informationen, wie beispielsweise die Angabe `<MinScaleDenominator>20000</MinScaleDenominator>`. Diese Angabe kann Ihnen helfen, den Darstellungsbereich der Ebene zu ermitteln.

Wenn Sie eine Karte von einem WMS anfordern möchten, können Sie sich aufgrund der Informationen des Capabilities-Dokumentes einen GetMap-Request zusammenstellen und über diesen Request eine Karte anfordern. Diese Aufgabe übernimmt für Sie normalerweise ein Client, wie beispielsweise MapServer, Mapbender oder Quantum GIS.

Gibt es mit Ihrer Anfrage ein Problem, erhalten Sie anstelle des Bilds eine Fehlermeldung, die entweder als Bild, Text oder XML zurückgeschickt wird. Auch wenn die Anfrage syntaktisch korrekt ist, kann es passieren, dass Sie nichts sehen, also ein leeres, völlig weißes Bild geliefert wird, z.B. wenn es für die gewünschte Gegend oder den genannten Maßstab keine Daten gibt. In diesem Fall erhalten Sie keine Fehlermeldung, sondern nur eine leere Karte.

Karten manuell von einem WMS anfordern

Jetzt verfügen wir über genügend Informationen, um einen eigenen Karten-Request zusammenzustellen. Über das Capabilities-Dokument haben Sie alle Informationen erhalten, um eine Karte von einem WMS anfordern zu können. Der Aufbau eines solchen Requests wird in Beispiel 12-2 gezeigt; er verdeutlicht, was hinter den Kulissen passiert. Abbildung 12-2 zeigt die Karte, die zurückgeliefert wird.

Die URL wurde hier aus Gründen der Lesbarkeit auf mehrere Zeilen verteilt. Wenn Sie den Aufruf abschicken möchten, muss die Anfrage allerdings in einer Zeile stehen.

Beispiel 12-2: WMS GetMap-Anfrage zum Anfordern einer Karte

```
http://wms.jpl.nasa.gov/wms.cgi
  ?request=GetMap
  &service=WMS
  &version=1.1.1
  &srs=EPSG:4326
  &format=image/jpeg
  &styles=
  &bbox=-180,-90,180,90
  &width=600
  &height=300
  &layers=global_mosaic
```

Selbst dieses einfache Beispiel mag Ihnen kompliziert erscheinen. Neben den hier aufgeführten Optionen gibt es noch ein paar weitere Parameter, die an dieser Stelle aber nicht angesprochen werden sollen. Die letzten vier Parameter (`layers`, `bbox`, `width` und `height`) können Sie leicht selbst anpassen, um beispielsweise eine größere Karte anzufordern, den Darstellungsbereich zu ändern oder um andere Ebenen anzuzeigen.

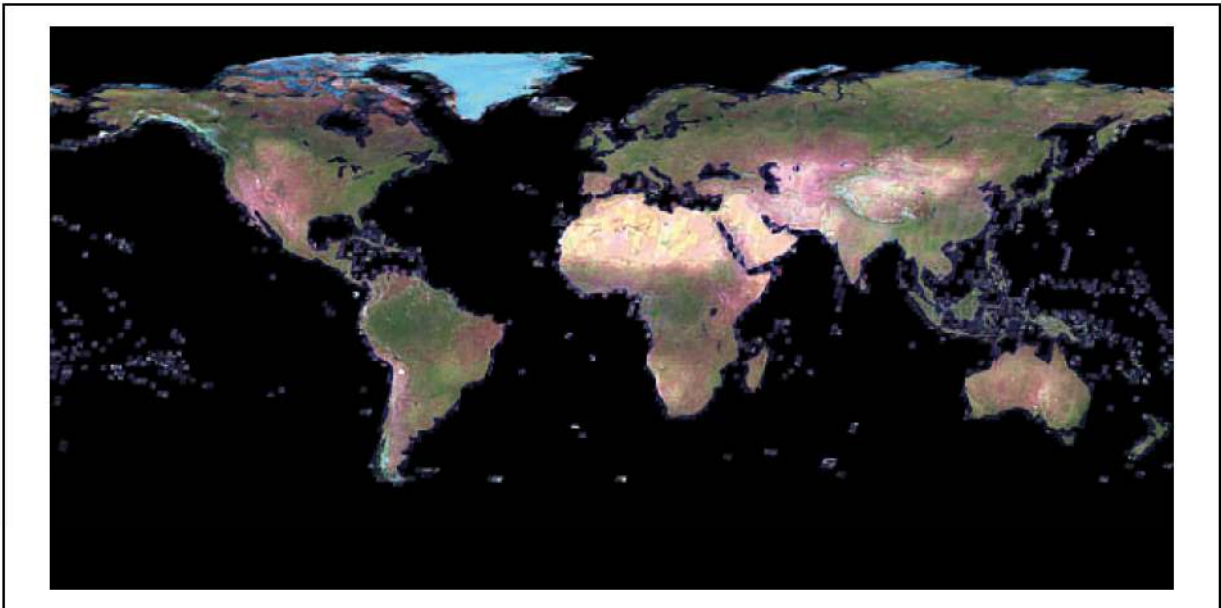


Abbildung 12-2: Karte, die auf den GetMap-Aufruf zurückgeliefert wird

Der Parameter `layers` kann noch um Ebenennamen erweitert werden. Beispiel 12-3 zeigt eine Karte, die eine weitere Ebene aus dem Capabilities-Dokument einbindet. Diese Ebene hilft, die Lücken in der Ebene `global_mosaic` aufzufüllen. Jetzt werden auch die Antarktis und Teile von Südostasien mit angezeigt.

Beispiel 12-3: Über einen GetMap-Aufruf können mehrere Ebenen von einem WMS angefordert werden

```
http://wms.jpl.nasa.gov/wms.cgi
  ?request=GetMap
  &service=WMS
  &version=1.1.1
  &srs=EPSG:4326
  &format=image/jpeg
  &styles=,
  &bbox=-180,-90,180,90
  &width=600
  &height=300
  &layers=modis,global_mosaic
```

Wie Sie sehen, kann der Aufruf leicht geändert und so eine Karte mit zusätzlichen Informationen angefordert werden. Abbildung 12-3 zeigt die aufgrund der Abfrage erzeugte Karte.

Die Reihenfolge, in der Sie die Parameter im Request angeben, spielt eine Rolle. Im Beispiel wurde zuerst die Ebene `modis` angegeben. Diese Ebene wird zuunterst ausgegeben, `global_mosaic` wird darüber dargestellt.

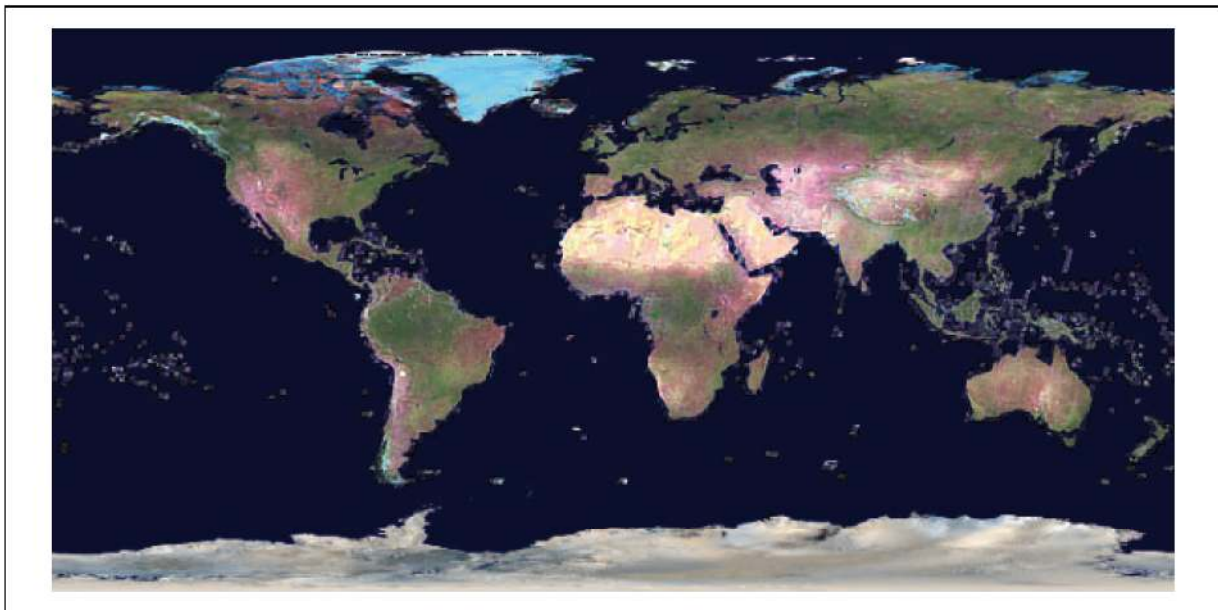


Abbildung 12-3: Karte, die durch das Anfordern mehrerer Layer des WMS zurückgeliefert wird

Wenn Sie ein wenig experimentieren möchten, können Sie die Einstellungen für den Parameter `bbox` verändern, um andere Bereiche der Welt anzufordern. Hierbei sollten Sie darauf achten, dass bei einer Bildgröße von 600×300 die Angabe unter `bbox` im Verhältnis 2:1 sein sollte.

Bodensee

`&bbox=8.736,47.321,9.9079,47.907`

Bagdad

`&bbox=44.22252,33.24291,44.58252,33.42291`

Gibraltar

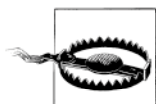
`&bbox=-6.825,35.3775,-4.575,36.5025`

Panamakanal

`&bbox=-80.30625,8.8725,-79.18125,9.435`

Sie können natürlich auch die Ebene `global_mosaic` komplett weglassen, um zu sehen, wie die Ebene `modis` für sich genommen aussieht.

Sämtliche verfügbaren Ebenen sind im Capabilities-Dokument aufgeführt und können dem Request hinzugefügt werden. Allerdings ist dies nicht unbedingt sinnvoll, da sich die Ebenen überlagern und möglicherweise verdecken. Außerdem stehen Layer nur für einen bestimmten Maßstabsbereich zur Verfügung (Angabe über den Parameter `ScaleHint`) und sind daher nicht immer sichtbar.



Beachten Sie, dass es sich bei dem in den Beispielen verwendeten WMS um den WMS eines Testservers handelt, d.h., es kann sein, dass sich der WMS ändert oder dass er nicht mehr zur Verfügung steht. Dann können Sie aber einfach einen anderen WMS auswählen und aufgrund der Kenntnisse, die Sie nun über WMS haben, ebenfalls manuell WMS-Requests erstellen, absetzen und so Karten anfordern.

UMN MapServer als WMS-Client

Das Absetzen von WMS-Abfragen über einen Webbrowser ist schon recht interessant, aber auch mühselig. Daher benötigen Sie einen Client, der für Sie die Anfragen zusammenstellt. Hier stehen Ihnen diverse WMS-Clients zur Verfügung. UMN MapServer, Quantum GIS, OpenLayers sowie Mapbender werden in diesem Buch als WMS-Clients vorgestellt. Darüber hinaus unterstützen aber noch viele andere Anwendungen das Einbinden von WMS. Über WMS-Clients können Sie die Informationen, die ein WMS liefert, mit eigenen Daten anzeigen lassen und verschiedene WMS in einem Projekt zusammenführen.

An dieser Stelle soll nun die Möglichkeit vorgestellt werden, UMN MapServer als WMS-Client einzusetzen. Das mag zunächst verwirrend erscheinen, da es sich bei MapServer ja, wie der Name schon sagt, um einen Server handelt und nicht um einen Client. Allerdings benötigt jeder Server auch Daten, die wiederum von einem anderen Server kommen können. Sobald ein MapServer jedoch von einem weiteren Server Daten bestellt, wird er dadurch selbst zum Client. Hier wird also der UMN MapServer als Client genutzt, um Karten zu erstellen, die von einem anderen Server bezogen werden. In Kapitel 13 werden zwei weitere Softwarepakete vorgestellt, die als Clients bezeichnet werden, obwohl zumindest der eine in manchen Fällen auch Serverfunktionen übernimmt. Die Trennlinie zwischen Client und Server ist je nach Blickwinkel an unterschiedlichen Stellen.

Um MapServer als WMS-Client nutzen zu können, müssen Sie in Ihrer MapServer-Mapdatei einen Layer definieren, der einen entfernten WMS anspricht und von diesem ein Bild anfordert. Dieses Bild wird dann anschließend in das Ausgabebild Ihres WMS integriert. Wir sprechen bei diesem Modell auch von einem kaskadierenden WMS, also einem WMS, der wiederum einen WMS anspricht. Diese Kaskadierung kann auch über mehrere WMS erfolgen.

Sie könnten einen Dienst aufsetzen, der nur auf entfernte Dienste zugreift. Interessanter ist aber sicherlich die Kombination eigener Daten mit Informationen, die über entfernte Dienste geliefert werden. Bedienen Sie sich der vielen Informationen, die über freie WMS zur Verfügung stehen. Es wäre sicherlich ein großer Aufwand, solche Daten selbst bereitzustellen und zu pflegen. Über die Kombination aus mehreren Diensten und Ihren eigenen Daten können Sie einfach ein aussagekräftiges Projekt aufsetzen.

Um MapServer als WMS-Client einzusetzen, müssen in der Mapdatei ein paar Einstellungen vorgenommen werden. Beispiel 12-4 zeigt eine Ebene, die den WMS aus Beispiel 12-2 als Datenquelle verwendet.

Beispiel 12-4: Definition einer Ebene in der Mapdatei, die auf einen entfernten WMS als Datenquelle zugreift

```
LAYER
  NAME global_landsat
  TYPE RASTER
  STATUS ON
```

Beispiel 12-4: Definition einer Ebene in der Mapdatei, die auf einen entfernten WMS als Datenquelle zugreift (Fortsetzung)

```
CONNECTIONTYPE WMS
CONNECTION "http://wms.jpl.nasa.gov/wms.cgi?" # WMS OnlineResource

METADATA
"wms_title" "Satellitenbilder (JPL)"      # gewünschter Titel
"wms_server_version" "1.1.1"             # WMS-Version
"wms_srs" "EPSG:4326"                    # angefordertes SRS
"wms_format" "image/jpeg"                # gewünschtes Bildformat
"wms_styles" ",pseudo_bright"            # angeforderter Stil der Layer
"wms_name" "modis,global_mosaic"         # Liste der Layer
END
END
```

Sie können den neuen Layer beispielsweise in die Mapdatei *global.map* aus den vorigen Kapiteln einbauen. Dazu können Sie die Definition entweder in die Mapdatei eintragen, oder Sie speichern die Definition in einer separaten Datei und binden diese über `INCLUDE` in die Mapdatei ein. Dies sieht dann beispielsweise so aus:

```
INCLUDE "wms_globaljpl.map"
```

Beachten Sie beim Einbinden in die Mapdatei die Reihenfolge der Layer. Ebenen, die in der Mapdatei zuerst definiert werden, werden als unterste Ebene im Bild ausgegeben.

Achten Sie darauf, dass Sie die Parameter `IMAGEPATH` und `IMAGEURL` im `WEB`-Objekt in der Mapdatei angeben, da das Bild, das von dem Kartendienst geliefert wird, temporär gespeichert wird.

Um ein zufriedenstellendes Ergebnis zu erzeugen, müssen Sie aber noch ein paar Änderungen vornehmen. Standardmäßig liefert MapServer ein Bild mit 8 Bit Farbtiefe zurück. Diese Farbtiefe ist nicht ausreichend für die Darstellung der Satellitenbilder. Deshalb ist es sinnvoll, das Bild in einer höheren Farbtiefe anzufordern.

Hierfür definieren Sie ein `OUTPUTFORMAT`-Objekt, das beim Anfordern eines PNG ein Bild mit einer Farbtiefe von 24 Bit ausgibt (`IMAGEMODE RGB`):

```
OUTPUTFORMAT
NAME png
DRIVER "GD/PNG"
MIMETYPE "image/png"
IMAGEMODE RGB
EXTENSION "png"
END
```

Verwenden Sie die Datei *global.map*, müssen Sie außerdem im Layer `countries` die Hintergrundfarbe der Länderpolygone entfernen, damit Sie die darunter liegende WMS-Ebene auch sehen. Dazu können Sie den Parameter `COLOR` entfernen oder die Hintergrundfarbe über die Angabe `COLOR -1 -1 -1` auf transparent setzen. Die vollständige *global.map*-Datei finden Sie am Ende dieses Kapitels (Beispiel 12-13).

Nun zu den Angaben, die innerhalb des Layers gemacht werden müssen. Die Definition `CONNECTIONTYPE WMS` ist notwendig, damit MapServer weiß, dass er eine WMS-Anfrage stellen soll. Über den Parameter `CONNECTION` wird die `OnlineResource` des Diensts angegeben. Diese können Sie über das Capabilities-Dokument ermitteln.



Handelt es bei dem WMS, den Sie ansprechen, um einen WMS, der über den UMN MapServer bereitgestellt wird, wird möglicherweise in der `OnlineResource` der Pfad zur Mapdatei über den zusätzlichen Parameter `map` angegeben. Eine entsprechende `OnlineResource` könnte dann wie folgt aussehen: http://www.gdi-rp-dienste2.rlp.de/cgi-bin/mapserv?map=/data/umn/rlp_relief.map. Nach der WMS-Spezifikation 1.1.1 sollte aber schon der Aufruf bis zum `?` alle notwendigen Informationen enthalten, um den Dienst anzusprechen, sodass kein zusätzlicher Parameter übergeben werden muss. In der MapServer-Dokumentation finden Sie eine Anleitung dazu unter http://mapserver.gis.umn.edu/docs/howto/wms_server.

Die Angaben im `METADATA`-Objekt entsprechen den WMS-Angaben, die Sie schon bei der Beschreibung des `GetMap`-Aufrufs kennengelernt und beim Anfordern einer Karte über den Browser verwendet haben. Wenn Sie einen WMS über einen Layer in der Mapdatei ansprechen möchten, müssen Sie allerdings nicht alle Parameter, die Sie für den `GetMap`-Request kennengelernt haben, angeben. MapServer braucht lediglich die notwendigen Informationen, um einen `GetMap`-Request zusammenstellen zu können. Die Angaben für `width`, `height` und `bbox` werden beispielsweise von MapServer selbst generiert, je nachdem, welcher Kartenausschnitt und welche Kartengröße angefordert werden soll. Über `"wms_name"` werden die Namen der Layer, die von dem entfernten WMS angefordert werden sollen, kommasepariert angegeben. Über `"wms_title"` können Sie dem Layer einen Titel geben, der von Clients für die Ansicht verwendet werden kann.

Der WMS-Layer bekommt von dem angesprochenen WMS ein Rasterbild zurück, daher geben Sie in der Layerdefinition `TYPE Raster` an. Der Layer besitzt kein `CLASS`-Objekt. Sie können die Darstellungsart der WMS-Ebene aber ändern, indem Sie über den Parameter `"wms_styles"` einen anderen Stil angeben. Dies ist nur möglich, sofern der Dienst neben dem Default-Style noch andere Styles unterstützt. Eine andere Möglichkeit bietet die Übergabe eines SLD (*OGC Styled Layer Descriptors*). Über SLD ist es möglich, bestimmte Stilvorgaben für einen WMS-Layer zu definieren. Dies ist eine optionale Funktionalität für einen WMS und wird nicht von jedem WMS unterstützt. In diesem Buch werden wir nicht weiter auf SLD eingehen. Eine sehr gute Beschreibung zur Verwendung von SLD mit MapServer finden Sie unter <http://mapserver.gis.umn.edu/docs/howto/sldhowto>.



Es ist möglich, bestimmte Informationen zu WMS-Ebenen abzufragen. Anstelle einer `getMap`-Anfrage kommt hierfür eine Anfrage vom Typ `getFeatureInfo` zum Einsatz. Details zu den hierfür benötigten Parametern finden Sie in der WMS-Spezifikation des OGC unter der Adresse <http://www.opengis.org/specs/?page=specs>.

WMS bietet die Interoperabilität zwischen entfernten Anwendungen. In den bisherigen Beispielen haben Sie MapServer als WMS-Client kennengelernt.

Übersicht verschiedener Open Source-WMS-Clients

Es ist eine der großen Stärken von WMS, dass es von keinem anderen Programm abhängig ist und an kein anderes Softwarepaket gebunden ist. Entsprechend gilt das ebenso, wenn MapServer als WMS-Server verwendet wird. Jedes Programm, das WMS unterstützt, kann die über den Dienst bereitgestellten Informationen anfordern.

Neben dem UMN MapServer gibt es eine Reihe weiterer Programme, die als WMS-Client verwendet werden können. Hier eine kleine Auswahl aus den Bereichen Desktop-GIS und WebGIS:

Open Source-Desktop-GIS, die WMS unterstützen:

- Quantum GIS (QGIS) (<http://www.qgis.org>)
- JUMP (<http://jump-project.org>)
- uDig (<http://udig.refractions.net>)
- gvSIG (<http://www.gvsig.gva.es>)
- OpenEV (<http://fwtools.maptools.org>)

Open Source-WebGIS-Clients, die WMS unterstützen:

- Cartoweb (<http://www.cartoweb.org/>)
- Mapbender (<http://www.mapbender.org>)
- MapBuilder (<http://communitymapbuilder.osgeo.org/>)
- OpenLayers (<http://www.openlayers.org/>)

Auch viele proprietäre GIS-Programme unterstützen inzwischen das Einbinden von OGC WMS-Kartendiensten, weil diese Art der verteilten Datenhaltung den Stand der Technik darstellt. Allerdings hat es etwas länger gedauert als bei vielen Open Source-Projekten, weil es nicht unbedingt im geschäftlichen Interesse eines proprietären Herstellers liegt, mit anderen Systemen kompatibel zu sein. Der Druck der Anwender hat aber dazu geführt, dass sich heute praktisch niemand mehr leisten kann, OGC WMS nicht zu unterstützen, egal ob es sich um Massenmarktsoftware wie GoogleEarth handelt oder um professionelle GIS-Software wie die ArcGIS-Server-Palette. Der Weg dorthin wurde allerdings auch wieder maßgeblich durch Open Source-Software bereitet: So wird z.B. die Professional-Variante von GoogleEarth mit der bereits vorgestellten Open Source-Bibliothek-GDAL ausgeliefert, und die ersten OGC WMS-Konnektoren für ArcView wurden als Open Source-Erweiterung durch Refrations entwickelt.

UMN MapServer als WMS-Server

MapServer kann auch als WMS-Server, also als Kartenserver, eingesetzt werden, um Karten über einen Dienst öffentlich oder im Intranet bereitzustellen. Um einen Dienst über MapServer bereitzustellen, müssen lediglich ein paar zusätzliche Parameter in der Mapdatei angegeben werden.

Die Anpassungen in der Mapdatei müssen sowohl im Map-Objekt (allgemeine Einstellungen) als auch in den einzelnen Layer-Objekten erfolgen. Die WMS-Parameter werden dabei im METADATA-Objekt angegeben.

Im Map-Objekt befindet sich das METADATA-Objekt innerhalb des WEB-Objekts. Das METADATA-Objekt enthält allgemeine Angaben zu Ihrem WMS, wie Titel, Beschreibung, Kontaktinformationen, unterstützte Projektionen.

Schauen Sie sich dazu nochmals die Capabilities-Dokumente der zwei Dienste an, die in diesem Kapitel schon vorgestellt wurden. Der zweite Dienst wird von GeoPortal.rlp über den UMN MapServer bereitgestellt.

```
http://wms.jpl.nasa.gov/wms.cgi?request=GetCapabilities&service=WMS&version=1.1.1
http://www.gdi-rp-dienste2.rlp.de/cgi-bin/mapserv?map=/data/umn/rlp_relief.map&REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.1
```

Beispiel 12-5 zeigt, wie die Anpassung für die Datei *global.map* aussieht. Die gesamte Mapdatei *global.map* finden Sie auch in Beispiel 12-13 am Ende dieses Kapitels. Die WMS-Parameter werden in dem Beispiel hervorgehoben.

Beachten Sie, dass es zahlreiche WMS-Parameter gibt, es müssen jedoch nur einige müssen angegeben werden. Eine Übersicht finden Sie in der MapServer-Dokumentation unter http://mapserver.gis.umn.edu/docs/howto/wms_server. Beachten Sie, dass die Namen der Parameter in der Mapdatei sich an die Namen der Tags im Capabilities-Dokument anlehnen.

Beispiel 12-5: METADATA-Objekt innerhalb des WEB-Objekts, über das die WMS-Parameter zur Beschreibung des Diensts angegeben werden

```
WEB
  TEMPLATE global.html
  IMAGEPATH "/srv/www/htdocs/tmp/"
  IMAGEURL "/tmp/"
  METADATA
    "wms_title" "My Global Map WMS Server"
    "wms_onlineresource" "http://localhost/cgi-bin/mapserv?map=/src/www/htdocs/global.map"
    "wms_srs" "EPSG:4326 EPSG:31466 EPSG:31467 EPSG:31468"
    "wms_abstract" "WMS Server powered by UMN MapServer."
  END
END
```

Auch die Angaben im LAYER-Objekt müssen erweitert werden. Im METADATA-Objekt sollten Sie den Parameter `wms_title` angeben. Zusätzlich kann die Angabe von `wms_srs`, `wms_abstract` und `wms_keywords` sinnvoll sein, sie ist aber nicht zwingend erforderlich.

Beispiel 12-6 zeigt, wie die WMS-Angaben eines Layers aussehen könnten. Die vollständige Mapdatei finden Sie in Beispiel 12-14.

Beispiel 12-6: WMS-Angaben für ein Layer in der MapServer-Mapdatei

```
LAYER
  NAME countries                                # Name für Layer muss angegeben werden
  TYPE POLYGON
  STATUS ON                                     # STATUS muss ON sein
  DATA countries_simpl
  METADATA                                     # Beginn des METADATA-Objekts
    "wms_title" "Country boundaries" # Titel der Ebene
    "wms_srs" "EPSG:4326"
  END                                           # Ende des METADATA-Objekts
  CLASSITEM 'NAME'
  CLASS
    NAME 'Bulgaria'
    EXPRESSION 'Bulgaria'
    STYLE
      OUTLINECOLOR 100 100 100
    END
  END
  CLASS
    NAME 'All Countries'
    EXPRESSION ('[NAME]' ne 'Bulgaria')
    STYLE
      OUTLINECOLOR 100 100 100
    END
  END
END
```

Anhand dieser Einstellungen wird die Ebene im WMS über den Namen `countries` zugänglich gemacht. Der Name wird im HTTP-Request, beispielsweise beim GetMap-Request, als Parameter angegeben.

Wenn Sie eine Ebene über einen WMS veröffentlichen, sollte `STATUS ON` angegeben werden. Der Client entscheidet über die Angabe im Request, welche Ebenen angefordert werden sollen.

Im METADATA-Objekt des Layers geben Sie einen `wms_title` für die Ebene an. Dieser Titel wird im Capabilities-Dokument ausgegeben und kann von einem Client als Titel für den Layer verwendet werden.

Ein Blick auf Ihr WMS Capabilities-Dokument

Wenn Sie die Anpassungen vorgenommen haben, können Sie sich das Capabilities-Dokument anschauen. Die Parameter, die Sie in der Mapdatei angegeben haben, werden im Capabilities-Dokument Ihren WMS ausgegeben. Schicken Sie den GetCapabilities-

Request ab und schauen Sie sich die Angaben an. Beachten Sie, dass Sie den GetCapabilities-Aufruf an Ihr System anpassen müssen. Der Aufruf setzt sich aus der Angabe unter `wms_onlineresource` und den Parametern `Request`, `VERSION` und `SERVICE` zusammen:

```
http://localhost/cgi-bin/mapserv?map=/src/www/htdocs/global.map?REQUEST=
GetCapabilities&SERVICE=WMS&VERSION=1.1.1
```

Die `OnlineResource` enthält in unserem Fall den Pfad zum `mapserv-Executable` und über den Parameter `map` den Pfad zur Mapdatei.

Geben Sie die Parameter `Request`, `VERSION` und `SERVICE` nicht an, erhalten Sie von MapServer die Meldung:

```
mapserv(): Web application error. No template provided.
```

Über den `GetCapabilities-Request` erhalten Sie das XML-Dokument mit dem Informationen zu Ihrem Dienst. MapServer zeigt Warnungen an, wenn die Konfiguration in der Mapdatei nicht korrekt erfolgte. Bevor Sie Ihren WMS in einen Client laden, sollten Sie immer prüfen, ob das `GetCapabilities-Dokument` zurückgegeben wird und die korrekten Angaben enthält. Im unten stehenden Beispiel wurde die `<Style>`-Definition weggelassen.

```
...
<Layer queryable="0" opaque="0" cascaded="0">
  <Name>countries</Name>
  <Title>Country boundaries</Title>
  <Style>
    ...
  </Style>
</Layer>
...
```

Jetzt kann der Dienst über einen WMS-Client geladen werden. Schauen Sie sich Ihren WMS nun über Quantum GIS an. Das Laden eines WMS über Quantum GIS wird in Kapitel 8 beschrieben und in Abbildung 8-12 verdeutlicht. Quantum GIS lädt den WMS über den `GetCapabilities-Aufruf` und kann anschließend dynamisch Karten über den `GetMap-Aufruf` von dem Dienst anfordern. Im Kapitel 13 werden Sie mit `OpenLayers` und `Mapbender` weitere WMS-Clients kennenlernen, mit denen Sie Ihren Dienst ansprechen können.

OGC Web Feature Service (WFS)

Neben WMS erhält auch der WFS (Web Feature Service) immer größere Bedeutung. Die Web Feature Service-Spezifikation wurde ebenfalls durch das Open Geospatial Consortium (OGC) entwickelt und ermöglicht den Zugriff auf Geodaten nach räumlichen und nicht räumlichen (alphanumerischen) Gesichtspunkten. Der WFS-Standard wird mittlerweile von vielen Programmen unterstützt. Der UMN MapServer ist nur eines dieser Programme und unterstützt WFS als Server und Client.

Für den WFS wurden zwei Stufen der Implementierung definiert, der Basis-WFS und der transaktionale WFS (OGC WFS-T).

Der Basis-WFS implementiert den rein lesenden Zugriff auf Daten. Dagegen erweitert der transaktionale WFS (OGC WFS-T) die Schnittstelle um Schreibaufträge. Über den WFS-T können Vektorgeometrien erzeugt, aktualisiert und gelöscht werden.

Basis-WFS (lesender Zugriff auf Daten des WFS)

- GetCapabilities (liefert Metadaten zum WFS als XML)
- DescribeFeatureType (Beschreibung der Struktur der einzelnen Feature Types)
- GetFeature (Anforderung von Feature Types, Rückgabe als GML)

Transactional WFS (WFS-T, schreibender Zugriff)

- bietet erweiternd zu den Basis-WFS-Operationen auch schreibenden Zugriff
- Transaction (Editieren von Daten – create, update, delete)
- LockFeature, GetFeatureWithLock

Ein OGC Web Feature Server (WFS) wird wie der OGC WMS über eine URL angesprochen, an die weitere Parameter angehängt werden. Während der WMS allerdings direkt Kartenbilder zurückgibt, liefert der WFS die Geodaten selbst in Form einer XML-Datei aus. Anders als die Ergebnisse eines WMS, die Sie direkt als Karte sehen können, liefert der WFS (Feature) also kein Bild (Map), sondern lediglich die Daten (Features). Die Weiterverarbeitung der Informationen erfolgt in dem OGC WFS-Client. Das XML-Format der Daten, die der WFS liefert, ist GML (Geography Markup Language), das ebenfalls durch das OGC standardisiert ist und derzeit in den Versionen 2 und 3 vorliegt.

Das Ergebnis eines OGC WFS muss vom Client weiterverarbeitet werden, bevor der Benutzer etwas damit anfangen kann. Der WMS dagegen liefert direkt ein fertiges Bild zurück. Wenn Sie einen WMS in Ihrer Mapdatei kaskadierend ansprechen, wird das Bild einfach zusammen mit Ihren anderen Daten angezeigt. Das Ergebnis eines WFS verlangt dem Client mehr Arbeit ab, bietet aber auch größere Flexibilität, weil Sie die Daten so weiterverarbeiten und auswerten können, wie Sie möchten.

Einsatzgebiete des WFS

Der WFS kann anders als der WMS für sehr unterschiedliche Zwecke verwendet werden. Hierzu zählen das Suchen nach Objekten (z.B. nach dem Namen einer Stadt), das Zoomen und Markieren der Ergebnisse (z.B. Highlight der Stadtgrenze), aber auch die gezielte Abfrage sowie Analyse, Weiterverarbeitung, Export und Import von Geometrien. Die WFS-T-Schnittstelle erlaubt professionellen Softwareprodukten, auch Daten auf dem Server zu verändern – natürlich nur, wenn der Server dies zulässt. So können ganze Verarbeitungsketten zusammengesetzt werden, die über die OGC WMS-Schnittstelle Karten darstellen sowie über WFS-T Daten verändern und analysieren können. Durch die Offenheit und starke Standardisierung der Kommunikationsschnittstellen ist es möglich, diese Architektur in die meisten bestehenden IT-Systemlandschaften zu integrieren.

Aufruf des WFS Capabilities-Dokuments

Wie bei WMS können auch die Fähigkeiten eines WFS-Servers mit einer URL-Abfrage überprüft werden. Hier ein einfaches Beispiel:

```
http://map.ns.ec.gc.ca/envdat/map.aspx?service=WFS&version=1.0.0&request=GetCapabilities
```

Schicken Sie den Request direkt im Browser ab oder speichern Sie das Ergebnis des WFS über `wget` in eine Datei:

```
>wget -O test.xml "http://map.ns.ec.gc.ca/envdat/map.aspx?service=WFS&version=1.0.0&request=getCapabilities"
```

Das Ergebnis des Aufrufs ist wie beim WMS ein XML-Dokument. Einen Ausschnitt des Capabilities-Dokuments sehen Sie in Beispiel 12-7.

Beispiel 12-7: Beispiel für ein WFS-Capabilities-Dokument

```
...
<Service>
  <Name>MapServer WFS</Name>
  <Title>ENVIRODAT - Atlantic Region Water Quality Chemistry Database</Title>
  <Abstract>ENVIRODAT is a repository of water quality information including chemical,
physical, biological, and selected hydrometric data which are stored for surface,
groundwater, wastewater, precipitation and various other water types</Abstract>
  ...
  <OnlineResource>http://map.ns.ec.gc.ca/envdat/map.aspx?</OnlineResource>
  ...
  <FeatureType>
    <Name>envirodat</Name>
    <Title>ENVIRODAT - Atlantic Region Water Quality Chemistry Database</Title>
    ...
    <SRS>EPSG:4326</SRS>
    <LatLongBoundingBox minx="-64.6622" miny="46.7594" maxx="-52.6808" maxy="55.2333" />
  </FeatureType>
```

Der WFS- DescribeFeatureType-Request

Um im nächsten Schritt Informationen über die einzelnen FeatureTypes zu bekommen, benötigen wir die OnlineResource und den Name des jeweiligen FeatureType.

Damit Sie erfahren, welche Informationen über einen FeatureType zurückgegeben werden, schicken Sie den DescribeFeatureType-Request ab. Über den folgenden Aufruf lassen Sie sich das Schema zum FeatureType `envirodat` ausgeben:

```
http://map.ns.ec.gc.ca/envdat/map.aspx?service=WFS&version=1.0.0&request=DescribeFeatureType&typename=envirodat
```

Der Aufruf liefert das folgende Schema zurück. Über das Schema erfahren Sie, wie die Daten aufgebaut sind. Für Sie ist wichtig, welche Spalten zur Verfügung stehen. Die Spalten werden jeweils über `element` angegeben. Diese verfügen über einen Namen, den Sie später verwenden können, um beispielsweise die Daten zu filtern oder um das Attribut für die Beschriftung anzugeben.


```

<schema targetNamespace="http://www.ec.gc.ca/envirodat/atlantic" elementFormDefault=
    "qualified" version="0.1">
<import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.
    net/gml/2.1.2/feature.xsd"/>
<element name="envirodat" type="envdat:envirodatType" substitutionGroup="gml:_Feature"/>
<complexType name="envirodatType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="msGeometry" type="gml:GeometryPropertyType" minOccurs="0"
            maxOccurs="1"/>

        <element name="Station_ID" type="string"/>
        <element name="lat" type="string"/>
        <element name="url_station_data" type="string"/>
        <element name="lon" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
</schema>

```

WFS-Datenanfragen über den GetFeature-Request

Im nächsten Schritt sollen Daten vom WFS manuell über den getFeature-Aufruf angefordert werden. Die Daten werden vom WFS als GML (Geography Markup Language) ausgegeben. GML ist ein weiterer Standard, der vom OGC definiert wurde, er liefert die Daten als XML zurück. Während WMS Ihnen direkt ein Kartenbild liefert, erhalten Sie beim WFS Daten, die dann zur weiteren Verarbeitung zur Verfügung stehen.

Das folgende Beispiel zeigt, wie Sie die Daten mit dem Befehl `wget` anfordern und in einer Datei speichern können.

```
> wget -O wfs_data.gml "http://map.ns.ec.gc.ca/envdat/map.aspx?service=WFS&version=
1.0.0&request=GetFeature&typename=envirodat"
```



Schauen Sie sich das GML in einem Editor an. Sie können sich auch über `ogrinfo` eine Zusammenfassung der Daten ausgeben lassen. Rufen Sie dazu `ogrinfo` wie folgt auf:

```
> ogrinfo wfs_data.gml envdat:envirodat -so
```

Sie erhalten eine Auflistung der Attributspalten und können darüber entscheiden, welches Attribut Sie beispielsweise für die Beschriftung verwenden wollen. Lassen Sie die Optionen `-so` weg, werden sämtliche Informationen ausgegeben.

UMN MapServer als WFS-Client

Der WFS kann als Datenquelle in einer Mapdatei verwendet werden. In Beispiel 12-8 wird das Einbinden eines externen WFS in eine Mapdatei demonstriert. Die wichtigsten Stellen wurden im Beispiel hervorgehoben. Wie bei WMS wurden auch hier die Parameter, die für den WFS angegeben werden müssen, im METADATA-Objekt definiert. In diesem Beispiel werden aber nur die notwendigen Parameter angegeben.

Beispiel 12-8: Beispiel für einen Layer in der Mapdatei, der auf eine externe WFS-Datenquelle zugreift

```
...
EXTENT -64.6622 46.7594 -52.6808 55.2333
...
LAYER
  NAME wfs_test
  STATUS ON
  TYPE POINT
  CONNECTIONTYPE WFS
  CONNECTION "http://map.ns.ec.gc.ca/envdat/map.aspx?"
  METADATA
    "wfs_version" "1.0.0"
    "wfs_srs" "EPSG:4326"
    "wfs_typename" "envirodat"
  END
  LABELITEM "envdat:Station_ID"
  CLASS
    STYLE
      SYMBOL 'circle'
      SIZE 10
      OUTLINECOLOR 150 150 0
      COLOR 250 250 0
    END
    LABEL
      SIZE SMALL
      COLOR 255 255 255
      POSITION UR
    END
  END
END
```

In Abbildung 12-4 ist zu sehen, wie die Weltdaten um einen Layer mit Messpunkten zur Wasserqualität in Neufundland (Kanada) erweitert wurden. Die Angaben des Parameters EXTENT wurden so angepasst, dass die Bounding Box zur WFS-Datenquelle passt. Die Information zur Bounding Box der Daten können Sie über das GML der Daten ableiten. Hier wird die Bounding-Box ausgegeben. Beispiel 12-15 zeigt die vollständige Mapdatei für den WFS-Client.

In der Mapdatei wird ein Symbol circle über die Angabe eines Symbol-Objekts definiert. Dieses Symbol wird dann im WFS-Layer zur Darstellung der Punkte verwendet. In der Mapdatei wird der Layer countries zuerst angegeben und daher auch an unterster Stelle in der Grafik ausgegeben.

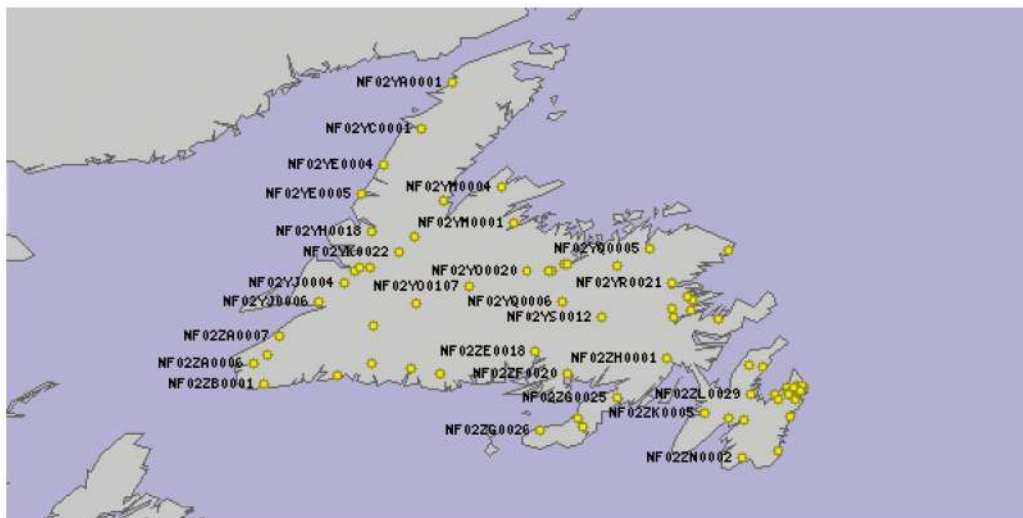


Abbildung 12-4: Eine Karte, die anhand von Daten aus einer WFS-Datenquelle die Position von Messstationen für die Wasserqualität anzeigt

Abbildung 12-5 zeigt eine Karte, in der der WMS-Layer `global_mosaic`, den wir bereits kennen, in Kombination mit der WFS-Ebene und einigen Informationen zu lokalen Transportwegen kombiniert wurde. Dieses Beispiel soll zeigen, wie einfach es ist, verschiedene Datenquellen in einer Karte zusammenzuführen.

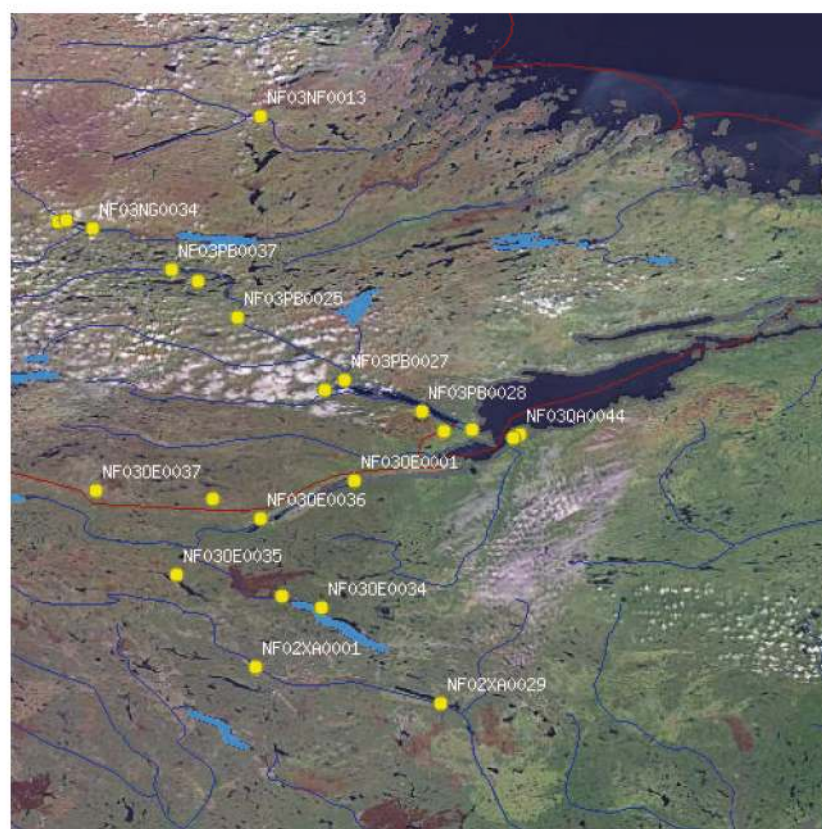


Abbildung 12-5: Eine aus WFS-, WMS- und lokalen Shapefile-Ebenen erstellte Karte

UMN MapServer als WFS-Server

MapServer kann als WFS-Server genutzt werden. Dies ist eine einfache Möglichkeit, um Vektordaten anderen zugänglich zu machen. Hierzu muss die Mapdatei als WFS konfiguriert werden. Dazu sind Angaben im MAP-Objekt (allgemeine Angaben zum WFS) und Angaben in den LAYER-Objekten notwendig. Eine ausführliche Beschreibung zu WFS und MapServer finden Sie unter:

http://mapserver.gis.umn.edu/docs/howto/wfs_server

Im MAP-Objekt müssen die METADATA-Angaben erweitert werden. Beispiel 12-9 zeigt das WEB-Objekt der Mapdatei mit den zusätzlichen WFS-Daten im METADATA-Teil.

Beispiel 12-9: Angaben für den WFS-Server erfolgen im METADATA-Objekt innerhalb des WEB-Objekts

```
WEB
  IMAGEPATH "/srv/www/tmp/"
  IMAGEURL  "/tmp/"
  METADATA
    "wfs_title" "My Global WFS Map Service"  #notwendig
    "wfs_onlineresource" "http://localhost/cgi-bin/mapserv?
                        map=/srv/www/htdocs/globalwfs.map"
    "wfs_srs" "epsg:4326"
  END
END
```

Dieses Beispiel zeigt die notwendigen und empfohlenen Einstellungen. Es wurde der `wfs_title`-Parameter eingefügt, der den Titel des WFS-Servers definiert. Daneben gibt es noch eine Reihe weiterer Parameter, aus denen Sie wählen können. Die Angaben zu `EXTENT` beziehen sich auf alle WFS-Anfragen. In diesem Beispiel deckt die Karte den gesamten Globus ab (`EXTENT -180 -90 180 90`). Wenn Sie die `EXTENT`-Definitionen des vorangegangenen Beispiels beibehalten hätten, wären nur die Objekte, die sich in diesem Bereich befinden, zurückgegeben worden.

Es gibt einen »MapServer OGC Web Services Workshop«, der Ihnen weitere Informationen und Mapdatei-Beispiele für den Einsatz von MapServer als OGC WMS und WFS liefert. Den Workshop finden Sie unter <http://ms-ogc-workshop.maptools.org/>.

Zusätzliche Informationen zu den verschiedenen Parametern gibt es in der MapServer-Dokumentation zur WFS-Unterstützung unter <http://mapserver.gis.umn.edu/doc/wfs-server-howto.html>.

In den LAYER-Objekten müssen ebenfalls Anpassungen erfolgen, und zwar nur in den Layern, die Sie per WFS zur Verfügung stellen möchten. Beachten Sie, dass WFS lediglich Vektordaten zur Verfügung stellen kann. GML unterstützt nur vektorbasierte Informationen und keine Rasterdaten. Für die Bereitstellung von Rasterdaten sollten Sie stattdessen WMS oder WCS verwenden (siehe Tabelle 12-1).

Beispiel 12-10 zeigt die Ebene `countries` mit den Änderungen, d.h. den für den WFS-Zugriff notwendigen und empfohlenen Parametern. Die vollständige Mapdatei zu diesem Beispiel finden Sie in Beispiel 12-16 am Ende dieses Kapitels. Über `gml_featureid` müssen

Sie eine Attributspalte angeben, die als ID in GML ausgegeben werden soll. Außerdem müssen Sie angeben, welche Attributspalten Sie über den WFS bereitstellen möchten. Solange Sie keine Angaben machen, werden keine Spalten ausgegeben. Die Angaben können über die Parameter `gml_include_items` sowie `gml_exclude_items` gemacht werden. Über den Parameter `wfs_maxfeatures` bestimmen Sie, wie viele Objekte Ihres FeatureType maximal im Ergebnis ausgegeben werden sollen.

Beispiel 12-10: Bereitstellen der Informationen einer Shape-Datei über WFS

```
LAYER
  NAME countries
  TYPE POLYGON
  STATUS ON
  DUMP TRUE # notwendig
  DATA countries_simpl
  METADATA
    "wfs_title" "Country"
    "wfs_featureid" "gid" # notwendig
    "gml_include_items" "all" # empfohlen
    "gml_exclude_items" "cat,fibs" # Spalten nicht ausgeben
    "wfs_maxfeatures" 2000 # max. Ausgabe an Objekten
  END
  PROJECTION
    "init=epsg:4326"
  END
END
```

In diesem Beispiel haben wir sämtliche CLASS-Objekte und einige andere Einstellungen wieder entfernt. Wenn Sie Ihre Daten nur per WFS bereitstellen möchten, werden diese Informationen nicht gebraucht, da die Daten vom WFS nicht gerendert werden und daher keine Angaben zum Stil notwendig sind. Wenn Sie möchten, können Sie eine Mapdatei aber auch mit den WMS- und den WFS-Angaben gleichzeitig versehen und dann entweder als WMS oder WFS nutzen.

Um Daten als GML auszugeben, muss im Layer der Parameter `DUMP` auf `TRUE` gesetzt sein. Mit dieser Einstellung erlauben Sie MapServer, die Daten auf Anfrage auszugeben. Für Ebenen, die explizit nur lokal benutzt werden sollen, können Sie den Wert auf `FALSE` setzen. Der Default für `DUMP` ist `FALSE`.

Prüfen des WFS Capabilities-Dokuments

Ihren WFS können Sie über den `GetCapabilities`-Aufruf prüfen. Schicken Sie diesen einfach im Browser ab. Beispiel 12-11 zeigt das Anfordern des Capabilities-Dokuments über `wget`. Im `GetCapabilities`-Aufruf müssen Sie die `OnlineResource` Ihres Diensts sowie die Parameter `Service`, `Request` und `Version` angeben. Im Beispiel sehen Sie ebenfalls Auszüge aus dem Capabilities-Dokument.

Beispiel 12-11: Anfordern des Capabilities-Dokuments zum WFS über wget sowie Anzeige der wichtigsten Informationen aus dem Dokument

```
> wget -O wfscapabilities.xml "http://localhost/cgi-bin/mapserv?map=/srv/www/htdocs/globalwfs.map&service=WFS&version=1.0.0&request=GetCapabilities"
```

```
...
<Service>
  <Name>MapServer WFS</Name>
  <Title>My Global WFS Map Service</Title>
  <OnlineResource>http://localhost/cgi-bin/mapserv?map=/srv/www/htdocs/globalwfs.map&</OnlineResource>
</Service>
...
<FeatureTypeList>
  <Operations>
    <Query/>
  </Operations>
  <FeatureType>
    <Name>countries</Name>
    <Title>Country Boundaries</Title>
    <SRS>EPSG:4326</SRS>
    <LatLongBoundingBox minx="-180" miny="-89.9999" maxx="180" maxy="83.6274" />
  </FeatureType>
</FeatureTypeList>
...
```

Aus den Angaben der Mapdatei generiert MapServer das Capabilities-Dokument. Hier finden sich allgemeine Angaben zum Server sowie genauere zu den FeatureTypes. Für jeden Layer in der Mapdatei, der über die WFS-Parameter verfügt, wird ein Feature-Type ausgegeben. Hier ist der Name des FeatureType von Interesse, da über den Namen die Beschreibung der Daten und die Daten selbst angefordert werden können.

Anfordern von Daten über GetFeature

Die Daten werden über den GetFeature-Aufruf angefordert. Geben Sie Name über die Angabe &typename=countries zusätzlich im Request an. Den vollständigen getFeature-Request und einen Teil der Ausgabe sehen Sie in Beispiel 12-12.

Beispiel 12-12: Anfordern von Daten über GetFeature von einem WFS

```
> wget -O countries.gml "http://localhost/cgi-bin/mapserv?map=/srv/www/htdocs/globalwfs.map&service=WFS&version=1.0.0&request=GetFeature&typename=countries"
```

```
...
<gml:featureMember>
  <countries>
    <gml:boundedBy>
      <gml:Box srsName="EPSG:4326">
        <gml:coordinates>
          -57.336025,54.494965 -57.245407,54.577198
        </gml:coordinates>
      </gml:Box>
    
```


Beispiel 12-12: Anfordern von Daten über GetFeature von einem WFS (Fortsetzung)

```
</gml:boundedBy>
<gml:polygonProperty>
  <gml:Polygon srsName="EPSG:4326">
    <gml:outerBoundaryIs>
      <gml:LinearRing>
        <gml:coordinates>
          -57.336025,54.494965 -57.330402,54.563805
          -57.245407,54.577198 -57.250492,54.506748
          -57.336025,54.494965
        </gml:coordinates>
      </gml:LinearRing>
    </gml:outerBoundaryIs>
  </gml:Polygon>
</gml:polygonProperty>
<gid>36354</gid>
<cat>38</cat>
<fibs>CA</fibs>
<name>Canada</name>
<f_code>FA001</f_code>
<total>32507874</total>
<male>16071958</male>
<female>16435916</female>
<ratio>97.799999999999997</ratio>
</countries>
</gml:featureMember>
...
```

Mit dieser Angabe fordern Sie alle Daten des FeatureType an. Über die zusätzliche Angabe `&maxfeatures=20` können Sie die Ergebnismenge einschränken.

Definitionen von Filtern zur gezielten Ausgabe von Informationen

Sie können auch einen Filter definieren, um das Ergebnis einzuschränken. Der Filter kann in dem Aufruf über den Parameter `FILTER` übergeben werden. Der Aufbau der Filterangabe richtet sich nach der *OpenGIS Filter Encoding Implementation Specification*. Sie lernen also an dieser Stelle einen weiteren OGC-Standard kennen.

Über den Einsatz von Filtern haben Sie zahlreiche Möglichkeiten, die Daten abzufragen. Sie können dabei je nach den Möglichkeiten, die der Server zur Verfügung stellt, Vergleichsoperatoren und logische Operatoren sowie räumliche Funktionen nutzen und kombinieren.

Ein Aufruf, der nur die Polygone der Schweiz ausgibt, sieht wie folgt aus:

```
http://localhost/cgi-bin/mapserv?map=/srv/www/htdocs/globalwfs.map&Service=
WFS&VERSION=1.0.0&Request=getFeature& Typename=countries&FILTER=<Filter>
<PropertyIsEqualTo><PropertyName>NAME</PropertyName><Literal>Switzerland</Literal>
</PropertyIsEqualTo></Filter>
```

Einsatz von WFS

Ein WFS kann von verschiedenen Clients geladen werden. In Abbildung 12-6 überlagern die Polygone des WFS eine Weltkarte und wurden wiederum mit Wolken bedeckt. Die Ergebnisse eines WFS können aber unter anderem auch für Analysen, Suchmodule oder Navigationen genutzt werden. Wenn Sie einen WFS zum Digitalisieren von neuen

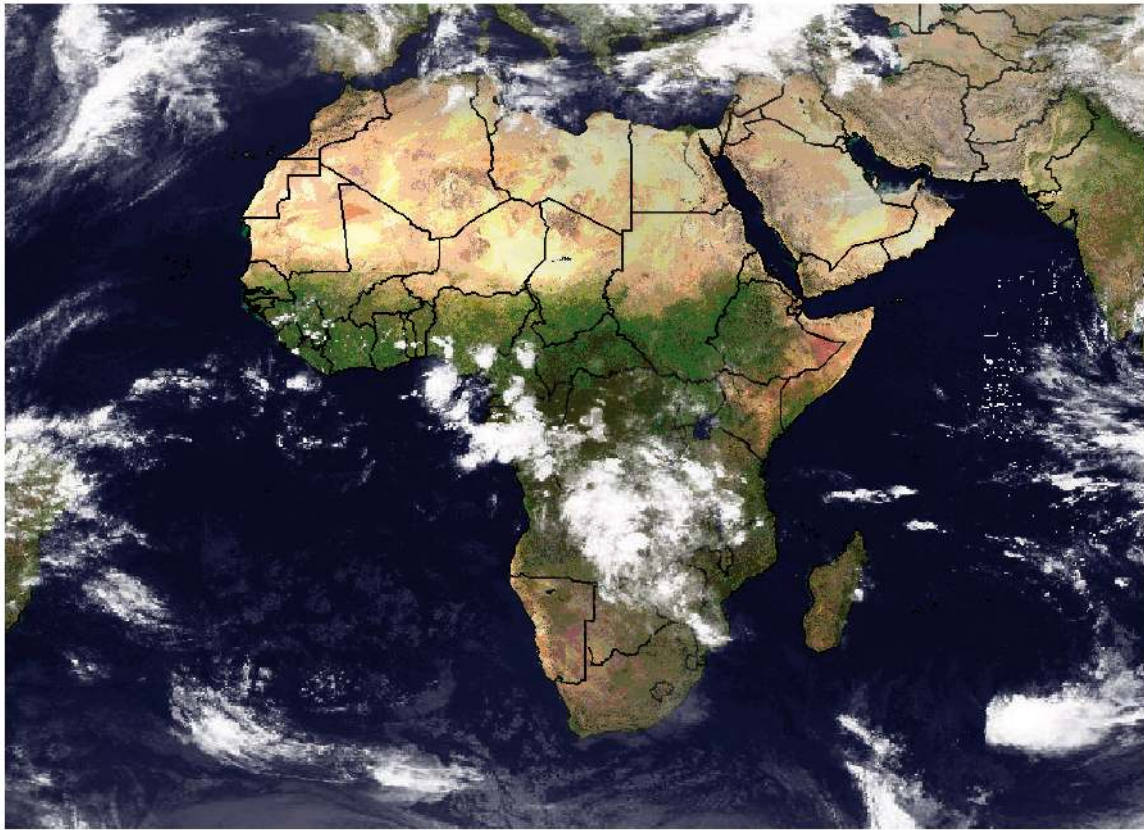


Abbildung 12-6: Unter Verwendung der vom WFS-Server heruntergeladenen GML-Datei und weiteren Bilddaten in OpenEV erstellte Beispielkarte

Objekten nutzen möchten, können Sie diese Aufgabe nicht über den UMN MapServer lösen, da MapServer nur Basis-WFS unterstützt, den schreibenden Zugriff auf Objekte also nicht ermöglicht. Hierzu können Sie GeoServer (<http://geoserver.org/>) oder deegree (<http://www.deegree.org/>) einsetzen. Beide Anwendungen unterstützen WFS-T und darüber den schreibenden Zugriff auf Objekte. Schauen Sie sich für weitere Informationen die jeweiligen Projektseiten an.

Mapdateien für den Einsatz als WMS und WFS

Dieser Abschnitt enthält die vollständigen in diesem Kapitel verwendeten Mapdateien (Beispiel 12-13 bis 12-16). Die wichtigen Bereiche wurden hervorgehoben. Auskommentierte Zeilen werden durch die Raute (#) gekennzeichnet.

Beispiel 12-13: UMN MapServer als WMS-Client

```
MAP
  NAME global
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250

  OUTPUTFORMAT
    NAME png
    DRIVER "GD/PNG"
    MIMETYPE "image/png"
    IMAGEMODE RGB
    EXTENSION "png"
  END

  UNITS DD

  WEB
    TEMPLATE global.html
    IMAGEPATH "/srv/www/htdocs/tmp/"
    IMAGEURL "/tmp/"
  END

  LAYER
    NAME global_landsat
    TYPE RASTER
    STATUS DEFAULT
    CONNECTIONTYPE WMS
    CONNECTION "http://wms.jpl.nasa.gov/wms.cgi?"
    MINSCALEDENOM 20000
    MAXSCALEDENOM 0
    METADATA
      "wms_server_version" "1.1.1"
      "wms_srs" "EPSG:4326"
      "wms_format" "image/jpeg"
      "wms_styles" ""
      "wms_name" "modis,global_mosaic"
    END
  END

  LAYER
    NAME countries
    TYPE POLYGON
    STATUS ON
    DATA countries_simpl
    LABELITEM 'NAME'
    LABELMAXSCALEDENOM 50000000
    CLASSITEM 'NAME'
    CLASS
      NAME 'Bulgaria'
      EXPRESSION 'Bulgaria'
      STYLE
        OUTLINECOLOR 100 100 100
```


Beispiel 12-13: UMN MapServer als WMS-Client (Fortsetzung)

```
        #COLOR 255 255 150
    END
    LABEL
        SIZE LARGE
    END
END
END
CLASS
    NAME 'All Countries'
    EXPRESSION ('[NAME]' ne 'Bulgaria')
    STYLE
        OUTLINECOLOR 100 100 100
        #COLOR 200 200 200
    END
END
END
END
```

Beispiel 12-14: Erweiterung der Mapdatei, um einen WMS bereitzustellen

```
MAP
    NAME global
    SIZE 600 300
    EXTENT -180 -90 180 90
    IMAGECOLOR 180 180 250

    OUTPUTFORMAT
        NAME png
        DRIVER "GD/PNG"
        MIMETYPE "image/png"
        IMAGEMODE RGB
        EXTENSION "png"
    END

    UNITS DD

    WEB

        IMAGEPATH "/srv/www/htdocs/tmp/"
        IMAGEURL "/tmp/"
        METADATA
            "wms_title" "My Global Map WMS Server"
            "wms_onlineresource" "http://localhost/cgi-bin/mapserv?
                                map=/srv/www/htdocs/globalwms.map"
            "wms_srs" "EPSG:4326"
        END
    END

    LAYER
        NAME countries
        TYPE POLYGON
        STATUS ON
        DATA countries_simpl
        METADATA
```

Beispiel 12-14: Erweiterung der Mapdatei, um einen WMS bereitzustellen (Fortsetzung)

```
    "wms_title" "Country boundaries"
END
MINSCALEDENOM 1000
MAXSCALEDENOM 1000000000
CLASSITEM 'NAME'
CLASS
    NAME 'Bulgaria'
    EXPRESSION 'Bulgaria'
    STYLE
        OUTLINECOLOR 100 100 100
    END
END
CLASS
    NAME 'All Countries'
    EXPRESSION ('[NAME]' ne 'Bulgaria')
    STYLE
        OUTLINECOLOR 100 100 100
    END
END
END
END
```

Beispiel 12-15: Beispiel einer Mapdatei für die Verwendung einer externen WFS-Datenquelle

```
MAP
    NAME global
    SIZE 600 300
    EXTENT -58 46 -54 52
    IMAGECOLOR 180 180 250

    UNITS DD

    WEB
        TEMPLATE global.html
        IMAGEPATH "/srv/www/htdocs/tmp/"
        IMAGEURL "/tmp/"
        METADATA
            "wms_title" "MapServer als WFS Client"
            "wms_onlineresource" "http://localhost/cgi-bin/mapserv?
                                map=/srv/www/htdocs/globalwfsclient.map"
        END
    END

    PROJECTION
        "init=epsg:4326"
    END

    SYMBOL
        NAME "circle"
        TYPE ellipse
        FILLED true
        POINTS
            1 1
```

*Beispiel 12-15: Beispiel einer Mapdatei für die Verwendung einer externen WFS-Datenquelle
(Fortsetzung)*

```
END
END

LAYER
  NAME countries
  TYPE POLYGON
  STATUS ON
  DATA countries_simpl
  METADATA
    "wms_title" "countries"
    "wfs_srs" "EPSG:4326"
  END
  PROJECTION
    "init=epsg:4326"
  END
  MINSCALEDENOM 1000
  MAXSCALEDENOM 1000000000
  CLASSITEM 'NAME'
  CLASS
    NAME 'Bulgaria'
    EXPRESSION 'Bulgaria'
    STYLE
      OUTLINECOLOR 100 100 100
      COLOR 255 255 150
    END
  END
  CLASS
    NAME 'All Countries'
    EXPRESSION ('[NAME]' ne 'Bulgaria')
    STYLE
      OUTLINECOLOR 100 100 100
      COLOR 200 200 200
    END
  END
END

LAYER
  NAME wfs_test
  STATUS ON
  TYPE POINT
  CONNECTIONTYPE WFS
  CONNECTION "http://map.ns.ec.gc.ca/envdat/map.aspx?"
  METADATA
    "wms_title" "envdat"
    "wms_srs" "EPSG:4326"
    "wfs_srs" "EPSG:4326"
    "wfs_version" "1.0.0"
    "wfs_typedname" "envirodat"
    "wfs_request_method" "GET"
    "wfs_service" "WFS"
  END
```


Beispiel 12-15: Beispiel einer Mapdatei für die Verwendung einer externen WFS-Datenquelle (Fortsetzung)

```
PROJECTION
  "init=epsg:4326"
END
LABELITEM "envdat:Station_ID"
CLASS
  STYLE
    SYMBOL 'circle'
    SIZE 6
    OUTLINECOLOR 150 150 0
    COLOR 255 250 0
  END
  LABEL
    SIZE TINY
    COLOR 0 0 0
    POSITION CL
  END
END
END
END
```

Beispiel 12-16: Einfaches Beispiel für die Bereitstellung des Layers countries per WFS

```
MAP
  NAME global
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250

  UNITS DD

  WEB
    TEMPLATE global.html
    IMAGEPATH "/srv/www/htdocs/tmp/"
    IMAGEURL "/tmp/"
    METADATA
      "wfs_title" "My Global WFS Map Service"
      "wfs_onlineresource" "http://localhost/cgi-bin/mapserv?
                           map=/srv/www/htdocs/globalwfsserver.map"
    END
  END

  PROJECTION
    "init=epsg:4326"
  END

  LAYER
    NAME countries
    TYPE POLYGON
    STATUS ON
    DUMP TRUE
    DATA countries_simpl
    METADATA
```

Beispiel 12-16: Einfaches Beispiel für die Bereitstellung des Layers countries per WFS (Fortsetzung)

```
    "wfs_title" "countries"
    "gml_featureid" "gid"
    "gml_include_items" "all"
  END
  PROJECTION
    "init=epsg:4326"
  END
END
END
```

Kartendienste nutzen und verwalten

Dieses Kapitel zeigt, wie Kartendienste von unterschiedlichen Servern in Clientumgebungen eingebunden werden können, z.B. um sie in Webseiten zu integrieren. Der Schwerpunkt liegt auf der Nutzung standardisierter Kartendienste, denn eine wachsende Zahl von Anbietern unterstützen diese Schnittstellen. Aus der Vielzahl verfügbarer Open Source-Projekte wird an den Beispielen der OSGeo-Projekte OpenLayers und Mapbender der Umgang mit OGC WMS und WFS aus zwei Perspektiven erklärt.

OpenLayers bildet als reine JavaScript-Bibliothek das eine Ende des Anwendungsspektrums ab und bietet eine sehr einfache und elegante Möglichkeit, Karten in Webseiten zu integrieren. Mapbender dagegen ist eine vollständige Entwicklungsumgebung zur Verwaltung großer Geodatenportale mit Metadatenverwaltung und Digitalisierungsmodulen. Er bildet das andere Ende des Anwendungsspektrums ab und beinhaltet Serverkomponenten zur Authentifizierung und Autorisierung, die für den sicheren Betrieb und die Orchestrierung von OGC-Architekturen benötigt werden.

Verschiedene Clients für Kartendienste

Es gibt eine Vielzahl von unterschiedlichen Clients, die OGC WMS-Kartendienste ansprechen können. Die folgende Übersicht teilt einige der bekannteren Softwarepakete in drei verschiedene Typen ein, die allerdings nicht als scharf abgegrenzte Klassen zu verstehen sind, sondern als unterschiedliche technische Ansätze, die sich oft überschneiden. Diese Liste erhebt auch keinen Anspruch auf Vollständigkeit, sondern bietet lediglich eine Übersicht der bekannteren Open Source-Clients. Es gibt eine Fülle weiterer Softwareprojekte, die OGC WMS-Kartendienste einbinden können. Eine sehr umfangreiche Liste finden Sie auf der Plattform <http://freegis.org>.

Desktop-GIS

Zu den Eigenschaften eines Desktop-GIS zählt unter anderem, dass es üblicherweise lokal auf einem Rechner installiert und ausgeführt wird sowie GIS-Funktionalität für den professionellen Anwender bietet. Zu den klassischen Desktop Free- und Open

Source-GIS gehört z.B. Quantum GIS (QGIS), das in den Kapiteln 8 und 9 eingehend beschrieben wurde. Quantum GIS bietet durch die Möglichkeit, GRASS direkt zu integrieren, ein enormes zusätzliches Potenzial an Funktionalität.

Hier folgt eine Auswahl von Desktop-GIS, die OGC WMS-Kartendienste unterstützen:

- GRASS (<http://www.osgeo.org/grass>)
Sprache: ANSI C, C++; Lizenz: GNU GPL
- gvSIG (<http://www.gvsig.gva.es/>)
Sprache: Java; Lizenz: GNU GPL
- OpenEV (<http://openev.sourceforge.net/>)
Sprache: Python; Lizenz: GNU LGPL
- OpenJUMP (<http://www.openjump.org/>)
Sprache: Java; Lizenz: GNU GPL
- Quantum GIS (<http://www.osgeo.org/qgis>)
Sprache: C++, Python und Java Bindings; Lizenz: GNU GPL
- uDig (<http://udig.refractions.net/>)
Sprache: Java; Lizenz: GNU LGPL

JavaScript-Clients

JavaScript-Clients bilden das andere Extrem der technologischen Bandbreite und nutzen ausschließlich die Funktionalität von Internetbrowsern, um Kartendienste anzuzeigen und Navigationsmöglichkeiten zu bieten. Durch die Beschränkung auf die Sprache JavaScript und deren Interpretationsspielraum in unterschiedlichen Browsern steht gegenüber Desktop-GIS nur ein eingeschränkter Funktionsumfang zur Verfügung. Die Software kann dafür aber ohne lokale Installation von jedem Rechner genutzt werden, der an das gleiche Netzwerk (Intra- oder Internet) angebunden ist und über einen Webbrowser verfügt.

Eine Anwendung, die im Internet bereitgestellt wird, ist also immer von der ganzen Welt nutzbar. JavaScript-Clients werden serverseitig von Administratoren konfiguriert und über einen Webserver bereitgestellt. Sie bieten sich sowohl für die einfache Integration von Kartenwerken in Webseiten an als auch als Auskunftssystem für große Anwendergruppen. Die Software-Community MapBuilder ist kein reines JavaScript-Projekt, da auch serverseitige Skripten genutzt werden können.

- Community MapBuilder (<http://www.osgeo.org/mapbuilder>)
Sprache: JavaScript, CSS, XSLT; Lizenz: GNU/LGPL
- OpenLayers (<http://www.osgeo.org/openlayers>)
Sprache: JavaScript; Lizenz: BSD

OpenLayers, das als reine Ajax-Anwendung (Asynchronous JavaScript and XML) implementiert wird und Anleihen aus der sogenannten »Web 2.0«-Welt übernimmt, wird in diesem Kapitel näher beschrieben. OpenLayers ist die derzeit aktuellste reine JavaScript-Bibliothek, die vollständig für die Ajax-Technologie ausgelegt ist.

Hybride WebGIS-Clients

Die meisten WebGIS-Anwendungen nutzen einen hybriden Ansatz, also JavaScript für die Benutzeroberflächen und serverseitige Programmmodule, die weitergehende GIS-Funktionen nutzen können. Neben reinen GIS-Funktionen zählt dazu auch eine Benutzerverwaltung sowie die Möglichkeit, Kartendienste gegen unberechtigten Zugriff zu sichern. Als Benutzeroberfläche werden dynamische HTML-Seiten erzeugt, die mit herkömmlichen Webbrowsern angezeigt werden können. Die meisten aktuellen Lösungen setzen dabei auf asynchrone Ausführung und unterstützen ebenfalls Ajax.

- CartoWeb (<http://www.cartoweb.org/>)
Sprache: PHP; Lizenz GNU GPL
- deegree iGeoPortal (<http://www.deegree.org>)
Sprache: Java; Lizenz: GNU LGPL
- ka-Map (<http://ka-map.maptools.org/>)
Sprache: JavaScript, PHP; Lizenz: individuelle Open Source-Lizenz
- Mapbender (<http://www.mapbender.org>)
Sprache: PHP, JavaScript, SQL; Lizenz GNU GPL

Mapbender ist eine Entwicklungsumgebung für hybride Anwendungen, die sowohl JavaScript nutzt als auch serverseitig PHP verwendet, um weitergehende Funktionalität zu implementieren. Die Alleinstellungsmerkmale der Geoportal-Software Mapbender sind die vollständig webbasierte Konfigurierbarkeit, eine umfangreiche Benutzerverwaltung und die Möglichkeit, Kartendienste gegen unbefugten Zugriff zu sichern.

Wohin geht die Reise?

Bisher wurden umfangreiche und komplexe Fragestellungen durch die Entwicklung sogenannter Enterprise-Lösungen beantwortet. Diese definieren eigene Schnittstellen und kommunizieren durch den Austausch von Beschreibungsdokumenten. Die mit diesem Ansatz assoziierte Technologie ist SOAP, das früher als »Simple Object Access Protocol« bezeichnet, später jedoch mit »Service Oriented Architecture Protocol« übersetzt wurde. Inzwischen ist der Begriff SOAP ein eigenständig definierter Begriff, der von der XML Protocol Working Group als Empfehlung des World Wide Web Consortium gepflegt wird.

Neuerdings wird allerdings immer mehr vom *programmierbaren Netz* (engl. the programmable web) gesprochen. Dabei nutzt die Software das Internet so, wie es bisher nur Menschen nutzen konnten. Voraussetzung dafür ist die Einhaltung der Regeln des Internet, das als RESTful orientierte Architektur bezeichnet wird und auf eine einzige, einheitliche und wohldefinierte Schnittstelle setzt. Der Begriff REST steht dabei für »Representational State Transfer«. Dieser Ansatz verfolgt die Grundidee, dass gerade komplexe Systeme nur durch kreative Kombinationen einfacher Komponenten und Schnittstellen nachhaltig gelöst werden können. Die ursprüngliche Definition der OGC-Standards, vor allem die WMS-Schnittstelle, wurde als RESTful-orientierte Lösung implementiert, allerdings gab es diesen Begriff damals noch gar nicht.

Die umfangreichen Vorteile, die eine konsequent RESTful-orientierte Architektur birgt, werden in dem Buch *Web Services mit REST* von Leonard Richardson und Sam Ruby (O'Reilly) eingehend beschrieben.

OpenLayers – Karten nach Maß

Das Projekt OpenLayers entwickelt eine Ajax-Bibliothek, die es ermöglicht, Kartendienste unterschiedlicher Anbieter per Knopfdruck zu integrieren. Dazu zählen neben allen Anbietern, die den aktuellen OGC WMS-Standard unterstützen, auch große proprietäre Anbieter aus der »Web 2.0«-Welt wie Yahoo!, Google und auch Microsoft. Auf den Webseiten des Projekts findet sich eine Vielzahl von Demoanwendungen, ein Besuch lohnt sich:

<http://openlayers.org/gallery/multiple.html>

Ganz, ganz einfach

Die einfachste Methode, eine Karte in eine Webseite zu integrieren, ist die Nutzung des Viewers der OpenLayers-Projekt-Webseite. Der Link zu diesem Viewer wird einfach in ein iframe Ihrer Webseite kopiert. Ein iframe (Inlineframe) ermöglicht es, andere Webdokumente in Ihre Seite einzubinden. Um das auszuprobieren, müssen Sie lediglich einen Editor haben und folgenden Code als iframe einbinden:

```
<iframe
  src="http://openlayers.org/viewer/"
  width="450" height="300"
  scrolling="no"
  marginwidth="0" marginheight="0"
  frameborder="0">
</iframe>
```

Wenn ein Benutzer die Webseite öffnet, lädt er automatisch und ohne es zu merken den Kartenviewer von der OpenLayers-Webseite hinzu. Das Ergebnis sehen Sie in Abbildung 13-1.

Die Karte kann auch mit einem anderen Maßstab eingebunden werden oder auf ein bestimmtes Gebiet zielen. Der folgende Ausschnitt zeigt eine Region im Süden Nordrhein-Westfalens.

```
<iframe
  src="http://openlayers.org/viewer/?center=50.73,7.07&zoom=10"
  width="100%" height="200"
  scrolling="no"
  marginwidth="0" marginheight="0"
  frameborder="0">
</iframe>
```

Das Ergebnis zeigt einen anderen Ausschnitt und die Anwendung in einer anderen Größe.

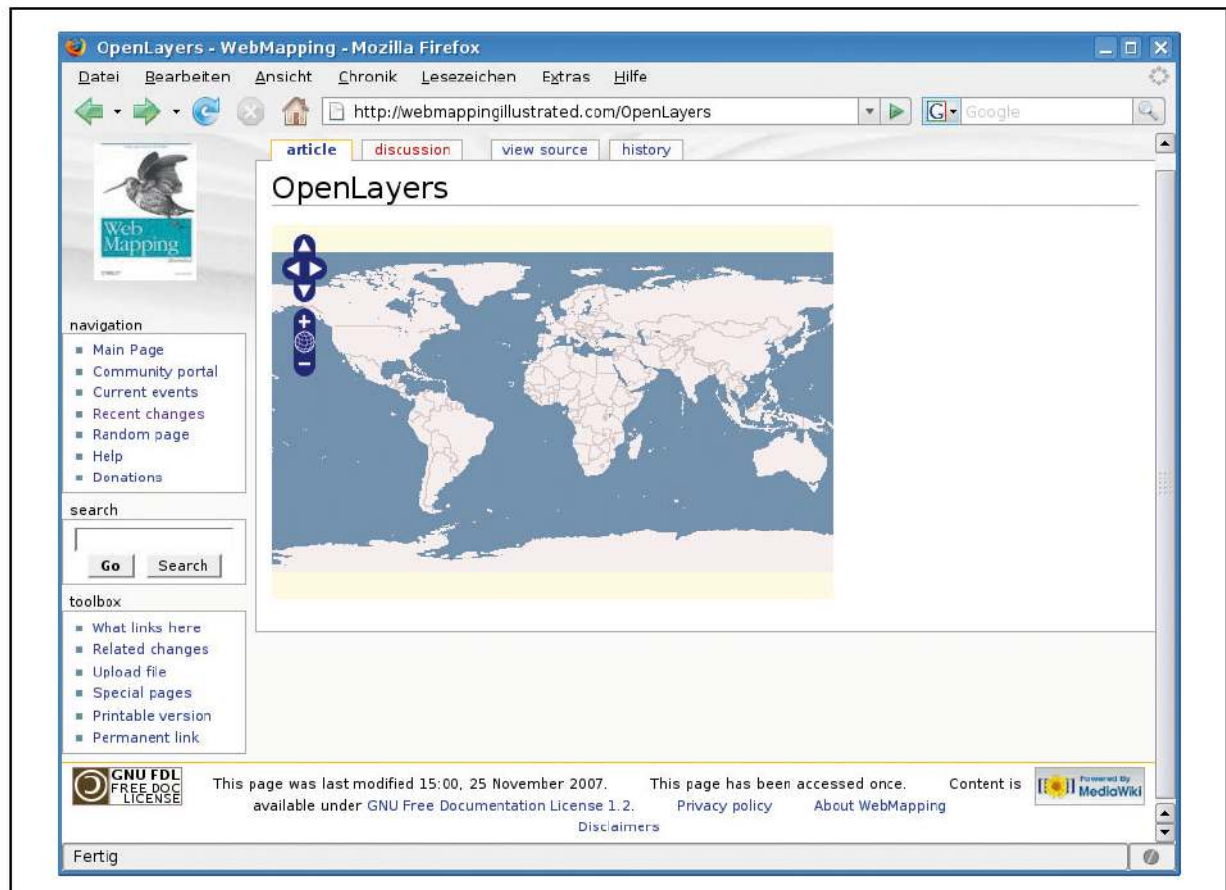


Abbildung 13-1: Einbinden des OpenLayers-Clients in Ihre Webseite

Nebenbei wurden auch noch die Parameter `width="100%"` `height="700"` geändert. Die 100% im Feld `width` beziehen sich auf die Breite des Browserfensters, die 700 im Feld `height` beziehen sich dagegen auf Pixel. Das Fenster wird also immer in der maximalen Breite genutzt und – falls möglich – 700 Pixel hoch sein. Der Parameter `zoom` ermöglicht es, zwischen den Maßstabsebenen 0 und in diesem Dienst derzeit maximal 16 zu wählen. Die Maßstabsebene 0 ist dabei der kleinste Maßstab und zeigt die ganze Welt, die Ebene 16 ist die Straßenebene, also ein großer Maßstab (siehe dazu auch die Erläuterungen zum Maßstab im Tipp in Kapitel 5 auf Seite 74).

