GEOMAJAS

Technology & Architecture

FOSDEM Brussels, Januari 2015

Jan De Moerloose

Geomajas – Technology & Architecture

Todays menu:

- What is Geomajas?
- Code samples
- (Security in Geomajas)
- Demo
- Q & A

What is Geomajas?

What is Geomajas?

Quotes from the website:

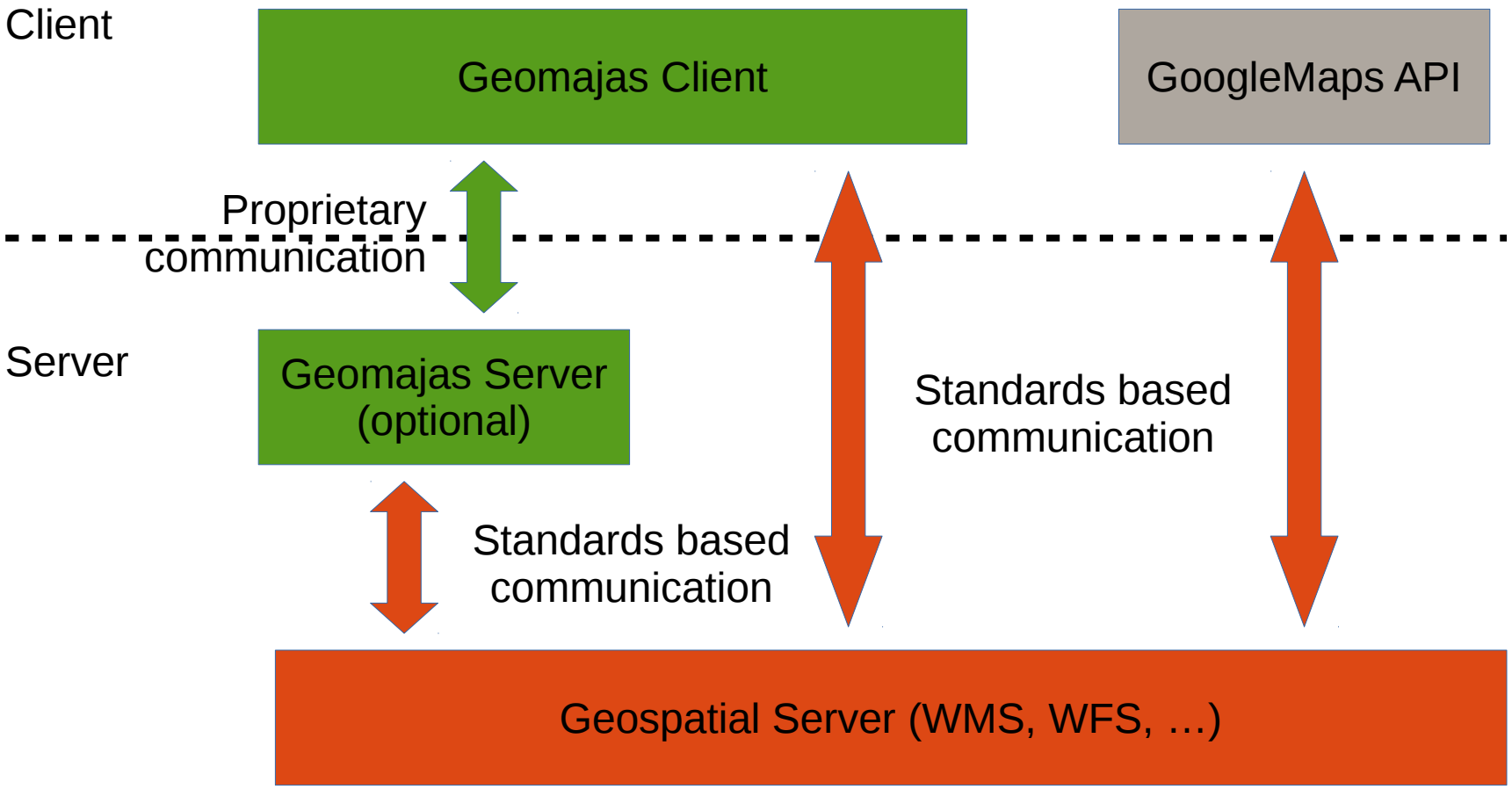
“Geomajas is the open source platform to create Web GIS applications”

“Geomajas is a collection of free and open source GIS libraries, tools and API's for a complete end-to-end web mapping solution”

What is unique about Geomajas?

