

Osnabrück, Februar 2007
Kai Behncke
Institut für Geoinformatik und Fernerkundung Osnabrück
kbehncke@igf.uni-osnabrueck.de

Einführung in WMS und WFS (mit praktischen Beispielen in UMN MapServer und Mapbender)

Fragen zu WMS/WFS und UMN MapServer bitte an die Mailinglist von
www.umn-mapserver.de
oder ins Forum auf www.umn-mapserver-community.de

Fragen zum Mapbender bitte an die Mailinglist von
http://www.mapbender.org/index.php/Mapbender_Mailing_Lists

Anregungen/Korrekturvorschläge zu dieser Einführung bitte an den Autoren senden.

Dank gebührt den vielen NutzerInnen der Mailinglisten/Foren von UMN MapServer und Mapbender, welche diese Anleitung erst ermöglichten.

Inhaltsverzeichnis

1. WMS - Was ist das eigentlich?	Seite 1
1.1 Wozu soll das gut sein?	Seite 1
1.2 Mögliche Abfrageoperationen	Seite 2
1.3 Erstellung eines eigenen WMS-Servers im UMN MapServer	Seite 3
1.4 GetCapabilities	Seite 6
1.5 GetMap	Seite 9
1.5.1 Veränderung einer GetMap-Ausgabe mittels SLD	Seite 11
1.6 GetFeatureInfo	Seite 14
Exkurs A: GetFeatureInfo im Mapbender	Seite 17
1.7 WMS-Layer in den UMN MapServer als WMS-Client einbinden	Seite 21
 2. WFS -Web Feature Service	 Seite 24
2.1 WFS-Server mit dem UMN MapServer	Seite 25
2.2 WFS-GetCapabilities	Seite 27
2.3 WFS-DescribeFeatureType	Seite 28
2.4 WFS-GetFeature	Seite 28
2.5 UMN MapServer als WFS-Client	Seite 31
Exkurs B: Einsatz einer WFS-Suche im Mapbender	Seite 33
Exkurs C: WFS-Tooltip im Mapbender	Seite 38

Voraussetzungen

Arbeitsumgebung der nachfolgend angeführten Beispiele war ein Debian Sarge 3.1 Betriebssystem in Verbindung mit dem UMN Mapserver 4.10.0 und Mapbender 2.4.0.

Auch wenn in diesem Fall ein Linux-Betriebssystem gewählt wurde, so können Sie die Beispiele natürlich auch unter Windows testen.

Die Beispieldaten für dieses Tutorium können Sie von:

http://www.selbstverwaltung-bundesweit.de/mapserver/wms_wfs_tutorium.zip herunterladen.

Für dieses Tutorium sollten Sie Erfahrungen im Umgang mit dem UMN MapServer und Grundkenntnisse im Umgang mit Mapbender besitzen.

1. WMS - was ist das eigentlich?

Unter OGC konformen WMS (Web Map Service) versteht man einen Standard, *“der sowohl die Syntax der Anfragen nach einem Kartenbild, als auch Format und Eigenschaften des Ergebnisses dieser Anfrage regelt. Von einem WMS (Dienst) werden nicht Geometriedaten angefordert, sondern deren visuelle Präsentation als Raster-Bild.”*¹ oder wie ERSTLING und SIMONIS schreiben: *“Web Map Services, die den Spezifikationen des OGC genügen, müssen eine Schnittstelle implementieren, die sowohl das Format der Anfrageoperationen (requests) wie auch die entsprechenden Antwortformen (responses) standardisiert.”*²

Hierbei spielt es keine Rolle von welchem Kartenserver die Kartenbilder geliefert werden: UMN MapServer, GeoServer, ArcIMS oder Deegree; wichtig ist nur, dass der Server einen OGC konformen WMS anbietet.

Anhand dieses Dienstes werden also Geodaten zu einem Rasterbild gerendert und als Response an die standardisierte Abfrage (den Clienten) zurückgegeben.

*“Die tatsächliche Art der Datenhaltung oder der Datenform (Raster oder Vektor) spielen dabei keine Rolle. Denn durch die WMS-Spezifikation von OGC wird lediglich eine system- und herstellerunabhängige Abfragemethode geschaffen, die von jedem GIS-System, welches den besagten OGC-Dienst unterstützt, eine Karte als Rasterbild anfordern kann.”*³

Definiert ist dieser Standard in der Web Map Service Implementation Specification des OGC.⁴

1.1 Wozu soll das gut sein?

Auch im Jahre 2007 ist es nach wie vor ein großes Problem “mal eben” Geodaten für die eigene Anwendung zu erhalten, an der Einschätzung von FISCHER hat sich leider nichts geändert:

*“Die wenigsten Besitzer von Geodaten rücken diese umsonst, oder überhaupt heraus.”*⁵

¹ teresstris & CCGIS 2004, S.27

² Erstling & Simonis 2005, S.110

³ Tscherkasski 2006, S.5

⁴ Open GIS Consortium (b), 2002

⁵ Fischer 2003, S.95

Wenn Geodatenanbieter Daten jedoch als WMS bereit stellen, dann können auch andere Nutzer darauf zugreifen, die Existenz des OGC konformen WMS-Standards trägt dann also zur Bereitstellung von Geodaten bei.

Nach FISCHER kann ein WMS auch zur Verteilung von Rechenlast genutzt werden, wenn einzelne Layer auf verschiedene Server verteilt werden.⁶

Mittels eines WMS können also verteilt vorliegende Karten in eine Anwendung integriert werden.

Von einem Server holt man sich z.B. eine Übersichtskarte von Deutschland, von einem anderen die TK100, von einem dritten die Gemeinden der Wesermarsch etc.

1.2 Mögliche Anfrageoperationen

Die Anfrageoperationen, welche in einem OGC konformen WMS integriert sein **müssen**, sind:

- GetCapabilities
- GetMap

optional ist die Operation:

- GetFeatureInfo

Mapserver, welche den OGC-Standard SLD (Styled Layer Descriptor) unterstützen, bieten weitere Abfrageoperationen:

- describeLayer
- getLegendGraphic
- getStyles
- putStyles

Der UMN MapServer unterstützt SLD in einem gewissen Rahmen seit Version 4.2.0.⁷ Mittels SLD ist es möglich, "von außen" die ausgelieferten Karten eines WMS zu verändern. Die Anfrageoperationen werden über eine URL im Browser abgeschickt, an späterer Stelle werden einige Beispiele behandelt.

⁶ ebda. S. 96

⁷ siehe History.Txt in den Sources des UMN MapServer 4.10.0

1.3 Erstellung eines eigenen WMS-Servers im UMN MapServer

Nachfolgend wird das Mapfile für die Erstellung eines eigenen WMS-Servers dargestellt. Sie finden es auch in den Beispieldaten im Ordner „wms_ogc“ (wms_server.map).

Zusätzlich soll die Anleitung unter:

http://mapserver.gis.umn.edu/docs/howto/wms_server

empfohlen werden.

```
MAP
NAME      'Testkarte'
STATUS ON
IMAGETYPE PNG
EXTENT    3430018 5876533 3491970 5948485
UNITS     METERS
SIZE      500 500
SHAPEPATH '/mapservertest/wms_ogc/data'
SYMBOLSET '/mapservertest/wms_ogc/symbols/symbols.sym'
FONTSET   '/mapservertest/wms_ogc/fonts/fonts.list'
```

```
WEB
TEMPLATE 'template.html'
  IMAGEPATH '/var/www/htdocs/tmp/'
  IMAGEURL  '/tmp/'
```

```
METADATA
'WMS_TITLE'      'Wesermarsch WMS' #zwingend notwendig
'WMS_ABSTRACT'   'Dies ist ein einfaches WMS-Beispiel' #optional
'WMS_FEATURE_INFO_MIME_TYPE' 'text/html' #optional.
```

```
'WMS_ONLINERESOURCE' 'http://127.0.0.1/cgi-
bin/mapserv?map=/mapservertest/wms_wfs/wms_server.map' #zwingend notwendig
'WMS_SRS'          'epsg:31467' #zwingend notwendig
END
END
```

```
PROJECTION
  'init=epsg:31467'
END
```

```
LAYER
NAME      'landkreis_export'
TYPE      POLYGON
STATUS    ON
DATA      'Lan.shp'
LABELITEM 'name'
```

```
METADATA
'WMS_TITLE' 'landkreis_export' #zwingend notwendig
```

```

'WMS_SRS' 'epsg:31467' #zwingend notwendig
'WMS_INCLUDE_ITEMS' 'all' #optional
END
CLASS

STYLE
OUTLINECOLOR 0 200 0
SYMBOL 0
COLOR 50 128 0
END

    LABEL
        TYPE TRUETYPE
        ANTIALIAS TRUE
        FONT 'arial'
        COLOR 0 0 0
        BACKGROUNDCOLOR 200 200 200
        SIZE 10
        BUFFER 2

    END
TEMPLATE getfeatureinfo.html

END
END
LAYER
SYMBOLSCALE 50000
NAME 'testdaten_export'
TYPE POINT
TOLERANCE 5
STATUS ON

METADATA
'WMS_TITLE' 'testdaten_export' #zwingend notwendig
'WMS_SRS' 'epsg:31467' #zwingend notwendig
END
DATA 'testdaten.shp'
CLASS
STYLE
SYMBOL 'triangle'
COLOR 120 0 0
SIZE 9
MINSIZE 9
MAXSIZE 20
END

END
END

```

END

Die **gelb markierten Stellen** sind in dem Mapfile die relevanten Positionen um einen WMS zu erstellen.

Natürlich sind deutlich mehr WMS-Parameter möglich.

Eine gute Übersicht über mögliche METADATA-Parameter findet man unter:

http://mapserver.gis.umn.edu/docs/howto/wms_server

Nun, wofür stehen die einzelnen Parameter?

Im Web-Bereich:

WMS_TITLE -> Titel der Karte

WMS_ABSTRACT -> Hier lassen sich Beschreibungen des Servers festlegen

WMS_FEATURE_INFO_MIME_TYPE -> **Achtung: Damit aber bei einer GetFeatureInfo-Anfrage Daten auch im HTML-Format ausgegeben werden können muss dieser Parameter gesetzt sein**

WMS_ONLINERESOURCE -> Wichtig für die GetCapabilities-Anfrage. Gibt die URL an, über die auf den Server zugegriffen wird.

WMS_SRS -> Projektionssystem(e)

Im Layer-Bereich:

WMS_TITLE -> Titel des Layers

WMS_SRS -> Projektionssystem des Layers

'WMS_INCLUDE_ITEMS' 'all' -> **Achtung aufgepaßt!! Dieser Parameter ist für den GetFeatureInfo-Aufruf gedacht. Bis zur Version 4.8.0 wurden bei diesem Aufruf standardmäßig immer alle Attribute eines Geoobjektes (also z.B. Flächengröße, Bevölkerungsdichte etc (je nach Daten)) ausgegeben. Will man dieses in Version 4.10.0 auch gewährleisten, so muss dieser Parameter angegeben werden.**

Ok, der Server ist eingerichtet?

Dann jetzt speziell zu den Anfrageoperationen.