Hinter den Kulissen

Wenn Sie sich den Quelltext dieser Seite ansehen, finden Sie dort eine ganze Reihe von OGC WMS-Aufrufen, denn OpenLayers nutzt ein Kachelungssystem, um den Bildaufbau auf viele kleinere Anfragen aufzuteilen, so wie das auch Google Maps-Anwendungen machen. Ein solcher OGC WMS-Aufruf sieht folgendermaßen aus:

```
http://labs.metacarta.com/wms-c/Basic.py?LAYERS=basic&SERVICE=WMS&VERSION=
1.1.1&REQUEST=GetMap&STYLES=&EXCEPTIONS=application%2Fvnd.ogc.se_text&FORMAT=
image%2Fjpeg&SRS=EPSG%3A4326&BBOX=7.03125%2C50.625%2C7.207031%2C50.800781&WIDTH=
256&HEIGHT=256
```


Wenn Sie diesen Aufruf einzeln in ein Browserfenster kopieren, erhalten Sie genau diesen Kartenausschnitt als Bild zurückgeliefert. So einfach macht der OGC WMS-Standard das Arbeiten mit Karten. Das Ergebnis des Aufrufs sehen Sie in Abbildung 13-2.

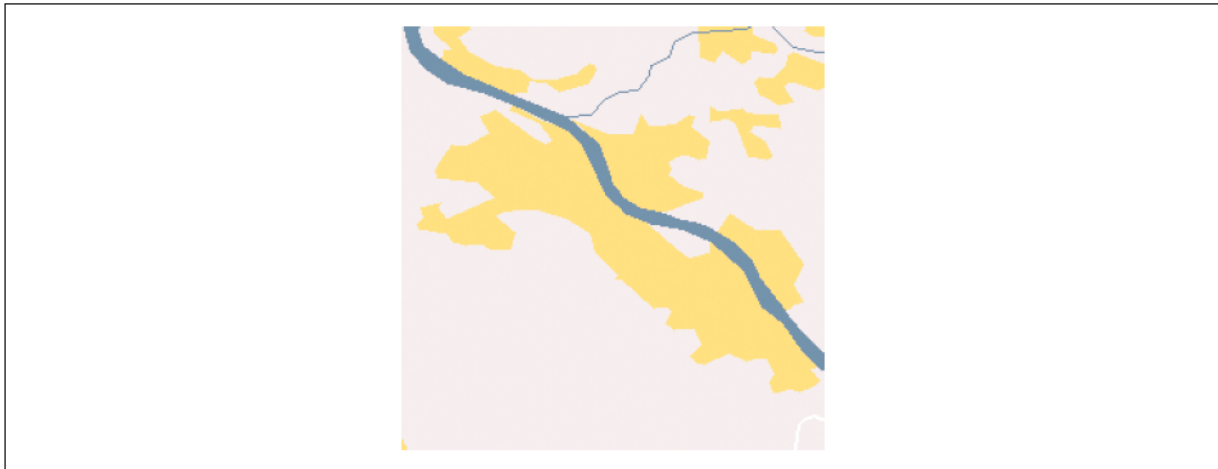


Abbildung 13-2: Kartenbild, das über einen GetMap-Aufruf ausgegeben wurde

Der OGC WMS-Aufruf im Detail

Im folgenden Abschnitt wird zur Wiederholung (siehe dazu auch Kapitel 12) der Aufruf in seine Bestandteile zerlegt. Sonderzeichen wie z.B. der Schrägstrich müssen in JavaScript maskiert werden und werden in der HTML-Notation als %2C angegeben. Im Beispieltext wurden diese Zeichen zur besseren Lesbarkeit durch die richtigen Zeichen ersetzt. Normalerweise müssen Sie diese Informationen nicht so detailliert auswerten, aber es hilft zu verstehen, wie die Anwendung funktioniert.

```
http://labs.metacarta.com/wms-c/Basic.py?  
LAYERS=basic&  
SERVICE=WMS&  
VERSION=1.1.1&  
REQUEST=GetMap&  
STYLES=&  
EXCEPTIONS=application/vnd.ogc.se_text&  
FORMAT=image/jpeg&  
SRS=EPSG:4326&  
BBOX=7.03125,50.625,7.207031,50.800781&  
WIDTH=256&  
HEIGHT=256
```

Bei dieser Art der Kommunikation zwischen Webserver und Client (hier also der Webbrowser) gilt, dass der erste Bestandteil, also die Basis-URL, durch ein Fragezeichen von den Aufrufparametern getrennt wird. Alle weiteren Parameter werden durch ein kaufmännisches Und-Zeichen (&) getrennt. Auf der linken Seite wird immer der Parameter mit dem Wert angegeben, getrennt durch ein Gleichheitszeichen. Der Parameter SERVICE hat also den Wert WMS. Deshalb spricht man bei dieser Art der Informationsübermittlung auch von »key-value pair« oder kurz KVP. Diese Kommunikation ist typisch für Internetanwendungen und ermöglicht es, in der URL zu sehen, welche Parameter übergeben werden.

Übersetzt bedeutet der Aufruf, dass der Server <http://labs.metacarta.com/wms-c/Basic.py> die Ebene basic als JPG-Bild mit 256 Pixeln Höhe und Breite als OGC WMS-Dienst der Version 1.1.1 liefern soll. Als Koordinatenausschnitt wird links unten (7.03125, 50.625) und rechts oben (7.207031, 50.800781) angegeben. Der Wert im Parameter Style gibt an, ob eine andere Darstellung verwendet werden soll, hier ist er leer, also soll der gleiche Stil verwendet werden.

Es gibt keine dummen Fragen

Ein weiterer Parameter sagt dem Server, wie er reagieren soll, falls ein Fehler auftritt. Wenn der Parameter EXCEPTIONS=application/vnd.ogc.se_text gesetzt ist, antwortet der Dienst mit einer Textmeldung. Alternativ kann der Parameter EXCEPTIONS=application/vnd.ogc.se_inimage gesetzt werden, in diesem Fall wird ein Bild in der Größe der angeforderten Karte zurückgeliefert, in der die Fehlermeldung als Text auftaucht. Dadurch sieht der Benutzer den Hinweis auf den Fehler direkt im Kartenbild, die Client-Software »merkt« in diesem Fall gar nichts von dem Fehler, denn es wird ganz korrekt ein Bild zurückgeliefert, wie in Abbildung 13-3 zu sehen.

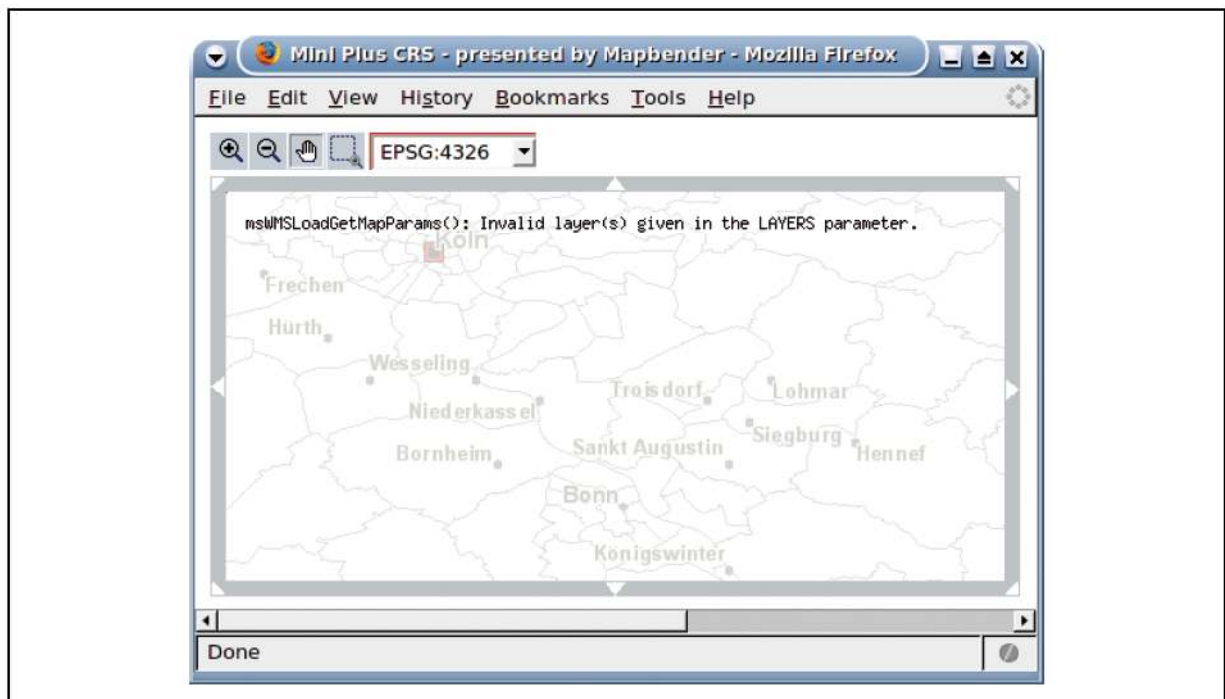


Abbildung 13-3: In dieser Anwendung ist zu sehen, wie ein Server mit einer Bildmeldung antwortet

Wir können einen Fehler provozieren, indem wir eine Ebene anfordern, die es nicht gibt. Wir raten hier einfach und fragen nach der Ebene »gibts_doch_gar_nicht«:

```
http://labs.metacarta.com/wms-c/Basic.py?LAYERS=gibts_doch_gar_nicht&SERVICE=
WMS&VERSION=1.1.1&REQUEST=GetMap&STYLES=&EXCEPTIONS=application%2Fvnd.ogc.se_text&
FORMAT=image%2Fjpeg&SRS=EPSG%3A4326&BBOX=7.03125%2C50.625%2C7.207031%2C50.800781&WIDTH=
256&HEIGHT=256
```

Karteninhalte wechseln

Diese Ebene gibt es wie erwartet nicht. Aber statt einer Fehlermeldung liefert dieser Dienst eine Liste mit Ebenen, die es gibt.

```
An error occurred: The requested layer (gibts_doch_gar_nicht) does not exist. Available layers are:  
* linktothepast  
* boston-freemap-remote-mapserver  
* boston  
* sigma-gif  
* zelda  
* mario-7-1  
* boston-freemap  
(und so weiter)
```

Jetzt kann getestet werden, ob die Ebene boston ein Ergebnis zurückliefert. Für den gewählten Ausschnitt (südliches Nordrhein-Westfalen) wird zwar ein Bild zurückgeliefert, aber es ist leer.

Es kann jedoch auch der Koordinatenausschnitt geändert werden. Der folgende Aufruf liefert einen Ausschnitt des Luftbilds von Boston. Diesen Ausschnitt sehen Sie in Abbildung 13-4.

```
http://labs.metacarta.com/wmsc/Basic.py?LAYERS=boston&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&STYLES=&EXCEPTIONS=application%2Fvnd.ogc.se_inimage&FORMAT=image%2Fjpeg&SRS=EPSG%3A4326&BBOX=-71.05957%2C42.297363%2C-71.037598%2C42.319336&WIDTH=256&HEIGHT=256
```

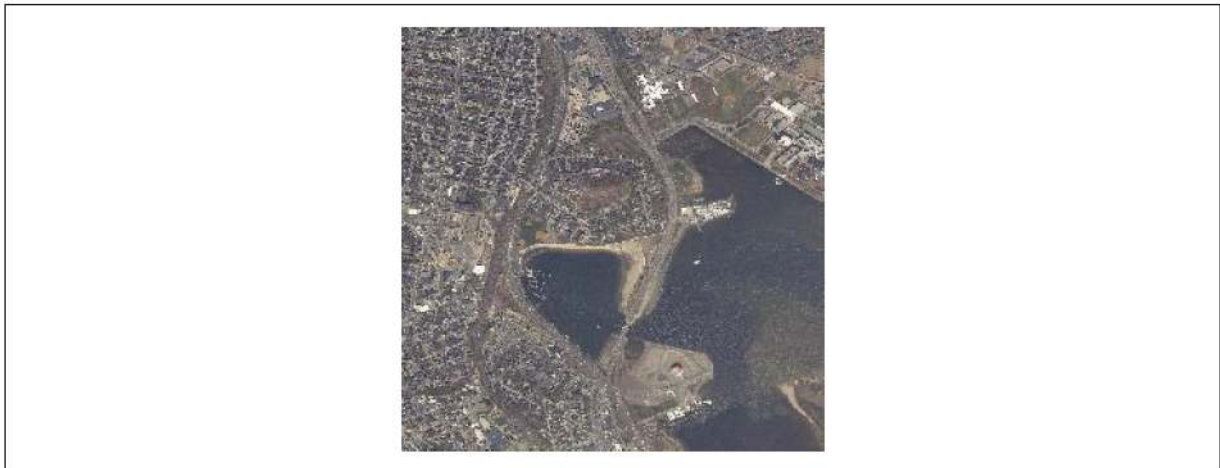


Abbildung 13-4: GetMap-Aufruf, der einen Ausschnitt von Boston liefert

OpenLayers auf dem eigenen Server

In den Beispielen oben haben wir OpenLayers als vorkonfigurierten Client von der Webseite des OpenLayers-Projekts eingebunden. Um OpenLayers dauerhaft auf dem eigenen Server zu betreiben, wird zunächst der Quellcode heruntergeladen und auf dem eigenen Server bereitgestellt. Die Software finden Sie auf der OpenLayers-Projektseite:

<http://www.openlayers.org/download/>

Damit HTML-Seiten auf die JavaScript-Bibliotheken zugreifen können, müssen sie über den Webserver Ihres Servers bereitgestellt werden.

Im Quellcode der HTML-Seite wird dann noch der Verweis auf die entfernte OpenLayers-Bibliothek geändert, er zeigt jetzt auf die neu angelegte, eigene URL. Je nachdem, wo Sie die JavaScript-Bibliotheken hinlegen, kann der Pfad ein anderer sein. So lange Sie den Code auf Ihrem eigenen Server installiert haben und auch von dort testen, können Sie als Quelladresse localhost angeben:

```
<script src="http://localhost/openlayers/OpenLayers.js"></script>
```

Bis hier verhält sich der Client genauso wie bisher. Allerdings nutzen Sie jetzt den Code auf Ihrer Festplatte und nicht mehr den des Projekts. Dadurch sind Sie unabhängig von der Verfügbarkeit oder der Belastung des OpenLayers-Servers. Andererseits müssen Sie ab jetzt darauf achten, auch Ihre lokale Kopie von OpenLayers zu aktualisieren, wenn Sie von Neuerungen in der Software profitieren möchten.

Andere OGC WMS-Kartendienste hinzufügen

Der Kartenserver des Projekts OpenLayers enthält zwar bereits eine Vielzahl von Ebenen und Inhalten, aber mit hoher Wahrscheinlichkeit nicht genau die Karten, die Sie für Ihre Fragestellung benötigen. Aus der ständig wachsenden Zahl der Kartendienste, die weltweit online gehen, können Sie jeden standardisierten WMS sofort in Ihr OpenLayers-Projekt einbinden. Jeder WMS-Kartendienst wird über die Funktion `OpenLayers.Layer.WMS()` als weitere Ebene hinzugefügt.

```
<html>
<head>
  <title>OpenLayers Germany</title>
  <script src="http://localhost/openlayers/OpenLayers.js"></script>
</head>
<body>
  <div style="width:100%; height:100%" id="map"></div>
  <script defer="defer" type="text/javascript">
    var map = new OpenLayers.Map('map');
    var wms = new OpenLayers.Layer.WMS( "OpenLayers WMS",
      "http://labs.metacarta.com/wms/vmap0", {layers: 'basic', format: 'png'} );
    map.addLayer(wms);
    map.zoomToMaxExtent();

    var twms = new OpenLayers.Layer.WMS( "Germany Map",
      "http://wms.wherogroup.com/cgi-bin/germany.xml?",
      {transparent: 'true', layers: 'roads',
       format: 'png'} );
    map.addLayer(twms);
  </script>
</body>
</html>
```

Eine Liste mit weiteren OGC WMS-Kartendiensten, die Sie in OpenLayers einbinden können, finden Sie auch auf der Webseite zum Buch:

<http://www.webmappingillustrated.com/>

Startausschnitt setzen

Nachdem der neue Dienst geladen wurde, sehen Sie zunächst immer noch die gesamte Welt, da der Startausschnitt durch den Aufruf der Funktion `map.zoomToMaxExtent()` gesetzt wurde, also auf die Ausdehnung des Kartendienstes *labs.metacarta.com*, der die ganze Welt abdeckt. Um auf einen speziellen Bereich zu zoomen, wird die Funktion `setCenter()` genutzt:

```
map.setCenter(new OpenLayers.LonLat(7, 50), 10);
```

In diesem Beispiel wird der Client beim Start auf die Koordinaten 50° N und 7° O zentriert. Beachten Sie, dass in OpenLayers die Reihenfolge der Koordinateneingabe der mathematischen Notation folgt, also zuerst der Wert auf der x-Achse, dann der auf der y-Achse. In geografischen Raumbezügen wird dagegen traditionell immer von Latitude/Longitude, also Breiten- und Längengrad gesprochen, so erfolgt auch meist die Angabe auf GPS-Geräten.

Der letzte Parameter gibt die Maßstabsebene an. Das ist ein OpenLayers-spezifischer Parameter, der mit einem Wert von 12 in diesen Breiten etwa einem Maßstab von 1:100.000 entspricht.

Markierung setzen

In der Karte kann auch eine Markierung gesetzt werden, die als Symbol überlagert dargestellt wird. Die Funktion `addMarker(marker)` zeichnet dabei ein Bild, das über eine URL referenziert ist, an die übergebene Koordinate. Das Bild wird so positioniert, dass es horizontal zentriert über der angegebenen Koordinate steht.

```
<html>
<head>
  <title>OpenLayers Example</title>
  <script src="http://localhost/openlayers/OpenLayers.js"></script>
</head>
<body>
  <div style="width:100%; height:100%" id="map"></div>
  <script defer="defer" type="text/javascript">
    var map = new OpenLayers.Map('map');
    var wms = new OpenLayers.Layer.WMS( "OpenLayers WMS",
      "http://labs.metacarta.com/wms/vmap0", {layers: 'basic', format: 'png'}
    );
    map.addLayer(wms);
    var twms = new OpenLayers.Layer.WMS( "Germany Map",
      "http://wms.wherogroup.com/cgi-bin/germany.xml?",
      {transparent: 'true', layers: 'roads',
       format: 'png'}
    );
    map.addLayer(twms);
    map.setCenter(new OpenLayers.LonLat(7.11, 50.71), 12);
    var markersLayer = new OpenLayers.Layer.Markers("Neue POI-Ebene");
    var iconSize = new OpenLayers.Size(21,25);
    var iconOffset = new OpenLayers.Pixel(-(iconSize.w/2), -iconSize.h);
```

```

var marker = new OpenLayers.Marker(
    new OpenLayers.LonLat(7.07, 50.73),
    new OpenLayers.Icon("http://www.openlayers.org/dev/img/marker.png",
        iconSize, iconOffset)
);
markersLayer.setVisibility(true);
markersLayer.addMarker(marker);
map.addLayer(markersLayer);
</script>
</body>
</html>

```

In Abbildung 13-5 sehen Sie den Client. Die Karte wird zentriert auf die geografische Koordinate (50,71N 7,11E) positioniert. An Koordinate (50,73N 7,07E) wird ein Symbol gezeichnet, das zentriert über dem Punkt steht (wie ein Zeiger).



Abbildung 13-5: Weltkarte vom metacarta-Server, überlagert mit Straßen in Deutschland vom WhereGroup-Server

OpenLayers ist seit Herbst 2007 ein graduiertes Projekt der Open Geospatial Foundation und wird von mehreren anderen Open Source-Projekten als Bibliothek verwendet.

Mapbender – Kartenanwendungen nach Maß

Mapbender ist eine in PHP, JavaScript und SQL implementierte Entwicklungsumgebung für Kartenanwendungen. Alle Informationen über die Oberfläche der Anwendungen werden zur Laufzeit dynamisch aus einer Datenbank gelesen, so wie das bei Content-Management-Systemen (CMS) üblich ist. Das gilt jedoch nur für die Anwendungsoberflächen, nicht für den »Content«. Die Karteninhalte selbst hingegen werden von den bereits bekannten OGC WMS-Diensten geliefert. Deshalb ist der Vergleich mit herkömmlichen »Content«-Management-Systemen etwas irreführend, die Inhalte gewöhnlich im gleichen System speichern. Hier geht es um die Nutzung und Verwaltung von vernetzten und verteilten Systemen.

Mapbender wird häufig zur Verwaltung von Kartendiensten in umfangreichen Geoportalen eingesetzt (siehe Abbildung 13-6). Anders als OpenLayers benötigt Mapbender für diese Aufgaben auch Softwarekomponenten, die serverseitig ausgeführt werden, es ist also keine reine Clientsoftware, sondern ein Hybrid. Das erweitert die Möglichkeiten der Software, da praktisch jede Funktion, die der Server bereitstellt, ausgeführt werden kann. Die Software muss sich also nicht auf die begrenzten Möglichkeiten eines Webrowsers und JavaScript beschränken. Die große Vielfalt an Funktionen und Erweiterungen bringt allerdings einen höheren Aufwand bei der Konfiguration mit sich.

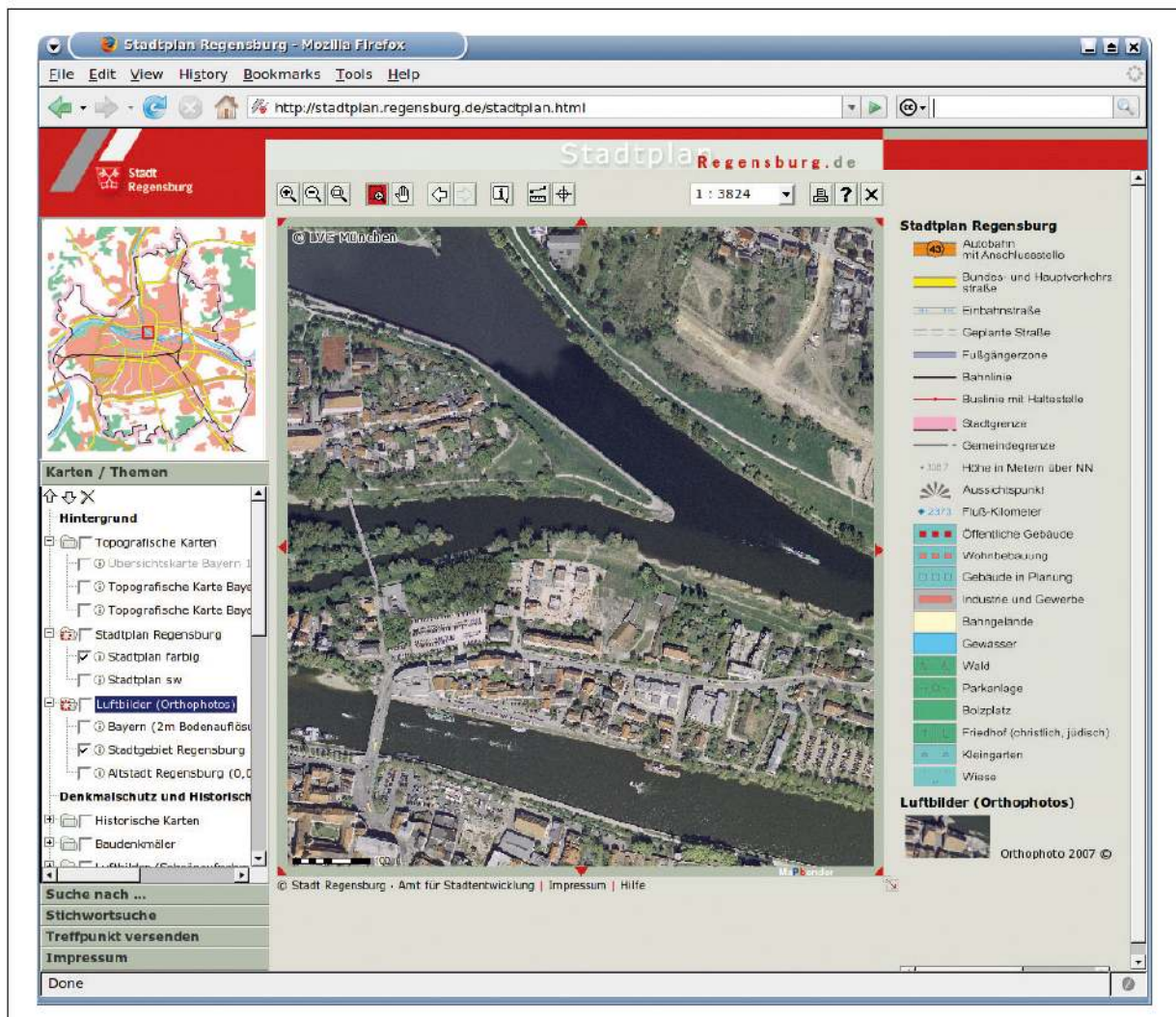


Abbildung 13-6: Der Stadtplandienst Regensburg, eine Mapbender-Anwendung

Ein Alleinstellungsmerkmal der Software Mapbender ist die vollständig über webbasierte Oberflächen implementierte Konfiguration. Hier passt auch wieder der Vergleich zum CMS: Editoren müssen nicht selbst komplexen HTML-Code schreiben, sondern nutzen die Administrationskomponenten, um Inhalte beizusteuern und anzuordnen.

Deshalb ist es auch nicht notwendig, dass Sie Mapbender sofort selbst installieren, um die Software kennenzulernen. So wie Sie in Wikipedia selbst Artikel anlegen und bear-

beiten können, ohne die Software auf Ihrem eigenen Rechner installieren zu müssen, können Sie auch Mapbender nutzen, eben um Kartenanwendungen zu erstellen.

Einen Demoversion von Mapbender finden Sie hier:

<http://www.mapbender.org/demo/>

Da Mapbender außerdem als freie Software lizenziert ist, steht es Ihnen zusätzlich offen, jederzeit eine eigene Installation aufzusetzen und zu pflegen.

Bevor Sie diesen Aufwand betreiben, sollten Sie genau ermitteln, wofür Sie die Software benötigen. Die folgenden Ausführungen zu möglichen Einsatzgebieten helfen Ihnen bei der Orientierung.

Einsatzgebiete und Funktionsumfang

Mapbender wird auf einem Server installiert, benötigt einen Webserver, die Skriptsprache PHP und eine zentrale Datenbank, um Benutzer, Kartendienste, Oberflächenelemente und die Beziehungen zwischen diesen drei Grundobjekten abzubilden. Das Installationspaket beinhaltet vorkonfigurierte Oberflächen für die Anzeige, Navigation und Abfrage der vom OGC standardisierten Dienste WMS und WFS sowie transaktionalen WFS (WFS-T). Die Anleitung zur Installation entnehmen Sie bitte der Online-dokumentation auf *<http://www.mapbender.org>*.

Die Administration enthält Module und Oberflächen für die Verwaltung der Kartendienste, Benutzer, Benutzergruppen und Berechtigungen. Die Mapbender-Datenbank ermöglicht eine exakte Benutzer- und projektspezifische Protokollierung aller Aktionen, Abfragen und Navigationen. Daraus lassen sich folgende Aufgabenbereiche ableiten:

- WebGIS-Client (OGC WMS, WFS, Katalog, Metadatenverwaltung und -suche)
- GeoCMS (Content-Management-System)
- Webbasierte Digitalisieroberfläche (OGC WFS-T-Client)
- Sicherung und Schutz gegen unbefugten Zugriff (Authentifizierung und Autorisierung)
- Zugriffsprotokollierung und Abrechnungssystem
- Orchestrierung vernetzter Geodaten- und Kartendienste

Mapbender lässt sich in praktisch jede bestehende heterogene Architektur und Webseite integrieren und kann jeden als OGC WMS standardisierten Kartendienst und als OGC WFS standardisierten Geometrieserver einbinden. Die Liste der Funktionen ist inzwischen sehr umfangreich, neben den üblichen Navigations- und Abfragemöglichkeiten kann Mapbender digitalisieren, dynamisch zwischen unterschiedlichen Koordinatensystemen wechseln und vieles mehr.

Durch die volle Unterstützung der OGC-Standards WMS und WFS wird Mapbender oft als Integrationsplattform für eine breite Palette professioneller GIS- und GDI-Software-

produkte unterschiedlicher Hersteller und Open Source-Projekte genutzt. Folgende Liste zeigt das Einsatzspektrum von Mapbender-Installationen:

- Stadtplandienste
- Geoportale und Geodateninfrastrukturen (GDI)
- Firmeninterne Geo-Auskunftssysteme
- Amtliche Katasterauskunft
- Öffentlicher Transport und Verkehr
- Flughafenverwaltung
- Straßenwacht
- Netzüberwachung (Telekom, Strom)
- Tourismusinformation
- Fahrzeugflottenmanagement
- Aus- und Fortbildung, Forschung
- Katastrophenmanagement (ad hoc und präventiv)
- Frühwarnsysteme
- Monitoring und Überwachung

Grundlagen – Wie funktioniert Mapbender?

Die Software Mapbender ist eine recht komplexe Anwendung, die seit über sechs Jahren entwickelt wird und vornehmlich in großen Unternehmen und der öffentlichen Verwaltung eingesetzt wird. Der Einsatz von Mapbender empfiehlt sich vor allem in Umgebungen, in denen größere Mengen an Geodaten von vielen unterschiedlich berechtigten Benutzern gepflegt und genutzt werden sollen. Mapbender wurde entwickelt, um diesen oftmals über Jahrzehnte gewachsenen Datenwald zu durchforsten, zu katalogisieren und nutzbar zu machen.

Komplexe Fragestellungen fordern aber nicht zwingend auch immer komplexe Lösungen, sondern umgekehrt: Oft sind die einfachsten Konzepte am besten zur Lösung komplexer Fragen geeignet. Dieses Grundparadigma wurde auch beim zweiten Redesign von Mapbender in 2005 berücksichtigt.

Die drei Grundobjekte in Mapbender

Es gibt in Mapbender lediglich drei Grundobjekte, die zueinander in Beziehung gesetzt werden. Über die Beziehungen werden die Berechtigungen abgebildet.

- Benutzer und Gruppen
- Oberflächen (Anwendungen, GUI)
- Karten- und Geometriedienste (OGC WMS und WFS)

Für die Arbeit mit Mapbender wird immer ein Benutzerkonto benötigt. Auch wenn eine Mapbender-Kartenanwendung ohne interaktive Anwendung bereitgestellt werden soll, erfolgen alle Zugriffe über ein Benutzerkonto. Soll die Anmeldung öffentlich sein, wird dazu ein eigenes Konto angelegt, z.B. mit dem Namen »Gast« und dem Passwort »Gast«. Durch die eindeutige Zuordnung einer Anwendung zu einem Benutzer besteht auch die Möglichkeit, die in der Anwendung enthaltenen Kartendienste vor unberechtigtem Zugriff zu schützen.

Wie bereits beschrieben wurde, sind alle Mapbender-Anwendungen dynamisch erstellte HTML-Seiten. Die Oberflächen bilden die Benutzerschnittstelle der Anwendungen ab und werden auch als Graphical User Interfaces oder kurz »GUI« bezeichnet. Sie enthalten alle Bedienelemente, ordnen sie in der HTML-Seite an, parametrisieren sie und setzen sie mit anderen Objekten in Beziehung. Durch die Auswahl der Oberflächenelemente und deren Zusammenspiel wird der Funktionsumfang einer Anwendung definiert.

Die standardisierten Webdienste (OGC WMS und WFS) werden über Verweise (Links) angesprochen. Das bedeutet, dass in Mapbender selbst keinerlei Karten erzeugt werden. Genau wie in OpenLayers werden lediglich Verweise auf Kartendienste gesetzt, die vom Webbrowser angefordert werden. Kartendienste können auch komplette Geo-Objekte im GML-Format zurückliefern, die der Client visualisieren kann oder dem Benutzer zur Navigation oder als Markierung bereitstellt.

Zusammenspiel der Grundobjekte

Die Anmeldung erzeugt für das entsprechende Benutzerkonto eine eindeutige Sitzung mit einer zugeordneten, intern durch PHP verwalteten Identifikationsnummer (Sitzungs-ID). Anhand dieser Sitzungs-ID, die bei jeder Anfrage mit übergeben wird, kann Mapbender ermitteln, welche GUI (Anwendung) der Benutzer verwenden darf.

Die Module einer GUI definieren den Funktionsumfang. Wenn eine Baumstruktur in der GUI enthalten ist, kann der Benutzer Ebenen an- und ausschalten. Ist eine Koordinatensystemauswahl vorgesehen, kann der Benutzer die Projektion wechseln. Ist sie nicht enthalten, kann der Benutzer die eingebundenen Kartendienste auch nicht umprojizieren.

In jede GUI werden Kartendienste (WMS) eingebunden, die dort vorkonfiguriert und überlagert werden können. Die Funktionen der Anwendung (GUI) können auf die eingebundenen Kartendienste angewendet werden. Auf gesicherte Dienste greifen Benutzer nie direkt zu, sondern immer nur durch den OWS-Proxy, der die Berechtigung für den Zugriff kontrolliert.

Benutzer und Gruppen

Mapbender-Benutzerkonten sind innerhalb von Mapbender systemweit eindeutig und werden über eine ID identifiziert. Alle Benutzer sind zunächst gleichberechtigt, es gibt keine benutzerspezifische Unterteilung in Administratoren und »normale« Benutzer.

Benutzer erhalten Administrierrechte ausschließlich, indem ihnen entsprechende Anwendungen (GUI) zugeordnet werden. Die Zuordnung erfolgt über den Link *Benutzerzugriff erteilen* im Administrationsbereich (z.B. in der Oberfläche *admin2_de*). Hier können Beziehungen zwischen Benutzern und Anwendungen hergestellt werden.

Benutzer, die eine Berechtigung für eine GUI erhalten, erben automatisch die Berechtigung für alle Dienste, die in dieser GUI enthalten sind. Hierüber wird auch die Sicherung von Diensten in Mapbender gelöst. Die Sitzungs-ID ist gleichzeitig das Ticket, über das Benutzer Dienste abfragen können, die gegen unberechtigten Zugriff geschützt wurden. Die Benutzerverwaltung besteht also lediglich aus dem Objekt Benutzer (Gruppen bestehen aus 1 bis n Benutzern) und dem Objekt Anwendung (GUI) sowie deren Beziehung.

Es gibt eine eindeutige Hierarchie, die mit dem Benutzer root beginnt. Der Wurzelbenutzer wird bei der Installation von Mapbender automatisch angelegt und ist OWNER von jedem neuen Benutzer. Benutzer, die von root angelegt wurden, und die Berechtigung für eine Administrationsoberfläche (GUI) erhalten haben werden so auch zu Administratoren und können selbst weitere Benutzer anlegen. Bei diesen neuen Benutzern ist der OWNER dann jedoch nicht root, sondern eben der Ersteller. Der Ersteller eines Benutzers kann das Attribut OWNER anderen Benutzern vererben und auch sich selbst aus der Gruppe der OWNER entfernen. Hierdurch wird die Mandantenfähigkeit im Portalbetrieb gewährleistet. Einzelne Fachbereiche können sich selbst verwalten und sind vollständig unabhängig von anderen Fachbereichen desselben Mapbender-Portals. Benutzer können Mitglieder der gleichen Mapbender-Gruppe werden, auch wenn sie aus unterschiedlichen Fachbereichen kommen, also unterschiedliche OWNER haben. So können einzelne Mitarbeiter bestimmter Fachbereiche mit Benutzern anderer Fachbereiche projektbezogen zusammenarbeiten.

Authentifizierung (Überprüfung der Echtheit)

Benutzer authentifizieren sich bei Mapbender (melden sich an) durch die Eingabe von Benutzerkennung und Passwort. Die Authentifizierung ist der Vorgang der Überprüfung einer behaupteten Authentizität, sie ist erfolgreich, wenn die Kombination aus Benutzername und Passwort stimmt. Im deutschen Sprachgebrauch wird der Begriff Authentifizierung oft synonym mit dem Begriff Authentisierung verwendet, was nicht ganz korrekt ist, aber den englischsprachigen Gebrauch wiedergibt, bei dem beide Vorgänge nicht differenziert als »Authentication« bezeichnet werden.

Benutzername und Passwort können interaktiv über ein Formular eingegeben werden (wie bei einer Onlinebanking-Webseite) oder der URL als Parameter `name` und `password` mit übergeben werden. Wenn in der Mapbender-Umgebung vertrauliche Daten zum Einsatz kommen, sollten diese Daten nur über ein verschlüsseltes Protokoll eingegeben und verschickt werden. Die im Internet übliche Verschlüsselung erfolgt über das Transportprotokoll SSL/TSL und wird durch die Anwendung HTTPS und den URL-Vorspann *https://* kenntlich gemacht.

Die Authentifizierung kann auch durch eine externe Benutzerverwaltung erfolgen (z.B. eine Systemanmeldung über LDAP). In diesem Fall übernimmt Mapbender die authentifizierten Benutzer des Drittsystems und kümmert sich nur noch um die Verwaltung der Berechtigungen.

Autorisierung (Berechtigung)

Wurden die Benutzerdaten bestätigt, ermittelt Mapbender, für welche Anwendungen die Berechtigungen dieses Benutzerkontos gelten. Dadurch wird es diesem Benutzerkonto ermöglicht, auf alle Kartendienste zuzugreifen, die in dieser Anwendung (GUI) enthalten sind – und zwar genau mit den Funktionen, die die jeweilige GUI bereitstellt.

Alle Berechtigungen werden also durch Beziehungen zwischen Benutzern und Anwendungen sowie zwischen Anwendungen und Diensten hergestellt. Ein Benutzer greift nicht direkt auf einen Dienst zu, sondern immer nur über eine Anwendung.

Die GUI – Anwendung und Berechtigung

Die Anwendung (GUI) ordnet alle Bedienelemente auf einer Oberfläche an und bindet Kartendienste ein. Alle Funktionen, die eine GUI bietet, kann auf die eingebundenen Kartendienste angewendet werden. Über diese Beziehung wird die Berechtigungsstruktur abgebildet, also nicht nur welche Kartendienste genutzt werden können, sondern auch mit welchen Funktionen.

In dem Beispiel (siehe Abbildung 13-7) darf Benutzer 1 die Dienste 1, 2, 3 und 4 mittels der Funktionen nutzen, die in GUI 1 definiert sind (ZoomIn und ZoomOut). Benutzer 3, 4 und 5 sind über die Gruppe A berechtigt, auf die Dienste von 1 bis 7 zuzugreifen, und zwar über die Funktionen der Anwendungen GUI 1 (für Dienste 1 bis 4) wie auch GUI 2 (für Dienste 4 bis 7). Benutzer der Gruppe A dürfen also zusätzlich Abfragen an die Kartendienste stellen.

Benutzer 2 hingegen darf nur auf die Dienste 4 bis 7 zugreifen, das aber inklusive der Funktion »Abfrage«. Benutzer N darf die Dienste 4 bis 7 über GUI 2 nutzen und auch als Einziger die Dienste 7 und N über GUI N, also Objekte im Datenbestand anlegen, ändern und löschen. Ein interessantes Detail in dieser Darstellung ist die Nutzung des gleichen Datenbestands sowohl durch rein lesende als auch einen schreibenden Dienst. Benutzer N kann Daten eingeben, die alle Benutzer der GUI 2 (Gruppe A und Benutzer 2) lediglich lesend nutzen können.

Kartendienst 4 kann als einziger Dienst von allen Benutzern verwendet werden. Der Dienst selbst »merkt« dabei allerdings keinen Unterschied zwischen Zugriffen durch Benutzer 1, 2, Benutzer N oder die Gruppe A (Benutzer 3 bis 5), da die WMS- und WFS-Standards des OGC keine Benutzerberechtigung kennen.

Deshalb wird über das Mapbender-Berechtigungsmodell auch die Sicherheitsinfrastruktur von OGC-Diensten gesteuert. Hierbei kapselt das Mapbender-Modul *OWS Proxy* Dienste und ermöglicht nur autorisierten Benutzern den Zugriff, die entweder bereits

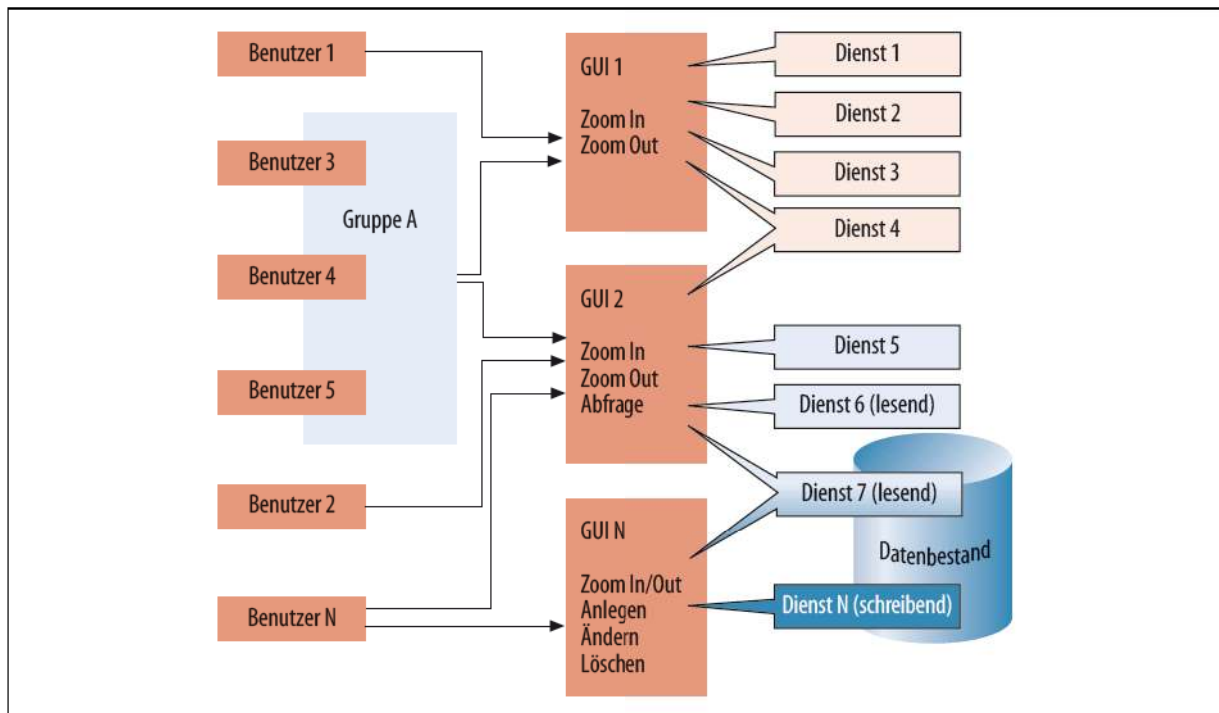


Abbildung 13-7: Berechtigungsstrukturen in der Mapbender-Benutzerverwaltung

über ein gültiges Ticket verfügen (Sitzungs-ID) oder die Authentifizierungsdaten (Benutzername und Passwort) in den Aufrufparametern mit übergeben.

Durch diese an sich einfache Konstruktion mit drei Basisobjekten und ihren Beziehungen werden hochkomplexe Organisationsstrukturen abgebildet und gesichert.

Kartendienste und deren Verwaltung

Mapbender wird mit einer umfangreichen Liste von OGC WMS-Kartendiensten ausgeliefert, die ohne technische Zugriffsbeschränkung genutzt werden können. Beachten Sie jedoch unbedingt das Copyright der Kartendienste und überprüfen Sie die Nutzungsbedingungen, bevor Sie die Karten in irgendeinem Kontext verwenden, der als »kommerzieller Einsatz« bezeichnet werden könnte, denn es ist möglich, dass dieser Einsatz nicht gestattet ist! Die Bezeichnung »kommerzielle Nutzung« ist leider ein sehr schwammiger Begriff, und es gibt auch noch keine allgemeingültigen auf Geodaten- oder Kartendienste zugeschnittenen Lizenzbestimmungen. Das erschwert die professionelle Nutzung enorm.

Kartendienst laden (OGC WMS Capabilities)

Um einen neuen Kartendienst nutzen zu können, benötigen Sie zuerst das OGC WMS Capabilities-Dokument. Es enthält alle Informationen und Eigenschaften des Diensts, die Clients für die Nutzung benötigen. Eine eingehende Beschreibung finden Sie in Kapitel 12.

Die Inhalte des Capabilities-Dokuments werden über die Administrationsoberfläche geladen, ausgelesen und in der Mapbender-Datenbank gespeichert. Das Mapbender-Berechtigungsmodell schreibt vor, dass bereits beim Hochladen jedes Diensts eine Beziehung zu einer GUI hergestellt werden muss.



Legen Sie sich eine eigene GUI an und nutzen Sie diese als Behälter für Ihre eigenen Dienste. Diese GUI dient lediglich Verwaltungszwecken und enthält keinerlei eigene Oberflächenelemente. Mehrere solcher reinen Verwaltungs-GUIs werden bei der Installation von Mapbender bereits angelegt, sie sind durch ein vorangestelltes *wms_* gekennzeichnet. In diesen GUIs finden Sie auch die Liste der mitgelieferten Verweise auf Kartendienste.

Sie können solche GUIs ohne Oberfläche, aber mit vielen WMS-Diensten auch hervorragend nutzen, um Ihren Anwendern das Hinzuladen weiterer Kartendienste zu vereinfachen. Die Metadatensuche erlaubt ebenfalls die Suche nach Kartendiensten, für die der angemeldete Benutzer eine Berechtigung besitzt. Handelt es sich um eine Vielzahl von Diensten, ist es unzweckmäßig, für jede mögliche Kombination eine eigene Oberfläche zu implementieren, die Organisation in GUIs mit unterschiedlichen Bezeichnungen erleichtert die Verwaltung und Nutzung enorm.

Die Informationen des Kartendienstes werden dabei in Mapbender in der Datenbank abgelegt. Wenn sich die Diensteseigenschaften ändern, muss der Dienst in Mapbender aktualisiert werden, da sonst möglicherweise falsche Anfragen an den Service gestellt werden.

In der Mapbender-Administration können Sie den Dienst GUI-abhängig konfigurieren und beispielsweise nur bestimmte Ebenen in einer GUI zur Anzeige aktivieren. Es gibt Dienste, die mehrere hundert Ebenen enthalten. Wenn ein Client bei jedem Aufruf alle Ebenen gleichzeitig anschaltet, wird auf der Karte nichts mehr zu erkennen sein. Sie können GUIs erzeugen, in denen unterschiedliche Ebenen eines WMS angezeigt werden.

Da Sie die fremden Dienste nicht umkonfigurieren können, pflegen Sie Ihre eigene Sicht auf die Dienste. Um das zu erleichtern, bietet Mapbender ein Werkzeug, das eine beliebige Auswahl von Diensten per Knopfdruck oder auch in regelmäßigen Abständen überprüft. Alle Änderungen in der Konfiguration oder der Struktur der Dienste werden angezeigt und, falls gewünscht, automatisch aktualisiert.

Kartendienst in GUI einbinden

Nachdem Sie den Kartendienst über die Mapbender-Administration in eine GUI geladen haben, können Sie ihn auch in allen anderen GUIs nutzen, für die Sie Administratorrechte besitzen. Die Berechtigungsstruktur erlaubt es jedem Benutzer mit Berechtigung auf eine GUI, auch die darin enthaltenen Dienste zu nutzen. So können Sie die Veröffentlichung eines neuen Kartendienstes komfortabel über die Zuordnung zu einer GUI regeln.

Zunächst sollten Sie neue Dienste in einer eigenen GUI laden und konfigurieren, bis die Darstellung Ihren Vorstellungen entspricht. Im nächsten Schritt können Sie den Dienst über den Verwaltungsbereich *WMS Zuordnung* und das Modul *WMS einbinden* in jede andere GUI übernehmen. In Abbildung 13-8 sehen Sie die Zuordnung einer GUI zu einem Benutzer.

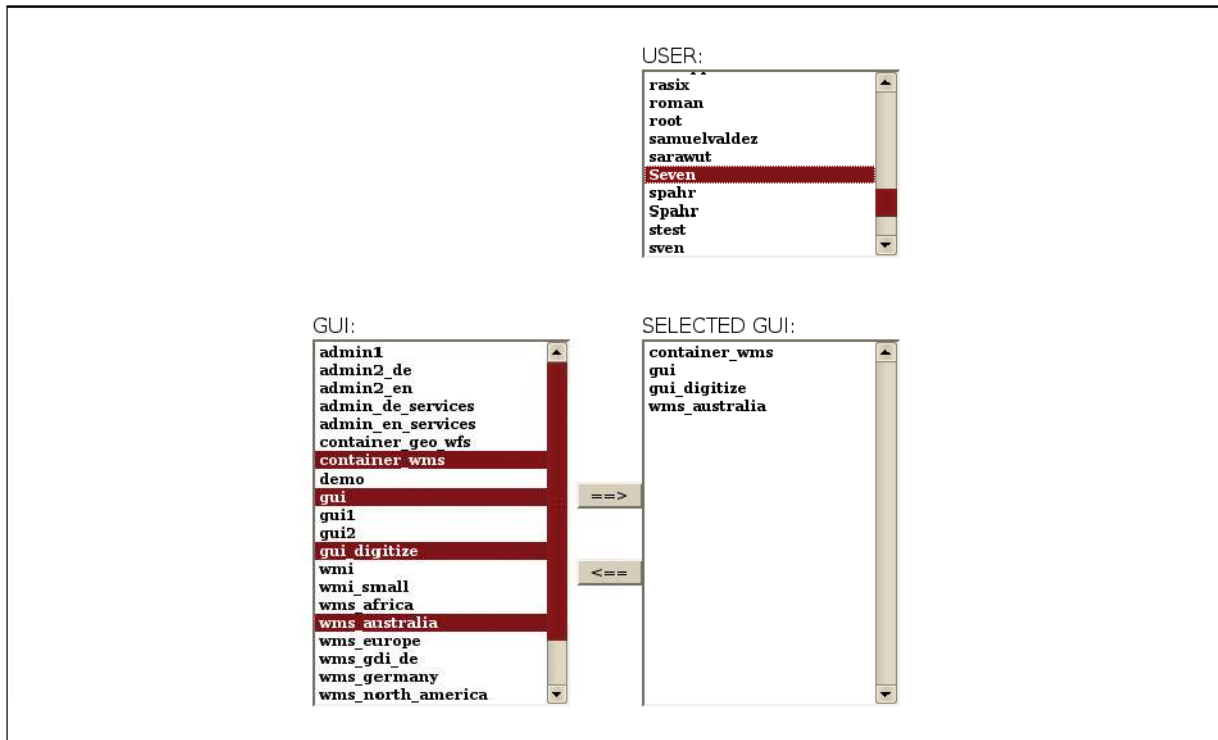


Abbildung 13-8: Zugriffsberechtigungen an Benutzer vergeben

Wählen Sie zunächst in der GUI-Auswahl Ihre Anwendung aus. Daraufhin werden in der Liste *WMS* alle bereits eingebundenen Kartendienste angezeigt. Wählen Sie als Nächstes aus der Liste *Load WMS* den Kartendienst, den Sie in Ihre Anwendung übernehmen möchten. Daraufhin werden in der Liste *FROM:* alle GUIs angezeigt, die diesen Dienst bereits verwenden. Wählen Sie eine GUI aus dieser Liste aus, daraufhin werden deren Einstellungen für den gewählten Kartendienst übernommen.

WMS-Einstellungen ändern

Die Eigenschaften eines Kartendienstes werden durch das Capabilities-Dokument definiert, z.B. wie viele Ebenen der Dienst beinhaltet, die Namen dieser Ebenen und so weiter.

Des Weiteren können für einige Parameter unterschiedliche Werte ausgewählt werden, z.B. für das Koordinatensystem oder das Bildformat. Für Karten mit Luftbildern ist das hochkomprimierte, aber verlustbehaftete JPEG-Format besser geeignet, für flächige, thematische Karten mit transparenten Bereichen dagegen eher das Format PNG. Außerdem unterstützt nicht jeder Dienst jedes Bildformat. Die Einstellungen eines Kartendienstes können individuell für jede GUI festgelegt werden, damit wird die höchste Flexibilität gewährleistet.

In Abbildung 13-9 sehen Sie die WMS-Einstellungen in der Mapbender-Administration. In der Abbildung ist oben rechts eine Liste zu sehen, sie enthält alle WMS, die in die Anwendung (GUI) eingebunden sind.

GUI

WMS-TITLE

up

down

remove

save

LINK: Capabilities

WMS ID: 893

Mapformat: image/png

Infoformat: text/html

Exceptionformat: application/vnd.ogc.se_ir

Visibility: visible

Nr.	ID	Parent	Name	Title	on/off	sel	sel_default	info	info_default	minScale 1	maxScale 1	Prio	setWFS
1	190C		Germany	Germany	☒	☒	☒	☐	☐	1	0	0	
2	190C	0	Topograph	topography	☒	☒	☒	☐	☐	0	0	1	setWFS
3	190C	0	Grenze	border	☒	☒	☒	☐	☐	750000	0	2	setWFS
4	190C	0	Bundeslaer	federal states	☒	☒	☒	☐	☐	750000	0	3	setWFS
5	1901	0	Staedte	places	☒	☒	☒	☐	☐	0	0	4	setWFS
6	1901	0	Postleitzahl	postcode areas	☒	☒	☒	☒	☒	0	0	5	setWFS
7	1901	0	Bahnhlinien	railroad lines	☒	☒	☒	☐	☐	0	0	6	setWFS
8	1901	0	Fluesse	rivers	☒	☒	☒	☐	☐	0	1250000	7	setWFS
9	1901	0	Strassen	roads	☒	☒	☒	☐	☐	0	0	8	setWFS
10	1901	0	Staedtepur	places as point	☒	☒	☒	☒	☒	0	0	9	setWFS
11	1901	0	Postleitzahl	postcode areas (i	☒	☒	☒	☐	☐	0	350000	10	setWFS
12	1901	0	Staedenar	places (names)	☒	☒	☒	☐	☐	0	0	11	setWFS
13	1901	0	Bundeslaer	federal states (ni	☒	☒	☒	☐	☐	1000000	0	12	setWFS

Abbildung 13-9: Einstellungen eines OGC WMS-Kartendienstes in der Mapbender-Administration

Die Grundeinstellungen betreffen Formate und die Sichtbarkeit in der Ebenen-Baumstruktur:

- Mapformat
- Infoformat
- Exceptionformat
- Visibility
- Auswahl des Bildformats
- Auswahl des Ergebnisformats

- Auswahl des Formats für Fehlermeldungen
- Sichtbarkeit in der Ebenen-Baumstruktur

Im unteren Bereich der Administration werden die Ebenen des Diensts aufgelistet. Jede Ebene wird durch eine in Mapbender eindeutige ID gekennzeichnet. Die weiteren Parameter im Einzelnen:

Parameter	Beschreibung
<i>Nr.</i>	Laufende Nummer.
<i>ID</i>	Eindeutige ID.
<i>Parent</i>	Nummer der Wurzelebene (außer in Gruppen immer 0 (null)).
<i>Name</i>	Name der Ebene. Wird in den Requests (GetMap, GetFeature, GetLegendGraphic) in der URL übergeben.
<i>Title</i>	Der Titel der Ebene kann Sonderzeichen enthalten und sollte für den Endbenutzer sprechend sein.
<i>on/off</i>	Definiert, ob die Ebene in der GUI angezeigt werden soll.
<i>sel</i>	Bedeutet selectable (selektierbar) und definiert, ob diese Ebene vom Benutzer an- und ausgeschaltet werden darf.
<i>sel_default</i>	Definiert, ob diese Ebene beim Laden der Anwendung aktiv ist.
<i>info</i>	Definiert Ebenen, die abfragbar sind, können zur Abfrage aktiviert oder deaktiviert werden.
<i>info_default</i>	Definiert, ob diese Ebene beim Laden der Anwendung zur Abfrage aktiviert ist.
<i>minScale</i>	Definiert die kleinste Maßstabszahl (den größten Maßstab), in der diese Ebene dargestellt wird. Dieser Wert wird vom Server vorgegeben und kann in Mapbender lediglich weiter eingeschränkt werden.
<i>maxScale</i>	Definiert die größte Maßstabszahl (den kleinsten Maßstab), in der diese Ebene dargestellt wird. Dieser Wert wird vom Server vorgegeben und kann in Mapbender lediglich weiter eingeschränkt werden.
<i>Prio</i>	Definiert die Reihenfolge, in der die Ebenen übereinandergezeichnet werden. Dabei gilt, dass die kleinste Zahl zuerst gezeichnet wird.
<i>setWFS</i>	Setzt einen Verweis auf einen WFS-Dienst, der mit diesem WMS-Dienst gekoppelt wird. Diese Kopplung wird vor allem für die Digitalisierung und interaktive »Clickable Maps« benötigt.

Bevor Sie diesen Dialog verlassen, müssen die Einstellungen immer über den großen Knopf *save* gesichert werden! Wenn Sie diese Einstellungen einmal zentral vorgenommen haben, können Sie den Dienst in alle weiteren GUIs, die Sie erstellen, direkt übernehmen. Die Einstellungen werden dann kopiert.

Bei einer Änderung in der Struktur des Diensts ist es aber erforderlich, dass Sie die Einstellungen kontrollieren. Aus Sicherheitsgründen werden alle neuen Ebenen und solche, die umbenannt wurden, zunächst in allen betroffenen GUIs automatisch deaktiviert.

Anwendung veröffentlichen

Um eine Anwendung zu veröffentlichen, müssen Sie die neue GUI lediglich über die Benutzerverwaltung den Benutzern oder der Gruppe zuordnen. Wenn Sie beabsichtigen, eine öffentliche Anwendung bereitzustellen, die jeder sehen darf, geben Sie dem Aufruf der GUI einfach einen öffentlich bekannten Benutzer mit, z.B. ein Gastkonto mit dem Passwort »Gast«.

Der Aufruf von Mapbender erfolgt dann über eine URL und wird mit dem Benutzernamen, dem Passwort und dem Namen der GUI parametrisiert. Wenn der entsprechende Benutzer lediglich Zugriff auf eine einzige GUI hat, wird diese automatisch gestartet, dann kann der Name der GUI entfallen.

Dazu hier ein Beispiel. Die Oberfläche, die die Mapbender-Anwender weltweit anzeigt, wird über die folgende URL geöffnet:

```
http://wms.wherogroup.com/mapbender/frames/login.php?name=mb&password=mb&mb_user_myGui=mapbender_user
```

Nutzung von Mapbender als Geoportal

Mapbender kann ähnlich wie eine Webseite oder ein CMS von vielen Benutzern gleichzeitig genutzt werden. Manche Mapbender-Installationen bei Internetservice-Betreibern oder großen Geodatenportalen bieten die Möglichkeit, sich selbst ein Benutzerkonto einzurichten. Dadurch können Sie die Software nutzen, ohne sie selbst installieren zu müssen. Die Open Source Geospatial Foundation betreibt einen Server, der diese Möglichkeit bietet.

Für die kommerzielle Nutzung (mit Ausfallsicherheit, Hotline, Backup usw.) finden Sie weitere Informationen unter der Webadresse:

<http://www.mapbender.org/ISP>



Das Internet ist ein dynamischer Ort, an dem sich ständig alles verändert. Deshalb kann es passieren, dass die Verweise im folgenden Text nicht funktionieren, dann müssen Sie gegebenenfalls die Basis-URL gegen eine neue Adresse austauschen. Aktuelle Informationen zu den Verweisen in diesem Buch finden Sie immer unter:

<http://www.webmappingillustrated.com/>

Eine weitere Installation erhalten Sie auf den Servern der Firma WhereGroup, die auch einen Geodatendienst mit einer einfachen Deutschlandkarte zur kostenfreien Nutzung anbietet. Die Seiten finden Sie unter der Adresse:

<http://wms.wherogroup.com/mapbender/>

Die folgenden Abschnitte erläutern, wie Sie auf einem Mapbender-Server bei der OSGeo ein Benutzerkonto erstellen und damit lernen können, die Anwendung Mapbender zu nutzen. Als Beispiel wird eine einfache, neue Oberfläche erstellt, und einige weltweit verfügbare Kartendienste werden eingebunden.

Weiter unten im Text wird auf spezielle Administrationsfunktionen eingegangen, beispielsweise wie weitere Benutzer angelegt, diese in Gruppen eingeteilt und Rechte zugeordnet werden können. Wie Sie professionelle Oberflächen erstellen, Funktionen hinzufügen, gesicherte Dienste hochladen und spezifische Rechte vergeben, lernen Sie am besten über die Onlinedokumentation und Tutorials:

<http://www.mapbender.org/index.php?title=Documentation>

Und wenn Sie mal nicht weiter wissen, profitieren Sie einfach von der Gemeinschaft der Mapbender-Anwender, die über eine Mailingliste und das Projekt-Wiki kommunizieren.

http://www.mapbender.org/Mailing_Lists

Alle im Folgenden beschriebenen Schritte können sowohl mit einer eigenen Installation als auch mit den oben angegebenen URLs nachvollzogen werden. Nachfolgend wird das Anlegen eines Benutzerkontos auf dem OSGeo-Mapbender-Server beschrieben.

Mapbender-Benutzer anlegen

Als Erstes legen Sie sich ein eigenes Benutzerkonto auf dem OSGeo-Mapbender-Server an. Gehen Sie auf die Seite <http://mapbender.telascience.org> und geben Sie einen Namen und ein Kennwort an. (Das Gleiche können Sie auch auf Ihrer eigenen Installation oder einer der anderen oben angegebenen Adressen tun.)



Prinzipiell sollten Sie mit der Angabe Ihrer E-Mail-Adresse vorsichtig sein. Auf jeden Fall sollten Sie die E-Mail-Adresse immer nur dann in einer Webseite eintragen, wenn Ihnen der Anbieter der Seiten bekannt ist und Sie dem Betreiber vertrauen! Leider ist das Netz immer noch nicht besonders sicher, und es ist immer noch sehr schwierig, die Echtheit einer Seite zu überprüfen. Diesbezüglich wurden inzwischen einige technische Möglichkeiten entwickelt, die aber leider für den normalen Internetanwender meist nicht nachvollziehbar sind. Fälscher dagegen lernen schnell und schaffen es, die neuen technischen Möglichkeiten zu unterlaufen oder zu umgehen.

Die Mapbender-Software nutzt die E-Mail-Adresse, um Ihnen ein neues Passwort zu schicken, falls Sie Ihr Passwort vergessen haben. Beachten Sie, dass weder Ihr Mapbender-Administrator noch einmal die Entwickler der Software Ihr Passwort auslesen könnten, da es in der Datenbank verschlüsselt wird! Deshalb bekommen Sie auch immer ein neues, wenn Sie Ihr Passwort vergessen haben.

Melden Sie sich mit dem neuen Benutzer und dem Passwort an. Beachten Sie, dass Ihr Account gesperrt wird, wenn Sie mehrfach (der Wert ist einstellbar) ein falsches Passwort eingegeben haben. Das ist eine weitere Sicherheitsmaßnahme, um zu vermeiden, dass jemand Passwörter durch wiederholte Eingabeversuche errät. Nur ein Mapbender-Administrator kann Ihren Account wieder freischalten, wenn die Anmeldung gesperrt wurde. Aber auch der Mapbender-Administrator kann Ihr Passwort nicht entschlüsseln, da es in der Datenbank verschlüsselt wird.

Anwendung öffnen

Nachdem Sie sich angemeldet haben, wird eine Liste der Ihnen zur Verfügung stehenden Oberflächen mit Namen und einer kurzen Beschreibung angezeigt. Durch einen Klick auf den Namen *gui*, *gui1*, *gui2* oder *gui_digitize* starten Sie eine der Standardkartenanwen-

dungen, die immer mit Mapbender ausgeliefert werden. Sie bestehen aus verschiedenen Komponenten, die auch beliebig miteinander kombiniert und im Browserfenster unterschiedlich positioniert werden können.

Oberflächenelemente

Als Beispielanwendung sehen wir uns die Oberfläche *gui* der Version 2.4.3 an (3. Quartal 2007) und identifizieren einige der Bedienelemente. Sie sehen die Oberfläche in Abbildung 13-10. Die Anordnung der Elemente ist beliebig, das folgende Beispiel zeigt eine der Basisoberflächen des Mapbender-Installationspakets, die nur einen Teil aller möglichen Funktionen abbildet.

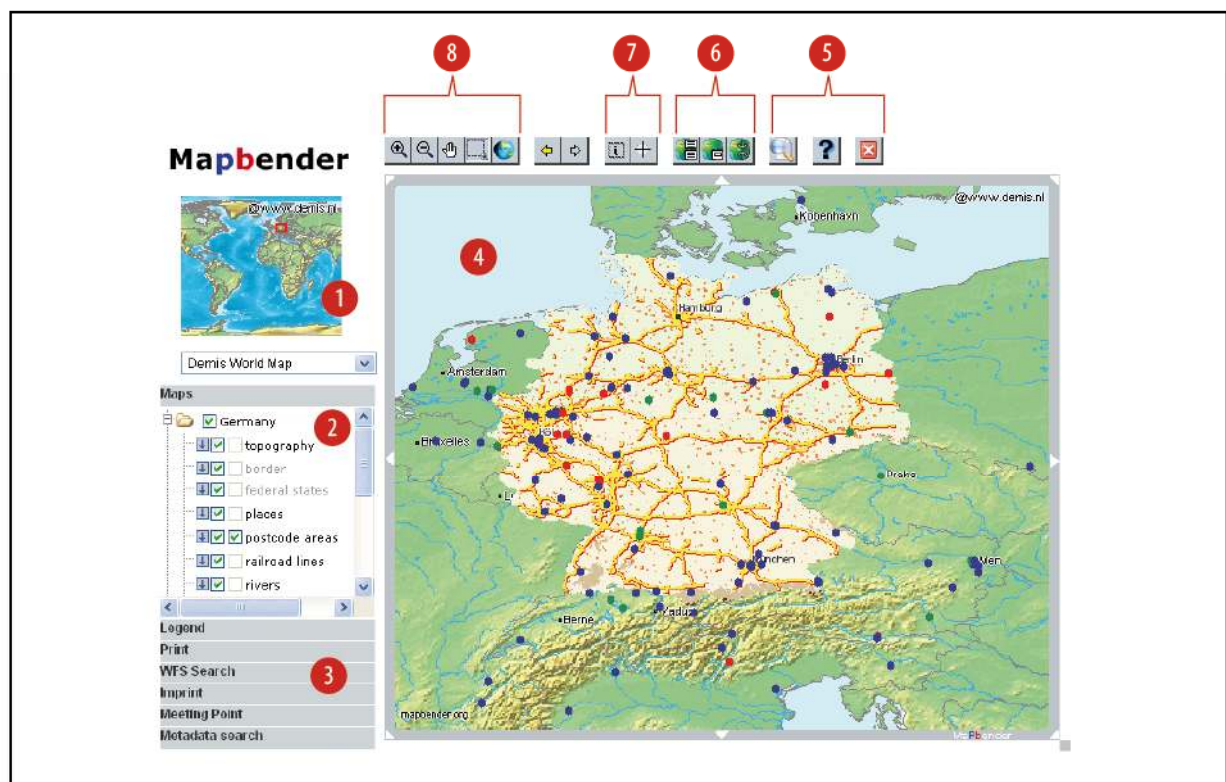


Abbildung 13-10: Aufbau einer Mapbender-Oberfläche

Die in der Abbildung gekennzeichneten Bereiche werden hier kurz erläutert.

1. Die Übersichtskarte zeigt das Gesamtprojekt und einen roten Kasten, der die Position der Hauptkarte (4) anzeigt. Die Karte ist interaktiv, ein mit der Maus aufgezo-
genes Fenster wird sofort in der Hauptkarte (4) angezeigt. Eine alternative
Übersichtskarte (in *gui2* enthalten) zeigt immer einen größeren Ausschnitt der
Hauptkarte und verändert sich relativ dazu.
2. Die Ebenenübersicht zeigt die Ebenen der eingebundenen Kartendienste. Diese
Liste kann über die Administration angepasst werden. Einige Ebenen sind mit
grauer Schrift hinterlegt. Dies ist ein Hinweis darauf, dass sie in der aktuellen

Ansicht nicht dargestellt werden, weil der Maßstab zu klein oder zu groß ist. Andere Ebenenübersichten (oder auch »GDE« für Geo Data Explorer) bieten andere Funktionen oder verhalten sich anders.

3. Der linke Bereich unter der Übersichtskarte besteht aus einzelnen Registerkarten, die durch Klick auf den Titel geöffnet und geschlossen werden können. Hier finden sich weitere Elemente wie die Legende und das Impressum sowie zusätzliche Funktionen wie ein Druckmodul, eine Metadatensuche und ein Modul, um einen Treffpunkt zu erzeugen. Diese Liste, deren Beschriftung und die Inhalte können erweitert und verändert werden.
4. Die Hauptkarte zeigt die geladenen Dienste und aktivierten Ebenen. Hier können beliebig viele Karten übereinandergelegt werden. Sie werden später feststellen, dass Karten unterschiedlich schnell geladen werden und sich manchmal gegenseitig verdecken. Das liegt daran, dass die voreingestellten Dienste von OGC WMS-Kartendiensten bezogen werden, die auf der ganzen Welt verteilt sind und unterschiedlich schnell antworten.
Die Reihenfolge kann sowohl vorab über die Administration festgelegt als auch nachträglich manuell verändert werden. Soll die Reihenfolge unveränderlich bleiben (z.B. um zu verhindern, dass ein Sachverhalt falsch dargestellt werden kann), kann dies auch unterbunden werden.
5. In Bereich 5 finden sich systemnahe Funktionen wie eine Onlinehilfe, der Abmeldeschalter und eine Metadatenanzeige.
6. Diese drei Knöpfe enthalten Funktionen zur Verwaltung der geladenen Ebenen. Hier können neue Dienste in das Kartenfenster (4) und die Ebenenübersicht (2) hinzugeladen werden, bestehende können entfernt oder die Reihenfolge und das Bildformat aller Dienste können verändert werden.
7. Der linke Knopf ermöglicht die Abfrage von Informationen zu Objekten in der Karte. Zunächst muss in der Ebenenübersicht (2) mit dem rechten Häkchen gewählt werden, welche Ebenen abgefragt werden sollen, dann wird der Infoschalter (7, links) angewählt und an die entsprechende Stelle in der Karte geklickt. Es liegen nicht überall Informationen vor.
Der rechte Knopf aktiviert die interaktive Koordinatenanzeige. Wenn Sie die Maus über das Kartenfenster bewegen, werden unter der Karte die Laufkoordinaten angezeigt. Die Genauigkeit der Koordinaten richtet sich nach der aktuellen Pixelgröße, je näher Sie hineinzoomen, umso genauer wird die Anzeige.
8. Im diesem Bereich befinden sich die üblichen Navigationsfunktionen: Hinein- und Herauszoomen, Wählen eines neuen Ausschnitts und Anzeigen des Gesamtprojekts.

Alle Elemente, die Sie hier sehen, können beliebig auf der Oberfläche angeordnet werden. Auf der Mapbender-Webseite finden Sie eine Übersicht weiterer Anwendungsoberflächen unter:

<http://www.mapbender.org/index.php/Gallery>

Wenn Sie Mapbender verlassen wollen, klicken Sie auf den Abmeldeschalter (5, ganz rechts), dadurch wird Ihre Sitzung beendet. Um eine andere Oberfläche zu starten, müssen Sie sich erneut anmelden.

Da das dauernde An- und Abmelden etwas mühsam ist, können Sie aus der Auswahl, die Sie nach der Anmeldung sehen, die gewünschte Oberfläche auch in einem neuen Fenster öffnen. Wählen Sie dazu aus dem Kontextmenü der Maus den Punkt *In neuem Fenster öffnen* oder *In neuem Tab öffnen*. Dann können Sie jederzeit wieder zur Auswahlliste in dem anderen Fenster zurückkehren. Das ist besonders hilfreich, während Sie eine neue Anwendung entwickeln oder wenn Sie die Anwendungen oft wechseln müssen.

Mapbender-Anwendungen entwickeln

Nachdem Sie sich mit einigen Oberflächen vertraut gemacht haben, werden wir nun Oberflächen verändern und eine eigene Anwendung entwickeln. Dazu wechseln Sie in die Administrationsoberfläche mit dem Namen *admin2_de*. Diese sehen Sie in Abbildung 13-11.

In der Administrations-GUI finden Sie eine lange Liste mit Funktionen. Schauen Sie sich dazu Abbildung 13-11 an. Jeder der Verweise auf der linken Seite öffnet in der rechten Hälfte des Browserfensters einen Detailbereich. Wenn Sie einen Detailbereich öfter benötigen, können Sie diesen auch in einem neuen Fenster öffnen, um zwischen verschiedenen Verwaltungsfenstern problemlos wechseln zu können.

Administrationsbereiche

Die Administration in Mapbender ist in fünf Teilbereiche untergliedert:

- WMS Verwaltung
- WMS Zuordnung
- GUI-Verwaltung
- Benutzerverwaltung
- Benutzerzugriff erteilen

Die erweiterte Administration bietet darüber hinaus noch die WFS- und Metadatenverwaltung.

Über die WMS Verwaltung werden die OGC-Kartendienste verwaltet, hier können OGC-Capabilities-Dokumente hochgeladen, aktualisiert und katalogisiert werden. Über die WMS Zuordnung werden die Karten den unterschiedlichen Anwendungen (GUI) zugeordnet. In der GUI-Verwaltung ist es möglich, neue Oberflächen zu erstellen und mit Elementen aus einer umfangreichen Liste vordefinierter Funktionen zu bestücken.

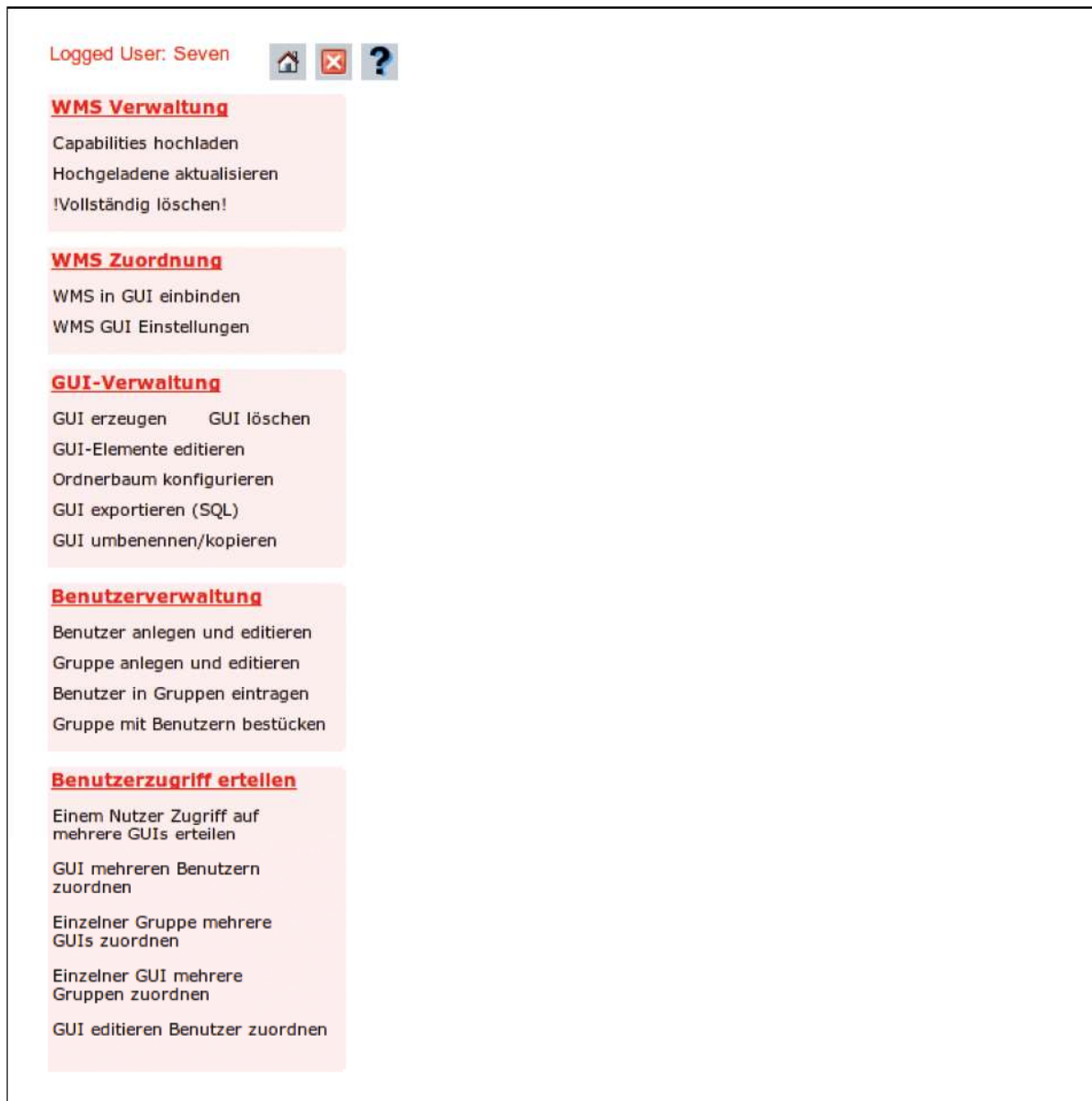


Abbildung 13-11: Mapbender-Administrationsoberfläche admin2_de

Über die Benutzerverwaltung können Benutzerkonten angelegt, verwaltet und Gruppen zugeordnet werden. Über den Bereich Benutzerzugriff erteilen werden die Berechtigungen an Gruppen und Benutzer vergeben.