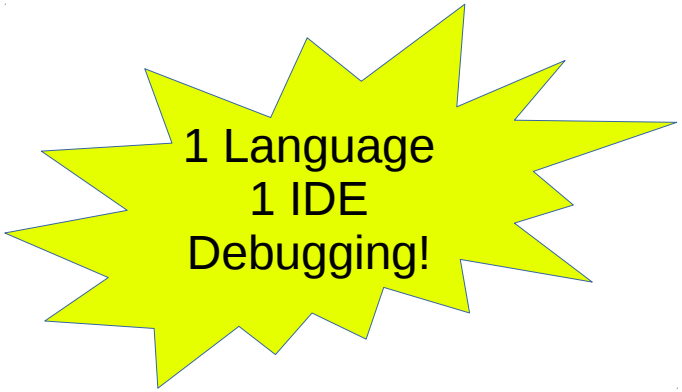
- Provides both client & server libraries
  - “Server layer” concept
  - Built-in security
- Language:
  - Client: GWT
  - Server: Java
- Lots of plugins

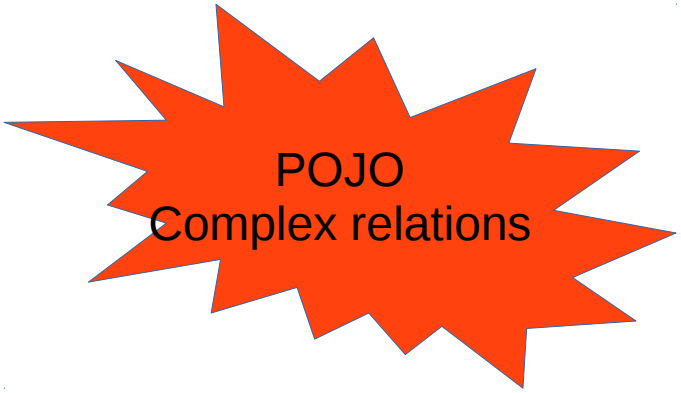
What is Geomajas?



Technologies

- Client:
  - GWT
- Server:
  - Java
  - Spring
  - Geotools
  - JTS
  - Hibernate Spatial

