Hochschule für Technik und Wirtschaft Dresden
Fakultät Geoinformation
Studiengang Vermessungswesen

DIPLOMARBEIT

Implementierung eines Feature Portrayal Service

vorgelegt von

Martin Domeyer

1. Gutachter: Prof. Dr.-Ing. F. Schwarzbach

2. Gutachter: Frau Prof. Dr.-Ing. M. Müller

Dresden, 13. Juli 2010

DANKSAGUNG

Für fachliche Anregungen und die Unterstützung zu dieser Diplomarbeit bedanke ich mich bei meinen beiden Gutachtern, Herrn Prof. Dr.-Ing. Frank Schwarzbach und Frau Prof. Dr.-Ing. Martina Müller.

Ein weiterer Dank geht an Frau Dipl.-Ing. Ines Schwarzbach und Herrn Dipl.-Ing. (FH) Stefan Schulze sowie das gesamte Labor Geoinformatik für die Hilfestellung bei allen Fragen.

Ganz besonders danke ich meiner Freundin Dore für den Beistand und vor allem auch die Geduld während der gesamten Studienzeit.

INHALTSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

ALK	Automatisierte Liegenschaftskarte
ASCII	American Standard Code for Information Interchange
BBOX	Bounding Box
CGI	Common Gateway Interface
CPS	Coverage Portrayal Service
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
DHDN	Deutsches Hauptdreiecksnetz
EPSG	European Petroleum Survey Group Geodesy
ESRI	Environmental Systems Research Institute
FPS	Feature Portrayal Service
GIS	Geoinformationssystem
GK	Gauß-Krüger-Koordinatensystem
GML	Geography Markup Language
GNU	GNU is not Unix
GNU LGPL	GNU Lesser General Public License
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
IDE	Integrated Development Environment
IP	Internet Protocol
ISO	International Organization for Standardization
JAR	Java Archive
Java EE	Java Enterprise Edition
Java SE	Java Standard Edition
JDK	Java Development Kit

JRE	Java Runtime Environment
JSP	JavaServer Page
KVP	Key Value Pair
MaSterStyLeS	Management System für Styles, Legenden und Symbole
MSI	Microsoft Installer
OGC	Open Geospatial Consortium
OpenJUMP	Open Java Unified Mapping Platform
OWS	OGC Web Service
PHP	PHP: Hypertext Preprocessor
PNG	Portable Network Graphics
QGIS	Quantum GIS
SE	Symbology Encoding
SLD	Styled Layer Descriptor
SQL	Structured Query Language
SRS	Spatial Reference System
SVN	Subversion
TOPP	The Open Planning Project
uDig	User-friendly Desktop Internet GIS
UMN	University of Minnesota
URL	Uniform Resource Locator
UTF-8	Unicode Transformation Format 8-Bit
WAR	Web Archive
WCS	Web Coverage Service
WFS	Web Feature Service
WMC	Web Map Context
WMS	Web Map Service
XML	Extensible Markup Language
XSD	XML Schema Definition

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGVERZEICHNIS

1 EINLEITUNG

Diese Diplomarbeit beschäftigt sich mit der «Implementierung eines Feature Portrayal Service». Das Ziel ist die Bereitstellung einer Lösung für den produktiven Einsatz im Labor Geoinformatik der Hochschule für Technik und Wirtschaft Dresden.

Im Einzelnen wurden folgende Teilaufgaben bearbeitet:

- Recherche nach geeigneten Open Source Frameworks und begründete Auswahl der Software,
- Konzeption und Realisierung eines Workaround,
- Recherche nach geeigneten Clients,
- Bereitstellung einer Webseite für das Geoportal der Hochschule mit einer geeigneten Client-Anwendung und
- Anbindung des vorhandenen Style Repository.

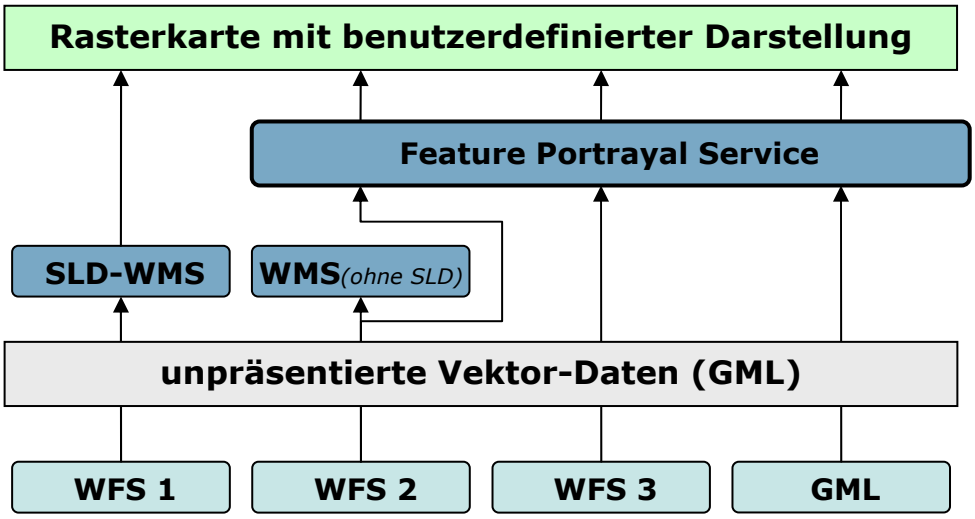
2 FACHLICHE SICHT

Fundamentale Voraussetzung für alle professionellen Kartendienste ist die Möglichkeit zur Kontrolle der grafischen Präsentation. Aus diesem Grund besteht ein starker Bedarf an der visuellen Darstellung von Geodaten. Diese transformiert rohe Informationen in wichtige und entscheidungstragende Werkzeuge.

Der Feature Portrayal Service (FPS) hat die Aufgabe, diese Anforderungen umzusetzen und damit wesentlich zur kartographischen Interoperabilität in Geodateninfrastrukturen beizutragen. Dabei ist jeder Nutzer in der Lage, beliebige Darstellungsvorschriften zu definieren.

In der *Abbildung 2-1* sind einige Anwendungsfälle aufgeführt, welche die Notwendigkeit des Dienstes zum Ausdruck bringen.

Abbildung 2-1: Anwendungsfälle



Die Web Feature Services (WFS) in der unteren Ebene enthalten die rohen Vektordaten. Diese Dienste liefern unpräsentierte Informationen im GML-Format.

Oftmals werden Web Map Services (WMS) aufgesetzt, die auf einen oder mehrere WFS als integrierte Datenquellen basieren. Handelt es sich dabei um einen SLD-fähigen WMS, könnte die Darstellung auch ohne einen Feature Portrayal Service erfolgen, jedoch nur für den intern definierten WFS. Einige Web Map Services bieten allerdings noch keine clientseitige Darstellungsmöglichkeit per SLD an und somit können aus diesen Diensten lediglich Karten mit vordefinierten Stilen generiert werden. Das ist der erste Fall, für den ein separater Darstellungsdienst zur benutzerdefinierten Präsentation genutzt werden muss. Dabei müssen die Rohdaten direkt aus der Datenquelle, also dem WFS, geholt werden.

Für Web Feature Services, die nicht innerhalb einer solchen Dienstekette implementiert wurden, kann eine Darstellung nur über den Feature Portrayal Service laufen. Dazu gehören auch so genannte Transaction WFS, mit denen der Datenbestand am Client editiert werden kann. Ebenso können auch direkt GML-Dateien vom Nutzer zur Visualisierung gebracht werden.

3 GRUNDLAGEN

Dieses Kapitel soll einen kurzen Überblick über die Spezifikationen des Open Geospatial Consortium (OGC) geben, die für das Verständnis dieser Arbeit zwingend erforderlich sind.

Das OGC [32] ist eine internationale Organisation, die frei zugängliche Standards zur interoperablen Nutzung von Geodaten verfasst.

3.1 Web Map Service (WMS)

Ein Web Map Service [3][2] liefert Bilder, die aus einem jeweils aktuellen räumlichen Datenbestand generiert werden. Bei diesem Datenbestand kann es sich um Geodatenbanken, Dateien (z.B. Shapefiles) oder andere Web Services handeln. In der Web Map Service Implementation Specification, deren aktuellster Stand die Version 1.3.0 ist, werden drei Requests beschrieben, die als Schnittstellen zwischen Dienst und Client dienen.

Die Schnittstelle GetCapabilities liefert das Angebot und die Fähigkeiten des Dienstes. Der Request mit den minimal erforderlichen Parametern würde wie folgt aussehen:

Listing 3-1: WMS - GetCapabilities-Request

```
http://host_adress/pfad?SERVICE=WMS&REQUEST=GetCapabilities.
```

Weitere optionale Parameter sind in der Spezifikation zu finden. Als Antwort erhält der Client eine XML-Datei, die die Eigenschaften des WMS nach den Regeln des OGC wiedergibt.

Da nun die Capabilities des Dienstes bekannt sind, kann eine Karte angefordert werden. Dies geschieht mit einem GetMap-Request.

Listing 3-2: WMS - GetMap-Request

```
http://host_adress/pfad?VERSION=1.1.1&REQUEST=GetMap
&LAYERS=layer_list&STYLES=style_list&SRS=EPSG:identifier
&BBOX=minx,miny,maxx,maxy&WIDTH=output_width
&HEIGHT=output_height&FORMAT=output_format
```

Dieses Interface fordert die gewünschte Karte im angegebenen Ausgabeformat an.

Die dritte Schnittstelle ist optional und stellt eine Sachdatenabfrage dar.

Listing 3-3: WMS - GetFeatureInfo-Request

```
http://host_adress/pfad?VERSION=1.1.1&REQUEST=GetFeatureInfo
&SRS=EPSG:identifier&BBOX=minx,miny,maxx,maxy
&WIDTH=output_width&HEIGHT=output_height
&FORMAT=output_format&QUERY_LAYERS=layer_list
&X=pixel_column&Y=pixel_row
```

3.2 Web Feature Service (WFS)

Die Web Feature Service Implementation Specification [20][19] liegt derzeit in der Version 1.1.0 vor. Im Gegensatz zu einem WMS stellt ein WFS Vektordaten zur Verfügung. Man erhält Geo-Objekte im GML-Format *(siehe Abschnitt 3.6)*, die keinerlei Darstellungsvorschriften unterliegen und somit unpräsentiert vorliegen. Die wichtigsten Schnittstellen lauten GetCapabilities, DescribeFeatureType und GetFeature.

Das GetCapabilities-Interface hat an dieser Stelle den gleichen Zweck wie bei einem WMS.

Listing 3-4: WFS - GetCapabilities-Request

```
http://host_adress/pfad?SERVICE=WFS&REQUEST=GetCapabilities
```

Per DescribeFeatureType wird eine Beschreibung der angegebenen Featuretypes angefordert.

Listing 3-5: WFS - DescribeFeatureType-Request

```
http://host_adress/pfad?VERSION=1.1.0&SERVICE=WFS
&REQUEST=DescribeFeatureType
```

Bei der zurückgegebenen Datei handelt es sich um ein XML-Schema-Dokument.

Um letztendlich eine GML-Datei zu erhalten, kommt der GetFeature-Request zum Einsatz.

Listing 3-6: WFS - GetFeature-Request

```
http://host_adress/pfad?VERSION=1.1.0&SERVICE=WFS
&REQUEST=GetFeature&TYPENAME=featuretype_list
```

Das Ergebnis sind alle ausgewählten Features, die zusätzlich auch nach verschiedenen Gesichtspunkten gefiltert werden können.

3.3 Styled Layer Descriptor (SLD)

SLD [8][10] wird benutzt, um die grafische Präsentation von Geodaten auch als Nutzer steuern zu können. Der derzeit aktuelle internationale Standard, die Version 1.1.0 mit dem Namen «Styled Layer Descriptor profile of the Web Map Service Implementation Specification» [10], erweitert die WMS-Spezifikation um die Möglichkeit benutzerdefinierter grafischer Darstellungen. Bei solchen als SLD-WMS bezeichneten Diensten ist der User also nicht von den angebotenen Styles des WMS abhängig.

Der GetMap-Request erhält nun den SLD-Parameter, der eine URL übergibt.

Listing 3-7: SLD-GetMap-Request

```
http://host_adress/pfad?VERSION=1.1.1
&REQUEST=GetMap&SRS=EPSG:identifier
&BBOX=minx,miny,maxx,maxy&WIDTH=output_width
&HEIGHT=output_height&FORMAT=output_format
&SLD=http://SLDhost_adress/SLD.xml
```

Wie hier zu erkennen ist, handelt es sich bei einem SLD-Dokument um eine benutzerdefinierte XML-Datei, deren Aufbau ausführlich in der SLD-Spezifikation erläutert wird. Es beinhaltet als Wurzelelement ein *StyledLayerDescriptor*-Element, das aus einer Reihe von benannten oder benutzerdefinierten Ebenen und Stilen besteht. Die Parameter *LAYERS* und *STYLES* können bei diesem Request weggelassen werden. Wird trotzdem der *LAYERS*-Parameter übergeben, werden nur diese Ebenen dargestellt und die SLD-Datei fungiert als Stilbibliothek. Die Angabe von *STYLES* wird nur berücksichtigt, wenn keine SLD-Stile mit diesem Namen existieren.

Alternativ kann die SLD-Syntax auch mit dem Request-Parameter *SLD_BODY* übergeben werden.

Listing 3-8: SLD-GetMap-Request mit SLD_BODY-Parameter

```
http://host_adress/pfad?VERSION=1.1.1
&REQUEST=GetMap&SRS=EPSG:identifier
&BBOX=minx,miny,maxx,maxy&WIDTH=output_width
&HEIGHT=output_height&FORMAT=output_format
&SLD_BODY=%3C%3Fxml+version%3D%221.0%22+
    encoding%3D%22UTF-8%22%3F%3E
    %3CStyledLayerDescriptor+version%3D%221.1.0%22%3E
    %3CUserLayer%3E [...] %3C%2FUserLayer%3E
    %3C%2FStyledLayerDescriptor%3E
```

Diese Methode bietet den Vorteil, dass der Nutzer die SLD-Datei nicht auf einem öffentlichen Webserver ablegen muss. Allerdings bringt sie auch einige Nachteile mit sich. So kann es, neben der Unübersichtlichkeit, Probleme mit zu langen URLs geben. Zusätzlich können Schwierigkeiten bei UTF-8-Zeichen außerhalb des 7-bit-ASCII-Bereichs auftreten, da HTTP das ISO Latin-1 character set benutzt.

In einer SLD-Datei können die Elemente *NamedLayer*, *NamedStyle*, *UserLayer* und *UserStyle* definiert werden. Mit *NamedLayer* ist eine Ebene gemeint, die bereits existiert und somit im Capabilities-Dokument des WMS auftaucht. Ebenso können diesem Dokument die vordefinierten Stile des Dienstes entnommen werden, welche als *NamedStyles* bezeichnet werden. *UserStyles* hingegen werden vom Benutzer neu definiert und auf die entsprechenden Layer angewendet. *UserLayer* greifen auf die Daten fremder Dienste zurück.

3.4 Symbology Encoding (SE)

Seit der SLD-Spezifikation der Version 1.1.0 wurde ein Teil aus dieser herausgelöst und als eigene Spezifikation veröffentlicht. Dieser Standard, genannt Symbology Encoding Implementation Specification [13], bietet die Möglichkeit, eigene Stile mit XML-Mitteln zu beschreiben. Im konkreten Anwendungsfall werden mit Symbology Encoding eigene Darstellungsregeln, also UserStyles, in SLD-Dokumenten definiert.

Dem Nutzer stehen *FeatureTypeStyles*, *Rules* und *Symbolizers* zur Verfügung, um die grafische Ausprägung der Karte nach den eigenen Wünschen steuern zu können.

3.5 Filter Encoding (FE)

Die Filterung von Geodaten kann per Filter Encoding [18] geschehen. In Kombination mit Symbology Encoding würden die Angaben zur Einschränkung der Datenmenge im *Rule*-Element stehen. Möglich sind sowohl raumbezogene als auch attributive Abhängigkeiten, die als *Spatial operators* und *Comparison operators* bezeichnet werden. Die Verknüpfung dieser Operatoren erfolgt mittels *Logical operators*.

3.6 Geography Markup Language (GML)

Das XML-Derivat GML [14] dient als Austauschformat für raumbezogene Objekte, genannt Features. Im Zusammenhang mit dieser Diplomarbeit ist GML als Rückgabe-Syntax nach der Anforderung von Daten eines WFS zu betrachten. Wichtigste Eigenschaft dieser GetFeature-Responses ist es, dass es sich um reine Vektordaten ohne jede Darstellungsvorschrift handelt.

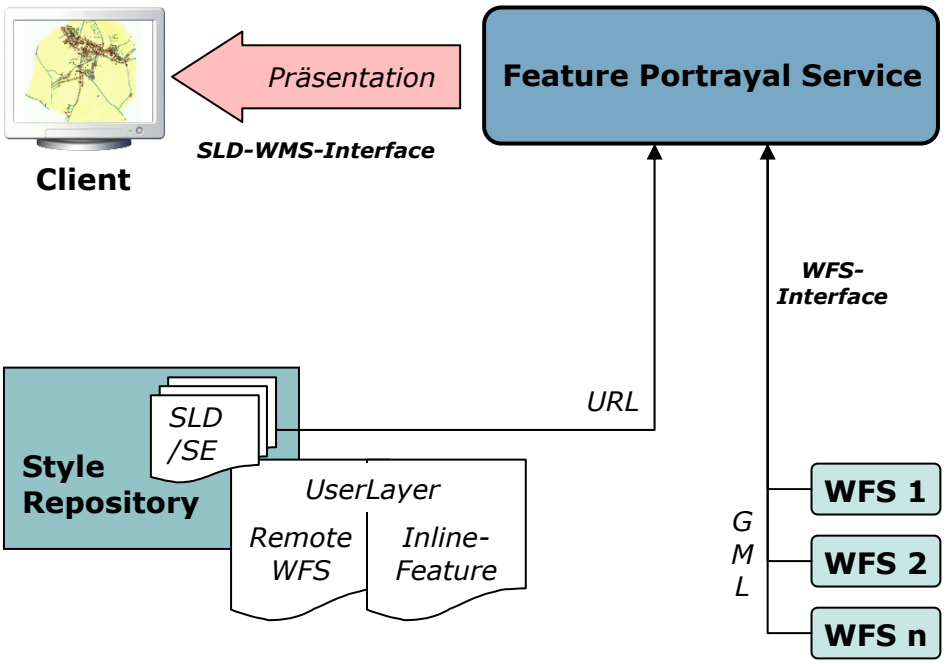
4 DEFINITION UND ARBEITSWEISE

4.1 Begriffsbestimmung

Der Begriff des Feature Portrayal Service, kurz FPS, entstammt der SLD-Spezifikation der Version 1.1.0 [10]. Es handelt sich um einen Darstellungsdienst, der die interoperable Nutzung von Vektor-Daten, also von so genannten Features, ermöglicht.

Dieser spezielle Web Map Service hat die Aufgabe, Geodaten von beliebigen Web Feature Services sowie rohe GML-Daten in einer gewünschten grafischen Ausprägung zu präsentieren. Dabei ist der jeweilige Nutzer in der Lage, eigene Darstellungen zu definieren und auf einen beliebigen Datenbestand anzuwenden.

Abbildung 4-1: Arbeitsweise eines Feature Portrayal Service



Datenquellen, Darstellungsvorschriften und der Feature Portrayal Service selbst sind insofern unabhängig voneinander, dass diese drei Komponenten auf verschiedenen und voneinander unabhängigen Servern liegen können. Diese Art der Implementierung bezeichnet man als Component WMS.

4.2 Integrated und Component WMS

Der SLD-Standard unterscheidet zunächst zwei Arten von Portrayal Engines. Die klassische Variante, der Integrated WMS, arbeitet nur in bestimmten Konfigurationen. Beispielsweise dient der Integrated WMS zur Darstellung von eigenen und vom Administrator definierten Daten oder Diensten. Die Datenquelle kann ein Web Feature Service sein, der aber in festem Zusammenhang mit dem WMS steht. Dieser Dienst muss also keine UserLayer unterstützen.

Im Gegensatz dazu gibt es die Gruppe der so genannten Component WMS, die in allen denkbaren Kombinationen funktionieren. Ein solcher Service muss UserLayer verstehen können, um Daten fremder Dienste symbolisieren zu können. Bei diesen fremden Diensten kann es sich um Web Coverage Services (WCS), also um Rasterdaten, und Web Feature Services (WFS) handeln. Dementsprechend existiert die Unterscheidung in Coverage Portrayal Services (CPS), mit deren Hilfe beispielsweise durch eine bestimmte Anordnung der Spektralkanäle die Darstellung der Rasterdaten bestimmt werden kann, und dem Kernstück dieser Diplomarbeit: dem Feature Portrayal Service (FPS).

4.3 Aufbau der SLD-Datei

Der Feature Portrayal Service unterstützt das WMS-Interface und besitzt meist keine vordefinierten Ebenen und Stile. Daher müssen

UserLayer in SLD-Dokumenten definiert werden können, um dem Dienst die entsprechenden Datenquellen mitzuteilen. *Listing 4-1* zeigt eine einfache SLD-Datei, die eine URL zu einem externen WFS enthält.

Listing 4-1: SLD mit RemoteWFS

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.1.0"
xmlns=http://www.opengis.net/sld
xmlns:ogc=http://www.opengis.net/ogc
xmlns:se=http://www.opengis.net/se
xmlns:ns="http://NamespaceHost_adress/nsp">
 <UserLayer>
    <se:Name>RemoteWFS</se:Name>
    <RemoteOWS>
        <Service>WFS</Service>
        <se:OnlineResource
        xmlns:xlink=http://www.w3.org/1999/xlink
        xlink:type="simple"
        xlink:href="http://RemoteWFShost_adress/pfad?"/>
    </RemoteOWS>
    <LayerFeatureConstraints>
        <FeatureTypeConstraint>
            <se:FeatureTypeName>
                nsp:Feature
            </se:FeatureTypeName>
        </FeatureTypeConstraint>
    </LayerFeatureConstraints>
    <UserStyle>
        <se:Name>RemoteWFSstyle</se:Name>
        <se:FeatureTypeStyle>
            <se:Rule>
                <se:PolygonSymbolizer>
                    <se:Geometry>
                        <ogc:PropertyName>
                            nsp:geometry
                        </ogc:PropertyName>
                    </se:Geometry>
                    <se:Fill>
                        <se:SvgParameter name="fill">
                            #000000
                        </se:SvgParameter>
                    </se:Fill>
                </se:PolygonSymbolizer>
            </se:Rule>
        </se:FeatureTypeStyle>
    </UserStyle>
 </UserLayer>
</StyledLayerDescriptor>
```

Es handelt sich bei diesem Beispiel um SLD 1.1.0 [10]. Innerhalb dieser Diplomarbeit wird allerdings aus noch zu nennenden Gründen sehr oft SLD 1.0.0 [8] zum Einsatz kommen, obwohl der Feature Portrayal Service in dieser Version noch nicht als solcher bezeichnet wird. Im Element *RemoteOWS* wird angegeben, dass es sich um einen fremden WFS handelt und über welche URL dieser abzurufen ist. Der *FeatureTypeConstraint* gibt an, welche Features des Dienstes dargestellt werden sollen. Die Art und Weise dieser Darstellung wird anschließend als *UserStyle* definiert. Im Beispiel werden Flächen mittels Symbology Encoding [13] mit schwarzer Farbe gefüllt. Eine Angabe unter *se:Geometry* kann entfallen, wenn alle Geometrie-attribute visualisiert werden sollen oder nur eines existiert. Es können auch mehrere Layer definiert werden, wobei jedoch jeder Ebene nur genau eine Datenquelle zugewiesen werden kann.

Neben Web Feature Services, deren Daten intern mittels der WFS-Schnittstellen vom FPS selbst besorgt werden, kann seit SLD 1.1.0 auch direkt für GML-Daten, die in das SLD-Dokument integriert werden, eine entsprechende grafische Präsentation definiert werden. Dazu kann anstelle des *RemoteOWS* das Element *InlineFeature* genutzt werden.

Listing 4-2: InlineFeature

```
<InlineFeature>
 <gml:FeatureCollection>
    ...
 </gml:FeatureCollection>
</InlineFeature>
```

Es beinhaltet wiederum eine oder mehrere *FeatureCollections*, welche die Geodaten beinhalten. Der GML-Namensraum muss zusätzlich in das XML-Dokument eingebunden werden.

4.4 Parameter eines FPS-Request

Um dem FPS die erstellte SLD-Datei mitzuteilen, muss diese auf einem öffentlich zugänglichen Webserver abgelegt werden. Anschließend kann der im *Listing 3-7* beschriebene Request genutzt werden.

Anstelle der Angabe der Datenquelle im *UserLayer* des SLD-Konstrukts könnten auch die optionalen Parameter *REMOTE_OWS_TYPE* und *REMOTE_OWS_URL* im GetMap-Request genutzt werden.

Listing 4-3: SLD-GetMap-Request mit REMOTE_OWS-Parametern

```
http://host_adress/pfad?VERSION=1.1.1
&REQUEST=GetMap&SRS=EPSG:identifier
&BBOX=minx,miny,maxx,maxy&WIDTH=output_width
&HEIGHT=output_height&FORMAT=output_format
&SLD=http://SLDhost_adress/SLD.xml
&REMOTE_OWS_TYPE=WFS
&REMOTE_OWS_URL=http://RemoteWFShost_adress/pfad?
```

4.5 Capabilities des Dienstes

Ob ein Web Map Service grundsätzlich die Funktionalitäten eines Feature Portrayal Service zur Verfügung stellt, ist den Capabilities des Dienstes zu entnehmen. SLD 1.1.0 erweitert die üblichen WMS-Eigenschaften um ein Element, dessen optionale Attribute, mit den Attributwerten *0 (TRUE)* oder *1 (FALSE)*, Aufschluss über die Art des SLD-WMS liefern.

Listing 4-4: UserDefinedSymbolization in WMS-Capabilities