GUI-Verwaltung

Der Bereich *GUI-Verwaltung* enthält alle Funktionen, um eigene Oberflächen zu erstellen und zu modifizieren. Hier finden Sie sechs Menüpunkte:

- *GUI erzeugen* – zum Erzeugen einer neuen, leeren Oberfläche.
- *GUI löschen* – hiermit löschen Sie eine Oberfläche vollständig, also Obacht.

- *GUI-Elemente editieren* – hier können die Eigenschaften aller Elemente der Oberfläche in Größe, Position und Funktionalität konfiguriert werden. Wenn Sie eigene Anwendungen entwickeln möchten, wird dies Ihr zweites Zuhause werden.
- *Ordnerbaum konfigurieren* – ermöglicht die manuelle Erstellung einer Baumstruktur. Hier können Sie z.B. einen englischsprachigen Kartendienst mit deutschen Beschriftungen versehen, Ebenen unterschiedlicher Dienste zusammenfassen und so weiter.
- *GUI exportieren (SQL)* – dieses Modul können Sie verwenden, um eine Oberfläche, die Sie z.B. auf dem OSGeo-Portal entwickelt haben, später in Ihre eigene Installation zu importieren oder auch um eine Sicherung anzufertigen.
- *GUI umbenennen/kopieren* – tut genau das, was man erwartet. Erweiternd können Sie im Detaildialog auswählen, ob beim Kopieren die Benutzerberechtigungen übernommen werden sollen oder nicht. Diese Funktion ist praktisch, um bestehende Oberflächen weiterzuentwickeln, ohne den aktuellen Tagesbetrieb zu stören.

GUI-Elemente editieren

Wählen Sie jetzt das Menü *GUI-Elemente editieren* aus. Im rechten Detailbereich wird eine Liste mit Namen von Oberflächen angezeigt, die Sie editieren dürfen. Falls Sie mit einem Konto auf einer Portalinstallation arbeiten, sehen Sie lediglich Oberflächen, denen Ihr Benutzername vorangestellt ist. Arbeiten Sie auf einer eigenen Installation und sind als Benutzer »root« angemeldet, sehen Sie dagegen alle Oberflächen der neu installierten Mapbender-Umgebung.

Wählen Sie jetzt aus der Liste die Oberfläche `<name>_gui` aus. In der Abbildung 13-12 setzt sich der Name aus dem Benutzernamen *Seven* und dem Suffix *gui* zusammen, also *Seven_gui*, in der Abbildung sehen Sie den Detailbereich.

In dieser Maske werden die HTML-Eigenschaften aller Objekte der GUI definiert. In der mittleren Spalte (blaue Schrift) finden Sie die Liste aller Elemente. Die Eigenschaften der Elemente können durch Anklicken des Optionsfelds links neben dem jeweiligen Elementnamen angezeigt werden. Im Beispiel in Abbildung 13-12 werden die Eigenschaften des Elements *mapframe1* angezeigt. Hierbei handelt es sich um das Hauptkartenfenster.

In der linken Hälfte des Fensters finden Sie alle HTML-Eigenschaften dieses Objekts, die wichtigsten davon werden hier beschrieben.

Eigenschaften	Beschreibung
Attributes:	Ebenfalls optional können hier weitere Attribute angegeben werden. In diesem Fall wird angegeben, dass für das Kartenfenster weder Scrollbalken noch eine Umrandung angezeigt werden soll. Diese würden lediglich die Navigation in der Karte behindern.
Left:	Angabe der Position vom linken Browserfenster in Bildschirmereinheiten. Diese Angabe erfolgt derzeit noch in Pixeln, soll aber in EM geändert werden, um die Barrierefreiheit zu verbessern.
Top:	Angabe der Position in Pixeln vom oberen Browserfenster (zukünftig in EM).
Width:	Angabe der Breite des Elements in Pixeln (zukünftig in EM).
Height:	Angabe der Höhe des Elements in Pixeln (zukünftig in EM).
Z-INDEX:	Angabe der Zeichenebene in der HTML-Seite.
Styles:	Dieses Attribut enthält Farb- und Schriftsatzangaben.
Content:	Im Content-Feld kann direkt JavaScript-Code hinterlegt werden.
Close-TAG:	Das schließende HTML-Tag (siehe oben).
JavaScript:	Verweis auf JavaScript-Code, der auch dynamisch über PHP geschrieben werden kann.
Modul:	Verweis auf JavaScript-Code.
Target:	Andere Oberflächenelemente, auf die von diesem Objekt Bezug genommen wird.
Requires:	Enthält einen Hinweis darüber, welche anderen Module von diesem Objekt benötigt werden. Dieses Feld dient nur der Information und wird nicht von der Software ausgewertet.
URL:	Link zur Onlinehilfe. Voreingestellt verweisen diese Links auf die Original-Mapbender-Onlinedokumentation. Sie können hier auch Links angeben, die auf Ihre eigenen Seiten verweisen.

Die vollständige Beschreibung aller HTML-Eigenschaften der Oberflächenelemente finden Sie in der Onlinedokumentation unter:

http://www.mapbender.org/Edit_GUI_Elements

Schaltflächen entfernen

Schaltflächen (engl. Buttons) sind kleine Bilder (z.B. GIF- oder PNG-Dateien), die in der HTML-Seite eingebettet sind und mit einem Ereignis versehen sind (z.B. einem Mausklick). Jede Schaltfläche kann in Mapbender einzeln positioniert werden (siehe die Eigenschaften Left und Top oben) oder auch ganz entfernt werden. Als Erstes werden wir den Knopf für die Anzeige der Metadaten der Kartendienste, die aktuell in Mapbender geladen sind (der dritte von rechts, ein Seitensymbol mit einer Lupe), entfernen.

Wählen Sie mit der Maus in der rechten Liste die Optionsschaltfläche neben dem Symbol mit der Seite und der Lupe aus. In der Liste werden nun die Eigenschaften dieses Objekts angezeigt. Um es komplett zu entfernen, können Sie nun auf die *Delete*-Schaltfläche klicken, damit wird diese Schaltfläche vollständig aus dieser Oberfläche gelöscht. Wenn Sie den Knopf später noch mal brauchen, können Sie den Button auch temporär entfernen, indem Sie die Eigenschaft On/Off auf 0 setzen. Damit Änderungen in den Eigenschaften der Funktion wirksam werden, müssen sie in die Datenbank zurückgeschrieben werden. Das geschieht durch Anklicken des Knopfs *save* ganz oben.

Wechseln Sie jetzt in die Oberfläche und laden Sie die Seite erneut (je nach Browser durch die Tastenkombination Strg-R oder durch einen Klick auf das Symbol *Seite erneut laden*). Die Schaltfläche mit der Seite und der Lupe wird jetzt nicht mehr angezeigt, stattdessen wird nur noch der graue Hintergrund angezeigt. Diesen können wir ebenfalls entfernen, indem wir das Element `layout_metadata` auswählen und die Eigenschaft `On/Off` auf 0 (null) setzen. Speichern nicht vergessen! Laden Sie die Oberfläche erneut und begutachten Sie das Ergebnis. Jetzt sind sowohl die Schaltfläche als auch der Hintergrundschatten nicht mehr zu sehen.

Eigenschaften von Schaltflächen ändern

Als Nächstes ändern wir den ToolTipp-Text der Schaltfläche *help*. Dieser wird angezeigt, wenn man die Maus einen Moment über dem entsprechenden Element stehen lässt. In der ausgelieferten Version steht da lediglich *help*. Das können wir ändern, indem wir in der rechten Liste das Element *help* (mit dem Fragezeichensymbol) auswählen. In den HTML-Eigenschaften *Attributes* gibt es mehrere Parameter, den Eintrag für *title* ändern Sie z.B. in: »Hier finden Sie die Onlinehilfe zu Mapbender«:

```
onmouseover = "mb_regButton('init_help')"  
title="Hier finden Sie die Onlinehilfe zu Mapbender"
```

Der andere Parameter definiert ein weiteres Ereignis, das eintritt, wenn Sie die Maus über dieses Feld bewegen, es steuert den Onmouseover-Effekt. Der Wert `mb_regButton('init_help')` ist ein Mapbender-JavaScript-Funktionsaufruf, der ebenfalls in dieser Seite geladen ist (sie werden durch das vorangestellte `mb_` gekennzeichnet).

Kartengröße ändern und fixieren, Rahmen entfernen

Um die Größe des angezeigten Kartenausschnitts zu verändern, wählen Sie aus der Liste das Element *mapframe1* aus, geben für die Parameter *Width* und *Height* andere Werte ein, z.B. 1000 und 800, und speichern diese. Anschließend wird das Kartenfenster wesentlich größer angezeigt. Den Rahmen um die Karte können Sie entfernen, indem Sie den Parameter `On/Off` für das Element *navFrame* auf 0 (null) setzen. Die Karte hat jetzt keinen Rahmen mehr.

Unten rechts an der Karte ist noch ein kleines graues Viereck zu sehen, darüber kann der Benutzer den Kartenausschnitt auf der HTML-Seite mit der Maus per Drag-and-Drop vergrößern oder verkleinern. Wenn Sie das unterbinden möchten, weil Ihre Seite nicht mehr Platz bietet außer für einen bestimmten Wert, können Sie auch diese Funktion entfernen, indem Sie für das Element *dragMapSize* den Parameter `On/Off` auf den Wert 0 (null) setzen. Wenn Sie die Anwendung neu laden, sehen Sie das Ergebnis.

Neue Oberflächen erstellen

Auf die gleiche Art und Weise, wie Sie Elemente aus der Anwendung entfernt (oder dort deaktiviert) haben, können Sie sich auch neue Anwendungen zusammenstellen. Im folgenden Abschnitt werden wir eine ganz einfache Kartenoberfläche zusammenstellen.

Zunächst erstellen wir eine neue Anwendung. Wählen Sie dazu im Administrationsbereich *GUI-Verwaltung* den Verweis *GUI erzeugen*. In der rechten Hälfte des Browserfensters werden zwei Felder angezeigt, *Name:* und *Description:*. Geben Sie hier den Namen Ihrer neuen Anwendung an. Dieser Name wird später auch im Titel des Browsers angezeigt.

Zunächst hat diese Oberfläche kein einziges HTML-Element, es ist noch völlig leer. Das sehen Sie, wenn Sie in der Administration in den Bereich *GUI-Elemente editieren* wechseln und die *gui* mit ihrem neuen Namen auswählen. Die rechte Hälfte ist komplett leer.

Oberflächenelemente hinzuladen

Das Erste, was eine Anwendung immer benötigt, ist das *Body*-Tag. Es ist auch das einzige Element, das zwingend erforderlich ist. Sie könnten jetzt alle Eigenschaften dieses Elements manuell eintragen, aber das wäre etwas mühsam. Stattdessen können Sie sich Elemente aus anderen Oberflächen kopieren. Wählen Sie dazu in der rechten oberen Hälfte des Fensters aus der Auswahlliste die Oberfläche *gui* aus. Jetzt werden alle Elemente dieser Oberfläche in der rechten Liste angezeigt. Sie sind grau hinterlegt, um deutlich zu machen, dass die Elemente zur Kopiervorlage gehören und nicht zur eigenen Oberfläche!

Wählen Sie aus der Liste das Element *body* aus. Die Eigenschaften werden jetzt in der linken Liste angezeigt. Beachten Sie unbedingt, dass sie jetzt noch nicht gespeichert sind! Denken Sie immer daran zu speichern, bevor Sie ein anderes Element der Anwendung auswählen.

Fahren Sie fort, indem Sie als Nächstes nacheinander die Schaltflächen *zoomIn1* (speichern) und *zoomOut1* (wieder speichern) anwählen. Als Letztes wählen Sie noch das Element *mapframe1* aus und speichern es ebenfalls. Wählen Sie in der GUI-Auswahlliste oben rechts jetzt wieder Ihre eigene GUI aus. In der rechten Hälfte werden nun die Elemente angezeigt, die Sie ausgewählt haben.

Jetzt können Sie Ihre neue GUI bereits starten (Abbildung 13-13).

Die neue Anwendung wird in der Auswahlliste angezeigt, die Sie direkt nach der Anmeldung sehen. Sie können aber auch einfach oben auf den Knopf *show* klicken, der die Anwendung in einem neuen Fenster öffnet (Sie müssen gegebenenfalls die Einstellungen Ihres Browsers ändern, um das tun zu können).

Ihre erste GUI sieht noch recht nackt aus, sie enthält nämlich gar keine Karten. Diese kommen im nächsten Schritt hinzu.

Kartendienst in Anwendungen einbinden

Es stehen Ihnen zwei Wege zur Verfügung, um Ihre Anwendung mit Karten zu bestücken. Entweder Sie laden einen WMS über den *GetCapabilities*-Request in die GUI, oder Sie nutzen die WMS, die Mapbender bereits kennt, und bestücken die Anwendung mit einem dieser Dienste.

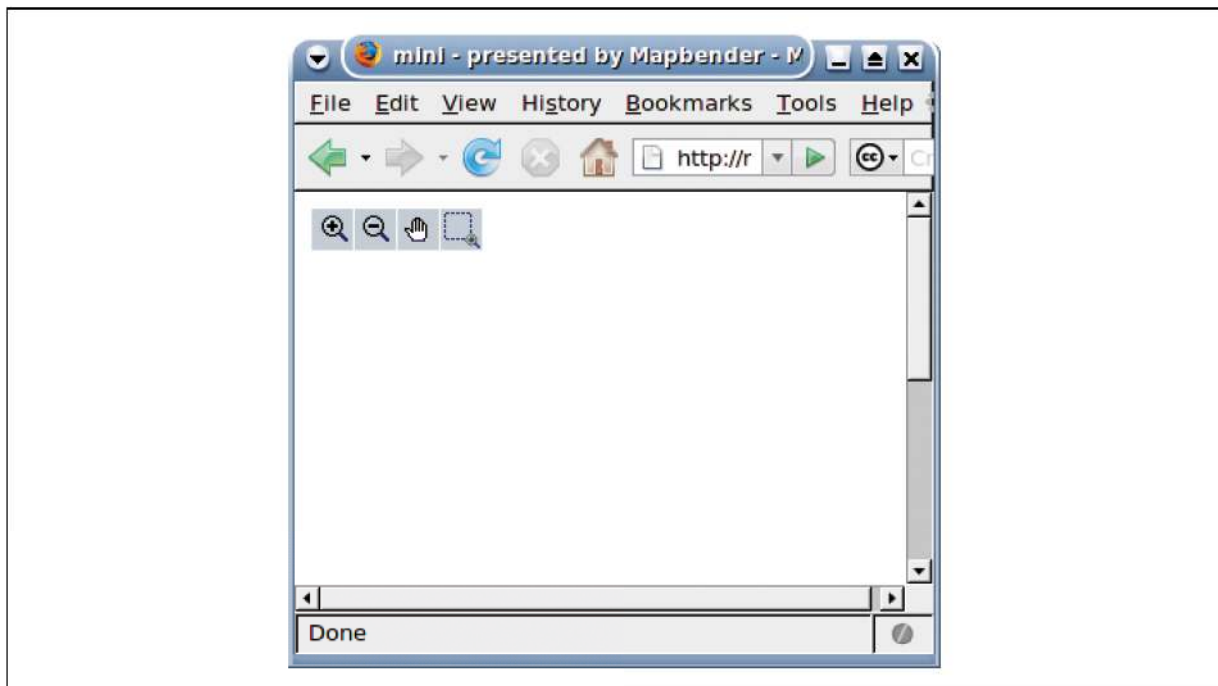


Abbildung 13-13: Eine minimalistische Kartenanwendung

Um einen WMS zu laden, gehen Sie in die Administrationsoberfläche *admin2_de* und laden im Bereich *WMS Verwaltung* über den Link *WMS laden* Ihren WMS. Wählen Sie Ihre GUI per Klick aus. Fügen Sie in das Textfeld *Link to WMS Capabilities URL* eine gültige Adresse eines WMS-Diensts ein. Über den Button können Sie dann den Dienst in die ausgewählte GUI laden. Schauen Sie sich dazu Abbildung 13-14 an. Den Aufbau des Diensts können Sie sich anschließend über den Administrationseintrag *WMS GUI Einstellungen* anzeigen lassen.

Der zweite Weg ist die Nutzung bestehender WMS. Wählen Sie in der Administrationsoberfläche *admin2_de* aus dem Bereich *WMS Zuordnung* den Verweis *WMS einbinden* aus. Die Liste zeigt alle Anwendungen, die Ihrem Benutzerkonto zugeordnet sind. Wenn Sie die neu erzeugte GUI auswählen, wird rechts eine Liste der eingebundenen Kartendienste angezeigt (diese ist zunächst noch leer), und im unteren Bereich werden alle Kartendienste angezeigt, die Sie verwenden können. Jetzt können Sie einen Dienst Ihrer Wahl (z.B. »Forstliche Verwaltungsgrenzen NRW«) anklicken (Abbildung 13-15).

In der rechten Hälfte sehen Sie eine Liste von Anwendungen, in denen dieser Dienst bereits verwendet wird. Durch Auswahl einer der angezeigten Anwendungen übernehmen Sie deren Voreinstellungen für Ihre neue Anwendung, z.B. die der Anwendung *gui1*. Mapbender fragt nach, ob Sie diese Einstellungen übernehmen möchten, bestätigen Sie die Auswahl mit OK. Wenn Sie jetzt Ihre Anwendung erneut laden, werden Sie unterhalb der zwei Zoomschaltflächen eine Karte sehen (Abbildung 13-16).

Ihre erste ganz einfache Anwendung ist fertig. Jetzt können Sie mit den Einstellungen experimentieren, neue Schaltflächen hinzufügen und diese anders positionieren.

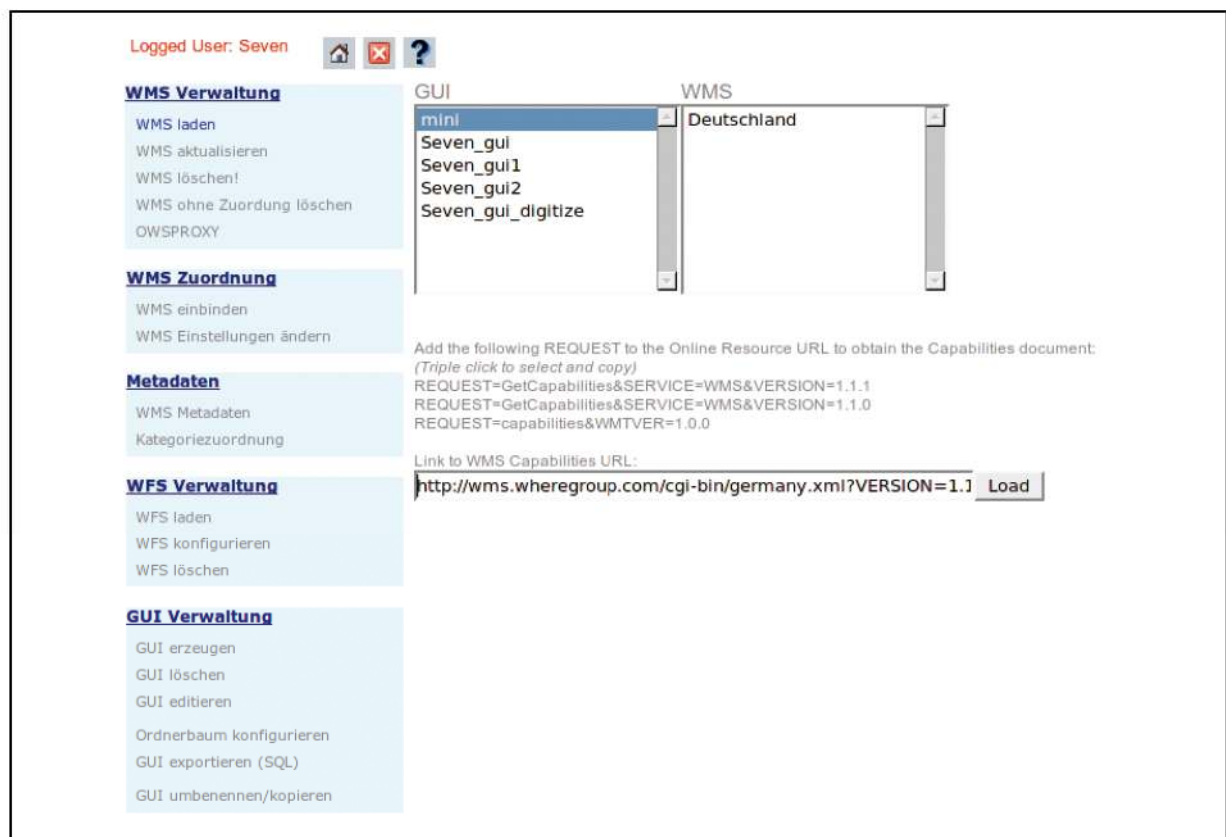


Abbildung 13-14: Laden eines WMS-Diensts über die Administrationsoberfläche

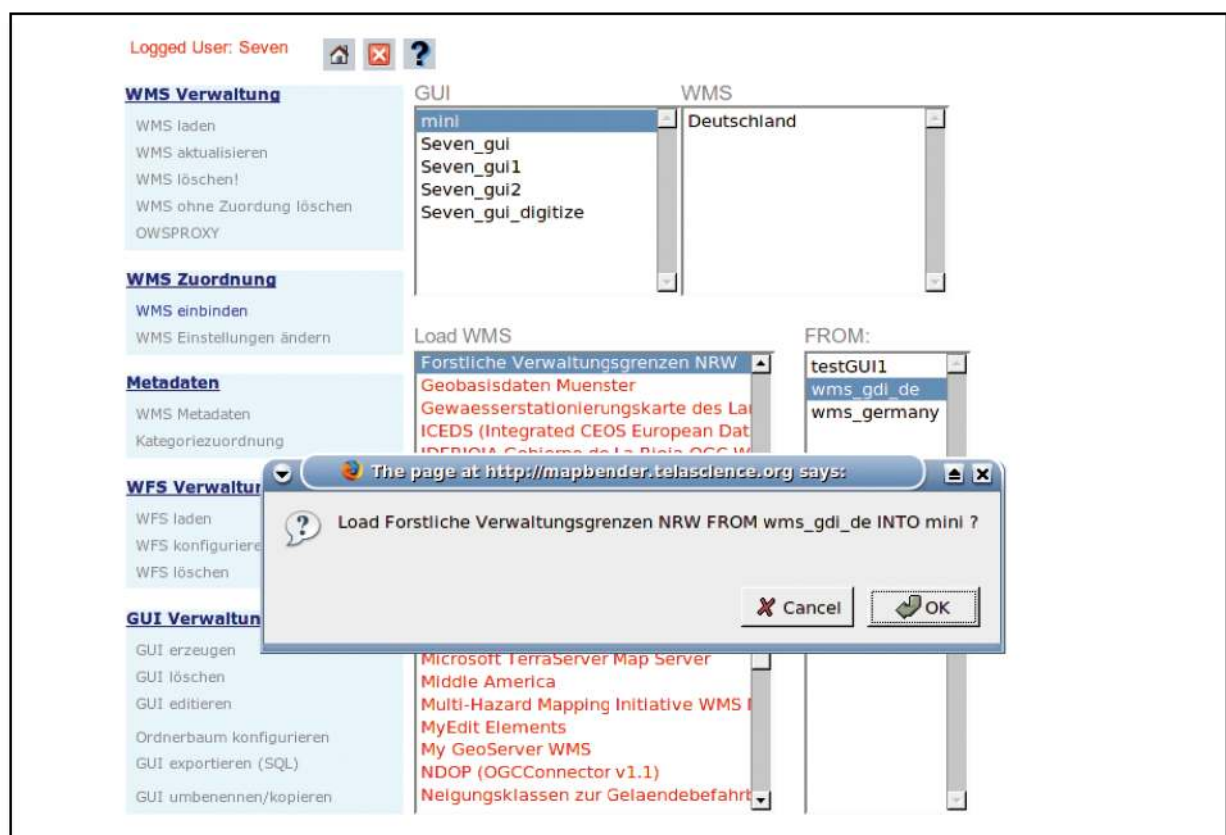


Abbildung 13-15: Sie können jetzt einen Dienst Ihrer Wahl anklicken.

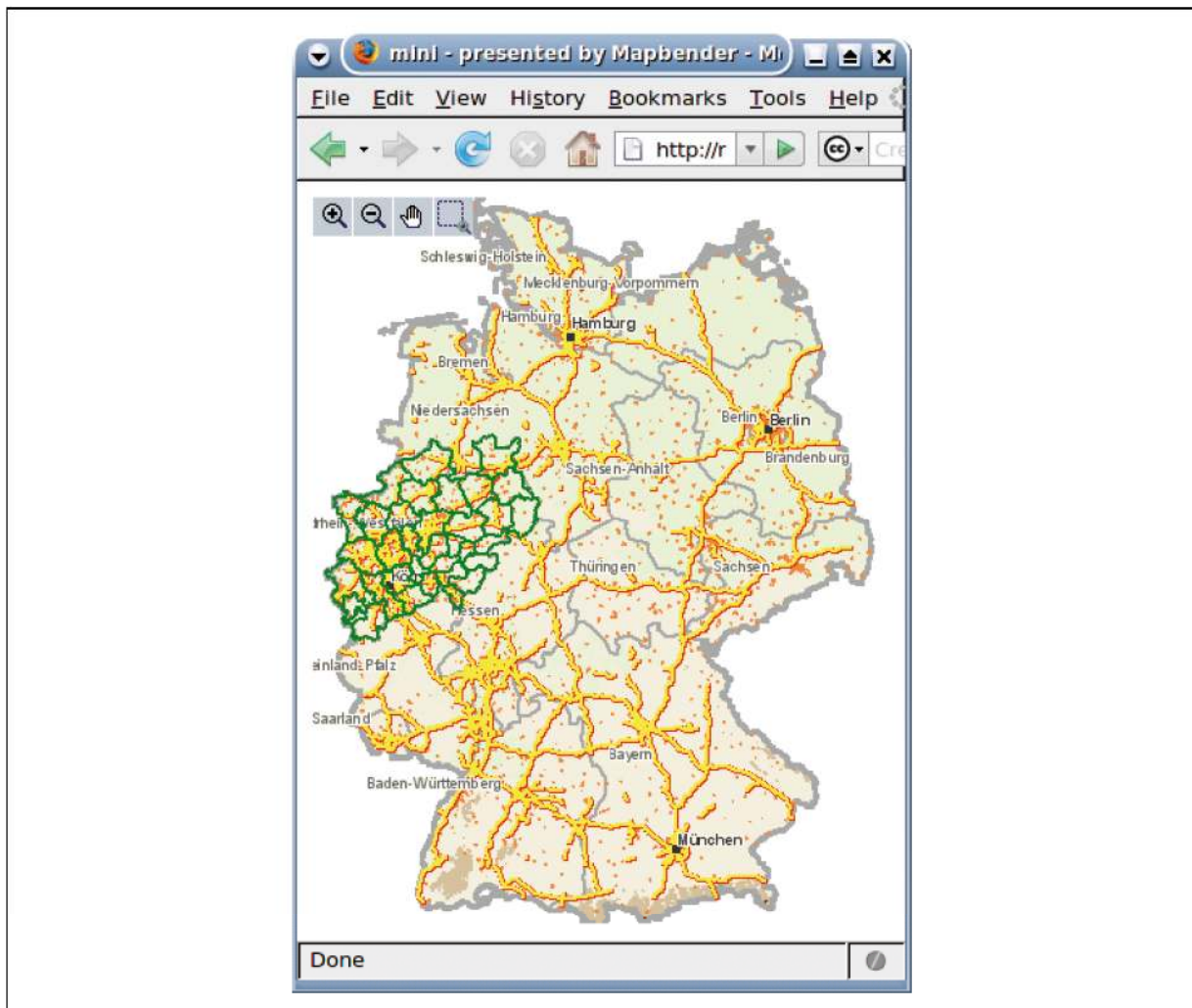


Abbildung 13-16: Ansicht der selbst erstellten Anwendung mit den Forstamtsgrenzen NRW

Mapbender-Anwendung in die eigene Webseite einbinden

Die neue Anwendung können Sie genau so wie OpenLayers über ein iframe in Ihre eigene Webseite einbinden. Dabei müssen Sie zusätzlich einen Benutzernamen und das Passwort als Parameter mit übergeben. Der parametrisierte Aufruf einer Mapbender-Anwendung in einem iframe sieht so aus (im HTML-Code ohne Umbrüche):

```
<iframe src='http://mapbender.telascience.org/current/frames/login.php?
    name=guest&password=guest&mb_user_myGui=mini'
    width='400' height='500' frameborder='0'>
</iframe>
```

Hierbei wird die Software Mapbender angewiesen, die GUI mini mit dem Benutzer guest und dem Passwort guest zu starten. Beachten Sie, dass dieses Passwort natürlich sichtbar in der HTML-Seite angezeigt wird. Dieser direkte Aufruf erfolgt also nicht durch die interaktive Anmeldung eines Benutzers am Portal, sondern durch Aufruf einer URL über einen allgemeinen Benutzer guest. Diesem Benutzerkonto dürfen selbstverständlich keine Anwendungen zugeordnet werden, die gesicherte Kartendienste enthalten.

Beispielanwendungen

Aktuelle Beispiele von Mapbender-Anwendungen finden Sie in der Projektgalerie der Mapbender-Webseite. Folgen Sie dem Link *Gallery* auf der Seite:

<http://www.mapbender.org/Gallery>

Stadtplandienste

Mapbender wird oft gleichzeitig als Stadtplandienst und verwaltungsinternes Geoportal genutzt, beispielsweise in:

- Bonn: <http://www.bonn.de>
- Mainz: <http://www.mainz.de>
- Nürnberg: <http://statistik.nuernberg.de/geoinf/spn/startseite.htm>
- Regensburg: <http://stadtplan.regensburg.de/>
- Wesseling: <http://www.wesseling.de/stadtinfos/stadtplan/index.php>
- ... und weitere online über die Mapbender-Galerie

Alle Stadtplandienste zeichnen sich durch ähnliche Funktionen und einen ähnlichen Aufbau der Anwendung aus. Eine Straßensuche mit tagesaktuell hausnummerngenauen Daten ist die wichtigste Komponente. Bei besonders gut gepflegten Webseiten bietet die Straßensuche auch Ergebnisse zu historischen Namen oder ist gleichzeitig mit der allgemeinen Suche kombiniert. Die Straßensuche ist wahrscheinlich eines der wichtigsten interaktiven Werkzeuge eines Stadtplandiensts.

Es gibt meist zwei oder drei Basiskarten, die wechselseitig eingeblendet werden können. Das Luftbild ist die mit Abstand beliebteste dieser Ebenen und wird statistisch am meisten abgerufen. Es ist wichtig, darauf zu achten, dass dieser Dienst sehr stabil ist, schnell antwortet und aktuelle Daten beinhaltet. Der Stadtplan ist maßstabsabhängig und zeigt in Übersicht das überregionale Verkehrsnetz und größere Ortsnamen. Je tiefer der Anwender hineinzoomt, um so nach detailreicher werden die Inhalte. Die Baublockebene entspricht ungefähr dem gewohnten Darstellungsmaßstab eines detaillierten Stadtplans. In manchen Fällen (Mainz oder Wesseling) werden in einem noch größeren Maßstab Gebäudeumrisse und Topografie dargestellt.

Eine eingescannte topografische Karte wird ebenfalls von einigen Diensten bereitgestellt. Diese Art von Karten eignet sich jedoch nicht besonders gut für die zoomstufenfreie Darstellung in digitaler Form.

Inhaltlich bieten Stadtplandienste mehrere Dutzend oder Hunderte von Informationsebenen, die als Ebenen mit Symbolen überlagert dargestellt werden. Die Informationen können von internen und externen Diensten beliebiger technischer Herkunft bereitgestellt werden. Oft wird im Katasteramt eine andere Software eingesetzt als in der Stadtplanung oder im Umweltamt. Da in diesen Beispielen jedoch alle Dienste die OGC-

Standards WMS und WFS erfüllen, können sie gemeinsam im Stadtplandienst zu einer homogenen Anwendung zusammengeführt werden.

Manche Stadtplandienste enthalten auch hochspezielle Karten wie Schallemissionskarten, die den Straßenverkehr bei Tag und bei Nacht abbilden (Stadt Mainz), oder historische Karten von 1630, 1811, 1860 oder auch eine eigene Ebene mit den Wirtshäusern von 1800 (Nürnberg).

Die Informationen können über eine Baumstruktur gruppiert in Ebenen an- und ausgeschaltet werden. Alternativ werden die einzelnen Ergebnisse einzeln dargestellt. Mauseffekte helfen, die gefundenen Objekte zu identifizieren oder um weitere Informationen dazu anzuzeigen. Wie solche Oberflächen anwenderfreundlich und ansprechend gestaltet werden, zeigen die Anwendungen der Suchmaschinenbetreiber Google Maps oder Yahoo! Maps.

Doch diese Art der zielorientierten Bereitstellung ist nur ein Teil des gesamten Spektrums. Auch die Katasterauskunft nutzt die Mapbender GUI, um rechtskonforme Katasterauszüge zu erstellen. Die gleichen gesicherten Dienste werden genutzt, um Katasterauszüge digital bereitzustellen. Zugriffe werden protokolliert und mit einer Gebührenanwendung weiterverarbeitet. Die Benutzerverwaltung erfolgt über Mapbender-Gruppen, OWS-Tickets regeln die Zugriffe auf gesicherte Dienste. Die gesamte Anwendung wird mit webbasierten Oberflächen über ein verschlüsseltes Protokoll und herkömmliche Webbrowser verwaltet und gepflegt.

Geoportale

Der Begriff Webportal beschreibt ein System aus Anwendungen, das Prozesse und Dienste integriert. Des Weiteren stellt es Funktionen zur Personalisierung, Sicherheit, Navigation und Benutzerverwaltung bereit sowie die Suche und Präsentation von Informationen. Ein Geoportal ist der Zugang zu einer dienstorientierten Geodateninfrastruktur. Die Aufgabe eines Geoportals besteht darin, seinen Anwendern möglichst geeignete Mittel zur Suche und Recherche nach Geodaten zu ermöglichen, dazu zählt auch eine Suche über räumliche Parameter. Das Ergebnis einer Recherche kann eine Karte, die Ebenen einer Karte, ein Ausschnitt einer Karte oder auch ein einzelnes Objekt sein.

Oft ist das Ergebnis nur der Verweis auf einen Dienst, der dann mit einer anderen Software weiterverarbeitet wird. Für diese Art von Daten gibt es ISO-Normen aus der 19000-Serie, die im Technical Committee 211 in Zusammenarbeit mit dem OGC entstanden sind.

Beispiele für Geoportale:

- Geoportal Rheinland-Pfalz (<http://www.geoportal.rlp.de/>)
- PortalU (<http://www.portalu.de/>)
- Geo Datenportal der Stadt Remscheid (<http://geoportal.remscheid.de/>)

Im Geoportal des Landes Rheinland-Pfalz wird Mapbender nicht nur zur Anzeige und Überlagerung der Kartenwerke verwendet, sondern vor allem für die Verwaltung der Kartendienste und deren Metadaten. Das Webportal ist mit der CMS-Software TYPO3 implementiert, die über serverseitige Skripten mit Mapbender kommuniziert, sodass das Portal wirkt wie eine homogene Software.

Das Geo Datenportal Remscheid (siehe Abbildung 13-17) stellt den Zugriff auf seine Dienste über eine schlichte, aber informative Eingangsseite bereit, die in kleinen Vorschaubildern Ausschnitte der Karten zeigt.



Abbildung 13-17: Das Geoportal der Stadt Remscheid

Ein Klick auf das jeweilige Vorschaubild öffnet die Anwendung über den hinterlegten URL-Verweis (Abbildung 13-18). Die Anwendungen sind mit unterschiedlichen Oberflächen und Funktionen ausgestattet, nutzen aber die gleichen Kartendienste.

Die Anwendungen bieten auch die Möglichkeit, Kartendienste der angrenzenden Städte, Kreise und landesweite Daten einzubinden. Damit integriert sich das Geodatenportal Remscheid optimal in die landesweite Geodateninfrastruktur NRW und die länderübergreifende GDI-DE.

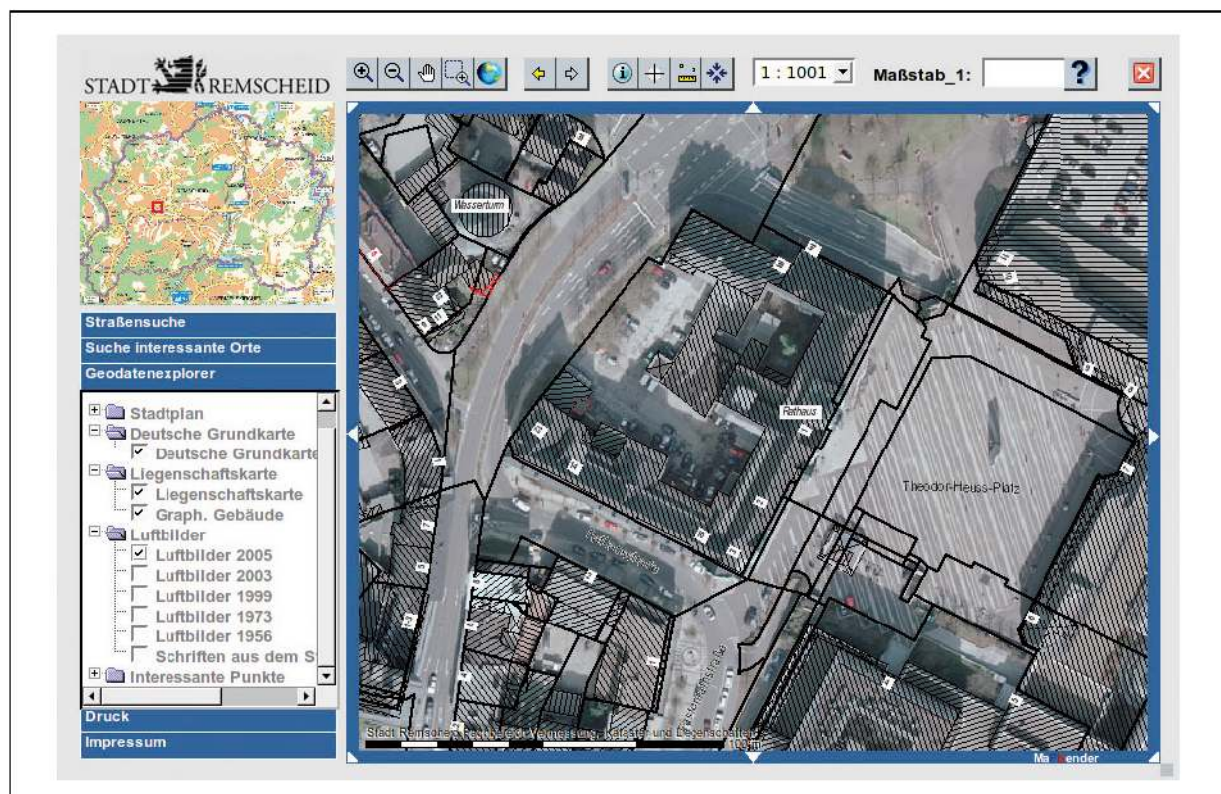


Abbildung 13-18: Die Geoportalanwendung Liegenschaftskarte der Stadt Remscheid

PostgreSQL/PostGIS – Einsatz von räumlichen Datenbanken

Dieses Kapitel richtet sich an Benutzer, die eine relationale Datenbank zur Erzeugung, Verwaltung und Bearbeitung von Geodaten nutzen möchten. Räumlich erweiterte Datenbanken verbinden die Leistungsfähigkeit relationaler Datenbanken bei alphanumerischen Daten mit geografischen Daten. Dieses Kapitel zeigt am Beispiel der Software PostgreSQL und PostGIS, wie räumliche Daten in eine Datenbank importiert, eigene Daten erstellt und manipuliert werden, wie Daten abgefragt und in eine Karte eingebunden werden können. Darüber hinaus geht das Kapitel darauf ein, wie Daten aus PostgreSQL/PostGIS in verschiedenen Anwendungen genutzt und in andere Formate überführt werden können.

Einführung in PostgreSQL/PostGIS

PostGIS ist ein räumlicher Aufsatz für die leistungsstärkste Open Source-Datenbank PostgreSQL. Der Fokus dieses Kapitels liegt darauf, den Umgang mit den tabellarischen Komponenten von PostgreSQL in Verbindung mit den räumlichen Funktionen von PostGIS zu vermitteln.



Für speziellere Informationen zu PostgreSQL empfehle ich Ihnen das Buch *Practical PostgreSQL* (<http://www.oreilly.de/catalog/ppostgresql/>) und die offizielle Website zu PostgreSQL (<http://www.postgresql.org>). Hier finden Sie einen Link (<http://www.postgresql.org/docs/books>), der Hinweise zu Büchern enthält. Beachten Sie auch den Link <http://www.postgresql.org/files/documentation/books/pghandbuch/html/>, der auf das deutschsprachige PostgreSQL-Handbuch von Peter Eisentraut verweist.

PostGIS ist eine Open Source-Erweiterung zu PostgreSQL, die von Refrations Research entwickelt wurde. Auf der Website (<http://postgis.org>) gibt es eine Zusammenfassung des Projekts.



PostGIS erweitert die objektrelationale Datenbank PostgreSQL um die Unterstützung räumlicher Objekte. Durch diese Erweiterung kann der PostgreSQL-Server als räumliches Datenbank-Backend für Geografische Informationssysteme (GIS) eingesetzt werden. PostgreSQL/PostGIS wird häufig im professionellen Umfeld genutzt, um ähnliche Aufgaben zu erfüllen wie die proprietären Werkzeuge ESRI ArcSDE sowie die Datenbank Oracle Spatial.

Was ist eine räumliche Datenbank?

Eine Datenbank ist ein Werkzeug zur Speicherung und für den Zugriff auf Informationen, die in tabellarischer Form abgelegt werden. Datenbanken speichern die Daten als Felder und Datensätze (Spalten und Zeilen oder Attribute und Werte). Abhängig von der verwendeten Datenbank können die Felder unterschiedliche Datentypen enthalten. Meistens handelt es sich hierbei um numerische Datentypen und Texte. Ein großer Vorteil von Datenbanken gegenüber unstrukturierten Dateien besteht in der Möglichkeit, die Daten nach Informationen, die auf bestimmte Suchkriterien passen, abzufragen. *Relationale Datenbanksysteme* ermöglichen es, Verknüpfungen zwischen Tabellen herzustellen. Hierbei erfolgt die Verknüpfung über Informationen, die in beiden Tabellen auftauchen. Weitere Vorteile sind Mehrbenutzerfähigkeit, Redundanzfreiheit und Konsistenzprüfung.



Weitere Informationen zu relationalen Datenbankmanagementsystemen finden Sie unter <http://de.wikipedia.org/wiki/RDBMS>.

Eine *räumliche Datenbank* ist im Prinzip genauso aufgebaut, ist zudem aber in der Lage, räumliche Informationen zu speichern. Hierbei hat der Begriff der *räumlichen Datenbank* für unterschiedliche Datenbanken und GIS-Produkte leicht abweichende Bedeutungen. So ist z.B. die ESRI Spatial Database Engine (SDE) keine räumliche Datenbank, sondern fungiert als Schnittstelle zwischen Clientprogrammen und herkömmlichen Datenbanken. Diese Schnittstelle ermöglicht es, räumliche Daten im SDE-Format in der Datenbank zu speichern. Um die Daten laden und bearbeiten zu können, wird ein ESRI-Produkt oder Zugang zu einem ESRI-Service benötigt.

Andere Datenbanken verfügen ebenfalls über räumliche Erweiterungen. Für Oracle gibt es die Erweiterung Oracle Spatial, IBM bietet den Spatial Extender für die DB2-Datenbank an, und auch Microsoft SQL Server 2008 verfügt über einen Spatial-Aufsatz. Auch für die Open Source-Datenbank MySQL gibt es eine räumliche Erweiterung. Alle diese Erweiterungen ähneln PostGIS. Sie ermöglichen die Speicherung von räumlichen Daten und den Zugriff auf diese mit Datenbankwerkzeugen, wobei keine spezielle GIS-Software für den Zugriff oder die Manipulation der Daten benötigt wird.

Die herausragenden Fähigkeiten von PostgreSQL/PostGIS sorgen dafür, dass sich viele Unternehmen für den Einsatz dieser Lösung als räumliches Datenbankmanagementsys-

tem entscheiden. Immer mehr Programme unterstützen PostGIS, hierzu gehören Desktop-GIS wie JUMP, QGIS, uDig, gvSIG sowie MapServer, GeoServer, Mapbender FME und viele mehr. PostGIS stellt eine Vielzahl von Funktionen für den Umgang mit räumlichen Daten bereit, ohne dass hierfür ein GIS-Client notwendig ist. Dies macht PostGIS zusätzlich attraktiv.

PostGIS bietet die Möglichkeit, räumliche Daten zu speichern und zu bearbeiten. Es handelt sich also nicht um einen reinen Datenspeicher, sondern auch um eine Umgebung für die Analyse räumlicher Daten.

Das OGC hat eine Spezifikation für die Speicherung, Bearbeitung und Abfrage von räumlichen Daten von SQL-Datenbanken entwickelt, die Simple Features Specification for SQL (SFS). Die Spezifikation können Sie über den folgenden Link <http://www.opengeospatial.org/standards/sfs> kostenfrei herunterladen. Die OGC-Standards sind eine wesentliche Voraussetzung für die Interoperabilität räumlicher Daten. Die Einhaltung offener Standards ist Voraussetzung, wenn Daten gemeinsam genutzt oder ein offener Zugang ermöglicht werden soll.

Hierbei ist PostGIS nach den Angaben der OGC-Ressourcenseite (siehe <http://www.opengeospatial.org/resources/products>) eine der robustesten Implementierungen der SFS-Standards. Da PostGIS die SFS-Standards vollständig umsetzt, können Sie über die gesamte Dauer Ihres Projekts mithilfe von standardisierten Funktionen arbeiten, ohne auf Umwege und Einschränkungen angewiesen zu sein. Viele andere Lösungen implementieren dagegen nur Teile der SFS-Standards.

Über PostgreSQL/PostGIS erfolgt die Darstellung der räumlichen Daten (als Rückgabe auf eine Datenbankabfrage) auf sehr intuitive Weise, wie Sie in den Beispielen dieses Kapitels noch sehen werden. Abhängig von Ihren Anforderungen gibt es eine Reihe verschiedener Methoden, mit denen Sie auf die Geometrien zugreifen können. Die Abfragen können über kommandozeilenbasierte Tools, Werkzeuge mit einer grafischen Benutzeroberfläche oder über ein Desktop-GIS erfolgen. Sie sind nicht von der Verwendung proprietärer Werkzeuge abhängig, um an die Informationen zu gelangen.

Im Zusammenhang mit PostGIS ist das Projekt *pgRouting* zu nennen, über das die Datenbank mit Routing-Funktionalität versehen wird. Weitere Informationen zu *pgRouting* finden Sie auf der Projektseite unter <http://pgrouting.postlbs.org/>.

Vorteile eines serverbasierten GIS

PostgreSQL ist ein objektrelationales Datenbankmanagementsystem. Über einen Datenbankserver werden die Daten bereitgestellt. Bei einer Datenbankabfrage verarbeitet der Server den Request, stellt die Daten zusammen und gibt das Ergebnis an das abfragende Programm zurück. Anstatt alle Datensätze selbst zur Berechnung an den Client zu übertragen, werden die nötigen Berechnungen direkt auf dem Server durchgeführt, und Sie erhalten nur das Ergebnis zurück. Selbst bei der Rechenleistung, die eine Menge moderner Computer mittlerweile besitzen, sind viele PCs nicht dafür konzipiert, mit der intensiven Belastung durch räumliche Operationen umzugehen. Müssten die

Daten zur Verarbeitung durch Ihr Programm auf Ihrem Rechner erst über das Netzwerk verschickt werden, könnte das für Netzwerk und Clientprogramm einen großen Performanceverlust bedeuten.

Das gleiche Problem tritt auch bei GIS und der Verwaltung räumlicher Daten auf. Desktop-GIS sind sehr stark von der Leistungsfähigkeit des Arbeitsplatz-Rechners abhängig, auf dem sie lokal installiert sind. Einfache räumliche Fragestellungen können dabei ohne Weiteres verarbeitet werden, aber schon die Darstellung komplexer Geometrieobjekte kann sehr langsam sein. Komplexe räumliche Analysen auf großen Datenmengen sollten deshalb von einem Server durchgeführt werden.

Stellen Sie sich beispielsweise vor, Sie müssten, basierend auf einem gemeinsamen Attribut, eine große Anzahl von Polygonen miteinander verknüpfen. Das GIS-Programm lädt die benötigten Polygone vom Server als temporäre Datei in den Arbeitsspeicher Ihres Computers. Dieser Prozess kann schon einen Flaschenhals darstellen, weil hierfür große Datenmengen über das Netzwerk transportiert werden müssen. Beginnt dann die Verknüpfung der Polygone, werden große Mengen an Arbeitsspeicher und Rechenleistung beansprucht, nicht zu vergessen die Belastung der Festplatten beim Durchsuchen der Daten.

Möglicherweise ist das Desktop-GIS auch gar nicht in der Lage, die von Ihnen geforderten Analysen durchzuführen. Oft können Sie in einem solchen Fall entweder die benötigten Softwareerweiterungen kaufen oder die Daten mit einem anderen Programm verarbeiten. Hierbei können aber schnell die gleichen Probleme auftreten wie im vorherigen Beispiel: Der PC benötigt sehr viel Rechenleistung und Arbeitsspeicher für das Kopieren und Berechnen der Daten, und zusätzlich müssen die Daten oftmals in ein anderes Format konvertiert werden. Es entstehen große Mengen an temporären Daten, die nicht zentral verwaltet werden.

PostGIS nutzt dagegen die Vorteile einer serverbasierten Datenbank, indem es zahlreiche GIS-Funktionen direkt auf dem Server bereitstellt. PostGIS bietet also nicht nur die Fähigkeit, räumliche Daten zu speichern, sondern besitzt auch die Möglichkeiten der Datenbearbeitung, die sonst nur in Desktop-GIS-Programmen zu finden sind. Die Anforderungen an Clientprogramme werden durch die Fähigkeiten von PostGIS stark reduziert, da die Möglichkeiten des Servers genutzt werden können. Dies ist eine wesentliche Stärke von PostGIS.

Desktop-GIS-Programme können zukünftig vornehmlich die Aufgabe der Datenvisualisierung übernehmen, die GIS-Funktionalität wird dagegen durch die räumliche Datenbank bereitgestellt. Dieses Modell setzt sich immer mehr durch. Durch die offene Beschreibung der Funktionalitäten von PostGIS können Entwickler räumliche Abläufe in ihre eigenen Programme integrieren und diese damit erweitern.

Voraussetzung für die Nutzung von PostGIS ist eine Standard-Installation der Datenbank PostgreSQL. PostGIS besteht aus drei Komponenten:

PostGIS-Bibliotheken

Die wichtigste Bibliothek ist *libpostgis.so* bzw., auf Windows, *libpostgis.dll*. Diese Bibliothek ist die Schnittstelle zwischen den PostgreSQL-Funktionalitäten und den räumlichen Fähigkeiten von PostGIS.

PostGIS-SQL-Skript mit Funktionen und Datentypen

Mit dem Skript *lwpostgis.sql*, das SQL-Befehle beinhaltet, werden Hunderte von PostGIS-spezifischen Funktionen und Datentypen geladen, und es werden zwei Tabellen erzeugt. In älteren PostGIS-Versionen wurde hierfür das Skript *postgis.sql* verwendet.

Skript zur Unterstützung von Projektionen

Wenn Sie das SQL-Skript *spatial_ref_sys.sql* laden, werden Informationen über Koordinatenreferenzsysteme und Projektionen in der Datenbank gespeichert.

Diese Skripten sind plattformunabhängig und setzen sich aus SQL-Befehlen zusammen. Sie können PostGIS auch selbst kompilieren und um eigene Funktionen erweitern. Dies ist aber immer seltener erforderlich, weil schon zahlreiche Funktionen vorliegen.

Die Dokumentation auf der PostGIS-Webseite ist ein guter Einstieg und liefert eine ausführlichere Beschreibung, als dies hier möglich ist. Die Adresse lautet <http://www.postgis.org/docs/>.



Die Beispiele in diesem Kapitel basieren auf der PostgreSQL-Version 8.2 und PostGIS in der Version 1.3. Dies sind die derzeit (Stand 10/2007) aktuellen Versionen. In neueren Versionen können Änderungen auftreten, an die die folgenden Beispiele angepasst werden müssen.

Installation von PostgreSQL und PostGIS

Für PostgreSQL und PostGIS gibt es mehrere Installationspakete und binäre (vorkompilierte) Versionen für unterschiedliche Plattformen.

Aktuelle Informationen zur Installation von PostgreSQL und PostGIS finden Sie auf den offiziellen Projekt-Websites:

PostgreSQL (<http://www.postgresql.org>)

PostGIS (<http://www.postgis.org>)

Auf der PostgreSQL-Website gibt es ausführliche Beschreibungen zur Installation von PostgreSQL für verschiedene Betriebssysteme:

<http://www.postgresql.org/docs/techdocs.4>

Den Download-Bereich finden Sie unter:

PostgreSQL (<http://www.postgresql.org/download>)

PostGIS (<http://www.postgis.org/download>)

PostGIS für Windows

Seit der PostgreSQL-Version 8.0 liegt PostgreSQL auch für Windows vor und kann über einen Installer ganz einfach auf Ihrem Rechner installiert werden. Während der Installation können Sie die PostGIS-Unterstützung direkt auswählen, Sie können PostGIS aber auch nachträglich installieren.

Die offiziellen Installationspakete für PostgreSQL finden Sie unter <http://www.postgresql.org/ftp>. Der Installer liegt im Verzeichnis *win32*.

Die nachträgliche Installation von PostGIS ist zu empfehlen, da Sie darüber eine aktuelle PostGIS-Version installieren können und nicht auf die im PostgreSQL-Paket verwendete Version angewiesen sind, die möglicherweise nicht der aktuellen PostGIS-Version entspricht. Auf der PostGIS-Webseite finden Sie einen PostGIS-Installer für Windows, über den Sie die Erweiterung problemlos hinzufügen können: <http://www.postgis.org/download/windows>.

PostGIS für Linux

Die einzelnen großen Linux-Distributionen besitzen oft ihre eigenen, leicht zu installierenden Versionen von PostgreSQL und PostGIS. Am häufigsten kommen hierbei vermutlich RPM- und Debian-Pakete zum Einsatz.

RPM-Pakete

Wenn Sie nach einer einfachen Installationsmethode suchen, ist die Verwendung von RPM-Paketen eine gute Wahl (vorausgesetzt, Ihr System unterstützt RPM wie Fedora, Open Suse oder Red Hat). Die Pakete können unter mehreren Adressen heruntergeladen werden. Zuerst müssen Sie PostgreSQL installieren. Danach können Sie entweder Ihre eigene PostGIS-Version kompilieren und installieren, oder Sie verwenden ein fertiges PostGIS-Paket.

RPM-Pakete für PostgreSQL finden Sie unter <http://www.postgresql.org/ftp/binary/v8.2.5/linux> im Verzeichnis *rpms*. Hier finden Sie Pakete für Red Hat und Fedora.

Wollen Sie PostGIS mit diesen Paketen selbst kompilieren, benötigen Sie außerdem die entsprechenden Quellcode-RPMs. Kompilieren Sie PostGIS selbst, brauchen Sie auf jeden Fall die Dateien aus dem Quellcodepaket für PostgreSQL.

Auf der PostGIS-Seite finden Sie ebenfalls RPM-Pakete für Fedora: <http://postgis.refractory.net/rpms/>.

Installation unter Debian/Ubuntu

Unter Debian oder Ubuntu kann PostgreSQL leicht über die Paketverwaltung installiert werden. Verwenden Sie den *Adept*-Paketmanager, um folgende Pakete zu installieren:

- postgres
- postgis

Beachten Sie, dass Sie danach dem Datenbankbenutzer postgres ein Passwort geben müssen. Wollen Sie zusätzlich grafische Clients installieren, wählen Sie diese Pakete:

- pgadmin3
- phppgadmin
- apache2

Wenn Sie apt-get für die Installation verwenden, beschränkt sich die Installation im Wesentlichen auf die Eingabe von:

```
> apt-get install postgresql-8.2 postgresql-contrib-8.2
> su postgres -c psql template1
template1=# ALTER USER postgres WITH PASSWORD '****';
template1=# \q
```

Anmelden an der Datenbank:

```
> psql -h localhost -U postgres
```

Installation des Clients PgAdmin3

```
> sudo apt-get install pgadmin3
```

Weitere Informationen zum Kompilieren von PostGIS in einer Debian-Umgebung finden Sie hier: <http://www.postgresql.org/docs/techdocs.4>.

Informationen zur Installation unter Ubuntu gibt es unter <http://wiki.ubuntuusers.de/PostgreSQL>.

PostgreSQL und PostGIS für Mac OS X

Vorkompilierte Pakete für PostgreSQL und PostGIS unter Mac OS X werden regelmäßig von William Kyngesburye zusammengestellt. Sie finden den Link zu den Paketen auf der PostGIS-Downloadseite oder direkt unter <http://www.kyngchaos.com/software/unixport/postgres>.

Weitere Installationshinweise finden Sie hier:

<http://www.postgresql.org/docs/techdocs.4>

<http://www.entropy.ch/software/macosx/postgresql/> (Marc Liyanaga)

<http://abscond.org/entry/show/2373> (James Darling)

PostgreSQL und PostGIS selbst kompilieren

Sie können PostgreSQL auch aus dem Quellcode kompilieren. Dies empfiehlt sich, wenn Sie die aktuelle Version nutzen möchten und diese für Ihr System nicht verfügbar ist. Sie finden den Quellcode sowohl bei PostgreSQL (<http://www.postgresql.org/download/>) als auch auf der PostGIS-Downloadseite (<http://www.postgis.org/download>).

Das Kompilieren und Installieren von PostgreSQL sieht im einfachsten Fall wie folgt aus: Zunächst wird PostgreSQL installiert. Anschließend wird ein Systembenutzer postgres angelegt. Ein Datenverzeichnis wird angelegt und dem Benutzer postgres zugewiesen.

Über `initdb` wird ein neuer PostgreSQL-Datenbankcluster erzeugt. Ein Datenbankcluster ist eine Sammlung von Datenbanken, die von einer einzelnen Serverinstanz verwaltet wird. Abschließend wird die Datenbank gestartet:

```
> ./configure
> gmake
> su
> gmake install
> adduser postgres
> mkdir /usr/local/pgsql/data
> chown postgres /usr/local/pgsql/data
> su - postgres
> /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
> /usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data >logfile 2>&1 &
```

Eine ausführliche Beschreibung zur Kompilierung finden Sie in der PostgreSQL-Dokumentation (<http://www.postgresql.org/docs/>, in Kapitel 14) sowie unter <http://www.designmagick.com/article/2/Starting-Out/Installing-PostgreSQL>.

Die Kompilierung von PostGIS erfolgt über die folgenden Schritte. Dabei muss die Proj.4-Bibliothek (siehe <http://proj.maptools.org/>) vorliegen, damit Sie Koordinatentransformationen durchführen können. Außerdem muss die GEOS-Bibliothek installiert sein. PostGIS nutzt diese Bibliothek, um viele der Geometrieoperationen durchzuführen. GEOS ist die C++-Variante der JTS (Java Topology Suite). Mehr zur GEOS-Bibliothek finden Sie unter <http://geos.refrations.net/>.

```
> gzip -d -c postgis-1.3.1.tar.gz | tar xvf -
> ./configure --with-proj=PATH --with-geos=PATH
> make
> make install
```

Eine ausführliche Anleitung zum Kompilieren von PostGIS finden Sie im README-Dokument, das dem Quellcode beiliegt.

<http://postgis.refrations.net/docs/ch02.html>

PostGIS einrichten Schritt für Schritt

Es sind nur wenige Schritte notwendig, um eine Datenbank mit räumlichen Fähigkeiten zu versehen. Eine ausführliche Beschreibung der Schritte finden Sie unter <http://postgis.refrations.net/docs> in Kapitel 2.

Je nachdem, auf welche Weise Sie PostgreSQL und PostGIS installiert haben, sind unterschiedliche Schritte notwendig, um eine Datenbank mit PostGIS-Funktionalität zu erstellen. Erfolgte die Installation unter Windows mit dem Installer und Sie wählten bei der Installation *PostGIS Erweiterung für räumliche (= spatial) Daten*, wurde direkt bei der Installation eine Datenbank mit dem Namen *template_postgis* als Vorlage (Template) angelegt, die über PostGIS-Funktionalität verfügt.

Liegt dieses PostGIS-Template (Datenbank mit PostGIS-Funktionalität, die als Vorlage verwendet werden kann) nicht vor, müssen Sie die folgenden Schritte durchführen, um es zu erzeugen:

1. Installation von PostgreSQL und Start des Datenbankservers.
2. Anlegen einer Datenbank, die zukünftig als Vorlage verwendet werden soll.
3. Aktivierung der Sprachunterstützung für PL/pgSQL.
4. Laden des Skripts *lwpostgis.sql*.
5. Laden des Skripts *spatial_ref_sys.sql*.

Diese Schritte werden in den folgenden Abschnitten ausführlich erklärt, sollen an dieser Stelle aber schon angesprochen werden.

Datenbank-Administrationswerkzeuge

Für die Administration und Datenverwaltung in PostgreSQL stehen verschiedene Clients zur Verfügung.

psql

Die Installation von PostgreSQL umfasst automatisch den Shell-basierten Client *psql*, den wir zu Beginn kennenlernen werden.

pgAdmin

Als Client mit grafischer Benutzeroberfläche ist *pgAdmin* (<http://www.pgadmin.org/>) zu empfehlen. Dieser Client ist unter Windows direkt im Installationspaket enthalten und kann unter Linux über die Paketverwaltung installiert werden.

PhpPgAdmin

Der Client *PhpPgAdmin* (<http://phpPgAdmin.sourceforge.net/>) verfügt ebenfalls über eine grafische Benutzeroberfläche und ermöglicht den webbasierten Zugriff auf die Daten. *PhpPgAdmin* kann über Webbrowser bedient werden, es ist keine Installation auf allen Clientrechnern erforderlich.

In diesem Kapitel werden wir *psql* und *pgAdmin* kennenlernen. Im ersten Teil dieses Kapitels werden die Daten mit dem Client *psql* angesprochen. Es folgt anschließend eine kurze Einführung in *pgAdmin*.

Aufbau einer PostGIS-Datenbank

Damit PostGIS eingerichtet werden kann, muss ein PostgreSQL-Datenbankserver auf Ihrem System bereits laufen. Die Funktionsweise von PostgreSQL ist der anderer Datenbankmanagementsysteme (z.B. Oracle, DB2, SQL Server, MySQL) sehr ähnlich. Der Server wartet auf Anfragen eines Clientprogramms, führt die Anfragen auf der Datenbank aus und gibt schließlich das Ergebnis an den Client zurück.

Mit dem Kommandozeilenwerkzeug `psql` können Sie testen, ob Ihr Datenbankserver bereits läuft. Öffnen Sie dazu ein Befehlsfenster oder eine Shell. Unter Windows können Sie `psql` über die Startleiste durch Auswahl von *Start* → *Programme* → *PostgreSQL-Befehlszeile* öffnen.

Über den Befehl `psql -l` erhalten Sie eine Liste der verfügbaren Datenbanken:

```
> psql -U postgres -l
      Liste der Datenbanken
  Name      | Eigentümer | Kodierung
-----+-----+-----
 template0  | postgres   | SQL_ASCII
 template1  | postgres   | SQL_ASCII
(2 rows)
```

Wie Sie sehen, gibt es auf diesem System zwei Datenbanken, die Teil jeder PostgreSQL-Installation sind. Die Datenbank `template1` ist die Standardvorlage zum Anlegen neuer Datenbanken und geht aus `template0` hervor, das nicht verändert werden sollte. `template1` wird kopiert, und alle Einstellungen der Vorlage werden in die neue Datenbank übernommen.

Je nach Konfiguration der Zugriffsrechte kann es sein, dass Sie sich mit einem Datenbankbenutzer an der Datenbank anmelden müssen. Im ersten Beispiel für den Einsatz von `psql` können Sie sehen, dass die Datenbank `template1` dem Benutzer `postgres` gehört. Um sich beim Server als ein bestimmter Benutzer anzumelden, verwenden Sie den Parameter `-U` (Username), gefolgt vom Benutzernamen.

`psql` verfügt noch über weitere zusätzliche Parameter, um sich an der Datenbank anzumelden. Läuft Ihr Datenbankserver auf einem anderen Rechner als auf dem, auf dem Sie den Befehl `psql` ausführen, müssen Sie `-h` für den Rechnernamen des entfernten Servers angeben. Mit diesem Befehl weisen Sie `psql` an, auf dem Rechner mit dem angegebenen Namen nach dem Datenbankserver zu suchen:

```
> psql -l -h myserver.com -U admin
```

Als Ausgabe erhalten Sie eine Liste der verfügbaren PostgreSQL-Datenbanken auf dem Rechner mit dem Namen `myserver.com`. Ist der PostgreSQL-Server nicht aktiv oder nimmt er keine Verbindungen an, erhalten Sie eine entsprechende Fehlermeldung:

```
psql: konnte nicht mit dem Server verbinden: Connection timed out
      Läuft der Server auf dem Host "myserver.com" und akzeptiert er
      TCP/IP-Verbindungen auf Port 5432?
```

In den folgenden Beispielen werden Sie weitere Parameter kennenlernen, die Sie dem Programm `psql` übergeben können. Die Hilfe zu `psql` rufen Sie wie folgt auf:

```
>psql --help
```

Zusätzlich stehen Ihnen verschiedene Hilfsprogramme zur Verfügung, mit denen Sie Prozesse in der Datenbank durchführen können. Sie werden beispielsweise `createdb` (zum Erstellen einer Datenbank), `createlang` (zum Laden einer Sprache) sowie `shp2pgsql` (zum Shape-Import in die Datenbank) kennenlernen.

Im Folgenden erstellen Sie eine eigene Datenbank und nutzen dabei `template1` als Vorlage. In diese neue Datenbank werden anschließend die PostGIS-Skripten geladen. Dann können Sie Ihre neue Vorlage verwenden, um Datenbanken mit PostGIS-Fähigkeiten zu erzeugen. Dieser Prozess wird in Abbildung 14-1 verdeutlicht.

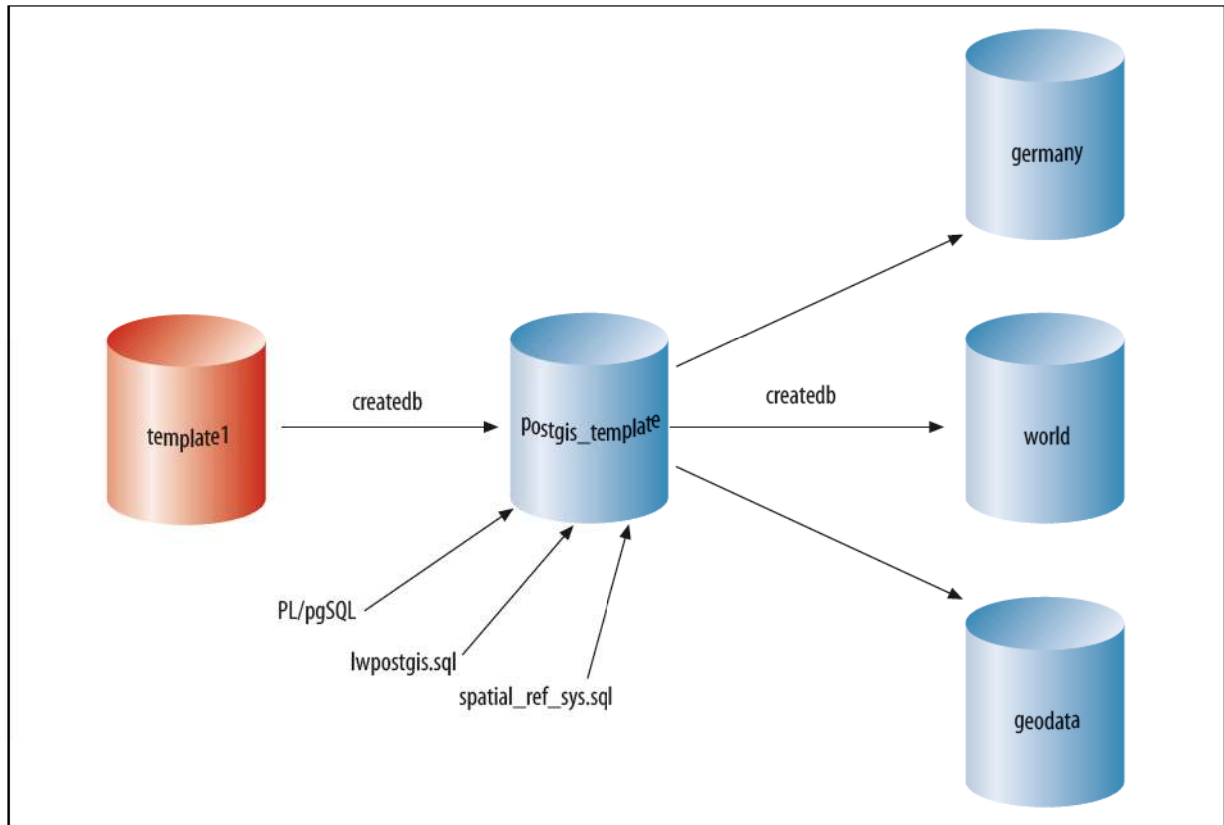


Abbildung 14-1: Anlegen einer Datenbank mit PostGIS-Funktionalität als Vorlage

Anlegen einer Datenbank

Als Erstes müssen Sie eine Datenbank anlegen. Diese soll zukünftig als Vorlage für räumliche Datenbanken dienen, die Sie erstellen wollen.

```
> createdb -U postgres -T template1 -E SQL_ASCII postgis_template
```

Das Hilfsprogramm `createdb` können Sie mit verschiedenen Parametern aufrufen. Im Beispiel wurde mit `-U` der Datenbankbenutzer angegeben und mit `-T` der Name der Datenbank, die als Vorlage verwendet werden soll. Mit `-E` geben Sie den gewünschten Zeichensatz an.



Beachten Sie, dass für deutschsprachige Umlaute ein spezieller Zeichensatz angegeben werden muss, die Voreinstellung `SQL_ASCII` kann dies nicht! Welchen Zeichensatz (Encoding) Sie auswählen, ist von den Quelldaten abhängig. Im deutschsprachigen Raum wird häufig Latin1 (aus der ISO-Reihe 8859) verwendet. Wenn Sie die Wahl haben, wird UTF8 empfohlen, da es alle Zeichen der meisten Sprachen unterstützt.

Schauen Sie sich die Beschreibung des Programms über `createdb --help` an. Die Möglichkeit, mit `--help` die Hilfe zum Programm ausgeben zu lassen, besteht für jedes der folgenden Programme, die Sie kennenlernen werden.

```
> createdb --help
createdb erzeugt eine PostgreSQL-Datenbank.
```

Benutzung:
`createdb [OPTION]... [DBNAME] [BESCHREIBUNG]`

Optionen:

<code>-D, --tablespace=TABLESPACE</code>	Standard-Tablespace der Datenbank
<code>-E, --encoding=KODIERUNG</code>	Kodierung für die Datenbank
<code>-O, --owner=EIGENTÜMER</code>	Eigentümer der neuen Datenbank
<code>-T, --template=TEMPLATE</code>	zu kopierende Template-Datenbank
<code>-e, --echo</code>	zeige die Befehle, die an den Server gesendet werden
<code>-q, --quiet</code>	unterdrücke alle Mitteilungen
<code>--help</code>	diese Hilfe anzeigen, dann beenden
<code>--version</code>	Versionsinformationen anzeigen

Verbindungsoptionen:

<code>-h, --host=HOSTNAME</code>	Name des Datenbankservers oder Socket-Verzeichnis
<code>-p, --port=PORT</code>	Port des Datenbankservers
<code>-U, --username=NAME</code>	Datenbankbenutzername
<code>-W, --password</code>	nach Passwort fragen

Wenn nichts anderes angegeben ist, dann wird eine Datenbank mit dem Namen des aktuellen Benutzers erzeugt.

Aktivieren der Sprache PL/pgSQL

PostgreSQL unterstützt die Verwendung interner Programmiersprachen. Diese Eigenschaft von PostgreSQL nutzt auch PostGIS. Die Funktionen von PostGIS sind in der Sprache PL/pgSQL und C programmiert. Die prozedurale Sprache PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) wird bei der Installation von PostgreSQL nicht zwingend aktiviert. Bei der Installation unter Windows mit dem Installer können Sie PL/pgSQL direkt auswählen. Die Sprachunterstützung liegt dann direkt in allen Datenbanken vor. Liegt PL/pgSQL nicht vor, kann die Sprachunterstützung mit dem folgenden Befehl geladen werden:

```
> createlang plpgsql postgis_template
```

Unsere neue Datenbank `postgis_template` wird so mit der Sprachunterstützung versehen.

Laden des Skripts lwpostgis.sql

Liegt die Sprachunterstützung nun vor, können Sie die PostGIS-Funktionen mit dem Skript *lwpostgis.sql* laden.



SQL-Skripten wie *lwpostgis.sql* sind Textdateien, die eine Liste mit SQL-Befehlen enthalten. Wenn Sie neugierig sind, wie die benutzten Befehle aussehen, können Sie die Datei in einem Texteditor oder einem Webbrowser öffnen.

Später werden Sie selbst ein paar SQL-Befehle schreiben. Wir werden SQL auch verwenden, um räumliche Daten in die Datenbank zu laden.

Zum Ausführen der SQL-Skripten verwenden wir ebenfalls das Kommandozeilenwerkzeug *psql*, das wir schon für die Ausgabe der verfügbaren Datenbanken kennengelernt haben. Da die Datei *lwpostgis.sql* auf dem gleichen Rechner liegt wie PostgreSQL selbst, müssen Sie den Pfad zur Datei kennen. Im folgenden Beispiel gehen wir davon aus, dass *psql* im gleichen Verzeichnis ausgeführt wird, in dem sich auch die Datei *lwpostgis.sql* befindet:

```
> psql -U postgres -f lwpostgis.sql -d postgis_template
```

Der erste Parameter *-f* gibt an, welche Datei geladen werden soll, geben Sie hier den vollständigen Pfad zur Datei an. Der zweite Parameter *-d* legt fest, in welcher Datenbank die Befehle ausgeführt werden sollen. Geben Sie hier *template1* an, stehen die PostGIS-Funktionen in allen neu angelegten Datenbanken automatisch zur Verfügung. Verwenden Sie Ihre neue Datenbank *postgis_template*.

Während das Skript geladen und ausgeführt wird, erscheint eine Reihe von Hinweisen und Warnungen auf dem Bildschirm. Die Warnungen können Sie in der Regel ignorieren. Meistens handelt es sich um Hinweise zu erfolgreich ausgeführten Befehlen. Bei einem wirklichen Fehler wird die Ausführung des Skripts angehalten, und Sie sehen eine entsprechende Fehlermeldung, wie diese:

```
...
psql:postgis.sql:37: ERROR:  function "histogram2d_in" already exists with same
argument types
psql:postgis.sql:42: ERROR:  current transaction is aborted, commands ignored until end
of transaction block
...
```

Diese Meldungen besagen beispielsweise, dass ein Teil der PostGIS-Funktionen bereits in die Datenbank geladen wurde.

Wurde das Skript erfolgreich ausgeführt, verfügt die neue Datenbank *postgis_template* nun über eine Vielzahl neuer Funktionen und Datentypen. Außerdem enthält die Datenbank nun die Tabellen *geometry_columns* und *spatial_ref_sys*. Bevor Sie sich die Tabellen genauer anschauen, müssen Sie noch ein weiteres Skript laden.

Laden des Skripts `spatial_ref_sys.sql`

Das zweite Skript mit dem Namen `spatial_ref_sys.sql` füllt die zuvor angelegte Tabelle `spatial_ref_sys` mit Informationen zu verschiedenen Projektionen. Damit PostGIS Daten umprojizieren (von einem Projektionssystem in ein anderes umwandeln) kann, werden diese Informationen in der Tabelle benötigt.

Die SQL-Datei wird auf die gleiche Weise geladen wie zuvor `lwpostgis.sql`:

```
> psql -U postgres -f spatial_ref_sys.sql -d postgis_template
```

Dieses Skript füllt die Tabelle `spatial_ref_sys` mit über 3.000 Einträgen. Während dieses Skript ausgeführt wird, erscheinen zahlreiche INSERT-Meldungen, beispielsweise:

```
...  
INSERT 797518 1  
INSERT 797519 1  
INSERT 797520 1  
COMMIT  
VACUUM
```

Die abschließende Meldung `COMMIT` ohne die Ausgabe von Fehlermeldungen bestätigt, dass die Tabelle erfolgreich gefüllt wurde. Über `VACUUM` wird die Datenbank reorganisiert.

Haben Sie vergessen, zuvor das Skript `lwpostgis.sql` auszuführen oder wurde `spatial_ref_sys.sql` bereits geladen, erhalten Sie Fehlermeldungen.

Überprüfen der PostGIS-Funktionalität der Datenbank

Nun schauen wir uns die Datenbank `postgis_template` genauer an und prüfen, ob alles korrekt installiert wurde. Anschließend nutzen wir `postgis_template` als Vorlage für neue Datenbanken, die wir anlegen.

Wenn Sie den Befehl `psql` für eine Verbindung zur Datenbank `postgis_template` verwenden, können Sie bereits einige Tests ausführen.