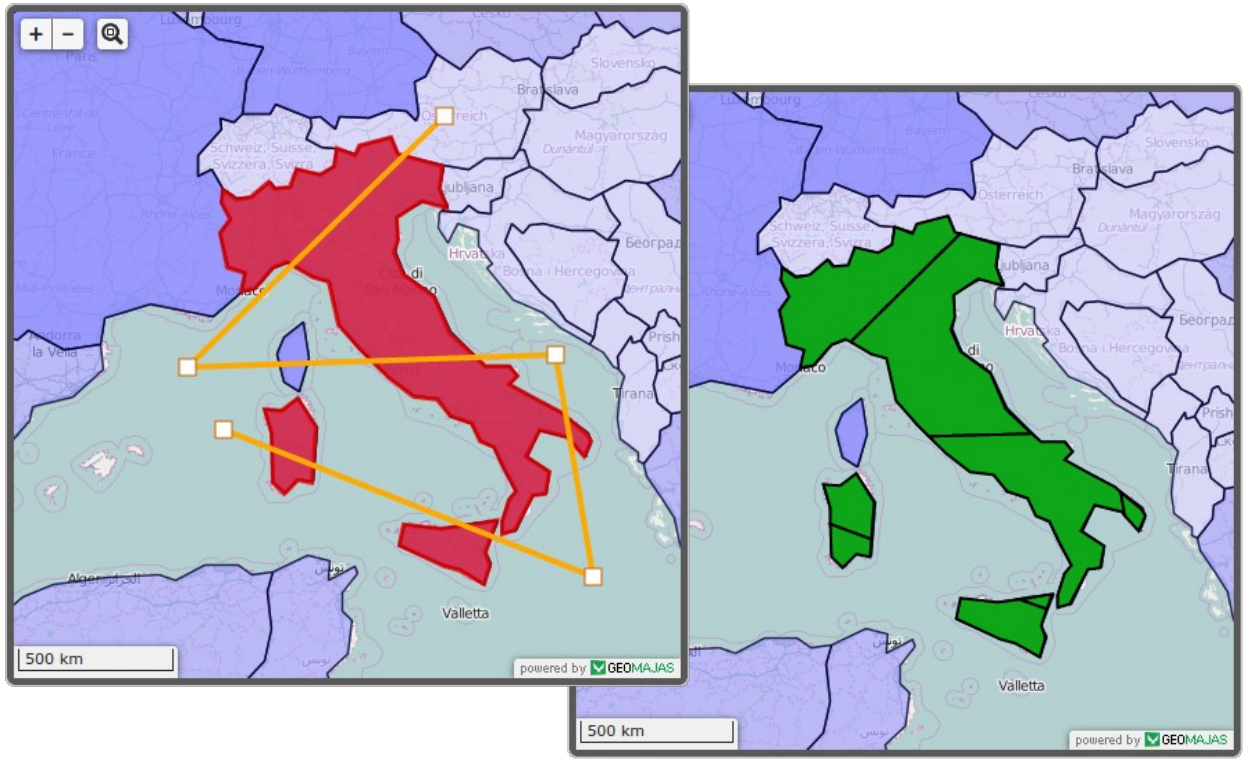




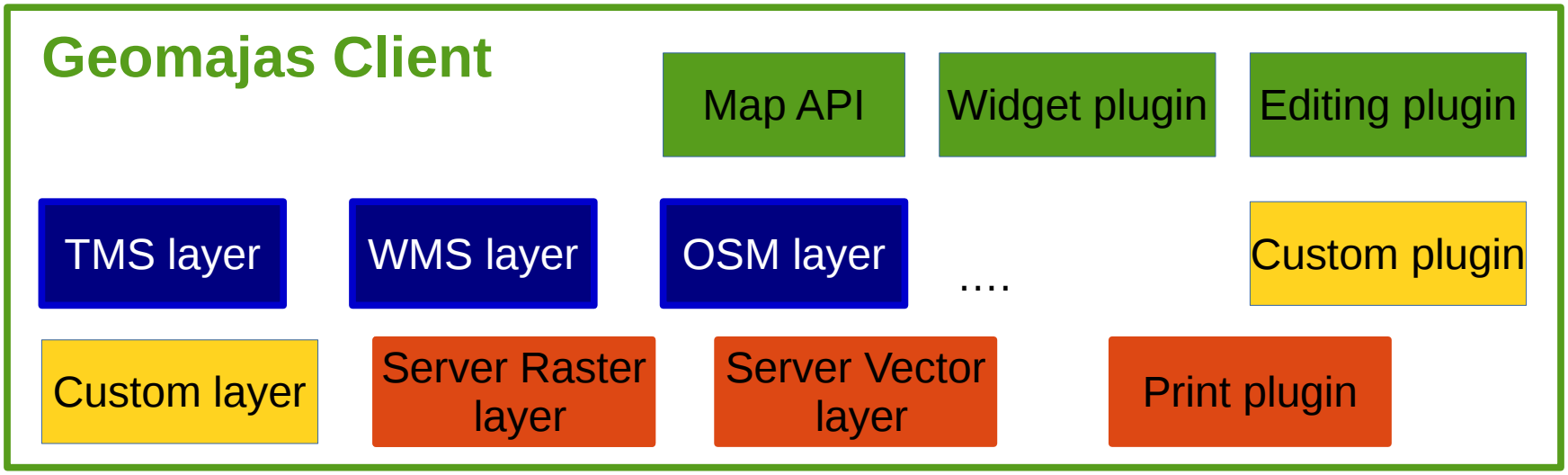
Screenshot

- Technical examples:
- http://apps.geomajas.org/geomajas-client-gwt2-example-2.2.0/

