FAKULTÄT FÜR !NFORMATIK

Erstellung von E-Learning Einheiten zur Implementierung OGC-konformer Web Clients

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Informatikmanagement

eingereicht von

Edin Pezerovic
Matrikelnummer 9509572

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuerin: Ass. Prof. Dr. Monika DiAngelo

Wien, 14.10.2009

(Unterschrift Verfasser) (Unterschrift Betreuerin)

Erstellung von E-Learning Einheiten zur Implementierung OGC-konformer Web Clients

Autor: Edin Pezerovic MSc (WU), BSc (WU)

Betreuerin: Ass. Prof. Dr. Monika DiAngelo

Institute of Computer Aided Automation

Automation Systems Group

Technische Universität Wien

Wien, am 9. Oktober 2009

Kurzfassung/Abstract

Diese Diplomarbeit beschäftigt sich mit der Erstellung eines eLearning Tutorials, dessen Inhalt die Implementierung eines Web-Clients zur Grundstücksuche ist. Für die Erstellung des Tutorials wird eLessons Markup Language (eLML) verwendet. Grundlage von eLML ist ein eigenes XML Schema, welches zur Validierung der Inhalte der Lektionen dient. Ein valides XML Dokument kann daraufhin mittels XSLT (XML Style Sheet Transformation) in verschiedene Formate umgewandelt werden. Die Darstellungsinformationen werden dabei nicht mit dem Inhalt vermischt, sondern kommen erst durch den Schritt der Generierung hinzu. Dies bietet den Vorteil, dass der gleiche Inhalt in unterschiedlichen Formaten und Darstellungsformen ausgegeben werden kann. Die Erzeugung einer neuen Darstellungsform wird durch ein Template bewerkstelligt. Der Inhalt des Tutorials ist die Implementierung eines Web-Clients, der eine geografische Grundstücksuche bietet. Durch Eingabe der Grundstücksnummer wird das entsprechende Grundstück in einem Kartenfenster markiert. Die Suche erfolgt unter Einhaltung der Spezifikation des Open Geospatial Consortium (OGC), genauso wie die Darstellung der Karte im Kartenfenster. Für die Aufbereitung der Karte wird ein Web Map Service (WMS), für die Suche der Grundstücke ein Web Feature Service (WFS) verwendet. Der Web-Client wird mittels der frei verfügbaren JavaScript Bibliothek OpenLayers implementiert und greift auf Services des Bundesamt für Eich- und Vermessungswesen zu.

This diploma thesis deals with the design of an e-learning tutorial for building a parcel search web page. The tutorial is created using the open source library eLML (eLessons Markup Language) which uses a generation approach. The content of the tutorial is written in XML and generated using XSLT into different output formats. eLML defines a XML schema which is used to validate the content. A valid content may be transformed into a e-learning tutorial of various formats. One of those formats is a pdf-file where eLML uses XSL-FO to transform the content. Another output format could be a html-file. By dividing content and style eLML offers the possibility to change the look of the outcome without affecting the content of the tutorial. The style is encapsulated into templates which are not difficult to create. The subject of the tutorial is the implementation of a web page for searching parcels. The user types a parcel number with optional wildcards and executes the search. The web client uses Web Feature Services (WFS) to query the result and displays it on top of a map. This map is requested from a Web Map Services (WMS) and displayed within a map window. The free open source library of OpenLayers is used for all accesses to services which are compliant to the specifications issued by the Open Geospatial Consortium (OGC). Those services are provided freely by the Austrian Public Agency for Land Surveying (Bundesamt für Eich- und Vermessungswesen, BEV) for the purpose of this tutorial.

Inhalt

Abbildungs- und Tabellenverzeichnis

Vorwort

Im Zuge einiger Seminare zum Thema Fachdidaktik schaffte es Professor DiAngelo, mein Interesse an dem Themengebiet der Fachdidaktik zu wecken. Vor allem die Beschäftigung mit den Robotern von LEGO® Mindstorms® gestaltete sich äußerst spannend. Daher war es ein Leichtes, Professor DiAngelo um die Betreuung meiner Magisterarbeit zu bitten. Für die große Hilfsbereitschaft, die zahlreichen Ratschläge und das mitgeteilte Wissen möchte ich mich hiermit bei ihr bedanken.

Weiteren Dank möchte ich meiner Familie aussprechen, insbesondere meiner Frau und meinem Sohn für ihr Verständnis, ihre Geduld und die Mithilfe beim Korrekturlesen.

1 Einleitung

Das Internet unterliegt laufenden Veränderungen und verwendeten Technologien sind bereits wieder veraltet, bevor die Mehrheit der Entwickler von ihnen Notiz genommen hat. Gleichzeitig werden die Anforderungen an Web-Applikationen immer umfangreicher. Selbst die Entwicklung einfacher Webseiten setzt ein Wissen in vielen verschiedenen Technologien voraus. Da sich in der heutigen Zeit soziale Netzwerke besonders hoher Beliebtheit erfreuen, wird dieses Vorgehen umso mehr beschleunigt. Es wird erwartet, dass alle Web-Sites eine eigene Community aufbauen, Wikis zur Verfügung stellen und im Idealfall alles mittels Geo-Tagging verlink- und suchbar zu gestalten. Daher wachsen in den letzten Jahren die Einstiegsbarrieren für junge Entwickler im Bereich Internet enorm. Ein Absolvent einer Universität, der sich zum Beispiel PHP im Selbststudium beigebracht hat, kann heute höchstens für kleine Vereine Homepages erstellen.

Einen Spezialbereich der Informationstechnologie stellt die Geoinformation (GIS) dar, über den nur ein geringer Prozentsatz der Entwickler fundierte Kenntnisse aufweisen. Die dahinter stehende Technologie existiert schon seit den frühen 70er Jahren, doch erst jüngst drängen die GIS-Hersteller, allen voran Google und Microsoft, in den Internet Bereich. Das große Schlagwort hier ist spätestens seit dem Jahr 2005 Web-GIS, denn 2004 kaufte Google die Firma Keyhole Corporation und veröffentlichte im Jahr 2005 das Produkt Google Earth/Maps. Anfangs als Spielzeug abgetan, entpuppte sich die Geoinformation als ein äußerst beliebtes Tool. Heutzutage existieren bereits digitale Kameras, die zu jedem Foto automatisch die GPS Koordinaten speichern. Ein weiteres Beispiel ist OpenStreetMap.org, eine Homepage, auf welcher Privatpersonen ehrenamtlich die Straßen ihrer Heimatgemeinden kartografisieren und über das Internet zur Verfügung stellen.

Aufgrund dieses starken Trends steigen die Anforderungen an die Webpräsenzen der Unternehmen enorm, Georeferenzierung – in welcher Form auch immer – auf den Web-Sites anzubieten (z.B.: ein Filialfinder, der dem Kunden die nächste Filiale auf einer Karte anzeigt).

Ziel dieser Arbeit ist folglich die Entwicklung eines eLearning Tutorials, das auch Entwicklern mit geringen Vorkenntnissen die Möglichkeit bietet, geografische Inhalte in Homepages einzubinden. Das Tutorial soll den Lernenden durch die einzelnen Schritte der Web-GIS Implementierung führen, um einen Web-Client zu erzeugen, der folgenden Kriterien entspricht:

- Verwendung ausschließlich offener Standards
- Exklusive Verwendung von Open Source
- Zugriff auf die geografischen Daten mittels standardisierter Schnittstellen
- Implementierung eine geografische Suche nach Grundstücken
- Automatische Markierung der Ergebnisse im Kartenfenster

Die hier angesprochenen Standards kommen vom Open Geospatial Consortium (OGC, 2009), welches sich seit 1994 um die Definition von anerkannten, offenen Standards für

Schnittstellen zu Geodaten-Services bemüht. Von diesen werden Web Map Services (WMS) und Web Features Services (WFS) verwendet (OGC, 2005b, 2006). WMS liefern die benötigte Karte für das Kartenfenster und WFS bieten die Möglichkeit, nach Attributen von geografischen Objekten zu suchen.

Der Web-Client implementiert nicht die Services, sondern nur den Zugriff auf diese. Daher müssen die Services (WMS und WFS) bereits zur Verfügung stehen. Das Bundesamt für Eich- und Vermessungswesen (BEV) hat für diese Diplomarbeit dankenswerterweise die benötigten Services implementiert und frei zur Verfügung gestellt. Durch das WMS ist es sogar möglich, alle Daten der Digitalen Katastralmappe darzustellen.

Der Zugriff auf die Services wird im Tutorial mittels OpenLayers umgesetzt. OpenLayers ist eine frei verfügbare JavaScript Bibliothek, die die meisten OGC-Standards unterstützt.

Das Tutorial ist so angelegt, dass Lernende ohne Lehrenden problemlos den Erklärungen folgen können. Daher ist es als eLearning Tutorial verfasst und mittels eLesson Markup Language (eLML) umgesetzt (eLML, 2009). eLML bietet die Möglichkeit, eLearning Tutorials unabhängig der Darstellungsform umzusetzen. Die verwendeten Technologien sind XML und XSL und bieten den Vorteil, dass das Ergebnis erst zum Zeitpunkt der Generierung festgelegt wird. Die Ausgabe kann als HTML, PDF, OpenDocument, LMS-Module oder LaTex Datei erfolgen.

Das folgende 2. Kapitel fasst die Hintergründe zu den Standards und dem Standardisierungskonsortium zusammen. Es wird erklärt, wer das OGC ist, wie es zu einem Standard kommt und welche der zur Zeit bestehenden Standards durch den Web-Client des Tutorials verwendet werden. Des Weiteren wird etwas über die historische Entwicklung von OpenLayer berichtet und schlussendlich die genaue Aufgabenstellung des Web-Clients beschrieben.

Kapitel 3 beschäftigt sich mit dem Thema eLearning und geht dabei auf Allgemeines sowie auf eLearning in Bezug auf Geoinformationssysteme ein. Den Abschluss bildet die Erklärung von eLML und warum dieses verwendet wird.

Im 4. Kapitel werden die didaktischen Hintergründe behandelt und das Design des eLearning Tutorials aufgezeigt. Zusätzlich werden Einsatzmöglichkeiten des Tutorials im Zuge einer Schulung genannt.

In Kapitel 5 wird das Thema durch eine Zusammenfassung sowie einen kurzen Ausblick abgerundet.

Der Anhang enthält das generierte Tutorial.

2 Web Services mit geografischem Kontext

Grundlage des eLearning Tutorials dieser Arbeit ist eine Web-Applikation, die geografische Daten visualisiert. Die dafür benötigten Daten werden durch Web Services geliefert, die offenen GIS-Standards entsprechen. Dieses Kapitel stellt eine Einführung in die Technologie der Geoinformationssysteme (GIS) mit den dort vorhandenen Standards dar. Zum Abschluss wird die Aufgabenstellung der Web-Applikation detailiert.

2.1 Open Geospatial Consortium (OGC)

Das Open Geospatial Consortium Inc. ist ein international tätiges Konsortium von 383 Unternehmen, Behörden und Hochschulen, die sich zusammen geschlossen haben, um öffentliche Standards für Schnittstellen zu erstellen (OGC, 2009). Diese Standards sind implementierungsunabhängig und sollen systemunabhängig die Vision eines geografischen Internets („geo-enabled web“) ermöglichen.

Vor der Gründung des OGC existierte eine Vielzahl von GIS Implementierungen, die vorwiegend im Verteidigungssektor und der Öffentlichen Verwaltung eingesetzt wurden. Jede dieser Implementierungen stellte Insellösungen dar, die ausschließlich für die Zwecke des jeweiligen Kunden implementiert oder angepasst wurden. Teilweise entwickelten Kunden selbst individuelle Geoinformationssysteme eigenständig entwickelt, die sogar später kommerziell verkauft wurden (GE-Energy, 2009). Warum keiner dieser Hersteller an eine Verbindung zwischen verschiedenen GIS unterschiedlicher Firmen dachte, scheint plausibel – je proprietärer eine Lösung war, desto garantierter war der zukünftige Umsatz für Upgrades und Wartung.

Im Jahre 1994 wurde dann OGC von acht Gründungsmitgliedern mit der Vision ins Leben gerufen, dass diverse Geoinformationssysteme direkt über das Netzwerk mittels offener Schnittstellen miteinander kommunizieren können. In den folgenden zehn Jahren wuchs die Mitgliederanzahl auf über 250 und beinhaltet neben traditionellen GIS Unternehmen auch solche, die an vorderster Front mit neuen Technologien arbeiten.

Partnerschaften mit anderen Standardisierungsgremien und –konsortien stärken den Rückhalt der erstellten Standards enorm. Als wichtigster Partner ist hier die International Organization for Standardization (ISO) zu nennen, mit der gemeinsam Standards bezüglich der Darstellung von geografischen Informationen zu Objekten und Phänomenen erstellt werden (ISO, 2009).

Seit dem Jahr 2000 wurden neben dem Hauptsitz in Massachusetts weitere Zweigniederlassungen der OGC in Europa, Asien sowie eine Niederlassung für die Förderung akademischer Partnerschaften gegründet.

Die aktuelle Vision des OGC lautet „Realization of the full societal, economic and scientific benefits of integrating electronic location resources into commercial and

institutional processes worldwide.“ (OGC, 2009), aus welcher das OGC folgende fünf strategischen Ziele ableitet (Pezerovic, 2009):

i. Bereitstellung freier und öffentlich verfügbarer Standards für den Markt, konkreter Werte für die Mitglieder und messbare Vorteile für die Benutzer;
ii. weltweit führend in der Erstellung und Etablierung von Standards zu sein, die es erlauben, raumbezogenen Inhalt und Services nahtlos in Geschäfts- und Zivilprozesse, das Internet mit GIS-Inhalten und Unternehmensapplikationen zu integrieren;
iii. Erleichterung des Einsatzes von offenen – um geografische Inhalte erweiterte – Referenzarchitekturen in die Unternehmensarchitekturen weltweit;
iv. Unterstützung der Bildung von neuen, innovativen Märkten und Applikationen für raumbezogene Technologien durch Verbesserung von definierten Standards;
v. Beschleunigung der Angleichung des Marktes für Forschungen im Bereich der Interoperabilität durch gemeinschaftliche Prozesse innerhalb des OGC.

Das OGC definiert einen Standardisierungsprozess, der die Erreichung der Ziele durch die Mitglieder fördern soll. Als offenen Standard verstehen die Mitglieder des OGC Spezifikationen, die nachstehende Punkte erfüllen:

- In einem öffentlichen und internationalen Prozess geschaffen
- Beteiligung der wichtigsten Wissensträger der Industrie am Standardisierungsprozess
- Recht der freien Verteilung
- Öffentlicher, uneingeschränkter Zugang zu den Spezifikationen
- Keine Diskriminierung oder Ausschließung von Personen oder Gruppen
- Technische Neutralität der Spezifikation und Nutzung

Alle Mitglieder sind hierdurch aufgerufen, innerhalb des OGC die Standardisierung einer neuen Spezifikation anzustoßen. Dies erfolgt, wenn eines der Mitglieder ein Problem in der Interoperabilität verschiedener Systeme entdeckt. Zum Beispiel gab es das Problem, dass nicht systemübergreifend geografische Karten abgefragt und angezeigt werden konnten. Hierfür fehlte es an der Spezifikation, wie solche Karten abgefragt und erstellt werden. Ein Mitglied kann nun den Wunsch vorbringen, solch eine Spezifikation zu erstellen. Alle Wünsche der Mitglieder werden gesammelt, in Gruppen diskutiert und letztendlich priorisiert. Es kann durchaus vorkommen, dass andere Mitglieder des OGC dieses Problem nicht als wichtig erachten oder gar kein Problem sehen und somit der Wunsch der Spezifikation durch das Mitglied abgelehnt wird.

Sobald ein Problem als solches erkannt und auch akzeptiert wird, erstellen die Mitglieder zusammen die Anforderungen der neuen Schnittstelle. Diese Anforderungen steuern bereits das endgültige Design der Spezifikation (OGC, 2009). Daraufhin wird der erste Entwurf („candidate standard“) der Spezifikation in Zusammenarbeit von einzelnen Mitglieder oder einer dauerhaft eingerichteten Working Group erzeugt und dem Specification Program (SP) zugeführt. Die Gruppe, die mit der Erstellung des Entwurfes

betraut wird, setzt sich in der Regel aus einer bunten Mischung der Mitglieder zusammen, wobei großer Wert auf eine internationale und größenmäßige Durchmischung der Unternehmen gelegt wird. Im Zuge des Specification Programs werden die Entwürfe den anderen Mitgliedern und der breiten Öffentlichkeit zur Begutachtung und Kommentierung vorgelegt. Dies erfolgt durch die Verfügbarmachung auf der Homepage des OGC. Kritik, Änderungs- oder Erweiterungswünsche werden durch die Mitglieder des SP eingearbeitet und wieder veröffentlicht.

Nach Einarbeitung der Rückmeldungen wird aus dem Entwurf eine Vorlage zur Spezifikation („draft standard"), über welche alle Mitglieder des OGC formal abstimmen. Durch den zyklischen Verbesserungsprozess werden fast alle Vorlagen einstimmig angenommen. Daraufhin wird der neue Standard auf der Homepage des OGC veröffentlicht und Hersteller dazu animiert, diesen neuen Standard umzusetzen.

2.2 Web Service Standards für geografische Inhalte

Wie im obigen Kapitel beschrieben befasst sich das Open Geospatial Consortium mit der Definition von Standards für Schnittstellen. Das Ziel dieser Arbeit ist die Erstellung eines eLearning Tutorials für die Implementierung eines Web-Clients zur Grundstücksuche, welcher auf Web Services mit geografischen Inhalten zugreifen muss. Dieses Subkapitel erläutert nun die Spezifikationen, die für die Entwicklung des Web-Clients benötigt werden. Ein Teil des eLearning Tutorials ist die genaue Beschreibung der Schnittstellen der folgenden beiden Standards, weswegen hier die Ausführungen auf einer höheren Ebene gehalten werden.

2.2.1 Web Map Service

Die Darstellung von geografischen Informationen erfolgt meistens in Form von Karten, welche dynamisch zur Laufzeit berechnet werden. Der Benutzer navigiert durch verschieben und vergrößern innerhalb der Karte, um den gewünschten Kartenausschnitt zu erhalten. Genau dafür werden Web Map Services (WMS) verwendet (OGC, 2006).

Geografische Informationen werden in zwei Formen in der GIS-Datenbank gespeichert:

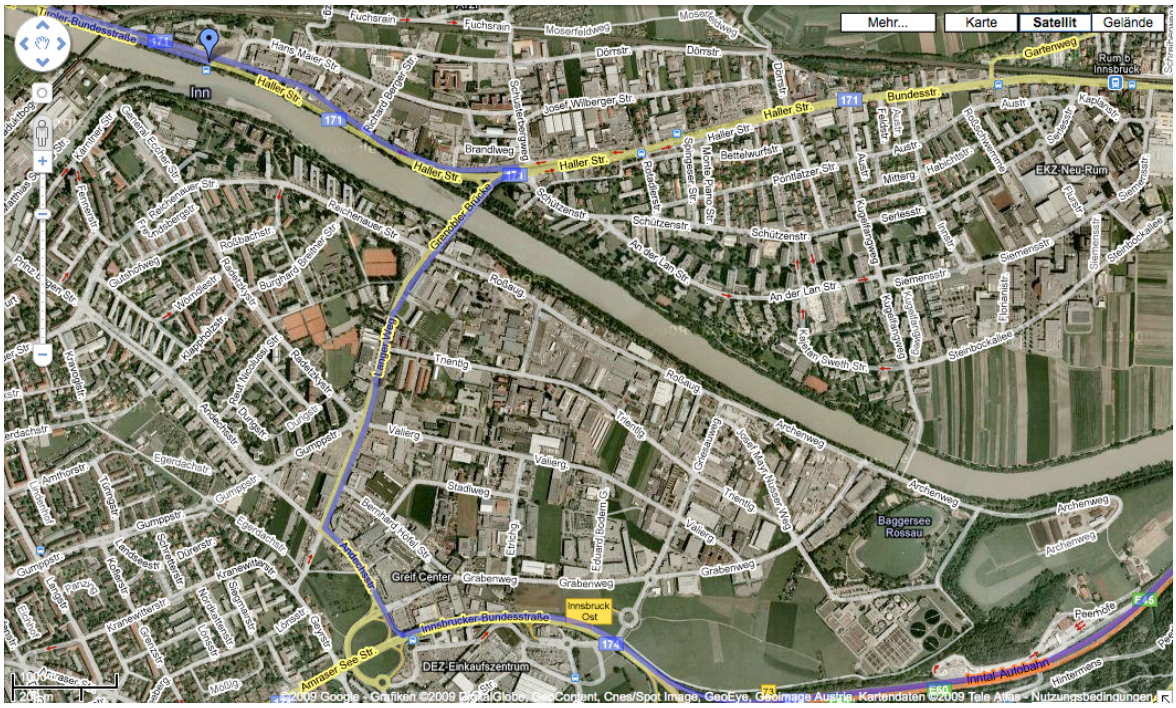
- Rasterformat: Grafiken, die die Oberfläche in Pixel auflösen (=Raster)
- Vektorformat: die Daten werden als Attributwerte in der Datenbank gespeichert und zur Laufzeit zur Berechnung und Erstellung der Grafik herangezogen

Das Problem des Rasterformates ist die Auflösung der Daten. Wenn der Benutzer das Kartenfenster zu stark vergrößert, entstehen je nach Komprimierungsverfahren des Bildes unschöne Artefakte. Man spricht davon, dass das Bild „aufpixelt". Im Gegensatz dazu können Vektordaten in jedem beliebigen Maßstab angezeigt werden, ohne Artefakte hervorzubringen.

Natürlich kann eine endgültige Karte auch aus Raster- und Vektordaten errechnet werden. Dabei wird ein Verfahren verwendet, welches Layering genannt wird und auch in der

Bildverarbeitung Verwendung findet. Beim Layering besteht ein Bild aus mehreren übereinanderliegenden Ebenen (layer), die zusammen das endgültige Bild ergeben. Die höheren Ebenen müssen gewisse Teile transparent darstellen, um noch Informationen der darunterliegenden Ebenen sichtbar erscheinen zu lassen. Zum Beispiel könnte auf ein Rasterbild mit Satellitenaufnahmen eine Ebene mit Grundstücksgrenzen gelegt werden. Dieses Verfahren ist auch bei Google Maps (Google, 2009) in Verwendung. Folgende Abbildung zeigt ein Beispiel von drei Ebenen, wobei hier Straßen und Routeninformationen über Satellitenbilder gelegt werden:

Abb. 1: Snapshot Google Maps



Diese Grafik wird durch ein WMS erzeugt, wobei die Anfrage die Koordinaten des Kartenfensters enthält. Die Koordinaten werden in Form einer Bounding Box mitgegeben, die das Rechteck des Kartenausschnittes durch zwei diagonale Ecken definiert.

Das Problem bei WMS liegt in der Art der Antwort. Da es sich hierbei lediglich um Grafiken handelt, können keine Objektinformationen ermittelt werden. In der oben dargestellten Grafik ist die Haller Straße 111 in Innsbruck mittels blauem Marker gekennzeichnet. Dieser Marker ist jedoch nur ein Teil der Pixelmenge des Bildes und hat keine weitere semantische Bedeutung. Um zusätzliche Informationen zu dem Marker ermitteln zu können, wäre eine Objektdefinition inklusive der geografischen Koordinaten des Objektes nötig. Solche Objekte werden durch die GML Spezifikation der OGC definiert (OGC, 2007).

2.2.2 Web Feature Service

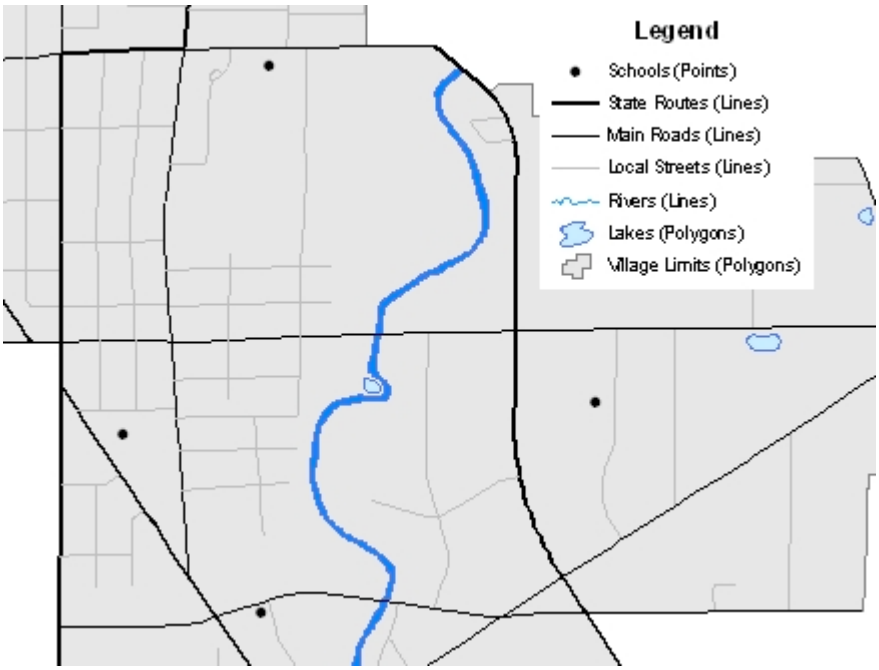
Es existieren Anwendungsfälle, wo Benutzer nicht eine erzeugte Karte anzeigen wollen, sondern direkten Zugriff auf die Attributwerte der geografischen Objekte (= Features) benötigen. Diese Attributwerte werden den Feldern der Vektordaten aus der GIS-Datenbank entnommen. Jeder Vektordatensatz entspricht in der GIS-DB einer Tabelle mit Attributen, die entweder Sachdaten oder geografische Daten enthalten können. Zumindest ein Feld der Tabelle muss geografische Werte speichern.

Ein Web Feature Service (WFS) (OGC, 2005b) bietet die Möglichkeit, auf solche Daten zuzugreifen, wobei die Inhalte als GML-Objekte geliefert werden (OGC, 2007). Im Gegensatz zu WMS muss hier nicht ausschließlich die Bounding Box des aktuellen Kartenfensters als Abfrage geliefert werden. Vielmehr ist eine attributative Abfrage im Zuge einer Suche möglich. Jedes Attribut der Tabelle in der GIS-DB kann somit Teil einer einschränkenden Abfrage gegen das WFS sein. Das Ergebnis ist dementsprechend auf die gewünschten Elemente eingeschränkt. Auf diese Art können Suchfunktionen leicht implementiert werden.

Die verwendete Abfragesprache ist als XML-Schema definiert und wurde von dem OGC als Filter Encoding spezifiziert (OGC, 2005a). Durch Verwendung von Filter Encoding ist eine Abfrage sowohl aufgrund von Sachdaten, als auch von Features einschränkbar. Bei Features gibt es beispielsweise Abfragen, in welchen ein Punkt innerhalb des Ergebnis-Features liegen muss. Die Auswertung der Abfrage erfolgt mittels Verschneidung der Anfrage-Features mit den gespeicherten Features des Layers.

Das Ergebnis enthält, wie bereits erwähnt, das GML-Objekt der Ergebnis-Features. Das GML-Objekt beinhaltet alle Attributwerte inklusive der geografischen Features. Geografische Features können unter anderem Punkte, Linien und Polygone sein, wobei jeweils die tatsächlichen Koordinaten im Ergebnis geliefert werden. Dies kann unter Umständen Probleme bei kostenpflichtigen Inhalten mit sich bringen, da WFS weder ein Verrechnungsmodel noch eine –möglichkeit bietet. Dies muss durch diejenige Organisation proprietär bewerkstelligt werden, die das WFS anbietet. Folgende Abbildung zeigt ein Beispiel einer Karte mit Punkt, Linie und Polygon (VNILES, 2009):

Abb. 2: Beispiele für GML-Objekte



2.3 Die Entwicklung von OpenLayers

Die oben vorgestellten Spezifikationen definieren lediglich die Schnittstelle zwischen Client und Server. Für die Darstellung der Karten ist eine Software auf Clientseite notwendig. OpenLayers ist eine frei verfügbare (open source) JavaScript Bibliothek, die in der Lage ist, Informationen von WFS und WMS in Kartenfenstern darzustellen.

OpenLayers (OpenLayers, 2009) wurde von dem Unternehmen MetaCarta (MetaCarta, 2009) in einer initialen Version entwickelt und im Jahr 2005 der Organisation Open Source Geospatial Foundation (OSGeo, 2009) zur weiteren Entwicklung überlassen. Das Besondere an OpenLayers stellt die Art der Kartendarstellung dar. Vor 2005 wurden bereits Kartenfenster in Webapplikationen eingebunden. Das Problem hierbei war, dass jede Interaktion des Benutzers mit der Karte ein Neuladen der Seite erzwang, sobald die Karte sich auch nur minimal veränderte. Erst mit Google Maps (Google, 2009) wurde im Jahr 2005 ein neues Konzept zum Laden und Darstellen von Karten entwickelt. Dieses Konzept sah die Aufteilung der darzustellenden Karte in sogenannte Tiles, die einzeln mittels AJAX vom Server geladen und letztendlich am Browser mittels Javascript wieder zusammengesetzt werden können. Der Vorteil liegt in der Größe der Grafiken, die nachgeladen werden müssen. Verschiebt der Benutzer den Kartenausschnitt, müssen nur die neu hinzugekommenen Tiles nachgeladen werden. Ein Neuaufbau der gesamten Seite ist somit nicht mehr nötig.

Google Maps stellte diese neue Art der Kartendarstellung 2005 auf O'Reilly's „Where 2.0 Conference" vor (O'Reilly, 2006). Dieser Vortrag inspirierte die Verantwortlichen bei MetaCarta, eine frei verfügbare JavaScript Bibliothek zu implementieren, welche die

gleiche Funktionalität bieten konnte. Bereits auf der Nachfolge-Konferenz konnte MetaCarta die ersten Ergebnisse präsentieren.

Durch OSGeo.org wird die Bibliothek laufend um neue Funktionalitäten erweitert. In der aktuellen Version 2.7 werden Zugriffe auf WMS und WFS unterstützt, wobei der WFS-Zugriff nicht vollständig der WFS-Spezifikation entspricht. Hier ist für die nachfolgende Version 2.8 eine Komplettunterstützung angekündigt.

Das eLearning Tutorial dieser Arbeit verwendet OpenLayers für die Abfrage von Karten vom WMS, die Suche nach Grundstücken mittels WFS und die Darstellung des Ergebnisses als eigenen Layer in der Karte.

2.4 Aufgabenstellung des Web-Clients

Das Ziel dieser Arbeit ist die Erstellung eines eLearning Tutorials, innerhalb welchen ein Web-Client zur Grundstücksuche implementiert wird. In diesem Subkapitel soll nun kurz der Inhalt des Web-Client zum leichteren Verständnis des eLearning Tutorials erklärt werden.

Der Web-Client dient vordergründig der Grundstücksuche, wobei anhand der Grundstücksnummer gesucht werden soll. Die hierzu benötigten Web Services wurden vom Bundesamt für Eich- und Vermessungswesen (BEV) zur Verfügung gestellt. Laut BEV definiert sich eine Grundstücksnummer aus folgenden Teilen:

- Katastralgemeindennummer: fünfstellige, identifizierende Nummer einer Katastralgemeinde mit führenden Nullen (zB: 01004)
- Bauflächenindikator: optionaler Punkt, der früher für die Identifikation von Gebäuden verwendet wurde (zB.: ‚.')
- Stammnummer: bis zu fünfstelliger Hauptteil der Grundstücksnummer (zB: 24)
- Unterteilungsnummer: bis zu fünfstelliger, optionaler Teil der Grundstücksnummer, der durch einen Schrägstrich von der Stammnummer getrennt angegeben wird; wurde früher für Teilungen von Grundstücken verwendet, um eine Art Teilungshistorie innerhalb der Grundstücksnummer darstellen zu können (zB: 7)

Die gesamte Grundstücksnummer des oben angegebenen Beispieles lautet somit ‚.24/7' in der Katastralgemeinde 01004. ‚.24/7' zeigt, dass es einst ein Grundstück mit der Nummer ‚.24' gab (Gebäudefläche), welches dann in mehrere Teile geteilt wurde, sagt jedoch nicht aus, dass das Grundstück ‚.24' in mindestens 7 Teile aufgeteilt wurde. Es kann durchaus aus einem Grundstück ‚.24/3' durch dessen Teilung hervorgegangen sein. Daher kann aus der Grundstücksnummer der historische Zusammenhang nicht herausgelesen werden.

Der Benutzer gibt auf der Suchseite des Web-Clients eine Grundstücksnummer an, wobei auch wildcards erlaubt sind. Nach Start der Suche werden die Suchkriterien an ein WFS zur Suche übergeben. Das WFS liefert dann eine Liste der Bounding Boxes der einzelnen Grundstücke zurück. Dies hat den Zweck, dass die gefundenen Grundstücke in ihrer

Gesamtheit im Kartenfenster dargestellt werden und kein Teil eines Grundstückes nicht sichtbar ist.

Als nächster Schritt wird das Kartenfenster so verschoben und vergrößert, dass das gesamte Ergebnis, also die Summe aller Bounding Boxes, innerhalb des Kartenfensters liegt. Für die Darstellung wird daraufhin der Karteninhalt an der nun gefundenen Stelle durch das WMS ermittelt und angezeigt. Der letzte Schritt stellt die Markierung der gefundenen Grundstücke im Kartenfenster dar. Hierfür wird aus dem Ergebnis des ersten WFS-Aufrufes ein neuerlicher WFS-Aufruf erzeugt, welcher die Einsetzpunkte der Grundstücksnummern laut BEV abfragt. Das BEV garantiert, dass die Einsetzpunkte der Grundstücksnummern, soweit möglich, innerhalb der Grundstücksfläche liegt. Daher ist dies der ideale Platz, um einen Marker für das Ergebnis zu setzen.

Folgende Abbildung zeigt ein Beispiel eines Suchergebnisses mit mehr als einem Treffer:

Abb. 3: Beispiel eines Suchergebnisses

3 eLearning Tutorial Grundlagen

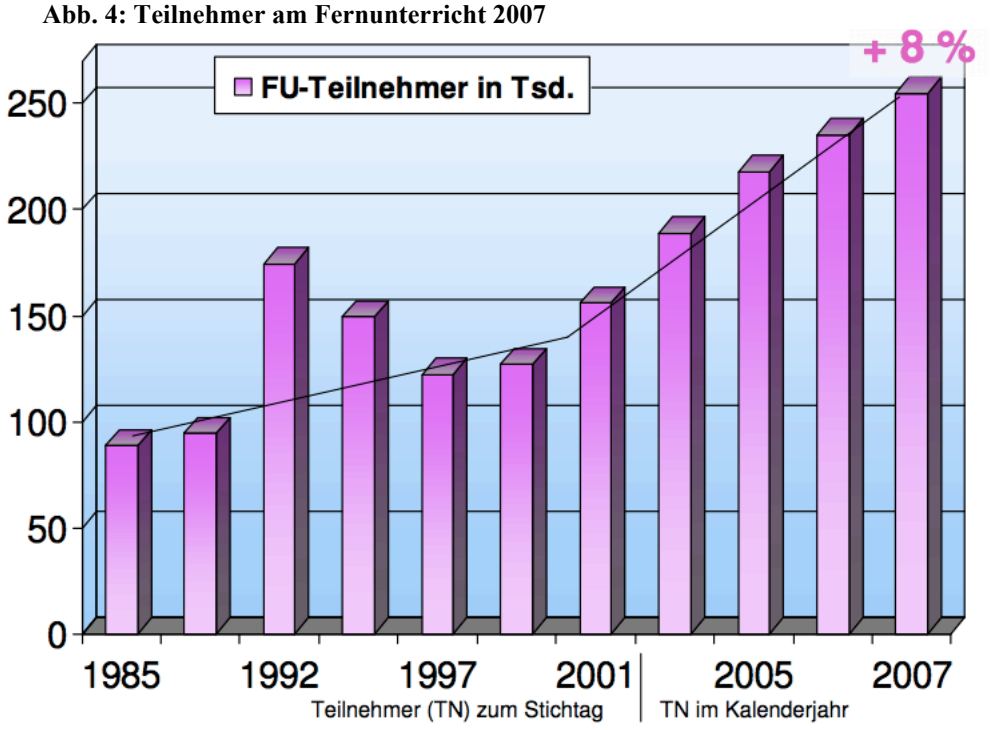
Der Schwerpunkt dieses Kapitels liegt in der Definition des Begriffes eLearning und dem davon betroffenen thematischen Umfeld. Beginnend mit dem Hintergrund werden verschiedene Begriffe im Naheverhältnis zu eLearning abgegrenzt und der Begriff eLearning, wie er in dieser Diplomarbeit gesehen wird, definiert. Anschließend werden aktuelle Initiativen zu eLearning im Gebiet der Geoinformation beleuchtet. Der letzte Teil des Kapitels enthält die Beschreibung des Projektes eLML, welches für die Implementierung des eLearning Tutorials verwendet wird.

3.1 Hintergrund und Einsatz von eLearning

Fernunterricht ist die Vermittlung von Wissen über eine räumliche oder zeitliche Trennung zwischen Informationsquelle und lernender Person. Statt einem direkten Treffen des Lehrenden und des Schülers, tritt hier eine Freiheit ins Spiel, wo Zeit und Ort des Unterrichtes frei gewählt werden. Die Vermittlung des Wissens erfolgt auf Basis des Austausches von Unterrichtsmaterial, sei es in Papierform oder elektronischer Form, zeitgleich oder zeitlich versetzt. Gewisse Kurse im Bereich Fernunterricht bedürfen einer zeitweiligen Präsenz an der Informationsquelle, um etwa Prüfungen oder Präsenzveranstaltungen abzuhalten. Solche hybriden Unterrichtsformen werden auch Blended Learning genannt.

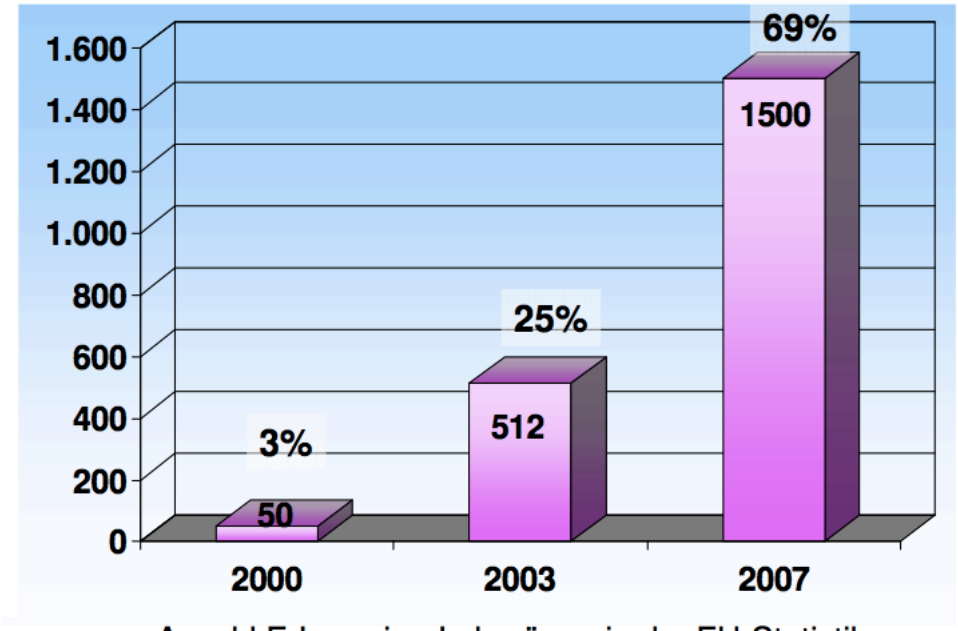
Der Fernunterrichtmarkt wuchs in den vergangenen Jahre stetig, wie Abbildung 1 zeigt (DistancE-Learning, 2008):

Abb. 4: Teilnehmer am Fernunterricht 2007



Ein großer Teil des Fernunterrichtangebots bildet eLearning. Abbildung 2 zeigt den Anteil von eLearning am Marktvolumen des Distance Learnings (DistancE-Learning, 2008):

Abb. 5: Marktanteil der eLearning Lehrgänge am Fernunterricht



Die nun folgenden Kapitel beschäftigen sich eingehender mit den Begriffen eLearning und Blended Learning.

3.1.1 eLearning 1.0 und eLearning Tools

Eine einheitliche Definition für eLearning gibt es nicht, da der Umfang, den eLearning abdeckt, so vielfältig ist, dass eine Definition unmöglich alles umfasst. Dennoch soll hier eine, für diese Arbeit gültige Definition, aus den bestehenden Definitionen abgeleitet werden.

„By eLearning, also e-Learning, we understand best practices for learning in the new economy, implying but not requiring benefits of networking and computers such as anywhere/anytime delivery, learning objects, and personalization. It often includes instructor led training.“ (Varis, 2006)

Dies definiert eLearning als einen Lernprozess der, optional durch Computer unterstützt, auf jeden Fall ungebunden bezogen auf Zeit und Ort stattfindet. Die folgende, tiefer gehende Definition beschreibt den in dieser Arbeit verwendeten Ansatz des Zugriffes auf Lernmaterialien über das Internet. Der Computer dient hierbei nicht als Grundlage sondern als Kommunikationsmittel zwischen Lehrer und Schüler.

„E-learning (or online education as it is still commonly termed) has been variously defined, but can be simply described as a learning process in which learners can

communicate with their instructors and their peers, and access learning materials, over the Internet or other computer networks. It therefore provides a means through which the powerful and pervasive computing and communications technologies can be applied to tertiary education – and to some of the key challenges now facing universities.“ (Curran, 2004)

Die ersten eLearning Möglichkeiten wurden durch Universitäten geschaffen. Diese dienten den Vortragenden als eine Vereinfachung der Verteilung der Lehrunterlagen. Für die Verteilung wurden Webapplikationen verwendet, die zumeist von den Mitarbeitern der Informatik Institute entwickelt wurden. Daraus entwickelten sich je Institut unterschiedliche, vor allem aber inkompatible Lösungen, da die Verwendung ausschließlich institutsintern angedacht war. Durch die vermehrte Verwendung entwickelten einige Universitäten Strategien, wie eLearning in den gesamten Unterricht einfließen kann. Das Potential von eLearning konnte nur dann voll ausgenutzt werden, wenn alle Institute eine gemeinsame Strategie verfolgten. Dadurch ist es für die Studenten einfacher, Lernmaterialien institutsübergreifend auf die gleiche Weise zu finden. Die Lernkurve bei der Benutzung der einzelnen Webseiten entfällt damit.

Der nächste Schritt folgte in die Richtung, den Lehrinhalt auch universitätsfremden Schülern zur Verfügung zu stellen. Diese Erweiterung zielt vor allem auf Studienabbrecher ab, denen auf diese Weise ermöglicht wird, Wissen aufbauen zu können. Daraus entwickelte sich auch der Ansatz der Open Universities (Curran, 2004), wo Studenten ein Abschluss ermöglicht wird, die vielleicht in ihrer Jugend nicht die Möglichkeiten eines Studiums hatten oder den Postgraduate Masterlehrgang nicht mehr in Präsenzform belegen wollen. Ziel ist es, den Zugriff auf Informationen jedem zur Verfügung zu stellen. Es ist jedoch problematisch dieses Ziel zu erreichen, da durch den Einsatz der Technologie des Internets und von Computern ein noch beträchtlicher Teil der Bevölkerung und hier vor allem die Personengruppen aus der Zielgruppe der Erwachsenen, von der Nutzung dieser Angebote ausgeschlossen wird. Trotzdem gilt diese Personengruppe als Zukunftsmarkt, da mit dem Alter auch die Wahrscheinlichkeit zunimmt, die Motivation und das Durchhaltevermögen für eine Ausbildung im zweiten Bildungsweg aufzubringen und die Ausbildung auch abzuschließen (Curran, 2004).

eLearning Methoden können auch Personen mit Behinderungen eine Ausbildung ermöglichen, die vielleicht sonst nicht so möglich gewesen wäre. In Japan gibt es bereits eine Reihe von Projekten, die den Fokus auf die Ausbildung von behinderten Menschen legen. Eines dieser Projekte, NPO (Unlimited Potential Ohita Project for disabled e-learners), entstand aus dem Ergebnis eines Lehrprogrammes der Firma Microsoft (Bunt-Kokhuis und Bolger, 2009). Ein weiteres Projekt der Universität von Nagano verwendet eLearning Methoden, um Studenten mit Hörbehinderungen das Lernen zu erleichtern.

Grundsätzlich dient die Technologie nur als Wegbereiter der Bildung und nicht als Garant für Erfolg, denn dieser ist abhängig von der Qualität des Lehrinhaltes.

„A commitment to enhancing the quality of teaching and learning, especially for students on-campus, is a recurrent aim of university e-learning strategies – especially in traditional, research-intensive universities.“ (Curran, 2004)

Hier wird eLearning als Bereichung des direkten Lehrens angesehen und steigert somit die Qualität des Lehrbetriebes. Qualität soll jedoch nicht nur als Teil des Lehrinhaltes betrachtet werden, der am Ende der Erstellung bedacht wird, sondern ein essenzieller Part bei der Entwicklung der eLearning Strategien. Die Definition der Qualität ist von der Zielgruppe abhängig und je nach Zielgruppe sind andere Kriterien als wichtig zu erachten. Daher kann Qualität auch nicht allgemein definiert werden (Ehlers, 2007). Die Zielgruppe und deren Bedürfnisse müssen immer im Fokus behalten werden, während die eLearning Strategie entwickelt und umgesetzt wird. Daher ist es für den Lernenden von besonderer Bedeutung, für seine Ansprüche die qualitativ beste Ausbildung zu finden. Daraus folgt ein Kompetenzbedarf, das Richtige auszuwählen.

„Digital competence implies the capacity to find, select, judge and evaluate good quality online content.“ (Bunt-Kokhuis und Bolger, 2009)

Die Suche nach Lehrinhalten gestaltet sich trotzdem schwierig. Die oben beschriebene Kompetenz muss von den Lernenden mühsam erworben werden, um sinnvolle Bildung zu erlangen. Einfacher wäre es, wenn die Lehrinhalte auf gleiche Art und Weise such- und vergleichbar wären. Diesen Weg versuchen einige Initiativen mittels Standards zu definieren. Dabei werden Standards für Metadaten, Lernobjekte und –architekturen definiert, die bei Einhaltung neue Möglichkeiten eröffnen (Varis, 2006):

- Vergleich und Kombination von Lehrinhalten verschiedener Quellen
- Wiederverwendung von Lehrinhalten
- Unabhängigkeit von der Technologie
- Minimierung der Investitionsrisiken in Technologie

Daher wird eine Definition von Standards im Bereich eLearning als kritisch für einen Erfolg und breite Umsetzung erachtet. Dies sollte auch bei der Strategiewahl bedacht werden.

Bei der eLearning Strategieentwicklung können zwei Ausrichtungen unterschieden werden (Seufert und Euler, 2004):

- Innenorientierung: dient der Verbesserung der Qualität der Hochschullehre
- Außenorientierung: dient dem Gewinn neuer Kunden und Zielgruppen

Die Qualitätsanforderungen an eLearning Inhalten spiegeln sich auch in der Definition der Nachhaltigkeit wider.

„Die Nachhaltigkeit von eLearning-Innovationen zielt auf eine dauerhafte Implementierung und Nutzbarmachung der Potenziale von eLearning in einer Organisation. Wesentliche Bedingungen für die Erzielung von Nachhaltigkeit sind der Nachweis eines pädagogischen Mehrwerts, eine ökonomische Effizienz im Ressourceneinsatz, die Flexibilität und Effizienz der Gestaltung der

Organisationsstrukturen und –prozesse, eine stabile und problemgerechte Technologie sowie die zielbezogene Gestaltung der Lern- und Organisationskultur. Insbesondere der letztgenannte Faktor bringt ein potenzialorientiertes Verständnis von Nachhaltigkeit zum Ausdruck.“ (Seufert und Euler, 2004)

Im Mittelpunkt des pädagogischen Mehrwerts stehen folgende Implementierungsfelder (Seufert und Euler, 2004):

- Gestaltung der verwendeten Medien (Tutorials, Homepages, Videos, ...)
- Aufbau des Unterrichtes in Form von Lernphasen (Abfolge verschiedener Methoden des Methodenpools der Universität Köln)
- Gesamtdidaktisches Design des Kurses (eLesson-Design anhand von ECLASS (Gerson, 2000))

Gerade bei der Erwachsenenbildung stehen die Lehrmethoden im Brennpunkt. Hier wird deutlich, dass eine Mischung zwischen Präsenzveranstaltungen und eLearning die Möglichkeiten und Bedürfnisse der Schüler am besten berücksichtigen (Scholze und Wiemann, 2007). Aufgrund der Unterschiede im Vorwissen, der verfügbaren Zeit und, bei internationalen Kursen, der Zeitzonen, sind eLearning Kurse zielführend, da jeder Teilnehmer zu der für ihn idealen Zeit am Kurs teilnehmen kann. Die Präsenz-veranstaltungen wiederum führen zu einer Gruppenbildung und Kohäsion, die den Lernerfolg positiv beeinflusst.

In den letzten Jahren kam der Begriff lebenslanges Lernen in Mode. eLearning bietet die Möglichkeit, lebenslanges Lernen zu unterstützen. In Betrieben liegt der Schwerpunkt beim eLearning im Benefit für das Unternehmen. Dadurch unterscheiden sich die Anforderungen an Kurse im betrieblichen Umfeld von denen in Hochschulen (Aceto und Dondi, 2009). Einen Großteil des Lebens verbringen die Menschen in der Arbeitswelt, wodurch dies auch einen erheblichen Teil der Zeit darstellt, die für Weiterbildung verwendet wird. Es muss bei der Entwicklung der eLearning Inhalte bedacht werden, dass verschieden Personengruppen darauf zugreifen könnten. Eine möglichst flexible Kombination der Lehrinhalte wäre wünschenswert.

Der Marktwert von betrieblichen eLearning Programmen in Europa steigt schnell an. Schätzungen zufolge wird eLearning ein Viertel des Schulungsmarktes umfassen und bis zu 6 Mrd. Dollar ausmachen (Curran, 2004). In den USA stellt die Verwendung von eLearning im betrieblichen Umfeld bereits 60% des Marktwertes. Somit ist in Europa noch Entwicklungspotential festzustellen.

eLearning Tools

Wie oben erwähnt gibt es zur Zeit sehr viele Insellösungen für eLearning. In den letzten Jahren entwickelten sich auch Plattformen, die Publikation von eLearning Inhalten unterstützen. Diese Plattformen werden „Learning Management Systems (LMS)“ genannt und bieten einfache Funktionen zur Verwaltung von Online Kursen, Informationsaustausch zwischen Lehrern und Schülern und der Abnahme von Prüfungen. Kurse werden als Module entwickelt und in ein LMS eingefügt. Dabei haben sich mehrere

Standards für die Entwicklung von Modulen hervorgehoben, wo Kursmodule in den verschiedenen LMS wiederverwendet werden können. Folgend werden zwei dieser Paketierungsarten für Kurse kurz vorgestellt:

- SCORM (Sharable Content Object Reference Model): stellt einen Standard dar, der von der Advanced Distribute Learning Initiative entwickelt wird, welche durch das Verteidigungsministerium der USA initiiert wurde (ADL, 2009)
- IMS Content Packaging: entwickelt durch das IMS Global Learning Consortium, welches von der National Learning Infrastructure Initiative gegründet wurde (IMS, 2009)

Diese Standards definieren, wie die Kommunikation zwischen Kursen und dem LMS funktioniert. Dadurch sind Module bei unterschiedlichen LMS wieder verwendbar. Folgende exemplarische LMS unterstützen die oben definierten Standards:

- OLAT (Online Learning And Training): wird von der Universität Zürich entwickelt und unterstützt IMS und SCORM (OLAT, 2009)
- Moodle: Open Source LMS welches ebenfalls SCORM und IMS unterstützt (Moodle, 2009)
- ATutor: Webbasiertes LMS mit sehr guter Unterstützung für Personen mit Behinderung, unterstützt SCORM und IMS (ATutor, 20009)

3.1.2 eLearning 2.0 und Blended Learning

Das Schlagwort Web 2.0 ist in den letzten Jahren nahe zu jedem ein Begriff geworden. Es beschreibt eine Art von Webdesigns, welche die gemeinsame, interaktive Nutzung von Informationen, Interoperabilität, Benutzer gesteuerte Darstellung und Zusammenarbeit vereinfachen. Im Vergleich zu herkömmlichen Webseiten können Benutzer bei Web 2.0 aktiv in die Gestaltung und den Inhalt der Webseite eingreifen, wobei das Miteinander einen gravierenden Aspekt darstellt. Der große Erfolg von Web 2.0 liegt in der sozialen Interaktion der Benutzer. Waren es früher die großen Familienclans, wo Wissen in der Gruppe geteilt wurde, fehlt solch eine soziale Komponente heutzutage fast vollkommen. Diese Komponente finden Benutzer von Web 2.0 Webseiten in der Verwendung und Mitgestaltung von Inhalten im Web.

Dies ist auch der Ansatz von eLearning 2.0. Ein Beispiel stellt die Implementierung eines eLearning Kurses für das Institut für Sozialwissenschaften der Freien Universität Berlin dar, wo Studenten durch Fakultätsmitarbeiter eingeschult werden, damit sie daraufhin den Kursinhalt in Zusammenarbeit erstellen (Schiederig, 2007). Die Vorgaben sind hier nicht auf den Inhalt bezogen, sondern die Fakultätsmitarbeiter lassen den Studenten freie Hand und lernen aufgrund der Resultate von den Erfahrungen der Studenten. Dies zeigt deutlich, dass dem Benutzer hierbei nicht eine Zuschauer- sondern eine mitwirkende, aktive Rolle zukommt (Hamburg und Hall, 2008), wobei der Inhalt in Gruppen- oder Einzelarbeit erstellt werden kann.

„E-Learning 2.0 is conceived as an interlocking set of open-source applications. learning is becoming a creative activity and that the appropriate venue is a platform rather than an application." (Blees und Rittberger, 2009)

Das Hauptaugenmerk liegt hier in der Definition, dass eLearning 2.0 die Tätigkeit des Lernens als eine kreative Aktivität betrachtet. Daher kann Lehrmaterial nur eine Grundlage für das Lernen an sich sein. Erst durch den kreativen Input der Schüler entsteht Wissen. Web 2.0 fördert diese Art der Kreativität durch Tools, welche eine einfache Erstellung von Inhalten mit webbasierter Verteilung und Zusammenarbeit unterstützen. Hierunter fallen Wikis, soziale Netzwerke, Tools zum verwalten von Lesezeichen, Blogs, etc. (Hamburg und Hall, 2008). Folgende Tabelle zeigt die Unterschiede zwischen eLearning 1.0 und 2.0 (Jokisalo und Riu, 2009):

Tab. 1: From eLearning 1.0 to eLearning 2.0

(e-)Learning 1.0	***(e-)Learning 2.0***
Learning Platform & Learning Management Systems (LMS)	Personal Learning Environments (PLEs)
Acquisition processes	Participation processes
Multimedia (interactivity)	Social networks / Communities of Practice (CoP)
Externally provided content	User-created content
Curricula	Learning diaries / e-portfolios
Course structure	Communication
Tutor availability	Learner and peer interaction
Quality assessed through experts	Quality assessed through learners and peers

Die Tabelle führt einen neuen Begriff ein, das Personal Learning Environment, welches eine Weiterentwicklung der bekannten LMS um Funktionalitäten des Web 2.0 darstellt.

„A Personal Learning Environment is a facility for an individual to access, aggregate, configure and manipulate digital artefacts of their ongoing learning experiences." (Lubensky, 2006)

„The values that underlie the PLE and Web 2.0 are the same: the fostering of social networks and communities, the emphasis on creation rather than consumption, and the decentralisation of content and control." (Downes, 2007)

Der Grundgedanke hinter PLE ist jedoch nicht mehr ausschließlich das Lernen, sondern vielmehr eine Art Portal, auf welchem sich die Schüler treffen um sich auszutauschen, zu lernen und auch Lerninhalte zu kreieren.