1.4 GetCapabilities

GREWE schreibt: *“Die Get Capabilities-Schnittstelle eines WMS ist in der Lage, eine Selbstauskunft über ihre Funktionalitäten zu geben. Ein WMS ist somit selbstbeschreibend (...). Durch eine Antwort auf eine Get Capabilities-Anfrage gibt der Server Auskunft über seine verfügbaren Operationen, Karten, Kartenlayer und räumlichen Referenzsysteme. Die Antwort, eine wohlgeformte und gültige XML-Datei, ist sowohl von einem Computer als auch von einem Menschen les- und interpretierbar.*

Aus den Informationen, welche über die Get Capabilities-Schnittstelle zur Verfügung gestellt werden, lässt sich ein gültiger Get Map Request generieren, welcher schließlich zur Darstellung von Geodaten im Browser führt.”⁸

Der Aufruf, bezogen auf unseren WMS, lautet wie folgt:

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wms_ogc/wms_server.map&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities

Als Ergebnis erhalten wir Informationen über den WMS im XML-Format.

Schauen wir uns die zurückgegeben Infos einmal kurz an.

Abbildung 1 stellt einen Auszug dar mit Title, Abstract und Onlineresource.

Abbildung1: Auszug aus dem XML-Dokument der GetCapabilities-Antwort

```
<!-- end of DOCTYPE declaration -->
- <WMT_MS_Capabilities version="1.1.1">
- <!--
  MapServer version 4.10.0 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=WMF OUTPUT=SVG SUPPORTS=PROJ SUPPORTS=FREETYPE :
-->
- <Service>
  <Name>OGC:WMS</Name>
  <Title>Wesermarsch WMS</Title>
  <Abstract>Dies ist ein einfaches WMS-Beispiel</Abstract>
  <OnlineResource xlink:href="http://127.0.0.1/cgi-bin/mapserv?map=/mapserver/test/wms_ogc/wms_server.map&"/>
</Service>
```

Die Infos in Abbildung 2 sind schon interessanter. Hier erhält man Informationen zu den EPSG-Codes, den aufrufbaren Layern und deren Bounding-Boxes (in Lat/Lon sowie in Gauß-Krüger-Koordinaten).

Auch erhält man die Information, dass der Layer “Landkreis Export” “queryable” ist, also Abfragen zu diesem möglich sind (dazu später mehr).

Abbildung 2: Auszug aus dem XML-Dokument der GetCapabilities-Antwort

```
<UserDefinedSymbolization SupportSLD="1" UserLayer="0" UserStyle="1" RemoteWFS="0"/>
- <Layer>
  <Name>Testkarte</Name>
  <Title>Wesermarsch WMS</Title>
  <SRS>epsg:31467</SRS>
  <LatLonBoundingBox minx="7.94115" miny="53.0181" maxx="8.88032" maxy="53.6692"/>
  <BoundingBox SRS="EPSG:31467" minx="3.43002e+06" miny="5.87653e+06" maxx="3.49197e+06" maxy="5.94848e+06"/>
- <Layer queryable="1" opaque="0" cascaded="0">
  <Name>landkreis_export</Name>
  <Title>landkreis_export</Title>
  <SRS>epsg:31467</SRS>
  <LatLonBoundingBox minx="8.1191" miny="53.0957" maxx="8.65646" maxy="53.732"/>
  <BoundingBox SRS="EPSG:31467" minx="3.44187e+06" miny="5.88502e+06" maxx="3.47699e+06" maxy="5.95553e+06"/>
</Layer>
- <Layer queryable="0" opaque="0" cascaded="0">
  <Name>testdaten_export</Name>
  <Title>testdaten_export</Title>
  <SRS>epsg:31467</SRS>
  <LatLonBoundingBox minx="8.24151" miny="53.156" maxx="8.55556" maxy="53.5994"/>
  <BoundingBox SRS="EPSG:31467" minx="3.44979e+06" miny="5.89164e+06" maxx="3.47027e+06" maxy="5.94081e+06"/>
</Layer>
```

Bei unserem selbst erstellten WMS wissen wir natürlich, welche Layer darstellbar sind, Rufen wir nun einmal die Capabilities eines externen Servers der LGN (Landesvermessung und Geobasisinformation Niedersachsen) auf:

<http://www.mapserver.niedersachsen.de/freezeogc/mapserverogc?SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.1.1>

In dem zurückgelieferten Dokument erhält man beispielsweise Informationen zu vier verschiedenen Layern. Sollten Sie versuchen diese aufzurufen: Das wird, bis auf einen, nicht funktionieren, da dafür die IP Ihres Rechners freigeschaltet werden muss.

Für Nutzer frei zugänglich ist aber die TK100 aus Niedersachsen.

Die XML-Daten zu diesem Layer sehen wie folgt aus (Abb. 3):

Abbildung 3: Auszug aus XML-Dokument eines WMS der LGN

```
- <Layer queryable="0" opaque="1">
  <Name>TK 100</Name>
  <Title>Topographische Karte 1:100000</Title>
  <LatLonBoundingBox minx="6.180364501677226" miny="50.50279594367625" maxx="13.683332705481225"
  maxy="54.94030655149304"/>
  <BoundingBox SRS="EPSG:31467" minx="3300000" miny="5600000" maxx="3800000" maxy="6100000"/>
  <BoundingBox SRS="EPSG:31493" minx="3300000" miny="5600000" maxx="3800000" maxy="6100000"/>
  <ScaleHint min="6.76160460880501" max="52.82503600628914"/>
</Layer>
```

Interessant ist hier z.B. der Bereich des ScaleHints.

Um den Scale übrigens in einen Maßstab umzurechnen benötigt man folgende Formel:

$(\text{ScaleHint} * \text{dpi} / 0,0254) / \sqrt{2}$ (mit dem Problem, dass die dpi-Auflösung in dem GetCapabilities-Dokument nicht angegeben wird, die TK100 der LGN wird standardmäßig übrigens in 170 dpi ausgeliefert (dann also zwischen 1: 32.000 und 1:250000 aufrufbar– Entsprechende Fehlermeldung erhalten Sie, wenn Sie diesen Layer in einem nicht entsprechenden Maßstab anfordern)).

Vorsicht: Wenn Sie mit der TK100 der LGN experimentieren, dann werden Sie sich vielleicht wundern, dass diese erstmal nur zwischen 1:13000 und 1:105000 angezeigt wird. Bedenken Sie, dass die Standardresolution des Mapservers bei 72 liegt. Wenn Sie diese im Map-Bereich umdeklariieren (RESOLUTION 170) dann wird die Karte auch zwischen 1:32000 und 1:250000 ausgegeben.

Es handelt sich beim “Scalehint” nicht um eine direkte Maßstabsangabe sondern es wird dabei die Größe der Diagonalen eines Bildpixels in der Natur in Metern angegeben.

Ein Beispiel:

Ein Mapserver fordert einen Ausschnitt an, welcher eine Fläche in der Natur von 100x100 m und einer Bildgröße von 100x100 Pixel hat.

Die Breite eines Pixel entspricht also einem Meter in der Natur. Laut Pythagoras ist der ScaleHint also die Wurzel aus 2.

In der Berechnung des ScaleHints spielt die Auflösung aber eine entscheidene Rolle. Wenn der Dienst mit 96 dpi antwortet, ergibt das einen Maßstab von genau 1:3780.

Der 72 dpi Dienst liefert aber einen Maßstab von 1:2835!

Wohlgemerkt: Das gelieferte Bild sieht auf dem Monitor bei beiden Diensten gleich aus, es ist ja die gleiche Fläche in der Natur, welche in beiden Fällen mit 100x100 Pixeln dargestellt wird.

Die Probleme treten dann auf, wenn eine Applikation einen Dienst in einer Auflösung bekommt (z.B. 96 dpi), intern aber von 72 dpi ausgeht und den ScaleHint mit 72 dpi in einen metrischen Maßstab umrechnet.

Wenn diese Applikation die DOPs von 1:500 - 1:3000 einbindet, wäre das obige Beispiel bei einem 72-dpi-Dienst zu sehen, bei einem 96-dpi-Dienst aber nicht zu sehen, obwohl die Parameter im GetMap-Aufruf die gleichen sind.⁹

Der Wert 0,0254 ergibt sich übrigens aus der Umrechnung von Zoll in Meter (vielen Dank an Herrn Frank).

⁹ http://www.geodaten-mv.de/geoportal/html/dienste_nutzung.html

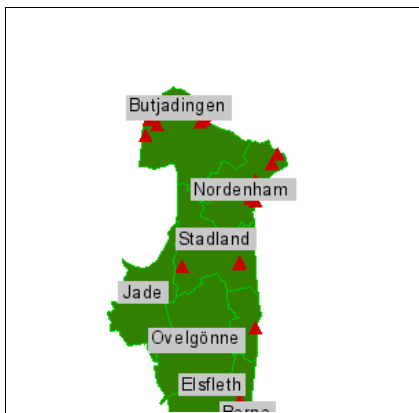
1.5 GetMap

Mittels GetMap kann man von einem Server eine Karte als Rasterbild anfordern. Versuchen wir dieses mal mit unserem selbst aufgebauten Wesermarsch-WMS. Die Anfrage sieht aus wie folgt:

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wms_ogc/wms_server.map&SERVICE=WMS&VERSION=1.1.0&REQUEST=GetMap&layers=landkreis_export,testdaten_export&BBOX=3430000,5899000,3490000,5953000&SRS=epsg:31467&Format=image/gif&width=300&height=300&BGCOLOR=ffffff&STYLES=,

Folgendes Bild wird zurückgegeben:

Abbildung 4: GetMap-Ausgabe



Nachfolgend eine erläuternde Übersicht zu den GetMap-Parametern.

Abbildung 5: Übersicht über GetMap-Parameter

Anfrage Parameter Beispiel	Verpflichtend/Optional	Beschreibung
VERSION=1.1.1	verpflichtend	Version des WMS Servers
REQUEST=GetMap	verpflichtend	Request-Typ (GetCapabilities GetMap GetFeatureInfo)
LAYERS=layer_list	verpflichtend	Kommaseparierte Liste mit einem oder mehreren Kartenlayern
STYLES=style_list	verpflichtend	Kommaseparierte Liste mit einer Zeichenvorschrift pro angefragtem Kartenlayer
SRS=namespace:identifizier1	verpflichtend	Räumliches Bezugssystem
BBOX=minx,miny,maxx,maxy	verpflichtend	Untere linke und obere rechte Ecke des Kartenausschnittes
WIDTH=output_width	verpflichtend	Breite des Kartenfensters in Pixeln
HEIGHT=output_width	verpflichtend	Höhe des Kartenfensters in Pixeln
FORMAT=supported_formats	verpflichtend	Ausgabeformat der Karte (z.B. png, jpg)
TRANSPARENT=TRUEIFALSE	optional	Hintergrundtransparenz
BGCOLOR=color_value	optional	Hexadezimaler RGB Farbwert des Hintergrundes

Quelle: GREWE 2006, S.30

Weitere GetMap-Parameter sind in der Web Map Service Implementation Specification zu finden.¹⁰

In der Praxis werden Sie feststellen, dass Sie beim UMN MapServer nicht immer alle "verpflichtenden" Parameter angeben müssen, um eine Karte zu produzieren. Defaultmäßig werden viele Werte belegt.

Auch eine Anfrage der Art von:

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wms_ogc/wms_server.map&SERVICE=WMS&VERSION=1.1.0&REQUEST=GetMap&layers=landkreis_export,testdaten_export

Achtung: Dies gilt nur bis Mapserver 4.10.2

Ab MapServer 5.0 MÜSSEN !!! Zwingend alle verpflichtenden Parameter (siehe S.9) angegeben werden!!!

liefert schon ein Ergebnis.

