Hochschule für Technik und Wirtschaft Berlin
- Campus Wilhelminenhof -

Belegarbeit

**im Studiengang Angewandte Informatik -
Schwerpunkt Ortsbasierte Informationssysteme**

Thema:	Neues Dateiformat zur Vereinfachung von SLD-Dateien
Autor:	Marcel Ebert MatNr. S0558606
Version vom:	24. September 2018
Betreuer:	Prof. Dr. Thomas Schwotzer

Inhaltsverzeichnis

Abbildungsverzeichnis **2**

Tabellenverzeichnis **2**

Listingverzeichnis **2**

Abkürzungsverzeichnis **3**

1 Aufgabe der Komponente **3**

2 Architektur **3**
2.1 Struktur von SLD-Light 3
2.2 Aufbau des Editors 4
2.3 Struktur der Objekte 5
2.4 Dateiupload 6
2.5 Genutzte Komponenten 7

3 Nutzung **7**
3.1 Code 7
3.2 Deployment / Runtime 8

4 Qualitätssicherung **8**

5 Vorschläge / Ausblick **8**

Literaturverzeichnis **10**

Anhang **11**

Abbildungsverzeichnis

1 UML Diagramm: SLD-Objekte 6
2 Dateiupload-Fenster 7
3 Ansicht des Editors 11

Tabellenverzeichnis

1 Zoomstufen 13

Listingverzeichnis

1 Die Klasse 'SSHConstants.java' 11
2 Beispiel für SLD-Light 11
3 Beispiel für SLD 11
4 Beispiel XML-Datei für Geoserver 13

1 Aufgabe der Komponente

Im Rahmen dieses Projektes soll ein neues Dateiformat entstehen, welches Styled-Layer-Descriptor(SLD)-Dateien verkürzt abbilden kann. Das SLD-Dateiformat wird von dem im OHDM-Projekt eingesetzten Geoserver benutzt, um Stile für verschiedene Objekte bzw. Ebenen (sogenannte Layer) zu definieren. Die Definition dieser Stile erfolgt innerhalb der SLD-Dateien über XML. Da SLD eine Vielzahl an Möglichkeiten bietet, diese aber häufig nicht vollständig benötigt werden, ist der Anteil an überflüssigem Code hoch. Das in diesem Projekt entstehende Dateiformat „SLD-Light“ hat die Aufgabe, den Anteil an überflüssigen Informationen zu minimieren, aber dennoch die am häufigsten benutzten Funktionalitäten zur Verfügung zu stellen.

Weiterhin wird ein Editor entwickelt, welcher dem Benutzer Möglichkeiten bietet SLD-Light-Dateien zu erzeugen, ohne dabei die Eigenschaften des Dateiformats kennen zu müssen. Das bedeutet der Editor bietet eine Benutzer-Schnittstelle mit welcher eine SLD-Light-Datei mit allen zur Verfügung stehenden Eigenschaften definiert, erzeugt und gespeichert werden kann. Außerdem bietet er Funktionalitäten um die erzeugte SLD-Light-Datei in eine SLD-Datei zu übersetzen, welche dann letztendlich vom Geoserver benutzt werden kann. Der Editor ermöglicht es, SLD-Dateien auf den Geoserver hochzuladen. Gespeicherte SLD-Light-Dateien können vom Editor wieder geöffnet werden, wodurch die Benutzeroberfläche wiederhergestellt wird, um eine einfache Veränderung der Datei zu ermöglichen.

2 Architektur

2.1 Struktur von SLD-Light

Im Folgenden soll der Aufbau des neuen Dateiformats erläutert werden. Ein Beispiel ist in Listing 2 zu sehen. Die SLD-spezifischen Begriffe werden nicht im Detail erläutert[1].

In der ersten Zeile der Datei steht der Name der „NamedLayer“. Darauf folgt die Angabe der Zoomstufen. Es müssen zwei Zahlen im Bereich von 0 bis 16 mit einem „-“ voneinander getrennt angegeben werden. Die erste Zahl steht für die Mindestzoomstufe und die zweite Zahl für die Maximalzoomstufe, welche für die Regel gelten sollen. Die Stufen von 0 bis 16 werden vom Editor beim Übersetzen zu entsprechenden Werten übertragen, die für SLD geeignet sind. Die Stufeneinteilung kann Tabelle 1 entnommen werden. Bei Angabe einer „0“ wird die Angabe der Zoomstufe in der SLD-Datei weggelassen.