Example:
~~Country~~ Polygon splitting



Client-side



Uses standards based services
Requires Geomajas Server

Code samples

Creating a map

```
// Create the mapPresenter and add an InitializationHandler:
MapConfiguration configuration = new MapConfigurationImpl();
configuration.setCrs(EPSG, CrsType.DEGREES);
configuration.setMaxBounds(new Bbox(-180, -90, 360, 180));
List<Double> resolutions = new ArrayList<Double>();
resolutions.add(0.703125);
resolutions.add(0.3515625);
resolutions.add(0.17578125);
resolutions.add(0.087890625);
resolutions.add(0.0439453125);
configuration.setResolutions(resolutions);
mapPresenter = GeomajasImpl.getInstance().createMapPresenter(configuration,
480, 480);

// Define the whole layout:
DecoratorPanel mapDecorator = new DecoratorPanel();
mapDecorator.add(mapPresenter.asWidget());
mapPanel.add(mapDecorator);
```

Adding a layer (TMS)

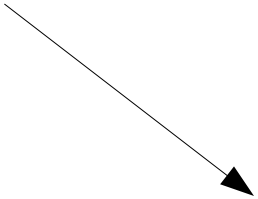
```
TmsClient.getInstance().getTileMap("d/proxy?
url=http://apps.geomajas.org/geoserver/gwc/service/tms/1.0.0" +
"/demo_world%3Asimplified_country_borders@EPSG%3A4326@png", new
Callback<TileMapInfo, String>() {

@Override
public void onSuccess(TileMapInfo result) {
TmsLayer layer = TmsClient.getInstance().createLayer(result);
mapPresenter.getLayersModel().addLayer(layer);
mapPresenter.getLayersModelRenderer().setAnimated(layer, true);
}

@Override
public void onFailure(String reason) {
Window.alert("We're very sorry, but something went wrong: " + reason);
}
});
```

Adding a layer (WMS)

```
// Now create a WMS layer and add it to the map:
TileConfiguration tileConfig = new TileConfiguration(256, 256, new
Coordinate(-180, -90),
mapPresenter.getViewPort());
WmsLayerConfiguration layerConfig = new WmsLayerConfiguration();
layerConfig.setBaseUrl(WMS_BASE_URL);
layerConfig.setFormat("image/png");
layerConfig.setVersion(getWmsVersion());
layerConfig.setLayers("demo_world:simplified_country_borders");
layerConfig.setMaximumResolution(Double.MAX_VALUE);
layerConfig.setMinimumResolution(2.1457672119140625E-5);

final WmsLayer wmsLayer = WmsClient.getInstance().createLayer("Blue
Marble",
mapPresenter.getViewPort().getCrs(), tileConfig, layerConfig, null);
wmsLayer.setMaxBounds(new Bbox(-180, -90, 360, 360));
mapPresenter.getLayersModel().addLayer(wmsLayer);
```

Adding a server-side layer

```
// Create the MapPresenter and add an InitializationHandler:
mapPresenter = GeomajasImpl.getInstance().createMapPresenter();
mapPresenter.setSize(480, 480);
mapPresenter.getEventBus().addMapCompositionHandler(new
MyMapCompositionHandler());

DecoratorPanel mapDecorator = new DecoratorPanel();
mapDecorator.add(mapPresenter.asWidget());
mapPanel.add(mapDecorator);

// Initialize the map, and return the layout:
GeomajasServerExtension.getInstance().initializeMap(mapPresenter, "gwt-app",
"mapLegend");
```



