GeoServer in action

Release 1.0

terrestris GmbH & Co. KG (D. Koch, N. Bühner)

09.03.2015

Herzlich Willkommen beim **GeoServer in action** Workshop auf der FOSSGIS 2015 in Münster.

Dieser Workshop soll Ihnen einen umfassenden Überblick über den GeoServer als Web-Mapping-Lösung geben. Bitte stellen Sie sicher, dass Sie die Schritte der *Vorarbeiten und generelle Informationen* (Seite 2)-Seite ausgeführt haben.

Dieser Workshop ist aus einer Reihe von Modulen zusammengestellt. In jedem Modul werden Sie eine Reihe von Aufgaben lösen, um ein bestimmtes Ziel zu erreichen. Jedes Modul baut iterativ Ihre Wissensbasis auf.

Die folgenden Module werden in diesem Workshop behandelt:

Vorarbeiten und generelle Informationen **(Seite 2)** Grundlegende Informationen zur Workshop-Umgebung (OSGeoLive, Pfade, URLs, Credentials) und notwendige Installationen (maven)

Basiswissen GeoServer **(Seite 5)** Basiswissen Geoserver, Kompilieren auf Basis des Source-Codes mit maven, Installation von Plugins und Extensions

REST-Schnittstelle **(Seite 11)** Einrichtung von Workspaces, Stores, Styles und Layern über die REST-Schnittstelle

Tipps, Tricks & Troubleshooting **(Seite 24)** Performance-Optimierung, Protokollierung, Debugging und hilfreiche Tipps

KAPITEL 1

Vorarbeiten und generelle Informationen

- Rechner mit OSGeoLive-Medium hochfahren
- Sprache auswählen (Deutsch für korrekte Tastaturbelegung)
- *Lubuntu ohne Installation ausprobieren* auswählen
- Benutzer: user; Passwort: user (wird vermutlich nicht benötigt)

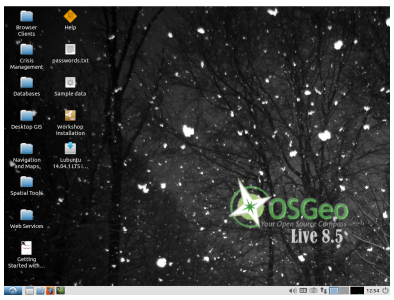


Abbildung 1.1: Die Startansicht der OSGeo Live 8.5 auf Ihrem Rechner.

1.1 Setup-Script ausführen

Es gibt ein Skript, welches ihr OSGeoLive-System für diesen Workshop einrichtet. Das Skript führt die folgenden Aktionen aus:

- Installation des *Build-Management-Tools* Maven
- Deinstallation der INSPIRE-Erweiterung vom GeoServer (wir werden diese Erweiterung später mit maven selbst kompilieren und auf dem GeoServer installieren)
- Download des Quellcodes der INSPIRE-Erweiterung für den GeoServer
- Initialisierung eines lokalen Maven-Repositories (dieser Schritt ist nicht zwingend nötig, beschleunigt aber die späteren Aufrufe von Maven-Befehlen von vielen Minuten auf wenige Sekunden)

Sollten Sie die OSGeoLive zwischenzeitlich neu starten, müssen Sie das Skript erneut ausführen!

Bemerkung: Sie können Inhalte aus der Zwischenablage, die sie etwa zuvor mit der Tastenkombination `STRG + C` kopiert haben im Terminal mit der Tastenkombination `STRG +`

UMSCHALT + V einfügen! Alternativ können Sie auch einen Rechtsklick in das Terminal machen und dort *Einfügen* wählen.

Um das Skript zu starten, führen Sie bitte den folgenden Befehl auf dem Terminal (Button  (LXTerminal) im unteren Systempanel) aus:

```
curl \
  http://workshops.terrestris.de/geoserver/geoserver_in_action/_static/setup_geoser
  sudo bash
```

Es kann einen Moment dauern bis das Skript durchgelaufen ist!

Machen Sie sich währenddessen schonmal mit den Pfaden und Zugangsdaten des GeoServers vertraut:

1.2 Pfade, URLs und Zugangsdaten

- GeoServer: http://localhost:8082/geoserver (muss zunächst gestartet werden, siehe unten)
- Zugangsdaten GeoServer: `admin:geoserver`
- GeoServer (Dateisystem): `/usr/local/lib/geoserver-2.6.1/`

1.3 Überprüfung der Maven-Installation

Sobald das Skript durchgelaufen ist, sollten Sie überprüfen, ob die Maven- Installation erfolgreich war, indem Sie folgenden Befehl auf dem Terminal ausführen:

```
mvn -v
```

Sie sollten folgende Ausgabe erhalten:

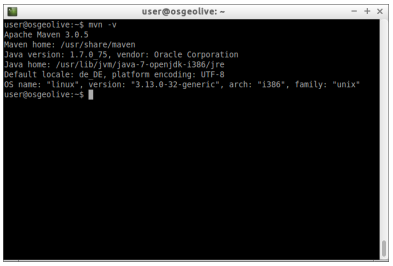


Abbildung 1.2: Erfolgreiche Installation von maven.

1.4 Starten des GeoServers

Der GeoServer kann durch einen Doppelklick auf **Start GeoServer** im Ordner **Web Services** auf dem Desktop der OSGeoLive gestartet werden:

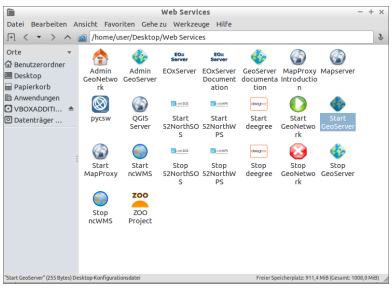


Abbildung 1.3: GeoServer starten.

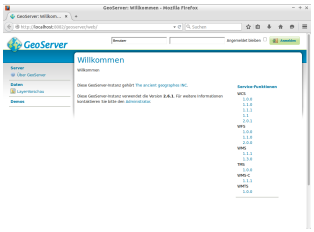


Abbildung 1.4: GeoServer-Weboberfläche nach erfolgreichem Start

Im *folgenden Abschnitt* (Seite 5) werden wir mit GeoServer-Basiswissen fortfahren.

KAPITEL 2

Basiswissen GeoServer

Der GeoServer ist ein offener, Java-basierter Server, der es ermöglicht Geodaten auf Basis der Standards des Open Geospatial Consortium (OGC) (insb. WMS und WFS) anzuzeigen und zu editieren. Eine besondere Stärke des GeoServers ist die Flexibilität mit der er sich um zusätzliche Funktionalität erweitern lässt.

Der GeoServer ist gut dokumentiert. Die Dokumentation unterteilt sich dabei in eine Benutzer- und eine Entwicklerdokumentation:

- Benutzerdokumentation: http://docs.geoserver.org/stable/user/
- Entwicklerdokumentation: http://docs.geoserver.org/stable/developer/

Die beiden Links verweisen auf die Dokumentationen der letzten stabilen Version. Das *stable* in der URL kann auch durch eine Versionsnummer ersetzt werden, falls man die Dokumentation einer bestimmten GeoServer-Version aufrufen möchte. Im Rahmen dieses Workshops wird die **Version 2.6.1** verwendet, also: http://docs.geoserver.org/2.6.1/user/

Abbildung 2.1: GeoServer-Weboberfläche nach erfolgreichem Login

Üblicherweise wird der GeoServer für einen Produktivbetrieb als (Java-)Standalone-Servlet in Form einer `.war`-Datei bereitgestellt, welche unter http://geoserver.org/download/ heruntergeladen werden kann. Die `.war`-Datei muss anschließend auf einem Servlet-Container, z.B. Tomcat oder Jetty veröffentlicht, d.h. *deployed* werden. Anschließend kann die Weboberfläche des GeoServers über den Browser aufgerufen werden.

Weitere Details zur klassischen WAR-Installation finden sich hier.

Bemerkung: Der GeoServer ist auf dem OSGeoLive-System bereits vorinstalliert und kann im Rahmen des Workshops unter http://localhost:8082/geoserver aufgerufen werden (siehe *hier* (Seite 3)). Diese Variante unterscheidet sich von dem klassischen *Deployment* als `.war`-Datei, da hier ein Java-Programm (`start.jar`) ausgeführt wird, welches programmatisch einen Jetty-Server mit dem Geoserver startet. Für die Inhalte des Workshops ist dies aber nicht von Bedeutung.

Im Folgenden werden wir uns zunächst einen Überblick über die GeoServer-Ordnerstruktur verschaffen. Anschließend wird erläutert wie sich GeoServer-Erweiterungen installieren lassen. Zum Abschluss dieses Modules wird erklärt wie sich der Quellcode des GeoServers (bzw. einzelner Erweiterungen) mit dem *Build-Management-Tool* Maven kompilieren lässt.

2.1 Ordnerstruktur des GeoServers

Im Folgenden wird die Ordnerstruktur des GeoServers erläutert. Ausgangspunkt ist das GeoServer-Verzeichnis:

```
/usr/local/lib/geoserver-2.6.1/
```

Dabei sind die folgenden Unterordner von besonderer Bedeutung:

Verzeichnis	Bedeutung
`bin/`	Enthält Skripte zum Starten und Stoppen des GeoServers (Jetty-Variante/OSGeoLive).
`data_dir/`	Konfiguration der GeoServer-Daten (z.B. *Arbeitsbereiche, Datenquellen, Layer oder Stile*). Für Produktivsysteme wird (durch Konfiguration des GeoServers) grundsätzlich empfohlen ein *DATA_DIR* außerhalb der Webapplikation zu verwenden, da sich der GeoServer auf diese Weise zu neueren Versionen updaten lässt ohne dass die Konfigurationsdaten verloren gehen. Details gibt es hier.
`data_dir/logs/`	Enthält die Log-Dateien des GeoServers. Auf das Logging wird zum Ende des Workshops auch *hier* (Seite 24) eingegangen.
`webapps/geoserver/WEB-INF/lib/`	Enthält `.jar`-Dateien, d.h. Java-Kompilate, die beim Starten des Servers in den *ClassPath* geladen werden. Dabei handelt es sich einerseits um Abhängigkeiten (*Dependencies*) des GeoServers zu anderen (OpenSource-)Bibliotheken, die benötigt werden, damit der GeoServer lauffähig ist, z.B. das Spring Framework. Andererseits werden Erweiterungen (ebenfalls in Form von `.jar`-Dateien) in diesen Ordner installiert.