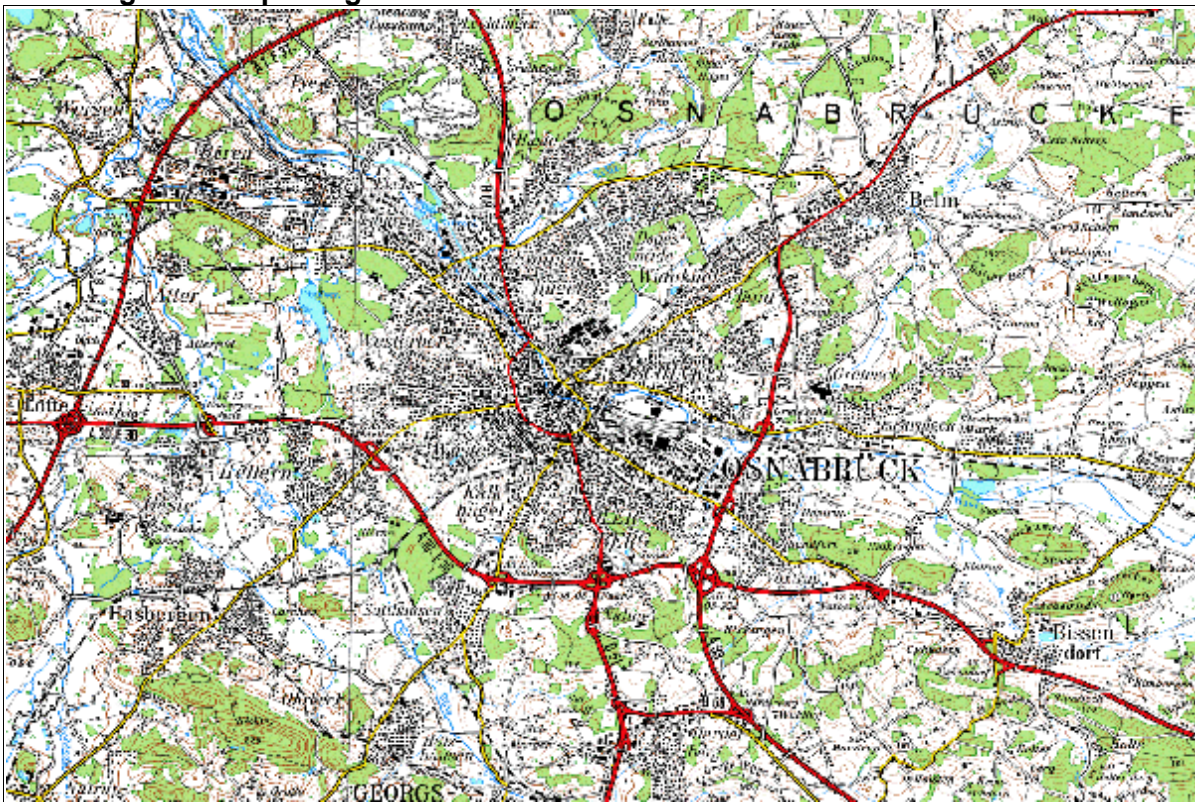
Es ist jedoch mit Sicherheit kein Nachteil, sich an die Vorgaben der WMS Implementation Specification zu halten.

Machen wir noch mal einen Test mit dem WMS der LGN.

Geben Sie mal ein:

<http://www.mapserver.niedersachsen.de/freezeogc/mapserverogc?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:31467&BBOX=3425700,5787300,3446400,5800500&layers=TK100&WIDTH=600&Height=400&FORMAT=image/png>

Abbildung 6: GetMap-Ausgabe des LGN-Servers



¹⁰ Open GIS Consortium (b), S.33

Diese Karte wird, wie gesagt, zwischen den Maßstäben 1:32000 und 1:250000 angezeigt, es dauert oft ein wenig, bis man das richtige Verhältnis zwischen Bounding Box und Höhe/Breite der Karte gefunden hat.

1.5.1 Veränderung einer GetMap-Ausgabe mittels SLD

Mittels SLD (Styled Layers Descriptors) können Sie auch externe WMS in einem gewissen Rahmen umgestalten.

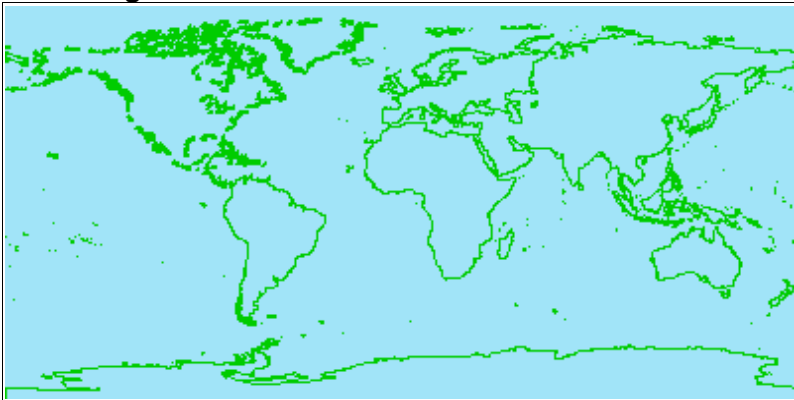
An dieser Stelle soll auf SLD nur am Rande eingegangen werden. Für einen tieferen Einstieg sei nachfolgender Link empfohlen: <http://mapserver.gis.umn.edu/docs/howto/sldhowto>

Rufen Sie mal:

http://www2.dmsolutions.ca/cgi-bin/mswms_world?SERVICE=WMS&VERSION=1.1.1&Request=GetMap&Layers=WorldGen_Outline&

auf, folgendes Bild erscheint:

Abbildung 7: Weltkarte



Wie auf Seite 2 erwähnt kann man hier auch den **GetStyles-Befehl** absenden, also:

http://www2.dmsolutions.ca/cgi-bin/mswms_world?SERVICE=WMS&VERSION=1.1.1&Request=GetStyles&Layers=WorldGen_Outline&

Als Ergebnis erhält man ein SLD-Dokument. Nachfolgend ein Auszug:

```
<Rule>
<ogc:Filter><ogc:PropertyIsEqualTo><ogc:PropertyName>NA3DESC</ogc:PropertyName><ogc:Literal>North
America</ogc:Literal></ogc:PropertyIsEqualTo></ogc:Filter>
<LineSymbolizer>
<Stroke>
<CssParameter name="stroke">#00cc00</CssParameter>
<CssParameter name="stroke-width">2</CssParameter>
<CssParameter name="stroke-dasharray">10 5 5 10 </CssParameter>
</Stroke>
</LineSymbolizer>
</Rule>
```

Wenn Sie in diesem SLD-Dokument Parameter verändern dann können Sie auch das Aussehen der Karte umgestalten. In diesem Falle nutzen wir dafür folgendes SLD-Dokument:

```
<StyledLayerDescriptor version="1.0.0" xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengespatial.net/sld/1.0.0/StyledLayerDescriptor.xsd">
<NamedLayer>
<Name>WorldGen_Outline</Name>
<UserStyle>
<FeatureTypeStyle>
<Rule>
<LineSymbolizer>
-<Stroke>
<CssParameter name="stroke">#0000ff</CssParameter>
<CssParameter name="stroke-width">10</CssParameter>
</Stroke>
</LineSymbolizer>
</Rule>
</FeatureTypeStyle>
</UserStyle>
</NamedLayer>
</StyledLayerDescriptor>
```

Dieses SLD-Dokument sorgt dafür, dass alle Linien blau und breit dargestellt werden. Layouttechnisch sicherlich nicht optimal, aber es geht an dieser Stelle ja auch eher darum zu zeigen, dass man externe WMS mit eigenen Daten verändern kann.

Theoretisch gibt es 2 Möglichkeiten ein SLD-Dokument einzubinden.

- Entweder über eine fürchterlich lange URL und den SLD_BODY-Parameter, z.B. so:

http://www2.dmsolutions.ca/cgi-bin/mswms_world?SERVICE=WMS&VERSION=1.1.1&Request=GetMap&Layers=WorldGen_Outline&SLD_BODY=<StyledLayerDescriptor version='1.0.0' xmlns='http://www.opengis.net/sld' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xsi:schemaLocation='http://www.opengis.net/sld http://schemas.opengespatial.net/sld/1.0.0/StyledLayerDescriptor.xsd'><NamedLayer><Name>WorldGen_Outline</Name><UserStyle><FeatureTypeStyle><Rule><LineSymbolizer><Stroke><CssParameter name='stroke'>%230000ff</CssParameter><CssParameter name='stroke-width'>10</CssParameter></Stroke></LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></NamedLayer></StyledLayerDescriptor>

Was natürlich sehr unbequem ist

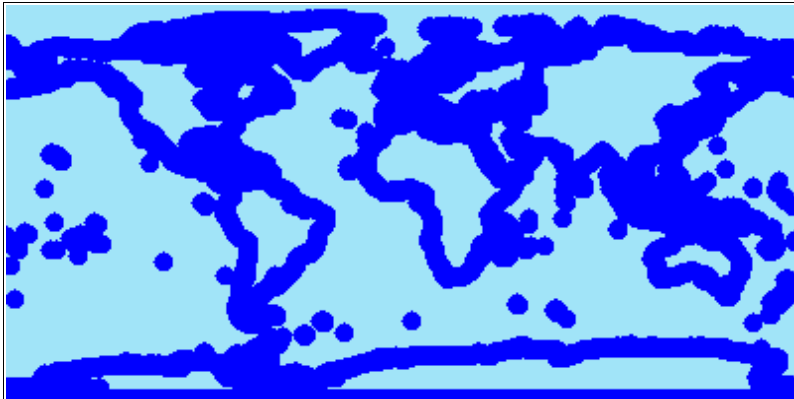
(Achtung:<CssParameter name="stroke">#0000ff</CssParameter> muss dann umgewandelt werden in: <CssParameter name="stroke">%23ff0000</CssParameter>).

- Über den einfachen SLD-Parameter mit remote-URL

http://www2.dmsolutions.ca/cgi-bin/mswms_world?SERVICE=WMS&VERSION=1.1.1&Request=GetMap&Layers=WorldGen_Outline&SLD=http://www.selbstverwaltung-bundesweit.de/line_worldgenoutline.xml

Das Ergebnis ist jeweils das gleiche (Abb.8):

Abbildung 8: Weltkarte mit SLD-verändert



1.6 GetFeatureInfo

Über den GetFeatureInfo-Aufruf lassen sich Informationen zu Geo-Objekten der Karte (Features) darstellen, jedoch nur dann, wenn der entsprechende Layer über das Attribut `queryable="1"` verfügt (siehe auch Abb. 2). Den Layer eines Mapfiles machen Sie ja abfragbar, wenn in der Layer-Sektion ein HTML-Abfragetemplate definiert ist. Bei unserem selbst erstellten WMS ist dieses bei dem Layer „landkreis_export“ der Fall.

Es werden die Attribute der Koordinatenstelle im Kartenbild präsentiert, welche mit X und Y angegeben wird.

Ein Beispiel:

Achtung: Ab MapServer 5.0 müssen alle verpflichtenden WMS-Parameter angegeben werden (siehe S.9) !!!!

http://localhost/cgi-bin/mapserv?map=/mapservertest/wms_ogc/wms_server.map&VERSION=1.1.1&SERVICE=WMS&REQUEST=getFeatureInfo&layers=landkreis_export&BBOX=3430018,5876533,3491970,5948485&Query_Layers=landkreis_export&WIDTH=500&Height=500&X=250&Y=250&FEATURE_COUNT=30&INFO_FORMAT=text/plain&

Folgende Infos werden dann im Browser ausgegeben:

Abbildung 9: Ergebnisse einer GetFeatureInfo-Abfrage im text/plain-Format

```
GetFeatureInfo results:
Layer 'landkreis_export'
  Feature 7:
    DEUMUNIC_I - '2166'
    ID - '03461008'
    NAME - 'Ovelgönne'
    AREA - '123.84'
    POPDENKM - '45.00'
    POPTOTAL - '5623.00'
    PO_M_T_95 - '2863.00'
    PO_F_T_95 - '2760.00'
```

Aber Achtung: Bis zur MapServer-Version 4.8 wurden **sämtliche** Attribute (welche ja in der dbf-Datei des Shapes oder aber z.B. in einer Geodatenbank liegen) standardmäßig ausgegeben.