Da eine Regel immer nur einen Zoom haben kann, folgt aus der Angabe der Zoomstufen, dass eine neue Regel begonnen wird. Aufgrund der niederen Wichtigkeit der Regelnamen gibt es keine Möglichkeit diesen in SLD-Light Dateien anzugeben. Der

[1] Erklärungen dafür können [PLPV18] entnommen werden

Regelname wird automatisch vergeben und setzt sich aus dem „NamedLayer“-Namen gefolgt „_rule_“ und einem Index beginnend bei 0 zusammen.

Nach dem Zoom folgt eine optionale Angabe eines Filternamens. Dieser kann wichtig sein, wenn die Regel sich nur auf bestimmte Elemente der Karte beziehen soll.

Als Nächstes werden die sogenannten „Symbolizer“ beschrieben. Diese bestimmen, wie die Objekte auf der Karte gezeichnet werden. Es gibt vier verschiedene Arten: PointSymbolizer, LineSymbolizer, PolygonSymbolizer und TextSymbolizer. Die Beschreibung im SLD-Light Format ist prinzipiell für alle gleich; zuerst wird der Typ angegeben und danach in runden Klammern die Eigenschaften, wobei die Reihenfolge dieser entscheidend ist. Im Folgenden wird die Reihenfolge der Eigenschaften beschrieben:

Punkt:	Point(wellKnownName, fillColor, size)
Linie:	Line(strokeColor, strokeWidth, strokeDashArray, perpendicularOffset)
Polygon:	Polygon(fillColor, fillOpacity, strokeColor, strokeWidth, strokeDashArray)
Text:	Text(label, fontSize, fontWeight, fill, anchorPointX, anchorPointY, displacementX, displacementY)

Weitere Regeln können hinzugefügt werden, indem weitere Angaben über Zoomstufen erfolgen. Dann ist klar, dass eine neue Regel angefangen wird und es können wieder Filter und Symbolizer hinzugefügt werden.

2.2 Aufbau des Editors

Das Hauptfenster des Editors besteht aus zwei Bereichen. Auf der linken Seite befindet sich eine grafische Benutzeroberfläche, mit dessen Hilfe die Eigenschaften für die SLD-Datei eingestellt werden können. Auf der rechten Seite werden die aus diesen Eigenschaften erstellten Dateien angezeigt (siehe Abbildung 3).

Beim Starten der Anwendung sind alle Bereiche weitestgehend leer. Um eine neue Regel hinzuzufügen klickt man auf den „Regel hinzufügen“-Button. Daraufhin werden vom Editor die Felder für „min Zoom“, „max Zoom“, „Filter“ sowie ein Dropdown-Menü für Symbolizer angezeigt. Um einen Symbolizer hinzuzufügen, muss man den Typ in dem Dropdown-Menü auswählen und auf „Objekt hinzufügen“ klicken. Der Editor schränkt den Benutzer aber insofern ein, dass für jede Regel nur jeweils ein Symbolizer eines bestimmten Typs hinzugefügt werden kann. Dies ist beabsichtigt, weil es prinzipiell wenig Sinn macht, in einer Regel zwei Symbolizer des gleichen Typs zu verwenden. Sobald ein Symbolizer hinzugefügt wurde, werden Eingabefelder mit den entsprechenden Eigenschaften erstellt und angezeigt. Für die Auswahl von Farben werden „ColorPicker“-Dialoge von JavaFX benutzt, damit der Benutzer nicht direkt mit den RGB-Farbcodes arbeiten muss.

Links neben jedem Symbolizer wird ein „Löschen“-Button angezeigt, welcher die Eingabefelder wieder entfernt. Um eine ganze Regel zu entfernen, muss man auf den „Regel löschen“-Button, welcher sich rechts neben jeder Regel befindet, drücken.

