http://intevation.de

Einführung in den UMN MapServer

Dokumentation

der
Intevation GmbH

Ansprechpartner:
Silke Reimer <silke.reimer@intevation.de>
Intevation GmbH, Georgstraße 4, 49074 Osnabrück
Tel: 0541/33508-32, Fax: 0541/33508-59

Datum: 7. Juli 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Allgemeiner Aufbau

Beim Erstellen einer Webmapping-Anwendung gibt es verschiedenen Aufgaben, die berücksichtigt werden müssen: Zunächst müssen die unterschiedlichen Datenquellen eingebunden werden. Für jede dieser Datenquellen sind Darstellungsvorschriften zu definieren, die regeln, wie einzelne Datensätze u.U. verbunden mit einer über Attributwerte definierten Klassifikation farblich und vom Muster her gestaltet werden sollen. Die daraus resultierenden Karten werden über das Intra- oder Internet an den Klienten geliefert, der in der Regel eine HTML-Seite u.U. angereichert mit JavaScript oder anderen Technologien zum Erstellen von Webanwendungen (Flash etc.) darstellt.

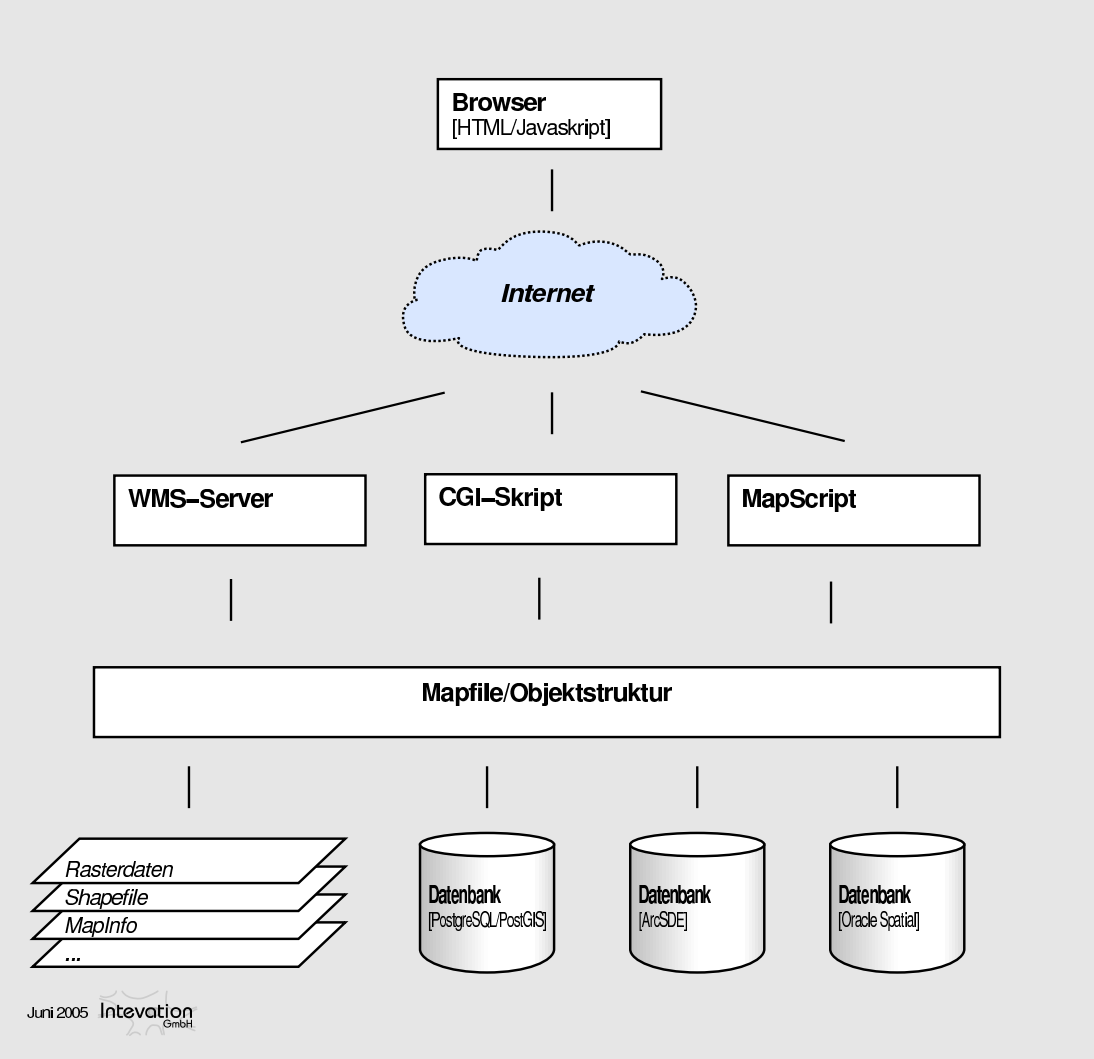

Abbildung 1: Überblick über die Einbettung der UMN MapServers in einfache und komplexe WebGIS-Anwendungen

Die Abbildung gibt einen Überblick über die Mechanismen, mit denen der UMN MapServer diese Aufgaben erfüllt: Die Datenanbindung erfolgt über das Mapfile oder – im Falle einer MapScript-Variante – durch die Definition einer Objektstruktur. Diese Objektstruktur entspricht dem Aufbau des Mapfiles und kann auch durch das Einlesen eines Mapfiles erzeugt werden. Alternativ kann diese Objektstruktur aber auch durch ein Skript aufgebaut werden (siehe Abschnitt 6). Die Darstellungsvorschriften sind ebenfalls Bestandteil der Mapfile-Konfiguration (siehe Abschnitt 4). U.U. sollten die Daten aufbereitet werden, um einen performanten Zugriff auch auf große Datenmengen zu gewährleisten. Abschnitt 3 beschäftigt sich mit diesem Thema.