Einstieg in den Umgang mit `psql`

Als Erstes melden Sie sich über `psql` an der Datenbank `postgis_template` an. Nach der Anmeldung haben Sie dann die Möglichkeit, eigene SQL-Anweisungen und `psql`-Befehle auszuführen. Für die Anmeldung an der Datenbank verwenden wir `psql` gefolgt vom Namen der gewünschten Datenbank, in unserem Fall also `postgis_template`. Je nach Konfiguration müssen Sie beispielsweise über `-U` noch den Datenbankbenutzer oder andere Parameter hinzufügen.

```
> psql -U postgres postgis_template  
Dies ist psql 8.2.4, das interaktive PostgreSQL-Terminal.
```

```
Geben Sie ein: \Copyright für Urheberrechtsinformationen  
               \h für Hilfe über SQL-Anweisungen
```

```

\? für Hilfe über interne Anweisungen
\g oder Semikolon, um eine Anfrage auszuführen
\q um zu beenden
postgis_template=#

```

Beim Start von psql werden die Programmversion und eine Reihe nützlicher Hinweise ausgegeben. Danach erscheint eine Eingabeaufforderung. Dieser Prompt entspricht dem Namen der verwendeten Datenbank, gefolgt von den Zeichen =#.

Das Programm ist jetzt bereit, SQL-Anweisungen (oder psql-spezifische Befehle) entgegenzunehmen. Schauen Sie unbedingt zuerst die Hilfe an, um eine Übersicht über die Möglichkeiten von psql zu bekommen.

```

\? für Hilfe über interne Anweisungen
\h für Hilfe über SQL-Anweisungen

```

Nun überprüfen Sie, ob die von PostGIS benötigten Tabellen angelegt wurden. Um alle Objekte in der Datenbank anzuzeigen, verwenden Sie den Befehl \d. Wollen Sie nur die Tabellen sehen, hängen Sie dem Befehl noch ein t an, sodass er nun \dt, lautet, wie im nächsten Beispiel gezeigt:

```

postgis_template=# \dt
                Liste der Relationen
 Schema |      Name      |  Typ  | Eigentümer
-----+-----+-----+-----
 public | geometry_columns | Tabelle | postgres
 public | spatial_ref_sys  | Tabelle | postgres
(2 Zeilen)

```

Über *lwpostgis.sql* wurden die zwei Tabellen in der Datenbank angelegt. Werden jetzt keine Tabellen angezeigt, gab es bei der Ausführung des Skripts einen Fehler.

Wenn das Skript *spatial_ref_sys.sql* erfolgreich ausgeführt wurde, sollte die Tabelle *spatial_ref_sys* jetzt zahlreiche Einträge enthalten. Zur Überprüfung können Sie Ihre erste SQL-Anweisung ausführen, die die Anzahl der Spalten einer Tabelle ermittelt. Achten Sie darauf, die SQL-Anweisung immer mit einem Semikolon oder mit \g abzuschließen:

```

postgis_template=# SELECT count(*) FROM spatial_ref_sys;
count
-----
    3162
(1 Zeile)

```



Um mehr über SQL-Befehle zu erfahren, ist der Wikipedia-Eintrag zu SQL ein guter Startpunkt: <http://de.wikipedia.org/wiki/SQL>, ebenso natürlich die PostgreSQL-Dokumentation unter <http://www.postgresql.org/docs/>. Das offizielle Handbuch zu PostgreSQL von Peter Eisentraut finden Sie unter <http://www.postgresql.org/files/documentation/books/pg handbook/html/>.

Lassen Sie sich von der Meldung (1 Zeile) nicht verwirren. Sie bezieht sich auf die Anzahl der Ausgabezeilen für die Abfrage. Das tatsächliche Ergebnis der Abfrage lautet 3162. Die Tabelle ist also mit Sicherheit nicht leer. Das sind gute Nachrichten. Beachten Sie, dass die Zahl der Einträge in der Tabelle *spatial_ref_sys* nicht statisch sind. Es kommen laufend weitere Definitionen für Projektionen hinzu.



Wie können Sie sich sicher sein, dass Sie über die richtige Anzahl an Einträgen in der Tabelle *spatial_ref_sys.sql* verfügen? Am einfachsten ermitteln Sie dies mit einem Befehl über die Kommandozeile. Mit folgendem Befehl zählen wir, wie oft das Wort *INSERT* in der Datei *spatial_ref_sys.sql* vorliegt. Den Befehl *grep* haben Sie bereits in Kapitel 6 kennengelernt:

```
> grep -c INSERT spatial_ref_sys.sql
3162
```

Die zur Datenbank hinzugefügten Funktionen machen einen großen Teil des PostGIS-Systems aus. Sie können sich eine Liste der im *postgis_template* verfügbaren Funktionen anzeigen lassen, um zu sehen, ob sie richtig geladen wurden. Mit dem *psql*-Befehl *\df* können Sie sich alle Funktionen einer Datenbank anzeigen lassen. In unserer Datenbank sind dies mehrere Hundert. Durch die Verwendung des Wildcard-Zeichens (*) mit dem Präfix *postgis* können Sie nur die Funktionen ausgeben lassen, deren Name mit *postgis* beginnt. Nicht alle PostGIS-Funktionen beginnen mit *postgis*, deshalb ist die Anzahl der Funktionen nicht identisch mit den weiter oben geladenen SQL-Skripten.

```
postgis_template=# \df postgis*
Liste der Funktionen
Schema | Name | Ergebnisdatentyp |
-----+-----+-----+
public | postgis_full_version | text |
public | postgis_geos_version | text |
public | postgis_gist_joinrel | double precision |
public | postgis_gist_sel | double precision |
public | postgis_jts_version | text |
public | postgis_lib_build_date | text |
public | postgis_lib_version | text |
public | postgis_proj_version | text |
public | postgis_scripts_build_date | text |
public | postgis_scripts_installed | text |
public | postgis_scripts_released | text |
public | postgis_uses_stats | boolean |
public | postgis_version | text |
(13 Zeilen)
```

Die Namen der Funktionen stehen in der zweiten Spalte mit der Überschrift *Name*. Diese Funktionen können direkt in SQL-Befehlen verwendet werden, um räumliche Operationen durchzuführen. Die Funktion *postgis_version* gibt Informationen über die verfügbare PostGIS-Installation aus:


```

postgis_template=# SELECT postgis_version();
               postgis_version
-----
1.3 USE_GEOS=1 USE_PROJ=1 USE_STATS=1
(1 Zeile)
postgis_template=#\q

```

Die Ausgabe gibt Auskunft darüber, welche PostGIS-Version verwendet wird (1.3). Außerdem sehen Sie, dass die GEOS- sowie die PROJ.4-Bibliothek als Teil von PostGIS zur Verfügung stehen. Die PROJ.4-Bibliothek wird verwendet, um Koordinaten auf ein Koordinatensystem zu projizieren. Eine ausführliche Beschreibung des PROJ.4-Projekts finden Sie in Anhang A. GEOS ist eine Bibliothek, die eine Vielzahl von Manipulationen und Analysen geometrischer Datentypen ermöglicht.

Weitere Informationen zu PROJ.4 und GEOS finden Sie auf den Projektseiten:

- PROJ.4: <http://www.remotesensing.org/proj/>
- GEOS: <http://geos.refrations.net/>

Mit \q melden Sie sich wieder von der Datenbank ab.

createdb – Anlegen einer räumlichen Datenbank

Die Datenbank `postgis_template` ist nun die Vorlage für neue Datenbanken, die Sie anlegen werden. Arbeiten Sie nicht direkt mit `postgis_template`, weil Sie sie dann nicht mehr als Vorlage nutzen können. Erstellen Sie stattdessen für jedes neue Projekt eine eigene Datenbank. Hierfür verwenden Sie das Hilfsprogramm `createdb` gefolgt vom gewünschten Datenbanknamen:

```

> createdb -U postgres -T postgis_template world -E LATIN1
CREATE DATABASE

```

In unserem Beispiel heißt die neue Datenbank `world`. Mit `-T` definieren Sie, dass die Datenbank `postgis_template` als Vorlage (Template) für diese Datenbank verwendet werden soll. Mit `-E` geben Sie den Zeichensatz an, der genutzt werden soll. Der ausgegebene Text (`CREATE DATABASE`) bestätigt, dass der Befehl erfolgreich ausgeführt wurde. Wenn Sie sich nun die verfügbaren Datenbanken ausgeben lassen, sollte `world` in der Liste auftauchen:

```

> psql -U postgres -l
           Liste der Datenbanken
    Name      | Eigentümer | Kodierung
-----+-----+-----
template0    | postgres  | SQL_ASCII
template1    | postgres  | SQL_ASCII
postgis_template | postgres  | SQL_ASCII
world        | postgres  | LATIN1
(4 Zeilen)

```

Melden Sie sich an der Datenbank an und lassen Sie sich zur Prüfung die Anzahl der Tabellen ausgeben.

```
> psql -U postgres world
Dies ist psql 8.2.4, das interaktive PostgreSQL-Terminal.
...
world=# \dt
               Liste der Relationen
Schema |      Name      |  Typ  | Eigentümer
-----+-----+-----+-----
public | geometry_columns | Tabelle | postgres
public | spatial_ref_sys | Tabelle | postgres
(2 Zeilen)
world=# \q
```

Geodatenhaltung in PostGIS

Die GIS-Objekte, die von PostGIS unterstützt werden, orientieren sich an der OGC *Simple Features-Spezifikation*. In der Spezifikation werden Geometrietypen sowie Funktionen, über die die Geometrieobjekte bearbeitet werden können, definiert. Hinzu kommt noch die Definition von Metadatentabellen.

Seit der PostGIS-Version 0.9 werden alle Funktionen und Objekte der OGC Simple Features-Spezifikationen in PostGIS vollständig umgesetzt. Dabei kann PostGIS sogar mehr, als in der Spezifikation definiert wird. So ergänzt PostGIS den Standard um weitere Funktionen und die Unterstützung von 3DZ-, 3DM- und 4D- Koordinaten.

Die OGC Simple Features-Spezifikation definiert zwei Standards, in denen räumliche Objekte formuliert werden können: die Formate *Well-Known Text (WKT)* und *Well-Known Binary (WKB)*. In diesen Formaten werden der Geometrietyp und die Koordinaten des Objekts gespeichert.

Hier ein paar Beispiele für die Definition räumlicher Objekte im WKT-Format, die der PostGIS-Dokumentation entnommen wurden:

```
POINT(0 0)
LINESTRING(0 0,1.5 1,1 2.3)
POLYGON((0 0,4.1 0.2,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
MULTIPOINT(0 0,1 2)
MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
GEOMETRYCOLLECTION(POINT(2 3),LINESTRING((2 3,3 4)))
```

Auf den Namen des Geometrietyps folgen in Klammern die Stützpunkte des Geometrieobjekts. Hierbei werden die Koordinaten durch ein Leerzeichen getrennt. Die einzelnen Stützpunkte werden durch Kommata getrennt. Beachten Sie, dass das Dezimaltrennzeichen der Punkt ist.

Eine ausführliche Beschreibung des Datenmodells von PostGIS finden Sie in der PostGIS-Dokumentation in Kapitel 4:

<http://postgis.refractory.net/docs/ch04.html>

Geodaten selbst erzeugen

An einem kleinen Beispiel soll verdeutlicht werden, wie Geometriespalten erzeugt und Geometrien angelegt werden. Unsere erste Tabelle soll zur Speicherung von Sehenswürdigkeiten (POI – Points of Interest) genutzt werden. Eine Tabelle mit einer Geometriespalte wird dabei in zwei Schritten erzeugt.

Zuerst wird die Tabelle über das folgende SQL angelegt:

```
world=# CREATE TABLE poi(
world(#  id serial,
world(#  name varchar(100),
world(#  beschreibung varchar(250)
world(# )WITH OIDS;
NOTICE: CREATE TABLE will create implicit sequence "poi_id_seq" for serial column
"poi.id"
CREATE TABLE
```

Die Tabelle erhält den Namen poi. Die Spalte id ist vom Typ serial und bewirkt, dass jeder Datensatz eine eindeutige ID erhält. Die Spalten name und beschreibung sind Textfelder. Die Tabelle wurde WITH OIDS angelegt, damit jeder Datensatz eine eindeutige vom System vergebene ID erhält.

Im zweiten Schritt wird die Geometriespalte über die Funktion AddGeometryColumn(schema_name,table_name,column_name,srid, type, dimension) hinzugefügt:

```
world=# SELECT AddGeometryColumn('public','poi','the_geom',31466, 'POINT', 2);
addgeometrycolumn
-----
public.poi.the_geom SRID:31466 TYPE:POINT DIMS:2
(1 Zeile)
```

Die Tabelle wurde im Schema public angelegt. Eine Datenbank kann ein bis viele Schemata enthalten. Diese dienen der Strukturierung in der Datenbank. Dabei ist public das Default-Schema. Der Name des Schemas wird der Funktion als erster Parameter übergeben, kann aber auch als leerer String übergeben werden, wenn das Default-Schema verwendet werden soll. Anschließend folgt der Tabellename (poi), dann der Name der Geometriespalte (the_geom). Über 31466 wird das *Koordinatenreferenzsystem* angegeben, das für die Daten verwendet werden soll. 31466 steht für *Gauß-Krüger-Zone 2*. Über POINT definieren wir den Geometrietyp. Der letzte Parameter (2) gibt die Dimension der Daten an. Die Funktion legt eine neue Tabellenspalte an, erzeugt *Constraints* (Einschränkungen) und fügt einen Eintrag in die Metadatentabelle geometry_columns ein.



Beachten Sie, dass die PostGIS-Funktionsnamen ab der PostGIS-Version 1.3.1 um das Präfix `ST_` erweitert wurden. Verwenden Sie eine ältere PostGIS-Version, müssen Sie daher die Funktionen ohne das Präfix angeben. Eine genauere Beschreibung dazu finden Sie weiter unten im Abschnitt zu PostGIS-Funktionen.

Nun füllen wir die Tabelle mit einigen Datensätzen. Hierzu wird die Funktion `ST_GeomFromText(text,[srid])` verwendet:

```
INSERT INTO poi (name, beschreibung, the_geom) VALUES ('Dom','sehr hoch, sehr alt', ST_GeomFromText('POINT(2567372 5645564)', 31466));
INSERT INTO poi (name, beschreibung, the_geom) VALUES ('Zoo','viele Tiere', ST_GeomFromText('POINT(2568490 5647550)', 31466));
INSERT INTO poi (name, beschreibung, the_geom) VALUES ('Rathaus','Altstadt, historisches Rathaus', ST_GeomFromText('POINT(2567430 5645197)',31466));
```

Schauen Sie sich die Daten nun über die folgenden `SELECT`-Anweisungen an. Damit Geometrien in einer für Sie lesbaren Form ausgegeben werden – also im WKT-Format – nutzen Sie die Funktionen `ST_AsText(geometry)` und `ST_AsEWKT(geometry)`:

```
world=# SELECT * FROM poi;
 id | name |      beschreibung      |      the_geom
-----+-----+-----+-----
  1 | Dom  | sehr hoch, sehr alt    | 0101000020EA7A0000...
  2 | Zoo  | viele Tiere            | 0101000020EA7A0000...
  3 | Rathaus | Altstadt, historisches Rathaus | 0101000020EA7A0000...
(3 Zeilen)
```

```
world=# SELECT name, ST_AsText(the_geom), ST_AsEWKT(the_geom) FROM poi;
 name |      astext      |      asewkt
-----+-----+-----
 Dom  | POINT(2567372 5645564) | SRID=31466;POINT(2567372 5645564)
 Zoo  | POINT(2568490 5647550) | SRID=31466;POINT(2568490 5647550)
 Rathaus | POINT(2567430 5645197) | SRID=31466;POINT(2567430 5645197)
(3 Zeilen)
```

Die erste Abfrage gibt alle Informationen zu einem Datensatz aus. Dabei wird die Geometrie in der »kanonischen« Form, die für uns nicht lesbar ist, ausgegeben. Um die Informationen der Geometriespalte in einer lesbaren Form auszugeben, können Sie die PostGIS-Funktionen `ST_AsText(geometry)` oder `ST_AsEWKT(geometry)` verwenden. Beide Funktionen geben die Geometrie als Well-known-Text aus. Dabei enthält die Ausgabe der Funktion `ST_AsEWKT(geometry)` zusätzlich die Information zum verwendeten `srid`.

Schauen Sie sich auch die Einträge in der Tabelle `geometry_columns` an:

```
world=# SELECT * FROM geometry_columns;
-[ RECORD 1 ]-----+-----
f_table_catalog |
f_table_schema  | public
f_table_name     | poi
f_geometry_column | the_geom
```


coord_dimension		2
srid		31466
type		POINT

Glückwunsch! Sie haben Ihre erste eigene Tabelle erstellt. In den nächsten Abschnitten werden Sie lernen, wie Sie Daten über den grafischen Datenbankclient pgAdmin verwalten und wie Sie die Daten in Quantum GIS visualisieren können.

pgAdmin als grafischer Datenbankclient

Wenn Sie sich auf der Kommandozeile nicht sonderlich wohlfühlen, sollten Sie unbedingt pgAdmin als grafischen Datenbankclient kennenlernen. Unter Windows starten Sie pgAdmin über *Start → Programme → PostgreSQL → pgAdmin III*. Unter Linux rufen Sie den Befehl `pgadmin3` auf (prüfen Sie zuerst, ob pgAdmin installiert wurde).

Sie müssen eine Verbindung zu Ihrem Datenbankserver aufbauen. Dies erfolgt über den Menüeintrag *Datei → Server hinzufügen*. In einem neuen Fenster müssen Sie die Verbindungsparameter für die Serverregistrierung angeben. Es kann auch sein, dass nach der Installation von PostgreSQL schon eine gültige Verbindung vorliegt. Verbinden Sie sich anschließend mit einem Doppelklick auf den gewünschten Server mit der Datenbank. Sie erhalten eine Baumstruktur mit einer Übersicht der Datenbanken. Hier finden Sie auch Ihre neuen Datenbanken `world` und `postgis_template`. Wählen Sie nun die Datenbank `world` aus und schauen Sie sich Ihre Tabellen an. Diese befinden sich in der Baumstruktur unter *Schemata – public – Tabellen* (1). Die Datensätze der Tabellen können Sie sich über den Button mit dem Tabellensymbol (2) anzeigen lassen. In pgAdmin können Sie ebenfalls SQL-Abfragen abschicken. Dazu öffnen Sie über den Button *SQL* (3) ein Abfragefenster. Tragen Sie in dem Fenster Ihre Abfrage ein und schicken Sie diese über den grünen Pfeil (4) ab. Die einzelnen Prozesse werden in Abbildung 14-2 veranschaulicht.

Visualisieren der Daten mit Quantum GIS

Nun ist es an der Zeit, dass wir uns die Daten in einem Desktop-GIS anzeigen lassen. In Kapitel 8 haben wir bereits Quantum GIS als Desktop-GIS kennengelernt, das in der Lage ist, Daten aus PostgreSQL/PostGIS anzuzeigen.

Starten Sie Quantum GIS. Über den Menüeintrag *Ebene → Füge PostGIS Ebene hinzu*, die Taste `D` oder den entsprechenden Button öffnen Sie das Fenster, über das Sie eine Verbindung zu einer Datenbank herstellen können. Hierzu geben Sie über den Button *Neu* die Verbindungsinformationen an, testen diese und speichern sie dann. Anschließend verbinden Sie sich mit der Datenbank und bekommen eine Liste der vorliegenden Geodaten, die Sie laden können. Wählen Sie `poi` durch einen Klick aus und fügen Sie die Daten hinzu. `poi` wird als Kartenebene geladen, und Sie sehen die drei Punkte im Kartenfenster. Wir werden Quantum GIS weiterhin nutzen, um die Ergebnisse unserer SQL-Abfragen zu visualisieren. Schauen Sie sich dazu auch die Beschreibung in Kapitel 8 an.

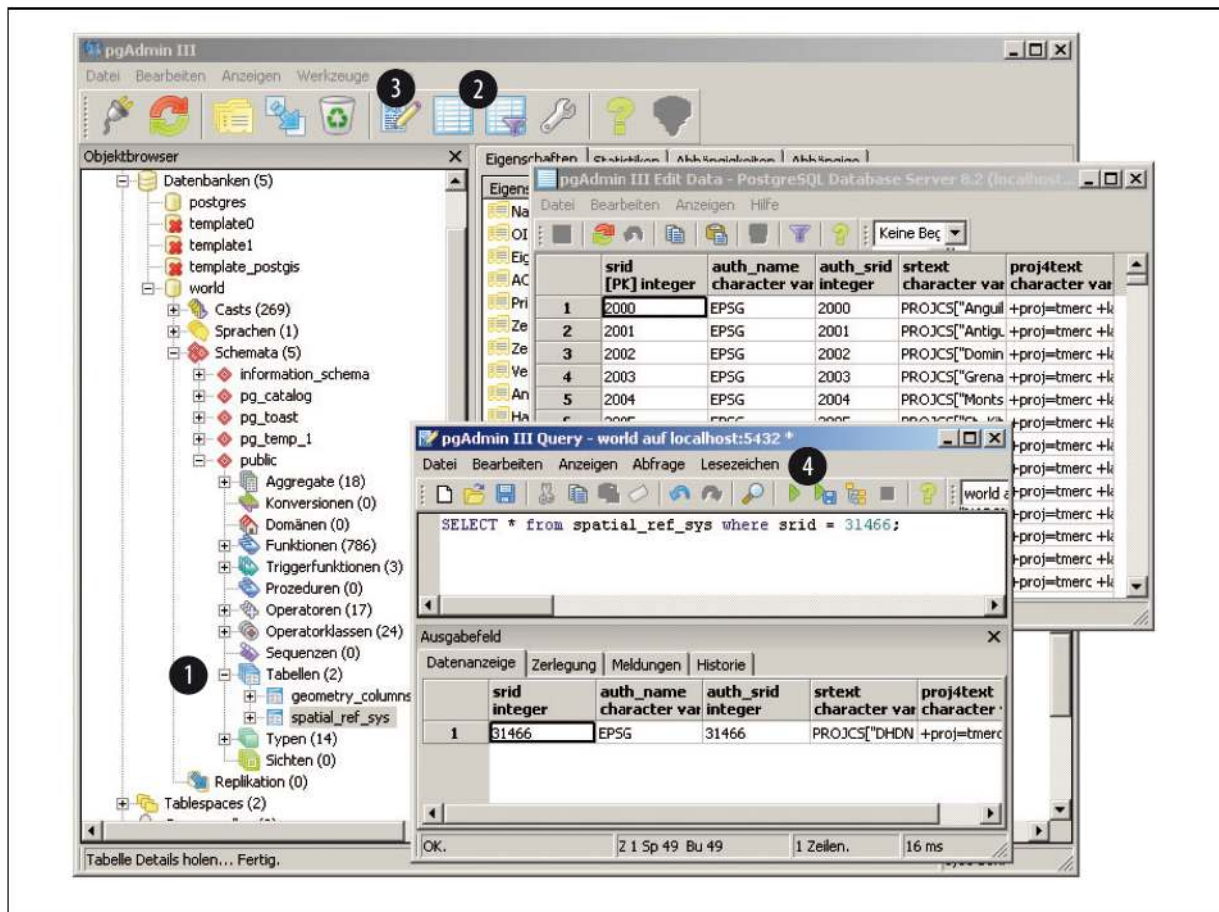


Abbildung 14-2: Der grafische Datenbankclient pgAdmin (die im Text beschriebenen Komponenten sind in der Abbildung durch Nummern markiert)

Laden von Geodaten

Es gibt verschiedene Möglichkeiten, Geodaten in die Datenbank zu überführen. Ein Weg besteht darin, die Daten in Form von SQL-Anweisungen mit `psql` manuell einzugeben. Dies haben wir bereits kennengelernt. Sie können aber auch ein Programm verwenden, das die Daten in ein SQL-Skript verpackt, das dann geladen werden kann. Oder Sie verwenden ein Werkzeug, das die Daten direkt in die Datenbank überführt.

Als Daten wollen wir die Weltdaten verwenden, die Sie schon in Kapitel 10 kennengelernt haben. Sie können die Daten über den folgenden Link herunterladen:

http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.2.tar.gz

Bei der Datei handelt es sich um eine mit `gzip` komprimierte `tar`-Datei, die Sie mit dem Befehl `tar` auf der Kommandozeile entpacken können:

```
> tar -xvzf freegis_worlddata-0.2.tar.gz
```

Unter Windows sind die meisten Zip-Programme (z.B. Filzip, WinZip) in der Lage, die Datei zu entpacken. Im Verzeichnis *geodata* finden Sie eine Shape-Datei mit dem Namen *countries_simpl_pop*.

shp2pgsql

PostGIS liegen zwei Kommandozeilenwerkzeuge namens `pgsql2shp` und `shp2pgsql` bei. Mit diesen ist es möglich, PostGIS-Tabellen nach Shape zu konvertieren und umgekehrt.

Das Werkzeug `shp2pgsql` konvertiert die Daten aus einer Shape-Datei und überführt sie in eine Reihe von SQL-Anweisungen. Sie können die Anweisungen entweder direkt an `psql` weiterreichen oder sie zum späteren Laden in einer Textdatei speichern.

Hier ein Beispiel für den Shape-Import in zwei Schritten:

```
> shp2pgsql -s 4326 -I countries_simpl_pop.shp world_countries > world_countries.sql
> psql -d world -f world_countries.sql
```

Die Anweisungen `shp2pgsql` und `psql` können mittels einer »Pipe« verkettet werden, sodass Sie die Daten auch direkt in die Datenbank laden können:

```
> shp2pgsql -s 4326 -I countries_simpl_pop.shp world_countries | psql -d world
```

Standardmäßig speichert `shp2pgsql` die Geometrien in einem Feld namens `the_geom`. Der Befehl `ogr2ogr` (zu dem wir im folgenden Abschnitt kommen) verwendet hierfür jedoch den Feldnamen `wkb_geometry`. Diese Namen können aber auch durch eigene Feldnamen ersetzt werden.

Über `-s` definieren Sie den SRID (spatial referencing identifier) der Daten. Wird dieser Parameter nicht angegeben, wird `-1` verwendet. Der Parameter `-I` bewirkt, dass die Daten mit einem *räumlichen Index* (GIST) versehen werden. Was sich dahinter verbirgt, werden Sie später noch erfahren.

Weitere Optionen von `shp2pgsql` können Sie sich über `shp2pgsql --help` ausgeben lassen.

ogr2ogr

`ogr2ogr` ist ein hervorragendes Programm, um Geodaten in die Datenbank zu überführen. Das Werkzeug ist Teil des GDAL/OGR-Pakets, das Sie schon aus den Kapiteln 3 und 7 kennen. `ogr2ogr` ist in der Lage, einen beliebigen von OGR-unterstützten Datentyp in die Datenbank zu überführen.

Mithilfe von `ogrinfo` können Sie sich Informationen zur Datei ausgeben lassen:

```
> ogrinfo countries_simpl_pop.shp -al -so
...
Feature Count: 2461
...
```

Prinzipiell sollten Sie immer feststellen, wie viele Daten konvertiert werden müssen, bevor Sie den nächsten Schritt ausführen. Je mehr Geometrien die Datei enthält, desto länger wird es dauern, diese in die Datenbank zu laden.

Um die Shape-Datei in die Datenbank world zu überführen, verwenden Sie den Befehl ogr2ogr auf folgende Weise:

```
> ogr2ogr -f "PostgreSQL" "PG:dbname=world user=postgres host=localhost port=5432"
countries_simpl_pop.shp -nln countries -a_srs EPSG:4326
```

Dies ist eines der einfachsten Beispiele für die Benutzung von ogr2ogr mit PostgreSQL/PostGIS. Der erste Parameter definiert das Zieldatenformat – in unserem Fall eine PostgreSQL-Datenbank. Über PG: geben Sie die Verbindungsinformationen für die Anmeldung an der Datenbank an. Geben Sie diese in doppelten Anführungszeichen an. Über -nln geben Sie den Namen der Tabelle an, die erstellt werden soll. Der Parameter -a_srs gibt das Referenzsystem an, in dem die Daten ausgegeben werden sollen.

Wie in Beispiel 14-1 gezeigt, können Sie mit psql überprüfen, ob die Daten erfolgreich geladen wurden. Hierfür lassen wir uns zuerst die verfügbaren Tabellen anzeigen und führen dann eine einfache SQL-Abfrage durch, um zu sehen, ob alle Datensätze korrekt gespeichert wurden.

Beispiel 14-1: Prüfen, ob die geladenen Geodaten in der Datenbank vorliegen

```
> psql -U postgres -d world
```

Dies ist psql 8.2.4, das interaktive PostgreSQL-Terminal.

Geben Sie ein: \copyright für Urheberrechtsinformationen
 \h für Hilfe über SQL-Anweisungen
 \? für Hilfe über interne Anweisungen
 \g oder Semikolon, um eine Anfrage auszuführen
 \q um zu beenden

```
world=# \dt
```

Liste der Relationen

Schema	Name	Type	Eigentümer
public	countries	table	postgres
public	geometry_columns	table	postgres
public	poi	table	postgres
public	spatial_ref_sys	table	postgres
public	world_countries	table	postgres

(5 Zeilen)

```
world=# SELECT count(*) FROM countries;
```

```
count
-----
  2461
(1 Zeile)
```

Über die SQL-Anweisung werden die Datensätze gezählt, und die Summe wird ausgegeben.

Mithilfe von `psql` können Sie sich den Tabellenaufbau nun anzeigen lassen. Verwenden Sie hierfür den Befehl `\d` gefolgt vom Namen der Tabelle:

Beispiel 14-2: Anzeige der Liste der Spalten und Datentypen der Tabelle `countries`

```
world=# \d countries
```

Tabelle »public.countries«		
Spalte	Typ	Attribute
ogc_fid	integer	not null default
nextval('countries_ogc_fid_seq'::regclass)		
wkb_geometry	geometry	
cat	double precision	
vmap	character(80)	
fibs	character(80)	
iso3166	character(80)	
name	character(80)	
comment	character(80)	
total	double precision	
male	double precision	
female	double precision	
ratio	numeric(24,15)	

Indexe:

```
»countries_pk« PRIMARY KEY, btree (ogc_fid)
```

Check-Constraints:

```
»enforce_dims_wkb_geometry« CHECK (ndims(wkb_geometry) = 2)
```

```
»enforce_geotype_wkb_geometry« CHECK (geometrytype(wkb_geometry) =  
'POLYGON' ::text OR wkb_geometry IS NULL)
```

```
»enforce_srid_wkb_geometry« CHECK (srid(wkb_geometry) = 32767)
```

Jede Spalte wird mit dem dazugehörigen Datentyp angezeigt. Abgesehen von der Geometriespalte (Typ `geometry`) kann PostgreSQL mit den genannten Datentypen auch ohne die Hilfe von PostGIS umgehen. Lediglich die Spalte mit dem Namen `wkb_geometry` ist vom Datentyp `geometry` und enthält räumliche Daten:

```
wkb_geometry | geometry
```



In einer Tabelle können mehrere Spalten vom Typ `geometry` vorliegen. Verwenden Sie mehr als ein `geometry`-Feld, können Sie sogar unterschiedliche Geometrietypen (Punkte, Linien oder Polygone) gemeinsam in einer Tabelle speichern.

Das Programm `ogr2ogr` hat die `geometry`-Spalte automatisch als `wkb_geometry` bezeichnet. (Hierbei steht das Präfix `wkb_` für *Well-Known Binary*). Sie können für die Geometriespalte aber auch einen beliebigen eigenen Namen definieren.

Zu jeder Geometriespalte werden Metadaten in der Tabelle `geometry_columns` – sowohl durch `shp2pgsql` als auch durch `ogr2ogr` – abgelegt. Über die Metadaten können andere Anwendungen Informationen über die Geodaten erhalten. Sie sollten darauf achten, dass zu jeder Geometriespalte entsprechende Metadaten vorliegen.

Im folgenden Beispiel werden die Informationen der Tabelle `geometry_columns` für die Tabelle `countries` ausgegeben. Dabei wird vorher mit `\x` in den erweiterten Anzeigemodus gewechselt.

```
world=#\x
Erweiterte Anzeige ist an.
world=# SELECT * FROM geometry_columns WHERE f_table_name = 'countries';
-[ RECORD 1 ]-----
f_table_catalog |
f_table_schema | public
f_table_name    | countries
f_geometry_column | wkb_geometry
coord_dimension | 2
srid            | 32767
type            | POLYGON
```



Zu Beginn des Beispiels weisen wir `psql` mit dem Befehl `\x` an, die Daten im erweiterten Anzeigemodus auszugeben. Diese Ausgabe ist etwas leichter zu lesen. Ein zweiter Aufruf von `\x` beendet den erweiterten Anzeigemodus wieder.

Über die `where`-Bedingung wird nur der Eintrag für die Tabelle `countries` ausgegeben. Die Daten liegen im Schema `public` vor. Der Name der Geometriespalte wird im Feld `f_geometry_column` gespeichert. PostGIS ist übrigens auch in der Lage, 2-D- oder 3-D-Daten zu speichern. Hierfür erhält das Feld `coord_dimension` entweder den Wert 2 (2-D) oder 3 (3-D). Das Feld `srid` enthält Informationen über das verwendete Koordinatenreferenzsystem. Ist der Wert -1, wurde kein System angegeben. Die Angabe `type | MULTI-POLYGON` besagt, dass in der Geometriespalte Multipolygone gespeichert werden können.



Laden Sie Ihre Daten in Quantum GIS. Sie erhalten die Ansicht einer Weltkarte mit den Ländern, die als Polygone vorliegen. Schauen Sie sich dazu auch Abbildung 14-3 an.

Räumliche Daten abfragen

Eine der Stärken von PostGIS ist die Fähigkeit, räumliche Daten in einem fortschrittlichen relationalen Datenbanksystem zu speichern, die auch den Anforderungen großer Datenmengen und komplexer Anfragen standhält. Die Daten werden über standardisierte SQL-Abfragen bereitgestellt und stehen anderen Programmen zur weiteren Verarbeitung zur Verfügung, ohne dass sie erst aus oder in proprietäre Datenformate konvertiert werden müssten. Eine weitere Stärke liegt in der Möglichkeit, die PostGIS-eigenen räumlichen Funktionen zu nutzen und so Analysen ebenfalls über SQL-Befehle in der Datenbank selbst vorzunehmen.

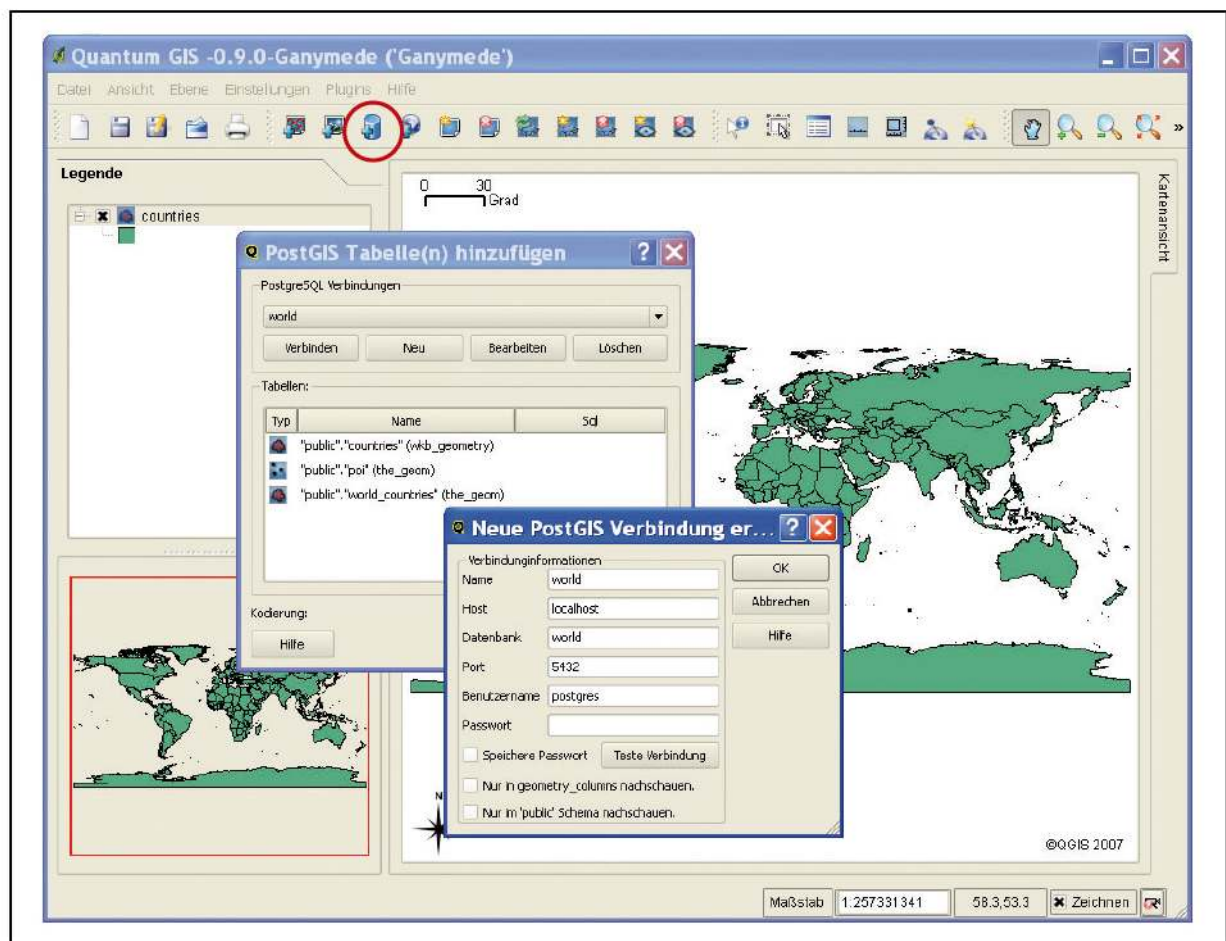


Abbildung 14-3: Visualisieren der Daten aus PostGIS mit dem Desktop-GIS Quantum GIS

Die folgenden Beispiele zur Abfrage räumlicher Daten sind absichtlich sehr einfach gehalten. PostGIS enthält Hunderte von Funktionen für die Verwaltung, Manipulation und Erzeugung räumlicher Daten. Hierzu gehört auch die Möglichkeit, Puffer um Geometrien zu erzeugen, Verschneidungen durchzuführen, Schnittmengen zu berechnen, Daten zu transformieren und vieles mehr.

Eine ausführliche Beschreibung der Funktionen finden Sie hier:

- <http://postgis.refractory.net/docs/ch06.html> (PostGIS-Dokumentation)
- http://openjump.org/PostGIS_Tutorial.pdf (PostGIS-Tutorial in deutscher Sprache von Uwe Delluege)

Unter <http://wiki.osgeo.org/index.php/PostGIS> finden Sie weitere hilfreiche Links.

Außerdem werden neue Funktionen und Hinweise zu den Funktionen auf den Mailinglisten diskutiert. Das Archiv finden Sie unter:

<http://postgis.refractory.net/pipermail/postgis-users/>

Um zu sehen, welche Daten in der Tabelle countries vorliegen, reicht schon eine sehr einfache SQL-Anweisung aus, um eine Liste ausgeben zu lassen. Mit dem folgenden

Befehl können Sie sämtliche Informationen der Tabelle, inklusive der räumlichen Informationen, ausgeben lassen.

```
world=# SELECT * FROM countries;
```

Die Ausführung dieses Befehls ohne weitere Einschränkungen ist nicht unbedingt empfehlenswert. Geben Sie daher über `limit` die Anzahl der Datensätze an, die ausgegeben werden sollen.

```
world=# SELECT * FROM countries limit 1;
```

Eine Liste aller Informationen der Länder ist vermutlich nicht besonders nützlich. Die Abfrage aus Beispiel 14-3 erscheint da schon sinnvoller. Hierbei werden nur die Informationen einer Spalte ausgegeben:

Beispiel 14-3: Eine einfache SELECT DISTINCT-Abfrage

```
# SELECT DISTINCT name FROM countries WHERE name = 'Switzerland';
      name
```

```
-----
Switzerland
```

```
(33 rows)
```

Die Abfrage gibt alle Datensätze mit dem Ländernamen Switzerland aus (`name='Switzerland'`).

In Beispiel 14-4 sehen Sie eine Abfrage, die alle Informationen zu einem Datensatz ausgibt. Dabei wird die Geometrie in der »kanonischen« Form ausgegeben, die wir bereits kennengelernt haben. Nutzen Sie die Funktionen `ST_AsText(geometry)` und `ST_AsEWKT(geometry)`, um sich die importierten Daten in einer lesbaren Form anzuschauen.

```
Select ST_AsEWKT(ogc_geometry) from countries WHERE name = 'Switzerland';
```

Beispiel 14-4: Abfrage aller Informationen eines Datensatzes

```
world=# \x
```

Erweiterte Anzeige ist an.

```
world=# SELECT * FROM countries WHERE name = 'Switzerland';
```

```
-[ RECORD 1 ]-----
ogc_fid      | 2202
wkb_geometry | 0103000020E61000000100000039.....
cat          | 215
vmap         | SZ
fibs         | SZ
iso3166      | CH
name         | Switzerland
comment      | -
total        | 7450867
male         | 3669066
female       | 3781801
ratio        | 97.000000000000000
```


Das Beispiel zeigt, wie Sie über eine Abfrage direkt auf die Geodaten zugreifen können. Hierfür ist kein spezielles GIS-Programm notwendig, sondern nur einfache SQL-Anweisungen. Die Ergebnisse liegen unter Umständen nicht im gewünschten Format vor, lassen sich aber leicht an die eigenen Bedürfnisse anpassen. Vielleicht wollen Sie eine eigene Applikation erstellen, die auf die Geometrien zugreift. In diesem Fall können Sie die räumlichen Informationen aus einer Tabelle abfragen und die Daten in das für Ihre Anwendung nötige Format umwandeln.

Erstellung von Sichten (Views) zur Speicherung der Abfragen

Um die Ergebnisse der SQL-Abfragen festzuhalten, können Sie die Resultate einer Abfrage entweder in einer neuen Tabelle oder die Abfrage in einer Sicht (View) speichern. Eine Sicht ist eine virtuelle Tabelle, die auf Grundlage der Definition der Sicht dynamisch erzeugt wird. Um eine Sicht der Abfrage aus Beispiel 14-4 zu erzeugen, müssen Sie am Anfang der SQL-Anfrage die Schlüsselwörter `CREATE VIEW <view name> AS` verwenden:

```
# CREATE VIEW query_countries_switzerland AS
  SELECT * FROM countries WHERE name = 'Switzerland' ;
```

Sie können nun auf die Sicht zugreifen wie auf jede andere Tabelle auch. Für den Zugriff geben Sie dabei den Namen der Sicht an.

```
SELECT * FROM query_countries_switzerland;
```

Hierbei müssen Sie darauf achten, dass die Geometriespalte in der Sicht aufgeführt ist, um die Daten grafisch darstellen zu können. Schauen Sie sich nun Ihre Sicht in Quantum GIS direkt an. In Abbildung 14-4 sehen Sie das Ergebnis.

GIST – räumlicher Index

Um den Zugriff auf die Daten zu optimieren, sollten für alle Geometriespalten ein räumlicher Index, *GIST* (*Generalized Search Tree*), erstellt werden. Hierfür verwenden Sie die SQL-Anweisung `CREATE INDEX` und erzeugen einen speziellen Index – GIST. Der Befehl zur Erstellung eines GIST lautet beispielsweise:

```
# CREATE INDEX gist_countries ON countries USING GIST (wkb_geometry GIST_GEOMETRY_OPS);
```

Als Name für den Index wurde `gist_countries` verwendet. Sie können hier einen beliebigen Namen vergeben. Der zweite Parameter (`countries`) gibt die Tabelle an. Außerdem wird mitgeteilt, für welche Geometriespalte der GIST erzeugt werden soll (`wkb_geometry`).

Nach jeder Anweisung zur Indexerstellung sollte der folgende Befehl ausgeführt werden:

```
# VACUUM ANALYSE <Name der Tabelle>;
```

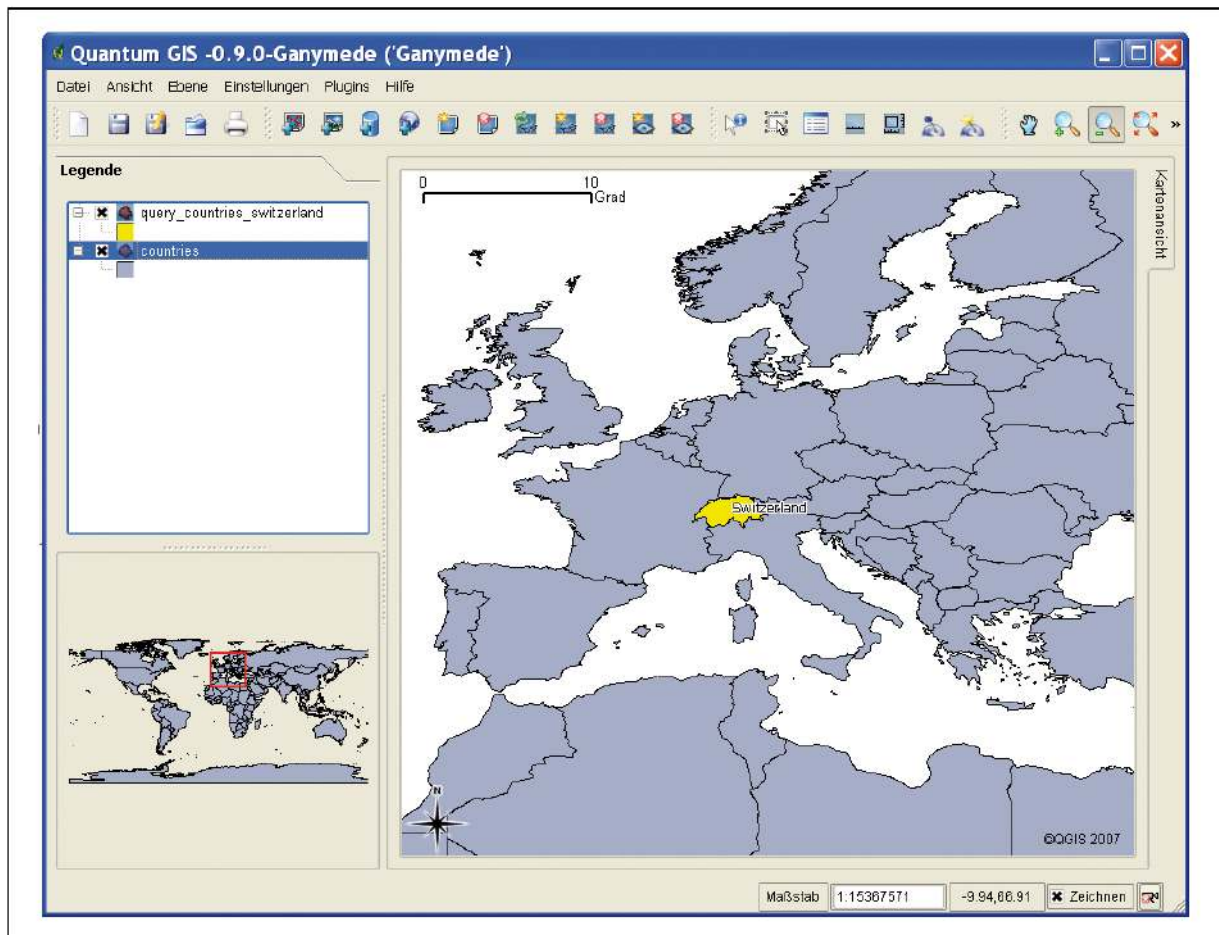


Abbildung 14-4: Anzeige des Polygons der Schweiz über eine Sicht, die in QGIS geladen wurde

Mit diesem Befehl wird die Datenbank reorganisiert, wodurch die Anfragen effizienter bearbeitet werden können. Wichtig ist hierbei die Angabe von ANALYSE. ANALYSE untersucht die Tabellen und speichert in einer Systemtabelle Informationen, die dann bei einer Abfrage ausgewertet werden.



Wenn Sie der Anweisung `VACUUM ANALYSE` keinen weiteren Parameter übergeben, wird der Befehl für alle Tabellen in der Datenbank ausgeführt. Liegen große Datenmengen und viele Tabellen in der Datenbank vor, kann dieser Prozess sehr lange dauern.

Ein GIST kann auch direkt beim Datenimport angelegt werden. Bei `shp2pgsql` erfolgt dies über die Angabe des Parameters `-I`. Schauen Sie sich dazu nochmals den Abschnitt über `shp2pgsql` an.

Einsatz von Indizes zur Zugriffsoptimierung

Ein GIST allein sorgt aber noch lange nicht für eine gute Performance. Sie müssen alle Datenbankabfragen möglichst effizient gestalten. Daher sollten Sie für alle Felder, die in WHERE-Abfragen verwendet werden sollen, Indizes anlegen. Es sollte eine eindeutige Spalte vorliegen und indiziert werden.

Als eindeutige Spalte können Sie auch die Spalte oid verwenden. Diese beinhaltet eine eindeutige, vom System vergebene laufende Nummer. Um die Spalte oid nutzen zu können, muss die Tabelle WITH OIDS angelegt worden sein.



Die Spalte oid wird in der Tabellenliste, die Sie durch den Befehl \dt erhalten, nicht angezeigt. Stattdessen können Sie sich die Werte über eine SELECT-Anweisung ausgeben lassen.

Vorstellung einiger PostGIS-Funktionen

Sie können PostGIS-Funktionen zur Analyse räumlicher Daten einsetzen. Früher waren spezielle GIS-Analysewerkzeuge für diese Aufgaben notwendig. Inzwischen können Sie die Analyse auch direkt in der PostGIS-Datenbank durchführen. In diesem Abschnitt sollen einige nützliche PostGIS-Funktionen vorgestellt werden. Eine ausführliche Erläuterung der Funktionen finden Sie in der PostGIS-Dokumentation in Kapitel 6 (<http://postgis.refractory.net/docs/ch06.html>).



Beachten Sie, dass die PostGIS-Funktionsnamen ab der PostGIS-Version 1.3.1 an den SQL/MM-Standard angepasst wurden. Das heißt, viele Funktionen wurden um das Präfix ST_ – für spatial type – erweitert, z.B. ST_Area(geometry). Die bisherigen Funktionen ohne Präfix liegen derzeit weiterhin vor, werden aber in zukünftigen Versionen entfallen. In diesem Abschnitt werden bereits die an SQL/MM orientierten Funktionsnamen verwendet. Wenn Sie mit einer älteren PostGIS-Version arbeiten, lassen Sie das Präfix ST_ einfach im Aufruf weg.

Speichern Sie sich die SQL-Abfragen direkt in einer Sicht und schauen Sie sich diese dann in Quantum GIS an. Beachten Sie, dass dazu die Geometriespalte in der Sicht vorliegen muss.

```
CREATE VIEW qry_switzerland AS SELECT * FROM countries where name =  
'Switzerland';
```

ST_Extent(geometry)

Über die Aggregatfunktion ST_Extent(geometry) können Sie den Darstellungsbereich Ihrer Daten ermitteln. Dieser Bereich wird als BOX mit der Angabe der südwestlichen und nordöstlichen Ausdehnung ausgegeben.

```
world=# SELECT ST_Extent(wkb_geometry) FROM countries;  
          st_extent  
-----  
BOX(-179.999908447266 -89.9999008178711,179.999908447266 83.6095886230469)  
(1 Zeile)
```

ST_X(geometry) und ST_Y(geometry)

Diese Funktionen geben die x- bzw. y-Koordinate eines Punkts zurück.

```
world=# SELECT name, ST_X(the_geom), ST_Y(the_geom) FROM poi;
 name | st_x | st_y
-----+-----+-----
 Dom   | 2567372 | 5645564
 Zoo   | 2568490 | 5647550
 Rathaus | 2567430 | 5645197
(3 Zeilen)
```

ST_TRANSFORM(geometry, integer)

Über ST_TRANSFORM(geometry, integer) können Geometrien in ein anderes Koordinatenreferenzsystem überführt werden. In dem folgenden Aufruf wird die neu berechnete Geometrie über ST_AsEWKT(geometry) anschließend in lesbarer Form ausgegeben.

```
world=# SELECT name, ST_AsEWKT(ST_Transform(the_geom,4326))
world=# FROM poi WHERE name='Zoo';
 name | st_asewkt
-----+-----
 Zoo  | SRID=4326;POINT(6.97424930834382 50.9590748500905)
(1 Zeile)
```

ST_Length(geometry)

Über ST_Length(geometry) wird die Länge einer Linie in der Einheit des verwendeten Projektionssystems ausgegeben. Im folgenden Beispiel wird über GeomFromText(geometry, integer) dynamisch eine Linie erzeugt und anschließend die Länge der Linie berechnet. Die Länge der Linie wird dabei in den Einheiten des verwendeten Projektionssystems ausgegeben. Im folgenden Beispiel sind dies Meter.

Schauen Sie sich auch ST_AREA(geometry) zur Flächenberechnung an.

```
world=# SELECT ST_Length(
world=# GeomFromText('LINESTRING(2567372 5645564,2568490 5647550)')
world=# ,31466)
world=# );
 st_length
-----
 2279.06121023548
(1 Zeile)
```

In welchem Land befindet sich eine bestimmte Koordinate?

Wenn Sie dies wissen wollen, können Sie es über die Funktion ST_Distance(geometry, geometry) ermitteln.

Eine schnelle Websuche nach dem Vulkan Mt. St. Helens bringt uns auf die Site <http://www.fs.fed.us/gpnp/volcanocams/msh/faqs/location.shtml>. Die Koordinaten für den Gipfel lauten 46.20 N, 122.18 W. In welchem Land befindet sich nun diese Koordinate?

Hier sehen Sie eine mögliche SQL-Anweisung, um diese Frage zu klären. Auf die Einzelheiten wird gleich näher eingegangen:

```
world=# SELECT name
world=# FROM countries
world=# WHERE distance( wkb_geometry,
world=#      ST_GeometryFromText('POINT(-122.18 46.20)',4326)
world=#      ) = 0;
name
-----
United States
(1 Zeile)
```

Erzeugen Sie am besten direkt einen View und schauen Sie sich das Ergebnis Ihrer Abfrage in Quantum GIS an. Achten Sie dabei darauf, dass eine eindeutige ID und die Geometriespalte in der SQL-Abfrage aufgeführt werden.

```
world=# CREATE VIEW qry_st_helena as
world=# SELECT ogc_fid, name, wkb_geometry FROM countries
world=# WHERE distance( wkb_geometry,
world=#      ST_GeomFromText('POINT(-122.18 46.20)',4326) ) = 0;
```

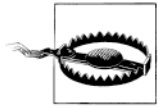
- SELECT ogc_fid, name, wkb_geometry gibt die Spalten an, die im Ergebnis der Abfrage ausgegeben werden sollen.
- FROM countries legt fest, in welcher Tabelle gesucht werden soll.
- Mit dem Schlüsselwort WHERE wird eine Bedingung definiert, über die die Ergebnisse einschränkt werden. In diesem Fall werden nur die Länder zurückgegeben, für die die Kriterien, die nach dem Schlüsselwort WHERE folgen, zutreffen.
- Die WHERE-Klausel besteht aus zwei Teilen: der PostGIS-Funktion und dem Kriterium, mit dem das Ergebnis der Funktion verglichen werden soll. ST_Distance(geometry,geometry) ist eine Funktion, mit der Entfernungen zwischen Geometrien berechnet werden können. Die Funktion benötigt zwei Geometrien als Parameter. Als erster Parameter wird in unserem Beispiel wkb_geometry, also die Geometriespalte der Tabelle countries, angegeben. Der zweite Parameter ist eine dynamisch erzeugte Geometrie: ST_GeomFromText('POINT(-122.18 46.20)',4326). Über die Funktion ST_GeomFromText(text,integer) wird ein Text, der eine Geometrie im *Well-Known Text*-Format (WKT) beschreibt, in eine Geometrie umgewandelt. Als zweiter Parameter muss der SRID angegeben werden. Die zuvor ermittelten Koordinaten von Mt. St. Helens werden der Funktion im WKT-Format übergeben. Das WKT-Format für einen Punkt sieht wie folgt aus:

```
POINT(-122.18 46.20)
POINT(<longitude> <latitude>)
```

oder:

```
POINT(X Y)
```

- Die Koordinaten werden mit Klammern umgeben und durch ein Leerzeichen voneinander getrennt angegeben. Als Dezimaltrennzeichen wird der Punkt verwendet. POINT gibt den Geometrietyp an.



Oftmals werden die Koordinaten fälschlicherweise in der Reihenfolge latitude, longitude angegeben. Häufig wird nicht bedacht, dass für die Datenbank zuerst die Ost/West-Werte (auf der x-Achse) und als Zweites die Nord/Süd-Werte (y-Achse) angegeben werden müssen. Längenangaben (longitude) beziehen sich auf die x-Achse, Breitenangaben (latitude) auf die y-Achse.

- In der letzten Zeile des obigen Beispiels werden die Entfernungen verglichen. Ergibt der Vergleich das Ergebnis 0, berühren sich die Geometrien oder überlappen sich.

Wenn Sie eine einfache Linie erzeugen wollen, können Sie die Stützpunkte einfach durch Kommata voneinander getrennt angeben. Der Geometrietyp ist in diesem Fall LINESTRING:

```
LINESTRING(-122.18 46.20, -110 50, -100 60.5)
```

Jetzt werden alle Länder, durch die diese Linie verläuft, ausgegeben:

```
world=# CREATE VIEW qry_st_helena as
world(# SELECT ogc_fid, name, wkb_geometry FROM countries
world(# WHERE distance( wkb_geometry,
world(#      ST_GeomFromText(
world(# 'LINESTRING(-122.18 46.20, -110 50, -100 60.5)',4326) ) = 0;
name
-----
Canada
United States
(2 Zeilen)
```

Welche Länder grenzen aneinander?

Sie können Geometrien innerhalb einer Tabelle miteinander vergleichen. Hierbei wird die Tabelle mit sich selbst verknüpft (*Self-Join*). In Beispiel 14-5 sehen Sie einen Self-Join, über den die angrenzenden Länder eines Landes ermittelt werden sollen, die Tabelle wird also mit sich selbst verglichen. Für die Tabellen in der FROM-Klausel wird jeweils ein Alias definiert (a und b), sodass die Tabelle nun über unterschiedliche Namen angesprochen werden kann.

Beispiel 14-5: Mithilfe eines Self-Joins werden die angrenzenden Länder über deren Bounding Box ermittelt (&&)

```
world=# SELECT DISTINCT a.name
world)# FROM countries a,
world)#      countries b
world)# WHERE b.name = 'Norway'
world)# AND a.wkb_geometry && b.wkb_geometry;
```

Beispiel 14-5: Mithilfe eines Self-Joins werden die angrenzenden Länder über deren Bounding Box ermittelt (&&) (Fortsetzung)

```
-----
name
-----
Estonia
Finland
Lativa
Norway
Russia
Saint Lucia
Svalbard
Sweden
(8 Zeilen)
```

Tabelle a wird in der SELECT-Anweisung angesprochen, liefert also die Ergebnisse. Die Ergebnisse des Vergleichs werden beschränkt von den Werten aus Tabelle b. Der erste Teil der WHERE-Klausel (`b.name = 'Norway'`) reduziert Tabelle b also auf die Polygone, die zum Land Nowegen vorliegen. Über `DISTINCT` wird jedes Ergebnis nur einmal ausgegeben.

In der WHERE-Klausel werden zwei Bedingungen definiert und durch `AND` verknüpft. Beide Bedingungen müssen erfüllt sein, damit ein Objekt im Ergebnis ausgegeben wird. Bei der zweiten Bedingung (`a.wkb_geometry && b.wkb_geometry`) kommt wieder PostGIS ins Spiel. Geometrien werden mit dem Operator `&&` verglichen. Dieser Operator überprüft, ob sich die Umgebungsrechtecke (Bounding Boxes) der Geometrien überschneiden. Die Bounding Box ist das kleinste Rechteck, das eine Geometrie umgibt, in Abbildung 14-5 durch die gestrichelte Linie angedeutet.

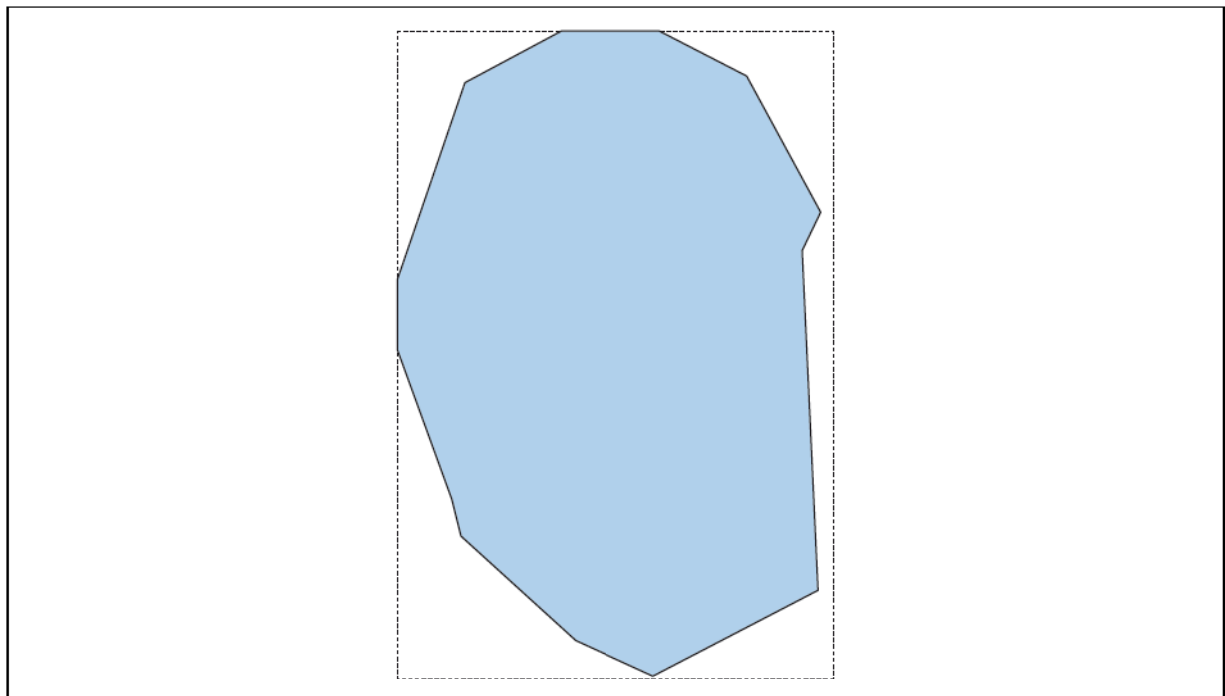


Abbildung 14-5: Ein Polygon und sein Umgebungsrechteck (Bounding Box)

Die Abfrage in Beispiel 14-5 vergleicht die Umgebungsrechtecke von Norwegen mit denen der anderen Geometrien (Abbildung 14-6). Beachten Sie die Überschneidungen. Gibt es zwischen dem Umgebungsrechteck von Norwegen (in Rot dargestellt) und dem Umgebungsrechteck eines anderen Landes (in Grün) eine Überschneidung, wird das betreffende Land als Treffer betrachtet.

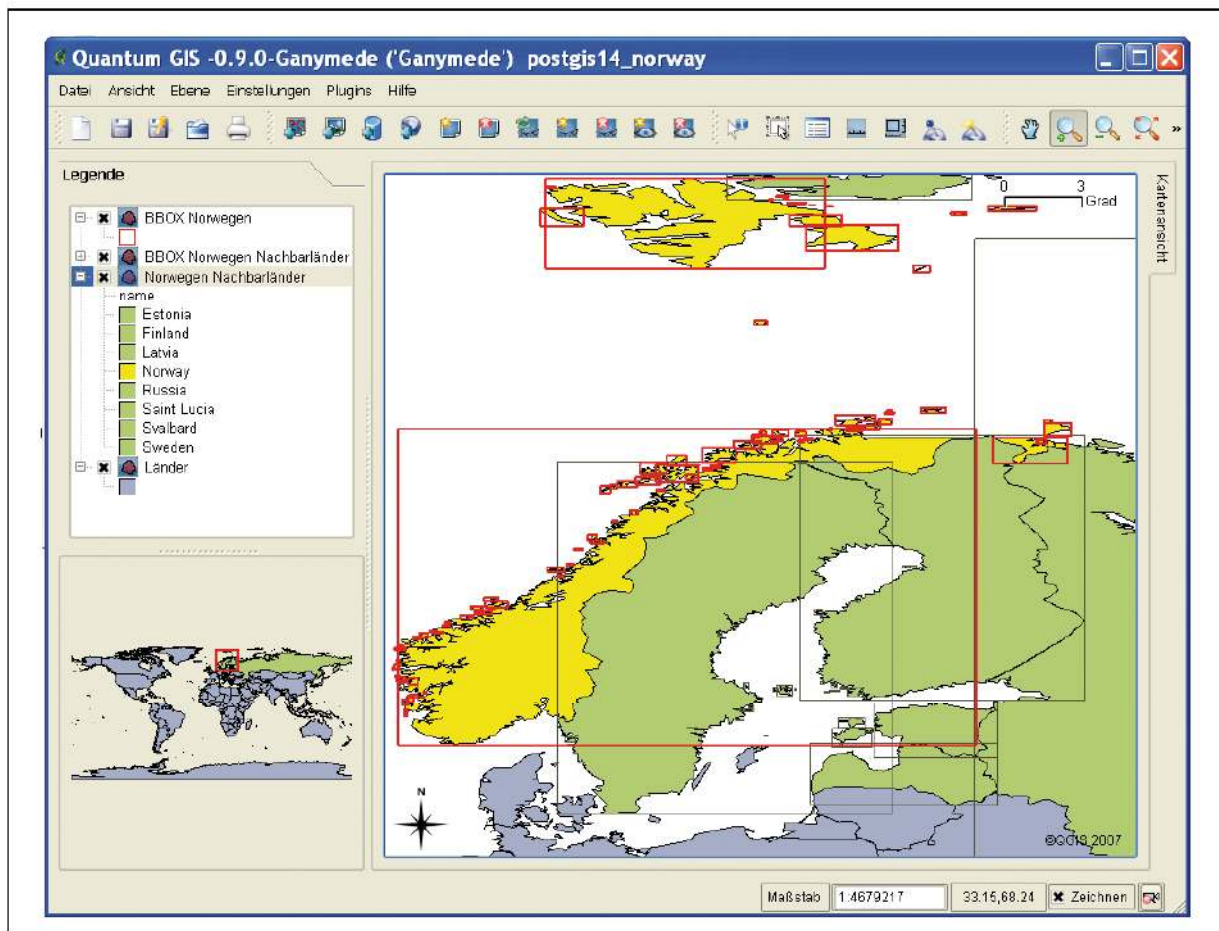


Abbildung 14-6: Darstellung von Norwegen und den Nachbarländern sowie deren Umgebungsrechtecke

Diese Verwendung der Bounding Box durch eine Abfrage über `&&` kann die Performance räumlicher Abfragen deutlich erhöhen. PostGIS muss keine komplexen Berechnungen für jeden Stützpunkt oder jede Linie eines Polygons durchführen. Eine einfache Überprüfung der Umgebungsrechtecke reicht in vielen Fällen völlig aus.

Dieses Beispiel soll Ihnen eigentlich zeigen, welche Länder an Norwegen angrenzen. Wenn Sie Abbildung 14-7 aber genauer betrachten, werden Sie feststellen, dass es Länder gibt, deren Umgebungsrechteck sich mit denen Norwegens überschneidet, obwohl die Geometrien nicht direkt aneinandergrenzen. Dies sind die Polygone von Estonia, Lativa, Svalbard und Saint Lucia. Saint Lucia ist ein Inselstaat in der Karibik, von dem sich ein Inselchen aufgrund eines Fehlers der Daten scheinbar in den Bereich westlich von Norwegen verirrt hat.

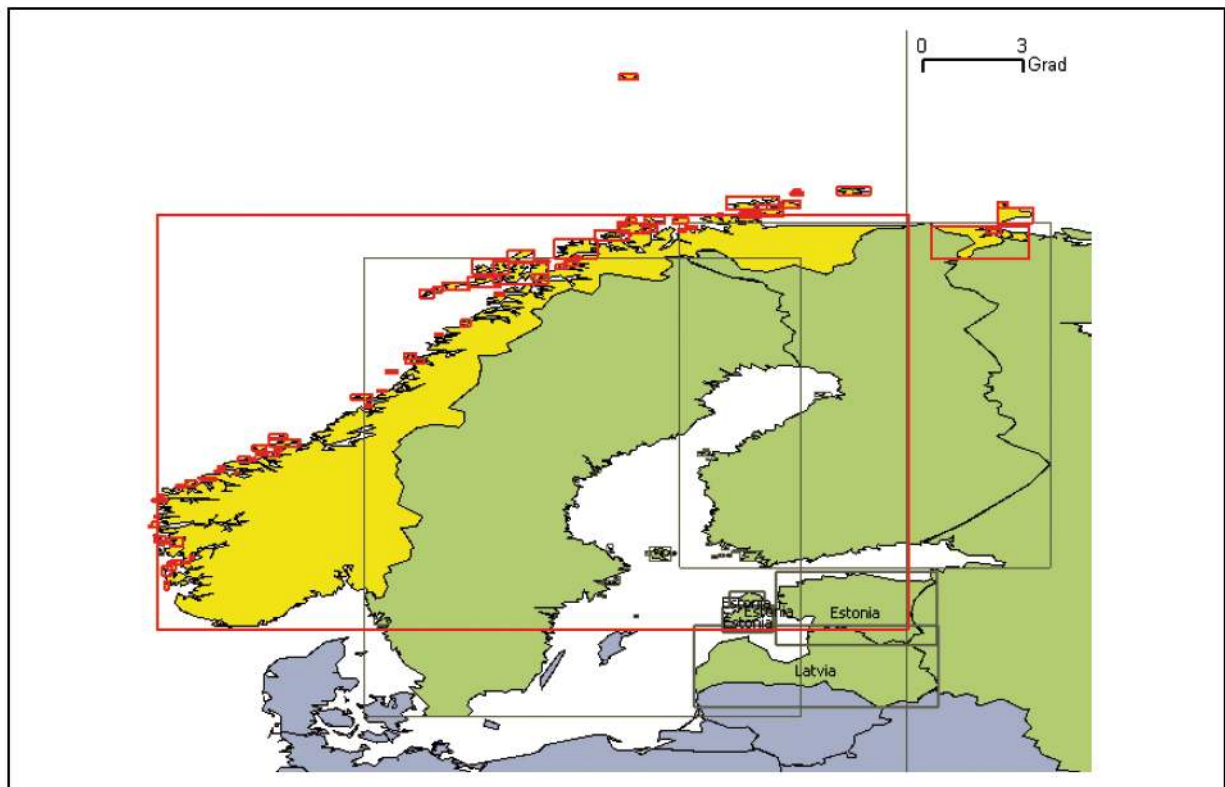


Abbildung 14-7: Das Umgebungsrechteck (Bounding Box) von Norwegen überlappt mit dem von Lettland (Latvia) und Estland, obwohl die Polygone nicht direkt aneinandergrenzen

Der Operator `&&` prüft lediglich das Überlappen oder Berühren der Umgebungsrechtecke. Er kann Ihnen aber nicht zuverlässig sagen, ob zwei Objekte tatsächlich aneinandergrenzen oder ob sie sich nur in relativer Nähe zueinander befinden. Besitzen zwei Geometrien keine sich überschneidenden oder berührenden Bounding Boxes, grenzen sie auf keinen Fall aneinander. Diese Funktion kann bei räumlichen Abfragen der Einschränkung von Ergebnismengen dienen, was die weitere Analyse erleichtert, da nur die Datensätze berücksichtigt werden, die einander überschneiden.

Um ein genaueres Ergebnis zu bekommen, kann die PostGIS-Funktion `ST_Distance(geometry, geometry)` verwendet werden, um festzustellen, wie weit die Länderpolygone tatsächlich voneinander entfernt sind. Im Ergebnis werden nur Länder mit gemeinsamen Grenzen ausgegeben. Beispiel 14-6 ist mit Beispiel 14-5 fast identisch, enthält aber eine weitere Bedingung in der `WHERE`-Klausel.

Beispiel 14-6: Die Verwendung der Funktion `ST_Distance(geometry, geometry)` zum Ermitteln benachbarter Polygone

```
world=# SELECT a.name
world)# FROM countries a,
world)# countries b
world)# WHERE b.name = 'Norway'
world)# AND a.wkb_geometry && b.wkb_geometry
world)# AND ST_Distance(a.wkb_geometry, b.wkb_geometry) = 0;
```

Beispiel 14-6: Die Verwendung der Funktion ST_Distance(geometry,geometry) zum Ermitteln benachbarter Polygone (Fortsetzung)

```
name
-----
Finland
Norway
Russia
Sweden
(4 Zeilen)
```

Der einzige Unterschied zum vorherigen Beispiel besteht in der letzten Zeile:

```
AND ST_Distance(a.wkb_geometry, b.wkb_geometry) = 0;
```

Da auch der Operator && weiterhin verwendet wird, wird ST_Distance(geometry,geometry) nur auf die Ergebnisse dieser Operation angewandt, also lediglich auf vier Datensätze. Latvia, Estonia sowie Saint Lucia und Svalbard sind nun nicht mehr Teil des Ergebnisses (Abbildung 14-8).

Achten Sie bei räumlichen Abfragen auf jedem Fall darauf, dass Ihre Geometriespalte über einen räumlichen Index (GIST) verfügt.

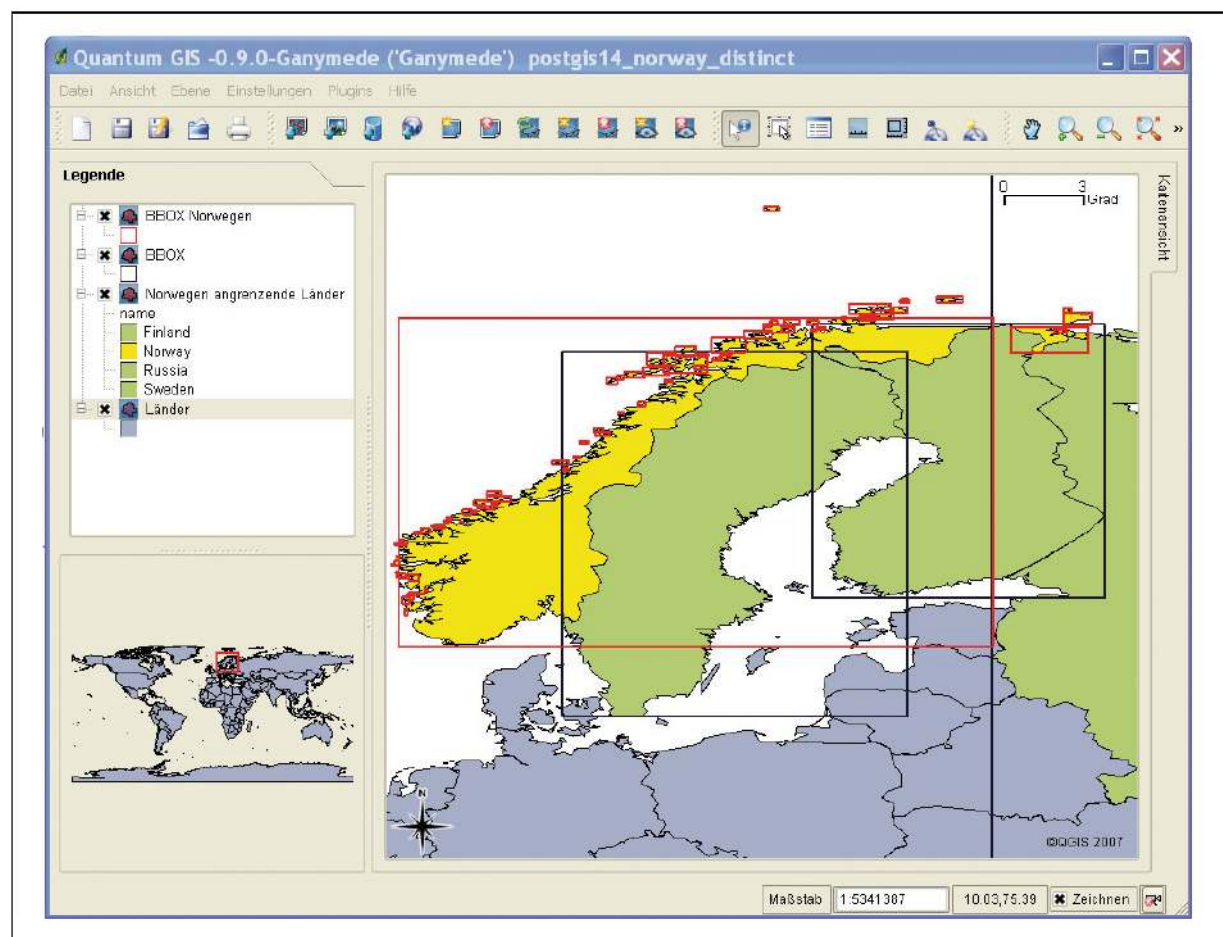


Abbildung 14-8: Das abschließende Ergebnis der Abfrage enthält nur noch Länder, die direkt an Norwegen angrenzen

Nutzen der PostGIS-Daten in anderen Applikationen

Die Visualisierung der Daten ist nicht immer das endgültige Ziel einer räumlichen Datenbankabfrage. Oftmals müssen Berichte oder Statistiken angefertigt werden. Abbildungen können jedoch Daten häufig besser darstellen, als dies mit einem einfachen Bericht möglich wäre. Daher dienen Karten oftmals als wichtiges Kommunikationsmittel.

Für die grafische Darstellung der PostGIS-Abfrageergebnisse gibt es verschiedene Möglichkeiten. Sie können beispielsweise ein Desktop-GIS verwenden oder die Daten über einen WMS oder WFS bereitstellen. Die Daten können aber auch für die Weiterverarbeitung in andere Formate überführt werden, z.B. GML, SVG, KML oder Shape.

Exportieren von PostGIS-Daten nach Shape oder GML

Das Werkzeug `ogr2ogr` ist in der Lage, PostGIS-Daten in viele andere Formate zu konvertieren. Hier ein Beispiel für den Export der Tabelle `countries` nach Shape:

```
> ogr2ogr -  
f "ESRI Shapefile" countries.shp "PG:dbname=world host=localhost user=postgres"  
countries
```

Mit dem Parameter `-f` wird angegeben, welches Ausgabeformat verwendet werden soll. Der Name der Ausgabedatei lautet in unserem Fall `countries.shp`. Die angesprochene Datenbank heißt `world`. Der letzte Parameter (`countries`) gibt an, welche Tabelle exportiert werden soll.