Seit Version 4.10.0 muss hierfür im entsprechenden Layer (hier also "landkreis_export") der Eintrag:

'wms_include_items' 'all' vorliegen.

Tabelle 1 auf der nächsten Seite zeigt, welche Parameter möglich sind.

Tabelle 1: GetFeatureInfo-Parameter

Request Parameter	verpflichtend/optional	Beschreibung
VERSION=version	v	Versionsangabe
REQUEST=GetFeatureInfo	v	Request-Beschreibung
Query_LAYERS=Layernamen-gaben	v	Layernamen, welche abgefragt werden sollen. Werden durch Komma unterteilt
INFO_FORMAT=output_format	o	Legt das Ausgabeformat der Daten fest
X=Pixelspalte	v	X-Koordinate (in Pixeln) des Geoobjektes, wird von oben links gemessen
Y=Pixelangabe	v	Y-Koordinate (in Pixeln) des Geoobjektes, wird von oben links gemessen
EXCEPTIONS=Exception-Format	o	Legt das Format für Fehlermeldungen fest (standard: application/vnd.ogc.se_xml)
Anbieterspezifische Parameter	o	Experimentelle Parameter

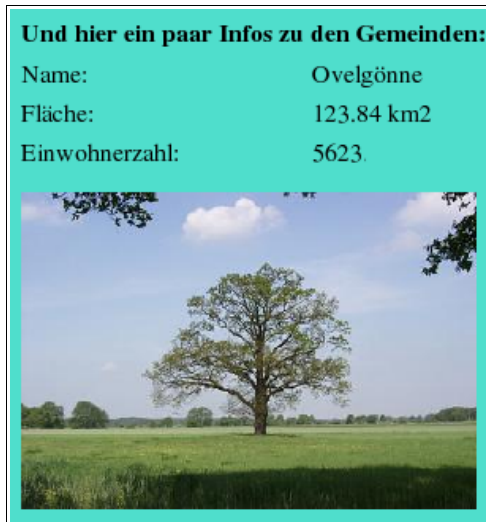
Quelle: Nach WMS Specification, S.40

Wenn Sie nun das Ausgabeformat auf text/html festlegen und auch ein HTML-Template erstellt und im Mapfile angegeben haben, dann können Sie auch einmal diesen Aufruf ausprobieren:

http://localhost/cgi-bin/mapserv?map=/mapserver/wms_ogc/wms_server.map&VERSION=1.1.1&SERVICE=WMS&REQUEST=getFeatureInfo&layers=landkreis_export&BBOX=3430018,5876533,3491970,5948485&Query_Layers=landkreis_export&WIDTH=500&Height=500&X=250&Y=250&FEATURE_COUNT=30&INFO_FORMAT=text/html&

Das Ergebnis im Browser dann wie folgt aus:

Abbildung 10: Ergebnisse einer GetFeatureInfo-Abfrage im text/html-Format



Folgendes HTM-Template wird verwendet:

```
<body bgcolor=#4fdecc><table><tr><td colspan="2"><b>
Und hier ein paar Infos zu den Gemeinden:</b></td></tr>
<tr><td>
Name:</td><td>[NAME]</td></tr>
<tr><td>Fläche:</td><td>[AREA] km2</td></tr><tr>
<td>
Einwohnerzahl: </td><td>[POPTOTAL]</td></tr>
<tr><td colspan="2"></img></td></tr></table>
</body>
```

Exkurs A: GetFeatureInfo im Mapbender



Anhand eines Beispiels im Mapbender soll einmal der Nutzen von GetFeatureInfo aufgezeigt werden.

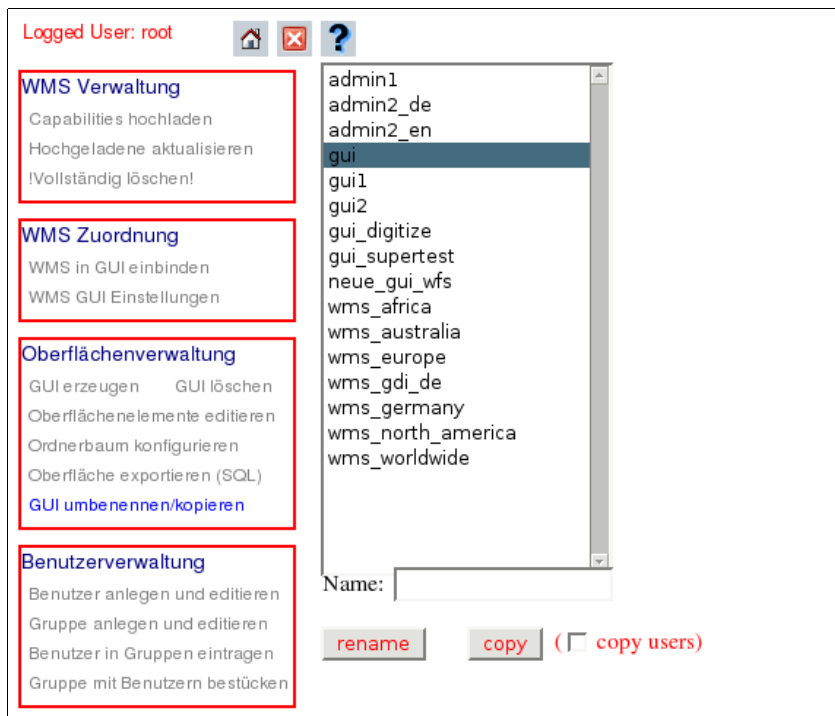
Es wird davon ausgegangen, dass Sie sich schon ein wenig mit dem Mapbender auseinandergesetzt haben.

Falls nicht, hier die Basisschritte:

Gehen Sie in der Mapbender GUI List auf “admin2_de”.

Im Bereich “Oberflächenverwaltung” gehen Sie auf “Gui umbenennen/kopieren” und kopieren hier einmal die “gui”, nennen Sie diese z.B. “gui_wesermarsch”.

Abbildung 11: Gui kopieren/umbenennen



Anschließend entfernen Sie erstmal alle vorhandenen WMS dieser gui.

Dieses machen Sie, indem Sie in der Sparte “WMS Zuordnung” auf “WMS Gui Einstellungen” gehen, die WMS auswählen und den remove-Button klicken.

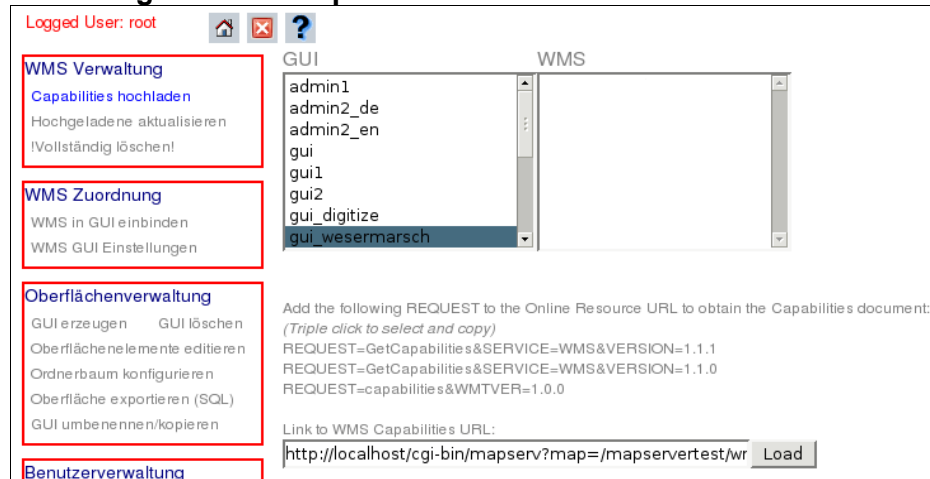
Dann müssen Sie im Bereich WMS-Verwaltung die Capabilities hochladen.

Gehen Sie auf die „gui_wesermarsch“ und geben Sie den Pfad zu Ihrem Mapfile ein, natürlich mit dem Request=GetCapabilities (siehe Abb. 12),

also z.B.:

http://localhost/cgi-bin/mapserv?map=/mapserverumgebung/wms_ogc/wms_server.map&REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.1

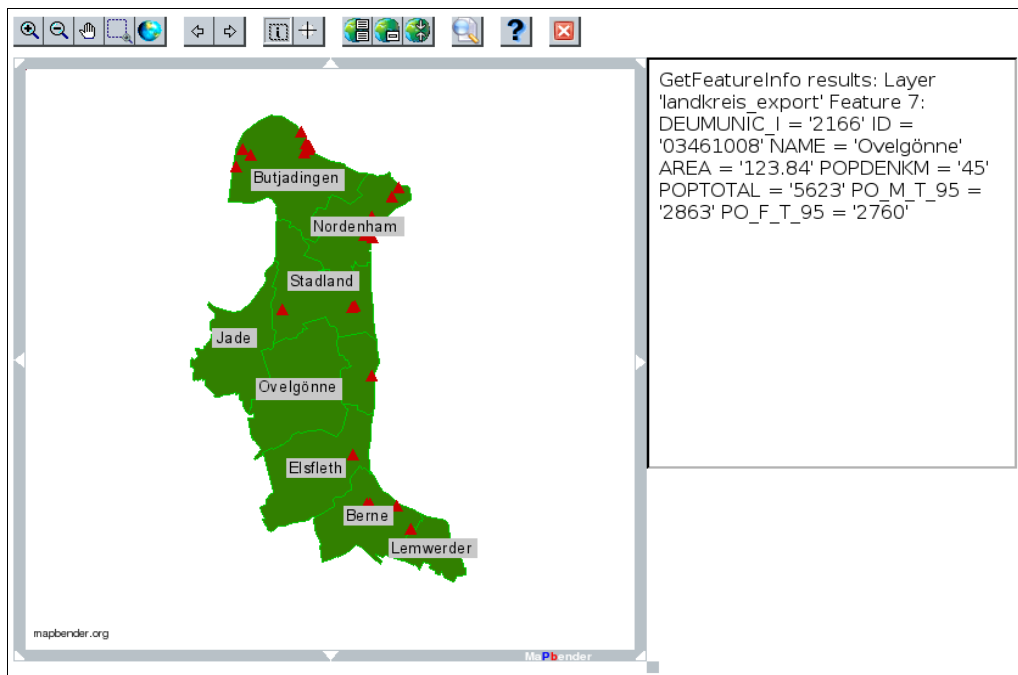
Abbildung 12: WMS-Capabilities hochladen



Wenn das Hochladen der Capabilities geklappt hat öffnen Sie mal die “gui_wesermarsch” und klicken Sie auf den “info”-Button. Die Attribute einer Fläche werden dann im text/plain-Format dargestellt (Abb. 13).

Schon ziemlich cool, aber noch nicht optimal (schließlich wollen wir das optisch noch besser darstellen). Wir brauchen also das `INFO_FORMAT=text/html`.