Das Ausliefern der Karten kann beim UMN MapServer über verschiedenen Mechanismen erfolgen. Als einfaches CGI-Skript liefert kennt der UMN MapServer verschiedenen Darstellungsmodi, die steuern, was der UMN MapServer nach außen liefern soll (mode=map liefert etwa eine statische Karte). Durch die Übergabe von weiteren Parameter können verschiedene Einstellungen vorgenommen werden. So lässt sich z.B. der Kartenausschnitt über den Parameter extent steuern. Die Parameter und Art und Weise, wie der UMN MapServer diese Parameter auswertet, sind UMN MapServer spezifisch. Alternativ dazu gibt es eine vom Open Geospatial Consortium (OGC) entwickelte Schnittstelle, die solche Abfragen und Einstellungen an einen Kartenserver normiert. Diese Spezifikation heißt WMS (Web Map Service). seit der Version 4.x des UMN MapServer wird auch diese Schnittstelle zur Lieferung von Karten unterstützt. Die dritte Möglichkeit zur Auslieferung der Karten stellt schließlich der Script-Mechanismus des UMN MapServer dar. Es werden verschiedenen Skriptsprachen unterstützt darunter PHP, Python und Perl (mehr dazu unter 6.

Es ist eigentlich nicht originäre Aufgabe eines Kartenservers, auch die Klienten-Seite einer Webmapping-Anwendung abzudecken, also das Erstellen von HTML-Seiten, mit denen man auf den Kartenserver zugreifen kann. Der UMN MapServer bietet jedoch auch hier verschiedenen Möglichkeiten an, diese zu erstellen. Über so genannte Templates (engl. Vorlagen) werden HTML-Seiten definiert, die der UMN MapServer beim Aufruf mit den dynamisch erzeugten Inhalten (Link zur Karte, Angaben über Kartenausschnitt, Abfrageergebnisse etc.) füllt (siehe Abschnitt 5). Beim Einsatz von MapScript werden die dynamischen Inhalte durch Einsatz der gewählten Programmiersprache an die richtige Stelle gesetzt (siehe Abschnitt 6).

Es gibt bereits eine Reihe von guter Dokumentation zum UMN MapServer – und zwar sowohl auf englisch als auch auf deutsch. Das vorliegenden Dokument will die in diesem Abschnitt genannten Aspekte ein wenig vertiefen, wobei in der Regel auf geeigneten Abschnitte der vorhandenen Dokumentation verwiesen wird. An einigen Stellen wird diese Dokumentation durch eigenen Bemerkungen ergänzt. Das gilt insbesondere für die Installation unter Windows (Abschnitt 2) wie die Datenaufbereitung (Abschnitt 3).

2 Installation unter Windows

Entwicklungswerkzeuge sind auf einer Windows-Installation meistens nicht vorhanden. Daher werden für eine Installation des UMN MapServers unter Windows meistens vorkompilierte Pakete benötigt. DM Solutions stellen vorkompilierte Windows-Binaries auf ihrer Homepage bereit[1]. Diese Pakete enthalten nicht nur den UMN MapServer als CGI-Skript, sondern auch die PHP MapScript Variante des UMN MapServers.

Im Folgenden wird die Installation eines Paketes von DM Solutions als CGI sowie als PHP MapScript unter Verwendung eines Apache als Webserver beschrieben. Wichtig zu wissen ist hierbei folgendes: Der UMN MapServer benötigt eine Reihe von Bibliotheken die u.a. die Anbindung an verschiedenen Datenformate sowie Projektionssupport u.ä. bieten. Soll eine Datenquelle eingebunden werden, die das vorliegenden Paket nicht unterstützt (etwa Oracle Spatial), dann muss ein anderes Binärpaket genutzt werden. Hier hilft entweder das Selbst-Kompilieren oder das Suchen im Internet. Wenn ein solches Paket nicht vorhanden ist, muss u.U. eine Firma beauftragt werden, das Paket zu bauen.

Zentrale Datei zur Konfiguration des Apache ist die Datei httpd.conf. Sie liegt in der Regel unter `Pfad/zum/Apache/conf/httpd.conf`. Sollten Sie sie dort nicht finden, suchen Sie bitte nach httpd.conf. Im Laufe diese Abschnitts wird einige Male auf diese Datei verwiesen, wobei der Pfad zu dieser Datei jedoch weggelassen wird.

2.1 UMN MapServer als CGI

Die Installation des UMN MapServers als CGI-Skript erfolgt in folgenden Schritten:

1. Anlegen eines neuen Systemverzeichnisses. Legen Sie das Unterverzeichnis „mapserv“ in Ihrem Systemverzeichnis (meistens `C:\Windows\system32` o.ä.) an. Öffnen Sie dann die Systemsteuerung und dort den Unterpunkt „System“. Wählen Sie den Tabulator „Erweitert“ und dort „Umgebungsvariablen“. Suchen Sie die Systemvariable „Path“, wählen sie „Bearbeiten“ und fügen Sie – mit einem Semikolon getrennt – das soeben erzeugte Verzeichnis hinzu. Nun bestätigen sie alle Dialoge mit „OK“ und starten Sie Windows neu.

2. Installation der benötigten Bibliotheken. Die vom UMN MapServer benötigten Bibliotheken befinden sich in den Zip-Archiven des von DM Solutions ausgelieferten Zip-Archiv. Öffnen Sie die einzelnen Archive und kopieren Sie die jeweiligen DLLs in das neu erzeugte Systemverzeichnis.

3. Installation des UMN MapServer Binaries: Kopieren Sie die Datei `mapserv.exe` in das CGI-Verzeichnis des Apache. In der Regel liegt diese unter

[1] `http://maptools.org/php_mapscript/index.phtml?page=downloads.html`

`Pfad/zum/Apache/cgi-bin`. Auskunft darüber gibt die Datei httpd.conf. Dort sollte sich ein Abschnitt der folgenden Art befinden:

```
ScriptAlias /cgi-bin/ C:/Apache/cgi-bin/

#
# "/usr/lib/cgi-bin" could be changed to whatever
# your ScriptAliased CGI directory exists, if you
# have that configured.
#
<Directory C:/Apache/cgi-bin/>
    AllowOverride None
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>
```

Dabei entspricht `C:/Apache/cgi-bin` dem Pfad zum CGI-Verzeichnis. Wichtig ist außerdem die Option ExecCGI. Sie regelt, dass die in diesem Verzeichnis liegenden Programme als CGI-Skript vom Apache ausgeführt werden. Dorthin also muss das Programm `mapserv.exe` kopiert werden.