Die erzeugte Shape-Datei kann nun von einem beliebigen Programm, das Shape als Format unterstützt, geladen werden.

Der Aufruf zur Erzeugung einer GML-Datei sieht ähnlich aus:

```
> ogr2ogr -  
f "GML" countries.gml "PG:dbname=world host=localhost user=postgres" countries
```

Für den Export nach Shape können Sie auch das Hilfsprogramm `pgsql2shp` verwenden. Dieses steht Ihnen mit der Installation von PostGIS zur Verfügung. Hierbei sieht der Aufruf wie folgt aus:

```
> pgsql2shp -u postgres -P postgres -f export_countries world countries
```

PostGIS-Daten mit einem Desktop-GIS visualisieren

Eine Stärke von PostGIS besteht darin, dass viele Applikationen direkt auf die Daten zugreifen können. Dies haben wir schon beim Desktop-GIS Quantum GIS kennengelernt. Es gibt noch viele weitere Desktop-GIS, die PostgreSQL/PostGIS unterstützen (OpenEV, JUMP, udig, gvSIG). Schauen Sie sich dazu nochmals das Kapitel 8 an.

PostGIS-Daten mit MapServer darstellen

Viele MapServer-Anwendungen verwenden PostGIS als Datenquelle. Die Datenverwaltung wird in PostGIS vorgenommen, während MapServer die Visualisierung der Daten übernimmt.

In den Kapiteln 10, 11 und 12 wurde beschrieben, wie Sie eigene MapServer-Anwendungen erstellen können.

In einem Beispiel soll nun gezeigt werden, wie Sie die Weltkarte der Beispiel-Mapdatei nun nicht mehr über eine Shape-Datei als Datengrundlage visualisieren, sondern über die Daten aus der Datenbank.

Für den Zugriff auf PostGIS-Daten müssen zusätzliche Parameter in der Mapdatei angegeben werden.

```
CONNECTIONTYPE POSTGIS
CONNECTION "dbname=<Datenbankname> host=<Rechnername des Datenbankservers>
           user=<Datenbankbenutzer> password=<Passwort> port=<Standardport ist 5432>"
DATA "<Geometriespalte> from <Tabellenname>"
```

Der CONNECTION-Parameter enthält die Verbindungsdaten. Der Parameter CONNECTIONTYPE teilt MapServer mit, welche Art von Datenquelle geladen werden soll. Hier werden die für eine Verbindung zum PostgreSQL-Server nötigen Informationen angegeben. Der Standardport der PostgreSQL-Datenbank ist 5432 und sollte hier explizit angegeben werden. Je nach Konfiguration der PostgreSQL-Benutzerrechte (siehe `pg_hba.conf`) kann der Parameter `password` auch weggelassen werden.



Über drei Konfigurationsdateien können Sie PostgreSQL konfigurieren und den Zugriff durch andere Clients steuern. Die Konfigurationsdateien heißen `postgresql.conf`, `pg_hba.conf` und `pg_ident.conf`. Eine ausführliche Beschreibung finden Sie unter »III. Server Administration« in der PostgreSQL-Dokumentation: <http://www.postgresql.org/docs/>.

In Beispiel 14-7 sehen Sie den kompletten Aufbau eines Layers der Mapdatei.

Beispiel 14-7: Einfache UMN MapServer-Layerdefinition für den Zugriff auf PostGIS

```
LAYER
NAME countries
TYPE POLYGON
STATUS ON
CONNECTIONTYPE POSTGIS
CONNECTION "dbname=world user=postgres host=localhost port=5432"
DATA "wkb_geometry from countries"
PROJECTION
  "init=epsg:4326"
END
METADATA
WMS_TITLE "Countries"
WMS_SRS "EPSG:4326"
```


Beispiel 14-7: Einfache UMN MapServer-Layerdefinition für den Zugriff auf PostGIS (Fortsetzung)

```
END
CLASS
  STYLE
    COLOR 200 200 200
    OUTLINECOLOR 100 100 100
  END
END
END
```

Die Ebene aus diesem Beispiel ist ein Layer der Mapdatei *global.map*. Mit dem Kommandozeilenprogramm *shp2img* können Sie sich ein Kartenbild ausgeben lassen.

```
> shp2img -m global.map -o fig13-6.png
```

Weitere Informationen und Beispiele für die Verwendung dieses Befehls finden Sie in Kapitel 10.

Bei der Layerdefinition wurden die WMS-Parameter angegeben. Die Daten werden über einen WMS bereitgestellt. Sie können sich das Ergebnis daher auch mit einem WMS-Client wie z.B. Quantum GIS oder Mapbender anschauen, indem Sie den WMS in dem jeweiligen Programm laden. In Abbildung 14-9 sehen Sie, wie der WMS in Mapbender geladen wurde.

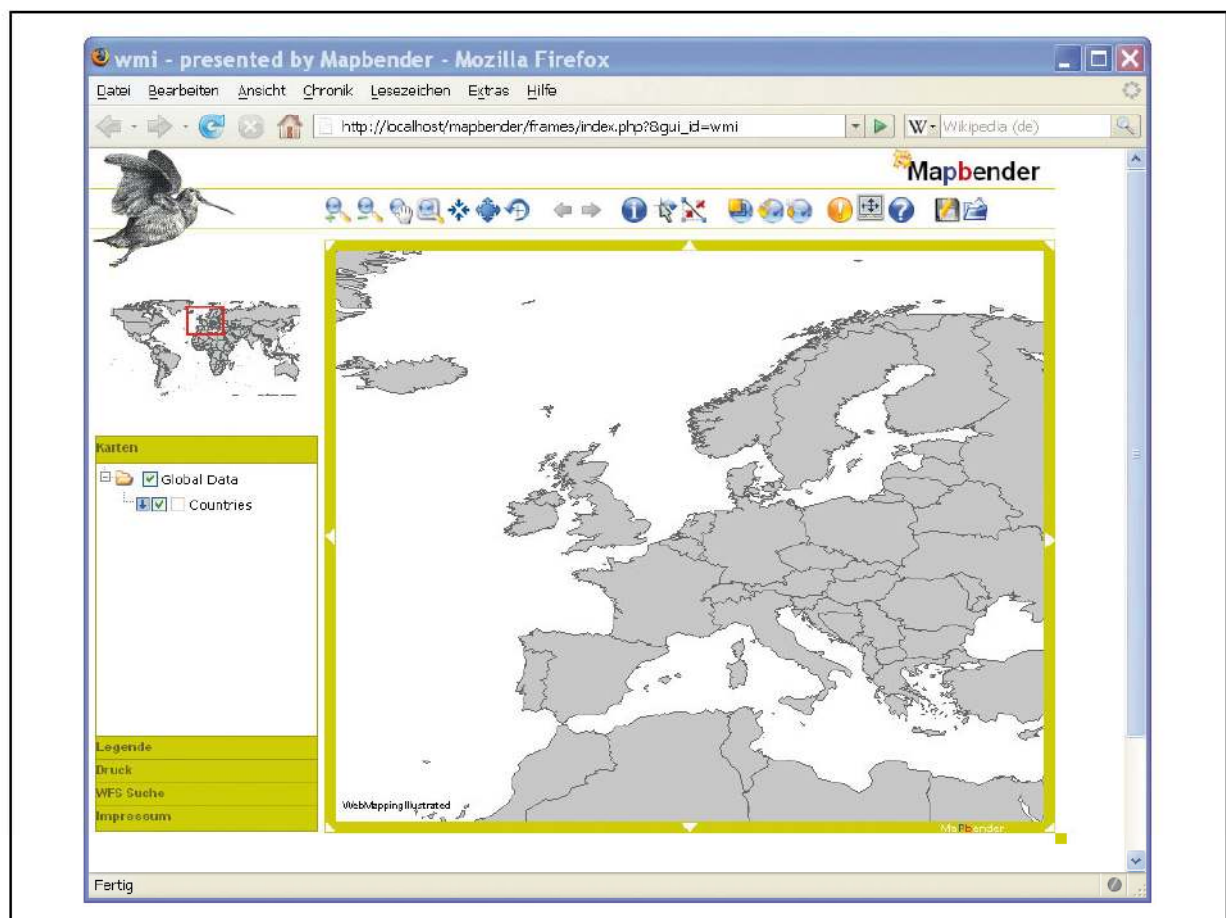


Abbildung 14-9: Visualisierung der Daten über einen MapServer-WMS mit Mapbender

Klassifizierung der Daten

Über den Parameter `EXPRESSION` im `CLASS`-Objekt können Sie die Daten klassifizieren (Beispiel 14-8). Das Ergebnis sehen Sie in Abbildung 14-10.



Weitere Informationen zum Parameter `EXPRESSION` finden Sie in Tabelle 10-1.

Beispiel 14-8: Über den Parameter `EXPRESSION` werden die Daten klassifiziert

```
LAYER
NAME countries
TYPE POLYGON
STATUS ON
CONNECTIONTYPE POSTGIS
CONNECTION "dbname=world user=postgres host=localhost port=5432"
DATA "wkb_geometry from countries"
PROJECTION
  "init=epsg:4326"
END
METADATA
  WMS_TITLE "Countries"
  WMS_SRS "EPSG:4326"
END
CLASS
  EXPRESSION ("[name]" = 'Norway')
  STYLE
    COLOR 255 255 0
    OUTLINECOLOR 255 0 0
  END
END
CLASS
  STYLE
    COLOR 100 100 100
    OUTLINECOLOR 100 100 100
  END
END
END
```

Einbinden von komplexeren Abfragen als Datenquelle

Die Abfragen, die über `DATA` angegeben werden, können auch komplexer sein. Sie müssen nicht immer alle Daten einer Tabelle anfordern. Wollen Sie beispielsweise nur die Länder mit einer Einwohnerzahl über 1 Milliarde anzeigen lassen, ist das Nachfolgende zu formulieren. Um die Lesbarkeit zu erhöhen, erfolgt die Angabe hier in mehreren Zeilen. Die Angabe muss in der Mapdatei aber in einer Zeile stehen:

```
DATA "wkb_geometry from (
  select ogc_fid, wkb_geometry, name, total
  from countries
  where total > 1000000000)
as foo USING UNIQUE ogc_fid USING SRID=4326"
```

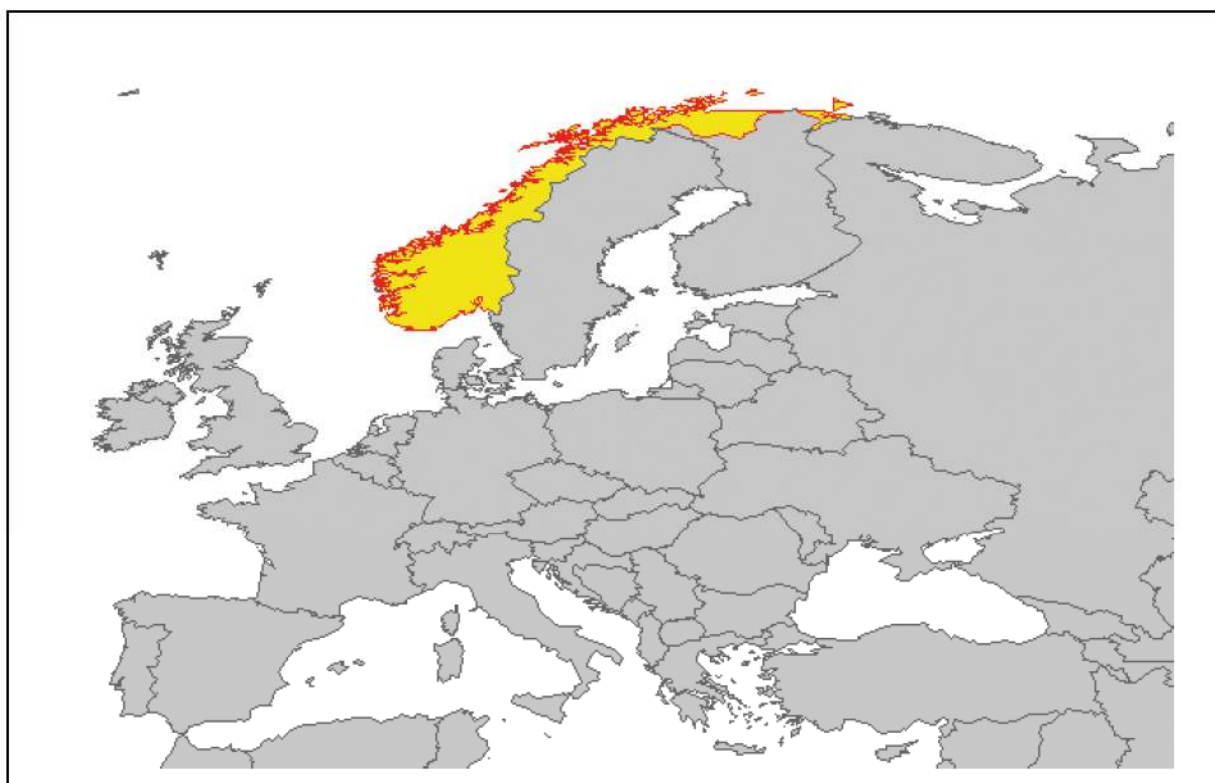


Abbildung 14-10: Anzeige von Norwegen über eine Klassifizierung der Daten in der MapServer-Mapdatei

Wie Sie sehen, weicht die Formulierung etwas von der bisherigen Angabe ab. Die Abfrage wird in runden Klammern angegeben. Das Schlüsselwort `foo` ist einfach ein Platzhalter. Sie können den Namen des Platzhalters frei wählen, müssen diesen aber angeben. MapServer benötigt zwei weitere Informationen. Jede von PostGIS zurückgelieferte Datenbankspalte muss eindeutig referenziert sein. Über `USING UNIQUE <Spaltenname>` muss die Tabellenspalte angegeben werden, die als Index verwendet werden soll, in unserem Fall die Spalte `ogc_fid`.

MapServer benötigt zusätzlich Informationen über die verwendete Projektion der Daten. Diese geben Sie über `USING SRID = <SRID>` an.

In der Abfrage dürfen sowohl PostGIS- als auch PostgreSQL-Funktionen verwendet werden. Sie können Sichten (Views) erzeugen und diese einbinden. Und Sie können mehrere Tabellen in einer Abfrage miteinander verknüpfen.

Dies ist ein Beispiel für die Verwendung einer PostGIS-Funktion:

```
DATA "wkb_geometry from ( SELECT ogc_fid, name, wkb_geometry from countries WHERE wkb_
geometry && ST_GeomFromText('POINT(13.5 57.9)',4326) ) as foo USING UNIQUE ogc_fid
USING SRID = 4326"
```

Wir verwenden den PostGIS-Vergleichsoperator (`&&`), um zu ermitteln, in welchen Umgebungsrechtecken der Geometrien der Tabellen `countries` sich ein bestimmter Punkt befindet. Der Punkt wurde dabei dynamisch über die Funktion `ST_GeomFromText(text,srid)` erstellt.

Mithilfe der PostGIS-Funktionen können Sie auch neue Geometrien erzeugen, die dann über MapServer visualisiert werden können. Mit der Funktion `ST_Buffer(geometry, double, [integer])` können Sie beispielsweise einen Puffer um eine Geometrie erzeugen.

Dazu legen Sie mit den gleichen Angaben wie in Beispiel 14-8 einen neuen MapServer-Layer an.

Hier sehen Sie ein Beispiel, in dem die Funktionen `ST_Buffer(geometry, double, [integer])` und `ST_Simplify(geometry, tolerance)` gemeinsam verwendet werden.

```
DATA "wkb_geometry from (  
  SELECT ogc_fid, ST_Buffer( ST_Simplify(wkb_geometry,0.5), 0.2)  
    AS wkb_geometry  
  FROM countries  
  WHERE name='Switzerland') as foo  
USING UNIQUE ogc_fid USING SRID = 4326"
```

Beide Funktionen erwarten eine Geometrie und einen numerischen Wert als Funktionsparameter. Die numerischen Werte werden in der Maßeinheit des verwendeten Projektionssystems angegeben. Die Funktion `ST_Simplify(geometry, tolerance)` sorgt dafür, dass, basierend auf dem von Ihnen angegebenen numerischen Wert, bestimmte Vertizes ignoriert werden. In diesem Beispiel beträgt die Toleranz 0.5 Grad. Die vereinfachte Geometrie wird dann an die Funktion `ST_Buffer(geometry, double, [integer])` weitergegeben. Diese Funktion erzeugt eine Fläche mit einer Breite von 0.2 Grad um die Geometrie.

Beschriftungen über PostGIS-Attribute

Wie bei anderen MapServer-Datenquellen, die über Attribute verfügen, können auch Informationen aus PostGIS-Ebenen verwendet werden, um die Beschriftungen für eine Karte zu erstellen. Beispiel 14-9 entspricht im Prinzip Beispiel 14-8. Allerdings wurden hier die für die Beschriftung nötigen Parameter ergänzt.

Beispiel 14-9: Durch die Angabe des Parameters LABELITEM und des LABEL-Objekts können Attribute zur Beschriftung genutzt werden

```
LAYER  
  NAME sweden  
  TYPE POLYGON  
  STATUS ON  
  CONNECTIONTYPE POSTGIS  
  CONNECTION "dbname=world user=postgres host=localhost port=5432"  
  DATA "wkb_geometry from (select ogc_fid, name, wkb_geometry from countries where name=  
    'Sweden') as foo USING UNIQUE ogc_fid USING SRID =4326"  
  PROJECTION  
    "init=epsg:4326"  
  END  
  METADATA  
    WMS_TITLE "Sweden"  
    WMS_SRS "EPSG:4326"  
  END  
  LABELITEM "name"  
  CLASS
```


Beispiel 14-9: Durch die Angabe des Parameters LABELITEM und des LABEL-Objekts können Attribute zur Beschriftung genutzt werden (Fortsetzung)

```
EXPRESSION ("[name]" = 'Sweden')
STYLE
  COLOR 0 255 255
  OUTLINECOLOR 0 0 255
END
LABEL
  COLOR 0 0 255
  TYPE TRUETYPE
  FONT ARIAL
  SIZE 14
END
END
END
```

In Beispiel 14-9 wurde zusätzlich ein LABEL-Block für eine Beschriftung definiert. Wenn Sie PostGIS-Daten ansprechen und zur Beschriftung verwenden wollen, müssen Sie Folgendes beachten: Die unter LABELITEM angegebene Beschriftungsspalte muss in der Abfrage, die über DATA angegeben wird, vorhanden sein. Die Abfrage in Beispiel 14-8 spricht die gesamte Tabelle an, hier stehen alle in der Tabelle enthaltenen Attribute für die Beschriftung zur Verfügung. In Beispiel 14-9 taucht die Spalte name in der Abfrage auf und kann so für die Beschriftung verwendet werden.

Eine weitere Schwierigkeit beim Erzeugen von Beschriftungen hat mit der Groß- und Kleinschreibung der Feldnamen zu tun. In PostgreSQL (wie in anderen Datenbanken) können neben den Feldnamen ohne explizite Definition der Groß- und Kleinschreibung auch Felder definiert werden, die Groß- und Kleinbuchstaben im Feldnamen enthalten. Sämtliche Feldnamen in den hier gezeigten Beispielen verwenden Kleinschreibung. Einige Werkzeuge erzeugen allerdings Feldnamen, die Großbuchstaben enthalten. Werden Großbuchstaben verwendet, muss der Parameter in doppelten Anführungszeichen in der Mapdatei angegeben werden. Das kann leicht zu Verwirrung führen, wenn Sie bereits doppelte Anführungszeichen um einen Feldnamen verwenden, wie z.B. bei LABELITEM "name" in Beispiel 14-8. Ist der Feldname name in der Datenbank in Großbuchstaben angegeben, muss er mit einem zusätzlichen Paar einfacher Anführungszeichen umgeben werden, wie hier: LABELITEM '"NAME"'.



Die hier besprochenen Besonderheiten für Feldnamen mit Großbuchstaben gelten für alle LAYER-Parameter, die sich auf PostGIS-Felder beziehen. Die Regeln gelten also auch für die Parameter DATA, CLASSITEM und EXPRESSION. Sie erleichtern den Umgang mit PostGIS-Daten, indem Sie alle Tabellen- und Spaltennamen in Kleinbuchstaben angeben.

ogr2ogr behält standardmäßig die Schreibweise der Tabellen- und Spaltennamen der Datenquelle bei. Wenn Sie ogr2ogr zum Laden der Daten verwenden, können Sie über eine Option angeben, dass Großschreibung für die Tabellen- und Spaltennamen ignoriert wird. Die Option lautet -lco LAUNDER=YES.

PostGIS-Support durch weitere Anwendungen

Auch wenn wir in diesem Buch nicht weiter darauf eingehen werden, gibt es noch eine Reihe weiterer Open Source-Anwendungen, die PostGIS unterstützen, beispielsweise OpenEV, uDIG, JUMP, Thuban, gvSIG, GRASS GIS, GeoServer und deegree.

Auch einige kommerzielle Anbieter haben ihre Programme mit PostGIS-Unterstützung versehen. Die Feature Manipulation Engine (FME) unterstützt das Lesen und Schreiben von PostGIS-Datensätzen. Dieses mächtige Werkzeug vereinfacht die Migration nach PostgreSQL/PostGIS. FME unterstützt die Umwandlung Dutzender Vektor- und Datenbankformate (<http://safe.com>).

Der GDV Spatial Commander (<http://www.gdv.com/download/scommander.php>) kann PostgreSQL/PostGIS-Daten visualisieren, manipulieren sowie importieren/exportieren.

Auch das Spatial Information System (SIS) von Cadcorp bietet ab Version 6.1 Unterstützung für PostGIS. Weitere Informationen finden Sie unter <http://www.cadcorp.com>.

Selbst wenn Sie PostGIS verwenden, kann es sein, dass Ihre Firma daneben noch die proprietären ESRI-Werkzeuge einsetzt. Eine häufige Frage von ESRI ArcGIS-Benutzern lautet: Wie kann ich mit ArcGIS auf PostGIS-Daten zugreifen? Zu diesem Thema gibt es ein Open Source-Projekt mit dem Namen PostGIS-ArcMap Connector, kurz: PgArc. Weitere Informationen hierzu finden Sie unter <http://pgarc.sourceforge.net/>. PgArc automatisiert den Import von PostGIS-Tabellen in temporäre Shape-Dateien und lädt sie dann in ArcMap. Die Ebenen können nach der Bearbeitung wieder in der PostGIS-Datenbank gespeichert werden, wobei die ursprüngliche Tabelle überschrieben wird. Ab 2008 soll es möglich sein, PostGIS-Daten über eine ESRI ArcSDE (siehe Anhang B) in ArcMap einzubinden.

Programmieren mit MapServer MapScript

Wenn Sie Programmiererfahrung haben, möchten Sie MapServer möglicherweise individuell anpassen. Sie möchten die MapServer-Funktionalitäten direkt in Ihre eigenen Applikationen einbauen und von dort aus steuern. Dies können Sie mithilfe von MapScript realisieren. MapScript ist die MapServer-Programmierschnittstelle (API – Application Program Interface). Dieses Kapitel soll Ihnen einen kurzen Einstieg in die Programmierung mit MapScript geben.

Was ist MapScript?

Über MapScript wird die Funktionalität von MapServer einer Vielzahl von Skriptsprachen verfügbar gemacht. Durch die Nutzung der MapServer-API können Sie einer Applikation leicht interaktive Karten hinzufügen. MapScript bietet eine Reihe von Objekten an, die zur Erzeugung der Karten in der Programmiersprache Ihrer Wahl verwendet werden können. Analog zum Aufbau der Mapdatei stehen z.B. Map-, Layer-, Klassen-, Style-Objekte usw. zur Verfügung. Die Eigenschaften der Objekte können mit einer Vielzahl an Methoden bearbeitet werden. Sie können bequem die Geodaten ansprechen und weiterhin die Programmiersprache Ihrer Wahl verwenden. MapScript ergänzt als Modul den Funktionsumfang der jeweiligen Skriptsprache. Sie können bequem mit den Geodaten interagieren und alle gewohnten Funktionen der Sprache uneingeschränkt weiter benutzen.

MapScript ist in der Lage, Mapdateien zu laden, zu bearbeiten und anzulegen. Mit MapScript können Sie Ebeneneinstellungen problemlos verändern, Sie können dynamisch neue Ebenen oder Layer erzeugen, Sie können die Klassen innerhalb der Mapdatei dynamisch anpassen, Ausgabebilder erzeugen, räumliche Daten exportieren (z.B. nach Shape) und vieles mehr. Außerdem haben Sie die Möglichkeit, über MapScript die GEOS-Funktionen zur Geometriebearbeitung zu nutzen und beispielsweise Puffer zu erzeugen. Da MapScript innerhalb einer bestehenden Umgebung verwendet wird, kann es mit anderen Funktionen der entsprechenden Sprache kombiniert werden.

Einige Programmiersprachen unterstützen MapScript. Hier sind PHP, Python, Perl, Java, Ruby sowie C# zu nennen.

Woher bekommen Sie MapScript?

MapScript können Sie auf verschiedene Weise bekommen: Es ist zum einen im MapServer-Quellcode enthalten, und im MS4W-Paket liegt MapScript ebenfalls vor. Sie finden Quellcode und Pakete unter <http://mapserver.gis.umn.edu/download>. Die wichtigste Voraussetzung für die Verwendung von MapScript ist, dass die Skriptsprache auf dem Rechner verfügbar ist. Möglicherweise werden auch die Header-Dateien für die entsprechende Sprache benötigt, wenn Sie die MapScript-Module selbst aus dem Quellcode kompilieren wollen.

MapScript selbst kompilieren

Wir werden hier nicht darauf eingehen, wie Sie MapScript für die jeweiligen Programmiersprachen kompilieren können. Es gibt allerdings eine gemeinsame Vorgehensweise.

Um MapScript für eine bestimmte Programmiersprache zu kompilieren, muss die Unterstützung für diese Sprache bei der Kompilierung von MapServer angegeben werden. Hierdurch werden im nächsten Schritt die benötigten Dateien in den MapScript-Verzeichnissen der einzelnen Sprachen eingerichtet.

In der MapServer-Verzeichnisstruktur gibt es das Verzeichnis `mapscript`. Dieses enthält für jede Programmiersprache, die MapScript unterstützt, ein eigenes Unterverzeichnis. Einige dieser Unterverzeichnisse enthalten bereits das für `make` benötigte Makefile, andere enthalten eine Reihe von Setup-Skripten. Durch das Ausführen dieser Dateien erhalten Sie (je nach Programmiersprache) die MapScript-Module oder -Bibliotheken. Oftmals helfen die beiliegenden Skripten auch bei der Installation der Erweiterungen. Für Python können Sie MapScript beispielsweise mit folgenden Befehlen kompilieren und installieren:

```
> python setup.py build
> python setup.py install
```

Neben den Sprachverzeichnissen gibt es noch eine Reihe weiterer Dateien und Ordner, die Teil des Simplified Wrapper and Interface Generator (SWIG) sind. Durch die Verwendung von SWIG kann MapScript für verschiedene Sprachen portiert werden. Diese Methode ist mittlerweile für viele der angebotenen Skriptsprachen (z.B. Python, Java) der Standard für die Erstellung der MapScript-Erweiterung. PHP beschreitet hier aber derzeit noch einen eigenen Weg. Die Syntax in den Funktionen und Schnittstellen von PHP MapScript kann sich leicht von denen der SWIG-Implementierungen unterscheiden. Zum Teil existieren in den verschiedenen Ausprägungen von MapScript Funktionen, die in der jeweils anderen fehlen.

Wenn Sie sich mit SWIG auskennen und eine `mapscript_wrap.c`-Datei für eine neue Sprache erstellen möchten, ist Ihr Beitrag sicherlich willkommen. Weitere Informationen zu SWIG finden Sie unter <http://www.swig.org>.

Dies sind Links zu den jeweiligen Beschreibungen der MapScript-Schnittstellen:

<http://mapserver.gis.umn.edu/docs/reference/phpmapscript-class> (PHP)

<http://mapserver.gis.umn.edu/docs/reference/mapscript> (SWIG)

Kompilierungsanweisungen finden Sie in den README-Dokumenten innerhalb des MapScript-Verzeichnisses für die betreffende Sprache.

Weitere Hinweise zur Installation von MapScript finden Sie unter:

<http://mapserver.gis.umn.edu/docs/howto/phpmapscript-install/> (PHP)

<http://mapserver.gis.umn.edu/docs/howto/javamapscript> (Java)

Bereitstellung von MapScript-Versionen

Anstatt MapScript selbst zu kompilieren, können Sie auch – je nach Programmiersprache – auf eine binäre Version zurückgreifen. Eine genauere Beschreibung liefert die MapServer-Website. Auf den folgenden Sites finden Sie unterschiedliche Versionen von MapScript:

- Die Firma DM Solutions pflegt PHP MapScript. Seit der MapServer-Version 4.8.2 stehen binäre PHP MapScript-Versionen über das MapServer for Windows- (MS4W-)Paket <http://maptools.org/ms4w/> zur Verfügung.
- Für Linux ist MapScript im FGS Linux Installer enthalten: <http://www.maptools.org/fgs/> (inkl. PHP MapScript).
- Das FWTools-Paket von Frank Warmerdam enthält eine Binärversion von MapScript für Python. Es werden sowohl Windows- als auch Linux-Plattformen unterstützt. Siehe <http://fwtools.maptools.org>.

Unterstützung im Umgang mit MapScript

Die meisten MapScript-Anfänger brauchen anfangs ein wenig Unterstützung. Die Suche auf der MapServer-Website liefert viele Ergebnisse zum Thema MapScript (<http://mapserver.gis.umn.edu/search?SearchableText=MapScript>). Hier finden Sie auch Installationsanleitungen, eine Beschreibung der Schnittstelle und Codebeispiele. Ansonsten erhalten Sie sicherlich über die MapServer-Mailinglisten Unterstützung. Dieses Kapitel enthält einfache Beispiele, die Ihnen den Einstieg erleichtern sollen.

Die MapScript-Objekte

Eine vollständige Übersicht über die Objekte, Variablen und Methoden in MapScript finden Sie auf der MapServer-Website unter <http://mapserver.gis.umn.edu/docs/reference/phpmapscript-class> oder in der MapScript-Referenz, die von Sean Gillies zusammengestellt wurde: <http://mapserver.gis.umn.edu/docs/reference/mapscript>.

Klassendiagramme für MapScript finden Sie unter den folgenden Adressen:

http://www.omsug.ca/dl/osgis2004/php_mapscript_mum2.ppt (PHP MapScript)

http://www2.dmsolutions.ca/mapserver/dl/mapserver_40_uml.zip

Auf die Klassen innerhalb einer Mapdatei kann über Objekte in MapScript zugegriffen werden. Tatsächlich gibt es sogar mehr Objekte, als Sie normalerweise zu Gesicht bekommen, da MapServer einen Großteil der Arbeit hinter den Kulissen durchführt. Der wichtigste Teil einer MapScript-Anwendung ist das Map-Objekt (`mapObj`). Das `mapObj` kann ein oder mehrere Ebenen-Objekte (`layerObj`) enthalten sowie ein Legenden-Objekt (`legendObj`), ein Maßstabsleisten-Objekt (`scalebarObj`) und ein Objekt zur Erstellung von Referenzkarten (`referenceMapObj`). In Abbildung 15-1 sehen Sie ein vereinfachtes Diagramm, das die hierarchische Struktur der Objekte widerspiegeln soll.

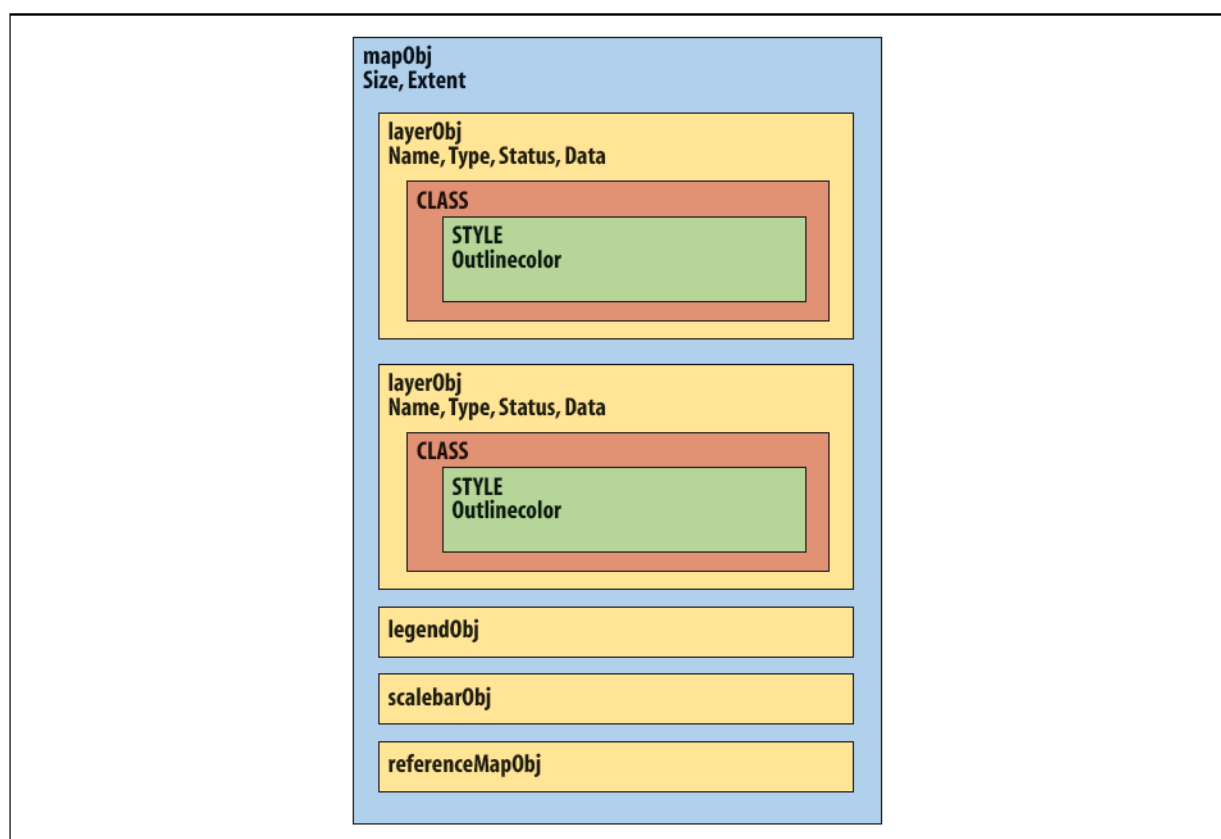


Abbildung 15-1: Die hierarchische Objektstruktur der MapScript-API

Das zentrale Objekt ist `mapObj`. Diesem Objekt können die Objekte `layerObj`, `classObj` und `styleObj` hinzugefügt werden. Andere Objekte, wie `webObj`, `scalebarObj`, `legendObj` und `colorObj`, werden nicht explizit angelegt, sie sind bereits Teil von `mapObj`. Wenn Sie diese Objekte dennoch manuell erzeugen wollen, werden Sie auf Fehler stoßen.

Sämtliche Parameter, die Sie in einer Mapdatei setzen können, lassen sich auch über MapScript steuern.

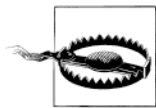
Die Verwendung der MapScript-API lässt sich am besten anhand einiger einfacher Beispiele erklären.

Beispiele für die Verwendung von MapScript

Auch beim Schreiben von MapScript-basierten Anwendungen spielen Mapdateien eine wesentliche Rolle. Wenn Ihnen bereits eine vorkonfigurierte Mapdatei zur Verfügung steht, so können Sie diese sofort mit MapScript bearbeiten. MapScript bietet aber auch die Möglichkeit, Map-Objekte, die dem Inhalt einer Mapdatei entsprechen, dynamisch zu generieren und daraus eine Karte zu erzeugen.

Viele Programmiersprachen sind in der Lage, MapScript zu verwenden. In den hier ausführlich erklärten Beispielen beschränken wir uns allerdings auf Python und PHP. Mit den folgenden Beispielen sollen die Möglichkeiten, die MapScript bietet, vorgestellt werden. Dabei werden die Beispiele zuerst mithilfe von Python und MapScript ausführlich erklärt. Anschließend werden die Skripten für den Einsatz von PHP und MapScript aufgeführt. Hierbei wird in einem zusätzlichen Beispiel aufgezeigt, wie über MapScript ein WMS dynamisch auch ohne Mapdatei erzeugt werden kann.

Um Python, Perl, PHP, Java und Ruby vergleichen zu können, wird am Ende dieses Kapitels jeweils noch ein Beispiel in der jeweiligen Sprache aufgeführt.



Um die Beispiele so einfach wie möglich zu halten, haben wir auf Fehlerüberprüfungen, Optimierungen und Modularisierung verzichtet. Diese Beispiele sind in der hier gezeigten Form nicht für den Einsatz in einer Produktionsumgebung gedacht.

Beispiel-Mapdatei

Die ersten beiden Beispiele verwenden eine einfache Mapdatei (`global.map`), die in Beispiel 15-1 aufgeführt ist. Im dritten Beispiel verwenden wir MapScript, um eine eigene Mapdatei zu erzeugen. Um das Beispiel kurz und einfach zu halten, wurden optionale Bestandteile wie Maßstabsleiste, Legende und Referenzkarte weggelassen.

Beispiel 15-1: Eine einfache Mapdatei `global.map`, die in den folgenden Beispielen verwendet wird

```
MAP
SIZE 600 300
EXTENT -180 -90 180 90
IMAGECOLOR 180 180 250
IMAGETYPE PNG

UNITS DD
WEB
  IMAGEPATH "/srv/www/htdocs/tmp/"
  IMAGEURL "/tmp/"
END

LAYER
  NAME countries
  TYPE POLYGON
  STATUS DEFAULT
```

Beispiel 15-1: Eine einfache Mapdatei *global.map*, die in den folgenden Beispielen verwendet wird (Fortsetzung)

```
DATA countries_simpl
CLASS
  NAME 'Countries'
  OUTLINECOLOR 100 100 100
  COLOR 200 200 200
END
END
END
```



Die verwendeten Daten mit den Polygonen der Länder können Sie von http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.1_simpl.tar.gz herunterladen.

Die komprimierte Datei enthält ein Shape-Datei mit dem Namen *countries_simpl*, die Sie schon aus den vorangegangenen Kapiteln dieses Buchs kennen. In der verwendeten Mapdatei wird davon ausgegangen, dass sich die *countries_simpl.**-Dateien im selben Verzeichnis wie die Mapdatei befinden.

Die Karte, die aus der in Beispiel 15-1 aufgeführten Mapdatei erzeugt wird, können Sie in Abbildung 15-2 sehen.

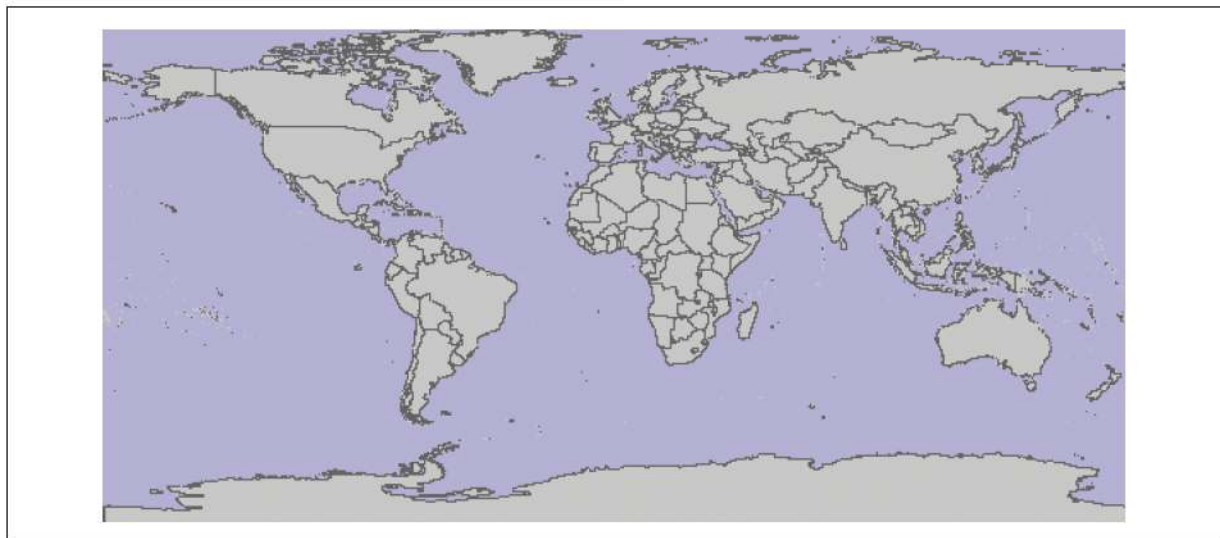


Abbildung 15-2: Einfache Weltkarte, die über die Mapdatei aus Beispiel 15-1 erstellt wurde

Einsatz von MapScript mit Python

In den folgenden Beispielen soll unter der Verwendung von Python der Einsatz von MapScript verdeutlicht werden. Die Prozesse werden dabei ausführlich erläutert, die Objekte vorgestellt.

Öffnen der Mapdatei und Zeichnen der Karte

Die Grundfunktionalität von MapServer besteht darin, eine Mapdatei einzulesen und aus den Informationen ein Kartenbild zu erzeugen. Dies lässt sich mit MapScript in wenigen Arbeitsschritten automatisieren, wie Beispiel 15-2 zeigt.

Beispiel 15-2: Das Öffnen einer Mapdatei und das Speichern der Karte in einer Datei

```
# map1.py
# Python MapScript Beispiel 1

import mapscript

# Angabe der Mapdatei, die verwendet werden soll
mapfile = "global.map"

# mapObj erzeugen, Initialisierung mit der Mapdatei
mapobject = mapscript.mapObj(mapfile)

# imgObj erzeugen, das ein Bild der Karte beinhaltet
mapimage = mapobject.draw()

# Speichern des Kartenbilds in eine Datei
mapimage.save("worldmap.png")
```

Die Zeilen, die in Beispiel 15-2 mit einem Doppelkreuz (#) beginnen, kennzeichnen Kommentare, die die jeweiligen Prozesse beschreiben. In anderen Programmiersprachen werden für Kommentare andere Zeichen benutzt. Wie Sie sehen, ist dieses Beispiel äußerst kurz. Hier eine ausführliche Beschreibung der einzelnen Codezeilen:

```
mapfile = "global.map"
```

Dieses Zeile greift noch nicht auf MapScript zu. Stattdessen wird der Variablen `mapfile` die Zeichenkette `global.map` zugewiesen. Sie bezeichnet den Namen der Mapdatei, die im folgenden Schritt geladen wird.

```
mapobject = mapscript.mapObj(mapfile)
```

Diese Zeile legt ein Objekt an, das den Kern unserer Applikation bilden wird – das sogenannte Map-Objekt, kurz `mapObj`. Entsprechend der Mapdatei (die das Map-Objekt und vermutlich eine Reihe von Ebenen, Projektionen und andere Einstellungen enthält) stellt `mapObj` die höchste Ebene der MapScript-Objekthierarchie dar. Alle anderen in diesem Beispiel verwendeten Objekte werden von `mapObj` abgeleitet.

Für `mapObj` stehen nun zahlreiche Attribute und Methoden zur Verfügung. In diesem Beispiel wird allerdings nur eine Methode verwendet.

```
mapimage = mapobject.draw()
```

Wird `mapobject` mit der Methode `draw` aufgerufen, erhalten wir ein Image-Objekt (`imgObj`) zurück, das in der Variablen `mapimage` gespeichert wird.

```
mapimage.save("worldmap.png")
```

Im letzten Arbeitsschritt verwenden wir die Methode `save()`, um das Bild in einer Datei zu speichern. Der Name für die Bilddatei wird hier direkt als String an die Methode übergeben. Es ist natürlich auch möglich, den Dateinamen in einer separaten Variablen, ähnlich `mapfile`, zu speichern.



Geben Sie in der Methode `save()` für den Dateinamen einen leeren String an, gibt MapScript das Bild über die Standardausgabe (`stdout`) aus. Abhängig von der verwendeten Programmiersprache und davon, wie das Programm ausgeführt wird, kann dies ein Terminal, ein Interpreter, eine Shell oder beispielsweise eine Webseite sein.

Und das ist auch schon alles! Mit diesem einfachen Beispiel sind Sie bereits in der Lage, die Funktionen von MapServer in Ihre eigenen Applikationen zu integrieren.

Manipulation der Eigenschaften des Map-Objekts

Beispiel 15-3 zeigt, wie MapScript für Python mit dem Map-Objekt interagiert. In diesem Beispiel legen wir ein `class`-Objekt an. Über die Eigenschaft `Expression` des `class`-Objekts wird ein bestimmtes Land in der Karte hervorgehoben.

Beispiel 15-3: Anlegen eines class-Objekts, Auswahl eines Landes und Darstellung des Landes in einer anderen Farbe

```
# map2.py
# Python MapScript Beispiel 2

import mapscript

# Angabe der Mapdatei, die verwendet werden soll
mapfile = "global.map"

# mapObj erzeugen, Initialisierung mit der Mapdatei
mapobject = mapscript.mapObj(mapfile)

# Auswahl des ersten Layers, Definition des Attributs,
# das klassifiziert werden soll
layer1 = mapobject.getLayer(0)
layer1.classitem = "NAME"

# Auswahl der ersten Klasse des Layers und Duplizieren dieser Klasse
classCanada = layer1.getClass(0)
layer1.insertClass(classCanada)

# Vergabe eines Namens für die neue Klasse
classCanada.name = "Canada"

# Setzen der Eigenschaft Expression,
# um nur eine Auswahl der Objekte zu zeichnen
classCanada.setExpression("Canada")
```

Beispiel 15-3: Anlegen eines class-Objekts, Auswahl eines Landes und Darstellung des Landes in einer anderen Farbe (Fortsetzung)

```
# Auswahl des Style-Objekts und Verändern der Farbe
canadaStyle = classCanada.getStyle(0)
canadaStyle.color.setRGB(250, 250, 125)

# imgObj erzeugen, das ein Bild der Karte beinhaltet
mapimage = mapobject.draw()

# Speichern des Kartenbilds in eine Datei
mapimage.save("worldmap.png")
```

Dieses Beispiel verändert die Eigenschaften der Ebene, fügt eine neue Klasse hinzu und verändert die Darstellungsfarbe für das betreffende Land. Hier zeigt sich das wohl mächtigste Leistungsmerkmal: die Fähigkeit, Filter anzulegen oder zu verändern, die einschränken, welche Objekte gezeichnet werden sollen. Die resultierende Karte sehen Sie in Abbildung 15-3.

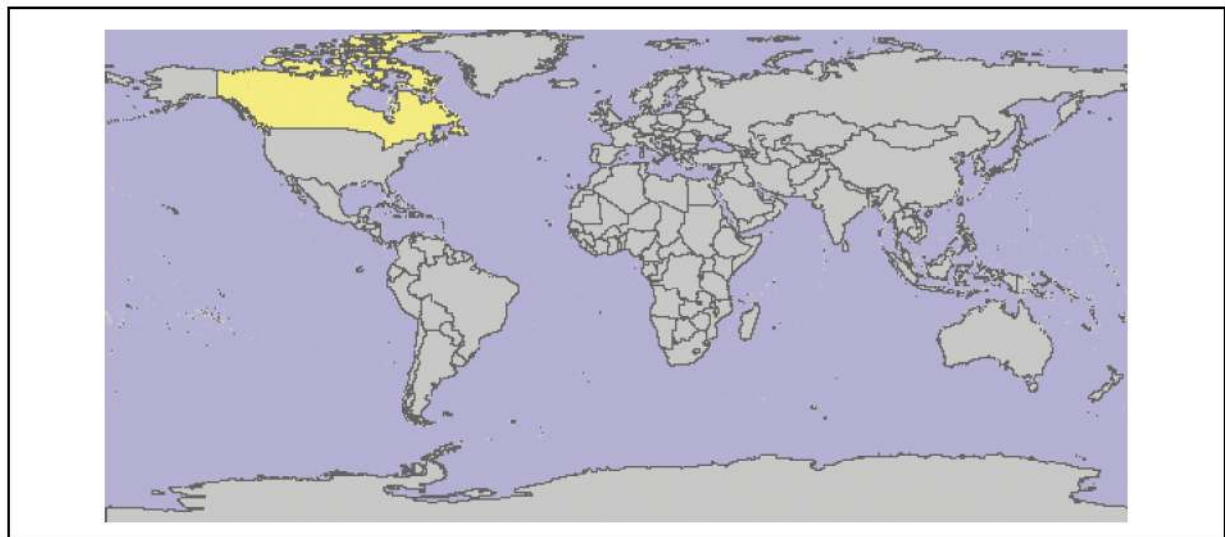


Abbildung 15-3: Die aus Beispiel 15-3 resultierende Karte. Über die zweite Klasse, die einen Filter enthält, wird lediglich Kanada dargestellt.

Es folgt eine genauere Betrachtung von Beispiel 15-3.

```
layer1 = mapobject.getLayer(0)
layer1.classitem = "NAME"
```

Die Einstellung der Eigenschaft `classitem` ist die erste Zeile, um die unser ursprüngliches Beispiel erweitert wurde. Genau wie das manuelle Setzen von `CLASSITEM` in der Mapdatei ermöglicht diese Angabe, einen Filter anzugeben und darüber zu definieren, welche Objekte über diese Klasse angezeigt werden sollen. In unserem Beispiel wird überprüft, ob das Attribut `"NAME"` dem als Expression angegebenen Wert entspricht. Der Filter wird erst später über die Funktion `setExpression()` gesetzt.

```
classCanada = layer1.getClass(0)
layer1.insertClass(classCanada)
```

Die erste Zeile spricht die erste für die Ebene definierte Klasse (0) an. In der hier verwendeten Mapdatei wurde nur eine Klasse definiert. Mithilfe der Funktion `insertclass()` wird anschließend eine neue Klasse erzeugt, indem die ursprüngliche Klasse einfach kopiert wird. Ziel dieses Beispiels ist es, Kanada innerhalb der Weltkarte hervorheben. Daher wird die ursprüngliche Klasse ohne Änderungen beibehalten. Aus der angelegten Kopie filtern wir dargegeben alle Länder außer Kanada aus.

```
classCanada.name = "Canada"  
classCanada.setExpression("Canada")
```

Wenn Sie die Eigenschaft `.name` einer Klasse setzen, wird diese Klasse bei der Erzeugung der Legende berücksichtigt. In diesem Beispiel wird zwar keine Legende erzeugt, Sie sollten dennoch für jede Klasse immer die Eigenschaft `.name` setzen, um einen späteren Zugriff auf die Klasse zu ermöglichen.

Die wichtigste Angabe in diesem Beispiel ist der Aufruf der Funktion `setExpression()`. Möglicherweise hätten Sie hier eine Art logische Anweisung, wie etwa eine SQL-Anweisung, erwartet:

```
...where country = 'Canada'
```

Auch diese Art von Anweisung ist möglich. In diesem Beispiel zeigen wir jedoch eine einfachere Methode. Der Wert, der über `setExpression()` gesetzt wird, wird hier zusammen mit der Eigenschaft `classitem` für die Ebene benutzt. Hieraus ergibt sich die Anweisung `[NAME] = "Canada"`, um Objekte auszuschließen, auf die dieses Kriterium nicht zutrifft.

Es ist auch möglich, Anweisungen wie diese direkt mit `setExpression()` anzugeben, ohne `classitem` für die Ebene zu benutzen. Allerdings ist es nicht immer ganz einfach, die Syntax richtig anzugeben, da diese zusätzlich von der verwendeten Datenquelle abhängig ist. Je einfacher Sie Ihre Syntax halten können, desto besser. Weitere Informationen zur Angabe von `EXPRESSION` finden Sie in Kapitel 10.

```
canadaStyle = classCanada.getStyle(0)  
canadaStyle.color.setRGB(250, 250, 125)
```

Die letzten beiden Zeilen ändern lediglich die Farbe der Objekte, die über diese Klasse dargestellt werden. Die Farbe wird über die Angabe der Werte für Rot, Grün und Blau auf Hellgelb gesetzt.

Dynamisches Erzeugen von Mapdateien

Das folgende Beispiel 15-4 greift nicht auf eine existierende Mapdatei zu, sondern generiert eine neue. Die erzeugte Mapdatei ähnelt der in Beispiel 15-1 verwendeten Mapdatei. Die Abschnitte sind so gestaltet, dass sie der Struktur der Mapdatei aus Beispiel 15-1 entsprechen. Auf diese Weise können Sie die Programmlogik leicht nachvollziehen.

Beispiel 15-4: Dynamisches Erzeugen einer neuen Mapdatei mithilfe von MapScript

```
# custommapfile.py
# Erzeugen einer Mapdatei unter Verwendung von Python MapScript

import mapsript

# Erzeugen der Mapdatei, Setzen verschiedener Eigenschaften
map = mapsript.mapObj()
map.name = "CustomMap"
map.setSize(600,300)
map.setExtent(-180.0,-90.0,180.0,90.0)
map.imagecolor.setRGB(180, 180, 250)
map.units = mapsript.MS_DD

# Setzen der Eigenschaften für das Web-Objekt
map.web.imagepath = "/srv/www/htdocs/tmp/"
map.web.imageurl = "/tmp/"

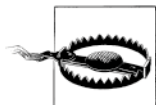
# Erzeugen eines layer-Objekts als Unterobjekt des map-Objekts
layer = mapsript.layerObj(map)
layer.name = "countries"
layer.type = mapsript.MS_LAYER_POLYGON
layer.status = mapsript.MS_DEFAULT
layer.data = "countries_simpl"
layer.template = "template.html"

# Erzeugen eines class-Objekts als Unterobjekt des layer-Objekts
class1 = mapsript.classObj(layer)
class1.name = "Countries"

# Erzeugen eines style-Objekts als Unterobjekt eines class-Objekts
style = mapsript.styleObj(class1)
style.outlinecolor.setRGB(100, 100, 100)
style.color.setRGB(200, 200, 200)

# Schreiben des map-Objekts in eine Mapdatei
map.save("custom.map")
```

Die mit diesem Skript erzeugte Mapdatei ist nicht mit Beispiel 15-1 identisch, da MapScript eine Reihe zusätzlicher Standardwerte definiert. Dennoch besitzt die Mapdatei die gleiche Funktionalität und erzeugt die gleiche Karte.



Die in diesem Beispiel verwendeten Objektnamen sind einfach, aber nicht unbedingt empfehlenswert, da sie große Ähnlichkeit mit den Namen der eigentlichen MapScript-Objekte haben. Würden Sie `class1` den Namen `class` geben, könnte es Schwierigkeiten geben, da `class` in vielen Programmiersprachen ein reserviertes Schlüsselwort ist.

Einsatz von PHP MapScript

PHP MapScript kommt sicherlich in vielen Anwendungen zum Einsatz. Daher sollen an dieser Stelle die Beispiele unter Verwendung von PHP MapScript aufgeführt werden. Dabei wird die gleiche Beispiel-Mapdatei verwendet. In einem Beispiel wird aufgeführt, wie über MapScript ein Layer erzeugt werden kann, der einen WMS als Datenquelle verwendet. Außerdem lernen Sie PHP MapScript als WMS-Server kennen.

Öffnen der Mapdatei und Zeichnen der Karte

Beispiel 15-5: Mithilfe von PHP MapScript eine Mapdatei laden und eine Karte in einer Datei ausgeben

```
<?php
// phpmascript_example_15-5.php

if (!extension_loaded("MapScript")) {
    dl('php_mapscript.' . PHP_SHLIB_SUFFIX );
}

// Gewuenschte Mapdatei wird gesetzt.
$mapfile= 'global.map';

// Map-Objekt wird mit der oben angegebenen Mapdatei initialisiert.
$mapobject = ms_newMapObj($mapfile) or die('MapFile kann nicht geladen werden.');
```

```
// Image-Objekt mit dem Kartenbild wird erstellt.
$mapimage = $mapobject-> draw();

// Speichert die Karte in eine Datei
$mapimage-> saveImage('weltkarte.png');
```

```
?>
```

Manipulation der Eigenschaften des Map-Objekts

Beispiel 15-6: Anlegen eines class-Objekts, Auswahl eines Landes und Darstellung des Landes in einer anderen Farbe (PHP MapScript)

```
<?php
// phpmascript_example_15-6.php

if (!extension_loaded("MapScript")) {
    dl('php_mapscript.' . PHP_SHLIB_SUFFIX );
}

// Gewuenschte Mapdatei wird gesetzt.
$mapfile= 'global.map';

// Map-Objekt wird mit der oben angegebenen Mapdatei initialisiert.
$mapobject = ms_newMapObj($mapfile) or die('MapFile kann nicht geladen werden.');
```

Beispiel 15-6: Anlegen eines class-Objekts, Auswahl eines Landes und Darstellung des Landes in einer anderen Farbe (PHP MapScript) (Fortsetzung)

```
// Erster Layer wird geholt und
// das Attributfeld fuer die Klassifikation ausgewaehlt.
$layer= $mapobject-> getLayer(0);
$layer-> set('classitem', 'NAME');

// Erste Klasse des Layers wird ermittelt und
// als Kopie dem Layer hinzugefuegt.
$classDeutschland = $layer-> getClass(0);
ms_newClassObj($layer, $classDeutschland);

// Vergabe eines aussagekraeftigen Namens
$classDeutschland-> set('name', 'Deutschland');

// Setzen der Expression zum Filtern
$classDeutschland-> setExpression( 'Germany' );

// Farbe und Umrissfarbe der Klasse werden gesetzt.
$deutschlandStyle = ms_newStyleObj( $classDeutschland );
$deutschlandStyle -> color -> setRGB ( 255, 255, 0 );
$deutschlandStyle -> outlinecolor -> setRGB ( 0, 0, 0 );

// Image-Objekt mit dem Kartenbild wird erstellt.
$mapimage = $mapobject-> draw();

// Speichert die Karte in eine Datei.
$mapimage-> saveImage('weltkarte.png');
?>
```

Dynamisches Erzeugen eines Map-Objekts, über das ein WMS im Layer-Objekt angesprochen wird

In diesem Beispiel wird ein Map-Objekt dynamisch erzeugt. Die Eigenschaften des Map-Objekts werden gesetzt, und die Ausgabeprojektion wird definiert. Es wird ein Layer-Objekt dynamisch erzeugt. Diesem Objekt wird ein WMS als Datenquelle zugewiesen, und es werden die notwendigen Eigenschaften für den Zugriff gesetzt. Abschließend wird ein Bild ausgegeben. Die Mapdatei wird nicht gespeichert, weil sie an dieser Stelle nicht benötigt wird.

Beispiel 15-7: Über PHP MapScript eine Mapdatei dynamisch erzeugen, die einen WMS im Layer anspricht und ein Bild ausgibt

```
<?php
// phpmascript_example_15-7.php
// Dynamisches Map-Objekt, Layer-Objekt nutzt WMS als Datenquelle.

if (!extension_loaded("MapScript")) {
    dl('php_mapscript.' . PHP_SHLIB_SUFFIX );
}
```

Beispiel 15-7: Über PHP MapScript eine Mapdatei dynamisch erzeugen, die einen WMS im Layer anspricht und ein Bild ausgibt (Fortsetzung)

```
// Map-Objekt wird erstellt und Eigenschaften werden gesetzt.
$mapobject = ms_newMapObj('');
$mapobject-> set('name', 'WMS und Projektionen');
$mapobject-> set('width', 300);
$mapobject-> set('height', 300);
$mapobject-> setExtent( 2554900 , 5511300 , 2628900 , 5585300 );
$mapobject-> set('units', MS_METERS);

// Ausgabeprojektion des Map-Objekts wird auf Gauß-Krüger 2 gesetzt.
$mapobject-> setProjection("init=epsg:31466");

// Eigenschaften des Web-Objekts werden gesetzt.
$mapobject-> web -> set ('imagepath','C:/ms4w/tmp/ms_tmp/');
$mapobject-> web -> set ('imageurl', '/ms_tmp/');

// WMS-Layer wird auf Basis des Map-Objekts eingebunden.
$layer = ms_newLayerObj($mapobject);
$layer-> set('name', 'gdi_rlp_relief');
$layer-> set('status', MS_ON);
$layer-> set('connectiontype', 7);
$layer-> set('connection', 'http://www.gdi-rp-dienste2.rlp.de/cgi-bin
                           /mapserv?map=/data/umn/rlp_relief.map');
// Metadaten fuer den WMS-Dienst werden gesetzt.
$layer-> setMetaData('wms_name', 'relief');
$layer-> setMetaData('wms_server_version', '1.1.1');
$layer-> setMetaData('wms_srs', 'EPSG:31466');
$layer-> setMetaData('wms_format', 'image/png');

// Projektion des Layers wird gesetzt.
$layer-> setProjection("init=epsg:31466");

$mapimage = $mapobject-> draw();
if (FALSE !== $mapimage ) {
    $url = $mapimage-> saveWebImage();
    $mapimage-> free();

    printf("<IMG SRC=%s>\n", $url);
}
?>
```

Probieren Sie dieses Skript direkt einmal aus. Legen Sie es in das DocumentRoot-Verzeichnis Ihres Webserver und rufen Sie es auf. Ein Aufruf könnte folgendermaßen aussehen:

`http://localhost/phpmapscript_example_15-7.php`

Sie erhalten das Bild, das in Abbildung 15-4 zu sehen ist, zurück.

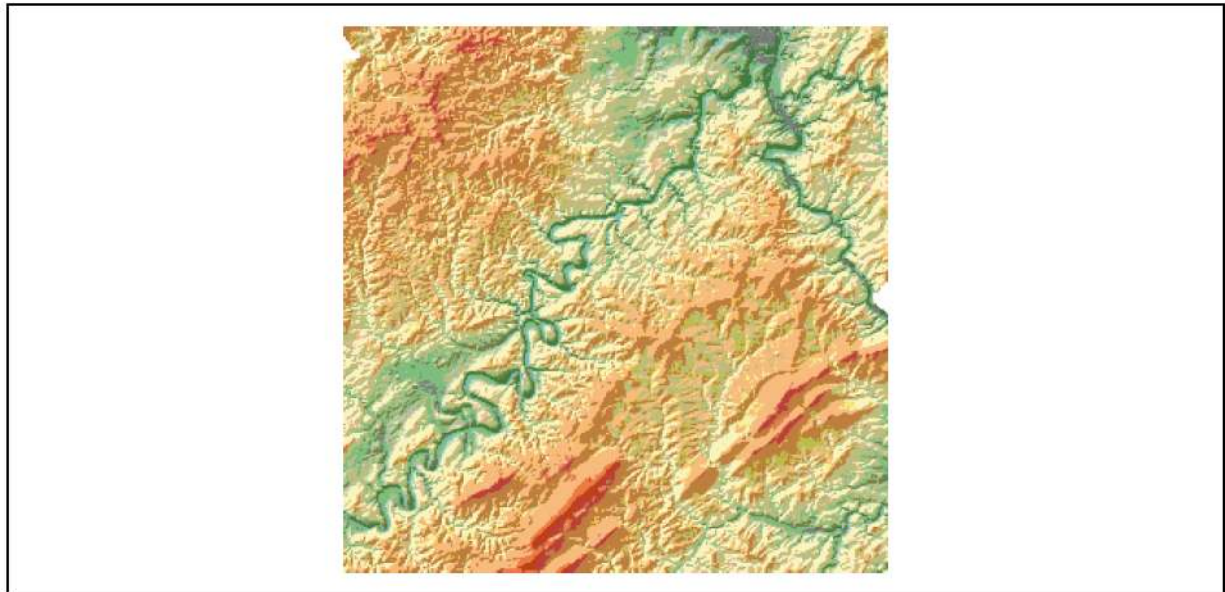


Abbildung 15-4: Die aus Beispiel 15-7 resultierende Karte

Nutzung von PHP MapScript als WMS-Server

Im letzten Beispiel soll gezeigt werden, wie PHP MapScript als WMS-Server eingesetzt werden kann.

In diesem Fall wird die Konfiguration aus einer Mapdatei gelesen. Hierfür wurde eine neue Mapdatei erstellt (`global_wms_server.map`) und um die notwendigen WMS-Parameter erweitert.

Beispiel 15-8: Mapdatei `global_wms_server.map`, in der ein WMS als Datenquelle genutzt wird

```
MAP
  NAME world
  EXTENT -180 -90 180 90
  SIZE 600 600
  IMAGECOLOR 180 180 250
  IMAGETYPE PNG

  UNITS DD
  WEB

  IMAGEURL "/tmp/"
  IMAGEPATH "/srv/www/htdocs/tmp/"
  METADATA
    "wms_title"      "WMS Demo Server"
    "wms_onlineresource" "http://localhost/phpmapscript_example_15-9.php?"
    "wms_srs"        "EPSG:31466 EPSG:31467 EPSG:4326"
  END
END
```

Beispiel 15-8: Mapdatei global_wms_server.map, in der ein WMS als Datenquelle genutzt wird (Fortsetzung)

```
PROJECTION
  "init=epsg:4326"
END

LAYER
  NAME "laender"
  STATUS ON
  TYPE POLYGON
  DATA countries_simpl
  METADATA
    "wms_title" "Länder der Erde"
  END
  PROJECTION
    "init=epsg:4326"
  END
  CLASS
    NAME "Länder"
    STYLE
      COLOR 200 200 200
      OUTLINECOLOR 100 100 100
    END
  END
END
END
```

Beachten Sie die Onlineresource in der Mapdatei `global_wms_server.map`. Hier wird auf das PHP verwiesen, das im folgenden Beispiel aufgeführt ist. Der WMS wird also nicht direkt über die Mapdatei angesprochen, sondern über das Skript, das die Einstellungen der Mapdatei lädt und über `owsdispatch` den Request auf das Map-Objekt anwendet. Das Ergebnis wird anschließend beispielsweise im Browser ausgegeben.

Dabei wird der Dienst wie gewohnt über den WMS-Request angesprochen. Der `GetCapabilities`-Aufruf könnte wie folgt aussehen:

```
http://localhost/phpmapscript_example_15-9.php?service=WMS&
Request=GetCapabilities&version=1.1.1
```

Ein `GetMap`-Aufruf sieht wie folgt aus:

```
http://localhost/phpmapscript_example_15-9.php?SERVICE=WMS&VERSION=
1.1.1&REQUEST=GetMap&LAYERS=laender&STYLES=&WIDTH=400&HEIGHT=200&BBOX=
-180,-90,180,90&SRS=EPSG:4326&FORMAT=image/png
```

Beispiel 15-9: PHP-Skript, das über PHP MapScript auf eine Mapdatei zugreift und als WMS-Server eingesetzt werden kann

```
<?php
// phpmapscript_example_15-9.php
// PHP MapScript als WMS-Server

if (!extension_loaded("MapScript")) {
    dl('php_mapscript.' . PHP_SHLIB_SUFFIX );
```

Beispiel 15-9: PHP-Skript, das über PHP MapScript auf eine Mapdatei zugreift und als WMS-Server eingesetzt werden kann

```
}

// Request-Objekt wird erstellt und Parameter werden geladen.
$request = ms_newowsrequestobj();
$request->loadparams();

// Buffer wird gestartet.
ms_iostreamstdoutbuffer();

// Gewuenschte Mapdatei wird gesetzt.
$mapfile= 'global_wms_server.map';

// Map-Objekt wird mit der oben angegebenen Mapdatei initialisiert.
$mapobject = ms_newMapObj($mapfile) or die('MapFile kann nicht geladen werden.');
```

```
// OWS-Request wird auf das MapObjekt angewendet.
$mapobject->owsdispatch($request);

// Content-Type wird ermittelt.
$contenttype = ms_iostreamstdoutbuffercontenttype();

// Content-Type wird geprueft. Bilder und XML-Rueckgaben werden unterschiedlich behandelt.
if( $contenttype == 'image/png' ){
    header('Content-type: ' . $contenttype);
    ms_iogetstdoutbufferbytes();
} else {
    // Buffer wird in Variable geschrieben.
    $buffer = ms_iogetstdoutbufferstring();
    header('Content-type: ' . $contenttype);
    echo $buffer;
}
ms_iostreamresethandlers();
?>
```

Verweise zu weiteren Beispielen

Hier finden Sie weitere Beispiele für Anwendungen und Open Source-Web-Mapping-Projekte, die MapScript verwenden:

- PHP MapScript HOWTO: <http://mapserver.gis.umn.edu/docs/howto>
- Programme, die PHP MapScript einsetzen: <http://maptools.org/>
- ka-Map :<http://ka-map.maptools.org>
- p.mapper: <http://www.pmapper.net/>. Hier finden Sie viele Demoanwendungen.
- Chameloen: <http://chameleon.maptools.org/>
- CartoWeb: <http://www.cartoweb.org/>

- Mobile Geographics: <http://www.mobilegeographics.com/mapserver/>. Einfache Beispiele für die Verwendung von PHP MapScript.
- Python Cartographic Library (PCL): <http://zcologia.org/>. Mapping-Werkzeuge für Python, die nicht unbedingt MapServer, aber ähnliche Objekte verwenden.

MapScript-Beispiele für verschiedene Programmiersprachen

In diesem Abschnitt erhalten Sie einfache Beispiele für den Einsatz von MapScript in verschiedenen Programmiersprachen. Die Beispiele 15-10 bis 15-12 sollen als Referenz für Programmierer dienen, die die Verwendung von MapScript in anderen Sprachen nutzen möchten. Die Beispiele führen die gleichen Schritte durch: Öffnen einer Mapdatei, Zeichnen der Karte und Speichern der Karte in einer Datei. Alle Beispiele sind so angelegt, dass sie in dem Verzeichnis ausgeführt werden, in dem sich *global.map* befindet. Außerdem erzeugen alle Beispiele ein Bild namens *worldmap.png*.

Beispiel 15-10: Einfaches Beispiel für die Verwendung von MapScript mit Perl

```
# map1.pl
# Perl MapScript Example 1

use mapscript;

# Set the map file to use
$mapfile = "global.map";

# Create a mapObj, initialized with the mapfile above
$mapobject = new mapscript::mapObj($mapfile);

# Create an imgObj that has an image of the map
$mapimage = $mapobject->draw();

# Save the mapimage to a file
$mapimage->save("worldmap.png");
```

Beispiel 15-11: Einfaches Beispiel für die Verwendung von MapScript mit Java

```
import edu.umn.gis.mapscript.*;

public class map1 {
    public static void main(String[] args) {
        System.loadLibrary("mapscript");

        mapObj mapobject;
        imageObj mapimage;
        String mapfile;

        // Set the map file to use
        mapfile = "global.map";

        // Create a mapObj, initilized with the mapfile above
        mapobject = new mapObj(mapfile);
```


Beispiel 15-11: Einfaches Beispiel für die Verwendung von MapScript mit Java (Fortsetzung)

```
// Create an imgObj that has an image of the map
mapimage=mapobject.draw();

// Save the mapimage to a file
mapimage.save("worldmap.png",mapobject);
}
}
```

Beispiel 15-12: Einfaches Beispiel für die Verwendung von MapScript mit Ruby

```
# map1.rb
# Ruby MapScript Example 1

require "mapscript"
include Mapscript

# Set the map file to use
mapfile = "global.map"

# Create a mapObj, initialized with the mapfile above
mapobject = MapObj.new(mapfile)

# Create an imgObj that has an image of the map
mapimage = mapobject.draw

# Save the mapimage to a file
mapimage.save("worldmap.png")
```

Eine kurze Einführung in die Kartenprojektionen

Kartenprojektionen gehören zu den kritischen Komponenten jeder kartografischen Anwendung, seien es Kartenausdrucke oder auch webbasierte, interaktive Karten. In beiden Fällen besteht die Aufgabe darin, geometrische Objekte den Nutzern bzw. Anwendern auf eine praktikable Art zu Verfügung zu stellen. Die grafische Darstellung soll eine optimale Interpretierbarkeit gewährleisten.

Gerade Einsteigern, die vorher noch nie etwas mit Kartografie zu tun hatten, ist dieses Thema zunächst etwas schwer zugänglich. Die große Vielfalt an unterschiedlichen Koordinaten- und Darstellungssystemen trägt hierzu entscheidend bei.

Um interaktive Web-Mapping-Anwendungen aufsetzen zu können, bedarf es einer gewissen Auseinandersetzung mit der Problematik der Koordinatensysteme und den gebräuchlichsten Abbildungen von georeferenzierten Objekten in die gewünschte Darstellungsform (Karte, Bild).

Die in den folgenden Abschnitten zu findende Einleitung soll den Leser auf die Verwendung von unterschiedlichen Projektionsparametern mit MapServer vorbereiten. Es geht in der Einführung zunächst um allgemeine Konzepte zu Kartenprojektionen und Koordinatentransformationen. Bei der Erläuterung der Grundlagen werden einige der gebräuchlichsten Modelle vorgestellt. Des Weiteren wird erläutert, welcher Syntax die MapServer-Konfigurationsdateien folgen und wie mit einfachen Mitteln Präsentationen in den unterschiedlichsten Abbildungssystemen erstellt werden können

Die Gestalt der Erde

Schon seit langer Zeit macht der Mensch sich über die Form der Erde Gedanken. Er war von Beginn an darauf angewiesen, sich auf ihr orientieren zu können. Als Jäger und Sammler reichte es ihm aus, sich auf seinen Instinkt zu verlassen, um von A nach B zu gelangen. Erst mit dem Sesshaftwerden tauchten neue Bedürfnisse auf. Er war plötzlich darauf angewiesen, auch nach weiten Wanderungen wieder zurück zu seinem Ausgangspunkt zu gelangen. Weiterhin gab es mit der landwirtschaftlichen Nutzung der Erde auch das Bedürfnis der Sicherung der Anbauflächen gegen Ansprüche Fremder.

Die Entstehung von (Stadt-)Staaten machte es zusätzlich notwendig, Herrschaftsbereiche zu dokumentieren.

All diese Entwicklungen führten dazu, dass der Mensch anfang, seine Umgebung auf Basis von geometrischen Darstellungen zu erfassen. Die frühen Arten von Karten können grob in zwei Kategorien aufgeteilt werden:

1. Karten zu Zwecken der Navigation, hauptsächlich für militärische und wirtschaftliche Zwecke (Planung von Kriegszügen sowie Durchführung von Handelsexpeditionen).
2. Karten zu Zwecken der Dokumentation sowohl des Grundbesitzes als auch der jeweiligen Herrschaftsbereiche.

Bei der Erstellung der Karten wurde versucht, sich ein Modell der Wirklichkeit zu schaffen. In den frühen bekannten Darstellungen lag meist die Vorstellung einer ebenen Fläche zugrunde. Die Darstellung selbst war oft nur schematisch, und die Positionen von realen Objekten ließen sich daher nicht eindeutig aus den Positionen der Kartensymbole ableiten. In Abbildung A-1 sehen Sie den Turiner Papyrus, eine altägyptische Landkarte, die als bedeutendste erhaltene topographische Karte aus der Zeit um 1160 v.Chr. gilt. Die Karte wurde von dem bekannten Schreiber Amennakhte erstellt und diente einer Forschungsreise Ramses IV. zum Wadi Hammamat.

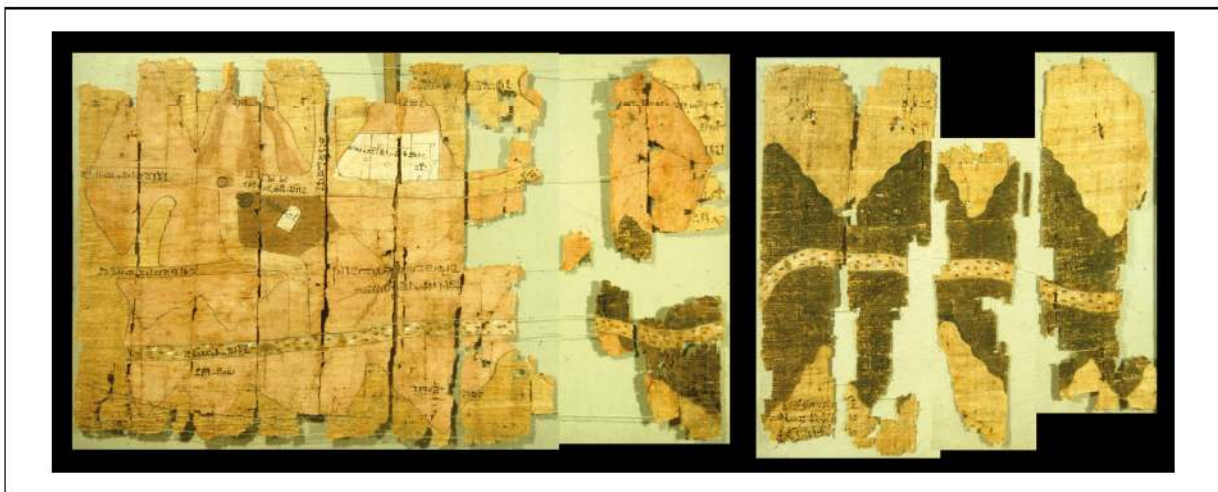


Abbildung A-1: Eine altägyptische Landkarte (Turiner Papyrus aus der Zeit um 1160 v.Chr.)

Eine Wende dieser Sicht gab es, nach bisherigen Erkenntnissen, erst ca. 500 v.Chr. Pythagoras (ca. 570–510 v.Chr.) soll einer der Ersten gewesen sein, der die Gestalt der Erde als Kugel postulierte. Eine erste Abschätzung der Erdgröße wird Eudoxus von Knidos (ca. 408–355 v.Chr.) zugeschrieben. Er soll auch die ersten Vorschläge für Positionsbestimmungen auf der Erdoberfläche aufgestellt haben.

Den Gelehrten zu dieser Zeit war die Problematik also geläufig. Es wurde damals schon nach Möglichkeiten gesucht, eindeutige Positionen festlegen zu können, wobei die Ansätze zur Bestimmung der Erdfigur sowie ihrer Abbildung in einer Karte eher philosophisch und mathematisch begründet waren. Die Annahmen, die zum Postulat des

Kugelmodells führten, basierten auf Beobachtungen der Umwelt. Beispiele hierfür sind beispielsweise der kreisförmige Erdschatten bei einer Mondfinsternis sowie die Tatsache, dass bei der Annäherung von Segelschiffen ans Festland zuerst die Spitzen der Masten zu erkennen sind.