Die Oberfläche besteht aus einer Verschachtelung von JavaFX-„GridPanes“[2]. Beim Hinzufügen neuer Elemente wird in die passende Rasterzelle eine neue „GridPane“ eingefügt, welche die neuen Eingabefelder zur Verfügung stellt. Damit ist die Benutzeroberfläche sehr flexibel und kann sich gut an Änderungen anpassen.

Im „Datei“-Reiter der Menüleiste finden sich vier Einträge, welche es ermöglichen den Editor zurückzusetzen, eine SLD-Light-Datei zu öffnen und die aktuelle Sitzung als SLD- oder SLD-Light-Datei abzuspeichern. Unter dem „Server“-Reiter der Menüleiste gibt es nur einen Eintrag, der ein neues Fenster öffnet, welches Dateiupload auf den Geoserver ermöglicht. Hier kann eine SLD-Datei und ein Workspace ausgewählt werden und durch einen Klick auf den „Hochladen“-Button wird diese Datei auf den Geoserver hochgeladen, sofern dieser erreicht werden kann (Hinweise dazu in Abschnitt 3.2). Es stehen drei verschiedene Workspaces zur Verfügung: „PRODUCTION“, „TEST“ und „DEFAULT“. Diese entscheiden, in welches Verzeichnis die Datei auf dem Geoserver abgelegt wird.

2.3 Struktur der Objekte

Die Architektur der Objekte, welche die Daten über eine SLD-Layer enthalten, ist in Abbildung 1 dargestellt. Jedes Objekt implementiert das „SLDObject“-Interface und muss damit Implementierungen für die Methoden „toSLD()“ und „toSLDLight()“ bereitstellen. Die Symbolizer-Objekte implementieren das Marker-Interface „Symbolizer“. Damit wird sichergestellt, dass zur Liste im „Regel“-Objekt nur Symbolizer-Objekte und keine anderen SLD-Objekte hinzugefügt werden können.

Das Erzeugen der Inhalte für SLD und SLD-Light erfolgt über Aufrufe der SLDObject-Methoden „toSLD()“ bzw. „toSLDLight()“. Das „NamedLayer“ Objekt benutzt die Ergebnisse der SLDObject-Methoden der „Regel“-Objekte und die „Regel“-Objekte benutzen die Ergebnisse der SLDObject-Methoden der Symbolizer, welche in ihrer Liste stehen. So wird rekursiv das Endergebnis erzeugt.

Jedes SLD-Objekt hat einen Konstruktor, der einen String Parameter erwartet. Diese Zeichenkette muss die Definition des jeweiligen Objektes in SLD-Light-Kodierung beinhalten. Ein „LineSymbolizer“ würde im Konstruktor zum Beispiel so etwas wie „Line(#CCCCCC, 2, 5 2, 3)“ erwarten und die eigenen Attribute daraus ableiten.