4. Testen der Installation. Rufen Sie im Browser folgende URL auf `http://localhost/cgi-bin/mapserv.exe`
   Es sollte folgende Antwort erscheinen:

   ```
   No query information to decode. QUERY_STRING is set, but empty.
   ```

5. Sollte der vorangegangene Punkt nicht erfolgreich gewesen sein, hilft unter Umständen folgende Analyse weiter: Öffnen einer DOS-Shell und Wechsel in das CGI-Verzeichnis des Apache. Dort wird folgendes Kommando aufgerufen `mapserv.exe -v`. U.U. erscheint nun eine Fehlermeldung, dass eine bestimmte DLL nicht gefunden wird. Diese sollte dann in das oben angelegt Systemverzeichnis nachinstalliert werden. Geht alles glatt, erhält man hiermit übrigens zusätzlich Informationen über die Fähigkeiten des UMN MapServers in der vorliegenden Installation, also z.B.

   ```
   MapServer version 4.4.2 OUTPUT=GIF OUTPUT=PNG
   OUTPUT=JPEG OUTPUT=WBMP SUPPORTS=PROJ
   SUPPORTS=FREETYPE SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT
   SUPPORTS=WFS_CLIENT INPUT=EPPL7 INPUT=JPEG INPUT=OGR
   INPUT=GDAL INPUT=SHAPEFILE
   ```

2.2 PHP MapScript

Die Installation von PHP MapScript erfordert zunächst die gleichen Schritte wie im vorangegangenen Abschnitt beschrieben, d.h. das Systemverzeichnis muss ebenfalls angelegt und die benötigten Bibliotheken dort abgelegt werden. Die darüber hinaus notwendigen Schritte sind zum einen die Installation von PHP und die Konfiguration des Apache zur Einbindung von PHP sowie die Installation des UMN MapServer als PHP-Erweiterung:

1. Installation von PHP: Oft wird mit dem Apache PHP schon mitgeliefert. Benötigt wird PHP als CGI-Installation. Wenn PHP nicht mitgeliefert wird, muss es selber noch installiert werden. Es ist als Version 4.3.11 auf der CD enthalten, kann aber auch unter `http://www.php.net/` herunter geladen werden. Danach muss es in beliebiges Verzeichnis ausgepackt werden, z.B. unter /Pfad/zum/Apache/PHP-4.3.11-Win32. Im Folgenden wird auf diese Verzeichnis als C:/PHP verwiesen, um die Pfad-Angaben nicht zu lang zu machen. Die eigentliche Installation von PHP ist damit abgeschlossen. Als nächstes erfolgt die Installation des Apache Webservers.

2. PHP-Konfiguration des Apaches: Die Konfigurationen erfolgen in der Konfigurationsdatei des Apache httpd.conf. PHP muss als CGI angesprochen werden. Dazu sind folgenden Konfigurationsangaben notwendig:

   ```
   ScriptAlias /php/ C:/php/
   AddType application/x-httpd-php .php
   AddType application/x-httpd-php .phtml

   Action application/x-httpd-php /php/php.exe
   ```

   ScriptAlias verknüpft das Installationsverzeichnis von PHP mit dem Apache-internen Verzeichnis php, auf das in der Folge referenziert werden kann. Mit AddType wird ein neuer Mime-Type, nämlich application/x-httpd-php definiert: Jede Datei, die mit php oder phtml endet, wird als application/x-httpd-php definiert. Mit Action wird festgelegt, was mit diesem Typ passieren soll, nämlich die Ausführung dieser Datei als PHP-Skript (Ausführung mit /php/php.exe). An dieser Stelle wird auf das Apache-interne Verzeichnis /php zugegriffen und das dort enthaltene Programm php.exe aufgerufen.

   Evtl. hat die Apache-Konfiguration php bereits als Modul eingebunden. Entsprechende Vorgaben müssen auskommentiert werden, damit PHP tatsächlich als CGI ausgeführt wird. In der httpd.conf muss daher die Zeile, die das Modul lädt auskommentiert werden (die Pfad-Angaben zum Modul können variieren. Es empfiehlt sich daher eine Suche nach LoadModule)

```
#LoadModule php4_module C:/Pfad/zum/Apache/php.dll
```

3. Überprüfung der Installation: Legen Sie in der DocumentRoot (s.u.) des Webservers die Datei phpinfo.phtml folgenden Inhalts an:

   ```
   <?php
   phpinfo();
   ?>
   ```

   Die DocumentRoot ist in der Regel das Verzeichnis Pfad/zum/Apache/htdocs. Gesteuert wird dies durch den Parameter DocumentRoot in der httpd.conf. Häufig ist dies unter Pfad/zum/Apache/htdocs.

   Der Aufruf von `http://localhost/phpinfo.phtml` im Browser sollte nun eine Übersicht der Konfiguration von PHP geben. Im ersten Absatz sollte dabei ein Hinweis auftauchen, dass PHP als CGI eingebunden ist („Server API: CGI“).

4. Festlegen des Extension-Verzeichnisses: Das Extension-Verzeichnis ist das Verzeichnis, in dem nach Erweiterungen gesucht wird, die dynamisch hinzu geladen wird, so wie es mit dem php_mapscript-Modul geschehen soll. Wo dieses Verzeichnis liegt, wird über die PHP-Konfigurationsdatei php.ini gesteuert, sofern diese existiert. Sie sollte im PHP-Verzeichnis (also hier C:/PHP) liegen. Evtl. müssen Sie dazu eine PHP.ini-Vorlage dorthin kopieren. Die PHP-Version auf der CD enthält zwei php.ini-Dateien. Kopieren Sie die Datei php.ini.recommended nach php.ini. Rufen Sie `http://localhost/phpinfo.phtml` erneut auf. Sie sollten jetzt im ersten Abschnitt u.a. folgende Angaben lesen: „Configuration File (php.ini) Path: C:/PHP/php.ini„. Öffnen Sie nun die Datei C:/PHP/php.ini und suchen Sie nach „extension_dir“. Tragen Sie hier das Verzeichnis ein, in dem nach Extensions gesucht werden soll, also z.B. „extensions“:

   ```
   extension_dir = "./Extensions"
   ```

5. Installation von php_mapscript: Kopieren Sie die Datei php_mapscript.dll aus dem Binär-Paket von DM Solutions nach C:/PHP/Extensions. Erweitern Sie phpinfo.phtml wie folgt:

   ```
   <?php
   dl("php_mapscript.dll");
   phpinfo();
   ?>
   ```

Rufen Sie nun `http://localhost/phpinfo.phtml` erneut auf. Auf der Seite sollte nun ein Abschnitt der folgenden Art auftauchen:

```
MapServer Versions: MapServer version 4.6.0 OUTPUT=GIF
                    OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP
                    OUTPUT=SVG SUPPORTS=PROJ
                    SUPPORTS=FREETYPE SUPPORTS=WMS_SERVER
                    INPUT=EPPL7 INPUT=GDAL INPUT=SHAPEFILE
PHP MapScript Version:
                    ($Revision: 1.235 $ $Date: 2005/06/14 16:03:35 $)
```

Die Angaben zu MapServer Versions geben an, welche Funktionalitäten der UMN MapServer hat. Es sind die gleichen Angaben, die man auch über `mapserv.exe -v` auf der DOS-Shell erhält (s.o.).

6. Fehleranalyse: Tauche die obige Angabe noch nicht auf, dann hilft es u.U. die Fehlerausgaben auf die HTML-Seite zu aktivieren. Öffnen Sie dazu die Datei php.ini und setzen Sie den Parameter display_errors auf On:

   ```
   display_errors On
   ```

   Oben auf der Ausgabe von phpinfo.phtml sollten jetzt Fehlermeldungen erscheinen, die u.U. einen Hinweis darauf geben, was beim Einbinden von PHP MapScript nicht funktioniert hat.

3 Datenaufbereitung

Ziel der Datenaufbereitung vor der Einbindung der Daten in das Mapfile ist es, auch große Datenmengen effizient ansprechen zu können. Dazu gibt es folgende Mechanismen:

- Kachelung von großen Rasterdaten. Diese Kachelung sorgt dafür, dass nicht jedes mal die gesamte Rasterdatei geladen werden muss, sondern u.U. nur ein kleiner Ausschnitt. Das dazu benötigte Tool ist *gdal_translate*.
- Zusammenfassung von vielen Raster- oder Shapedateien in Bildkataloge. Raster- oder Vektordaten gleicher Thematik, die auf verschiedene Kacheln aufgeteilt sind, können damit zu einem Shapefile zusammengefasst werden, so dass nur ein Layer im Mapfile angelegt werden muss (s.u.). Die dafür verwendeten Tools sind *gdaltindex* (Raster) und *tile4ms* (Vektor). Ein Bildkatalog enthält Shapes in Form eines Rechtecks, dass den räumlichen Extent der zugehörigen Datei abbildet. Die zugehörige DBF-Tabelle enthält ein Attribut, dessen Wert den Pfad zu der zugehörigen Datei enthält.

Die genannten Tools sind teilweise aus der Entwicklung des UMN MapServers entstanden, teilweise gehören sie in den Bereich der Rasterdatenverarbeitung und werden daher von der Rasterdatenbibliothek gdal verwaltet. Genauere Informationen über die Tools findet man unter `http://www.umn-mapserver.de/doc42/utils.html` und `http://www.gdal.org/gdal_utilities.html`. Im Folgenden werden die genannten Tools anhand von Beispielen erläutert. Grundlage dafür ist die Datei material/frida/data/satelit-os.tif auf der CD.

3.1 Kachelung

3.1.1 gdalinfo

Mit gdalinfo lassen sich Informationen zu einer Rasterdatei anzeigen:

```
gdalinfo.exe satelit-os.tif
Driver: GTiff/GeoTIFF
Size is 1000, 500
Coordinate System is ''
Origin = (6.999000,53.001000)
Pixel Size = (0.002000,-0.002000)
Metadata:
  TIFFTAG_DOCUMENTNAME=satelit-os.tif
  TIFFTAG_SOFTWARE=ImageMagick 6.0.6 09/26/04 Q16 http://www.imagemagick.org
Corner Coordinates:
Upper Left  (   6.9990000,  53.0010000)
Lower Left  (   6.9990000,  52.0010000)
Upper Right (   8.9990000,  53.0010000)
Lower Right (   8.9990000,  52.0010000)
Center      (   7.9990000,  52.5010000)
Band 1 Block=1000x500 Type=Byte, ColorInterp=Red
  Overviews: 500x250, 250x125, 125x63
Band 2 Block=1000x500 Type=Byte, ColorInterp=Green
  Overviews: 500x250, 250x125, 125x63
Band 3 Block=1000x500 Type=Byte, ColorInterp=Blue
  Overviews: 500x250, 250x125, 125x63
```

Man erkennt, dass die Datei einen Ausschnitt von ca (7, 52) bis (8, 53) hat. Man beachte, dass der Bildursprung (Origin) (7, 53) ist, d.h. er entspricht der linken *oberen* Ecke und nicht wie sonst in Koordinatensystem gewohnt der linken *unteren*. Dies gilt es bei der weiteren Bearbeitung des Bildes zu beachten.

3.1.2 gdal_translate

Im folgenden soll die Datei satelit-os.tif in 4 Kacheln zerlegt werden, die wie folgt aufgebaut sind

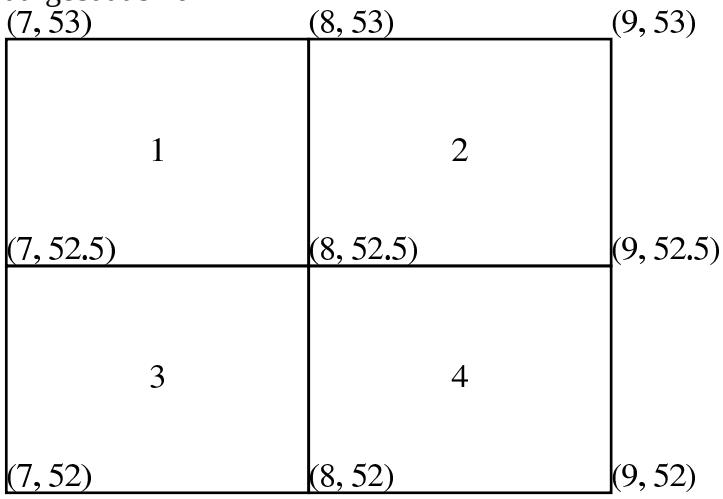

Abbildung 2: Aufbau der Kacheln für satelit-os.tif

Mit gdal_translate können diese Kacheln erzeugt werden. Dazu wird über den Parameter -projwin die BoundingBox der resultierenden Kachel angegeben. Die Übergabeparameter sind:

ulx: Upperleft X-Coordinate (Obere linke X-Koordinate)
uly: Upperleft Y-Coordinate (Obere linke Y-Koordinate)
lrx: Lowerright X-Coordinate (Untere rechte X-Koordinate)
lry: Lowerright Y-Coordinate (Untere rechte Y-Koordinate)