„The 'pedagogy' behind the PLE – if it could be still called that – is that it offers a portal to the world, through which learners can explore and create, according to their own interests and directions, interacting at all times with their friends and community." (Downes, 2007)

Auf dieses Konzept setzt auch Storytelling auf, wo der Austausch von Wissen im Vordergrund liegt. Das Projekt HiStory sammelt Geschichten und persönliche Erfahrungen von Senioren in Blogs (EU, 2009). Ziel ist es, Senioren wieder sozial in die Gesellschaft einzubeziehen. Die Erfahrung wird gesammelt und geteilt, wodurch relevantes Wissen aufgebaut und den Senioren gleichzeitig ein wichtiger gesellschaftlicher Stellenwert gegeben wird.

„Storytelling is presumed as an important form of learning process." (Strahovnik und Mecava, 2009)

Mittels Web 2.0 Technologien werden Möglichkeiten geschaffen, damit Personen „Geschichten" erzählen. Dies bringt etwas von dem Ursprünglichen wieder, als Menschen zusammensaßen und Erfahrungen ausgetauscht wurden. Gefördert werden vor allem die Fähigkeiten, eigene Eindrücke in Worte zu fassen und den Geschichten anderer Personen zu folgen.

Blended Learning

Blended Learning definiert eine hybride Form des Unterrichts, wo zwei oder mehr Arten des Unterrichts miteinander kombiniert werden. Ein Beispiel wäre die Kombination einer Präsenzveranstaltung mit eLearning Tutorials.

„In addition, many universities are adopting a blended strategy, replacing some part of their contiguous teaching with online sessions. It is reported that the University of Central Florida now offers about 100 courses that meet half the time in classrooms and half online." (Curran, 2004)

Hier stellt die Einbeziehung von eLearning Material in den klassischen Unterricht zwar noch einen geringen Teil dar, der jedoch ständig wächst. An der Virginia Tech University haben bereits 2002 mehr als 1500 Studenten Online- und Präsenzkurse besucht (vgl. Curran, 2004). Diese Durchmischung von Fernunterricht und direkten Vorlesungen, sogenannten Sozialphasen, bekämpft den normalerweise einhergehenden Motivationsverlust bei den Studierenden (Seufert und Euler, 2004).

Die Kombination von Blended Learning und eLearning 2.0 kann die Motivation und den Lernerfolg der Studenten beträchtlich beinflussen. Hierfür werden Web 2.0 Technologien sowohl in die Präsenzphasen als auch in die Online Kurse eingebettet. Zum Beispiel kann ein Wiki in beiden Varianten das Verständnis erhöhen, da Studenten während des Vortrages Nichtverstandenes im Wiki nachlesen können. Dadurch kann ein Verständnis des Stoffgebietes erzielt werden, welches ohne Wiki beim Vortrag nicht möglich wäre. Einige Web 2.0 Technologien werden bereits erfolgreich eingesetzt. Darunter fallen auch Videomitschnitte bei der Ausbildung von Lehrkräften. Diese werden bei Übungsvorträgen oder –stunden durch die Lehrkraft gefilmt. Die Kollegen geben danach Feedback und das Video wird im Anschluss analysiert. Dies fördert das eigene Sicherheitsgefühl der Studenten und motiviert auch Feedback zu geben. Wie eine Online Umfrage ergeben hat, sehen 77% der Studenten die Videoaufnahmen als Vorteil an.

„According to the online survey after the course the majority of students (77%) strongly or mostly agree that working with video recordings from teaching practice helps them to understand TEFL topics and/or to make connections between various topics.“ (Kupetz und Ziegenmeyer, 2006)

Idealerweise führt der Einsatz von Blended Learning zu einer Verbesserung in der Informationsaufnahme und Qualität der Lehre. Durch direktes Feedback erfährt der Lehrende, welche Teile der eLearning Kurse zu verbessern sind. Die Effektivität wächst hierbei auf beiden Seiten. Es hängt vom Inhalt des Kurses und dem Niveau der Studierenden ab, ob ein Kurs als Präsenzveranstaltung geführt werden muss oder die Studenten sich den Lehrinhalt mittels eLearning selbst beibringen. Blended Learning und eLearning 2.0 bieten alle benötigten Möglichkeiten, um beide Varianten umzusetzen.

3.2 eLearning und Geoinformationssysteme

Geoinformationssysteme (GIS) stellen an sich bereits Informationstechnologie dar. Vor allem der Geografieunterricht lässt sich durch Einsatz von GIS bereichern, wobei der Einsatz vielfältiger Natur sein kann. Grund ist hier die immer weitere Verbreitung von Karten, die zudem von virtuellen Communities, die kollaborativ Inhalte kreieren, erstellt werden (Fischer, 2009). Diese Gruppierungen werden von Tools wie OpenStreetMap, Google Maps/Earth oder Microsoft Virtual Earth unterstützt. Durch die Verwendung von geografischen Tools lernen die Gruppenmitglieder die damit verbundenen Technologien spielerisch kennen.

Des Weiteren steigen die Angebote an GIS-eLearning Möglichkeiten in letzter Zeit stark an. Einige Unternehmen im Schulungsbereich bieten bereits Schulungen für GIS an, insbesondere vor dem Hintergrund der zwingenden Einführung von INSPIRE (Inspire, 2009) in den nächsten Jahren. Mehr als die Hälfte der teilnehmenden Institutionen der VESTA-GIS (VESTA-GIS, 2009) Umfrage gaben an, LMS wie WebCT/Blackboard, Moodle oder OLAT zu verwenden (Traun, 2009).

Im Folgenden werden einige Projekte im GIS Umfeld präsentiert, die didaktische Methoden in Verbindung mit eLearning verwenden.

Projekt eduGI

Das Projekt eduGI wird von der Europäischen Union gefördert und setzt sich zum Ziel, vorhandene eLearning Kurse zwischen acht europäischen GI-Instituten auszutauschen (Mäs und Reinhardt, 2007). Aufgabe der acht Institute war die Erstellung eines eLearning Kurses, der bei zwei Partnerinstituten abgehalten wird. Im Gegenzug wurden zwei unterschiedliche eLearning Kurse von zwei Partnerinstituten am jeweiligen Institut abgehalten. In Österreich nahm das Institut für Geoinformation und Kartographie der TU Wien teil. Durch die Art der eLearning Kurse waren die Studenten gezwungen, selbständig zu arbeiten, was durch die umfangreichen Materialien ermöglicht wurde. Zusätzlich zu den eLearning Materialien wurden sogenannte „synchrone Sitzungen“ abgehalten, bei denen Vortragende und Studenten gleichzeitig über die eLearning-Plattform verbunden waren.

Dies hatte den Zweck, Fragen zu beantworten und Problemthemen zu vertiefen (Mäs und Reinhardt, 2007). Die im Anschluss durchgeführte Kursevaluation zeigte ein durchwegs positives Feedback zu der Art und Weise der Vorträge.

Projekte geoinformation.net, gimolus und WEBGEO

Diese Projekte wurden alle im Rahmen des BMBF-Programms „Neue Medien“ gefördert und durch Hochschulinstitute gepflegt (Schiewe, 2006). Die Schulungsunterlagen stehen in Form von einzelnen Lernmodulen zur Verfügung und können teilweise (geoinformation.net) auch als Powerpoint-Folien heruntergeladen werden. Die Zielgruppe sind Studenten der Institute sowie externe Personen, auch wenn aufgrund des mangelnden Marketings letztere kaum die Lernmaterialien nutzen. Die Stärke des Projektes geoinformation.net liegt in der Möglichkeit, interaktive Animationen als Erklärung für komplexe Vorgänge heranzuziehen. Zum Beispiel zeigt eine Animation, wie eine Pufferung bei der räumlichen Abfrage funktioniert (Schwarz und Asche, 2006).

Projekt FerGI

Das Projekt Fernstudienmaterialien Geoinformatik (FerGI) wird von einem Konsortium von vier niedersächsischen Hochschulen betrieben (Schiewe, 2006). Ziel des Projektes ist die Implementierung, Nutzung und Evaluation von eLearning-Modulen zu Themen der Geoinformatik. Bei der Implementierung werden Lernplattform-unabhängige Module entwickelt, um den heterogenen Nutzeranforderungen besser zu genügen (Schiewe et al., 2006). Die entwickelten Module sind frei zugänglich und nutzbar, jedoch steht keine Betreuung der Studierenden zur Verfügung. Die Module wurden als Online-Module entwickelt, können aber auch heruntergeladen und Offline verwendet werden. Um Blended Learning zu forcieren, werden auch Testaufgaben und Kommunikationsmöglichkeiten durch Foren geboten. Daher eignen sich die Module sowohl zum Selbststudium als auch begleitend zum klassischen Unterricht.

Projekt geo-kiosk

Die Projekte geoinformation.net, WEBGEO, FerGI und gimolus wurden durch sogenannte öffentliche Anschubförderung umgesetzt. Nach Ablauf oder Ausklingen dieser Förderungen sahen die Projekte einer unsicheren Zukunft entgegen. Um ein langsames Auflösen der unterstützenden Konsortien und ein Veralten der Lernmaterialien zu verhindern, hat sich im Rahmen des BMBF-Programms „E-Learning-Dienste für die Wissenschaft“ ein Konsortium aus diesen vier Projekten gebildet. Dieses Konsortium läuft unter dem Namen „geo-kiosk“ und bietet die Lernmodule der vier Projekte sowie weitere neue Materialien an (Schiewe, 2006). Als Ausbauschritt sollen qualitätssichernde Maßnahmen zur Steigerung der Qualität dienen, um die Lehrinhalte auch international an Kunden verkaufen zu können. Auf diese Weise soll eine stabile Finanzierung der Projekte garantiert werden.

Projekt GITTA

Das Projekt GITTA (Geographic Information Technology Training Alliance) wurde im Rahmen der „Swiss Virtual Campus“ (SVC) Initiative finanziert und wird von zehn Instituten der schweizerischen Hochschulen betreut (Fisler et al., 2006). Ziele des Projektes waren das verstreute GIS-Wissen in einen zu schaffenden GIS-Ausbildungspool einzubringen, um den Frontalunterricht durch eLearning und Blended Learning Einheiten zu ersetzen (Niederhuber et al., 2005). Nach erfolgreichem Abschluss des Projekte wurde eine öffentliche Freigabe der Inhalte beschlossen, welche in ein Open Source Projekt des GITTA-Konsortium einflossen. Das Konsortium stellt gleichzeitig eine Community dar, die sich um die Weiterpflege der Inhalte kümmert. Externe Partner sind willkommen, müssen sich aber beim Konsortium anmelden. Dieses prüft, ob die neuen Autoren über ausreichend Wissen verfügen, um die Lektionen pflegen und erweitern zu können. Die Technologie hinter GITTA wurde als Open Source Projekt „eLesson Markup Language“ (eLML) veröffentlicht und wird in zahlreichen Projekten als Grundlage verwendet. eLML hat seitdem einige Preise gewonnen, unter anderem 2008 den mediendidaktischen Hochschulpreis Medida Prix 2008 (Niederhuber et al., 2009).

Projekt UNIGIS

UNIGIS ist eine der ersten GIS distance learning Initiativen und bietet seit 1993 international anerkannte, akademische Fernstudiengänge mit Qualifikationen in GIScience und GISystems in Form von Hochschulabschlüssen (MSc(GIS)) (Car, 2008). Das UNIGIS Konsortium besteht aus mehr als einem Dutzend Partneruniversitäten, dem auch die Universität Salzburg als eines der Gründungsmitglieder angehört. 2001 wurde das Lehrangebot auf eLearning umgestellt, wobei Blackboard als Grundlage für die Distribution von Inhalten und der Kommunikation zwischen den Studierenden und den Lehrenden dient (Traun und Fally, 2007). Zusätzlich sind Projekte Teil der Ausbildung von UNIGIS. Diese werden von den Studierenden selbständig durchgeführt, wobei die Gruppenzusammenstellung durch die Studierenden zu erfolgen hat. Die Kommunikation innerhalb der Gruppe erfolgt wiederum über die Lernplattform (Traun und Fally, 2007). Dadurch soll kollaboratives Lernen zwischen den Studierenden gefördert werden. Ob synchrone oder asynchrone Kommunikationsmedien der Plattform verwendet werden, ist ebenfalls den Studenten überlassen. Es hat sich mit der Zeit gezeigt, dass synchrone Kommunikation bei Problemstellungen, Fragen oder Teambuilding bevorzugt werden (Traun und Fally, 2007).

Projekt GeoloGIS

Dieses Projekt zielt auf Schüler der Oberstufe in Belgien ab. Es soll eine Art erweiterte Realität geboten werden, wo Schüler zu den geologischen Landschaften in Belgien zusätzliche Information von einem eLearning Tool erhalten (Bral et al., 2009). Geplant sind Videos, geologische Karten und 3D-Visualisierungen. Das Projekt befindet sich noch in einem frühen Stadium.

Projekt Geoland.at

Geoland.at ist ein Projekt der österreichischen Bundesländer und wurde durch die österreichischen Landeshauptleute initiiert. „Das Ziel von Geoland ist, einen offenen, auf internationalen Standards basierenden, freien und österreichweiten Zugriff auf Geodaten und Services der Bundesländer zu ermöglichen." (Schieriau, 2007) Des Weiteren werden wesentliche Teile der INSPIRE (Infrastructure for Spatial Information in Europe) Richtlinie der EU mit geoland.at umgesetzt, wobei Ziel der Richtlinie ist, das Angebot von Geodaten der europäischen Gemeinschaft den einzelnen Mitgliedern und der Öffentlichkeit zur Verfügung zu stellen. Die in geoland.at bereitgestellten Daten können mittels WMS abgefragt und frei verwendet werden. Daher eignen sich diese Daten sehr gut für die Verwendung im Unterricht. Zusätzlich wird ein eigener Geodaten-Viewer angeboten, damit die Daten auch visualisiert werden können.

mLearning Projekte

Einige Projekte beschäftigen sich mit dem Sammeln von GPS Daten. Schüler werden mit Handhelds ausgestattet, die GPS Koordinaten aufnehmen und aufzeichnen können. Diese Daten werden dann im Anschluss in GIS-Systeme geladen. Das Projekt IMST einer Wiener Hauptschulklasse (Haller, 2008) zeigt, wie zehnjährigen Schülern der Umgang mit PDAs und GPS beigebracht werden kann, indem sie ein Problem mittels der zur Verfügung gestellten PDAs lösen. Auf diese Weise kann ein doch eher trockenes Stoffgebiet der Geografie interessant und motivierend gelehrt werden. Zudem fördert dies den fächerübergreifenden Unterricht, der in Österreich in den Lehrplänen verpflichtend vorgeschrieben ist (Zumbach und Bachleitner, 2007).

Projekt eScenario

Unter Zusammenarbeit der ETH Zürich, EPF Lausanne und der Universität von Bern entstand eine Lernplattform, die das Management von natürlichen Gefahren unterstützt. Ziel der Plattform ist die Unterstützung des Frontalunterrichtes durch eLearning Module. Grundlage der Plattform bildet NAHRIS, ein Vorgängerprojekt des SVC bezüglich Gefahren in der Natur (Kos et al., 2008). Die Module sind in fünf Phasen aufgebaut, wobei sich Frontalunterricht mit gemeinschaftlichen und eLearning Inhalten abwechseln. Erste Erfahrungen zeigen eine hohe Akzeptanz, wobei 30% der Studierenden den Kurs wegen des hohen Arbeitspensums abbrechen.

3.3 eLesson Markup Language (eLML)

Dieses Subkapitel stellt die technologische Basis des eLearning Tutorials vor. Es werden historische Zusammenhänge mit dem Projekt GITTA erwähnt, sowie das didaktische Fundament hinter dem eLML Schema.

3.3.1 Historischer Zusammenhang mit Projekt GITTA

eLML (eLML, 2009) stellt die Technologie hinter dem Projekt GITTA dar. Es wurde nach Beendigung der Projektphase von GITTA als eigenständiges Projekt eingerichtet, wobei der Open Source Ansatz gewählt wurde. Eine eigene Homepage (eLML, 2009) wurde für die weitere Pflege der Technologie eingerichtet und externe Partner angeworben. Seitdem verwenden einige Projekte eLML als technologische Grundlage, einige davon sind exemplarisch auf der eLML-Homepage angeführt.

Hinter eLML steckt ein XML-Schema, welches zur Validierung der Module verwendet wird. Die Module stellen XML-Dokumente dar, die mittels XSLT (XML Style Sheet Transformation) in ein Ausgabeformat gebracht werden. Dadurch, dass die Module keinerlei Darstellungsinformationen enthalten, können aus dem gleichen Ursprungsdokument viele Ausgaben erzeugt werden. Hierunter fallen beispielhaft HTML, PDF oder OpenDocument. Genauso ist es möglich, mit eLML erzeugte Lektionen als IMS oder SCORM Pakete zu generieren, das die Verwendung in den meisten LMS ermöglicht. Diese Flexibilität ist der Grund, warum das in dieser Arbeit erstellte eLearning Tutorial mittels eLML erstellt wird.

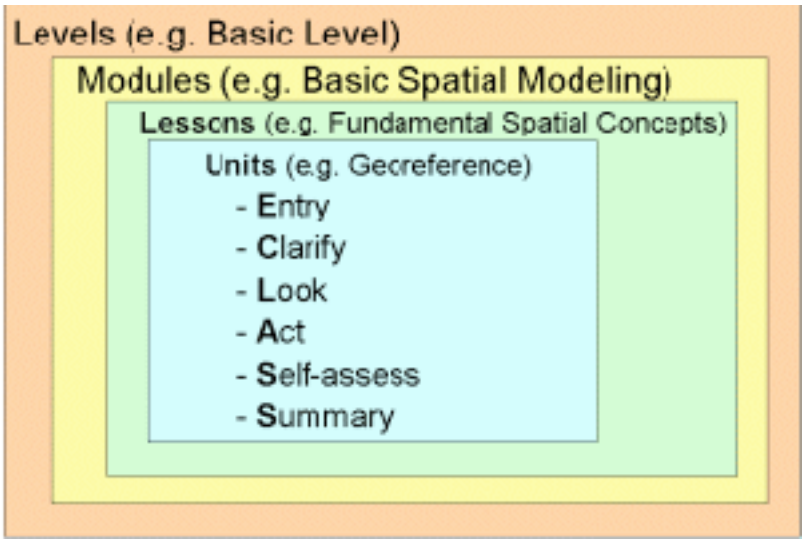
3.3.2 Standardisierter Aufbau und didaktische Grundlagen einer eLesson (E-CLASS)

Die didaktischen Richtlinien in eLML wurden aus einem Modell namens ECLASS (Gerson, 2000) entwickelt und angepasst. Die Abkürzung ECLASS steht in diesem Zusammenhang für (Fisler et al., 2006):

- Entry: steht für einen Einführungsteil der Lektion
- Clarify: spiegelt einen Theorieblock wieder
- Look: dient dazu, wiederholende Beispiele zu erstellen
- Act: beinhaltet praktische Aufgaben, um den gelehrten Inhalt zu vertiefen
- Self-assessment: Übungen oder Gruppenarbeiten zum Ende der Lektionen
- Summary: Zusammenfassung am Ende einer Lektion

Jede dieser Lektionen ist Bestandteil eine Lerneinheit. Die Menge der Lerneinheiten bilden zusammen eine eLesson, also einen Kurs. Mittels eLML können auch mehrere zusammengehörige Kurse erstellt werden. Dies erfolgt bei dem eLearning Tutorial dieser Arbeit, welches in vier Kurse aufgeteilt ist. Folgende Abbildung zeigt einen Überblick über die Struktur der Module in GITTA und soll ein Gefühl für die hinter eLML stehende Struktur von eLML bieten (eLML, 2009).

Abb. 6: Aufbau eines Tutorials

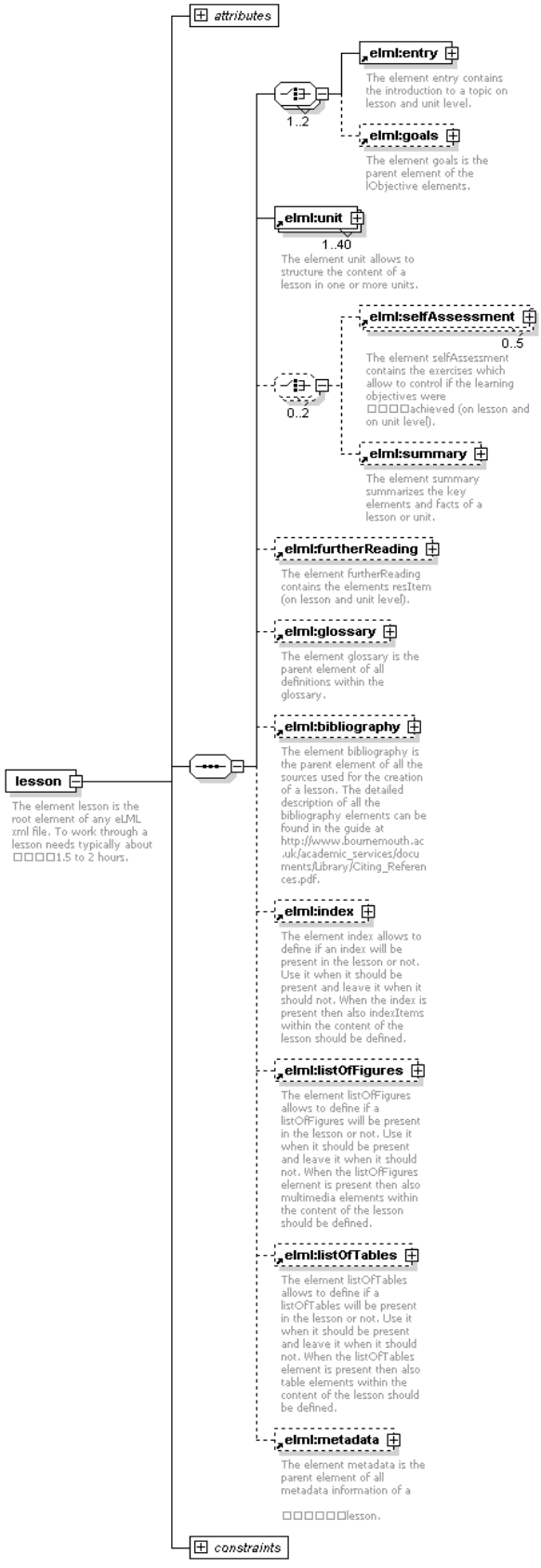


3.3.3 XML-Schemabeschreibung

Wie im vorherigen Subkapitel erwähnt, ist eLML ausschließlich ein XML-Schema. Dieses definiert den Aufbau von Lerneinheiten, Modulen und Lektionen. Folgend soll das Schema etwas genauer verdeutlicht werden.

Die oberste Einheit ist ein Modul, das einem Kurs gleichgestellt ist. Jedes Modul besteht aus lessons, also Lerneinheiten. Diese müssen nachstehende Struktur aufweisen:

Abb. 7: Aufbau des Elementes lesson



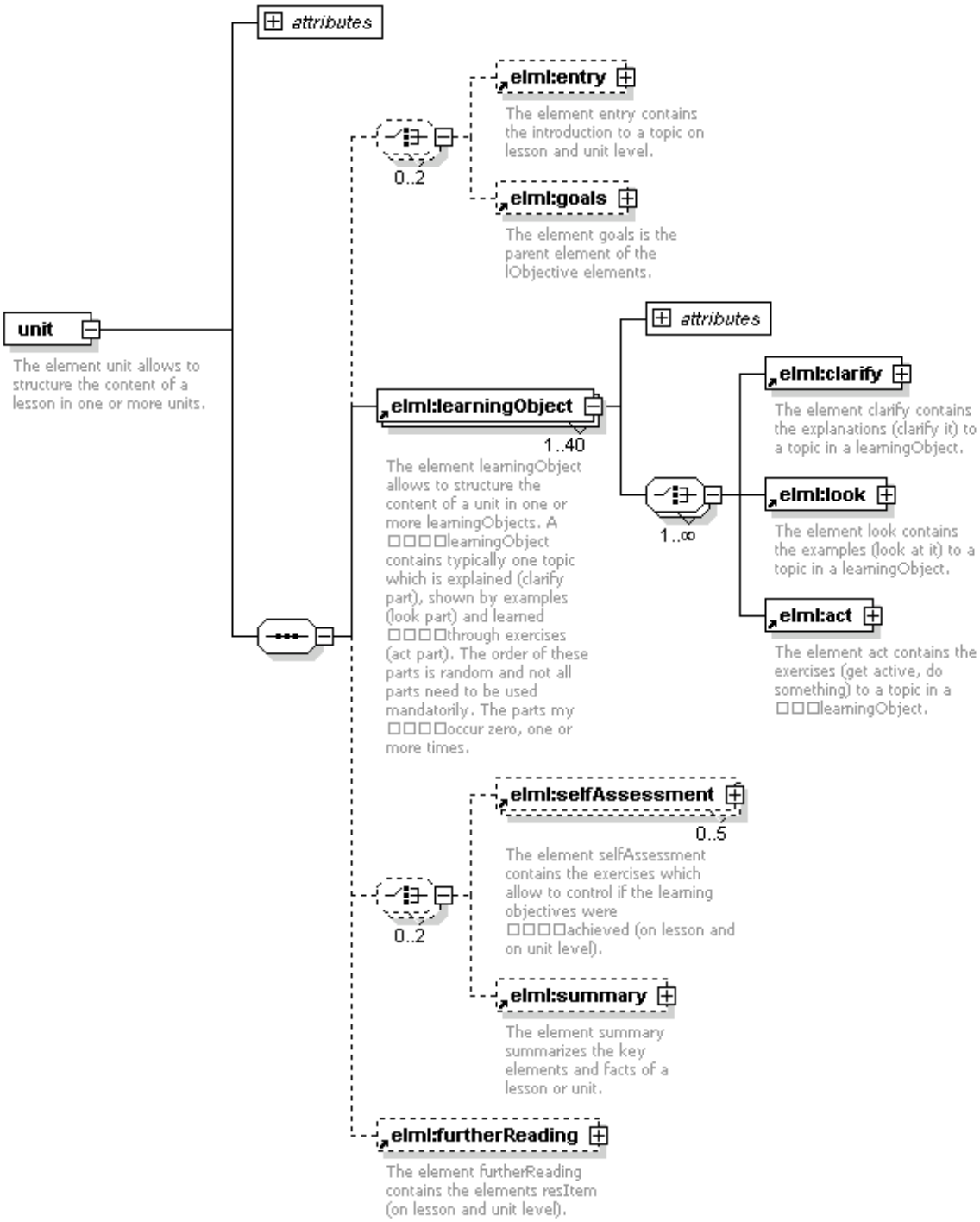
Das Element Entry stellt die Einleitung zur Lerneinheit dar und kann optional von Zielen (Element goals) gefolgt sein. Der Hauptteil einer Lesson stellen die Units dar. Eine Unit ist eine Lektion und beinhaltet das Stoffgebiet des Kurses. Der genaue Aufbau wird im Anschluss erklärt.

Nach den Units folgen einige optionale Elemente (Fisler et al., 2006):

- selfAssessment: kann Übungen enthalten
- summary: Zusammenfassung der Lerneinheit
- furtherReading: Verweise auf die Bibliografie, welche weitere interessante Inhalte zu dieser Lerneinheit enthält
- glossary: Glossar
- bibliography: Liste der Referenzen
- metadata: Metadaten zu der Lerneinheit, wie Sprache, Verantwortlichkeit, etc.

Folgende Abbildung zeigt den Aufbau von Units.

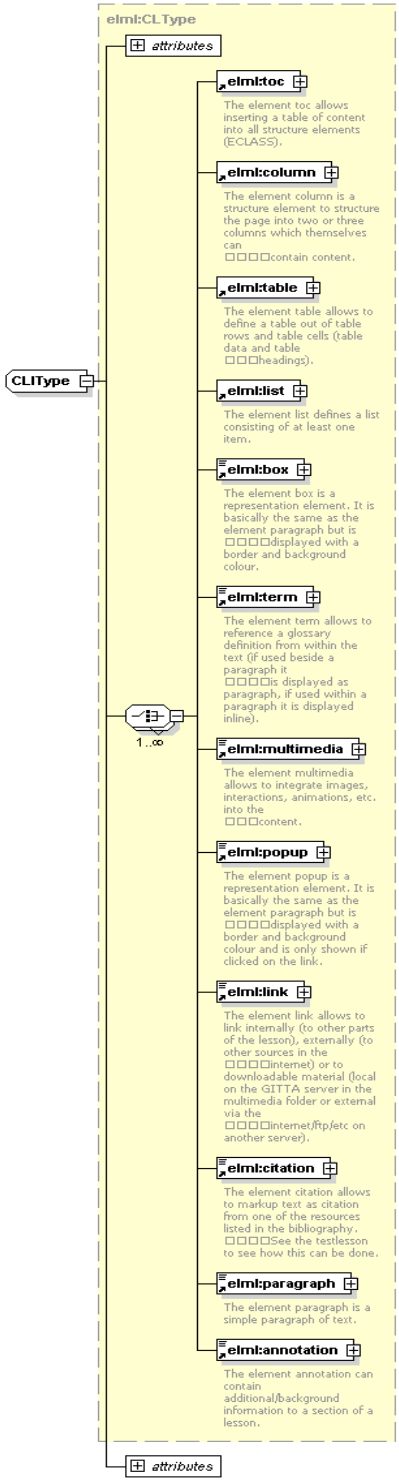
Abb. 8: Schema des Elementes unit



Jede unit besteht aus einer Anzahl von Lernobjekten (learningObject). Diese enthalten den Kern des Stoffgebietes in Form von Theorie (clarify), Beispielen (look) und Übungen (act) (Fisler und Schneider, 2008). Zusätzlich können Lektionen auch die optionalen Felder entry, goals, selfAssessment, summary und furtherReading enthalten, die den gleichen Inhalt haben, wie auf lesson-Ebene.

Der eigentliche Inhalt einer Lektion wird in den Elementen clarify und look abgelegt. Diese enthalten den formatierten Text, Tabellen oder multimediale Inhalte, die durch XML-Tags erzeugt werden. Folgende Abbildung zeigt die Liste der möglichen Elemente. Eine genaue Beschreibung mit Beispielen findet sich auf der eLML Homepage (eLML, 2009).

Abb. 9: Schema der Content-Elemente



4 Aufbau und Inhalt der eLessons

In Kapitel 3.3 wurde erklärt, wie in eLML eLessons aufgebaut werden. Eine eLesson stellt in etwa einen Kurs dar. Das eLearning Tutorial dieser Arbeit wird mittels eLML erstellt und als PDF generiert. Das Ergebnis ist im Anhang ersichtlich.

Wie in Kapitel 3.1.1 erläutert, ist die Qualität eines Tutorials maßgeblich vom pädagogischen Mehrwert abhängig, welcher durch folgende Implementierungsfelder beeinflusst werden kann:

- Gestaltung der verwendeten Medien
- Aufbau des Unterrichtes in Form von Lernphasen
- Gesamtdidaktisches Design des Kurses

Die Berücksichtigung aller dieser Implementierungsfelder wird durch eLML erleichtert, wodurch die Fokussierung auf den Inhalt des Tutorials erfolgen kann. Des Weiteren schränkt eLML den Entwickler eines eLearning Modules nicht auf bestimmte pädagogische Methoden ein, sondern bietet durch die Vielzahl an Ausgabeformaten Möglichkeiten, verschiedenste Methoden einzusetzen. Zusätzlich erhielt das Projekt Gitta für die Implementierung und den Aufbau von eLML einige Auszeichnungen. Dies führte zu der Entscheidung, eLML als Grundlage für die Entwicklung des eLearning Tutorials einzusetzen.

Im Folgenden sollen die oben erwähnten Implementierungsfelder in Bezug auf eLML betrachtet werden.

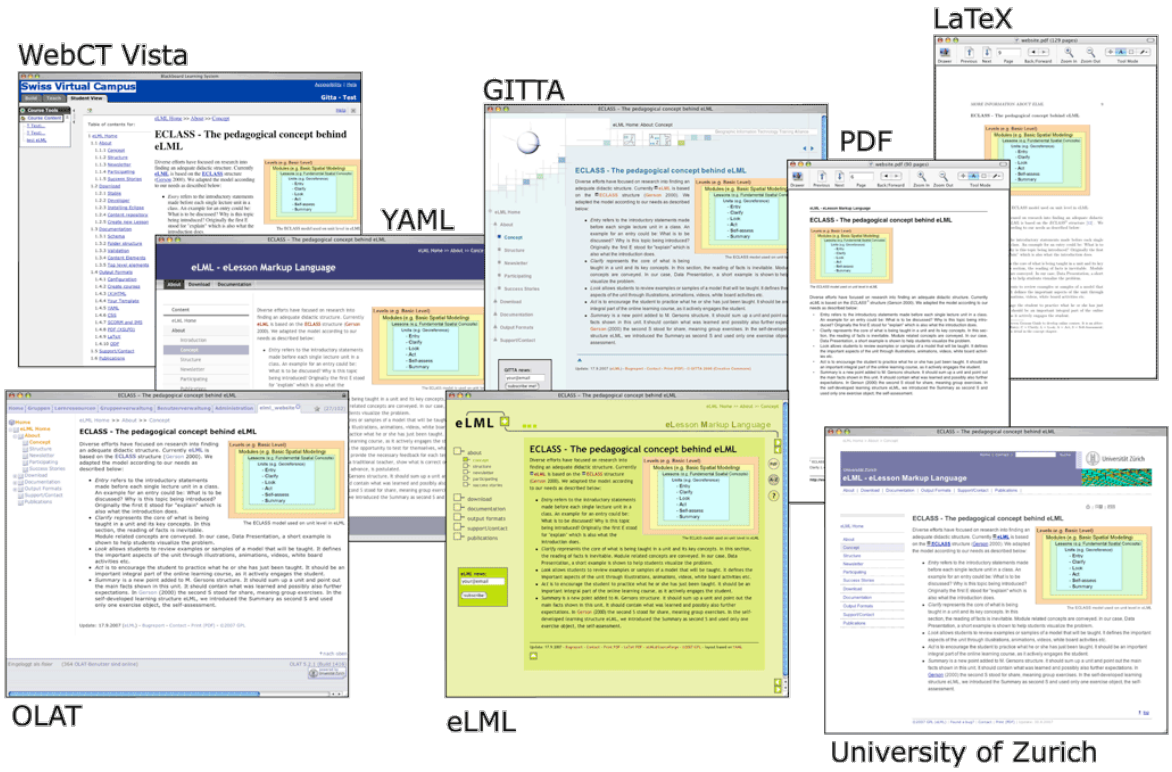
4.1 Gestaltung der verwendeten Medien

Durch den Einsatz von eLML stehen mehrere Ausgabeformate zur Auswahl:

- XHTML
- PDF
- LaTex
- Open Document Format
- DocBook
- LMS Modul (SCORM und IMS Content Package)

Folgende Abbildung zeigt einen Überblick über die möglichen Ausgabeformate als Grafik:

Abb. 10: Ausgabemöglichkeiten durch eLML



Die Gestaltung des Tutorials ist dabei weitgehend unabhängig vom gewählten Format. Dabei kann der Inhalt der Lerneinheiten unter anderem folgende Elemente enthalten:

- Text
- Tabellen
- Bilder
- Querverweise
- Glossar
- Bibliografien
- Videos
- Popups
- Boxen
- Begriffsdefinitionen
- Formatierungen
- Listen
- Links
- Java Applets
- etc.

Natürlich kann nicht jedes Outputformat alle diese Elemente darstellen. Videos können zum Beispiel in keinen dokumentenbasierten Ausgabeformaten angezeigt werden, sondern nur bei XHTML oder LMS-Modulen. Die Entwicklung kann dies bei der Erstellung des eLearning Tutorials bedenken, indem das Element ausschließlich bei XHTML zu verwenden ist. eLML prüft diese Einstellung je Element und ignoriert bei PDF die

Ausgabe. Sollte trotzdem ein Element verwendet werden, welches in dem gewünschten Ausgabeformat nicht dargestellt werden kann, erscheint stattdessen eine Grafik mit dem Hinweis, das dieser Content ausschließlich in den Onlineversionen ersichtlich ist. Dies erleichtert wiederum die Entwicklung, da alle Ausgabeformate gleich behandelt werden können.

Natürlich kann ein Tutorial auch auf ein bestimmtes Ausgabeformat hin abzielen, wie dies in dem hier entstandenen Tutorial der Fall ist. Es wurde die Darstellung auf die Ausgabe als PDF-File optimiert, was jedoch die Generierung als XHTML oder LMS-Modul nicht behindert. Das Tutorial schöpft lediglich die Möglichkeiten der beiden Ausgabeformate nicht aus.

Ein weiterer Punkt bei der Gestaltung der verwendeten Medien stellt das Design des Ergebnisses dar. Meistens werden Tutorials durch Unternehmen erstellt, deren Corporate Identity auch durch das Tutorial verwendet werden muss. eLML bietet hier die Möglichkeit, die Darstellung den eigenen Wünschen mittels Templates anzupassen. Die vorhandenen Templates erstellen aus dem XML-Dokument des Kurses eine Homepage im Design der eLML- oder der GITTA-Homepage. Diese Generierung soll nur als Grundlage für die eigenen Entwicklungen dienen.

Die Entwicklung eines Templates kann die Darstellung komplett verändern, wie das Moodle Tutorial des Institutes für Fernstudien- und eLearningforschung (IFel, 2009) zeigt.

4.2 Aufbau des Unterrichtes in Form von Lernphasen

Um den Unterricht interessant zu gestalten, ist Abwechslung in den Lehrformen nötig. Dies kann durch einen Mix aus Präsenzveranstaltungen und Selbststudium erfolgen. Das hier erstellte Tutorial ist darauf ausgelegt, sowohl als Skriptum für eine Präsenzveranstaltung als auch als Grundlage für das Selbststudium zu dienen. Durch eine Kombination der beiden ist auch der Einsatz des Tutorials in Form von Blended Learning möglich.

eLML fördert die Unterstützung der unterschiedlichen Lehrformen durch die Ausgabeformate und die verschiedenen unterstützten Inhalte. Die Generierung als PDF-Dokument kann dem Selbststudium und als Skriptum dienen. Wenn als Ausgabeformat XHTML gewählt wird, können multimediale Inhalte eine Präsenzveranstaltung bereichern.

Durch den Aufbau der XML-Struktur, fördert eLML die Verwendung von Teilen einer Lehreinheit. Lehreinheiten bestehen aus sogenannten learningObjects, die ein Stoffgebiet umfassen, welches in etwa 10 Minuten erfolgreich gelernt werden kann. Beim Ausgabeformat XHTML wird jedes dieser learningObjects als eigene HTML-Seite generiert. Daher können einzelne learningObjects in der Präsenzveranstaltung behandelt und andere als Selbststudium gelernt werden. Die genaue Teilung zwischen den beiden Arten fällt durch den Aufbau von eLML leicht und bildet eine gute Grundlage für den Einsatz von Blended Learning.

Der Unterricht kann hierbei durch mehrere Methoden bereichert werden. Folgend sollen einige Methoden angeführt werden, die im Zusammenhang mit dem Tutorial als sinnvoll erscheinen:

- Teile des Tutorials werden als Aufgaben vergeben. Beim nächsten Termin wird einer der Studenten aufgefordert, eine Zusammenfassung zu geben. Zusätzlich kann dies auch dazu verwendet werden, ein Feedback für den Inhalt zu erlangen.
- Behandeln der learningObjects in Form von Workshops. Die Studenten arbeiten direkt am PC am eigenen Beispiel des Web-Clients.
- Zu Beginn werden die Ziele der Lehreinheit besprochen und am Ende durch ein eLML-unterstütztes Selfassessment kontrolliert. Im Anschluss werden Probleme und Verständnisschwierigkeiten diskutiert und erklärt. Dies dient ebenfalls dem Tutorialentwickler als Feedback, welche Teile des Tutorials unverständlich formuliert sind und einer Adaption bedürfen.
- Um den Lernerfolg zu fördern, bietet jede Lehreinheit eine Zusammenfassung, die kurz alle wichtigen Punkte der Lehreinheit anschneidet. Sollten hier noch Bereiche bestehen, die nicht verstanden wurden, kann das jeweilige Lernobjekt wiederholt werden.
- Sinnvoll wäre der Einsatz von Blitzlicht, um das Gefühl der Studenten und den Lernerfolg festzuhalten. Gerade bei komplexen Themen, wie bei diesem Tutorial, ist es leicht möglich, dass Studenten den Überblick über das Stoffgebiet verlieren. Blitzlicht zeigt dies ziemlich schnell auf, worauf der Lehrende reagieren kann. Andernfalls würde ein Defizit erst bei etwaigen Prüfungen oder Projekten auffallen.
- Im Zuge des Workshops sind Lehrphasen einzuplanen, die Freiarbeit bieten. Die Studenten sollen hierbei selbständig den definierten Bereich des Web-Clients verlassen und dadurch neue Funktionalitäten ausprobieren. Ziel ist hierbei das spielerische Entdecken der Technologie, ohne einen genau vorgegebenen Weg zu haben. Im Anschluss präsentieren die Studenten, was sie im Zuge der Freiarbeit geschaffen haben. Dadurch kann der Tutor ein Feedback bekommen, auf welche Weise die Beispiele im Tutorial verändert werden können, um diese interessanter zu gestalten. Die nachhaltige Verbesserung des Tutorials sollte Ziel des Tutors sein.
- Teile des Tutorials können komplett als Aufgaben des Selbststudiums im Sinne des Problem Based Learnings vergeben werden. Zuvor wird mit den Studenten gemeinsam eine Aufgabenstellung erarbeitet, die dann durch die Studenten gelöst wird. Das nötige Wissen wird dabei durch die Studenten eigenständig erworben. Dies führt zu unterschiedlichen und kreativen Lösungen, die in den Änderungszyklus des Tutorials einfließen können.

Diese Methoden sollten durch den Tutor angewendet werden, um den Lernerfolg der Studenten zu steigern und gleichzeitig die Qualität des Tutorials zu erhöhen. Das Hauptaugenmerk muss hierbei auf die praktischen Komponenten der Schulung gelegt werden, da das Thema sehr technisch und komplex ist.

4.3 Gesamtdidaktisches Design des Kurses

Zielgruppe des Tutorials sind Personen mit unterschiedlichen technischem Grundwissen. Daher muss es möglich sein, an beliebigen Stellen mit dem Tutorial zu beginnen. Dies wurde bei der Entwicklung in Betracht gezogen, indem das Tutorial in vier Kurse eingeteilt wurde:

1) Kurs 1: Vorbedingungen für den Zugriff auf die Web Services und die Implementierung
2) Kurs 2: Zugriff auf ein WMS und Darstellung einer Karte
3) Kurs 3: Zugriff auf ein WFS
4) Kurs 4: Durchführung einer Suche mittels WFS und Darstellung des Suchergebnisses innerhalb des Kartenfensters

Die einzelnen Kurse bauen aufeinander auf, können jedoch auch einzeln belegt werden. Personen, die bereits bestimmte Vorkenntnisse haben, können direkt bei einem der aufbauenden Kurse einsteigen. Hierfür muss nur festgestellt werden, ob die Vorraussetzungen der einzelnen Kurse erfüllt sind. Jeder Kurs ist wie folgt aufgebaut:

- Einführung: hier wird kurz erklärt, was Inhalt des Kurses ist. Gleichzeitig wird ein Ausblick auf das Ergebnis des Kurses geboten
- Ziele: eine Liste der Ziele bietet die Möglichkeit, die Erwartungen an den Lernerfolg abzuschätzen
- Lerneinheiten: diese bilden den Hauptteil eines Kurses und werden nachstehend genauer beschrieben
- Zusammenfassung: abschließend wird eine Zusammenfassung über den Inhalt der einzelnen Lerneinheiten gegeben

Die Kombination aus Einführung, Zielen und der Zusammenfassung bietet einem Studenten die Möglichkeit, das eigene Wissen gegen den Inhalt des Kurses abzugleichen. Wenn nötig, können einzelne Lerneinheiten oder der gesamte Kurs absolviert werden. Ob eine Lerneinheit für den Studenten sinnvoll ist, kann dieser wiederum aus dem Inhalt ablesen. Jede Lerneinheit im Tutorial ist wie folgt aufgebaut:

- Einführung: kurze Einführung, die den Inhalt der Lerneinheit skizziert
- Ziele: Liste der Ziele dieser Lerneinheit. Diese Ziele spiegeln sich als Teil der Kursziele wider. Daher kann schnell ein Konnex zwischen den nicht erfüllten Zielen des Kurses und der dazugehörigen Lerneinheit geschaffen werden
- Lernobjekte: eine Liste von Lernobjekten, welche jeweils einen Teilbereich der Lerneinheit abdecken. Diese entsprechen vom Umfang dem Lernpensum von 10 Minuten.
- Zusammenfassung: jede Lerneinheit enthält eine eigene Zusammenfassung über den darin vermittelten Stoff. Die Zusammenfassung ist wiederum Teil der Zusammenfassung des Kurses und bietet genauso wie die Ziele einen Konnex zum Teilgebiet des Kurses.

Jede Lerneinheit besteht aus Lernobjekten, welche den eigentlichen Inhalt des Tutorials darstellen. Diese enthalten Texte, Beispiele und auch Komplettlösungen für die Aufgaben. Auf diese Weise wir dem Studenten zumindest eine funktionierende Lösung geboten, die vor allem im Zuge der Freiarbeit als Grundlage für eigene Ideen dienen soll. Die Studenten sollen angehalten werden, die Lösung nicht zu kopieren, sondern eigene kreative Lösungen zu finden.

Das Tutorial wurde auf die Ausgabe mittels PDF abgestimmt, wodurch einige Abstriche in der Form gemacht werden mussten. Eines ist die Darstellung von Quellcode als Grafiken. eLML kann Quellcode inklusive Syntaxhighlighting nur bei der Ausgabeform XHTML darstellen. Sobald das Ausgabeformat PDF gewählt wird, erscheint stattdessen nur ein Platzhalter, der die Nichtdarstellbarkeit signalisiert. Über den Umweg der Darstellung mittels Grafiken ist auch der Quellcode im PDF sichtbar. Dafür ist es im Zuge einer Schulung nötig, neben dem Tutorial den Studenten auch den Quellcode zur Verfügung zu stellen.

Im Anschluss an die Kurse enthält das Tutorial ein Glossar mit den wichtigsten Begriffen des Inhaltes. Dadurch wird dem Studenten eine schnelle Informationsquelle im Falle von Fragen geboten.

Das vollständige Tutorial kann dem Anhang entnommen werden.

5 Zusammenfassung und Ausblick

Das Hauptziel des eLearning Tutorials ist die Anleitung, wie eine Webapplikation mit einem Kartenfenster implementiert werden kann. Diese Webapplikation bietet die Möglichkeit, nach Grundstücken zu suchen und das Ergebnis im Kartenfenster darzustellen.

Hinter dieser einfach klingenden Aufgabenstellung verbergen sich etliche Technologien, die für die Realisierung nötig sind. Angefangen bei der Authentisierung und Generierung eines Security-Tokens bis hin zum manuellen Aufbau des WFS-Requests für die Grundstücksuche. Das Tutorial beschränkt sich nicht auf die Teile, die der tatsächlichen Darstellung des Kartenfensters dienen, sondern versucht die Gesamtheit aller nötigen Schritte zu erklären. Die Probleme, die aufgrund fehlender oder unvollständiger Software auftreten, werden ausgeführt und Lösungen geboten. Letztendlich soll der Lernende eine funktionsfähige Applikation erstellen.

Im Zuge der Implementierung des Web-Clients gab es Probleme mit dem aktuellen Zustand der JavaScript Bibliothek OpenLayers. Offiziell unterstützt OpenLayers den Zugriff auf WFS, jedoch ist dieser nur auf nicht einschränkendes Lesen von Objekten beschränkt. Um eine Suche implementieren zu können, muss die Abfragesprache Filter Encoding (OGC, 2005a) direkt im JavaScript manuell zusammengestellt werden.

Ein weiteres Problem stellt der Zugriff mittels JavaScript auf das WFS allgemein dar. Sobald JavaScript auf eine URL zugreift, prüft der Browser, ob die Domain der URL die gleiche ist, von der auch die Webseite selber geladen wurde. Diese Sicherheitseinstellung wird Sandboxkonzept genannt und soll verhindern, dass manipulierte Seiten schadhaften Quellcode von fremden Webseiten nachladen. Im Falle des Web-Client ist dies jedoch gewünscht, da der WFS-Server nicht gleich dem HTTP-Web-Server ist. Hierfür ist es nötig, eine Weiterleitung am HTTP-Web-Server umzusetzen. Das Tutorial zeigt auch, wie dies geschehen kann. Das genaue Vorgehen bei diesem und den anderen Problemen kann im Tutorial nachgelesen werden.

Das Tutorial selbst wurde mittels eLML erstellt. eLML wurde im Zuge des Projektes GITTA der ETH Zürich, gefördert durch den Swiss Virtual Campus, entwickelt und nach Ende des Projektes als eigenständiges Projekt herausgelöst. Seitdem ist eLML als Open Source Projekt verfügbar und wird von engagierten Institutsmitarbeitern der ETH Zürich und vielen Freiwilligen gepflegt. In eLML besteht ein Tutorial aus eLessons, welche wiederum aus Units (Lehreinheiten) aufgebaut sind. Zusammengestellt werden die Inhalte mittels XML, wobei durch eLML ein eigenes XML-Schema definiert wurde.

Der Vorteil der Umsetzung mittels XML liegt in der Darstellungsunabhängigkeit. Zum Zeitpunkt der Erstellung des Inhaltes ist es völlig egal, in welcher Form das Ergebnis angeboten werden soll. Mit Ausnahme von multimedialen Inhalten, wie Videos oder Audiodateien, kann ein eLML Modul auch in PDF generiert werden. Wenn als Ausgabeformat HTML gewählt wird, bietet eLML zusätzlich die Möglichkeit, eigene

Templates anzugeben. Dadurch kann ein eLearning Tutorial in das Corporate Design des Unternehmens/Institutes eingebunden werden.

Die Vorgangsweise, wie ein neues Tutorial angelegt wird, beschreibt die eLML Homepage (eLML, 2009) nahezu genau. Die Generierung von PDF-Dateien ist etwas komplizierter, doch mit etwas Geduld ist auch dies leicht möglich. Die Beschreibung auf der Homepage könnte in diesem Bereich allerdings etwas genauer sein.

Zusammenfassend kann gesagt werden, dass sich eLML sehr gut für die Umsetzung eines eLearning Tutorials eignet und noch dazu frei verfügbar ist.

Das hier erstellte Tutorial wurde nur mit den von eLML zur Verfügung gestellten Templates generiert. Ein Ansatz für Verbesserung wäre die Entwicklung eines eigenen Templates, um die Darstellung des Ergebnisses genauer zu kontrollieren. Dementsprechend sieht das in HTML generierte Tutorial wie die eLML Homepage oder wie eine GITTA-Lektion aus. Das genaue Vorgehen bei der Implementierung eines Templates wird ebenfalls auf der eLML Homepage beschrieben.

Literatur

Aceto, S. und Dondi, C. (2009): e-Learning, Lifelong Learning and Innovation in the working world. *eLearning Papers, No. 14* (Mai 2009).

ADL (2009): Advanced Distribute Learning. Verfügbar von http://www.adlnet.gov/Pages/Default.aspx, abgefragt am 3. September 2009

ATutor (20009): ATutor Learning Content Management System. Verfügbar von http://www.atutor.ca/, abgefragt am 2. September 2009

Blees, I. und Rittberger, M. (2009): Web 2.0 Learning Environment: Concept, Implementation, Evaluation. *eLearning Papers, No. 15* (Juni 2009).

Bral, L., Maeyer, P. D., Wit, B. D., Jacobs, P., Ongena, T., Zwartjes, L. und Weghe, N. V. D. (2009): *GeoloGIS - Development of an Educational Software Module to Understand the Geological Structure of Belgium.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation IV. AGIT 2009, Salzbrug.

Bunt-Kokhuis, S. v. d. und Bolger, M. (2009): Talent competences in the new eLearning generation. *eLearning Papers, No. 15* (Juni 2009).

Car, A. (2008): *Towards a Quality Assurance Concept for Postgraduate Distance Learning Programmes for Professionals.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation III. AGIT 2008, Salzburg.

Curran, C. (2004): Strategies for E-Learning in Universities. Dublin City University.

DistancE-Learning, F. (2008): Fernunterrichtsstatistik 2007. Verfügbar von http://www.forum-distance-learning.de/content/downloads/FU-Statistik/FU_Statistik2007.pdf, abgefragt am 1. September 2009

Downes, S. (2007): Emerging Technologies for Learning. *National Research Council of Canada, Volume 2*.

Ehlers, U.-D. (2007): Towards greater quality literacy in a eLearning Europe. *eLearning Papers, No. 2* (Januar 2007).

eLML (2009): eLML Homepage. Verfügbar von http://www.elml.org, abgefragt am 9. August 2009

EU (2009): Seniors tell about HiStory. Verfügbar von http://www.history-project.eu, abgefragt am 5. September 2009

Fischer, F. (2009): *Learning in Geocommunities - an Explorative View on Geo-Social Network Communities.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation IV. AGIT 2009, Salzburg.

Fisler, J., Bleisch, S. und Weibel, R. (2006): *Das E-Leraning-Projekt GITTA - frei zugängliche Inhalte für die akademische Ausbildung in Geoinformation.* In: T. Jekel, A. Koller & J. Strobl, Hrsg. Lernen mit Geoinformation. AGIT 2006, Salzburg.

Fisler, J. und Schneider, F. (2008): Creating, Handling and Implementing E-Learning Courses and Content using the Open Source Tools OLAT and ELML at the University of Zurich. Universität Zürich.

GE-Energy (2009): GE Energy - Geospatial Asset Management Smallworld. Verfügbar von http://www.gepower.com/prod_serv/products/gis_software/en/index.htm, abgefragt am 7. September 2009

Gerson, S. (2000): E-CLASS: Creating a Guide to Online Course Development For Distance Learning Faculty. *Online Journal of Distance Learning Administration, Volume 3* (Issue 4).

Google (2009): Google Maps. Verfügbar von http://maps.google.com, abgefragt am 7. September 2009

Haller, P. (2008): *Navigieren mit Handheld und GPS.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation III. AGIT 2008, Salzburg.

Hamburg, I. und Hall, T. (2008): Informal learning and the use of Web 2.0 within SME training strategies. *eLearning Papers, No. 11* (November 2008).

IFel (2009): Moodle Tutorial. Verfügbar von http://moodle.ifel.ch/tutorial/moodle_intro/de/html/startseite.html, abgefragt am 3. Oktober 2009

IMS (2009): IMS Global Learning Consortium. Verfügbar von http://www.imsglobal.org/, abgefragt am 2. September 2009

Inspire (2009): INSPIRE Homepage. Verfügbar von http://inspire.jrc.ec.europa.eu/, abgefragt am 1. Oktober 2009

ISO (2009): ISO/TC 211 - Geographic information/Geomatics. Verfügbar von http://www.isotc211.org/, abgefragt am 7. September 2009

Jokisalo, E. und Riu, A. (2009): Informal learning in the era of Web 2.0. *eLearning Papers, No. 14* (Mai 2009).

Kos, A., Stopper, R., Muff, O. und Loew, S. (2008): *Application of Active Learning and Web-based Geographical Information for Natural Hazard Management.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation III. AGIT 2008, Salzburg.

Kupetz, R. und Ziegenmeyer, B. (2006): Flexible learning activities fostering autonomy in teaching training. English Department, University of Hannover.

Lubensky, R. (2006): The present and future of Personal Learning Environments (PLE) Verfügbar von http://www.deliberations.com.au/2006/12/present-and-future-of-personal-learning.html, letzte Aktualisierung am 31. August 2006, abgefragt am 3. September 2009

Mäs, S. und Reinhardt, W. (2007): *Europaweiter Austausch von GI e-Learning Kursen: ein Erfahrungsbericht.* In: T. Jekel, A. Koller & J. Strobl, Hrsg. Lernen mit Geoinformation II. AGIT 2007, Salzburg.