```
<UserDefinedSymbolization
SupportSLD="1" UserLayer="1" UserStyle="1" RemoteWFS="1"
InlineFeatureData="1" RemoteWCS="0"/>
```

Für den Einsatz als Feature Portrayal Service muss der Dienst unbedingt *SupportSLD*, *UserLayer*, *UserStyle* und *RemoteWFS* unterstützen. Die Möglichkeit für einen FPS, GML-Daten direkt einzubinden (*InlineFeatureData*) ist optional.

5 IMPLEMENTIERUNG

5.1 Auswahl eines Framworks

Im Internet ist eine Vielzahl von Produkten im Umlauf, bei denen sowohl die WMS- als auch die SLD-Spezifikation umgesetzt wurden. Daher sollten diese Lösungen der erste Anlaufpunkt zur Implementierung eines Feature Portrayal Service sein. Die Entwicklung eines komplett eigenen Dienstes würde nur dann Sinn ergeben, wenn sich alle vorhandenen Services als absolut ungeeignet erweisen würden.

Um eine Auswahl bezüglich eines geeigneten Open Source Frameworks zur Implementierung des Dienstes zu treffen, sind intensive Software-Untersuchungen notwendig. Die in Frage kommenden Softwarelösungen werden dabei auf die drei bekanntesten Open Source WMS-Implementierungen beschränkt, da bei diesen die Wahrscheinlichkeit der Unterstützung der notwendigen Funktionen am Höchsten ist. Dabei handelt es sich um die Produkte deegree, MapServer und GeoServer.

5.1.1 Anforderungsanalyse

Bevor die einzelnen Dienste untersucht werden können, müssen die Anforderungen an einen Feature Portrayal Service aufgeführt werden, um anschließend dementsprechende Testumgebungen einrichten zu können.

Tabelle 5-1 beinhaltet alle Sachverhalte, auf welche die verschiedenen Dienste untersucht werden müssen. Weiterhin ist der letzten Spalte zu entnehmen, ob eine Umsetzung dieser Sachverhalte obligatorisch oder optional ist.

WMS-Capabilities

Bei den WMS-Capabilities handelt es sich lediglich um die jeweilige Unterstützung laut dem Response-Dokument nach einem GetCapabilities-Request. Die tatsächliche und korrekte Implementierung der einzelnen Punkte wird an dieser Stelle noch nicht geprüft. Da *InlineFeatureData*, wie in *Abschnitt 4.5* bereits erwähnt, optional ist, ist dessen Umsetzung nicht zwingend erforderlich.

WMS-GetMap-Interface

Weiterhin sollen alle relevanten GetMap-Request-Möglichkeiten ausprobiert werden. Die einfache Abfrage aus *Listing 3-2* ohne SLD wird nur der Vollständigkeit halber mit aufgeführt. Für den Einsatz als Feature Portrayal Service muss der Dienst natürlich sowohl mit einem *SLD*- als auch einem *SLD_BODY*-Parameter umgehen können.

Styled Layer Descriptor / Symbology Encoding 1.0.0

Bei der Prüfung der vollständigen Unterstützung von Styled Layer Descriptor, beziehungsweise Symbology Encoding, der Version 1.0.0 wird zunächst eine einfache Kombination aus *NamedLayer* und *NamedStyle* getestet, um die grundsätzliche Umsetzung nachzuweisen.

Wie bereits in *Abschnitt 4.3* genannt, kommen bei einem Portrayal Service allerdings benutzerdefinierte Ebenen in Verbindung mit ebenfalls vom User gesteuerten Stilen zum Einsatz. Dazu wird zuerst nur der *UserStyle* selbst zur Darstellung eines *NamedLayer* genutzt, bevor die typische FPS-Kombination erarbeitet wird. Dabei muss wiederum unterschieden werden, um welche Version es sich bei dem als Datenquelle eingesetzten fremden Web Feature Service handelt, da man nicht davon ausgehen kann, dass beide Veröffentlichungen korrekt implementiert wurden.

Die Unterstützung von *InlineFeature* spielt bei SLD 1.0.0 eigentlich keine Rolle (*siehe Abschnitt 4.3*). Allerdings wird es an dieser Stelle aufgeführt, da GeoServer trotzdem den Ansatz einer Lösung dafür anbietet (*siehe Abschnitt 5.1.4*).

Styled Layer Descriptor / Symbology Encoding 1.1.0

Da der Feature Portrayal Service der SLD-Version 1.1.0 zu entnehmen ist, muss deren Support unbedingt getestet werden. Dabei handelt es sich wiederum um alle bereits im vorherigen Absatz geschilderten Tests. *InlineFeature* ist laut der Spezifikation optional.

Filter Encoding

In einem SLD-Dokument muss die Möglichkeit bestehen, die Daten einer benutzerdefinierten Ebene entsprechend der Filter Encoding Implementation Specification [18] anhand diverser Aspekte zu differenzieren, um nur einen gewünschten Teil der zur Verfügung gestellten Datensätze zu präsentieren oder einzelne Elemente aufgrund unterschiedlicher Attributwerte verschiedenartig darzustellen. Dieser Fall soll mittels einer Beispiellösung nachvollzogen werden.

Zusammenfassung

Tabelle 5-1: Anforderungen an einen FPS

Bezeichnung		Beschreibung	oblig.
WMS-Capabilities	***SupportSLD***	Test, ob das jeweilige Attribut laut der WMS-Capabilities (*siehe 4.5*) unterstützt wird	ja
	UserLayer		ja
	UserStyle		ja
	RemoteWFS		ja
	InlineFeatureData		nein
Get-Map	***einfach***	WMS-GetMap ohne SLD	ja
	SLD	mit SLD-Parameter	ja
	SLD_BODY	mit SLD_BODY-Parameter	ja
SLD/SE 1.0.0	***NamedStyle***	*siehe 3.3*	nein
	UserStyle		ja
	UserLayer: WFS 1.0.0	RemoteWFS ist WFS 1.0.0	ja
	UserLayer: WFS 1.1.0	RemoteWFS ist WFS 1.1.0	ja
	InlineFeature	GML-Daten im SLD-Dokument	nein
SLD/SE 1.1.0	***NamedStyle***	*siehe 3.3*	nein
	UserStyle		ja
	UserLayer: WFS 1.0.0	RemoteWFS ist WFS 1.0.0	ja
	UserLayer: WFS 1.1.0	RemoteWFS ist WFS 1.1.0	ja
	InlineFeature	GML-Daten im SLD-Dokument	nein
Filter Encoding		*siehe 3.5*	ja

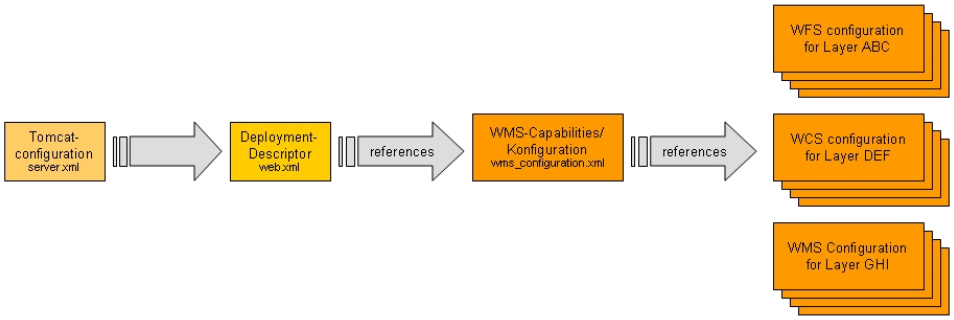
Diese Untersuchungen wurden mit den jeweils aktuellsten veröffentlichten Versionen der zur Verfügung stehenden Frameworks durchgeführt. Als Testumgebung wurde das Paket XAMPP Lite 1.7.2 für Windows [22] mit dem Tomcat-AddOn 6.0.20 eingesetzt. Bei allen aufgesetzten Framworks wurden Beispiel-ALK-Daten von Großschönau integriert. Dabei handelte es sich um ein ESRI-Shapefile, das die Flurstücke dieses Gebietes beinhaltet. Die zugehörigen Koordinaten entstammen dem Deutschen Hauptdreiecksnetz (DHDN) und befinden sich im fünften Streifen des Gauß-Krüger-Koordinatensystems. Alle Konfigurationen und Test-Dokumente befinden sich auf der zu dieser Diplomarbeit gehörenden CD *(Anlage D, Ordner \frameworks\)*.

5.1.2 Untersuchung des deegree WMS

Das deegree-Projekt [25], dessen Hauptmitglieder die Firma lat/lon GmbH [29] in Bonn und die Arbeitsgruppe GIS des Geographischen Instituts der Universität Bonn sind, ist ein Open Source Produkt, das unter der GNU Lesser General Public License (GNU LGPL) geschützt ist. Die aktuellste stabile Version zum Zeitpunkt dieser Diplomarbeit war deegree 2.3, welche die wesentlichsten Bausteine zum Aufbau einer Geodateninfrastruktur zur Verfügung stellt.

Die Konfiguration erfolgt mit Hilfe diverser XML-Dokumente. *Abbildung 5-1* stellt die Konfigurationsdokumente eines deegree WMS dar.

Abbildung 5-1: Files affecting the deegree WMS configuration [12]



Für die Programmierung von deegree wurde die Java-Servlet-Technologie genutzt. Das bedeutet, ein deegree Web Server wird von einem zentralen Servlet kontrolliert - dem Dispatcher. Da Java eine plattformunabhängige Programmiersprache ist, ist auch deegree plattformunabhängig.

Der zu Testzwecken mit den ALK-Daten vorkonfigurierte deegree WMS, welcher sich in der digitalen Anlage befindet, kann direkt in einen Apache Tomcat über den Tomcat Manager eingebunden werden.

WMS-Capabilities

Listing 5-1: deegree WMS - GetCapabilities-Request

```
http://127.0.0.1:8080/deegree-wms/services?
SERVICE=WMS&REQUEST=GetCapabilities
```

Laut dem *UserDefinedSymbolization*-Element der WMS-Capabilities werden, ausgenommen von *InlineFeatureData*, alle relevanten Attribute unterstützt.

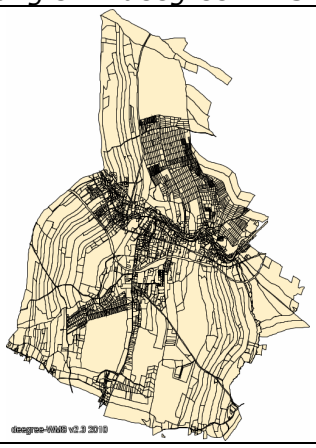
WMS-GetMap-Interface

Listing 5-2: deegree WMS - GetMap-Request

```
http://localhost:8080/deegree-wms/services?VERSION=1.1.1
&REQUEST=GetMap&LAYERS=ALK&STYLES=default:app:FLST
&SRS=EPSG:31469&BBOX=5474086,5637004,5479155,5643637
&WIDTH=500&HEIGHT=708&FORMAT=image/png
```

Bei dem einfachen GetMap-Request aus *Listing 5-2* erhält man eine Karte im angeforderten Standard-Stil.

Abbildung 5-2: deegree WMS - Karte im Standard-Stil



Bei den beiden GetMap-Anfragen mit SLD erscheint dasselbe Bild, was deren Funktionalität beweist.

Listing 5-3: deegree WMS - GetMap mit SLD-Parameter

```
http://localhost:8080/deegree-wms/services?
VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:31469
&BBOX=5474086,5637004,5479155,5643637
&WIDTH=500&HEIGHT=708&FORMAT=image/png
&SLD=file:///E:/xampplite/_Testdaten/deegree_simpleSLD-100.xml
```

Listing 5-4: deegree WMS - GetMap mit SLD_BODY-Parameter

```
http://localhost:8080/deegree-wms/services?
VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:31469
&BBOX=5474086,5637004,5479155,5643637&WIDTH=500
&HEIGHT=708&FORMAT=image/png&SLD_BODY=%3C?xml%20
version=%221.0%22%20encoding=%22UTF-8%22?%3E
%3CStyledLayerDescriptor%20version=%221.1.0%22%20xmlns=
%22http://www.opengis.net/sld%22%3E%3CNamedLayer%3E
%3CName%3EALK%3C/Name%3E%3CNamedStyle%3E
%3CName%3Edefault:app:FLST%3C/Name%3E%3C/NamedStyle%3E
%3C/NamedLayer%3E%3C/StyledLayerDescriptor%3E
```

Styled Layer Descriptor / Symbology Encoding 1.0.0

Da es sich bei der im *Listing 5-3* angegebenen SLD-Datei um die Version 1.0.0 mit einem *NamedLayer* und einem zugehörigen *NamedStyle* handelt, steht fest, dass diese Variante grundsätzlich unterstützt wird.

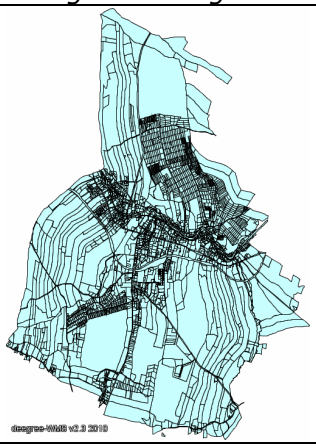
Will man eine selbst definierte Darstellung erzwingen, ist mindestens ein *UserStyle* notwendig.

Listing 5-5: deegree WMS - NamedLayer mit UserStyle

```
http://localhost:8080/deegree-wms/services?
VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:31469
&BBOX=5474086,5637004,5479155,5643637
&WIDTH=500&HEIGHT=708&FORMAT=image/png
&SLD=file:///E:/xampplite/_Testdaten/deegree_UserStyleSLD-100.xml
```

Wie zu erwarten war, erhält man die gewünschte Darstellung.

Abbildung 5-3: deegree WMS - Karte mit UserStyle



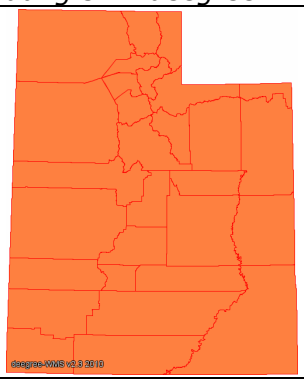
Um nun den Zugriff auf fremde Web Feature Services zu testen, werden zunächst Daten des deegree Demo Servers präsentiert. Dazu wird in der Datenquelle im *UserLayer* die entsprechende URL zum *RemoteOWS* angegeben. Es handelt sich hier um die Beispieldaten aus dem US-Bundesstaat Utah.

Listing 5-6: deegree WMS - UserLayer (deegree WFS)

```
http://localhost:8080/deegree-wms/services?
REQUEST=GetMap&VERSION=1.1.1&SERVICE=WMS
&FORMAT=image/png&BBOX=228563,4094785,673992,4653592
&SRS=EPSG:26912&WIDTH=500&HEIGHT=627
&SLD=file:///E:/xampplite/_Testdaten/deegreeWFS_SLD-100.xml
```

An dieser Stelle ist zum ersten Mal festzustellen, dass deegree tatsächlich als Framework für einen Feature Portrayal Service in Frage kommt, da dieser Request die gewünscht Karte zurückbekommt.

Abbildung 5-4: deegree WMS - Karte mit UserLayer



Nach dem Response kann man der Log-Datei des Tomcat entnehmen, dass bei diesem fremden Dienst, der beide WFS-Versionen unterstützt, die WFS 1.1.0-Schnittstelle genutzt wird.

Listing 5-7: deegree WMS - UserLayer (MapServer WFS 1.0.0)

```
http://localhost:8080/deegree-wms/services?
REQUEST=GetMap&VERSION=1.1.1&SERVICE=WMS
&FORMAT=image/png&BBOX=5.8,47.2,15.1,55.1
&SRS=EPSG:4326&WIDTH=500&HEIGHT=425&SLD=
file:///E:/xampplite/_Testdaten/MapServerWFS100_SLD-100.xml
```

Nimmt man hingegen, wie in *Listing 5-7* geschehen, einen WFS, der nur die Version 1.0.0 verarbeitet hat, erhält man eine Fehlermeldung,

dass der angefragte Featuretype nicht existieren würde. Beim Debugging des Problems wird klar, dass es in Wirklichkeit an der älteren WFS-Version liegt. Es kann also geschlussfolgert werden, dass deegree nur WFS 1.1.0 verarbeiten kann.

Allerdings ergaben die beiden folgenden Tests, dass auch Remote Web Feature Services der Version 1.1.0 nicht dargestellt werden können, wenn sie nicht mit dem deegree Framework implementiert wurden.

Listing 5-8: deegree WMS - UserLayer (MapServer WFS 1.1.0)

```
http://localhost:8080/deegree-wms/services?
REQUEST=GetMap&VERSION=1.1.1&SERVICE=WMS
&FORMAT=image/png&BBOX=4507747,5586671,4563610,5630925
&SRS=EPSG:31468&WIDTH=500&HEIGHT=396&SLD=
file:///E:/xampplite/_Testdaten/MapServerWFS110_SLD-100.xml
```

Listing 5-9: deegree WMS - UserLayer (GeoServer WFS)

```
http://localhost:8080/deegree-wms/services?
REQUEST=GetMap&VERSION=1.1.1&SERVICE=WMS
&FORMAT=image/png&BBOX=5474086,5637004,5479155,5643637
&SRS=EPSG:31469&WIDTH=500&HEIGHT=708
&SLD=file:///E:/xampplite/_Testdaten/GeoServerWFS_SLD-100.xml
```

Während auf den Request aus *Listing 5-8* lediglich mit einer allgemeinen Fehlermeldung geantwortet wird, erhält man bei *Listing 5-9* mit einem GeoServer WFS, der in diesem Fall selbst lokal aufgesetzt wurde, nur eine leere Karte.

Styled Layer Descriptor / Symbology Encoding 1.1.0

Um die eben genannten Untersuchungen auch mit der SLD-Version 1.1.0 durchführen zu können, muss zunächst deren grundsätzliche Funktionsweise in deegree auf den Prüfstand gestellt werden.

Listing 5-10: deegree WMS - GetMap mit SLD 1.1.0

```
http://localhost:8080/deegree-wms/services?
VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:31469
&BBOX=5474086,5637004,5479155,5643637
&WIDTH=500&HEIGHT=708&FORMAT=image/png
&SLD=file:///E:/xampplite/_Testdaten/deegree_simpleSLD-110.xml
```

Wie anhand der deegree-Dokumentation [12] bereits erwartet, wird dieser SLD-Standard nicht verstanden. Somit erübrigen sich auch alle anderen Tests mit dieser Version.

Filter Encoding

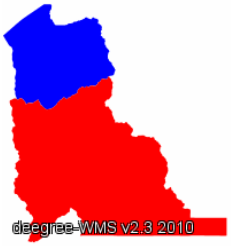
Um abschließend noch die Filtermöglichkeiten auszuprobieren, kommen wieder die bereits bei *Listing 5-6* genutzten Utah-Beispieldaten zum Einsatz. In diesem Fall enthält das SLD-Dokument noch einige Filteranweisungen, um nur Utah und Salt Lake in unterschiedlichen Farben darzustellen.

Listing 5-11: deegree-WMS - GetMap mit SLD und Filter Encoding

```
http://localhost:8080/deegree-wms/services?
REQUEST=GetMap&VERSION=1.1.1&SERVICE=WMS
&FORMAT=image/png&BBOX=393000,4400000,515000,4535000
&SRS=EPSG:26912&WIDTH=200&HEIGHT=221
&SLD=file:///E:/xampplite/_Testdaten/deegreeWFS_FilterSLD-100.xml
```

Die zurückgegebene Karte in *Abbildung 5-5* zeigt, dass Filter Encoding bei deegree angewandt werden kann und auch korrekt funktioniert.

Abbildung 5-5: deegree WMS - Karte nach Filter Encoding

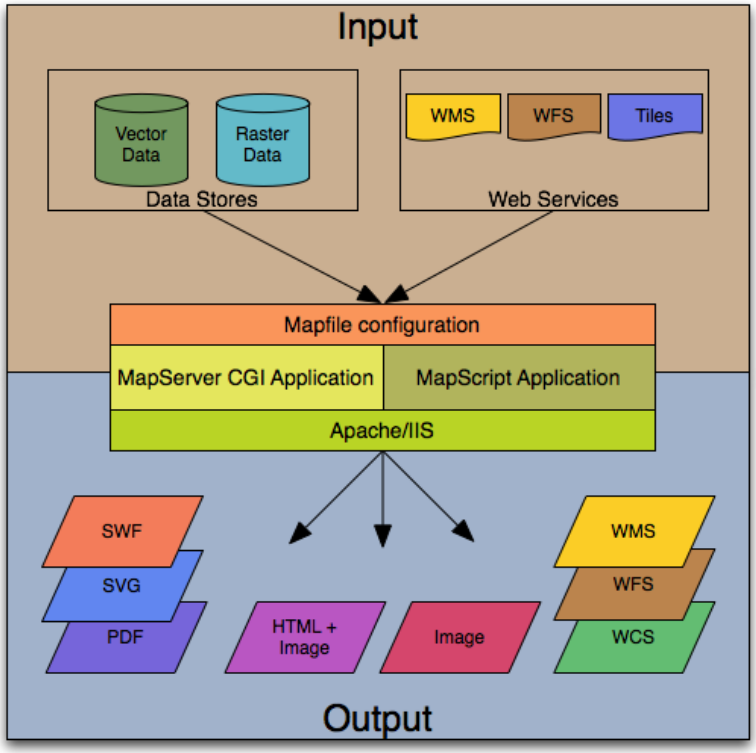


5.1.3 Untersuchung des MapServer WMS

Der Mapserver [31], aufgrund seiner ursprünglichen Entwicklung an der University of Minnesota auch als UMN MapServer bekannt, kann neben seiner Funktion als OGC-konformer Dienst auch als proprietärer Kartendienst genutzt werden. Bei der zum Zeitpunkt der Untersuchungen aktuellsten Veröffentlichung handelte es sich um die Version 5.6.1 [17].

Es handelt sich um ein CGI-Programm, welches bei jedem Request vom Webserver ausgeführt wird. Unter Berücksichtigung der Request-Parameter und dem so genannten *Mapfile*, der zentralen Konfigurationsdatei des MapServer, wird ein entsprechendes Bild an den Client zurückgegeben.

Abbildung 5-6: The basic architecture of MapServer applications. [17]



WMS-Capabilities

Listing 5-12: MapServer WMS - GetCapabilities-Request

```
http://localhost/cgi-bin/mapserv?
SERVICE=WMS&REQUEST=GetCapabilities
```

Im Gegensatz zum deegree WMS kann man bei MapServer schon an diesem Punkt erkennen, dass er nicht als Feature Portrayal Service geeignet ist. Das erhaltene Capabilities-Dokument zeigt, dass weder *UserLayer* noch *RemoteWFS* unterstützt werden. Demzufolge wird auch der Support von *InlineFeatureData* verneint.

WMS-GetMap-Interface

Listing 5-13: MapServer WMS - GetMap-Request

```
http://localhost/cgi-bin/mapserv?
VERSION=1.1.1&REQUEST=GetMap&LAYERS=ALK&STYLES=default
&SRS=EPSG:31469&BBOX=5474086,5637004,5479155,5643637
&WIDTH=500&HEIGHT=708&FORMAT=image/png
```

Dieser Request sowie die entsprechenden Anfragen mit dem SLD-Dokument *MapServer_simpleSLD-100.xml*, welches wiederum in der digitalen Anlage zu finden ist, liefert korrekterweise dieselbe Karte wie in *Abbildung 5-2*.

Styled Layer Descriptor / Symbology Encoding 1.0.0

Um das Bild aus *Abbildung 5-3* mittels *UserStyle* in SLD 1.0.0 auch mit MapServer zu erhalten, muss ein entsprechendes SLD-Dokument auf einen Webserver abgelegt werden, da MapServer keine zu Testzwecken lokal abgelegten Files verarbeiten kann.

Listing 5-14: MapServer WMS - NamedLayer mit UserStyle

```
http://localhost/cgi-bin/mapserv?
VERSION=1.1.1&REQUEST=GetMap&LAYERS=ALK
&SRS=EPSG:31469&BBOX=5474086,5637004,5479155,5643637
&WIDTH=500&HEIGHT=708&FORMAT=image/png&SLD=http://
www.htw-dresden.de/~s58783/MapServer_UserStyleSLD-100.xml
```

Während dieser Request noch funktioniert, wird bei der Integration eines *UserLayer* ein Exception-Dokument zurückgegeben, da der Dienst diese Arbeitsweise nicht anbietet, was so auch der Dokumentation [17] zu entnehmen ist. Dazu kommen die Abfragen aus *Listing 5-7* und *Listing 5-8* wieder zum Einsatz, wobei die deegree-durch die MapServer-WMS-URL ersetzt wird und die SLD-Dokumente über einen Webserver aufgerufen werden. Nach dem Fehlschlagen des *UserLayer*-Tests kann auf den *InlineFeature*-Versuch verzichtet werden.

Styled Layer Descriptor / Symbology Encoding 1.1.0

Der Einsatz von SLD 1.1.0 ergibt das gleiche Ergebnis wie mit SLD 1.0.0. Die Requests sind identisch, bis auf die Tatsache, dass die SLD-Dokumente, die sich ebenfalls in der digitalen Anlage befinden, auf die neuere Version umgeschrieben wurden. SLD 1.1.0 wird also unterstützt.

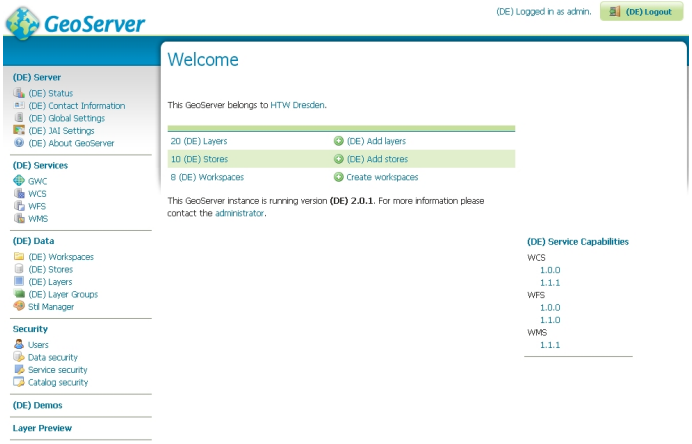
Filter Encoding

Bei einem Feature Portrayal Service werden Filtermethoden in Verbindung mit benutzerdefinierten Ebenen genutzt. Aufgrund der fehlenden Möglichkeit, diese zu erstellen, ist eine Untersuchung der Filtereigenschaften an dieser Stelle überflüssig.

5.1.4 Untersuchung des GeoServer WMS

Ein weiteres, wie schon deegree in Java geschriebenes, Open Source Produkt heißt GeoServer [28]. Die ersten Entwicklungsschritte wurden im Jahr 2001 von The Open Planning Project (TOPP) in New York durchgeführt. GeoServer implementiert die WMS-, WFS- und WCS-Spezifikation des Open Geospatial Consortium. Im Gegensatz zu den beiden anderen Frameworks wird dieser Dienst über eine grafische Benutzeroberfläche im Browser konfiguriert [6].

Abbildung 5-7: GeoServer GUI



WMS-Capabilities

Das *UserDefinedSymbolization*-Element gleicht dem des deegree Servers. Wie man sich also nach dem folgenden Request vergewissern kann, werden außer *InlineFeatureData* alle FPS-Voraussetzungen erfüllt.

Listing 5-15: GeoServer - GetCapabilities-Request

```
http://localhost:8080/geoserver/wms?
SERVICE=WMS&REQUEST=GetCapabilities
```

WMS-GetMap-Interface

Die GetMap-Untersuchungen gleichen sowohl in der Durchführung als auch dem positiven Ergebnis denen der beiden vorangegangenen Software-Lösungen. Deshalb wird an dieser Stelle nicht noch einmal näher darauf eingegangen.

Styled Layer Descriptor / Symbology Encoding 1.0.0

Listing 5-16: GeoServer - NamedLayer mit UserStyle

```
http://localhost:8080/geoserver/wms?
VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:31469
&BBOX=5474086,5637004,5479155,5643637&WIDTH=500
&HEIGHT=708&FORMAT=image/png&SLD=
file:///E:/xampplite/_Testdaten/GeoServer_UserStyleSLD-100.xml
```

Die Prüfungen mit den vordefinierten Ebenen, einmal in Verbindung mit *NamedStyle* und einmal mit *UserStyle (Listing 5-16)*, werden bestanden.

Für die Kontrolle der *UserLayer*-Möglichkeiten wurden die relevanten Listings aus *Abschnitt 5.1.2* bemüht, wobei die Dienst-URL mit der des GeoServer ersetzt wurde. Zu beachten ist, dass bei diesem Framework unbedingt ein *Name* für jeden *UserLayer* vergeben werden muss. Bei keinem der Tests konnte ein Bild generiert werden. Während GeoServer bei einem MapServer WFS als Remote Web Feature Service *(Listing 5-7 und Listing 5-8)* meldet, den angefragten FeatureType

nicht finden zu können, wird bei einem deegree WFS *(Listing 5-6)* die in *Listing 5-17* visualisierte Fehlermeldung zurückgegeben.

Listing 5-17: GeoServer - Exception bei UserLayer (deegree WFS)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE ServiceExceptionReport SYSTEM "http://localhost:8080/
geoserver/schemas/wms/1.1.1/WMS_exception_1_1_1.dtd">
<ServiceExceptionReport version="1.1.1">
 <ServiceException code="Internal error ">
    org.vfny.geoserver.wms.WmsException:
        Rendering process failed
    Rendering process failed
    Exception rendering layer DefaultMapLayer[app:
        CountyBoundaries_edited, VISIBLE, UNSELECTED,
        style=StyleImpl[ name=Default Styler], data=
        org.geotools.data.wfs.v1_1_0.WFSFeatureSource
        @309a10, query=Query:
    feature type: app:CountyBoundaries_edited
    filter: Filter.INCLUDE
    [properties:  ALL ]]
    Server does not support 'text/xml;
        subtype=gml/3.1.1' output format: []
 </ServiceException>
</ServiceExceptionReport>
```

Da dieses Ausgabeformat jedoch eigentlich von deegree unterstützt wird, hilft diese XML-Exception im Hinblick auf das Aufspüren der Fehlerursache nicht weiter.