Abbildung 13: GetFeatureInfo-Ausgabe im text/plain-Format



Damit HTML-Infos auch im Mapbender dargestellt werden muss hierfür im METADATA-Bereich des Mapfiles der Eintrag: 'WMS_FEATURE_INFO_MIME_TYPE' 'text/html' vorhanden sein, also:

METADATA

```
'WMS_TITLE'          'Wesermarsch WMS'  
'WMS_ABSTRACT'       'Dies ist ein einfaches Beispiel fuer einen WMS'  
'WMS_FEATURE_INFO_MIME_TYPE' 'text/html'  
'WMS_ONLINERESSOURCE' http://127.0.0.1/cgi-  
bin/mapserv?map=/mapserver/test/wms_wfs/wfs_hope/wms_server.map'  
'WMS_SRS'            'epsg:31467'  
END
```

Die Mapbender-Verwaltungsebene hat die sehr angenehme Eigenschaft, dass man bequem auf das Format "text/html" umschalten kann.

Unter „WMS Gui Einstellungen“ können Sie GUI und WMS auswählen und das Infoformat festlegen (Abb. 14).

Abbildung 14: Umstellung auf text/html

Logged User: root

WMS Verwaltung
Capabilities hochladen
Hochgeladene aktualisieren
Vollständig löschen!

WMS Zuordnung
WMS in GUI einbinden
WMS GUI Einstellungen

Oberflächenverwaltung
GUI erzeugen GUI löschen
Oberflächenelemente editieren
Ordnerbaum konfigurieren
Oberfläche exportieren (SQL)
GUI umbenennen/kopieren

Benutzerverwaltung
Benutzer anlegen und editieren
Gruppe anlegen und editieren
Benutzer in Gruppen eintragen
Gruppe mit Benutzern bestücken

Benutzerzugriff erteilen
Einem Nutzer Zugriff auf mehrere Oberflächen erteilen
Oberfläche mehreren Benutzern erteilen

GUI: admin1, admin2_de, admin2_en, gui, gui1, gui2, gui_digitize, **gui_supertest**

WMS-TITLE: 0 - Wesermarsch WMS

up, down, remove

save

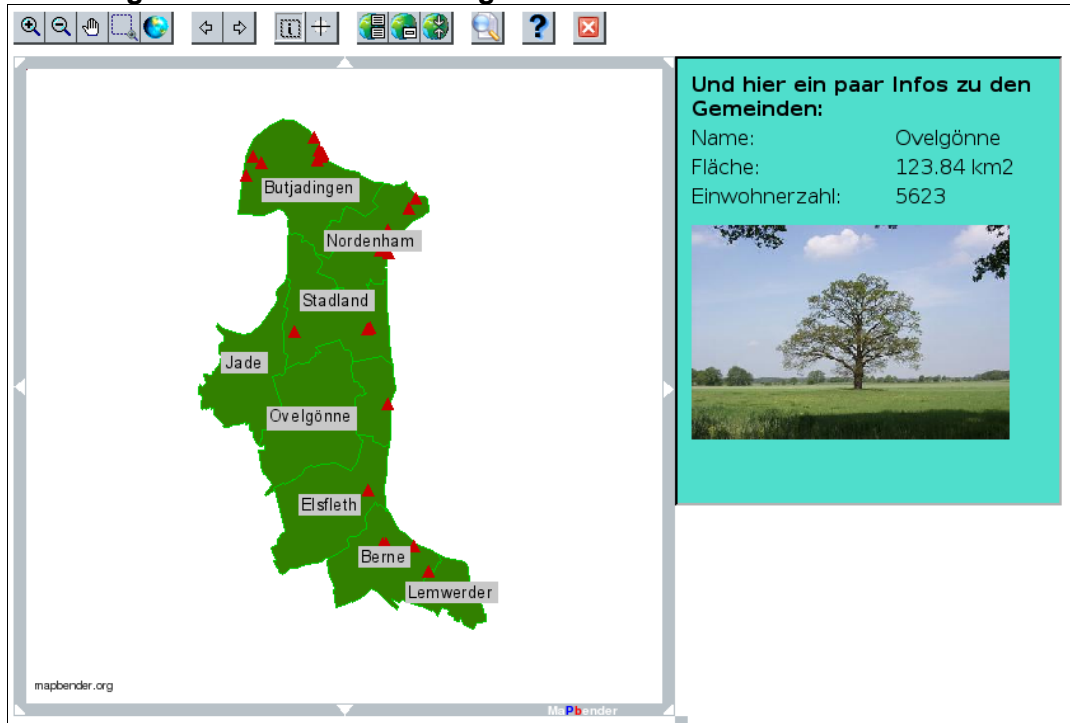
LINK: Capabilities WMS ID: 912

EPSG: EPSG:31467
Mapformat: image/png
Infoformat: text/html
Exceptionformat: application/vnd.ogc.se_in
Visibility: visible

Nr.	ID	Parent	Name	Title	on/off	sel	sel_default	info	info_default	minScale	maxScale
1	191		Testkarte	Wesermarsch V	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0
1	191	0	landkreis_	landkreis_expoi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0	0

Eine Ausgabe im HTML-Format sieht z.B. dann ungefähr so wie in Abb.15 aus (je nachdem wie Sie das Layout definieren).

Abbildung 15: GetFeatureInfo-Ausgabe im text/html-Format



1.7 WMS-Layer in den UMN MapServer als WMS-Client einbinden

Die Ausgabe von WMS-Karten über die Angabe einer URL ist immer recht umständlich. Sie können den UMN MapServer auch als Klienten für die Einbindung von externen WMS-Layern benutzen. Nachfolgend das Mapfile (auch dieses liegt im Ordner wms_ogc (wms_client.map)).

Zusätzlich soll die Anleitung unter

http://mapserver.gis.umn.edu/docs/howto/wms_client empfohlen werden.

```
MAP
NAME      'Testkarte'
STATUS ON
IMAGETYPE jpeg
IMAGECOLOR 255 255 255

EXTENT    3430018 5876533 3491970 5948485
UNITS     METERS
SIZE      500 500
SHAPEPATH '/mapservertest/wms_ogc/data'
SYMBOLSET '/mapservertest/wms_ogc/symbols/symbols.sym'
FONTSET   '/mapservertest/wms_ogc/fonts/fonts.list'
```

WEB

```
    TEMPLATE './template.html'
    IMAGEPATH '/var/www/htdocs/tmp/'
    IMAGEURL  '/tmp/'
```

END

```
PROJECTION
  'init=epsg:31467'
END
```

```
LAYER
NAME 'BRD'
TYPE RASTER
STATUS DEFAULT
CONNECTION 'http://wms1.ccgis.de/cgi-
bin/mapserv?map=/data/umn/germany/germany.map'
CONNECTIONTYPE WMS
METADATA
  'wms_srs' 'EPSG:31467'
  'wms_name' 'Germany'
  'wms_format' 'image/png'
  'wms_server_version' '1.1.1'
```

END
END

LAYER
NAME 'TK100'
TYPE RASTER
STATUS DEFAULT
CONNECTION 'http://www.mapserver.niedersachsen.de/freezoneogc/mapserverogc?'
CONNECTIONTYPE WMS

METADATA
'wms_server_version' '1.1.1'
'wms_name' 'TK100'
'wms_srs' 'EPSG:31467'
'wms_format' 'image/png'
END
END

LAYER
NAME 'Wesermarsch'
TYPE RASTER
STATUS DEFAULT
CONNECTION 'http://127.0.0.1/cgi-
bin/mapserv?map=/mapservertest/wms_ogc/wms_server.map'
CONNECTIONTYPE WMS

METADATA
'wms_server_version' '1.1.1'
'wms_name' 'landkreis_export'
'wms_srs' 'EPSG:31467'
'wms_format' 'image/png'
END
END

LAYER
NAME 'Testdaten'
TYPE RASTER
STATUS DEFAULT
CONNECTION 'http://localhost/cgi-
bin/mapserv?map=/mapservertest/wms_ogc/wms_server.map'
CONNECTIONTYPE WMS

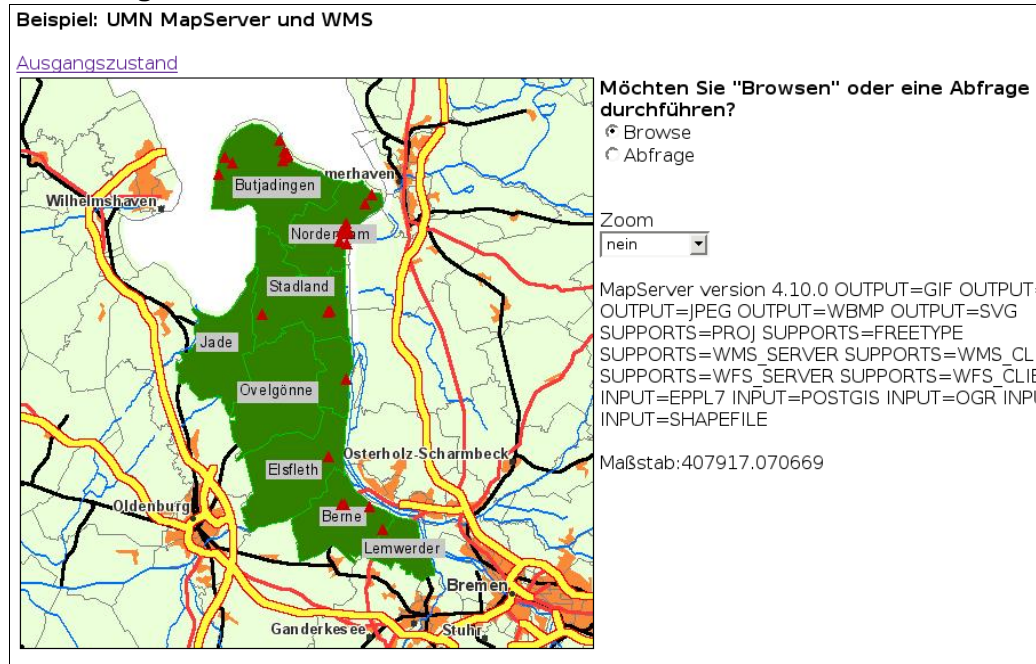
METADATA
'wms_server_version' '1.1.1'
'wms_name' 'testdaten_export'
'wms_srs' 'EPSG:31467'
'wms_format' 'image/png'

END
END
END

Bei den Einträgen im Mapfile sehen Sie erneut, dass Sie beim UMN MapServer nicht alle in Abbildung 5 vorgeschriebenen Parameter setzen müssen.

Wenn Sie nämlich nun in Ihrem Browser das Mapfile "wms_client.map" aufrufen (http://localhost/cgi-bin/mapserv?map=/mapserver/test/wms_ogc/wms_client.map), dann erhalten Sie folgendes Bild:

Abbildung 16: Mehrere WMS in einem Client



An dieser Stelle werden die externen WMS-Layer Germany, TK100, landkreis_export und testdaten_export eingebunden.

Landkreis_export und testdaten_export ließen sich jetzt natürlich auch ohne WMS-Verbindung einbinden (die Daten liegen ja lokal auf Ihrem Rechner); die Verbindung über

CONNECTION 'http://127.0.0.1/cgi-bin/mapserv?map=/mapserver/test/wms_ogc/wms_server.map'

dürfte das Prinzip der WMS-Einbindung aber besser verdeutlichen.

Wenn der Rechner nun eine IP hat, welche auch "von außen" ansprechbar ist, dann könnten Sie die Wesermarschdaten nun auch anderen Nutzern im WWW zur Verfügung stellen.