Bemerkung: Die hier erläuterte Struktur bezieht sich auf die Jetty-Umgebung im OSGeoLive-System. In anderen Umgebungen (z.B. auf einem Tomcat mit klassischer `.war`-Datei-Installation) kann die Struktur abweichen.

Im *folgenden Abschnitt* (Seite 6) wird erklärt wie sich die Funktionalität des GeoServers durch das Einbinden zusätzlicher Module erweitern lässt.

2.2 Installieren von Erweiterungen

Eine Stärke des GeoServers besteht darin, dass seine Funktionalität durch das Einbinden von zusätzlichen Modulen erweitert werden kann. So gibt es z.B. Erweiterungen, die es ermöglichen Vektor-Datenquellen zu verwenden, die ihre Daten aus bestimmten SQL-Datenbanken beziehen (z.B. MySQL oder Oracle). Ebenso gibt es Erweiterungen, die es ermöglichen *Excel* (und weitere) als Ausgabeformat zu unterstützen.

2.2.1 Übersicht über verfügbare Erweiterungen

Auf http://geoserver.org/release/2.6.1/ finden Sie im Bereich *Extensions* eine Auflistung zahlreicher Erweiterungen, die als stabile betrachtet werden und im Rahmen eines Release-Prozesses bereitgestellt werden. Darüberhinaus gibt es noch *Community-Extensions*, die einen experimentellen oder instabilen Status haben und kein Teil des offiziellen *Release*-Prozesses sind.

Im Rahmen des Workshops werden wir (exemplarisch) das WPS-Modul installieren. WPS steht für *Web Processing Service* und ist (wie WMS und WFS) ein Standard des OGC, in dem Regeln für das Anfragen und Antworten von (räumlichen) Prozessen, definiert sind.

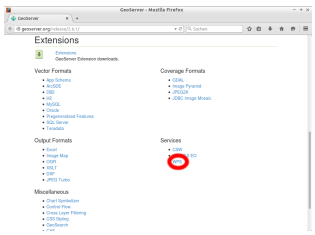


Abbildung 2.2: GeoServer-Erweiterungen. Rot markiert ist die WPS-Erweiterung.

2.2.2 Installieren der WPS-Erweiterung

Führen Sie die folgenden Schritte aus, um diese Erweiterung zu installieren:

1. Stoppen Sie den Geoserver durch einen Doppelklick auf **Stop GeoServer** im Ordner **Web Services** auf Ihrem Desktop.
2. Laden Sie die WPS-Erweiterung von `hier` (oder der offiziellen Release-Seite) herunter. Wählen Sie bitte *Datei speichern* und nicht *Öffnen mit*!
3. Wechseln Sie im Terminal in das im *vorigen Abschnitt* (Seite 6) erläuterte Verzeichnis zur Installation von Erweiterungen:

   ```
   cd /usr/local/lib/geoserver-2.6.1/webapps/geoserver/WEB-INF/lib/
   ```

4. Da Sie root-Rechte benötigen, um in das `lib`-Verzeichnis schreiben zu können, muss der folgende Befehl zum Entpacken des Archivs mit `sudo` ausgeführt werden:

   ```
   sudo unzip /home/user/Downloads/geoserver-2.6.1-wps-plugin.zip
   ```

5. Wenn die `jar`-Dateien erfolgreich in das `lib`-Verzeichnis entpackt wurden, muss der GeoServer wieder gestartet werden. Dazu klicken Sie auf **Start GeoServer** im Ordner **Web Services** auf dem Desktop.

Sobald der GeoServer hochgefahren ist, können wir in seiner Weboberfläche überprüfen, ob die Installation der WPS-Erweiterung erfolgreich war. Dazu loggen wir uns zunächst mit den Zugangsdaten `admin:geoserver` ein. Anschließend muss in dem Menü auf der linken Seite (im unteren Bereich) auf `Demos` geklickt werden. Hier findet sich nun ein Eintrag `WPS Request-Builder`, den es an dieser Stelle zuvor nicht gegeben hat.

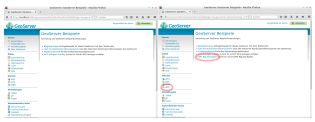


Abbildung 2.3: GeoServer-Weboberfläche (Bereich *Demo*) vor und nach der WPS-Installation

---

Bemerkung: Wenn Sie auf `WPS Request-Builder` klicken und dort den ersten verfügbaren WPS-Prozess *JTS:area* wählen, im Bereich "Process inputs" *TEXT* und *application/wkt* wählen, als Dateneingabe `POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))` verwenden und anschließend auf *Execute process* klicken, müssten Sie das Ergebnis *550.0* erhalten.

Im *folgenden Abschnitt* (Seite 8) wird erklärt wie Sie Erweiterungen (bzw. den GeoServer als solchen) auf Basis des Quellcodes mit maven kompilieren.

2.3 Kompilieren auf Basis des Quellcodes

In diesem Abschnitt wird gezeigt wie sich der Quellcode des GeoServers bzw. einzelner Module mit dem *Build-Management-Tool* Maven kompilieren lässt. Maven wird von der GeoServer-Community für die Erstellung und Verwaltung der Software eingesetzt. Anschließend könnten Sie z.B. den Original-Code für spezielle Anwendungsfälle anpassen (oder erweitern) und eine derart modifizierte GeoServer-Version einsetzen.

Bemerkung: Maven ist eine auf Java basierende OpenSource-Software zur standardisierten Erstellung und Verwaltung von (Java-)Programmen. Ausgehend von einer Validierung der Quelldateien, über das Kompilieren, Testen, Verpacken bis hin zum Installieren der Software bietet Maven Unterstützung für den gesamten Lebenszyklus einer Software. Leider können wir an dieser Stelle nicht näher auf Maven eingehen, da dies den Rahmen des Workshops sprengen würde. Einen Einstieg in die Maven-Welt finden Sie z.B. unter http://maven.apache.org/guides/.

Bevor wir konkrete Maven-Befehle aufrufen, um sogenannte *Maven-Artefakte* zu erzeugen (welche in unserem Falle Java-Kompilaten, also `.jar`-Dateien, entsprechen) wollen wir zunächst aufzeigen wie Sie grundsätzlich mit dem GeoServer-Quellcode und Maven arbeiten können.

Anschließend werden wir exemplarisch den Quellcode der INSPIRE-Erweiterung des GeoServers mit Maven kompilieren und auf dem GeoServer installieren.

2.3.1 GeoServer-Quellcode und Maven

Der GeoServer-Quellcode wird mit git verwaltet. *Git* ist eine freie Software zur verteilen Versionsverwaltung von Dateien, mit dem z.B. auch der Linux-Kernel verwaltet wird. Den Quellcode des GeoServers finden Sie hier:

https://github.com/geoserver/geoserver

Wenn Sie sich dort in den Ordner `src/extension/inspire` durchklicken, können Sie feststellen, dass es in jedem dieser drei Ordner eine Datei `pom.xml` gibt. *POM* steht für Project Object Model und es handelt sich hier um die Konfigurationsdateien für Maven, in denen z.B. Abhängigkeiten zu anderen Bibliotheken definiert sind.

Jede Datei `pom.xml` repräsentiert dabei ein eigenes Maven-Modul. Die `pom.xml` im Ordner `src/` ist die zentrale Maven-Konfigurationsdatei auf höchster Ebene, also des GeoServers als Gesamtsystem. Die `pom.xml` im Ordner `src/extension/` definiert ein Maven-Submodul *extension*, die `pom.xml` in `src/extension/inspire/` ist wiederum ein Submodul von *extension*.

Wenn Sie auf dem Terminal unterwegs sind und in ein Verzeichnis navigieren, das eine `pom.xml` enthält, können Sie dort Maven-Befehle ausführen. Der Befehl `mvn package` würde z.B. das entsprechende Artefakt bauen und in einen Unterordner `target/` ablegen.

Details zur Verwendung von Maven mit dem GeoServer finden Sie unter http://docs.geoserver.org/latest/en/developer/maven-guide/index.html.

2.3.2 Kompilieren der INSPIRE-Erweiterung

Wir werden nun die INSPIRE-Erweiterung des GeoServers auf Basis des Quellcodes selber kompilieren. Anschließend kann die Erweiterung analog zum *vorigen Abschnitt* (Seite 6) im GeoServer installiert werden.

Bemerkung: Das Herunterladen des (gesamten) GeoServer-Quellcodes und insbesondere das (erstmalige) Ausführen bestimmter Maven-Befehle kann u.U. sehr lange dauern. Im Falle von Maven ist dies darauf zurückzuführen, dass Maven zunächst alle benötigten Abhängigkeiten auflöst und diese anschließend aus öffentlich verfügbaren Maven- *Repositories* in ein lokales Maven-*Repository* auf ihrem Rechner herunterlädt bzw. installiert. Um diesen Prozess zu beschleunigen hat das Script, welches Sie zu Beginn des Workshops ausgeführt haben, bereits vorbereitende Maßnahmen getroffen.

Zu Beginn des Workshops haben Sie ein Skript ausgeführt, welches gewisse Vorbereitungen getroffen hat, damit die Paketierung der INSPIRE-Erweiterung bei der erstmaligen Ausführung der Maven-Befehle nur wenige Sekunden statt vieler Minuten dauert.

Sie sollten in Ihrem Home-Verzeichnis `/home/user` ein Verzeichnis `src/` vorfinden, in welchem sich **ausschließlich** der Quellcode der INSPIRE-Erweiterung (in Version 2.6.1) befindet. Das Skript hat darüberhinaus alle (im nächsten Schritt) benötigten Abhängigkeiten in ihr lokales Maven-Repository `~/.m2` vorinstalliert, welche Maven ansonsten selbst installieren (und was entsprechend länger dauern) würde.