Die Kacheln werden nun in der Reihenfolge 1,2,3,4 wie folgt erzeugt:

```
gdal_translate.exe -projwin 7 53 8 52.5 satelit-os.tif sat1.tif
gdal_translate.exe -projwin 8 53 9 52.5 satelit-os.tif sat2.tif
gdal_translate.exe -projwin 7 52.5 8 52 satelit-os.tif sat3.tif
gdal_translate.exe -projwin 8 52.5 9 52 satelit-os.tif sat4.tif
```

gdalinfo zeigt, dass die Kacheln richtig erzeugt worden sind (man achte auf die veränderten Werte für Lower Left, Center etc.)

```
gdalinfo.exe sat1.tif
Driver: GTiff/GeoTIFF
Size is 500, 250
Coordinate System is ''
Origin = (6.999000,53.001000)
```

```
Pixel Size = (0.002000,-0.002000)
Metadata:
  TIFFTAG_DOCUMENTNAME=satelit-os.tif
  TIFFTAG_SOFTWARE=ImageMagick 6.0.6 09/26/04 Q16 http://www.imagemagick.org
Corner Coordinates:
Upper Left  (   6.9990000,  53.0010000)
Lower Left  (   6.9990000,  52.5010000)
Upper Right (   7.9990000,  53.0010000)
Lower Right (   7.9990000,  52.5010000)
Center      (   7.4990000,  52.7510000)
Band 1 Block=500x16 Type=Byte, ColorInterp=Red
Band 2 Block=500x16 Type=Byte, ColorInterp=Green
Band 3 Block=500x16 Type=Byte, ColorInterp=Blue
```

3.2 Erzeugung von Bildkatalogen

Die unter Abschnitt 3.1.2 erzeugen Kacheln sollen nun in einem so genannten Bildkatalog zusammengefasst werden. Das ist ein Shapefile, das als einzelnen Shapes die BoundingBoxen der einzelnen Kacheln enthält. Die zugehörige Dbase-Datei enthält ein Attribut, das den Pfad zu der zugehörigen Kachel enthält. Über diesen Pfad findet der UMN MapServer die richtige Rasterdatei im Dateisystem.

Das Kommando, mit dem solche Bildkataloge erstellt werden können heißt gdaltindex. Es bekommt als Übergabe Parameter den Namen des Shapefiles das erzeugt werden soll sowie eine Liste von Rasterdateien, die in den Bildkatalog eingehängt werden sollen. Wenn die Shapedatei schon existiert, werden die Rasterdateien an den Bildkatalog angehängt, so dass sich auch nachträglich noch weitere Rasterdaten hinzufügen lassen.

Der Aufruf kann demnach z.B. folgendermaßen aussehen:

```
gdaltindex sat-index pfad/zur/den/rasterdaten/sat1.tif
gdaltindex sat-index pfad/zur/den/rasterdaten/sat2.tif
gdaltindex sat-index pfad/zur/den/rasterdaten/sat3.tif
gdaltindex sat-index pfad/zur/den/rasterdaten/sat4.tif
```

Wenn man vermeiden möchte, das Kommando häufiger auf der Kommandozeile eingeben zu müssen, schreibt man sich am besten eine Batchdatei. Dies geht z.B. wie folgt:

1. Öffnen Sie eine DOS-Shell
2. führen Sie folgendes Kommando aus

   ```
   dir /b /n pfad/zu/den/Rasterdaten/*.tif > kachel.bat
   ```

3. Editieren Sie *kachel.bat* so, dass jeweils `gdaltindex` mit der Angabe des zu erzeugenden Bildkatalogs zu Beginn der Zeile steht

```
gdaltindex tileshape pfad/zu/den/Rasterdaten/1.tif
gdaltindex tileshape pfad/zu/den/Rasterdaten/2.tif
...
```

4. Führen Sie die Datei kachel.bat aus

Sie sollten jetzt ein neues Shapefile namens tileshape haben, das Sie als Bildkatalog in Ihr Mapfile einbinden können (siehe Abschnitt 4).

4 Datenanbindung und Darstellung – das Mapfile

Es existiert bereits eine Vielzahl an Dokumentationen über die Konfiguration der Datenanbindung sowie die Darstellung der Daten. Es soll hier daher kein weiterer Versuch gemacht werden, eine weitere evtl. bessere Dokumentation zu schreiben. Statt dessen wird ein kurzer Überblick über die Aufgaben des Mapfiles gegeben und auf die verschiedenen Quellen verwiesen, über die detaillierte Informationen darüber beschafft werden können, wie die zugehörige Mapfile-Konfiguration aussieht:

Generelle Einstellungen Dies betrifft z.B. den Kartenausschnitt, die Hintergrundfarbe der Karte u.ä. Informationen gibt es über

- Das Tutorial, Section 1.1
- UMN MapServer-Handbuch, Kapitel 2.3
- Online Dokumentation[2]

Anbindung verschiedener Datenquellen Es gibt eine Reihe unterschiedlicher Datenquellen, die eingebunden werden. Prinzipiell wird pro Ebene bzw. Layer eine Datenquelle angebunden. Welche dies ist, wird über die Parameter DATA, TYPE, CONNECTION und CONNECTIONTYPE geregelt. Siehe dazu u.a.