2. WFS -Web Feature Service

“Steigen die Ansprüche an die Funktionalität und den Automatisierungsgrad einer Web-GIS-Lösung, so kann es interessant sein, neben der Fähigkeit zur Kartendarstellung auch die Analysefähigkeiten eines GIS im Web zu nutzen und statt auf digitale Karten auch auf die zugrunde liegenden Geodaten mittels Internet-Technologie zuzugreifen. Exakt diese Möglichkeit eröffnet ein so genannter Web Feature Service(...).”¹¹

Während ein WMS also nur eine Rasterkarte an den Clienten liefert (natürlich auch mit Abfrageoptionen durch GetFeatureInfo) bietet ein WFS deutlich mehr. *“The server sends back geographic coordinate data such as line, point, or polygon features(...)you’ll like WFS because it offers more flexibility.”¹²*

Letztlich liefert ein WFS **Geodaten** (Vektordaten inkl. Attribute) , welche natürlich auch visualisiert werden können.

Wie Sie sehen werden bietet ein WFS thematische und räumliche Selektionsmöglichkeiten, was ein entscheidender Unterschied zu einem WMS ist.

Mit den Daten eines WFS kann man letztlich viel mehr machen, als mit einem Rasterbild von einem WMS:¹³

- Tooltipmöglichkeit
- Highlightfunktion
- räumliche Analysen
- umfangreiche Filter
- Verschneidungen mit anderen Datensätzen
- hoch aufgelöste Ausgaben
- Suche über Parameter (Straßenname, Objekt-ID, etc.)
- editieren der Daten (WFS-T)
- Weiterverarbeitung, Weiterleitung der Daten

Nach der Web Feature Service Implementation Specification müssen 2 Klassen von WFS unterschieden werden:

- Basic WFS (hierzu gehört auch der UMN MapServer)
*“A basic WFS would implement the GetCapabilities, DescribeFeatureType and GetFeature operations.
This would consider a READ-ONLY web feature service.”¹⁴*
- Transaction WFS (hierzu gehört z.B. Geoserver (<http://geoserver.org>))
*“A transaction web feature Service would support all the operations of a basic web feature service and in addition it would implement the Transaction operation.
Optionally, a transaction WFS could implement the LockFeature operation.”¹⁵*

¹¹ Donaubaue 2005, S.93

¹² Mitchell 2005, S.228

¹³ <http://intevation.de/pipermail/mapserver-de/2007-February/002869.html>
<http://intevation.de/pipermail/mapserver-de/2007-February/002870.html>

¹⁴ Open GIS Consortium (a), S.3

¹⁵ ebda.

Bedeutet konkret, dass ein Transaction-WFS auch schreibenden Zugriff auf Features zulässt. Features können neu erstellt werden, Eigenschaften können geändert oder Features komplett gelöscht werden.

Die LockFeature Operation ermöglicht es, dass Features mit einer Veränderungssperre belegt werden.

Da der UMN MapServer ein "Basic WFS" ist werden in der Folge die Operationen

- GetCapabilities
- DescribeFeatureType
- GetFeature

beschrieben.

Anders als beim WMS, welcher als Datenquellen Raster- oder Vektordaten akzeptiert, werden beim WFS nur Vektordaten verwandt.

2.1 WFS-Server mit dem UMN MapServer

Zunächst erstellen wir einmal ein passendes Mapfile (liegt auch im Ordner wfs_ogc (wfs_client.map)).

Als zusätzliche Hilfe soll auch auf die Seite

http://mapserver.gis.umn.edu/docs/howto/wfs_server

verwiesen werden.

Nachfolgend das Mapfile:

```
MAP
NAME          Testkarte
IMAGETYPE     PNG
EXTENT        3430018 5876533 3491970 5948485
UNITS         METERS
#DEBUG ON
PROJECTION
    'init=epsg:31467'
END
SIZE 500 500
SHAPEPATH     '/mapservertest/wfs_ogc/data'
SYMBOLSET     '/mapservertest/wfs_ogc/symbols/symbols.sym'
FONTSET       '/mapservertest/wfs_ogc/fonts/fonts.list'
IMAGECOLOR    200 200 200
```

Grundeinstellungen-----

```
WEB
    TEMPLATE './template.html'
    IMAGEPATH '/var/www/htdocs/tmp/'
    IMAGEURL  '/tmp/'
```

```

METADATA
"WFS_TITLE" "WFS-Test"          #zwingend notwendig
"WFS_SRS" "epsg:31467"          #dringend empfohlen
"WFS_ABSTRACT" "Ein kleiner WFS-Test" #optional
"WFS_ONLINERESOURCE" "http://localhost/cgi-
bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map" # dringend empfohlen
END
END

```

```

LAYER
NAME      Gemeindelayer
TYPE      POLYGON
STATUS     DEFAULT

```

```

DATA      Lan.shp
DUMP TRUE #zwingend notwendig
METADATA
"WFS_TITLE" "Gemeinde"          #zwingend notwendig
"WFS_SRS" "epsg:31467" #dringend empfohlen
"gml_featureid" "id" #zwingend notwendig (ab MapServer 4.10.0)
"GML_INCLUDE_ITEMS" "all" #optional (liefert alle vorhandenen Parameter aus)
END
CLASS
NAME 'gemeindeklasse'
STYLE
OUTLINECOLOR 200 0 0
COLOR 0 200 0
SIZE 22
END
END
END

```

```

LAYER
NAME      'Testdatenlayer'
TYPE      POINT
TOLERANCE 5
STATUS    ON
DATA 'testdaten.shp'

```

```

DUMP TRUE #zwingend notwendig
METADATA
"WFS_TITLE" "Testdaten" #zwingend notwendig
"WFS_SRS" "epsg:31467" #dringend empfohlen

```

```
"gml_featureid" "id" #zwingend notwendig (ab MapServer 4.10.0)
"GML_INCLUDE_ITEMS" "all" #optional (liefert alle vorhandenen Parameter aus)
END
```

```
CLASS
STYLE
SYMBOL 'triangle'
COLOR 200 0 0
SIZE 9
END
END
END
END
```

Die gelb markierten Bereiche kennzeichnen die Stellen, welche den WFS erstellen.

2.2 WFS-GetCapabilities

Ein Aufruf der Art von:

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.0.0&REQUEST=GetCapabilities

sorgt dafür, dass ein XML-Dokument mit den Eigenschaften des WFS ausgegeben wird.

Abbildung 17: Auszug aus XML-Dokument

```
<Name>Gemeindelayer</Name>
<Title>Gemeinde</Title>
<SRS>epsg:31467</SRS>
<LatLongBoundingBox minx="8.1191" miny="53.0957" maxx="8.65646" maxy="53.732"/>
</FeatureType>
- <FeatureType>
  <Name>Testdatenlayer</Name>
  <Title>Testdaten</Title>
  <SRS>epsg:31467</SRS>
  <LatLongBoundingBox minx="8.24151" miny="53.156" maxx="8.55556" maxy="53.5994"/>
</FeatureType>
```

Wie bei dem GetCapabilities-Aufruf zu einem WMS-Layer werden auch hier Metadaten präsentiert.

2.3 DescribeFeatureType-Abfrage

Geben Sie einmal folgende URL ein:

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.0.0&REQUEST=DescribeFeatureType&TYPENAME=Gemeindelayer&

Erneut wird ein XML-Dokument ausgegeben (siehe Abb. 18).

Abbildung 18: Auszug aus XML-Dokument

```
- <schema targetNamespace="http://mapserver.gis.umn.edu/mapserver" 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="Gemeindelayer" type="ms:GemeindelayerType" substitutionGroup="gml:_Feature"/>
- <complexType name="GemeindelayerType">
- <complexContent>
- <extension base="gml:AbstractFeatureType">
- <sequence>
  <element name="msGeometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="1"/>
  <element name="DEUMUNIC_I" type="string"/>
  <element name="ID" type="string"/>
  <element name="NAME" type="string"/>
  <element name="AREA" type="string"/>
  <element name="POPDENKM" type="string"/>
  <element name="POPTOTAL" type="string"/>
  <element name="PO_M_T_95" type="string"/>
  <element name="PO_F_T_95" type="string"/>
  </sequence>
</extension>
</complexContent>
</complexType>
</schema>
```

Es wird hier die Struktur der Daten beschrieben (z.B. ob String oder Geometry).

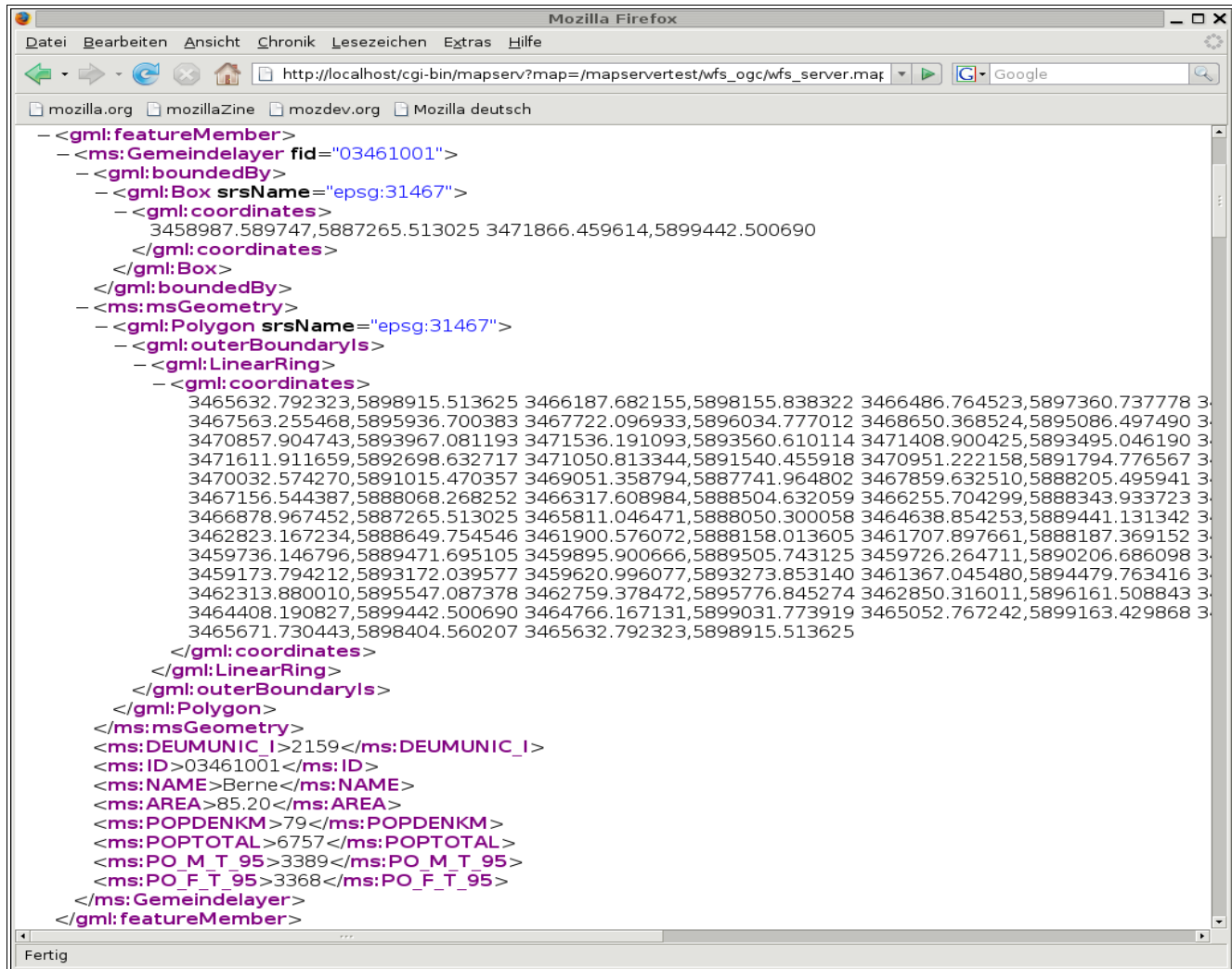
2.4 GetFeature-Abfrage

Ein GetFeature-Aufruf sieht z.B. so aus:

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.0.0&REQUEST=GetFeature&TYPENAME=Gemeindelayer

Der Typename legt hier den abfragbaren Layer fest.