MetaCarta (2009): Geographic Search and Referncing Solutions - MetaCarta - At the Forefront of the GeoWeb. Verfügbar von http://www.metacarta.com/, abgefragt am 5. September 2009

Moodle (2009): Moogle.org: open-source community-based tools for learning. Verfügbar von http://moodle.org/, abgefragt am 2. September 2009

Niederhuber, M., Grossmann, T., Weibel, R. und Bleisch, S. (2009): *Integration von GITTA-Lektionen in den Hochschulunterricht - Beispiele von Lehr- und Lernszenarien.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation IV. AGIT 2009, Salzburg.

Niederhuber, M., Heinimann, H.-R. und Hebel, B. (2005): e-Learning basierte Fallstudien zur akademischen Ausbildung in der Geoinformatik: Methodisches Konzept, Umsetzung und Erfahrungen. Professur Forstliches Ingenieurwesen, Eidgenössische Technische Hochschule Zürich.

O'Reilly (2006): O'Reilly Where 2.0 Conference. Verfügbar von http://conferences.oreillynet.com/where2005, letzte Aktualisierung am 10. März 2006, abgefragt am 6. Septemer 2009

OGC (2005a): OpenGIS Filter Encoding Implementation Specification. Standardisiert durch: Open Geospatial Consortium Inc. Verfügbar von: http://portal.opengeospatial.org/files/?artifact_id=8340. Abgefragt am: 6. September 2009

OGC (2005b): Web Feature Service Implementation Specification Standardisiert durch: Open Geospatial Consortium Inc. Verfügbar von: http://portal.opengeospatial.org/files/?artifact_id=8339. Abgefragt am: 6. September 2009

OGC (2006): OpenGIS® Web Map Server Implementation Specification. Standardisiert durch: Open Geospatial Consortium Inc. Verfügbar von: http://portal.opengeospatial.org/files/?artifact_id=4756. Abgefragt am: 6. September 2009

OGC (2007): OpenGIS® Geography Markup Language (GML) Encoding Standard. Standardisiert durch: Open Geospatial Consortium Inc. Verfügbar von: http://portal.opengeospatial.org/files/?artifact_id=20509. Abgefragt am: 6. September 2009

OGC (2009): Open Geospatial Consortium Inc. Verfügbar von http://www.opengeospatial.org/, abgefragt am 7. September 2009

OLAT (2009): OLAT - The Open Source LMS. Verfügbar von http://www.olat.org, abgefragt am 2. September 2009

OpenLayers (2009): OpenLayers - Homepage. Verfügbar von http://www.openlayers.org/, abgefragt am 1. September 2009

OSGeo (2009): OSGeo.org - Your Open Source Compass. Verfügbar von http://www.osgeo.org/, abgefragt am 5. September 2009

Pezerovic, E. (2009): *Ein OGC-konformer Web Feature Service Prototyp zur Suche von Grundstücken: Beispiel einer Webapplikation unter Verwendung des österreichischen Katasters.* Masterarbeit, Institut für Wirtschaftsgeographie und Geoinformatik, Wirtschaftsuniversität Wien, Wien.

Schiederig, K. (2007): Using e-Learning for social sciences: practical lessons from the Free University of Berlin. *eLearning Papers, No. 3* (März 2007).

Schieriau, H. (2007): *Geoland.at - ein Web-GIS Portal for den Unterricht?* In: T. Jekel, A. Koller & J. Strobl, Hrsg. Lernen mit Geoinformation II. AGIT 2007, Salzburg.

Schiewe, J. (2006): *Verwertungsmodelle für E-Learning-Materialien zur Geoinformatik*. 2. GIS-Ausbildungstagung 2006. Institut für Geoinformatik und Fernerkundung, Universität Osnabrück.

Schiewe, J., Grendus, B. und Plass, C. (2006): *Nutzung und Nachhaltigkeit von E-Learning-Materialien für die Geoinformatik - das Beispiel FerGI.* In: T. Jekel, A. Koller & J. Strobl, Hrsg. Lernen mit Geoinformation. AGIT 2006, Salzburg.

Scholze, T. und Wiemann, S. (2007): Successful Blended Learning Projects in 2006: Experiences in different formal, non-formal and informal learning environments. *eLearning Papers, No. 3* (März 2007).

Schwarz, J.-A. und Asche, H. (2006): *Vermittlung von GIS-relevantem Grundlagenwissen mithilfe von dynamischen interaktiven Lernangeboten.* In: T. Jekel, A. Koller & J. Strobl, Hrsg. Lernen mit Geoinformation. AGIT 2006, Salzburg.

Seufert, S. und Euler, D. (2004): Nachhaltigkeit von eLearning-Innovationen: Ergebnisse einer Delphi-Studie. Universität St. Gallen.

Strahovnik, V. und Mecava, B. (2009): Storytelling and Web 2.0 Services: A synthesis of old and new ways of learning. *eLearning Papers, No. 15* (Juni 2009).

Traun, C. (2009): *Continuing Professional Education Requirements and Offers in GI within Europe.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Learning with Geoinformation IV. AGIT 2009, Salzburg.

Traun, C. und Fally, M. (2007): *Problemorientiertes kollaboratives Lernen in online-basierten Geoinformatik-Fernstudien.* In: T. Jekel, A. Koller & J. Strobl, Hrsg. Lernen mit Geoinformation II. AGIT 2007, Salzburg.

Varis, T. (2006): eLearning and Higher Education. *eLearning Papers, No. 1* (November 2006).

VESTA-GIS (2009): VESTA-GIS Homepage. Verfügbar von http://www.vesta-gis.eu/, abgefragt am 1. Okober 2009

VNILES (2009): The Village of Niles - What is GIS. Verfügbar von http://www.vniles.com, abgefragt am 7. September 2009

Zumbach, J. und Bachleitner, S. (2007): *M-Learning im Geographie- und Wirtschaftskundeunterricht.* In: T. Jekel, A. Koller & K. Donert, Hrsg. Lernen mit Geoinformation II. AGIT 2007, Salzburg.

Anhang

eLearning Tutorial: Implementierung eines OGC-konformen Web-Clients zur Grundstückssuche

von

Edin Pezerovic

Inhaltsverzeichnis

1. Vorraussetzungen für die Grundstückssuche

Die in diesem Kurs enthaltenen Lerneinheiten erstellen zusammen einen Web-Client, um Ergebnisse einer Grundstückssuche in einem Kartenfenster darstellen zu können.

Der Web-Client bietet dem Benutzer die Möglichkeit nach Katastralgemeindenummer und Grundstücksnummer zu suchen. Diese beiden Werte bilden zusammen den eindeutigen Schlüssel für ein Grundstück der Digitalen Katastralmappe des Bundesamtes für Eich- und Vermessungswesen. Nach der Eingabe des Schlüssels, wobei auch sogenannte Wildcards eingegeben werden dürfen, wird eine Anfrage an ein Web Feature Service (WFS) abgeschickt, um die Suche durchzuführen. Ein WFS liefert aufgrund der Suchkriterien eine Liste von Vektorobjekten. Diese entsprechen den Grundstücken (umschließende Rechtecke oder die Einsetzpunkte der Grundstücksnummern). Für die Anzeige des Ergebnisses wird eine Karte mit den Grundstücksgrenzen an der geografischen Position des Ergebnisses von einem Web Map Service abgefragt. Diese Karte wird im Kartenfenster angezeigt und die Grundstücke aus der Anfrage an das WFS markiert.

Sowohl WFS als auch WMS sind durch das Open Geospatial Consortium spezifiziert. Durch diese Spezifikation ist es möglich, Client- und Serversoftware von verschiedenen Herstellern beliebig zu kombinieren, solange sich die Komponenten an die Spezifikation der Schnittstellen halten. In diesem Kurs wird für die Zugriffe auf die Services (WFS und WMS) OpenLayers verwendet. OpenLayers bietet die Möglichkeit, aus einer HTML-Seite heraus diese Services zu verwenden. Das Ergebnis des Web-Clients (nach den vier Kursen) zeigt folgende Abbildung.

Ergebnis der vier Lerneinheiten

Die nun folgenden Lerneinheiten bilden eine Einführung in die Programmierung solch eines Web-Clients. Es werden sowohl die Grundlagen als auch konkrete Beispiele für die Implementierung gegeben. Alle Beispiele verwenden OpenLayers als Client-Bibliothek und Services vom BEV, die für diesen Kurs zur Verfügung gestellt wurden.

Lernziele

- Kennenlernen des Begriffs Token
- Verstehen des Unterschiedes zwischen Session und Access Token
- Erklären von Timeouts bei Token
- Erklären der Aufgaben eines Interceptors
- Kennenlernen des BEV-Interceptors
- Generieren eines Tokens mittels Webapplikation
- Aufruf der XML-Schnittstelle des Interceptors
- Verstehen des Usermanagements beim BEV
- Verstehen der Struktur der XML-Antwort des Interceptors
- Erklären der Elemente der XML-Antwort des Interceptors
- Verstehen der Verwendung von XSD-Files
- Überblick über JAXB
- Alternativen zu JAXB
- Verwenden von Schema-Compiler XJC
- Verstehen der Verwendung von JavaBeans
- Verstehen des Unterschiedes zwischen Unmarshalling und Marshalling
- Verstehen von Unmarshalling mittels JAXB
- Verstehen von Marshalling mittels JAXB
- Verwenden der Validierung mittels JAXB
- Definition des Servletsbegriffes
- Verstehen einer HTTP-Session

1.1 Grundlagen zum Thema Token

In dieser Lerneinheit werden die Grundlagen zum Thema Token behandelt.
Nach einer eingehenden Definition des Begriffes Token wird die Verwendung des Tokens für die Absicherung von Services beschrieben. Abschließend zeigt ein Beispiel, wie Tokens bei den Services des BEV Verwendung finden.

Lernziele

- Kennenlernen des Begriffs Token
- Verstehen des Unterschiedes zwischen Session und Access Token
- Erklären von Timeouts bei Token

1.1.1 Was ist ein Token?

In der IT-Welt ist der Begriff Token wie folgt zu verstehen:

> Ein Token ist ein Objekt, welches ein zumeist exklusives Recht repräsentiert, eine Aktion durchzuführen.

Ein Token stellt somit eine Berechtigung dar, auf ein Service zuzugreifen. Beim Zugriff auf das Service muss der Aufrufer authentisiert und autorisiert werden. Um diesen zeitaufwendigen Vorgang nicht jedes Mal durchführen zu müssen, wird nach einmaligem Anmelden ein Token generiert. Das Mitschicken des Tokens impliziert automatisch eine erfolgreiche Anmeldung.
Das Objekt des Tokens kann dabei verschiedenste Ausprägungen annehmen. Die üblichste Form eines Tokens stellt ein eindeutiger Text dar. Dieser Text wird durch eine berechtigte Komponente erzeugt und muss beim Aufruf jeder nachfolgenden Aktion mitgegeben werden. Auf diese Weise kann geprüft werden, ob ein Zugriff berechtigt ist. Folgende zwei Varianten des Tokens sind in diesem Zusammenhang von Interesse:

- Session Token
- Access Token

1.1.2 Wofür werden Token verwendet?

Die bereits beschriebenen Tokenvarianten Session Token und Access Token werden nun näher erklärt.

Session Token
Benutzerinteraktion benötigt fast immer mehr als einen Zugriff auf serverseitige Ressourcen. Beim ersten Zugriff werden für den Benutzer verschiedene Informationen ermittelt, wie zum Beispiel die zugestandenen Rechte. Um diese bereits vorhandenen Informationen nicht jedes Mal neu ermitteln zu müssen, ist es von Vorteil, einen wiederkehrenden Benutzer zu erkennen. Aufgrund dessen ist es möglich, vorher abgespeicherte Information wieder zu verwenden und die Neuermittlung einzusparen. Damit ein Benutzer als wiederkehrender Benutzer erkannt wird, muss dieser eine Art von Kennung mitschicken. Beim ersten Zugriff generiert aus diesem Grund der Server einen Token, den er zusammen mit der Antwort an den Aufrufer zurückgibt. Bei allen nachfolgenden Aufrufen übermittelt der Benutzer diesen Token an den Server, welcher dann diese beiden Anfragen dem gleichen Benutzer zuordnen kann. Solch ein Token wird Session Token genannt und findet vor

allem in der Webentwicklung Verwendung. Hierbei werden Session-IDs generiert und als sogenannte HTTP-Cookies im Browser abgelegt. Bei der nächsten Anfrage an den gleichen Server (entpricht der gleichen Domain) wird das Cookie mitgeschickt.

Access Token

Gewisse Services im Internet bedürfen einer Authentisierung und Autorisierung. Dies erfolgt meist mittels Benutzername und Passwort oder Zertifikat. Aus Sicherheitsgründen ist es nicht vorteilhaft, Benutzername und Passwort jedes Mal an den Server zu übermitteln. Dadurch wird der Identitätsdiebstahl erleichtert. Stattdessen ist es sinnvoller, wenn die Übermittlung lediglich einmal erfolgt und dabei im Falle einer erfolgreichen Anmeldung ein Token generiert wird. Ab dem Zeipunkt gilt der Token stellvertretend für einen angemeldeten Benutzer. Solche Token werden Access Token genannt, da sie den Zugriff auf beschränkte Services ermöglichen. Zusätzlich können verschiedene Applikationen übereinkommen, einen gemeinsamen Token zu verwenden. Dadurch ist es ausreichend, wenn der Benutzer bei einer Applikation authentisiert wird. Alle beteiligten Applikationen sind dann ebenfalls verfügbar. Diese Vorgangsweise wird als Single Sign On (SSO) bezeichnet und findet durchwegs bei Portalen Einsatz. Die beteiligten Applikationen können zusätzlich den Token gegen einen Token Server validieren. Dadurch kann zusätzlich Missbrauch vorgebeugt werden. Der Kerberos Ticket Server entspricht solch einem Token Server für die Validierung ausgestellter Token.

1.1.3 Beispiel eines Token beim BEV

Die im Zuge dieses Tutorials verwendeten Services wurden vom Bundesamt für Eich- und Vermessungswesen (BEV) zur Verfügung gestellt. Diese Services sind durch einen, wie oben beschriebenen, Token geschützt. Daher ist es notwendig, einen Token bei jedem Request mitzuschicken. Ein vom BEV generierter Token sieht wie folgt aus: "BEVIntern-cc143258-43b6-4f85-8e9a-eae2b3b74db5". Wenn nun auf das WMS des BEV zugegriffen werden soll, muss der Token mitgegeben werden.

Beispiel eines Zugriffes auf das WMS

```
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_KM50/BEVIntern-
cc143258-43b6-4f85-8e9a-eae2b3b74db5
```

1.1.4 Token und Timeouts

Grundsätzlich ist es am sichersten, wenn der Benutzer sich bei jedem Zugriff neu anmelden muss. Dadurch wird jedes Mal festgestellt, ob der Benutzer dazu berechtigt ist. Aufgrund der oben beschriebenen Punkte sind jedoch Token praktikabler und führen zu mehr Benutzerfreundlichkeit. Das Mitschicken eines Tokens ist jedoch anfälliger gegenüber Identitätsdiebstahl.

> Identitätsdiebstahl ist die missbräuchliche Nutzung personenbezogener Daten oder Identitäten einer natürlichen Person durch Dritte.

Um den Identitätsdiebstahl zu erschweren, laufen Token in der Regel nach kurzer Zeit ab. Bei Online-Banking Anwendungen ist der Zeitraum der Gültigkeit des Tokens sehr begrenzt, meistens nicht mehr als einige wenige Minuten. Normale Zeitspannen für Webanwendungen sind im Bereich von 30 Minuten zu finden. Das BEV kann beim Anlegen des Users diese Zeitspanne definieren.

1.1.5 Zusammenfassung

Diese Lerneinheit zeigt, wofür ein Token verwendet werden kann. Token sind Objekte, die ein exklusives Recht repräsentieren. Mittels Token können gewissen Informationen übergeben werden, unter anderem Session und Access Informationen. Session Token bezeichnen die Zusammengehörigkeit zweier Zugriffe und verbinden somit unabhängige Ereignisse (Zugriffe) zu einer zusammengehörigen Kommunikation. Access Token eröffnen die Möglichkeit, auf gesicherte Ressourcen zuzugreifen. Beim BEV werden Token als Access Token verwendet, mittels welcher auf Web Services zugegriffen werden darf. Ein Token ist jedoch nicht unbegrenzt gültig. Sogenannte Timeouts definieren den Zeitraum, in welchem ein Token Gültigkeit bewahrt.

1.2 Zugriffe mittels Interceptoren abfangen

In dieser Lerneinheit wird erklärt, wie ein Interceptor verwendet werden kann, um Zugriffe auf schützenswürdige Inhalte zu kontrollieren. Im Zusammenspiel mit dem Tokenhandling kann ein Interceptor sensible Inhalte kapseln und nur denjenigen Benutzern zur Verfügung stellen, die dafür berechtigt wurden. Diese Berechtigung ist nur dem Interceptor bekannt und befreit alle durch den Interceptor gekapselten Komponenten von der Aufgabe, sich um Berechtigungen kümmern zu müssen.
Nach einer kurzen Einführung wird der Interceptor vom BEV vorgestellt.

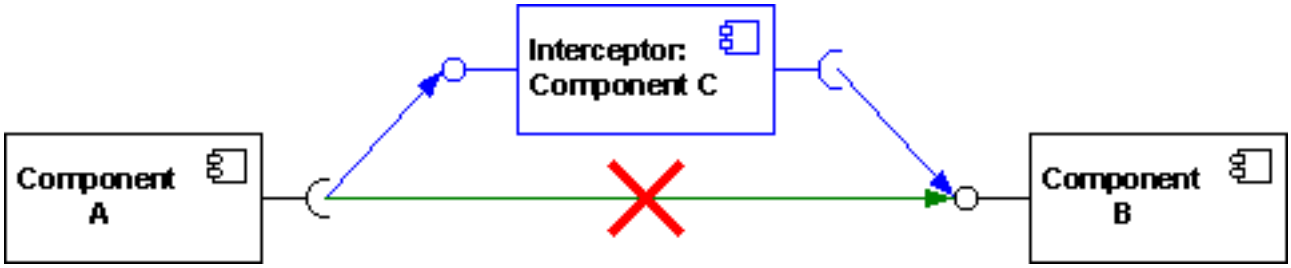
Lernziele

- Erklären der Aufgaben eines Interceptors
- Kennenlernen des BEV-Interceptors
- Generieren eines Tokens mittels Webapplikation
- Aufruf der XML-Schnittstelle des Interceptors
- Verstehen des Usermanagements beim BEV

1.2.1 Was wacht ein Interceptor?

Ein Interceptor ist ein Entwurfsmuster aus der Softwareentwicklung. Der Begriff Interceptor leitet sich vom Englischen Wort "intercept" ab, welches in etwa "abfangen" bedeutet. Somit fängt ein Interceptor eine Anfrage an ein Sevice ab.

Beispiel eines Interceptors



Beim obigen Beispiel versucht Component A auf Component B zuzugreifen. Da jedoch der direkte Aufruf nicht möglich ist (grüner Pfeil), muss Component A die Anfrage an den Interceptor (Component C) stellen. Nur der Interceptor hat direkten Zugriff auf Component B. Auf diese Weise kann der Interceptor alle Zugriffe auf Component B kontrollieren.
Die Möglichkeit, Zugriffe auf bestimmte Komponenten zu kontrollieren wird im Zusammenhang mit Token zur Berechtigung von Benutzern verwendet. Beim ersten Zugriff muss sich der Benutzer am Interceptor anmelden. Durch diese Anmeldung wird für eine bestimmte Zeit ein Token durch den Interceptor erstellt. Bei den einzelnen Zugriffen auf die Softwarekomponenten, die durch den Interceptor geschützt werden, muss der Benutzer den generierten Token mitgeben. Der Zugriff erfolgt dennoch mittels Interceptor. Der Benutzer sendet die gleiche Anfrage an den Interceptor, die auch an die Softwarekomponente direkt ergehen würde. Der Interceptor prüft, ob ein Token vorhanden und noch gültig ist. Sollte dies der Fall sein, leitet der Interceptor die Anfrage an die gewünschte Komponente weiter.

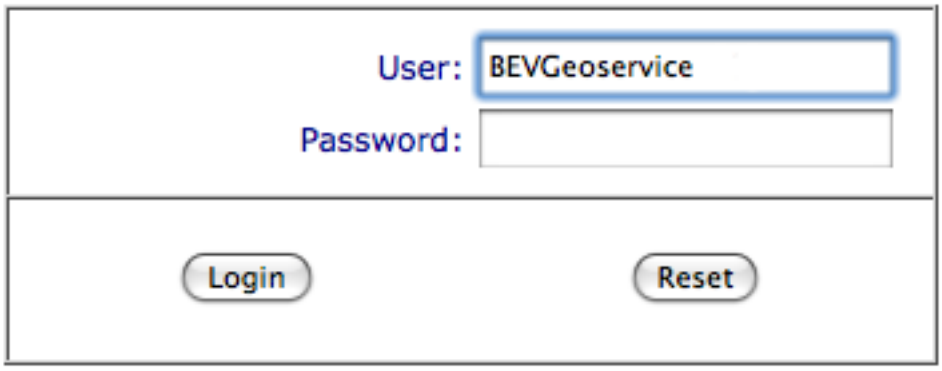
1.2.2 Zugriffsmöglichkeiten beim Interceptor des BEV

Beim BEV ist der Interceptor als Webapplikation umgesetzt. Der Grund liegt in den Komponenten, die der Interceptor schützt. Dies sind ausschließlich Web Services (WMS und WFS), die ebenfalls mittels HTML abgefragt werden. Daher liegt es nahe, den Interceptor als Web Applikation anzulegen. Der Interceptor generiert einen Access Token und Beispiele für die einzelnen Zugriffmöglichkeiten auf die Services hinter dem Interceptor.
Grundsätzlich bestehen beim Interceptor des BEV zwei Möglichkeiten, um einen Token zu generieren. Dies kann durch Interaktion eines Benutzers mit dem Interceptor erfolgen (Loginmaske) oder ohne Benutzerinteraktion durch rein technische Zugriffe mittels XML. Beide Varianten werden im Folgenden erläutert.

1.2.3 Tokengenerierung mittels HTML-Zugriff

Der Interceptor bietet die Möglichkeit per Webseite einen Token zu generieren. Dies erfolgt durch Aufruf der Webseite mit der URL "http://217.13.180.218/GeoServer2/login.jsp" welche folgende Loginmaske liefert:

Interceptor-Loginmaske



Nach Angabe des Benutzernamens und des Passwortes wird der Token generiert. Zusätzlich werden noch gültige Zugriffe auf die geschützten GeoServices vom Interceptor geliefert.

1.2.4 Ergebnis eines HTML-Zugriffes

Folgende Abbildung zeigt ein Beispiel einer Antwort auf eine HTML-Anfrage:

Beispiel einer HTML-Antwort (Teil 1)

User 'BEVGeoservice-03' successfully authorized in user management.

INTERCEPTOR-TOKEN:

COMMON:
---

id => BEVIntern-64528f5f-aad1-4990-bf4c-eb958a3f5b67
user (id, name) => 32efcd47-047a-49c7-a7a7-f08085746bbe, BEVGeoservice-03
group (id, name) => a84f5559-ce5c-4398-9e11-42230ddfd1ca, BEVIntern

valid from => Sat Aug 29 15:39:18 CEST 2009
valid to => Sun Aug 30 15:39:18 CEST 2009

created by appl. version => 2.2.0

SERVICE 'ASP_WFS'
---

name => ASP_WFS
description => BEV DKM service with wfs access
type => WFS
url => http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-64528f5f-aad1-4990-bf4c-eb958a3f5b67
bounding box => crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description: SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich
query filters:

examples:
WFS GetCapabilities (1.1.0)
WFS DescribeFeatureType (1.1.0) - GrundstuecksRechtecke
WFS DescribeFeatureType (1.1.0) - KatastralgemeindenRechtecke
WFS GetFeature (1.1.0) - Grundstuecksnummer, KGGNR = 910111992
WFS GetFeature (1.1.0) - Grundstuecksnummer, KGGNR = 91013511/1
WFS GetFeature (1.1.0) - Katastralgemeinde, KGNR = 91011
WFS GetFeature (1.1.0) - Katastralgemeinde, KGNR = 91013

SERVICE 'GeoRef'
---

name => GeoRef
description => BEV other web applications: BEV GeoRef
type => WEBAPP
url => http://217.13.180.218/GeoServer2/Interceptor/WebApp/GeoRef/BEVIntern-64528f5f-aad1-4990-bf4c-eb958a3f5b67/
bounding box => -- NOT CONFIGURED --
query filters:
-- NOT CONFIGURED --
examples:
Start (Homepage)

Beispiel einer HTML-Antwort (Teil 2)

SERVICE 'ASP_DKM'
---

name => ASP_DKM
description => BEV internal DKM service with wms access
type => WMS
url => http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-64528f5f-aad1-4990-bf4c-eb958a3f5b67
bounding box => crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description: SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich
query filters:

examples:
WMS GetCapabilities (1.1.1)
WMS GetMap (Layer: Verwaltungsgrenzen)
WMS GetMap (Layer: Katastralgemeinden)
WMS GetFeatureInfo (Layer: Katastralgemeinden)

SERVICE 'ASP_KM50'
---

name => ASP_KM50
description => BEV internal KM50 service with wms access
type => WMS
url => http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_KM50/BEVIntern-64528f5f-aad1-4990-bf4c-eb958a3f5b67
bounding box => crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description: SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich
query filters:

examples:
WMS GetCapabilities (1.1.1)
WMS GetMap (Layer: KM50)

1.2.5 Tokengenerierung mittels XML-Zugriff

Eine weitere Möglichkeit, die der Interceptor bietet, um einen Token zu generieren ist mittels Zugriff auf die XML-Schnittstelle. Hierbei wird ein HTTP-Get Request an die URL des Interceptors geschickt, wobei der Benutzername und das Passwort als Parameter angehängt werden. Die URL für den Aufruf der XML-Schnittstelle lautet "http://217.13.180.218/GeoServer2/GatewayHttp". An die Anfrage müssen noch folgende Parameter angehängt werden:

- username: enthält den Benutzernamen
- password: enthält das Passwort
- showinfoxml: teilt dem Interceptor mit, dass die Antwort als XML erwartet wird

Beispiel einer XML-URL

```
http://217.13.180.218/GeoServer2/GatewayHttp?
username=MYUSER&password=MYPWD&showinfoxml
```

1.2.6 Ergebnis eines XML-Zugriffes

Folgende Abbildung zeigt ein Beispiel einer XML-Antwort einer Anfrage an den Interceptor:

Beispiel einer XML-Antwort (Teil 1)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TOKEN>
    <ID>BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c</ID>
    <USERID>32efcd47-047a-49c7-a7a7-f08085746bbe</USERID>
    <USERNAME>BEVGeoservice-03</USERNAME>
    <GROUPID>a84f5559-ce5c-4398-9e11-42230ddfd1ca</GROUPID>
    <GROUPNAME>BEVIntern</GROUPNAME>
    <VALIDFROM>Thu Aug 27 10:39:20 CEST 2009</VALIDFROM>
    <VALIDTO>Fri Aug 28 10:39:20 CEST 2009</VALIDTO>
    <APPVERSION>2.2.0</APPVERSION>
    <SERVICES>
        <SERVICE>
            <NAME>ASP_WFS</NAME>
            <DESCRIPTION>BEV DKM service with wfs access</DESCRIPTION>
            <TYPE>WFS</TYPE>
            <URL>http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c</URL>
            <BOUNDINGBOX>crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description:
SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich</BOUNDINGBOX>
            <QUERYFILTERS />
            <EXAMPLES>
                <EXAMPLE>
                    <NAME>WFS GetCapabilities (1.1.0)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetCapabilitie
s&amp;SERVICE=WFS&amp;VERSION=1.1.0 </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WFS DescribeFeatureType (1.1.0) - GrundstuecksRechtecke</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=DescribeFeatur
eType&amp;typename=GrundstuecksRechtecke </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WFS DescribeFeatureType (1.1.0) - KatastralgemeindenRechtecke</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=DescribeFeatur
eType&amp;typename=KatastralgemeindenRechtecke </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WFS GetFeature (1.1.0) - Grundstuecksnummer, KGGNR = 910111992</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetFeature
&amp;typename=ASP_WFS:GrundstuecksRechtecke&amp;
filter=%3CFilter+xmlns=%22http://www.opengis.net/ogc%22%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3EKGGNR%3C/PropertyName%3E%3C
Literal%3E910111992%3C/Literal%3E%3C/PropertyIsEqualTo%3E%3C/Filter%3E </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WFS GetFeature (1.1.0) - Grundstuecksnummer, KGGNR = 91013511/1</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetFeature
&amp;typename=ASP_WFS:GrundstuecksRechtecke&amp;
filter=%3CFilter+xmlns=%22http://www.opengis.net/ogc%22%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3EKGGNR%3C/PropertyName%3E%3C
Literal%3E91013511/1%3C/Literal%3E%3C/PropertyIsEqualTo%3E%3C/Filter%3E </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WFS GetFeature (1.1.0) - Katastralgemeinde, KGNR = 91011</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetFeature
&amp;typename=ASP_WFS:KatastralgemeindenRechtecke&amp;
filter=%3CFilter+xmlns=%22http://www.opengis.net/ogc%22%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3EKGNR%3C/PropertyName%3E%3CL
iteral%3E91011%3C/Literal%3E%3C/PropertyIsEqualTo%3E%3C/Filter%3E </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WFS GetFeature (1.1.0) - Katastralgemeinde, KGNR = 91013</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetFeature
&amp;typename=ASP_WFS:KatastralgemeindenRechtecke&amp;
filter=%3CFilter+xmlns=%22http://www.opengis.net/ogc%22%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3EKGNR%3C/PropertyName%3E%3CL
iteral%3E91013%3C/Literal%3E%3C/PropertyIsEqualTo%3E%3C/Filter%3E</URL>
                </EXAMPLE>
            </EXAMPLES>
        </SERVICE>
```

Beispiel einer XML-Antwort (Teil 2)

```
        <SERVICE>
            <NAME>GeoRef</NAME>
            <DESCRIPTION>BEV other web applications: BEV GeoRef</DESCRIPTION>
            <TYPE>WEBAPP</TYPE>
            <URL>http://217.13.180.218/GeoServer2/Interceptor/WebApp/GeoRef/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c/
</URL>
            <BOUNDINGBOX />
            <QUERYFILTERS />
            <EXAMPLES>
                <EXAMPLE>
                    <NAME>Start (Homepage)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/WebApp/GeoRef/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c/start.php</URL>
                </EXAMPLE>
            </EXAMPLES>
        </SERVICE>
        <SERVICE>
            <NAME>ASP_DKM</NAME>
            <DESCRIPTION>BEV internal DKM service with wms access</DESCRIPTION>
            <TYPE>WMS</TYPE>
            <URL>http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c</URL>
            <BOUNDINGBOX>crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description:
SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich</BOUNDINGBOX>
            <QUERYFILTERS />
            <EXAMPLES>
                <EXAMPLE>
                    <NAME>WMS GetCapabilities (1.1.1)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetCapabilitie
s&amp;SERVICE=WMS&amp;VERSION=1.1.1 </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WMS GetMap (Layer: Verwaltungsgrenzen)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetMap&amp;
SERVICE=WMS&amp;VERSION=1.1.1&amp;LAYERS=5&amp;FORMAT=image/png&amp;SRS=EPSG:31255&amp;BBOX=-290000,166000,-29000,293000&amp;
WIDTH=823&amp;HEIGHT=558&amp;STYLES=default </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WMS GetMap (Layer: Katastralgemeinden)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetMap&amp;
SERVICE=WMS&amp;VERSION=1.1.1&amp;LAYERS=0&amp;FORMAT=image/png&amp;SRS=EPSG:31255&amp;BBOX=-290000,166000,-29000,293000&amp;
WIDTH=823&amp;HEIGHT=558&amp;STYLES=default </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WMS GetFeatureInfo (Layer: Katastralgemeinden)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetFeatureInfo
&amp;SERVICE=WMS&amp;VERSION=1.1.1&amp;LAYERS=0&amp;STYLES=default&amp;FORMAT=image/png&amp;SRS=EPSG:31255&amp;
BBOX=-290000,166000,-29000,293000&amp;WIDTH=823&amp;HEIGHT=558&amp;QUERY_LAYERS=1&amp;X=353&amp;Y=158</URL>
                </EXAMPLE>
            </EXAMPLES>
        </SERVICE>
        <SERVICE>
            <NAME>ASP_KM50</NAME>
            <DESCRIPTION>BEV internal KM50 service with wms access</DESCRIPTION>
            <TYPE>WMS</TYPE>
            <URL>http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_KM50/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c
</URL>
            <BOUNDINGBOX>crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description:
SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich</BOUNDINGBOX>
            <QUERYFILTERS />
            <EXAMPLES>
                <EXAMPLE>
                    <NAME>WMS GetCapabilities (1.1.1)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_KM50/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetCapabiliti
es&amp;SERVICE=WMS&amp;VERSION=1.1.1 </URL>
                </EXAMPLE>
                <EXAMPLE>
                    <NAME>WMS GetMap (Layer: KM50)</NAME>
                    <URL>
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_KM50/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetMap&amp;
SERVICE=WMS&amp;VERSION=1.1.1&amp;LAYERS=0&amp;FORMAT=image/png&amp;SRS=EPSG:31255&amp;BBOX=-290000,166000,-29000,293000&amp;
WIDTH=823&amp;HEIGHT=558&amp;STYLES=default</URL>
                </EXAMPLE>
            </EXAMPLES>
        </SERVICE>
    </SERVICES>
</TOKEN>
```

1.2.7 Usermanagement beim Interceptor des BEV

Der Interceptor generiert einen Token, sobald ein Benutzer authentisiert wurde. Diese Authentisierung und Autorisierung erfolgt anhand des Benutzernamens und des zugehörigen Passwortes. Dafür muss ein Benutzer beim Interceptor durch das BEV angelegt worden sein. Im Zuge des Anlegen eines Benutzers wird auch festgelegt, wie lange ein Token gültig bleibt.

1.2.8 Zusammenfassung

Interceptoren werden als Zwischenhändler bei der Anfrageverarbeitung verwendet. Diese Lerneinheit zeigt, wofür diese Interceptoren verwendet werden können. Das BEV verwendet einen Interceptor, um die in der vorherigen Lerneinheit beschriebenen Token zu generieren. Des Weiteren prüft der Interceptor im Zuge jedes Zugriffes, ob der Token noch gültig ist. Hierfür muss vorher durch den Interceptor ein Token erstellt worden sein. Dies kann mittels HTML- oder XML-Zugriff erfolgen. Beide Varianten greifen auf die im Interceptor definierten User zu und prüfen Username und Passwort gegen die Datenbank des Usermanagements.

1.3 Inhalte der XML-Antwort des Interceptors

In dieser Lerneinheit wird die Struktur und der Inhalt der XML-Antwort des Interceptors erklärt. In der vorhergehenden Unit lag der Schwerpunkt auf der Aufgabe des Interceptors. Diese Unit beschäftigt sich mit der Umsetzung des Zugriffes auf die XML-Antwort des Interceptors. Großes Augenmerk wird auf die automatisierte Weiterverarbeitung der XML-Antwort gelegt.

Lernziele

- Verstehen der Struktur der XML-Antwort des Interceptors
- Erklären der Elemente der XML-Antwort des Interceptors
- Verstehen der Verwendung von XSD-Files

1.3.1 Struktur der XML-Antwort

Die in der vorherigen Unit erwähnte XML-Antwort gilt beim Interceptor als Bestätigung der Authetisierung. Im Zuge dieser Bestätigung werden folgende Informationen an den Aufrufer übermittelt:

- generierter Token
- Benutzerinformationen
- Gültigkeit des Tokens
- Zugeordnete Services
- Beispiele für Aufrufe der Services

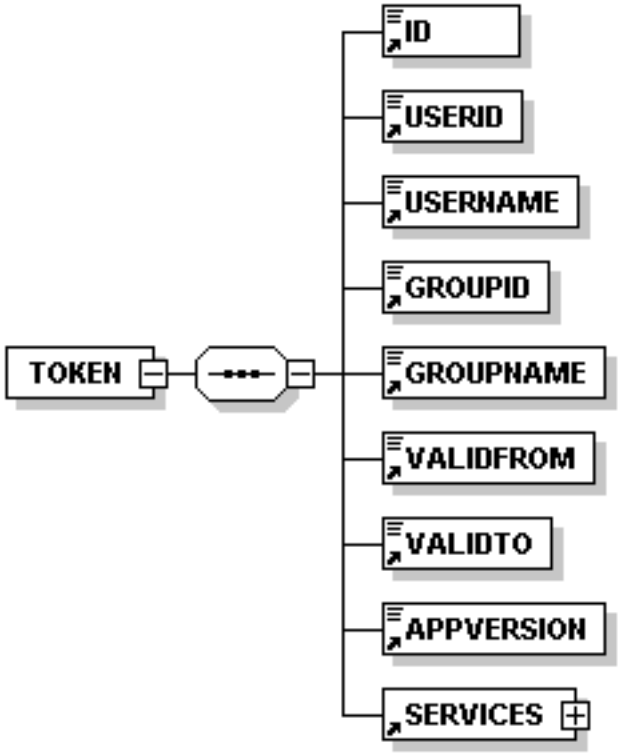
Die einzelnen Teile der XML-Antwort werden im Folgenden näher erklärt.

1.3.2 Erklärung der einzelnen Elemente

Element Token

Das Element Token bildet das Rootelement der XML-Antwort. Darin sind die restlichen Informationen der XML-Antwort enthalten. Das wichtigste Element unterhalb des Elementes Token ist die ID, welche den Token darstellt. Die Gültigkeit des Tokens wird durch die Elemente ValidFrom und ValidTo repräsentiert. Zusätzlich werden noch Informationen zum angemeldeten Benutzer übertragen. Darunter fallen Username, UserID und die Gruppenzugehörigkeit durch die Elemente GroupID und Groupname. Folgende Grafik zeigt einen Überblick über das Element Token:

Aufbau des Elementes Token

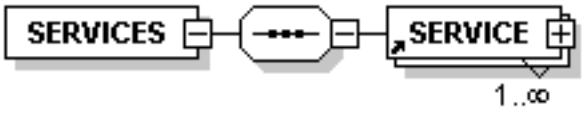


Wie der Grafik zu entnehmen ist, enthält das Element Token die hier beschriebenen Elemente. Des Weiteren sind noch Services angeführt.

Element Services

Das Element Services ist ein Subelement vom Tag Token und enthält die dem angemeldeten Benutzer zugeordnete Liste an Services. Diese können mehrfach vorkommen, daher auch die Ausprägung des Elementes als Liste. Folgende Grafik zeigt das Element Services:

Aufbau des Elementes Services



Element Service

Das Element Service beschreibt ein dem Benutzer zugesichertes Service. Die übermittelten Informationen enthalten den Namen des Services, eine Beschreibung und den Typ des Services. Dieser kann folgende Ausprägungen haben:

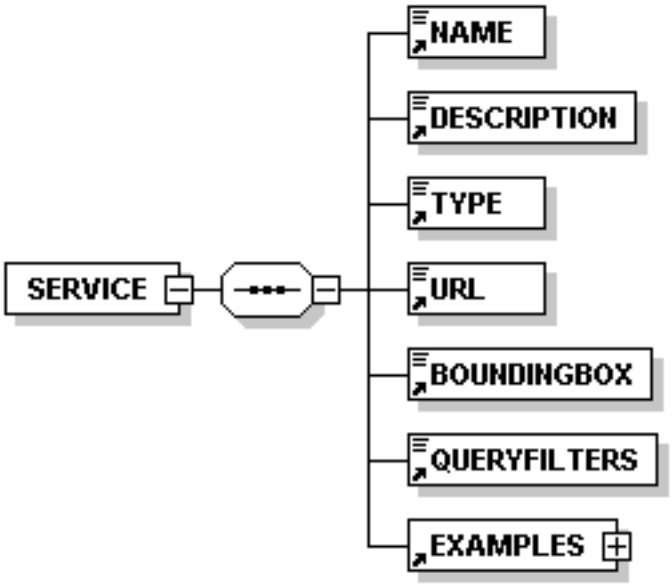
- WFS
- WMS
- WEBAPP

Die einzelnen Ausprägungen sollen selbsterklärend sein. Das nächste Element unterhalb von Service ist die URL. Diese wird für den Zugriff auf das Service verwendet und enthält bereits den generierten Token. Daher kann für den späteren Zugriff diese URL ohne Veränderung verwendet werden. Zuletzt sei noch das Element BoundingBox erwähnt, welches das umschließende Rechteck um alle Daten des Services, somit des dahinter liegenden Layers, definiert. Hier ein Beispiel für die Angabe einer BoundingBox:

```
crs: EPSG:31255; xmin: -298000.0; ymin: 131000.0; xmax: 298000.0; ymax: 439000.0; description: SPATIALEXTENT ID: e8892629-e8a8-497b-b085-01f2890876e5; name: Österreich
```

Folgende Grafik zeigt den Aufbau des Elementes Service:

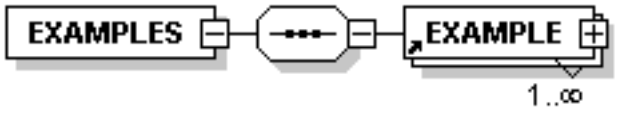
Aufbau des Elementes Service



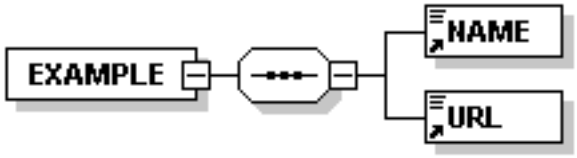
Element Examples

Das Element Service enthält zusätzlich noch Beispiele für Zugriffe auf Daten des Services. Diese befindet sich innerhalb des List-Elementes Examples. Jeder Eintrag vom Element Examples spiegelt eine vollständige URL dar, die Daten liefert. Zusätzlich ist im Element Name ein beschreibender Name für die URL enthalten.

Aufbau des Elementes Examples



Aufbau des Elementes Example



1.3.3 Was ist ein XSD-File?

Die Beschreibung der Elemente der XML-Antwort ist formalisierbar. Diese Formalisierung geht soweit, dass der übermittelte Inhalt der XML-Antwort geprüft werden kann. Um ein XML File beschreiben zu können, werden XSD-Files verwendet. XSD steht für XML Schema Description und enthält ein sogenanntes Schema einer XML Datei. Das Schema beschreibt den genauen Aufbau und die einzelnen Typen der XML-Antwort. Das XSD-File bietet die Möglichkeit, die XML-Antwort softwaretechnisch zu verarbeiten. Ohne ein XSD-File muss der genaue Aufbau vorher bekannt sein.

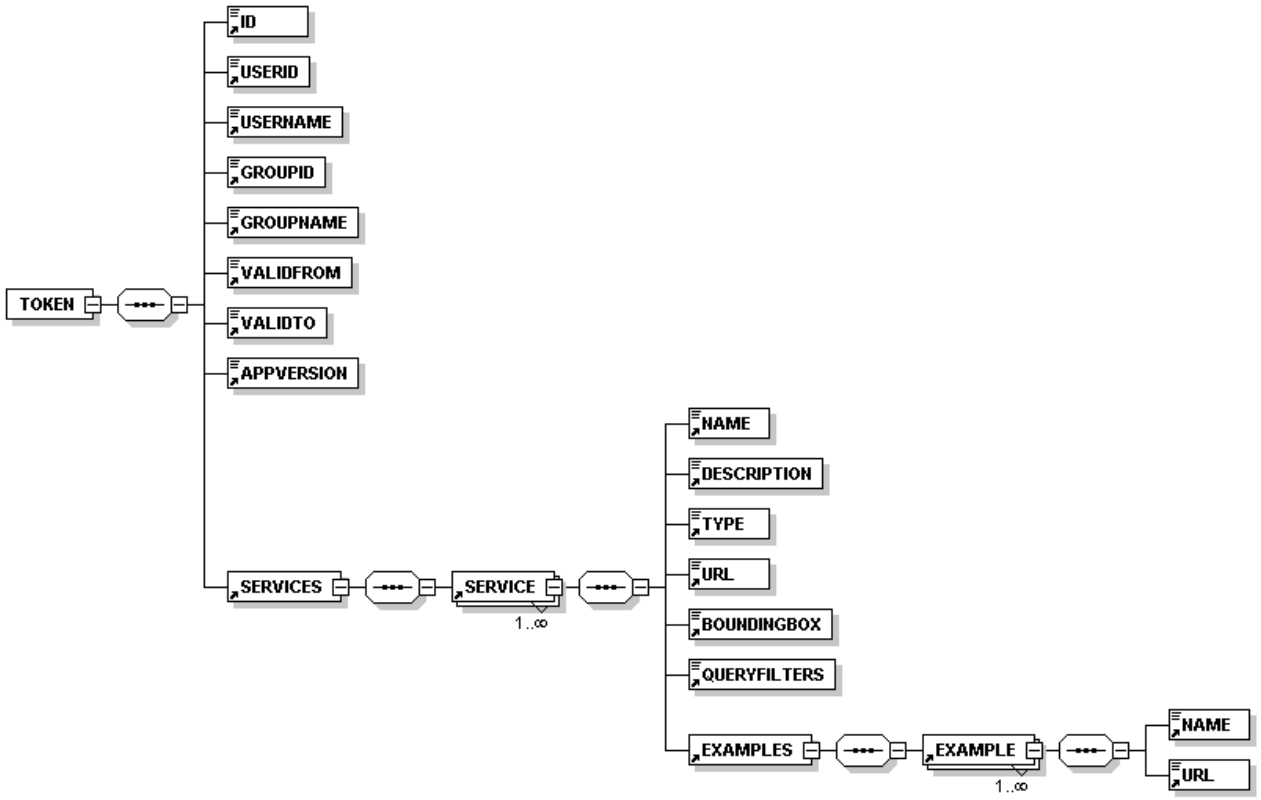
Unterschied XSD- zu DTD-File

Die Struktur eines XML-Files kann auch durch DTD-Files (Document Type Definition) definiert werden. Im Vergleich zu XSD-Files sind diese jedoch nicht gleichwertig verwendbar. Die Definition durch DTD-Files ist auf einer gröberen Ebene und ermöglicht dadurch kaum Validierungen von Inhalten. XSD-Files gehen hier einen anderen Weg, indem nahezu jede Feinheit definierbar ist. Dadurch ist eine Validierung von Inhalten gegen ein XSD-File möglich.

1.3.4 Schemagrafik der XML-Antwort

Ein XSD-File kann auch als Schemagrafik dargestellt werden. Folgende Abbildung zeigt den Gesamtüberblick über das Schema der XML-Antwort des Interceptors:

Aufbau des Elementes Examples



1.3.5 Zusammenfassung

Für die Verarbeitung einer XML-Antwort des Interceptors sind einige Grundlagen nötig. Diese werden in dieser Lerneinheit behandelt und enthalten eine Definition und Erklärung der Struktur der Interceptor-Antwort, genaue Erklärung der einzelnen Elemente und einen Überblick über die Schemagrafik der XML-Antwort. Im Zuge dessen werden auch Grundlagen zum Thema Schema und XSD behandelt.

1.4 Java und der Zugriff auf XML

In dieser Lerneinheit wird die Möglichkeit vorgestellt, mittels JAXB XML-Dokumente in Java zu verarbeiten. Die Voraussetzung für die Verarbeitung von XML-Dateien mittels JAXB ist das in der vorherigen Unit beschriebene XSD-File.

Lernziele

- Überblick über JAXB
- Alternativen zu JAXB
- Verwenden von Schema-Compiler XJC
- Verstehen der Verwendung von JavaBeans

1.4.1 Was ist JAXB?

JAXB steht für Java Architecture for XML Binding und erlaubt Java Klassen an XML Repräsentanzen zu binden. Binden bedeutet in diesem Zusammenhang eine Zuordnung zwischen Javaobjekten und den entsprechenden Werten im XML-File zu schaffen. Zum Beispiel korrespondiert das Element "Name" im XML-File mit dem Attribut "name" eines Javaobjektes. Um diese Zuordnung aufbauen zu können, benötigt JAXB das XSD-File der XML-Datei. Aufgrund des XSD-Files werden die Java-Klassen generiert, was den Aufwand erspart, diese eigenhändig zu implementieren. Der Vorteil an den generierten Java-Klassen ist, dass die Komplexität der Zuordnung vom Entwickler gekapselt wird. Der Entwickler muss sich um diese Feinheiten nicht mehr kümmern und es muss lediglich mit JavaBeans gearbeitet werden.

1.4.2 JAXB Alternativen

Neben JAXB gibt es noch eine Reihe von anderen Bibliotheken, um von Java aus XML-Dateien verarbeiten zu können.

SAX
SAX steht für Simple API for XML und gehört zu den ereignisbasierten XML-Bibliotheken. Hierbei liest der Parser das XML-Dokument durch und ruft bei bestimmten Ereignissen Rückrufroutinen auf, die zu Beginn dem Parser mitgeteilt wurden. Der aktive Part bei dieser Art liegt im Parser. Der Vorteil ist der geringe Speicherverbrauch, wodurch sich SAX für große Dateien gut eignet. Nachteil ist die umständliche Programmierart mittels Rückrufroutinen.

DOM
DOM steht für Document Object Model und gehört zu den baumbasierten XML-Bibliotheken. Mittels DOM wird eine baumartige Objektrepräsentanz der XML-Datei im Speicher aufgebaut. Der Speicherbedarf ist hier sehr groß, wodurch es für große XML-Dateien nicht brauchbar ist. Die verwendeten Objekte sind vom generischen Typ Node. Der Vorteil liegt in der leichten Programmierbarkeit, da beliebig innerhalb des XML-Dokumentes gelesen werden kann.

StAX

StAX steht für Streaming API for XML und stellt eine Mischung zwischen eventbasierten und baumbasierten XML-Bibliotheken dar. Hierbei wird eine Art Cursor durch die XML-Datei bewegt, während Daten gelesen werden. Die aktive Seite ist hierbei das Programm und nicht der Parser. Es wird jedoch keine Repräsentanz des Inhaltes im Speicher abgelegt.

JDOM

JDOM stellt eine XML-Bibliothek aufbauend auf SAX und DOM dar, wobei es zu den baumbasierten XML-Bibliotheken gehört. Es bietet eine vereinfachte DOM-Programmierschnittstelle, kann jedoch zum Parsen SAX verwenden. Dadurch ergibt sich die Komination der Vorteile indem das XML-Dokument effizient geparst wird, trotzdem danach einfach im Zugriff ist. Der Nachteil des Speicherverbrauches bleibt jedoch bestehen.

1.4.3 Schema-Compiler XJC

Wie bereits erwähnt erstellt JAXB eine Zuordnung zwischen JavaBeans und der Struktur des XML-Files. Dieses Mapping wird mittels Annotations innerhalb der JavaBeans aufgebaut. Folgende Abbildung zeigt einen Ausschnitt einer solchen Konfiguration innerhalb einer Java-Klasse:

JAXB-Annotations innerhalb eines JavaBeans

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "id",
    "userid",
    "username",
    "groupid",
    "groupname",
    "validfrom",
    "validto",
    "appversion",
    "services"
})
@XmlRootElement(name = "TOKEN")
public class TOKEN {

    @XmlElement(name = "ID", required = true)
    protected String id;
    @XmlElement(name = "USERID", required = true)
    protected String userid;
    @XmlElement(name = "USERNAME", required = true)
    protected String username;
    @XmlElement(name = "GROUPID", required = true)
    protected String groupid;
    @XmlElement(name = "GROUPNAME", required = true)
    protected String groupname;
    @XmlElement(name = "VALIDFROM", required = true)
    protected String validfrom;
    @XmlElement(name = "VALIDTO", required = true)
    protected String validto;
    @XmlElement(name = "APPVERSION", required = true)
    protected String appversion;
    @XmlElement(name = "SERVICES", required = true)
    protected SERVICES services;
```

Diese Konfiguration mittels Annotation kann per Hand erstellt werden, nur wächst der Aufwand mit der Größe des Schemas. Etwaige Änderungen müssen ebenfalls nachgezogen werden. Um diesen Aufwand streichen zu können kann der XJC (steht für XML Java Binding Compiler) verwendet werden. XJC benötigt ein XSD-File und generiert aufgrund dessen die Baumstruktur der JavaBeans. Folgend ein Beispiel für einen XJC Aufruf, um die JavaBeans für die Antwort des Interceptors zu generieren:

```
xjc -d '.' -p "com.pe.gstsuche" interceptor.xsd
```

Die verwendeten Optionen stehen für das Zielverzeichnis (-d), in dem die generierten Klassen gespeichert werden und das zu verwendende Java-Package (-p), in welchem die Klassen angelegt werden.

1.4.4 Was sind JavaBeans?

JavaBeans sind Klassen in Java die folgenden Regeln entsprechen:

- die Klasse muss einen öffentlichen Standardkonstruktor besitzen
- die Klasse muss serialisierbar sein
- die Klasse muss alle Attribute kapseln und mittels öffentlichen Zugriffsmethoden zur Verfügung stellen (Getters und Setters)

Klassen dieser Art sind leicht verständlich und bieten zudem die Möglichkeit, mittels Introspektion Informationen auszulesen. Introspektion ist das Auslesen der Meta-Daten einer Klasse und wird von Entwicklungsumgebungen verwendet.

1.4.5 Zusammenfassung

Diese Lerneinheit beschäftigte sich mit der Thematik der Verarbeitung von XML-Dokumenten in Java. Es wurden unterschiedliche Alternativen vorgestellt und JAXB als empfohlene Variante genauer erklärt. Hierbei wurde der Schema-Compiler XJC und die Verwendung der generierten JavaBeans behandelt.

1.5 XML-Binding mittels Java

In dieser Lerneinheit werden die Erkenntnisse der vorherigen Lerheinheit verwendet, um in Java ein XML-Dokument zu verarbeiten. Dabei werden die Inhalte des XML-Dokumentes in Java ausgelesen, und als Javaobjekthierarchie abgebildet. Im Anschluß wird diese Objekthierarchie wieder in ein XML-Dokument umgewandelt.

Lernziele

- Verstehen des Unterschiedes zwischen Unmarshalling und Marshalling
- Verstehen von Unmarshalling mittels JAXB
- Verstehen von Marshalling mittels JAXB
- Verwenden der Validierung mittels JAXB

1.5.1 Unmarshalling einer XML-Nachricht

Unmarshalling ist der Prozess der Erstellung einer Objekthierachrie für ein XML-Dokument. Der Objektbaum repräsentiert danach den Inhalt des XML-Dokumentes. JAXB bietet die Möglichkeit, mit wenigen Befehlen diese Abbildung durchzuführen.

JAXB-Unmarshaller

```
JAXBContext jc = JAXBContext.newInstance("com.pe.gstsuche");
Unmarshaller u = jc.createUnmarshaller();
```

Die Abbildung zeigt, wie ein Unmarshaller erzeugt wird. Zuerst wird der JAXBContext benötigt, um eine bestimmte Struktur des XML-Dokumentes mappen zu können. Welches Schema zu verwenden ist, wird mittels Parameter der Methode newInstance angegeben. In dem Beispiel oben wird das Package der generierten Klassen aus der vorherigen Lerneinheit verwendet (siehe XJC). Die Methode createUnmarshaller des JAXBContextes liefert den benötigten Unmarshaller. Das Unmarshalling ist nun denkbar einfach, wie die folgende Abbildung zeigt.

JAXB-Unmarshalling

```
TOKEN token = (TOKEN)u.unmarshal(is);
```

Die Methode unmarshall liefert bereits den gewünschten Objektbaum. Der Parameter der Methode enthält den InputStream des Dokumentes. Ergebnis ist ein Objekt vom Typ TOKEN, das bereits im Kapitel Element Token erklärt wurde.