Wechseln Sie nun in das Verzeichnis mit der `pom.xml` und dem Quellcode der INSPIRE-Erweiterung:

```
cd ~/src/extension/inspire/
```

Wir können nun den Maven-Befehl ausführen, mit dem sich die INSPIRE-Erweiterung (auf Basis des aktuellen Quellcodes) paketieren lässt:

```
mvn package
```

Sie können die Paketierung auch beschleunigen, indem Sie auf das Ausführen von Tests verzichten:

```
mvn package -DskipTests
```

Ebenso könnten Sie nur die Tests ausführen:

```
mvn test
```

Es ist auch möglich auf der Ebene des *extension*-Moduls die INSPIRE-Erweiterung zu *bauen*. Hierzu muss zunächst in das *extension*-Verzeichnis gewechselt werden, um dort den `mvn package`-Befehl unter Verwendung eines bestimmten Profils zu verwenden:

```
cd ~/src/extension/

mvn package -P inspire -DskipTests
```

Wenn einer der `mvn package`-Befehle erfolgreich durchgelaufen ist, finden Sie im Verzeichnis `~/src/extension/inspire/target/` das erzeugte Kompilat (`gs-inspire-2.6.1.jar`), welches wie im *vorigen Abschnitt* (Seite 6) auf dem GeoServer installiert werden kann.

```
sudo cp \
  ~/src/extension/inspire/target/gs-inspire-2.6.1.jar \
  /usr/local/lib/geoserver-2.6.1/webapps/geoserver/WEB-INF/lib
```

Sobald dies geschehen ist (GeoServer-Neustart nicht vergessen!), erscheint bei der WMS-Konfiguration in der GeoServer- Weboberfläche ein zusätzlicher Bereich *INSPIRE* (unten).

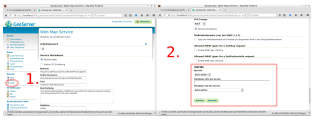


Abbildung 2.4: GeoServer-Weboberfläche (Bereich *WMS*) nach der INSPIRE-Installation

Im *folgenden Abschnitt* (Seite 11) werden werden Sie die GeoServer- REST-Schnittstelle kennenlernen, mit der sich viele Dinge, die Sie über die Weboberfläche konfigurieren können, auf programmatischem Wege ausführen können.

KAPITEL 3

REST-Schnittstelle

Die REST-API erlaubt das Lesen, Schreiben, Aktualisieren und Entfernen von (fast) allen GeoServer Katalogelementen direkt über das HTTP-Protokoll, d.h. ohne die Verwendung der graphsichen GeoServer Benutzeroberläche. Hierzu zählen z.B. Arbeitsbereiche, Datenspeicher, Layer, Stile und Gruppenlayer. In diesem Modul werden die Grundprinzipien der REST-API vom GeoServer anhand von einfachen Beispielen erläutert. Bei der Auswahl der folgenden Beispiele wurde auf eine hohe Praxistauglichkeit und Wiederverwendbarkeit geachtet.

Doch zunächst: Was ist REST?

REST (oder auch *ReST*) steht Representational State Transfer und ist ein Architekturstil zur Realisierung von Web Services und wir daher auch häufig im Zusammenhang mit dem Begriffspaar *RESTful Webservices* genannt. Kernelement des Architekturstils sind die sog. Ressourcen. Eine Ressource sollte dabei die folgenden Anforderungen erfüllen (nach [1]):

- **Adressierbarkeit:** Jede Ressource (z.B. ein GeoServer Layer) muss über eine eindeutige *URI* (Unique Resource Identifier) erreichbar sein.
- **Zustandslosigkeit:** Jede Kommunikation über REST enthält alle Informationen, die für Server oder Client zum Verständnis notwendig sind, Zustandsinformationen (z.B. Sessions) werden nicht von Server oder Client gespeichert. Hieraus ergibt sich eine einfache Umsetzung der Skalierbarkeit von Webservices (z.B. für die Lastverteilung).
- **Einheitliche Schnittstelle:** Der Zugriff auf jede Ressource muss über standardisierte Methoden erlangt werden können. Übersetzt auf das HTTP-Protokoll sollten Operationen auf Ressourcen über die Standard-HTTP Methoden (siehe Tabelle) implementiert werden.
- **Entkopplung von Ressourcen und Repräsentation:** Jede Ressource kann in unterschiedlichen Darstellungsformen/Repräsentationen (z.B. im HTML-, JSON- und XML-Format) existieren.

[1] Fielding, R.T. (2000): Architectural Styles and the Design of Network-based Software Architectures. Abrufbar unter: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Tabelle 3.1: Übersicht der HTTP-Methoden, verändert nach http://de.wikipedia.org/wiki/Representati
23.02.2015

Operation	Beschreibung
GET	fordert die angegebene Ressource vom Server an. GET weist keine Nebeneffekte auf. Der Zustand am Server wird nicht verändert, weshalb GET als sicher bezeichnet wird.
POST	fügt eine neue (Sub-)Ressource unterhalb der angegebenen Ressource ein. Da die neue Ressource noch keinen URI besitzt, adressiert der URI die übergeordnete Ressource. Als Ergebnis wird der neue Ressourcenlink dem Client zurückgegeben. POST kann im weiteren Sinne auch dazu verwendet werden, Operationen abzubilden, die von keiner anderen Methode abgedeckt werden.
PUT	die angegebene Ressource wird angelegt. Wenn die Ressource bereits existiert, wird sie geändert.
PATCH	ein Teil der angegebenen Ressource wird geändert. Hierbei sind Nebeneffekte erlaubt.
DELETE	löscht die angegebene Ressource.
HEAD	fordert Metadaten zu einer Ressource an.
OPTIONS	prüft, welche Methoden auf einer Ressource zur Verfügung stehen.
CONNECT	Dient dazu, die Anfrage durch einen TCP-Tunnel zu leiten. Wird meist eingesetzt, um eine HTTPS-Verbindung über einen HTTP-Proxy herzustellen.
TRACE	Gibt die Anfrage zurück, wie sie der Zielserver erhält. Dient etwa dazu, um Änderungen der Anfrage durch Proxyserver zu ermitteln.

Üblicherweise wird zur Realisierung von REST-Diensten das HTTP-Protokoll genutzt, deren Operationen in der o.g. Tabelle aufgezeigt sind. Jede Anfrage an einen REST- Dienst über eine dieser Operationen wird neben der REST-Nachricht (sofern vorhanden) einen Statuscode an den Client senden. Die folgendene Tabelle enthält eine Übersicht der zu erwartenden Statuscodes und der damit verbundenen Fehlerquellen:

Tabelle 3.2: HTTP-Statuscodes, verändert nach http://docs.geoserver.org/stable/en/user/rest/api/det
codes, 23.02.2015

Statuscode	Statustext	Beschreibung
200	OK	Request war erfolgreich
201	Created	Eine neue Ressource (z.B. ein Layer) wurde erfolgreich erstellt
403	Forbidden	Keine Berechtigung vorhanden
404	Not Found	Ressource oder Endpoint nicht gefunden
405	Method Not Allowed	Falsche Operation für Ressource oder Endpoint (z.B. Request mit GET, aber nur PUT/POST erlaubt)
500	Internal Server Error	Fehler beim Ausführen (z.B. Syntaxerror im Request oder Fehler beim Verarbeiten)

Im vorliegenden Modul werden wir nun an Beispielen die GeoServer-REST API nutzen, um die bestehende Ressouren auszulesen, einen neuen Layer anzulegen, einen bestehenden Layer zu editieren und abschließend einen bestehenden Layer zu entfernen:

3.1 Katalog auslesen

Wie bereits in der *Einführung zu REST* (Seite 11) aufgeführt, ist eine zentrale Bedingung von REST die Adressierbarkeit, d.h. jede Katalogkonfiguration im GeoServer besitzt eine eindeutige URL. Da sich diese Konfiguration logischerweise je nach Instanz unterscheidet, aber die Voraussetzung für die späteren Manipulations-Operationen notwendig ist, muss es eine Möglichkeit geben, diese Ressourcen abzubilden. Das Auslesen des existierenden Katalogs und der entsprechenden eindeutigen REST URLs ist direkt über den Browser möglich und dabei interaktiv, d.h. navigierbar aufgebaut.

Bemerkung: Wir werden in diesem Workshop die REST-Schnittstelle über das Kommandozeilentool *cURL* ansprechen. *cURL* ist ein Programm zur Client-Server Kommunikation und unterstützt neben vielen anderen auch das HTTP Netzwerkprotokoll. Für einfache Debugging-Prozesse zu empfehlen, sind für den produktiven Einsatz der REST-Schnittstelle Clientbibliotheken in den Programmiersprachen Java, Python, PHP und Ruby, die eine direkte Kommunikation aus dem Programmcode heraus erlauben, verfügbar und zu empfehlen.

GeoServer Configuration API

Öffnen Sie ein Browserfenster und geben Sie dort die folgende Adresse ein (Die Ressource benötigt eine Authentifizierung mit einem gültigen GeoServer Nutzer, geben Sie hier **admin:geoserver** ein):

```
http://localhost:8082/geoserver/rest
```

Nach Aufruf der Adresse erscheint im Browser eine einfache Liste, die eine Übersicht der aktuellen GeoServer Configuration API bietet (siehe Abbildung).

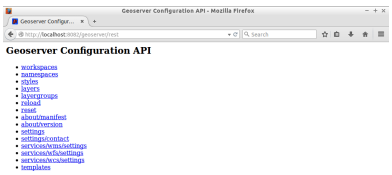


Abbildung 3.1: Startansicht der GeoServer Configuration API.