Abbildung 19: Auszug aus XML-Dokument bei GetFeature-Abfrage



Diesmal liefert das XML-Dokument die Geometrien der einzelnen Features (in dem XML-Unterformat GML) sowie Werte einzelner Spalten zu den jeweiligen Features aus.

Im Rahmen des GetFeature-Anfrage können dann auch räumliche oder attribute Filter gesetzt werden.

Siehe hierzu z.B.: <http://mapserver.gis.umn.edu/docs/howto/filterencoding/>

Folgende Parameter sind möglich:¹⁶

Vergleichsoperatoren

PropertyIsEqualTo (=)

PropertyIsNotEqualTo (<>)

PropertyIsGreaterThan (>)

PropertyIsLessThan (<)

PropertyIsLessThanOrEqualTo (<=)

PropertyIsGreaterThanOrEqualTo (>=)

PropertyIsBetween

PropertyIsLike

Logische Operatoren OR, AND, NOT

Räumliche Anfragen BBOX, Dwithin, Intersects

Arithmetische Operatoren

Ein Aufruf derart von

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.0.0&REQUEST=GetFeature&TYPENAME=Gemeindelayer&Filter=<Filter><PropertyIsGreaterThan><PropertyName>AREA</PropertyName><Literal>100</Literal></PropertyIsGreaterThan></Filter>

also mit diesem Filter;

```
<Filter><PropertyIsGreaterThan>
<PropertyName>AREA</PropertyName>
<Literal>100</Literal>
</PropertyIsGreaterThan></Filter>
```

liefert in dem XML-Dokument nur die Features, welche größer als 100 km² sind.

Auch räumliche Abfragen sind möglich, z.B.

http://localhost/cgi-bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.0.0&REQUEST=GetFeature&TYPENAME=Gemeindelayer&Filter=<Filter><BBOX><PropertyName>Name>NAME</PropertyName><Box srsName='EPSG:31467'><coordinates>3456500,5901700 3481900,5882300</coordinates></Box></BBOX></Filter>

In diesem Fall schneidet die Boundix-Box Elsfleth, Berne und Lemwerder. Nur zu diesen Gemeinden werden dann Daten ausgegeben.

Der Filter dafür also:

```
<Filter><BBOX>
<PropertyName>Name>NAME</PropertyName>
<Box srsName='EPSG:31467'>
<Coordinates>3456500,5901700 3481900,5882300</Coordinates>
</Box></BBOX>
</Filter>
```

¹⁶ Emde 2005

2.5 UMN MapServer als WFS-Client

siehe dazu auch hier: http://mapserver.gis.umn.edu/docs/howto/wfs_client

Hier nun ein Beispiel für ein Mapfile als WFS-Client (wfs_client.map).

```
MAP
NAME          Testkarte
IMAGETYPE     PNG
EXTENT        3430018 5876533 3491970 5948485
UNITS         METERS
#DEBUG ON
PROJECTION
    'init=epsg:31467'
END
SIZE 500 500
SHAPEPATH     '/mapservertest/wfs_ogc/data'
SYMBOLSET     '/mapservertest/wfs_ogc/symbols/symbols.sym'
FONTSET       '/mapservertest/wfs_ogc/fonts/fonts.list'
IMAGECOLOR    200 200 200

# Grundeinstellungen-----

WEB
    TEMPLATE './template.html'
    IMAGEPATH '/var/www/htdocs/tmp/'
    IMAGEURL  '/tmp/'
END
LAYER
    NAME      Gemeindelayer
    TYPE      POLYGON
    STATUS    DEFAULT

    CONNECTIONTYPE WFS #zwingend notwendig
    CONNECTION 'http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_server.map' #zwingend #notwendig
    METADATA
        "WFS_VERSION" "1.0.0" #zwingend notwendig
        "WFS_TYPERNAME" "Gemeindelayer" #zwingend notwendig
        "WFS_FILTER" "<PropertyIsGreaterThan><PropertyName>AREA</PropertyName><Literal>100</Literal></PropertyIsGreaterThan>" #optional, Hier ein Beispiel für #einen Filter
    END
```

```

        CLASS
            STYLE
                OUTLINECOLOR 255 0 0
                COLOR 0 0 250
            END
        END
    END
END

```

LAYER

```

NAME      'Testdatenlayer'
TYPE      POINT
TOLERANCE 5
STATUS    DEFAULT

```

```

CONNECTIONTYPE WFS #zwingend notwendig
CONNECTION 'http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_server.map' #zwingend notwendig
METADATA
"WFS_VERSION" "1.0.0" #zwingend notwendig
"WFS_TYPERNAME" "Testdatenlayer" #zwingend notwendig
"WFS_REQUEST_METHOD" "GET" #Eigentlich optional, Defaultmäßig wird die
                             #"POST"-Methode angewandt, in diesem Beispiel
                             #gab es aber Schwierigkeiten, wenn NICHT mit
                             #"GET" gearbeitet wird, eventuell ein Bug in der
                             #Version 4.10.0 (??).
END

```

```

CLASS
STYLE
SYMBOL 'square'
COLOR 200 0 0
SIZE 9
END
END
END
END

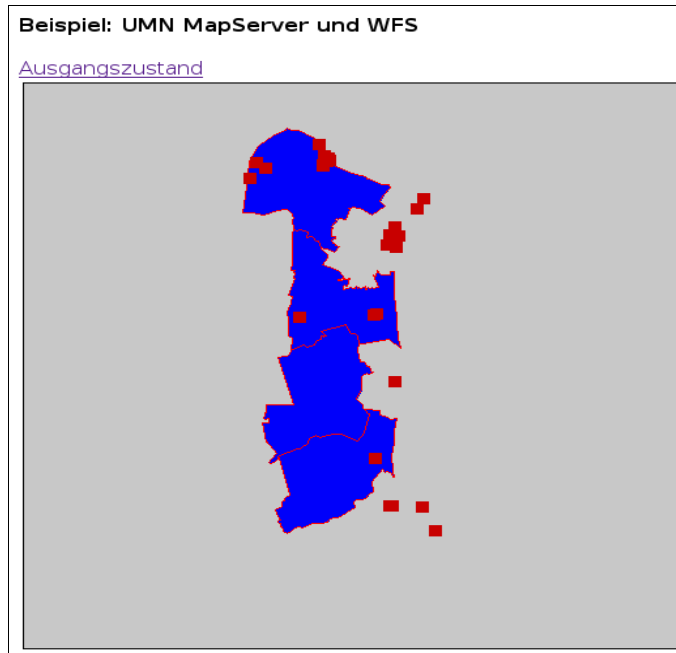
```

Als Ergebnis des Aufrufs

http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_client.map

wird folgendes Bild geliefert:

Abbildung 20: Ausgabe eines WFS mit Filter



Im Gegensatz zu einem “simplen” WMS, welcher nun die gesamte Wesermarsch analog zu den Werten des WMS-Servers geliefert hätte (in einem hellen grün) haben wir hier die Möglichkeit, die Farbe der Polygone zu ändern und über den Filter zu sagen, dass wir nur Gemeinden mit einer Größe > 100 km² haben wollen.

Exkurs B: Einsatz einer WFS-Suche im Mapbender

Mapbender

Im Mapbender ist eine WFS-Suche integriert. Diese funktioniert über das Modul `mod_wfs_gazetter.php`.

Hilfreich hierzu auch das WIKI: http://www.mapbender.org/index.php/WFS_Konfiguration

Der einfachste Weg um “auf die Schnelle” dieses Modul z.B. an eine Gemeindesuche anzupassen sieht wie folgt aus:

1. Erst einmal die “gui” kopieren und umbenennen sowie WMS-Capabilities hochladen (siehe Seite 18-19).
2. WFS-Capabilities hochladen. Gehen Sie hierfür in der Mapbender GUI List auf den Bereich “admin1” und dann auf “Load-WFS”.

Laden Sie hier die Capabilities Ihres eigenen WFS in eine gui (hier dann „gui_wesermarsch“), in diesem Fall:

http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.0.0&REQUEST=GetCapabilities&

Abbildung 21: WFS-Capabilities hochladen

Logged User: root

LOAD WMS
LOAD myWMS
UPDATE WMS
EDIT GUI-WMS
DELETE WMS
NEW GUI
DELETE GUI
EDIT GUI-ELEMENTS
GUI -> USER
myGUI -> USER
myGUI -> myUSER
GUI -> GROUP
myGUI -> GROUP
myGUI -> myGROUP
EXPORT GUI
EDIT USER
EDIT myUSER

GUI

admin1
admin2_de
admin2_en
gui
gui1
gui2
gui_digitize
gui_wesermarsch

Add the following REQUEST to the Online Resource URL to obtain the Capabilities document:
(Triple click to select and copy)
REQUEST=getCapabilities&VERSION=1.0.0&SERVICE=WFS

Link to WFS Capabilities URL:
/ICE=WFS&VERSION=1.0.0&REQUEST=GetCapabilities& Load

Bitte beachten Sie: Das Hochladen eines WFS sorgt in dem Mapbender-Clienten nicht dafür, dass Geodaten visualisiert werden, dafür müssen Sie die selben Geodaten als WMS einspielen.

Anschließend gehen Sie im Bereich “admin1” auf “WFS-Conf”, wählen den neu hochgeladenen WFS aus und klicken auf den Radio-Button “Gemeindelayer” (siehe Abb.22).

Abbildung 22: WFS-Konfiguration im Mapbender

Logged User: root

LOAD WMS
LOAD myWMS
UPDATE WMS
EDIT GUI-WMS
DELETE WMS
NEW GUI
DELETE GUI
EDIT GUI-ELEMENTS
GUI -> USER
myGUI -> USER
myGUI -> myUSER
GUI -> GROUP
myGUI -> GROUP
myGUI -> myGROUP
EXPORT GUI
EDIT USER
EDIT myUSER
USER -> GUI
USER -> myGUI
myUSER -> myGUI
USER -> GROUP
USER -> myGROUP
RENAME COPY GUI
myUSER -> myGROUP
EDIT GROUP
EDIT myGROUP
GROUP -> USER
GROUP -> myUSER
myGROUP -> myUSER
GROUP -> GUI
GUI -> myUSER
GUI -> myGROUP
myUSER -> GROUP
myGROUP -> USER

WFS Configuration

[edit WFS Configuration](#)

Select WFS: 7 WFS-Test

ID: 7
Name: MapServer WFS
Title: WFS-Test
Abstract: Ein kleiner WFS-Test
Capabilities: http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_server.map&
FeaturTypes: http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_server.map&
Feature: http://localhost/cgi-bin/mapserv?map=/mapservertest/wfs_ogc/wfs_server.map&

☒ Gemeindelayer
☐ Testdatenlayer

SRS: epsg:31467

Abstract:
Label:
Label_id:
Button:
Button_id:

Style:

Buffer: 1

ResultStyle:

Es öffnen sich verschiedene Eingabefenster (Abb.23).