1.5.2 Marshalling einer XML-Nachricht

Marshalling bildet nun den Weg in die Gegenrichtung. Als Input dient diesmal ein Objektbaum, der mit den zu speichernden Daten befüllt ist. Ergebnis des Marshallings ist ein XML-Dokument mit den Inhalten des Objektbaumes.

JAXB-Marshaller

```
38  JAXBContext jc = JAXBContext.newInstance("at.gv.bev.gdb.webservice.data");
39  Marshaller m = jc.createMarshaller();
40  m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.FALSE);
```

Die Abbildung zeigt, wie ein Marshaller erzeugt wird. Zuerst wird der JAXBContext benötigt, um eine bestimmte Struktur des XML-Dokumentes mappen zu können. Welches Schema zu verwenden ist, wird mittels Parameter der Methode newInstance angegeben. Die Methode createMarshaller des JAXBContextes liefert den benötigten Marshaller. Dem Marshaller kann noch mitgeteilt werden, dass die Ausgabe "schön" formatiert erfolgen soll. Normalerweise würde der Marshaller alle XML-Daten innerhalb einer Zeile ausgeben. Mit der Konfiguration laut letzter Zeile der vorherigen Abbildung erzeugt der Marshaller ein XML-Dokument, in welchem die einzelnen Elemente eingerückt sind, und damit die Lesbarkeit auch für Menschen erheblich erhöht wird. Das Marshalling ist nun denkbar einfach, wie die folgende Abbildung zeigt.

JAXB-Marshalling

```
51          SOAPElement requestparam = SOAPFactory.newInstance().createElement("requestparam");
52          m.marshal(abgleichEigentuemer, requestparam);
```

Die Methode marshall erzeugt aus dem übergebenen Objektbaum (hier abgleichEigentuemer) ein XML-Element. Dieses Element wird an ein bestehendes, sogenanntes Rootelement (die Wurzel) nagehängt (hier requestparam). Dieses Rootelement muss vorher noch ausdrücklich angelegt werden (siehe Zeile davor).

1.5.3 Validierung mittels JAXB

Die oben beschriebenen Aktionen des Marshallings und Unmarshallings prüfen nicht, ob alle Regeln, die im Schema definiert sind, auch eingehalten werden. Hierunter fallen selbst einfache Regeln wie eine verpflichtende Befüllung eines Elementes. Diese Prüfung, auch Validierung genannt, kann jedoch bei JAXB angegeben werden. Folgende Abbildung zeigt ein Beispiel solch einer Validierung.

JAXB Validierung

```
SchemaFactory sf =
    SchemaFactory.newInstance(javax.xml.XMLConstants.W3C_XML_SCHEMA_NS_URI);
Schema schema =
    sf.newSchema(TokenServlet.class.getClassLoader().getResource("com/pe/gstsuche/interceptor.xsd"));
u.setSchema(schema);
```

Die Abbildung verdeutlicht, dass für die Validierung ein Schema benötigt wird. Dieses Schema entspricht dem oben angesprochenen Schema, welches für die Generierung der JAXB-Klassen verwendet wurde. Zuerst wird die SchemaFactory für XML-Schemas instanziert, welche dann im zweiten Schritt das Java-Gegenstück des Schemas erzeugt. Für diesen zweiten Schritt wird das XSD-File benötigt. Die hier gezeigte Verwendung des TokenServlets dient alleine der Möglichkeit, das XSD-File aus einem WAR-File (Web Archive) auslesen zu können. Da dies ein Teil des TokenServlets werden soll, ist diese Vorgehensweise nötig. Zuletzt wird noch das Schema dem Unmarshaller für die Validierung übergeben. Sobald der Unmarshaller ein Schema hat, wird beim Unmarshalling validiert. Das Vorgehen beim Marshalling ist identisch.

1.5.4 Arbeiten mit JavaBeans

Nach dem Unmarshalling sind die Inhalte des XML-Dokuments in Form von JavaBeans verfügbar. Zugriffe auf die Inhalte der JavaBeans erfolgen immer mittels Getter und Setter (vergleiche Was sind JavaBeans?). Eine Sonderstellung haben Attribute, die multiple Inhalte enthalten können. Diese werden bei JAXB als List-Objekte geführt. Ein Hinzufügen erfolgt mittels der Methode add und ein Auslesen mittels Methode get. Zu bedenken gilt, dass List-Attribute bei JAXB initial immer null sind. Somit muss vor dem ersten Aufruf der Methode add das Attribute mittels Setter mit einer Liste initialisiert werden. Andernfalls ist eine NullPointerException vorprogrammiert.

1.5.5 Zusammenfassung

Diese Lerneinheit befasst sich eingehender mit der Verarbeitung von XML-Dokumenten mittels JAXB. Dabei werden die Aktionen Marshalling und Unmarshalling im Zuge des Speicherns bzw. Lesens eines XML-Dokumentes behandelt. Unmarshalling befasst sich mit dem Auslesen des XML-Dokumentes und Darstellung als Objekthierarchie in Java und Marshalling betrifft den umgekehrten Weg, und zwar das Speichern eines Objektbaumes in Java als XML-Dokument. Bei beiden Richtungen kann der Inhalt validiert werden. Dies erfolgt einfach mittels Angabe des Schemas. Zum Abschluß werden noch etwaige Fallstricke bei der Verwendung von JavaBeans und JAXB erläutert.

1.6 Tokenverarbeitung im Servlet

In dieser Lerneinheit wird ein Servlet erstellt, welches die gesamte bisher beschriebene Tokenverarbeitung übernimmt. Darin enthalten ist der Zugriff auf den Interceptor, das Unmarshalling des Ergebnis-XMLs des Interceptors und das Abspeichern in der Session des Users.

Lernziele

- Definition des Servletsbegriffes
- Verstehen einer HTTP-Session

1.6.1 Was ist ein Servlet?

Servlets sind die Java-Antwort auf dynamische Web-Inhalte. Historisch betrachtet wurden CGI-Scripts (Common Gateway Interface) für die Erstellung von Webinhalten verwendet. Hierbei ruft der Benutzer eine URL auf, hinter der das CGI-Script gespeichert ist. Nach Ablauf des CGI-Scripts wird die Antwort an den Benutzer gesendet. Dies erfolgt meist in Form einer HTML-Seite. Das Problem bei CGI-Scripts liegt in der Serverarchitektur. Bei jedem Aufruf, der bei einem Interface ankommt, wird ein neuer Serverprozess für das CGI-Script gestartet. Dies limitiert die Höchstanzahl der gleichzeitigen Zugriffe auf die Webinhalte, da Prozessresourcen allgemein sehr eingeschränkt sind.
Um dieses Problem zu beheben, wurde die Java Servlet Technologie durch Sun entwickelt. Hierbei greift der Benutzer mittels URL auf einen sogenannten Servlet-Container zu. Dieser entscheidet aufgrund der URL, welches Servlet auszuführen ist. Servlets sind Javaprogramme, die das Interface HttpServlet implementieren müssen. Nach der Auswahl des Servlets nimmt der Container (auch Webcontainer oder Webserver genannt) einen freien Workerthread aus seinem Pool und beauftragt diesen, das Servlet abzuarbeiten. Sollte kein Thread frei sein, wird ein neuer angelegt und verwendet. Der Threadpool enthält Subthreads zum Prozess des Containers, wodurch lediglich der Prozess des Containers die gesamten Prozessressourcen benötigt. Subthreads haben ebenfalls einen gewissen Overhead, jedoch um einiges geringer als Prozesse und durch die Verwendung von Threadpools werden diese nicht immer neu angelegt. Dies erspart einige Zeit und damit auch CPU-Ressourcen. Ergebnis ist ein System, welches im Vergleich zu CGI-Scripts um ein Vielfaches mehr an Benutzern gleichzeitig bedienen kann .

1.6.2 Gründe für die Verwendung eines Servlets

Wie bereits unter Was ist ein Servlet? beschrieben, bietet die Servlet Technologie einen Geschwindigkeitsvorteil gegenüber herkömmlichen Scripts. Zusätzlich erhöht die Verwendung von Servlets die Stabilität des Systems auf der Serverseite, da die meisten Probleme in Bezug auf Stabilität auf mangelnde Ressourcen zurückzuführen sind.
Im Zuge der historischen Entwicklung wurden zusätzlich zu den Servlets auch sogenannte Java Server Pages (oder kurz JSPs) entwickelt. Diese ähneln PHP- oder ASP-Seiten und bieten die Möglichkeit, dynamische Inhalte direkt in den HTML-Code einzutragen. Bei Servlets muss der HTML-Code ebenfalls als Text auf den OutputStream geschrieben werden, was schnell zu einem unübersichtlichen Quellcode führt. Aufbauend auf vielen Webprojekten entwickelte sich die MVC-Architektur als sinnvollste für Webanwendungen. MVC steht für Model View Controller und beschreibt die Trennung zwischen dem Daten-/Objektmodell (Model), der Darstellungsschicht (View) und der Orchestrierung der einzelnen Komponenten (Controller). Umgelegt auf Servlets und JSPs bedeutet dies:

- Modell entspricht normalen Javaklassen

- View wird mittels JSPs erzeugt
- Controller wird durch Servlets bereitgestellt

Kurz gesagt kümmert sich jene Technologie um einen Teil des MVC-Frameworks, welche dazu am besten und einfachsten in der Lage ist. Wenn eine javalastige Implementierung nötig ist, sind Servlets zu verwenden (Controller). Um die HTML-Ausgabe zu erstellen werden JSPs verwendet, da damit die HTML-Tags am leichtesten zu erstellen sind.
Genau diesem Model folgend wurde der Prototyp für dieses eLearing Tutorial entwickelt. Servlets kümmern sich um das Tokenhanlding und die Weiterleitung von Anfragen, die einen Proxy benötigen. JSPs werden für die Darstellung der Webmaske mit dem Kartenfenster verwendet.

1.6.3 Zwischenspeichern des Tokens

Nachdem nun geklärt ist, dass ein Servlet mit dem Tokenhandling befasst ist, bleibt nur noch ein Problem zu lösen. Jeder Zugriff auf den Interceptor, um einen Token zu generieren, benötigt Zeit, die im Internet immer kostbar ist. Daher ist es ratsam, den Token einmal zu generieren und dann bis zum Ablauf der Benutzerinteraktion (Benutzersession) zu verwenden. Hierfür wird HttpSession bei Servlets verwendet, auf die auch JSPs zugreifen können. Somit meldet sich das Tokenservlet beim Interceptor an und speichert die Antwort mit dem Token in der HttpSession ab. Ab diesem Zeitpunkt wird nur noch der zwischengespeichete Token verwendet.

1.6.4 Beispiel für ein vollständiges Tokenservlet

Zusammenfassend soll nun der gesamte Quellcode eines Tokenservlets gezeigt und erklärt werden. Alle Punkte dieses Kurses werden nun gemeinsam zu einer Lösung zusammengebaut. Das TokenServlet erhält von der Konfiguration im Web.xml Username und Passwort für den Zugriff auf den Interceptor. Folgende Abbildung zeigt die Imports und die Variablen des Servlets.

Imports und Variablen

```
1  package com.pe;
2
3  import com.pe.gstsuche.TOKEN;
4  import java.io.*;
5
6  import java.net.InetSocketAddress;
7  import java.net.Proxy;
8  import java.net.SocketAddress;
9  import java.net.URL;
10 import java.net.URLConnection;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import javax.servlet.*;
14 import javax.servlet.http.*;
15 import javax.xml.bind.JAXBContext;
16 import javax.xml.bind.JAXBException;
17 import javax.xml.bind.Unmarshaller;
18 import javax.xml.validation.Schema;
19 import javax.xml.validation.SchemaFactory;
20 import org.xml.sax.SAXException;
21
22 /**
23  *
24  * @author pezerovic
25  */
26 public class TokenServlet extends HttpServlet
27 {
28
29     private static final long serialVersionUID = 7216429307747257138L;
30
31     public static final String TOKEN_KEY = "com.pe.gstsuche.TOKEN_KEY";
32
33     private Proxy proxy;
34
35     private String username;
36
37     private String password;
38
39     private ServletConfig config;
40
```

Die Init-Methode des Servlets wird vom Container bei der Instanziierung des Servlets mit den Parametern aus dem Web.xml aufgerufen. Die beiden Parameter Username und Passwort werden in Servletvariablen gespeichert. Normalerweise sollten keine Instanzvariablen bei Servlets verwendet werden, da die Servlets für mehrere Requests verwendet werden. Dadurch könnten andere Requests auf Daten des vorhergehenden Requests zugreifen. In diesem Fall handelt es sich um Konfigurationsdaten die ohnehin von statischer Natur sind und daher kein Sicherheitsrisiko darstellen. Zusätzlich wird noch der Proxy für Zugriffe auf den Interceptor definiert. Dies erfolgt optional, da nicht immer ein Proxy nötig ist, um auf das Internet zuzugreifen.

Init Methode

```
41     @Override
42     public void init(ServletConfig config) throws ServletException
43     {
44         super.init(config);
45         username = config.getInitParameter("username");
46         password = config.getInitParameter("password");
47
48         SocketAddress proxyAddress = new InetSocketAddress("10.2.1.101", 8080);
49         proxy = new Proxy(Proxy.Type.HTTP, proxyAddress);
50         this.config = config;
51     }
```

Die Methode Processrequest kümmert sich um die Verarbeitung der Anfrage des Benutzers und prüft als erstes, ob bereits ein Token in der HttpSession abgelegt ist. Wenn dies der Fall ist, muss nicht erneut eine Authentisierung am Interceptor stattfinden. Ansonsten wird die XML-URL des Interceptors aufgerufen und die XML-Antwort mittels JAXB auf JavaBeans abgebildet. Das resultierende Tokenobjekt wird in der HttpSession

abgelegt. Im Zuge des Unmarshallings wird noch der XML-Stream validiert. Nachdem nun ein Token in der HttpSession abgelegt ist, wird die Anfrage mittels RequestDispatcher an die zuständige View (gstsuche.jsp) weitergeleitet.

ProcessRequest Methode

```
    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        if(request.getSession().getAttribute(TOKEN_KEY) == null)
        {
            try
            {
                URL url =
                    new URL("http://217.13.180.218/GeoServer2/GatewayHttp?username=" + username + "&password=" +
                            password +
                            "&showinfoxml");

                URLConnection urlConnection = url.openConnection(proxy);
                InputStream is = urlConnection.getInputStream();


                JAXBContext jc = JAXBContext.newInstance("com.pe.gstsuche");
                Unmarshaller u = jc.createUnmarshaller();
                if(true)
                {
                    SchemaFactory sf =
                        SchemaFactory.newInstance(javax.xml.XMLConstants.W3C_XML_SCHEMA_NS_URI);
                    Schema schema =
                        sf.newSchema(TokenServlet.class.getClassLoader().getResource("com/pe/gstsuche/interceptor.xsd"));
                    u.setSchema(schema);
                }

                TOKEN token = (TOKEN)u.unmarshal(is);

                config.getServletContext().log(token.getID());

                request.getSession().setAttribute(TOKEN_KEY, token);
            }
            catch(SAXException ex)
            {
                Logger.getLogger(TokenServlet.class.getName()).log(Level.SEVERE, null, ex);
            }
            catch(JAXBException ex)
            {
                Logger.getLogger(TokenServlet.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        RequestDispatcher dispatcher = request.getRequestDispatcher("/gstsuche.jsp");
        if(dispatcher != null)
        {
            dispatcher.forward(request, response);
        }
    }
```

Den Abschluss bilden drei Standardmethoden eines Servlets. Die Methode doGet wird bei einem GET-Request aufgerufen (zum Beispiel bei der Eingabe einer URL in der Browseradressleiste), die Methode doPost, wenn die Anfrage mittels POST-Request gestellt wurde (zum Beispiel ein Submit einer Form). Beide Methoden reichen die Anfrage an die oben beschriebene Methode ProcessRequest weiter. Die Methode ServletInfo soll dazu dienen, eine kurze Beschreibung des Servlets zurückzugeben.

doGet und doPost Methoden

```
107
108     // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
109     /**
110      * Handles the HTTP <code>GET</code> method.
111      * @param request servlet request
112      * @param response servlet response
113      */
114     @Override
115     protected void doGet(HttpServletRequest request, HttpServletResponse response)
116         throws ServletException, IOException
117     {
118         processRequest(request, response);
119     }
120
121     /**
122      * Handles the HTTP <code>POST</code> method.
123      * @param request servlet request
124      * @param response servlet response
125      */
126     @Override
127     protected void doPost(HttpServletRequest request, HttpServletResponse response)
128         throws ServletException, IOException
129     {
130         processRequest(request, response);
131     }
132
133     /**
134      * Returns a short description of the servlet.
135      */
136     @Override
137     public String getServletInfo()
138     {
139         return "Short description";
140     }
141     // </editor-fold>
142 }
143
```

1.6.5 Zusammenfassung

In dieser Lerneinheit wird die Erfahrung der vorherigen Lerneinheiten in einem Servlet zusammengefasst. Einführend wird erklärt, was ein Servlet ist und wofür es verwendet wird. Danach wird die Verwendung des Servlets als Zwischenspeicher für Token erläutert, damit der langsame Zugriff auf den Interceptor nicht bei jedem Zugriff neu erfolgt. Des Weiteren muss die Web-Maske den Token kennen, um auf die Inhalte hinter dem Interceptor zugreifen zu können. Abschließend wird ein komplettes Beispiel für ein Servlet mit Tokenhandling gezeigt.

1.7 Zusammenfassung

Der erste Kurs befasst sich mit den Grundlagen, um auf Web Services mit geografischen Inhalten zugreifen zu können. Hierbei müssen Punkte wie Zugriffseinschränkungen, Token, Interceptoren, XML und Java Servlets erklärt werden.

Die erste Lerneinheit zeigt, wofür ein Token verwendet werden kann. Token sind Objekte, die ein exklusives Recht repräsentieren. Mittels Token können gewissen Informationen übergeben werden, unter anderem Session und Access Informationen. Session Token bezeichnen die Zusammengehörigkeit zweier Zugriffe und verbinden somit unabhängige Ereignisse (Zugriffe) zu einer zusammengehörigen Kommunikation. Access Token eröffnen die Möglichkeit, auf gesicherte Ressourcen zuzugreifen. Beim BEV werden Token als Access Token verwendet, mittels welcher auf Web Services zugegriffen werden darf. Ein Token ist jedoch nicht unbegrenzt gültig. Sogenannte Timeouts definieren den Zeitraum, in welchem ein Token Gültigkeit bewahrt.

Interceptoren werden als Zwischenhändler bei der Anfrageverarbeitung verwendet. Lerneinheit 2 zeigt, wofür diese Interceptoren verwendet werden können. Das BEV verwendet einen Interceptor, um die in der vorherigen Lerneinheit beschriebenen Token zu generieren. Des Weiteren prüft der Interceptor im Zuge jedes Zugriffes, ob der Token noch gültig ist. Hierfür muss vorher durch den Interceptor ein Token erstellt worden sein. Dies kann mittels HTML- oder XML-Zugriff erfolgen. Beide Varianten greifen auf die im Interceptor definierten User zu und prüfen Username und Passwort gegen die Datenbank des Usermanagements.

Für die Verarbeitung einer XML-Antwort des Interceptors sind einige Grundlagen nötig. Diese werden in Lerneinheit 3 behandelt und enthalten eine Definition und Erklärung der Struktur der Interceptor-Antwort, genaue Erklärung der einzelnen Elemente und einen Überblick über die Schemagrafik der XML-Antwort. Im Zuge dessen werden auch Grundlagen zum Thema Schema und XSD behandelt.

Die vierte Lerneinheit beschäftigte sich mit der Thematik der Verarbeitung von XML-Dokumenten in Java. Es wurden unterschiedliche Alternativen vorgestellt und JAXB als empfohlene Variante genauer erklärt. Hierbei wurde der Schema-Compiler XJC und die Verwendung der generierten JavaBeans behandelt.

Die Lerneinheit 5 befasst sich eingehender mit der Verarbeitung von XML-Dokumenten mittels JAXB. Dabei werden die Aktionen Marshalling und Unmarshalling im Zuge des Speicherns bzw. Lesens eines XML-Dokumentes behandelt. Unmarshalling befasst sich mit dem Auslesen des XML-Dokumentes und Darstellung als Objekthierarchie in Java und Marshalling betrifft den umgekehrten Weg, und zwar das Speichern eines Objektbaumes in Java als XML-Dokument. Bei beiden Richtungen kann der Inhalt validiert werden. Dies erfolgt einfach mittels Angabe des Schemas. Zum Abschluß werden noch etwaige Fallstricke bei der Verwendung von JavaBeans und JAXB erläutert.

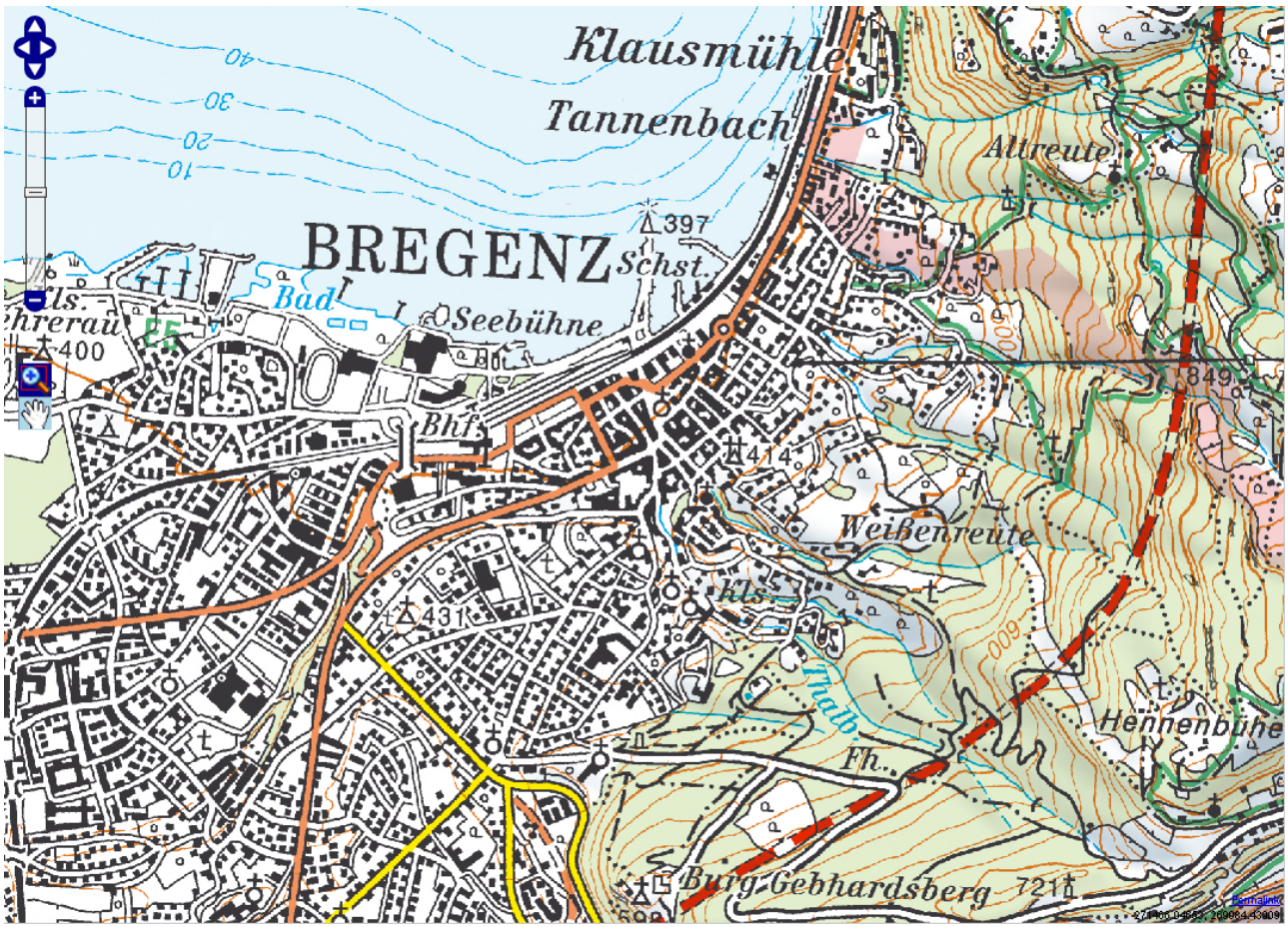
In der Lerneinheit 6 wird die Erfahrung der vorherigen Lerneinheiten in einem Servlet zusammengefasst. Einführend wird erklärt, was ein Servlet ist und wofür es verwendet wird. Danach wird die Verwendung des Servlets als Zwischenspeicher für Token erläutert, damit der langsame Zugriff auf den Interceptor nicht bei jedem Zugriff neu erfolgt. Des Weiteren muss die Web-Maske den Token kennen, um auf die Inhalte hinter dem Interceptor zugreifen zu können. Abschließend wird ein komplettes Beispiel für ein Servlet mit Tokenhandling gezeigt.

1.8 Abbildungsverzeichnis

2. Anzeige einer Karte

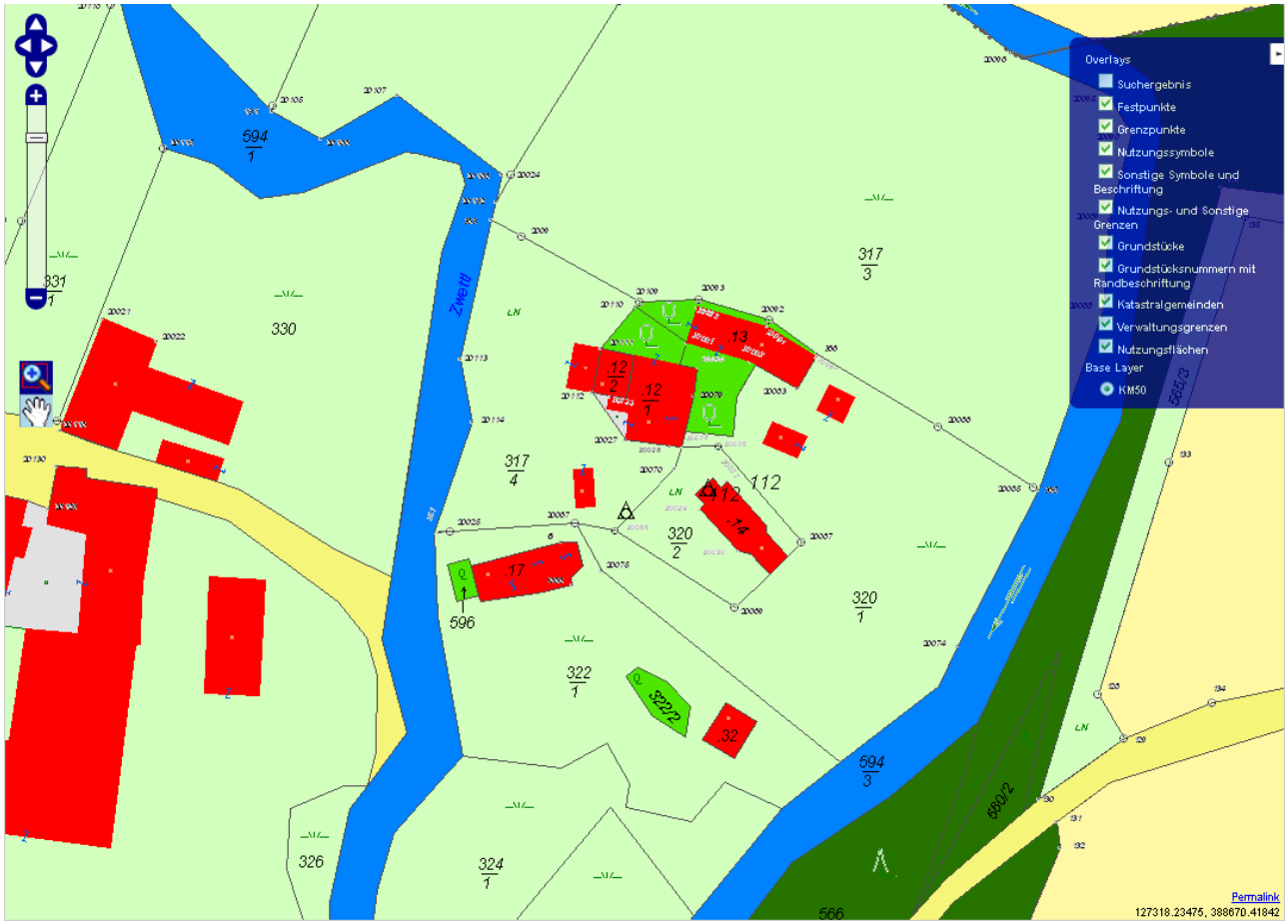
Diese eLesson vermittelt das Wissen, um eine Karte in einem Kartenfenster anzeigen zu können. Für die Darstellung wird die JavaScript-Bibliothek OpenLayers und für die Erstellung der Karte ein Web Map Service verwendet. Die angezeigte Karte enthält mehrere Layer, die einzeln aktivierbar sind. Das erwünschte Ergebnis dieser eLesson zeigen die folgenden zwei Abbildungen. Die erste zeigt eine Karte, bei der alle Layer, bis auf den Grundlayer (BaseLayer), deaktiviert sind.

Anzeige des BaseLayers



Wenn alle verfügbaren Layer aktiviert werden, sieht das Ergebnis wie folgt aus.

Anzeige aller Layer



Lernziele

- Verstehen der Definition eines Web Map Service
- Bennen und Erklären der Methoden eines Web Map Service
- Verstehen des Konzepts hinter OpenLayers
- Anzeige einer Map mit OpenLayers
- Zugriff auf ein WMS
- Anzeige einer Karte in der Map von OpenLayers
- Durchführen von Panning und Zooming in der Karte
- Verstehen und Erklären von Koordinatensystemen
- Kennen der verwendeten Koordinatensysteme in Österreich
- Konfigurieren der Koordinatensysteme in OpenLayers
- Konfigurieren des Zoomverhaltens in OpenLayers
- Verstehen der Grundlagen des Layerings
- Kennen der zur Verfügung stehenden Layer
- Gleichzeitiges Anzeigen mehrerer Layer in der Karte
- Transparent setzen eines Layers
- Verstehen des Unterschiedes zwischen normalen und Base-Layer
- Verstehen der Funktion von Tiles

2.1 Grundlagen zum Thema Web Map Service

In dieser Lerneinheit werden die Grundlagen zum Thema Web Map Service behandelt. Vertiefend werden die einzelnen Methoden besprochen, die ein Web Map Service dem Aufrufer anbietet.

Lernziele

- Verstehen der Definition eines Web Map Service
- Bennen und Erklären der Methoden eines Web Map Service

2.1.1 Was ist ein Web Map Service?

Ein Web Map Service (WMS) erzeugt dynamisch Karten von räumlichen Daten aus geografischen Informationen. WMS ist durch einen Open Geospatial Consortium-Standard (OGC) definiert und wird von mehreren Unternehmen als Service angeboten. Der Standard definiert eine Karte als die Darstellung geografischer Informationen in Form eines digitalen Bildes. Daher enthält eine Karte nicht die für ihre Erstellung notwendigen Daten, sondern nur das daraus resultierende Bild.
Das Open Geospatial Consortium aktualisiert den WMS Standard laufend. Die Version 1.3.0 liegt dem hier erzeugten Client zu Grunde. Bei der Implementierung muss auf die unterstützte Version durch die verwendete Software geachtet werden. Zum Beispiel wird in der Version 1.3.0 die Version 3 der Geographic Markup Language (GML) verwendet. Diese vertauscht die Reihenfolge von Längen- und Breitengrad im Vergleich zur Vorversion. Daher ist in diesem Bereich die GML nicht abwärtskompatibel.

2.1.2 Welche Methoden bietet ein WMS?

Der WMS-Standard definiert die Methoden, die ein OGC-konformes WMS anbieten muss. Die zu implementierenden Methoden sind:

- GetCapabilities
- GetMap
- GetFeatureInfo

Die Methode GetFeatureInfo ist als einzige optional und muss nicht implementiert sein, um OGC-Konformität zu erlangen. Diese Methoden sind für die Verwendung mittels Hypertext Transfer Protocol (HTTP) gedacht. Dadurch sind alle Methoden mittels sogenannter Key-Value-Pairs (KVP) aufrufbar. Folgender Ausschnitt zeigt ein Beispiel eines Aufrufes eines WMS:

Beispiel eines WMS Aufrufes

```
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-b8c22afd-91af-4898-bcb3-0e1342cb1919?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.3.0
```

2.1.3 Methode GetCapabilities

Der Zweck der verpflichtenden Methode GetCapabilities ist die Abfrage von Metadaten vom WMS-Server. Die Metadaten enthalten die unterstützten Parameter, vorhandenen Layer und allgemeine Kontaktinformationen. Die folgende Tabelle zeigt eine Übersicht über die möglichen Parameter.

Parameter der Methode GetCapabilities

Parameter	Pflichtfeld	Beschreibung
VERSION=version	Nein	Version der Anfrage. Kann auch eine ältere Version sein (zB.: 1.1.1 bei einem 1.3.0 WMS), solange das WMS auch abwärtskompatibel umgesetzt ist.
SERVICE=WMS	Ja	Typ des Services
REQUEST=GetCapabilities	Ja	Name der Methode
FORMAT=MIME_type	Nein	Format, in welchem die Antwort erwartet wird. Jeder WMS konforme Server muss zumindest text/ xml unterstützen. Wird ein nicht unterstütztes Format angegeben, hat der WMS-Server mit text/xml zu antworten.
UPDATESEQUENCE=string	Nein	Dient zur Steuerung des Cacheverhaltens und kann die Werte none, any, equal, lower und higher annehmen

2.1.4 Beispiel für die Methode GetCapabilities

Für ein besseres Verständis soll ein Beispiel für eine Anfrage und die zugehörige Antwort gegeben werden. Hierfür wird das WMS der Digitalen Katastralmappe (DKM) des BEV verwendet. Dieses dient auch in der Folge als Grundlage für die Implementierung.

Anfrage an das DKM-WMS

```
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-5d5b9d3b-c73e-4785-8e0f-734b88386874?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.1&FORMAT=text/xml
```

Die Anfrage enthält den oben beschriebenen Token. Eine mögliche Antwort soll im Folgenden besprochen werden. Die Antwort enthält als erstes Informationen zum Service, wie den Namen, die URL und die Kontaktinformationen der verantwortlichen Stelle/Person.

Service-Metadaten

```
<?xml version="1.0" encoding="UTF-8"?>
<WMT_MS_Capabilities version="1.1.1">
    <Service>
        <Name>DKM</Name>
        <Title>WMS-Service der DKM</Title>
        <Abstract>WMS-Service der DKM, Stand Juli 2008</Abstract>
        <KeywordList>
            <Keyword/>
        </KeywordList>
        <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-5d5b9d3b-c73e-4785-8e0f-734b8838
            xlink:type="simple"/>
        <ContactInformation>
            <ContactPersonPrimary>
                <ContactPerson>DI Stefan Klotz
                </ContactPerson>
                <ContactOrganization>BEV</ContactOrganization>
            </ContactPersonPrimary>
            <ContactPosition>Projekt Koordinator</ContactPosition>
            <ContactAddress>
                <AddressType/>
                <Address>Schiffamtsgasse 1-3</Address>
                <City>Wien</City>
                <StateOrProvince>Wien</StateOrProvince>
                <PostCode>A-1020</PostCode>
                <Country>Austria</Country>
            </ContactAddress>
            <ContactVoiceTelephone>+43-1-21110-0</ContactVoiceTelephone>
            <ContactFacsimileTelephone>+43-1-21110-0</ContactFacsimileTelephone>
            <ContactElectronicMailAddress>kundencenter@bev.gv.at</ContactElectronicMailAddress>
        </ContactInformation>
        <Fees/>
        <AccessConstraints/>
    </Service>
```

Weiters enthält die Antwort die Beschreibung der Methoden, die das WMS unterstützt. Folgende Abbildung zeigt einen Ausschnitt aus einer möglichen Antwort.

Methoden-Metadaten

```
<Capability>
    <Request>
        <GetCapabilities>
            <Format>application/vnd.ogc.wms_xml</Format>
            <DCPType>
                <HTTP>
                    <Get>
                        <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                            xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-5d5b9d3b-c73
                            xlink:type="simple"/>
                    </Get>
                </HTTP>
            </DCPType>
        </GetCapabilities>
        <GetMap>
            <Format>image/bmp</Format>
            <Format>image/jpeg</Format>
            <Format>image/tiff</Format>
            <Format>image/png</Format>
            <Format>image/gif</Format>
            <Format>image/svg+xml</Format>
            <DCPType>
                <HTTP>
                    <Get>
                        <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                            xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-5d5b9d3b-c73
                            xlink:type="simple"/>
                    </Get>
                </HTTP>
            </DCPType>
        </GetMap>
        <GetFeatureInfo>
            <Format>application/vnd.ogc.wms_xml</Format>
            <Format>text/xml</Format>
            <Format>text/html</Format>
            <Format>text/plain</Format>
            <DCPType>
                <HTTP>
                    <Get>
                        <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                            xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_DKM/BEVIntern-5d5b9d3b-c73
                            xlink:type="simple"/>
                    </Get>
                </HTTP>
            </DCPType>
        </GetFeatureInfo>
    </Request>
```

Ein WMS kann Karten in verschiedenen Koordinatensystemen liefern. Welche untertützt werden, steht ebenfalls in der hier beschriebenen Antwort. Für die Aufstellung der Koordinatensysteme werden EPSG Codes verwendet. EPSG Codes (European Petroleum Survey Group) sind standardisierte Codes, die eindeutig Koordinatensysteme beschreiben. Zusätzlich wird die maximale Ausdehnung der Layer in Längen- und Breitengraden sowie im Default-Koordinatensystem angegegen.

Koordinatensysteme

```
    <Layer>
        <Title>Layers</Title>
        <SRS>EPSG:4326</SRS>
        <SRS>EPSG:31254</SRS>
        <SRS>EPSG:31255</SRS>
        <SRS>EPSG:31256</SRS>
        <SRS>EPSG:31257</SRS>
        <SRS>EPSG:31258</SRS>
        <SRS>EPSG:31259</SRS>
        <SRS>EPSG:31287</SRS>
        <LatLonBoundingBox maxx="17.4083247821552" maxy="49.0172762433533" minx="9.46807492103876" miny="46.25341301384
        <BoundingBox SRS="EPSG:31255" maxx="298000.0" maxy="439000.0" minx="-298000.0" miny="131000.0"/>
```

Abschließend enthält die Antwort eine Aufstellung aller vorhandenen Layer. Hierbei werden Namen, Ausdehnung und der Default-Stil beschrieben.

Layer-Metadaten

```
99      <Layer queryable="1">
100         <Name>0</Name>
101         <Title>Katastralgemeinden</Title>
102         <Abstract>Katastralgemeinden</Abstract>
103         <SRS>EPSG:4326</SRS>
104         <SRS>EPSG:31255</SRS>
105         <LatLonBoundingBox maxx="17.235165" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318215"/>
106         <BoundingBox SRS="EPSG:31255" maxx="285624.5849" maxy="432800.1432" minx="-287675.8898" miny="137722.55"/>
107         <Style>
108             <Name>default</Name>
109             <Title>Katastralgemeinden</Title>
110             <LegendURL height="10" width="100">
111                 <Format>image/png</Format>
112                 <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
113                     xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
114                     xlink:type="simple"/>
115             </LegendURL>
116         </Style>
117         <ScaleHint max="INF" min="0.792000"/>
118     </Layer>
```

2.1.5 Methode GetMap

Die Methode GetMap liefert die fertiggestellte Karte und bietet folgende Parameter.

Parameter der Methode GetMap

Parameter	Pflichtfeld	Beschreibung
VERSION=1.3.0	Ja	Version der Anfrage
REQUEST=GetMap	Ja	Name der Methode
LAYERS=layer_list	Ja	durch Beistrich getrennte Liste der Layernamen, die in der Karte enthalten sein sollen
STYLES=style_list	Ja	durch Beistrich getrennte Liste der Styles je Layer
CRS=namespace:identifier	Ja	Name des Referenzsystems (Koordinatensystem)
BBOX=minx,miny,maxx,maxy	Ja	Koordinaten des umschließenden Rechtecks
WIDTH=output_width	Ja	Breite der Ausgabe in Pixel
HEIGHT=output_height	Ja	Höhe der Ausgabe in Pixel
HEIGHT=output_height	Ja	Ausgabeformat der Karte
TRANSPARENT=TRUE|FALSE	Nein	Transparente Anzeige des Hintergrunde der Karte (Default False)
BGCOLOR=color_value	Nein	Hexadezimaler Farbcode für den Hintergrund (Default 0xFFFFFF)
EXCEPTIONS=exception_format	Nein	Format, in welchem Fehler gemeldet werden sollen (Default XML)
TIME=time	Nein	Zeitpunkt für die Darstellung der Karte (bei historischen Karten)
ELEVATION=elevation	Nein	Angabe der Höhe über dem Meeresspiegel für die Darstellung (falls vorhanden)

2.1.6 Beispiel für die Methode GetMap

Die Methode GetMap liefert ausschließlich eine Grafik, die die Karte repräsentiert. Hier soll nun ein Beispiel für eine Anfrage und die daraus resultierende Grafik gezeigt werden.

Kartenabfrage

```
http://217.13.180.218/GeoServer2/Interceptor/Wms/ASP_KM50/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c?REQUEST=GetMap&SERVICE=WMS&VERSION=1.1.1&
LAYERS=0&FORMAT=image/png&SRS=EPSG:31255&BBOX=-290000,166000,-29000,293000&
WIDTH=823&HEIGHT=558&STYLES=default
```

Das Ergebnis der Anfrage zeigt die folgende Abbildung.

KM50 Von Tirol und Vorarlberg

2.1.7 Methode GetFeatureInfo

Die Methode GetFeature Info liefert zusätzliche Information zu einem bestimmten Punkt innerhalb einer bereits erstellten Karte. Wenn der Benutzer zum Beispiel innerhalb der angezeigten Karte mit dem Informations-Icon hineinklickt, dann liefert die Methode GetFeatureInfo die Informationen des Layers an der geklickten Stelle. Diese Methode ist in der Spezifikation optional und wird in diesem WebClient nicht verwendet. Weitere Informationen inklusive der Beschreibung der Parameter können der Spezifikation entnommen werden.

2.1.8 Zusammenfassung

Diese Lerneinheit enthält die Einführung in das Thema Web Map Service (WMS). Beginnend wird der Begriff und die Spezifikation eines WMS erläutert, gefolgt von der Beschreibung der einzelnen Methoden, die ein WMS enthalten kann. Jede der Methoden wird durch ein eigenes Beispiel zusätzlich veranschaulicht.

2.2 Anzeige einer Karte mittels OpenLayers

In dieser Lerneinheit wird erklärt, wie eine Karte mittels OpenLayers angezeigt werden kann. Dafür wird eine HTML-Seite erstellt, welche ein Kartenfenster enthält. Innerhalb dieses Fensters wird die Möglichkeit zu Pannen und zu Zoomen geboten. Das Ergebnis dieser Lerneinheit zeigt folgende Abbildung.

Anzeige einer Karte

Lernziele

- Verstehen des Konzepts hinter OpenLayers
- Anzeige einer Map mit OpenLayers
- Zugriff auf ein WMS
- Anzeige einer Karte in der Map von OpenLayers
- Durchführen von Panning und Zooming in der Karte

2.2.1 Was ist OpenLayers?

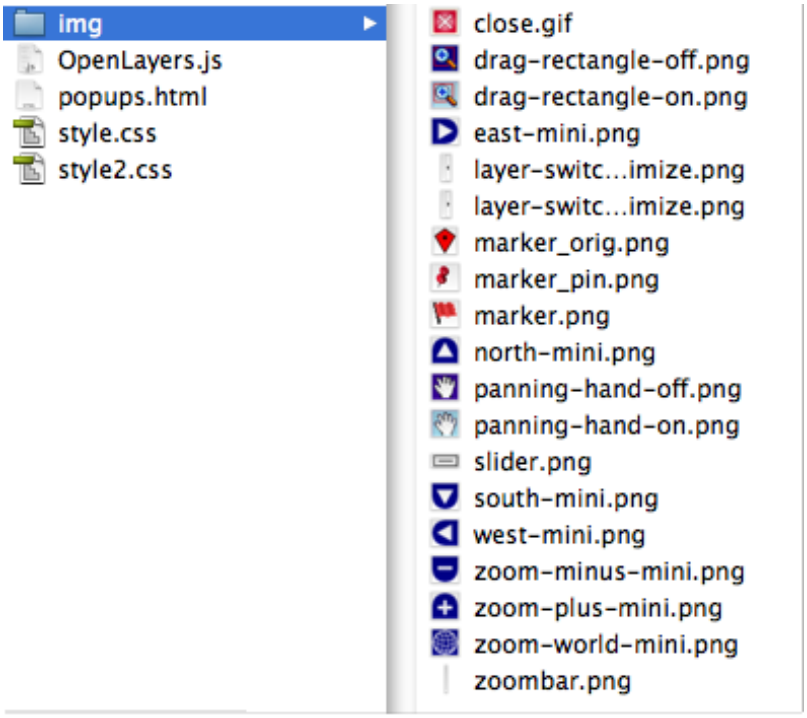
OpenLayers ist eine JavaScript Bibliothek, die es ermöglicht, dynamische Karten in eine HTML-Seite einzubetten. Die erste Version der JavaScript Bibliothek wurde von der Firma MetaCarta entwickelt. Im Jahr 2005 kam die Firma MetaCarta beim Besuch der "Where 2.0 Conference" auf die Idee, solch eine JavaScript Bibliothek zu erzeugen. Davor gab es nur Bibliotheken für die Darstellung, bei denen ein Submit-Response Zyklus nötig war. Das bedeutet dass, um eine neue Karte zu zeichnen, ein Submit gegen den Server nötig ist.

Somit wird die gesamte Seite immer wieder neu geladen. Erst durch Google Maps konnte gezeigt werden, dass auch asynchrone Aktualisierungen möglich waren. Später wurde solch ein Vorgehen als AJAX (Asynchronous JavaScript And XML) bekannt. Zusätzlich führte Google Maps sogenannte Tiles ein. Es musste nicht mehr die gesamte Karte neu geladen werden, sondern nur noch Teile. Diese Teile (tiles genannt) werden dann im Browser zusammengesetzt. Der Benutzer merkt den Unterschied vor allem beim initialen Laden der Karte. Während noch andere Tiles nachgeladen werden, sieht der Benutzer bereits geladene Tiles im Kartenfenster. Somit wirkt die Seite für den Benutzer viel schneller und aktiver, auch wenn die Zeit um alle Daten zu laden womöglich länger wird.
OpenLayers ist unter der BSD-Lizenz frei verfügbar. Die Weitereintwicklung wird von der OSGeo (Open Source Geospatial Foundation) vorangetrieben, wobei mindestens 2 neue Versionen pro Jahr veröffentlicht werden. Die Unterstützung für die Entwicklung ist durch Foren und sehr umfangreiche Beispiele gegeben. Ziel der Entwicklung ist die Unterstützung der OGC-Standards. Genau auf diese baut auch OpenStreetMap auf, welches als Grundlage OpenLayers einsetzt.

2.2.2 Anzeige einer Map

Um eine Map (Karte) in einer HTML-Seite anzuzeigen, sind einige Vorbedingungen zu erfüllen. Natürlich muss als erstes die JavaScript Bibliothek heruntergeladen werden. Zusätzlich werden von OpenLayers für die Darstellung des Kartenfenster zwei CSS Files und einige Grafiken benötigt. Folgende Grafik zeigt einen Überblick über die benötigten Artefakte und die dafür nötige Struktur innerhalb des Verzeichnisbaumes.

OpenLayers Verzeichnis



Innerhalb der HTML-Seite müssen nun die JavaScript Bibliothek und die CSS-Files referenziert werden.

Referenz auf OpenLayers Artefakte

```
<LINK href="openlayers/style.css" type="text/css" rel="stylesheet" />
<LINK href="openlayers/style2.css" type="text/css" rel="stylesheet" />
<LINK href="bevci/BEV_STYLES_GENERAL.css" type="text/css" rel="stylesheet" />
<script src="openlayers/OpenLayers.js"></script>
<style type="text/css">
    #controls {
        float: left;
        text-align: right;
    }
    .smallmap {
        border: #ccc 1px solid;
        width: 1074px;
        height: 770px;
    }

</style>
```

Die oben gezeigte Grafik enthält ein interessantes Detail, nämlich eine CSS-Stildefinition innerhalb des HTML-Textes. Diese definiert die Größe des Kartenfensters (.smallmap) und die Position der Bedienelemente für Zoom und Pan (#controls). Um nun eine Karte innerhalb der HTML-Seite anzeigen zu können, muss ein Objekt vom Typ OpenLayers.Map angelegt werden. Dieses stellt den Rahmen für die Karte dar. Alle geografischen Informationen werden innerhalb dieses Objektes verwaltet und angezeigt.

Map Objekt anlegen

```
    map = new OpenLayers.Map( 'map', {
        controls: [
            new OpenLayers.Control.PanZoomBar(),
            new OpenLayers.Control.MouseToolbar(),
            new OpenLayers.Control.LayerSwitcher({'ascending':false}),
            new OpenLayers.Control.Permalink(),
      //      Bug: zeigt immer NaN an
      //      new OpenLayers.Control.Scale(),
      //      new OpenLayers.Control.ScaleLine(),
            new OpenLayers.Control.Permalink('permalink'),
            new OpenLayers.Control.MousePosition()
        ]
        , zoom : 15
        , maxExtent: new OpenLayers.Bounds(-287638, 139391, 284887, 432527)
        , maxResolution: 550
        //, resolutions: [550, 450, 390, 310, 230, 150, 70, 60, 50, 40, 30, 20, 10, 2, 0.2, 0.02]
        , minResolution: 0.05
        , units: 'meters'
        , projection: "<%= epsgCode %>"
    });
```

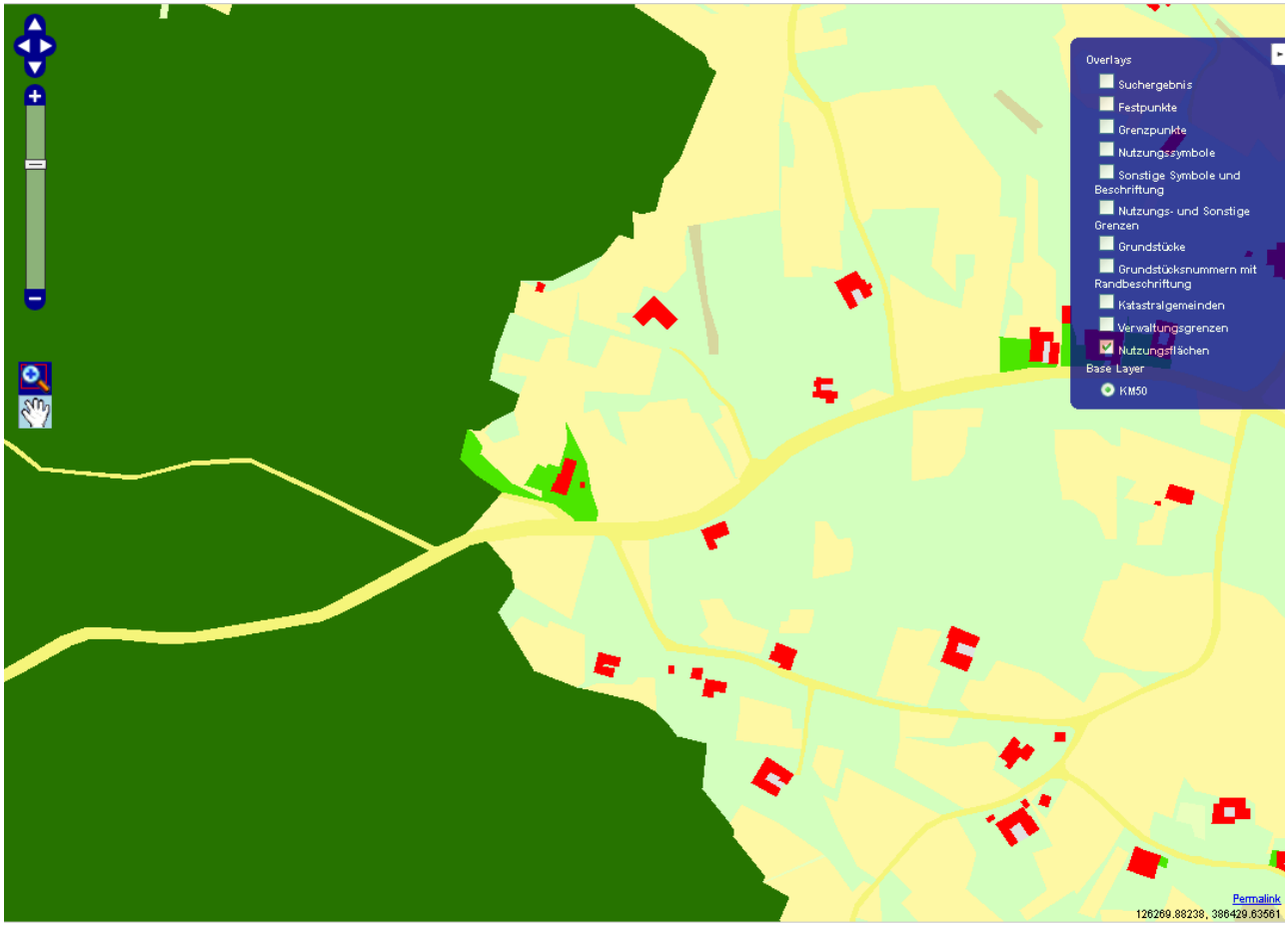
Die Grafik zeigt, wie ein Map Objekt angelegt werden kann. Zuerst wird ein Name für die Map definiert ('map'), da mehrere Kartenfenster je HTML-Seite existieren können. Zusätzlich werden alle Bedienelemente aktiviert, die benötigt werden. In diesem Fall sind dies:

- PanZoomBar: steuert die Anzeige der Zoom- und Pan-Bedienelemente (Anzeige einer Scrollbar für die Zoomstufe)
- MouseToolBar: aktiviert das Zoomen mittels Mausrades

- LayerSwitcher: aktiviert eine ausklappende Liste aller eingefügten Layer
- Permalink: bietet die Möglichkeit, mittels URL genau die gleiche Darstellung zu erlangen, die aktuell angezeigt wird (für Bookmarks sehr sinnvoll)
- MousePosition: dadurch wird die Anzeige der Koordinaten unterhalb des Mauszeigers aktiviert. Die aktuellen Koordinaten erscheinen im rechten unteren Fensterrand im Koordinatensystem der Karte

Folgende Abbildung zeigt ein Kartenfenster mit den oben beschriebenen Bedienelementen. Die PanZoomBar ist in der linken oberen Ecke, der LayerSwitcher in der rechten oberen Ecke, der Permalink und die MousePostion übereinander in der rechten unteren Ecke zu finden.

Kartenfenster mit Bedienelementen



Zustätzlich kann noch die Anzahl der Zoomschritte (zoom), die maximale Ausdehnung des Kartenfensters (maxExtent), die größte (maxResolution) und kleinste Zoomstufe (minResolution) und das Koordinatensystem (units, projection) angegeben werden. Nachdem das Map-Objekt angelegt wurde, muss der Ausgangspunkt definiert werden. Dieser bildet jenen Punkt, der beim ersten Anzeigen der Karte genau in der Mitte des Kartenfensters angezeigt werden soll. Des Weiteren kann noch die Breite des logischen Kartenfensters angegeben werden.

Map Zentrieren

```
map.setCenter(new OpenLayers.LonLat(xKoordinate, yKoordinate), zoom);
```

2.2.3 Zugriff auf ein WMS

Um sichtbare Elemente auf der Map darzustellen, werden sogenannte Layer angelegt. Ein Layer bildet eine Art Ebene innerhalb der Karte. Eine dieser Ebenen ist die BaseMap, also die unterste Ebene mit der grundlegenden Karte. Diese wird im Fall des WebClients durch ein WMS geliefert und stellt die ÖK50 dar. Folgende Abbildung zeigt, wie auf das WMS zugegriffen wird.