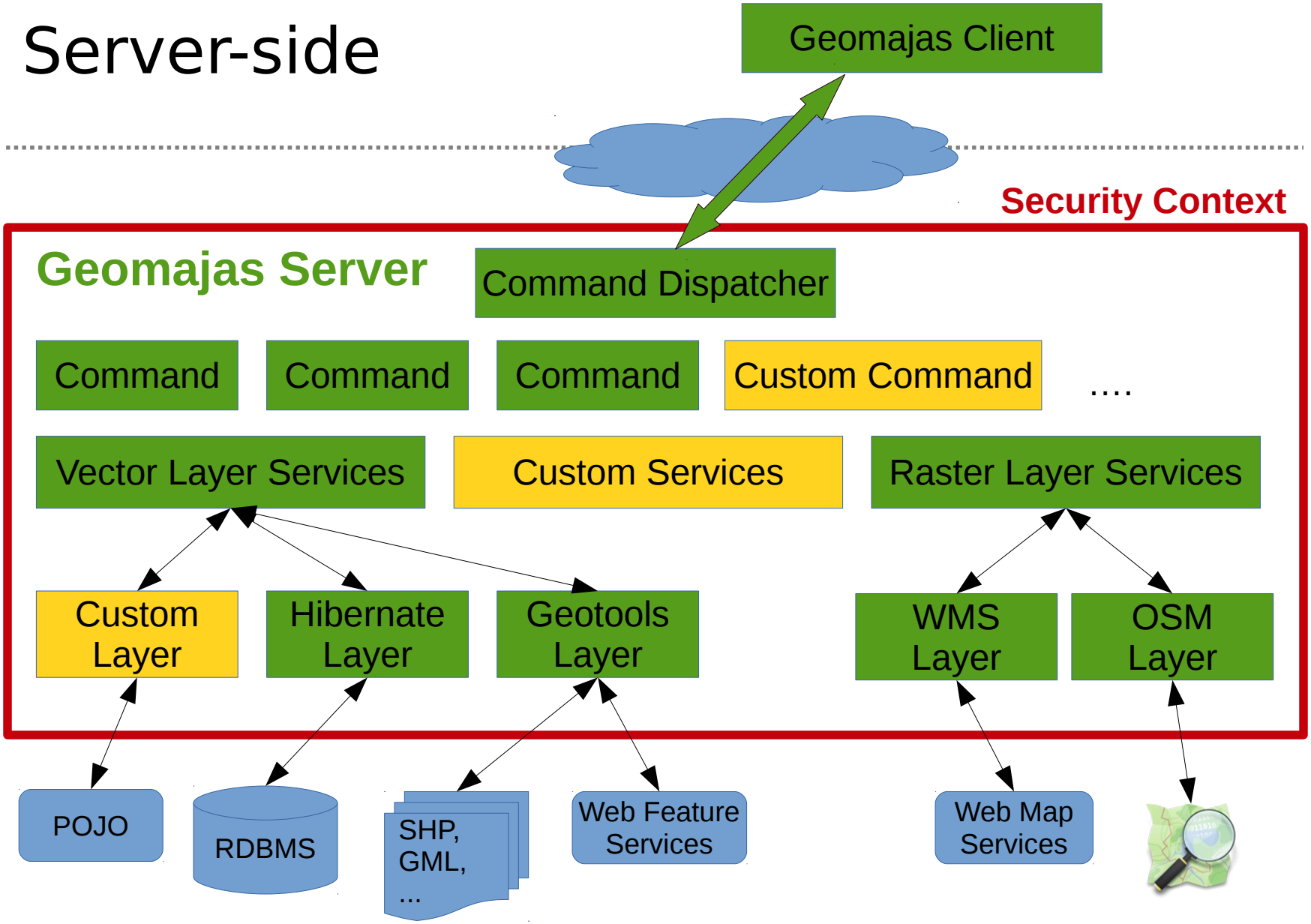
The map “mapLegend” is defined in XML on the server side

```
<bean name="mapLegend" class="org.geomajas.configuration.client.ClientMapInfo">
    <property name="backgroundColor" value="#FFFFFF" />
    <property name="lineSelectStyle">
        <bean class="org.geomajas.configuration.FeatureStyleInfo">
        </bean>
    </property>
    <property name="pointSelectStyle">
        <bean class="org.geomajas.configuration.FeatureStyleInfo">
        </bean>
    </property>
    <property name="polygonSelectStyle">
        <bean class="org.geomajas.configuration.FeatureStyleInfo">
        </bean>
    </property>
    <property name="crs" value="EPSG:4326" />
    <property name="scaleBarEnabled" value="true" />
    <property name="panButtonsEnabled" value="true" />
    <property name="scaleConfiguration">
        <bean class="org.geomajas.configuration.client.ScaleConfigurationInfo">
            <property name="maximumScale" value="1:100" />
        </bean>
    </property>
    <property name="initialBounds">
        <bean class="org.geomajas.geometry.Bbox">
            <property name="x" value="-128.5"/>
            <property name="y" value="16"/>
            <property name="width" value="64.5"/>
            <property name="height" value="35"/>
        </bean>
    </property>
    <property name="layers">
        <list>
            <ref bean="clientLayerWms" />
            <ref bean="clientLayerCountries110m" />
            <ref bean="clientLayerRivers50m" />
            <ref bean="clientLayerPopulatedPlaces110m" />
        </list>
    </property>
</bean>
```