Abb. 23: WFS-Konfiguration im Mapbender

Logged User: root

1 Mapbender User
8 Gemeinden

GazetterID: 8

Abstract: Gemeinden

Label: Gemeinden:

Label_id: a

Button: Suche abschicken

Button_id: b

Style: `STYLE:body{font-family:Verdana,Arial,sans-serif;font-size: 12px;line-height:2;}`

Buffer: 1

ResultStyle: `Result-Style:.even{color:green;font-family:Verdana,Arial,sans-serif;font-size: 18px;}.uneven{`

WZ-Graphics: ☒

ID	name	type	geom	gid	search	pos	style_id	upper	label	label
218	msGeometry	GeometryPropertyType	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	<input type="checkbox"/>		0
219	DEUMUNIC_I	string	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	<input type="checkbox"/>		0
220	ID	string	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	<input type="checkbox"/>		0
221	NAME	string	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	c	<input type="checkbox"/>	Gemeindena	d
222	AREA	string	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	<input type="checkbox"/>		0
223	POPDENKM	string	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	0	<input type="checkbox"/>		0

Tätigen Sie hier z.B. folgende Angaben (siehe auch:
http://www.mapbender.org/index.php/WFS_Konfiguration).

Abstract: Gemeinden

Label: Gemeinden:

Label_id:a

Button:Suche Abschicken

Button_id:b

STYLE:body{
font-family:Verdana,Arial,sans-serif;
font-size: 12px;
line-height:2;
}

BUFFER:0.01

```

Result-Style:.even{
color:green;
font-family:Verdana,Arial,sans-serif;
font-size: 18px;
}
.uneven{
color:red;
font-family:Verdana,Arial,sans-serif;
font-size: 11px;
}

```

WZ-Graphics:Ja

Dann außerdem in der unteren Tabelle in der Spalte
msGeometry "geom" anklicken

Bei "Name" die Search-Checkbox anklicken und bei pos **1** angeben, bei style_id: **c**

label:**Gemeindename**

label:**d**

show (anklicken)

position:**1**

Achtung: Achten Sie darauf, welche Gazetter-Id vergeben wird (hier 8).

und abspeichern.

Anschließend: gehen Sie in den Bereich „admin2_de“ und wählen bei
„Oberflächenelemente editieren“ die jeweils erstellte gui (hier gui_wesermarsch).

Dann klicken Sie auf das Modul "Gazetteer_mbUser"

Ändern Sie hier z.B. den Comment.

Wichtig: Tauschen Sie den "src-Bereich" aus.

Also anstelle von:

```

../php/mod_wfs_gazetteer.php?sessionID&wfs_conf=1&target=mapframe1,overview&resultFrame=wfsresult

```

geben Sie bei wfs_conf die Zahl Ihres WFS ein, z.B.:

```

../php/mod_wfs_gazetteer.php?sessionID&wfs_conf=8&target=mapframe1,overview&resultFrame=wfsresult

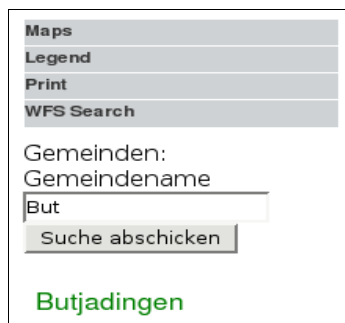
```

und abspeichern.

Rufen Sie nun die "Gui_wesermarsch" auf. Sie brauchen in der WFS-Suche den zu
suchenden Namen nicht ausschreiben, ein Teilstring reicht auch.

Die Suche sollte ungefähr so aussehen:

Abbildung 24: WFS Suche



Maps
Legend
Print
WFS Search

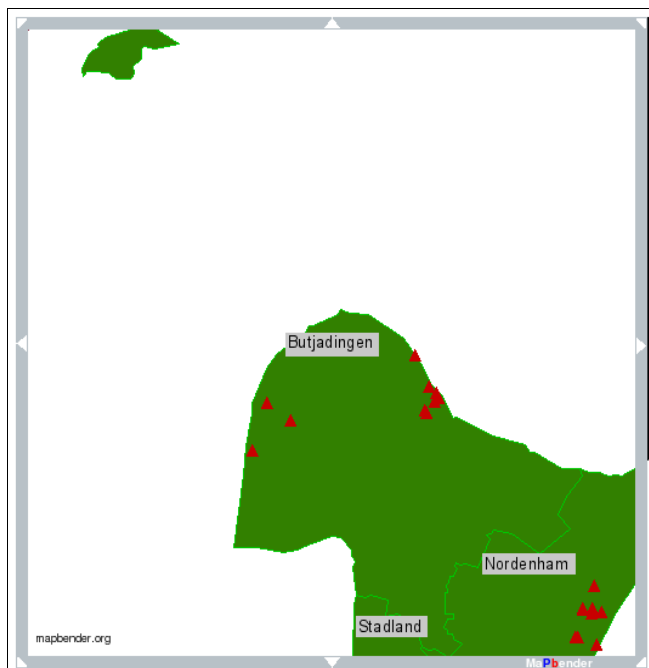
Gemeinden:
Gemeindenname

Butjadingen

Der grün geschriebene String wird ausgegeben, wenn bei der Suche ein Objekt gefunden wird. Wenn Sie diesen String mit dem Mauszeiger überfahren, so wird die Gemeinde highlighted.

Klicken Sie auf den grünen Link, so wird automatisch auf die Gemeinde gezoomt.

Abbildung 25: Zoom nach WFS-Suche



Exkurs C: WFS-Tooltip im Mapbender



Siehe auch hier: <http://www.mapbender.org/index.php/UseMap>

Zunächst führen Sie folgenden SQL-Befehl in der Mapbender-Datenbank aus:

```
INSERT INTO gui_element(fkey_gui_id, e_id, e_pos, e_public, e_comment,
e_element, e_src, e_attributes, e_left, e_top, e_width, e_height,
e_z_index, e_more_styles, e_content, e_closeTag, e_js_file, e_mb_mod,
e_target, e_requires) VALUES('gui_wesermarsch', 'wz_graphics', '0002', '1',
'Module for wfs onmouseover infos', 'div', "", "", '1', '1', '1', '1',
'1', "", "", 'div', '../extensions/wz_jsgraphics.js', "", 'mapframe1',
");
```

Den Namen „gui_wesermarsch“ natürlich gegebenenfalls ändern.

Anschließend führen Sie folgenden SQL-Befehl aus:

```
INSERT INTO gui_element(fkey_gui_id, e_id, e_pos, e_public, e_comment, e_element,
e_src, e_attributes, e_left, e_top, e_width, e_height, e_z_index, e_more_styles, e_content,
e_closeTag, e_js_file, e_mb_mod, e_target, e_requires) VALUES('gui_wesermarsch',
'usemap', '0002', '1', 'clickable map', 'iframe', '../html/mod_blank.html', "", '1', '1', '1', '1', '1',
'visibility:hidden;', "", 'iframe', 'mod_usemap.php', "", 'mapframe1', "");
```

Als nächstes öffnen Sie die Datei:

/var/www/htdocs/mapbender/http/javascripts/mod_usemap.php

In Zeile 21 tragen Sie den Pfad zu Ihrem WFS-Server ein, z.B.:

```
var mod_usemap_wfs = "http://localhost/cgi-
bin/mapserv?map=/mapserver/test/wfs_ogc/wfs_server.map&SERVICE=WFS&VERSION=1.
0.0&REQUEST=GetFeature&TYPENAME=Testdatenlayer";
```

Dann öffnen Sie:

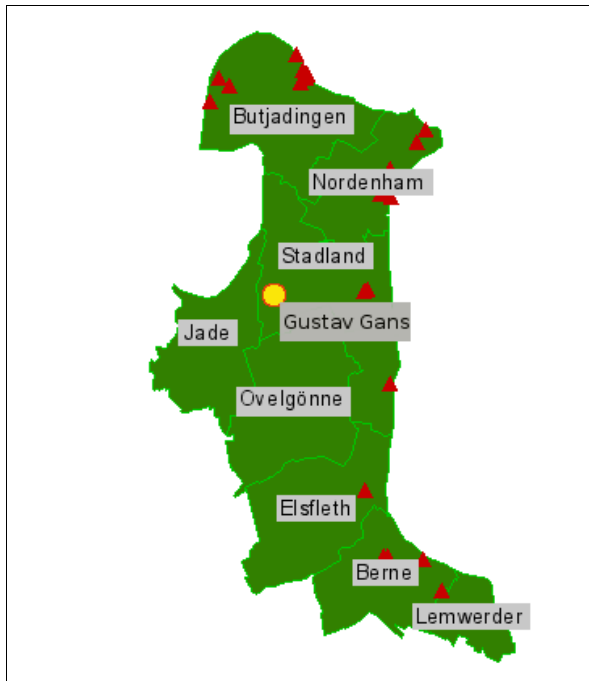
/var/www/htdocs/mapbender/http/php/mod_usemap.php

Hier muss festgelegt werden, welches WFS-Attribut in der Karte angezeigt werden soll.
Tragen Sie in Zeile 39 ein:

```
echo "um_title[um_title.length] = ".$g->getValueBySeparatedKey($ii,"INHABER")."."";
```

Wenn Sie dann in der Karte mit der Maus über einem Punkt „mouven“ dann wird dieser
gehighlighted und
der Wert aus dem Attributfeld „Inhaber“ wird präsentiert.

Abb. 26: WFS-Tooltip im Mapbender



Ich hoffe diese Anleitung konnte im Bereich WMS/WFS etwas Hilfestellung bieten.

Mit freundlichen Grüßen, Kai Behncke

3. Quellen

CCGIS & terrestris (2004): "Praxishandbuch WebGIS mit Freier Software"

Donaubauer, Andreas (2005): "Web Feature Services – Geodienst für den Zugriff auf objektstrukturierte Geodaten", in: Bernhard, Lars; Fitzke, Jens; Wagner, Roland M.: Geodateninfrastruktur, S.93-107

Emde, Astrid (2005): "UMN MapServer als Web Feature Service" . Foliensammlung UMN Anwenderkonferenz 2005

Erstling, Reinhard; **Simonis**, Ingo (2005): "Web Map Service", in: Bernhard, Lars; Fitzke, Jens; Wagner, Roland M.: Geodateninfrastruktur, S.108-129

Fischer, Thorsten (2003): "UMN MapServer 4.0"

Grewe, Klaus (2006): "ISO, OGC & Co", in: GIS-Business. 06/2006, S.28-31

Mitchell, Tyler (2005) : "Web Mapping Illustrated"

Open GIS Consortium Inc.(a) (2002): "Web Feature Service Implementation Specification"

Open GIS Consortium Inc. (b) (2002): "Web Map Service Implementation Specification"

Tscherkasski, Evgeni (2006): "Konzeption und Implementierung eines erweiterten Clients für den Web-Map-Service (WMS) der Stadt Bochum". Diplomarbeit an der Fachhochschule Bochum, Fachbereich Vermessungswesen und Geoinformatik.

Internet:

<http://www.mapbender.org>

http://www.mapbender.org/index.php/WFS_Konfiguration

<http://www.mapbender.org/index.php/UseMap>

http://mapserver.gis.umn.edu/docs/howto/wfs_server

http://mapserver.gis.umn.edu/docs/howto/wfs_client

http://mapserver.gis.umn.edu/docs/howto/wms_client

http://mapserver.gis.umn.edu/docs/howto/wms_server

<http://www.umn-mapserver.de>

<http://www.umn-mapserver-community.de>