Zugriff auf die ÖK50

```
layer= new OpenLayers.Layer.WMS(
  "KM50",
  "<%= km50Url %>",
  {layers: '0', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png'},
  {buffer:2}
);
```

Der Zugriff erfolgt, indem ein Objekt vom Typ OpenLayers.Layer.WMS instanziiert wird, welches die weiteren Aktivitäten bezüglich Abfrage und Anzeige übernimmt. Parameter des Konstrukturs sind:

- Name des Layers (KM50)
- URL des WMS (URL auf das km50-WMS des Interceptors; siehe XML-Antwort des Interceptors)
- Name des Layers, der vom WMS abgefragt werden soll (layer '0'; siehe GetCapabilities)
- CRS (Coordinate Reference System; sollte dem Koordinatensystem vom Objekt Map entsprechen)
- Format der Antwort des WMS (png)
- Buffer, der angibt, wieviele Tiles um die sichtbare Anzeige herum zusätzlich geladen werden sollen, um ein etwaiges Pan zu beschleunigen (2)

Ein so angelegter Layer kümmert sich selbständig um Anzeige und Nachladen von den benötigten Tiles. Den aktuellen Kartenausschnitt erhält das Objekt WMS vom Objekt Map.

2.2.4 Anzeige der Karte in der Map

Wenn der Layer mit dem Zugriff auf das WMS definiert ist, muss dieser nur noch zur Map hinzugefügt werden. Dies erfolgt mit folgendem Befehl:

Hinzufügen eines Layers

```
map.addLayer(layer);
```

Daraufhin wird die ÖK50 automatisch im Kartenfenster angezeigt. Zoom und Pan der Karte funktionieren, ohne dass eine zusätzliche Zeile programmiert werden muss.

Anzeige der ÖK50

2.2.5 Panning der Karte

Panning kann mit dem Schwenken der Karte übersetzt werden. Hierbei nimmt der Benutzer die Karte und verschiebt den Kartenausschnitt. Dies erfogt bei OpenLayers durch Auswahl des Pan-Bedienelementes. Danach kann durch Klick mit der Maus ein Pan durchgeführt werden. Solange die linke Maustaste gedrückt gehalten wird, verschiebt sich der Karteninhalt unterhalb des Mauszeigers. Folgende Abbildung zeigt die BaseMap mit aktiviertem Pan.

Aktiviertes Pan-Bedienelement

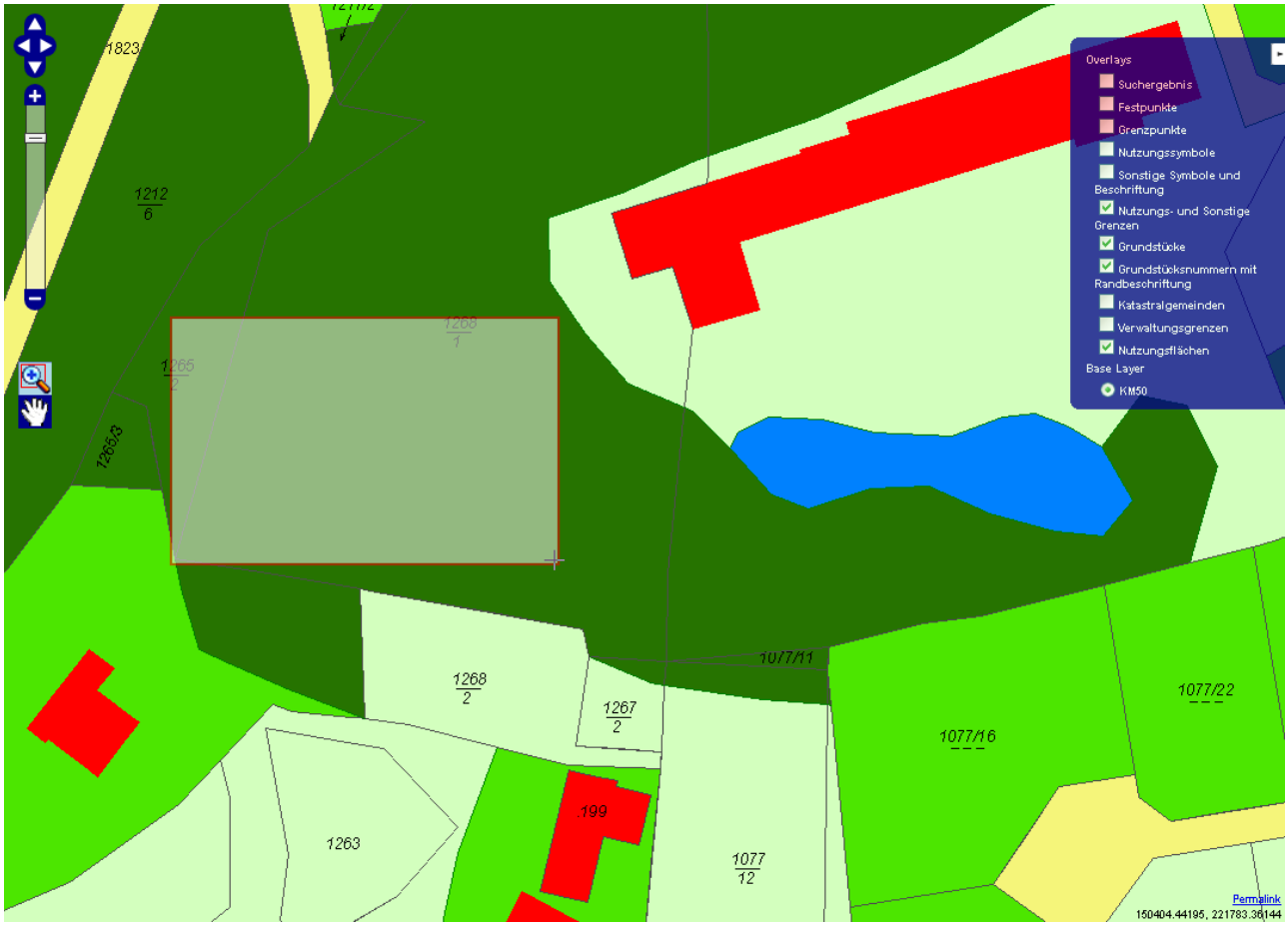
2.2.6 Zoomen der Karte

Zoomen stellt eine Veränderung des ausgewählten Kartenausschnittes durch vergrößern oder verkleinern dar. Bei OpenLayers erfolgt dies durch Auswahl des Zoom-Bedienelementes und aufziehen der Zoom-Box. Die Zoom-Box stellt jenen Kartenausschnitt dar, der nach dem Vergrößern die gesamte Karte abdecken soll. Es wird somit der Rand des nächsten Kartenfensters in die Karte eingezeichnet. Das Verkleinern erfolgt mittels Klick auf das Minuszeichen der ZoomBar. Ebenso kann durch Klick auf das Pluszeichen der ZoomBar die Karte vergrößert werden. Folgende Abbildung zeigt die BaseMap mit aktiviertem Zoom.

Aktiviertes Zoom-Bedienelement

Beim Zoomen kann entweder durch Doppelklick die Karte um eine Stufe vergrößert werden, oder es wird die oben beschriebene Zoom-Box aufgezogen. Die zweite Variante zeigt nachfolgende Grafik.

Aufgezogene Zoom-Box



2.2.7 Zusammenfassung

In dieser Lerneinheit werden grundlegende Kartenfunktionen erläutert. Zuerst erfolgt eine kurze Einführung in OpenLayers und die Erklärung, wie innerhalb von OpenLayers eine Karte dargestellt wird. Danach folgt die Information, was in OpenLayers eine Map darstellt und zum Unterschied dazu, was ein WMS-Layer ist. Eine Map bildet die Referenz auf das Kartenfenster in OpenLayers und ein WMS-Layer entspricht einer Ebene innerhalb der Karte, die eine Grafik enthält, welche durch ein WMS erzeugt wird. Innerhalb der Map bietet OpenLayers die Möglichkeit, die Karte zu pannen und zu zoomen.

2.3 Konfiguration der Mapdarstellung

In dieser Lerneinheit werden die Einstellungsmöglichkeiten der Map in OpenLayers erklärt. Weiters werden die in Österreich gebräuchlichen Koordinatensysteme vorgestellt.

Lernziele

- Verstehen und Erklären von Koordinatensystemen
- Kennen der verwendeten Koordinatensysteme in Österreich
- Konfigurieren der Koordinatensysteme in OpenLayers
- Konfigurieren des Zoomverhaltens in OpenLayers

2.3.1 Was ist ein Koordinatensystem?

Ein Koordinatensystem stellt die Beschreibung eines Punktes und dessen Lage auf der Erde dar. Die einfachste Variante ist die Referenz mittels Längen- und Breitengraden. Dies stellt auch die Standardeinstellung von OpenLayers dar. Wenn kein Koordinatensystem konfiguriert wird, müssen alle Koordinaten in WGS84 angegeben werden. WGS84 (World Geodetic System 1984) wird über einen Referenzellipsoid, die Abweichung zur Erdoberfläche (dem Geoid) und Referenzstationen definiert. Weite Verbreitung findet das Koordinatensystem seit der Einführung von GPS (Global Position System), welches WGS84 als geodätische Grundlage verwendet. Im Bereich der Geoinformationssysteme gibt es zwei Referenzsysteme, die das WGS84 als Basis haben.

- EPSG 4326: Angabe der Koordinaten mittels Breiten- und Längengrad
- CRS 84: Angabe der Koordinaten mittels Längen- und Breitengrad

Wie ersichtlich liefern beide Koordinatensysteme das gleiche Ergebnis. Leider sind die Angaben der Längen- und Breitengrade vertauscht. Bis zur Version 1.1.1 der GML Spezifikation musste ein OGC-konformes WMS/ WFS die Daten in EPSG 4326 liefern. Seit der Version 1.3.0 ist nun CRS 84 als Standard definiert. Dies führt zu großen Problemen, die bei der Umsetzung von WMS/WFS Clients auftreten.

2.3.2 Koordinatensysteme in Österreich

Um die Verzeichnung der Punkte im äußeren Bereich der Staatsfläche zu minimieren wurden in Österreich eigene Koordinatensysteme definiert. Da Österreich eine zu große Ausdehnung entlang des Breitengrades hat, werden drei Koordinatensysteme für die Darstellung Österreichs verwendet. Folgende Auflistung zeigt die unterschiedlichen Koordinatensysteme, die in Österreich in Verwendung sind:

Koordinatensystem in Österreich

EPSG code	Projection name	Base GeogCRS	Map Projection	Longitude Orig	Latitude Orig	Prime Meridian	false easting	false northing
3416	ETRS89 / Austria Lambert	ETRS89	Austria Lambert	13d 20m	47d 30m	Greenwich	400000	400000
31251	MGI (Ferro) / Austria GK West Zone	MGI (Ferro)	Austria Gauss-Kruger West Zone	28d	0d	Ferro	0	-5000000
31252	MGI (Ferro) / Austria GK Central Zone	MGI (Ferro)	Austria Gauss-Kruger Central Zone	31d	0d	Ferro	0	-5000000
31253	MGI (Ferro) / Austria GK East Zone	MGI (Ferro)	Austria Gauss-Kruger East Zone	34d	0d	Ferro	0	-5000000
31254	MGI / Austria GK West	MGI	Austria Gauss-Kruger West	10d 20m	0d	Greenwich	0	-5000000
31255	MGI / Austria GK Central	MGI	Austria Gauss-Kruger Central	13d 20m	0d	Greenwich	0	-5000000
31256	MGI / Austria GK East	MGI	Austria Gauss-Kruger East	16d 20m	0d	Greenwich	0	-5000000
31257	MGI / Austria GK M28	MGI	Austria Gauss-Kruger M28	10d 20m	0d	Greenwich	150000	-5000000
31258	MGI / Austria GK M31	MGI	Austria Gauss-Kruger M31	13d 20m	0d	Greenwich	450000	-5000000
31259	MGI / Austria GK M34	MGI	Austria Gauss-Kruger M34	16d 20m	0d	Greenwich	750000	-5000000
31281	MGI (Ferro) / Austria West Zone	MGI (Ferro)	Austria West Zone	28d	0d	Ferro	0	0
31282	MGI (Ferro) / Austria Central Zone	MGI (Ferro)	Austria Central Zone	31d	0d	Ferro	0	0
31283	MGI (Ferro) / Austria East Zone	MGI (Ferro)	Austria East Zone	34d	0d	Ferro	0	0
31284	MGI / Austria M28	MGI	Austria M28	10d 20m	0d	Greenwich	150000	0
31285	MGI / Austria M31	MGI	Austria M31	13d 20m	0d	Greenwich	450000	0
31286	MGI / Austria M34	MGI	Austria M34	16d 20m	0d	Greenwich	750000	0
31287	MGI / Austria Lambert	MGI	Austria Lambert	13d 20m	47d 30m	Greenwich	400000	400000
31288	MGI (Ferro) / M28	MGI (Ferro)	Austria zone M28	28d	0d	Ferro	150000	0
31289	MGI (Ferro) / M31	MGI (Ferro)	Austria zone M31	31d	0d	Ferro	450000	0
31290	MGI (Ferro) / M34	MGI (Ferro)	Austria zone M34	34d	0d	Ferro	750000	0

Um die Implementierung der Webapplikation zu vereinfachen, wird vom WMS und WFS ganz Österreich im Koordinatensystem EPSG:31255 geliefert. Ansonsten müsste die Webapplikation je nach dem Gebiet von Österreich, ein anderes Koordinatensystem definieren.

2.3.3 Koordinatensystem in OpenLayers konfigurieren

In Openlayers wird ein Koordinatensystem beim Anlegen des Objektes OpenLayers.Map angegeben. OpenLayers unterstützt eine Vielzahl von Koordinatensystemen, jedoch nicht die Österreichischen. Daher muss das Koordinatensystem definiert werden. Folgende Abbildung zeigt nochmals das Anlegen einer Map mittels OpenLayers:

Koordinatensystem konfigurieren

```
map = new OpenLayers.Map( 'map', {
    controls: [
        new OpenLayers.Control.PanZoomBar(),
        new OpenLayers.Control.MouseToolbar(),
        new OpenLayers.Control.LayerSwitcher({'ascending':false}),
        new OpenLayers.Control.Permalink(),
//      Bug: zeigt immer NaN an
//      new OpenLayers.Control.Scale(),
//      new OpenLayers.Control.ScaleLine(),
        new OpenLayers.Control.Permalink('permalink'),
        new OpenLayers.Control.MousePosition()
    ]
    , zoom : 15
    , maxExtent: new OpenLayers.Bounds(-287638, 139391, 284887, 432527)
    , maxResolution: 550
    //, resolutions: [550, 450, 390, 310, 230, 150, 70, 60, 50, 40, 30, 20, 10, 2, 0.2, 0.02]
    , minResolution: 0.05
    , units: 'meters'
    , projection: "<%= epsgCode %>"
});
```

Durch folgende Parameter wird das Koordinatensystem definiert:

- projection: spezifiziert den Namen des Koordinatensystems (EPSG:31255)
- units: gibt an, welche Einheiten verwendet werden (meters)

- maxExtent: definiert die maximalen Werte des Kartenfensters und somit der Koordinaten, indem die Koordinaten des umschließenden Rechteckes mitgegeben werden (links unten nach rechts oben; [-287638, 139391]; [284887, 432527])

2.3.4 Konfiguration der Zoommöglichkeiten

Die Karte in OpenLayers ist, wie bereits erwähnt, vergrößer- und verkleinerbar. In welchem Rahmen dies erfolgen kann, sollte definiert werden. Ohne diese Definition läßt OpenLayers das Verkleinern bis auf jenen Punkt zu, wo die gesamte Erdkugel drei mal im Kartenfenster dargestellt wird. Die Einstellung erfolgt mittels folgender Parameter:

- maxResolution: Information über die maximale Größe des Kartenausschnittes; Die Angabe ist schwer verständlich und definiert, wieviele Einheiten ein Pixel der Grafik maximal referenzieren darf. In dem untenstehenden Ausschnitt bedeutet die Angabe von 550, dass ein Pixel der Grafik maximal 550 Metern entsprechen darf. Bei einer Kartenbreite von 1074 Pixel entspricht das 590 km Breite der Grafik (Luftlinie). Dies reicht aus, um ganz Österreich darzustellen.
- minResolution: die entsprechende Angabe für das untere Ende des Zoombereiches. Information über die minimale Größe des Kartenausschnittes. Die Angabe entspricht wiederum 0.05 Meter je Pixel der Grafik. Bei einer Kartenbreite von 1074 Pixel entspricht das 53 Metern.
- zoom: definiert, wieviele Schritte für das Zoomen vorhanden sein sollen (hier 15 Stufen der Vergrößerung)

Zoomkonfiguration

```
map = new OpenLayers.Map( 'map', {
    controls: [
        new OpenLayers.Control.PanZoomBar(),
        new OpenLayers.Control.MouseToolbar(),
        new OpenLayers.Control.LayerSwitcher({'ascending':false}),
        new OpenLayers.Control.Permalink(),
//      Bug: zeigt immer NaN an
//      new OpenLayers.Control.Scale(),
//      new OpenLayers.Control.ScaleLine(),
        new OpenLayers.Control.Permalink('permalink'),
        new OpenLayers.Control.MousePosition()
    ]
    , zoom : 15
    , maxExtent: new OpenLayers.Bounds(-287638, 139391, 284887, 432527)
    , maxResolution: 550
    //, resolutions: [550, 450, 390, 310, 230, 150, 70, 60, 50, 40, 30, 20, 10, 2, 0.2, 0.02]
    , minResolution: 0.05
    , units: 'meters'
    , projection: "<%= epsgCode %>"
});
```

2.3.5 Konfiguration der Bedienelemente

Die angezeigte Karte bietet unterschiedliche Bedienelemente. Diese müssen bei der Instanziierung des Mapobjektes angegeben werden. Die Angabe erfolgt mittels Parameter 'controls'. Eine genaue Definition kann unter Anzeigen einer Map gefunden werden.

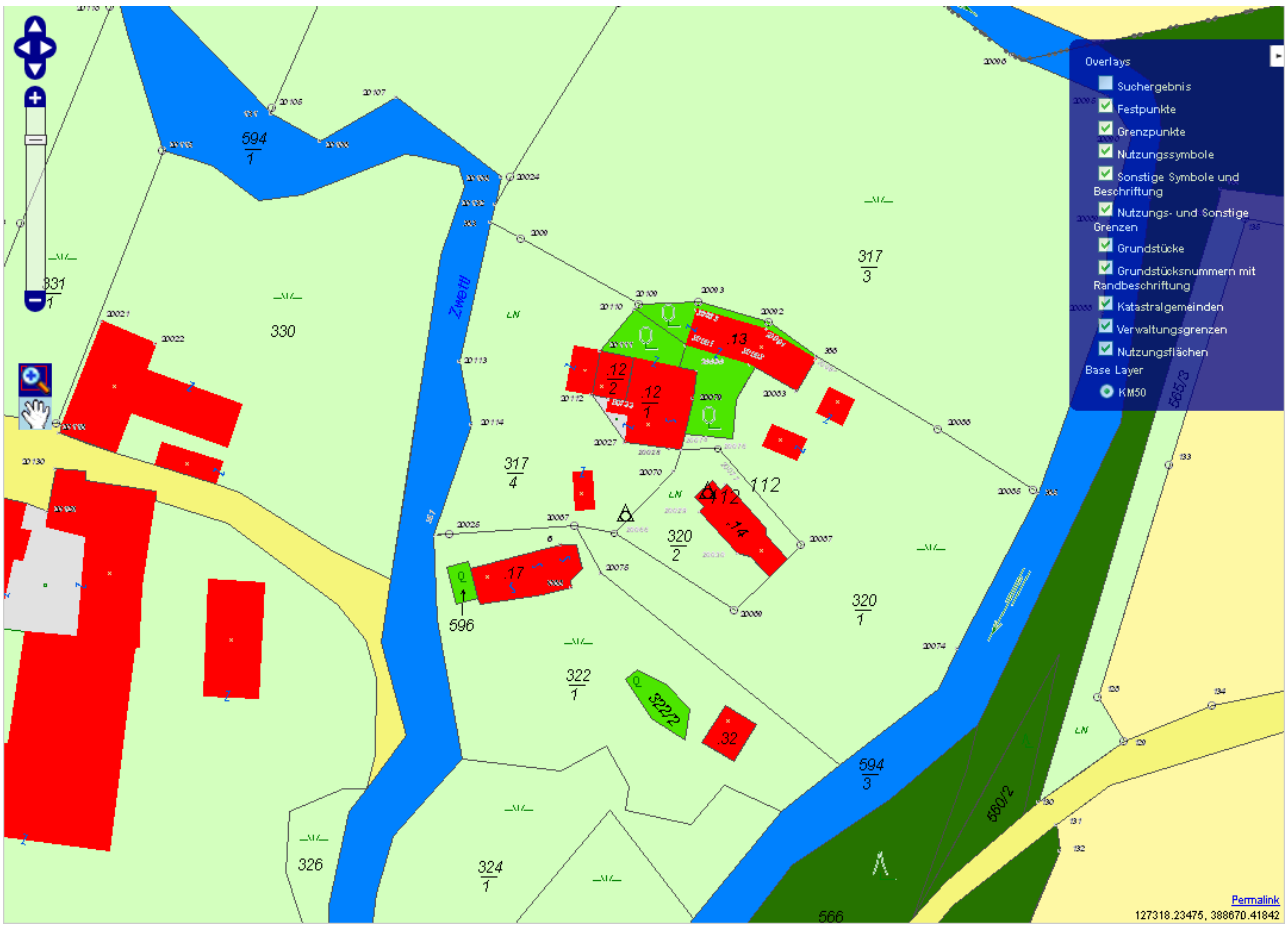
2.3.6 Zusammenfassung

Um eine Karte im Kartenfenster von OpenLayers darzustellen, muss ein Koordinatensystem für die Karte definiert werden. Diese Lerneinheit zeigt, wie in OpenLayers ein Koordinatensystem konfiguriert werden kann. Sollte kein Koordinatensystem angegeben werden, verwendet OpenLayers automatisch die Angabe der Koordinaten in Längen- und Breitengraden. Im Anschluß werden die in Österreich verwendeten Koordinatensysteme vorgestellt. Abschließend wird erklärt, wie die einzelnen Bedienelemente und die Zoommöglichkeit in OpenLayers konfiguriert werden kann.

2.4 Anzeige von mehreren Layer in der Karte

In dieser Lerneinheit wird erklärt, wie Layer in eine Karte hinzugefügt werden können. Die BaseMap wurde bereits in der vorherigen Lerneinheit zur Karte hinzugefügt. Es ist jedoch möglich, mehr als einen Layer in der Karte anzuzeigen. Wie dies funktioniert, erklärt diese Lerneinheit. Folgende Grafik zeigt das Ergebnis:

Anzeige aller Layer



Lernziele

- Verstehen der Grundlagen des Layerings
- Kennen der zur Verfügung stehenden Layer
- Gleichzeitiges Anzeigen mehrerer Layer in der Karte
- Transparent setzen eines Layers
- Verstehen des Unterschiedes zwischen normalen und Base-Layer
- Verstehen der Funktion von Tiles

2.4.1 Was ist ein Layer?

Die Layertechnik ist ein weitverbreitetes Verfahren, um Grafikdateien zu strukturieren. Dabei werden die Grafiken in mehreren Ebenen aufgebaut, wobei die erzeugte Grafik die Summe aller sichtbaren Ebenen darstellt. Ebenen können daher auch ausgeblendet werden, um die resultierende Grafik zu verändern. Wenn nun zwei Layer übereinander gelegt werden, muss natürlich der obere Layer gewisse transparente Teile enthalten.

Ansonsten würde der Inhalt des oberen Layers den unteren komplett abdecken. Bei Geoinformationssystemen wird ebenfalls die Layertechnik angewendet, um Karten zu erzeugen. Ein WMS kann beliebig viele Layer anbieten, die der Client dann nach Belieben kombinieren kann.

2.4.2 Welche Layer stehen zur Verfügung?

Das BEV hat für dieses Tutorial zwei WMS zur Verfügung gestellt, die eine Vielzahl an Layern bieten. Das erste WMS heißt ASP_KM50 und bietet nur einen Layer (0). Dieser Layer ist wie folgt definiert:

Layer ÖK50

```
<Layer>
  <Name>0</Name>
  <Title>KM50</Title>
  <Abstract>KM50</Abstract>
  <SRS>EPSG:4326</SRS>
  <SRS>EPSG:31255</SRS>
  <LatLonBoundingBox maxx="17.4083247821552" maxy="49.0172762433533" minx="9.46807492103876" miny="46.2534130138412" />
  <BoundingBox SRS="EPSG:31255" maxx="298000.0" maxy="439000.0" minx="-298000.0" miny="131000.0" />
  <Style>
    <Name>default</Name>
    <Title>KM50</Title>
    <LegendURL height="87" width="100">
      <Format>image/png</Format>
      <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:href=
"http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_KM50/BEVIntern-d62bf6e3-6994-44e4-9135-50bd98b096d5/default0.png"
        xlink:type="simple" />
    </LegendURL>
  </Style>
  <ScaleHint max="INF" min="3.960000" />
</Layer>
```

Das zweite WMS bietet elf verschiedene Layer an. Diese sollen nun im Anschluß vorgestellt werden. Layer '0' enthält die Features der Katastralgemeinden. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Katastralgemeinden

```
<Layer queryable="1">
    <Name>0</Name>
    <Title>Katastralgemeinden</Title>
    <Abstract>Katastralgemeinden</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.235165" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318215"/>
    <BoundingBox SRS="EPSG:31255" maxx="285624.5849" maxy="432800.1432" minx="-287675.8898" miny="137722.55"/>
    <Style>
        <Name>default</Name>
        <Title>Katastralgemeinden</Title>
        <LegendURL height="10" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="INF" min="0.792000"/>
</Layer>
```

Layer '1' enthält die Features der Nutzungsflächen. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Nutzungsflächen

```
<Layer queryable="0">
    <Name>1</Name>
    <Title>Nutzungsflaechen</Title>
    <Abstract>Nutzungsflaechen</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.235165" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318215"/>
    <BoundingBox SRS="EPSG:31255" maxx="285624.5849" maxy="432800.1432" minx="-287675.8898" miny="137722.55"/>
    <Style>
        <Name>default</Name>
        <Title>Nutzungsflaechen</Title>
        <LegendURL height="351" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="3.960000" min="1.980000"/>
</Layer>
```

Layer '2' enthält die Features der Grundstücksnummern mit Randbeschriftungen. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Grundstücksnummern mit Randbeschriftungen

```
<Layer queryable="0">
    <Name>2</Name>
    <Title>Grundstuecksnummern mit RandBeschriftung</Title>
    <Abstract>Grundstuecksnummern mit RandBeschriftung</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.235165" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318215"/>
    <BoundingBox SRS="EPSG:31255" maxx="285624.5849" maxy="432800.1432" minx="-287675.8898" miny="137722.55"/>
    <Style>
        <Name>default</Name>
        <Title>Grundstuecksnummern mit RandBeschriftung</Title>
        <LegendURL height="28" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="1.980000" min="0.000000"/>
</Layer>
```

Layer '3' enthält die Features der Grundstücksnummern. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Grundstücksnummern

```
<Layer queryable="0">
    <Name>3</Name>
    <Title>Grundstuecksnummern</Title>
    <Abstract>Grundstuecksnummern</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.233751" maxy="49.0172762433533" minx="9.46807492103876" miny="46.32767"/>
    <BoundingBox SRS="EPSG:31255" maxx="285528.9848" maxy="432642.6263" minx="-287560.1409" miny="138770.241"/>
    <Style>
        <Name>default</Name>
        <Title>Grundstuecksnummern</Title>
        <LegendURL height="28" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="0.795960" min="0.000000"/>
</Layer>
```

Layer '4' enthält die Features der Grundstücksgrenzen. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Grundstücke

```
<Layer queryable="1">
    <Name>4</Name>
    <Title>Grundstuecke</Title>
    <Abstract>Grundstuecke</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.235165" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318215"/>
    <BoundingBox SRS="EPSG:31255" maxx="285624.5849" maxy="432800.1432" minx="-287675.8898" miny="137722.55"/>
    <Style>
        <Name>default</Name>
        <Title>Grundstuecke</Title>
        <LegendURL height="13" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="9.900000" min="0.000000"/>
</Layer>
```

Layer '5' enthält die Features der Verwaltungsgrenzen. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Verwaltungsgrenzen

```
<Layer queryable="0">
    <Name>5</Name>
    <Title>Verwaltungsgrenzen</Title>
    <Abstract>Verwaltungsgrenzen</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.235165" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318215"/>
    <BoundingBox SRS="EPSG:31255" maxx="285624.5849" maxy="432800.1432" minx="-287675.8898" miny="137722.55"/>
    <Style>
        <Name>default</Name>
        <Title>Verwaltungsgrenzen</Title>
        <LegendURL height="69" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="INF" min="0.000000"/>
</Layer>
```

Layer '6' enthält die Features der Nutzungs- und sonstigen Grenzen. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Nutzungs- und sonstige Grenzen

```
<Layer queryable="0">
    <Name>6</Name>
    <Title>Nutzungs- und Sonstige Grenzen</Title>
    <Abstract>Nutzungs- und Sonstige Grenzen</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.224915" maxy="49.0172762433533" minx="9.46807492103876" miny="46.33321"/>
    <BoundingBox SRS="EPSG:31255" maxx="284886.4723" maxy="432526.9088" minx="-287637.2579" miny="139390.89"/>
    <Style>
        <Name>default</Name>
        <Title>Nutzungs- und Sonstige Grenzen</Title>
        <LegendURL height="57" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="1.980000" min="0.000000"/>
</Layer>
```

Layer '7' enthält die Features der sonstigen Symbole und Beschriftungen. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer sonstige Symbole und Beschriftungen

```
<Layer queryable="0">
    <Name>7</Name>
    <Title>Sonstige Symbole und Beschriftung</Title>
    <Abstract>Sonstige Symbole und Beschriftung</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.231788" maxy="49.0172762433533" minx="9.46807492103876" miny="46.319074"/>
    <BoundingBox SRS="EPSG:31255" maxx="285386.5268" maxy="432611.0739" minx="-287646.3383" miny="137816.865"/>
    <Style>
        <Name>default</Name>
        <Title>Sonstige Symbole und Beschriftung</Title>
        <LegendURL height="302" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="1.980000" min="0.000000"/>
</Layer>
```

Layer '8' enthält die Features der Nutzungssymbole. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Nutzungssymbole

```
<Layer queryable="0">
    <Name>8</Name>
    <Title>Nutzungssymbole</Title>
    <Abstract>Nutzungssymbole</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.233691" maxy="49.0172762433533" minx="9.46807492103876" miny="46.327913"/>
    <BoundingBox SRS="EPSG:31255" maxx="285523.7875" maxy="432657.2145" minx="-287559.7027" miny="138797.223"/>
    <Style>
        <Name>default</Name>
        <Title>Nutzungssymbole</Title>
        <LegendURL height="1087" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="1.980000" min="0.000000"/>
</Layer>
```

Layer '9' enthält die Features der Grenzpunkte. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Grenzpunkte

```
<Layer queryable="1">
    <Name>9</Name>
    <Title>Grenzpunkte</Title>
    <Abstract>Grenzpunkte</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.235186" maxy="49.0172762433533" minx="9.46807492103876" miny="46.318248"/>
    <BoundingBox SRS="EPSG:31255" maxx="285625.7356" maxy="432808.2582" minx="-287728.2797" miny="137728.75"/>
    <Style>
        <Name>default</Name>
        <Title>Grenzpunkte</Title>
        <LegendURL height="341" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="0.594000" min="0.000000"/>
</Layer>
```

Layer '10' enthält die Features der Festpunkte. Die folgende Abbildung zeigt die Beschreibung des Layers.

Layer Festpunkte

```
<Layer queryable="1">
    <Name>10</Name>
    <Title>Festpunkte</Title>
    <Abstract>Festpunkte</Abstract>
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:31255</SRS>
    <LatLonBoundingBox maxx="17.234733" maxy="49.0172762433533" minx="9.46807492103876" miny="46.320712"/>
    <BoundingBox SRS="EPSG:31255" maxx="285621.4372" maxy="432246.271" minx="-287591.4328" miny="137996.73"/>
    <Style>
        <Name>default</Name>
        <Title>Festpunkte</Title>
        <LegendURL height="223" width="100">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Output/ASP_DKM/BEVIntern-5d5b9d3b-c73e
                xlink:type="simple"/>
        </LegendURL>
    </Style>
    <ScaleHint max="19.800000" min="0.000000"/>
</Layer>
```

2.4.3 Mehrere Layer anzeigen

Die unter dem Kapitel Welche Layer stehen zur Verfügung? beschriebenen Layer sollen nun zur Karte hinzugefügt werden. Je nach Layer müssen jedoch andere Parameter angegeben werden. Dies hängt sowohl vom Inhalt, als auch von der Erstellungsart des Layers ab. Die Parameter werden in den folgenden Kapiteln zentral besprochen. Hier wird nur gezeigt, wie die einzelnen Layer hinzugefügt werden. Da die Layer teilweise aufeinander aufbauen, werden sie hier bereits in einer sinnvollen Anordnung hinzugefügt. Diese kann natürlich nach Belieben verändert werden, dies kann jedoch zu weniger sinnvollen Ergebnissen führen.
Folgende Abbildung zeigt, wie der Layer '1' (Nutzungsflächen) zur BaseMap hinzugefügt wird.

Layer Nutzungsflächen zur Karte hinzufügen

```
layer= new OpenLayers.Layer.WMS(
  "Nutzungsflächen",
  "<%= wmsUrl %>",
  {layers: '1', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
  {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
);
map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Nutzungsflächen im Kartenfenster dargestellt werden.

Darstellung Layer Nutzungsflächen

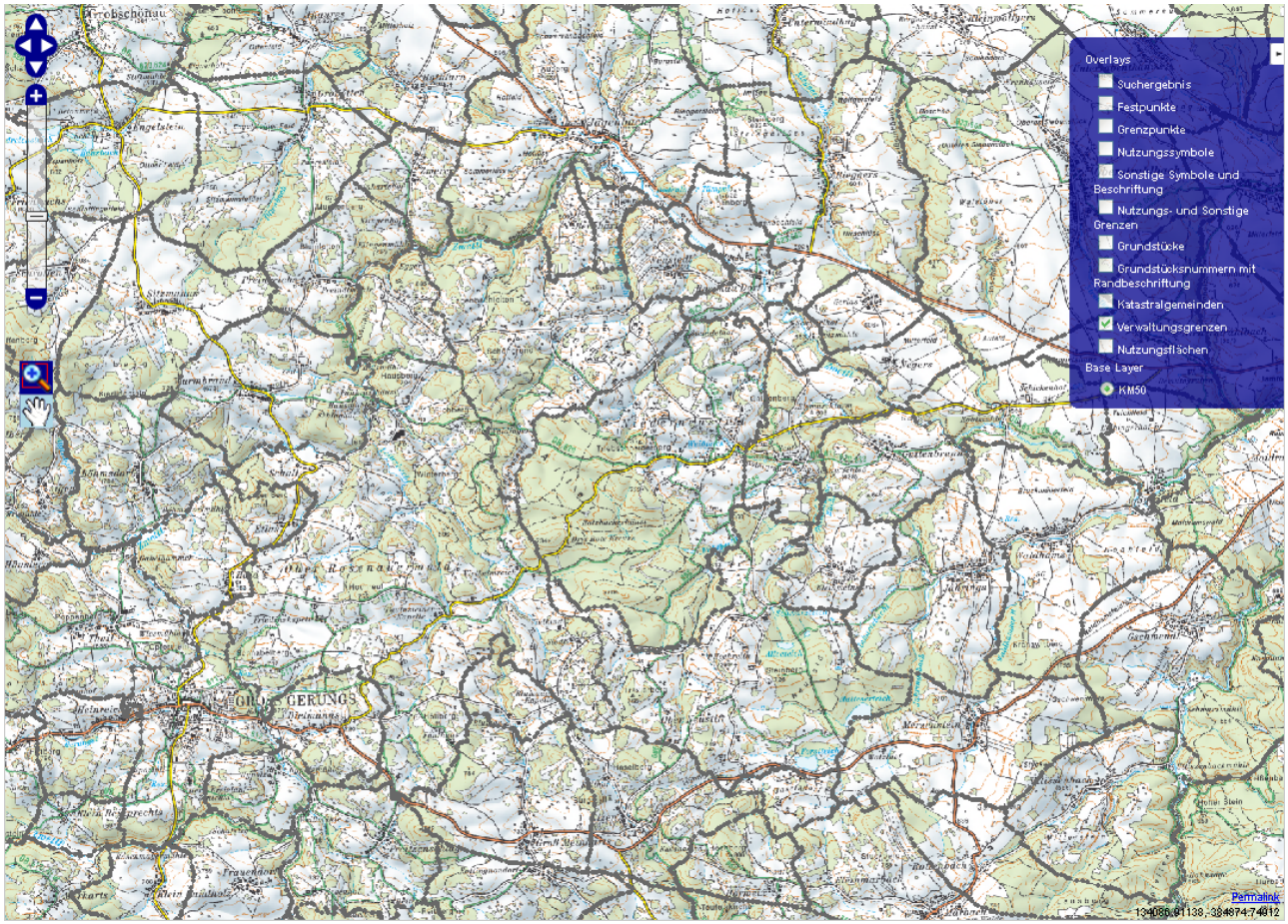
Folgende Abbildung zeigt, wie der Layer '5' (Verwaltungsgrenzen) zur BaseMap hinzugefügt wird.

Layer Verwaltungsgrenzen zur Karte hinzufügen

```
169             layer= new OpenLayers.Layer.WMS(
170               "Verwaltungsgrenzen",
171               "<%= wmsUrl %>",
172               {layers: '5', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
173               {buffer:2, isBaseLayer: false}
174             );
175             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Verwaltungsgrenzen im Kartenfenster dargestellt werden.

Darstellung Layer Verwaltungsgrenzen



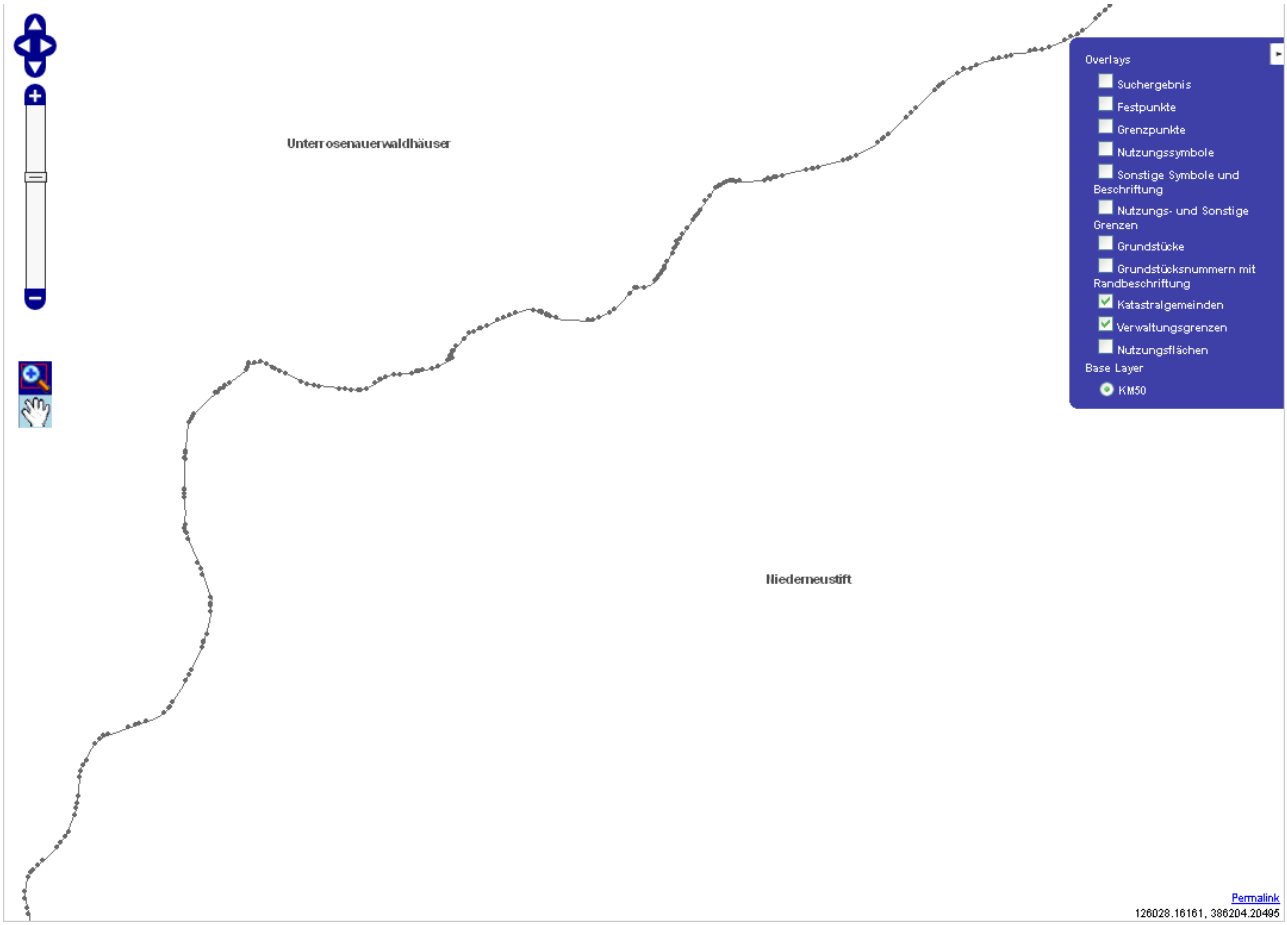
Folgende Abbildung zeigt, wie der Layer '0' (Katastralgemeinden) zur BaseMap hinzugefügt wird.

Layer Katastralgemeinden zur Karte hinzufügen

```
177             layer= new OpenLayers.Layer.WMS(
178               "Katastralgemeinden",
179               "<%= wmsUrl %>",
180               {layers: '0', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
181               {ratio:1.2, singleTile: true, isBaseLayer: false}
182             );
183             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Katastralgemeinden (Texte) im Kartenfenster dargestellt werden. Um die Grenze zwischen zwei Katastralgemeinden darzustellen, wurde zusätzlich noch der Layer Verwaltungsgrenzen aktiviert.

Darstellung Layer Katastralgemeinden



Folgende Abbildung zeigt, wie die Layer '2' und '3' (Grundstücksnummern mit Randbeschriftungen) zur BaseMap hinzugefügt werden.

Layer Grundstücksnummern mit Randbeschriftungen zur Karte hinzufügen

```
185             layer= new OpenLayers.Layer.WMS(
186               "Grundstücksnummern mit Randbeschriftung",
187               "<%= wmsUrl %>",
188               {layers: '2,3', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
189               {buffer:0, singleTile: true, isBaseLayer: false}
190             );
191             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Grundstücksnummern mit Randbeschriftungen im Kartenfenster dargestellt werden. Zusätzlich werden die Grenzen der Grundstücke aktiviert, um die Abgrenzung zwischen den Grundtücksnummern zu verdeutlichen.

Darstellung Layer Grundstücksnummern mit Randbeschriftungen

Folgende Abbildung zeigt, wie der Layer '4' (Grundstücksgrenzen) zur BaseMap hinzugefügt wird.

Layer Grundstücksgrenzen zur Karte hinzufügen

```
193             layer= new OpenLayers.Layer.WMS(
194               "Grundstücke",
195               "<%= wmsUrl %>",
196               {layers: '4', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
197               {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
198             );
199             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Grundstücksgrenzen im Kartenfenster dargestellt werden.

Darstellung Layer Grundstücksgrenzen

Folgende Abbildung zeigt, wie der Layer '6' (Nutzungs- und sonstige Grenzen) zur BaseMap hinzugefügt wird.

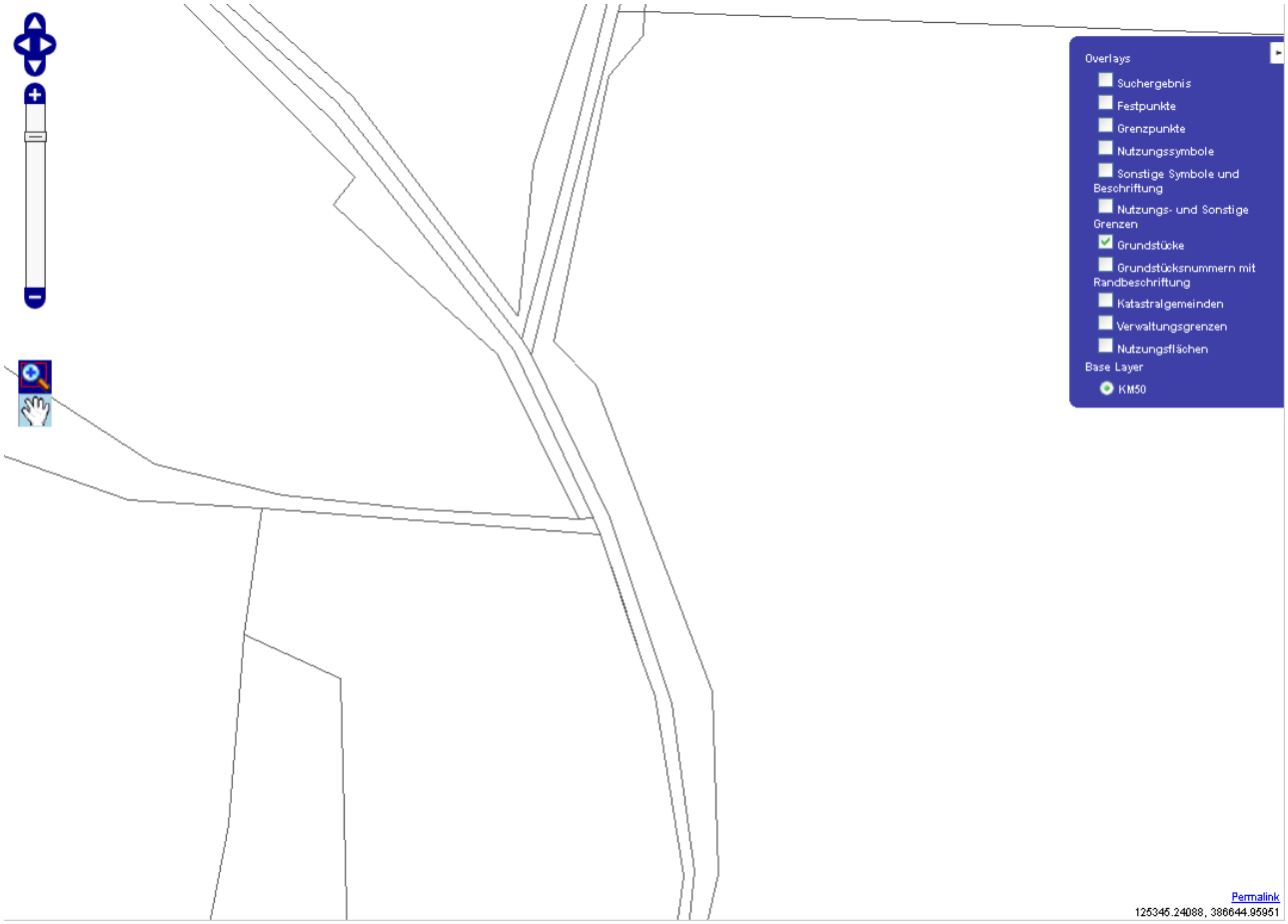
Layer Nutzungs- und sonstige Grenzen zur Karte hinzufügen

```
201             layer= new OpenLayers.Layer.WMS(
202               "Nutzungs- und Sonstige Grenzen",
203               "<%= wmsUrl %>",
204               {layers: '6', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
205               {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
206             );
207             map.addLayer(layer);
```

Die folgenden Abbildungen zeigen, wie die Inhalte des Layers Nutzungs- und sonstige Grenzen im Kartenfenster dargestellt werden. Da es schwer ist festzustellen, welche Linie eine Nutzungsgrenze ist, wird gleich darauf die gleiche Karte nochmals dargestellt, diesmal jedoch ohne Nutzungsgrenzen. Der Unterschied zwischen den zwei folgenden Abbildungen sind die Nutzungsgrenzen.

Darstellung Layer Nutzungs- und sonstige Grenzen (aktiv)

Darstellung Layer Nutzungs- und sonstige Grenzen (inaktiv)



Folgende Abbildung zeigt, wie der Layer '7' (Sonstige Symbole und Beschriftungen) zur BaseMap hinzugefügt wird.

Layer Sonstige Symbole und Beschriftungen zur Karte hinzufügen

```
209             layer= new OpenLayers.Layer.WMS(
210               "Sonstige Symbole und Beschriftung",
211               "<%= wmsUrl %>",
212               {layers: '7', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
213               {buffer:2, isBaseLayer: false}
214             );
215             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Sonstige Symbole und Beschriftungen im Kartenfenster dargestellt werden.

Darstellung Layer Sonstige Symbole und Beschriftungen

Folgende Abbildung zeigt, wie der Layer '8' (Nutzungssymbole) zur BaseMap hinzugefügt wird.

Layer Nutzungssymbole zur Karte hinzufügen

```
217             layer= new OpenLayers.Layer.WMS(
218               "Nutzungssymbole",
219               "<%= wmsUrl %>",
220               {layers: '8', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
221               {buffer:2, isBaseLayer: false}
222             );
223             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Nutzungssymbole im Kartenfenster dargestellt werden.

Darstellung Layer Nutzungssymbole

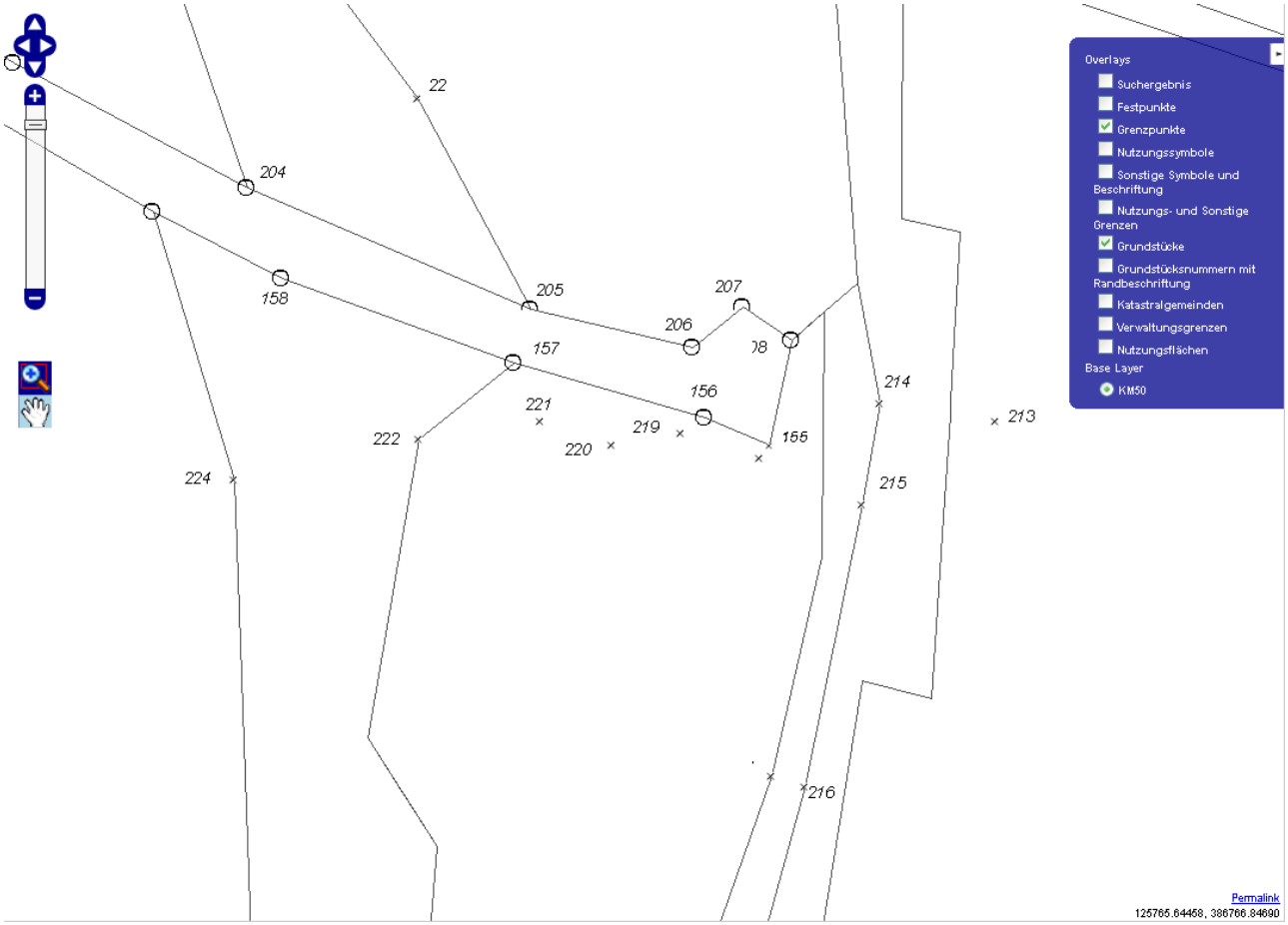
Folgende Abbildung zeigt, wie der Layer '9' (Grenzpunkte) zur BaseMap hinzugefügt wird.

Layer Grenzpunkte zur Karte hinzufügen

```
225             layer= new OpenLayers.Layer.WMS(
226               "Grenzpunkte",
227               "<%= wmsUrl %>",
228               {layers: '9', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
229               {buffer:2, isBaseLayer: false, visibility: false}
230             );
231             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Grenzpunkte im Kartenfenster dargestellt werden.

Darstellung Layer Grenzpunkte



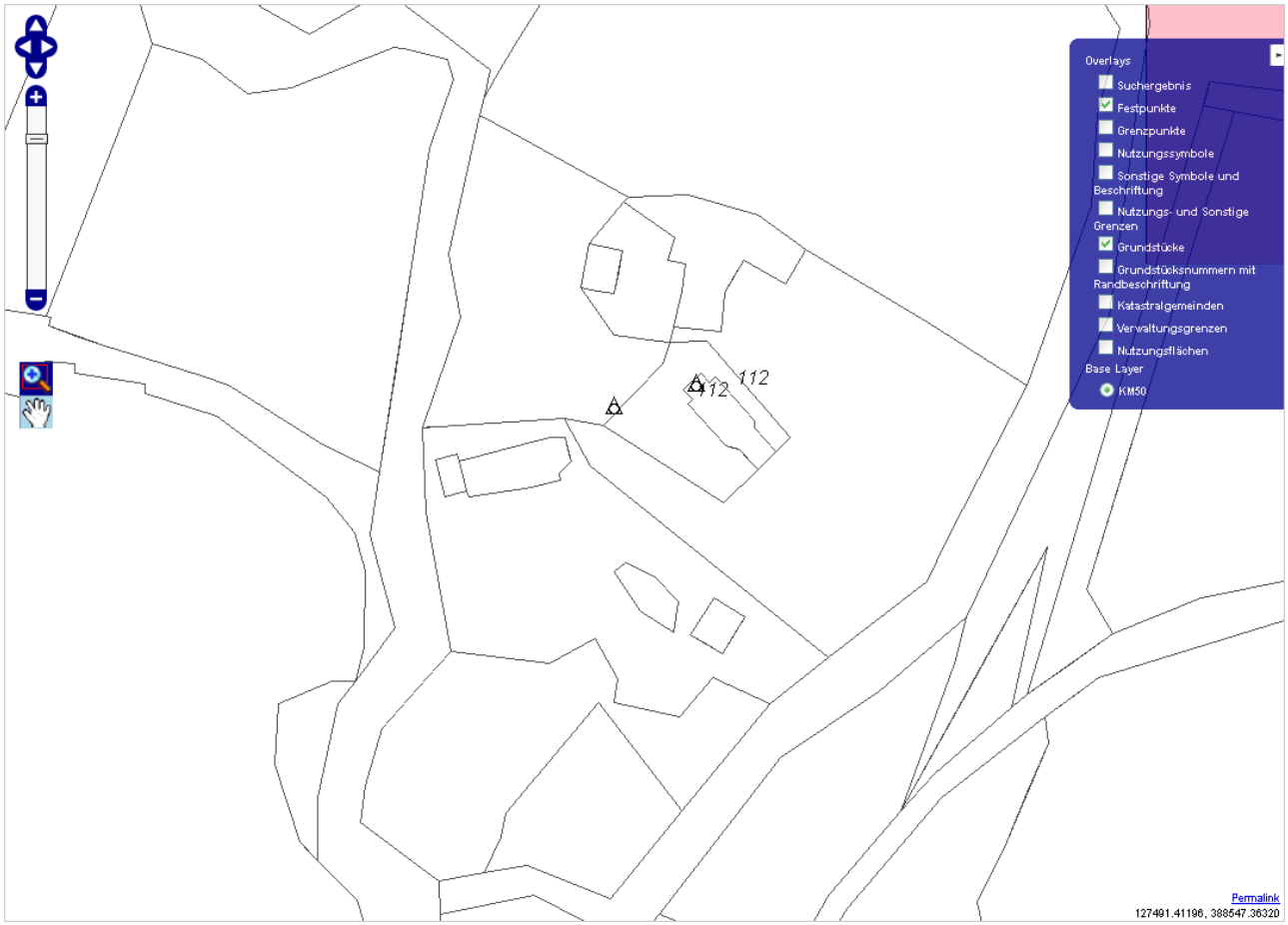
Folgende Abbildung zeigt, wie der Layer '10' (Festpunkte) zur BaseMap hinzugefügt wird.