Die Listenansicht ist voll navigierbar und eindeutig zuordenbar. Eine Auswahl im Browser (z.B. des Eintrags *workspaces*) navigiert den Browser zur eindeutigen URL *http://localhost:8082/geoserver/rest/workspaces*. Der Aufbau der Liste (bei Auswahl eines Arbeitsbereichs) folgt dabei der logischen Struktur des GeoServer-Katalogs:

```
workspace
  |
  +--datastore
       |
       +--featuretype
```

GeoServer Configuration API Formate

Die obigen Aktionen im Browser rufen einen Endpunkt standardmäßig im *HTML*-Format auf. Der GeoServer unterstützt darüberhinaus die Formate *JSON* (JavaScript Object Notation) und

XML (Extensible Markup Language), die insbesondere bei der Manipulation einer Ressource relevant sind. Vergleichen Sie die folgenden Ausgaben im Browser:

```
http://localhost:8082/geoserver/rest/workspaces

http://localhost:8082/geoserver/rest/workspaces.json

http://localhost:8082/geoserver/rest/workspaces.xml
```

sowie

```
http://localhost:8082/geoserver/rest/workspaces/topp/datastores/states_shapefile/fe

http://localhost:8082/geoserver/rest/workspaces/topp/datastores/states_shapefile/fe

http://localhost:8082/geoserver/rest/workspaces/topp/datastores/states_shapefile/fe
```

Die obigen Befehle können ebenfalls direkt über cURL aufgerufen werden. Öffnen Sie hierzu das Terminal (über den Button  (LXTerminal) im unteren Systempanel) und fügen Sie dort den folgenden Befehl ein. Der aktuelle Pfad im Terminal ist dabei irrelevant.

```
curl \
  -v \
  -u admin:geoserver \
  -XGET \
  -H "Accept: text/html" \
  http://localhost:8082/geoserver/rest/workspaces
```

Der obige Request ist analog zu dem ersten hier aufgeführten Beispiel *http://localhost:8082/geoserver/rest/workspaces* und wird entsprechend die selbe Antwort im HTML-Format liefern (siehe Output aller vorhandenen Arbeitsbereiche im HTML-Format). Das Format der Response ist jedoch auch über cURL steuerbar. Hierzu muss lediglich der *Accept* Header des Requests von *text/html* auf *text/json* bzw. *text/xml* gesetzt werden.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>GeoServer Configuration</title>
    <meta name="ROBOTS" content="NOINDEX, NOFOLLOW"/>
</head>
<body>

Workspaces
<ul>
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/it.geosolutions.html
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/cite.html">cite</a>
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/tiger.html">tiger</a
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/sde.html">sde</a></l
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/topp.html">topp</a><
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/sf.html">sf</a></li>
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/nurc.html">nurc</a><
  <li><a href="http://localhost:8082/geoserver/rest/workspaces/cartaro.html">cartar
</ul>
</body>
</html>
```

Nachdem wir die grundlegenden Funktionen zum Auslesen der Katalogeinstellungen kennengelernt haben, können Sie mit dem Kapitel *Katalogeinträge erzeugen* (Seite 15) fortfahren.

3.2 Katalogeinträge erzeugen

Nachdem wir die REST-API kennengelernt und über die `GET`-Operation einige Informationen unseres GeoServers abgerufen haben, werden wir im nächsten Schritt einen neuen Arbeitsbereich inklusive eines Datenspeichers und Feature Types über REST anlegen.

3.2.1 Arbeitsbereich erstellen

Öffnen Sie das Terminal (falls noch nicht geschehen) und geben Sie den folgenden Befehl zum Erstellen eines neuen Workspaces namens *fossgis* ein:

```
curl \
  -v \
  -u admin:geoserver \
  -XPOST \
  -H "Content-type: text/xml" \
  -d "<workspace>
        <name>fossgis</name>
      </workspace>" \
  http://localhost:8082/geoserver/rest/workspaces
```

Der obige Aufruf unterscheidet sich in zwei wesentlichen Punkten von den bisherigen Read-Operationen: Wir nutzen für das Erstellen einer neuen Ressource im Gegensatz zur HTTP-Operation `GET` die Operation `POST` und schicken einen `XML` Content `<workspace>...</workspace>` an eine eindeutige URL der REST-API. Die Adresse haben wir durch die bisherigen Schritte identifizieren können. Nach Aufruf des Befehls sollte im Terminal folgende Ausgabe erscheinen:

```
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8082 (#0)
* Server auth using Basic with user 'admin'
> POST /geoserver/rest/workspaces HTTP/1.1
> Authorization: Basic YWRtaW46Z2Vvc2VydmVy
> User-Agent: curl/7.35.0
> Host: localhost:8082
> Accept: */*
> Content-type: text/xml
> Content-Length: 69
>
* upload completely sent off: 69 out of 69 bytes
< HTTP/1.1 201 Created
< Date: Tue, 24 Feb 2015 10:03:56 GMT
< Location: http://localhost:8082/geoserver/rest/workspaces/fossgis
* Server Noelios-Restlet-Engine/1.0..8 is not blacklisted
< Server: Noelios-Restlet-Engine/1.0..8
< Transfer-Encoding: chunked
<
* Connection #0 to host localhost left intact
```

An dieser Stelle sind zwei Informationen für uns entscheidend:

1. `HTTP/1.1 201 Created`: Der Aufruf wurde erfolgreich bearbeitet und die Ressource erstellt.
2. `http://localhost:8082/geoserver/rest/workspaces/fossgis`: Die (eindeutige) URL unseres neuen Arbeitsbereichs.

Wir können nun überprüfen, ob der Arbeitsbereich tatsächlich angelegt wurde, indem wir

- *klassisch* die GeoServer GUI öffnen und dort über den Menüeintrag `Arbeitsbereiche` alle vorhandenen Arbeitsbereiche auflisten oder
- *über die REST-API* eine Auflistung aller Arbeitsbereiche anfordern (im `XML` -Format):

```
curl \
  -v \
  -u admin:geoserver \
  -XGET \
  -H "Accept: text/xml" \
  http://localhost:8082/geoserver/rest/workspaces
```

In beiden Fällen werden wir erkennen, dass ein neuer Arbeitsbereich namens *fossgis* vorhanden ist.

3.2.2 Datenspeicher erstellen

Nachdem wir einen neuen Arbeitsbereich erstellt haben, werden wir diesem einen neuen Datenspeicher hinzufügen. Der folgende Befehl erzeugt (Operation `POST`!) einen neuen PostGIS Datenspeicher (`dbtype`) mit den Namen *natural_earth* und den folgenden DB-Verbindungsparametern:

- **Hostadresse**: localhost (`host`)
- **Hostport**: 5432 (`port`)
- **Datenbankname**: natural_earth2 (`database`)
- **Benutzer**: user (`user`)
- **Passwort**: user (`passwd`)

```
curl \
  -v \
  -u admin:geoserver \
  -XPOST \
  -H "Content-type: text/xml" \
  -d "<dataStore>
        <name>natural_earth</name>
          <connectionParameters>
            <host>localhost</host>
            <port>5432</port>
            <database>natural_earth2</database>
            <user>user</user>
            <passwd>user</passwd>
            <dbtype>postgis</dbtype>
          </connectionParameters>
      </dataStore>" \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores
```

Bemerkung: Die hier genutzte Datenbank *natural_earth2* ist auf der OSGeo-Live vorinstalliert und entstammt dem Natural Earth Projekt. Weiterführende Informationen zum Datensatz sowie die Daten selbst finden Sie unter http://www.naturalearthdata.com/.

Die erfolgreiche Anlage des Datenspeichers wird uns erneut über den Statuscode `HTTP/1.1 201 Created` bestätigt und kann wiederum über die GUI oder das Auslesen über die API kontrolliert werden:

```
curl \
  -v \
  -u admin:geoserver \
  -XGET \
  -H "Accept: text/xml" \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth.
```

Die Antwort sollte wie folgt aussehen:

```
<dataStore>
  <name>natural_earth</name>
  <type>PostGIS</type>
  <enabled>true</enabled>
  <workspace>
    <name>fossgis</name>
    <atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="alternate" href="http:
  </workspace>
  <connectionParameters>
    <entry key="port">5432</entry>
    <entry key="passwd">crypt1:36+tgAgu2EC0mGyPR7MNkQ==</entry>
    <entry key="dbtype">postgis</entry>
    <entry key="host">localhost</entry>
    <entry key="user">user</entry>
    <entry key="database">natural_earth2</entry>
    <entry key="namespace">http://fossgis</entry>
  </connectionParameters>
  <__default>false</__default>
  <featureTypes>
    <atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="alternate" href="http:
  </featureTypes>
</dataStore>
```

3.2.3 Stil erstellen und hochladen

Nachdem Arbeitsbereich und Datenspeicher angelegt wurden, können wir im nächsten Schritt einen neuen Stil mit dem Namen `states_provinces` im Arbeitsbereich `fossgis` anlegen, der die Stildatei (SLD, *Styled Layer Descriptor*) `states_provinces.sld` assoziiert. Führen Sie dazu zunächst den folgenden Befehl aus:

```
curl \
  -v \
  -u admin:geoserver \
  -XPOST \
  -H "Content-type: text/xml" \
  -d "<style>
        <name>states_provinces</name>
        <filename>states_provinces.sld</filename>
      </style>" \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/styles
```

Erneut wird uns die erfolgreiche Anlage mit dem Status `HTTP/1.1 201 Created` bestätigt und wir prüfen dies wiederum über die GUI

Nachdem die Ressource im GeoServer publiziert wurde, können wir über REST den Stil selbst (*states_provinces.sld*) an die Ressource binden. Hierzu können wir die HTTP-Operation `PUT` nutzen, um eine Datei auf den Server zu kopieren:

```
curl \
  -v \
  -u admin:geoserver \
  -XPUT \
  -H "Content-type: application/vnd.ogc.sld+xml" \
  -d @states_provinces.sld \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/styles/states_provinces
```

Wichtig: Der obige Befehl setzt zwei Dinge voraus:

1. Es existiert eine SLD-Datei mit dem Namen *states_provinces.sld* und einem validen SLD Inhalt. Die entsprechende Datei kann `hier` heruntergeladen werden.
2. Der Pfad zur Datei *states_provinces.sld* ist korrekt. Im obigen Beispiel liegt die Datei im gleichen Ordner aus dem cURL aufgerufen wurde. Wechseln Sie also ggf. im Terminal zum Ordner der Datei oder passen Sie den cURL-Aufruf an.

War der Befehl erfolgreich, enthält die Antwort im Terminal den Teilstring `< HTTP/1.1 200 OK`.

3.2.4 Layer anlegen

Der nächste logische Schritt ist das Anlegen eines Layers auf Basis einer Geometrietabelle aus unserem neuen Datenspeichers *natural_earth*. Als Beispiel werden wir die Tabelle *ne_10m_admin_1_states_provinces_shp* mit dem folgenden Befehl im Arbeitsbereich *fossgis* veröffentlichen:

```
curl \
  -v \
  -u admin:geoserver \
  -XPOST \
  -H "Content-type: text/xml" \
  -d "<featureType>
        <name>states_provinces</name>
        <nativeName>ne_10m_admin_1_states_provinces_shp</nativeName>
        <nativeCRS>EPSG:4326</nativeCRS>
        <enabled>true</enabled>
      </featureType>" \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth/
```

Und erneut begrüßt uns nach erfolgreichem Hinzufügen der Ressource die Statusmeldung `HTTP/1.1 201 Created` und selbstverständlich können wir an dieser Stelle erneut das Ergebnis über die GUI oder die REST-API begutachten:

```
curl \
  -v \
  -u admin:geoserver \
  -XGET \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth/
```

Die XML-Repräsentation des Layers ist demnach wie folgt (gekürzt):

```
<featureType>
  <name>states_provinces</name>
```

```
<nativeName>ne_10m_admin_1_states_provinces_shp</nativeName>
<namespace>
  <name>fossgis</name>
  <atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="alternate" href="http:
</namespace>
<title>ne_10m_admin_1_states_provinces_shp</title>
<keywords>
  <string>ne_10m_admin_1_states_provinces_shp</string>
  <string>features</string>
</keywords>
<nativeCRS>
  GEOGCS["WGS 84",
    DATUM["World Geodetic System 1984",
    SPHEROID["WGS 84", 6378137.0, 298.257223563, AUTHORITY["EPSG","7030"]],
    AUTHORITY["EPSG","6326"]
  ],
  PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]],
  UNIT["degree", 0.017453292519943295],
  AXIS["Geodetic longitude", EAST],
  AXIS["Geodetic latitude", NORTH],
  AUTHORITY["EPSG","4326"]]
</nativeCRS>
<srs>EPSG:4326</srs>
<nativeBoundingBox>
  <minx>-181.800003051758</minx>
  <maxx>181.800018310547</maxx>
  <miny>-60.1882858276367</miny>
  <maxy>84.3496398925781</maxy>
  <crs>EPSG:4326</crs>
</nativeBoundingBox>
<latLonBoundingBox>
  <minx>-181.800003051758</minx>
  <maxx>181.800018310547</maxx>
  <miny>-60.1882858276367</miny>
  <maxy>84.3496398925781</maxy>
  <crs>GEOGCS[&quot;WGS84(DD)&quot;,
DATUM[&quot;WGS84&quot;,
  SPHEROID[&quot;WGS84&quot;, 6378137.0, 298.257223563]],
PRIMEM[&quot;Greenwich&quot;, 0.0],
UNIT[&quot;degree&quot;, 0.017453292519943295],
AXIS[&quot;Geodetic longitude&quot;, EAST],
AXIS[&quot;Geodetic latitude&quot;, NORTH]]</crs>
</latLonBoundingBox>
<projectionPolicy>FORCE_DECLARED</projectionPolicy>
<enabled>true</enabled>
<store class="dataStore">
  <name>natural_earth</name>
  <atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="alternate" href="http:
</store>
<maxFeatures>0</maxFeatures>
<numDecimals>0</numDecimals>
<overridingServiceSRS>false</overridingServiceSRS>
<circularArcPresent>false</circularArcPresent>
<attributes>
  <attribute>
    <name>adm1_code</name>
    <minOccurs>0</minOccurs>
    <maxOccurs>1</maxOccurs>
```

```
      <nillable>true</nillable>
      <binding>java.lang.String</binding>
    </attribute>

    (...)

    <attribute>
      <name>the_geom</name>
      <minOccurs>0</minOccurs>
      <maxOccurs>1</maxOccurs>
      <nillable>true</nillable>
      <binding>com.vividsolutions.jts.geom.MultiPolygon</binding>
    </attribute>
  </attributes>
</featureType>
```

Bereits zu diesem Zeitpunkt können wir den Layer über die Layervorschau des GeoServers oder über einen beliebigen Client (z.B. QGIS) aufrufen. Öffnen Sie hierzu die Weboberfläche vom GeoServer und rufen Sie dort die Layervorschau (Layer-Vorschau) auf. Öffnen Sie dort den Layer in der interaktiven Vorschau durch einen Klick auf OpenLayers (OpenLayers). Die Startansicht der Vorschau sollte dabei in etwa der folgenden Abbildung entsprechen.

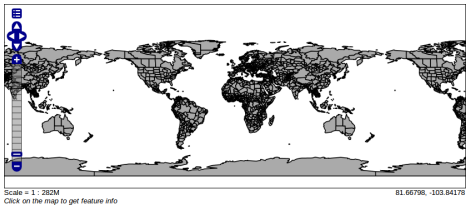


Abbildung 3.2: Hallo Welt!

3.2.5 Layerstil zuordnen

Vergessen wir jedoch nicht unseren Layerstil *states_provinces* aus den vorherigen Kapiteln, den wir im Folgenden dem Layer *states_provinces* zuweisen wollen:

```
curl \
  -v \
  -u admin:geoserver \
  -XPUT \
  -H "Content-type: text/xml" \
  -d "<layer>
        <defaultStyle>
          <name>states_provinces</name>
          <workspace>fossgis</workspace>
        </defaultStyle>
      </layer>" \
  http://localhost:8082/geoserver/rest/layers/fossgis:states_provinces
```

Bestätigt durch den Status `HTTP/1.1 200 OK` können wir die Layervorschau erneut aufrufen und werden sehen, dass der neue Stil (hellgraue Landesflächen, dunkelgraue Landesgrenzen und Beschriftung) dem Layer zugeordnet wurde:

Abbildung 3.3: Layer *states_provinces* mit zugehörigem Stil und zentriert auf Nordrhein- Westfalen

Herzlichen Glückwunsch! Sie haben mit nur wenigen Befehlen im Terminal einen Layer im GeoServer angelegt!

Lernen wir nun im *nächsten Kapitel* (Seite 21) das Editieren von Katalogkonfigurationen.

3.3 Katalogeinträge editieren

Neben dem Hinzufügen von Katalogressourcen ist der häufigste Anwendungsfall in der Administration des GeoServers das Editieren bestehender Ressourcen. Selbstverständlich kann uns auch hier die REST-API in wiederkehrenden Prozeduren zur Seite stehen. Als Beispiel werden wir mit dem folgenden cURL das standardmäßige Ausgabeprojektionssystem des Layers `states_provinces` zu *EPSG:900913* ändern:

```
curl \
  -v \
  -u admin:geoserver \
  -XPUT \
  -H "Content-type: text/xml" \
  -d "<featureType>
        <enabled>true</enabled>
        <srs>EPSG:900913</srs>
        <projectionPolicy>REPROJECT_TO_DECLARED</projectionPolicy>
      </featureType>" \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth/
```

Nachdem dieser Schritt mit `HTTP/1.1 200 OK` bestätigt wurde, können wir anschließend automatisch über REST die neue native Bounding Box des Layers mit dem Parameter *recalculate=nativebbox,latlonbbox* berechnen lassen:

```
curl \
  -v \
  -u admin:geoserver \
  -XPUT \
  -H "Content-type: text/xml" \
  -d "<featureType>
        <enabled>true</enabled>
      </featureType>" \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth/
```

Betrachten wir nun den Layer in der Layerübersicht des GeoServers (Layer) sehen wir, dass der Layer - wie zu erwarten - mit dem angegebenen Koordinatensystem *EPSG:900913* und einer Bounding Box im entsprechenden Koordinatensystem konfiguriert ist:

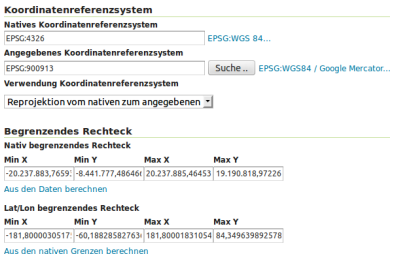


Abbildung 3.4: Ausschnitt der Layerkonfiguration *states_provinces* im nativen Koordinatensystem EPSG:900913

Das Ergebnis unserer Änderung können wir uns abschließend über die GeoServer Layervorschau ( Layer-Vorschau) anschaulich illustrieren lassen:

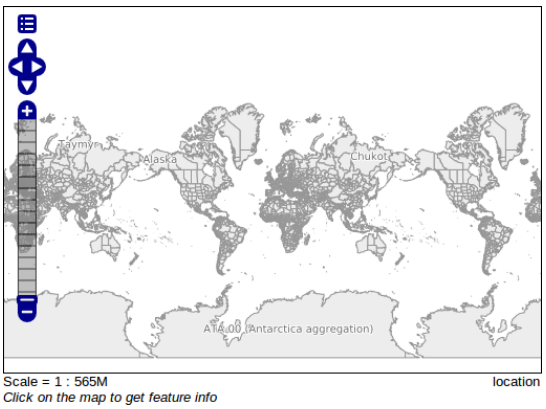


Abbildung 3.5: Layer *states_provinces* mit zugehörigem Stil in EPSG:900913

Wichtiger Hinweis: Das REST-Paradigma beschreibt, dass Repräsentationen (hier in den Formaten `XML` und `JSON`) sowohl gelesen als auch geschrieben werden können. Dies bedeutet, dass jede XML-Antwort (siehe z.B. *Layer anlegen* (Seite 18)) in der obigen Form abgeändert (z.B. mit neuem Layernamen oder als deaktiviert markiert) und direkt über die REST-API an den Server gesendet werden kann.

Nachdem wir bestehende Ressourcen editiert haben, wird das *letzte Kapitel* (Seite 22) dieses Moduls erläutern, wie Sie Ressourcen entfernen können.

3.4 Katalogeinträge entfernen

Um eine Ressource über die REST-API zu entfernen, können wir die Operation `DELETE` nutzen. Das folgende Beispiel zeigt den (zweischrittigen) cURL Aufruf zum Entfernen eines Layers (**Hinweis**: Durch den Befehl wird der oben erzeugte Layer *states_provinces* aus dem Arbeitsbereich *fossgis* gelöscht!):

Zunächst entfernen wir den Layer:

```
curl \
  -v \
  -u admin:geoserver \
  -XDELETE \
  http://localhost:8082/geoserver/rest/layers/fossgis:states_provinces
```

Und anschließend den FeatureType des Datenspeichers:

```
curl \
  -v \
  -u admin:geoserver \
  -XDELETE \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth/
```

In dem obigen Beispiel wird der FeatureType *states_provinces* durch den gleichnamigen Layer referenziert, wodurch die zweischrittige Lösung notwendig ist. Das alleinige Ausführen des zweiten Befehls würde in diesem Fall die Fehlermeldung *feature type referenced by layer(s)* provozieren. Um automatisch alle referenzierten Objekte (z.B. auch Gruppenlayer) zu entfernen, sollte der Parameter *recurse=true* gesetzt werden:

```
curl \
  -v \
  -u admin:geoserver \
  -XDELETE \
  http://localhost:8082/geoserver/rest/workspaces/fossgis/datastores/natural_earth/
```

Wurden die Ressourcen erfolgreich entfernt, wird der Status `HTTP/1.1 200 OK` ausgegeben.

Bemerkung: Möchten Sie sich - auf Grundlage der Beispiele dieses Workshops oder Ihrer eigenen Daten - weiter mit der REST-API beschäftigen, finden Sie in der offiziellen Dokumentation viele Hinweise und gute Beispiele:

http://docs.geoserver.org/stable/en/user/rest/

Sie haben das Modul *REST-Schnittstelle* (Seite 11) erfolgreich beendet, Sie können nun mit Modul *Tipps, Tricks & Troubleshooting* (Seite 24) fortfahren.

KAPITEL 4

Tipps, Tricks & Troubleshooting

Im letzten Modul des Workshops werden wir Hinweise liefern, die insbesondere beim Produktivbetrieb eines GeoServers hilfreich sein können. Dazu gehen wir zunächst auf die *Protokollierung* des GeoServers und die Möglichkeiten des integrierten *GeoWebCaches (GWC)* ein. Anschließend wird gezeigt wie sich das GeoServer-Datenverzeichnis auslagern lässt. Den Abschluß bilden die eher *informativen* Abschnitte mit Hinweisen auf hilfreiche Optionen in der GeoServer-Weboberfläche und Tuning-Möglichkeiten der *Java Virtual Machine (JVM)*.

4.1 Protokollierung

Bei jeglichen Fehlern, die sich auf den GeoServer zurückführen lassen (wie z.B. keine oder falsche Antwort eines Kartendienstes) ist das Protokoll die erste Anlaufstelle. Das GeoServer Protokoll lässt sich dabei entweder direkt über die GUI oder aus dem Dateisystem (`/usr/local/lib/geoserver-2.6.1/data_dir/logs`) aufrufen. Rein informativen Protokollmeldungen steht dabei das Kürzel `INFO` vor. Bei schwerwiegenden Fehlern findet sich dort jedoch das Kürzel `ERROR`.

Für die Protokollierung des GeoServers lassen sich verschiedene Profile einstellen. Diese unterscheiden sich in der Sensitivität, in der die Prozesse des GeoServers protokolliert werden. Das zu verwendende Protokoll kann über die GeoServer-Weboberfläche im Bereich `Einstellungen -> Global` konfiguriert werden. Der Wechsel eines Profils wirkt sich sofort aus, d.h. der GeoServer muss *nicht* neu gestartet werden!

Abbildung 4.1: Protokollierung in der GeoServer-Weboberfläche

Ist die Checkbox bei *Ausführliche Fehlerausgaben* gesetzt, wird der volle Java-Stacktrace in die Log-Datei geschrieben. Da hierdurch größere Log-Dateien verursacht werden, ist diese Einstellung nur für das Debuggen zu empfehlen.

Hier eine kurze Erläuterung einiger Protokoll-Profile:

- `DEFAULT_LOGGING`: Mittleres Protokolllevel auf fast allen Modulebenen des GeoServers.

- `GEOSERVER_DEVELOPER_LOGGING`: Ausführliche Protokollierung auf Ebene des Moduls GeoServer. Nur sinnvoll, wenn der GeoServer debuggt wird.
- `GEOTOOLS_DEVELOPER_LOGGING`: Ausführliche Protokollierung auf Ebene des Moduls Geo-Tools. Diese Auswahl kann nützlich sein, wenn überprüft werden soll, welche SQL Statements (z.B. bei einer GetFeature Abfrage) an die Datenbank gesendet werden.
- `PRODUCTION_LOGGING`: Minimale Protokollierung, nur Fehler werden ausgegeben. Diese Einstellung ist für den Produktiveinsatz zu wählen.
- `VERBOSE_LOGGING`: Ausführliche Protokollierung auf allen Ebenen des GeoServes. Nur sinnvoll, wenn der GeoServer debuggt wird.

Wir wollen das Protokoll des GeoServers nun *live* über das Dateisystem beobachten. Dazu führen müssen die folgenden Schritte ausgeführt werden.

1. Stellen Sie das `VERBOSE_LOGGING`-Profil ein.
2. Der GeoServer auf der OSGeoLive ist so konfiguriert, dass nicht automatisch in die Protokolldatei, sondern in die Standardausgabe des Java-Prozesses geschrieben wird. Wir müssen also den **Haken bei** *in die Standardausgabe schreiben* **entfernen**, sodass das Protokoll in die Datei `/usr/local/lib/geoserver-2.6.1/data_dir/logs/geoserver.log` geschrieben wird.
3. Speichern Sie die Einstellungen (unten).
4. Öffnen Sie die Konsole und führen Sie den folgenden Befehl aus:

   ```
   less +F /usr/local/lib/geoserver-2.6.1/data_dir/logs/geoserver.log
   ```

Bemerkung: Normalerweise würde man statt `less +F` den Befehl `tail -f` verwenden. Dies funktioniert auf der OSGeoLive aber aus unbekannten Gründen nicht. Die *Live*-Beobachtung der Datei kann durch das Drücken der Tastenkombination `STRG + C` beendet werden.

---

5. Öffnen Sie eine Layer-Vorschau und beobachten Sie wie sich das Protokoll verändert.

Die folgende Abbildung stellt diese Schritte dar:

Abbildung 4.2: Protokolleinstellungen und Layer-Vorschau

Sie können nun *live* beobachten wie sich der Inhalt der Logdatei verändert. Dabei sehen Sie immer das Ende der Protokolldatei. Jede Interaktion in der Vorschau des Layers (z.B. Zoomen oder Klicken) kann nun im Protokoll nachvollzogen werden. Sie können das Protokollprofil auch ändern, um zu beobachten wie sich die Sensitivität der Ausgabe verändert.

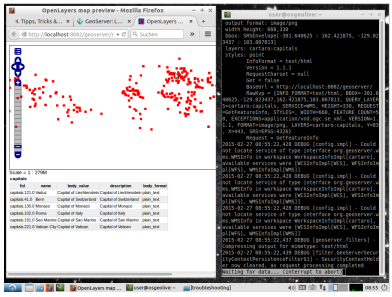


Abbildung 4.3: Live-Beobachtung der Protokollierung

Im *folgenden Abschnitt* (Seite 26) geht es mit dem Thema *GeoWebCache (GWC)* weiter.

4.2 Layer cachen mit GWC

Die häufigste Anforderung an einen GeoServer ist das Bereitstellen einer OGC-konformen WMS Schnittstelle und damit das Ausgeben von Kartenmaterial im Rasterformat. Aus diesem Grund kann das Cachen dieser Anfragen einen entschiedenen Einfluss auf die Performance des Servers haben und sollte nach Möglichkeit auf jedem (produktiven) System vorgenommen werden. Für das Cachen von Kartenkacheln existieren eine Vielzahl guter OpenSource Caching-Engines, wir werden an dieser Stelle jedoch den standardmäßig im GeoServer integrierten GeoWebCache (GWC) nutzen, der als Proxy zwischen Client und GeoServer fungiert (siehe Abbildung).

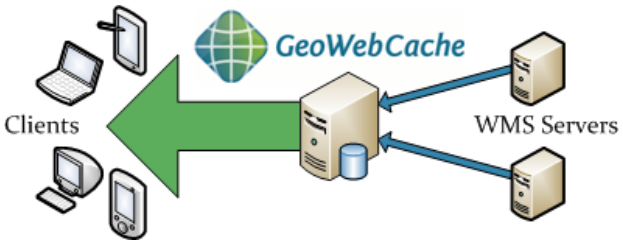


Abbildung 4.4: Funktionsübersicht des GWC als Proxy, http://geowebcache.org/docs/current/_images/how

Prinzipiell bietet der GWC zwei Methoden zum Anlegen der Kartenkacheln:

1. **On-The-Fly-Prozessierung**: Wird ein GWC-Layer erstmalig von einem Client angefordert, werden die entsprechenden Kartenkacheln einmalig live gerendert und anschließend in einem GWC-Datenverzeichnis abgelegt. Der nächste Aufruf des Layers im gleichen Kartenausschnitt erhält nun eine (deutlich schnellere) Antwort aus dem Cache.

2. **Vorrechnen von Kartenkacheln**: Die Kacheln eines Layers werden in einer definierten Bounding Box und in definierten Zoomstufen entlang eines Gridsets vorberechnet und abgelegt. Im Gegensatz zur On-The-Fly Berechnung benötigt diese Methode je nach verfügbarer Ressourcen eine deutliche höhere Rechenzeit, jedoch erhalten alle Clients eine direkte Antwort aus dem Cache, sodass die gefühlte Performance für den Enduser höher ist.

Wir werden im Folgenden nicht alle Konfigurationsmöglichkeiten des GWC kennenlernen können und aus diesem Grund werden wir uns auf die grundlegende Konfiguration am Bei-

spiel der On-The-Fly-Prozessierung beschränken und beispielhaft einen Cache für den Layer *topp:states* vorbereiten.

Auf der OSGeoLive werden die vorgerechneten Kartenkacheln im Verzeichnis `/usr/local/lib/geoserver-2.6.1/data_dir/gwc/` abgelegt. Navigieren Sie zunächst im Terminal () zu diesem Verzeichnis und lassen Sie sich den Inhalt mit dem Befehl `ls -lh` (oder einer vergleichbaren Operation) anzeigen. Die Ausgabe sollte dabei in etwa wie folgt aussehen und aktuell nur die globale Konfigurationsdatei *geowebcache.xml* für den GWC sowie ein leeres temporäres Verzeichnis enthalten.