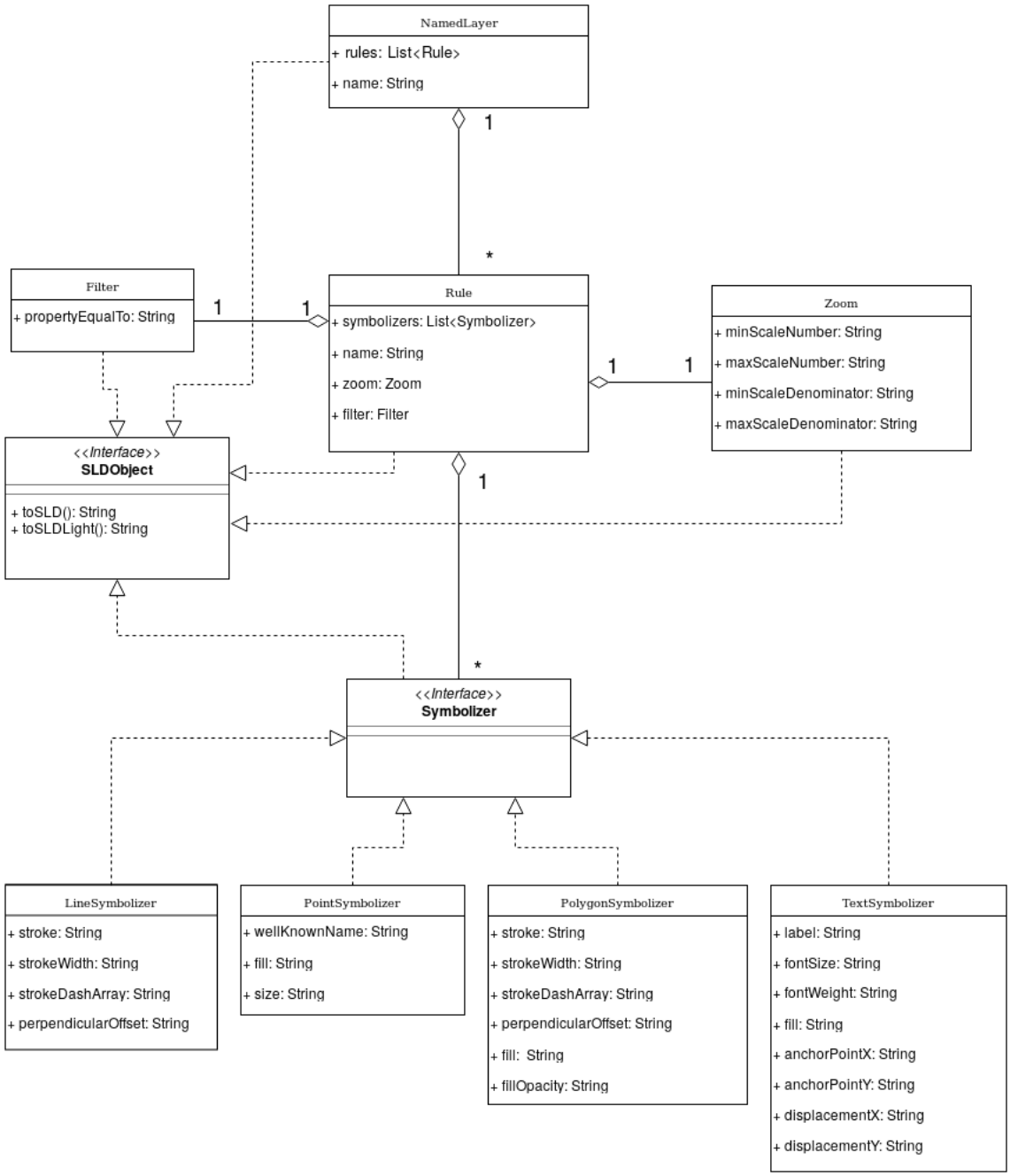

Abbildung 1: UML Diagramm: SLD-Objekte

2.4 Dateiupload

Der Editor ermöglicht es dem Benutzer eine SLD-Datei und einen Workspace auszuzuwählen, in welchen die Datei platziert werden soll (siehe Abbildung 2). Hierbei ist anzumerken, dass der Upload der SLD-Datei nicht ausreicht, da der Geoserver auf weitere Informationen in einer weiteren XML-Datei zurückgreift. Die XML-Datei wird von der Anwendung zusätzlich erzeugt und muss den gleichen Namen (bis auf die Dateiendung), wie die SLD-Datei haben. Ein Beispiel kann in Listing 4 betrachtet werden.

[2]Layout, welches das Anordnen von Elementen in einem Raster, ermöglicht

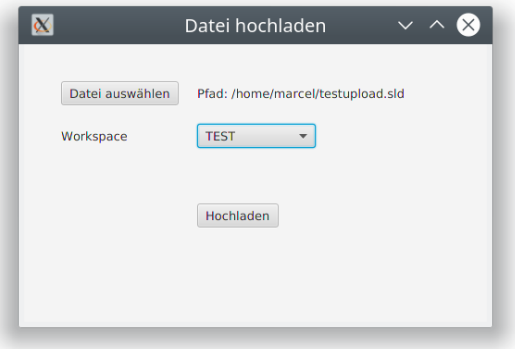

Abbildung 2: Dateiupload-Fenster

Die Workspace-ID kann XML-Dateien, welche sich auf dem Geoserver befinden, entnommen werden. Sie steht in der „workspace.xml“-Datei, welche sich in jedem Workspace-Verzeichnis befindet. Es ist unklar, wie sich die Style-ID zusammensetzt. Zwischen den XML-Dateien im gleichen Workspace unterscheidet sich bei dieser ID nur der Teil hinter dem letzten Doppelpunkt, was auf einen internen Nummerierungsmechanismus hindeutet. Die entwickelte Anwendung verwendet momentan bei einem Upload immer die gleiche Style-ID. Unerwünschte Folgen davon sind nicht bekannt, aber nicht auszuschließen.