Neben der abstrakten Figur der Erde stand auch ihre Größe immer wieder zur Disposition. Die erste nachweisbare Abschätzung des Radius der Erdkugel verdanken wir Eratosthenes (276–195 v.Chr.), der auch der Leiter der bekannten Bibliothek von Alexandria gewesen sein soll. Er beobachtete den Winkel des Sonneneinfalls zur Mittagszeit an zwei verschiedenen Orten, die ungefähr auf demselben Längengrad lagen. Aus der ihm grob bekannten Entfernung der beiden Orte leitete er dann den Radius der Erde ab. Da die der Messung zugrunde liegende Längeneinheit (1 Stadion = 600 Fuß) regional unterschiedlich war, kann das absolute Ergebnis nicht genau qualifiziert werden. Werden Minimal- und Maximalwerte für die Einheit Fuß zugrunde gelegt, ergibt sich eine Spanne zwischen 6.000 und 8.000 km. Ungeachtet dessen ist die erstmalige Erkenntnis der Erdgröße so bedeutend, dass Eratosthenes auch als Urvater der Erdmessung bezeichnet wird.

Die Kugelform der Erde war bis zum 17. Jahrhundert allgemein anerkannt, auch wenn viele Quellen den europäischen Wissenschaftlern des Mittelalters vorwerfen, sie hätten die Erkenntnisse der griechischen Antike zugunsten eines Scheibenmodells verworfen. Selbst altnordische Schriften des 11. Jahrhunderts beschreiben die Erde als Kugel.

Mit der Weiterentwicklung der messtechnischen Verfahren, vor allem der Streckenmessung im Zeitalter der Renaissance, gab es erstmals die Möglichkeit, die Figur der Erde noch genauer zu erfassen. Bei diesen Messungen stellte sich heraus, dass je nach geographischer Breite die Länge von Meridianbogenabschnitten variierten. Dies wird in Abbildung A-2 verdeutlicht.

Isaac Newton vermutete schon, dass die Erde an den Polen abgeflacht sei. Er argumentierte insbesondere mit der beobachteten Rotation der Erde um ihre Achse. Da der Erdkörper elastisch sei, müssten die Zentrifugalkräfte zu einer Auswölbung am Äquator führen. Der praktische Beweis für die Abplattung an den Polen konnte aber erst Mitte des 18. Jahrhunderts erbracht werden. Seit dieser Zeit wurde die Form der Erde näherungsweise als sogenannter Rotationsellipsoid angesehen. Es handelt sich hierbei um einen zu einer Achse (in diesem Fall der Drehachse bzw. Erdachse) symmetrischen Rotationskörper, der an den Polen etwas abgeflacht ist.

Eindeutige Festlegung von Objektpositionen

Da eine dreidimensionale Betrachtung von räumlichen Objekten den Rahmen dieser kurzen Vorstellung sprengen würde, beschränkt sich der folgende Teil auf die Behandlung von zwei Dimensionen (der Lage von Objekten auf der Erdoberfläche). Bei der Abbildung von Teilen der Erdoberfläche auf einer Karte spielt die Höhe eher eine untergeordnete Rolle.

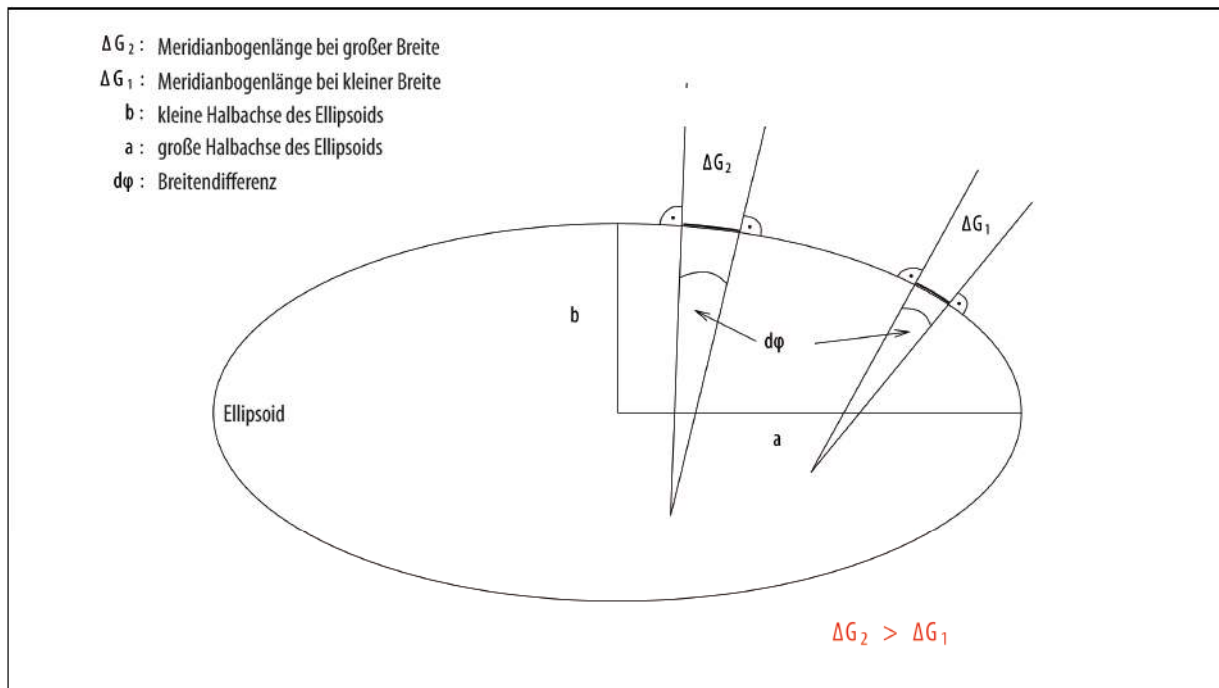


Abbildung A-2: Breitenabhängigkeit der Meridianbogenabschnitte

Für die Beschreibung der Lage (2-D) eines Objekts auf der Erdoberfläche gibt es verschiedene Möglichkeiten, die sich nach der Größe des zu untersuchenden bzw. abzubildenden Gebiets gliedern lassen. Die einfachste Lösung ist die Verwendung eines »ebenen« rechtwinkligen Koordinatensystems. Für kleinräumige Gebiete mit Ausdehnungen von wenigen Kilometern ist eine solche Vereinfachung sicher sinnvoll. Sie können sich leicht vorstellen, dass es kaum einen Unterschied macht, ob Sie die zu beschreibenden Objekte auf eine Ebene beziehen oder versuchen, ihre genaue Position auf einem Rotationsellipsoid festzulegen. Wird jedoch das kartografisch darzustellende Gebiet größer, wird als Bezugsfläche eine Kugel oder, bei gehobeneren Genauigkeitsansprüchen, ein Ellipsoid gewählt.

Geografische Koordinatensysteme

Wie wird ein bestimmter Punkt auf der Erde beschrieben? Früher wurden Positionen mithilfe astronomischer Beobachtungen fixiert. Spätestens seit der Erkenntnis, dass die Erde eine kugelförmige Gestalt aufweist, geschah dies durch Winkelmessungen zu Fixsternen bzw. der Sonne und über die genaue Bestimmung von Zeitdifferenzen. Unabhängig davon, wie aufwendig diese Variante der Ortsbestimmung im Zeitalter von GPS anmutet: Sie führte zu eindeutigen Positionen auf der Erdoberfläche und war bis zum Ende des 80er-Jahre des 20. Jahrhunderts beispielsweise auf hoher See die einzige Möglichkeit, den Standort herauszufinden.

Die über solche Verfahren erhaltenen Positionen werden in Winkelwerten angegeben und üblicherweise als geografische Koordinaten bezeichnet. Sie beziehen sich in erster Näherung auf die schon angesprochene Kugel als Bezugsfläche. Dies wird in Abbildung

A-3 verdeutlicht. Die geografische Breite ist, für den Fall der Kugel als Erdmodell, der Höhenwinkel, unter dem der Punkt P von der Äquatorebene aus erscheint. Sie wird von 0 ausgehend bis +90 Grad nach Norden und bis –90 Grad nach Süden gezählt. Die geografische Länge ist der Winkel zwischen den senkrecht zur Äquatorebene stehenden Ebenen durch P und M sowie durch Greenwich und M.

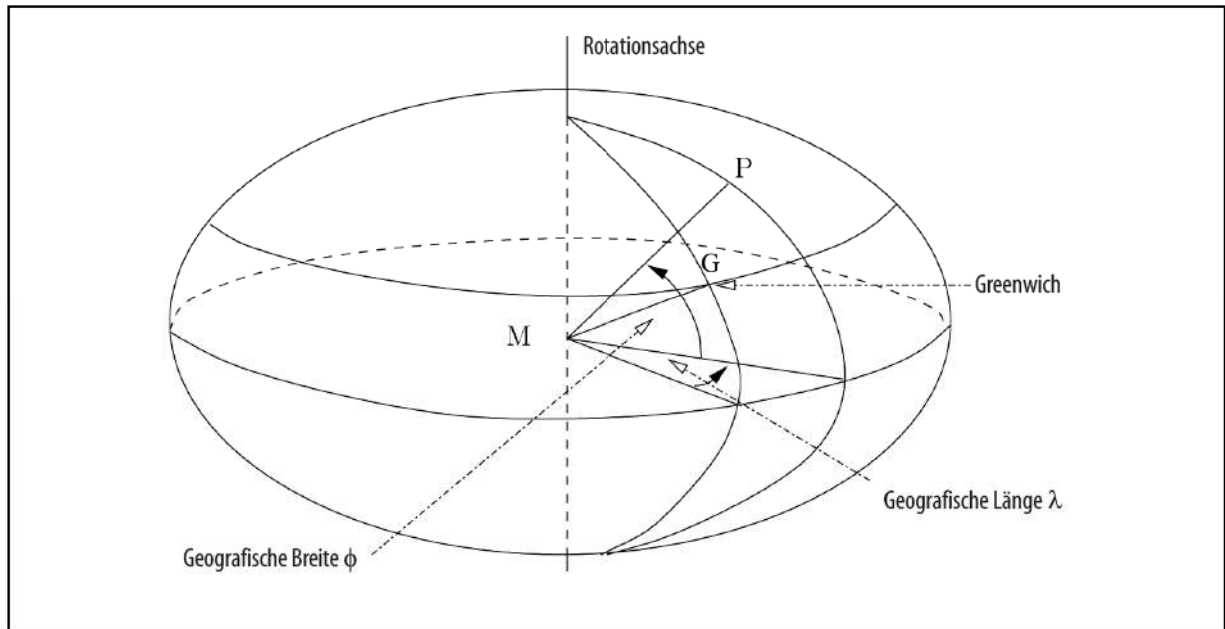


Abbildung A-3: Definition des geografischen Koordinatensystems

Interessanterweise wurden die geografischen Koordinaten schon durch Hipparchos von Nicäa (Hipparch, ca. 190–120 v.Chr.) definiert. Die Zählung der geografischen Länge begann aber nicht beim Meridian von Greenwich, sondern auf der Insel El Hierro (Kanaren). Diese Festlegung hatte fast zwei Jahrtausende Bestand, bis man sich 1884 auf die Mittelachse eines bestimmten Teleskops im Observatorium von Greenwich einigte. Erst seit dieser Zeit gilt der Längengrad, der durch Greenwich verläuft, als Nullmeridian.

Bestimmung der geografischen Breite

Die geografische Breite lässt sich entweder über die Messung des Winkels vom Horizont zum Polarstern (Nordhalbkugel – nachts) ermitteln oder über den Winkel zwischen Horizont und Sonne zum Zeitpunkt des höchsten Sonnenstands. Im ersten Fall werden für eine Positionsgenauigkeit von 60 Meilen keine Korrekturen benötigt, im zweiten Fall sind zusätzlich Korrekturdaten aus einem Almanach (bzw. einem nautischen Jahrbuch) notwendig.

Bestimmung der geografischen Länge

Die Bestimmung der geografischen Länge ist um einiges schwieriger als die der geografischen Breite. Es wird ein Differenzwinkel zum Meridian von Greenwich benötigt. Die Bestimmung der geografischen Länge bedingt dabei eine sehr genaue Bestimmung der Zeit. Die Ortszeit kann über die Beobachtung des Sonnenstands ermittelt werden. Wird diese mit der Ortszeit von Greenwich verglichen, kann die geografische Länge sehr einfach ermittelt werden. Da die Erde sich innerhalb von 24 Stunden um ihre Achse dreht, entspricht ein Zeitunterschied von einer Stunde einem Längenunterschied von 15 Grad. Eine andere Möglichkeit ist die Ableitung einer sehr genauen Ortszeit aus Winkelbeobachtungen zu Planeten und Gestirnen. Auf See gestaltet sich das aber äußerst schwer, deshalb wurde dort der genauen Zeitmessung der Vorrang gegeben.

Die geografischen Koordinaten bieten aufgrund ihrer Eigenschaften eine einfache Möglichkeit, alle Punkte der Erdoberfläche eindeutig zu beschreiben. Sie gehören zu den verbreitetsten Koordinatensystemen und werden in dieser Form schon seit über 2000 Jahren verwendet. Werden genauere Punktfestlegungen benötigt, stößt man mit ihnen jedoch sehr schnell an Grenzen. Sie eignen sich außerdem nicht dazu, um auf ihnen zu navigieren oder um Messungen durchzuführen. Für solche Zwecke werden sogenannte Gebrauchskoordinatensysteme verwendet. Schon Hipparch und im Folgenden Ptolemäus (ca. 100–175 n.Chr.) schufen aus diesem Grund Rechenvorschriften zur Abbildung bzw. Projektion der geografischen Koordinaten einer Ebene (Kartenbild).

Verebnete Koordinatensysteme/Projektionen

Wie schon zuvor erwähnt, soll die grafische Darstellung von georeferenzierten Objekten in einer Karte die optimale Interpretierbarkeit gewährleisten. Je nach Anwendungsfall werden bestimmte Anforderungen an eine kartografische Darstellung gestellt. Sie können sich vorstellen, dass bei der Projektion der Kugelfläche in eine Ebene die gekrümmte Fläche verzerrt werden muss. Es gibt keine Möglichkeit, die Kugelfläche unverzerrt in die Ebene zu überführen. Es gibt eine Vielzahl von Projektionen, die jeweils für einen bestimmten Zweck definiert wurden.

Häufig wird die folgende Klassifizierung nach den zu erhaltenden geometrischen Eigenschaften vorgenommen:

Längentreue (äquidistante) Abbildung

Soll eine Karte genutzt werden, um Strecken abzugreifen, muss die Abbildung der auf der Bezugsfläche gemessenen Strecken in der Karte längentreu sein. Mithilfe eines (einheitlichen) Maßstabs soll zurück auf die Wirklichkeit geschlossen werden können.

Winkeltreue (konforme) Abbildung

Gerade bei der Navigation kommt es darauf an, die in der Natur gemessenen Winkel (Richtungswinkel, Kurswinkel) in der Karte wiederzufinden. Hier wird die

Karte sowohl zur Bestimmung von Positionen genutzt (grafische Positionsbestimmung durch Übertragen von real gemessenen Winkeln/Richtungen in die Karte) als auch zur Festlegung und Planung von Kursen.

Flächentreue (äquivalente) Abbildung

Flächen werden ihrem Maßstab entsprechend richtig abgebildet. Karten, die nach einem solchen Verfahren erstellt wurden, eignen sich besonders zu Übersichtszwecken.

Andere Projektionen wiederum vermitteln die Abbildungsverzerrungen. Es ist nicht sinnvoll, an dieser Stelle auf alle möglichen Projektionen einzugehen. Es gibt Hunderte solcher Verfahren, die gekrümmte Oberfläche der Erde mehr oder weniger verzerrt in eine ebene Form zu überführen.

Bei der folgenden Auflistung verschiedener Projektionen wird davon ausgegangen, dass man sich das Verfahren der Projektion bildlich vorstellt. Im Grunde entstehen die Projektionen durch mehr oder weniger komplexe Rechenverfahren, die eine mathematische Abbildung der gekrümmten Fläche in eine Ebene vollführen. Es gibt dabei unterschiedliche Interpretationsansätze, die jedoch den Rahmen dieser Einführung sprengen würden. Einen intuitiven Zugang erhalten Sie, indem Sie sich die Projektion auch als eine solche vorstellen, beispielsweise wie das Prinzip des Diaprojektors. Dieser durchleuchtet ein transparentes Objekt und stellt dessen Bild auf einer Leinwand dar. Im Fall der Projektionen der Erdfigur müssen Sie sich die reale Figur der Erde als transparente Glaskugel (-Ellipsoid) vorstellen. Im Innern dieser Kugel befindet sich eine Lichtquelle, die die auf der Oberfläche vorhandenen Objekte auf einen irgendwie geformten Schirm projiziert. Die Projektionen lassen sich dann nach Form und Position der Lichtquelle sowie der Form und Position des Schirms klassifizieren. Wichtig ist dabei die Tatsache, dass der Schirm immer aus einer abwickelbaren Fläche (Zylinder, Kegel, Ebene) besteht.

Im Folgenden seien nun die wichtigsten Projektionsarten kurz beschrieben (siehe auch Abbildung A-4):

Zylinderprojektionen (Zylinderentwürfe)

Hierbei wird die Projektionsfläche wie ein Zylinder um den Erdball gelegt. Zylinderprojektionen können den Globus komplett umgeben oder auch schneiden. Bei dieser Projektion liegt die Lichtquelle meist im Zentrum der Kugel. Der obere und der untere Bereich der Karte werden verzerrt dargestellt.

Konische Projektionen (Kegelentwürfe)

Diese Projektionen sehen aus wie eine Art Kegel, die der Erdkugel aufgesetzt wurde. Wenn Sie den Kegel der Erdkugel am Nordpol aufsetzen, verläuft der untere Rand des Kegels entlang eines Breitengrads. Die Spitze des Kegels bildet dabei die Details des Nordpols ab. Die Darstellung ist entlang des gemeinsamen Breitengrads am genauesten.

Orthografische Projektionen

Diese Entwürfe sehen aus, als hätte sie ein Astronaut vom Weltraum aus gezeichnet. Bei dieser Projektion ist der Globus nicht vom Mittelpunkt beleuchtet, sondern die Lichtquelle steht hinter der Erdkugel. Das Blatt Papier wird direkt auf die Oberfläche gelegt. Es wird von parallelen Lichtstrahlen getroffen. Die orthografische Projektion ist nicht für Weltkarten geeignet, da immer nur ein bestimmter Teil der Erde angezeigt werden kann. Wenn Sie die geografischen Gegebenheiten auf der Rückseite ignorieren, sieht eine solche Karte wie ein Faksimile der Erde aus.

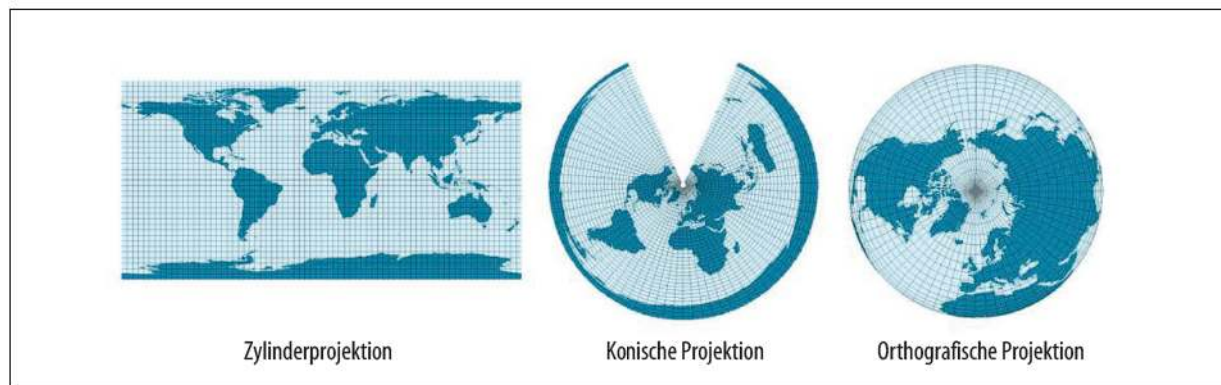


Abbildung A-4: Drei Projektionsarten (Abbildungen von Stefan Kühn)

Mit steigender Leistungsfähigkeit der Computer ist eine kugelförmige Abbildung zwar möglich, allerdings sind diese realistisch anmutenden Darstellungen nicht immer hilfreich. Für die Projektion räumlicher Daten spielen mathematische Berechnungen, ohne die die Erstellung präziser Darstellungen gar nicht möglich wäre, eine wichtige Rolle. Glücklicherweise müssen Sie die Berechnungen und Hintergründe nicht kennen, um Projektionen nutzen zu können. Sie müssen eigentlich nur wissen, dass die Glaskugel-analogie durch mathematische Berechnungen ersetzt wird. Jede Kartenprojektion hat ihre Stärken und Schwächen. Je nach Anwendung ist eine bestimmte Projektion besser geeignet als eine andere. So erhalten Sie bei zylindrischen Projektionen eine verständliche Karte der Erde, allerdings werden die Pole extrem verzerrt dargestellt. Auf der anderen Seite empfiehlt sich die konische Projektion für die Darstellung der Pole, während Weltkarten nicht abgeleitet werden können. Insgesamt gibt es mehrere Hundert verschiedener Projektionsmodelle, die sich allerdings in einige wenige Hauptgruppen (zylindrisch, konisch usw.) unterteilen lassen. Die für eine Implementierung notwendigen Details können sehr unterschiedlich sein.

Netze aus fest vermarkten Punkten und Datumsdefinitionen

In den bisherigen Abschnitten wurden die geografischen Koordinaten sowie deren Abbildung in die Ebene behandelt. Leider ist es in den meisten Fällen jedoch nicht damit getan, die Transformationen zwischen geografischen Koordinaten und ihren Abbildungen durchzuführen. Die meisten existierenden Geodaten wurden nicht dadurch erfasst, dass ihre geografischen Koordinaten bestimmt wurden. Dies wäre zu

aufwendig und, wie wir später noch sehen werden, zu ungenau gewesen. Im vorletzten Jahrhundert wurde in den meisten Ländern damit begonnen, sogenannte Festpunkt-netze aufzubauen. Diese Netze wurden angelegt, um als Anschlusspunkte für weitere Verdichtungsmessungen zu dienen. Man wollte die gesamte Landesfläche genauestens erfassen. Dabei kam es weniger auf den militärischen Bedarf an (auch wenn das Militär oft eine tragende Rolle spielte), vielmehr waren öffentliche und private Interessen maßgeblich. Für eine topografische Aufnahme, die dem Militär zu Planungszwecken dienen sollte, waren diese Messungen zu genau. Es sollte ein einheitliches Bezugssystem geschaffen werden, das sowohl den Anforderungen der Katasterbehörden zum Zwecke der Besteuerung genüge, als auch dafür geeignet war, als Grundlage für ingenieurtechnische Planungen zu dienen. Prinzipiell wurde – meist flächendeckend für ein Land – ein hochgenaues Lagenetz von vermarkten Punkten aufgebaut. Der Abstand der Hauptpunkte betrug dabei ca. 30 Kilometer. Das Netz selbst wurde über Winkel- und Streckenmessungen bestimmt. Die Messwerte wurden um die Erdkrümmung korrigiert (hier kamen entweder Kugelmodelle oder auch Ellipsoidmodelle mit unterschiedlichsten Parametern zum Einsatz), und das Netz wurde einer mathematischen Ausgleichung unterzogen.

Als Ergebnis lag danach ein – für die damaligen Verhältnisse – hochgenaues Bezugssystem aus vermarkten Punkten vor. In diesem System konnte eine sogenannte Nachbarschaftsgenauigkeit von 1cm auf 1km gewährleistet werden.

Die Koordinaten der Festpunkte lagen nach der Ausgleichung als geografische (ellipsoidische) Koordinaten vor. Sie bezogen sich also entweder auf ein Kugelmodell oder ein Ellipsoidmodell.

Bei diesem Verfahren wurden die direkte Bestimmung der geografischen Koordinaten oft in nur einem einzigen Punkt des Netzes durchgeführt. Dieser Punkt definiert dann die Orientierung sowie die Position des gesamten Festpunktfelds (der sogenannte Datumspunkt). Es wurde an dieser Stelle festgestellt, dass sich direkt gemessene geografische Koordinaten von denen unterscheiden, die über die Ausgleichung erhalten wurden. Das geschaffene Netz war nur in sich selbst genau. Beim Anlegen der Netze wurde damals natürlich nicht daran gedacht, eine globale Lösung zu schaffen. Es wurde immer nur bis zur Grenze des eigenen Landes gedacht.

Das geodätische Datum und der Begriff des Bezugssystems

Mit diesem kleinen Exkurs wären wir beim geodätischen Datum und den damit verbundenen Problemen angelangt. Das geodätische Datum wird bestimmt durch die Koordinaten des Netzes aus Festpunkten. Prinzipiell definieren die absolut gemessenen geografischen Koordinaten im Netz sowie die Wahl der Ellipsoidparameter das geodätische Datum. Abbildung A-5 veranschaulicht die Unterschiede zwischen den verschiedenen Datumsdefinitionen relativ gut. Jedem der verschiedenen Systeme liegt eine eigene Bezugsfläche zugrunde.

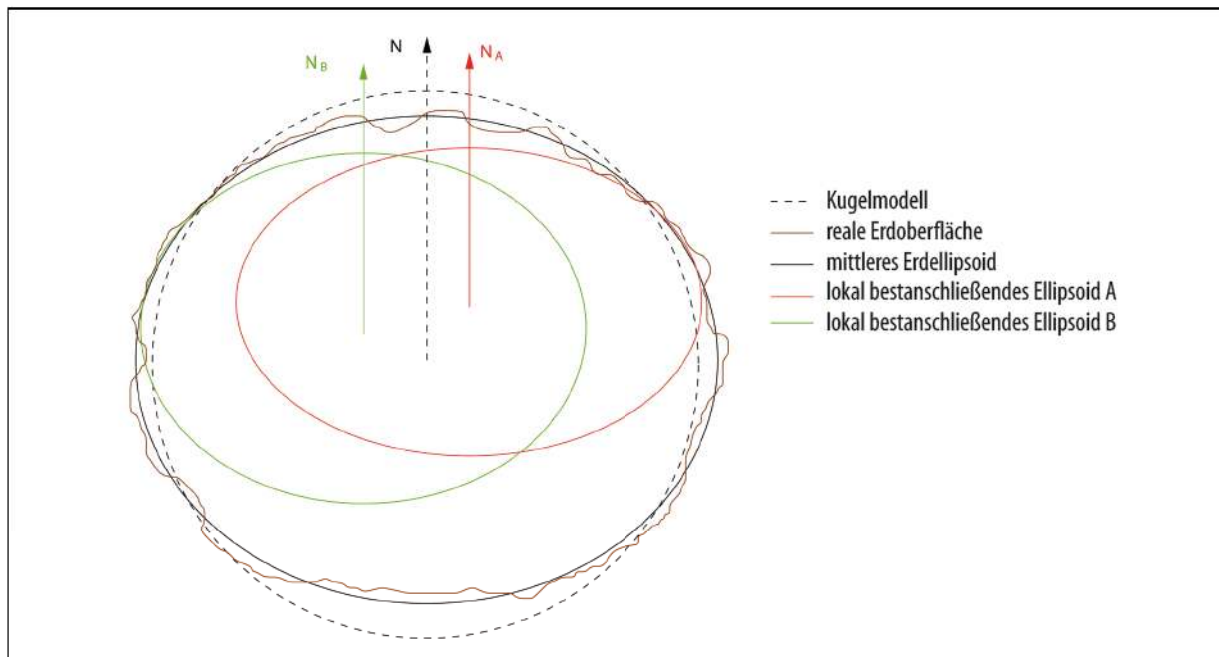


Abbildung A-5: Position unterschiedlicher Bezugsellipsoide

In der Praxis müssen Sie sich über diese Problematik im Klaren sein. Wollen Sie beispielsweise Daten mit Koordinaten im deutschen Hauptdreiecksnetz (zentrales deutsches Festpunktnetz) mit Daten aus Frankreich verknüpfen oder auch nur in gemeinsamen Karten oder Websites präsentieren, haben Sie es hier schon mit zwei unterschiedlichen geodätischen Datumsdefinitionen zu tun. Zusätzlich zu dieser Tatsache sind auch die verwendeten Projektionen unterschiedlich: In Frankreich wurden die Koordinaten über eine Lambert-Projektion verebnet, in Deutschland geschah dies über eine Gauß-Krüger-Projektion.

Zusammenfassend muss an dieser Stelle der Begriff des Koordinatenreferenzsystems (Coordinate Reference System) erklärt werden. Wie oben erwähnt, beziehen sich Koordinaten von Objekten auf der Erdoberfläche immer auf ein bestimmtes Datum. Das Datum und die verwendeten Koordinaten bestimmen das sogenannte Koordinatenreferenzsystem (engl. CRS). Neben dem Begriff des Koordinatenreferenzsystems werden im Zusammenhang mit Koordinatentransformationen auch die Begriffe Bezugssystem und Referenzsystem verwendet. Diese beiden Begriffe bezeichnen jedoch mehr als nur ein geodätisches Datum. Das Bezugssystem beinhaltet alle physikalischen und geometrischen Parameter, die notwendig sind, um Positionen von Objekten eindeutig zu beschreiben. Im Gegensatz zum geodätischen Datum werden nicht nur geometrische Zusammenhänge beschrieben, vielmehr gehören zu den Parametern eines Bezugssystems auch Vereinbarungen über zu verwendende physikalische Konstanten wie beispielsweise die Gravitationskonstante oder die Rotationsgeschwindigkeit der Erde. Prinzipiell werden durch die Definition eines solchen Bezugssystems die Freiheitsgrade bei der Bestimmung von relativen Positionen einzelner Objekte minimiert.

Transformationen zwischen unterschiedlichen Koordinatenreferenzsystemen

Sollen Koordinaten von einem CRS in ein anderes überführt werden, müssen die Transformationsparameter zwischen Ausgangsdatum und Zieldatum bekannt sein. Eine Koordinatentransformation erhalten Sie, indem Sie diese Parameter auf das Ausgangssystem anwenden. Der Übergang von einem Bezugsellipsoid zum anderen ist nicht besonders intuitiv. Einfacher ist es, sich die Punkte in einem räumlichen kartesischen Koordinatensystem vorzustellen. Abbildung A-6 zeigt hierzu eine schematische Darstellung. Hier wird ein Festpunkt in zwei unterschiedlichen kartesischen Koordinatensystemen beschrieben. Er hat folglich jeweils zwei Koordinatentripel $[X\ Y\ Z]$. Um nun das eine System in das andere zu überführen, sind sechs verschiedene Operationen notwendig: je drei Verschiebungen entlang der jeweiligen Achse (danach erhalten Sie identische Ursprungspunkte) sowie drei Drehungen um jeweils eine Achse. Falls sich die Systeme zusätzlich noch im Maßstab unterscheiden, kommt zu den sechs Parametern noch ein siebter hinzu. Es ist festzuhalten, dass der Übergang von einem geodätischen Datum in ein anderes über eine sogenannte 7-Parameter-Transformation möglich ist.

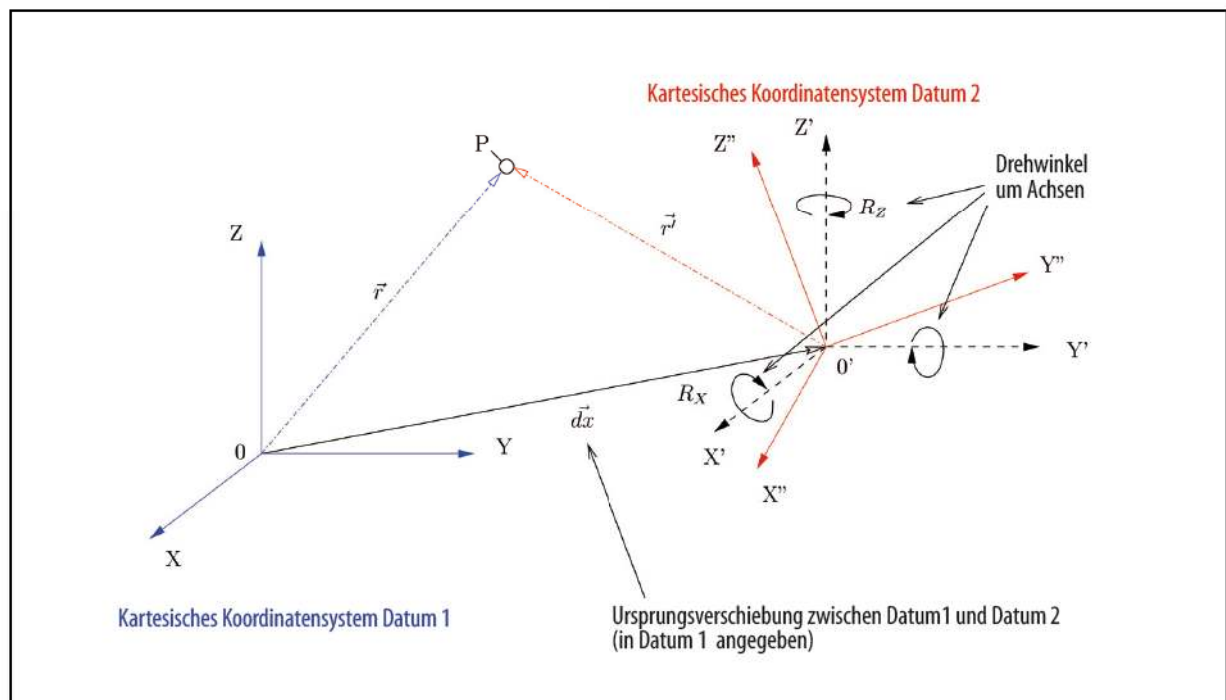


Abbildung A-6: Punktdefinition in verschiedenen kartesischen Koordinatensystemen

Die Parameter für den Übergang von einem in ein anderes System erhalten Sie grundsätzlich durch eine Ausgleichung. Für eine gewisse Zahl von identischen Punkten müssen Koordinaten sowohl in dem einen als auch in dem anderen System vorliegen. Je mehr Punkte vorhanden sind, desto genauer ist die Schätzung der Transformationsparameter. Da aber die Netze meist größere Spannungen aufweisen, gibt es eine Obergrenze für die erreichbare globale Transformationsgenauigkeit. In kleinen lokalen

Gebieten kann jedoch eine sehr gute Genauigkeit erreicht werden. Dies liegt darin begründet, dass die Nachbarschaftsgenauigkeit (die Genauigkeit zwischen nahe zusammenliegenden Punkten) in den Netzen meist sehr hoch ist. Global, beispielsweise vom Rhein bis zur Oder, muss mit realen Abweichungen bis zu einigen Dekametern gerechnet werden.

Wenn davon ausgegangen wird, dass die Positionen vorliegender Objekte fehlerfrei bestimmt wurden, könnten diese, bei gegebenen Transformationsparametern, relativ einfach in ein anderes Koordinatenreferenzsystem überführt werden. Leider ist es so, dass sich die Koordinaten der meisten Geodaten auf ältere Koordinatenreferenzsysteme beziehen. Die Realisierungen dieser Systeme (Festpunktnetze) sind fehlerbehaftet, und die Netze weisen mehr oder weniger große Spannungen (im Fall des DHDN können dies mehrere Meter sein) auf. Eine mögliche Lösung für dieses Problem ist das Vorhalten einer Datei mit Korrekturdaten. In einer solchen Datei werden die Koordinatendifferenzen (die sogenannten Shiftwerte) zweier unterschiedlicher Referenzsysteme vorgehalten. Die Transformation vom einen in das andere erfolgt nach einer Interpolation über das Anbringen von einfachen Verschiebungen. Die Genauigkeit hängt dabei von der Dichte der Stützpunkte ab.

Übersichten über existierende Koordinatenreferenzsysteme

Nachdem wir uns nun mit der Problematik von Koordinaten- und Referenzsystemen auseinandergesetzt haben, stellt sich sicher die Frage, wie Sie an die benötigten Informationen kommen, um vorhandene Geodaten in ein anderes Koordinatenreferenzsystem zu überführen. Wie zuvor schon erwähnt, werden hierzu entweder Transformationsparameter oder eine Tabelle mit Shiftwerten benötigt.

Da die Zahl der auf der Welt vorhandenen Koordinatenreferenzsysteme sehr groß ist, gibt es einige Institutionen, die Register von Parametersätzen zur Transformation zwischen verschiedenen CRS sowie die bestimmenden Parameter der CRS selbst verwalten. Die wichtigste Organisation ist derzeit das Surveying & Positioning Committee der International Organisation of Oil and Gas Producers (OGP). Diese Organisation hat im Jahr 2005 die Aufgabe von der European Petroleum Survey Group (EPSG) übernommen. Die dort verwaltete Datenbank mit den Transformationsinformationen wird als EPSG Geodetic Parameter Dataset bezeichnet und liegt bei Drucklegung in der Version 6.14 vor (<http://www.epsg.org>).

Die Datensätze können entweder als MS-Access-Dateien oder als SQL-Skripten kostenfrei heruntergeladen werden. Hierbei werden sowohl PostgreSQL als auch MySQL und Oracle unterstützt. Eine Neuerung stellt ein zusätzliches Webfrontend dar. Dieses Frontend wird als EPSG Geodetic Parameter Registry Service bezeichnet und steht unter <http://www.epsg-registry.org/> zu Verfügung (siehe Abbildung A-7). Die Datenbank enthält zwar sehr viele, aber leider eben nicht alle auf der Welt verwendeten CRS.

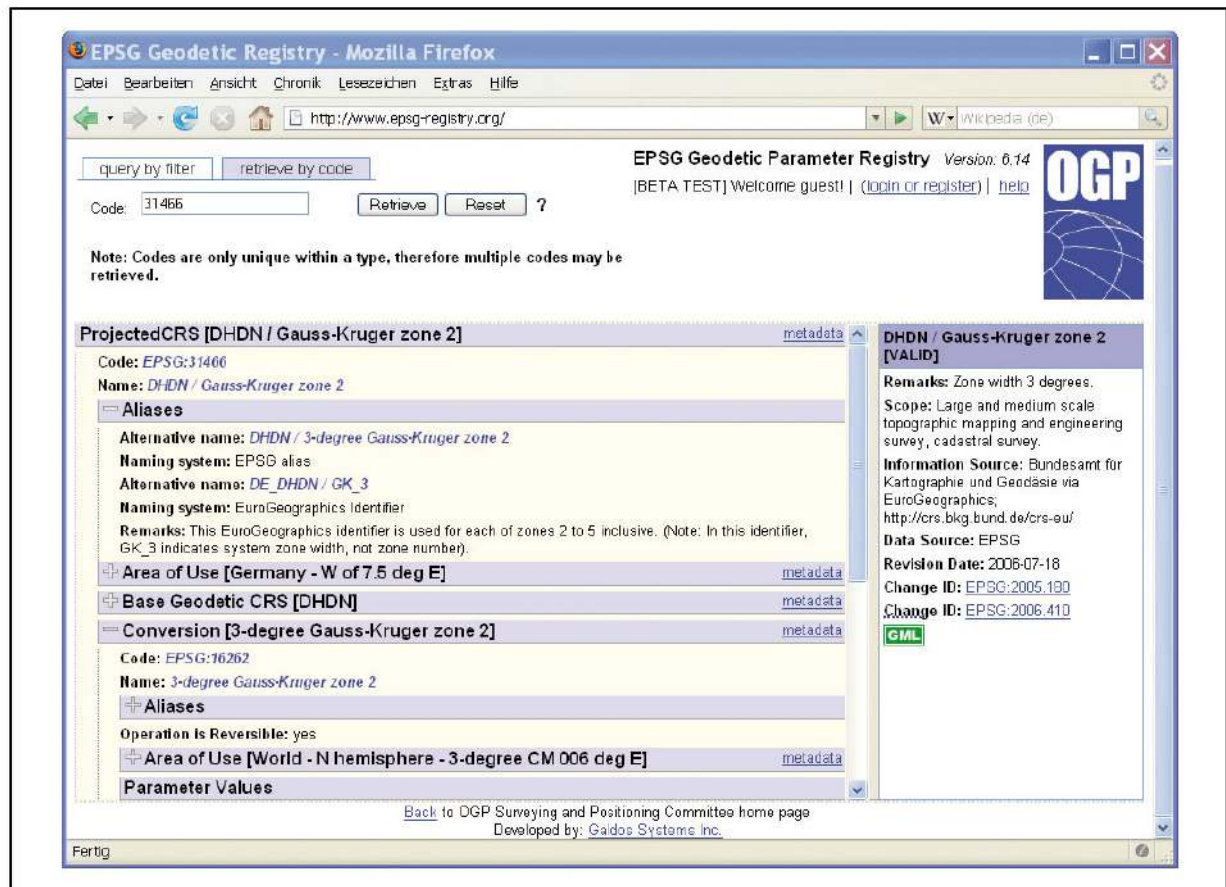


Abbildung A-7: EPSG Geodetic Parameter Registry Service-Informationsausgabe für das Koordinatenreferenzsystem

Neben der Möglichkeit, das Frontend der Registry zu nutzen, wird auch ein Webservice-Interface angeboten, das über Software direkt angesprochen werden kann. Der Service stellt Such- sowie Download-Funktionalitäten, beispielsweise als GML, zur Verfügung. In Abbildung A-8 ist das Ergebnis einer Abfrage für das Koordinatenreferenzsystem EPSG:31466 dargestellt (<http://www.epsg-registry.org/export.htm?gml=urn:x-ogc:def:crs:EPSG:31466>).

Hierbei handelt es sich um ein Gebrauchskoordinatensystem, das vor allem in den westlichen Bundesländern verwendet wird. Es werden nur die grundlegenden Beschreibungen zurückgegeben. Die weiteren Informationen sind über Xlinks miteinander verknüpft. Der folgende Link führt zur Rückgabe einer GML-Datei, die die Ellipsoiddimensionen des Bessel-Ellipsoids enthält:

<http://www.epsg-registry.org/export.htm?gml=urn:x-ogc:def:ellipsoid:EPSG:7004>

Neben der OGP gibt es noch weitere Institutionen, die ebenfalls Informationen zu unterschiedlichen Koordinatenreferenzsystemen zu Verfügung stellen. Hier ist insbesondere das Bundesamt für Kartografie und Geodäsie zu erwähnen. Unter <http://crs.bkg.bund.de/crs-eu/> können verschiedenste deutsche sowie paneuropäische Parametersätze abgerufen werden (siehe Abbildung A-9).

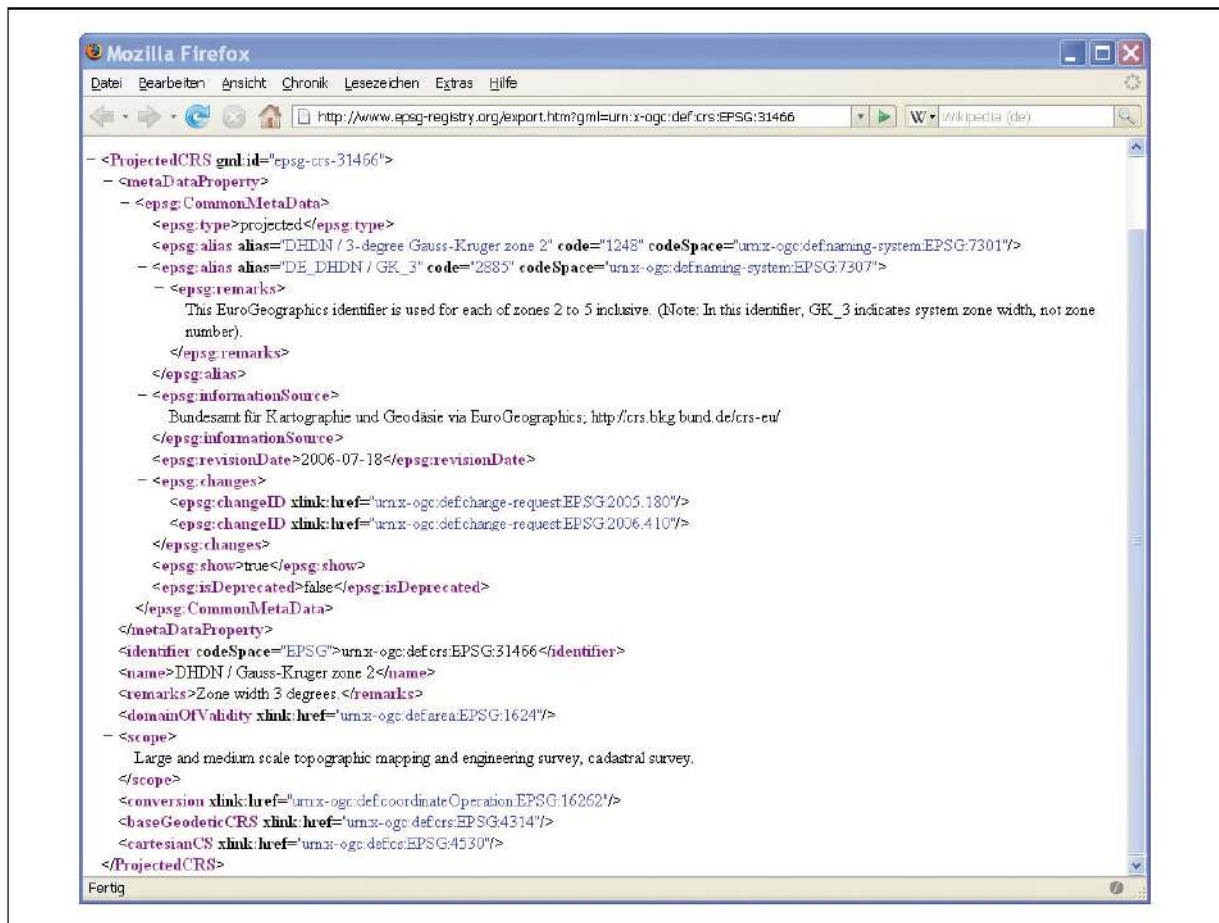


Abbildung A-8: EPSG Geodetic Parameter Registry Service – Ausgabe der Abfrage für EPSG:31466 als GML

Die Angaben auf der Seite enthalten zusätzlich Informationen über die Genauigkeit der Transformation. In Abbildung A-10 sehen Sie den Auszug einer Transformation vom Deutschen Hauptdreiecksnetz (DHDN) in das neue europäische Bezugssystem ETRS89. Die Parameter wurden aus 109 identischen Punkten abgeleitet und ergeben eine Transformationsgenauigkeit von 3 Metern.

Neben den beiden zuvor genannten Registries gibt es seit Kurzem auch eine Seite, die versucht, eine pragmatische Lösung anzubieten: <http://spatialreference.org>. Es ist ein Versuch, die Beschreibung von unterschiedlichen Koordinatenreferenzsystemen in den verschiedensten Formaten anzubieten. Es handelt sich dabei um einen sehr pragmatischen Ansatz. Während die beiden zuvor beschriebenen CRS-Registries eine mehr oder weniger standardisierte Form der Bereitstellung der Parameter gewählt haben, wird hier versucht, unterschiedlichste Transformationsbibliotheken/-programme mit Parametern in der von ihnen benötigten Form zu versorgen. Für die Bereitstellung der Daten werden die frei verfügbaren Bibliotheken GDAL und PROJ.4 verwendet. Ein schönes Beispiel für das derzeitige Angebot zeigt Abbildung A-11. Dort wurde über die URL <http://spatialreference.org/ref/epsg/31466/> ebenfalls das CRS EPSG:31466 abgefragt. Zusätzlich zu den von den Anbietern zur Verfügung gestellten Parametern können Nutzer bei diesem System auch eigene Transformationsansätze veröffentlichen.

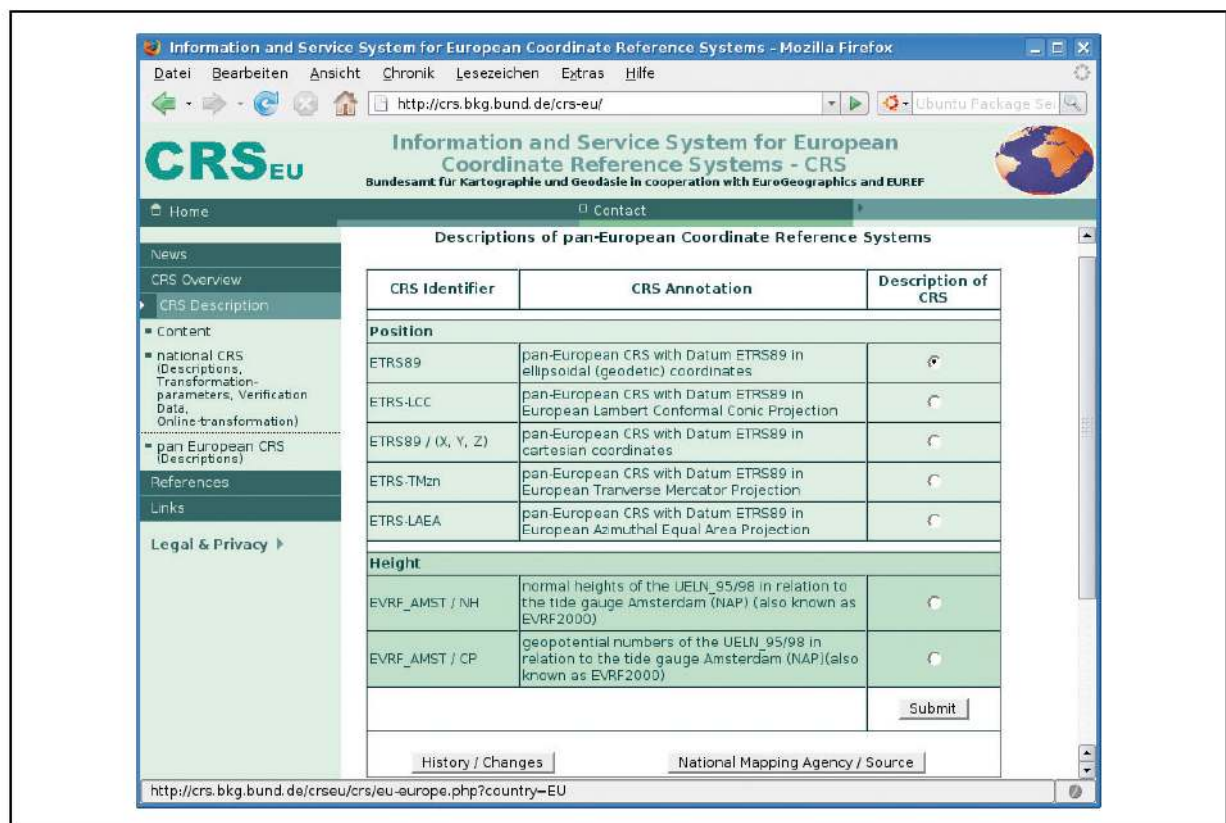


Abbildung A-9: CRS-Informationsseite des Bundesamts für Kartographie und Geodäsie

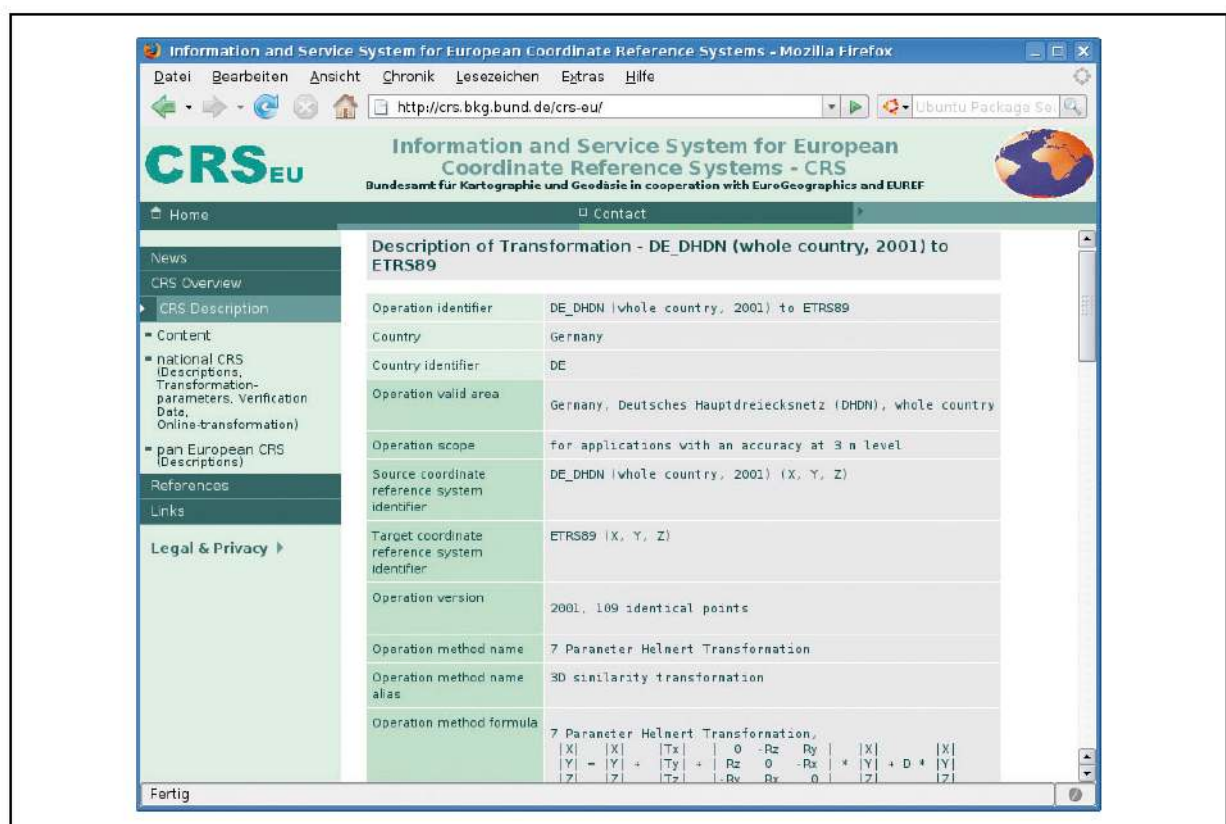


Abbildung A-10: CRS-Informationsseite – Ausgabe der Informationen zur Transformation von DHDN nach ETRS89

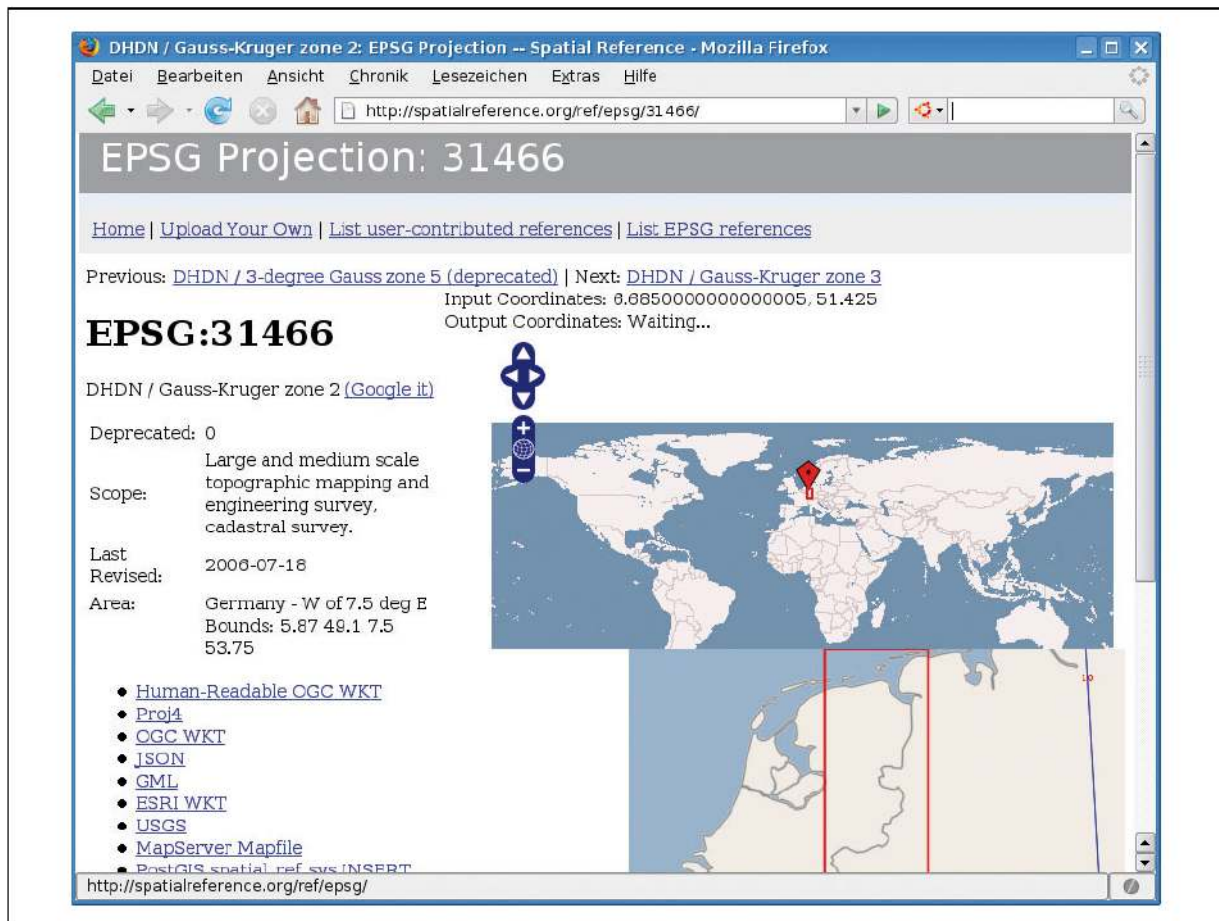


Abbildung A-11: Informationsausgabe zu EPSG:31466 von <http://spatialreference.org>

Praktische Umsetzung von Koordinatentransformationen mit PROJ.4

Die wichtigste Bibliothek, die beinahe von der gesamten freien GIS-Software verwendet wird, ist das Programm PROJ.4. Es wurde Anfang der 80er-Jahre zunächst in RATFOR (einem Fortran-Präprozessor) entwickelt. Mit der C-Migration Mitte der 80er konnte es auf Unix-Systemen genutzt werden.

Nach der Installation des Programmpakets (unter Debian/Ubuntu reicht ein `$ apt-get install proj`) stehen, neben den auch von MapServer verwendeten PROJ.4-Bibliotheken, verschiedene Kommandozeilenbefehle zur Verfügung. Die Eingabe von `proj` gibt beispielsweise folgende Meldung aus:

```
> proj
Rel. 4.5.0, 22 Oct 2006
usage: proj [ -beEfIlormsStTvVwW [args] ] [ +opts[=arg] ] [
files ].
```

Eine ausführliche Hilfe zur Verwendung der Parameter liefern die Manpages (`man proj`).

Wie der Name des Programms schon vermuten lässt, ist es in der Lage, geografische Koordinaten in eine ebene Darstellung zu überführen. Es lassen sich also Kartenprojektionen ableiten. Eine Liste der unterstützten Projektionsarten erhalten Sie nach der

Eingabe von `proj -l` auf der Kommandozeile. Es werden derzeit mehr als 120 unterschiedliche Projektionsverfahren zu Verfügung gestellt.

Neben dieser Funktion gibt es auch die Möglichkeit, Datumsübergänge berechnen zu lassen. Für diesen Zweck ist das Kommandozeilenprogramm `cs2cs`, das ebenfalls zum PROJ.4-Paket gehört, vorgesehen. Um die Liste der vordefinierten Ellipsoidparameter anzeigen zu lassen, genügt es, den Befehl `proj -le` einzugeben.

Die folgende Liste zeigt einen Auszug aus den Ergebnissen der Abfrage:

```
>proj -le
MERIT a=6378137.0      rf=298.257      MERIT 1983
SGS85 a=6378136.0      rf=298.257      Soviet Geodetic System 85
GRS80 a=6378137.0      rf=298.257222101 GRS 1980(IUGG, 1980)
IAU76 a=6378140.0      rf=298.257      IAU 1976
airy a=6377563.396     b=6356256.910   Airy 1830
APL4.9 a=6378137.0.    rf=298.25       Appl. Physics. 1965
NWL9D a=6378145.0.    rf=298.25       Naval Weapons Lab., 1965
mod_airy a=6377340.189 b=6356034.446   Modified Airy
andrae a=6377104.43    rf=300.0        Andrae 1876 (Den., Iclnd.)
aust_SA a=6378160.0    rf=298.25       Australian Natl & S. Amer. 1969
GRS67 a=6378160.0      rf=298.2471674270 GRS 67(IUGG 1967)
bessel a=6377397.155    rf=299.1528128   Bessel 1841
bess_nam a=6377483.865 rf=299.1528128   Bessel 1841 (Namibia)
.....
```

Um die Transformationen durchführen zu können, sind neben den Ellipsoiddimensionen auch die Transformationsparameter notwendig. Eine Liste der vordefinierten Datumsparameter erhalten Sie über die Eingabe von `proj -ld`.

```
>proj -ld
 datum_id ellipse definition/comments
WGS84 WGS84 towgs84=0,0,0
GGRS87 GRS80 towgs84=-199.87,74.79,246.62
Greek_Geodetic_Reference_System_1987
NAD83 GRS80 towgs84=0,0,0
North_American_Datum_1983
NAD27 clrk66 nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
North_American_Datum_1927
potsdam bessel towgs84=606.0,23.0,413.0
Potsdam Rauenberg 1950 DHDN
carthage clark80 towgs84=-263.0,6.0,431.0
Carthage 1934 Tunisia
hermannskogel bessel towgs84=653.0,-212.0,449.0
Hermannskogel
ire65 mod_airy towgs84=482.530,-130.596,564.557,-1.042,-0.214,-0.631,8.15
Ireland 1965
nzgd49 intl towgs84=59.47,-5.04,187.44,0.47,-0.1,1.024,-4.5993
New Zealand Geodetic Datum 1949
```

Die Liste, die in diesem Fall zurückgegeben wird, ist nicht besonders lang. Sie müssen jedoch beachten, dass dies nur die schon vorkonfigurierten Datumsübergangsdefinitionen sind. Sie erkennen hier deutlich, dass PROJ.4 verschiedene Möglichkeiten erlaubt, eine Datumstransformation zu berechnen.

In der Liste werden drei verschiedene aufgeführt:

towgs84=-199.87,74.79,246.62

Dieser Eintrag transformiert das jeweilige Datum über eine reine Verschiebung des Mittelpunkts ins WGS84. Die Achsen der beiden Bezugsellipsoide weisen hier in gleiche Richtungen (sie werden nicht gedreht!).

towgs84=482.530,-130.596,564.557,-1.042,-0.214,-0.631,8.15

In diesem Fall werden eine Verschiebung, Drehungen um alle drei Achsen sowie eine Änderung des Maßstabs durchgeführt.

nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat

Bei diesem Eintrag erfolgt der Übergang vom einen zum anderen Datum über die Verwendung einer Datei mit Koordinatendifferenzen (Shiftwerten). Dieser Ansatz führt, natürlich abhängig von der Qualität der verwendeten Differenzdaten, zur genauesten Koordinatentransformation. Lokale Unregelmäßigkeiten können bei einem solchen Verfahren berücksichtigt werden. (Weitere Informationen unter http://www.geod.nrcan.gc.ca/software/ntv2_e.php).

Wie schon im vorigen Abschnitt beschrieben, erhalten Sie mit der dritten Möglichkeit bei spannungsbehafteten Netzen die besten Ergebnisse. Da es sich bei dem in Deutschland verwendeten deutschen Hauptdreiecksnetz (DHDN) ebenfalls um ein spannungsbehaftetes Netz handelt, wurde Ende 2007 von der Arbeitsgemeinschaft der Vermessungsverwaltungen ein einheitlicher Transformationsdatensatz zu Verfügung gestellt. Mit diesem Datensatz ist es möglich, einen relativ genauen Übergang vom DHDN auf das in Zukunft innerhalb von Deutschland und Europa genutzte Bezugssystem ETRS89 zu erhalten. Die erreichbare Genauigkeit wird dabei mit unter 1 Meter angegeben. Diese Transformation wird für topografische Datensätze empfohlen, deren Erfassungsgenauigkeit 3 Meter nicht übersteigt. Der Transformationsparametersatz kann unter http://crs.bkg.bund.de/crseu/crs/dscrtrans/BeTA/de_dhdn2etrs_beta.php heruntergeladen werden. In einem ebenfalls dort erhältlichen PDF-Dokument wird die Nutzung des Datensatzes (Format *NTv2 – National Transformation Version 2*) unter der Verwendung der PROJ.4 eingehend beschrieben. Des Weiteren sind dort auch Beispieldaten zur Kontrolle der Transformation angegeben.

Die Transformation über PROJ.4 erfolgt nach dem für PROJ.4 typischen Verfahren. Zunächst werden die Ausgangsdaten mit der inversen Projektion in geografische Koordinaten auf dem jeweiligen Bezugsellipsoid (= geodätische Koordinaten) überführt, danach erfolgt der Datumsübergang mithilfe des NTv2-Datensatzes (Potsdam/Bessel nach WGS84/WGS84). Da der Datumsunterschied zwischen WGS84 und ETRS89 vernachlässigbar klein ist, können die so erhaltenen geodätischen Koordinaten über die Projektionsfunktion von PROJ.4 wieder in eine verebnete Darstellung überführt werden. Hier wird im Beispiel die Universale Transversale Mercatorprojektion (UTM) in der Zone 32N gewählt.

Beispiel aus der PDF-Dokumentation:

```
cs2cs +proj=tmerc +lat_0=0 +lon_0=6 +k=1.000000 +x_0=2500000 +y_0=0 +ellps=bessel  
+units=m +nadgrids=./BETA2007.gsb +to +proj=utm +ellps=GRS80 +zone=32<CR> 2490000.00  
5652000.00<CR>  
279488.01 5654871.71 0.00
```

Die aufgezeigte Vorgehensweise führt zu einer Genauigkeit der ebenen Koordinaten von unter 1 Meter. Es muss aber erwähnt werden, dass die ETRS89-Koordinaten seit 1989 (daher der Name) fest mit der europäischen Kontinentalplatte verbunden sind. Würde man heute hochgenaue absolute GPS-Positionsbestimmungen durchführen, lägen die erhaltenen Werte genau um die Kontinentalverschiebung der letzten 18 Jahre neben den real vorhandenen 1989 vermarkten Punkten. Wie bereits erwähnt, spielt diese Abweichung bei topografischen Daten keine Rolle. Anders sieht es bei Daten mit rechtlicher Wirksamkeit wie zum Beispiel den Liegenschaftskarten aus. Hier sollte eine eigene Transformation vermieden werden. Falls Sie zu Präsentationszwecken doch eine Transformation durchführen, sollten Sie deutlich auf eventuell vorhandene Ungenauigkeiten hinweisen.

Für viele Anwendungen (vor allem großmaßstäbliche) reichen in der Regel einfache Koordinatentransformationen aus. Hier liefert PROJ.4 eine eigene kleine Registry mit aus. In der Datei `epsg`, die nach der Installation im Programmpfad von PROJ.4 vorhanden ist, sind viele Parameter von Koordinatenreferenzsystemen aus der EPSG-Datenbank aufgelistet.

Einfache Transformationen

Um mit verhältnismäßig geringem Aufwand eine Transformation zwischen zwei unterschiedlichen CRS durchzuführen, kann das Programm `cs2cs` folgendermaßen aufgerufen werden (die verwendeten Parameter bis auf den Datumsübergang entsprechen dabei denen des vorigen Beispiels):

```
cs2cs +init=epsg:31466 +to +init=epsg:25832<CR>2490000.00 5652000.00<CR>279491.50 56548  
66.82 2.07
```

Das Ergebnis basiert diesmal auf der relativ einfachen Annahme, dass der Datumsübergang zwischen DHDN und ETRS89 auf einer reinen Verschiebung der Ellipsoidmittelpunkte beruht. Wie Sie beim Vergleich des Ergebnisses mit dem NTv2-Verfahren erkennen können, ergibt sich ein Lageunterschied von 6 Meter!

Das hier aufgeführte Beispiel sollte den Leser für die Probleme beim Umgang mit Koordinatentransformationen sensibilisieren und ihm eine kurze Hilfestellung bei der Nutzung von PROJ.4 geben.

Weitere Informationen finden Sie unter <http://proj.maptools.org/>.

Beispiele für Kartenprojektionen mit MapServer

Die Projektionseinstellungen in der MapServer-Mapdatei werden für zwei unterschiedliche Bereiche vorgenommen: einerseits für die Angabe der Projektion im MAP- Objekt für die Ausgabekarte und andererseits für die Angabe der Projektion der Ausgangsdaten, die in den Layern definiert wird. Beispiel A-1 zeigt ein Beispiel für die Projektionsangabe für die Ausgabekarte. Wie in den folgenden Beispielen zu erkennen ist, nutzt MapServer die gleichen Parameter, die auch schon bei der Beschreibung des PROJ.4-Programmpakets aufgeführt wurden. MapServer nutzt dabei die von PROJ.4 bereitgestellte Library.

Erfolgt die Angabe der Projektion nur im Map-Objekt, geht MapServer davon aus, dass diese Projektion auch für die Layer zutrifft.

Beispiel A-1: MapServer-PROJECTION-Objekt: Beispiel für die Verwendung einer bestimmten Projektion (hier: Albers flächentreuer Kegelentwurf)

```
PROJECTION
  "proj=aea"
  "ellps=WGS84"
  "lat_0=10"
  "lon_0=-90"
END
```

Die erste Zeile in Beispiel A-1 ist gleichzeitig die wichtigste: "proj=aea". Sie gibt an, welche Projektion verwendet werden soll. aea ist die Abkürzung für Albers Equal Area, bzw. Albers flächentreuer Kegelentwurf. Für jedes PROJECTION-Objekt muss über den Parameter proj der Name einer Projektion angegeben werden. Zudem sollten Sie auch über den Parameter ellps ein Ellipsoid angeben, z.B. ellps=WGS84. Einige Funktionen benötigen zwar keinen ellps-Parameter, schlagen aber fehl, wenn Sie ihn nicht angeben. Wenn Sie nicht sicher sind, was Sie hier angeben sollen, verwenden Sie an dieser Stelle WGS84 – es werden dann die Parameter des dem World Geodetic System 1984 (WGS84) zugrunde liegenden Bezugsellipsoids verwendet. Die Parameter stimmen mit denen des GRS80 (Geodetic Reference System 1980) überein. Alle übrigen Parameter sind optional.

Über den Parameter lat_0 wird die Ursprungslatitude in diesem Fall mit 10° Nord angegeben. Die projizierten Nord/Süd-Koordinaten beginnen hier an der Ursprungsbreite mit 0. Ohne die Angabe von lat_0 wird hier standardmäßig der Wert 0°, also der Äquator, verwendet.

Der Parameter lon_0 definiert den Hauptmeridian. lon_0=-90 legt den Hauptmeridian auf 90° West bzw. -90° fest. Die projizierten Ost/West-Werte starten an diesem Meridian mit 0. Der Standardwert von lon_0 ist 0°, entspricht also dem Meridian durch Greenwich.



Zusätzliche Informationen zum Umgang mit Projektionen mit MapServer finden Sie unter <http://mapserver.gis.umn.edu/docs/reference/mapfile/projection>.

Es folgen drei Beispiele für EPSG-Definitionen, die der epsg-Datei von PROJ.4 entnommen wurden. In der ersten Zeile wird der Name des Projektionssystems angegeben. Die zweite Zeile beginnt mit der Angabe des EPSG-Codes. Anschließend folgt in derselben Zeile eine Liste der Optionen, die für dieses Projektionssystem gelten.

```
# NAD83 / UTM zone 10N
<26910> +proj=utm +zone=10 +ellps=GRS80 +datum=NAD83
      +units=m +no_defs no_defs
<>

# WGS84 / LCC Canada
<42101> +proj=lcc +ellps=WGS84 +lat_0=0 +lon_0=-95
      +lat_1=49 +lat_2=77 +x_0=0 +y_0=-8000000.0
      +datum=WGS84 +units=m no_defs
<>

# DHDN / Gauss-Kruger zone 3
<31467> +proj=tmerc +lat_0=0 +lon_0=9 +k=1.000000 +x_0=3500000 +y_0=0 +ellps=bessel
+units=m +no_defs no_defs <>
```

Anstatt diese Informationen wie in Beispiel A-1 manuell festzulegen, reicht es oft aus, den entsprechenden EPSG-Code in der Mapdatei anzugeben.

```
PROJECTION
  "init=epsg:26910"
END
```

Hiermit werden alle Projektionsoptionen so eingestellt, dass die Karte in der UTM-Zone 10N mit dem Datum NAD83 projiziert wird. UTM-Zonen und andere Projektionen werden im folgenden Abschnitt erklärt.

Auf ähnliche Weise legt die Einstellung "init=epsg:42101" fest, dass als Projektionsmodell Lamberts flächentreuer Kegelentwurf (Lambert Conformal Conic, LCC) mit speziellen Einstellungen für die Darstellung von Kanada verwendet werden soll. Diese Definition gehört nicht zum offiziellen Standard und wurde unabhängig von der EPSG entwickelt. Sie ist daher nicht unbedingt Teil der Standard-EPSG-Datei. Viele zusätzliche Definitionen liegen PROJ.4 als separate Datei bei. Wenn Sie die gewünschten EPSG-Definitionen nicht finden, kann es gut sein, dass Ihnen jemand aus der MapServer-Mailingliste und dem MapServer-IRC-Channel weiterhelfen kann. Um zusätzliche Projektionen nutzen zu können, müssen Sie lediglich die zusätzlichen zwei Zeilen in die epsg-Datei einfügen, damit stehen sie Ihrer MapServer-Anwendung direkt zur Verfügung.

Ein Beispiel wäre das Koordinatenreferenzsystem des Landes Ruanda. Die epsg-Datei sowie die EPSG-Registry enthalten hierfür keinen Eintrag. Wenn Sie jedoch recherchieren, finden Sie heraus, dass es sogar in Ruanda ein einheitlich vorgegebenes Koordina-

tenreferenzsystem gibt. Es handelt sich dabei um eine Transversale Mercatorprojektion mit der Ursprungsbreite auf dem Äquator und dem Hauptmeridian 30E. Das System verfügt sowohl über einen eigenen Maßstabsfaktor als auch über eigene Minimalwerte für x und y. Das Bezugsellipsoid ist das Clarke Ellipsoid von 1880. Der folgende Eintrag in der epsg-Datei von PROJ.4 führt dazu, dass Sie in der MapServer-Konfiguration direkt den EPSG-Code 50001 verwenden können, um Daten aus einem beliebigen System in dieses System zu transformieren beziehungsweise Daten, die aus Ruanda stammen, in andere, weiter verbreitete Systeme zu überführen.

```
# Rwanda / Clarke 80
<50001> +proj=tmerc +lat_0=0 +lon_0=30 +k=0.999900 +x_0=500000
+y_0=10000000 +ellps=clrk80 +units=m +no_defs <>
```

EPSG oder epsg?

Bei der Verwendung von EPSG-Codes im PROJECTION-Objekt der MapServer-Mapdatei sollte der Text epsg prinzipiell kleingeschrieben werden. *epsg* verweist auf die Datei mit dem Namen epsg. In anderem Zusammenhang, zum Beispiel bei der Anforderung einer Karte eines WMS über einen getMap-Request, muss der EPSG dagegen in Großbuchstaben angegeben werden. Auch die Definition des WMS-Parameters wms_srs im META-DATA-Objekt erfolgt in Großbuchstaben:

```
"wms_srs" "EPSG:26910"
```

Dieses Problem tritt bei der Verwendung von EPSG-Codes mit MapServer häufig auf.



Im Prinzip können Sie epsg im PROJECTION-Objekt auch in Großbuchstaben schreiben. Allerdings müssen Sie dann auf Linux-/Unix-Betriebssystemen die epsg-Datei in EPSG umbenennen, da diese Betriebssysteme Groß- und Kleinschreibung unterscheiden.

Vorstellung ausgewählter Kartenprojektionen

Dieser Abschnitt soll einige häufig verwendete Kartenprojektionen vorstellen. Dabei werden für die Darstellung globale Daten, die im geografischen Koordinatensystem vorliegen, verwendet. Jedes Beispiel zeigt eine Projektion in einem passenden Maßstab und wird jeweils kurz beschrieben. Für ein umfassendes Verständnis sollten Sie jedoch zusätzliche Dokumente heranziehen.

Zur Referenz sind auch die entsprechenden MapServer-PROJECTION-Einstellungen angegeben. Sofern Sie für alle Ihre Ebenen eine gültige Projektion definiert haben, können diese Beispiele als Projektion der Ausgabekarte verwendet werden.



Für die Kartenbilder in diesem Abschnitt wurde anstelle des internen MapServer-Gitternetzgenerators (definiert mit dem Objekt GRID) eine Shape-Datei mit einem globalen Gitternetz verwendet. Das MapServer-GRID-Objekt erzeugt nur sehr einfache Linien, die von vielen Projektionen nicht korrekt dargestellt werden.

Im FWTools-Paket gibt es ein Python-Skript `mkgraticule.py`, mit dem das Gitternetz für die Karten in diesem Abschnitt erzeugt wurde. Weitere Informationen zu den FWTools finden Sie unter <http://fwtools.maptools.org/> und in den Kapiteln 3 und 6 in diesem Buch.

Einfache zylindrische Projektion

Abbildung A-12 zeigt eine einfache Weltkarte. Die globalen Koordinaten, Breiten- und Längengrade werden auf einem rechteckiges Gitternetz dargestellt. Jedes Rechteck hat auf der x- und der y-Achse die gleiche Größe. Dies ist bei MapServer die Standardprojektion, wenn eine `latlon`-Projektion verwendet wird. Dieses Modell wird auch als einfache zylindrische Projektion oder als *Plate Carrée* bezeichnet und stammt aus der Gruppe der äquidistanten zylindrischen Projektionen.

Diese Projektion eignet sich besonders gut zur Abbildung äquaturnaher Regionen. Je weiter Sie sich vom Äquator entfernen, desto verzerrter erscheint die Abbildung. Die Umrisse der Antarktis in Abbildung A-12 sind hierfür ein gutes Beispiel.