More complex stuff: map controller sample

```
private GeometryEditService editService;

private MapController startRectangleController;

public PolygonAreaController() {
    GeometryEditor editor = Editing.getInstance().createGeometryEditor(mapPresenter);
    editService = editor.getEditService();

}

@Override
public void onActivate(MapPresenter mapPresenter) {
   // activate the rectangle controller first
   mapPresenter.setMapController(startRectangleController);
}

public void setGeometry(Geometry geometry) {
   editService.start(geometry);
   editService.setEditingState(GeometryEditState.IDLE);
}

/**
 * Controller for drawing the initial rectangle.
 */
  class StartRectangleController extends AbstractRectangleController {

   @Override
   public void execute(Bbox worldBounds) {
      Geometry location = GeometryService.toPolygon(worldBounds);
      setGeometry(location);
   }

  }
}
```

Server-side


Geomajas Client
Security Context
Geomajas Server
Command Dispatcher
Command
Command
Command
Custom Command
....
Vector Layer Services
Custom Services
Raster Layer Services
Custom Layer
Hibernate Layer
Geotools Layer
WMS Layer
OSM Layer
POJO
RDBMS
SHP, GML, ...
Web Feature Services
Web Map Services

Domain Model Support

- ORM via Hibernate Layer
- Spatial via Hibernate Spatial
- Support for Associations (single, multiple)
- Customizable through API

Layer Access

```
@UserImplemented
public interface VectorLayer extends Layer<VectorLayerInfo> {
    boolean isCreateCapable();
    boolean isUpdateCapable();
    boolean isDeleteCapable();
    Object create(Object feature) throws LayerException;
    Object saveOrUpdate(Object feature) throws LayerException;
    void delete(String featureId) throws LayerException;
    Iterator<?> getElements(Filter filter, int offset, int maxResultSize) throws LayerException;
    Envelope getBounds(Filter filter) throws LayerException;
    Envelope getBounds() throws LayerException;
}
```

1 java interface to implement

Object feature

your custom java object

Filter filter

Generic filter to pass to your ORM layer

Feature Access

```
@UserImplemented
public interface FeatureModel {
    void setLayerInfo(VectorLayerInfo vectorLayerInfo) throws LayerException;
    Attribute getAttribute(Object feature, String name) throws LayerException;
    Map<String, Attribute> getAttributes(Object feature) throws LayerException;
    String getId(Object feature) throws LayerException;
    Geometry getGeometry(Object feature) throws LayerException;
    void setAttributes(Object feature, java.util.Map<String, Attribute> attributes) throws LayerException;
    void setGeometry(Object feature, Geometry geometry) throws LayerException;
    Object newInstance() throws LayerException;
    Object newInstance(String id) throws LayerException;
    int getSrid() throws LayerException;
    String getGeometryAttributeName() throws LayerException;
    boolean canHandle(Object feature);
}
```