2.5 Genutzte Komponenten

Zur Erstellung einer Verbindung mit dem OHDM-Server wird die „sshj“-Bibliothek[3] genutzt. Diese ermöglicht es „ssh“-, „scp“- und „sftp“-Verbindungen in Java zu erzeugen. In der entwickelten Anwendung werden nur SFTP-Verbindungen benötigt. Die anderen Dateien, welche sich im „lib“-Ordner des Projektes befinden, sind Abhängigkeiten, welche für die korrekte Ausführung der „sshj“-Bibliothek benötigt werden. Aktuelle Releases der Bibliothek können von der GitHub-Seite des Projektes heruntergeladen oder über Maven in das Projekt eingebunden werden. Die Einbindung wird auf der Projektseite detailliert erklärt.

3 Nutzung

3.1 Code

Der Code des Projektes wird auf `https://github.com/OpenHistoricalDataMap/SLDlight` zur Verfügung gestellt. Als Programmiersprache wird Java verwendet und zur Erstellung der grafischen Oberfläche wird JavaFX genutzt.

[3]Projektseite: `https://github.com/hierynomus/sshj/tree/master`

Wo findet man den Code. Struktur des Codes. (In Prototyphase ausfüllen, kann dort sehr kurz sein. Ab Alpha-Phase konkret beschreiben.)

3.2 Deployment / Runtime

Um das Programm aus dem Code zu erzeugen, muss der Projektordner in einer Java-Entwicklungsumgebung geöffnet und die „sshj“-Bibliothek zum Projekt hinzugefügt werden. Dazu können entweder die Dateien im „lib“-Ordner als Bibliotheken zum Projekt hinzugefügt oder über Maven eingebunden werden[4]. Da JavaFX verwendet wird, muss darauf geachtet werden, dass die Java-Installation auf dem PC auch JavaFX enthält, da es sonst zu Fehlern kommen kann.

Des Weiteren muss eine Klasse mit dem Dateinamen „SSHConstants.java“ im „app.sftp“-Package erstellt werden. In dieser Klasse müssen Benutzername und Passwort für einen Benutzer angegeben werden, welcher auf dem OHDM-Server registriert ist. Der Aufbau kann Listing 1 entnommen werden.

Außerdem benötigt der angegebene Benutzer bestimmte Rechte um Dateien im Verzeichnis des Geoservers abzulegen. Es genügt den Benutzer der Gruppe „tomcat7“ auf dem Geoserver hinzuzufügen. Dies ist beispielsweise mit Verwendung des Befehls sudo usermod −a −G tomcat7 [user−name] möglich.

Damit die Anwendung eine Verbindung zum Geoserver herstellen kann, muss sich der PC im Netzwerk der HTW Berlin befinden. Falls der Anwender sich nicht direkt im Netzwerk der HTW befindet, kann auf eine Verbindung über VPN zurückgegriffen werden.[5]

4 Qualitätssicherung

Die Qualität der Komponente wird nicht gesichert. Aus Zeitgründen wurde auf das Erstellen von Tests verzichtet.

5 Vorschläge / Ausblick

Ziel dieses Projektes war es, ein neues Dateiformat zu entwickeln, welches das Erstellen von „Styled-Layer-Descriptor“-Dateien vereinfacht. Weiterhin sollte eine Anwendung erstellt werden, welche Dateien des neuen Dateiformats in SLD-Dateien überführen kann und eine grafische Benutzeroberfläche bietet, um SLD- bzw SLD-Light-Dateien zu erzeugen.