Layer Festpunkte zur Karte hinzufügen

```
233             layer= new OpenLayers.Layer.WMS(
234               "Festpunkte",
235               "<%= wmsUrl %>",
236               {layers: '10', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
237               {buffer:2, isBaseLayer: false, visibility: false}
238             );
239             map.addLayer(layer);
```

Die folgende Abbildung zeigt, wie die Inhalte des Layers Festpunkte im Kartenfenster dargestellt werden.

Darstellung Layer Festpunkte



2.4.4 Transparenzeinstellung eines Layers

Im vorherigen Kapitel wurden die einzelnen Layer der Karte hinzugefügt. Nun soll gezeigt werden, wie verschiedene Anzeigemechanismen funktionieren. Der erste wäre, dass der Layer transparent erscheinen soll. Dies führt dazu, dass Bereiche des Layers, die leer sind, den oder die Layer unterhalb des aktuellen Layers durchscheinen lassen. Dadurch kann zum Beispiel über eine Karte die Grenze der Katastralgemeinde gelegt werden. Folgende Abbildung zeigt, wie Transparenz als Parameter gesetzt wird (siehe transparent).

Layer Katastralgemeinden transparent hinzufügen

```
177             layer= new OpenLayers.Layer.WMS(
178               "Katastralgemeinden",
179               "<%= wmsUrl %>",
180               {layers: '0', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
181               {ratio:1.2, singleTile: true, isBaseLayer: false}
182             );
183             map.addLayer(layer);
```

Zusätzlich kann noch ein vorteilhafter Effekt aktiviert werden, der im Falle des Zoomings dem Benutzer schnelleren Response gibt. Wenn der Benutzer die Karte verkleinert oder vergrößert, dauert es eine Zeit lang, bis die ersten Daten angezeigt werden. Diese Zeitspanne kann dahingehend überbrückt werden, indem OpenLayers das vorherige Bild des Layers von sich aus verkleinert oder vergrößert und dem Benutzer anzeigt. Die Darstellung ist natürlich von geringer Qualität und wird durch die tatsächliche Grafik ersetzt, sobald diese vom WMS geladen wurde. Folgender Ausschnitt zeigt, wie dies konfiguriert wird (siehe transitionEffect).

Layer Nutzungs- und sonstige Grenzen mit TransitionEffect hinzufügen

```
201             layer= new OpenLayers.Layer.WMS(
202               "Nutzungs- und Sonstige Grenzen",
203               "<%= wmsUrl %>",
204               {layers: '6', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
205               {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
206             );
207             map.addLayer(layer);
```

Es kann auch vorkommen, dass nicht alle Layer von Anfang an dargestellt werden sollen. Dies beschleunigt das Laden der Karte, bietet jedoch trotzdem dem Benutzer die Möglichkeit, bei Bedarf den Layer einzublenden. Um einen Layer ohne Anzeige hinzuzufügen, muss dieser "unsichtbar" angelegt werden. Die folgende Abbildung zeigt die nötige Konfiguration (siehe visibility).

Layer Grenzpunkte unsichtbar hinzufügen

```
225             layer= new OpenLayers.Layer.WMS(
226               "Grenzpunkte",
227               "<%= wmsUrl %>",
228               {layers: '9', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
229               {buffer:2, isBaseLayer: false, visibility: false}
230             );
231             map.addLayer(layer);
```

Das Ergebnis dieser Konfiguration ist ein Layer, der anwählbar, jedoch defaultmäßig deaktiviert ist. Folgende Abbildung zeigt ein Kartenfenster, in welchem dieser Layer inaktiv ist (genauso wie der Layer Festpunkte).

Layer Grenzpunkte inaktiv im LayerSwitcher

2.4.5 Baselayer

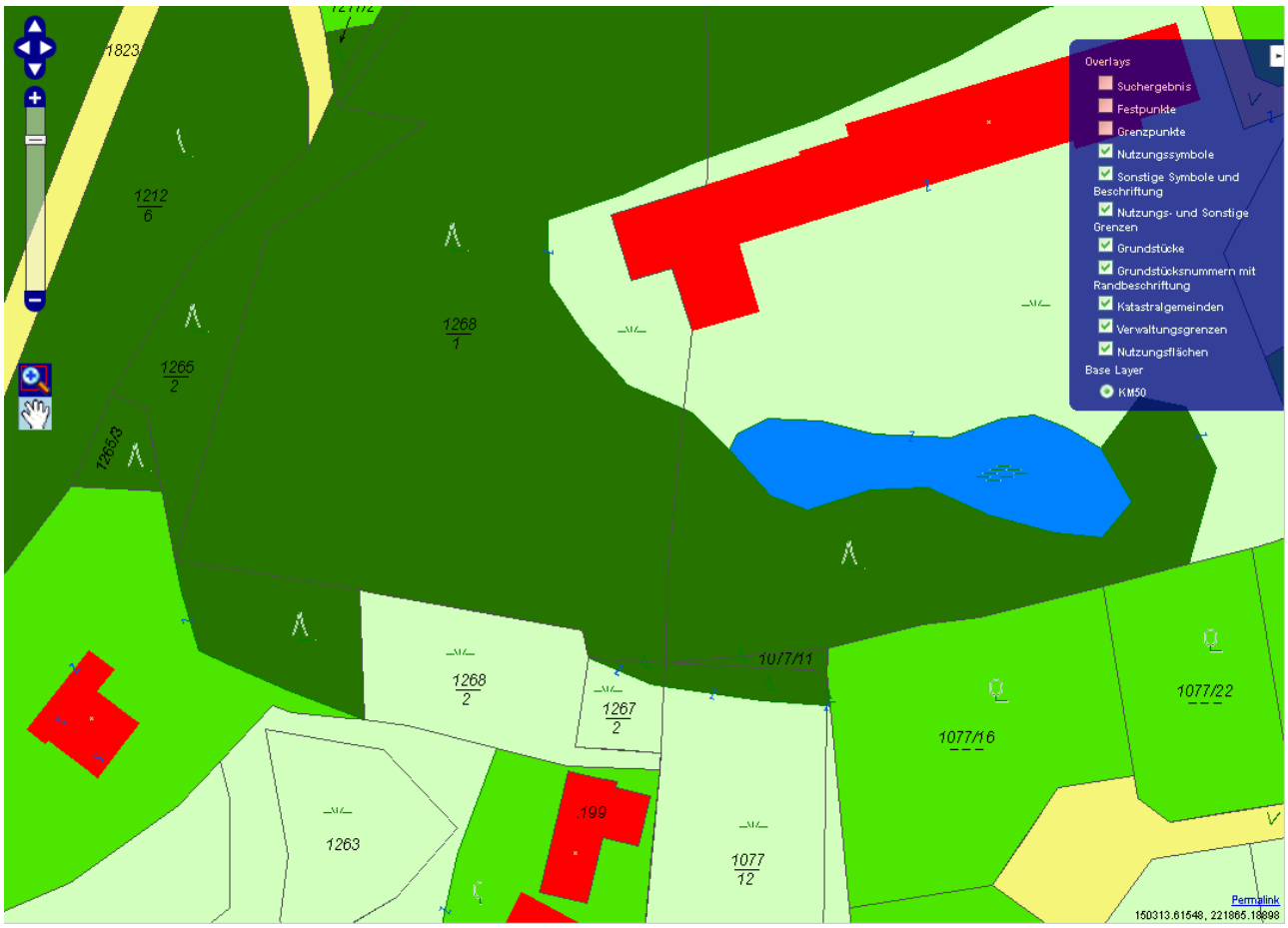
OpenLayers unterscheidet BaseLayer von normalen Layer. Der Unterschied liegt in der Möglichkeit, den Layer auf der Karte zu deaktivieren. Sobald ein Layer als BaseLayer definiert ist, kann dieser nicht abgewählt werden. Daher ist sehr darauf zu achten, welchen Layer der Benutzer aufgezwungen bekommt. Folgende Grafik zeigt die Konfiguration, damit ein Layer nicht zum BaseLayer wird und somit abwählbar bleibt (siehe isBaseLayer).

Layer Verwaltungsgrenzen nicht als BaseLayer hinzufügen

```
169             layer= new OpenLayers.Layer.WMS(
170               "Verwaltungsgrenzen",
171               "<%= wmsUrl %>",
172               {layers: '5', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
173               {buffer:2, isBaseLayer: false}
174             );
175             map.addLayer(layer);
```

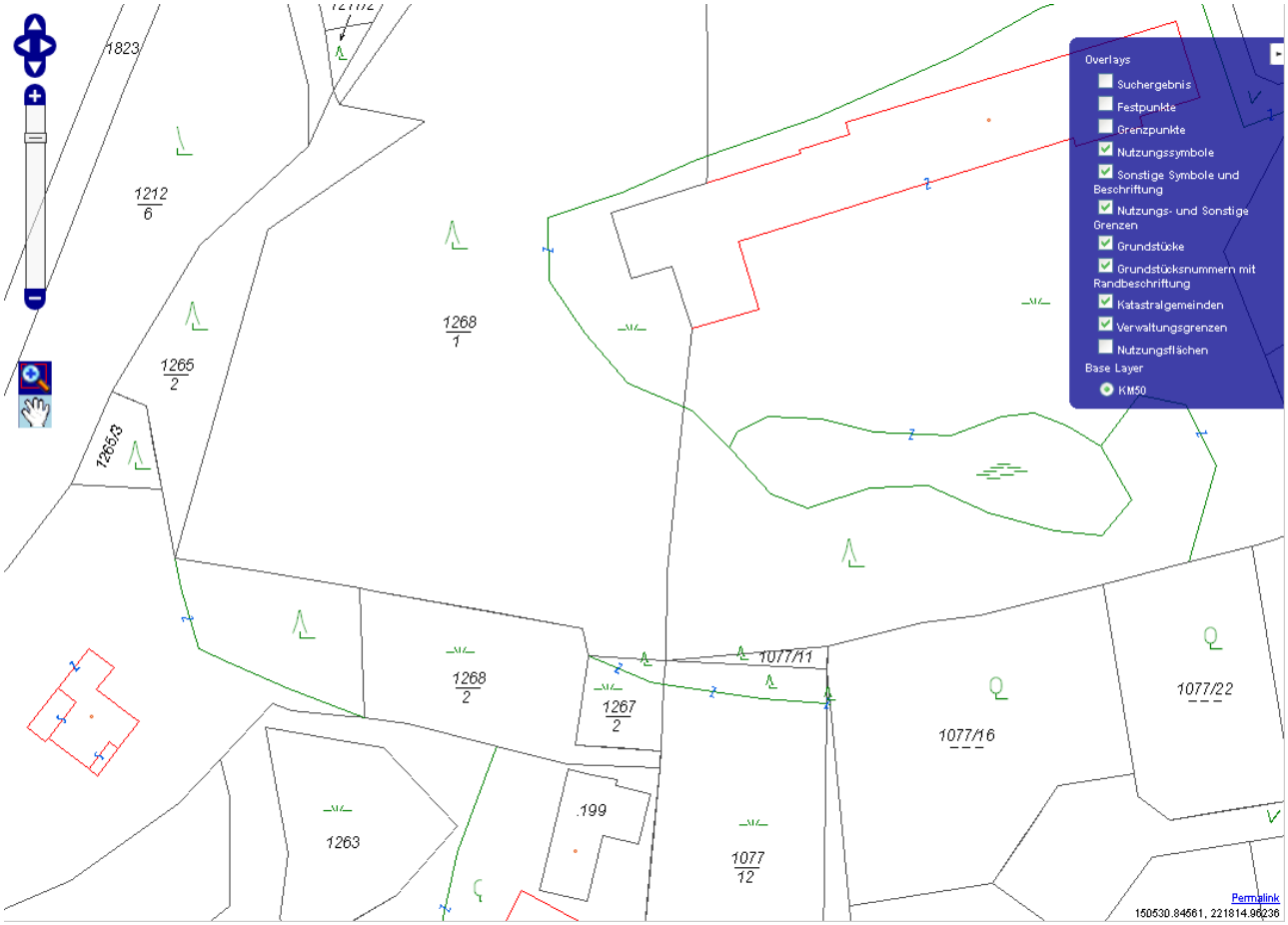
Diese Konfiguration fügt den Layer Verwaltungsgrenzen abwählbar zur Karte hinzu. Dieser wird innerhalb des LayerSwitchers im Bereich Overlay angeführt, und ist somit abwählbar. Folgende Grafik zeigt einen ausgefahrenen LayerSwitcher mit allen Layer als Overlays.

Layer Verwaltungsgrenzen abwählbar



Zum Vergleich dazu die gleiche Grafik, wobei lediglich die Nutzungsfläche deaktivert wurden. Die Grafik hat automatisch ein anderes Erscheinungsbild, ohne dass diese neu vom Server (WMS) geholt werden muss. Alles wird durch OpenLayers im Browser neu gerechnet.

Layer Nutzungsflächen abgewählt



2.4.6 Layer und Tiles

Tiles wurden bereits kurz angesprochen und sollen nun genauer betrachtet und konfiguriert werden. Es besteht die Möglichkeit, die Karte in kleinere Teile zu teilen und Stück für Stück zu laden. Der Benutzer bekommt dadurch viel früher einen Teil der Karte zu sehen und hat ein subjektives Gefühl der Schnelligkeit. Ganz nach dem Motto, es ist besser schnell ein Feedback zu bekommen, auch wenn der gesamte Prozess länger dauert, als einen schnellen Prozess zu haben, der erst ganz am Ende alles liefert. Um einen Layer in Tiles aufzuteilen reicht es, den Parameter singleTile wegzulassen.

Layer Nutzungssymbole mit Tiles hinzufügen

```
217             layer= new OpenLayers.Layer.WMS(
218               "Nutzungssymbole",
219               "<%= wmsUrl %>",
220               {layers: '8', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
221               {buffer:2, isBaseLayer: false}
222             );
223             map.addLayer(layer);
```

Wie aus der Grafik ersichtlich ist, fehlt die Angabe von singleTile. Dadurch wird der Layer mittels Tiles vom WMS geladen. Zusätzlich ist hier noch der Parameter buffer anzumerken. Dieser enthält den Wert 2, welcher besagt, dass OpenLayers 2 Tiles außerhalb des sichtbaren Kartenbereiches zusätzlich anfordern soll, um etwaiges Panning zu beschleunigen. Im Falle eines Pannings wären bereits die nähesten Tiles vorhanden und müssen nicht erst extra vom Server (WMS) geholt werden. Natürlich lädt OpenLayers zuerst die sichtbaren Tiles herunter und erst im Anschluß daran die nicht sichtbaren.

Durch Angabe des Parameters singleTile wird OpenLayers mitgeteilt, diesen Layer in einem Stück vom WMS anzufordern. Dies widerspricht den oben ausgeführten gefühlten Ladegeschwindigkeiten, ist jedoch zumindest bei zwei Layern unbedingt notwendig. Diese Layer sind Katastralgemeinde und Grundstücknummern mit Randbedingungen (Layer 0, 2, 3), da diese Beschriftungen enthalten, die ansonsten in jedem Tile vorkämen. Wenn zum Beispiel eine Katastralgemeinde über 2 Tiles geht, findet sich in jedem Tile der Name der Katastralgemeinde wieder. Da dies zu unanschaulichen Karten führt, ist es sinnvoller, den gesamten Layer in einem einzelnen Tile anzufordern. Folgende Abbildung zeigt, wie dies funktioniert (siehe singleTile).

Layer Katastralgemeinden mit singleTile hinzufügen

```
177             layer= new OpenLayers.Layer.WMS(
178               "Katastralgemeinden",
179               "<%= wmsUrl %>",
180               {layers: '0', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
181               {ratio:1.2, singleTile: true, isBaseLayer: false}
182             );
183             map.addLayer(layer);
```

Die Abbildung zeigt eine Besonderheit von singleTile-Layern. Nachdem diese aus genau einem Tile bestehen, kann kein Buffer angegeben werden. Ein Buffer würde um das eine singleTile noch weitere Tiles abfragen. Nachdem jedoch die Abfrage in einem Tile erfolgen muss, müsste das Tile größer sein. Womit man wieder am Anfang stünde, da um das singleTile wieder ein Buffer zu legen wäre. Dieses Problem umgeht OpenLayers durch einen eigenen Parameter, der ratio heißt. Dieser definiert einen Faktor für die Größe des angeforderten Kartenausschnittes. In dem obigen Beispiel bedeutet die Angabe von 1.2, dass eine Karte vom WMS abgefragt wird, die im Vergleich zum Kartenfenster der Webapplikation 1,2 mal so breit wie auch hoch ist. Auf diese Weise kann ein größerer Ausschnitt errechnet werden und das Panning bleibt weiterhin schnell.

2.4.7 Zusammenfassung

Innerhalb einer Karte können mehrere Ebenen mit geografischen Informationen angezeigt werden. Diese Ebenen werden Layer genannt. Diese Lerneinheit geht zuerst der Definition eines Layers nach, gefolgt von einer Aufstellung, welche Layer vom BEV zur Verfügung gestellt werden. Als anschauliches Beispiel werden danach alle Layer des BEV in einer einzigen Karte vereint. Um noch brauchbare Informationen aus der Karte auslesen zu können, wird daraufhin erklärt, wie Transparenzen bei Layern eingestellt werden können. Abschließend wird die Konfiguration von Tiles und deren Bedeutung erläutert.

2.5 Zusammenfassung

Dieser Kurs vermittelt das Wissen, wie mittels OpenLayers ein Kartenfenster angelegt und mit einer Karte befüllt werden kann. Hierfür werden Begriffe wie Layer, Map, WMS, Koordinatensystem und Tiles erklärt. Folgend eine kurze Übersicht über den Inhalt der einzelnen Lerneinheiten dieses Kurses.
Die erste Lerneinheit enthält die Einführung in das Thema Web Map Service (WMS). Beginnend wird der Begriff und die Spezifikation eines WMS erläutert, gefolgt von der Beschreibung der einzelnen Methoden, die ein WMS enthalten kann. Jede der Methoden wird durch ein eigenes Beispiel zusätzlich veranschaulicht.
In Lerneinheit 2 werden grundlegende Kartenfunktionen erläutert. Zuerst erfolgt eine kurze Einführung in OpenLayers und die Erklärung, wie innerhalb von OpenLayers eine Karte dargestellt wird. Danach folgt die Information, was in OpenLayers eine Map darstellt und zum Unterschied dazu, was ein WMS-Layer ist. Eine Map bildet die Referenz auf das Kartenfenster in OpenLayers und ein WMS-Layer entspricht einer Ebene innerhalb der Karte, die eine Grafik enthält, welche durch ein WMS erzeugt wird. Innerhalb der Map bietet OpenLayers die Möglichkeit, die Karte zu pannen und zu zoomen.
Um eine Karte im Kartenfenster von OpenLayers darzustellen, muss ein Koordinatensystem für die Karte definiert werden. Die dritte Lerneinheit zeigt, wie in OpenLayers ein Koordinatensystem konfiguriert werden kann. Sollte kein Koordinatensystem angegeben werden, verwendet OpenLayers automatisch die Angabe der Koordinaten in Längen- und Breitengraden. Im Anschluß werden die in Österreich verwendeten Koordinatensysteme vorgestellt. Abschließend wird erklärt, wie die einzelnen Bedienelemente und die Zoommöglichkeit in OpenLayers konfiguriert werden kann.
Innerhalb einer Karte können mehrere Ebenen mit geografischen Informationen angezeigt werden. Diese Ebenen werden Layer genannt. Die Lerneinheit 4 geht zuerst der Definition eines Layers nach, gefolgt von einer Aufstellung, welche Layer vom BEV zur Verfügung gestellt werden. Als anschauliches Beispiel werden danach alle Layer des BEV in einer einzigen Karte vereint. Um noch brauchbare Informationen aus der Karte auslesen zu können, wird daraufhin erklärt, wie Transparenzen bei Layern eingestellt werden können. Abschließend wird die Konfiguration von Tiles und deren Bedeutung erläutert.

2.6 Abbildungsverzeichnis

2.7 Tabellenverzeichnis

3. Zugriff auf ein Web Feature Service

Diese eLesson vermittelt das Wissen, um auf Vektordaten eines Web Feature Services zugreifen zu können. Die zu verwendende Schnittstelle wird durch das Open Geospatial Consortium (OGC) definiert und als Standard aufgelegt. Dadurch bieten bereits einige Hersteller sowohl serverseitige als auch clientseitige Implementierungen an. Solange die Implementierungen dem Standard konform sind, können diese miteinander kombiniert werden. Diese eLesson verwendet einen dieser standardisierten Clients, um auf ein standardisiertes WFS zuzugreifen. Das WFS wurde durch das BEV zur Verfügung gestellt und dient hier dem Testen.

Lernziele

- Beschreiben eines Web Feature Services
- Beschreiben der Methoden eines Web Feature Service
- Verstehen der Verwendung von JavaScript
- Erklären des Sandboxkonzeptes eines Browser
- Verwenden von AJAX und XMLHttpRequest
- Mittels OpenLayers auf ein WFS zugreifen
- Erklären der ProxyHost Konfiguration in OpenLayers
- Programmieren eines ProxyHosts für GET- und POST-Zugriffe
- Verstehen von Parameter Encoding
- Kennen der zur Verfügung stehenden WFS
- Verstehen der Queries bei WFS
- Verwenden von Filter Encoding
- Erklären der Verwendung von Spatial Operators
- Einsetzen von Comparison Operators
- Einschränken des Ergebnises mittels Logical Operators

3.1 Grundlagen zum Thema Web Feature Service

In dieser Lerneinheit werden die Grundlagen zum Thema Web Feature Service behandelt. Es werden die einzelnen Methoden, die im Standard definiert sind, erklärt und in drei Gruppen aufgeteilt.

Lernziele

- Beschreiben eines Web Feature Services
- Beschreiben der Methoden eines Web Feature Service

3.1.1 Was ist ein Web Feature Service?

Ein Web Feature Service (WFS) bietet die Möglichkeit Vektordaten zu übertragen. Im Gegensatz zum WMS werden hier keine Grafiken erzeugt, sondern die tatsächlichen Features (geografische Objekte) mittels Geographic Markup Language (GML) übertragen. Mit diesen Daten kann der Client das Objekt selbständig in Karten zeichnen. Mittels OpenLayers ist es dabei möglich, Features über bereits eingeblendete Karten zu legen. Diese stellen dann innerhalb der Map einen eigenen Layer dar.
Die Spezifikation listet eine Reihe von Methoden auf, die ein WFS implementieren kann. Drei davon sind verpflichtende Methoden:

- GetCapabilities (verpflichtend)
- DescribeFeatureType (verpflichtend)
- GetFeature (verpflichtend)
- GetFeatureWithLock
- GetGmlObject
- LockFeature
- Transaction

Des Weiteren definiert die Spezifikation noch drei Gruppen von WFS Arten, die eine Implementierung umsetzen kann. Jedes WFS gehört zu einer der folgenden drei Gruppen:

- Basic WFS: implementiert die Methoden GetCapabilities, DescribeFeatureType und GetFeature
- XLink WFS: implementiert alle Basic WFS Methoden und zusätzlich GetGmlObject
- Transaction WFS: implementiert alle Basic WFS Methoden und zusätzlich die Methode Transaction. Optional kann hier noch die Implementierung der Methoden LockFeature und GetFeatureWithLock erfolgen.

Die einzelnen Methode werden im Anschluss genauer erklärt.

3.1.2 Methode GetCapabilities

Jedes WFS muss in der Lage sein, seine Möglichkeiten zu beschreiben. Dies erfolgt mittels Methode GetCapabilities. Die Antwort auf eine GetCapabilities Anfrage beinhaltet Informationen über die unterstützten Methoden, die zuständige Person für das WFS, eine Liste der lieferbaren FeatureTypes und eine Liste der Filter, die das WFS unterstützt. Die Methode GetCapabilities unterstützt folgende Parameter:

Parameter der Methode GetCapabilities

Parameter	Pflichtfeld	Beschreibung
VERSION=1.1.0	Nein	Version der Anfrage
SERVICE=WFS	Nein	Typ des Services
REQUEST=GetCapabilities	Ja	Name der Methode

3.1.3 Beispiel für die Methode GetCapabilities

In diesem Kapitel soll eine beispielhafte Anfrage mit der zugehörigen Antwort gezeigt werden. Die Methode GetCapabilities kann mittels GET-Request und Key-Value-Pair (KVP) Parametern aufgerufen werden. Folgende URL stellt einen gültigen Aufruf dar.

Beispiel eines WFS Aufrufes

```
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.1.0
```

Die Antwort auf diesen Request enthält mehrere Informationen, die kurz beschrieben werden sollen. Ganz am Anfang der Antwort stehen der Name, der Typ und die unterstützte Version des Services. Danach folgen Informationen über diejenige Person, die für dieses Service verantwortlich ist, inklusive deren Kontaktinformationen. Folgende Abbildung zeigt diese Metadaten.

Service-Metadaten

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:WFS_Capabilities xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:ows="http://www.opengis.net/ows"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="1.1.0"
    xsi:schemaLocation="http://www.opengis.net/gml http://schemas.opengis.net/gml/3.1.1/base/gml.xsd
                        http://www.opengis.net/ogc http://schemas.opengis.net/filter/1.1.0/filter.xsd
                        http://www.opengis.net/ows http://schemas.opengis.net/ows/1.0.0/owsAll.xsd
                        http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
    <ows:ServiceIdentification>
        <ows:Title>WFS-Service der DKM</ows:Title>
        <ows:Abstract>WFS-Service der DKM</ows:Abstract>
        <ows:Keywords>
            <ows:Keyword/>
        </ows:Keywords>
        <ows:ServiceType>WFS
        </ows:ServiceType>
        <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
        <ows:Fees/>
        <ows:AccessConstraints/>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>BEV</ows:ProviderName>
        <ows:ProviderSite xlink:href="http://www.bev.gv.at"/>
        <ows:ServiceContact>
            <ows:IndividualName>DI Stefan Klotz</ows:IndividualName>
            <ows:PositionName>Projekt Koordinator</ows:PositionName>
            <ows:ContactInfo>
                <ows:Phone>
                    <ows:Voice>+43-1-21110-0</ows:Voice>
                    <ows:Facsimile>+43-1-21110-0</ows:Facsimile>
                </ows:Phone>
                <ows:Address>
                    <ows:DeliveryPoint/>
                    <ows:City>Wien</ows:City>
                    <ows:AdministrativeArea/>
                    <ows:PostalCode>A-1020</ows:PostalCode>
                    <ows:Country/>
                    <ows:ElectronicMailAddress/>
                </ows:Address>
                <ows:OnlineResource
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e781
                <ows:HoursOfService>Wochentags 09:00 - 16:00</ows:HoursOfService>
                <ows:ContactInstructions>e-Mail kundendienst@bev.gv.at</ows:ContactInstructions>
            </ows:ContactInfo>
        </ows:ServiceContact>
    </ows:ServiceProvider>
```

Daraufhin sind alle unterstützten Methoden des Sevices angeführt.

Service-Methoden

```
<ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-
                <ows:Post
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="AcceptVersions">
            <ows:Value>1.1.0</ows:Value>
            <ows:Value>1.0.0</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="AcceptFormats">
            <ows:Value>text/xml</ows:Value>
        </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="DescribeFeatureType">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-
                <ows:Post
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="outputFormat">
            <ows:Value>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</ows:Value>
        </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetFeature">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-
                <ows:Post
                    xlink:href="http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="resultType">
            <ows:Value>results</ows:Value>
            <ows:Value>hits</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputFormat">
            <ows:Value>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</ows:Value>
        </ows:Parameter>
    </ows:Operation>
    <ows:ExtendedCapabilities>
        <ows:Constraint name="serviceAxisOrderForSwappableSRS">
            <ows:Value>latitude,longitude</ows:Value>
        </ows:Constraint>
    </ows:ExtendedCapabilities>
</ows:OperationsMetadata>
```

Das Wichtigste an einem WFS sind die gelieferten Daten. Diese werden ebenfalls grob in der Antwort beschrieben. Hierbei werden die Namen der Features angegeben und grundlegende Informationen, wie zum Beispiel die möglichen Koordinatensysteme für die Abgabe.

Liste der FeatureTypes

```
<wfs:FeatureTypeList>
  <wfs:FeatureType>
    <wfs:Name>ASP_WFS:GrundstuecksRechtecke</wfs:Name>
    <wfs:Title>GrundstuecksRechtecke</wfs:Title>
    <wfs:DefaultSRS>EPSG:31255</wfs:DefaultSRS>
    <wfs:OtherSRS>EPSG:4326</wfs:OtherSRS>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
    </wfs:OutputFormats>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>9.4052369537545228 46.327670137886301</ows:LowerCorner>
      <ows:UpperCorner>17.23375146832625 49.032104102733868</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  </wfs:FeatureType>
  <wfs:FeatureType>
    <wfs:Name>ASP_WFS:GrundstuecksEinsetzpunkte</wfs:Name>
    <wfs:Title>GrundstuecksEinsetzpunkte</wfs:Title>
    <wfs:DefaultSRS>EPSG:31255</wfs:DefaultSRS>
    <wfs:OtherSRS>EPSG:4326</wfs:OtherSRS>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
    </wfs:OutputFormats>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>9.4052369537545228 46.327670137886301</ows:LowerCorner>
      <ows:UpperCorner>17.23375146832625 49.032104102733868</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  </wfs:FeatureType>
  <wfs:FeatureType>
    <wfs:Name>ASP_WFS:KatastralgemeindenRechtecke</wfs:Name>
    <wfs:Title>KatastralgemeindenRechtecke</wfs:Title>
    <wfs:DefaultSRS>EPSG:31255</wfs:DefaultSRS>
    <wfs:OtherSRS>EPSG:4326</wfs:OtherSRS>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
    </wfs:OutputFormats>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>9.4327886435610484 46.401035227143304</ows:LowerCorner>
      <ows:UpperCorner>17.177014305348173 49.029002596024235</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  </wfs:FeatureType>
</wfs:FeatureTypeList>
```

Um nach Features suchen zu können, bieten WFS eine Query-Funktionalität an. Diese funktioniert ähnlich einer normalen SQL-Anweisung, mit der Ausnahme, dass auch räumliche Abfragen möglich sind. Welche Abfrage genau unterstützt werden, zeigt der letzte Teil der GetCapabilities-Antwort.

Filter Capabilities des Services

```
<ogc:Filter_Capabilities>
  <ogc:Spatial_Capabilities>
    <ogc:GeometryOperands>
      <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
    </ogc:GeometryOperands>
    <ogc:SpatialOperators>
      <ogc:SpatialOperator name="BBOX" />
      <ogc:SpatialOperator name="Equals" />
      <ogc:SpatialOperator name="Disjoint" />
      <ogc:SpatialOperator name="Intersects" />
      <ogc:SpatialOperator name="Crosses" />
      <ogc:SpatialOperator name="Touches" />
      <ogc:SpatialOperator name="Within" />
      <ogc:SpatialOperator name="Contains" />
      <ogc:SpatialOperator name="Overlaps" />
    </ogc:SpatialOperators>
  </ogc:Spatial_Capabilities>
  <ogc:Scalar_Capabilities>
    <ogc:LogicalOperators />
    <ogc:ComparisonOperators>
      <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
    </ogc:ComparisonOperators>
  </ogc:Scalar_Capabilities>
  <ogc:Id_Capabilities>
    <ogc:EID />
    <ogc:FID />
  </ogc:Id_Capabilities>
</ogc:Filter_Capabilities>
</wfs:WFS_Capabilities>
```

3.1.4 Methode DescribeFeatureType

Aufbauend auf die Informationen des GetCapabilities Requests können zusätzliche Informationen zu einem bestimmten FeatureType abgefragt werden. Diese Informationen werden in Form eines XSD-Files geliefert, welches das GML-Schema referenziert. Die Methode DescribeFeatureTypes bietet folgende Parameter:

Parameter der Methode DescribeFeatureType

Parameter	Pflichtfeld	Beschreibung
VERSION=1.1.0	Nein	Version der Anfrage
SERVICE=WFS	Nein	Typ des Services
OUTPUTFORMAT=text/xml; subtype=gml/3.1.1	Nein	Ausabeformat der Antwort. text/xml; subtype=gml/3.1.1 ist das einzige Ausgabeformat, das unterstützt werden muss.
TYPENAME=typname	Nein	Abgefragter Typ; wenn keine Angabe erfolgt, werden alle FeatureTypes zurückgeliefert
REQUEST=DescribeFeatureType	Ja	Name der Methode

3.1.5 Beispiel für die Methode DescribeFeatureType

Um die Methode DescribeFeatureType genauer zu verdeutlichen, soll ein anschauliches Beispiel den Inhalt erklären. Die Anfrage an das WFS erfolgt wiederum mittels URL.

Beispiel für DescribeFeatureType

```
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=DescribeFeatureType&typename=GrundstuecksRechtecke
```

Die folgende Abbildung zeigt die Antwort auf die Anfrage. Wie zu erkennen ist, handelt es sich hierbei um ein herkömmliches XSD-File.

DescribeFeatureType GrundstuecksRechtecke (Teil 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
           xmlns:gml="http://www.opengis.net/gml"
           elementFormDefault="qualified"
           targetNamespace="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer">
    <xs:import namespace="http://www.opengis.net/gml"
               schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
    <xs:element name="GrundstuecksRechtecke"
                substitutionGroup="gml:_Feature"
                type="ASP_WFS:GrundstuecksRechteckeType"/>
    <xs:complexType name="GrundstuecksRechteckeType">
        <xs:complexContent>
            <xs:extension base="gml:AbstractFeatureType">
                <xs:sequence>
                    <xs:element name="OBJECTID" type="xs:int"/>
                    <xs:element name="GNR">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="11"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="RSTATUS">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="1"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="TYP" type="xs:short"/>
                    <xs:element name="KG">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="5"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="KGGNR">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="16"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="GEMNR" type="xs:int"/>
                    <xs:element minOccurs="0" name="BEZNR" type="xs:short"/>
                    <xs:element minOccurs="0" name="LANDNR" type="xs:short"/>
```

DescribeFeatureType GrundstuecksRechtecke (Teil 2)

```
                    <xs:element minOccurs="0" name="VANR">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="2"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="XMIN" type="xs:double"/>
                    <xs:element minOccurs="0" name="YMIN" type="xs:double"/>
                    <xs:element minOccurs="0" name="XMAX" type="xs:double"/>
                    <xs:element minOccurs="0" name="YMAX" type="xs:double"/>
                    <xs:element minOccurs="0" name="KGNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="GEMNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="BEZNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="VANAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="LANDNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="SHAPE" type="gml:PolygonPropertyType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

3.1.6 Methode GetFeature/GetFeatureWithLock

Die Methode GetFeature liefert ein bestimmtes Feature oder eine Liste von Features zu einer Anfrage. Diese Anfrage wird mittels Filter Encoding als Query übergeben. Die Antwort ist ein XML-File, welches GML-Objekte enthält. Folgende Parameter können bei der Anfrage übergeben werden:

Parameter der Methode GetFeature/GetFeatureWithLock

Parameter	Pflichtfeld	Beschreibung
VERSION=1.1.0	Nein	Version der Anfrage
SERVICE=WFS	Nein	Typ des Services
REQUEST=[GetFeature| GetFeatureWithLock]	Ja	Name der Methode. GetFeatureWithLock sperrt die Features gegen Änderungen anderer Benutzer, damit eine nachfolgende Änderung nicht fremde Daten überschreibt.
OUTPUTFORMAT=text/ xml;subtype=gml/3.1.1	Nein	Ausgabeformat der Antwort. text/xml;subtype=gml/3.1.1 muss unterstützt werden
RESULTTYPE=[results|hits]	Nein	Angabe, ob das Ergebnis verlangt wird (results; Default) oder nur die Anzahl der gefundenen Features (hits)
PROPERTYNAME=propertynames	Nein	Liste der Properties, die im Ergebnis enthalten sein sollen
FEATUREVERSION=[ALL | N]	Nein	Wenn das Feature historisiert wird, Angabe ob nur die aktuelle Version geliefert werden soll oder auch historische Versionen
MAXFEATURES=N	Nein	Begrenzung des Ergebnisses in der Anzahl
EXPIRY=N	Nein	Im Falle von GetFeatureWithLock gibt dieser Parameter an, wie lange der Lock gehalten werden soll
SRSNAME=srsname	Nein	Name des Koordinatensystems, das für die Ausgabe verwendet werden soll
TYPENAME=typname	Ja	Name der FeatureTypes, die von der Query betroffen sind
FEATUREID=id	Nein	Zugriff mittels ID des Features
FILTER=filter	Nein	Angabe des Filters für die Anfrage
BBOX=koordinaten	Nein	Abfrage mittels Koordinaten anstatt Filter
SORTBY=propertynames	Nein	Liste der Properties, anhand welcher sortiert werden soll

3.1.7 Beispiel für die Methode GetFeature/GetFeatureWithLock

Folgendes Beispiel soll den Zugriff auf ein Feature verdeutlichen.

GetFeature GrundstuecksRechteck

```
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=GetFeature&typename=ASP_WFS:GrundstuecksRechtecke&filter=<Filter+xmlns="http://www.opengis.net/ogc"><PropertyIsEqualTo><PropertyName>KGGNR</PropertyName><Literal>910111992</Literal></PropertyIsEqualTo></Filter>
```

Ergebnis dieser Anfrage ist folgendes XML-File.

GetFeature GrundstuecksRechtecke

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
                       xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
                       xmlns:gml="http://www.opengis.net/gml"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xsi:schemaLocation="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer
                                           http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a
                                           http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
    <gml:boundedBy>
        <gml:Envelope srsName="EPSG:31255">
            <gml:lowerCorner>138770.24100000001 -287560.1409</gml:lowerCorner>
            <gml:upperCorner>432642.6263 285528.98479999998</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
        <ASP_WFS:GrundstuecksRechtecke gml:id="F23__8505563">
            <ASP_WFS:OBJECTID>8505563</ASP_WFS:OBJECTID>
            <ASP_WFS:GNR>1992</ASP_WFS:GNR>
            <ASP_WFS:RSTATUS>E</ASP_WFS:RSTATUS>
            <ASP_WFS:TYP>1</ASP_WFS:TYP>
            <ASP_WFS:KG>91011
            </ASP_WFS:KG>
            <ASP_WFS:KGGNR>910111992</ASP_WFS:KGGNR>
            <ASP_WFS:GEMNR>80227</ASP_WFS:GEMNR>
            <ASP_WFS:BEZNR>802</ASP_WFS:BEZNR>
            <ASP_WFS:LANDNR>8</ASP_WFS:LANDNR>
            <ASP_WFS:VANR>91</ASP_WFS:VANR>
            <ASP_WFS:XMIN>-262665.56069999997</ASP_WFS:XMIN>
            <ASP_WFS:YMIN>249850.0405</ASP_WFS:YMIN>
            <ASP_WFS:XMAX>-262254.31359999999</ASP_WFS:XMAX>
            <ASP_WFS:YMAX>250176.7139</ASP_WFS:YMAX>
            <ASP_WFS:KGNAME>Mellau</ASP_WFS:KGNAME>
            <ASP_WFS:GEMNAME>Mellau</ASP_WFS:GEMNAME>
            <ASP_WFS:BEZNAME>Bregenz</ASP_WFS:BEZNAME>
            <ASP_WFS:VANAME>Bregenz</ASP_WFS:VANAME>
            <ASP_WFS:LANDNAME>Vorarlberg</ASP_WFS:LANDNAME>
            <ASP_WFS:SHAPE>
                <gml:Polygon>
                    <gml:outerBoundaryIs>
                        <gml:LinearRing>
                            <gml:posList>249850.0405 -262665.56069999997 250176.7139 -262665.56069999997 250176.7139
                        </gml:LinearRing>
                    </gml:outerBoundaryIs>
                </gml:Polygon>
            </ASP_WFS:SHAPE>
        </ASP_WFS:GrundstuecksRechtecke>
    </gml:featureMember>
</wfs:FeatureCollection>
```

3.1.8 Methode GetGmlObject

Die Methdoe GetGmlObject bietet die Möglichkeit, mittels ID auf ein bestimmtes Feature zuzugreifen. Diese Methode ist im WFS-Standard optional definiert und muss nicht implementiert sein. Im Falle des WFS beim BEV ist diese Methode nicht vorhanden. GetGmlObject unterstützt folgende Parameter:

Parameter der Methode GetGmlObject

Parameter	Pflichtfeld	Beschreibung
VERSION=1.1.0	Nein	Version der Anfrage
SERVICE=WFS	Nein	Typ des Services
TRAVERSEXLINKDEPTH=n	Ja	Angabe, wie tief XLink-Referenzen verfolgt werden sollen
TRAVERSEXLINKEXPIRY= minutes	Nein	Dieser Parameter spezifiziert die Zeit in Minuten, die das WFS auf nachfolgende Anfragen mittels XLink-Referenzen wartet
REQUEST=GetGmlObject	Ja	Name der Methode
GMLOBJECTID=ID	Ja	XML-ID des verlangten Objektes

3.1.9 Beispiel für die Methode GetGmlObject

Aufgrund der fehlenden Unterstützung durch das WFS des BEV kann hier nur ein allgemeines Beispiel angegeben werden. Folgende URL fragt die ID InWaterA_1M.1234 eines vorangegangenen GetFeature Request ab.

GetGmlObject Beispiel

```
http://www.someserver.com/wfs.cgi?SERVICE=WFS&VERSION=1.1.0&
REQUEST=GetGmlObject&TRAVERSEXLINKDEPTH=2&TRAVERSEXLINKEXPIRY=1&
GMLOBJECTID=t1
```

GetGmlObject Beispiel

```
<Town gm:id="t1">
   <gml:name>Bedford</gml:name>
   <gml:directedNode orientation="+"> <!-- xlink:href="#n1" -->
      <gml:Node gml:id="n1">
         <gml:pointProperty
         xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall"/>
      </gml:Node>
   </gml:directedNode>
</Town>
```

3.1.10 Methode LockFeature

Um einen Änderungsprozess abbilden zu können, muss ein Feature zur Änderung gesperrt werden. Solange das Feature durch den Benutzer geändert wird, ist es für Änderungen durch andere Personen gesperrt. Da jedoch HTTP keinen Status auf Seiten des Servers zwischenspeichert, muss diese Sperre explizit durch den Benutzer angefordert werden. Die Methode LockFeature führt genau diese Sperre auf Seiten des Servers durch und bietet folgende Parameter:

Parameter der Methode LockFeature

Parameter	Pflichtfeld	Beschreibung
VERSION=1.1.0	Nein	Version der Anfrage
SERVICE=WFS	Nein	Typ des Services
TYPENAME=typname	Ja	Name des FeatureTypes, dessen Instanzen gesperrt werden sollen
EXPIRY=minutes	Nein	Dieser Parameter spezifiziert die Zeit in Minuten, die das WFS das Feature sperren soll
REQUEST=LockFeature	Ja	Name der Methode
LOCKACTION=[ALL | SOME]	Nein	Angabe, ob alle Features gelockt werden sollen oder ob es ausreichend ist, das nur einige der gewünschten Features gelockt werden
FEATUREID=ID	Nein	ID des Features
FILTER=filter	Nein	Filter, der die gewünschten Features einschränkt
BBOX=koordinaten	Nein	Angabe der Einschränkung mittels Koordinaten

3.1.11 Beispiel für die Methode LockFeature

Da diese Methode vom WFS des BEV ebenfalls nicht unterstützt wird, soll hier ein allgemeines Beispiel den Zugriff veranschaulichen.

LockFeature Beispiel

```
http://www.someserver.com/wfs.cgi?SERVICE=WFS&VERSION=1.1.0&
REQUEST=LockFeature&LOCKACTION=ALL&TYPENAME=INWATER_1M&
FILTER=(<Filter><Within><PropertyName>wkbGeom</PropertyName><gml:Envelope>
<gml:lowerCorner>10 10</gml:lowerCorner> <gml:upperCorner>20 20</gml:upperCorner> </
gml:Envelope></Within></Filter>)
```

Ergebnis ist bei diesen Methoden das gesperrte Feature auf der Serverseite.

3.1.12 Methode Transaction

Die Methode Transaction bildet den Abschluß nach einer Reihe von Änderungen. Dadurch werden die Änderungen in der Datenbank festgeschrieben. Die Unterstützung dieser Methode ist optional, wobei WFS, die diese Methode unterstützen, meistens als WFS-T bezeichnet werden. Es werden folgende Parameter unterstützt:

Parameter der Methode Transaction

Parameter	Pflichtfeld	Beschreibung
VERSION=1.1.0	Nein	Version der Anfrage
SERVICE=WFS	Nein	Typ des Services
REQUEST=Transaction	Ja	Name der Methode
OPERATION=Delete	Ja	Art der Operation
TYPENAME=typename	Ja	Name des FeatureTypes
RELEASEACTION=[ALL|SOME]	Nein	Gibt an, welche Locks freizugeben sind; wenn SOME angegeben wird, werden nur die veränderten Features freigegeben
FEATUREID=ID	Nein	ID des Features
FILTER=filter	Nein	Filter, der die gewünschten Features einschränkt
BBOX=koordinaten	Nein	Angabe der Einschränkung mittels Koordinaten

3.1.13 Beispiel für die Methode Transaction

Das WFS, welches vom BEV zur Verfügung gestellt wird, unterstützt keine Transaction Methode. Daher kann hier nur ein allgemeines Beispiel gezeigt werden. Folgende URL löscht das Feature mit der ID RoadL_1M.1013.

Transaction Beispiel

```
http://www.someserver.com/wfs.cgi?SERVICE=WFS&VERSION=1.1.0&
REQUEST=Transaction&OPERATION=Delete&FEATUREID=RoadL_1M.1013
```

Nach positiver Durchführung ist das Feature mit der gewünschten ID aus der Datenbank gelöscht.

3.1.14 Zusammenfassung

Diese Lerneinheit führt in die Thematik der Web Feature Services (WFS) ein. Es wird zuerst erklärt, was ein WFS ist und welche Methoden ein WFS bieten kann. Dabei wird zwischen verpflichtenden und optionalen Methoden unterschieden. Jede der Methoden wird im Anschluß durch ein Beispiel veranschaulicht.

3.2 Probleme bei Zugriffen auf WFS mittels OpenLayers

In dieser Lerneinheit wird erklärt, welche Probleme beim Zugriff auf WFS entstehen können. In der vorherigen Lerneinheit werden die Grundlagen zu WFS erklärt. Die einzelnen Methoden eines WFS können mit den beschriebenen URLs leicht ausgeführt werden. Sobald jedoch eine Webapplikation auf ein WFS zugreifen will, kommt es unweigerlich zu Problemen. Diese beruhen auf dem Sandboxkonzept in JavaScript, welches verhindert, das auf fremde Domains zugegriffen wird. Folgend wird das Problem genauer erläutert und in der nächsten Lerneinheit eine Lösungsmöglichkeit vorgestellt.

Lernziele

- Verstehen der Verwendung von JavaScript
- Erklären des Sandboxkonzeptes eines Browser
- Verwenden von AJAX und XMLHttpRequest
- Mittels OpenLayers auf ein WFS zugreifen

3.2.1 Was ist JavaScript

JavaScript ist eine Scriptsprache, die den Objektzugriff zulässt. Die angesprochenen Objekte können clientseitige DOM-Bäume eines Web-Browsers oder auch beliebige Objekte einer serverseitigen Implementierung unter Java sein. Die Einfachheit der Sprache zielt auf ein Publikum ab, welches keinen starken Hintergrund zum Programmieren besitzt. Clientseitig können mittels JavaScript alle Eigenschaften des Browsers ausgelesen und zum großen Teil auch verändert werden.
Der Sprachkern von JavaScript beruht auf ECMAScript und bildet eine

- dynamische
- schwach typisierte
- auf Prototypen basierte
- objektorientierte
- klassenlose
- funktionale

Sprache. JavaScript wurde durch viele Sprachen beeinflusst und dahingehend entwickelt, Java zu ähneln. Heutzutage unterstützen alle Browser JavaScript, da es auch ein Teil des HTML Standards ist.

3.2.2 Sandboxkonzept der Browser

Browser stellen eine sehr unsichere Art der Anwendungen dar. Sie eröffnen dem Benutzer die Möglichkeit, über das Internet auf andere Rechner zuzugreifen. Der Zugriff auf Ressourcen des eigenen Rechners sollte jedoch unterbunden werden. Dafür wurde das sogenannte Sandboxkonzept für Browser entwickelt. Alle HTML-Seiten befinden sich quasi in einer Sandkiste und können innerhalb dieser Grenzen jegliche zur Verfügung stehenden Methoden verwenden. Sobald eine HTML-Seite außerhalb der Sandbox einen Zugriff benötigt, wird dieser geprüft. Folgende Punkte werden durch den Browser unterbunden:

- Zugriff auf das Dateisystem
- Lesen, schreiben, löschen von Dateien
- Schließen des Browser
- Aus- und einblenden von Symbolleisten

- Veränderung der Startseite des Browsers
- Auslesen der Historie
- Auslesen von Cookies/Usernamen/Passwörter fremder Seiten
- Zugriff auf Webseiten, außer auf die Ursprungsseite der HTML-Seite

3.2.3 Was ist XMLHttpRequest?

XMLHttpRequest ist ein Objekt der JavaScript Bibliothek, welches (nicht standardisierten) asynchronen Zugriff auf Webseiten ermöglicht. Durch die Verwendung von XMLHttpRequest kann JavaScript Quellcode Inhalte von Webseiten abfragen und das Ergebnis in die aktuell angezeigte Webseite einfügen, welche für diesen Vorgang nicht aktualisiert werden muss. Der Vorteil dieser asynchronen Variante liegt in der gefühlten Aktivität der Webseite. Es entfällt der langwierige Neuaufbau der gesamten Seite, wenn nur gewisse Teile der Seite aktualisiert werden. Des Weiteren können auf diese Weise auch Pollingmechanismen implementiert werden, die auf Trigger am Server warten. Dies führt im Endeffekt zu einer Möglichkeit, dass der Server aktiv Daten an einen bestimmten Browser schickt. XMLHttpRequest spielt hierbei eine wichtige Rolle bei AJAX (Asynchronous JavaScript and XML), indem es den Teil der Asynchronität übernimmt.

3.2.4 Zugriff auf WFS mittels OpenLayers

Das Sandboxkonzept des Browser unterbindet einige Funktionalitäten, die als selbstverständlich gelten. Vor allem der letzte Punkt stellt für den Zugriff auf ein WFS ein Problem dar (Zugriff auf Inhalte von Webseiten mit anderer Domain als die Domain der aktuellen HTML-Seite). WFS werden von GIS-Servern zur Verfügung gestellt. Diese bilden eine Laufzeitumgebung für WFS und WMS, jedoch nicht für normale HTML-Seiten und kommen von einem Webserver. Daraus folgt, dass die Domains der beiden Server unterschiedlich sind. Die Domain ist jedoch der Teil der URL, anhand welcher der Browser entscheidet, ob der Zugriff auf die Ursprungsseite erfolgt oder nicht. Dadurch wird der Zugriff auf das WFS vom Browser unterbunden.
Um zu verstehen, warum das zum Problem wird, muss noch geklärt werden, wie OpenLayers auf Daten eines WFS zugreift. OpenLayers verwendet einen XMLHttpRequest, um Daten von einem WFS asynchron zu holen. Da XMLHttpRequest eine JavsScript API ist, greift das Sandboxkonzept. Folgende Abbildung zeigt, wie in OpenLayers mittels POST-Request auf ein WFS zugegriffen wird.

Zugriff auf ein WFS mittels OpenLayers

```
277                 OpenLayers.Request.POST({
278                     url: '<%= wfsSucheUrl %>',
279                     data: requestString,
280                     callback: handleResultRechtecke
281                 });
```

Der Aufruf enthält drei Parameter:

- url: enthält die URL des WFS
- data: enthält den XML-String, welcher an das WFS zu senden ist
- callback: referenziert eine Funktion, der das Ergebnis des Aufrufes übergeben wird

OpenLayers führt daraufhin den Aufruf aus, und übergibt die Antwort an die Callback-Methode. Der Aufruf erfolgt asynchron und der JavaScript-Code aus der Abbildung wartet nicht auf die Abarbeitung des Aufrufes. Es muss gewährleistet sein, dass die Verarbeitung des nachfolgenden JavaScript-Codes weiterläuft, auch wenn noch keine Daten vom WFS geladen wurden. Somit darf auch nicht davon ausgegangen werden, dass die Daten des WFS zur Verfügung stehen.

3.2.5 Zusammenfassung

Das Hauptaugenmerk dieser Lerneinheit liegt auf der Erklärung des Zugriffes auf ein WFS von OpenLayers. Dieser erfolgt mittels XMLHttpRequest in JavaScript, der zu Problemen aufgrund des Sandboxkonzeptes der Browser führt. Es wird erklärt, wie ein XMLHttpRequest funktioniert und warum dies zu Problemen führt. Abschließend wird gezeigt, wie mittels OpenLayers auf ein WFS zugegriffen werden kann.

3.3 Einrichtung eines ProxyServlets

In dieser Lerneinheit wird erklärt, wie die Zugriffsprobleme von OpenLayers mittels ProxyServlets behoben werden können. In der vorherigen Lerneinheit wurde das Problem mit dem Zugriff auf ein WFS beschrieben. Eine Lösung des Problems bietet ein Proxy, der auf dem gleichen Webserver deployt sein muss, wie die HTML-Seite mit dem Kartenfenster. Dadurch nimmt der Browser an, dass es sich um die gleiche Domain handelt und lässt den Zugriff auf das WFS zu. Der Proxy muss nur die Anfrage an das richtige WFS weiterleiten. Genauso muss auch mit der Antwort des WFS umgegangen werden, indem der Proxy diese als Antwort zum Browser weiterreicht. In diesem Kapitel wird nun die Vorgehensweise genauer erklärt.

Lernziele

- Erklären der ProxyHost Konfiguration in OpenLayers
- Programmieren eines ProxyHosts für GET- und POST-Zugriffe
- Verstehen von Parameter Encoding

3.3.1 Was ist ein ProxyHost in OpenLayers

Das Sandboxproblem ist den Entwicklern von OpenLayers ebenfalls bekannt, weswegen sie versucht haben, eine Lösung in die OpenLayers API einzubauen. Da die OpenLayers API nicht die Domain des WFS verändern kann, wird eine erleichterte Konfiguration eines Proxys geboten. Folgende Abbildung zeigt die Konfiguration des ProxyHosts in OpenLayers.