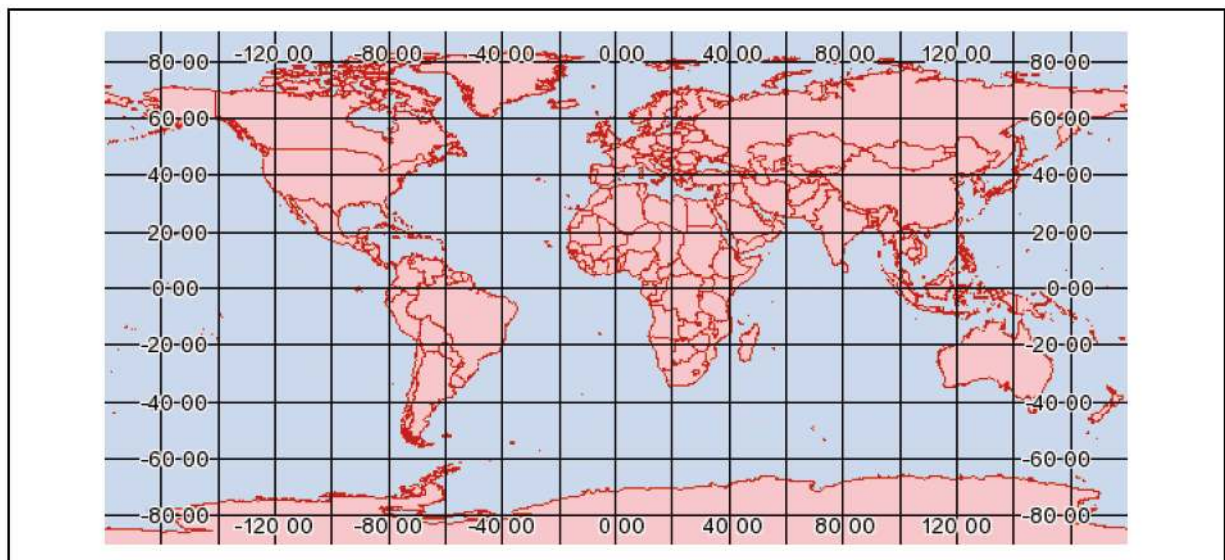


Abbildung A-12: Weltkarte in einer einfachen zylindrischen Projektion

MapServer-Syntax

Diese Projektion lässt sich leicht definieren. Sie brauchen hierzu nur den Namen der Projektion (`proj`) und des Ellipsoids (`ellps`) angeben. Die Angaben sollen anhand einer einfachen Mapdatei veranschaulicht werden:

*Beispiel A-2: Aufbau einer einfachen Mapdatei mit den Angaben
PROJECTION und EXTENT*

```
MAP
NAME world
SIZE 600 300
EXTENT -180 -90 180 90
UNITS METERS
```

Beispiel A-2: Aufbau einer einfachen Mapdatei mit den Angaben PROJECTION und EXTENT (Fortsetzung)

```
METADATA
  "wms_title" "World"
  "wms_onlineresource" "http://localhost/cgi-bin/mapserv?map=/srv/www/htdocs/
globalwmsclient.map"
  "wms_srs" "EPSG:4326"
END
PROJECTION
  "proj=latlon"
  "ellps=WGS84"
END

LAYER
  NAME countries
  TYPE POLYGON
  STATUS ON
  DATA countries_simpl
  METADATA
    "wms_title" "countries"
    "wms_srs" "EPSG:4326"
  END
  PROJECTION
    "init=epsg:4326"
  END
  CLASS
    STYLE
      COLOR 255 160 122
      OUTLINECOLOR 255 0 0
    END
  END
END
END
END
```

Orthografische Projektion

Abbildung A-13 zeigt ein Beispiel einer orthografischen Projektion. Diese Projektion stellt die Erde wie eine Kugel dar.

MapServer-Syntax

Die Abkürzung für diese Projektion lautet `ortho`. Anhand der Parameter `lat_0` und `lon_0` können Sie die Karte an einem bestimmten Längen- und Breitengrad zentrieren.

```
PROJECTION
  "proj=ortho"
  "lat_0=45"
  "lon_0=45"
  "ellps=WGS84"
END
```

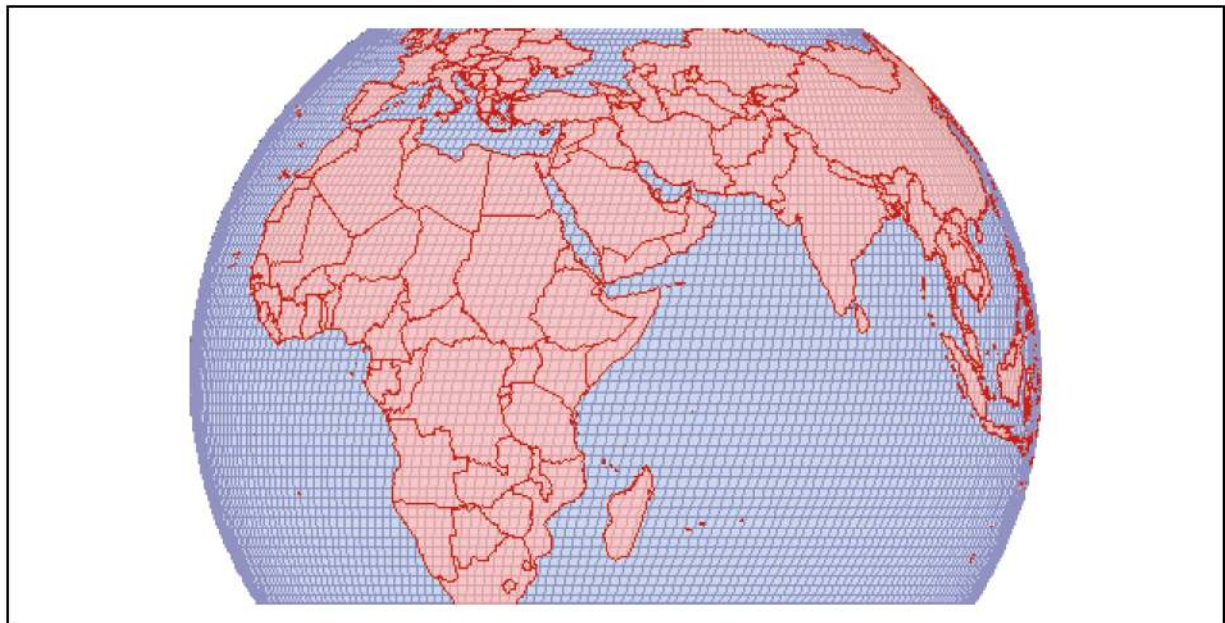
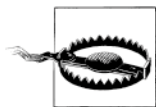


Abbildung A-13: Karte der östlichen Hemisphäre unter Verwendung einer orthografischen Projektion

Der Darstellungsbereich für diese Karte wurde in Metern angegeben:

```
EXTENT -8710000 -2739000 8710000 4493000
```

Da der Darstellungsbereich breiter ist als die Erde (nämlich 17.420 km), ist die Wölbung der Sphäre auf der Karte sichtbar. Nord- und Südpol sind ebenfalls aufgrund der Wahl des Bereichs ausgeblendet.



Zwar sehen orthografische Projektionen recht schön aus, allerdings erschweren bestimmte Einschränkungen in MapServer deren Benutzung. Gegenden in Polnähe oder in der Nähe der Außenbereiche (bei 180° oder -180° Länge) werden möglicherweise nicht oder nur lückenhaft dargestellt. Wenn die Objekte Ihrer Karte aus Linien mit nur wenigen Stützpunkten bestehen, werden diese vermutlich gar nicht angezeigt.

Mercatorprojektion

Abbildung A-14 zeigt ein Beispiel für die Mercatorprojektion. Auch diese Projektion ist für die Darstellungen von Bereichen in Äquatornähe besonders gut geeignet. Wie Sie sehen, ist das Gitternetz nicht regelmäßig, sondern zu den Polen hin gestreckt, wodurch besonders weit nördlich oder südlich gelegene Bereiche verzerrt dargestellt werden.

MapServer-Syntax

Die Abkürzung der Mercatorprojektion lautet merc:

```
PROJECTION
  "proj=merc"
  "ellps=WGS84"
END
```

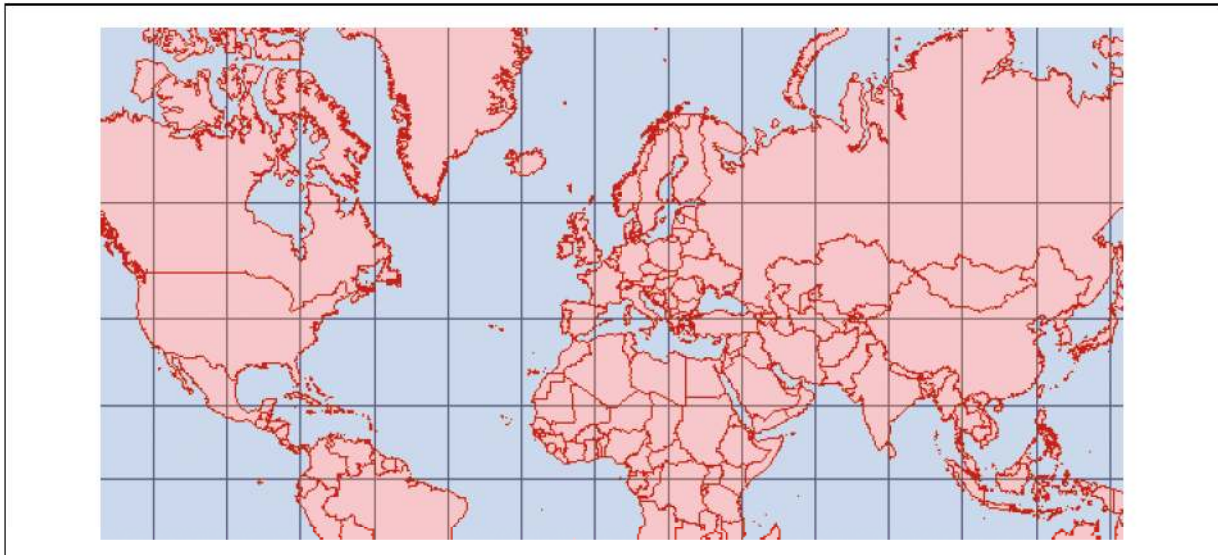



Abbildung A-14: Karte der nördlichen Breitengrade in der Mercatorprojektion

Der Darstellungsbereich wurde in Metern angegeben:

EXTENT -14896048 0 16000000 11779439

Transverse Mercatorprojektion

Abbildung A-15 präsentiert Ihnen ein Beispiel für die Transverse Mercatorprojektion. Wie die einfache Mercatorprojektion ist auch dieses Modell für die Darstellung der Bereiche in der Nähe der geografischen Ursprungslatitude (Latitude of Origin) geeignet, hier liegt die Ursprungslatitude bei 10° Nord.

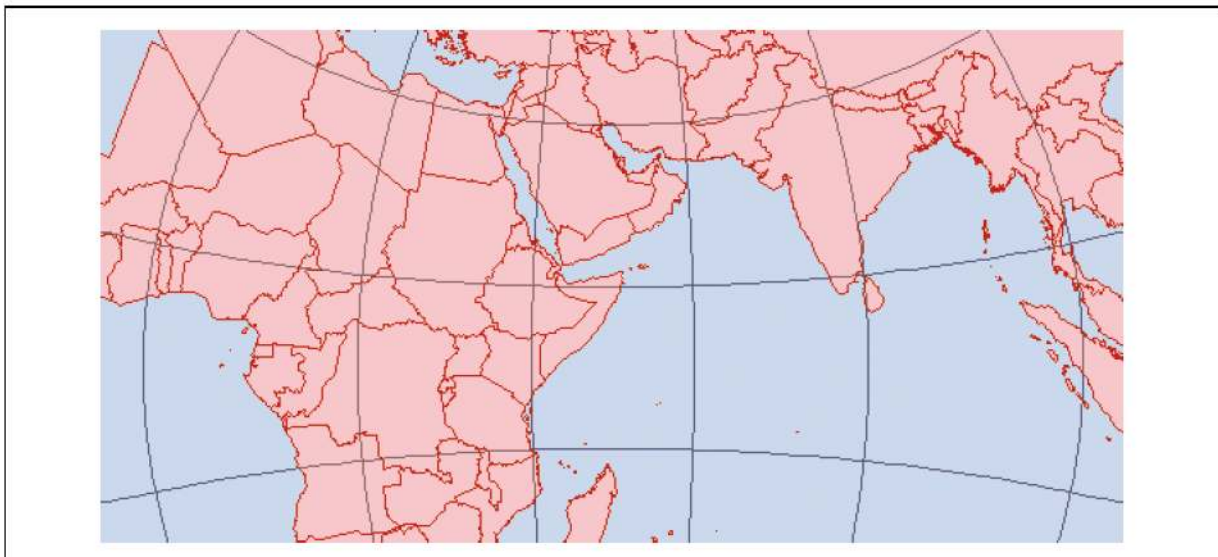


Abbildung A-15: Karte von Afrika und Südasien in der Transversen Mercatorprojektion

MapServer-Syntax

Der Kurzname dieser Projektion lautet tmerc:

```
PROJECTION
  "proj=tmerc"
  "lat_0=10"
  "lon_0=80"
  "ellps=WGS84"
END
```

Der Darstellungsbereich dieser Karte wurde in Metern angegeben:

```
EXTENT -500000 -3500000 500000 3500000
```

Albers flächentreuer Kegelentwurf

In Abbildung A-16 sehen Sie ein Beispiel für *Albers flächentreuen Kegelentwurf*. Er wurde entwickelt, um die relativen Größen der verschiedenen Gegenden der Erde korrekt wiederzugeben.

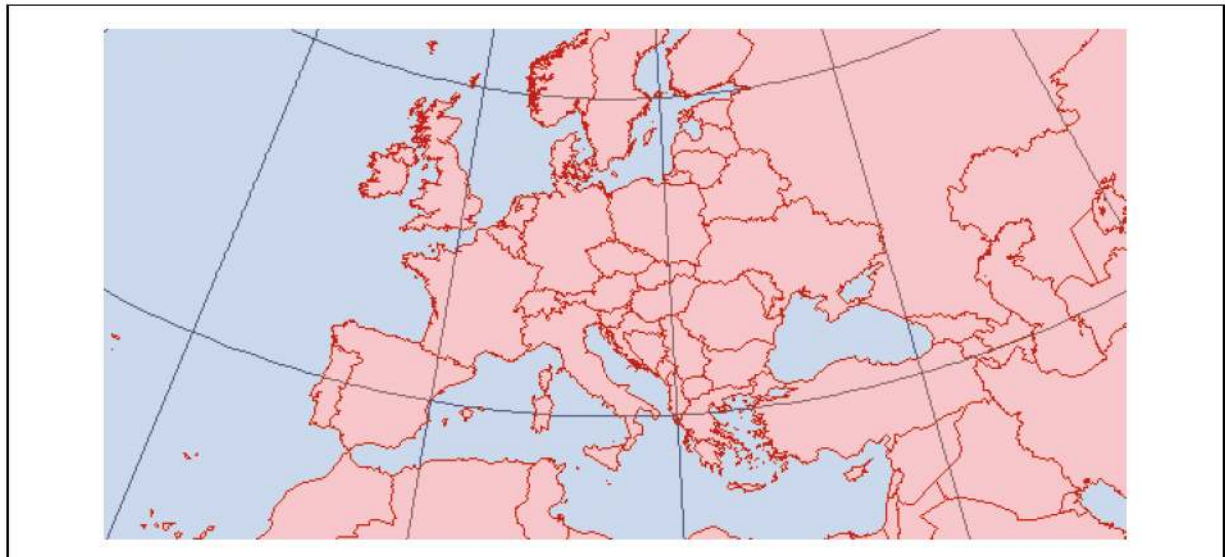


Abbildung A-16: Karte von Europa nach Albers flächentreuem Kegelentwurf

MapServer-Syntax

Der Kurzname für diese Projektion lautet aea. Es wurden außerdem eine erste und eine zweite Parallele definiert, wodurch der Bereich zwischen diesen Parallelen präziser abgebildet werden kann. Zudem wurde der Hauptmeridian über den Parameter lon_0 gesetzt.

```
PROJECTION
  "proj=aea"
  "lon_0=15"
  "lat_0=55"
  "lat_1=35"
  "ellps=WGS84"
END
```

Der Darstellungsbereich für diese Karte wurde in Metern angegeben:

```
EXTENT -100861 -2500195 100819 1000195
```

Stereografische Projektion

Abbildung A-17 zeigt ein Beispiel für eine stereografische Projektion.

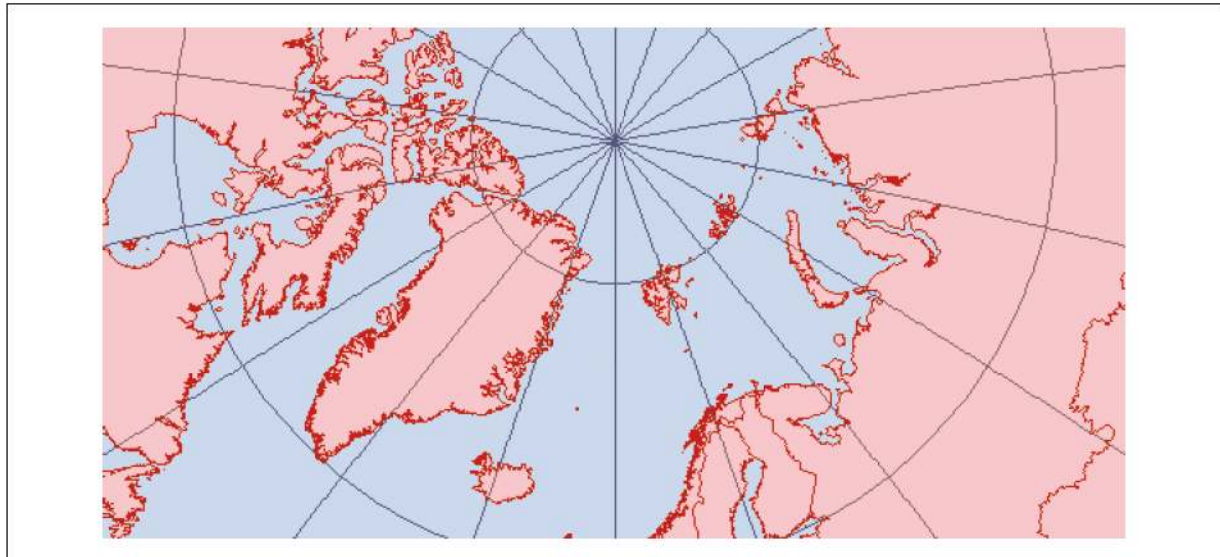


Abbildung A-17: Karte, die den Nordpol in der stereografischen Projektion abbildet

MapServer-Syntax

Der Kurzname für diese Projektion lautet `stere`. Als Hauptmeridian wurde der Standardwert 0° verwendet, daher erfolgt keine Angabe im `PROJECTION`-Objekt. `lat_0` setzt die Ursprungslatitude auf 80° Nord.

```
PROJECTION
  "proj=stere"
  "lat_0=80"
  "ellps=WGS84"
END
```

Der Darstellungsbereich für diese Karte wurde in Metern angegeben:

```
EXTENT -800861 -2000195 800819 2000195
```

UTM – Universale Transversale Mercator-Projektion

Die *Universale Transversale Mercator-Projektion*, kurz UTM, ist vermutlich das am häufigsten verwendete Projektionsmodell. Es verwendet ein System aus 60 vordefinierten Zonen von jeweils 6° Breite und 90° Höhe vom Äquator zu den Polen. Die Nummerierung der Zonen beginnt bei -180° West und erhöht sich nach Osten. Am Äquator erfolgt eine Trennung in eine nördliche und eine südliche Zone. Je nachdem, ob sich die

Zone nördlich oder südlich vom Äquator befindet, wird ein N bzw. ein S angefügt, z.B. 1N oder 1S.

Die UTM-Zonen bilden ein regelmäßiges Gitter, das lediglich im nördlichen Europa Unregelmäßigkeiten aufweist. Der Hauptmeridian liegt in der Mitte der Zone und überquert die erste Parallele, den Äquator, im Zentrum der Zone. Anstelle von Längen- und Breitengraden werden die Koordinaten über die Zone und das *Easting*- und *Northing*-Koordinatenpaar angegeben. Ostwert (Easting) ist dabei die projizierte Distanz der Position vom Hauptmeridian der Zone, und Nordwert (Northing) ist die projizierte Distanz der Position vom Äquator.

Die UTM-Projektion wurde für die Darstellung regionaler und lokaler Karten entwickelt. Sie kann nicht für globale Übersichten, z.B. große Länder, verwendet werden. Sobald der Darstellungsbereich die Größe einer Zone überschreitet, sollten Sie möglichst eine andere Projektion wählen.

Abbildung A-18 zeigt eine Karte des regelmäßigen UTM-Zonengitters. In dieser Karte wurden die Unregelmäßigkeiten Nordeuropas nicht berücksichtigt. Die Abbildung sollte aber ausreichen, um eine Vorstellung vom UTM-Konzept zu bekommen.

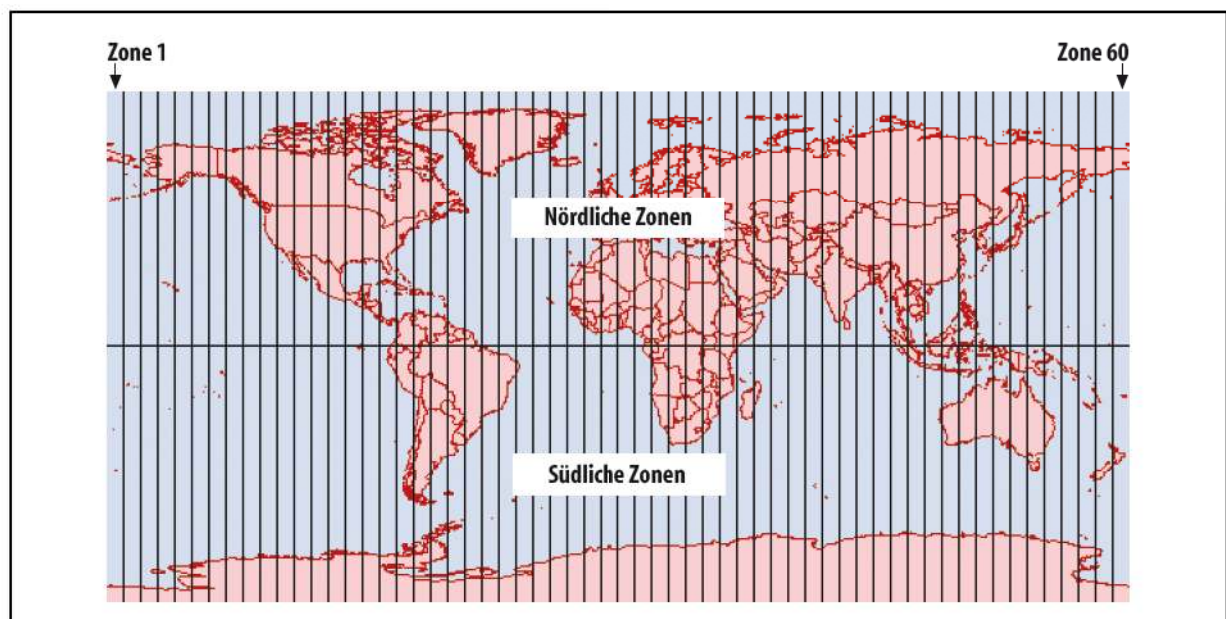


Abbildung A-18: Karte mit einer ungefähren Darstellung der UTM-Zonen

Als Beispiel zeigt Abbildung A-19 eine Karte von Kuba unter Verwendung der UTM-Zone 18N. Die gezeigten Längengrade sollen die Breite der UTM-Zone veranschaulichen. Da Kuba sich über mehr als eine UTM-Zone erstreckt, ist dieses Projektionsmodell vermutlich nicht optimal.

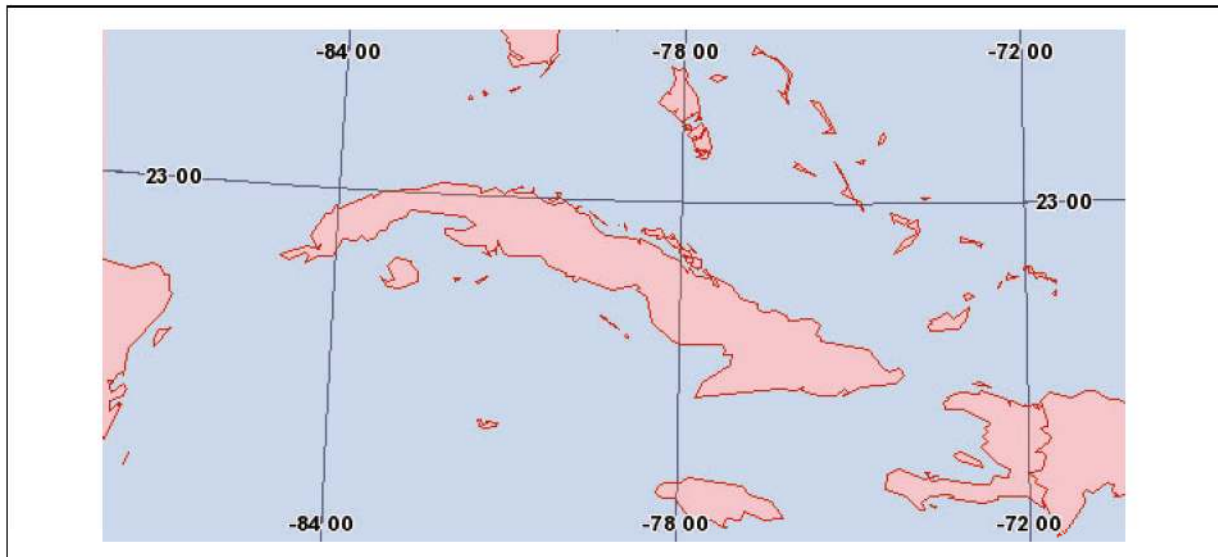


Abbildung A-19: Karte von Kuba in der Projektion UTM Zone 18N

MapServer-Syntax

```
PROJECTION
  "proj=utm"
  "zone=18N"
  "ellps=WGS84"
END
```

Der Darstellungsbereich der Karte wurde in Metern angegeben:

```
EXTENT -848000 2225000 990000 2565000
```

Einsatz von Projektionen in anderen Anwendungen

MapServer ist nur eine von vielen Anwendungen, die die PROJ.4 zur Projektion von Koordinaten verwendet. In den Kapiteln 2 und 6 in diesem Buch werden zwei Hilfsprogramme vorgestellt, in denen Sie mit ein paar Kommandozeilenoptionen Koordinatentransformationen durchführen können. Mit den Kommandozeilenwerkzeugen `gdalwarp` und `ogr2ogr` (Teil des GDAL/OGR-Pakets, siehe <http://gdal.org>) können Sie Raster- bzw. Vektordaten zwischen verschiedenen Formaten umwandeln. Mit der Option `-t_srs` können Sie dabei das gewünschte geografische Referenzsystem angeben.

Mit `gdalwarp` können Sie beispielsweise ein Bild, das in geografischen Breiten- und Längengraden vorliegt, in die Mercatorprojektion konvertieren:

```
> gdalwarp -t_srs "+proj=merc +ellps=WGS84" in.tif out.tif
```

Mit der Option `-s_srs` geben Sie auch das Referenzsystem der Datenquelle an. Dies ist besonders wichtig, wenn die verwendeten Daten noch keine eigenen Projektionsinformationen enthalten.

`ogr2ogr` ist ein Konvertierungsprogramm für Vektordaten. Das folgende Beispiel nutzt `ogr2ogr`, um eine Shape-Datei von einem Referenzsystem in ein anderes zu überführen.

Dabei wird das SRS für die Quelldatei ebenfalls angegeben, weil für die Quelldaten keine Projektionsinformationen (und keine .prj-Datei) existierten.

```
> ogr2ogr -s_srs "+proj=latlong" -  
t_srs "+proj=merc +ellps=WGS84" city_merc.shp city_in.shp
```

Weiterführende Informationen

Dieser Anhang kann nur eine Einführung in die Thematik der Projektionen liefern. Um mehr über Kartenprojektionen, technische Spezifikationen und kartografische Prinzipien zu erfahren, können Ihnen die folgenden Links weiterhelfen:

GDAL/OGR-Bibliotheken und -Hilfsprogramme

Kommandozeilenwerkzeuge und Bibliotheken für den umfassenden Zugriff auf Raster-(GDAL-) und Vektordaten (OGR), inklusive der Unterstützung für verschiedene Projektionsmodelle: <http://gdal.org>.

PROJ.4 – Cartographic Projections Library

Hauptseite für PROJ.4 mit weiteren Links zu technischen Dokumenten, die den Umgang mit den Kommandozeilentools proj sowie cs2cs näher beschreiben: <http://www.remotesensing.org/proj/>.

Deutschsprachige Wikipedia-Seiten

Hier existieren interessante Informationen zu folgenden Stichpunkten (Seitennamen): Erdgür, Geodätisches Datum, Kartenentwurfslehre, Aristoteles, Ptolemäus, Deutsches Hauptdreiecksnetz, Erdradius, Erdmessung, Eudoxos von Knidos, Ferro-Meridian, Kartennetzentwurf, Kartografie, Triangulation.

Eine sehr schöne und intensive Auseinandersetzung mit den Lehren des Ptolemäus:

<http://wwwuser.gwdg.de/~fhasele/ptolemaeus/>.

pimath - geo

Historisches zur Geodäsie und Beiträge zu Ellipsoid, Geoid und Sphäroid: <http://www.pimath.de/geo/index.htm>.

Wie die Erde zu einer Scheibe wurde

Artikel aus der Onlinepräsentation des Spiegel zur historischen Entwicklung des heutigen Weltbilds: <http://www.spiegel.de/wissenschaft/weltall/0,1518,381627,00.html>.

Bestimmung der Erdgröße

Eine schöne Abhandlung über einfache, auch praktisch durchführbare Messverfahren zur Bestimmung der Größe der Erde: www.didaktik.physik.uni-essen.de/heraeus_2005/DVD/Steinruecken/erdgroesse.pdf.

Ptolemaios-Forschungsstelle

Website mit einer Menge Informationen zur ptolemäischen Geografie: <http://www.ptolemaios.unibe.ch/>.

Online Koordinatentransformationen

Seite des Bundesamts für Kartografie und Geodäsie (BKG) mit einem Formular für Online-Koordinatentransformationen. Hier können sowohl Einzelwerte als auch ganze Dateien transformiert werden. Interessant zur Kontrolle der Transformationen mit unterschiedlichen Programmsystemen: https://upd.geodatenzentrum.de/geodaten/gdz_rahmen.gdz_div?gdz_spr=deu&gdz_akt_zeile=3&gdz_anz_zeile=6&gdz_user_id=0.

Coordinate Transformation Service (CTS)

Onlinedienst des BKG zur Integration in eigene Anwendungen. Es wird sowohl HTTP GET als auch HTTP POST unterstützt: https://upd.geodatenzentrum.de/geodaten/gdz_rahmen.gdz_div?gdz_spr=deu&gdz_akt_zeile=3&gdz_anz_zeile=6&gdz_unt_zeile=1&gdz_user_id=0, URL des Diensts: <https://upd.geodatenzentrum.de/cts/server?REQUEST=GetCapabilities>.

Prototyp-Koordinatenreferenzsystem-Registry

Abschlussbericht zu einer prototypischen Implementierung einer Registry für Koordinatenreferenzsysteme und Maßeinheiten der Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV), Stand August 2007: <http://www.adv-online.de/extdeu/binarywriterservlet?imgUid=e9460faf-3815-a411-a3b2-1718a438ad1b&uBasVariant=22222222-2222-2222-2222-222222222222&isDownload=true>.

Bücher

- *Meister der Kartographie* von Leo Bagrow & R. A. Skelton, Safari-Verlag, Berlin 1963
- *Vermessung und Ortung mit Satelliten* von Manfred Bauer, 4. Auflage, Wichmann, Heidelberg 1997
- *Vermessungskunde* von Heribert Kahmen, 18. Auflage, de Gruyter, New York 1993
- *Rechenverfahren und Auswertemodelle der Landesvermessung* von Bernhard Heck, 3. Auflage, Wichmann, Heidelberg 2003

MapServer-Referenz für den Zugriff auf Vektordaten

Dieses von Jeff McKenna und Tyler Mitchell erstellte Dokument wurde 2004 auf einem Workshop im Rahmen des Open Source-GIS/MapServer-Anwendertreffens in Ottawa, Kanada vorgestellt.

Es handelt sich dabei um eine umfangreiche Anleitung zur Verwendung verschiedener Vektordatenformate mit MapServer. Für die deutsche Auflage dieses Buchs wurden die Angaben überarbeitet und aktualisiert. In diesem Kapitel wird nur ein Teil der unterstützten Formate vorgestellt. Weitere Informationen zum Vektordatenzugriff finden Sie auf der MapServer-Website unter:

http://mapserver.gis.umn.edu/docs/reference/vector_data

Auf der FOSS4G-Konferenz 2007 fand ein Workshop zum Thema MapServer statt. Schauen Sie sich die Workshop-Unterlagen an. Hier stehen ebenfalls Demodaten zum Download bereit:

<http://www.foss4g2007.org/workshops/W-09/>

Weitere Daten und Beispielanwendungen finden Sie unter:

<http://mapserver.gis.umn.edu/download/current/>

Falls Sie Kommentare, Ideen oder Verbesserungsvorschläge zu diesem Dokument haben, können Sie die Autoren unter einer der folgenden Adressen kontaktieren:

Tyler Mitchell: tylermitchell@shaw.ca

Jeff McKenna: jmckenna@dmsolutions.ca

Vektordaten

Was sind Vektordaten? Dieses Zitat enthält eine gute Beschreibung zu Vektordaten:

Vektor: Eine Abstraktion der wirklichen Welt, bei der Positionsdaten in Form von Koordinaten dargestellt werden. Die einfachsten räumlichen Einheiten von Vektordaten sind Punkte, Linien und Polygone. So ist eine Linie beispielsweise eine Folge zusammenhängender Punkte, und ein Polygon besteht aus einer Folge miteinander verbundener Linien.

Vektorbasierte Grafiken werden mathematisch als eine Serie von Punkten und Linien definiert. Vektorbasierte Grafiken sind von der Auflösung des Ausgabemediums (z.B. Drucker oder Monitor) unabhängig. Sie erscheinen immer in der größten Auflösung des Ausgabemediums. Jedes Objekt ist eigenständig und enthält Informationen, wie Farbe, Form, Umriss und Position.

Quelle: http://coris.noaa.gov/glossary/glossary_l_z.html#v

MapServer ist in der Lage, auf verschiedene dateibasierte und datenbankbasierte Vektorformate zuzugreifen, kann aber auch Rasterdaten ansprechen. In dieser Zusammenfassung wird nur der Umgang mit Vektordaten behandelt. Dies beinhaltet die jeweilige Angabe in der MapServer-Mapdatei sowie den Datenzugriff mithilfe des Kommandozeilenwerkzeugs ogrinfo.

Zugriff auf Vektordaten mit UMN MapServer

MapServer bietet zwei Möglichkeiten für den Datenzugriff:

In MapServer eingebauter, formatspezifischer Datenzugriff

Die einfachste Form des Datenzugriffs nutzt die MapServer-eigenen Zugriffsmöglichkeiten, die beim Kompilieren eingebunden wurden. Diese sind allerdings auf wenige Vektorformate beschränkt, wie ESRI Shape, PostGIS, Oracle Spatial und ArcSDE. Standardmäßig wird von MapServer die Einbindung von ESRI Shape unterstützt.

Nutzung externer Bibliotheken für den Datenzugriff

Eine Stärke von MapServer besteht in seiner Fähigkeit, Datenformate über eine Pseudo-Plugin-Umgebung einzubinden. Die wichtigste externe Bibliothek ist dabei die GDAL/OGR, die den Zugriff auf eine Vielzahl zusätzlicher Raster- (GDAL) und Vektorformate (OGR) ermöglicht.

Verwendung von OGR für den Zugriff auf Vektordaten

Bei Bedarf verwendet MapServer hinter den Kulissen die OGR-Bibliothek. MapServer kann auch ohne die OGR-Bibliothek eingesetzt werden, doch ist sie je nach verwendetem Datenformat absolut notwendig. Da MapServer über OGR auf zahlreiche Datenformate zugreifen kann, ist es nicht nötig, die Unterstützung für diese Formate direkt in MapServer zu integrieren. Stattdessen werden die Erweiterungen in OGR vorgenommen und können dann in MapServer genutzt werden. Im Prinzip wandeln die Bibliotheken die Daten in ein internes, speicherbasiertes Format um, das MapServer verarbeiten kann. Dabei ist die Einbindung von Daten über OGR nahtlos und intuitiv.

Verschiedene Datenformattypen

Die Geodatenformate können in drei Kategorien unterteilt werden, die jeweils unterschiedlich behandelt werden müssen. Hier eine Liste der drei Typen und ein paar Beispiele:

Dateibasierte Formate

Shape, Microstation Design Files (DGN), GeoTIFF

Verzeichnisbasierte Formate

ESRI ArcInfo Coverages, U.S. Census TIGER

Datenbankbasierte Formate

PostgreSQL/PostGIS, ESRI ArcSDE, MySQL, Oracle Spatial

Jeder Datentyp besteht aus einer Datenquelle und einem oder mehreren Layern. Die folgenden beiden Definitionen gelten für MapServer und OGR:

Datenquelle

Eine Gruppe von Layern, die in einem gemeinsamen Repository (Depot) gespeichert werden. Dies kann eine Datei sein, die verschiedene Layer verwaltet, oder ein Verzeichnis, das mehrere Dateien enthält.

Layer

Eine Teilmenge der Datenquelle, die Informationen eines Vektordatentyps (Punkte, Linien, Polygone) enthält.

Dateibasierte Daten

Dateibasierte Daten bestehen aus einer oder mehreren Dateien, die in einem beliebigen Ordner vorliegen. In vielen Fällen (z.B. DGN) wird zur Speicherung der Daten nur eine Datei benötigt. ESRI Shape-Dateien bestehen dagegen aus mindestens drei Dateien mit unterschiedlichen Dateierweiterungen (*SHP*, *DBF*, *SHX*). In diesem Fall werden alle drei Dateien benötigt, da jede intern für eine andere Aufgabe zuständig ist.

Die Dateinamen werden normalerweise auch als Name der Datenquelle benutzt. Die enthaltenen Layer lassen sich allerdings nicht immer vom Dateinamen ableiten. Bei Shape-Dateien gibt es pro Shape-Datei eine Datenquelle und einen Layer, der den gleichen Namen trägt wie die Datei.

Verzeichnisbasierte Daten

Verzeichnisbasierte Daten bestehen aus einer oder mehreren Dateien, die auf bestimmte Weise in einem übergeordneten Verzeichnis gespeichert sind. Das Verzeichnis selbst ist dabei die Datenquelle. Oft ist es so, dass verschiedene Dateien innerhalb des Verzeichnisses die verfügbaren Layer repräsentieren. In manchen Fällen (z.B. Coverages) muss zusätzlich auf weitere Verzeichnisse zugegriffen werden, die sich an anderer Stelle im Verzeichnisbaum befinden.

ESRI ArcInfo Coverages bestehen zum Beispiel aus mehreren *ADF*-Dateien innerhalb eines Verzeichnisses. Hierbei enthält die Datei *PAL.ADF* die Polygondaten, während in der *ARC.ADF* Linien und Kreisbogen gespeichert werden. Das Verzeichnis enthält die Datenquelle, und jede *ADF*-Datei steht für einen Layer.

Datenbankverbindungen

Datenbankverbindungen besitzen eine Gemeinsamkeit mit datei- oder verzeichnisbasierten Strukturen: Auch sie stellen räumliche Informationen zur Interpretation durch MapServer zur Verfügung. Diese Aussage stellt die Abläufe, die innerhalb von MapServer ablaufen, stark vereinfacht dar. Doch im Wesentlichen braucht MapServer lediglich den Zugang zu den Koordinaten, die die Vektordaten aufbauen.

Datenbankverbindungen stellen Geodaten bereit, die temporär z.B. im Arbeitsspeicher gehalten und von MapServer gelesen werden, um daraus eine Karte zu erstellen.

Eine wichtige Unterscheidung zwischen verschiedenen Datenbanken muss getroffen werden. Bei den Datenbanken, die hier angesprochen werden, handelt es sich um *räumliche* Datenbanken, die Geometrien in einem eigenen Datentyp speichern können. Wenn Sie eine Datenbank vorliegen haben, die keinen eigenen Datentyp für Geometrien aufweist, können Sie zumindest Punkte über die Koordinaten speichern. Für alles, was darüber hinausgeht, ist aber eine räumliche Datenbank erforderlich. Für einige (Open Source und proprietäre) Datenbanken gibt es entsprechende räumliche Erweiterungen. Eine der leistungsstärksten räumlichen Erweiterungen ist PostGIS, eine Erweiterung für die PostgreSQL-Datenbank. PostGIS ermöglicht nicht nur die Speicherung räumlicher Daten, sondern auch deren Manipulation mithilfe von SQL-Befehlen. Eine weitere Open Source-Datenbank mit räumlichen Fähigkeiten ist MySQL.

Für die Verbindung mit der Datenbank sind die folgenden Verbindungsinformationen notwendig:

Host

Angabe der Adresse des Datenbankservers.

Datenbankname

Der Name der Datenbank, zu der die Verbindung aufgebaut werden soll.

Benutzername/Passwort

Die Zugangsberechtigungen sind normalerweise auf bestimmte Benutzer beschränkt. Über die Angabe von Benutzer und Passwort wird die Berechtigung geprüft.

Port

Angabe des Ports, über den auf die Datenbank zugegriffen werden kann.



Einige Datenbanken (z.B. Oracle) verwenden einen Bezeichner (Name Service Identifier), der sowohl den Rechner- als auch den Datenbanknamen enthält.

Für den Zugriff auf räumliche Daten werden folgende Angaben benötigt:

Name der Tabelle/der Sicht

Der Name der Tabelle oder der Sicht, in der sich die Geometrien befinden.

Name der Geometriespalte

Der Name der Tabellenspalte, in der die räumlichen Informationen vorliegen.

Vorstellung verschiedener Vektordatenformate

Anhand von Beispielen wird im Folgenden der Zugriff auf verschiedene Datenformate veranschaulicht. Dabei werden zunächst die wichtigsten Informationen über das jeweilige Format aufgeführt. Anschließend wird der Zugriff über ogrinfo und die korrekte Syntax für die MapServer-Mapdatei beschrieben. Die Formate werden abschnittsweise behandelt, dabei kann die Länge der Abschnitte ein oder mehrere Seiten betragen.

Die Abschnitte wurden so angelegt, dass sie unabhängig voneinander betrachtet werden können. Bestimmte Angaben wiederholen sich daher. Die Auflistung der Formate erfolgt nach der Häufigkeit ihrer Verwendung, die sich durch eine Umfrage in der MapServer-Nutzergemeinschaft ergeben hat (Stand 2004).

Die folgenden Formate werden behandelt:

- ESRI Shape-Dateien (SHP)
- PostGIS/PostgreSQL-Datenbank
- MapInfo-Dateien (TAB/MID/MIF)
- Oracle Spatial-Datenbank
- Web Feature Service (WFS)
- Geography Markup Language (GML)
- Virtueller Datenzugriff (ODBC/OVF)
- TIGER/Line-Dateien
- ESRI Arc Info Coverage Dateien (ADF)
- ESRI ArcSDE-Datenbank (SDE)
- Microstation Design-Dateien (DGN)
- IHO S-57-Dateien
- Spatial Data Transfer Standard-Dateien (SDTS)
- MapServer Feature-Objekte
- National Transfer Format-Dateien (NTF)



Die Einbindung von Daten aus einer MySQL-Datenbank, an dieser Stelle nicht beschrieben werden. Eine Beschreibung finden Sie unter folgendem Link: http://mapserver.gis.umn.edu/docs/reference/vector_data/mygis.

ESRI Shape (SHP)

Auch bekannt als Shapefile, Shape-Datei oder ESRI ArcView. ESRI ist die Firma, die dieses Format eingeführt hat; ArcView war dabei das erste Programm, das Shape-Dateien verwendete.

Liste der Dateien

Shape-Dateien bestehen aus mindestens drei Dateien mit gleichem Namen, aber unterschiedlichen Dateierweiterungen. MapServer benötigt für den Zugriff auf Shape die folgenden drei Dateien:

```
Countries_area.dbf
Countries_area.shp
Countries_area.shx
```

Datenzugriff/Verbindungsmethode

- Der Zugriff auf Shape ist direkt in MapServer integriert. Er kann zwar auch über OGR erfolgen, dies wird aber nicht empfohlen und soll hier auch nicht weiter vertieft werden.
- Der Pfad zur Shape-Datei wird benötigt. Bei der Angabe muss keine Dateierweiterung angegeben werden.
- Shape-Dateien enthalten immer nur Daten eines Layers, daher entfällt eine weitere Unterscheidung.

Zugriff über ogrinfo

- Das Verzeichnis kann als Datenquelle genutzt werden.
- Jede Shape-Datei eines Verzeichnisses entspricht einem Layer.
- Eine Shape-Datei kann auch als Datenquelle benutzt werden. In diesem Fall ist als Layer der Name der Shape-Datei anzugeben.

Beispiel für die Benutzung von ogrinfo mit Angabe eines Verzeichnisses, das mehrere Shape-Dateien enthält:

```
> ogrinfo /data/shapefiles/
INFO: Open of '/data/shapefiles/'
using driver 'ESRI Shapefile' successful.
1: wpg_h2o (Line String)
2: wpg_roads (Line String)
3: wpg_roads_dis (Line String)
4: wpgrestaurants (Point)
```

Beispiel für die Verwendung von ogrinfo mit Angabe einer einzelnen Shape-Datei:

```
> ogrinfo /data/shapefiles/Countries_area.shp
Had to open data source read-only.
INFO: Open of 'Countries_area.shp'
using driver 'ESRI Shapefile' successful.
1: Countries_area (Polygon)
```

Beispiel für den Einsatz von ogrinfo zur Ausgabe der Struktur der Daten:

```
> ogrinfo -summary /data/shapefiles/Countries_area.shp Countries_area
Had to open data source read-only.
INFO: Open of 'Countries_area.shp'
using driver 'ESRI Shapefile' successful.

Layer name: Countries_area
Geometry: Polygon
Feature Count: 27458
Extent: (-180.000000, -90.000000) - (180.000000, 83.627419)
```



```
Layer SRS WKT:
(unknown)
FAC_ID: Integer (5.0)
TILE: Integer (3.0)
ARCLIST: String (254.0)
NAM: String (77.0)
PERIMETER: Real (22.17)
POLYGONCOU: Integer (6.0)
NA2DESC: String (45.0)
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME my_shapefile
  STATUS ON
  TYPE POLYGON
  DATA countries_area
  CLASS
    NAME "Countries"
    STYLE
      OUTLINECOLOR 255 0 0
  END
END
END
```

PostGIS/PostgreSQL-Datenbank

PostGIS ist die von Refraction Research entwickelte räumliche Erweiterung für die PostgreSQL-Datenbank. Eine ausführliche Beschreibung zu PostGIS finden Sie auch in Kapitel 14 in diesem Buch.

PostGIS-Support in MapServer

PostGIS wird direkt von MapServer unterstützt, muss aber beim Kompilieren berücksichtigt werden. In den meisten Fällen müssen die PostgreSQL- und PostGIS-Bibliotheken (*.dll* oder *.so*) im Systempfad vorliegen, damit die Funktionalität genutzt werden kann. Hierzu gehören die Bibliotheken *libpq* und *libpostgis*.

Layerangabe in der MapServer-Mapdatei

- Angabe von CONNECTIONTYPE POSTGIS.
- Angabe des CONNECTION-Parameters:

```
"host=rechnername dbname=datenbankname user=benutzername password=benutzerpasswort
port=postgresport"
```

Die CONNECTION-Parameter können in beliebiger Reihenfolge angegeben werden und sind je nach Konfiguration der Datenbank optional. Der *datenbankname* muss angegeben werden. Der Standardwert für *rechnername* ist *localhost*, der Standardwert für *port* ist 5432 – der PostgreSQL-Standardport.

- Angabe des DATA-Parameters: "*geometriespalte from tabellenname*". Wurde *shp2pgsql* zum Laden der Daten verwendet, erhält die Geometriespalte standardmäßig den Namen *the_geom*, bei Verwendung von *ogr2ogr* lautet der Name *wkb_geometry*.

Beispiel:

```
LAYER
  NAME pg_test
  STATUS ON
  TYPE POLYGON
  CONNECTIONTYPE POSTGIS
  CONNECTION "host=mapserver.com dbname=world user=julio port=5432"
  DATA "wkb_geometry from province"
  CLASS
    ...
  END
END
```

Zusätzliche Informationen zu PostGIS und MapServer finden Sie unter http://mapserver.gis.umn.edu/docs/reference/vector_data/postgis und in der PostGIS-Dokumentation unter <http://postgis.refractory.net/docs/> in Kapitel 4.

MapInfo-Dateien (TAB/MID/MIF)

MapInfo-Dateien sind auch als TAB- oder MID/MIF-Dateien bekannt.

Liste der Dateien

Zu den TAB-Dateien gehören Dateien mit den Endungen *.DAT*, *.ID* und *.MAP*. Hier ein Beispiel:

```
border.DAT border.ID border.MAP border.TAB
```

Der Begriff MID/MIF bezieht sich auf die Dateien mit den Endungen *.MID* und *.MIF*.

Datenzugriff/Verbindungsmethode

- Der Zugriff auf TAB- und MID/MIF-Dateien wird in MapServer über OGR ermöglicht.
- Der Parameter `CONNECTIONTYPE OGR` muss angegeben werden.
- Der Pfad zur (*.tab*- oder *.mif*-)Datei muss mit Dateieindung angegeben werden.
- Der Pfad kann relativ zum Parameter `SHAPEPATH` angegeben werden.

MapInfo-Dateien enthalten bereits Zeichenvorschriften. Diese können durch die Angabe von `STYLEITEM "AUTO"` im LAYER-Block der Mapdatei verwendet werden.

Wenn Sie die Einstellung `STYLEITEM "AUTO"` benutzen, muss der Layer einen leeren `CLASS`-Block enthalten.

Zugriff über ogrinfo

Beispiel für die Verwendung von `ogrinfo` mit einer einzelnen TAB-Datei:

```
> ogrinfo elev5_poly.TAB
Had to open data source read-only.
INFO: Open of 'elev5_poly.TAB'
using driver 'MapInfo File' successful.
1: elev5_poly (Polygon)
```

Beispiel für die Verwendung von ogrinfo zur Ausgabe der Struktur der Daten:

```
> ogrinfo elev5_poly.TAB elev5_poly
Had to open data source read-only.
INFO: Open of 'elev5_poly.TAB'
using driver 'MapInfo File' successful.

Layer name: elev5_poly
Geometry: Polygon
Feature Count: 2236
Extent: (-141.000000, 60.000000) - (-124.403310, 69.300251)
Layer SRS WKT:
GEOGCS["unnamed",
  DATUM["MIF 0",
    SPHEROID["WGS 84 (MAPINFO Datum 0)",6378137.01,298.257223563],
    TOWGS84[0,0,0,0,0,0,0]],
  PRIMEM["Greenwich",0],
  UNIT["degree",0.0174532925199433]]
AREA: Real (0.0)
PERIMETER: Real (0.0)
ELEV5_: Integer (0.0)
ELEV5_ID: Integer (0.0)
TYPE: Real (4.0)
ELEV5: Real (4.0)
...
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME Elevation_Poly_5
  STATUS ON
  TYPE POLYGON
  CONNECTIONTYPE OGR
  CONNECTION "../hypso/elev5_poly.TAB"
  STYLEITEM "AUTO"
  CLASS
    NAME "Elevation Poly 5"
  END
END
```

Oracle Spatial-Datenbank

- MapServer unterstützt Oracle Spatial über OGR.
- OGR muss dabei mit Oracle Spatial-Unterstützung kompiliert werden, und MapServer wiederum muss mit OGR-Unterstützung kompiliert werden.
- MapServer unterstützt auch einen eigenen Zugriff auf Oracle Spatial.

Weitere Informationen zu Oracle Spatial und MapServer finden Sie in der MapServer-Dokumentation unter http://mapserver.gis.umn.edu/docs/reference/vector_data/oracle_spatial und http://mapserver.gis.umn.edu/docs/howto/oracle_spatial_howto.

Layerangabe in der MapServer-Mapdatei für den Zugriff über OGR

```
LAYER
...
CONNECTIONTYPE OGR
CONNECTION "OCI:user/pwd@service"
DATA "tablename"
...
END
```

Beispiel:

```
LAYER
NAME "Ottawa"
CONNECTIONTYPE OGR
CONNECTION "OCI:jeff/blah@ora_cities"
DATA "CITIES"
TYPE POINT
...
END
```

Layerangabe in der MapServer-Mapdatei bei Verwendung der nativen Unterstützung

```
LAYER
...
CONNECTIONTYPE oraclespatial
CONNECTION "user/pwd@service"
DATA "GEOMETRY FROM tablename"
...
END
```

Web Feature Service (WFS)

WFS ist eine Spezifikation des Open Geospatial Consortium (OGC). Nähere Informationen zu WFS finden Sie unter <http://www.opengeospatial.org/>. Schauen Sie sich zu WFS auch das Kapitel 12 in diesem Buch an.

WFS ermöglicht es einer Clientapplikation, räumliche Daten als GML (Geography Markup Language) von unterschiedlichen Web Feature Services abzufragen. GML wurde ebenfalls vom OGC entwickelt und baut auf der Standard-Websprache XML auf.

Im Unterschied zum Web Map Service, der nur eine fertige Rastergrafik der Daten zurückgibt, gibt der WFS eine Teilmenge der eigentlichen Daten im GML-Format zurück.

Eine ausführliche Beschreibung zum Einsatz von MapServer als WFS-Client finden Sie unter http://mapserver.gis.umn.edu/docs/howto/wfs_client.

Anfordern des Capabilities-Dokument

Mithilfe der Anfrage GetCapabilities an einen WFS-Server wird die Konfiguration des Diensts als Capabilities-Dokument im XML-Format zurückgegeben. Dieses Dokument enthält Informationen zu den verfügbaren FeatureTypes, Projektionen usw.

Beispiel für einen WFS GetCapabilities-Aufruf

```
http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
?VERSION=1.0.0
&SERVICE=WFS
&REQUEST=GetCapabilities
```

Ausschnitt aus einem Capabilities-Dokument

Beispiel B-1: Ausschnitt aus dem gelieferten XML des GetCapabilities-Requests

```
...
<FeatureTypeList>
  <Operations>
    <Query/>
  </Operations>
  <FeatureType>
    <Name>park</Name>
    <Title>Parks</Title>
    <SRS>EPSG:42304</SRS>
    <LatLongBoundingBox minx="-173.433" miny="41.4271" maxx="-13.0481" maxy="83.7466" />
  </FeatureType>
  <FeatureType>
    <Name>road</Name>
    <Title>Roads</Title>
    <SRS>EPSG:42304</SRS>
    <LatLongBoundingBox minx="-148.059" miny="35.882" maxx="-33.7745" maxy="72.5503" />
  </FeatureType>
  <FeatureType>
    <Name>popplace</Name>
    <Title>Cities</Title>
    <SRS>EPSG:42304</SRS>
    <LatLongBoundingBox minx="-172.301" miny="36.3541" maxx="-12.9698" maxy="83.4832" />
  </FeatureType>
</FeatureTypeList>
...
```

Datenzugriff/Verbindungsmethode

- MapServer unterstützt WFS als Basic WFS, also für den lesenden Zugriff auf Geodaten.
- MapServer unterstützt WFS in der Version 1.0.0.
- Angabe des Parameters CONNECTIONTYPE WFS.
- Angabe der OnlineResource über den Parameter CONNECTION.

WFS-FeatureTypes können über eine Layer-Definition in der Mapdatei angesprochen werden. Das GML kann aber auch direkt mit einem GetFeature-Request über einen Webbrowser ausgegeben werden. Dabei muss im GetFeature-Request der Parameter typename mit der Angabe des FeatureType-Namens angegeben werden. Den gewünschten FeatureType können Sie über die Angaben im GetCapabilities-Dokument ermitteln.

Bei der Einbindung von WFS-FeatureTypes in der Mapdatei müssen im Layer in einem META-DATA-Block zusätzliche Angaben gemacht werden.

Sie können die maximal per GML übergebenen Objekte mit dem Parameter MaxFeatures beschränken (z.B. &MAXFEATURES=100).

WFS GetFeature-Request

Über den folgenden GetFeature-Request fordert ein GML für den FeatureType road an. (Im Abschnitt »Anfordern des Capabilities-Dokument« weiter oben wurde beschrieben, wie Sie eine Liste der möglichen FeatureTypes für diesen Testserver erhalten.) Die URL wurde hier lediglich aus Gründen der Lesbarkeit mit Zeilenumbrüchen versehen.

```
http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
?VERSION=1.0.0
&SERVICE=WFS
&REQUEST=GetFeature&TYPENAME=road
&MAXFEATURES=100
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME "wfs_gmap_roads"
  STATUS ON
  TYPE LINE
  CONNECTIONTYPE WFS
  CONNECTION "http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap?"
  METADATA
    "wfs_version" "1.0.0"
    "wfs_srs" "EPSG:42304"
    "wfs_typename" "road"
    "wfs_request_method" "GET"
    "wfs_service" "WFS"
  END
  CLASS
    NAME "roads"
    STYLE
      COLOR 0 0 0
    END
  END
END # layer
```

Geography Markup Language (GML)

- Auch bekannt als GML/XML.
- Dateiformat.
- GML ist ein XML-Format zum Datenaustausch. Es enthält die Geometrie und die zugehörigen Attribute (»Features«).
- GML ist eine Spezifikation des Open Geospatial Consortium für den standardisierten Datenaustausch und die Datensicherung.

Liste der Dateien

GML-Daten werden üblicherweise in einer einzelnen Textdatei mit der Dateiendung *.gml* gespeichert (z.B. *coal_dep.gml*). Gelegentlich wird auch *.xml* als Dateiendung verwendet.

Oftmals liegen GML-Dateien auch XML-Schemadokumente bei, die den Aufbau des GML beschreiben. Schemadateien können beispielsweise über ogr2ogr erstellt werden.

Ausschnitt aus einer GML-Datei

Im GML werden die Attribute und Angaben zur Geometrie in XML-Tags ausgegeben:

```
<gml:featureMember>
  <Coal_Deposits fid="1">
    <UNKNOWN>0.000</UNKNOWN>
    <NA>0.000</NA>
    <ID>2</ID>
    <ID2>2</ID2>
    <MARK>7</MARK>
    <COALKEY>110</COALKEY>
    <COALKEY2>110</COALKEY2>
    <ogr:geometryProperty>
      <gml:Point>
        <gml:coordinates>78.531,50.694</gml:coordinates>
      </gml:Point>
    </ogr:geometryProperty>
  </Coal_Deposits>
</gml:featureMember>
```

Datenzugriff/Verbindungsmethode

- MapServer verwendet OGR für den Zugriff auf GML.
- Es muss der Parameter CONNECTIONTYPE OGR verwendet werden.
- Angabe des Pfads zur GML-Datei mit der entsprechenden Dateierweiterung.
- Eine GML-Datei kann aus mehreren Layern bestehen und mehrere FeatureTypes enthalten.

Zugriff über ogrinfo

Beispiel für die Verwendung von ogrinfo mit einer einfachen GML-Datei:

```
> ogrinfo /data/gml/coal_dep.gml
Had to open data source read-only.
INFO: Open of 'coal_dep.gml'
using driver 'GML' successful.
1: Coal_Deposits
```

Beispiel für die Verwendung von ogrinfo zur Untersuchung der Struktur eines Layers:

```
> ogrinfo -summary /data/gml/coal_dep.gml Coal_Deposits
Had to open data source read-only.
INFO: Open of 'coal_dep.gml'
using driver 'GML' successful.

Layer name: Coal_Deposits
Geometry: Unknown (any)
Feature Count: 266
Extent: (23.293650, 37.986340) - (179.272550, 80.969670)
Layer SRS WKT:
(unknown)
UNKNOWN: Real (0.0)
NA: Real (0.0)
ID: Integer (0.0)
ID2: Integer (0.0)
```

```
MARK: Integer (0.0)
COALKEY: Integer (0.0)
COALKEY2: Integer (0.0)
LONG: Real (0.0)
LAT: Real (0.0)
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME coal_deposits
  TYPE POINT
  STATUS ON
  CONNECTIONTYPE OGR
  CONNECTION "gml/coal_dep.gml"
  CLASS
    STYLE
      COLOR 0 0 0
      SYMBOL 'circle'
      SIZE 6
    END
  END
END
```

Virtueller Datenzugriff (ODBC/OVF)

Der virtuelle Datenzugriff wird in MapServer über OGR realisiert. Hiermit kann auf Datenbanken ohne räumliche Unterstützung sowie auf einfache Textdateien zugegriffen werden. Die Daten müssen eine x- und eine y-Spalte enthalten. Der Zugriff kann über eine ODBC-Verbindung oder über einen direkten Verweis auf eine Textdatei erfolgen.

Unterstützte Datenbanken

Der virtuelle Datenzugriff über die OGR-Erweiterung wurde mit den folgenden Datenbanken getestet und sollte in der Regel alle ODBC-Datenquellen unterstützen:

- Oracle
- MySQL
- SQL Server
- Access
- PostgreSQL

Unterstützte Typen von Textdateien

Der virtuelle Datenzugriff wird für das CSV-Format (Trennzeichen: Komma) sowie das TSV-Format (Trennzeichen: Tabulator) und für Dateien mit selbst definiertem Trennzeichen unterstützt.

Data Source Name (DSN)

Um die Verbindung mit Ihrer Datenquelle über ODBC herstellen zu können, müssen Sie einen DSN-Eintrag erstellen. Auf einigen Windows-Systemen *muss* SYSTEM-DSN verwendet werden.

Test der ODBC-Verbindung

Testen Sie Ihre Verbindung mit ogrinfo. Die Syntax für diesen Befehl lautet:

```
> ogrinfo ODBC:Benutzer/Passwort@DSN Tabellename
```

Beispiele für die Verwendung von ogrinfo

Beispiel für den Zugriff auf eine kommaseparierte Textdatei (CSV) über ODBC. Die Textdatei *coal_dep.txt* enthält eine Reihe von Breiten- und Längenangaben:

```
unknown,na,id,id2,mark,coalkey,coalkey2,long,lat
0.000,0.000,1,1,7,87,87,76.90238,51.07161
0.000,0.000,2,2,7,110,110,78.53851,50.69403
0.000,0.000,3,3,3,112,112,83.22586,71.24420
0.000,0.000,4,4,6,114,114,80.79896,73.41175
```

Für den Zugriff wurde Data_txt als DSN verwendet. Übergeben Sie ogrinfo als DSN den Wert *Data_txt*, erhalten Sie eine Liste der möglichen Dateien im jeweiligen Verzeichnis:

```
> ogrinfo ODBC:jeff/test@Data_txt
INFO: Open of `ODBC:jeff/test@Data_txt'
using driver `ODBC' successful.
1: coal_dep.csv
2: coal_dep.txt
3: coal_dep_nf.txt
4: coal_dep_trim.txt
...
```

Die Angabe von Benutzername und Passwort ist nicht immer notwendig.

```
> ogrinfo ODBC:@Data_txt
```

Der Befehl zur Anzeige weiterer Informationen zu einem bestimmten Layer lautet:

```
> ogrinfo ODBC:@Data_txt coal_dep.txt
INFO: Open of 'ODBC:@Data_txt'
using driver 'ODBC' successful.
```

```
Layer name: coal_dep.txt
Geometry: Unknown (any)
Feature Count: 266
Layer SRS WKT:
(unknown)
UNKNOWN: String (255.0)
NA: String (255.0)
ID: String (255.0)
ID2: String (255.0)
MARK: String (255.0)
COALKEY: String (255.0)
COALKEY2: String (255.0)
LONG: String (255.0)
LAT: String (255.0)
OGRFeature(coal_dep.txt):0
  UNKNOWN (String) = 0.000
....
```

Über die Angaben in einer ovf-Datei wird ein virtueller Layer erzeugt. Weitere Informationen zu ovf-Dateien finden Sie unter http://gdal.org/ogr/drv_vrt.html.

Aufbau einer *ovf*-Datei:

```
<OGRVRTDataSource>
  <OGRVRTLayer name="mylayer">
    <SrcDataSource>ODBC:user/pass@DSN</SrcDataSource>
    <SrcLayer>tablename</SrcLayer>
    <GeometryType>wkbPoint</GeometryType>
    <LayerSRS>WGS84</LayerSRS>
    <GeometryField encoding="PointFromColumns" x="x" y="y"/>
  </OGRVRTLayer>
</OGRVRTDataSource>
```

Beispiel einer *ovf*-Datei für *coal_dep.txt*:

```
<OGRVRTDataSource>
  <OGRVRTLayer name="coal">
    <SrcDataSource>ODBC:Data_txt</SrcDataSource>
    <SrcLayer>coal_dep.txt</SrcLayer>
    <GeometryField encoding="PointFromColumns" x="Long" y="Lat"/>
    <GeometryType>wkbPoint</GeometryType>
  </OGRVRTLayer>
</OGRVRTDataSource>
```

Layerangabe in der MapServer-Mapdatei

Layerangabe bei Verwendung einer *ovf*-Datei:

```
LAYER
  NAME "coal-test"
  STATUS ON
  TYPE POINT
  CONNECTIONTYPE OGR
  CONNECTION "coal.ovf"
  DATA "coal"
  METADATA
    "wms_srs" "4326"
    "wms_title" "coal-test"
  END
  CLASS
    STYLE
      COLOR 255 0 0
      SIZE 6
      SYMBOL "star"
    END
  END
END
```

Der Inhalt der *ovf*-Datei kann auch direkt in der Mapdatei angegeben werden:

```
LAYER
  NAME "coal-test"
  STATUS ON
  TYPE POINT
  CONNECTIONTYPE OGR
  CONNECTION "<OGRVRTDataSource>
  <OGRVRTLayer name='coal'>
  <SrcDataSource>ODBC:@Data_txt</SrcDataSource>
  <SrcLayer>coal_dep.txt</SrcLayer>
```

```

<GeometryField encoding='PointFromColumns' x='Long' y='Lat' />
<GeometryType>wkbPoint</GeometryType>
</OGRVRTLayer>
</OGRVRTDataSource>
DATA "coal"
METADATA
    "wms_srs" "4326"
    "wms_title" "coal-test"
END
CLASS
    STYLE
        COLOR 255 0 0
        SIZE 6
        SYMBOL "star"
    END
END
END

```

TIGER/Line-Dateien

TIGER/Line-Dateien werden vom U.S. Census Bureau (Behörde für Bevölkerungsstatistik der Vereinigten Staaten) verwendet. Oftmals werden diese Dateien einfach als TIGER-Dateien bezeichnet. Nähere Informationen zu diesem Format finden Sie unter <http://www.census.gov/geo/www/tiger/>.

Verwendete Dateien

Bei TIGER/Line-Dateien handelt es sich um Textdateien und verzeichnisbasierte Datenquellen. So enthält das Verzeichnis TGR06059 für einen bestimmten Bezirk mehrere Einzeldateien:

```

TGR06059.RT1 TGR06059.RT2 TGR06059.RT4 TGR06059.RT5
TGR06059.RT6 TGR06059.RT7 TGR06059.RT8 TGR06059.RTA
TGR06059.RTC TGR06059.RTH TGR06059.RTI TGR06059.RTP
TGR06059.RTR TGR06059.RTS TGR06059.RTT TGR06059.RTZ

```

Datenzugriff/Verbindungsmethode

- Der Zugriff auf TIGER/Line-Daten erfolgt über OGR.
- Im CONNECTION-String muss der vollständige Pfad zu dem Verzeichnis angegeben werden, das die benötigten Dateien enthält. Nach der Pfadangabe wird dem Parameter CONNECTION, durch ein Komma getrennt, die Zahl der gewünschten Layer angehängt, z.B.: CONNECTION "/tiger/data,0".
- Die Nummer des Layers entspricht der von ogrinfo zurückgegebenen Nummer minus eins.

Zugriff über ogrinfo

Beispiel für die Verwendung von ogrinfo unter Angabe des TIGER-Verzeichnisses zum Ermitteln der Layernummern:

```

> ogrinfo TGR06059 (NOTE that this is a directory)
ERROR 4: Tiger Driver doesn't support update.
Had to open data source read-only.

```

```

INFO: Open of 'TGR06059'
using driver 'TIGER' successful.
1: CompleteChain (Line String)
2: AltName (None)
3: FeatureIds (None)
4: ZipCodes (None)
5: Landmarks (Point)
6: AreaLandmarks (None)
7: Polygon (None)
8: PolygonCorrections (None)
9: EntityNames (Point)
10: PolygonEconomic (None)
11: IDHistory (None)
12: PolyChainLink (None)
13: PIP (Point)
14: TLIDRange (None)
15: ZeroCellID (None)
16: OverUnder (None)
17: ZipPlus4 (None)

```

Für den Layer CompleteChain gibt ogrinfo die Nummer 1 aus. Im CONNECTION-Parameter der Mapdatei müssen Sie diesen Wert minus eins angeben. Der für diesen Layer benötigte Wert lautet also tatsächlich 0.

Beispiel für die Verwendung von ogrinfo zur Untersuchung der Struktur des TIGER-Layers mit dem Namen CompleteChain:

```

> ogrinfo TGR06059 CompleteChain
ERROR 4: Tiger Driver doesn't support update.
Had to open data source read-only.
INFO: Open of 'TGR06059'
using driver 'TIGER' successful.

Layer name: CompleteChain
Geometry: Line String
Feature Count: 123700
Extent: (-118.125898, 33.333992) - (-117.412987, 33.947512)
Layer SRS WKT:
GEOGCS["NAD83",
  DATUM["North_American_Datum_1983",
    SPHEROID["GRS 1980",6378137,298.257222101]],
  PRIMEM["Greenwich",0],
  UNIT["degree",0.0174532925199433]]
MODULE: String (8.0)
TLID: Integer (10.0)
SIDE1: Integer (1.0)
SOURCE: String (1.0)
FEDIRP: String (2.0)
FENAME: String (30.0)
FETYPE: String (4.0)
FEDIRS: String (2.0)
CFCC: String (3.0)
FRADDL: String (11.0)
TOADDL: String (11.0)
FRADDR: String (11.0)

```



```
TOADDR: String (11.0)
FRIADDL: String (1.0)
TOIADDL: String (1.0)
FRIADDR: String (1.0)
TOIADDR: String (1.0)
ZIPL: Integer (5.0)
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME Complete_Chain
  TYPE LINE
  STATUS ON
  CONNECTIONTYPE OGR
  CONNECTION "/path/to/data/tiger/TGR06059,0"
  CLASS
    STYLE
      COLOR 153 102 0
    END
  END
END
```

ESRI ArcInfo Coverage-Dateien

ESRI ArcInfo Coverage-Dateien werden oftmals auch einfach als Coverages oder ADF-Dateien bezeichnet.

Liste der Dateien

Coverages bestehen aus einer Reihe von Dateien, die sich in einem gemeinsamen Verzeichnis befinden, das den Namen des Coverage trägt. Die Dateien repräsentieren verschiedene Layer, die die verschiedenen Topologien oder Geometriertypen beinhalten.

```
> ls /data/coverage/brazil
aat.adf arc.adf arx.adf bnd.adf lab.adf prj.adf tic.adf tol.adf
```

Zu einem Coverage gehört außerdem ein Verzeichnis mit dem Namen info, das sich auf der gleichen Verzeichnisebene befindet wie das Coverage-Verzeichnis. Soll das Coverage (mit den üblichen Werkzeugen des Dateisystems) kopiert werden, dürfen Sie nicht vergessen, das info-Verzeichnis mitzukopieren. Das info-Verzeichnis enthält eine Reihe von Kataloginformationen zum Coverage:

```
> ls /data/coverage/info
arc0000.dat arc0001.dat arc0002.dat arc.dir
arc0000.nit arc0001.nit arc0002.nit
```

Datenzugriff/Verbindungsmethode

- Als Verbindungsparameter muss der CONNECTIONTYPE OGR verwendet werden. MapServer besitzt keine eigene Unterstützung für Coverages.
- Angabe des Pfads zum Coverage-Verzeichnis.
- Anhand der Layernummer wird festgelegt, welche Objekte dargestellt werden sollen (diese Zahl entspricht dem von ogrinfo zurückgegebenen Wert minus eins).

Zugriff über ogrinfo

- Das Verzeichnis ist die Datenquelle.
- Die Layer befinden sich innerhalb dieses Verzeichnisses.

Beispiel für die Verwendung von ogrinfo unter Angabe des Coverage-Verzeichnisses:

```
> ogrinfo /data/coverage/brazil
INFO: Open of 'brazil'
using driver 'AVCBin' successful.
1: ARC (Line String)
2: CNT (Point)
3: LAB (Point)
4: PAL (Polygon)
```

Hier ein Beispiel für die Verwendung von ogrinfo zur Untersuchung der Layerstruktur:

```
> ogrinfo -summary /data/coverage/brazil PAL
Had to open data source read-only.
INFO: Open of 'brazil'
using driver 'AVCBin' successful.

Layer name: PAL
Geometry: Polygon
Feature Count: 1
Extent: (1272793.274958, 795381.617050) - (1287078.382785, 807302.747284)
Layer SRS WKT:
(unknown)
ArcIds: IntegerList (0.0)
AREA: Real (18.5)
PERIMETER: Real (18.5)
F_OPER#: Integer (5.0)
F_OPER-ID: Integer (5.0)
OPER: String (2.0)
FCODE: String (10.0)
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME Brazil_bounds
  TYPE POLYGON
  STATUS ON
  CONNECTIONTYPE OGR
  CONNECTION "/data/coverage/brazil,3"
  CLASS
    NAME "Brazil Admin Areas"
    STYLE
      OUTLINECOLOR 153 102 0
    END
  END
END
```

ESRI ArcSDE(SDE)

ArcSDE ist eine von ESRI entwickelte Erweiterung für den Zugriff auf verschiedene Datenbanken, z.B. SQL Server-, Oracle- und DB2-Datenbanken. ArcSDE wird auch als Spatial Database Engine bezeichnet, meistens aber einfach nur SDE genannt.

SDE-Unterstützung

- SDE wird direkt von MapServer unterstützt.
- MapServer 5.0 unterstützt SDE 8.1, 8.2, 8.3, 9.0, 9.1 und 9.2 sowie SDE für Coverages unter Linux-, Windows- und Solaris-Plattformen.
- MapServer unterstützt auch SDE-Rasterlayer, auf die wir hier aber nicht weiter eingehen werden.

Weiterführende Informationen zu SDE und MapServer finden Sie in der MapServer-Dokumentation unter http://mapserver.gis.umn.edu/docs/reference/vector_data/arc_sde.

Datenzugriff/Verbindungsmethode

1. Installieren Sie die SDE C API-Client-Bibliotheken.
2. Kompilieren Sie MapServer mit SDE-Unterstützung.
3. Definieren Sie die Layer in der Mapdatei.

Layerangabe in der MapServer-Mapdatei

1. Angabe von CONNECTIONTYPE SDE.
2. Die Verbindungsangaben für den Parameter CONNECTION erfolgen kommasepariert: *rechnername,instanzname,datenbankname,benutzername,password*
3. Angaben des DATA-Parameters: *tabellenname, geometriespalte*

Beispiel:

```
LAYER
  NAME test
  TYPE POLYGON
  STATUS ON
  CONNECTION "sde.dms.ca,port:5151,sde,user,password"
  CONNECTIONTYPE SDE
  DATA "NTDB.WATER,SHAPE"
  CLASS
    ...
  END
END
```

Microstation Design Files (DGN)

Diese Dateien werden auch als DGN-Dateien bezeichnet.

Liste der Dateien

Die Daten werden in einer einzelnen Datei gespeichert, die typischerweise die Dateierendung *.dgn* aufweist, beispielsweise: *0824t.dgn*.

Datenzugriff/Verbindungsmethode

- MapServer kann über OGR auf DGN-Dateien zugreifen.
- Angabe von CONNECTIONTYPE OGR.
- Angabe des Pfads zur *.dgn*-Datei mit Dateiergung.
- Alle Objekttypen werden bei DGN-Dateien in einem Layer vorgehalten. Dieser Layer trägt den Namen *elements* und ist der erste und einzige Layer.
- Welcher Objekttyp aus der DGN-Datei ausgelesen wird, hängt von der Definition des Parameters TYPE in der Mapdatei ab.
- DGN-Dateien enthalten Objekte vom Typ POINT, LINE, POLYGON und ANNOTATION.
- DGN-Dateien enthalten Informationen zur Darstellung der Daten (Farbe, Ausgestaltung usw.). Um diese Angaben zu nutzen, muss in der Mapdatei die Angabe STYLEITEM "AUTO" verwendet werden.



Die Standardfarbe für Objekte in DGN-Dateien ist Weiß. Beachten Sie, dass DGN-Objekte in Karten mit weißem Hintergrund nicht sichtbar sind.

Zugriff über ogrinfo

Aufruf von ogrinfo mit einer einzelnen DGN-Datei

Da eine DGN-Datei unterschiedliche Geometrietypen enthalten kann, wird bei Verwendung von ogrinfo im Ergebnis beim Layer *elements* kein Typ ausgegeben.

```
> ogrinfo /data/dgn/0824t.dgn
Had to open data source read-only.
INFO: Open of '0824t.dgn'
using driver 'DGN' successful.
1: elements
```

Verwendung von ogrinfo zur Ausgabe der Struktur der Daten

DGN-Dateien sind keine echten GIS-Dateien. Das Format entstand aus Formaten computer-gestützter Konstruktionsprogramme (CADD).

Die Dateien enthalten eine Reihe von Schlüsselattributen, die Sie in allen DGN-Dateien wiederfinden werden. Die meisten Attribute beziehen sich dabei auf die grafische Gestaltung der enthaltenen Objekte, wie ColorIndex, Style usw.

Die Information über die verwendete Projektion ist in DGN-Dateien nicht immer enthalten. Dies kann zu Problemen führen, wenn Sie versuchen, die DGN-Daten korrekt mit anderen Programmen darzustellen.