[4] Hilfe bei Verwendung von Intellij IDEA als IDE siehe `https://www.jetbrains.com/help/idea/library.html`

[5] Mehr Informationen über die Benutzung des VPN an der HTW Berlin siehe: `https://anleitungen.rz.htw-berlin.de/de/vpn/`

Beide Teile wurden erfolgreich umgesetzt, wobei es trotzdem Bereiche gibt, in denen die Anwendung noch verbessert werden könnte.

Bei Anwendung des Editors in der Praxis wird sich zeigen, ob etwaige notwendige Attribute fehlen. Es wurden die Eigenschaften verwendet, welche am Häufigsten vorkommen bzw. am Wichtigsten schienen. Außerdem sollte die Zusammensetzung der Style-ID für die XML-Datei des Geoservers noch einmal überprüft werden. Weiterhin wäre es hilfreich, wenn man nicht nur eine SLD-Light-Datei mit dem Editor öffnen könnte, sondern auch eine SLD-Datei. Dies bringt allerdings andere Probleme mit sich, da die SLD-Datei Attribute definiert haben kann, die von dem Editor bzw. SLD-Light nicht unterstützt werden.

Ergebnisformulierend ist festzustellen, dass dieses Projekt gezeigt hat, dass die Entwicklung eines neuen Dateiformats, sowie einer Anwendung, welche dieses Format übersetzen und auf dem GeoServer platzieren kann, möglich ist und dazu beiträgt die Erstellung bzw. Bearbeitung von „Styled-Layer-Descriptor“-Dateien zu vereinfachen.

Literaturverzeichnis

[PLPV18] Pumphrey, Mike ; Lewis, Clint ; Petkov, Alex ; Vega, Eduin Yezid C.: *GeoServer User Manual*. http://docs.geoserver.org/stable/en/user/, 2018

Anhang

```java
package app.sftp;

public class SSHConstants {
    public static String USERNAME = "";
    public static String PASSWORD = "";
}
```

Listing 1: Die Klasse 'SSHConstants.java'

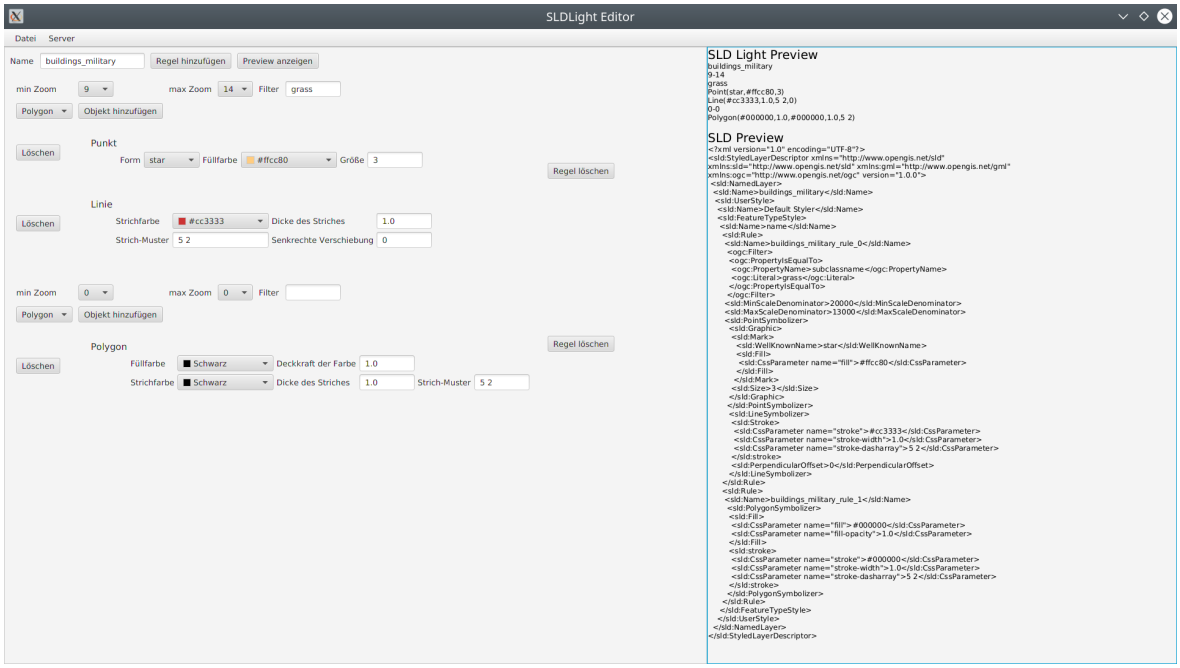

Abbildung 3: Ansicht des Editors

```
buildings_military
9-14
grass
Point(star,#ffcc80,3)
Line(#cc3333,1.0,5 2,0)
0-0
Polygon(#000000,1.0,#000000,1.0,5 2)
```

