Geomajas TMS layer plug-in

Geomajas Developers and Geosparc

Geomajas TMS layer plug-in

by Geomajas Developers and Geosparc

1.1.0

Table of Contents

List of Tables

List of Examples

Chapter 1. Introduction

This plug-in contains a raster layer definition for accessing raster data from TMS servers as specified by OSGeo. See the Specification [http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification]. Although not an official OGC specification, TMS is easy to understand and has the advantage of high performance.

This plug-in does not support all TileMapService capabilities. It only supports a single TileMap configuration, represents a single layer. The supported TMS version is 1.0.0.

Chapter 2. Configuration

This chapter describes the perils of configuring a TMS layer for your Geomajas application. There are 2 obstacles to conquer: getting the TMS plug-in jar on your class path, and configuring a TMS layer using the Geomajas configuration.

1. Java dependencies

If you are using Maven, you're in luck. All you have to do is add the following dependency to your configuration:

Example 2.1. TMS layer dependency, using geomajas-dep

```
<dependency>
    <groupId>org.geomajas.plugin</groupId>
    <artifactId>geomajas-layer-tms</artifactId>
</dependency>
```

If you are not using geomajas-dep to manage versions, then you need to mention the version as well.

If you are not a Maven user, make sure that there is a recent version of JaxB (2.x) on your class path. Normally the Geomajas server has already required this, but we're mentioning it just in case.

1.1. Configuring a TMS layer

When configuring a TMS layer, there are 2 basic options to chose from: either configure your own RasterLayerInfo object (as with all other Geomajas raster layers), or you let the TMS plug-in configure the layer automatically, based upon the TMS layer description (see the TileMap resource in the TMS specification [http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification]).

1.2. Option 1: using the TMS layer description (TileMap)

This option is the easiest one, but only works if your TMS actually has a TileMap description. If no such description is available, you need to configure a RasterLayerInfo yourself.

This options requires you to specify the TMS layer description URL, upon which the TMS plug-in will retrieve the description and build a `RasterLayerInfo` automatically. The CRS, bounding box, resolutions, etc will be taken from the TileMap description. Below is an example of such a configuration:

```
<bean name="layerTms" class="org.geomajas.layer.tms.TmsLayer" >
    <property name="baseTmsUrl" value="http://tms.osgeo.org/1.0.0/landsat/" />
</bean>
```

Note

While this is the easiest way to configure a TMS layer, it can cause application startup to fail if the specified URL cannot be read or parsed. To prevent application startup failure when the TMS service is down during application startup, use the RasterLayerInfo configuration option mentioned below.

1.3. Option 2: configuring a RasterLayerInfo

This option would can be used in case there is no TMS description available, or in case you wish to use different configuration options. You would still need to specify the "baseTmsUrl" and a `RasterLayerInfo` object:

```
<bean name="layerTms" class="org.geomajas.layer.tms.TmsLayer" >
    <property name="baseTmsUrl" value="http://tms.osgeo.org/1.0.0/landsat/" />
    <property name="extension" value="jpg" />
    <property name="layerInfo" ref="layerTmsInfo" />
</bean>

<bean name="layerTmsInfo" class="org.geomajas.configuration.RasterLayerInfo">
    <property name="crs" value="EPSG:4326"/>
    <property name="maxExtent">
        <bean class="org.geomajas.geometry.Bbox">
            <property name="x" value="-180"/>
            <property name="y" value="-90"/>
            <property name="width" value="360"/>
            <property name="height" value="180"/>
        </bean>
    </property>
    <property name="resolutions">
        <list>
            <value>0.5</value>
            <value>0.25</value>
            <value>0.125</value>
            <value>0.0625</value>
            <value>0.03125</value>
            <value>0.015625</value>
            <value>0.0078125</value>
            <value>0.00390625</value>
            <value>0.001953125</value>
            <value>0.0009765625</value>
            <value>0.00048828125</value>
            <value>0.000244140625</value>
            <value>0.000122070312</value>
        </list>
    </property>
    <property name="dataSourceName" value="landsat" />
    <property name="tileWidth" value="512"/>
    <property name="tileHeight" value="512"/>
</bean>
```

Note that this time, an extension and a link to a `RasterLayerInfo` have been added. The above configuration will generate tiles like:

```
http://tms.osgeo.org/1.0.0/landsat/0/0/0.jpg
```

1.4. Using authentication

The optional HTTP authentication allows the TMS layer to send user credentials to the TMS server in the HTTP headers. Although this is an option, often TMS servers will be secured, and require such authentication. Below is an example of a TMS layer configuration that uses the HTTP authentication:

```
<bean name="layerTms" class="org.geomajas.layer.tms.TmsLayer" >
    <property name="baseTmsUrl" value="http://tms.osgeo.org/1.0.0/landsat/" />
    <property name="extension" value="jpg" />
    <property name="layerInfo" ref="layerTmsInfo" />

    <property name="authentication">
        <bean class="be.geomajas.layer.common.proxy.LayerHttpAuthentication">
            <property name="user" value="<the user name>" />
            <property name="password" value="<password>" />
```

```
            <property name="realm" value="<optional realm>" />
            <property name="applicationUrl" value="<the URL for this web applic
        </bean>
    </property>
</bean>
```

Let us go over the properties for the authentication bean:

Table 2.1. TMS authentication properties

Name	Description
user (required)	The user login name.
password (required)	The users password.
realm (optional)	The HTTP realm for this user. This is an optional value.
authenticationMethod (optional)	Authentication method to use. Options are `LayerAuthenticationMethod.BASIC` (default) and `LayerAuthenticationMethod.URL`.
userKey *(optional)*	Key which is used to pass the user name when using the URL authentication type.
passwordKey *(optional)*	Key which is used to set the password when using the URL authentication type.

1.5. Using the proxy

When using the optional Proxy setting you can provide extra HttpRequest interceptors which will be used by the proxy to further customise your requests (add extra parameters / headers for security or do some logging). To add interceptors add a bean of type LayerHttpServiceInterceptors to your spring configuration. This contains a map of HttpRequestInterceptor lists. The key is the name of a layer or the prefix to the baseUrl property of the RasterLayerInfo object.

To add an interceptor to all layers use an empty key (eg. ""). To add an interceptor to a specific layer use the name of this layer.

Sample bean configuration:

```
<bean name="layerTms" class="org.geomajas.layer.tms.TmsLayer" >
    <property name="baseTmsUrl" value="http://tms.osgeo.org/1.0.0/landsat/" />
    <property name="extension" value="jpg" />
    <property name="layerInfo" ref="layerTmsInfo" />
    <property name="useProxy"  value="true" />
</bean>

<bean name="interceptors" class="org.geomajas.layer.common.proxy.LayerHttpServi
   <property name="map">
      <map>
         <entry key="">
            <list>
               <bean class="org.geomajas.layer.tms.sample.AddSomeLeetHeadersHtt
            </list>
         </entry>
      </map>
   </property>
</bean>
```

Chapter 3. How-to