Proxyhost in OpenLayers

```
130        OpenLayers.ProxyHost="ProxyHost?";
```

Diese simple Konfiguration bewirkt, dass alle WFS-URLs nicht mehr direkt verwendet werden. Stattdessen werden alle Anfragen an die URL des ProxyHosts geschickt und die tatsächliche URL einfach angehängt. Ein Beispiel soll dies besser erklären. Folgende URL greift auf die Methode GetCapabilities des WFS zu:

Beispiel eines WFS Aufrufes ohne ProxyHost

```
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.1.0
```

Direkt in die Browseradresszeile eingegeben funktioniert der Zugriff problemlos. Sobald jedoch die URL (ohne ProxyHost) aus OpenLayers heraus verwendet wird, wird der Zugriff verweigert. Wenn nun der ProxyHost, wie oben beschrieben, konfiguriert ist, sieht die URL, die OpenLayers intern absetzt, wie folgt aus:

Beispiel eines WFS Aufrufes mit ProxyHost

```
ProxyHost?http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.1.0
```

Durch das Voranstellen von "ProxyHost?" wird aus der URL eine zur aktuellen Domain relative URL und diese fällt somit nicht mehr unter die Sandboxbeschränkung. Diese Art der Problemumgehung vordert jedoch zwei Voraussetzungen. Zum einen muss dieser ProxyHost bei der aktuellen Domain auch vorhanden sein, zum anderen wird alles nach dem ersten Fragezeichen decoded. Decoding stellt die Darstellung von verbotenen Zeichen in URL durch % und eine Nummer dar. Zum Beispiel ist ein Leerzeichen innerhalb einer URL als %20 dargestellt. Der ProxyHost kann in Form eines Servlets umgesetzt werden (siehe Beispiel für ein vollständiges ProxyServlet).

3.3.2 Proxy für GET-Requests

Ein HTTP-GET-Request enspricht einer Anfrage, bei der alle Parameter Teil der URL sind. Folgender GetCapabilities Aufruf stellt solch eine Anfrage dar.

Beispiel eines GET-Requests

```
http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.1.0
```

Um einen Proxy zu implementieren, der GET-Requests weiterleiten kann, müssen alle Parameter ausgelesen und weitergegeben werden. Glücklicherweise kann innerhalb eines Servlets die gesamte URL nach dem Servletnamen ermittelt werden. Somit kann dieser Teil der URL als neue URL verwendet werden. Folgender Ausschnitt zeigt den Zugriff auf die URL.

Zugriff auf den gesamten Parameterstring

```
47        String newUrl = request.getQueryString();
```

Mit dem ermittelten Parameterstring wird die neue URL erstellt. Wie bereits in Beispiel eines WFS Aufrufes mit ProxyHost gezeigt wird, hängt OpenLayers die tatsächliche URL einfach der ProxyHost-URL an. Daher ist der Parameterstring gleichzeitig die neue, authentische URL des WFS. Folgende Abbildung zeigt, wie daraus eine URL in Java erzeugt wird.

Erzeugen einer URL aus dem Parameterstring

```
53        URL url = new URL(newUrl);
```

Diese URL muss nur noch für den Zugriff auf das WFS verwendet werden.

Zugriff auf das WFS

```
66        URLConnection urlConnection = url.openConnection(proxy);
```

Die Antwort des WFS wird durch den Proxy ausgelesen und als eigene Antwort an den Benutzer weitergeleitet. Dies erfolgt, indem der InputStream der WFS-Anfrage in den OutputStream der Anfrage an den Proxy kopiert wird (siehe Abbildung).

Antwort weiterleiten

```
75          InputStream is = urlConnection.getInputStream();
76          OutputStream os = response.getOutputStream();
77          int i;
78          while((i = is.read()) > -1)
79          {
80              os.write(i);
81          }
82          os.close();
```

3.3.3 Proxy für POST-Requests

Ein POST-Request enthält im Gegensatz zum GET-Request die Parameter nicht als Teil der URL, sondern im Body des HTTP-Requests mitgeschickt. Der Vorteil liegt in der nahezu unbeschränkten Größe des Body. Des Weiteren können auf diese Weise auch File-Uploads erfolgen, die selber Parameter für das WFS darstellen. Alle bisher beschriebenen Methoden eines WFS und die zugehörigen Parameter können auch in Form von XML-Files definiert werden.
Um einen Proxy für POST-Request umzusetzen, muss der Inhalt des HTTP-Body an das WFS weitergeleitet werden. Dies erfolgt mittels Kopierns des InputStreams in den OutputStream der Anfrage an das WFS. Hierfür muss der Parameterstring als URL für die Anfrage an das WFS verwendet werden.

Zugriff auf den Parameterstring

```
47          String newUrl = request.getQueryString();
```

Mit dem Parameterstring wird die neue URL erzeugt.

Erzeugen einer URL aus dem Parameterstring

```
53          URL url = new URL(newUrl);
```

Mit der neuen URL wird eine Verbindung zum WFS aufgebaut.

Aufbau der Verbindung zum WFS

```
66          URLConnection urlConnection = url.openConnection(proxy);
```

Die neue Verbindung wird genutzt, um den HTTP-Body an das WFS zu übergeben. Folgende Abbildung zeigt die Implementierung des Umkopierens mittels Java-Servlets.

HTTP-Body kopieren

```
55          InputStream inputStream = request.getInputStream();
56          BufferedReader rd = new BufferedReader(new InputStreamReader(inputStream));
57          String data = "";
58          String line;
59          while((line = rd.readLine()) != null)
60          {
61              data += line;
62          }
63          rd.close();
64          config.getServletContext().log("Body: " + data);
65
66          URLConnection urlConnection = url.openConnection(proxy);
67          ((HttpURLConnection) urlConnection).setRequestProperty("Content-Type", "text/xml");
68          urlConnection.setDoOutput(true);
69          OutputStreamWriter wr = new OutputStreamWriter(urlConnection.getOutputStream());
70          wr.write(data);
71          wr.flush();
72          wr.close();
```

Zuvor muss noch der Content-Type auf text/xml gesetzt werden, da das WFS des BEV diesen Content-Type erwartet. Genau genommen ignoriert der Interceptor den HTTP-Body, wenn nicht text/xml ausgewählt wurde. Nach der Übergabe des XML-Files als Anfrage an das WFS wartet das Servlet auf die Antwort vom WFS. Diese Antwort muss an den Aufrufer zurückgegeben werden. Dies erfolgt durch Kopieren der XML-Antwort des WFS in die eigene Response. Folgende Abbildung zeigt das Vorgehen des Kopierens der Antwort.

Antwort kopieren

```
75      InputStream is = urlConnection.getInputStream();
76      OutputStream os = response.getOutputStream();
77      int i;
78      while((i = is.read()) > -1)
79      {
80          os.write(i);
81      }
82      os.close();
```

3.3.4 Was ist Parameter Encoding?

OpenLayers hängt die tatsächliche URL des WFS hinter die URL für den ProxyHost. Da jedoch eine URL bestimmte Zeichen nicht enthalten darf, müssen diese maskiert werden. Dieses Maskieren wird Encoding genannt. Zum Beispiel wird ein Leerzeichen innerhalb der URL als %20 dargestellt. Für das Encoding gibt es eine eindeutige Mapping-Liste. Aus diesem Grund ist das Maskieren auf Seiten des Servers reversibel und wird Decoding genannt. Im Falle des Prototyps müssen die Parameter decoded werden, da sonst die URL keine Gültigkeit hat. Das Decoding erfolgt zwischen dem Auslesen des Parameterstrings und dem Aufbau der neuen URL an das WFS. Folgende Abbildung zeigt den Quellcode zum Decoding der Parameter.

Parameter Decoding

```
47      String newUrl = request.getQueryString();
48      if(newUrl.contains("%"))
49      {
50          newUrl = URLDecoder.decode(newUrl, "UTF-8");
51      }
52      config.getServletContext().log("URL: " + newUrl);
53      URL url = new URL(newUrl);
```

Das Decoding auf Seiten des Servers erfolgt mittels Objekt URLDecoder. Die Abbildung zeigt den gesamten Vorgang vom Auslesen des Parameterstings über das Decoding (falls % vorhanden und somit encoded) bis hin zum Anlegen der neuen URL.

3.3.5 Beispiel für ein vollständiges ProxyServlet

Die vorherigen Kapitel haben die einzelnen Teile eines sinnvollen ProxyServlets erklärt. Hier soll nun ein gesamtes Servlet für GET- und POST-Requests inklusive Decoding gezeigt werden. Folgende Abbildung zeigt den Initialisierungsteil des Servlets. Hier wird ausschließlich die IP eines etwaigen HTTP-Proxys zwischengespeichert. Wenn kein Proxy für den Internetzugriff benötigt wird, ist dieser Teil hinfällig.

Initialisierung des ProxyServlets

```java
package com.pe;

import java.io.*;

import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.SocketAddress;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLDecoder;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 *
 * @author pezerovic
 */
public class ProxyHostServlet extends HttpServlet
{

    private static final long serialVersionUID = 72164293077472571381;

    private Proxy proxy;

    private ServletConfig config;

    @Override
    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);
        SocketAddress proxyAddress = new InetSocketAddress("10.2.1.101", 8080);
        proxy = new Proxy(Proxy.Type.HTTP, proxyAddress);
        this.config = config;
    }
```

Die Methode processRequest wird bei jeder Anfrage an das Servlet aufgerufen und ist für folgende Punkte zuständig:

- Setzen des Response Content-Types auf XML
- Auslesen des Parameterstrings
- Decoding der Parameter
- Erstellen der neuen URL
- Auslesen des POST-Bodys
- Aufbau der Verbindung zum WFS mittels neuer URL
- Weiterleiten des POST-Bodys
- Auslesen der WFS-Antwort und kopieren als eigene Antwort

Folgende Abbildung zeigt den dazugehörigen Quellcode der Methode processRequest.

Methode processRequest des ProxyServlets

```
37      /**
38       * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
39       * @param request servlet request
40       * @param response servlet response
41       */
42      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
43          throws ServletException, IOException
44      {
45          response.setContentType("application/xml");
46
47          String newUrl = request.getQueryString();
48          if(newUrl.contains("%"))
49          {
50              newUrl = URLDecoder.decode(newUrl, "UTF-8");
51          }
52          config.getServletContext().log("URL: " + newUrl);
53          URL url = new URL(newUrl);
54
55          InputStream inputStream = request.getInputStream();
56          BufferedReader rd = new BufferedReader(new InputStreamReader(inputStream));
57          String data = "";
58          String line;
59          while((line = rd.readLine()) != null)
60          {
61              data += line;
62          }
63          rd.close();
64          config.getServletContext().log("Body: " + data);
65
66          URLConnection urlConnection = url.openConnection(proxy);
67          ((HttpURLConnection) urlConnection).setRequestProperty("Content-Type", "text/xml");
68          urlConnection.setDoOutput(true);
69          OutputStreamWriter wr = new OutputStreamWriter(urlConnection.getOutputStream());
70          wr.write(data);
71          wr.flush();
72          wr.close();
73
74
75          InputStream is = urlConnection.getInputStream();
76          OutputStream os = response.getOutputStream();
77          int i;
78          while((i = is.read()) > -1)
79          {
80              os.write(i);
81          }
82          os.close();
83      }
```

Die restlichen Methoden des ProxyServlets bilden die Standardmethoden eines Servlets. Die Methoden doGet und doPost leiten die Verarbeitung an die Methode processRequest weiter.

Standardmethoden des ProxyServlets

```
85      // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
86      /**
87       * Handles the HTTP <code>GET</code> method.
88       * @param request servlet request
89       * @param response servlet response
90       */
91      @Override
92      protected void doGet(HttpServletRequest request, HttpServletResponse response)
93          throws ServletException, IOException
94      {
95          processRequest(request, response);
96      }
97
98      /**
99       * Handles the HTTP <code>POST</code> method.
100      * @param request servlet request
101      * @param response servlet response
102      */
103     @Override
104     protected void doPost(HttpServletRequest request, HttpServletResponse response)
105         throws ServletException, IOException
106     {
107         processRequest(request, response);
108     }
109
110     /**
111      * Returns a short description of the servlet.
112      */
113     @Override
114     public String getServletInfo()
115     {
116         return "Short description";
117     }
118     // </editor-fold>
119 }
120
```

3.3.6 Zusammenfassung

Um die Probleme des Sandboxkonzeptes der Browser zu umgehen, wird in dieser Lerneinheit erklärt, wie ein Proxy für den Zugriff von OpenLayers aus erstellt wird. Der Proxy behandelt sowohl GET- als auch POST-Requests und bietet somit die Möglichkeit, Zugriffe auf das WFS mittels Key-Value-Pairs (KVP) oder XML-Dokumente als Anhang (POST) automatisch weiterzuleiten. Bei KVP müssen hierfür die Parameter decoded werden, weswegen die Begriffe Encoding und Decoding erläutert werden. Abschließend wird ein umfassendes Beispiel für ein Servlet gezeigt, welches alle Punkte des Proxys vereint.

3.4 Queries und Filter Encoding

In dieser Lerneinheit wird erklärt, wie Abfragen an WFS abgesetzt und auch eingeschränkt werden können. Im Gegensatz zu einem WMS ist bei einem WFS selten das Ziel, alle Features eines Kartenausschnittes zu ermitteln. Vielmehr geht es meist darum, eine Art der Suche mittels WFS abzubilden. Das Ergebnis der Suche kann innerhalb des aktuellen Kartenausschnittes liegen, muss jedoch nicht. Stattdessen wird eher das Ergebnis eines WFS-Aufrufes dazu dienen, in einen neuen Kartenausschnitt zu pannen. Daher definiert der WFS-Standard eine Schnittstelle, wie Abfragen eingeschränkt werden können. Die folgenden Kapitel werden diese Punkte genauer beleuchten.

Lernziele

- Kennen der zur Verfügung stehenden WFS
- Verstehen der Queries bei WFS
- Verwenden von Filter Encoding
- Erklären der Verwendung von Spatial Operators
- Einsetzen von Comparison Operators
- Einschränken des Ergebnises mittels Logical Operators

3.4.1 Welche WFS stehen zur Verfügung?

Für dieses Tutorial wurde vom BEV ein WFS zur Verfügung gestellt. Es bietet 3 FeatureTypes an, die im folgenden gezeigt werden sollen. Die folgende Abbildung zeigt die drei FeatureTypes, wie sie von der Methode GetCapabilities geliefert werden.

FeatureTypes des WFS

```
<wfs:FeatureTypeList>
  <wfs:FeatureType>
    <wfs:Name>ASP_WFS:GrundstuecksRechtecke</wfs:Name>
    <wfs:Title>GrundstuecksRechtecke</wfs:Title>
    <wfs:DefaultSRS>EPSG:31255</wfs:DefaultSRS>
    <wfs:OtherSRS>EPSG:4326</wfs:OtherSRS>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
    </wfs:OutputFormats>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>9.4052369537545228 46.327670137886301</ows:LowerCorner>
      <ows:UpperCorner>17.23375146832625 49.032104102733868</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  </wfs:FeatureType>
  <wfs:FeatureType>
    <wfs:Name>ASP_WFS:GrundstuecksEinsetzpunkte</wfs:Name>
    <wfs:Title>GrundstuecksEinsetzpunkte</wfs:Title>
    <wfs:DefaultSRS>EPSG:31255</wfs:DefaultSRS>
    <wfs:OtherSRS>EPSG:4326</wfs:OtherSRS>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
    </wfs:OutputFormats>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>9.4052369537545228 46.327670137886301</ows:LowerCorner>
      <ows:UpperCorner>17.23375146832625 49.032104102733868</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  </wfs:FeatureType>
  <wfs:FeatureType>
    <wfs:Name>ASP_WFS:KatastralgemeindenRechtecke</wfs:Name>
    <wfs:Title>KatastralgemeindenRechtecke</wfs:Title>
    <wfs:DefaultSRS>EPSG:31255</wfs:DefaultSRS>
    <wfs:OtherSRS>EPSG:4326</wfs:OtherSRS>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
    </wfs:OutputFormats>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>9.4327886435610484 46.401035227143304</ows:LowerCorner>
      <ows:UpperCorner>17.177014305348173 49.029002596024235</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  </wfs:FeatureType>
</wfs:FeatureTypeList>
```

Der FeatureType ASP_WFS:GrundstuecksRechtecke liefert (wie der Name schon andeutet) die minimalen Rechtecke, die das Grundstück noch beinhalten. Dies wird oft auch als BoundingBox bezeichnet. Der zweite FeatureType ASP_WFS:KatastralgemeindenRechtecke liefert dementsprechend die minimalen Rechtecke um die Polygone der Katastralgemeinden. Der dritte FeatureType ASP_WFS:GrundstuecksEinsetzpunkte enthält diejenigen Punkte, an denen die Grundstücksnummer in der Digitalen Katastralmappe (DKM) eingesetzt wird. Jeder dieser FeatureTypes hat zusätzliche Attribute, die ein Aufruf der Methode DescribeFeatureType genauer beschreibt. Folgende Abbildung zeigt die Beschreibung des FeatureTypes ASP_WFS:GrundstuecksRechtecke.

DescribeFeatureType GrundstuecksRechtecke (Teil 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
           xmlns:gml="http://www.opengis.net/gml"
           elementFormDefault="qualified"
           targetNamespace="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer">
    <xs:import namespace="http://www.opengis.net/gml"
               schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
    <xs:element name="GrundstuecksRechtecke"
                substitutionGroup="gml:_Feature"
                type="ASP_WFS:GrundstuecksRechteckeType"/>
    <xs:complexType name="GrundstuecksRechteckeType">
        <xs:complexContent>
            <xs:extension base="gml:AbstractFeatureType">
                <xs:sequence>
                    <xs:element name="OBJECTID" type="xs:int"/>
                    <xs:element name="GNR">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="11"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="RSTATUS">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="1"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="TYP" type="xs:short"/>
                    <xs:element name="KG">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="5"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="KGGNR">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="16"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="GEMNR" type="xs:int"/>
                    <xs:element minOccurs="0" name="BEZNR" type="xs:short"/>
                    <xs:element minOccurs="0" name="LANDNR" type="xs:short"/>
```

DescribeFeatureType GrundstuecksRechtecke (Teil 2)

```
                    <xs:element minOccurs="0" name="VANR">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="2"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="XMIN" type="xs:double"/>
                    <xs:element minOccurs="0" name="YMIN" type="xs:double"/>
                    <xs:element minOccurs="0" name="XMAX" type="xs:double"/>
                    <xs:element minOccurs="0" name="YMAX" type="xs:double"/>
                    <xs:element minOccurs="0" name="KGNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="GEMNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="BEZNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="VANAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="LANDNAME">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element minOccurs="0" name="SHAPE" type="gml:PolygonPropertyType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

Die folgende Abbildung zeigt die Beschreibung der Attribute des FeatureTypes ASP_WFS:KatastralgemeindenRechtecke.

DescribeFeatureType KatastralgemeindenRechtecke (Teil 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
           xmlns:gml="http://www.opengis.net/gml"
           elementFormDefault="qualified"
           targetNamespace="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer">
  <xs:import namespace="http://www.opengis.net/gml"
             schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd" />
  <xs:element name="KatastralgemeindenRechtecke" substitutionGroup="gml:_Feature"
              type="ASP_WFS:KatastralgemeindenRechteckeType" />
  <xs:complexType name="KatastralgemeindenRechteckeType">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="OBJECTID" type="xs:int" />
          <xs:element minOccurs="0" name="KG">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="5" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element minOccurs="0" name="KGNR" type="xs:int" />
          <xs:element minOccurs="0" name="XMIN" type="xs:double" />
          <xs:element minOccurs="0" name="YMIN" type="xs:double" />
          <xs:element minOccurs="0" name="XMAX" type="xs:double" />
          <xs:element minOccurs="0" name="YMAX" type="xs:double" />
          <xs:element minOccurs="0" name="KGNR_C">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="5" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element minOccurs="0" name="GEMNR" type="xs:int" />
          <xs:element minOccurs="0" name="BEZNR" type="xs:short" />
          <xs:element minOccurs="0" name="LANDNR" type="xs:short" />
          <xs:element minOccurs="0" name="VANR">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="2" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
```

DescribeFeatureType KatastralgemeindenRechtecke (Teil 2)

```
45          <xs:element minOccurs="0" name="KGNAME">
46            <xs:simpleType>
47              <xs:restriction base="xs:string">
48                <xs:maxLength value="50" />
49              </xs:restriction>
50            </xs:simpleType>
51          </xs:element>
52          <xs:element minOccurs="0" name="GEMNAME">
53            <xs:simpleType>
54              <xs:restriction base="xs:string">
55                <xs:maxLength value="50" />
56              </xs:restriction>
57            </xs:simpleType>
58          </xs:element>
59          <xs:element minOccurs="0" name="BEZNAME">
60            <xs:simpleType>
61              <xs:restriction base="xs:string">
62                <xs:maxLength value="50" />
63              </xs:restriction>
64            </xs:simpleType>
65          </xs:element>
66          <xs:element minOccurs="0" name="VANAME">
67            <xs:simpleType>
68              <xs:restriction base="xs:string">
69                <xs:maxLength value="50" />
70              </xs:restriction>
71            </xs:simpleType>
72          </xs:element>
73          <xs:element minOccurs="0" name="LANDNAME">
74            <xs:simpleType>
75              <xs:restriction base="xs:string">
76                <xs:maxLength value="50" />
77              </xs:restriction>
78            </xs:simpleType>
79          </xs:element>
80          <xs:element minOccurs="0" name="SHAPE" type="gml:PolygonPropertyType" />
81        </xs:sequence>
82      </xs:extension>
83    </xs:complexContent>
84  </xs:complexType>
85</xs:schema>
```

Die folgende Abbildung zeigt die Beschreibung der Attribute des FeatureTypes ASP_WFS:GrundstuecksEinsetzpunkte.

DescribeFeatureType GrundstuecksEinsetzpunkte (Teil 1)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  targetNamespace="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
            elementFormDefault="qualified"
            xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:gml="http://www.opengis.net/gml">
  <xs:import namespace="http://www.opengis.net/gml"
             schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd" />
  <xs:element name="GrundstuecksEinsetzpunkte"
              type="ASP_WFS:GrundstuecksEinsetzpunkteType"
              substitutionGroup="gml:_Feature" />
  <xs:complexType name="GrundstuecksEinsetzpunkteType">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="OBJECTID" type="xs:int" />
          <xs:element name="GNR">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="11" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="RSTATUS">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="1" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="TYP" type="xs:short" />
          <xs:element name="KG">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="5" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="KGGNR">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="16" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element minOccurs="0" name="GEMNR" type="xs:int" />
          <xs:element minOccurs="0" name="BEZNR" type="xs:short" />
          <xs:element minOccurs="0" name="LANDNR" type="xs:short" />
          <xs:element minOccurs="0" name="VANR">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="2" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
```

DescribeFeatureType GrundstuecksEinsetzpunkte (Teil 2)

```
            <xs:element minOccurs="0" name="XMIN" type="xs:double" />
            <xs:element minOccurs="0" name="YMIN" type="xs:double" />
            <xs:element minOccurs="0" name="XMAX" type="xs:double" />
            <xs:element minOccurs="0" name="YMAX" type="xs:double" />
            <xs:element minOccurs="0" name="KGNAME">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:maxLength value="50" />
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element minOccurs="0" name="GEMNAME">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:maxLength value="50" />
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element minOccurs="0" name="BEZNAME">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:maxLength value="50" />
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element minOccurs="0" name="VANAME">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:maxLength value="50" />
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element minOccurs="0" name="LANDNAME">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:maxLength value="50" />
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element minOccurs="0" name="SHAPE"
              type="gml:PointPropertyType" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

3.4.2 Queries in WFS

Abfragen gegen WFS werden im WFS-Standard Queries genannt. Eine solche Query enthält Filter, die die Einschränkungen des Ergebnisses definieren. Wenn auf das WFS mittels GET-Request zugegriffen wird, enthält der Parameter FILTER den Filter der Query. Bei einem POST-Request wird das gesamte Query-Element im XML-Baum der Anfrage mitgeschickt. Folgende URL zeigt, wie ein Filter bei einem GET-Request mitgegegen werden kann.

GetFeature GrundstuecksRechteck

http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVIntern-e78108a1-94ba-4872-afa0-736a41097b0c? REQUEST=GetFeature&typename=ASP_WFS:GrundstuecksRechtecke&filter=<Filter+xmlns="http://www.opengis.net/ogc"><PropertyIsEqualTo><PropertyName>KGGNR</PropertyName><Literal>910111992</Literal></PropertyIsEqualTo></Filter>

Der Prototyp, der im Zuge dieses Tutorials erstellt wird, greift mittels POST-Requests auf das WFS zu. Folgende Abbildung zeigt den Aufbau eines Query-Elementes für die Suche nach Grundstücken.

Eröffnender Query-Tag

```
80                '<Query typeName="ASP_WFS:' + stringGrundstuecksRechtecke + '" srsName="<%= epsgCode %>" > ';
```

Filter-Tag

```
90              '<ogc:Filter>' +
91                '<ogc:And>' +
92                  '<ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">' +
93                    '<ogc:PropertyName>' +
94                      'GNR' +
95                    '</ogc:PropertyName>' +
96                    '<ogc:Literal>';
97            searchFilterInBetween =
98                    '</ogc:Literal>' +
99                  '</ogc:PropertyIsLike>' +
100                 '<ogc:PropertyIsEqualTo>' +
101                   '<ogc:PropertyName>' +
102                     'KG' +
103                   '</ogc:PropertyName>' +
104                   '<ogc:Literal>';
105           searchFilterEnd =
106                   '</ogc:Literal>' +
107                 '</ogc:PropertyIsEqualTo>' +
108               '</ogc:And>' +
109             '</ogc:Filter>';
```

Schließender Query-Tag

```
86                '</Query> ' +
```

Die Abbildung enthält die JavaScript-Strings, weswegen die Tags alle unter Hochkomma angegeben werden. Des Weiteren sind im Filter-Tag Variablen definiert, an deren Stelle die Suchwerte stehen werden. Die so definierten Variableninhalte ergeben zum Zeitpunkt des Requests einen kompletten XML-Filter Encoding Baum. Folgende Abbdilung zeigt solch einen XML-Request.

Beispielhafter Request

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <GetFeature xmlns="http://www.opengis.net/wfs"
3              xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
4              xmlns:ogc="http://www.opengis.net/ogc"
5              xmlns:gml="http://www.opengis.net/gml"
6              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7              xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd
8                                  http://www.opengis.net/gml/3.1/geometry.xsd"
9              service="WFS"
10             version="1.1.0"
11             outputFormat="text/xml; subtype=gml/3.1.1"
12             maxFeatures="10"
13             handle="" >
14     <Query typeName="ASP_WFS:GrundstuecksRechtecke" srsName="EPSG:31255" >
15         <ogc:Filter>
16             <ogc:And>
17                 <ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">
18                     <ogc:PropertyName>GNR</ogc:PropertyName>
19                     <ogc:Literal>37/*</ogc:Literal>
20                 </ogc:PropertyIsLike>
21                 <ogc:PropertyIsEqualTo>
22                     <ogc:PropertyName>KG</ogc:PropertyName>
23                     <ogc:Literal>23431</ogc:Literal>
24                 </ogc:PropertyIsEqualTo>
25             </ogc:And>
26         </ogc:Filter>
27     </Query>
28 </GetFeature>
29
```

3.4.3 Einschränkungen mittels Filter Encoding

Wie im vorherigen Kapitel bereits erklärt, dienen Filter Encodings dazu, das Ergebnis eines WFS einzuschränken. Dabei können verschiedene Arten von Operatoren auf die übergebenen Werte angewendet werden. Folgende Liste zeigt einen Überblick über die verschiedenen Gruppen von Operatoren:

- Spatial Operators: Einschränkung des Ergebnisses aufgrund von geografischen Kriterien
- Comparison Operators: Einschränkung des Ergebnisses aufgrund von Vergleichen mit Werten
- Logical Operators: dienen dem logischen Verketten von mehrfachen Spatial und Comparison Operators

Die einzelnen Gruppen werden in den folgenden Kapiteln genauer beschrieben.

3.4.4 Einschränkung mittels Spatial Operators

Ein Spatial Operator dient dazu, das Ergebnis aufgrund von geografischen Kriterien einzuschränken. Dies erfolgt durch Abfragen, deren Argument ein geografisches Feature sein muss. Folgende Operator existieren:

- Equals: Features sind ident
- Disjoint: Features sind weder ident noch berühren oder überdecken sie sich
- Touches: Features berühren sich nur mit ihren Grenzen
- Within: Abfrage-Feature ist innerhalb des WFS-Features
- Overlaps: Features überdecken sich zumindest teilweise
- Crosses: die Grenze des Abfrage-Features schneidet die Grenze des WFS-Features
- Intersects: Features berühren, schneiden oder überdecken sich
- Contains: WFS-Feature ist innerhalb des Abfrage-Features
- DWithin: genau wie Within nur mit Buffer, der als Distance angegeben wird
- Beyond: genau wie Disjoint nur mit Buffer, der als Distance angegeben wird
- BBOX: entspricht Not Disjoint

Diese Abfragen erfolgen auf Serverseite mittels Verschneidung des übergebenen Features und dem WFS-Layer in der Datenbank. Das Ergebnis einer Einschränkung ist immer True oder False.

3.4.5 Einschränkung mittels Comparison Operators

Die Abfrage kann auch mittels Werten in Attributen des WFS-Layers eingeschränkt werden. Dafür werden die Comparison Operators verwendet. Das Ergebnis einer Einschränkung ist immer True oder False. Folgende Liste zeigt einen Überblick über die verwendbaren Operatoren:

- PropertyIsEqualTo: entspricht der Gleichheit der Werte
- PropertyIsNotEqualTo: entspricht der Ungleichheit der Werte
- PropertyIsLessThan: Attributwert ist kleiner als der übergebene Wert
- PropertyIsGreaterThan: Attributwert ist größer als der übergebene Wert
- PropertyIsLessThanOrEqualTo: Attributwert ist kleiner gleich dem übergebenen Wert
- PropertyIsGreaterThanOrEqualTo: Attributwert ist größer gleich dem übergebenen Wert
- PropertyIsLike: Attributwert wird auf Gleichheit geprüft, wobei wildcards im übergebenen Wert vorhanden sein dürfen
- PropertyIsNull: Attributwert ist nicht belegt

- PropertyIsBetween: Attributwert liegt zwischen zwei übergebenen Werten (Gleichheit mit einem der übergebenen Werte gilt als dazwischen)

3.4.6 Einschränkung mittels Logical Operators

Um komplexe Abfragen definieren zu können, müssen einzelne Einschränkungen kombiniert werden können. Hierfür sind die Logical Operators zuständig. Dabei werden die Ergebnisse der Einzelabfragen/ Einzeleinschränkungen (True oder False) miteinander kombiniert, um ein endgültiges Ergebnis zu erhalten (wieder True oder False). Folgende Liste zeigt einen Überblick über die vorhandenen logischen Operatoren:

- And: entspricht einem logischen Und
- Or: entspricht einem logischen Oder
- Not: entspricht einem logischen Nicht

3.4.7 Zusammenfassung

In dieser Lerneinheit wird erklärt, wie Zugriffe auf ein WFS inhaltlich eingeschränkt werden können. Hierfür werden Queries und die Spezifikation der Filter Encoding verwendet. Die Filter Encoding Spezifikation wird im Anschluß vorgestellt und die einzelnen Einschränkungsmöglichkeiten erläutert. Dabei werden Beispiele für Spatial, Comparison und Logical Operators gezeigt.

3.5 Zusammenfassung

Dieser Kurs befasst sich mit Zugriffen auf Web Feature Services und die Probleme, die dabei auftreten können. Dabei werden Begriffe wie WFS, Sandboxkonzept, KVP, Queries und Filter Encoding erläutert. Folgend ein kurzer Überblick über die einzelnen Lerneinheiten des Kurses.

Die erste Lerneinheit führt in die Thematik der Web Feature Services (WFS) ein. Es wird zuerst erklärt, was ein WFS ist und welche Methoden ein WFS bieten kann. Dabei wird zwischen verpflichtenden und optionalen Methoden unterschieden. Jede der Methoden wird im Anschluß durch ein Beispiel veranschaulicht.

Das Hauptaugenmerk der Lerneinheit 2 liegt auf der Erklärung des Zugriffes auf ein WFS von OpenLayers. Dieser erfolgt mittels XMLHttpRequest in JavaScript, der zu Problemen aufgrund des Sandboxkonzeptes der Browser führt. Es wird erklärt, wie ein XMLHttpRequest funktioniert und warum dies zu Problemen führt. Abschließend wird gezeigt, wie mittels OpenLayers auf ein WFS zugegriffen werden kann.

Um die Probleme des Sandboxkonzeptes der Browser zu umgehen, wird in der dritten Lerneinheit erklärt, wie ein Proxy für den Zugriff von OpenLayers aus erstellt wird. Der Proxy behandelt sowohl GET- als auch POST-Requests und bietet somit die Möglichkeit, Zugriffe auf das WFS mittels Key-Value-Pairs (KVP) oder XML-Dokumente als Anhang (POST) automatisch weiterzuleiten. Bei KVP müssen hierfür die Parameter decoded werden, weswegen die Begriffe Encoding und Decoding erläutert werden. Abschließend wird ein umfassendes Beispiel für ein Servlet gezeigt, welches alle Punkte des Proxys vereint.

In der Lerneinheit 4 wird erklärt, wie Zugriffe auf ein WFS inhaltlich eingeschränkt werden können. Hierfür werden Queries und die Spezifikation der Filter Encoding verwendet. Die Filter Encoding Spezifikation wird im Anschluß vorgestellt und die einzelnen Einschränkungsmöglichkeiten erläutert. Dabei werden Beispiele für Spatial, Comparison und Logical Operators gezeigt.

3.6 Abbildungsverzeichnis

3.7 Tabellenverzeichnis

4. Suche und Anzeige des Suchergebnisses im Kartenfenster

Diese eLesson vermittelt das Wissen, um nach Vektordaten zu suchen und das Ergebnis im Kartenfenster anzuzeigen. Für die Suche nach geografischen Daten werden Web Feature Services (WFS) verwendet. Das Ergebnis wird mittels Fähnchen in der Karte markiert und die Karte auf das Ergebnis ausgerichtet. Der genaue Vorgang wird in den einzelnen Lerneinheiten vorgestellt. Nach Fertigstellung dieser eLesson ist der Web-Client in der Lage, Suchergebnisse im Kartenfenster darzustellen. Folgende Abbildung zeigt solch ein Suchergebnis.

Markierung des Suchergebnisses

Lernziele

- Definieren eines Vectorlayers in OpenLayers
- Befüllen des Vectorlayers in OpenLayers mit GML-Objekten
- Erklären der Verwendung von Markern
- Anzeige von Markern in einem Vectorlayer
- Definition einer Bounding Box
- Durchführung einer Suche nach Bounding Boxes
- Erstellen einer umschliessenden Bounding Box um eine Menge von GML-Objekten
- Durchführen einer Suche nach den Einsetzpunkte der Grundstücksnummern
- Parsen der GML-Antwort des WFS
- Einfügen der Marker an den Einsetzpunkten der Grundstücksnummern
- Verstehen der Probleme bei der Verwendung von CRS84 und EPSG4326

4.1 Grundlagen zum Thema Vectorlayers

In dieser Lerneinheit werden die Grundlagen zum Thema Vectorlayer in OpenLayers behandelt. Vectorlayer können beliebige GML-Objekte aufnehmen und darstellen, wobei die Interpretation des GML-Objektes zur Gänze dem Vectorlayer überlassen wird.

Lernziele

- Definieren eines Vectorlayers in OpenLayers
- Befüllen des Vectorlayers in OpenLayers mit GML-Objekten
- Erklären der Verwendung von Markern
- Anzeige von Markern in einem Vectorlayer

4.1.1 Was ist ein Vectorlayer in OpenLayers?

Wie schon erwähnt, besteht in OpenLayers eine Map (Karte) aus mehreren übereinander liegenden Layer. Diese können ein- und ausgeschalten werden, um eine endgültige Karte zu erzeugen. Bisher wurden bei dem hier vorgestellten Beispiel nur Layer von Web Map Services (WMS) verwendet. Es können jedoch viele andere Arten von Layer in einem Kartenfenster angezeigt werden, wie zum Beispiel auch VectorLayer. Ein VectorLayer hat die Eigenschaft, dass er beliebige GML-Objekte beinhalten kann. Dies bietet den Vorteil, einen Layer komplett dynamisch mittels JavaScript erzeugen zu können. Solange nur GML-Objekte zum Layer hinzugefügt werden, zeigt OpenLayers diese auch an.

4.1.2 Befüllung des Vectorlayers

Der VectorLayer stellt GML-Objekte im Kartenfenster dar. Diese Objekte müssen nur zum Layer hinzugefügt werden. Folgende Abbildung zeigt, wie vorhandene Features des VectorLayers entfernt und durch neue ersetzt werden.

Features in VectorLayer ersetzen

```
356             vectorLayer.destroyFeatures();
357             vectorLayer.addFeatures(features);
```

Mit diesen zwei Befehlen (destroyFeatures und addFeatures) wird der Inhalt des VectorLayers ersetzt. Damit der Inhalt im Kartenfenster angezeigt wird, muss der Layer zur Map hinzugefügt worden sein. Folgende Abbildung zeigt, wie der VectorLayer zur Map hinzugefügt wird.

VectorLayer zur Map hinzufügen

```
252             vectorLayer = new OpenLayers.Layer.Vector("Suchergebnis", {style: layer_style});
253             map.addLayer(vectorLayer);
```

Obiger Befehl fügt den Vectorlayer zur Map hinzu. Sollte sich der Inhalt des VectorLayers nach dem Hinzufügen zur Map ändern, muss dies der Map mitgeteilt werden. Der Grund ist, dass der VectorLayer neu gezeichnet werden muss und im internen Cache von OpenLayers die alte Darstellung zwischengespeichert ist. Folgende Abbildung zeigt, wie die Darstellung des VectorLayers erneuert werden kann.

VectorLayer neu zeichen

```
367             vectorLayer.redraw();
```

4.1.3 Beispiel eines Vectorlayers

Der VectorLayer kann als Teil der Karte angezeigt werden. Genau wie die anderen Layer ist es auch möglich, den VectorLayer auszublenden. Folgende Grafik zeigt einen VectorLayer, der Punkt-Features enthält, deren Darstellung als Fähnchen realisiert ist.

Aktivierter VectorLayer

4.1.4 Was sind Marker?

Wie der Name schon sagt, markieren Marker gewisse Punkte. In OpenLayers ist ein Marker eine Repräsentanz eines Punktobjektes. Die Darstellungsform des Punktobjektes kann in OpenLayers zusätzlich gewählt werden. Zum Beispiel kann ein Marker in Form eines Fähnchens angezeigt werden. Auf jeden Fall kennzeichnet ein Marker Punkte von besonderem Interesse, wie zum Beispiel das Suchergebnis.

4.1.5 Anzeige von Marker im Vectorlayer

Um Marker in OpenLayers darzustellen, muss ein VectorLayer erzeugt werden. Bei der Instanziierung des VectorLayers wird die Darstellungsform der Marker angegeben. Hierbei sind einige Konfigurierungsmöglichkeiten gegeben. Folgende Abbildung zeigt, wie ein VectorLayer unter Angabe der Darstellungsform angelegt werden kann.

VectorLayer mit Marker-Konfiguration

```
252             vectorLayer = new OpenLayers.Layer.Vector("Suchergebnis", {style: layer_style});
253             map.addLayer(vectorLayer);
```

Die Abbildung zeigt, dass die Darstellungskonfiguration mittels sogenannter Stile erfolgt. Der gewünschte Stil wird dem Konstruktor des Objektes OpenLayers.Layer.Vector als Parameter mitgegeben, wobei der Name des Parameters 'style' lautet. Der Inhalt des Parameters ist vom Typ OpenLayers.Feature.Vector.Style. Dieses Objekt beinhaltet die möglichen Konfigurationen der Marker. Die nachfolgende Abbildung zeigt eine beispielhafte Konfiguration.

Stil-Konfiguration

```
241             layer_style = OpenLayers.Util.extend({}, OpenLayers.Feature.Vector.style['default']);
242             layer_style.graphicOpacity = 1;
243             layer_style.pointRadius = 12;
244             layer_style.graphicYOffset = -23;
245             layer_style.graphicXOffset = -11;
246             layer_style.fillOpacity = 0.0;
247             layer_style.externalGraphic = "openlayers/img/marker.png";
248             layer_style.fillColor = "#FFFFFF";
249             layer_style.strokeColor = "#FF0000";
250             layer_style.strokeWidth = 3;
```

Die einzelnen Einstellmöglichkeiten werden im Anschluss in Kürze erklärt. Grundsätzlich kann ein Marker in Form einer externen Grafik oder eines gezeichneten Kreises dargestellt werden. Beide Formenen haben eigene Attribute, die die Darstellung steuern.

Marker als Grafik

- graphicOpacity: Transparenz der Markergrafik
- graphicYOffset: Versatz der Grafik in Y-Richtung; dient der Ausrichtung der Grafik, da der "hot spot" die linke untere Ecke bildet; hiermit wird dieser hot spot verschoben
- graphicXOffset: Versatz der Grafik in X-Richtung
- externalGraphic: URL der anzuzeigenden Grafik

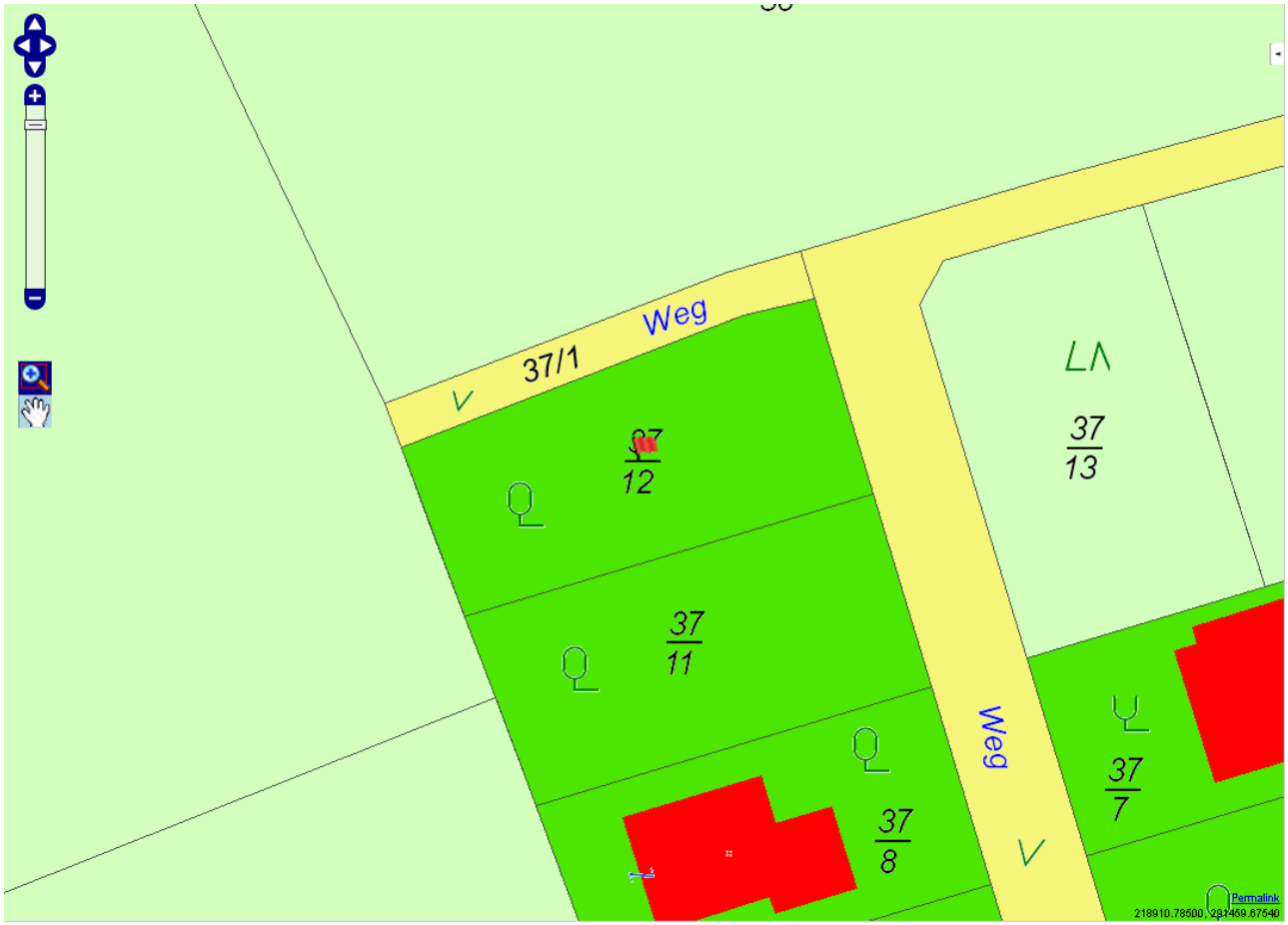
Marker als Kreis

- pointRadius: Radius des Markers
- fillOpacity: Transparenz des Markers
- fillColor: Farbe der Fläche des Markers
- strokeColor: Farbe der Umrandung des Markers
- strokeWidth: Breite der Umrandung des Markers in Pixel

4.1.6 Beispiel für die Anzeige von Markern

Folgende Abbildung zeigt ein Beispiel eines Markers in Form einer externen Grafik. Diese Grafik stellt ein rotes Fähnchen dar.

Marker als Grafik



Das nächste Beispiel zeigt Marker in Form von Kreisen.

Marker als Kreis

Natürlich ist der VectorLayer als ganzes deaktivierbar, wodurch die Marker nicht angezeigt werden. Dies erfolgt durch Abwahl des Suchergebnisses im LayerSwitcher. Folgende Abbildung zeigt das gleiche Suchergebnis, ohne Anzeige der Marker.

Marker nicht angezeigt

4.1.7 Zusammenfassung

Diese Lerneinheit befasst sich mit dem Thema Vectorlayer. OpenLayers kann beliebige GML-Objekte innerhalb der Karte darstellen. Hierfür werden Vectorlayer als Kartenebene verwendet. Nach einer kurzen allgemeinen Einführung in Vectorlayer wird die Befüllung dieser durch Erklärung und Beispiele gezeigt. Um Suchergebnisse in der Karte darzustellen, können Marker verwendet werden. Marker sind Grafiken oder Formen, die in einem Vectorlayer dargestellt werden. Dabei dienen GML-Objekte als geografische Positionierung der Marker. Eine Liste von GML-Punkten dient den Markern als Positionen, um ein Suchergebnis darstellen zu können. Abschließend zeigt ein Beispiel, wie dies in OpenLayers erfolgen kann.

4.2 Suche nach den Bounding Boxes

In dieser Lerneinheit wird erklärt, wie nach Bounding Boxes der Grundstücke gesucht werden kann. Die BoundingBoxes werden für die Ausrichtung des Kartenfensters benötigt. Nach dieser Ausrichtung zeigt das Kartenfenster alle Grundstücke des Ergebnisses in ihrer Gesamtheit dar.

Lernziele

- Definition einer Bounding Box
- Durchführung einer Suche nach Bounding Boxes
- Erstellen einer umschliessenden Bounding Box um eine Menge von GML-Objekten

4.2.1 Was ist eine Bounding Box?

Eine Bounding Box bildet das umschließende Rechteck um ein geografisches Feature. Die Ausdehnung der Bounding Box wird dabei durch die minimalen und maximalen Koordinaten in X- und Y-Richtung definiert. Die linke untere Ecke wird durch die minimalen X- und Y-Koordinaten festgelegt, und die rechte obere Ecke durch jeweils maximale Koordinaten. Mit diesen beiden Koordinaten wird die Bounding Box aufgespannt. Ein Zoom/Pan auf die Bounding Box garantiert, dass das der Bounding Box zugrundeliegende Feature in seiner Gesamtheit im Kartenfenster angezeigt wird. Andernfalls besteht die Gefahr, das Teile des Features nicht mehr dargestellt werden können, und der Benutzer die Karte manuell verkleinern oder verschieben muss. Die hier gewählte Variante mit Bounding Boxes dient also der höheren Benutzerfreundlichkeit.

4.2.2 Suche nach Bounding Boxes

Dieses Tutorial zeigt, wie Grundstücke mittels WFS gesucht werden können. Daher erfolgt auch die Suche nach den Bounding Boxes mittels WFS. Hierfür wird der FeatureType GrundstuecksRechtecke des WFS ASP_WFS verwendet, welcher bereits die Bounding Boxes der einzelnen Grundstücke enthält. Wenn mehr als ein Grundstück im Ergebnis erscheint, muss die Bounding Box sozusagen um alle Bounding Boxes berechnet werden, da im Ergebnisfenster alle Grundstücke angezeigt werden sollen. Die folgende Abbildung zeigt, wie diese globale Bounding Box ermittelt wird.

Bounding Box ermitteln

```
294             internalvectorLayer = new OpenLayers.Layer.Vector("internalSuchergebnis", {style: layer_style});
295
296             internalvectorLayer.addFeatures(features);
```

In OpenLayers gibt es die Möglichkeit, die Bounding Box um alle Features eines VectorLayers zu ermitteln. Aus diesem Grund wurde in der obigen Abbildung, ein interner VectorLayer erzeugt und alle Features des Suchergebnisses eingefügt. Folgende Abbildung zeigt, wie daraus die resultierende Bounding Box, in OpenLayers DataExtent genannt, ermittelt wird.

Data Extent ermitteln

```
310             map.zoomToExtent(internalvectorLayer.getDataExtent());
```

Jetzt fehlt noch die eigentliche Suche nach den Bounding Boxes der Grundstücke, also den Features, die in den internen VectorLayer hinzugefügt werden. Diese Suche wird mittels WFS durchgeführt. Die nachfolgende Abbildung zeigt, wie eine Suche gegen das WFS abgesetzt wird.

Suche durchführen

```
277                 OpenLayers.Request.POST({
278                     url: '<%= wfsSucheUrl %>',
279                     data: requestString,
280                     callback: handleResultRechtecke
281                 });
```

Die Suche erfolgt mittels POST-Request, der asynchron durchgeführt wird. Als Parameter werden die URL des WFS, der eigentliche Requeststring und eine Callback-Methode benötigt. Der Requeststring muss dem Filter Encoding entsprechen. Folgende Abbildung zeigt, wie dieser String in JavaScript zusammengesetzt wird.

Requeststring zusammensetzen

```
265                 requestString = requestBegin +
266                                 maxfeat +
267                                 requestAfterMaxFeatures +
268                                 requestGrundstuecksRechtecke +
269                                 searchFilterStart +
270                                 gstnr +
271                                 searchFilterInBetween +
272                                 kgnr +
273                                 searchFilterEnd +
274                                 requestEnd;
```

Die einzelnen Teile des Requeststring werden in den folgenden Abbildungen genauer dargestellt. Die Variablen maxfeat (Maximale Anzahl an Ergebnissen), gstnr (Grundstücksnummer) und kgnr (Katastralgemeindenummer) werden durch Eingaben des Benutzers festgelegt.

Inhalt Variable requestBegin

```
62              var requestBegin =
63                  '<?xml version="1.0" encoding="utf-8"?> ' +
64                  '<GetFeature xmlns="http://www.opengis.net/wfs" ' +
65                       'xmlns:ASP_WFS="<%= wfsNameSpace %>"  ' +
66                       'xmlns:ogc="http://www.opengis.net/ogc" ' +
67                       'xmlns:gml="http://www.opengis.net/gml" ' +
68                       'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ' +
69                       'xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd ' +
70                       'http://www.opengis.net/gml/3.1/geometry.xsd" ' +
71                       'service="WFS" ' +
72                       'version="1.1.0" ' +
73                       'outputFormat="text/xml; subtype=gml/3.1.1" ' +
74                       'maxFeatures="';
```

Die Variable requestBegin enthält den Aufruf der Methode GetFeatures inklusive der Angabe der maximalen Features im Ergebnis.

Inhalt Variable requestAfterMaxFeatures

```
75              var requestAfterMaxFeatures= '" ' +
76                       'handle="" > ';
```

Inhalt Variable requestGrundstuecksRechtecke

```
79              var requestGrundstuecksRechtecke =
80                  '<Query typeName="ASP_WFS:' + stringGrundstuecksRechtecke + '" srsName="<%= epsgCode %>" > ';
```

Der String stringGrundstuecksRechtecke enthält den Namen des WFS, in diesem Fall GrundstuecksRechtecke.

Inhalt Variable searchFilterStart

```
89              searchFilterStart =
90                  '<ogc:Filter>' +
91                    '<ogc:And>' +
92                      '<ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">' +
93                        '<ogc:PropertyName>' +
94                          'GNR' +
95                        '</ogc:PropertyName>' +
96                        '<ogc:Literal>';
```

Inhalt Variable searchFilterInBetween

```
97            searchFilterInBetween =
98                    '</ogc:Literal>' +
99                  '</ogc:PropertyIsLike>' +
100                 '<ogc:PropertyIsEqualTo>' +
101                   '<ogc:PropertyName>' +
102                     'KG' +
103                   '</ogc:PropertyName>' +
104                   '<ogc:Literal>';
```

Inhalt Variable searchFilterEnd

```
105           searchFilterEnd =
106                   '</ogc:Literal>' +
107                 '</ogc:PropertyIsEqualTo>' +
108               '</ogc:And>' +
109             '</ogc:Filter>';
```

Inhalt Variable requestEnd

```
85            var requestEnd =
86                  '</Query> ' +
87                '</GetFeature>';
```

Damit ist der Aufbau eines Requests im JavaScript definiert. Folgende Abbildung zeigt aufgrund der Eingabe eines Benutzer, wie der zugehörige Request aussieht.