Listing 2: Beispiel für SLD-Light

```
<?xml version="1.0" encoding="UTF-8"?>
<sld:StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:sld="http://www
 <sld:NamedLayer>
  <sld:Name>buildings_military</sld:Name>
   <sld:UserStyle>
    <sld:Name>Default Styler</sld:Name>
    <sld:FeatureTypeStyle>
     <sld:Name>name</sld:Name>
```

```
9         <sld:Rule>
10         <sld:Name>buildings_military_rule_0</sld:Name>
11          <ogc:Filter>
12           <ogc:PropertyIsEqualTo>
13            <ogc:PropertyName>subclassname</ogc:PropertyName>
14            <ogc:Literal>grass</ogc:Literal>
15           </ogc:PropertyIsEqualTo>
16          </ogc:Filter>
17         <sld:MinScaleDenominator>20000</sld:MinScaleDenominator>
18         <sld:MaxScaleDenominator>13000</sld:MaxScaleDenominator>
19         <sld:PointSymbolizer>
20           <sld:Graphic>
21            <sld:Mark>
22              <sld:WellKnownName>star</sld:WellKnownName>
23              <sld:Fill>
24               <sld:CssParameter name="fill">#ffcc80</sld:CssParameter>
25              </sld:Fill>
26             </sld:Mark>
27            <sld:Size>3</sld:Size>
28           </sld:Graphic>
29          </sld:PointSymbolizer>
30           <sld:LineSymbolizer>
31            <sld:Stroke>
32             <sld:CssParameter name="stroke">#cc3333</sld:CssParameter>
33             <sld:CssParameter name="stroke-width">1.0</sld:CssParameter>
34             <sld:CssParameter name="stroke-dasharray">5 2</sld:CssParameter>
35            </sld:stroke>
36            <sld:PerpendicularOffset>0</sld:PerpendicularOffset>
37           </sld:LineSymbolizer>
38        </sld:Rule>
39        <sld:Rule>
40         <sld:Name>buildings_military_rule_1</sld:Name>
41          <sld:PolygonSymbolizer>
42           <sld:Fill>
43            <sld:CssParameter name="fill">#000000</sld:CssParameter>
44            <sld:CssParameter name="fill-opacity">1.0</sld:CssParameter>
45           </sld:Fill>
46           <sld:stroke>
47            <sld:CssParameter name="stroke">#000000</sld:CssParameter>
48            <sld:CssParameter name="stroke-width">1.0</sld:CssParameter>
49            <sld:CssParameter name="stroke-dasharray">5 2</sld:CssParameter>
50           </sld:stroke>
51          </sld:PolygonSymbolizer>
52        </sld:Rule>
53       </sld:FeatureTypeStyle>
54     </sld:UserStyle>
55   </sld:NamedLayer>
56 </sld:StyledLayerDescriptor>
```

Listing 3: Beispiel für SLD

```
<style>
  <id>StyleInfoImpl-4e1160b8:160efbd149b:40ff</id>
  <name>buildings_military</name>
  <workspace>
    <id>WorkspaceInfoImpl-355a8844:15b19d1e2c3:-7ffe</id>
  </workspace>
  <format>sld</format>
  <languageVersion>
    <version>1.0.0</version>
  </languageVersion>
  <filename>buildings_military.sld</filename>
</style>
```

Listing 4: Beispiel XML-Datei für Geoserver

Zoomstufe	SLD-Wert
0	keine Angabe
1	100000
2	80000
3	60000
4	50000
5	40000
6	30000
7	25000
8	22000
9	20000
10	18000
11	16000
12	15000
13	14000
14	13000
15	12000
16	8000

Tabelle 1: Zoomstufen