Geomajas WMS layer plug-in

Geomajas Developers and Geosparc

Geomajas WMS layer plug-in

by Geomajas Developers and Geosparc

1.11.0

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 WMS servers. It has support for WMS versions (1.0.0, 1.1.0, 1.1.1 and 1.3.0)

Since version 1.8.0 of this plug-in, there is also built-in support for basic and digest HTTP authentication. On top of this, the option remains to add user name and password parameters as GET parameters to each WMS GetMap request.

A command exists to access the GetFeatureInfo request which is available on some WMS servers.

Chapter 2. Configuration

1. Dependencies

Make sure sure you include the correct version of the plug-in in your project. Use the following excerpt (with the correct version) in the dependencyManagement section of your project:

```
 <dependency>
    <groupId>org.geomajas.plugin</groupId>
    <artifactId>geomajas-plugin-wms-all</artifactId>
    <version>1.8.0</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

If you are using geomajas-dep, this includes the latest released version of the wms layer (at the time of publishing of that version). If you want to overwrite the caching plug-in version, make sure to include this excerpt *before* the geomajas-dep dependency.

You can now include the actual dependency without explicit version.

Example 2.1. WMS layer dependency

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

If you are using the GWT face and want the access the GetFeatureInfo capability of your WMS server, then you should use the following dependency instead.

Example 2.2. WMS layer dependency

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

In that case you should also put "<inherits name="org.geomajas.layer.wms.GeomajasWms"/>" in your .gwt.xml file.

2. WMS layer configuration

2.1. Defining a WMS layer

A complete WMS layer configuration looks as follows:

Example 2.3. WMS layer configuration

```
    <bean name="wmsBluemarbleInfo" class="org.geomajas.configuration.RasterLaye
        <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="bluemarble" />
        <property name="tileWidth" value="512"/>
        <property name="tileHeight" value="512"/>
    </bean>

    <bean name="wmsBluemarble" class="org.geomajas.layer.wms.WmsLayer" >
        <property name="layerInfo" ref="wmsBluemarbleInfo" />

        <!-- When configuring your own applications, please do not use this WMS
        <property name="baseWmsUrl" value="http://apps.geomajas.org/geoserver/w
        <property name="version" value="1.1.1"/>
        <property name="format" value="image/jpeg"/>
        <property name="styles" value=""/>
    </bean>
```

Caution

The correct display of a WMS layer with respect to other layers can only be assured if the CRS of the layer is the same as the CRS of the map. If the CRS values differ, the effect may be translation or distortion of the layer data when drawn on the map. Change the CRS of the map if necessary.

The first property is the layer info object, which describes the metadata common to all raster layers. Visit the Geomajas developer guide [http://files.geomajas.org/maven/trunk/geomajas/docbook-devuserguide/html/master.html#conf-raster] to learn more about it. This guide will focus on the WMS specific configuration options.

The most important parameter here, is the dataSourceName, which must point to a layer name as specified by the WMS server. Note that comma separated values are supported, as the layer name is literally used in the WMS requests.

As you can see, the bean class refers to the actual layer type: `org.geomajas.layer.wms.WmsLayer`. The layer object contains some extra properties which are specifically tied to the WMS server. Some of these are required, some are optional.

Table 2.1. WMS layer properties

Name	**Description**
baseWmsUrl (required)	The base url of the WMS server. This is the base part (excluding the request parameters) of the url that would be called to execute a WMS request.
version (optional)	Version of the WMS protocol which should be used. Check your server configuration for possible values. This defaults to "1.1.1".
format (optional)	The mime type in which the images should be returned, for example "image/gif". Check your server configuration for possible values. This defaults to "image/png".
styles (optional)	Some WMS servers support multiple styles for their layers. Check your server configuration for possible values.
parameters (optional)	You can define additional parameters which can be passed to the WMS server for each GetMap request. These are name/value pairs which are passed in `org.geomajas.configuration.Parameter` objects. See "Adding extra parameters" for more information.
authentication (optional)	Optional basic or digest HTTP authentication. This bean is of the type: `org.geomajas.layer.common.proxy.LayerAuthentication`. When set, the WMS request is automatically proxied to assure that credentials are not leaked to the client. See "Using authentication" for more information.
useProxy (optional)	Allows you to force the use of a proxy to get the image. This can be useful if you want to hide the WMS URL (excluding the parameters that is). This is automatically enabled when the authentication property is set (to assure that credentials are not visible on the user).
useCache (optional)	Allows you to force caching of WMS tiles.This can be useful if you want to unload the used WMS server. This automatically enables the useProxy property to be able to cache the images.
enableFeatureInfoSupport (optional)	Setting this to true allows you to use the SearchByPoint command to access the GetFeatureInfo capability of the WMS server.

2.2. Adding extra parameters to the WMS requests

In the previous paragraph, we mentioned the possibility to configure extra parameters to be passed to the WMS server when executing the GetMap requests. This option can be useful for example for configuring the use of transparency.

Below is an example of a WMS layer configuration which uses transparency:

```
<bean name="wms03" class="org.geomajas.layer.wms.WmsLayer" >
    <property name="layerInfo" ref="wms03Info" />

    <!-- When configuring your own applications, please do not use this WMS ser
    <property name="baseWmsUrl" value="http://apps.geomajas.org/geoserver/wms" 
    <property name="version" value="1.1.1"/>
    <property name="format" value="image/png;%20mode=24bit"/>
    <property name="styles" value=""/>
    <property name="parameters">
        <list>
            <bean class="org.geomajas.configuration.Parameter">
                <property name="name" value="transparent" />
                <property name="value" value="TRUE" />
            </bean>
        </list>
    </property>
</bean>
```

2.3. Using authentication

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

```
<bean name="wmsOrtho" class="be.geomajas.layer.wms.WmsLayer" >
    <property name="layerInfo" ref="wmsOrthoInfo" />

    <property name="baseWmsUrl" value="http://ogc.beta.agiv.be/ogc/wms/orthoklm
    <property name="version" value="1.3.0" />
    <property name="format" value="image/jpeg" />
    <property name="styles" value="" />

    <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 2.2. WMS 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.

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

Chapter 3. How-to

Usage examples for the WMS layer.

1. Using the GetFeatureInfo capability

As an example, let's show some code which implements a click (or mouse up) event which should read the feature information. It assumes the map widget is known in the variable mapWidget. :

Example 3.1. Using the SearchByPoint command

```
public void onMouseUp(MouseUpEvent event) {
    Coordinate worldPosition = getWorldPosition(event);
    Point point = mapWidget.getMapModel().getGeometryFactory().createPoint(worl

    final SearchByPointRequest rasterLayerRequest = new SearchByPointRequest();
    rasterLayerRequest.setLocation(point.getCoordinate());
    rasterLayerRequest.setCrs(mapWidget.getMapModel().getCrs());
    rasterLayerRequest.setSearchType(SearchByPointRequest.SEARCH_FIRST_LAYER);
    rasterLayerRequest.setPixelTolerance(pixelTolerance);
    rasterLayerRequest.setLayerIds(getServerLayerIds(mapWidget.getMapModel()));
    rasterLayerRequest.setBbox(toBbox(mapWidget.getMapModel().getMapView().getB
    rasterLayerRequest.setScale(mapWidget.getMapModel().getMapView().getCurrent

    GwtCommand commandRequest = new GwtCommand(SearchByPointRequest.COMMAND);
    commandRequest.setCommandRequest(rasterLayerRequest);
    GwtCommandDispatcher.getInstance().execute(commandRequest,
      new AbstractCommandCallback<SearchByPointResponse>() {
        public void execute(SearchByPointResponse response) {
            // ... process response
        }
    });
}
```

This is the default.
Pixel tolerance if known. Currently not used by the WMS layer.
Here you can do what needs to be done with the response.

Note that the features returned are not defined anywhere in the configuration. The feature is filled with whatever data is retrieved from the WMS server.