Beispielhafter Request

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <GetFeature xmlns="http://www.opengis.net/wfs"
3              xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
4              xmlns:ogc="http://www.opengis.net/ogc"
5              xmlns:gml="http://www.opengis.net/gml"
6              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7              xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd
8                                  http://www.opengis.net/gml/3.1/geometry.xsd"
9              service="WFS"
10             version="1.1.0"
11             outputFormat="text/xml; subtype=gml/3.1.1"
12             maxFeatures="10"
13             handle="" >
14     <Query typeName="ASP_WFS:GrundstuecksRechtecke" srsName="EPSG:31255" >
15         <ogc:Filter>
16             <ogc:And>
17                 <ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">
18                     <ogc:PropertyName>GNR</ogc:PropertyName>
19                     <ogc:Literal>37/*</ogc:Literal>
20                 </ogc:PropertyIsLike>
21                 <ogc:PropertyIsEqualTo>
22                     <ogc:PropertyName>KG</ogc:PropertyName>
23                     <ogc:Literal>23431</ogc:Literal>
24                 </ogc:PropertyIsEqualTo>
25             </ogc:And>
26         </ogc:Filter>
27     </Query>
28 </GetFeature>
29
```

4.2.3 Pan und Zoom auf ergebende Bounding Box

Die Antwort des WFS muss als erstes geparst werden. Hierfür bietet OpenLayers für alle bisherigen Versionen der GML eigene Parser an. Folgende Abbildung zeigt, wie innerhalb der Callback-Methode die Antwort der Version GMLv3 geparst wird.

Parsen der Antwort

```
288                 var g =  new OpenLayers.Format.GML.v3({'featureType': stringGrundstuecksRechtecke,
289                    'featureNS' : '<%= wfsNameSpace %>',
290                    'geometryName' : 'ASP_WFS:SHAPE', 'xy' : false, 'srsName': '<%= epsgCode %>'});
291                 html = ""
292                 features = g.read(responsedText);
```

Aus den geparsten Features muss nun die Bounding Box berechnet werden. Dies erfolgt mittels internem VectorLayer, der das Ergebnis beinhaltet.

Bounding Box ermitteln

```
294                 internalvectorLayer = new OpenLayers.Layer.Vector("internalSuchergebnis", {style: layer_style});
295
296                 internalvectorLayer.addFeatures(features);
```

Nachdem die Bounding Box ermittelt wurde, muss das Kartenfenster auf den Ausschnitt verschoben und vergrößert werden. Dies erfolgt mittels eines einzelnen Befehls, der die Bounding Box als Parameter nimmt. Folgende Abbildung zeigt den Befehl zum Pan und Zoom auf die Bounding Box.

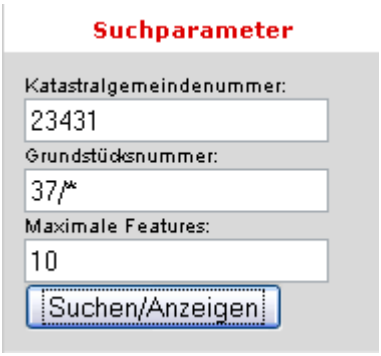
Pan und Zoom auf Bounding Box

```
310                 map.zoomToExtent(internalvectorLayer.getDataExtent());
```

4.2.4 Beispiel einer Suche

Für das hier gezeigte Beispiel wird von folgender Benutzereingabe ausgegangen:

Suchparameter



Aufgrund dieser Suchparameter wird der Request an das WFS zusammengestellt. Folgende Abbildung zeigt den daraus resultierenden Request.

Beispielhafter Request

```
<?xml version="1.0" encoding="utf-8"?>
<GetFeature xmlns="http://www.opengis.net/wfs"
            xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:gml="http://www.opengis.net/gml"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd
                                http://www.opengis.net/gml/3.1/geometry.xsd"
            service="WFS"
            version="1.1.0"
            outputFormat="text/xml; subtype=gml/3.1.1"
            maxFeatures="10"
            handle="" >
    <Query typeName="ASP_WFS:GrundstuecksRechtecke" srsName="EPSG:31255" >
        <ogc:Filter>
            <ogc:And>
                <ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">
                    <ogc:PropertyName>GNR</ogc:PropertyName>
                    <ogc:Literal>37/*</ogc:Literal>
                </ogc:PropertyIsLike>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>KG</ogc:PropertyName>
                    <ogc:Literal>23431</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:And>
        </ogc:Filter>
    </Query>
</GetFeature>
```

Das WFS verarbeitet diese Anfrage und liefert die Liste der Bounding Boxes der Grundstücke zurück. Zur Verdeutlichung des Ergebnisses zeigt folgende Grafik die Grundstücke mittels Fähnchen markiert. Ergebnis dieses Aufrufes ist trotzdem eine Liste der Bounding Boxes, auch wenn hier Fähnchen verwendet werden.

Ergebnis der Anfrage

Das XML-Dokument des Suchergebnisses beinhaltet die Bounding Boxes. Folgende Abbildung zeigt einen Ausschnitt aus dem Ergebnis.

Ergebnis der Bounding Box Suche als XML (nur Teil)

```
<wfs:FeatureCollection xsi:schemaLocation='http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer http://21
    <gml:boundedBy>
        <gml:Envelope srsName='EPSG:31255'>
            <gml:lowerCorner>138770.24100000001 -287560.1409</gml:lowerCorner>
            <gml:upperCorner>432642.6263 285528.98479999998</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
        <ASP_WFS:GrundstuecksRechtecke gml:id='F23__2322781'>
            <ASP_WFS:OBJECTID>2322781</ASP_WFS:OBJECTID>
            <ASP_WFS:GNR>37/13</ASP_WFS:GNR>
            <ASP_WFS:RSTATUS>E</ASP_WFS:RSTATUS>
            <ASP_WFS:TYP>2</ASP_WFS:TYP>
            <ASP_WFS:KG>23431
            </ASP_WFS:KG>
            <ASP_WFS:KGGNR>2343137/13</ASP_WFS:KGGNR>
            <ASP_WFS:GEMNR>32333</ASP_WFS:GEMNR>
            <ASP_WFS:BEZNR>323</ASP_WFS:BEZNR>
            <ASP_WFS:LANDNR>3</ASP_WFS:LANDNR>
            <ASP_WFS:VANR>23</ASP_WFS:VANR>
            <ASP_WFS:XMIN>218959.99609999999</ASP_WFS:XMIN>
            <ASP_WFS:YMIN>291436.163</ASP_WFS:YMIN>
            <ASP_WFS:XMAX>218989.65700000001</ASP_WFS:XMAX>
            <ASP_WFS:YMAX>291475.19949999999</ASP_WFS:YMAX>
            <ASP_WFS:KGNAME>Schleinz</ASP_WFS:KGNAME>
            <ASP_WFS:GEMNAME>Walpersbach</ASP_WFS:GEMNAME>
            <ASP_WFS:BEZNAME>Wiener Neustadt</ASP_WFS:BEZNAME>
            <ASP_WFS:VANAME>Wr. Neustadt</ASP_WFS:VANAME>
            <ASP_WFS:LANDNAME>Niederösterreich</ASP_WFS:LANDNAME>
            <ASP_WFS:SHAPE>
                <gml:Point>
                    <gml:pos>291454.07840000093 218974.38129999954</gml:pos>
                </gml:Point>
            </ASP_WFS:SHAPE>
        </ASP_WFS:GrundstuecksRechtecke>
    </gml:featureMember>
    <gml:featureMember>
        <ASP_WFS:GrundstuecksRechtecke gml:id='F23__2322783'>
            <ASP_WFS:OBJECTID>2322783</ASP_WFS:OBJECTID>
            <ASP_WFS:GNR>37/1</ASP_WFS:GNR>
            <ASP_WFS:RSTATUS>E</ASP_WFS:RSTATUS>
            <ASP_WFS:TYP>1</ASP_WFS:TYP>
            <ASP_WFS:KG>23431
            </ASP_WFS:KG>
            <ASP_WFS:KGGNR>2343137/1</ASP_WFS:KGGNR>
            <ASP_WFS:GEMNR>32333</ASP_WFS:GEMNR>
            <ASP_WFS:BEZNR>323</ASP_WFS:BEZNR>
            <ASP_WFS:LANDNR>3</ASP_WFS:LANDNR>
            <ASP_WFS:VANR>23</ASP_WFS:VANR>
            <ASP_WFS:XMIN>218914.08530000001</ASP_WFS:XMIN>
            <ASP_WFS:YMIN>291454.34830000001</ASP_WFS:YMIN>
            <ASP_WFS:XMAX>218950.96460000001</ASP_WFS:XMAX>
            <ASP_WFS:YMAX>291471.1974</ASP_WFS:YMAX>
            <ASP_WFS:KGNAME>Schleinz</ASP_WFS:KGNAME>
            <ASP_WFS:GEMNAME>Walpersbach</ASP_WFS:GEMNAME>
            <ASP_WFS:BEZNAME>Wiener Neustadt</ASP_WFS:BEZNAME>
```

4.2.5 Zusammenfassung

Diese Lerneinheit befasst sich mit der Problematik der Anzeige des Suchergebnisses. Wenn ein Grundstück gefunden wird, sollte es als Ganzes im Kartenfenster angezeigt werden. Hierfür kommt das Konzept der Bounding Boxes, als umschließende Rechtecke, zum Einsatz. Wenn die Suche als Ergebnis diese Bounding Boxes liefert, muss das Kartenfenster so ausgerichtet werden, dass alle Bounding Boxes dargestellt werden können. Es wird das umschließende Rechteck um die Bounding Boxes berechnet, und die Karte von OpenLayers auf diese Position verschoben und vergrößert. Sowohl für die Suche als auch für die korrekte Anzeige des Kartenfensters werden Beispiele gezeigt.

4.3 Suche nach Einsetzpunkten der Grundstücksnummern

In dieser Lerneinheit wird erklärt, wie nach den Einsetzpunkten der Grundstücksnummern gesucht werden kann. In der vorherigen Lerneinheit wird nach den Bounding Boxes der Grundstücke gesucht. Nachdem nun das geografische Ausmaß des Ergebnisses bekannt ist, müssen die tatsächlichen Einsetzpunkte für die Marker ermittelt werden. Dies erfolgt, indem die Marker auf die Einsetzpunkte der Grundstücksnummern gesetzt werden.

Lernziele

- Durchführen einer Suche nach den Einsetzpunkte der Grundstücksnummern
- Parsen der GML-Antwort des WFS
- Einfügen der Marker an den Einsetzpunkten der Grundstücksnummern
- Verstehen der Probleme bei der Verwendung von CRS84 und EPSG4326

4.3.1 Suche nach den Einsetzpunkten

Die ermittelten Bounding Boxes dienen dazu, die Karte in den richtigen Ausschnitt zu bewegen, nützen jedoch nicht der Darstellung des Ergebnisses. Ziel der Suche nach Bounding Boxes war die Darstellung der gesamten Grundstückspolygone im Kartenfenster. Für die Markierung des Ergebnisses werden nun Marker auf die Grundstücke des Ergebnisses gelegt. Die einfachste Variante wäre, den Marker in den Schwerpunkt, also die Mitte, der Bounding Box zu setzen. Dies wird meist zielführend sein, doch wirft es Probleme auf, wenn ein Grundstück nicht die übliche Form hat. Ein extremes Beispiel wäre ein Grundstück in Form eines Ringes oder eines Hufeisens. Bei beiden ist der Schwerpunkt der Bounding Box außerhalb des Grundstückes und die Markierung würde ein falsches Ergebnis suggerieren. Das gleiche Problem hat das BEV beim Beschriften der Grundstücke mit den Grundstücksnummern. Diese sollten womöglich in der Mitte des Grundstückes liegen, auf jeden Fall im Grundstück. Der FeatureType GrundstuecksEinsetzpunkte bietet genau diese Punkte als Features an. Somit muss nur noch für das Ergebnis der Suche nach Bounding Boxes die jeweiligen Einsetzpunkte der Grundstücksnummern ermittelt werden. Folgende Grafik zeigt, wie der Request für die Suche der Einsetzpunkte zusammengestellt wird.

Aufbau des Requests

```
316                 requestString = requestBegin +
317                                 maxfeat +
318                                 requestAfterMaxFeatures +
319                                 requestGNREinsetzpunkte +
320                                 objectidFilterStart;
321                 if (objectIds.length > 1) {
322                     requestString += objectidOrStart;
323                 }
324
325                 for(var objectId in objectIds) {
326                     requestString += objectidPropertyStart +
327                                      objectIds[objectId] +
328                                      objectidPropertyEnd ;
329                 }
330
331
332                 if (objectIds.length > 1) {
333                     requestString += objectidOrEnd;
334                 }
335                 requestString += objectidFilterEnd +
336                                  requestEnd;
```

Die einzelnen Teile der Suchanfrage werde im Anschluß genauer gezeigt.

Inhalt Variable requestBegin

```
62          var requestBegin =
63              '<?xml version="1.0" encoding="utf-8"?> ' +
64              '<GetFeature xmlns="http://www.opengis.net/wfs" ' +
65                  'xmlns:ASP_WFS="<%= wfsNameSpace %>"  ' +
66                  'xmlns:ogc="http://www.opengis.net/ogc" ' +
67                  'xmlns:gml="http://www.opengis.net/gml" ' +
68                  'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ' +
69                  'xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd ' +
70                  'http://www.opengis.net/gml/3.1/geometry.xsd" ' +
71                  'service="WFS" ' +
72                  'version="1.1.0" ' +
73                  'outputFormat="text/xml; subtype=gml/3.1.1" ' +
74                  'maxFeatures="';
```

Inhalt Variable requestAfterMaxFeatures

```
75          var requestAfterMaxFeatures= '" ' +
76                  'handle="" > ';
```

Inhalt Variable requestGNREinsetzpunkte

```
83          var requestGNREinsetzpunkte =
84              '<Query typeName="ASP_WFS:' + stringGNREinsetzpunkte + '" srsName="<%= epsgCode %>" > ';
```

Inhalt der Object-ID Variablen

```
111         objectidFilterStart =
112             '<ogc:Filter>';
113         objectidOrStart =
114               '<ogc:Or>';
115         objectidPropertyStart =
116               '<ogc:PropertyIsEqualTo>' +
117                 '<ogc:PropertyName>' +
118                   'OBJECTID' +
119                 '</ogc:PropertyName>' +
120                 '<ogc:Literal>';
121         objectidPropertyEnd =
122                 '</ogc:Literal>' +
123               '</ogc:PropertyIsEqualTo>';
124         objectidOrEnd =
125               '</ogc:Or>';
126         objectidFilterEnd =
127             '</ogc:Filter>';
```

Inhalt Variable requestEnd

```
85          var requestEnd =
86              '</Query> ' +
87            '</GetFeature>';
```

Um den Request besser zu verdeutlichen, soll nun ein zusammengesetzer Request gezeigt werden.

Fertiger Request nach Einsetzpunkten

```
<?xml version="1.0" encoding="utf-8"?>
<GetFeature xmlns="http://www.opengis.net/wfs"
            xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:gml="http://www.opengis.net/gml"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd
                                http://www.opengis.net/gml/3.1/geometry.xsd"
            service="WFS"
            version="1.1.0"
            outputFormat="text/xml; subtype=gml/3.1.1"
            maxFeatures="10"
            handle="" >
    <Query typeName="ASP_WFS:GrundstuecksEinsetzpunkte" srsName="EPSG:31255" >
        <ogc:Filter>
            <ogc:Or>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322781</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322783</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322784</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322785</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322789</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322790</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322792</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322793</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322794</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322797</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Or>
        </ogc:Filter>
    </Query>
</GetFeature>
```

Nach Betrachtung des Requests stellt sich die Frage, warum nicht der gleiche Request nochmals für die Einsetzpunkte abgesetzt wird, der schon bei den Bounding Boxes verwendet wurde. Die Antwort ist ziemlich einfach, denn es ist nicht garantiert, dass das Ergebnis das gleiche ist. Wenn eine Einschränkung des Ergebnisses eingegeben wurde (ist sehr empfehlenswert, da dies die Performance positiv beeinflusst), werden zum Beispiel maximal 10 Grundstücke geliefert. Interessant ist, dass die Reihenfolge der Grundstücke nicht garantiert werden kann, wodurch die ersten 10 Grundstücke durchaus andere sein können. Somit würde die Anfrage 10 Bounding Boxes liefern, das Kartenfenster würde auf diese 10 Grundstücke ausgerichtet werden, um dann 10 Grundstücke zu markieren, die vielleicht nicht einmal im Kartenfenster dargestellt werden! Die

Lösung dieses Problems stellt die oben gezeigte Suche dar, wobei hier nicht nach der Benutzereingabe gesucht wird, sondern nach den Primärschlüsseln des Ergebnisses der Suche nach Bounding Boxes. Dies führt garantiert zum richtigen Ergebnis.
Die Suche nach den Einsetzpunkten wird mittels folgendem Befehl an das WFS geschickt. Diese Suche erfolgt wiederum asynchron mittels POST-Request. Parameter sind die URL des WFS, der oben dargestellte Requeststring und eine Callback Methode, die das Parsing der Antwort übernimmt.

Abschicken des Requests

```
340                 OpenLayers.Request.POST({
341                     url: '<%= wfsEinsetzpunkteUrl %>',
342                     data: requestString,
343                     callback: handleResultEinsetzpunkte
344                 });
```

4.3.2 Parsing der GML-Antwort

Die Antwort der Suche nach den Einsetzpunkten muss noch geparst werden. Dies erfolgt wiederum mittels GML Parser von OpenLayers. Folgende Grafik zeigt, wie der Parser die Antwort verarbeitet.

Parsing der Antwort

```
351                 var g =  new OpenLayers.Format.GML.v3({'featureType': stringGNREinsetzpunkte,
352                    'featureNS' : '<%= wfsNameSpace %>',
353                    'geometryName' : 'ASP_WFS:SHAPE', 'xy' : false, 'srsName': '<%= epsgCode %>'});
354                 html = ""
355                 features = g.read(responsedText);
```

4.3.3 Einfügen der Marker

Nachdem die Antwort nun in Form von GML-Objekten vorliegt, müssen die zugehörigen Marker in VectorLayer eingefügt werden. Dies erfolgt mittels folgenden Codezeilen.

Hinzufügen der Marker zum VectorLayer

```
356                 vectorLayer.destroyFeatures();
357                 vectorLayer.addFeatures(features);
```

Da der VectorLayer verändert wurde, müssen die neuen Daten im Kartenfenster angezeigt werden. Folgende Abbildung zeigt, wie der VectorLayer im Kartenfenster aktualisiert wird.

Aktualisieren des VectorLayers

```
367                 vectorLayer.redraw();
```

4.3.4 Probleme zwischen CRS84 und EPSG4326

Im Zuge der Standardisierung von GML wurde ein Problem standardisiert, nämlich in welcher Reihenfolge die Koordinaten der Längen- und Breitengrade angegeben werden. Die Koordinatensysteme CRS84 und EPSG4326 definieren beide WGS84, nur in unterschiedlicher Reihenfolge der Koordinaten. In CRS84 kommen zuerst die Längen-, in EPSG4326 zuerst die Breitengrade. Um solch ein Problem umgehen zu können, gibt es in OpenLayers ein nicht dokumentiertes Attribut. Das Attribut 'xy' steuert die Reihenfolge der Koordinaten und wird beim Parser angegeben. Folgende Abbildung zeigt ein Beispiel der Angabe des Attributes.

Parsing der Antwort

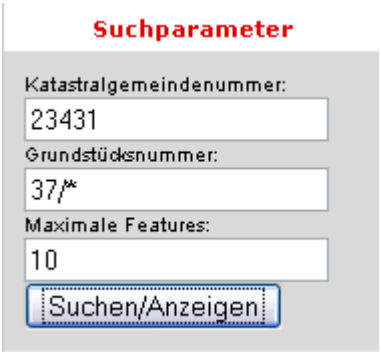
```
351                 var g =  new OpenLayers.Format.GML.v3({'featureType': stringGNREinsetzpunkte,
352                    'featureNS' : '<%= wfsNameSpace %>',
353                    'geometryName' : 'ASP_WFS:SHAPE', 'xy' : false, 'srsName': '<%= epsgCode %>'});
354                 html = ""
355                 features = g.read(responsedText);
```

Das Attribut 'xy' erwartet einen Booleanausdruck, wobei false die Standardreihenfolge Breiten- vor Längengrad umkehrt.

4.3.5 Beispiel einer Suche

Zum Abschluß soll ein zusammenfassendes Beispiel den Request und das Ergebnis zu einer Eingabe zeigen. Für dieses Beispiel dient folgende Eingabe als Suchkriterium:

Suchparameter



Dies führt zu folgender Suchanfrage nach den Bounding Boxes.

Beispielhafter Request für Bounding Boxes

```
<?xml version="1.0" encoding="utf-8"?>
<GetFeature xmlns="http://www.opengis.net/wfs"
            xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:gml="http://www.opengis.net/gml"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd
                                http://www.opengis.net/gml/3.1/geometry.xsd"
            service="WFS"
            version="1.1.0"
            outputFormat="text/xml; subtype=gml/3.1.1"
            maxFeatures="10"
            handle="" >
    <Query typeName="ASP_WFS:GrundstuecksRechtecke" srsName="EPSG:31255" >
        <ogc:Filter>
            <ogc:And>
                <ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">
                    <ogc:PropertyName>GNR</ogc:PropertyName>
                    <ogc:Literal>37/*</ogc:Literal>
                </ogc:PropertyIsLike>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>KG</ogc:PropertyName>
                    <ogc:Literal>23431</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:And>
        </ogc:Filter>
    </Query>
</GetFeature>
```

Das Ergebnis ist ein XML-Dokument mit allen Bounding Boxes. Folgende Abbildung zeigt den ersten Teil des Ergebnisses.

Ergebnis der Bounding Box Suche als XML (nur Teil)

```
<wfs:FeatureCollection xsi:schemaLocation='http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer http://21
    <gml:boundedBy>
        <gml:Envelope srsName='EPSG:31255'>
            <gml:lowerCorner>138770.24100000001 -287560.1409</gml:lowerCorner>
            <gml:upperCorner>432642.6263 285528.98479999998</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
        <ASP_WFS:GrundstuecksRechtecke gml:id='F23__2322781'>
            <ASP_WFS:OBJECTID>2322781</ASP_WFS:OBJECTID>
            <ASP_WFS:GNR>37/13</ASP_WFS:GNR>
            <ASP_WFS:RSTATUS>E</ASP_WFS:RSTATUS>
            <ASP_WFS:TYP>2</ASP_WFS:TYP>
            <ASP_WFS:KG>23431
            </ASP_WFS:KG>
            <ASP_WFS:KGGNR>2343137/13</ASP_WFS:KGGNR>
            <ASP_WFS:GEMNR>32333</ASP_WFS:GEMNR>
            <ASP_WFS:BEZNR>323</ASP_WFS:BEZNR>
            <ASP_WFS:LANDNR>3</ASP_WFS:LANDNR>
            <ASP_WFS:VANR>23</ASP_WFS:VANR>
            <ASP_WFS:XMIN>218959.99609999999</ASP_WFS:XMIN>
            <ASP_WFS:YMIN>291436.163</ASP_WFS:YMIN>
            <ASP_WFS:XMAX>218989.65700000001</ASP_WFS:XMAX>
            <ASP_WFS:YMAX>291475.19949999999</ASP_WFS:YMAX>
            <ASP_WFS:KGNAME>Schleinz</ASP_WFS:KGNAME>
            <ASP_WFS:GEMNAME>Walpersbach</ASP_WFS:GEMNAME>
            <ASP_WFS:BEZNAME>Wiener Neustadt</ASP_WFS:BEZNAME>
            <ASP_WFS:VANAME>Wr. Neustadt</ASP_WFS:VANAME>
            <ASP_WFS:LANDNAME>Niederösterreich</ASP_WFS:LANDNAME>
            <ASP_WFS:SHAPE>
                <gml:Point>
                    <gml:pos>291454.07840000093 218974.38129999954</gml:pos>
                </gml:Point>
            </ASP_WFS:SHAPE>
        </ASP_WFS:GrundstuecksRechtecke>
    </gml:featureMember>
    <gml:featureMember>
        <ASP_WFS:GrundstuecksRechtecke gml:id='F23__2322783'>
            <ASP_WFS:OBJECTID>2322783</ASP_WFS:OBJECTID>
            <ASP_WFS:GNR>37/1</ASP_WFS:GNR>
            <ASP_WFS:RSTATUS>E</ASP_WFS:RSTATUS>
            <ASP_WFS:TYP>1</ASP_WFS:TYP>
            <ASP_WFS:KG>23431
            </ASP_WFS:KG>
            <ASP_WFS:KGGNR>2343137/1</ASP_WFS:KGGNR>
            <ASP_WFS:GEMNR>32333</ASP_WFS:GEMNR>
            <ASP_WFS:BEZNR>323</ASP_WFS:BEZNR>
            <ASP_WFS:LANDNR>3</ASP_WFS:LANDNR>
            <ASP_WFS:VANR>23</ASP_WFS:VANR>
            <ASP_WFS:XMIN>218914.08530000001</ASP_WFS:XMIN>
            <ASP_WFS:YMIN>291454.34830000001</ASP_WFS:YMIN>
            <ASP_WFS:XMAX>218950.96460000001</ASP_WFS:XMAX>
            <ASP_WFS:YMAX>291471.1974</ASP_WFS:YMAX>
            <ASP_WFS:KGNAME>Schleinz</ASP_WFS:KGNAME>
            <ASP_WFS:GEMNAME>Walpersbach</ASP_WFS:GEMNAME>
            <ASP_WFS:BEZNAME>Wiener Neustadt</ASP_WFS:BEZNAME>
```

Aus dem Ergebnis dieser Anfrage wird eine erneute Suche zusammengestellt, welche die Einsetzpunkte der Grundstücke ermittelt. Als Suchkriterium dient hierbei die ObjectID aus den einzelnen Features der Bounding Box Suche. Folgende Abbildung zeigt den daraus resultierenden Request an das WFS.

Request nach Einsetzpunkten

```
<?xml version="1.0" encoding="utf-8"?>
<GetFeature xmlns="http://www.opengis.net/wfs"
            xmlns:ASP_WFS="http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:gml="http://www.opengis.net/gml"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd
                                http://www.opengis.net/gml/3.1/geometry.xsd"
            service="WFS"
            version="1.1.0"
            outputFormat="text/xml; subtype=gml/3.1.1"
            maxFeatures="10"
            handle="" >
    <Query typeName="ASP_WFS:GrundstuecksEinsetzpunkte" srsName="EPSG:31255" >
        <ogc:Filter>
            <ogc:Or>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322781</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322783</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322784</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322785</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322789</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322790</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322792</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322793</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322794</ogc:Literal>
                </ogc:PropertyIsEqualTo>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>OBJECTID</ogc:PropertyName>
                    <ogc:Literal>2322797</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Or>
        </ogc:Filter>
    </Query>
</GetFeature>
```

Das Ergebnis dieses Requests wird dann in Form von Markern im Kartenfenster dargestellt. Folgende Abbildung zeigt das Ergebnis im Kartenfenster.

Ergebnis der Anfrage

4.3.6 Beispiel eines vollständigen Web-Clients

Zusammenfassend soll nun ein vollständiger Web-Client gezeigt werden, der die in allen Lerneinheiten erarbeiteten Inhalte umfasst. Dieser Web-Client ist noch weit davon entfernt, produktiv zum Einsatz zu kommen, liefert jedoch einen sehr guten Anhaltspunkt für eigene Implentierungen.

Vollständiger Web-Client (Teil 1)

```
<%@ page import="com.pe.*" %>
<%@ page import="com.pe.gstsuche.*" %>

<!--      OpenLayers.loadURL( 'http://217.13.180.218/GeoServer2/Interceptor/Wfs/ASP_WFS/BEVTest-d1118bef-dc85-4f68-af07-da808c3417eb',
      OpenLayers.loadURL( 'http://217.13.180.218/BEVIntern/services/ASP_WFS/MapServer/WFSServer',
-->

<%
  TOKEN token = (TOKEN)session.getAttribute(TokenServlet.TOKEN_KEY);
  System.out.println("hier schon da" + token.getSERVICES());
  String wmsUrl = "http://service-not-found";
  String km50Url = "http://service-not-found";
  String wfsSucheUrl = "http://service-not-found";
  for (SERVICE service : token.getSERVICES().getSERVICE()) {
      if (service.getNAME().equals("ASP_DKM"))
          wmsUrl = service.getURL();
      if (service.getNAME().equals("ASP_KM50"))
          km50Url = service.getURL();
      if (service.getNAME().equals("ASP_WFS"))
          wfsSucheUrl = service.getURL();
  }
  wfsSucheUrl = "http://217.13.180.218/BEVIntern/services/ASP_WFS/MapServer/WFSServer";
  String wfsEinsetzpunkteUrl = "http://217.13.180.218/BEVIntern/services/ASP_WFS/MapServer/WFSServer";
  String wfsNameSpace = "http://rz1-asp-01/BEVIntern/services/ASP_WFS/MapServer/WFSServer";
  String epsgCode = "EPSG:31255";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML xmlns="http://www.w3.org/1999/xhtml">
    <HEAD>
        <TITLE>WFS/WMS Web Client - Pezerovic</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <LINK href="openlayers/style.css" type="text/css" rel="stylesheet" />
        <LINK href="openlayers/style2.css" type="text/css" rel="stylesheet" />
        <LINK href="bevci/BEV_STYLES_GENERAL.css" type="text/css" rel="stylesheet" />
        <script src="openlayers/OpenLayers.js"></script>
        <style type="text/css">
            #controls {
                float: left;
                text-align: right;
            }
            .smallmap {
                border: #ccc 1px solid;
                width: 1074px;
                height: 770px;
            }

        </style>

        <SCRIPT type="text/javascript">
            var xKoordinate = 150000;
            var yKoordinate = 300000;
            var zoom = 15;
            var map, layer;
            var vectorLayer;
            var layer_style;
            var requestString, searchFilterStart, searchFilterInBetween, searchFilterEnd;
            var objectidFilterStart, objectidFilterInBetween, objectidFilterEnd;
            var feattypeString;
            var maxfeat;

            var requestBegin =
                '<?xml version="1.0" encoding="utf-8"?> ' +
                '<GetFeature xmlns="http://www.opengis.net/wfs" ' +
                    'xmlns:ASP_WFS="<%= wfsNameSpace %>"  ' +
                    'xmlns:ogc="http://www.opengis.net/ogc" ' +
                    'xmlns:gml="http://www.opengis.net/gml" ' +
                    'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ' +
                    'xsi:schemaLocation="http://www.opengis.net/ogc/filter/1.0.0/filter.xsd ' +
                    'http://www.opengis.net/gml/3.1/geometry.xsd" ' +
                    'service="WFS" ' +
                    'version="1.1.0" ' +
                    'outputFormat="text/xml; subtype=gml/3.1.1" ' +
                    'maxFeatures="';
            var requestAfterMaxFeatures= '" ' +
                    'handle="" > ';
            var stringGrundstuecksRechtecke =
                    'GrundstuecksRechtecke';
            var requestGrundstuecksRechtecke =
                    '<Query typeName="ASP_WFS:' + stringGrundstuecksRechtecke + '" srsName="<%= epsgCode %>" > ';
```

Vollständiger Web-Client (Teil 2)

```
        var stringGNREinsetzpunkte =
            'GrundstuecksRechtecke';
        var requestGNREinsetzpunkte =
            '<Query typeName="ASP_WFS:' + stringGNREinsetzpunkte + '" srsName="<%= epsgCode %>" > ';
        var requestEnd =
            '</Query> ' +
          '</GetFeature>';

        searchFilterStart =
          '<ogc:Filter>' +
            '<ogc:And>' +
              '<ogc:PropertyIsLike wildCard="*" singleChar="_" escapeChar="#">' +
                '<ogc:PropertyName>' +
                  'GNR' +
                '</ogc:PropertyName>' +
                '<ogc:Literal>';
        searchFilterInBetween =
                '</ogc:Literal>' +
              '</ogc:PropertyIsLike>' +
              '<ogc:PropertyIsEqualTo>' +
                '<ogc:PropertyName>' +
                  'KG' +
                '</ogc:PropertyName>' +
                '<ogc:Literal>';
        searchFilterEnd =
                '</ogc:Literal>' +
              '</ogc:PropertyIsEqualTo>' +
            '</ogc:And>' +
          '</ogc:Filter>';

        objectidFilterStart =
          '<ogc:Filter>';
        objectidOrStart =
            '<ogc:Or>';
        objectidPropertyStart =
            '<ogc:PropertyIsEqualTo>' +
              '<ogc:PropertyName>' +
                'OBJECTID' +
              '</ogc:PropertyName>' +
              '<ogc:Literal>';
        objectidPropertyEnd =
              '</ogc:Literal>' +
            '</ogc:PropertyIsEqualTo>';
        objectidOrEnd =
            '</ogc:Or>';
        objectidFilterEnd =
          '</ogc:Filter>';

        function init(){
          OpenLayers.ProxyHost="ProxyHost?";

          map = new OpenLayers.Map( 'map', {
            controls: [
              new OpenLayers.Control.PanZoomBar(),
              new OpenLayers.Control.MouseToolbar(),
              new OpenLayers.Control.LayerSwitcher({'ascending':false}),
              new OpenLayers.Control.Permalink(),
            //  Bug: zeigt immer NaN an
            //  new OpenLayers.Control.Scale(),
            //  new OpenLayers.Control.ScaleLine(),
              new OpenLayers.Control.Permalink('permalink'),
              new OpenLayers.Control.MousePosition()
            ]
            , zoom : 15
            , maxExtent: new OpenLayers.Bounds(-287638, 139391, 284887, 432527)
            , maxResolution: 550
            //, resolutions: [550, 450, 390, 310, 230, 150, 70, 60, 50, 40, 30, 20, 10, 2, 0.2, 0.02]
            , minResolution: 0.05
            , units: 'meters'
            , projection: "<%= epsgCode %>"
          });

          layer= new OpenLayers.Layer.WMS(
            "KM50",
            "<%= km50Url %>",
            {layers: '0', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png'},
            {buffer:2}
          );
          map.addLayer(layer);
```

Vollständiger Web-Client (Teil 3)

```
            layer= new OpenLayers.Layer.WMS(
              "Nutzungsflächen",
              "<%= wmsUrl %>",
              {layers: '1', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Verwaltungsgrenzen",
              "<%= wmsUrl %>",
              {layers: '5', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Katastralgemeinden",
              "<%= wmsUrl %>",
              {layers: '0', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {ratio:1.2, singleTile: true, isBaseLayer: false}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Grundstücksnummern mit Randbeschriftung",
              "<%= wmsUrl %>",
              {layers: '2,3', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:0, singleTile: true, isBaseLayer: false}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Grundstücke",
              "<%= wmsUrl %>",
              {layers: '4', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Nutzungs- und Sonstige Grenzen",
              "<%= wmsUrl %>",
              {layers: '6', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false, transitionEffect: 'resize'}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Sonstige Symbole und Beschriftung",
              "<%= wmsUrl %>",
              {layers: '7', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Nutzungssymbole",
              "<%= wmsUrl %>",
              {layers: '8', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Grenzpunkte",
              "<%= wmsUrl %>",
              {layers: '9', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false, visibility: false}
            );
            map.addLayer(layer);

            layer= new OpenLayers.Layer.WMS(
              "Festpunkte",
              "<%= wmsUrl %>",
              {layers: '10', VERSION: "1.1.1", crs: '<%= epsgCode %>', format:'image/png', transparent: 'true'},
              {buffer:2, isBaseLayer: false, visibility: false}
            );
            map.addLayer(layer);
```

Vollständiger Web-Client (Teil 4)

```
                layer_style = OpenLayers.Util.extend({}, OpenLayers.Feature.Vector.style['default']);
                layer_style.graphicOpacity = 1;
                layer_style.pointRadius = 12;
                layer_style.graphicYOffset = -23;
                layer_style.graphicXOffset = -11;
                layer_style.fillOpacity = 0.0;
                layer_style.externalGraphic = "openlayers/img/marker.png";
                layer_style.fillColor = "#FFFFFF";
                layer_style.strokeColor = "#FF0000";
                layer_style.strokeWidth = 3;

                vectorLayer = new OpenLayers.Layer.Vector("Suchergebnis", {style: layer_style});
                map.addLayer(vectorLayer);

                map.setCenter(new OpenLayers.LonLat(xKoordinate, yKoordinate), zoom);
            }

            function gstSuchen() {
                var kgnr = parseInt(document.getElementById("kgnr").value, 10);
                var gstnr = document.getElementById("gstnr").value;

                maxfeat = document.getElementById("maxfeat").value;


                requestString = requestBegin +
                                maxfeat +
                                requestAfterMaxFeatures +
                                requestGrundstuecksRechtecke +
                                searchFilterStart +
                                gstnr +
                                searchFilterInBetween +
                                kgnr +
                                searchFilterEnd +
                                requestEnd;
                document.getElementById("requestID").value = requestString;

                OpenLayers.Request.POST({
                     url: '<%= wfsSucheUrl %>',
                     data: requestString,
                     callback: handleResultRechtecke
                 });

            }
            function handleResultRechtecke(req) {
                var responsedText = req.responseText;
                document.getElementById("responseID").value = responsedText;

                var g =  new OpenLayers.Format.GML.v3({'featureType': stringGrundstuecksRechtecke,
                   'featureNS' : '<%= wfsNameSpace %>',
                   'geometryName' : 'ASP_WFS:SHAPE', 'xy' : false, 'srsName': '<%= epsgCode %>'});
                html = ""
                features = g.read(responsedText);
                vectorLayer.destroyFeatures();
                internalvectorLayer = new OpenLayers.Layer.Vector("internalSuchergebnis", {style: layer_style});

                internalvectorLayer.addFeatures(features);

                var objectIds = new Array(features.length);

                var pos = 0;
                for(var feat in features) {
                    html += "Feature: Geometry: "+ features[feat].geometry+",";
                    html += "<ul>";
                    for (var j in features[feat].attributes) {
                        html += "<li>"+j+":"+features[feat].attributes[j]+"</li>";
                    }
                    html += "</ul>"
                    objectIds[pos++] = features[feat].attributes['OBJECTID'];
                }
                map.zoomToExtent(internalvectorLayer.getDataExtent());
                document.getElementById("responseFeatID").value = html;
                einsetzpunkteSuchen(objectIds);
            }
            function einsetzpunkteSuchen(objectIds) {

                requestString = requestBegin +
                                maxfeat +
                                requestAfterMaxFeatures +
                                requestGNREinsetzpunkte +
                                objectidFilterStart;
```

Vollständiger Web-Client (Teil 5)

```
                if (objectIds.length > 1) {
                    requestString += objectidOrStart;
                }

                for(var objectId in objectIds) {
                    requestString += objectidPropertyStart +
                                     objectIds[objectId] +
                                     objectidPropertyEnd ;
                }


                if (objectIds.length > 1) {
                    requestString += objectidOrEnd;
                }
                requestString += objectidFilterEnd +
                                 requestEnd;

                document.getElementById("requestIDEP").value = requestString;

                OpenLayers.Request.POST({
                     url: '<%= wfsEinsetzpunkteUrl %>',
                     data: requestString,
                     callback: handleResultEinsetzpunkte
                 });

            }
            function handleResultEinsetzpunkte(req) {
                var responsedText = req.responseText;
                document.getElementById("responseIDEP").value = responsedText;

                var g =  new OpenLayers.Format.GML.v3({'featureType': stringGNREinsetzpunkte,
                   'featureNS' : '<%= wfsNameSpace %>',
                   'geometryName' : 'ASP_WFS:SHAPE', 'xy' : false, 'srsName': '<%= epsgCode %>'});
                html = ""
                features = g.read(responsedText);
                vectorLayer.destroyFeatures();
                vectorLayer.addFeatures(features);

                for(var feat in features) {
                    html += "Feature: Geometry: "+ features[feat].geometry+",";
                    html += "<ul>";
                    for (var j in features[feat].attributes) {
                        html += "<li>"+j+":"+features[feat].attributes[j]+"</li>";
                    }
                    html += "</ul>"
                }
                vectorLayer.redraw();
                document.getElementById("responseFeatIDEP").value = html;
            }

        </SCRIPT>
    </HEAD>
    <BODY onload="init()" leftMargin="0" topMargin="0" marginheight="0" marginwidth="0">

        <TABLE cellSpacing="0" cellPadding="0" width="100%" border="0">
            <TBODY>
                <TR>
                    <TD class="kopfleiste" vAlign="top" align="right">
                        <IMG height="50" src="bevci/BG_KOPFLEISTE.png" width="489" />
                        <IMG height="50" alt="BEV - Bundesamt für Eich- und Vermessungswesen" src="bevci/KOPFLEISTE.png" width="770" />
                    </TD>
                </TR>
            </TBODY>
        </TABLE>

        <TABLE cellSpacing="0" cellPadding="0" width="100%" border="0" id="body" >
            <TBODY>
                <TR>
                    <TD vAlign="top" width="85%">
                        <TABLE class="grau4" height="30" cellSpacing="0" cellPadding="0" width="100%" border="0">
                            <TBODY>
                                <TR>
                                    <TD width="100%" colSpan="3"><h1>WFS/WMS Web Client</h1></TD>
                                </TR>
                            </TBODY>
                        </TABLE>
```

Vollständiger Web-Client (Teil 6)

```
                    <TABLE class="weiss0" cellSpacing="0" cellPadding="0" width="100%" border="0">
                        <TBODY>
                            <TR>
                                <TD width="100%" colSpan="3">
                                    <DIV class=smallmap id=map></DIV>

                                    <DIV style="display:inline"> <!-- none inline -->
                                        <table>
                                            <tbody>
                                                <tr>
                                                    <td>Request:<br/>
                                                    <input type="text" name="requestID" id="requestID" size="140" /></td>
                                                </tr>
                                                <tr>
                                                    <td>Response:<br/>
                                                    <input type="text" name="responseID" id="responseID" size="140" /></td>
                                                </tr>
                                                <tr>
                                                    <td>ResponseFeatures:<br/>
                                                    <input type="text" name="responseFeatID" id="responseFeatID" size="140" /></td>
                                                </tr>
                                                <tr>
                                                    <td>Request-EP:<br/>
                                                    <input type="text" name="requestIDEP" id="requestIDEP" size="140" /></td>
                                                </tr>
                                                <tr>
                                                    <td>Response-EP:<br/>
                                                    <input type="text" name="responseIDEP" id="responseIDEP" size="140" /></td>
                                                </tr>
                                                <tr>
                                                    <td>ResponseFeatures-EP:<br/>
                                                    <input type="text" name="responseFeatIDEP" id="responseFeatIDEP" size="140" /></td>
                                                </tr>
                                            </tbody>
                                        </table>
                                    </DIV>

                                </TD>
                            </TR>
                        </TBODY>
                    </TABLE>
                </TD>
                <TD style="vertical-align:top" width="15%">
                    <TABLE class="grau1" cellSpacing="0" cellPadding="0" width="100%" border="0">
                        <TBODY>
                            <TR align="bottom">
                                <TD vAlign="top" width="100%">
                                    <h2>Suchparameter</h2>
                                </TD>
                            </TR>
                        </TBODY>
                    </TABLE>
                    <TABLE class="grau3" cellSpacing="0" cellPadding="0" width="100%" border="0">
                        <TBODY>
                            <TR align="top">
                                <TD vAlign="top" width="100%" style="padding:10px">
                                    <table border="1">
                                        <tbody>
                                            <tr>
                                                <td>Katastralgemeindenummer:<br/>
                                                    <input type="text" name="kgnr" id="kgnr" value="24351" size="21" />
                                                </td>
                                            </tr>
                                            <tr>
                                                <td>Grundstücksnummer:<br/>
                                                    <input type="text" name="gstnr" id="gstnr" value="981" size="21" />
                                                </td>
                                            </tr>
                                            <tr>
                                                <td>Maximale Features:<br/>
                                                    <input type="text" name="maxfeat" id="maxfeat" value="10" size="21" />
                                                </td>
                                            </tr>
                                            <tr>
                                                <td>
                                                    <input type="button" value="Suchen/Anzeigen" onclick="gstSuchen()" />
                                                </td>
                                            </tr>
                                        </tbody>
                                    </table>
```

Vollständiger Web-Client (Teil 7)

```
                                    </TD>
                                </TR>
                            </TBODY>
                        </TABLE>
                        <TABLE class="grau2" cellSpacing="0" cellPadding="0" width="100%" border="0">
                            <TBODY>
                                <TR align="left">
                                    <TD vAlign="top" width="100%"> </TD>
                                </TR>
                            </TBODY>
                        </TABLE>
                    </TD>
                </TR>
            </TBODY>
        </TABLE>
    </BODY>
</HTML>
```

4.3.7 Zusammenfassung

Nach dem korrekten Positionieren wird in dieser Lerneinheit gezeigt, wie das Suchergebnis am Besten dargestellt wird. Da die Karte bereits ausgerichtet ist, müssen noch die Grundstücke des Suchergebnisses mittels Marker gekennzeichnet werden. Dies erfolgt durch Verwendung der Einsetzpunkte der Grundstücksnummern, da diese garantiert innerhalb des Grundstückes, meist mittig, liegen. Zusätzlich wird noch das Problem bei WGS84 mit der Reihenfolge der Längen- und Breitengrade besprochen. Abschließend wird ein komplettes Beispiel für einen Web-Client gezeigt.

4.4 Zusammenfassung

Dieser Kurs zeigt, wie das Ergebnis einer Suche mittels WFS in einem Kartenfenster dargestellt werden kann. OpenLayers bietet mit den Vectorlayern die Funktionialitäten, um beliebige GML-Objekte darzustellen. Die Suche liefert die Punkte, welche als Marker gekennzeichnet werden. Dabei werden Themen wie Vectorlayer, Bounding Box, Marker, Einsetzpunkte, WGS84 und GML angesprochen.

Die erste Lerneinheit befasst sich mit dem Thema Vectorlayer. OpenLayers kann beliebige GML-Objekte innerhalb der Karte darstellen. Hierfür werden Vectorlayer als Kartenebene verwendet. Nach einer kurzen allgemeinen Einführung in Vectorlayer wird die Befüllung dieser durch Erklärung und Beispiele gezeigt. Um Suchergebnisse in der Karte darzustellen, können Marker verwendet werden. Marker sind Grafiken oder Formen, die in einem Vectorlayer dargestellt werden. Dabei dienen GML-Objekte als geografische Positionierung der Marker. Eine Liste von GML-Punkten dient den Markern als Positionen, um ein Suchergebnis darstellen zu können. Abschließend zeigt ein Beispiel, wie dies in OpenLayers erfolgen kann.

Die Lerneinheit 2 befasst sich mit der Problematik der Anzeige des Suchergebnisses. Wenn ein Grundstück gefunden wird, sollte es als Ganzes im Kartenfenster angezeigt werden. Hierfür kommt das Konzept der Bounding Boxes, als umschließende Rechtecke, zum Einsatz. Wenn die Suche als Ergebnis diese Bounding Boxes liefert, muss das Kartenfenster so ausgerichtet werden, dass alle Bounding Boxes dargestellt werden können. Es wird das umschließende Rechteck um die Bounding Boxes berechnet, und die Karte von OpenLayers auf diese Position verschoben und vergrößert. Sowohl für die Suche als auch für die korrekte Anzeige des Kartenfensters werden Beispiele gezeigt.

Nach dem korrekten Positionieren wird in der dritten Lerneinheit gezeigt, wie das Suchergebnis am Besten dargestellt wird. Da die Karte bereits ausgerichtet ist, müssen noch die Grundstücke des Suchergebnisses mittels Marker gekennzeichnet werden. Dies erfolgt durch Verwendung der Einsetzpunkte der Grundstücksnummern, da diese garantiert innerhalb des Grundstückes, meist mittig, liegen. Zusätzlich wird noch das Problem bei WGS84 mit der Reihenfolge der Längen- und Breitengrade besprochen. Abschließend wird ein komplettes Beispiel für einen Web-Client gezeigt.

4.5 Abbildungsverzeichnis

4.6 Glossar

Access Token:
enthält die Sicherheitsinformationen einer angemeldeten Session/Sitzung

AJAX:
siehe Asynchronous JavaScript and XML

Asynchronous JavaScript and XML:
eine Gruppe von zusammenhängenden Webtechnologien, die es ermöglichen, Browserinhalte asynchron zu ersetzen; hauptsächliche Verwendung für Rich Internet Applications oder interaktive Webanwendungen

Authentisierung:
stellt die Prüfung einer behaupteten Identität dar; im Internet zumeist durch Username und Passwort

Autorisierung:
die Zuweisung und Überprüfung von Zugriffsrechten auf Daten und Dienste

Baselayer:
Layer in OpenLayers, der als unterster Layer der Karte angezeigt wird

BEV:
siehe Bundesamt für Eich- und Vermessungswesen

Bundesamt für Eich- und Vermessungswesen:
österreichische Bundesbehörde mit den Aufgabenbereichen Vermessung und Geoinformation und Mess- und Eichwesen

Callback:
eine ausführbare Funktion, die als Parameter an eine andere Methode übergeben wird; ermöglicht der aufgerufenen Methode den aufrufenden Code auszuführen

CGI:
siehe Common Gateway Interface

Common Gateway Interface:
ein Standardprotokoll für die Schnittstelle zwischen Applikationen und einem Webserver

Comparison Operator:
Vergleichsoperator bei Filter Encoding

Container:
Webserver, Applicationserver

Coordinate Reference System:
auf Koordinaten basierendes System, um geografische Punkte zu referenzieren

CRS:
siehe Coordinate Reference System

Data Extent:
von OpenLayers gerechnetes kleinestes mögliches Rechteck, welches eine Liste von Features enthält

Decoding:
Demaskierung einer encodeten URL (Ersatz der Prozentzeichen durch das reservierte Zeichen; zB.: '*' statt %2A)

Digitale Katastralmappe:
Grafischer Datenbestand des Katasters im Koordinatensystem der Österreichischen Landesvermessung in digitaler Form

Document Object Model:
eine Variante, um HTML und XML Dateien mittels Objektzugriffen anzusprechen

Document Type Definition:
eine Menge von Deklarationen die den Aufbau eines XML-Dokumentes beschreibt (nur eingeschränkt möglich)

DOM:
siehe Document Object Model

Domain:
ein bekannter Netzwerkname, unter welchem Netzwerkkomponenten wie Server ansprechbar sind

DTD:
siehe Document Type Definition

Einsetzpunkt der Grundstücksnummer:
derjenige Punkt, an dem die Grundstücksnummer in der Digitalen Katastralmappe angezeigt wird

Encoding:
Maskierung von reservierten Zeichen einer URL durch die Darstellung mittels Prozentzeichen (zB.: %2A statt '*')

Entwurfsmuster:
eine generell verwendbare Lösung für ein allgemein auftretendes Problem

EPSG:
siehe European Petroleum Survey Group

European Petroleum Survey Group:
pflegt ein System von weltweit eindeutigen 4- bis 5-stelligen Schlüsselnummern für Koordinatenreferenzsysteme (EPSG-Codes)

Feature:
geografisches, digitales Objekt

Filter Encoding:
Spezifikation für die Einschränkung von Suchergebnissen

Geographic Markup Language:
spezifizierte Beschreibungssprache für Features

Geoid:
Das Geoid ist eine Bezugsfläche im Schwerefeld der Erde zur Vermessung und Beschreibung der Erdfigur. In guter Näherung wird das Geoid durch den mittleren Meeresspiegel der Weltmeere repräsentiert und ist damit in seiner Form außerhalb der Landmassen sichtbar.

Geoinformationssystem:
Informationssystem zur Erfassung, Bearbeitung, Organisation, Analyse und Präsentation geografischer Daten

GeoService:
ein von einem Geoserver zur Verfügung gestelltes WebService (meist WMS oder WFS)

Getter:
Methode, die den Inhalt eines Properites liefert

GIS:
siehe Geoinformationssystem

Global Position System:
ein globales Navigationssatellitensystem zur Positionsbestimmung und Zeitmessung

GML:
siehe Geographic Markup Language

GPS:

siehe Global Position System

Grundstücksnummer:
eindeutige Nummer eines Grundstückes in Österreich

Identitätsdiebstahl:
die missbräuchliche Nutzung personenbezogener Daten (der Identität) einer natürlichen Person durch Dritte

Interceptor:
steltl ein Entwurfsmuster dar, wo der Aufruf einer Methode durch den Interceptor abgefangen und kontrolliert wird

Java Architecture for XML Binding:
API um Daten aus einem XML-Dokument an Java-Klassen zu binden

JavaBean:
serialisierbare Klassen mit öffentlichem Standardkonstruktor und Kapselung der Properties durch Getter und Setter

JAXB:
siehe Java Architecture for XML Binding

JDOM:
bietet die Darstellung eines XML-Dokumentes als Baum im Hauptspeicher

Kapselung:
abschotten der Properties einer Java-Klasse (nur mehr durch Getter und Setter zugreifbar)

Katastralgemeindenummer:
eindeutige Nummer einer Katastralgemeinde in Österreich

Key-Value-Pair:
Kombination aus Schlüssel und Wert durch Gleichzeichen getrennt

Koordinatensystem:
dient der Positionsangabe von Punkten im Raum

KVP:
siehe Key-Value-Pair

Layer:
gezeichnete Ebene in der Karte

Layerswitcher:
bietet die Möglichkeit, einzelne Layer auszublenden

Logical Operator:
Logischer Operator beim Filter Encoding

Map:
stellt das Objekt eines Kartenfensters bei OpenLayers dar

Marker:
Punktobjekt zur Markierung in OpenLayers

Marshalling:
Umwandlung eines Java-Objektbaumes in ein XML-Dokument

MetaCarta:
Unternehmen, welches OpenLayers anfänglich entwickelte

MVC:
steht für das Entwurfsmuster Model-View-Controller

OGC:
siehe Open Geospatial Consortium

Open Geospatial Consortium:
fördert die Entwicklung raumbezogener Informationsverarbeitung auf Basis allgemeingültiger Standards zum Zweck der Interoperabilität

OpenLayers:
von MetaCarta entwickelte freie JavaScript Bibliothek für den Zugriff auf geografische Webservices und die Darstellung des Ergebnisses

Panning:
verschieben der Karte

Proxy:
ein Vermittler von Zugriffen auf Ressourcen

ProxyHost:
Parameter in OpenLayers, der den URL-Prefix für den Proxy-Zugriff definiert

Rasterdaten:
Bildinformationen, wobei die Daten in ein definiertes Raster aufgeteilt werden und jede entstandene Zelle einen Wert zugeteilt bekommt (Zelle = Pixel)

Request:
Anfrage an einen Server

Response:
Antwort des Servers

Sandboxkonzept:
stellt das Sicherheitskonzept der Browser für JavaScript dar

SAX:
siehe Simple API for XML

Servlet:
Java-Klassen, deren Instanzen innerhalb eines Java-Webservers Anfragen von Clients entgegen nehmen und beantworten

Session Token:
ein eindeutiger Indentifier, welcher vom Server generiert und zum Client geschickt wird, um eine Interaktionssession zu kennzeichnen; wird danach bei jedem Request an den Server statt Username und Passwort übermittelt

Setter:
Methode, um den Inhalt eines Properties zu verändern

Simple API for XML:
API zum Parsen von XML-Daten mittels Callbackfunktionen

Single Sign On:
entspricht einer Einmalanmeldung, nach welcher der Benutzer auf alle Rechner und Dienste, für die er berechtigt ist, zugreifen kann, ohne sich neu anmelden zu müssen

Softwarekomponente:
Teil einer Software

Spatial Operator:
Verschneidungsoperator für geografische Abfrage bei Filter Encoding

SSO:
siehe Single Sign On

StAX:
siehe Streaming API for XML

Streaming API for XML:
Verarbeitung eines XML-Baumes mittels Cursor

Tiles:
Teile einer Karte die mosaikartig ein Computerbild ergeben

Timeout:
Zeitraum bis zum Ablauf der Gültigkeit eines Tokens

Token:
ein Objekt, welches ein zumeist exklusives Recht repräsentiert, eine Aktion durchzuführen

Transactional WFS:
WFS, welches Daten am Server im Zuge einer Transaktion verändern kann

Unmarshalling:
Darstellen eines XML-Dokumentes als Java-Objektbaum

Validierung:
Prüfung des Inhaltes eines XML-Dokumentes gegen die Vorgaben aus dem XSD

Vektordaten:
räumliche Daten, die aus Punkt, Linie und Polygon aufgebaut sind

Verschneidung:
Überlagerung von unterschiedlichen Layern am Server zur Gewinnung neuer Objekte; dient der Abfrage, ob Objekte der einzelnen Layer einander berühren, beinhalten, etc.

Verwaltungsgrenzen:
Grenze einer Verwaltungseinheit; beim WMS des BEV sind die die Grenzen der Katastralgemeinden

WAR-File:
siehe WebArchive

WebArchive:
ZIP-File mit allen Servlets und JSP-Files für das Deployment auf einem WebContainer

Web-Client:
Client, welcher auf Ressourcen im Internet zugreift (zB.: auf WFS, WMS)

Web Feature Service:
OGC Spezifikation für WebServices, die GML-Objekte liefern

Web Map Service:
OGC Spezifikation für WebServices, die fertige Karten liefern

WFS:
siehe Web Feature Service

WFS-T:
siehe Transactional WFS

WGS84:
siehe World Geodetic System 1984

Wildcard:
Platzhalter bei Abfragen; kann eine oder mehrere Stellen symbolisieren

WMS:
siehe Web Map Service

World Geodetic System 1984:
ein geodätisches Referenzsystem als einheitliche Grundlage für Positionsangaben auf der Erde und im erdnahen Weltraum

XJC:

siehe XML Java Binding Compiler

XML:

dient der Darstellung von hierarchischen Daten in Textform

XMLHttpRequest:

eine API für asynchrone Aufrufe in JavaScript

XML Java Binding Compiler:

erstellt aus einem XSD-File die Objekthierarchie in Java

XSD:

dient dem Definieren von Strukturen für XML-Dokumente

Zertifikat:

bestätigt den Besitzer oder die Eigenschaften eines öffentlichen Schlüssels

Zooming:

vergrößern/verkleinern der Karte