Die Variante, deren korrekte Funktionsweise am ehesten zu erwarten wäre, ist der Zugriff auf einen *RemoteOWS*, der selbst eine GeoServer-Implementierung darstellt, also die Umsetzung von *Listing 5-9* mit angepasster Service-URL. Doch auch hier kann kein Bild generiert werden.

Listing 5-18: GeoServer - Exception bei UserLayer (GeoServer WFS)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE ServiceExceptionReport SYSTEM "http://localhost:8080/
geoserver/schemas/wms/1.1.1/WMS_exception_1_1_1.dtd">
<ServiceExceptionReport version="1.1.1">
 <ServiceException code="Internal error ">
     org.vfny.geoserver.wms.WmsException:
         Rendering process failed
     Rendering process failed
     Error rendering feature
     Could not aquire feature:java.net.SocketTimeoutException:
         Read timed out
     Read timed out
     Read timed out
 </ServiceException>
</ServiceExceptionReport>
```

Diese Fehlermeldung besagt, dass bei GeoServer ein zeitliches Abbruchkriterium programmiert wurde, das eintritt, bevor die Daten komplett vom fremden WFS besorgt werden konnten. Da ein Feature Portrayal Service jedoch in der Lage sein muss, auch größere Datenmengen darstellen zu können, ist ein solches Time-out kontraproduktiv. Wird der WFS als feste Datenquelle in GeoServer definiert, kann dieses Zeitkriterium konfiguriert werden. Allerdings wurde auf Anfrage in der GeoServer Community bestätigt, dass die Möglichkeit der Anpassung des Wertes bei benutzerdefinierten Ebenen derzeit nicht vorgesehen ist. Es besteht jedoch die Möglichkeit den Quellcode mit Hilfe der Entwickler dementsprechend umzuschreiben.

Dasselbe Problem tritt beim Einfügen eines *InlineFeature*-Elements auf. Nur kleine Datenmengen können hier visualisiert werden. Im Beispiel aus *Listing 5-19* handelt es sich um ein simples Rechteck.

Listing 5-19: GeoServer_InlineFeature_SLD-100.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.0.0"
xmlns:gml=http://www.opengis.net/gml
xmlns:ogc=http://www.opengis.net/ogc
xmlns="http://www.opengis.net/sld">
 <UserLayer>
    <Name>Rechteck</Name>
    <InlineFeature>
        <FeatureCollection>
            <featureMember>
                <BodyPart>
                    <Type>Face</Type>
                    <polygonProperty>
                        <gml:Polygon>
                            <gml:outerBoundaryIs>
                                <gml:LinearRing>
                                    <gml:coordinates>
                                    -10,10 10,10 10,-10 -10,-10 -10,10
                                    </gml:coordinates>
                                </gml:LinearRing>
                            </gml:outerBoundaryIs>
                        </gml:Polygon>
                    </polygonProperty>
                </BodyPart>
            </featureMember>
        </FeatureCollection>
    </InlineFeature>
    <UserStyle>
        [...]
    </UserStyle>
 </UserLayer>
</StyledLayerDescriptor>
```

Listing 5-20: GeoServer - UserLayer (InlineFeature)

```
http://localhost:8080/geoserver/wms?
REQUEST=GetMap&VERSION=1.1.1&SERVICE=WMS
&FORMAT=image/png&BBOX=-12,-12,12,12
&SRS=EPSG:4326&WIDTH=60&HEIGHT=60&SLD=
file:///E:/xampplite/_Testdaten/GeoServer_InlineFeature_SLD-100.xml
```

Abbildung 5-8: Karte mit InlineFeature

Allerdings handelt es sich hier um keine echte InlineFeature-Lösung, da SLD 1.0.0 zum Einsatz kommt. InlineFeature wird jedoch erst in SLD 1.1.0 definiert, wo auch die entsprechende XML-Syntax ein wenig anders aussieht.

Styled Layer Descriptor / Symbology Encoding 1.1.0

Wie schon deegree, unterstützt auch GeoServer SLD 1.1.0 nicht. Wird beispielsweise eine Karte angefordert, welche per *UserStyle* in dieser Version dargestellt werden soll, erhält man diese Karte in einem Standard-Stil.

Filter Encoding

Da in erster Linie nur größere Datenmengen gefiltert werden, die ja bei GeoServer nicht angefordert werden können, wurde auch hier auf die Untersuchung von Filter Encoding verzichtet.

5.1.5 Ergebnisse und Software-Entscheidung

Die in *Tabelle 5-2* dargestellten Testergebnisse waren die Grundlage für die Auswahl eines Frameworks. Alle obligatorischen Elemente sind fett hervorgehoben.

Tabelle 5-2: Testergebnisse

Bezeichnung		deegree **deegree 2.3**	MapServer **MapServer 5.6.1**	GeoServer **GeoServer 2.0.1**
WMS-Capabilities	***SupportSLD***	✓	✓	✓
	UserLayer	✓	✗	✓
	UserStyle	✓	✓	✓
	RemoteWFS	✓	✗	✓
	InlineFeatureData	✗	✗	✗
Get-Map	***einfach***	✓	✓	✓
	SLD	✓	✓	✓
	SLD_BODY	✓	✓	✓
SLD/SE 1.0.0	*NamedStyle*	✓	✓	✓
	UserStyle	✓	✓	✓
	UserLayer: WFS 1.0.0	✗	✗	✓*2
	UserLayer: WFS 1.1.0	✓*1	✗	
	InlineFeature	✗	✗	✓*3
SLD/SE 1.1.0	*NamedStyle*	✗	✓	✗
	UserStyle	✗	✓	✗
	UserLayer: WFS 1.0.0	✗	✗	✗
	UserLayer: WFS 1.1.0	✗	✗	✗
	InlineFeature	✗	✗	✗
Filter Encoding		✓	–*4	–*4

*1 nur im Zusammenhang mit deegree WFS
*2 nur bei kleinen Datenmengen, eine Unterscheidung der WFS-Versionen ist anhand der Fehlermeldungen nicht möglich
*3 keine echte InlineFeature-Lösung, da SLD 1.0.0
*4 Test in FPS-Konstellation nicht möglich

Während MapServer aufgrund der Resultate als Feature Portrayal Service nicht in Frage kommen kann, liegen deegree und GeoServer beim Stand der Implementierung recht nah beieinander. Beide Softwarelösungen unterstützen zwar nur SLD 1.0.0, sind dafür aber in der Lage *UserLayer* zu verarbeiten. Entscheidender Nachteil von GeoServer ist, dass aufgrund des in *Abschnitt 5.1.4* genannten Time-outs die Funktionsweise als FPS nicht wirklich getestet werden konnte. Somit besteht die Gefahr, nach einer Anpassung des Quellcodes auf

weitere Probleme zu stoßen. Bei deegree hingegen ist völlig klar, was unter welchen Umständen funktioniert und was nicht. Ein weiteres Argument für den Einsatz von deegree ist außerdem die Tatsache, dass im Geoinformatik-Labor der Fakultät Geoinformation der Hochschule für Technik und Wirtschaft Dresden bisher bereits einige deegree-Dienste aufgesetzt wurden. GeoServer wäre in dieser Umgebung eine Umstellung und passt daher nicht optimal in die Infrastruktur des Labors.

Aus den oben genannten Gründen fiel also die Entscheidung, den Feature Portrayal Service mithilfe des deegree Frameworks zu implementieren.

5.2 Installation und Konfiguration

Die folgenden Installationsschritte entstammen teilweise der Projektarbeit [4], die dieser Diplomarbeit vorangegangenen ist.

Alle benötigten Komponenten wurden auf der virtuellen Maschine mit der IP-Adresse *141.56.141.5* im Labor Geoinformatik installiert. Dazu wurde eine Remotedesktopverbindung eingerichtet.

Abbildung 5-9: Remotedesktopverbindung

5.2.1 Java SE Runtime Environment (JRE)

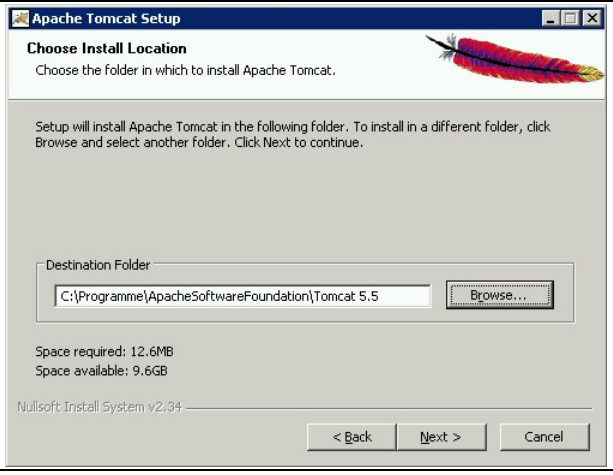
Da das deegree Framework mit Java Servlets arbeitet, ist zumindest eine entsprechende Laufzeitumgebung notwendig. Diese kann beispielsweise auf der Java-Webseite von Sun Microsystems [37] heruntergeladen und anschließend installiert werden. Im Zuge dieser Diplomarbeit wurde mit der Version JRE 6, Update 17, gearbeitet.

5.2.2 Apache Tomcat

Um diese Servlets umzusetzen, wird eine so genannte Servlet Engine benötigt. In diesem Fall handelt es sich dabei um den Apache Tomcat Server [24], der als Standalone-Server eingesetzt wurde, wobei es auch möglich wäre, ihn auf einen Apache HTTP Server [23] aufzusetzen. Zur Installation wurde der Windows Service Installer zur Version 5.5.28 von der Tomcat-Homepage heruntergeladen.

Beim Installationsvorgang wurde der Installationsordner insofern angepasst, dass keine Leerzeichen enthalten sind, da dies eventuell zu Problemen führen könnte.

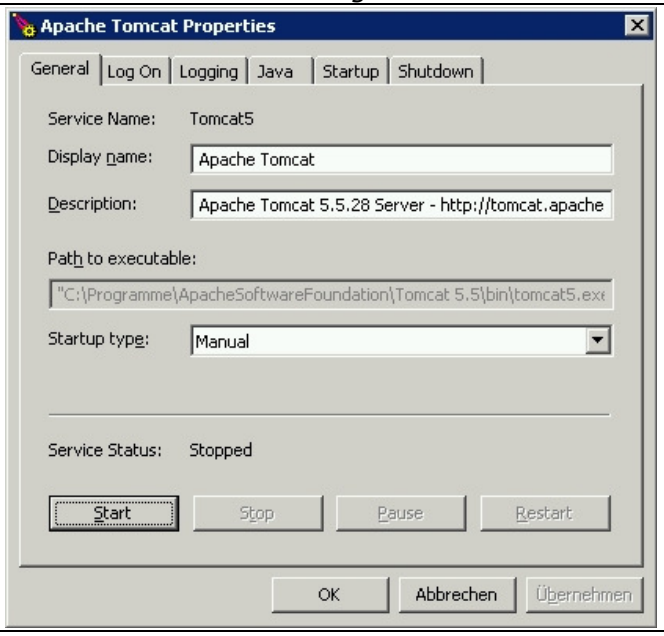
Abbildung 5-10: Tomcat - Installationsordner



Als Standard-Port wird Port 8080 vorgeschlagen. Auch das vorgeschlagene Administrator-Login wurde unverändert übernommen. In einem weiteren Schritt wird der Nutzer nach dem Pfad der JRE-Installation gefragt *(siehe Abschnitt 5.2.1)*.

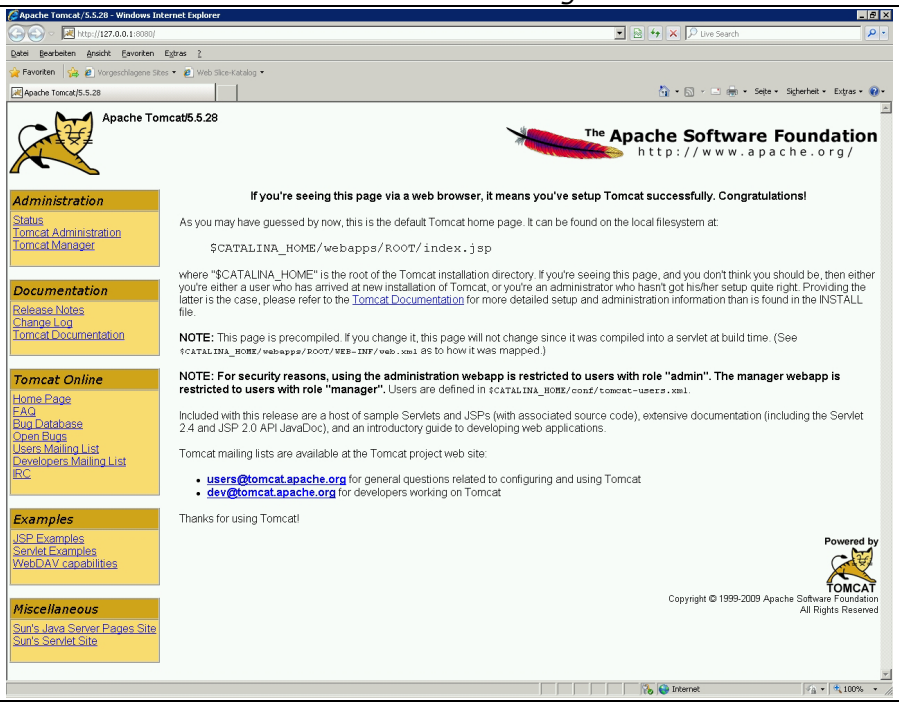
Ist die Installation abgeschlossen, kann Tomcat über *Startmenü → Programme → Apache Tomcat 5.5 → Monitor Tomcat* geöffnet werden. Es erscheint ein entsprechendes Symbol im System Tray, über welches man per *Rechtsklick → Start service* den Tomcat Server starten kann (alternativ: *Doppelklick auf das Symbol → Reiter General → Start*).

Abbildung 5-11: Tomcat - Start des Server



Ist dies geschehen, kann im Browser unter der URL *http://127.0.0.1:8080/* die Tomcat-Startseite aufgerufen werden. Die IP-Adresse *127.0.0.1* steht für den *localhost* und könnte auch durch diesen Begriff ersetzt werden. Das Anzeigen der Seite aus *Abbildung 5-12* ist gleichbedeutend mit der erfolgreichen Installation des Apache Tomcat.

Abbildung 5-12: Tomcat - Startseite



Möchte man die Angabe des Ports (8080) vermeiden, muss Tomcat über Port 80 erreichbar sein, was entweder während des Installationsvorganges oder danach in der Datei *server.xml* im *conf*-Verzeichnis des Tomcat eingestellt werden kann.

Abbildung 5-13: Tomcat - Port-Konfiguration in server.xml

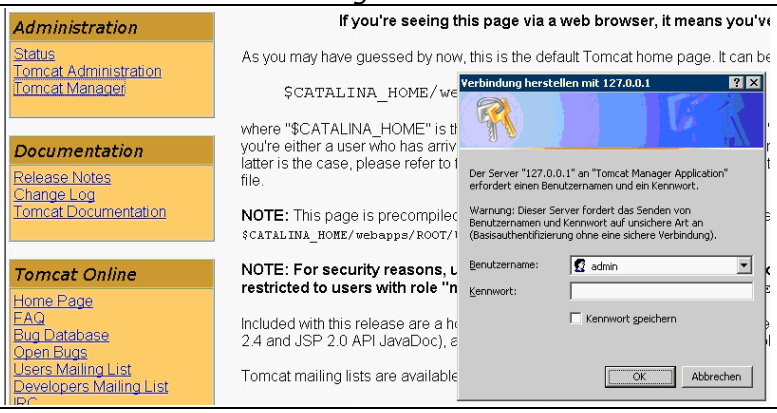
```
94      <Connector
95  port="80"                maxHttpHeaderSize="8192"
96                  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
97                  enableLookups="false" redirectPort="8443" acceptCount="100"
98                  connectionTimeout="20000" disableUploadTimeout="true" />
```

Dazu darf Port 80 natürlich noch nicht belegt sein, beispielsweise durch einen Apache HTTP Server. In diesem Fall wäre allerdings auch eine Verbindung zwischen Apache und Tomcat mittels dem Apache-Modul *mod_jk* möglich, welches dann den Datenaustausch zwischen beiden Komponenten regelt. Der Tomcat Server wird in dieser Arbeit ab sofort ohne Angabe des Ports angesprochen, ist also unter *http://127.0.0.1/* erreichbar. Damit ist die Installation des Tomcat endgültig fertig gestellt.

5.2.3 deegree Web Map Service

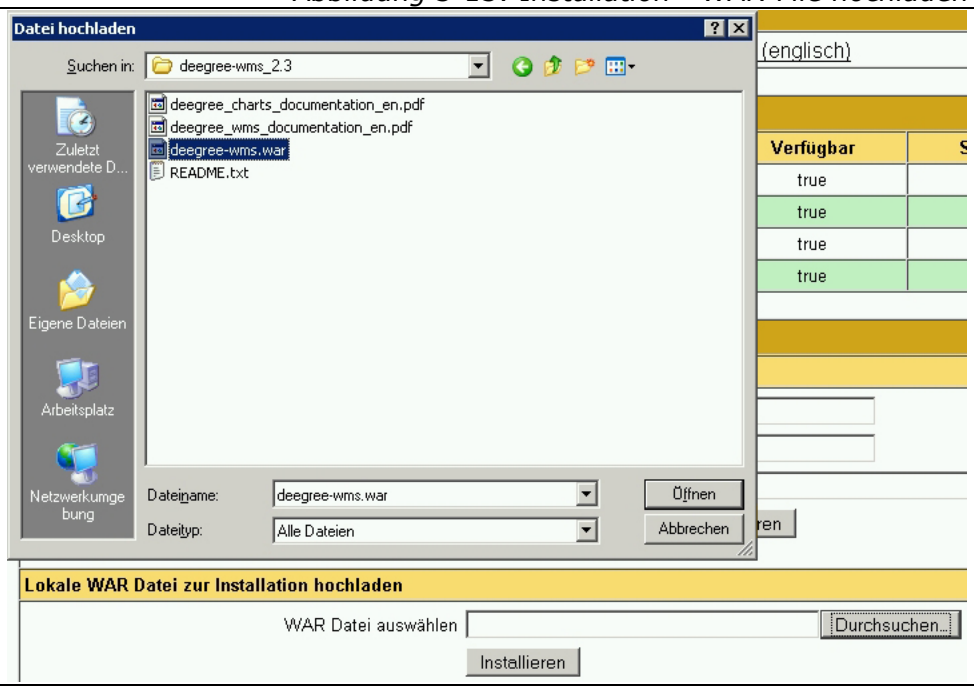
Der WMS der Version deegree 2.3 [12] steht auf der deegree Webseite [25] zum Herunterladen bereit. Nach dem Download der ZIP-Datei kann diese entpackt werden. Sie enthält eine Datei mit der Endung *.war*, was im Prinzip ein weiteres Archiv ist, das in den Tomcat Server integriert werden muss. Dazu gibt es zwei Möglichkeiten. Eine Variante wäre, dieses Archiv in das *webapps*-Verzeichnis des Tomcat zu kopieren und den Server anschließend neu zu starten. Die Alternative ist die Installation über den Tomcat Manager, der über einen Klick auf den entsprechenden Link auf der Tomcat Startseite aufgerufen wird. Anschließend muss der Benutzername und das Kennwort eingegeben werden, welche bei der Installation des Tomcat Servers festgelegt wurden *(Abschnitt 5.2.2)*. Auch diese Angaben könnten natürlich noch nachträglich geändert werden. Dazu müsste die Datei *tomcat-users.xml* im *conf*-Verzeichnis editiert werden.

Abbildung 5-14: Installation - Tomcat Manager



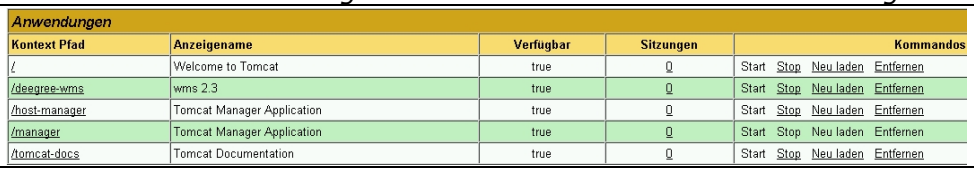
Anschließend werden alle bereits in Tomcat integrierten Anwendungen aufgeführt und es können lokale WAR-Dateien zur Installation hochgeladen werden.

Abbildung 5-15: Installation - WAR-File hochladen



Nun wird der Web Map Service installiert und mit aufgelistet. Ein Neustart ist in diesem Fall nicht notwendig.

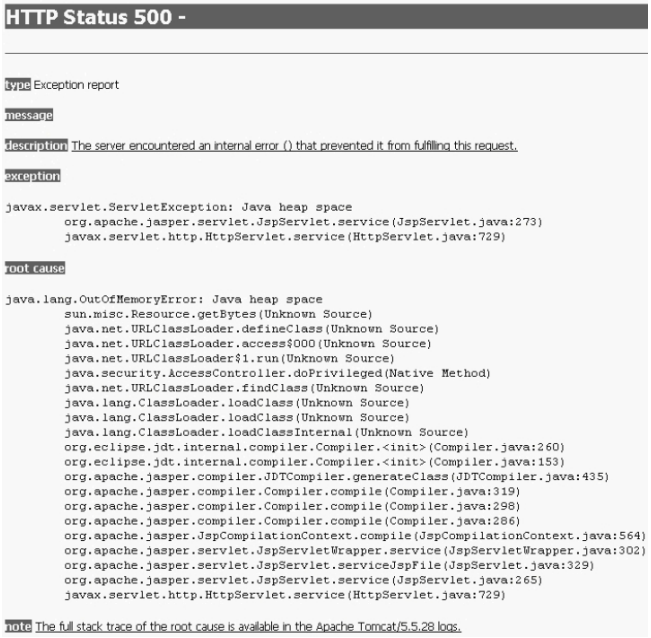
Abbildung 5-16: Installation - Tomcat-Anwendungsliste



Anwendungen				
Kontext Pfad	Anzeigename	Verfügbar	Sitzungen	Kommandos
/	Welcome to Tomcat	true	0	Start Stop Neu laden Entfernen
/deegree-wms	wms 2.3	true	0	Start Stop Neu laden Entfernen
/host-manager	Tomcat Manager Application	true	0	Start Stop Neu laden Entfernen
/manager	Tomcat Manager Application	true	0	Start Stop Neu laden Entfernen
/tomcat-docs	Tomcat Documentation	true	0	Start Stop Neu laden Entfernen

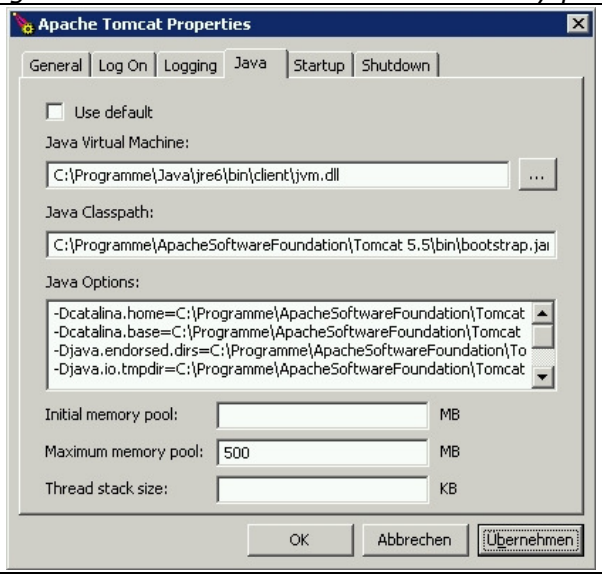
Damit ist die Installation eines vorkonfigurierten WMS abgeschlossen, dessen Startseite unter *http://127.0.0.1/deegree-wms/* erreichbar sein sollte. Allerdings erhält man zunächst eine Fehlermeldung, die auf ein Problem mit dem Java Heap Space hinweist. Das bedeutet, dass der zugewiesene dynamische Speicherbereich nicht ausreicht, um Tomcat mit deegree laufen zu lassen. Standardmäßig sind hier 64 MB eingestellt.

Abbildung 5-17: Installation - Java Heap Space Exception



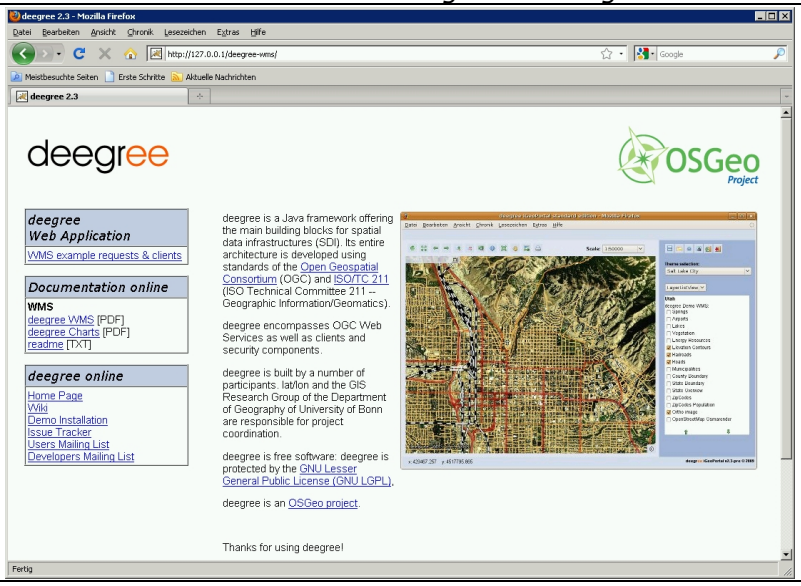
Durch Wählen der Option *Configure…* im Kontextmenü nach einem Rechtsklick auf das System Tray-Icon des Tomcat gelangt man in die Oberfläche, mit deren Hilfe dieses Problem behoben werden kann. Dazu ist dort im Reiter *Java* unter *Maximum memory pool* der zu vergebende Speicherbereich einzutragen. 500 MB sollten in jedem Fall ausreichend sein.

Abbildung 5-18: Installation - Maximum memory pool des Tomcat



Nun kann der Server wieder neu gestartet werden und die Startseite des deegree WMS aufgerufen werden. Hier kann der Web Map Service unter seinen Vorkonfigurationen getestet werden.

Abbildung 5-19: deegree FPS - Startseite



Als erster Schritt einer eigenen Konfiguration wurde in der Datei *web.xml* aus dem *WEB-INF*-Verzeichnis des WMS nach dem einleitenden Tag mit dem Namen *servlet-mapping* das URL-Pattern des OWS geändert.

Listing 5-21: deegree FPS - URL-Pattern in web.xml

```
<servlet-mapping>
 <servlet-name>owservice</servlet-name>
 <url-pattern>/fps</url-pattern>
</servlet-mapping>
```

Das Servlet, welches zuvor im servlet-Abschnitt deklariert wurde, wird somit diesem URL-Pattern zugeordnet. Demzufolge lautet nun die komplette URL des Dienstes *http://141.56.141.5/deegree-wms/fps*, an welche später das jeweilige Interface mit den zugehörigen Request-Parametern angefügt werden muss.

Anschließend wurden die Capabilities im *wms-configuration.xml*-Dokument umkonfiguriert. Diese Datei ist im *deegree-wms*-Verzeichnis unter *\WEB-INF\conf\wms* zu finden. Dort wurden zunächst die *DefaultOnlineResource* und die jeweilige *OnlineResource* für die verschiedenen Schnittstellen durch den eben genannten Link ersetzt.

Besondere Beachtung gilt für die Konfiguration mit dem Ziel eines Feature Portrayal Service der Zeile aus *Listing 5-22*.

Listing 5-22: deegree FPS - UserDefinedSymbolization

```
<UserDefinedSymbolization
SupportSLD="1" UserLayer="1" UserStyle="1" RemoteWFS="1" />
```

Hier müssen alle vier Attribute den Wert *1*, also *TRUE*, bekommen.