- Tutorial, Section 1.1 (Shapefiles), 1.5 (Raster), 1.6 (WMS)
- UMN MapServer Handbuch, Kapitel 2.10 (Shapefiles), 2.9 (Raster und Bildkataloge aus Rasterdaten), 4.1.2 (PostGIS/PostgreSQL)
- Online-Dokumentation[3]
- Weitere Online-Dokumentation
  - Raster[4]
  - WMS[5]

Darstellung von Vektordaten Eine sehr gute Zusammenfassung der Möglichkeiten zur Erzeugung von einfachen und komplexen Symbolen für die Darstellung von Vektordaten findet sich Online auf den Seiten von Mapmedia[6]. Sie ist im Rahmen eines Praktikums von Peter Freimuth bei der MapMedia GmbH entstanden. Es kann auch der vollständige Praktikumsbericht[7] herunter geladen werden.

[2] `http://mapserver.gis.umn.edu/doc44/mapfile-reference.html#map`
[3] deutsch: `http://www.umn-mapserver.de/doc44/mapfile-reference44.html#layer`
englisch: `http://mapserver.gis.umn.edu/doc44/mapfile-reference.html#layer`
mit den jeweiligen Parametern
[4] `http://mapserver.gis.umn.edu/doc42/raster-howto.html`
[5] `http://mapserver.gis.umn.edu/doc42/wms-client-howto.html`
[6] `http://www.mapmedia.de/dokumente/umn_signaturen_howto/index.html`
[7] `http://www.mapmedia.de/dokumente/umn_signaturen_howto/Praktikumsarbeit.zip`

In diesem Zip-Archiv finden sich auch die Symbol-Dateien, mit denen die Beispiele erzeugt worden sind. Einführende Informationen bekommt man über das Tutorial, Section 1.3 und 1.4

Klassifizierung von Vektordaten Unter http://mapserver.gis.umn.edu/doc44/mapfile-reference.html#class wird unter dem Punkt „EXPRESSION“ erklärt, welche Möglichkeiten für die Klassifizierung von Geodaten es gibt. Demnach gibt es

- Stringvergleiche: Dazu wird auf Layerebene ein CLASSITEM angegeben und unter EXPRESSION ein String angegeben auf den verglichen wird. Folgendes Beispeil zeigt alle Grünflächen vom Typ „Wald“:

```
LAYER
    NAME "gruenflaeche"
    TYPE POLYGON
    STATUS DEFAULT
    DATA "test"
    CLASSITEM "typ"
    CLASS
        NAME "Wald"
        EXPRESSION "wald"
        COLOR 0 255 0
    END
END
```

- Reguläre Ausdrücke: Ein regulärer Ausdruck wird in Slashes (/) eingefasst. Folgendes Beispiel zeigt alle Grünflächen, deren Typ-Beschreibung das Wort „wald“ enthalten, also z.B. „Nadelwald“, „Lauubwald“ etc.

```
LAYER
    NAME "gruenflaeche"
    TYPE POLYGON
    STATUS DEFAULT
    DATA "test"
    CLASSITEM "typ"
    CLASS
        NAME "gruenflaeche"
        EXPRESSION /wald/
        COLOR 255 0 0
    END
END
```

  In folgendem Beispiel tauchen in der Typbeschreibung der Grünfläche entweder „Wald“ oder „Park“ auf (Oder-Verknüpfung)

```
LAYER
```

```
        NAME "gruenflaeche"
        TYPE POLYGON
        STATUS DEFAULT
        DATA "test"
        CLASSITEM "typ"
        CLASS
            NAME "gruenflaeche"
            EXPRESSION /Wald|Park/
            COLOR 255 0 0
        END
    END
```

- Logische Ausdrücke: Logische Ausdrücke benötigen nicht mehr die Angabe des CLASSITEM. Logische Ausdrücke setzen sich aus Vergleichen von Attributen mit Werten zusammen, die wiederum durch logische Operatoren verbunden werden können. Der gesamte Ausdruck wird in Klammen gefasst, die Attributnamen in eckige Klammern. Für String-Vergleiche werden sowohl der Attributname als auch der Wert in einfache Hochkommata gesetzt. Ein Beispiel:

  ```
  EXPRESSEION ([POPULATION] > 50000 AND '[LANGUAGE]' eq 'FRENCH')
  ```

  Das Zahl-Attribut POPULATION soll einen Wert größer als 50000 habe und gleichzeitig soll das Character-Attribut LANGUAGE auf FRENCH gesetzt sein.

  Mögliche Vergleichsoperatoren sind =,¿,¡,¡=,¿=,=,or,and,lt,gt,ge,le,eq. Mögliche logische Operatoren sind AND sowie OR.

Maßstabsbalken Der Maßstabsbalken wird durch das SCALEBAR-Objekt im Mapfile definiert. Siehe dazu

- Tutorial, Section 2.3
- UMN MapServer Handbuch, Kapitel 2.14
- Online-Dokumentation[8]

Legende Die Legende wird durch das LEGEND-Objekt im Mapfile definiert. Außerdem muss in jeder Klasse, die in der Legende auftauchen soll der Parameter NAME definiert werden. Siehe dazu

- Tutorial, Section 2.4
- UMN MapServer Handbuch, Kapitel 2.15

[8] `http://www.umn-mapserver.de/doc44/mapfile-reference44.html#scalebar`

- Online-Dokumentation[9]

Übersichtskarte Die Referenzkarte wird durch das REFERENCE-Objekt im Mapfile definiert. Siehe dazu

- Tutorial, Section 2.5
- Online-Dokumentation[10]

[9] `http://www.umn-mapserver.de/doc44/mapfile-reference44.html#legend`
[10] `http://www.umn-mapserver.de/doc44/mapfile-reference44.html#referencemap`

5 Erstellung von Oberflächen: Templates

Auch für das Erstellen von Oberflächen durch Verwendung von HTML-basierten Templates ist schon eine Menge geschrieben worden. An dieser Stelle sollen daher wieder nur die groben Prinzipien dargestellt werden. Für detailliertere Informationen wird wieder auf die entsprechende Dokumentation verwiesen. Da der UMN MapServer eine Reihe von temporären Dateien im Dateisystem hinterlässt wird in einem weiteren Abschnitt beschrieben, wie dafür gesorgt werden kann, diese regelmäßig zu entfernen. Am Ende wird die Installation von dynamap beschrieben, einer HTML/JavaScript-basierten Oberfläche, die sich schnell installieren lässt und eine angenehmere Navigation auf einer Karte zulässt, als eine rein HTML-basierte Oberfläche.