```
-rw-rw-r-- 1 root users 4,8K Jan 14 19:58 geowebcache.xml
drwxrwxr-x 2 root users    3 Jan 14 19:58 tmp/
```

Bemerkung: Das Verzeichnis `gwc/` wird per default auf root-Ebene im GeoServer Datenverzeichnis angelegt. Soll dieses geändert werden, sollte der folgende Eintrag in die web.xml des GeoServers eingetragen werden (Dabei unbedingt die Berechtigungen des neuen Ordners beachten!):

```
<context-param>
  <param-name>GEOWEBCACHE_CACHE_DIR</param-name>
  <param-value>/dir/to/gwc_cache/</param-value>
</context-param>
```

Wie wir aus dem obigen Ordnerliste entnehmen können, existiert zum aktuellen Zeitpunkt noch kein Verzeichnis mit vorberechneten Kartenkacheln, weshalb wir im Folgenden die notwendigen Schritte zum Anlegen eines GWC-Layers vollziehen werden:

1. Öffnen Sie in der GeoServer GUI unter 

die Konfiguration für den Layer *states* und wählen Sie dort den Reiter *Kartenkachel-Cache*.
2. Um einen gecachten Layer zu erzeugen, muss die erste Checkbox *Erzeuge einen gecachten Layer für diesen Layer* aktiviert sein (Für den ausgewählten Layer bereits eingestellt).
3. Überprüfen Sie nun die weiteren Einstellungen und passen Sie diese ggf. an Ihre Bedürfnisse an:
   - **Metatile-Faktor**: Metatiles sind größere Kartenkacheln (Tiles), aus denen die zu speichernden Kacheln herausgeschnitten werden. Der Faktor gibt hierbei die Größe der Metatiles an, ein Faktor von 3x3 bedeutet, dass die Bildbreite der Zielkachel um den Faktor drei erhöht wird und sich damit für eine Kachelgröße von 256px eine Metatile-Kachelgröße von 768px ergibt (siehe Abbildung).

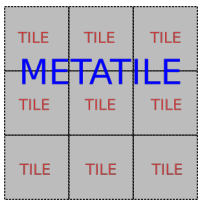


Abbildung 4.5: Mit Metatiles wird die Zielkachel (TILE) aus einer größeren Kachel (METATILE) herausgeschnitten, http://geowebcache.org/docs/current/_images/metatile.png

Metatiles werden in erster Linie benötigt, um doppelte Kartenbeschriftungen (z.B. von Straßenlayern) in zwei aneinanderliegenden Kacheln zu vermeiden.

- **Kachel-Umrandung**: Zusätzlicher Rahmen (in px), der um eine Kachel angefordert werden soll. Nur sinnvoll, wenn in Verbindung mit der Vernwendung von Metatiles Problemen bei der Darstellung von Labels und/oder Features am Kachelrand auftreten.
- **Bildformat für Kacheln**: Das Standard Bildformat für die Kacheln. Dieses Format sollte unbedingt in Abhängigkeit Ihrer Clients bzw. des dort definierten Anfrage-Bildformats für die gecachten Layer gewählt werden, andernfalls kann der Cache nicht genutzt werden!
- **STYLES**: Existieren für den Layer mehrere Stile, die im Client gewechselt werden können und somit gecacht werden sollten, müssen diese hier ausgewählt werden. In aller Regel wird es jedoch genügen, ausschließlich den Standard-Layerstil (*LAYER DEFAULT*) als Wert zu setzen.
- **Verfügbare Rastergittersätze**: Das Gitternetz definiert das Netz, über das die gespeicherten Kacheln abgefragt werden und definiert damit die räumlichen Index der Einzelkacheln. Die Einzelkachel im rechteckigen Grid ist dabei über ein *x,y,z* Koordinatentripel identifizierbar (siehe Abbildung). Die *x,y* Koordinaten bestimmen die horizontale/vertikale Position, die *z*-Koordinate das Zoomlevel. Ein Grid ist dabei stets für genau ein Koordinatensystem gültig. Wählen Sie hier die Projektion bzw. das Grid, in dem der Layer gecacht werden soll.

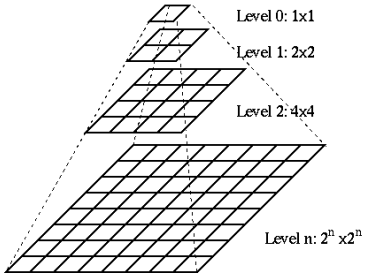
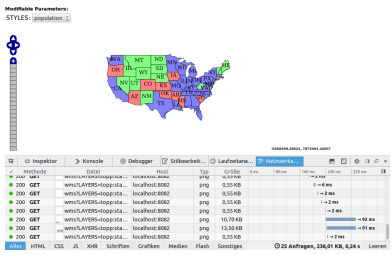


Abbildung 4.6: Gridset eines Tile-Layers, http://3.bp.blogspot.com/_0_xIiXP5xuY/S5pEpCjenaI/AAAAAAAAAAA/h/Image_Pyramid.gif

Hinweis: Die Standardeinstellungen für einen neuen gecachten Layer können Sie unter dem Menüeintrag  Caching Standards anpassen.

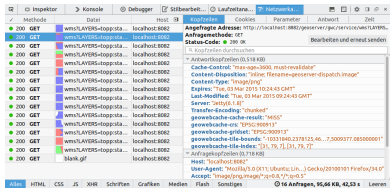
4. Im nächsten Schritt werden wir prüfen, ob der Layer korrekt gecacht wird und die *HTTP-Response Header* eines GWC-Layers analysieren. Öffnen Sie hierzu die GWC Layervorschau unter  Gecachte Layer und wählen Sie dort unter dem Layereintrag *topp:states* in der Combobox *Vorschau* den Wert *EPSG:900913/png*. Nachdem die Vorschau in einem neuen Tab/Fenster geöffnet wurde, öffnen wir über `F12` die Entwicklerkonsole des Browsers und navigieren in dieser zum Reiter *Netzwerkanalyse* (siehe Abbildung). **Wichtig:** Nach Aktivieren der Konsole bzw. der Netzwerkanalyse muss die aktuelle Seite ggf. neu geladen werden!



5. Leeren Sie die Netzwerk-Übersicht (

unten rechts in der Entwicklerkonsole) und zoomen Sie (einmalig!) zu einem beliebigen Kartenausschnitt.

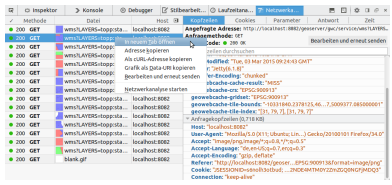
6. Im Reiter *Netzwerkanalyse* erscheinen alle Requests des Kartenclients an den GeoServer. Wählen Sie aus dieser Liste einen beliebigen WMS-Request aus und öffnen Sie im rechten Bereich den Reiter *Kopfzeilen* (siehe Abbildung):



   Die Ansicht zeigt neben den *Anfragekopfzeilen* des Clients auch die *Antwortkopfzeilen* des Servers. Für uns sind die Folgenden Headerinformationen von Relevanz:

   - `geowebcache-cache-result`: Wurde die Kachel aus dem Cache ausgeliefert, wird der Wert *HIT*, andernfalls *MISS* ausgegeben. Das obige Beispiel sollte *MISS* ausgeben, da zum aktuellen Zeitpunkt kein Cache vorhanden ist.
   - `geowebcache-crs`: Das Koordinatensystem der Kachel.
   - `geowebcache-gridset`: Der Name des zu Grunde liegenden Gridsets.
   - `geowebcache-tile-bounds`: Die BoundingBox der Kachel.
   - `geowebcache-tile-index`: Der Index der Kachel (X,Y,Z) im Gridset.

7. Nachdem die obige Beispiel-Kachel erstmalig angefordert wurde, wird der GWC diese Kachel im Cache ablegen und künftige Requests werden die Antwort aus selbigem erhalten. Um dies zu überprüfen werden wir nun die o.g. Kachel neu laden, indem wir in der Entwicklerkonsole das Kontextmenü aufrufen und *In neuem Tab öffnen* auswählen (siehe Abbildung).



8. Öffnen Sie in dem neuen Fenster/Tab erneut die Entwicklerkonsole über `F12` und aktivieren Sie den Reiter *Netzwerkanalyse*. Betrachten Sie nun den Response-Header `geowebcache-cache-result`. Was fällt Ihnen auf?

9. Navigieren Sie abschließend - falls nicht mehr geöffnet - im Terminal zum GWC-Verzeichnis (`/usr/local/lib/geoserver-2.6.1/data_dir/gwc/`) und prüfen

Sie dort den (neuen) Ordner `topp_states`, in dem der GWC die Kacheln aus selbigem Layer abgelegt hat. Die obige Kachel lässt sich dabei anhand der Header-Informationen `geowebcache-gridset` und `geowebcache-tile-index` eindeutig identifizieren:

```
topp_states/ (Layername)
  |
  +-- EPSG_900913_07/ (Gridset + Zoomstufe)
        |
        +-- 01_04/ (inter berechnete Notation auf Basis von Gridset + Zoomstufe)
              |
              +-- 0031_0079.png (Kachelindex)
```

Sie haben in diesem Kapitel erfolgreich die Basiskonfiguration für einen GWC-Layer vorgenommen und Methoden kennengelernt, selbige Layer über die Entwicklerkonsole im Browser zu analysieren. Fahren Sie nun mit dem *nächsten Kapitel* (Seite 31) (Datenverzeichnis auslagern) fort.

Zusatzinformation: Vorrechnen der Kartenkacheln

In der Praxis wird es in aller Regel unter gegebenen Ressourcen notwendig sein, den Layercache für häufig angeforderte Layer im Voraus zu berechnen. Die folgende Liste führt die notwendingen Schritte am Beispiel des Layers *topp:states* in Kurzform auf. Weiterführende Informationen finden Sie in der GWC Dokumentation.

1. Öffnen Sie die die GWC Administrationsoberfläche über http://localhost:8082/geoserver/gwc und wählen Sie unter *A list of all the layers and automatic demos* den Eintrag *Seed this Layer* unterhalb von Layer *topp:states* aus. In diesem Dialog kann über die oberen beiden Auswahlboxen geprüft werden, ob für den aktuell ausgewählten Layer bereits ein Task läuft oder geplant ist. Falls gewünscht, kann dieser abgebrochen werden.
2. Um einen neuen Task zu starten, sind folgende Einstellungen notwendig:

- **Number of tasks to use:** Anzahl der Threads für diesen Seed. Um den Server (und damit die Reaktionsgeschwindigkeit des GeoServers) in einer produktiven Umgebung nicht zu sehr mit dem Rechenvorgang zu belasten, sollte hier der minimale Wert (*01*) gewählt werden. Ist der GeoServer (noch) nicht produktiv kann ein höherer Wert gewählt werden.
- **Type of operation:** Auswahl der Seed-Operation. *Reseed* generiert alle Kacheln neu, *Seed* nur die fehlenden und *Truncate* löscht alle existierenden Kacheln. Für die obigen Layer empfiehlt sich die Operation *Reseed*.
- **Grid Set:** Auswahl des Projektionssystems bzw. des Gridsets.
- **Format:** Auswahl des Bildformats.
- **Zoom start:** Auswahl der kleinsten Zoomstufe (kleiner Maßstab).
- **Zoom stop:** Auswahl der höchsten Zoomstufe (großer Maßstab). Eine höhere Zahl repräsentiert eine detailliertere Kartenansicht. Je höher der Wert, desto höher auch der Rechenaufwand und der belegte Speicher!
- **Bounding box:** Optionaler Parameter zur Angabe des BoundingBox. Falls keine Werte angegeben werden, werden die Angaben des Layers selbst genutzt. Diese sollten in aller Regel korrekt sein, sodass hier keine Angabe erforderlich ist. Ausnahme: Nur ein begrenztes Gebiet soll gecacht werden.

3. Nachdem alle Einstellungen vorgenommen wurden, wird der (Re-)Seed über den Button "Submit" gestartet.

Bemerkung: Die obigen Schritte zum Anlegen eines Layercache sind auch über eine REST-API möglich. Informationen und gute Beispiele zu dieser API finden Sie unter http://docs.geoserver.org/stable/en/user/geowebcache/rest/index.html

---

4.3 GeoServer-Datenverzeichnis auslagern

Es wird empfohlen das GeoServer-Datenverzeichnis (siehe *hier* (Seite 6)) auszulagern. Der entscheidende Vorteil besteht darin, dass sich etwa ein GeoServer im Produktivbetrieb auf diese Weise elegant zu einer aktuelleren Version updaten lässt, ohne dass die Daten/Konfigurationen des GeoServers verloren gehen bzw. separat gesichert werden müssen.

Um das Datenverzeichnis auszulagern, führen Sie die folgenden Schritte durch:

1. Wir kopieren das bestehende Datenverzeichnis aus der GeoServer-Installation (`data_dir/`) in ein neues Verzeichnis (`gs_data_dir/`) in unserem Home-Verzeichnis, welches anschließend als Datenverzeichnis des GeoServers verwendet werden soll:

   ```
   cp -r /usr/local/lib/geoserver-2.6.1/data_dir/ /home/user/gs_data_dir
   ```

2. Das Datenverzeichnis des GeoServers wird über die Umgebungsvariable (`GEOSERVER_DATA_DIR`) gesteuert. Bei einer klassischen WAR-Installation, etwa auf einem Tomcat, kann dieser Wert in der Datei (`web.xml`) gesetzt werden. Im Falle der OSGeoLive müssen wir diese Variabel jedoch im Startup-Skript des GeoServers setzen. Führen Sie bitte den folgenden Befehl aus, um das Skript `startup.sh` mit dem Texteditor *medit* und den benötigten root-Rechten zu öffnen:

   ```
   sudo medit /usr/local/lib/geoserver-2.6.1/bin/startup.sh
   ```

4. Fügen Sie an den Anfang der Datei folgende Zuweisung ein, um das in Schritt 1 erstellte Verzeichnis als GeoServer-Datenverzeichnis zu verwenden (und speichern Sie anschließend die Datei):

   ```
   GEOSERVER_DATA_DIR=/home/user/gs_data_dir
   ```

5. Starten Sie den GeoServer neu und beobachten Sie anschließend im Abschnitt *Serverstatus*, wie sich der Pfad des Datenverzeichnisses verändert hat.

Bemerkung: Analog zur obigen Konfiguration kann auch das Verzeichnis für den GeoWebCache (GWC) gesteuert werden. In diesem Fall muss die Variable `GEOWEBCACHE_CACHE_DIR` gesetzt werden.

Der *folgende Abschnitt* (Seite 32) liefert wertvolle Hinweise zur Problemlösung und Performanceoptimierung für den GeoServer im Produktivbetrieb.

4.4 Einstellungen in der GeoServer GUI

Es gibt zahlreiche Möglichkeiten den GeoServer über die Weboberfläche zu konfigurieren. Wir wollen in diesem Abschnitt kurz auf hilfreiche Optionen hinweisen:

- **Vektordaten aus Datenbanken**: Bei Datenbank-Quellen, die sehr viele (z.B. mehrere Millionen) Features enthalten, können die Verbindungsparameter (für den Datenbankzugriff) einen entscheidenden Einfluss auf die Performance des GeoServers haben. Details zu den Parametern finden Sie hier.

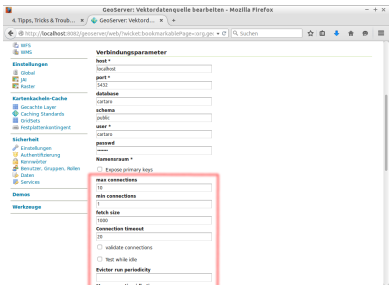


Abbildung 4.7: Verbindungs-Einstellungen (PostGIS) bei Vektordatenspeicher

- Der GeoServer verwendet die **Java Advanced Imaging (JAI)**-Bibliothek zur Bildmanipulation (z.B. zum Erzeugen der Kacheln, die ausgeliefert werden). Im Bereich *JAI* der Weboberfläche können verschiedene Parameter eingestellt werden, die hier dokumentiert sind.
- Wenn Sie über den GeoServer **sehr viele** verschiedene (Vektor-)Layer bereitstellen, sollte der Wert **Größe des Feature Type Caches** mindestens so groß sein wie die Anzahl der bereitgestellten Feature Types. Ansonsten kann es zwischenzeitlich zu spürbar längeren Antwortzeiten bei WFS-Anfragen kommen. Der Wert kann in der Weboberfläche im Bereich *Einstellungen -> Global* konfiguriert werden.
- Sollten Sie einen öffentlich zugänglichen GeoServer verwalten, ist es dringend zu empfehlen, dass sie **WFS-T(ransaktionen) ausstellen**, falls diese schreibende Funktionalität nicht erlaubt ist bzw. benötigt wird. Dazu müssen Sie den Radio-Button im Bereich *Dienste -> WFS -> Dienstgüte* von "Vollständig" auf "Basis" wechseln. Anderenfalls wäre es denkbar, dass Ihre Daten unerwünscht per WFS-T von Dritten manipuliert werden.
- Sehr hilfreich kann das Anlegen von **SQL-Views** (auf Ebene des Geoservers und nicht der Datenbank(!)) sein. Diese Funktionalität erreichen Sie, wenn Sie im Bereich *Daten -> Layer -> Layer hinzufügen* eine Vektor-Datenbankquelle auswählen und dort *SQL View konfigurieren* klicken. Nützlich ist insbesondere die Möglichkeit die SQL-Views zu parametrisieren, etwa um bestimmte Features durch entsprechende Parameter herauszufilter. Ausführliche Informationen dazu finden Sie hier.

Der *letzte Abschnitt* (Seite 32) zeigt wie Sie die *Java Virtual Machine (JVM)* für den GeoServer tunen können.

4.5 Java Virtual Machine (JVM) tunen

Der GeoServer läuft, wie bereits in Kapitel *Basiswissen GeoServer* (Seite 5), innerhalb eines JAVA Servlet Containers wie z.B. *Apache Tomcat* oder *Jetty*. Ein JAVA Container kann dabei mit

gezielten Startup-Parametern gestartet werden, die das Laufzeitverhalten der in ihm deployten Servlets massiv beeinflussen können. Die folgende Tabelle führt einige Parameter auf, die in produktiven GeoServer Instanzen bedacht werden sollten. Die Inhalte stellen dabei keinen Anspruch an Vollständigkeit und bedürfen einer kritischen Überprüfung für jede Installation (u.a. in Abhängigkeit der verfügbaren Hardwareressourcen, erwartbaren Zugriffszahlen, Einsatzzweck des GeoServers). Im Tomcat werden die Parameter an die JAVA_OPTS (global) oder CATALINA_OPTS (Container) übergeben.

Es wird grundsätzlich empfohlen das **Oracle (Sun) JDK statt OpenJDK** zu verwenden, da ersteres die Performance des GeoServers erhöht. Details dazu gibt es z.B. hier.

Tabelle 4.1: Container Parameter

Parameter	Beschreibung	Beispiel
-server	Für Server optimierte JVM.	
-Xms	Steuert die Anfangsgröße des Java-Heap-Speichers. Für Produktivsysteme werden Werte im Bereich von 2 bis 4 GB empfohlen.	-Xms2g
-Xmx	Steuert die maximale Größe des Java-Heap-Speichers. d.h. dieser Wert darf nicht kleiner sein als `-Xms`. Es gibt unterschiedliche Meinungen darüber, ob für `-Xms` und `-Xmx` dieselben Werte, d.h. `Xms=Xmx` verwendet werden sollen oder nicht. Wir verwenden auf unseren Systemen `Xms=Xmx` und haben damit keine schlechten Erfahrungen gemacht!	-Xmx2g
-XX:PermSize	Definiert die anfänliche Größe des Speichers, der für die permanente Objektgenerierung reserviert ist.	-XX:PermSize=256m
-XX:MaxPermSize	Definiert die maximale Größe des Speichers, der für die permanente Objektgenerierung reserviert ist. Ein Wert von maximal 256 MB ist hier absolut ausreichend.	-XX:MaxPermSize=256m
-Djavax.servlet.request.encoding	Zeichenkodierung eingehender Anfragen (Standard: ISO 8559-1)	-Djavax.servlet.request.encoding=UTF-8
-Djavax.servlet.response.encoding	Zeichenkodierung ausgehender Antworten (Standard: ISO 8559-1)	-Djavax.servlet.response.encoding=UTF-8
-Dfile.encoding	Zeichenkodierung beim Umgang mit statischen Dateien (Standard: Default des Betriebssystems, daher ggf. wichtig bei Windows-Systemen)	-Dfile.encoding=UTF-8
-XX:+UseParallelGC	Garbage Collection für Mehrkern-Systeme (siehe hier)	
-XX:+UseParallelOldGC	s.o.	

Der Workshop ist an dieser Stelle beendet. Wir hoffen, dass Sie interessante Dinge über den GeoServer gelernt haben und bedanken uns für Ihre Aufmerksamkeit.

KAPITEL 5

Autoren

Daniel Koch: koch [at] terrestris [dot] de

Nils Bühner: buehner [at] terrestris [dot] de