Eigentlich ist für den Betrieb des WMS als Feature Portrayal Service kein *NamedLayer* notwendig. Da für ein reibungsloses Arbeiten des deegree WMS aber ein Layer erwartet wird, wurde hier ein leerer Layer mit den geforderten Mindestangaben definiert.

Listing 5-23: deegree FPS - leere Ebene

```
<Layer queryable="1" cascaded="0" noSubsets="0"
        xmlns:app="http://www.deegree.org/app">
 <Name>deegree FPS</Name>
 <Title>deegree FPS</Title>
 <LatLonBoundingBox miny="-90" maxy="90" minx="-180"
        maxx="180" />
</Layer>
```

Alle getätigten Anpassungen in der Datei *wms_configuration.xml* können der *Anlage A* entnommen werden.

Da bei diesem Service keine Daten einzubinden sind, ist dessen Konfiguration hiermit abgeschlossen und es können bereits seine Capabilities angefordert werden.

Listing 5-24: deegree FPS - GetCapabilities-Request

```
http://127.0.0.1/deegree-wms/fps?
SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities
```

Zuvor ist allerdings wiederum ein Neustart des Tomcat erforderlich.

Damit ist der Feature Portrayal Service auf deegree-Basis bereits fertig konfiguriert. Die komplette Implementierung befindet sich in der digitalen *Anlage D (Ordner \deegree-wms\)*.

5.3 Defizitanalyse

Trotz der theoretischen Möglichkeit, deegree als Feature Portrayal Service nutzen zu können, gibt es noch einige Aspekte, die dem im Wege stehen. Im Großen und Ganzen sind diese Punkte der *Tabelle 5-2* zu entnehmen.

In erster Linie handelt es sich um die Erkenntnis, dass nicht alle Datenquellen für einen UserLayer in Frage kommen, obwohl ein Feature Portrayal Service selbstverständlich in allen denkbaren Kombinationen funktionstüchtig sein muss. Die Lösung dieses Problems ist hauptsächlicher Gegenstand des nächsten Kapitels.

Auf die Unterstützung von InlineFeature kann verzichtet werden, da es sich um ein optionales Element handelt *(siehe Abschnitt 5.1.1)*. Außerdem müsste es mit SLD 1.0.0-Mitteln implementiert werden, da deegree derzeit nur diese Version umgesetzt hat. Somit würde es sich lediglich um eine Pseudo-Lösung hierfür handeln.

Der fehlende Support von SLD 1.1.0 resultiert aus dem derzeitigen Entwicklungsstand von deegree. Da zu erwarten ist, dass sich dies in kommenden Veröffentlichungen ändern wird *(siehe Abschnitt 8.3)*,

wäre ein komplexer Workaround für diesen Sachverhalt mit einem unverhältnismäßig großen Aufwand verbunden.

Aus den genannten Gründen soll nur die Lösung des ersten Problems, also die Integration beliebiger WFS, primärer Gegenstand dieser Diplomarbeit sein. Um diesbezüglich eine exakte Fehleranalyse durchführen zu können, wurde ein WFS-Servlet entwickelt, in welches jeweils immer die DescribeFeatureType- und GetFeature-Response-Dokumente der verschiedenen Web Feature Services eingebunden wurden. Anschließend konnten GetMap-Requests an den deegree FPS gesendet werden, die innerhalb der SLD-Syntax auf dieses WFS-Servlet anstelle der eigentlichen Dienste zeigen. So konnten zunächst die originären Response-Dokumente integriert werden, welche man bei den Anfragen an den RemoteWFS erhält. Das führt bei einem GetMap-Request mit jeweils angepasster SLD-Datei zu demselben Fehler, wie beim Zugriff auf den echten RemoteWFS. Anschließend wurden die Servlet-Antworten systematisch verändert, bis tatsächlich eine Karte im Browser angezeigt wurde. Auf diese Weise konnte ermittelt werden, an welcher Stelle die Fehler verursacht werden.

Bisher wurden Probleme mit MapServer und Geoserver als RemoteWFS festgestellt, deswegen wurden diese beiden WFS-Lösungen getestet. Die Ergebnisse sind in *Tabelle 5-3* zusammengefasst. Der Eclipse-Workspace zu dem zur Fehlersuche verwendeten WFS-Servlet ist in der digitalen *Anlage D (Ordner \workspace\TestWorkaround\)* zu finden.

Tabelle 5-3: Defizitanalyse deegree FPS

RemoteWFS	Fehlerbeschreibung	Lösung
MapServer	DescribeFeatureType-Response: Element-Typen werden mit *type="string"* angegeben	ersetzen durch *type="xsd:string"*
	GML-Datei: URL zum DescribeFeatureType-Request zeigt nicht auf WFS-Fassade	durch URL der WFS-Fassade ersetzen
	GML-Datei: deegree versteht keinen OUTPUTFORMAT-Parameter im Describe-FeatureType-Request	OUTPUTFORMAT-Parameter entfernen
GeoServer	GML-Datei: URL zum DescribeFeatureType-Request zeigt nicht auf WFS-Fassade	durch URL der WFS-Fassade ersetzen
	GML-Datei: Angabe des Koordinatenrefernezsystems (z.B. *urn:x-ogc:def:crs:EPSG:31469*)	entsprechender Eintrag in deegree CRS Registry muss vorhanden sein oder Nutzen der WFS 1.0.0 - Schnittstelle
	GML-Datei: Rechts- und Hochwert sind vertauscht (Ergebnis ist verdreht)	Nutzen der WFS 1.0.0 - Schnittstelle
	weitere GML-Fehler	Nutzen der WFS 1.0.0 - Schnittstelle
allgemein	WFS 1.0.0 wird nicht unterstützt	Fassade muss WFS 1.0.0/1.1.0 für RemoteWFS und WFS 1.1.0 zur Kommunikation mit deegree WMS unterstützen

Der hier mit GeoServer aufgetretene Fall, dass deegree die Angabe des Koordinatenreferenzsystems nicht kennt, kann rein theoretisch überall auftreten. Die Lösung liegt hier im File *deegree-crs-configuration.xml*.

Diese Datei kann dem deegree SVN entnommen werden und an eigene Belange angepasst werden. Fehlende CRS-Angaben müssen hier ergänzt werden. Wird diese Datei in das Verzeichnis *WEB-INF**classes*\ des deegree WMS kopiert, werden die entsprechenden Angaben, welche im Archiv *WEB-INF**lib**deegree2.jar* gemacht sind, überschrieben.

6 WORKAROUND

Wie bereits in der Defizitanalyse aus dem *Abschnitt 5.3* erwähnt, muss zur Bereitstellung eines ernsthaften Feature Portrayal Service insbesondere das Problem mit unterschiedlich implementierten RemoteWFS gelöst werden.

6.1 Möglichkeiten

Nachfolgend werden diverse Lösungsmöglichkeiten diskutiert, aus denen ein Workaround-Ansatz ausgewählt wird.

6.1.1 Kommerzielle Produkte

Die Alternative der kommerziellen WMS-Lösungen soll an dieser Stelle nur der Vollständigkeit halber genannt werden. Da sich diese Diplomarbeit jedoch auf Open Source Software stützen soll, wurden mögliche Produkte nicht untersucht.

6.1.2 Veränderung des deegree-Codes

Rein theoretisch wäre es sinnvoll, den Hebel direkt an der Quelle anzusetzen. So könnte man versuchen, mit Hilfe der deegree Community den Quellcode selbst umzuschreiben, um die vorhandenen Probleme aus dem Weg zu räumen. Eine Anpassung durch die Entwickler selbst ist ohne eine entsprechende Finanzierung nicht zu erwarten.

In der Praxis jedoch stellt sich diese Variante als relativ schwierig dar. Einerseits wäre es sehr aufwändig, sich in den deegree-Code und die

Funktionsweise aller Komponenten hineinzudenken, andererseits wäre man hier sehr auf die Hilfe der Entwickler angewiesen. Das hat zur Folge, dass sich a priori kaum ein Bearbeitungszeitraum abschätzen lässt.

6.1.3 Kaskade

Ein weiterer nur theoretischer Ansatz ist die Einbindung der nicht unterstützten fremden Web Feature Services durch eine WFS-Kaskade. Dabei könnte ein deegree WFS aufgesetzt werden, da dieser ja als einziger Web Feature Service als RemoteOWS in Frage kommt *(siehe Abschnitt 5.1.2)*. Anschließend müsste dieser so konfiguriert werden, dass ein bestimmter anderer WFS als eingebundene Datenquelle dient.

Doch auch diese Möglichkeit kommt nicht in Betracht. Zunächst einmal würde es sich hierbei um einen sehr umständlichen Workaround handeln, da die notwendigen Kaskaden nicht dynamisch erzeugt werden können. Bei jeder neuen Kaskade müsste der deegree WFS umkonfiguriert werden, was einen Neustart der Servlet Engine mit sich bringt. Außerdem wird das Einbinden eines WFS als Datenquelle des eigenen WFS von deegree derzeit nicht angeboten. Dieser Umstand führt zu dem Entschluss, diese Möglichkeit zu verwerfen.

6.1.4 Fassade

Neben den nicht zufrieden stellenden vorherigen Überlegungen bleibt noch die Möglichkeit einer so genannten Fassade. Die Fassade ist ein Entwurfsmuster aus der objektorientierten Softwareentwicklung und wird beispielsweise im Architekturkonzept der gdi.initiative.sachsen erwähnt [5]. Sie verfolgt den Zweck, spezialisierte Anforderungen an Schnittstellen zu bedienen.

Im konkreten Fall würde der Feature Portrayal Service nicht direkt mit den fremden Web Feature Services kommunizieren, sondern mit einer WFS-Fassade, welche die notwendigen Schnittstellen anbietet. Diese Architektur hat den Vorteil, dass die Fassade einen ganz normalen WFS darstellt, wobei intern zunächst die Daten der eigentlichen RemoteWFS besorgt und dahingehend verändert werden, dass auch der deegree WMS mit diesen umgehen kann.

Da diese Möglichkeit als einziger Ansatz wirklich als guter Workaround in Frage kommt, wird deren Umsetzung in den folgenden Abschnitten erläutert.

6.2 Realisierung einer FPS-Fassade

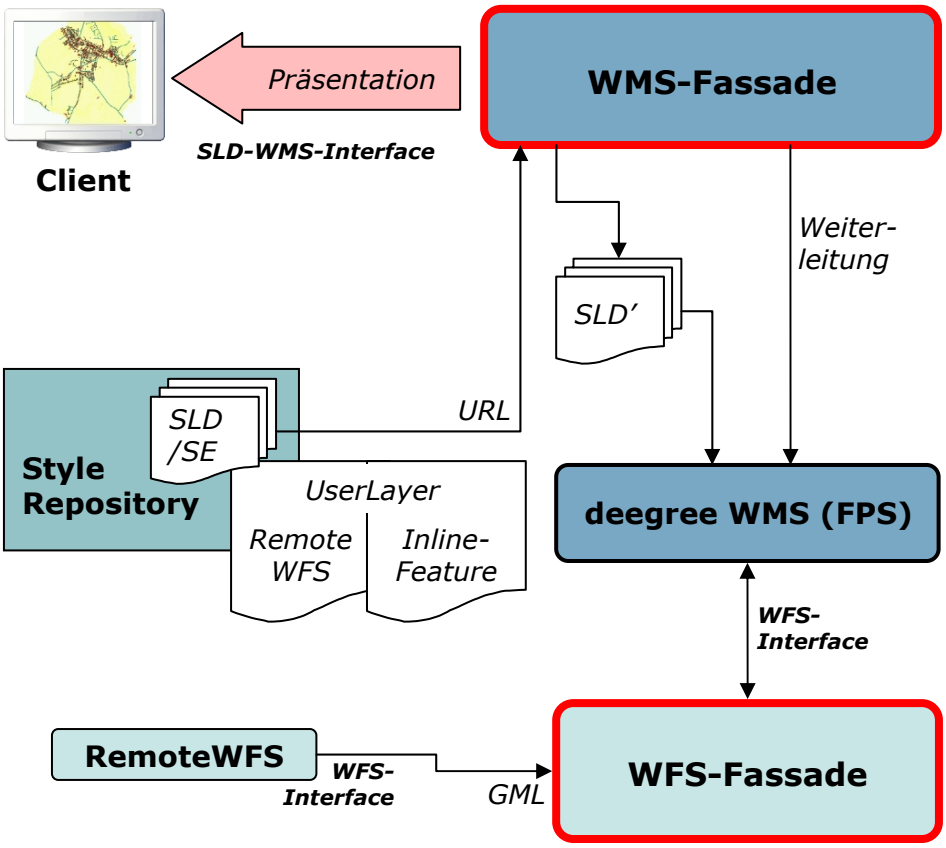
6.2.1 Konzeption

Um die Arbeitsweise der in *Abschnitt 6.1.4* erläuterten Fassade verständlich zu machen, wurde der einfache Feature Portrayal Service in der *Abbildung 6-1* entsprechend erweitert.

Die FPS-Fassade selbst beinhaltet zwei Fassaden, die auch getrennt voneinander implementiert werden könnten. Da sie aber möglichst aufeinander abgestimmt sein sollten, sind sie in einem Paket zusammengefasst.

Anfangs existierte auch der Gedanke, den Client-Request direkt an den deegree WMS weiterzuleiten, wenn dieser so verarbeitet werden kann. Voraussetzung dafür wäre, dass bekannt ist, ob der RemoteWFS eine deegree-Implementierung ist oder nicht. Da es allerdings keine sichere Methode gibt, um dies herauszufinden, kann dieser Aspekt nicht umgesetzt werden.

Abbildung 6-1: Arbeitsweise der FPS-Fassade



WMS-Fassade

Die WMS-Fassade ist notwendig, da sie dafür sorgt, dass die Datenquelle im SLD-Dokument verändert wird, bevor der Request an den eigentlichen Feature Portrayal Service, den deegree WMS, weitergeleitet wird. Das *RemoteOWS*-Element im editierten SLD-File beinhaltet die URL zur WFS-Fassade. So wird umgangen, dass der deegree WMS direkt mit dem RemoteWFS kommuniziert.

Für den Nutzer ist diese WMS-Fassade der Feature Portrayal Service, der direkt angesprochen wird. Dem Client kann völlig egal sein, was hinter dieser Fassade geschieht, denn der Request zum Anfordern einer Karte geht an die URL der Fassade und die SLD-Syntax

beinhaltet ganz normal die Datenquelle, also den gewünschten RemoteWFS.

WFS-Fassade

Da dem deegree WMS die URL zur WFS-Fassade per SLD mitgeteilt wird, muss diese Fassade die deegree-Requests verarbeiten, indem die Daten vom eigentlichen RemoteWFS abgefragt, gegebenenfalls bearbeitet und an den deegree WMS zurückgegeben werden, um dann von diesem visualisiert zu werden.

Ablauf

Um die FPS-Fassade entwickeln zu können, muss bekannt sein, welche Schritte nacheinander durchlaufen werden müssen. Es sind Kenntnisse über die Arbeitsweise von deegree erforderlich. Dazu muss in der Datei *\WEB-INF\classes\log4j.properties* eingestellt werden, dass alle den WMS betreffenden Debug-Meldungen im Logfile im *logs*-Verzeichnis des Tomcat gespeichert werden sollen.

Listing 6-1: log4j.properties des deegree WMS

```
log4j.logger.org.deegree.ogcwebservices.wms=DEBUG
```

Anschließend kann dieser Datei entnommen werden, dass deegree zunächst einen DescribeFeatureType-Request an den WFS sendet, um danach per GetFeature-Request eine GML-Datei anzufordern. In dieser GML-Datei steht wiederum eine URL zu einem DescribeFeatureType-Request, der an die WFS-Fassade umgeleitet werden muss.

Auf diesem Weg konnte der Ablauf und das Zusammenspiel der Komponenten in achtzehn Schritten in *Tabelle 6-1* zusammengefasst werden.

Tabelle 6-1: FPS-Fassade - Ablauf

		FPS-Fassade	
	Client		
1.	WMS-GetCapabilities-Request	**WMS-Fassade**	
2.		WMS-GetCapabilities-Response	
3.	WMS-GetMap-Request mit SLD		
4.		RemoteWFS-URL speichern	
5.		SLD edit.: Remote-WFS=WFS-Fassade	
6.		WMS-GetMap-Request mit editierter SLD weiterleiten	
	deegree WMS		
7.	SLD-Datei lesen		
8.	WFS-Describe-FeatureType-Request	**WFS-Fassade**	
9.		WFS-Describe-FatureType-Request	**RemoteWFS**
10			WFS-DescribeFea-tureType-Response
11.		editiertes WFS-Des-cribeFeatureType-Do-kument zurückgeben	
12.	WFS-GetFeature-Request		
13.		WFS-GetFeature-Request	
14.			WFS-GetFeature-Response (GML)
15.		editiertes GML-Doku-ment zurückgeben	
16.	Karte generieren		
17.	generiertes Bild an Client zurückgeben		
	Client		
18.	Karte anzeigen		

Diese Tabelle zeigt auf, was genau in der FPS-Fassade realisiert werden muss. Die zur Verfügung zu stellenden Schnittstellen werden in *Tabelle 6-2* zusammengefasst.

Tabelle 6-2: FPS-Fassade - Interfaces

Web Map Service	GetCapabilities
	GetMap
Web Feature Service	DescribeFeatureType
	GetFeature

6.2.2 Umsetzung

Zur Programmierung des Workaround war eine Lösung notwendig, mit der die genannten KVP-Requests verarbeitet werden können. Obwohl auch die Skriptsprache PHP als mögliche Implementierungsalternative in Betracht gezogen werden könnte, fiel die Entscheidung auf eine Umsetzung mittels der Java-Servlet-Technologie. Diese Entscheidung fiel unter anderem aufgrund der Tatsache, dass auch deegree diese Technologie nutzt und deshalb mit dem Apache Tomcat bereits ein Servlet-Container zur Verfügung steht.

Für die Umsetzung war eine entsprechende Entwicklungsumgebung notwendig. In diesem Fall kam die Eclipse IDE for Java EE Developers [26] zum Einsatz. Zusätzlich wurde das entsprechende Java Development Kit (JDK) [37] installiert.

Der komplette Quelltext der FPS-Fassade ist in *Anlage B* abgebildet. Zusätzlich befindet sich der Eclipse-Workspace auf der beiliegenden CD *(Anlage D, Ordner \workspace\Workaround\)*.

WMS-Fassade

Die Servlet-Klasse *WMS* bedient die WMS-Schnittstellen GetCapabilities und GetMap.

Die FPS-Fassade stellt denjenigen Dienst dar, mit dem der Client kommuniziert. Da die meisten Clients als erstes die Capabilities abfragen, muss die WMS-Fassade diese zurückgeben können *(siehe Tabelle 6-1 - Schritte 1 und 2)*.

Listing 6-2: WMS-Fassade - GetCapabilities-Response

```
if (request.getParameter(requestKey).equals("GetCapabilities")) {
 response.setContentType("text/xml");
 PrintWriter out = response.getWriter();
 out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?> [...] "
        + request.getRequestURL().toString()
        + "?\" xlink:type=\"simple\"/> [...] xlink:href=\""
        + request.getRequestURL().toString()
        + "?\" xlink:type=\"simple\"/> [...] xlink:href=\""
        + request.getRequestURL().toString()
        + "?\" xlink:type=\"simple\"/> [...] xlink:href=\""
        + request.getRequestURL().toString()
        + "?\" xlink:type=\"simple\" [...]
        <Layer cascaded=\"0\" noSubsets=\"0\" opaque=\"0\"
        queryable=\"1\"><Name>FPS</Name><Title>FPS</Title>
        <Abstract>Feature Portrayal Service</Abstract><SRS>
        EPSG:4326</SRS><LatLonBoundingBox maxx=\"180.0\"
        maxy=\"90.0\" minx=\"-180.0\" miny=\"-90.0\"/></Layer>
        [...] </WMT_MS_Capabilities>");
 out.close();
}
```

In *Listing 6-2* ist zu sehen, dass die URL-Angabe zu dieser Fassade aus dem erhaltenen Request gefiltert wird. Der Grund hierfür ist der Gedanke, die FPS-Fassade auf einem beliebigen Webserver installieren zu können ohne diese URLs an die neue Umgebung anpassen zu müssen. Weiterhin gibt die Fassade vor, eine Ebene mit dem Namen *FPS* anzubieten, da viele Clients, welche in der Regel nicht vordergründig als FPS-Client gedacht sind, unbedingt einen *LAYERS*-Parameter generieren wollen. Dieser wird dann natürlich den Capabilities entnommen.

Bei einem GetMap-Request muss dieses Servlet dafür sorgen, dass die Anfrage an den deegree WMS, also den eigentlichen Portrayal Service, weitergeleitet wird. Allerdings muss deegree anschließend mit der

WFS-Fassade kommunizieren. Dazu wird aus der originären SLD-Datei die RemoteWFS-URL herausgefiltert, gespeichert und durch die URL zur WFS-Fassade ersetzt. Diese Schritte erfolgen in einer Hilfsklasse mit dem Namen *Assistant*, welche die Weitergabe wichtiger Informationen der WMS-Fassade an die WFS-Fassade regelt und einige notwendige Methoden zur Verfügung stellt. So zum Beispiel die Methode aus *Listing 6-3*, welche die oben genannten SLD-spezifischen Aktionen ausführt und die URL zum editierten SLD-Dokument zurückgibt.

Listing 6-3: WMS-Fassade - Methode „getSldUrl"

```
public URL getSldUrl(String requestURL) throws IOException {
 br = new BufferedReader(new InputStreamReader
     (sldUrl.openStream()));
 bw = new BufferedWriter(new FileWriter(sldFile));
 while ((s = br.readLine()) != null) {
     if (s.contains("xlink:href=")){
         remoteWFSUrl=s.substring(s.indexOf("\"",
             s.indexOf("xlink:href="))+1,s.indexOf("\"", s.indexOf("\"",
             s.indexOf("xlink:href="))+1));
         s=s.replace(remoteWFSUrl,
             requestURL.replace("wms", "wfs")+"?");
         if(!remoteWFSUrl.endsWith("?")){
             remoteWFSUrl=remoteWFSUrl+"&";
         }
     }
     if(s.contains("<FeatureTypeName>")){
         typeName=s.substring(s.indexOf(">",
             s.indexOf("<FeatureTypeName>"))+1,s.indexOf("<",
             s.indexOf("<FeatureTypeName>")+1));
     }
     bw.write(s);
     bw.newLine();
 }
 br.close();
 bw.close();
 sldUrl=new URL(requestURL.replace
     ("fps-fassade/wms", "fps-fassade/SLD.xml"));
 return sldUrl;
}
```

Zunächst wird die URL zum eigentlichen Web Feature Service in der Variable *remoteWFSUrl* gespeichert, bevor sie durch die eigene ersetzt wird. Der Name des Featuretypes wird später noch benötigt und

deshalb an dieser Stelle mit gesichert. Das neue SLD-File, welches an deegree weitergegeben wird, kann unter einer URL mit der entsprechenden IP-Adresse und dem notwendigen Zusatz aufgerufen werden. Nach der Installation auf der im *Abschnitt 5.2* genannten virtuellen Maschine lautet der Link also *http://141.56.141.5/fps-fassade/SLD.xml*.

Daraufhin kann die WMS-Fassade den weiterzuleitenden Request zusammenbauen. *Listing 6-4* zeigt, dass dieser aus den Request-Parametern generiert wird, die an die Fassade geschickt wurden. Dabei wird der Wert des SLD-Parameters angepasst und ein eventueller *LAYERS*-Parameter, der beispielsweise von einem Client angefügt wurde, wird ignoriert.

Listing 6-4: WMS-Fassade - deegree-Request generieren

```
for (Map.Entry<String,String> eintrag : parameters.entrySet()){
 if (eintrag.getKey().equals(sldKey)){
     deegreeWMSUrl=deegreeWMSUrl.concat("SLD="+
         assistant.getSldUrl(request.getRequestURL().toString())+"&");
 }
 else{
     if (!eintrag.getKey().equals("LAYERS")){
         deegreeWMSUrl=deegreeWMSUrl.concat(eintrag.getKey()+"="
             +request.getParameter(eintrag.getKey().toString())+"&");
     }
 }
}
```

Abschließend wird die Weiterleitung angestoßen und die WMS-Fassade wird verlassen.

Listing 6-5: WMS-Fassade - Weiterleitung an deegree

```
response.sendRedirect(response.encodeRedirectURL(
deegreeWMSUrl.substring(0,deegreeWMSUrl.length()-1)));
```

WFS-Fassade

Nach der Verarbeitung des GetMap-Request beim deegree WMS, schickt dieser die notwendigen Anfragen an die WFS-Fassade.

Listing 6-6 zeigt einen Ausschnitt aus dem Umgang mit dem DescribeFeatureType-Request *(siehe Tabelle 6-1 - Schritte 9 bis 11)*.

Listing 6-6: WFS-Fassade - Verarbeitung DescribeFeatureType-Request

```
if (request.getParameter(requestKey).equals
    ("DescribeFeatureType")) {
 dftRequestUrl=assistant.getRemoteWFSUrl();
 parameters=request.getParameterMap();
 for (Map.Entry<String,String> eintrag : parameters.entrySet()){
     dftRequestUrl=dftRequestUrl.concat(eintrag.getKey()+"="
         +request.getParameter(eintrag.getKey().toString())+"&");
 }
 dftRequestUrl=dftRequestUrl.replace
     ("VERSION=1.1.0", "VERSION=1.0.0");
 br = new BufferedReader(new InputStreamReader(new URL
     (dftRequestUrl.substring(0,
     dftRequestUrl.length()-1)).openStream()));
 while ((s = br.readLine()) != null) {
     if(s.contains("ExceptionReport")){
         dftRequestUrl=dftRequestUrl.replace
             ("VERSION=1.0.0", "VERSION=1.1.0");
         break;
     }
 }
 br.close();
 if (unrealDFT==false) {
     br = new BufferedReader(new InputStreamReader(new URL
         (dftRequestUrl.substring(0,
         dftRequestUrl.length()-1)).openStream()));
     while ((s = br.readLine()) != null) {
         s=s.replaceAll("type=\"string\"", "type=\"xsd:string\"");
         out.println(s);
     }
     br.close();
     out.close();
[...]
}
```

Die Anfrage wird zunächst an den eigentlichen RemoteWFS gesendet. Die Antwort wird in der WFS-Version 1.0.0 angefordert, da mit dieser, wie in *Tabelle 5-3* dargestellt, mehreren Problemen aus dem Weg

gegangen werden kann, vor allem in Bezug auf GeoServer-Implementierungen.

Zu Beginn wird der Request einmal probehalber abgesetzt, um eine eventuelle Inkompatibilität des RemoteWFS mit WFS 1.0.0 festzustellen, was sich in Form einer zurückerhaltenden Exception bemerkbar machen würde. Das wäre zum Beispiel bei einem deegree Web Feature Service der Fall und die Fassade würde doch wieder auf die WFS 1.1.0-Schnittstelle zurückgreifen müssen.

Um das ebenfalls bereits in *Tabelle 5-3* behandelte Problem mit MapServer zu umgehen, werden die Element-Typen in der SLD-Syntax um den Namensraum *xsd* erweitert, falls dieser noch nicht vorhanden ist.

Mit der Anweisung *out.println(s);* werden die vom echten RemoteWFS eingelesenen Zeilen nach eventueller Veränderung an den deegree WMS weitergegeben.

Eine ähnliche Vorgehensweise wird bei der Beantwortung eines GetFeature-Request eingeschlagen. Die GML-Daten werden nur insofern editiert, dass der enthaltene DescribeFeatureType-Request wieder auf diese Fassade zeigen muss, da deegree unter Umständen diesen Link erneut abfragt.