5.1 Prinzipielle Verwendung von Templates

Das Prinzip des Templates geht davon aus, dass der UMN MapServer als CGI-Skript gestartet wird. Dazu wird dem UMN MapServer als Parameter das Mapfile mitgegeben, das er einlesen soll und zur Darstellung verwenden soll (map=/pfad/zur/mapdatei). Über so genannten Modi wird dem UMN MapServer mitgeteilt, was er darstellen soll (mode=map erzeugt z.B. eine statische Karte, mode=legend erzeugt eine Legendengraphik usw.). Weitere Parameter geben an, wie die Karte aussehen soll. So lässt sich etwas über den Parameter extent der Ausschnitt der Karte und über den Parameter layer steuern, welche Kartenebenen angezeigt werden sollen. Eine Übersicht darüber, welche CGI-Parameter zur Steuerung zur Verfügung stehen, gibt es in der Online-Dokumentation[11].

Die Parameter sollen nun nicht von einem Benutzer per Hand eingegeben werden, sondern werden innerhalb eine HTML-Formulare (<form>-Element) als Input-Parameter angegeben und an den UMN MapServer übergeben. Das bereits erwähnt Template ist eine solche HTML-Seite, in der Platzhalter für die dynamischen Inhalte der Seite, also die Karte, die Legende, die Übersichtskarte, aber auch für die einzusetzenden Parameter gelassen werden. Diese Platzhalter werden in eckige Klammern ([]) eingefasst. Der UMN MapServer liest das Template in bestimmten Modi (z.B. mode=browse) ein, sucht alle Platzhalter des Templates und füllt sie mit den dynamischen Inhalten. Dieses ausgefüllte Template wird dann an den Browser geschickt, der damit die neu gerenderte Karte darstellen kann. Eine Übersicht über die Platzhalter, die verwendet werden können findet sich in der Online-Dokumentation[12]. Im WEB-Objekt des Mapfiles wird über den Parameter TEMPLATE festgelegt, welches Template verwendet werden soll.

[11] `http://mapserver.gis.umn.edu/doc42/cgi-reference.html`
[12] `http://www.umn-mapserver.de/doc42/mapserver_template.html`

Das Tutorial führt in die Verwendung von Templates zur Steuerung der UMN MapServer ein. Dabei sind vor allem die Abschnitte 1.9, 2.1, 2.2 interessant.

5.1.1 Templates in Abfragen

Ähnlich funktioniert die Rückgabe einer Abfrage durch den UMN MapServer. Anstatt eines Templates werden nun allerdings mehrere Templates definiert, um die Fälle abzudecken, in denen Ergebnisse aus mehr als einem Layer oder mehrere Ergebnisse aus einem Layer zurückgeliefert werden. Daher existieren Template-Definitionen im WEB-Objekte (HEADER und FOOTER (beide optional), in jedem abfragbaren Layer-Objekt und zwar dort HEADER und FOOTER (beide optional) auf LAYER-Ebenen sowie TEMPLATE (zwingend notwendig) innerhalb jeder abfragbaren Klasse (CLASS-Objekt). Die Rückgabe sieht dann schematisch wie folgt aus:

```
HEADER (WEB)
    HEADER (LAYER1)
        TEMPLATE (LAYER1)
        TEMPLATE (LAYER1)
    FOOTER (LAYER1)
    HEADER (LAYER2)
        TEMPLATE (LAYER2)
    FOOTER (LAYER2)
HEADER (WEB)
```

Dieses Vorgehen wird im Tutorial Schritt für Schritt angewandt, wobei zunächst nur das zwingend erforderliche Template definiert wird (Beispiel 3.1), danach der Header und Footer des Layers (Beispiel 3.2) und schließlich der Header und Footer des Web-Objekts (Beispiel 3.3)

5.2 Entfernen alter Graphiken

Der UMN MapServer legt die von ihm erzeugen Graphiken (Karten, Legenden etc.) im Dateisystem ab. Um zu verhindern, dass die Festplatte deswegen zu voll wird, sollten die Dateien regelmäßig abgeräumt werden. Dies geschieht durch das Einrichten eines regelmäßigen Jobs (so genannter Geplante Task) unter Windows. Folgende Schritte sind dazu notwendig:

1. Kopieren Sie bin/delete_temp_files.bat an eine beliebige Stelle im Dateisystem, z.B. C:/Programme

2. Passen Sie den Pfad in delete_temp_files.bat nach den aktuellen Bedürfnissen an

3. Öffnen Sie die Systemsteuerung, dort die Geplanten Tasks und klicken Sie auf „Geplanten Task hinzufügen“. Ein Assistent öffnet sich. Klicken Sie zunächst auf „Weiter“, dann auf „Durchsuchen“, so dass Sie das Skript „delete_temp_files.bat“ als einzurichtenden geplanten Task auswählen können. Danach geben Sie an, wann und wie oft der Task ausgeführt werden soll, also z.B. „täglich“, bestätigen mit „Weiter“ und Angabe einer Uhrzeit. Danach können Sie einen Benutzer auswählen, als der der Task ausgeführt werden soll. Den Rest können Sie mit „Weiter“ bzw. „Fertig stellen“ bestätigen.

4. Überprüfen Sie am nächsten Tag, ob der geplante Task wie gewünscht die von Ihnen erzeugen Dateien löscht. Andernfalls sollten Sie überprüfen, ob die vorangegangenen Schritte von Ihnen tatsächlich korrekt ausgeführt worden sind.

5.3 Installation von dynamap

Die Installation von dynamap erfordert zunächst die Installation der dynamap-spezifischen Scripte sowie Dateien in die DocumentRoot. Dazu wird das Verzeichnis material/dynamap z.B. in das htdocs-Verzeichnis des Apache gelegt.