Im Zusammenhang mit den Einheiten können ebenfalls Probleme auftreten. In manchen Fällen werden Kilometerangaben oder Fuß (feet) verwendet, obwohl diese Information über ogrinfo nicht ersichtlich ist. Manchmal besteht die einzige Möglichkeit, die richtigen Einheiten herauszufinden, darin, die Datei mit Microstation-Software zu öffnen.

```
> ogrinfo -summary /data/dgn/0824t.dgn elements
INFO: Open of '0824t.dgn'
using driver 'DGN' successful.
```


Layer name: elements
Geometry: Unknown (any)
Feature Count: 22685
Extent: (-513183.050000, 150292.930000) - (-224583.220000, 407463.360000)
Layer SRS WKT:
(unknown)
Type: Integer (2.0)
Level: Integer (2.0)
GraphicGroup: Integer (4.0)
ColorIndex: Integer (3.0)
Weight: Integer (2.0)
Style: Integer (1.0)
EntityNum: Integer (8.0)
MSLink: Integer (10.0)
Text: String (0.0)

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME dgn
  TYPE LINE
  STATUS ON
  CONNECTIONTYPE OGR
  CONNECTION "dgn/0824t.dgn,0"
  STYLEITEM "AUTO"
  CLASS
  END
END
```

IHO S-57-Dateien

- Auch als S57 bekannt. Das IHO S-57-Format ist ein vektorbasiertes Format für den Datenaustausch digitaler hydrografischer Daten.
- IHO S-57 wurde von der International Hydrographic Organization (IHO) entwickelt. Weitere Informationen zur IHO finden Sie unter <http://www.iho.shom.fr/>.

Liste der Dateien

Einzelne S-57-Datendateien tragen die Endung *.000*, z.B. *US1BS02M.000*.

Datenzugriff/Verbindungsmethode

- Der Zugriff auf S-57-Daten erfolgt in MapServer über OGR.
- Es muss CONNECTIONTYPE OGR angegeben werden.
- Geben Sie den vollständigen Pfad oder einen relativen Pfad ausgehend vom SHAPEPATH zur *.000*-Datei als Wert für den Parameter CONNECTION an.
- Zusätzlich muss im Parameter CONNECTION auch eine Layernummer angegeben werden (diese entspricht der von ogrinfo zurückgegebenen Ebenennummer minus eins).

Besondere Anmerkungen

Um S-57-Dateien lesen zu können, benötigt der zugrunde liegende OGR-Code zwei Dateien aus der GDAL/OGR-Installation: *s57objectclasses.csv* und *s57attributes.csv*. Diese Dateien finden Sie im Verzeichnis */GDAL/data/*. Sollten Sie eine MapServer-Fehlermeldung wie die folgende erhalten:

```
msDrawMap(): Image handling error. Failed to draw layer named 's57'. msOGRFileOpen():  
OGR error. GetLayer( 9) failed for OGR connection
```

müssen Sie MapServer möglicherweise über den Parameter `CONFIG` im Map-Objekt Ihrer Mapdatei mitteilen, wo diese Dateien zu finden sind.

```
CONFIG GDAL_DATA "C:\gdal\data"
```

Zugriff über ogrinfo

Beispiel für die Verwendung von `ogrinfo` mit einer S-57-Datei, um die OGR-Referenznummer zu ermitteln:

```
> ogrinfo us1bs02m.000  
ERROR 4: S57 Driver doesn't support update.  
Had to open data source read-only.  
INFO: Open of 'us1bs02m.000'  
using driver 'IHO S-57 (ENC)' successful.  
1: ADMARE (Polygon)  
2: CBLSUB (Line String)  
3: CTNARE  
4: COALNE (Line String)  
5: DEPAARE  
6: DEPCNT (Line String)  
7: LNDARE  
8: LNDELV  
9: LNDGRN  
10: LNDMRK  
11: LIGHTS (Point)  
12: OBSTRN  
13: RDOSTA (Point)  
14: SEAARE  
15: SBDARE  
16: SLCONS  
17: SOUNDG (Multi Point)  
18: UWTROC (Point)  
19: WATTUR  
20: WRECKS  
21: M_COVR (Polygon)  
22: M_NPUB (Polygon)  
23: M_NSYS (Polygon)  
24: M_QUAL (Polygon)  
25: C_ASSO (None)
```

Beispiel für die Verwendung von `ogrinfo` zur Untersuchung der Struktur eines S-57-Layers:

```
> ogrinfo us1bs02m.000 DEPAARE  
ERROR 4: S57 Driver doesn't support update.  
Had to open data source read-only.
```

```

INFO: Open of 'us1bs02m.000'
using driver 'IHO S-57 (ENC)' successful.

Layer name: DEPARE
Geometry: Unknown (any)
Feature Count: 297
Extent: (165.666667, 48.500000) - (180.000000, 60.750000)
Layer SRS WKT:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]
GRUP: Integer (3.0)
OBJL: Integer (5.0)
RVER: Integer (3.0)
AGEN: Integer (2.0)
FIDN: Integer (10.0)
FIDS: Integer (5.0)
LNAM: String (16.0)
LNAM_REFS: StringList (16.0)
DRVAL1: Real (0.0)
DRVAL2: Real (0.0)
QUASOU: String (0.0)
SOUACC: Real (0.0)
VERDAT: Integer (0.0)
INFORM: String (0.0)
NINFOM: String (0.0)
NTXTDS: String (0.0)
SCAMAX: Integer (0.0)
SCAMIN: Integer (0.0)
TXTDSC: String (0.0)
RECDAT: String (0.0)
RECIND: String (0.0)
...

```

Layerangabe in der MapServer-Mapdatei

```

LAYER
  NAME s57
  TYPE POLYGON
  STATUS ON
  CONNECTIONTYPE OGR
  CONNECTION "/s57/us1bs02m.000,4"
  CLASS
    STYLE
      COLOR 247 237 219
      OUTLINECOLOR 120 120 120
    END
  END
END

```

Spatial Data Transfer Standard-(SDTS-)Dateien

Hierbei handelt es sich um ein Datenformat der U.S. Geological Survey (USGS). SDTS-Dateien können Raster- und Vektordaten enthalten. Das Rasterformat wird allerdings nicht von MapServer unterstützt. Nur die Vektorformate inklusive VTP und DLG können verwendet werden.

Liste der Dateien

SDTS-Dateien werden meistens nach US-Bundesstaaten zusammengefasst, zum Beispiel für den Staat Maryland, USA.

Die Dateien liegen für verschiedene Themenbereiche vor, z.B. Hydrografie, Transportwesen und administrative Grenzen.

Dieses Beispiel verwendet Daten zum Transportwesen. Insgesamt handelt es sich um 35 Dateien mit der Dateiendung *DDF*.

MDTRAHDR.DDF	MDTRARRF.DDF	MDTRCATS.DDF
MDTRDQCG.DDF	MDTRFF01.DDF	MDTRLE02.DDF
MDTRNA03.DDF	MDTRNO03.DDF	MDTRSPDM.DDF
MDTRAMTF.DDF	MDTRBFPS.DDF	MDTRCATX.DDF
MDTRDQHL.DDF	MDTRIDEN.DDF	MDTRLE03.DDF
MDTRNE03.DDF	MDTRPC01.DDF	MDTRSTAT.DDF
MDTRARDF.DDF	MDTRBMTA.DDF	MDTRDDSH.DDF
MDTRDQLC.DDF	MDTRIREF.DDF	MDTRNA01.DDF
MDTRNO01.DDF	MDTRPC02.DDF	MDTRXREF.DDF
MDTRARDM.DDF	MDTRCATD.DDF	MDTRDQAA.DDF
MDTRDQPA.DDF	MDTRLE01.DDF	MDTRNA02.DDF
MDTRNO02.DDF	MDTRPC03.DDF	

Datenzugriff/Verbindungsmethode

- Der Zugriff auf SDTS-Dateien wird in MapServer über OGR realisiert.
- Es muss der Parameter `CONNECTIONTYPE OGR` angegeben werden.
- Es muss der Pfad zur Katalogdatei (`????CATD.DDF`) inklusive Dateiendung angegeben werden.
- Ein SDTS-Katalog kann mehrere Layer enthalten. Einige dieser Layer enthalten nur Attribute, aber keine Geometrien.

Zugriff über ogrinfo

Aufruf von `ogrinfo` mit Angabe einer Katalogdatei (beachten Sie, dass die ersten sieben Zeilen keine räumlichen Informationen enthalten):

```
> ogrinfo /data/sdts/MD/MDTRCATD.DDF
Had to open data source read-only.
INFO: Open of 'MDTRCATD.DDF'
using driver 'SDTS' successful.
1: ARDF (None)
2: ARRF (None)
3: AMTF (None)
4: ARDM (None)
5: BFPS (None)
```


6: BMTA (None)
 7: AHDR (None)
 8: NE03 (Point)
 9: NA01 (Point)
 10: NA02 (Point)
 11: NA03 (Point)
 12: N001 (Point)
 13: N002 (Point)
 14: N003 (Point)
 15: LE01 (Line String)
 16: LE02 (Line String)
 17: LE03 (Line String)
 18: PC01 (Polygon)
 19: PC02 (Polygon)
 20: PC03 (Polygon)

Beispiel für die Verwendung von ogrinfo zur Ausgabe der Layerstruktur:

```

> ogrinfo -summary /data/sdts/MD/MDTRCATD.DDF LE01
Had to open data source read-only.
INFO: Open of 'MDTRCATD.DDF'
using driver 'SDTS' successful.

Layer name: LE01
Geometry: Line String
Feature Count: 780
Extent: (-80.000289, 36.999774) - (-74.999711, 40.000225)
Layer SRS WKT:
GEOGCS["NAD27",
  DATUM["North_American_Datum_1927",
    SPHEROID["Clarke 1866",6378206.4,294.978698213901]],
  PRIMEM["Greenwich",0],
  UNIT["degree",0.0174532925199433]]
RCID: Integer (0.0)
SNID: Integer (0.0)
ENID: Integer (0.0)
ENTITY_LABEL: String (7.0)
ARBITRARY_EXT: String (1.0)
RELATION_TO_GROUND: String (1.0)
VERTICAL_RELATION: String (1.0)
OPERATIONAL_STATUS: String (1.0)
ACCESS_RESTRICTION: String (1.0)
OLD_RAILROAD_GRADE: String (1.0)
WITH_RAILROAD: String (1.0)
COVERED: String (1.0)
HISTORICAL: String (1.0)
LIMITED_ACCESS: String (1.0)
PHOTOREVISED: String (1.0)
LANES: Integer (2.0)
ROAD_WIDTH: Integer (3.0)
BEST_ESTIMATE: String (1.0)
ROUTE_NUMBER: String (7.0)
ROUTE_TYPE: String (9.0)
  
```

Layerangabe in der MapServer-Mapdatei

```
LAYER
  NAME sdts_maryland
  TYPE LINE
  CONNECTIONTYPE OGR
  CONNECTION "data/sdts/MD/MDTRCATD.DDF,14"
  STATUS ON
  CLASS
    STYLE
      COLOR 0 0 0
    END
  END
END
```

MapServer-Features

MapServer verfügt über die Möglichkeit, durch die Angabe von Koordinaten in der Mapdatei, eigene Objekte zu erzeugen. Die Koordinaten werden dabei in FEATURE-Objekten in der Mapdatei definiert. MapServer-Features haben kein bestimmtes Datei- oder Datenbankformat und bedürfen daher auch keiner DATA- oder CONNECTION-Angabe in der Mapdatei.

Über das FEATURE-Objekt können Punkte, Linien und Polygone so definiert werden, als würden sie von einer externen Datenquelle stammen. Hierzu müssen die Koordinatenpaare in einer speziellen Syntax angegeben werden. Zusätzlich können die Objekte über den Parameter TEXT mit einem Text für eine Beschriftung versehen werden. Über den Parameter WKT können Geometrien als Wellknown-Text angegeben werden.

Weitere Informationen zu MapServer-Features (Inline Features) finden Sie unter <http://mapserver.gis.umn.edu/docs/reference/mapfile/feature>.

Datenzugriff/Verbindungsmethode

Dies ist eine interne MapServer-Option, für die keine externen Bibliotheken benötigt werden.

Layerangabe in der MapServer-Mapdatei

Im FEATURE-Objekt wird lediglich ein Geometrieobjekt beschrieben.

Punkte

Werden mehrere Punkte innerhalb eines Layers definiert, gelten für sie die gleichen CLASS-Angaben, z.B. für Farbe und Stil. Die Koordinaten werden in der Einheit angegeben, die für die Projektion des betroffenen Layers gilt. In diesem Beispiel gehen wir von Dezimalgrad als Einheit aus.

Beachten Sie, dass im ersten FEATURE-Block die Definition der Punkte über WKT erfolgt:

```
LAYER
  NAME inline_stops
  TYPE POINT
  STATUS ON
  PROJECTION
    "init=epsg:4326"
  END
  FEATURE
```

```

WKT "POINT(72.36 33.82)"
TEXT "My House"
END
FEATURE
  POINTS
    69.43 35.15
    71.21 37.95
    72.02 38.60
  END
  TEXT "My Stores"
END
CLASS
  STYLE
    COLOR 0 0 250
    SYMBOL 'circle'
    SIZE 6
  END
  LABEL
    SIZE medium
    COLOR 0 0 255
  END
END
END

```

Linien

Linien bestehen einfach aus einer Liste von Punkten. Allerdings muss TYPE LINE anstelle von TYPE POINT im Layer angegeben werden:

```

LAYER
  NAME inline_track
  TYPE LINE
  STATUS ON
  FEATURE
    POINTS
      72.36 33.82
      70.85 34.32
      69.43 35.15
      70.82 36.08
      70.90 37.05
      71.21 37.95
    END
  END
  CLASS
    STYLE
      COLOR 255 0 0
      WIDTH 3
    END
  END
END

```

Polygone

Polygone werden wie die Linien im vorangegangenen Beispiel angegeben. Dabei muss das letzte Koordinatenpaar dem ersten entsprechen, um das Polygon zu schließen. Anstelle von TYPE LINE muss für den Layer TYPE POLYGON angegeben werden.

National Transfer Format-(NTF-)Dateien

NTF-Dateien werden in erster Linie von der britischen Ordnance Survey (OS) verwendet. Weitere Informationen zu Ordnance Survey finden Sie unter <http://www.ordnancesurvey.co.uk>.

Liste der Dateien

NTF-Dateien verwenden die Dateierweiterung *.ntf*.

Datenzugriff/Verbindungsmethode

- Der Zugriff auf NTF-Dateien erfolgt über OGR.
- Der Parameter CONNECTION muss den gesamten Pfad oder den relativen Pfad ausgehend vom SHAPEPATH zur NTF-Datei enthalten.
- Im Parameter CONNECTION muss außerdem eine Layernummer angegeben werden (entsprechend der Ausgabe von ogrinfo minus eins).

Zugriff über ogrinfo

Beispiel für die Verwendung von ogrinfo zur Ausgabe der Layernummern:

```
> ogrinfo llcontours.ntf
ERROR 4: NTF Driver doesn't support update.
Had to open data source read-only.
INFO: Open of 'llcontours.ntf'
using driver 'UK .NTF' successful.
1: LANDLINE_POINT (Point)
2: LANDLINE_LINE (Line String)
3: LANDLINE_NAME (Point)
4: FEATURE_CLASSES (None)
```

Für LANDLINE_LINE gibt ogrinfo 2 als Nummer des Layers zurück. In der Mapdatei lautet die Angabe daher 1.

Beispiel für die Verwendung von ogrinfo zur Ausgabe der Struktur eines NTF-Layers:

```
> ogrinfo llcontours.ntf LANDLINE_LINE
ERROR 4: NTF Driver doesn't support update.
Had to open data source read-only.
INFO: Open of 'llcontours.ntf'
using driver 'UK .NTF' successful.

Layer name: LANDLINE_LINE
Geometry: Line String
Feature Count: 491
Extent: (279000.000000, 187000.000000) - (280000.000000, 188000.000000)
Layer SRS WKT:
PROJCS["OSGB 1936 / British National Grid",
    GEOGCS["OSGB 1936",
        DATUM["OSGB_1936",
            SPHEROID["Airy 1830",6377563.396,299.3249646,
                AUTHORITY["EPSG","7001"]],
            AUTHORITY["EPSG","6277"]],
```



```

        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433],
        AUTHORITY["EPSG","4277"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",49],
    PARAMETER["central_meridian",-2],
    PARAMETER["scale_factor",0.999601272],
    PARAMETER["false_easting",400000],
    PARAMETER["false_northing",-100000],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AUTHORITY["EPSG","27700"]]
LINE_ID: Integer (6.0)
FEAT_CODE: String (4.0)
...

```

Layerangabe in der MapServer-Mapdatei

```

LAYER
  NAME ntf_uk
  TYPE LINE
  CONNECTIONTYPE OGR
  CONNECTION "../ntf/llcontours.ntf,1"
  STATUS ON
  CLASS
    NAME "Contours"
    STYLE
      COLOR 0 150 200
    END
  END
END

```


Symbole

- < = -Operator 183
- = -Operator 183
- > = -Operator 183

A

- ADF-Datei 411
- AdV, Koordination des Amtlichen deutschen Vermessungswesens 83
- AGG 192
- Albers flächentreuer Kegelentwurf 403
- Apache Webserver 14
- Äquator 381
- Arbeitsgemeinschaft der Vermessungsverwaltungen (AdV) 394
- ARC.ADF-Datei 411
- Ausgabeformate
 - festlegen 190–192
 - Standardformat PNG 191
 - unterstützte Formate 191
 - WBMP 191
- awk-Befehl (Unix) 106

B

- Beobachtungen
 - Datenerhebung 18
 - Positionierung der Daten 19
 - Visualisierung der Daten in einer Karte 19
- Beschriftungen, Karte erweitern um 176
- Bildgröße anpassen, an den Darstellungsbereich der Karte 213
- Bildkatalog mit gdaltindex erzeugen 36
- Bounding Box 93
- Breitenabhängigkeit 380
- British Columbia (Canada) 79
- Bundesamt für Kartographie und Geodäsie 389

C

- CAD, Computer Aided Design 22
- CADD-Programme 430
- Canada GeoBase-Portal 82
- Canadian Government 82
- Capabilities-Dokument 240
- CartoWeb 229, 273, 373
- CGI-Programm 201
- Chameleon 229, 373
 - Webkartografie-System, Demo 373
- CIA World Factbook 76
- CLASSITEM 182, 183
- CLASS-Objekt 182, 183
- classObj-Objekt 360
- COLOR 181
- colorObj-Objekt 360
- column-Befehl (Unix) 107
- Coordinate Transformation Service (CTS) 408
- createdb, Programm 321
- cut-Befehl (Unix) 107
- Cywin, Programm zur Emulation einer Linux-Umgebung unter Windows 98

D

- Datawarehouse 10
- dateibasierte Daten 411
- Daten
 - Formate
 - proprietär 21
 - verstehen 21
 - Inhalte untersuchen 86–97
 - kartografische 14
 - konvertieren und betrachten 27–42
 - Sammlungen 78
 - (siehe auch Geodaten)
- Datenbanken (siehe räumliche Datenbanken)
- Datenbankverbindungen 412

- Datenbedarf einschätzen
 - Maßstab der Karte 73
 - Vektordaten, Typen 72
- Datenkonvertierung 24
- Datumsdefinitionen 384
- Debian Linux-Paketsammlungen 53
- DebianGIS 53
- deegree 234
 - WFS-T 254, 263
- Demodaten 85
- Desktop-GIS 121
 - GRASS GIS, Geographic Resources Analysis Support System 122
 - gvSIG 122, 313
 - JUMP Unified Mapping Project 122
 - Open Source Software Image Map (OSSIM) Info Sheet 122
 - OpenEV 122
 - OpenMap 122
 - Quantum GIS (QGIS) 122, 313
 - SAGA 122
 - Thuban 122
 - uDig 313
 - User-friendly Desktop Internet GIS (uDig) 122
- DGN, Intergraph Microstation Design 31
- Digital Elevation Model (DEM) 141
- Digital Terrain Model (DTM) 141
- digitale Karten 1–15
 - anpassen 2
 - im Vergleich zu konventionellen Karten 2
 - persönliche Karten 3
 - Schwierigkeiten 2
 - technologische Grenzen 3
- digitale Werkzeuge, Abhängigkeit von 20
- Digitales Geländemodell (DGM) 141
- Digitales Höhenmodell (DHM) 141
- DM Solutions 50, 359
- 3-D-Ansichten 74–76, 141
 - OpenEV 141
 - OSSIM 141

E

- Earth Observation Portal 82
- Eisentraut Peter
 - PostgreSQL: Das Offizielle Handbuch 325

- EPSG, European Petroleum Survey Group 388
 - EPSG Geodetic Parameter Dataset 388
 - EPSG Geodetic Parameter Registry Service 388
 - EPSG-Codes oder Projektions-Parameter 209
- ESRI ArcInfo Coverage-Dateien 411, 427
 - Beispiel für die Angabe in der Mapdatei 428
 - Beispiele für die Verwendung von ogrinfo 428
 - Datenzugriff/Verbindungsmethode 427
- ESRI ArcSDE 411, 429
 - Beispiel für die Angabe in der Mapdatei 429
 - SDE-Unterstützung 429
 - Verbindung mit SDE 429
- ESRI Shape (shp) 411, 413
 - (siehe auch Shape)
 - Beispiel für die Angabe in der Mapdatei 415
 - Datenzugriff/Verbindungsmethode 414
 - ogrinfo-Beispiele 414
- expand/unexpand-Befehl (Unix) 107
- EXPRESSION 183
- EXTENT 177, 204

F

- farbige Karten 180–183
- Farbtiefe erhöhen 192
- Feature Manipulation Engine (FME) 24, 313
- fertige Karten 77
- Festpunktnetz 385
- FGS Linux Installer 51
- FGS, Free Open Source Software GIS Suite 51
- FILTER 184
- FILTER-Schlüsselwort 183
- Flexible Internet Spatial Template (FIST) 229
- FOSS4G-Konferenz 6, 409
- FOSSGIS-Konferenz 8
- Frameworks zur Applikationsentwicklung 229
- FreeGIS 169
 - freegis.org-Informationsplattform 82
- frei verfügbare GIS-Anwendungen 27
- freie Geodaten 80–83
 - Linkliste 82
- Freshmeat-Website 98
- FWTools 36, 359, 399
 - Installation 85
 - Website 37

G

- Gauß-Krüger-Zone 2 329
- GDAL 32–37, 58, 192
 - gdal_merge.py 36
 - gdal_translate 35, 116–119
 - Bilder in andere Formate konvertieren 116
 - Erzeugen einer JPEG-Vorschau für Satellitenbilder 118
 - Liste der von GDAL unterstützten Formate 116
 - Umwandlung von Teilbildern 118
 - Verringern der Ausgabebildgröße 118
 - zur Ausgabe von Teilbereichen eines Bildes 118
 - gdaladdo 36
 - gdalinfo 35, 93
 - Parameter -mm, Informationen zu den Farbkanälen 97
 - gdaltindex 36, 168
 - gdalwarp 36
 - Hilfsprogramme 34
 - installieren 58
 - Liste der unterstützten Formate 116
 - OGR (siehe OGR)
 - ogrindex 168
 - SWIG und 34
 - unterstützte Rasterdatenformate 33
- GDAL/OGR 28, 85, 406, 410
 - GDAL (Geospatial Data Abstraction Library)
 - siehe GDAL
 - gdalinfo-Befehl 93
 - gdalwarp-Werkzeug 406
 - Pakete 406
 - Website für Bibliotheken und Werkzeuge 407
 - (siehe auch OGR, GDAL)
- GD-Bibliothek 64, 192
- GDI-DE, Geodateninfrastruktur Deutschland 81, 82
- Gebrauchskoordinatensystem 382, 389
- Gemeinde Zwischenwasser 82
- GeoConnections Discovery Portal 82, 238
- Geodaten 14, 24, 71–83, 169–171
 - 3-D-Ansichten 74–76
 - Analyse 85–107
 - CSV, comma-separated values 73
 - Datenanalyse per SQL 40
 - Datenbedarf einschätzen 71–76
 - Datenquellen im Netz 77
 - Datensammlungen 78
 - Demodaten 85
 - Deutschland 81
 - eingescannte Karten 71
 - finden 20
 - Formate 410
 - dateibasierte Daten 411
 - Datenbankverbindungen 412
 - verzeichnisbasierte Daten 411
 - freie Geodaten 80
 - Höhenmodelle 77
 - INSPIRE 81
 - Konvertierung (siehe Geodaten konvertieren)
 - Links zu freien Geodaten 79
 - ogrinfo zur Ausgabe von Detailinformationen 89–93
 - OpenStreetMap 80
 - OSGeo 5, 80
 - OSGeo Public Geospatial Data Project 82
 - Österreich 82
 - Rasterdaten 71
 - Typen 73
 - Satellitenbilder 77
 - Schweiz 81
 - standardisierte Dienste als Datenquellen 77
 - Vektordaten 71
 - vorgefertigte Karten 76
 - Werkzeuge zur Datenanalyse 85
 - worlddata, Ländergrenzen (FreeGIS) 169
- Geodaten konvertieren 109–119
 - Erzeugen einer JPEG-Vorschau für Satellitenbilder 118
 - GDAL/OGR (siehe GDAL/OGR)
 - gdal_translate (siehe GDAL)
 - GDAL-unterstützte Ausgabeformate für Rasterdaten 117
 - Gründe für Konvertierung 109
 - ogr2ogr (siehe ogr2ogr)
 - PostgreSQL/PostGIS (siehe PostGIS)
 - Quantum-GIS-Plugin SPIT zur Konvertierung von Shape nach PostgreSQL/PostGIS 115
 - Rasterdaten 115–119
 - Shape nach PostgreSQL/PostGIS (siehe shp2pgsql)
 - Vektordaten 110–115
- Geodateninfrastruktur Rheinland-Pfalz XIX, 52
- Geodatenportal der Stadt Remscheid 308
- geografische Breite 381

Geografische Informationssysteme (GIS) XIII
 geografische Koordinatensysteme 380
 geografische Länge 382
 Geography Markup Language (siehe GML)
 GEOLAND.AT, Geodatenportal der österreichischen Länder 83
 geometrische Berechnungen 20
 Geometry Engine Open Source (GEOS) 40
 geometry, Datentyp 335
 GeoNetwork opensource 234
 Geoportale 308
 Geoportal AT 82
 Geoportal Rheinland-Pfalz 308
 GeoPortal.rlp 83
 Georeferenzieren 161
 GEOS-Bibliothek 327
 GeoServer 41, 52, 234, 313
 WFS-T 263
 Geospatial Data Abstraction Library (siehe GDAL)
 GeoTIFF-Bilder 31, 411
 GIF-Format 176
 Gillies, Sean 359
 GIS Data Depot 82
 GIS, serverbasiert 314
 GISCO, Dienst des Europäischen Statistischen Amtes (Eurostat) 82
 GIS-Knoppix 53
 GIST (Generalized Search Tree) 339
 Global Change Master Directory 82
 Global Land Cover 82
 global.map 351
 GML (Geography Markup Language) 31, 110, 418, 420
 Beispiel für die Angabe in der Mapdatei 422
 Beispiele für die Verwendung von ogrinfo 421
 Datenzugriff/Verbindungsmethode 421
 ogr2ogr zur Konvertierung von und nach GML 111
 Text in einer GML-Datei 421
 GNU/Free Software Foundation 98
 Google Earth 8
 Google Maps 8
 GPS (Global Positioning System) 1, 150
 GPsBabel 159
 GPS-Positionsbestimmungen 395
 GPX 151
 ogr2ogr zum Import und Export von GPX-Dateien 151

GPX (GPS Exchange Format) 151, 157
 GRASS 122, 272
 Grassmuck, Volker
 Buch Freie Software zwischen Privat- und Gemeineigentum 7
 grep-Befehl (Unix) 100, 107
 GRS80 (Geodetic Reference System 1980) 396
 GTOPO, digitale Höhenmodelle 82
 gvSIG 122, 178, 272

H

häufige Fehler bei der Kartenerstellung 19
 head/tail-Befehl (Unix) 106
 Höhenmodelle 77
 HostGIS 53
 HTML-Client (MapServer) 197
 Ebenen-Auswahl 218
 Legende als separates Bild einfügen 223
 Maßstabsleiste 225
 Navigation 214
 Übersichtskarte 226
 Zoom-Funktionen bereitstellen 214
 HTML-Template (siehe HTML-Client)

I

IBM Spatial Extender 312
 iGeoPortal 273
 IHO S-57-Dateien 431
 Beispiel für die Angabe in der Mapdatei 433
 Beispiele für die Verwendung von ogrinfo 432
 besondere Anmerkungen 432
 Datenzugriff/Verbindungsmethode 431
 IMAGECOLOR 181
 IMAGEPATH 199
 IMAGETYPE 191
 IMAGEURL 199
 IMAGI-RP, Interministerieller Ausschuss für Geoinformation Rheinland-Pfalz 52
 Indizes zur Zugriffsoptimierung 341
 INSPIRE, Infrastructure for Spatial Information in Europe 82
 interaktive Karten 2, 9, 193–229
 Intergraph Microstation Design Files (DGN) 31
 Interoperabilität von Programmen verschiedener Hersteller 4

J

join-Befehl (Unix) 106
JPEG-Format 176
JUMP 53, 122, 178

K

ka-Map 229, 273, 373
Kanton Solothurn SO!GIS 81, 83
Kanton Thurgau 82, 83
Karten
 häufig wiederkehrende Aufgaben bei der
 Erstellung 18
 über die MapServer-Hilfsprogramme
 erstellen 168
Kartenbild 44
Kartendatei, Kommentare 173
Kartendienst 271
 standardisierter (siehe Web Map Service)
Kartenprojektionen 208, 377–408
 äquidistante zylindrische Projektionen 399
 Ausgleichung 385
 Beispiele 398–406
 Albers flächentreuer Kegelentwurf 403
 Mercatorprojektion 401
 orthografische Projektion 400
 stereografische Projektionen 404
 transverse Mercatorprojektion (TM) 402
 Bezugssystem 385, 386
 Datumspunkt 385
 Deutsches Hauptdreiecksnetz (DHDN) 386,
 394
 einfache zylindrische Projektionen 399
 einheitlicher Transformationsdatensatz 394
 Ellipsoidparameter 385
 ETRS89 394
 flächentreue (äquivalente) Abbildung 383
 Gauß-Krüger-Projektion 386
 geodätisches Datum 385
 Gravitationskonstante 386
 Interpolation 388
 kartesisches Koordinatensystem 387
 konische Projektionen 383
 Koordinatenreferenzsystem (Coordinate Re-
 ference System) 386
 Koordinatentransformationen 386
 Lambert-Projektion 386
 längentreue (äquidistante) Abbildung 382
Links und Bücher zum Thema 407

NTv2, National Transformation Version 2
 394

orthografische Projektionen 384
Referenzsystem 386
Rotationsgeschwindigkeit der Erde 386
Transformationen 387
Transformationsparametersatz 394
Universale Transversale Mercatorprojektion
 (UTM) 394, 404
Verebnete Koordinatensysteme/Projektionen
 382
Verwendung mit anderen Applikationen 406
winkeltreue (konforme) Abbildung 382
zylindrische Projektionen 383

Kartenserver 9

KML, Keyhole Markup Language 157

Knoppix 52, 53

Kommandozeilen-Werkzeuge 406

 Beispiele 169–190

 farbige Karten 180–183

 Karte mit Beschriftungen versehen 176

 Legenden erstellen 184–187

 Maßstabsangaben hinzufügen 187–189

 ogrinfo, zur Untersuchung eines

 Shapefiles mit den

 Landesgrenzen der Welt 170

 shp2img 174

 legend 168

 shp2pdf 168

 tile4ms 168

konische Projektionen 383

Konvertieren und Betrachten 27–42

Koordinatenreferenzsystem (Spatial Reference
 System, SRS) 170, 329

Kuba in der UTM 18N-Projektion 406

L

LABELITEM 177

Lamberts flächentreuer Kegelentwurf (Lambert
 Conformal Conic, LCC) 95, 209

Landesvermessungsamt Nordrhein-Westfalen
 83

layerObj-Objekt 360

Legende

 eingebettet, entfernen 223

 erzeugen 184–187

 erzeugen über das Hilfsprogramm legend 168

 hinzufügen zur Website 223

LEGEND-Objekt 184
 POSITION zur Positionierung in der Karte 186
 STATUS-Parameter ON, OFF oder EMBED 185
 legendObj-Objekt 360
 Linux Documentation Project-Website 107
 Live-CD WebGIS.rlp XV, 52
 Liyanaga, Marc 317
 Logische Operatoren 183
 = oder eq (ist gleich) 183
 > oder gt (größer als) 183
 >= oder ge (größer oder gleich) 183
 AND 184
 ne (ungleich) 183
 OR 184
 Luftbildinterpretation 151

M

Mapbender 11, 14, 234, 271, 281–309
 Administration 297–304
 Anwendung veröffentlichen 292, 306
 Beispielanwendungen 307–309
 Benutzer- und Gruppenverwaltung 285
 Beschreibung 281–284
 Demoversion zum Testen 283
 Digitalisieren mit Mapbender über WFS-T 283
 Dokumentation 293
 Einsatzgebiete und Funktionsumfang 283
 Geoportale 293, 308
 Grundlagen 284–297
 GUI (grafische Benutzeroberfläche) 287
 Kartendienst in GUI einbinden 289
 Oberflächenelemente 295
 PostgreSQL/PostGIS 313
 Stadtplandienste 307
 Übersicht über laufende Anwendungen 296
 Unterstützung von OGC WMS, WFS, WMC, SLD, KML 283
 WMS-Einstellungen anpassen 290
 WMS-Kartendienst laden 288
 WMS-Kartendienste verwalten 288
 MapBuilder 272
 Mapdatei
 # zur Erzeugung von Kommentaren 173
 .map-Dateiendung 174
 Beschriftung über Bitmap-Zeichensatz 182
 Beschriftung über TrueType-Zeichensatz 182

Einbinden von Daten aus PostgreSQL/PostGIS 354
 ESRI ArcInfo Coverage-Dateien 428
 ESRI ArcSDE-Datenbank (SDE) 429
 ESRI-Shapefiles (SHP) 415
 GML Zugriff über OGR 422
 IHO S-57-Dateien 433
 klassifizierte Karten erzeugen 180–182
 Links zu Dokumentationen 171
 MapInfo-Dateien (TAB/MID/MIF) 417
 MapServer, eingebettete Merkmale 436
 Microstation Design-Dateien (DGN) 431
 National Transfer-Format-Dateien (NTF) 439
 Objekte in der Mapdatei 171
 Parameter 174
 PostGIS/PostgreSQL-Datenbank 415
 Regeln und Empfehlungen 173
 shp2img zum Erzeugen von Karten 168
 Spatial Data Transfer Standard-Dateien (SDTS) 436
 temporäres Verzeichnis 195
 TIGER/Line-Dateien 427
 Übersicht 171
 Verwendung der OGR-Unterstützung 418
 VirtualSpatialData (ODBC/OVF) 424
 WFS, Web Feature Service 420
 MapInfo
 Dateien (TAB/MID/MIF) 416
 Beispiel für die Angabe in der Mapdatei 417
 Beispiele für ogrinfo 416
 Datenzugriff/Verbindungsmethode 416
 TAB-Dateien 31
 MapLab 53
 Demo 373
 MAP-Objekt 207
 mapObj-Objekt 360
 Mapping Hacks GIS RPMs für Fedora Core 53
 MapScript 193, 357, 375
 Anwendungen 373
 API, hierarchische Objektstruktur 360
 Beispiele 361–375
 Anlegen einer neuen Klasse, Auswahl eines Landes und dessen Darstellung in einer anderen Farbe 364
 einfache Weltkarte 362
 Java 374

- Karteneinstellungen ändern 364–366
- Mapdatei über MapScript erzeugen 366
- Map-Objekt manipulieren 364–366, 368
- Perl 374
- PHP MapScript 365, 368
- PHP MapScript als WMS-Server 371
- Ruby 375
- WMS-Layer als Datenquelle über PHP
 - MapScript erzeugen 369
- Binärversionen 359
- Dokumentation 359
- Download 358
- Einführung 357
- Hilfe bekommen 359
- insertclass()-Funktion 366
- Karte über Mapdatei erzeugen 363
- Klassendiagramme 360
- Objekte 359
 - classObj 360
 - colorObj 360
 - layerObj 360
 - legendObj 360
 - mapObj 360
 - scalebarObj 360
 - styleObj 360
 - webObj 360
- selbst kompilieren 358
- setExpression()-Funktion 365, 366
- mapserv (Executable) 191, 193
 - mapserv-Datei 68
- MapServer 43–70, 193–229, 259, 313
 - Abhängigkeiten bei der Installation 63–66
 - Bibliotheken für Bildformate 64
 - Bibliotheken zur Bildverarbeitung 64
 - Optionen für Webservices 64
 - Unterstützungsbibliothek für
 - Datenformate und Applikationen 64
 - Projektionsmodelle 64
 - zusätzliche Bibliotheken für
 - Grafikformate 64
 - AGG (Anti-Grain Geometry) 58
 - Anwendungen
 - Arbeitsweise 44
 - Diagramm der grundsätzlichen
 - Arbeitsschritte 44
 - Apache HTTP Webserver 194
 - Konfiguration 194
 - API 46
 - Ausgabekarte 48
 - CGI-Executable 45, 46
 - Datenquellen (siehe Geodaten)
 - Datenzugriff und Leistungsfähigkeit 31
 - Definition von Features (Inline Features) 436
 - Dokumentation 171
 - epsg-Datei enthält Projektionsinformationen
 - 210
 - EXTENT, Darstellungsbereich festlegen 204
 - Frameworks zur Applikationsentwicklung
 - 229
 - GD-Bibliothek 64
 - Hauptbestandteile 45–48
 - Hauptfunktion 44
 - Hilfsprogramme 168
 - Beispiele 169–192
 - Legende 184, 186
 - scalebar 168, 187–189
 - shp2img 168, 174
 - shp2pdf 168
 - shptree 168
 - sortshp 169
 - tile4ms 168
 - Hinzufügen einer WFS-Datenquelle 257
 - Installation 49–69
 - Debian/Ubuntu 52
 - DebianGIS 53
 - GIS-Distributionen für Linux 51
 - GIS-Knoppix 53
 - HostGIS 53
 - Linux Red Hat Package Manager (RPM)-
 - Dateien 53
 - Linux-Binaries 51
 - Mac OS X 54
 - Mapping Hacks GIS RPMs für Fedora
 - Core 53
 - mapserv-Datei 68
 - mehrere Versionen 68
 - Plattformen 49
 - Refractions Research 53
 - RPM-Pakete 68
 - Sammlungen von Debian Linux-Paketten
 - 53
 - Test der Installation 69
 - Windows-Binaries 50
 - Installations- und Konfigurierungsoptionen
 - 59–63
 - Anzeigen des Hilfetextes für die
 - Konfigurierung 59
 - DIR- und PATH-Argumente 63

- Eigenschaften und Pakete 59
 - include- und lib-Dateien 62
 - Konfigurierung für LibJPEG 62
 - Parameter 61
 - Unterstützung für JPEG-Bilder 62
 - IRC-Channel 70
 - Kommerzieller Support 70
 - Kompilierung aus dem Quellcode 54–68
 - Abhängigkeiten 63
 - configure 59
 - Erzeugung des Makefiles 59
 - Fehler 57
 - Kompilieren mit make 66
 - Konfiguration und Kompilierungsprozess 55–59, 61
 - nicht standardmäßig eingebundene Optionen 58
 - PROJ4-Dateien 57
 - Kompilierungsprozess 66–68
 - Linux mit Apache 194
 - Mailinglisten 70, 82
 - Mapdatei 171
 - klassifizierte Karten 180–183
 - Konfigurationsdatei 47
 - MapServer-Hilfsprogramme
 - legend 168
 - shp2img 178–179
 - MS4W (MapServer for Windows) 50, 194
 - OGC Webservices Workshop, Website 259
 - Performanz 31
 - Portabilität 32
 - PostGIS-Sichten 353
 - Projektion 204–212
 - Projektionen bei der Verwendung von WMS (Web Map Service) 242
 - Quellcode herunterladen 54
 - Referenz für Vektordaten 409–439
 - Ressourcen 69–70
 - RPM-Versionen 53
 - statische Karten erzeugen 167–192
 - Test mit einer Mapdatei 197–203
 - Testen der Installation 69
 - Tests und Fehlersuche 196
 - unterstützte Formate 413–439
 - unterstützte OGC-Standards 237
 - unterstützte Plattformen 49
 - unterstützte Vektordatenformate 413–439
 - Vektordaten einbinden 409–439
 - Vektordatenzugriff 410
 - Verbindungsparameter CONNECTION 350
 - Verwendung als eigenständiges Programm 167–192
 - Verwendung als WFS-Server 259
 - Verwendung mit Webdiensten 237–263
 - Visualisierung von PostGIS-Daten 350
 - Web Map Service (WMS) 239
 - Web-Mapping-Werkzeuge 23
 - Webserver 47
 - Website 50, 69
 - WMS-Client 247
 - WMS-Server 251
 - Mapserver (siehe Kartenserver)
 - MapServer-Features
 - Beispiel für die Angabe in der Mapdatei 436
 - Datenzugriff/Verbindungsmethode 436
 - MapStorer 52, 167
 - MapTools.org
 - MS4W 50
 - Website 50
 - Maßstab 74
 - geeigneter Maßstab einer Karte 73
 - Maßstabsleiste
 - INTERVALS 3 188
 - scalebar, Hilfsprogramm zum Erzeugen einer Maßstabsleiste 168, 189
 - SIZE x y 188
 - TRANSPARENT 189
 - UNITS KILOMETERS 188
 - Zweck 187
 - McKenna, Jeff 409
 - Mercatorprojektion 401
 - Metadaten 15
 - Microstation Design-Dateien (DGN) 411, 429
 - Beispiel für die Angabe in der Mapdatei 431
 - Beispiele für die Verwendung von ogrinfo 430
 - Datenzugriff/Verbindungsmethode 430
 - MINFEATURESIZE 177
 - Mobile Geographics 374
 - MySQL 411
- ## N
- NASA Topography 83
 - National Geophysical Data Center Interactive Map Services 238
 - National Oceanic and Atmospheric Administration (NOAA) 78

National Transfer Format-Dateien (NTF) 438
 Beispiel für die Angabe in der Mapdatei 439
 Beispiele für die Verwendung von ogrinfo 438
 Datenzugriff/Verbindungsmethode 438
Naturpark-Scout 157
nl-Befehl (Unix) 105
Nullmeridian 381

0

offene Standards und Interoperabilität 4
OGC (siehe Open Geospatial Consortium)
OGC Web Map Services 238
OGR 410
 Hilfsprogramme 38
 ogr2ogr 39
 ogrinfo 38
 Projekt-Website 37
 unterstützte Vektorformate 37
ogr2ogr 39, 178, 204, 333, 406
 Konvertierung von Shape in andere Formate 113
 Verwendung mit ogrinfo 178
 -where zum Extrahieren bestimmten Objekten 112
 -where-Option 178
 zur Ausgabe von Teilmengen 112
 zur Konvertierung nach GML 39
 zur Konvertierung von PostGIS-Daten 349
 zur Konvertierung von Shape in andere Formate 113
 zur Konvertierung von Shape nach GML 111
ogrinfo 38, 178, 410
 Auflisten der Daten in einer Shape-Datei 99
 Beispiele
 ESRI ArcInfo Coverage-Dateien 428
 ESRI Shapefiles (SHP) 414
 GML 421
 IHO S-57-Dateien 432
 MapInfo-Dateien (TAB/MID/MIF) 416
 Microstation Design-Dateien (DGN) 430
 National Transfer Format-Dateien (NTF) 438
 Spatial Data Transfer Standard-Dateien (SDTS) 434
 TIGER/Line-Dateien 425
 VirtualSpatialData (ODBC/OVF) 423
 detaillierte Informationen zu räumlichen Daten 89–93

 Filtern der Ausgaben von 171
 --help-Parameter 90
 -sql-Option 91
 -summary-Option 93
 zur Untersuchung der Struktur einer Ebene/Datei 414, 417
ogrtindex 168
Ohloh Open Source-Netzwerk 8
ominiverdi Live-CD 53
Open Geospatial Consortium (OGC) 24, 234
 OGC-Projektseite 313
 OGC-Referenzimplementierung 234
 OGC-Simple Features-Spezifikation 328
 OGC-Transactional Web Feature Server (WFS-T) 41
 OGC-Web Services (OWS) 25
 Simple Features Specification for SQL (SFS) 313
 Spezifikationen 418
 Well-Known Binary (WKB) 328
 Well-Known Text (WKT) 328
Open Source
 GIS-Anwendungen 27
 Initiative 8
 Software 6
Open Source Desktop-GIS 122
Open Source Geospatial Foundation (OSGeo) 5
 Kommerzieller Support 70
Open Source GIS/MapServer Anwendertreffen (siehe FOSS4G und FOSSGIS)
Open Source Software Image Map (OSSIM) Info Sheet 122
OpenEV 28, 122, 178, 272
 Projektseite 142
 zum Betrachten von PostGIS-Daten 349
OpenJUMP 272
OpenLayers 11, 14, 234, 271, 274–281
 Beispielanwendungen 274
 Client als iframe in Webseite einbinden 274
OpenMap 122
OpenStreetMap, freie Geodaten 82
Operatoren 183
Oracle Spatial-Datenbank 31, 312, 411, 417
 Beispiel-Kartendatei
 Verwendung mit MapServer-nativer Unterstützung 418
 Verwendung mit OGR-Unterstützung 418
Organisation of Oil and Gas Producers (OGP) 388

- orthografische Projektionen 384, 400, 401
- OSGeo 5
- OSGeo (siehe Open Source Geospatial Foundation)
- OSGeo Public Geospatial Data Project 82
- OSSIM 122
- OUTLINECOLOR 181
- OUTPUTFORMAT-Objekt 192
- ovf-Dateien 423, 424

P

- p.mapper 229, 373
- PAL.ADF-Datei 411
- PDF-Ausgabe über shp2pdf 168
- pgAdmin 319
- pgRouting
 - Einsatz im Naturpark-Scout 157
- PHP MapScript HOWTO 373
- PhpPgAdmin 319
- Planen eines Projektes 149
- Plate Carrée 399
- PortalU, Umweltportal Deutschland 15, 83, 308
- POSITION-Einstellung (Legende) 186
- PostGIS 31, 40, 53, 311, 347, 411
 - ArcMap Connector-(PgArc-)Projekt 356
 - Bestandteile 314
 - Betrachtung der Daten in OpenEV 349
 - Binaries 315
 - Darstellung der Daten mit MapServer 350
 - Debian Linux-Umgebung 317
 - Desktop-GIS-Unterstützung 314
 - Dokumentations-Website 315
 - Erstellung von Beschriftungen in MapServer 354
 - für Linux 316
 - für Windows 316
 - GIST, räumlicher Index 339
 - Installation unter Debian/Ubuntu 316
 - Installation von PostgreSQL und PostGIS 315
 - Installationspakete 315
 - lwpostgis.sql 315
 - SQL-Skript laden 323
 - Mac OS X-Umgebung 317
 - ogr2ogr
 - für den Export nach Shape, GML u. a. 349
 - Hilfsprogramm zur Datenkonvertierung 349
 - zum Laden von Daten 334
 - zur Konvertierung nach 114

- pgRouting 313
- pgsql2shp für den Export nach Shape 349
- PostGIS-Installer 316
- Quantum GIS zur Visualisierung der Daten 331
- Quellcode, Kompilierung aus 317
- räumliche Datenabfragen 336
- räumlicher Aufsatz für PostgreSQL 311
- RPM-Pakete 316
- serverbasierte Datenbanken 314
- shp2pgsql-Werkzeug 333
- Sichten (Views) in MapServer ansprechen 353
- spatial_ref_sys.sql 315, 324
- SRID, Spatial Reference Identifier 330
- Testen der Funktionalität 324
- Übersicht 311
- Unterstützung durch Desktop-GIS 313
- Vergleichsoperator für Bounding Boxes (&&) 353
- Website 311, 315
- Well-Known Binary (WKB) 335
- Well-Known Text (WKT) 343
- Zugriff auf kartografische Daten in anderen Applikationen 349
- PostGIS/PostgreSQL-Datenbank 415
 - Beispiel für Kartendatei-Syntax 415
- PostGIS-Funktionen 341–348
 - &&-Operator 344, 345
 - AddGeometryColumn(schema_name, table_name, column_name, srid, type, dimension) 329
 - ST_ (spatial type), Hinweis zum Präfix 330, 341
 - ST_AREA(geometry) 342
 - ST_AsEWKT(geometry) 330, 338
 - ST_AsText(geometry) 330, 338
 - ST_Buffer(geometry, double, [integer]) 354
 - ST_Contains(geometry, geometry) 41
 - ST_Distance(geometry, geometry) 342, 343
 - ST_Extent 341
 - ST_GeomFromText(text, [srid]) 330
 - ST_GeomFromText(text, integer) 343
 - ST_Length(geometry) 342
 - ST_Simplify(geometry, tolerance) 354
 - ST_TRANSFORM(geometry, integer) 342
 - ST_X(geometry) 342
 - ST_Y(geometry) 342

- PostgreSQL 40, 311
 - createdb, Hilfsprogramm zum Anlegen einer Datenbank 321
 - createlang, Hilfsprogramm zum Laden von Sprachen 322
 - Das Offizielle Handbuch von Peter Eisentraut 311
 - Datenabfragen in Sichten (Views) speichern 339
 - Einrichtung 319
 - Hilfe 311
 - INSERT, Einfügen von Daten 330
 - Literatur 311
 - offizielle Website für Installationspakete 316
 - pgAdmin, grafischer Datenbank-Client 319, 331
 - PhpPgAdmin, webbasierter Datenbank-Client 319
 - PL/pgSQL, prozedurale Sprache 322
 - psql, kommandozeilenbasierter Client 319
 - Schema 329
 - SELECT SQL, Anweisung zur Ausgabe von Informationen 330
 - Sichten (Views) 353
 - template1 320
 - Übersicht 313
 - Website 315
(siehe auch PostGIS)
 - professionelle und persönliche Karten im Vergleich 3
 - PROJ.4 392–395
 - cs2cs, Datumsübergänge berechnen 393
 - epsg-Datei 395
 - proj 392
 - proj -l, unterstützte Projektionsarten ausgeben 393
 - proj -le, Ellipsoidparameter ausgeben 393
 - Transformation über PROJ.4 394
 - PROJ.4-Bibliothek 209, 327
 - PROJECTION-Objekt 206
 - Projektionen 384
 - 3-D-Ansichten 141
 - OpenEV 141
 - OSSIM 141
 - 3-D-Perspektive 74–76
 - Ändern der Projektion 206–212
(siehe auch Kartenprojektionen)
 - Projektionsarten 384
 - Projektziele
 - Analyse 23
 - Betrachtung 23
 - Erstellung und Manipulation 24
 - Konvertierung 24
 - Weitergabe/Veröffentlichung 25
 - Prototyp Koordinatenreferenzsystem-Registry 408
 - psql, PostgreSQL-Kommandozeilenwerkzeug 320
 - \dt, Tabellen auflisten 328
 - \q, psql beenden 327
 - Python Cartographic Library (PCL) 374
 - Python MapScript (siehe MapScript) 359
- ## Q
- Quantum GIS (QGIS) 28, 41, 53, 122, 178, 272
 - Benutzeroberfläche 126
 - Digitalisieren mit QGIS 153
 - Editieren von Geometrien 154
 - Georeferenzieren von Rasterdaten 161
 - GPS-Daten importieren 157
 - GRASS-Plugin 152
 - Navigation 133
 - PostgreSQL/PostGIS-Daten laden 136
 - Punktdaten aus einer Textdatei laden 160
 - UMN Mapserver-Mapdatei exportieren 140
 - Vektorebene erzeugen 153
 - Visualisierung von Geodaten 128
 - WMS laden 137
 - Quelldaten (siehe Geodaten)
 - Querachsiger winkeltreuer Zylinderentwurf, Universal Transverse Mercator (UTM) 88
- ## R
- RADARSAT 93
 - Rasterdaten 28
 - Aufbau der wld/tfw-Datei 161
 - Farbkanäle 97
 - Festlegen des Ausgabeformats 190–192
 - gdalinfo (siehe GDAL)
 - Graustufenbild 97
 - im Vergleich mit Vektordaten 28, 71
 - Informationsausgabe über gdalinfo 93–97
 - mehrere Farbkanäle 97
 - PNG 191
 - Typen 73

- räumliche Datenbanken 40, 311–356, 412
 - Daten laden 332
 - Datenabfragen 336
 - IBM Spatial Extender (DB2) 312
 - Microsoft SQLServer 2008 312
 - MySQL 312
 - Oracle Spatial 312
 - PostGIS (siehe PostGIS)
 - PostgreSQL (siehe PostgreSQL)
 - Spatial Database Engine (SDE) 312
 - Übersicht 312
- räumliche Informationen und Karten 17
- räumlicher Index (GIST) 333, 339
- rechtwinkliges Koordinatensystem 380
- Refractions Research 53, 311
 - OGC Services Survey Website 238
 - Website 40
- Reguläre Ausdrücke 183
 - \$ (endet auf Zeichenfolge) 184
 - ./ (beliebiges Zeichen) 184
 - ^ (beginnt mit der Zeichenfolge) 184
- Relationale Datenbanksysteme (RDBMS) 312
- RESTful 273
- Rotationsellipsoid 379

S

- SAGA (System für Automatisierte Geowissenschaftliche Analysen) 122
- Scalable Vector Graphic (SVG) 41
- SCALEBAR-Objekt 188, 225
- scalebarObj-Objekt 360
- Schweizerischer Umweltdatenkatalog 83
- sed-Befehl (Unix) 106
 - zum Entfernen von Zeilen und Zeilenanfängen 102
 - zum Finden bestimmter Textmuster 101
 - zum Neuformatieren von Ausgaben 101
- Shape 411
 - Indizierung über shptree 168
 - Konvertierung nach GML 111
 - sortshp zum Sortieren der Datensätze 169
 - Untermengen hervorheben 352
- shp2img (MapServer-Hilfsprogramm) 168, 174, 186
 - Angabe unterschiedlicher Werte für EXTENT 178–179
 - Syntax 174
- shp2pgsql 115, 333

- shp-Präfix 169
- shptree (MapServer-Hilfsprogramm) 168
- Simplified Wrapper and Interface Generator (SWIG) 34, 358
- SO!GIS Kanton Solothurn 83
- SOAP 273
- sort-Befehl (Unix) 106
 - Erstellung einer nach Höhenangaben geordneten Liste 104
- sortshp (MapServer-Hilfsprogramm) 169
- Sourceforge-Website 98
- Spatial Data Transfer Standard-Dateien (SDTS) 434
 - Beispiel für Kartendatei-Syntax 436
 - Beispiele für die Verwendung von ogrinfo 434
 - Datenzugriff/Verbindungsmethode 434
- Spatial Information System (SIS) 356
- SRS (Spatial Reference System, Koordinatenreferenzsystem) 88
- SRTM, Topografische Daten der NASA 83
- Stadt Rostock, WebGIS 10
- Stadtplandienste 307
- statische Karten 8
- Stereografische Projektion 404
- Styled Layer Descriptors (SLD) 249
- STYLE-Objekt 182
- styleObj-Objekt 360
- Subversion (SVN) 55
- SWIG, Simplified Wrapper and Interface Generator 34, 358
- Swisstopo, Bundesamt für Landestopographie 81

T

- TAB-Dateien 416
 - (siehe auch MapInfo)
- TEMPLATE 199
- Terraview 53
- Texas Online Map Library 77
- Textverarbeitungswerkzeuge 97
 - Installation auf Nicht-GNU-Plattformen 98
- Thuban 53, 122
- TIGER-Dateiformat (US-Statistikbehörde) 31
 - TIGER/Line-Dateien 425
 - Beispiel für Kartendatei-Syntax 427
 - Beispiele für die Verwendung von ogrinfo 425
 - Datenzugriff/Verbindungsmethode 425

Tikiwiki-Website 229
Tim-Online (NRW) 83
Transformationen
 cs2cs 395
 einfache Transformationen 395
transverse Mercatorprojektion (TM) 402
TrueType-Fonts 182
tsort-Befehl (Unix) 106

U

U.S. Census TIGER (siehe TIGER-Dateiformat)
U.S. Geologic Survey 83
U.S. Maps and Data 83
U.S. National Atlas 83
U.S. National Geophysical Data Center
 (NGDC) 77
U.S. National Geospatial Data Clearinghouse 83
Übersichtskarte, Erweiterung einer Applikation
 um 226
Ubuntu, Paketsammlung 53
uDig 122, 178, 272
Umweltdatenkatalog Baden-Württemberg 83
Umweltdatenkatalog Niedersachsen 83
uniq-Befehl (Unix) 104–106
 -c-Option 105
UNITS, Parameter der Mapdatei zur Angabe der
 Einheiten 187
Unix-Befehle
 awk 101, 106
 colrm 107
 column 107
 cut 107
 expand/unexpand 107
 grep 100, 107
 head/tail 106
 join 106
 look 107
 nl 105
 paste 106
 sed 106
 zum Entfernen von Zeilen und
 Zeilenanfängen 102
 zum Finden bestimmter Textmuster 101
 zur Neuformatierung von Textausgaben
 101
 sort 104, 106
 tsort 106

uniq 104–106
 -c-Option 105
 wc (word count) 100, 107
User-friendly Desktop Internet GIS (uDig) 122

V

VACUUM ANALYSE 339
Vasudevan, Venu 231
Vektordaten 28
 Definition 409
 im Vergleich mit Rasterdaten 28, 71
 Referenzdokument 409–439
 Typen 72
 verzeichnisbasierte Daten 411
Vergleichsoperatoren (siehe Logische Operato-
 ren)
VirtualSpatialData (ODBC/OVF) 422
 Anlegen des DSN 422
 Beispiel für die Angabe in der Mapdatei 424
 Beispiele für die Verwendung von ogrinfo 423
 unterstützte Datenformate 422
 Zugriff auf einfache Textdateien 422
Visualisierungswerkzeuge 23

W

Warmerdam, Frank 359
 (siehe auch FWTools; GDAL/OGR)
WBMP 191
wc 100
wc-Befehl, word count (Unix) 100
Web Feature Service (WFS) 31, 238, 253–263,
 271, 420
 Beispiel für die Angabe in der Mapdatei 420
 Capabilities-Dokument 260
 Datenzugriff/Verbindungsmethode 419
 Fähigkeiten (Capabilities) des Anbieters 255
 FILTER 262
 GetCapabilities-Abfrage 418
 getFeature 256
 Mapbender als WFS-Client 283
 MapServer als Server 259
 MapServer WFS Howto 259
 Web Feature Service (WFS) 418
 WFS-Anfrage 420
Web Map Service (WMS) 10, 31, 238, 271
 Angabe der Projektion 242
 Capabilities, Aufbau 242

- Capabilities-Dokument 240
 - Desktop-GIS mit WMS-Unterstützung 271
 - EXCEPTIONS-Parameter 277
 - germany-Projekt 159
 - GetMap-Aufruf (Beispiel) 275
 - Hybride WebGIS-Clients 273
 - Interoperabilität 250
 - JavaScript-Clients mit WMS-Unterstützung 272
 - Karte manuell über einen GetMap-Request anfordern 244
 - kaskadierender WMS 247
 - Mapbender als WMS-Client 281–309
 - MapServer als WMS Server 251
 - OpenLayers als WMS-Client 274–281
 - REQUEST-Parameter 239
 - wget zum Anfordern des Capabilities-Dokuments 241
 - WMS-Clients 271
 - Web Services mit REST (O'Reilly) 274
 - webbasierte Karten 9, 14
 - Dienstanbieter 10
 - Server 13
 - Websites 10
 - webbasierte Kartografie 8
 - Webbrowser 247
 - Webdienste, Überblick 231
 - WebGIS.rlp Live-CD 52, 53
 - Web-Mapping 25
 - Web-Mapping-Werkzeuge 23
 - WEB-Objekt 199
 - webObj-Objekt 360
 - Webserver 14
 - Webservices 231–269
 - Verwendung mit MapServer 237–263
 - Zugriff auf Karten anhand von 231–269
 - Well-Known Binary (WKB) 335
 - Well-Known Text (WKT) 343
 - Weltkarte 399
 - Werkzeuge für die digitale Kartografie 22
 - Werkzeuge zur Textbearbeitung 106
 - WFS (siehe Web Feature Service)
 - WKT, Well-known Text 88
 - WMS (siehe Web Map Service)
 - World Geodetic System 1984 (WGS84) 396
- X**
- XML-basierte Datenformate 110
- Z**
- zylindrische Projektionen 383
 - GDAL/OGR
 - (siehe auch OGR, GDAL)

Über die Autoren

Tyler Mitchell ist Geograf, begeisterter Open Source-Verfechter und Executive Director der Open Source Geospatial Foundation (OSGeo). Er verfügt über mehr als zehn Jahre professionelle Erfahrung mit Geoinformationen und GIS im Naturressourcen-Management und in der Forstwirtschaft in West-Kanada.

Auf Open Source-Software ist er gestoßen, als er auf der Suche nach Werkzeugen war, die er bei seiner Arbeit mit Geoinformationen nutzen konnte. Auf seiner Suche fand er Open Source-Tools, die das Geodaten-Management und den Datentransfer dramatisch verbessern können.

Astrid Emde befasst sich schon seit einigen Jahren mit Geodaten, GIS und WebGIS und hierbei vor allem mit Lösungen im Open Source-Bereich. Seit Beginn des Mapbender-Projekts gehört sie zum Hauptentwicklerteam und arbeitet aktiv mit. Ebenso engagiert sie sich bei der Open Source Geospatial Foundation (OSGeo).

Als Mitarbeiterin der Firma WhereGroup kann sie ihre Begeisterung für die Ideen hinter Freier Software, der Open Source-Entwicklung und Standards einbringen, da in den dort umgesetzten Projekten eben diese Ideen Anwendung finden. Zusätzlich zu ihrer Tätigkeit als Projektleiterin und Entwicklerin hält Astrid auch Workshops auf nationalen und internationalen Veranstaltungen (FOSSGIS, FOSS4G, Agit) und gibt Schulungen zu Themen wie UMN MapServer, PostgreSQL/PostGIS, Mapbender und WMS/WFS.

Arnulf Christl kam Anfang der 90er zum ersten Mal mit der Geoinformationsverarbeitung in Kontakt. 1998 gründete er den GIS-Dienstleister CCGIS in Bonn, der seit 2003 Freie und Open Source-Software verwendet. 2007 fusionierte das Unternehmen in die WhereGroup GmbH & Co. KG, in der er weiterhin als Geschäftsführer mit technischem Schwerpunkt tätig ist. Über das OGC Principal Membership der WhereGroup ist er im Technical and Planning Committee des OGC aktiv und wurde in das OGC Architecture Board gewählt.

Er ist Gründungsmitglied der Open Source Geospatial Foundation (<http://www.osgeo.org>), die zum Ziel hat, Freie und Open Source-Software in der räumlichen Datenverarbeitung, freien Zugang zu Geodaten sowie Wissensverbreitung und Bildung zu fördern. Arnulf arbeitet als Systemarchitekt für Geodateninfrastrukturen und ist aktiver Entwickler der Software Mapbender, er hält regelmäßig Vorträge auf Konferenzen und Messen, am liebsten zum Thema Freie und Open Source-GIS-Softwarearchitekturen.

Über den Übersetzer

Jörgen W. Lang lebt als freier Übersetzer, Webdesigner, Programmierer, Dozent und Musiker im Elsass. Mit dem Web kam er 1996 zum ersten Mal in Kontakt. Im Laufe der folgenden Jahre eignete er sich im Selbststudium ein umfassendes Wissen rund um die Webprogrammierung und das Design von Webseiten an.

Seine Übersetzertätigkeit begann im Jahr 2000 mit dem Buch *CGI-Programmierung mit Perl* und umfasst mittlerweile ein Dutzend Titel rund um das Web und zu verwandten Themen. Neben seiner Arbeit als Übersetzer steht Jørgen auch als Dozent zu den Themen Webdesign und Webprogrammierung zur Verfügung. Während seiner übrigen Zeit widmet sich Jørgen meistens dem einen oder anderen Musikprojekt. Zu erreichen ist er unter der E-Mail-Adresse jwl@worldmusic.de.

Kolophon

Das Tier auf dem Cover von *Web-Mapping mit Open Source-GIS-Tools* ist eine Bekassine (*Gallinago gallinago*). Die Bekassine ist ein etwa taubengroßer, watender Schnepfenvogel, der durch seine kurzen Beine, die spitz zulaufenden Flügel und einen ausgesprochen langen Schnabel auffällt. Männchen und Weibchen haben einen stark gemusterten Rücken mit einigen markanten, sandfarbenen Längsstreifen, einen weißen Bauch und dunkelbraun gefleckte Flanken.

Wird die Bekassine aufgescheucht, flüchtet sie in einem steilen Zickzack-Flug. Dabei stößt sie einen kratzenden, heiseren Laut aus. Während der Balz im Frühjahr kann man schon von Weitem das »Wummern« der Männchen hören, ein Geräusch, das durch das Auseinanderstrecken der äußeren Schwanzfedern beim Sturzflug erzeugt wird. Der dabei entstehende tiefe, schwirrende Laut hat der Bekassine den Namen »Himmelsziege« eingebracht und soll die Weibchen anlocken.

Das Nest wird gut versteckt auf dem Boden zwischen Pflanzen in feuchten Wiesen und Marschländern angelegt. Ein weicher Untergrund und die Nähe zu kleinen Tümpeln sind dabei unerlässlich. Dort legen die Weibchen bis zu vier Eier. Die Jungtiere schlüpfen nach 19 bis 20 Tagen und laufen als Nestflüchter sofort den Eltern hinterher.

Noch im 19. Jahrhundert galt die Bekassine als »gemein«, das heißt als häufig vorkommend, doch die Jagd auf den wohlschmeckenden Vogel hat ihn mittlerweile so weit dezimiert, dass er heute auf der Roten Liste der vom Aussterben bedrohten Tiere steht. Die Entwässerung großer Flächen an den Küsten hat in den letzten Jahrzehnten den Rückgang der Populationen noch befördert.

Der Umschlagsentwurf dieses Buches basiert auf dem Reihenlayout von Edie Freedman und stammt von Ellie Volckhausen, die hierfür einen einzelnen alten Stich verwendet hat. Das Coverlayout der deutschen Ausgabe wurde von Michael Oreal mit InDesign CS3 unter Verwendung der Schriftart ITC Garamond von Adobe erstellt. Als Textschrift verwenden wir die Linotype Birka, die Überschriftenschrift ist die Adobe Myriad Condensed und die Nichtproportionalchrift für Codes ist LucasFont's TheSans Mono Condensed. Die in diesem Buch enthaltenen Abbildungen stammen von Robert Romano, Jessamyn Read, Lesley Borash und Michael Oreal und wurden mit Adobe Photoshop CS3 und Macromedia FreeHand MX erzeugt. Mary Anne Weeks Mayo und Geesche Kieckbusch haben das Kolophon geschrieben.

Webdesign

HTML mit CSS & XHTML von Kopf bis Fuß



Elisabeth Freeman & Eric Freeman
692 Seiten, 2006, 44,90 €
ISBN 978-3-89721-453-8

Sie wollen die Web-Grundtechnologien HTML und CSS erlernen, fürchten aber nüchtern-technische Lehrbücher? Dann ist HTML mit CSS und XHTML von Kopf bis Fuß genau das Richtige für Sie! Statt langweiliger Vorträge finden Sie hier viele

visuelle Überraschungen, witzige Comics, geistreiche Dialoge und unterhaltsame Beispiele. Und trotzdem vermittelt das Buch das technische Know-how, das man braucht, wenn man im Web-Umfeld arbeiten oder arbeiten möchte. Angefangen bei den Grundkenntnissen in HTML, XHTML und CSS über notwendige Infos zu Zeichensätzen und Farben bis hin zur Einhaltung der Webstandards werden alle wichtigen Themen behandelt – worauf warten Sie also noch?

HTML & XHTML, 3. Auflage



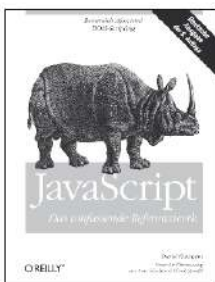
Chuck Musciano & Bill Kennedy
608 Seiten, 2007, 39,90 €, gebundene Ausgabe, ISBN 978-3-89721-494-1

Dieses Buch ist eine umfassende und komplette Übersicht über HTML 4.01 und XHTML 1.0, die über eine Einführung weit hinausgeht. Es behandelt detailliert jedes HTML-Element und erklärt die jeweilige Funktionsweise und das Zusammenspiel

mit anderen Elementen. Das Buch ist besonders geeignet für HTML-Autoren, die professionelle Webseiten erstellen und dazu das ganze Spektrum an Möglichkeiten mit HTML und XHTML nutzen wollen.

JavaScript –

Das umfassende Referenzwerk, 3. Auflage



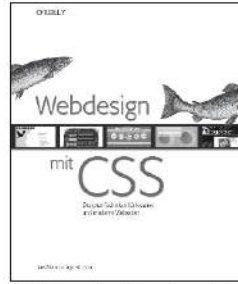
David Flanagan
1100 Seiten, 2007, 54,90 €, gebundene Ausgabe, ISBN 978-3-89721-491-0

JavaScript ist als Baustein des WWW mittlerweile fast so bekannt wie HTML. Für das Erstellen dynamischer Webseiten kommt man um die clientseitige Skriptsprache nicht mehr herum. Mit der Verbreitung von Ajax wird JavaScript inzwischen aber auch zur Programmierung

komplexer Anwendungen mit vielen Tausend Zeilen Code verwendet.

JavaScript – Das umfassende Referenzwerk wurde von Kultautor David Flanagan für diese Neuauflage komplett überarbeitet und um viele Kapitel ergänzt, die den modernen Einsatz von JavaScript und Ajax dokumentieren. Die dritte Auflage ist die deutsche Ausgabe der fünften englischen Auflage.

Webdesign mit CSS, 2. Auflage



Jens Meiert, Ingo Helmdach
264 Seiten, 2007, 29,90 €
komplett in Farbe
ISBN 978-3-89721-712-6

Standardkonformes Webdesign muss nicht zwangsläufig zu langweiligen Designs führen, sondern ermöglicht durchaus originelle Ideen. Dieses durchgehend vierfarbige Buch bietet Anregungen und Beispiele für

CSS-Standard-basierte Designs, die zur eigenen Inspiration studiert werden können, aber auch sofort angewendet und für die eigenen Bedürfnisse angepasst werden können. Das Buch wendet sich an Webdesigner, Autoren und Entwickler, die sich mit den XHTML- und CSS-Standards prinzipiell auskennen und nun auf der „Design-Leiter“ ein paar Stufen höher klettern wollen. Die neue Auflage wurde komplett überarbeitet sowie um neue Designs erweitert. Die Beispieldateien aller 30 Designs können heruntergeladen und frei verwendet werden.

Cascading Style Sheets –

Das umfassende Handbuch, 2. Auflage



Eric A. Meyer
576 Seiten, 2007, 39,90 €, gebundene Ausgabe, ISBN 978-3-89721-493-4

Eric Meyer, eine international anerkannte Größe in der CSS-Gemeinde, behandelt in diesem Buch ausführlich die aktuellen Standards CSS2 und CSS2.1. Detailliert beschreibt er alle CSS-Eigenschaften, erläutert, wie sie mit anderen Eigenschaften

zusammenwirken und zeigt anhand vieler praktischer Beispiele, wie sie angewendet werden. Durch die Referenzen im Anhang zu allen CSS-Eigenschaften und Selektoren kann dieses Buch auch sehr gut als Nachschlagewerk verwendet werden.

CSS Kochbuch, 2. Auflage



Christopher Schmitt
544 Seiten, 2007, 39,90 €, gebundene Ausgabe, ISBN 978-3-89721-492-7

Aktualisiert und deutlich erweitert, wartet die zweite Auflage des *CSS Kochbuchs* mit dem bewährten Mix aus schnellen, praxisorientierten Lösungen und begleitenden Diskussionen auf. In über 150 Rezepten behandelt Autor Christopher Schmitt gängige bis vertrackte

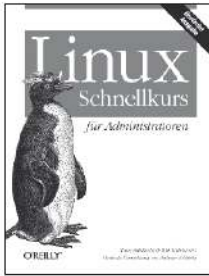
CSS-Aufgaben, denen sich Webdesigner immer wieder gegenüber sehen. Den passenden Code dazu gibt es zum Herunterladen – CSS für alle Lebenslagen, fertig zum sofortigen Einsatz auf Ihrer Website.

O'REILLY®

anfragen@oreilly.de • <http://www.oreilly.de> • +49 (0)221-97 31 60-0

Linux

Linux-Schnellkurs für Administratoren

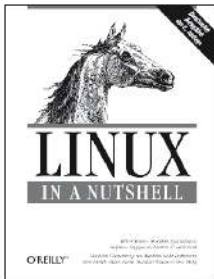


Tom Adelstein, Bill Lubanovic
352 Seiten, 2007, 39,90 €
ISBN 978-3-89721-722-5

Solide Kenntnisse der Linux-Systemadministration sind gefragt denn je. Egal, ob Sie ein Unix-Veteran, ein erfahrener MCSE oder ein fortgeschrittener Linux-Anwender sind – größere Linux-Systeme aufbauen und verwalten zu können, ist ein Pfund, mit dem sich wuchern lässt.

Dieser *Linux-Schnellkurs für Administratoren* fasst die Schritte zusammen, die nötig sind, um die verschiedensten Systeme aufbauen zu können, angefangen von alleinstehenden SOHO-Rechnern, Web- und LAN-Servern bis hin zu lastverteilten Clustern und Servern, die per Virtualisierung konsolidiert wurden. Sie werden hier auch alles Notwendige über die Werkzeuge lernen, die Sie für die Einrichtung und Pflege dieser produktiven Systemumgebungen benötigen.

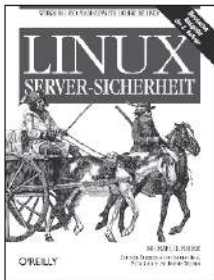
Linux in a Nutshell, 4. Auflage



Ellen Siever, Stephen Figgins & Aaron Weber
956 Seiten, 2005, 40,- €
ISBN 978-3-89721-426-2

Linux in a Nutshell ist das Nachschlagewerk für Benutzer, Programmierer und Systemadministratoren und beschreibt alle wichtigen Linux-Befehle, Editoren und andere Werkzeuge.

Linux Server-Sicherheit, 2. Auflage

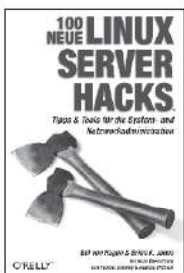


Michael D. Bauer
608 Seiten, 2005, 44,- €
ISBN 978-3-89721-413-2

Linux Server-Sicherheit erklärt die Prinzipien verlässlicher Systeme und der Netzwerksicherheit, zeigt typische Risiken auf, gibt praktische Tipps für gängige Sicherungsaufgaben und kombiniert das mit profunden und detaillierten Beschreibungen der technischen Werkzeuge, die für einen

bestimmten Einsatz empfohlen werden.

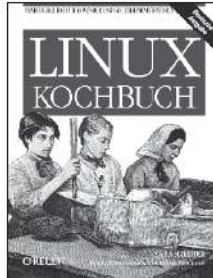
100 neue Linux Server Hacks



Bill von Hagen & Brian K. Jones
552 Seiten, 2006, 34,90 €
ISBN 978-3-89721-461-3

Endlich 100 neue Linux Server Hacks! Der erste Band mit Hacks hat nichts von seiner Nützlichkeit verloren, aber es gibt noch so viele weitere Tipps, Tricks und Werkzeuge, mit denen Sie Ihren Alltag als Sysadmin effizienter gestalten können. Lösungen zur Sicherung Ihres Systems, zur Benutzerverwaltung, zum Speichermanagement, zum Monitoring – hier zeigen Ihnen Experten, wie's geht.

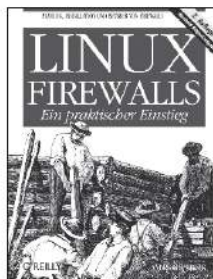
Linux Kochbuch



Carla Schroder
618 Seiten, 2005, 44,- €
ISBN 978-3-89721-405-7

Im *Linux Kochbuch* finden Sie die hochkonzentrierte Erfahrung vieler Jahre Linux-Administration in Form praktischer Rezepte. Jede Lösung dieser Sammlung lässt sich direkt auf typische Alltagsprobleme anwenden und spart Stunden mühevollen Ausprobierens. Themen sind unter anderem das Installieren von Software, die Benutzerverwaltung, das Dateimanagement, die Anpassung des Kernels, Boot-Loader und Multi-Boot, Systemreparaturen mit Knoppix, Drucken mit CUPS, Backups und Systemsicherung, Versionskontrolle, Postfix, Spambekämpfung, Apache, Samba und vieles mehr.

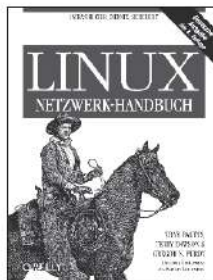
Linux-Firewalls – Ein praktischer Einstieg, 2. Aufl.



Andreas G. Lessig
656 Seiten, 2006, 42,- €
ISBN 978-3-89721-446-0

Das Thema Sicherheit ist aktueller denn je und kein verantwortungsbewusster Linux-User oder Systemadministrator kommt an der Konfiguration einer Firewall vorbei. Der praxisorientierte Einstieg stellt typische Firewall-Lösungen vor und dokumentiert sorgfältig, wie sie Schritt für Schritt zu realisieren sind: von unkomplizierten Floppy-Firewalls für den Privathaushalt bis zu anspruchsvolleren Konzepten für Unternehmen, die eine sicher konfigurierte Standarddistribution, Paketfilterung und Proxies kombinieren.

Linux Netzwerk-Handbuch, 3. Auflage



Tony Bautts, Terry Dawson & Gregor N. Purdy
382 Seiten, 2005, 38,- €
ISBN 978-3-89721-414-9

Dieses Buch bietet Ihnen das nötige Rüstzeug, das Sie für die Einrichtung und Administration von Linux-Netzwerken brauchen. Beginnend bei grundlegenden Informationen, widmen sich die Autoren rasch zentralen Themen wie TCP/IP und PPP, Firewalls, Masquerading, Accounting und E-Mail. Völlig neu in dieser Auflage sind die Kapitel zu IMAP, Samba, OpenLDAP, Wireless Networking und IPv6.

Linux – kurz & gut



Daniel J. Barrett, 204 Seiten, 2004, 9,90 €
ISBN 978-3-89721-501-6

Eine praktische, anwenderorientierte Kurzreferenz, die auf engstem Raum alle wichtigen Konzepte, Befehle und Optionen vorstellt. Die Referenz ist auf SUSE 9 zugeschnitten, lässt sich aber im Wesentlichen auf alle gängigen Linux-Distributionen übertragen.

O'REILLY®

anfragen@oreilly.de • <http://www.oreilly.de> • +49 (0)221-97 31 60-0