Listing 6-7: WFS-Fassade - Verarbeitung GetFeature-Request

```
br = new BufferedReader(new InputStreamReader
      (new URL(assistant.getGfRequestUrl()).openStream()));
while ((s = br.readLine()) != null) {
 if (s.contains(assistant.getRemoteWFSUrl())){
      try{
           s=s.replace(s.substring(s.indexOf(assistant.getRemoteWFSUrl
                ()),s.indexOf(" ", s.indexOf(assistant.getRemoteWFSUrl())
                +1)), assistant.getDftRequestUrl());
           if(s.toLowerCase().contains("subtype")){
                s=s.replace(s.substring(s.indexOf("subtype"),
                     s.indexOf(" ", s.indexOf("subtype")+2)), "");
           }
      }
      catch(Exception ex){
           s=s.replace(s.substring(s.indexOf(assistant
                .getRemoteWFSUrl()),s.indexOf("\"", s.indexOf(assistant
                .getRemoteWFSUrl())+1)), assistant.getDftRequestUrl());
      }
 }
 out.println(s);
}
br.close();
out.close();
```

Der zwingend zu entfernende *subtype*-Ausschnitt gehört zu einem *OUTPUTFORMAT*-Parameter, der verworfen werden muss *(siehe Tabelle 5-3)*. Im *catch*-Block aus *Listing 6-7* wird der Fall behandelt, dass die Angabe des Links mit einem Anführungszeichen statt eines Leerzeichens endet.

Zusätzlich muss noch die Variante berücksichtigt werden, dass der deegree WMS nicht immer zuerst einen DescribeFeatureType-Request absetzt. Daraus resultiert die Notwendigkeit, diesen vor der Verarbeitung eines GetFeature-Request zu erzwingen.

Listing 6-8: WFS-Fassade - Erzwingen von DescribeFeatureType

```
if (dftTest==false){
 unrealDFT=true;
 response.sendRedirect(response.encodeRedirectURL
      (request.getRequestURL().toString()+"?SERVICE=WFS
      &VERSION=1.1.0&REQUEST=DescribeFeatureType
      &TYPENAME="+assistant.getTypeName()));
}
```

Die Variablen *dftTest* und *unrealDFT* dienen der Kontrolle, ob dieser Request überhaupt erzwungen werden muss und ob es sich um eine solche herbeigeführte Abfrage handelt. Weitere Kontrollstrukturen zur optimalen Kooperation der Komponenten sind dem kompletten Quellcode in der *Anlage B* zu entnehmen.

6.2.3 Installation und Anwendung

Installation

Zur Installation der FPS-Fassade müssen zunächst die erforderlichen Voraussetzungen vorhanden sein. In jedem Fall muss zumindest die Java-Laufzeitumgebung der Version 1.5.x oder höher installiert sein. Zusätzlich wird als Servlet Container der Apache Tomcat 5.5.x oder höher erwartet, wobei natürlich auch andere Servlet Engines in Frage kommen würden, dessen Funktionsweise im Zusammenhang mit dieser Diplomarbeit jedoch nicht getestet wurde.

Sind diese Voraussetzungen erfüllt, kann der fertig konfigurierte deegree FPS installiert werden. Dazu kann das in der *Anlage D* zu findende Verzeichnis *deegree-wms*, wie in *Abbildung 6-2* geschehen, per Tomcat Manager installiert werden, wobei der Pfad zum Ordner entsprechend anzupassen ist.

Abbildung 6-2: Installation des FPS

Es ist auch möglich, einen anderen als den zu dieser Arbeit gehörenden deegree WMS zu nutzen, solange dieser die Mindest-anforderungen an einen Feature Portrayal Service erfüllt. Dieser muss übrigens auch nicht zwingend in denselben Webserver integriert werden, wie die FPS-Fassade.

Nun kann auch der Workaround eingerichtet werden. Die FPS-Fassade liegt als WAR-File (*Anlage D, Ordner \war\fps-fassade.war*) vor und kann wiederum mit Hilfe des Tomcat Manager oder per Hineinkopieren des Archivs in das *webapps*-Verzeichnis installiert werden.

Abbildung 6-3: Installation der FPS-Fassade

Anschließend muss in der Datei *fps-fassade.conf* im Ordner *\WEB-INF\conf* aus dem *fps-fassade*-Verzeichnis unter den *webapps* des Tomcat die URL zu dem deegree WMS angegeben werden, der als Darstellungsdienst eingesetzt werden soll. Danach kann der Tomcat neu gestartet werden.

Die genannten Schritte wurden auch auf der virtuellen Maschine im Labor Geoinformatik *(siehe Abschnitt 5.2)* durchgeführt.

Anwendung

Um die FPS-Fassade zu nutzen, müssen die GetMap-Requests mit *SLD*-Parameter direkt an sie gesendet werden. Bei der aufgesetzten Lösung innerhalb der Infrastruktur des Labors lautet die zugehörige URL

http://141.56.141.5/fps-fassade/wms?, gefolgt von den Abfrage-Parametern. Das SLD-Dokument selbst muss dabei auf einem öffentlich zugänglichen Webserver liegen und der Version 1.0.0 entsprechen.

Anschließend wird der Request verarbeitet und der Nutzer erhält bei einer fehlerfreien Anfrage eine Karte zurück. Die intern ablaufenden Prozesse sind der *Tabelle 6-1* zu entnehmen.

An der im Browser angezeigten URL ist zu erkennen, dass der Request an den deegree WMS weitergeleitet wurde und ein eigenes SLD-Dokument übergeben wurde, welches unter dem im Request angegebenen Link erreichbar ist.

Weiterhin werden während eines Durchlaufs mehrere LOG-Dateien im Verzeichnis *\WEB-INF\logs* angelegt. Diese enthalten Informationen zu den einzelnen Requests und dienen vor allem zu Debugging-Zwecken.

6.2.4 Beispiel-Requests

Dieser Abschnitt soll anhand von Beispielen die Funktionsweise der FPS-Fassade aufzeigen und beweisen. Es werden jeweils die Requests ohne und mit FPS-Fassade dargestellt.

Die in diesem Abschnitt benutzten SLD-Dateien befinden sich im Ordner *\sld* auf der digitalen *Anlage D*.

Tabelle 6-3: FPS-Fassade mit deegree WFS (1.0.0/1.1.0)

SLD-File		siehe Anlage: deegreeWFS_SLD-100.xml
ohne FPS-Fassade	***Request***	http://141.56.141.5/deegree-wms/fps?REQUEST=GetMap&version=1.1.1&service=WMS&FORMAT=image/png&BBOX=228563,4094785,673992,4653592&SRS=EPSG:26912&WIDTH=500&HEIGHT=627&SLD=http://www.htw-dresden.de/~s58783/deegreeWFS_SLD-100.xml
	Ergebnis	
mit FPS-Fassade	***Request***	http://141.56.141.5/fps-fassade/wms?REQUEST=GetMap&version=1.1.1&service=WMS&FORMAT=image/png&BBOX=228563,4094785,673992,4653592&SRS=EPSG:26912&WIDTH=500&HEIGHT=627&SLD=http://www.htw-dresden.de/~s58783/deegreeWFS_SLD-100.xml
	Ergebnis	

Tabelle 6-4: FPS-Fassade mit MapServer WFS (1.0.0)

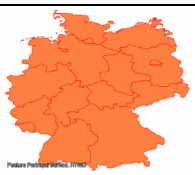
SLD-File		siehe Anlage: MapServerWFS100_SLD-100.xml
ohne FPS-Fassade	***Request***	http://141.56.141.5/deegree-wms/fps?REQUEST=GetMap&version=1.1.1&service=WMS&FORMAT=image/png&BBOX=5.8,47.2,15.1,55.1&SRS=EPSG:4326&WIDTH=500&HEIGHT=425&SLD=http://www.htw-dresden.de/~s58783/MapServerWFS100_SLD-100.xml
	Ergebnis	<?xml version="1.0" encoding="UTF-8"?> <ServiceExceptionReport> <ServiceException locator="unknown"> **Queried feature type "ms:Bundeslaender (ms=http://mapserver.gis.umn.edu/mapserver)" is not served by this WFS.** </ServiceException> </ServiceExceptionReport>
mit FPS-Fassade	***Request***	http://141.56.141.5/fps-fassade/wms?REQUEST=GetMap&version=1.1.1&service=WMS&FORMAT=image/png&BBOX=5.8,47.2,15.1,55.1&SRS=EPSG:4326&WIDTH=500&HEIGHT=425&SLD=http://www.htw-dresden.de/~s58783/MapServerWFS100_SLD-100.xml
	Ergebnis	

Tabelle 6-5: FPS-Fassade mit MapServer WFS (1.0.0/1.1.0)

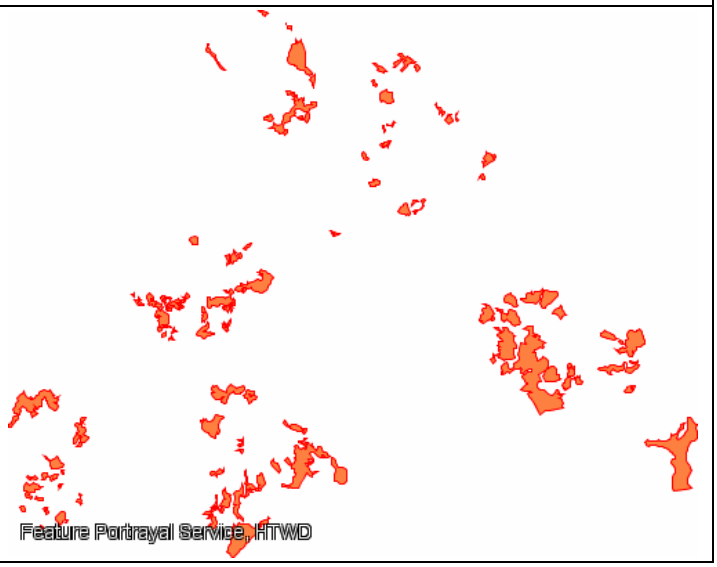
SLD-File		siehe Anlage: MapServerWFS110_SLD-100.xml
ohne FPS-Fassade	***Request***	http://141.56.141.5/deegree-wms/fps?REQUEST=GetMap&version=1.1.1&service=WMS&FORMAT=image/png&BBOX=4507747,5586671,4563610,5630925&SRS=EPSG:31468&WIDTH=500&HEIGHT=396&SLD=http://www.htw-dresden.de/~s58783/MapServerWFS110_SLD-100.xml
	Ergebnis	<?xml version="1.0" encoding="UTF-8"?> <ServiceExceptionReport> <ServiceException locator= "ServiceInvokerForUL%3A+null"> **Couldn't perform query!** </ServiceException> </ServiceExceptionReport>
mit FPS-Fassade	***Request***	http://141.56.141.5/fps-fassade/wms?REQUEST=GetMap&version=1.1.1&service=WMS&FORMAT=image/png&BBOX=4507747,5586671,4563610,5630925&SRS=EPSG:31468&WIDTH=500&HEIGHT=396&SLD=http://www.htw-dresden.de/~s58783/MapServerWFS110_SLD-100.xml
	Ergebnis	

Tabelle 6-6: FPS-Fassade mit GeoServer WFS (1.0.0/1.1.0)

SLD-File		siehe Anlage: GeoServerWFS_SLD-100.xml
ohne FPS-Fassade	***Request***	http://141.56.141.5/deegree-wms/fps? VERSION=1.1.1&SERVICE=WMS &REQUEST=GetMap&SRS=EPSG:31469 &BBOX=5474086,5637004,5479155,5643637 &WIDTH=500&HEIGHT=708&FORMAT=image/png &SLD=http://www.htw- dresden.de/~s58783/GeoServerWFS_SLD-100.xml
	Ergebnis	<?xml version="1.0" encoding="UTF-8"?> <ServiceExceptionReport> <ServiceException locator="org.deegree.ogcwebservices .wms.DefaultGetMapHandler"> **java.util.concurrent.CancellationException** </ServiceException> </ServiceExceptionReport>
mit FPS-Fassade	***Request***	http://141.56.141.5/fps-fassade/wms? VERSION=1.1.1&SERVICE=WMS &REQUEST=GetMap&SRS=EPSG:31469 &BBOX=5474086,5637004,5479155,5643637 &WIDTH=500&HEIGHT=708&FORMAT=image/png &SLD=http://www.htw- dresden.de/~s58783/GeoServerWFS_SLD-100.xml
	Ergebnis	

7 ENTWICKLUNG EINES FPS-CLIENTS

Um den Feature Portrayal Service, beziehungsweise die FPS-Fassade, optimal nutzen zu können, ist ein entsprechender Client sinnvoll. Im Folgenden werden unterschiedliche Lösungsvarianten diskutiert.

7.1 Mögliche Client-Lösungen

In diesem Abschnitt werden drei Möglichkeiten vorgestellt, mit deren Hilfe ein FPS-Client entwickelt werden kann. Es wurden allerdings auch andere Lösungen getestet, welche jedoch nicht die notwendigen Funktionalitäten mitbringen, die für diesen Zweck erforderlich sind.

Das iGeoPortal [11] von deegree bietet derzeit kein Modul zur Integration benutzerdefinierter Stile an. Mit dem Desktop-Client iGeoDesktop [9] können zwar eigene Darstellungen definiert und SLD-Dateien eingelesen werden, allerdings sind dabei nur die UserStyles von Interesse. Etwaige definierte UserLayer werden ignoriert, da die Stile direkt einem Dienst zugeordnet werden.

Weiterhin wurden unter anderem uDig (User-friendly Desktop Internet GIS) [38], OpenJUMP [33] und Quantum GIS [36] geprüft. Alle kommen derzeit nicht als FPS-Client in Frage.

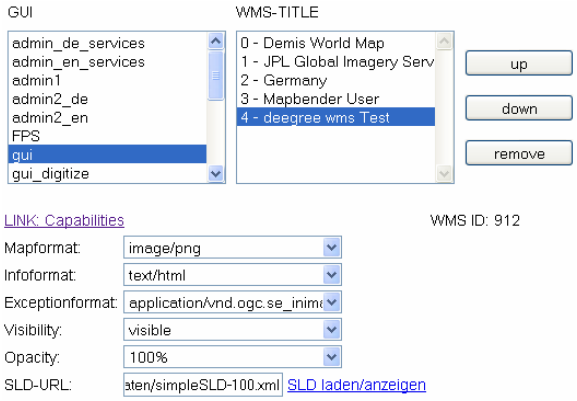
7.1.1 Mapbender

Die Software Mapbender [30] wurde mittels PHP und JavaScript entwickelt und dient der Visualisierung von OGC-Diensten. Zur Installation der Version 2.6 [1] ist ein Webserver mit PHP-Unterstützung notwendig, wie beispielsweise XAMPP [22]. Durch diese

PHP-Skripte wird unter anderem auf eine Administrationsdatenbank zugegriffen. Bei dieser handelt es sich um PostgreSQL [35] mit PostGIS als räumlichen Aufsatz.

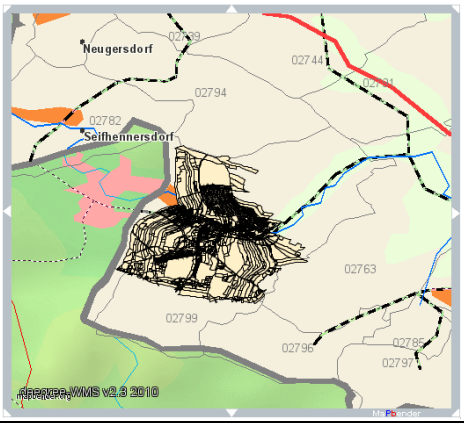
Als Nutzer mit Administrator-Rechten kann hier einem Web Map Service eine SLD-Datei zugeordnet werden. Dieser WMS gehört dann zu einer grafischen Benutzeroberfläche, welche auch für einfache User zugänglich ist.

Abbildung 7-1: Mapbender - WMS-Einstellungen



Styled Layer Descriptor wird an dieser Stelle korrekt umgesetzt, auch mit einem *UserLayer*-Element. Begibt man sich in die entsprechende Oberfläche, präsentiert der Feature Portrayal Service die gewünschten Daten.

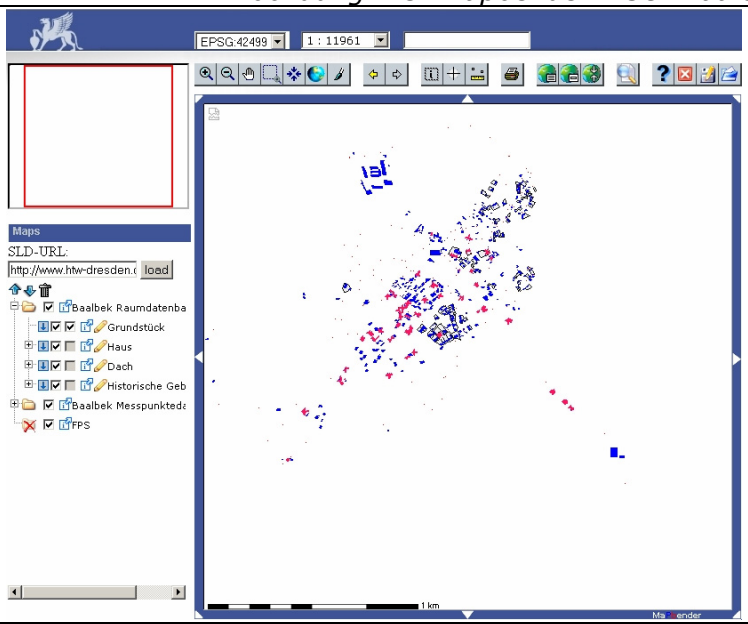
Abbildung 7-2: Mapbender - GUI mit FPS



Das Problem bei diesem Client ist, dass jeder Nutzer in der Lage sein muss, diverse, per SLD vereinbarte Darstellungen, eigenständig anzuwenden. Bei Mapbender sind hierfür aber unbedingt Administrator-Rechte notwendig. Aus diesem Grund kann Mapbender in seiner aktuellen Umsetzung nicht als ernsthafter FPS-Client in Betracht gezogen werden.

Die einzige Möglichkeit besteht in der manuellen Anpassung von Mapbender. Beispielsweise wurden im Zuge der Diplomarbeit von Andreas Richter [15] Buttons eingebaut, mit denen die Darstellung für bestimmte Ebenen durch einen SLD-Editor verändert werden können.

Abbildung 7-3: Mapbender - GUI Baalbek [32]



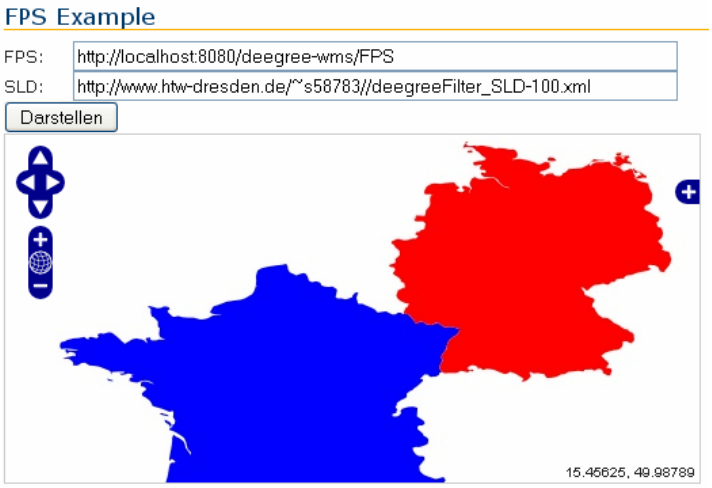
Allerdings müsste man diese Möglichkeit für alle vorhandenen Ebenen verfügbar machen. Die Zuweisung einer SLD-URL zu einem beliebigen Layer wurde in diesen Client nicht eingebaut. Somit zeigt das Ergebnis dieser Diplomarbeit, die natürlich andere Ziele als die Bereitstellung eines FPS-Clients verfolgte, eine Richtung auf, in die man weiter-entwickeln müsste, um Mapbender auch für die Nutzung eines Feature Portrayal Service nutzen zu können.

7.1.2 OpenLayers

OpenLayers [34] ist eine JavaScript-Bibliothek, welche alle Werkzeuge mitbringt, um eine dynamische Karte in die eigene Webseite einzubinden. Es werden Methoden angeboten, um sowohl proprietäre Kartendienste wie Google Maps als auch Web Map und Web Feature Services zu verarbeiten.

Um OpenLayers auszuprobieren, wurde eine einfache HTML-Seite erstellt. Der Nutzer kann einen Dienst und eine SLD-Datei angeben. Anschließend wird der Layer der Karte hinzugefügt.

Abbildung 7-4: OpenLayers - Hinzufügen eines WMS mit SLD-Parameter



Der Vorteil dieser Lösung ist die Tatsache, dass der Client selbst zusammengebaut wird. Das ist zwar mit einem verhältnismäßig großen Aufwand verbunden, jedoch hat man die volle Kontrolle über das Produkt.

Da es möglich ist, mit einem solchen Client die an die Dienste zu sendenden Request-Parameter im Quelltext selbst zu definieren, bietet sich OpenLayers sehr gut als FPS-Client an. Aus diesem Grund fiel die Entscheidung zur Auswahl eines Clients auf diese Variante.

7.1.3 Gaia

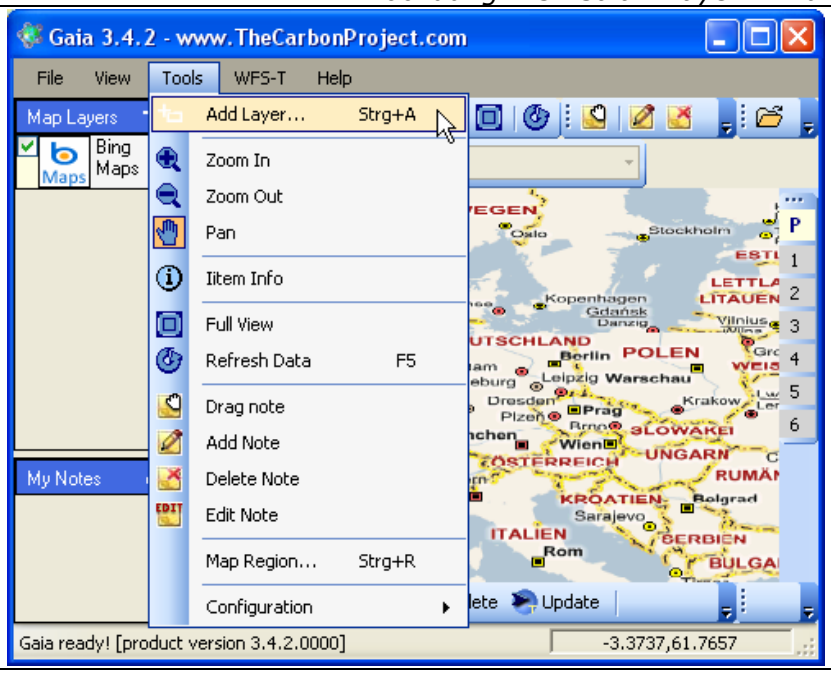
Da als mögliche Clients natürlich auch Desktop-Lösungen in Frage kommen, soll an dieser Stelle die Software Gaia [27] vorgestellt werden. Es handelt sich um das einzige getestete Desktop-GIS, mit dem die Einbindung eines Feature Portrayal Service durchführbar ist.

The Carbon Project heißt das Unternehmen, welches diesen Client entwickelt. Neben diversen OGC- und proprietären Kartendiensten wird eine Vielzahl an Dateiformaten unterstützt. Die in dieser Diplomarbeit zum Einsatz gekommene Version lautet Gaia 3.4.2 [7].

Die Installation von Gaia ist denkbar einfach. Nach dem Download des MSI-Paketes kann der Client installiert werden, indem der Installer ausgeführt wird.

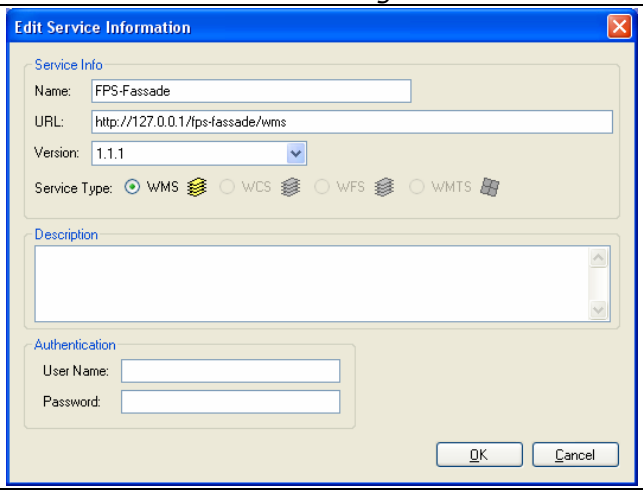
Um beispielsweise die FPS-Fassade nutzen zu können, muss ein Layer hinzugefügt werden.

Abbildung 7-5: Gaia - Layer hinzufügen



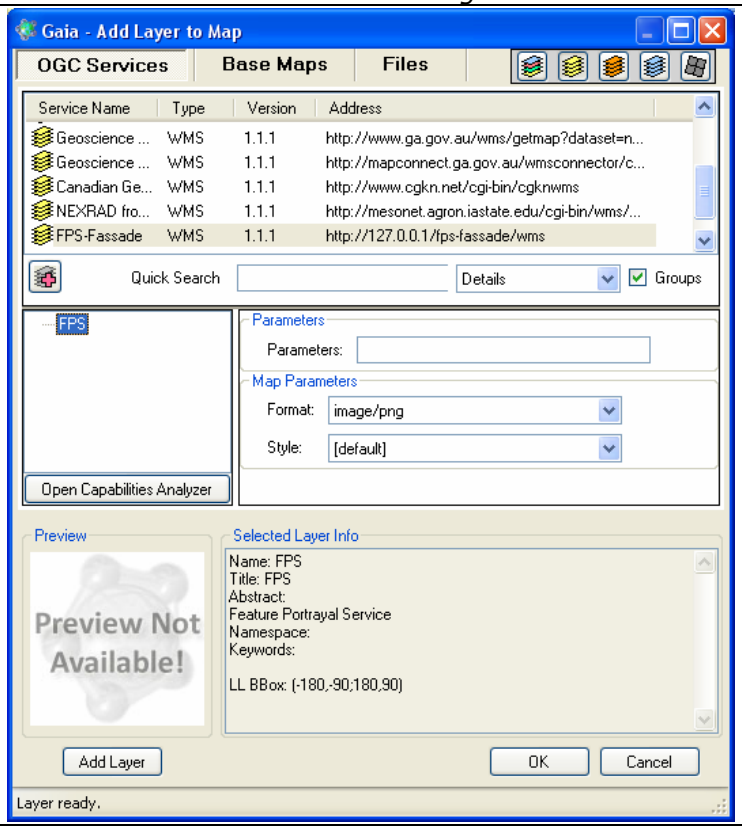
Im erscheinenden Fenster muss die Fassade über den entsprechenden Button als neuer Dienst hinzugefügt werden.

Abbildung 7-6: Gaia - Dienst hinzufügen



Nun wird der Dienst mit in der Liste aufgeführt und kann genutzt werden.

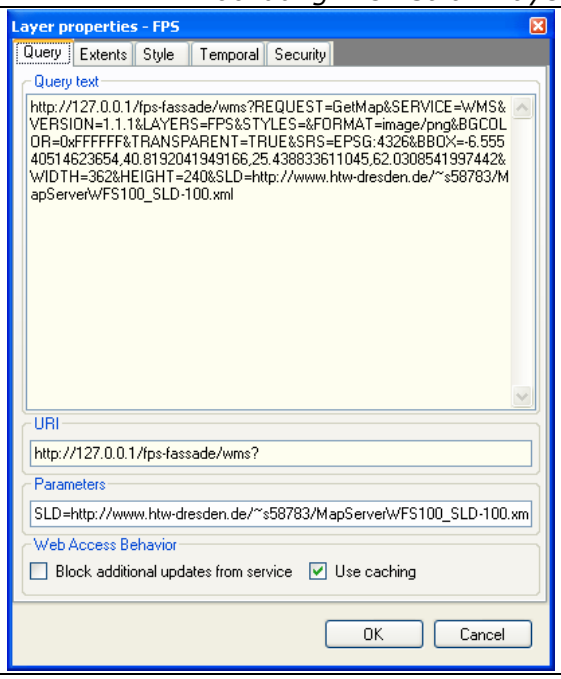
Abbildung 7-7: Gaia - Dienst-Auswahl



Da Gaia natürlich nicht direkt als FPS-Client entwickelt wurde, kann eine Ebene nur dann über den Button *Add Layer* hinzugefügt werden, wenn der WMS auch über einen Layer verfügt. Aus diesem Grund wurde ein leerer Layer in die Capabilities der WMS-Fassade eingepflegt *(siehe Listing 6-2)*. Weil für die Ebene keine Darstellung existiert, will Gaia einen Standard-Stil benutzen. Die Angabe des SLD-Parameters könnte bereits an dieser Stelle erfolgen.