Im nächsten Schritt wird das Beispiel-Template von Frida, das dynamap für die Navigation verwendet als Template im Mapfile eingetragen. Dazu wird die Datei z.B. in das gleiche Verzeichnis gelegt, in dem das betreffende Template liegt (Kopieren von material/frida/html/frida.html nach Pfad/zum/mapfile/meintemplate.html und der Parameter TEMPLATE im WEB-Objekt des Mapfiles angepasst:

```
...
WEB
   TEMPLATE "meintemplate.html"
   ...
END
```

Als nächstes müssen die Pfade im Template auf die aktuelle Installation angepasst werden. Dazu wird das Template geöffnet und alle Vorkommen von /mapserver/schulung/ durch die URL zur DocumentRoot ersetzt. Wenn dynamap direkt in der DocumentRoot abgelegt wird, wird /mapserver/schulung einfach gelöscht. Außerdem muss u.U. die Action im Form-Element angepasst werden. Achtung: Unter Windows heißt das CGI-Binary nicht bloß mapserv, sondern mapserv.exe!

Als letztes muss die Liste der Layer im Template auf die gewünschte Layer-Liste angepasst werden. So wird z.B.

```
<!-- Strassen -->
    <tr>
      <td><input type="checkbox" name="layers" value="strassenall"
```

```
                    [strassenall_check] onClick="doUpdate()">
      </td>
      <td>
        <small>Strassen</small>
      </td>
    </tr>
    <!-- Ende Strassen -->
```

ersetzt durch

```
<!-- Gebäude -->
   <tr>
     <td><input type="checkbox" name="layers" value="gebaeude"
                 [gebaeude_check] onClick="doUpdate()">
     </td>
     <td>
       <small>Gebäude</small>
     </td>
   </tr>
   <!-- Ende Gebäude -->
```

6 PHP MapScript

MapScript stellt eine Möglichkeit dar, den UMN MapServer in komplexeren Anwendungen zu verwenden. Dabei wird der UMN MapServer selber im Wesentlichen als Programm zum Rendern von Karten verwendet. MapScript baut dabei auf die Objektstruktur auf, die innerhalb des Mapfiles verwendet wird. Jedes Attribut eines Objektes oder Unterobjektes lässt sich über MapScript verändern. Des weiteren lassen sich neue Objekte erzeugen, löschen, ihre Reihenfolge verändern. Man kann die Karte rendern, die Legende rendern – kurz man kann sämtliche Funktionen des UMN MapServer über MapScript steuern.

Die am häufigsten benutzt MapScript Variante ist sicherlich PHP MapScript. Es gibt aber auch eine Reihe von anderen Skript-Sprachen, über die man den UMN MapServer steuern kann, z.B. Python und Perl. Prinzipiell wichtig zu wissen bei der Programmierung mit MapScript sind folgende Punkte

- Die Objektstruktur des Mapfiles wird bei MapScript voll eingehalten. So gibt es ein Mapobjekt, das als Unterobjekte mehrere Layer-Objekte hat, die wiederum Class-Objekte als Unterobjekte enthalten können usw.
- Die Objektstruktur kann entweder aus einem Mapfile eingelesen werden, wie es in den folgenden Beispiel-Skripten getan wird. Es ist aber auch möglich, die gesamte Objekt-Struktur selbstständig aufzubauen.
- Skripte, die in MapScript geschrieben sind, werden vom Webserver prozessiert ohne als Templates wie im vorangegangenen Abschnitt beschrieben betrachtet zu werden. Das heißt, dass ein automatisches Ersetzen von Platzhaltern nicht erfolgt. Statt dessen werden mit den Skript-typischen Kommandos die entsprechenden dynamischen Inhalte eingesetzt. Es folgt ein Beispiel, das ein Bild erzeugt, es im Dateisystem speichert und dessen URL als Quelle eines dynamischen Bildes in eine HTML-Seite einbettet:

```
<?php

    dl("php_mapscript.dll");

    $map_file="C:\htdocs\dynamap\frida\map\frida.map";
    $map = ms_newMapObj($map_file);

    // Bild zeichnen und URL holen
    $image=$map->draw();
    $image_url=$image->saveWebImage();
```

```
    ?>
<HTML>
     <HEAD>
    <TITLE>Mapserver Karte</TITLE>
    </HEAD>
    <BODY>

    <input type=image src="<?php echo $image_url?>">

</BODY>
</HTML>
```

- Wie bereits gesagt, lassen sich sämtliche Attribute eines Objektes über MapScript steuern. So ändert folgendes Beispiel den Status aller Layer auf OFF und setzt nur diejenigen Layer auf ON, die über die URL explizit angefordert worden sind:

```
<?php

    dl("php_mapscript.dll");

    $map_file="C:\htdocs\dynamap\frida\map\frida.map";
    $map = ms_newMapObj($map_file);

    // Zunächst alle Layer unsichtbar machen.
    for ($i=0; $i<$map->numlayers;$i++){
        $mylayer = $map->getLayer($i);
        $mylayer->set("status", MS_OFF);
    }

    // Jetzt die Liste der darzustellenden Layer holen...
        $layers = split(",",$_GET['layers']);

    // ... und sie wieder sichtbar machen.
        for ($i=0; $i<count($layers);$i++){
        $mylayer = $map->getLayerByName($layers[$i]);
        $mylayer->set("status", MS_ON);
    }

    // Bild zeichnen und URL holen
    $image=$map->draw();
    $image_url=$image->saveWebImage();
```

```
    ?>
<HTML>
     <HEAD>
    <TITLE>Mapserver Karte</TITLE>
    </HEAD>
    <BODY>

    <input type=image src="<?php echo $image_url?>">

</BODY>
</HTML>
```

Informationen über den vollen Umfang der Möglichkeiten mit PHP MapScript erhält man über die Seite http://mapserver.gis.umn.edu/doc44/phpmapscript-class-guide.html. Ein sehr gutes Beispiel für ein PHP MapScript basiertes Tool ist p.mapper[13]. P.mapper erlaubt mit wenigen Konfigurationsschritten das Aufsetzen einer Javascript-basierten Webmapping-Anwendung. Serverseitig kommt dabei wie gesagt PHP MapScript zum Einsatz, das für den Aufbau einer Legende, das Rendern der Karte etc. sorgt.

Die drei Skript insert.php, drawpoint.php und write_point.php unter material/php bieten außerdem ein Beispiel dafür, wie über eine Webbasierte Oberfläche ein Punkt eingegeben wird, der schließlich in einem Shapefile gespeichert wird.

[13] http://pmapper.sourceforge.net/