Geomajas admin plug-in guide

Geomajas Developers and Geosparc

Geomajas admin plug-in guide

by Geomajas Developers and Geosparc

1.0.0-M1

Table of Contents

List of Examples

Chapter 1. Introduction

The administration plugin adds the capability to dynamically maintain the Geomajas Spring application context. Prior to this plugin, the application context was loaded once and for all at startup and any changes to the context would necessarily involve a restart of the complete web application. While this is standard practice for Spring-configured systems, it prevents common GIS administrative tasks like:

- dynamically configuring layers: vector layers, raster layers, styling, attributes, caching, security, ...
- dynamically configuring fysical datastores: geodatabases, shapefiles

or more specific Geomajas-related tasks like:

- dynamically configuring applications: maps, layers, tools, etc...
- dynamically configuring services and commands

There are various technical roads that could be pursued to reach this goal, some of them involving a complete rehaul of the existing service infrastructure. For this plugin we have tried to follow the least intrusive, yet most generic option, which is to directly perform application context changes on a runtime system. Initial testing has shown that this is a viable road, although some limitations have to be taken into account:

- Spring Dependency Injection, and more particularly autowiring, will need to be reapplied to reestablish bean dependencies within the context. This is not part of the standard bean definition registration mechanism, although it can be achieved partially by manually rewiring certain bean types.
- Application context persistence is a separate issue that comes into play when the context becomes dynamic. It will be handled by a separate persistence service.
- Concurrency issues may pop up as a consequence of dynamic configuration. As Geomajas is a stateless system, this should have limited consequences in most cases, although some glitches may appear during the configuration stage. The overall approach here should be to avoid locks by only sharing immutable objects. In the end it is up to the service implementors to ensure proper behavior after reconfiguration.

1.

Chapter 2. Use

How to use the admin plug-in.

Chapter 3. Configuration

Configuration for the admin plug-in.

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-runtimeconfig-all</artifactId>
    <version>1.0.0</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

If you are using geomajas-dep, this includes the latest released version of the caching plug-in (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 3.1. Plug-in dependency

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

If you want to use the admin widget for the GWT face, you need the following dependency:

Example 3.2. Plug-in GWT dependency

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

Chapter 4. How-to

This chapter details the extension possibilities of the admin plug-in.

1.