Nachdem die Ebene der Karte hinzugefügt wurde, können per Rechtsklick deren Einstellungen bearbeitet werden.

Abbildung 7-8: Gaia - Layer-Einstellungen



Hier können beliebige Request-Parameter hinzugefügt werden, in diesem Fall also der SLD-Parameter. Der komplette Request, der so auch in einem Browser abgesetzt werden könnte, wird unter *Query text* angegeben.

Als Ergebnis ist die durch den Feature Portrayal Service visualisierte Ebene über dem Basis-Layer zu sehen.

Abbildung 7-9: Gaia - Karte mit FPS-Layer

Gaia kann also aufgrund der Möglichkeit, eigene Parameter-Definitionen anzugeben, gut als FPS-Client genutzt werden.

7.2 Webseite mit OpenLayers-Client

Bei der Erstellung einer Webseite für das Geoportal des Labors kam OpenLayers 2.8 zum Einsatz. Das ZIP-Archiv zu dieser Version wurde heruntergeladen und entpackt. Für die Verwendung der JavaScript-Bibliothek sind die Verzeichnisse *\img* und *\theme* sowie die Datei *\build\OpenLayers.js* notwendig. Letztere wird dann in den Header der HTML-Seite eingebettet.

Listing 7-1: Client - Header

```
<head>
 <title>Feature Portrayal Service</title>
 <link rel="shortcut icon" href="images/favicon.ico"/>
 <meta name="author" content="Martin Domeyer"/>
 <meta name="language" content="de"/>
 <link rel="stylesheet" type="text/css"
     href="styles/style.css"/>
 <script src='http://maps.google.com/maps?file=api&amp;v=2
     &amp;key=ABQIAAAAwmeZhL3tCC-bwNeKqGLxEhTnRhy2Ic
     OaBxKEqiT7zBpZuljP1xR2VkfCqaczGADGRtUbyo2DznmaDw'
     ></script>
 <script src="OpenLayers.js"></script>
 <script type="text/javascript">
     [...]
 </script>
</head>
```

Das letzte hier zu sehende *script*-Element enthält den kompletten JavaScript-Code, der für den Client erforderlich ist.

Im Body der Seite muss zunächst Platz für die Karte geschaffen werden. Dazu wird ein *div*-Block definiert, in welchen später die Karte per JavaScript eingefügt wird. Dazu ist eine *ID* notwendig. Durch die Angabe der Klasse in *Listing 7-2* wird aus der verlinkten CSS-Datei gelesen, wie groß der Client dargestellt werden soll.

Listing 7-2: Client - Block für die OpenLayers-Karte

```
<div id="map" class="mapArea">
 <noscript>
    <h2 align="center">
        <br/>
        JavaScript ist in Ihrem Browser deaktiviert!
        <br/>
        Um den FPS Client nutzen zu können,
        muss JavaScript aktiviert sein!
        <br/>
    </h2>
 </noscript>
</div>
```

Das *noscript*-Element erzeugt eine entsprechende Fehlermeldung, falls JavaScript im Browser deaktiviert ist.

Damit nun eine Karte erzeugt wird, wird beim Laden des Body die Funktion *init()* ausgeführt.

Listing 7-3: Client - Funktion init()

```
function init() {
 [...]
 map = new OpenLayers.Map('map', options);
 baseLayer();
 map.setCenter(new OpenLayers.LonLat(1528830, 6627340), 2);
}
```

Im Konstruktor des *Map*-Objekts muss die in *Listing 7-2* definierte *ID* angegeben werden. In den *options* stecken wichtige Angaben zu den Eigenschaften der Karte, wie beispielsweise die Projektion oder die Steuerelemente.

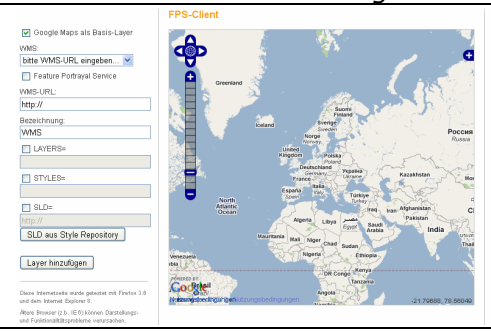
Die Funktion *baseLayer()* sorgt dafür, dass im Hintergrund Karten aus Google Maps angezeigt werden. Dazu ist auch die Angabe des Google Maps-API-Schlüssels im Header *(siehe Listing 7-1)* notwendig.

Listing 7-4: Client - Funktion baseLayer()

```
function baseLayer() {
 [...]
    gmapskarte = new OpenLayers.Layer.Google(
        "Google Maps: Karte",
        {'sphericalMercator': true}
    );
    gmapssatellit = new OpenLayers.Layer.Google(
        "Google Maps: Satellit",
        {type: G_SATELLITE_MAP, 'sphericalMercator': true}
    );
    gmapshybrid = new OpenLayers.Layer.Google(
        "Google Maps: Hybrid",
        {type: G_HYBRID_MAP, 'sphericalMercator': true}
    );
    map.addLayers([gmapskarte,gmapssatellit,gmapshybrid]);
 [...]
}
```

Der Basis-Layer dient zu Übersichtszwecken und zur einfacheren Orientierung in der Anwendung. Die *sphericalMercator*-Angabe bei der Erzeugung der Layer ist notwendig, da diese mit WMS-Ebenen überlagert werden sollen. Dadurch passen die Layer später geografisch übereinander. Die vereinbarten Koordinaten in der Methode *setcenter() (Listing 7-3)* für die Karte resultieren aus dieser Angabe. Der Client enthält demzufolge nach dem Laden der Seite nur den Basis-Layer, wobei der Nutzer zwischen verschiedenen Ansichten aus Google Maps wählen kann.

Abbildung 7-10: Client - Laden der Seite



Die linke Spalte beinhaltet ein Formular, durch das dynamisch Ebenen hinzugefügt werden können. Der mit deegree implementierte FPS und die FPS-Fassade können direkt als WMS ausgewählt werden. Der Haken bei *Feature Portrayal Service* wird in diesem Fall automatisch gesetzt und nur das Textfeld zur Eingabe einer SLD-Datei wird aktiviert. Außerdem besteht auch die Möglichkeit, den Client für klassische Web Map Services zu nutzen, also auch die Parameter *LAYERS* und *STYLES* zu berücksichtigen. Bei einem Klick auf *Layer hinzufügen* wird die Funktion *addWms()* ausgeführt.

Listing 7-5: Client - Funktion addWms()

```
function addWms() {
 if (document.form.layersparam.checked
        && document.form.stylesparam.checked
        && document.form.sldparam.checked) {
    layer = new OpenLayers.Layer.WMS(
        document.getElementById("wmsname").value,
        document.getElementById("wmsurl").value,
        {layers: document.getElementById("paramlayers").value,
            styles: document.getElementById("paramstyles").value,
            sld: document.getElementById("paramsld").value,
            transparent: 'TRUE'},
        {singleTile: true}
    );
 }
 [...]
 if (!document.form.layersparam.checked
        && !document.form.stylesparam.checked
        && document.form.sldparam.checked) {
    layer = new OpenLayers.Layer.WMS(
        document.getElementById("wmsname").value,
        document.getElementById("wmsurl").value,
        {sld: document.getElementById("paramsld").value,
            transparent: 'TRUE'},
        {singleTile: true}
    );
 }
 [...]
 layer.isBaseLayer = false;
 map.addLayer(layer);
 reloadLayers();
}
```

Beim Hinzufügen einer Ebene wird getestet, welche Kontrollkästchen zu den Request-Parametern aktiviert sind, um die entsprechenden OpenLayers-Anweisungen auszuführen. Die Bilder werden als *singleTiles* angefordert, da einige Dienste, wie auch der zu dieser Diplomarbeit gehörende Feature Portrayal Service, Copyright-Texte im Response-Bild ausgeben. Dieser Text würde bei der Aufteilung in mehrere Kartenbereiche mehrmals im Client auftauchen. Die erzeugten WMS-Ebenen werden transparent über dem Basis-Layer angezeigt.

Nach dem Anfügen einer neuen Ebene erzeugt die Funktion *reloadLayers()* eine neue Layer-Übersicht. Dazu werden die angebotenen OpenLayers-Methoden genutzt, um die Anzahl und Namen der aktuell vorhandenen Layer herauszufinden.

Listing 7-6: Client - Funktion reloadLayers()

```
function reloadLayers() {
 [...]
 for (var i = 3; i < map.getNumLayers(); i++) {
     [...]
     p.appendChild(document.createTextNode(map.layers[i].name));
     ausgabe.appendChild(p);
 }
 p = document.createElement("p");
 element = document.createElement("input");
 attribut = document.createAttribute("type");
 attribut.nodeValue = "button";
 element.setAttributeNode(attribut);
 [...]
 attribut = document.createAttribute("onclick");
 attribut.nodeValue = "removeWMS()";
 element.setAttributeNode(attribut);
 p.appendChild(element);
 ausgabe.appendChild(p);
}
```

Außerdem werden Kontrollkästchen für jeden Layer und eine Schaltfläche erzeugt, um Ebenen auch wieder entfernen zu können. In der *Abbildung 7-11* ist der linke Teil der Webseite nach einigen User-Eingaben zu sehen.

Abbildung 7-11: Client - hinzugefügte Ebenen

Das Löschen von Ebenen erfolgt mit der Funktion *removeWMS()*. Nachdem alle ausgewählten Layer entfernt wurden, wird auch die Layer-Übersicht neu geladen.

Listing 7-7: Client - Funktion removeWMS()

```
function removeWMS() {
 var l = map.layers.length, j = 0;
 for (var i = 3; i < l; i++) {
     try {
         if (document.getElementById('layerslist')
                 .checklayers[i-3].checked) {
             map.removeLayer(map.layers[i+j]);
             j=j-1;
         }
     [...]
 }
 reloadLayers();
}
```

7.3 Anbindung des Style Repository

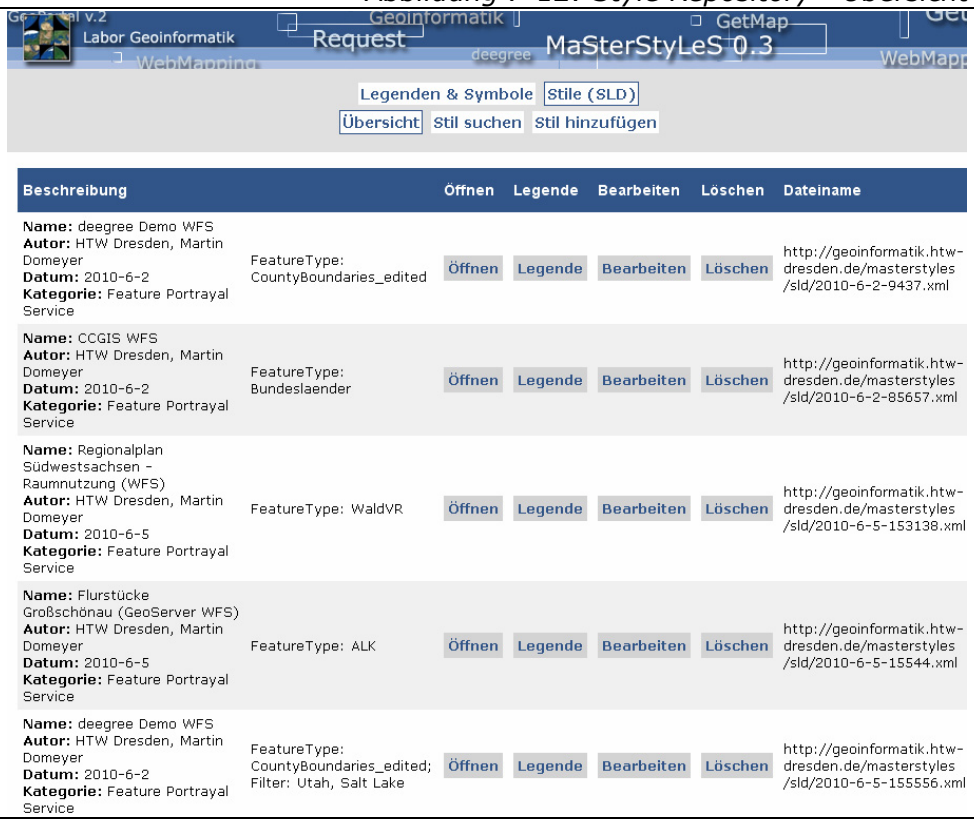
7.3.1 MaSterStyLeS

Auf der Homepage des Labors Geoinformatik der Hochschule kann auf den Legendenservice MaSterStyLeS (Management System für Styles, Legenden und Symbole) zugegriffen werden. Entwickelt wurde dieser Legendenservice im Zuge der Diplomarbeit von Stefan Schulze [16], welche auch eine umfassende Bedienungsanleitung beinhaltet.

Eine Anwendungsmöglichkeit von MaSterStyLeS ist der Einsatz als Style Repository. Der Nutzer kann eigene SLD-Files hochladen. Jeder Stil erhält eine URL, die anschließend den jeweiligen Diensten mitgeteilt werden kann. Der Vorteil des Ablegens der Dokumente im Style Repository liegt darin, dass diese dann einer breiten Masse an Benutzern zugänglich gemacht werden. Außerdem verfügt nicht jeder Anwender über einen öffentlich zugänglichen Webserver, um seine eigenen Darstellungsdefinitionen zu übermitteln.

Für den Upload eigener SLD-Dateien muss man sich bei dem Legendenservice mit Passwort anmelden. Anschließend kann eine Datei über *Stile (SLD) → Stil hinzufügen* hochgeladen werden. Auf diese Art und Weise wurden während dieser Arbeit mehrere Beispiel-Files in das Style Repository geladen.

Abbildung 7-12: Style Repository - Übersicht



Es ist darauf hinzuweisen, dass am Legendenservice seit der Entwicklung einige Änderungen vollzogen wurden und infolge dessen derzeit einige Probleme auftreten, die laut Aussagen der Labor-Mitarbeiter demnächst behoben werden sollen. Beispielsweise sind die in *Abbildung 7-12* zu sehenden Schaltflächen zur Bearbeitung der Stile auch dann vorhanden, wenn sich der Nutzer nicht eingeloggt hat. Außerdem können derzeit zwar SLD-Dateien hochgeladen werden, diese werden jedoch nicht in der Konfigurationsdatei für die Styles gespeichert. Daraus folgt, dass sie nach der Aktualisierung der Seite nicht mehr in der Liste stehen. Deshalb wurde die XML-Datei zur Konfiguration immer nur manuell angepasst. Weiterhin entleert sich diese Datei manchmal aus bisher unbekannten Gründen und muss aus einem Backup wiederhergestellt werden. Ebenso kann es passieren,

dass die SLD-Dokumente selbst wiederhergestellt werden müssen. Die Verwendung des Style Repository ist aus den genannten Gründen erst nach dessen Überarbeitung wirklich empfehlenswert.

7.3.2 Integration in die Webseite

In der *Abbildung 7-10* ist die rechte Spalte der Webseite ohne Inhalt zu sehen. Hier sollen die Stile aus dem Style Repository angezeigt und zur Auswahl angeboten werden.

Dazu bietet MaSterStyLeS den in *Listing 7-8* gezeigten Request an. Als Ergebnis wird eine HTML-Seite generiert, welche die vorhandenen Stile wie in *Abbildung 7-12* auflistet.

Listing 7-8: Style Repository - view-Request

```
http://geoinformatik.htw-dresden.de/masterstyles/sms?
service=sld&request=view
```

Um diese Abfrage auszuführen und die zurückgegebene HTML-Seite zu verarbeiten, reichen die einfachen JavaScript-Mittel nicht aus. Deshalb wurde eine JavaServer Page (JSP) erstellt. Diese Entscheidung fiel aufgrund der Tatsache, dass wegen der FPS-Fassade mit dem Apache Tomcat Server bereits ein Java EE-konformer Server zur Verfügung steht. Sowohl Servlets als auch JavaServer Pages sind Bestandteil des Java EE-Standards.

Zu Beginn beinhaltete die JSP lediglich den Quellcode der in *Abschnitt 7.2* entwickelten HTML-Webseite. Damit die JSP ordnungsgemäß kompiliert werden kann, sind einige Angaben notwendig, die in der page-Direktive gemacht werden können.

Listing 7-9: Webseite - page-Direktive

```
<%@ page language="java" contentType="text/html;charset=ISO-
8859-1" pageEncoding="ISO-8859-1" import="java.io.BufferedReader"
import="java.io.InputStreamReader" import="java.net.URL"%>
```

Hier werden unter anderem diejenigen Klassen importiert, die für den Java-Code innerhalb der JSP notwendig sind.

Somit kann die JavaScript-Funktion zum Laden des Style Repository aus dem Response-Dokument des view-Request generiert werden.

Listing 7-10: Client - Funktion loadMasterstyles()

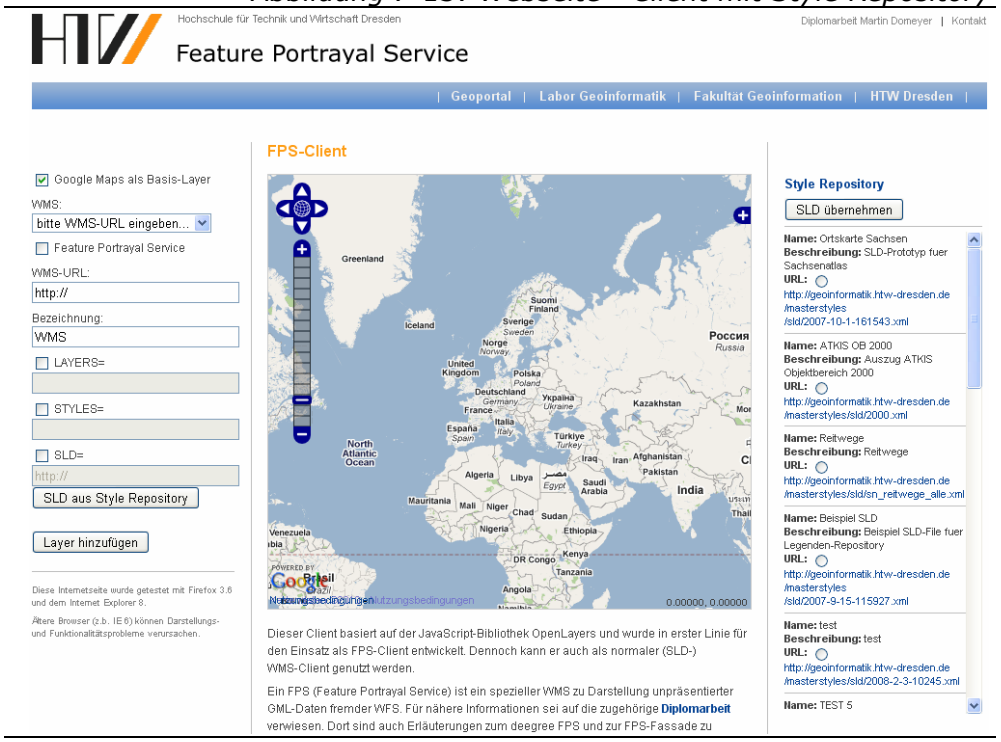
```
function loadMasterstyles() {
 [...]
 <%
    BufferedReader br = new BufferedReader(
        new InputStreamReader(new URL(
        "http://geoinformatik.htw-dresden.de/masterstyles/sms?
        service=sld&amp;request=view").openStream()));
    [...]
    while ((s.indexOf("<strong>Name:</strong> ",j)) != -1) {
        [...]
        out.println("element.appendChild(document.createTextNode(
            \"Name: \"))");
        out.println("div.appendChild(element)");
        i=s.indexOf("<strong>Name:</strong> ",j);
        out.println("div.appendChild(document.createTextNode(
            \""+s.substring(i+23,s.indexOf("<",i+23))+"\"));");
        [...]
        out.println("element.appendChild(document.createTextNode(
            \"URL: \"))");
        out.println("div.appendChild(element)");
        j=s.indexOf("<td>http",i);
        out.println("element = document.createElement(\"input\")");
        out.println("attribut = document.createAttribute(\"type\")");
        out.println("attribut.nodeValue = \"radio\"");
        out.println("element.setAttributeNode(attribut);");
        [...]
        out.println("attribut = document.createAttribute(\"value\")");
        out.println("attribut.nodeValue =
            \""+s.substring(j+4,s.indexOf("<",j+4))+"\"");
        out.println("element.setAttributeNode(attribut);");
        out.println("div.appendChild(element);");
        [...]
    }
 %>
}
```

Die HTML-Seite, die als Antwort auf den Request aus *Listing 7-8* zurückgegeben wird, muss entsprechend gelesen werden. Dazu wird eine *InputStreamReader*-Instanz erzeugt. Aus diesem Stream werden

dann die Zeichenketten herausgefiltert, die in die JavaScript-Befehle eingebettet werden müssen. Zu jedem Stil wird ein Radio-Button erzeugt, dem die URL zum SLD-Dokument als Wert angefügt wird. Aufgrund der in *Abschnitt 7.3.1* erläuterten Probleme konnte nicht getestet werden, ob die Stile auch ohne Anmeldung korrekt aus dem Legendenservice geladen werden. In diesem Fall würde die auszuwertende HTML-Seite etwas anders aussehen, müsste aber mit den erzeugten Java-Anweisungen auch dann das korrekte Ergebnis liefern.

Wird nun die JSP aufgerufen, sind auf der rechten Seite die Informationen aus dem Style Repository aufgelistet.

Abbildung 7-13: Webseite - Client mit Style Repository



Nach der Auswahl einer SLD-Datei und dem Betätigen der Schaltfläche *SLD übernehmen* wird die ausgewählte URL in das Textfeld für den SLD-Parameter auf der linken Seite neben dem Client geschrieben.

7.4 Installation und Anwendung

Da in diesem Kapitel lediglich Ausschnitte aus dem Quellcode gezeigt wurden, kann der komplette Inhalt der JavaServer Page in *Anlage C* betrachtet werden. Der Eclipse-Workspace ist Teil der beiliegenden digitalen *Anlage D (Ordner \workspace\Webseite)*.

Installation

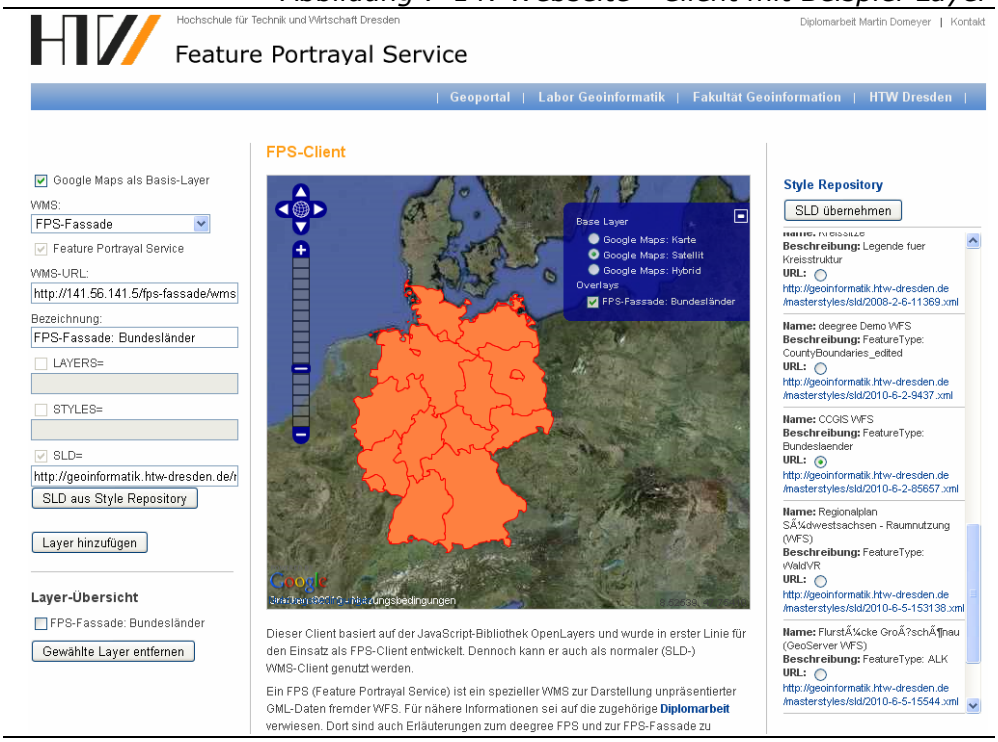
Zur Installation muss die Seite aufgrund der JSP-Technologie wieder in den Apache Tomcat integriert werden. Das WAR-Archiv dazu befindet sich in der digitalen *Anlage D (Ordner \war\fps.war)* und beinhaltet alle notwendigen Verzeichnisse und Dateien, auch die zur Einbindung von OpenLayers. Anschließend kann die Webseite aufgerufen werden. Auch diese wurde auf der virtuellen Maschine innerhalb der Infrastruktur des Labors Geoinformatik installiert und ist somit unter *http://141.56.141.5/fps/* zu erreichen.

Anwendung

In der *Abbildung 7-13* erkennt man, dass sich unterhalb des Clients einige Zeilen zur Unterstützung der Nutzer befinden. Sie enthalten unter anderem einen Link zur Webseite dieser Diplomarbeit, um dort nähere Informationen über die Funktionsweise des Workaround zu erhalten.

Zur Demonstration eines Beispiels wurde der Karte ein FPS-Layer mit einer SLD-Datei aus dem Style Repository hinzugefügt und wie in *Abbildung 7-14* dargestellt.

Abbildung 7-14: Webseite - Client mit Beispiel-Layer



8 ZUSAMMENFASSUNG UND AUSBLICK

8.1 Ergebniseinschätzung

Bereits durch die Erkenntnisse der vorangegangenen Projektarbeit [4] war von Beginn bekannt, dass die Implementierung eines Feature Portrayal Service und die Entwicklung eines dazugehörigen Clients nicht allzu einfach werden würde. Doch im Zuge dieser Diplomarbeit konnten beide Anforderungen umgesetzt werden.

Die Schwierigkeiten bei der Implementierung des Dienstes auf der Basis eines Open Source Frameworks konnten durch die Umsetzung eines Workaround, der FPS-Fassade, umgangen werden. Dabei standen nur die wesentlichsten Funktionalitäten im Blickpunkt, da die Behebung einiger Probleme bei zukünftigen Software-Veröffentlichungen zu erwarten ist. Die Fassade ermöglicht es, einen Feature Portrayal Service umfangreich zu nutzen und dessen Funktionsweise zu verstehen.

Auch Kartenanwendungen mit SLD-Unterstützung sind noch nicht flächendeckend gängige Praxis. Die Recherchen und Softwaretests im Zuge dieser Diplomarbeit haben gezeigt, dass es noch einige Clients gibt, die entweder keine Einbindung von Styled Layer Descriptor vorgesehen haben oder deren Integration insofern beeinträchtigen, dass SLD nur zur Gestaltung interner Ebenen dient und die Angabe einer Datenquelle in einem UserLayer ignoriert wird. Somit gibt es derzeit nur relativ wenige Produkte in der Open Source Web- und Desktop-GIS-Landschaft, mit denen der Feature Portrayal Service genutzt werden kann. Für das Geoportal des Labors Geoinformatik der

Hochschule wurde ein OpenLayers-Client in eine Webseite eingebaut. Dieser ermöglicht die Einbindung eines beliebigen FPS, verbunden mit der Übergabe einer SLD-Datei aus dem Style Repository.

Alles in allem konnten die zu bearbeitenden Aufgabenstellungen erfüllt werden. Als praktische Ergebnisse befinden sich der deegree FPS, die FPS-Fassade und der FPS-Client in der digitalen *Anlage D*. Nach deren Installation auf dem virtuellen Rechner im Labor Geoinformatik ergeben sich folgende URLs:

- deegree FPS: *http://141.56.141.5/deegree-wms/fps?*,
- FPS-Fassade: *http://141.56.141.5/fps-fassade/wms?*,
- FPS-Client: *http://141.56.141.5/fps/*.

8.2 Einschränkungen und Anregungen

8.2.1 Umsetzung der OGC-Spezifikationen

Die Untersuchung der unterschiedlichen Frameworks hat gezeigt, dass die Dienste der verschiedenen Anbieter leider in einigen Fällen nicht in dem Maß kompatibel zueinander sind, wie es zu erwarten wäre. Das Open Geospatial Consortium [32] hat internationale Standards definiert, auf deren Grundlage diese Dienste entwickelt wurden. Jedoch hat sich ergeben, dass an der einen oder anderen Stelle minimale Interpretationsdifferenzen existieren. Das hat zur Folge, dass die Schnittstellen, die eigentlich problemlos untereinander anwendbar sein müssten, zwischen unterschiedlichen Implementierungen zu Fehlern führen können.

Gerade für einen solchen Dienst, wie den Feature Portrayal Service, ist es aber absolut notwendig, dass die Spezifikationen hundertprozentig

identisch umgesetzt werden. Ob es in Zukunft dazu kommt, bleibt jedoch abzuwarten.

8.2.2 FPS-Fassade

Die Fassade wurde entwickelt, um trotz der Defizite der vorhandenen Frameworks *(siehe Tabelle 5-2)* einen funktionierenden Feature Portrayal Service anbieten zu können.

Der Fokus lag in der Fähigkeit, möglichst alle beliebigen Web Feature Services einbinden zu können. Die zweite Variante, die laut SLD 1.1.0 nicht zwingend erforderliche direkte Integration von GML-Daten als *InlineFeature*, wurde hinten angestellt. Weiterhin kann mit der FPS-Fassade derzeit immer nur genau ein *UserLayer* verarbeitet werden.

Die beiden eben genannten Einschränkungen wurden aufgrund der Annahme in Kauf genommen, in näherer Zukunft neue Veröffentlichungen mit verbesserten Möglichkeiten zum Aufsetzen eines FPS vorzufinden. Sollten sich diese aktuelleren Versionen wider Erwarten nicht als bessere FPS-Grundlagen erweisen, könnte die Fassade dementsprechend erweitert werden.

8.2.3 FPS-Client

Zum erarbeiteten Client ist zu erwähnen, dass dieser nicht den absoluten Schwerpunkt dieser Arbeit darstellen sollte und deshalb nur die wesentlichsten FPS-spezifischen Anforderungen umgesetzt wurden.

Optimal wäre jedoch ein Client, welcher die komplette Bandbreite der Arten und Anwendungsmöglichkeiten von Kartendiensten unterstützt. OpenLayers bietet dafür eine sehr gute Basis. Mit den verfügbaren Methoden könnte der Client um das Speichern und Laden von aktuellen

Kartenkonstellationen mit Web Map Context (WMC), um das Editieren der Geodaten von Transaction WFS oder um die Nutzung eines SLD-Editors erweitert werden. Das sind nur einige Anregungen für zukünftige Projekte.

Da die Daten bei einem FPS nicht direkt vom eingebundenen Dienst kommen, können die Grenzen einer Ebene nicht direkt aus dessen Capabilities entnommen werden. Somit müsste man die Eigenschaften der WFS aus den SLD-Dateien abfragen. Dadurch könnte eine Funktion eingebaut werden, mit der auch auf die Ausdehnung eines FPS-Layer gezoomt werden kann.

Im Blickpunkt dieser Diplomarbeit stand die Implementierung eines funktionierenden Dienstes. Die SLD-Dokumente, die in das Style Repository geladen wurden, beinhalten recht einfache Darstellungen. Zu Anschauungszwecken könnten auch komplexere grafische Ausgestaltungen definiert werden, eventuell auch für einen ganz speziellen Anwendungsfall.

Gesondert soll an dieser Stelle die Anzeige von Legenden erwähnt werden, die in dieser Diplomarbeit keine Rolle spielten. Zu einer guten Karte gehört immer auch eine Legende. Eine nahe liegende Möglichkeit besteht in der Nutzung einer Lösung wie MaSterStyLeS *(siehe Abschnitt 7.3.1)*. Dort wird aus einem neu hochgeladenen SLD-File direkt eine Legende aus dem UserStyle abgeleitet und als Bild zur Verfügung gestellt. Dieses Bild könnte anschließend in die Client-Oberfläche als visuelle Hilfe integriert werden. MaSterStyLeS sieht diese Funktionalität jedoch bisher nur für Dokumente mit NamedLayers vor.

8.3 Ausblick

Mit deegree 3 befindet sich derzeit die nächste Generation des deegree Frameworks in der Entwicklung. Mit dieser neuen Version können auch einige Neuerungen erwartet werden. So wird beispielsweise die aktuellste SLD Spezifikation implementiert sein und somit wird auch die Verarbeitung von GML-Daten als InlineFeature möglich sein.

Eine erste stabile Version ist in naher Zukunft zu erwarten, da bereits erste Testumgebungen zur Prüfung der OGC-Konformität geschaffen wurden.

Auch die Definition des Feature Portrayal Service selbst wird voraussichtlich in der Zukunft noch geändert und erweitert. So könnten beispielsweise zusätzliche Ausgabeformate spezifiziert werden. Auch ein eigenständiger FPS-Standard wäre denkbar. Dafür existierte auch bereits ein Discussion Paper beim Open Geospatial Consortium [21], was jedoch mittlerweile als Retired Document stillgelegt wurde.

Sicher ist, dass der grafischen Darstellung von Kartendiensten auch in Zukunft eine sehr große Bedeutung zukommen wird.

LITERATURVERZEICHNIS

Literaturquellen

[1] Christl, A.; Emde, A.; Schulz, M.: *Mapbender Dokumentation*, Version 2.6, The Open Source Geospatial Foundation, 2009

[2] de La Beaujardiere, J.: *OpenGIS® Web Map Server Implementation Specification*, Version 1.3.0, OGC® 06-042, Open Geospatial Consortium Inc., 2006.

[3] de La Beaujardiere, J.: *Web Map Service Implementation Specification*, Version 1.1.1, OGC 01-068r3, Open Geospatial Consortium Inc., 2002.

[4] Domeyer, M.: *Implementierung eines Portrayal Service*, Hochschule für Technik und Wirtschaft Dresden, 2010.

[5] Expertengruppe Architekturkonzept: *Architekturkonzept der gdi.initiative.sachsen*, Version 1.0, gdi.initiative.sachsen, 2009.

[6] GeoServer: *GeoServer User Manual*, Version 2.0.1, GeoServer, 2010.

[7] Goldstein, N.: *Gaia 3.4 User's Guide*, Carbon Project, Inc., 2009.

[8] Lalonde, W.: *Styled Layer Descriptor Implementation Specification*, Version 1.0.0, OGC 02-070, Open Geospatial Consortium Inc., 2002.

[9] Loos, A.: *deegree desktop Benutzerhandbuch*, lat/lon GmbH, 2010.

[10] Lupp, Dr. M.: *Styled Layer Descriptor profile of the Web Map Service Implementation Specification*, Version 1.1.0, OGC 05-078r4, Open Geospatial Consortium Inc., 2007.

[11] Mays, J.: *deegree iGeoPortal - Standard Edition v2.3*, lat/lon GmbH, Universität Bonn, 2010.

[12] Mays, J.: *deegree Web Map Service v2.3*, lat/lon GmbH, Universität Bonn, 2010.

[13] Müller, Dr. M.: *Symbology Encoding Implementation Specification*, Version 1.1.0, OGC 05-077r4, Open Geospatial Consortium Inc.,

2006.

[14] Portele, C.: *OpenGIS® Geography Markup Language (GML) Encoding Standard*, Version 3.2.1, OGC 07-036, Open Geospatial Consortium Inc., 2007.

[15] Richter, A.: *Erweiterung der Funktionalität von WMS Clients*, Hochschule für Technik und Wirtschaft Dresden, 2009.

[16] Schulze, S.: *Anwendungen der SLD-Spezifikation des OGC*, Hochschule für Technik und Wirtschaft Dresden, 2006.

[17] The MapServer Team: *MapServer Documentation*, Version 5.6.1, MapServer, 2010.

[18] Vretanos, P. A.: *OpenGIS® Filter Encoding Implementation Specification*, Version 1.1.0, OGC 04-095, Open Geospatial Consortium Inc., 2005.

[19] Vretanos, P. A.: *Web Feature Service Implementation Specification*, Version 1.1.0, OGC 04-094, Open Geospatial Consortium Inc., 2005.

[20] Vretanos, P. A.: *Web Feature Service Implementation Specification*, Version 1.0.0, OGC 02-058, Open Geospatial Consortium Inc., 2002.

[21] Woodward, B.; Whiteside, A.: *Feature Portrayal Service*, Version 0.0.30, OGC 05-110, Open Geospatial Consortium Inc., 2005.

Internetquellen

[22] Apache Friends - XAMPP: http://www.apachefriends.org/de/xampp.html (Stand: 16.06.2010)

[23] Apache HTTP Server Project: http://httpd.apache.org/ (Stand: 27.03.2009)

[24] Apache Tomcat: http://tomcat.apache.org/ (Stand: 27.03.2009)

[25] deegree: http://www.deegree.org/ (Stand: 09.07.2010)

[26] Eclipse: http://www.eclipse.org/

(Stand: 20.06.2010)

[27] Gaia: http://www.thecarbonproject.com/gaia.php
(Stand: 06.07.2010)

[28] GeoServer: http://geoserver.org/
(Stand: 17.06.2010)

[29] lat/lon GmbH: http://www.lat-lon.de/
(Stand: 16.06.2010)

[30] Mapbender: http://www.mapbender.org/
(Stand: 28.05.2010)

[31] MapServer: http://mapserver.org/
(Stand: 25.05.2010)

[32] Open Geospatial Consortium Inc.: http://www.opengeospatial.org/
(Stand: 09.07.2010)

[33] OpenJUMP: http://www.openjump.org/
(Stand: 06.07.2010)

[34] OpenLayers: http://openlayers.org/
(Stand: 06.07.2010)

[35] PostgreSQL: http://www.postgresql.org/
(Stand: 28.05.2010)

[36] Quantum GIS: http://www.qgis.org/
(Stand: 06.07.2010)

[37] Sun Microsystems - Java: http://java.sun.com/
(Stand: 19.06.2010)

[38] uDig: http://udig.refractions.net/
(Stand: 06.07.2010)

A KONFIGURATION DES DEEGREE FPS

wms_configuration.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<WMT_MS_Capabilities xmlns:deegree="http://www.deegree.org/wms"
xmlns:sld="http://www.opengis.net/sld" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1.1" updateSequence="1.1.0">
  <deegree:DeegreeParam>
    <deegree:DefaultOnlineResource xlink:type="simple"
            xlink:href="http://141.56.141.5/deegree-wms/fps" />
    <deegree:CacheSize>100</deegree:CacheSize>
    <deegree:MaxLifeTime>3600</deegree:MaxLifeTime>
    <deegree:RequestTimeLimit>45</deegree:RequestTimeLimit>
    <deegree:MapQuality>0.95</deegree:MapQuality>
    <deegree:MaxMapWidth>1500</deegree:MaxMapWidth>
    <deegree:MaxMapHeight>1500</deegree:MaxMapHeight>
    <deegree:AntiAliased>true</deegree:AntiAliased>
    <!--
     copyright note that will be drawn to the left bottom side of the maps;
     you can also reference a graphic file using absolute path to file e.g. c:/images/mylogo.jpg
    -->
    <deegree:Copyright>Feature Portrayal Service, HTWD</deegree:Copyright>
    <deegree:FeatureInfoRadius>10</deegree:FeatureInfoRadius>
    <deegree:DefaultPNGFormat>image/png; mode=24bit</deegree:DefaultPNGFormat>
  </deegree:DeegreeParam>
  <Service>
    <Name>FPS</Name>
    <Title>Feature Portrayal Service</Title>
    <Abstract>SLD 1.0.0 implementation</Abstract>
    <KeywordList>
      <Keyword>deegree</Keyword>
      <Keyword>wms</Keyword>
      <Keyword>fps</Keyword>
      <Keyword>component wms</Keyword>
    </KeywordList>
    <OnlineResource xlink:type="simple" xlink:href=
            "http://141.56.141.5/deegree-wms/fps?" />
    <ContactInformation>
      <ContactPersonPrimary>
        <ContactPerson>Martin Domeyer</ContactPerson>
        <ContactOrganization>Hochschule fuer Technik und Wirtschaft
                    Dresden</ContactOrganization>
      </ContactPersonPrimary>
      <ContactPosition>Student</ContactPosition>
      <ContactAddress>
        <Address>Friedrich-List-Platz 1</Address>
        <City>Dresden</City>
        <StateOrProvince>Sachsen</StateOrProvince>
        <PostCode>01069</PostCode>
        <Country>Germany</Country>
      </ContactAddress>
      <ContactElectronicMailAddress>martin.domeyer@googlemail.com
                </ContactElectronicMailAddress>
    </ContactInformation>
    <Fees>none</Fees>
```

```
    <AccessConstraints>none</AccessConstraints>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:type="simple"
                      xlink:href="http://141.56.141.5/deegree-wms/fps?" />
            </Get>
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <GetMap>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/png; mode=8bit</Format>
        <Format>image/png; mode=24bit</Format>
        <Format>image/jpg</Format>
        <Format>image/jpeg</Format>
        <Format>image/tif</Format>
        <Format>image/bmp</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:type="simple"
                      xlink:href="http://141.56.141.5/deegree-wms/fps?" />
            </Get>
            <Post>
              <OnlineResource xlink:type="simple"
                      xlink:href="http://141.56.141.5/deegree-wms/fps" />
            </Post>
          </HTTP>
        </DCPType>
      </GetMap>
      <GetFeatureInfo>
        <Format>application/vnd.ogc.gml</Format>
        <Format>text/plain</Format>
        <Format>text/html</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:type="simple"
                      xlink:href="http://141.56.141.5/deegree-wms/fps?" />
            </Get>
          </HTTP>
        </DCPType>
      </GetFeatureInfo>
      <GetLegendGraphic>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/jpeg</Format>
        <Format>image/jpg</Format>
        <Format>image/tif</Format>
        <Format>image/bmp</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:type="simple"
                      xlink:href="http://141.56.141.5/deegree-wms/fps?" />
            </Get>
```

```
            </HTTP>
          </DCPType>
        </GetLegendGraphic>
      </Request>
      <Exception>
        <Format>application/vnd.ogc.se_xml</Format>
        <Format>application/vnd.ogc.se_inimage</Format>
        <Format>application/vnd.ogc.se_blank</Format>
      </Exception>
      <UserDefinedSymbolization
            SupportSLD="1" UserLayer="1" UserStyle="1" RemoteWFS="1" />
      <Layer queryable="1" cascaded="0" noSubsets="0"
            xmlns:app="http://www.deegree.org/app">
          <Name>deegree FPS</Name>
        <Title>deegree FPS</Title>
        <LatLonBoundingBox miny="-90" maxy="90" minx="-180" maxx="180" />
      </Layer>
    </Capability>
  </WMT_MS_Capabilities>
```

B QUELLCODE DER FPS-FASSADE

WMS.java

```
package fassade;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class WMS extends HttpServlet {

  private static final long serialVersionUID = 1L;
  public static Assistant assistant = new Assistant();
  private Map<String,String> parameters;
  private String requestKey, sldKey, deegreeWMSUrl, s;
  private BufferedWriter bwDebug;
  private BufferedReader br;

  public WMS() {
      super();
  }

  @SuppressWarnings("unchecked")
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
          ServletException, IOException {
      try{
          //Groß-/Kleinschreibung akzeptieren:
          requestKey="";
          sldKey="";
          parameters=request.getParameterMap();
          for (Map.Entry<String,String> eintrag : parameters.entrySet()){
              if (eintrag.getKey().equalsIgnoreCase("REQUEST")){
                  requestKey=eintrag.getKey();
              }
              else{
                  if (eintrag.getKey().equalsIgnoreCase("SLD")){
                      sldKey=eintrag.getKey();
                  }
              }
          }
          //Sicherstellung, dass REQUEST-Parameter existiert:
          if(requestKey.equals("")){
              PrintWriter out = response.getWriter();
              out.println("REQUEST-Parameter erwartet!");
              out.close();
          }
          else{
```

```
//bei GetMap-Request:
if (request.getParameter(requestKey).equals("GetMap")) {
    //Sicherstellung, dass SLD-Parameter existiert:
    if(sldKey.equals("")){
        PrintWriter out = response.getWriter();
        out.println("SLD-Parameter erwartet!");
        out.close();
    }
    else{
        bwDebug = new BufferedWriter(new FileWriter("webapps/fps-fassade/
                WEB-INF/logs/WmsGetMapDebug.log"));
        //damit bei WFS keine Endlos-Weiterleitung möglich ist:
        assistant.setGfRequestUrl("");
        assistant.setSldUrl(request.getParameter(sldKey));
        bwDebug.write("- SLD-URL "+request.getParameter(sldKey)
                +" an Assistant übergeben");
        bwDebug.newLine();
        //deegreeWMSUrl aus fps-fassade.conf lesen:
        br=new BufferedReader(new FileReader("webapps/fps-fassade/
                WEB-INF/conf/fps-fassade.conf"));
        while ((s = br.readLine()) != null) {
            if (s.contains("deegreeWMSUrl")){
                deegreeWMSUrl=s.substring(s.indexOf("=", 1)+1).trim();
            }
        }
        br.close();
        //URL zur Weiterleizung zusammenbauen
        for (Map.Entry<String,String> eintrag : parameters.entrySet()){
            if (eintrag.getKey().equals(sldKey)){
                deegreeWMSUrl=deegreeWMSUrl.concat("SLD="+assistant.
                        getSldUrl(request.getRequestURL().toString())+"&");
                bwDebug.write("- RemoteWFS-URL gespeichert: "
                        +assistant.getRemoteWFSUrl());
                bwDebug.newLine();
                bwDebug.write("- FeatureTypeName gespeichert: "
                        +assistant.getTypeName());
                bwDebug.newLine();
                bwDebug.write("- SLD-File editiert und lokal abgelegt");
                bwDebug.newLine();
            }
            else{
                //eventuellen LAYERS-Parameter ignorieren
                if (!eintrag.getKey().equals("LAYERS")){
                    deegreeWMSUrl=deegreeWMSUrl.concat(eintrag.getKey()+"="
                        +request.getParameter(eintrag.getKey().toString())+"&");
                }
            }
        }
        WFS.assistant=assistant;
        bwDebug.write("- Assistant-Instanz an WFS übergeben");
        bwDebug.newLine();
        bwDebug.write("- Weiterleitung an deegree-WMS: "
                +deegreeWMSUrl.substring(0,deegreeWMSUrl.length()-1));
        bwDebug.newLine();
        bwDebug.close();
        //Weiterleitung an deegree-WMS:
        response.sendRedirect(response.encodeRedirectURL(
                deegreeWMSUrl.substring(0,deegreeWMSUrl.length()-1)));
    }
}
else{
    //bei GetCapabilities-Request:
    if (request.getParameter(requestKey).equals("GetCapabilities")) {
```

```
                    response.setContentType("text/xml");
                    PrintWriter out = response.getWriter();
                    out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?><!DOCTYPE
                        WMT_MS_Capabilities SYSTEM \"http://schemas.opengis.net/wms/
                        1.1.1/WMS_MS_Capabilities.dtd\" [<!ELEMENT
                        VendorSpecificCapabilities EMPTY>]><WMT_MS_Capabilities
                        updateSequence=\"1.1.0\" version=\"1.1.1\"><Service><Name>FPS
                        </Name><Title>Feature Portrayal Service</Title><Abstract>SLD
                        1.0.0 implementation</Abstract><KeywordList><Keyword>wms
                        </Keyword><Keyword>fps</Keyword><Keyword>component wms
                        </Keyword></KeywordList><OnlineResource xmlns:xlink=\
                        "http://www.w3.org/1999/xlink\" xlink:href=\""
                        + request.getRequestURL().toString()
                        + "?\" xlink:type=\"simple\"/><ContactInformation>
                        <ContactPersonPrimary><ContactPerson>Martin Domeyer
                        </ContactPerson><ContactOrganization>Hochschule fuer Technik und
                        Wirtschaft Dresden</ContactOrganization></ContactPersonPrimary>
                        <ContactPosition>Student</ContactPosition><ContactAddress>
                        <AddressType>postal</AddressType><Address>Friedrich-List-Platz 1
                        </Address><City>Dresden</City><StateOrProvince>Sachsen
                        </StateOrProvince><PostCode>01069</PostCode><Country>
                        Germany</Country></ContactAddress>
                        <ContactElectronicMailAddress>martin.domeyer@googlemail.com
                        </ContactElectronicMailAddress></ContactInformation><Fees>none
                        </Fees><AccessConstraints>none</AccessConstraints></Service>
                        <Capability><Request><GetCapabilities><Format>application/vnd.
                        ogc.wms_xml</Format><DCPType><HTTP><Get><OnlineResource
                        xmlns:xlink=\"http://www.w3.org/1999/xlink\" xlink:href=\""
                        + request.getRequestURL().toString()
                        + "?\" xlink:type=\"simple\"/></Get></HTTP></DCPType>
                        </GetCapabilities><GetMap><Format>image/gif</Format><Format>
                        image/png</Format><Format>image/png; mode=8bit</Format>
                        <Format>image/png; mode=24bit</Format><Format>image/jpg
                        </Format><Format>image/jpeg</Format><Format>image/tif
                        </Format><Format>image/bmp</Format><DCPType><HTTP><Get>
                        <OnlineResource xmlns:xlink=\http://www.w3.org/1999/xlink\
                        xlink:href=\""
                        + request.getRequestURL().toString()
                        + "?\" xlink:type=\"simple\"/></Get><Post><OnlineResource
                        xmlns:xlink=\"http://www.w3.org/1999/xlink\" xlink:href=\""
                        + request.getRequestURL().toString()
                        + "?\" xlink:type=\"simple\"/></Post></HTTP></DCPType>
                        </GetMap></Request><Exception><Format>application/
                        vnd.ogc.se_xml</Format><Format>application/vnd.ogc.se_inimage
                        </Format><Format>application/vnd.ogc.se_blank</Format>
                        </Exception><UserDefinedSymbolization RemoteWFS=\"1\"
                        SupportSLD=\"1\" UserLayer=\"1\" UserStyle=\"1\"/><Layer
                        cascaded=\"0\" noSubsets=\"0\" opaque=\"0\" queryable=\"1\">
                        <Name>FPS</Name><Title>FPS</Title><Abstract>Feature Portrayal
                        Service</Abstract><SRS>EPSG:4326</SRS><LatLonBoundingBox
                        maxx=\"180.0\" maxy=\"90.0\" minx=\"-180.0\" miny=\"-90.0\"/>
                        </Layer></Capability></WMT_MS_Capabilities>");
                    out.close();
                }
            }
        }
    }
    catch (Exception e) {
        PrintWriter out = response.getWriter();
        bwDebug.write("- ERROR: "+e.getMessage());
        bwDebug.close();
        out.println("fps-fassade/wms: Es ist ein Fehler aufgetreten!");
        out.println(e.getMessage());
```

```
            out.close();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
        doGet(request, response);
    }

}
```

WFS.java

```
package fassade;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.URL;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class WFS extends HttpServlet {

    private static final long serialVersionUID = 1L;
    public static Assistant assistant;
    private String requestKey, typeNameKey, dftRequestUrl, s;
    private Map<String,String> parameters;
    private BufferedWriter bwDebug;
    private BufferedReader br;
    private boolean unrealDFT=false, dftTest=false;

    public WFS() {
        super();
    }

    @SuppressWarnings("unchecked")
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
        response.setContentType("text/xml");
        PrintWriter out = response.getWriter();
        try{
            //Groß-/Kleinschreibung akzeptieren:
            parameters=request.getParameterMap();
            for (Map.Entry<String,String> eintrag : parameters.entrySet()){
                if (eintrag.getKey().equalsIgnoreCase("REQUEST")){
                    requestKey=eintrag.getKey();
                }
                else{
                    if (eintrag.getKey().equalsIgnoreCase("TYPENAME")){
                        typeNameKey=eintrag.getKey();
                    }
```

```
        }
    }
    //Sicherstellung, dass REQUEST-Parameter existiert, unnötig,
            da Request nie vom Nutzer kommt
    if (request.getParameter(requestKey).equals("DescribeFeatureType")) {
        bwDebug = new BufferedWriter(new FileWriter("webapps/fps-fassade/
                WEB-INF/logs/WfsDescribeFeatureTypeDebug.log"));
        //DFT-Request zum wahren RemoteWFS zusammenbauen:
        dftRequestUrl=assistant.getRemoteWFSUrl();
        parameters=request.getParameterMap();
        for (Map.Entry<String,String> eintrag : parameters.entrySet(){
            dftRequestUrl=dftRequestUrl.concat(eintrag.getKey()
                    +"="+request.getParameter(eintrag.getKey().toString())+"&");
        }
        //Request kommt von deegree und erfolgt immer in der Version 1.1.0,
                WFS 1.0.0 hat sich aber als stabiler erwiesen:
        dftRequestUrl=dftRequestUrl.replace("VERSION=1.1.0", "VERSION=1.0.0");
        //Test, ob RemoteWFS Version 1.0.0 unterstützt (z.B. deegree2 tut es nicht):
        br = new BufferedReader(new InputStreamReader(new URL(
                dftRequestUrl.substring(0, dftRequestUrl.length()-1)).openStream()));
        while ((s = br.readLine()) != null) {
            //wenn Exception-Response, dann kein WFS 1.0.0-Support
                    -> wieder auf 1.1.0 stellen:
            if(s.contains("ExceptionReport")){
                bwDebug.write("- kein WFS 1.0.0-Support: "
                        +dftRequestUrl.substring(0, dftRequestUrl.length()-1));
                bwDebug.newLine();
                dftRequestUrl=dftRequestUrl.replace("VERSION=1.0.0",
                        "VERSION=1.1.0");
                break;
            }
        }
        br.close();
        bwDebug.write("- besorge DescribeFeatureType-Response von "
                +dftRequestUrl.substring(0, dftRequestUrl.length()-1));
        bwDebug.newLine();
        //wenn es sich um echten DFT-Request handelt:
        if (unrealDFT==false) {
            bwDebug.write("- \"echter\" DescribeFeatureType-Request:");
            bwDebug.newLine();
            //laden, editieren und weitergeben des DFT-Response von RemoteWFS:
            br = new BufferedReader(new InputStreamReader(new URL(
                    dftRequestUrl.substring(0, dftRequestUrl.length()-1)).openStream()));
            while ((s = br.readLine()) != null) {
                //korrigieren der fehlerhaften Datentyp-Angaben bei MapServer-WFS
                s=s.replaceAll("type=\"string\"", "type=\"xsd:string\"");
                out.println(s);
            }
            br.close();
            out.close();
            bwDebug.write("- DescribeFeatureType-Response zurückgegeben");
            bwDebug.newLine();
        }
        //beim zweiten Durchlaufen -> setze dftTest=false
                (für nächsten Darstellungsprozess):
        if(assistant.getGfRequestUrl().contains(request.getParameter(typeNameKey))
                && assistant.getGfRequestUrl().contains(assistant.getRemoteWFSUrl())){
            dftTest=false;
        }
        else{
            dftTest=true;
        }
        bwDebug.write("- setze dftTest="+dftTest);
```

```
            bwDebug.newLine();
            //belegen von dftRequestUrl und gfRequestUrl des Assistant:
            assistant.setDftRequestUrl(dftRequestUrl, request.getRequestURL().toString());
            bwDebug.write("- dftRequestUrl: "+assistant.getDftRequestUrl());
            bwDebug.newLine();
            bwDebug.write("- gfRequestUrl: "+assistant.getGfRequestUrl());
            bwDebug.newLine();
            bwDebug.close();
            //bei unechtem DFT-Request -> Request wird an GetFeature zurückgeleitet
            if(unrealDFT==true){
                unrealDFT=false;
                //aktueller DFT-Request ohne Parameter landet bei GetFeature:
                response.sendRedirect(response.encodeRedirectURL(
                        request.getRequestURL().toString()));
            }
        }
    }
    //GetFeature-Request (da WFS-Requests immer von deegree-WMS kommen, ist sicher das
            es sich um GetFeature handelt, wenn kein DFT-Request):
    catch (Exception e) {
        bwDebug = new BufferedWriter(new FileWriter("webapps/fps-fassade/WEB-INF/
                logs/WfsGetFeatureDebug.log"));
        //wenn kein DFT-Request vorangegangen ist (macht deegree nicht immer),
                muss dieser erzwungen werden:
        if (dftTest==false){
            unrealDFT=true;
            bwDebug.write("- DescribeFeatureType-Request wird erzwungen: "+request.
                    getRequestURL().toString()+"?SERVICE=WFS&VERSION=1.1.0&
                    REQUEST=DescribeFeatureType&TYPENAME="+assistant.getTypeName());
            bwDebug.newLine();
            bwDebug.close();
            response.sendRedirect(response.encodeRedirectURL(request.getRequestURL().
                    toString()+"?SERVICE=WFS&VERSION=1.1.0&REQUEST=
                    DescribeFeatureType&TYPENAME="+assistant.getTypeName()));
        }
        else{
            //für Folgerequests zurücksetzen:
            dftTest=false;
            bwDebug.write("- besorge GML-Daten von "+assistant.getGfRequestUrl());
            bwDebug.newLine();
            //GML-Datei laden, editieren und weitergeben:
            br = new BufferedReader(new InputStreamReader(new URL(
                    assistant.getGfRequestUrl()).openStream()));
            while ((s = br.readLine()) != null) {
                //ersetzen der DFT-Request-URL in der GML-Datei durch Request an
                        WFS-Fassade:
                if (s.contains(assistant.getRemoteWFSUrl())){
                    try{
                        s=s.replace(s.substring(s.indexOf(assistant.getRemoteWFSUrl()),
                                s.indexOf(" ", s.indexOf(assistant.getRemoteWFSUrl())+1)),
                                assistant.getDftRequestUrl());
                        //bei vorhandenem OUTPUTFORMAT-Parameter muss subtype-
                            Konstrukt separat entfernt werden (da Leerzeichen dazwischen):
                        if(s.toLowerCase().contains("subtype")){
                            s=s.replace(s.substring(s.indexOf("subtype"),
                                    s.indexOf(" ", s.indexOf("subtype")+2)), "");
                        }
                    }
                    //wenn URL nicht mit Leerzeichen, sondern mit " endet:
                    catch(Exception ex){
                        s=s.replace(s.substring(s.indexOf(assistant.getRemoteWFSUrl()),
                                s.indexOf("\"", s.indexOf(assistant.getRemoteWFSUrl())+1)),
                                assistant.getDftRequestUrl());
```

```
                    }
                }
                out.println(s);
            }
            br.close();
            bwDebug.write("- GML editiert und weitergegeben");
            bwDebug.newLine();
            bwDebug.close();
            out.close();
        }
    }
}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
    doGet(request, response);
}

}
```

Assistant.java

```
package fassade;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;

public class Assistant {

  private URL sldUrl;
  private BufferedReader br;
  private File sldFile=new File("webapps/fps-fassade/SLD.xml");
  private BufferedWriter bw;
  private String s, remoteWFSUrl, typeName, dftRequestUrl="", gfRequestUrl="";

  public void setSldUrl(String sldUrl) throws MalformedURLException {
      this.sldUrl = new URL(sldUrl);
  }

  public URL getSldUrl(String requestURL) throws IOException {
      //SLD-Datei lesen, editieren und lokal ablegen:
      br = new BufferedReader(new InputStreamReader(sldUrl.openStream()));
      bw = new BufferedWriter(new FileWriter(sldFile));
      while ((s = br.readLine()) != null) {
          //RemoteWFS-URL durch URL der WFS-Fassade ersetzen:
          if (s.contains("xlink:href=")){
              remoteWFSUrl=s.substring(s.indexOf("\"", s.indexOf("xlink:href="))
                      +1,s.indexOf("\"", s.indexOf("\"", s.indexOf("xlink:href="))+1));
              s=s.replace(remoteWFSUrl, requestURL.replace("wms", "wfs")+"?");
              //z.B. bei MapServer (Angabe Mapfile):
              if(!remoteWFSUrl.endsWith("?")){
                  remoteWFSUrl=remoteWFSUrl+"&";
```

```
                }
            }
            //Name des Feauretypes speichern:
            if(s.contains("<FeatureTypeName>")){
                typeName=s.substring(s.indexOf(">", s.indexOf("<FeatureTypeName>"))+1,
                        s.indexOf("<", s.indexOf("<FeatureTypeName>")+1));
            }
            bw.write(s);
            bw.newLine();
        }
        br.close();
        bw.close();
        //lokale SLD-Datei zurückgeben:
        sldUrl=new URL(requestURL.replace("fps-fassade/wms", "fps-fassade/SLD.xml"));
        return sldUrl;
    }

    public String getRemoteWFSUrl() {
        return remoteWFSUrl;
    }

    public String getTypeName() {
        return typeName;
    }

    public void setDftRequestUrl(String dftRequestUrl, String requestURL) {
        //DFT-Request-URL editieren (zu WFS-Fassade):
        this.dftRequestUrl = dftRequestUrl.substring(0, dftRequestUrl.length()-1).
                replace(remoteWFSUrl, requestURL+"?").replaceAll("&", "&amp;");
        //GetFeature-Request-URL erzeugen:
        this.gfRequestUrl = dftRequestUrl.substring(0, dftRequestUrl.length()-1).
                replace("DescribeFeatureType", "GetFeature");
    }

    public String getDftRequestUrl() {
        return dftRequestUrl;
    }

    public String getGfRequestUrl(){
        return gfRequestUrl;
    }

    public void setTypeName(String typeName) {
        this.typeName = typeName;
    }

    public void setGfRequestUrl(String gfRequestUrl) {
        this.gfRequestUrl = gfRequestUrl;
    }

}
```

C QUELLCODE DER CLIENT-WEBSEITE

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.io.BufferedReader"
    import="java.io.InputStreamReader" import="java.net.URL"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
    <title>Feature Portrayal Service</title>
    <link rel="shortcut icon" href="images/favicon.ico"/>
    <meta name="author" content="Martin Domeyer"/>
    <meta name="language" content="de"/>
    <link rel="stylesheet" type="text/css" href="styles/style.css"/>
    <script src='http://maps.google.com/maps?file=api&amp;v=2&amp;
            key=ABQIAAAAwmeZhL3tCC-bwNeKqGLxEhTnRhy2IcOaBxKEqiT7zBpZuljP
            1xR2VkfCqaczGADGRtUbyo2DznmaDw'></script>
    <script src="OpenLayers.js"></script>
    <script type="text/javascript">
        var map, gmapskarte, gmapssatellit, gmapshybrid, ausgabe, ausgabediv, element,
                attribut;
        function init() {
            var options = {
                projection: new OpenLayers.Projection("EPSG:4326"),
                displayProjection: new OpenLayers.Projection("EPSG:4326"),
                units: "m",
                maxExtent: new OpenLayers.Bounds(-20037508, -20037508, 20037508,
                        20037508),
                controls: [
                    new OpenLayers.Control.LayerSwitcher(),
                    new OpenLayers.Control.MousePosition(),
                    new OpenLayers.Control.PanZoomBar({zoomWorldIcon:'true'}),
                    new OpenLayers.Control.KeyboardDefaults(),
                    new OpenLayers.Control.Navigation()
                ]
            };
            map = new OpenLayers.Map('map', options);
            baseLayer();
            map.setCenter(new OpenLayers.LonLat(1528830, 6627340), 2);
            loadMasterstyles();
        }
        function baseLayer() {
            if (document.form.baselayer.checked) {
                gmapskarte = new OpenLayers.Layer.Google(
                    "Google Maps: Karte",
                    {'sphericalMercator': true}
                );
                gmapssatellit = new OpenLayers.Layer.Google(
                    "Google Maps: Satellit",
                    {type: G_SATELLITE_MAP, 'sphericalMercator': true}
                );
                gmapshybrid = new OpenLayers.Layer.Google(
                    "Google Maps: Hybrid",
                    {type: G_HYBRID_MAP, 'sphericalMercator': true}
```

```
            );
            map.addLayers([gmapskarte,gmapssatellit,gmapshybrid]);
        }
        else {
            map.removeLayer(gmapskarte);
            map.removeLayer(gmapssatellit);
            map.removeLayer(gmapshybrid);
        }
    }
    function wmsChange() {
        var wmslist = document.getElementById("wmschoice");
        for (var i = 0; i < wmslist.options.length; i++) {
            if (wmslist.options[i].selected) {
                document.getElementById("wmsurl").value = wmslist.options[i].value;
                if (wmslist.options[i].value != "http://") {
                    if (wmslist.options[i].value ==
                            "http://141.56.141.5/deegree-wms/fps") {
                        document.getElementById("wmsname").value = "FPS";
                    }
                    else {
                        document.getElementById("wmsname").value = "FPS-Fassade";
                    }
                    document.form.fpsrequest.checked = true;
                    document.form.fpsrequest.disabled = true;
                    wmsType();
                }
                else {
                    document.getElementById("wmsname").value = "WMS";
                    document.form.fpsrequest.checked = false;
                    document.form.fpsrequest.disabled = false;
                    wmsType();
                }
                break;
            }
        }
    }
    function wmsType() {
        if (document.form.fpsrequest.checked) {
            document.form.layersparam.checked = false;
            document.form.layersparam.disabled = true;
            document.getElementById("paramlayers").disabled = true;
            document.form.stylesparam.checked = false;
            document.form.stylesparam.disabled = true;
            document.getElementById("paramstyles").disabled = true;
            document.form.sldparam.checked = true;
            document.form.sldparam.disabled = true;
            document.getElementById("paramsld").disabled = false;
        }
        else {
            document.form.layersparam.disabled = false;
            document.form.stylesparam.disabled = false;
            document.form.sldparam.disabled = false;
            if (document.form.layersparam.checked) {
                document.getElementById("paramlayers").disabled = false;
            }
            else {
                document.getElementById("paramlayers").disabled = true;
            }
            if (document.form.stylesparam.checked) {
                document.getElementById("paramstyles").disabled = false;
            }
            else {
                document.getElementById("paramstyles").disabled = true;
```

```
            }
            if (document.form.sldparam.checked) {
                document.getElementById("paramsld").disabled = false;
            }
            else {
                document.getElementById("paramsld").disabled = true;
            }
        }
    }
    function loadMasterstyles() {
        ausgabediv = document.getElementById('styledIv');
        ausgabe = document.getElementById('stylelist');
        ausgabediv.removeChild(ausgabe);
        element = document.createElement("form");
        attribut = document.createAttribute("id");
        attribut.nodeValue = "stylelist";
        element.setAttributeNode(attribut);
        ausgabediv.appendChild(element);
        ausgabe = document.getElementById('stylelist');
        var div;
        <%
            BufferedReader br = new BufferedReader(new InputStreamReader(new URL(
                    "http://geoinformatik.htw-dresden.de/masterstyles/sms?service=sld
                        &amp;request=view").openStream()));
            String l,s="";
            int i=0,j=1;
            while ((l = br.readLine()) != null) {
                s=s.concat(l);
            }
            br.close();
            s = s.replaceAll("\n","").replaceAll("\r","");
            while ((s.indexOf("<strong>Name:</strong> ",j)) != -1) {
                out.println("div = document.createElement(\"div\")");
                out.println("attribut = document.createAttribute(\"class\")");
                out.println("attribut.nodeValue = \"mBox\"");
                out.println("div.setAttributeNode(attribut);");
                out.println("element = document.createElement(\"b\")");
                out.println("element.appendChild(
                        document.createTextNode(\"Name: \"))");
                out.println("div.appendChild(element)");
                i=s.indexOf("<strong>Name:</strong> ",j);
                out.println("div.appendChild(document.createTextNode(
                        \""+s.substring(i+23,s.indexOf("<",i+23))+"\"));");
                j=s.indexOf("<strong>Autor:</strong> ",i);
                i=s.indexOf("<strong>Datum:</strong> ",j+1);
                j=s.indexOf("<strong>Kategorie:</strong> ",i+1);
                out.println("div.appendChild(document.createElement(\"br\"))");
                out.println("element = document.createElement(\"b\")");
                out.println("element.appendChild(
                        document.createTextNode(\"Beschreibung: \"))");
                out.println("div.appendChild(element)");
                i=s.indexOf("<td>",s.indexOf("<",j+28)+1);
                out.println("div.appendChild(document.createTextNode(
                        \""+s.substring(i+4,s.indexOf("<",i+4))+"\"));");
                out.println("div.appendChild(document.createElement(\"br\"))");
                out.println("element = document.createElement(\"b\")");
                out.println("element.appendChild(document.createTextNode(\"URL: \"))");
                out.println("div.appendChild(element)");
                j=s.indexOf("<td>http",i);
                out.println("element = document.createElement(\"input\")");
                out.println("attribut = document.createAttribute(\"type\")");
                out.println("attribut.nodeValue = \"radio\"");
                out.println("element.setAttributeNode(attribut);");
```

```
                    out.println("attribut = document.createAttribute(\"name\")");
                    out.println("attribut.nodeValue = \"sldurl\"");
                    out.println("element.setAttributeNode(attribut);");
                    out.println("attribut = document.createAttribute(\"value\")");
                    out.println("attribut.nodeValue =
                            \""+s.substring(j+4,s.indexOf("<",j+4))+"\"");
                    out.println("element.setAttributeNode(attribut);");
                    out.println("div.appendChild(element);");
                    out.println("div.appendChild(document.createElement(\"br\"))");
                    out.println("element = document.createElement(\"a\")");
                    out.println("attribut = document.createAttribute(\"href\")");
                    out.println("attribut.nodeValue =
                            \""+s.substring(j+4,s.indexOf("<",j+4))+"\"");
                    out.println("element.setAttributeNode(attribut);");
                    out.println("attribut = document.createAttribute(\"target\")");
                    out.println("attribut.nodeValue = \"_blank\"");
                    out.println("element.setAttributeNode(attribut);");
                    out.println("element.appendChild(document.createTextNode(
                            \""+s.substring(j+4,s.indexOf("<",j+4))+"\"));");
                    out.println("div.appendChild(element);");
                    out.println("ausgabe.appendChild(div)");
                }
            %>
        }
        function addMasterstylesSLD() {
            var check = false;
            for (var i = 0; i < document.getElementById('stylelist').sldurl.length; i++) {
                if (document.getElementById('stylelist').sldurl[i].checked) {
                    document.getElementById("paramsld").value =
                            document.getElementById('stylelist').sldurl[i].value;
                    check = true;
                    break;
                }
            }
            if (!check) {
                alert("Bitte zuerst auf der rechten Seite eine SLD-Datei auswählen!");
            }
        }
        function addWms() {
            if (document.form.layersparam.checked && document.form.stylesparam.checked
                    && document.form.sldparam.checked) {
                layer = new OpenLayers.Layer.WMS(
                    document.getElementById("wmsname").value,
                    document.getElementById("wmsurl").value,
                    {layers: document.getElementById("paramlayers").value, styles:
                            document.getElementById("paramstyles").value, sld: document.
                            getElementById("paramsld").value, transparent: 'TRUE'},
                    {singleTile: true}
                );
            }
            if (document.form.layersparam.checked && document.form.stylesparam.checked
                    && !document.form.sldparam.checked) {
                layer = new OpenLayers.Layer.WMS(
                    document.getElementById("wmsname").value,
                    document.getElementById("wmsurl").value,
                    {layers: document.getElementById("paramlayers").value, styles: docu-
                            ment.getElementById("paramstyles").value, transparent: 'TRUE'},
                    {singleTile: true}
                );
            }
            if (document.form.layersparam.checked && !document.form.stylesparam.checked
                    && document.form.sldparam.checked) {
                layer = new OpenLayers.Layer.WMS(
```

```
                    document.getElementById("wmsname").value,
                    document.getElementById("wmsurl").value,
                    {layers: document.getElementById("paramlayers").value, sld:
                        document.getElementById("paramsld").value, transparent: 'TRUE'},
                    {singleTile: true}
                );
            }
            if (document.form.layersparam.checked && !document.form.stylesparam.checked
                    && !document.form.sldparam.checked) {
                alert("STYLES- oder SLD-Parameter notwendig!");
            }
            if (!document.form.layersparam.checked && document.form.stylesparam.checked
                    && document.form.sldparam.checked) {
                alert("LAYERS-Parameter notwendig!");
            }
            if (!document.form.layersparam.checked && document.form.stylesparam.checked
                    && !document.form.sldparam.checked) {
                alert("LAYERS-Parameter notwendig!");
            }
            if (!document.form.layersparam.checked && !document.form.stylesparam.checked
                    && document.form.sldparam.checked) {
                layer = new OpenLayers.Layer.WMS(
                    document.getElementById("wmsname").value,
                    document.getElementById("wmsurl").value,
                    {sld: document.getElementById("paramsld").value, transparent: 'TRUE'},
                    {singleTile: true}
                );
            }
            if (!document.form.layersparam.checked && !document.form.stylesparam.checked
                    && !document.form.sldparam.checked) {
                alert("Parameter notwendig!");
            }
            layer.isBaseLayer = false;
            map.addLayer(layer);
            reloadLayers();
        }
        function reloadLayers() {
            ausgabediv = document.getElementById('layersdiv');
            ausgabe = document.getElementById('layerslist');
            ausgabediv.removeChild(ausgabe);
            element = document.createElement("form");
            attribut = document.createAttribute("id");
            attribut.nodeValue = "layerslist";
            element.setAttributeNode(attribut);
            ausgabediv.appendChild(element);
            ausgabe = document.getElementById('layerslist');
            element = document.createElement("h2");
            element.appendChild(document.createTextNode("Layer-Übersicht"));
            ausgabe.appendChild(element);
            var p;
            for (var i = 3; i < map.getNumLayers(); i++) {
                p = document.createElement("p");
                element = document.createElement("input");
                attribut = document.createAttribute("type");
                attribut.nodeValue = "checkbox";
                element.setAttributeNode(attribut);
                attribut = document.createAttribute("name");
                attribut.nodeValue = "checklayers";
                element.setAttributeNode(attribut);
                p.appendChild(element);
                p.appendChild(document.createTextNode(map.layers[i].name));
                ausgabe.appendChild(p);
            }
```

```
                p = document.createElement("p");
                element = document.createElement("input");
                attribut = document.createAttribute("type");
                attribut.nodeValue = "button";
                element.setAttributeNode(attribut);
                attribut = document.createAttribute("value");
                attribut.nodeValue = "Gewählte Layer entfernen";
                element.setAttributeNode(attribut);
                attribut = document.createAttribute("onclick");
                attribut.nodeValue = "removeWMS()";
                element.setAttributeNode(attribut);
                p.appendChild(element);
                ausgabe.appendChild(p);
            }
            function removeWMS() {
                var l = map.layers.length, j = 0;
                for (var i = 3; i < l; i++) {
                    try {
                        if (document.getElementById('layerslist').checklayers[i-3].checked) {
                            map.removeLayer(map.layers[i+j]);
                            j=j-1;
                        }
                    } catch (e) {
                        if (document.getElementById('layerslist').checklayers.checked) {
                            map.removeLayer(map.layers[i]);
                        }
                    }
                }
                reloadLayers();
            }
        </script>
    </head>
    <body style="font-size: 100%;" onload="init()">
        <div id="wrapper">
            <div id="kopfbereich">
                <div id="ls">
                    <a href="http://www.htw-dresden.de/" target="_blank">
                        <img src="images/logo_htw.gif" alt="Logo Hochschule f&uuml;r
                            Technik und Wirtschaft Dresden" title="Logo Hochschule
                            f&uuml;r Technik und Wirtschaft Dresden"/>
                    </a>
                    <a href="index.jsp">
                        <img src="images/logo_htw_fps.gif" alt="Schriftzug Feature
                        Portrayal Service" title="Schriftzug Feature Portrayal Service"/>
                    </a>
                    <div id="servicenav">
                        <a href=http://141.56.141.5/DA_DOMEYER_2010/
                                target="_blank">
                            Diplomarbeit Martin Domeyer
                        </a>
                         
                        <span>|</span>
                         
                        <a href="mailto:martin.domeyer@googlemail.com"
                                target="_blank">
                            Kontakt
                        </a>
                    </div>
                </div>
                <div id="kopfimg"></div>
                <div id="navbartop">
                    <p>
                        |  
```

```
                <a href="http://www.htw-dresden.de/fakultaet-geoinformation/
                        labore/geoinformatik/geoportal.html" title="Geoportal"
                        target="_blank">
                    Geoportal
                </a>
                  |   
                <a href="http://www.htw-dresden.de/fakultaet-geoinformation/
                        labore/geoinformatik.html" title="Labor Geoinformatik"
                        target="_blank">
                    Labor Geoinformatik
                </a>
                  |   
                <a href="http://www.htw-dresden.de/fakultaet-geoinformation.
                    html" title="Fakult&auml;t Geoinformation" target="_blank">
                    Fakult&auml;t Geoinformation
                </a>
                  |   
                <a href="http://www.htw-dresden.de/" title="HTW Dresden"
                        target="_blank">
                    HTW Dresden
                </a>
                  |
            </p>
        </div>
    </div>
    <div id="inhaltsbereich_gesamt">
        <div id="navigation">
            <div class="zwischenraum"></div>
            <form name="form" action="">
                <p>
                    <input type="checkbox" name="baselayer" checked
                            onchange="baseLayer()"/>
                    Google Maps als Basis-Layer
                </p>
                <p>
                    WMS:
                    <br/>
                    <select id="wmschoice" size="1" onchange="wmsChange()">
                        <option value="http://">
                            bitte WMS-URL eingeben...
                        </option>
                        <option value="http://141.56.141.5/fps-fassade/wms">
                            FPS-Fassade
                        </option>
                        <option value="http://141.56.141.5/deegree-wms/fps">
                            deegree FPS
                        </option>
                    </select>
                </p>
                <p>
                    <input type="checkbox" name="fpsrequest"
                            onchange="wmsType()"/>
                    Feature Portrayal Service
                </p>
                <p>
                    WMS-URL:
                    <input type="text" id="wmsurl" value="http://" size="30"/>
                </p>
                <p>
                    Bezeichnung:
                    <input type="text" id="wmsname" value="WMS" size="30"/>
                </p>
                <p>
```

```
                <input type="checkbox" name="layersparam"
                        onchange="wmsType()"/>
                LAYERS=
                <input type="text" id="paramlayers" size="30" disabled/>
            </p>
            <p>
                <input type="checkbox" name="stylesparam"
                        onchange="wmsType()"/>
                STYLES=
                <input type="text" id="paramstyles" size="30" disabled/>
            </p>
            <p>
                <input type="checkbox" name="sldparam"
                        onchange="wmsType()"/>
                SLD=
                <input type="text" id="paramsld" value="http://" size="30"
                        disabled/>
                <input type="button" value="SLD aus Style Repository"
                        onclick="addMasterstylesSLD()"/>
            </p>
            <br/>
            <input type="button" name="addwmsbutton"
                    value="Layer hinzuf&uuml;gen" onclick="addWms()"/>
        </form>
        <div class="layerOverview" id="layersdiv">
            <form id="layerslist" style="font-size:0.75em; color:#808080">
                <p>Diese Internetseite wurde getestet mit Firefox 3.6 und dem
                        Internet Explorer 8.</p>
                <p>Ältere Browser (z.b. IE 6) können Darstellungs- und
                        Funktionalitätsprobleme verursachen.</p>
            </form>
        </div>
    </div>
    <div id="inhaltsbereich">
        <div id="inhalt">
            <h1>
                FPS-Client
            </h1>
            <div id="map" class="mapArea">
                <noscript>
                    <h2 align="center">
                        <br/>
                        JavaScript ist in Ihrem Browser deaktiviert!
                        <br/>
                        Um den FPS Client nutzen zu können, muss JavaScript
                                aktiviert sein!
                        <br/>
                    </h2>
                </noscript>
            </div>
            <br/>
            <p>
                Dieser Client basiert auf der JavaScript-Bibliothek OpenLayers
                und wurde in erster Linie für den Einsatz als FPS-Client
                entwickelt. Dennoch kann er auch als normaler
                (SLD-) WMS-Client genutzt werden.
            </p>
            <p>
                Ein FPS (Feature Portrayal Service) ist ein spezieller WMS zur
                Darstellung unpräsentierter GML-Daten fremder WFS. Für
                nähere Informationen sei auf die zugehörige
                <a href=http://141.56.141.5/DA_DOMEYER_2010/
                target="_blank">Diplomarbeit</a>
```

```
                    verwiesen. Dort sind auch Erläuterungen zum deegree FPS und
                    zur FPS-Fassade zu finden.
                </p>
                <p>
                    Auf der linken Seite neben dem Client können beliebige Ebenen
                    dynamisch hinzugefügt und auch wieder entfernt werden. Soll
                    ein Parameter angegeben werden, muss zuvor das
                    dazugehörige Kontrollkästchen aktiviert werden.
                </p>
                <p>
                    Dir URL im SLD-Parameter kann direkt aus dem
                    <a href="http://www.htw-dresden.de/fakultaet-geoinforma-
                    tion/labore/geoinformatik/geoportal/legendenservice.html"
                    target="_blank">Style Repository der HTW Dresden</a>
                    eingefügt werden. Dazu muss in der rechten Spalte ein
                    SLD-Dokument ausgewählt und übernommen werden.
                </p>
                <p>
                    Durch einen Klick auf das Plus in der oberen rechten Ecke der
                    Karte kann die Layer-Liste aufgeklappt werden. Dort kann die
                    Sichtbarkeit der Ebenen eingestellt werden. Zur Navigation
                    stehen neben den vorhandenen Steuerelementen die typischen
                    Maus-Funktionalitäten für Pan und Zoom zur Verfügung.
                    Alternativ können die Plus- und Minus- sowie die Pfeiltasten
                    der Tastatur genutzt werden. Bei gedrückter SHIFT-Taste kann
                    ein Rechteck gezogen werden, um einen neuen
                    Kartenausschnitt zu definieren.
                </p>
            </div>
            <div id="randspalte">
                <div class="zwischenraum"></div>
                <div class="mBox">
                    <h2>
                        <a href="http://www.htw-dresden.de/fakultaet-geoinfor-
                        mation/labore/geoinformatik/geoportal/legendenservice.
                        html" title="Legendenservice" target="_blank">
                            Style Repository
                        </a>
                    </h2>
                    <input type="button" value="SLD übernehmen"
                            onclick="addMasterstylesSLD()"/>
                </div>
                <div id="styledIv" style="height:500px; overflow:auto;">
                    <form id="stylelist"></form>
                </div>
            </div>
        </div>
    </div>
  </div>
 </body>
</html>
```

D DIGITALE ANLAGE

Inhalt der beiliegenden CD

\DIPLOMARBEIT_DOMEYER_2010.pdf (digitale Version dieser Diplomarbeit)

\deegree-wms (als FPS vorkonfigurierter deegree WMS)

\frameworks

- ***konfigurationen*** (WMS-Konfigurationen zu Testzwecken)
  - ***deegree-wms***
  - ***geoserver***
  - ***mapserv.map***
- ***sld*** (Test-SLD-Dokumente)

\sld (SLD-Dokumente)

\war (Web Archive zur Integration in eine Servlet Engine)

- ***fps-fassade.war***
- ***fps.war***

\workspace (Eclipse-Workspaces)

- ***TestWorkaround***
- ***Workaround***
- ***Webseite***

EIDESSTATTLICHE ERKLÄRUNG

Hierdurch erkläre ich, dass ich die Diplomarbeit mit dem Titel „Implementierung eines Feature Portrayal Service" selbstständig verfasst und ausschließlich die angegebenen Literaturquellen benutzt habe.

Dresden, 13.07.2010

Martin Domeyer