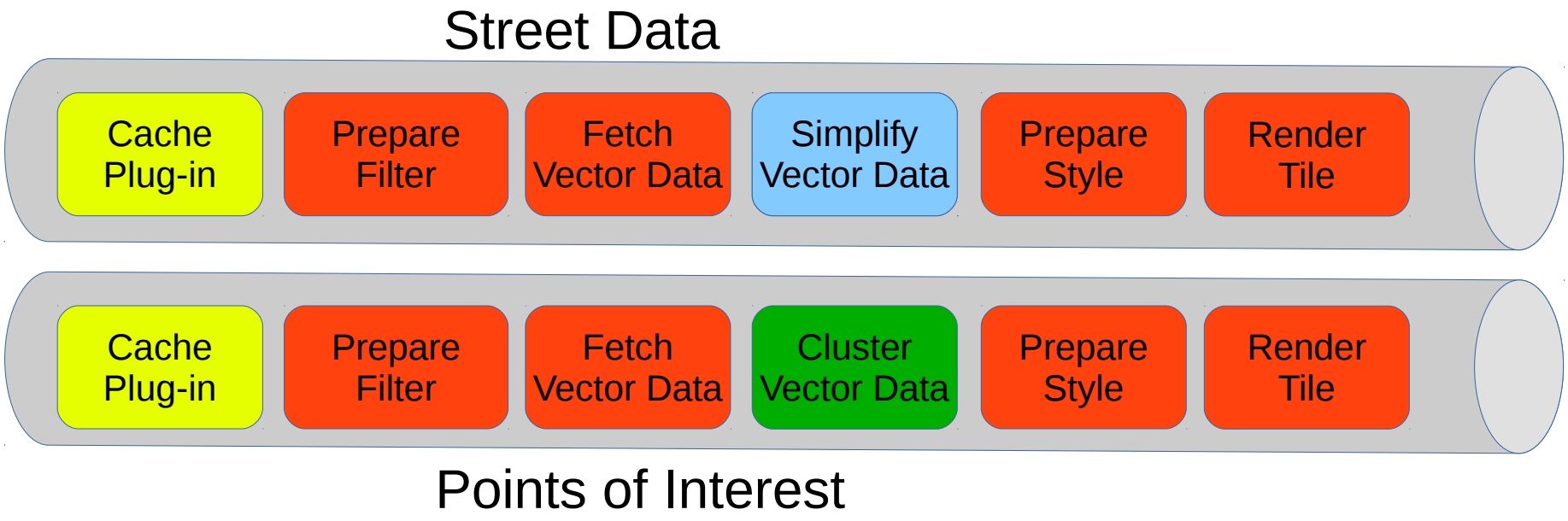
1 java interface to implement

Object feature

your custom java object

Geomajas & Spring

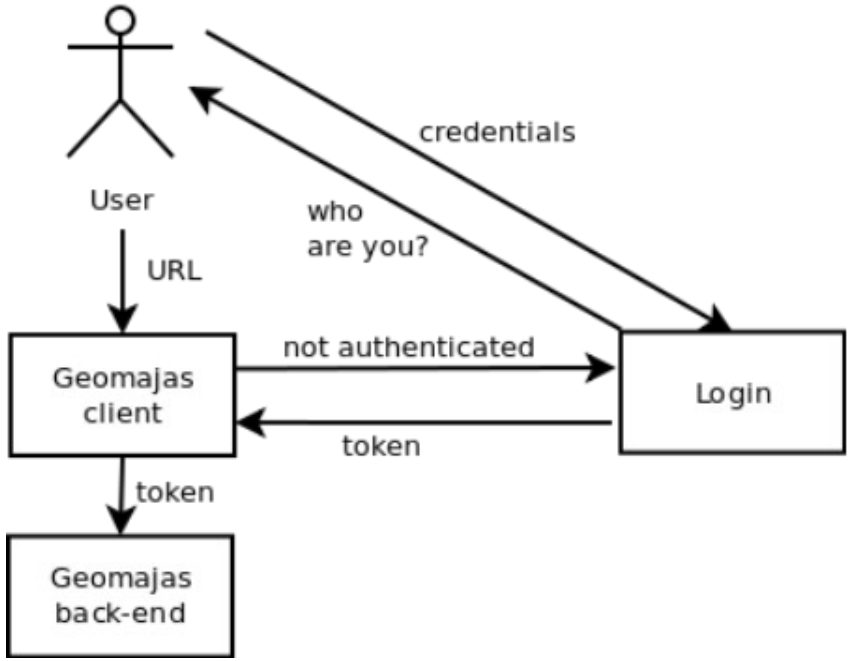
- Spring is everywhere!
- Example: pipelines in services
  - VectorLayerService.getTile(...);



Security in Geomajas

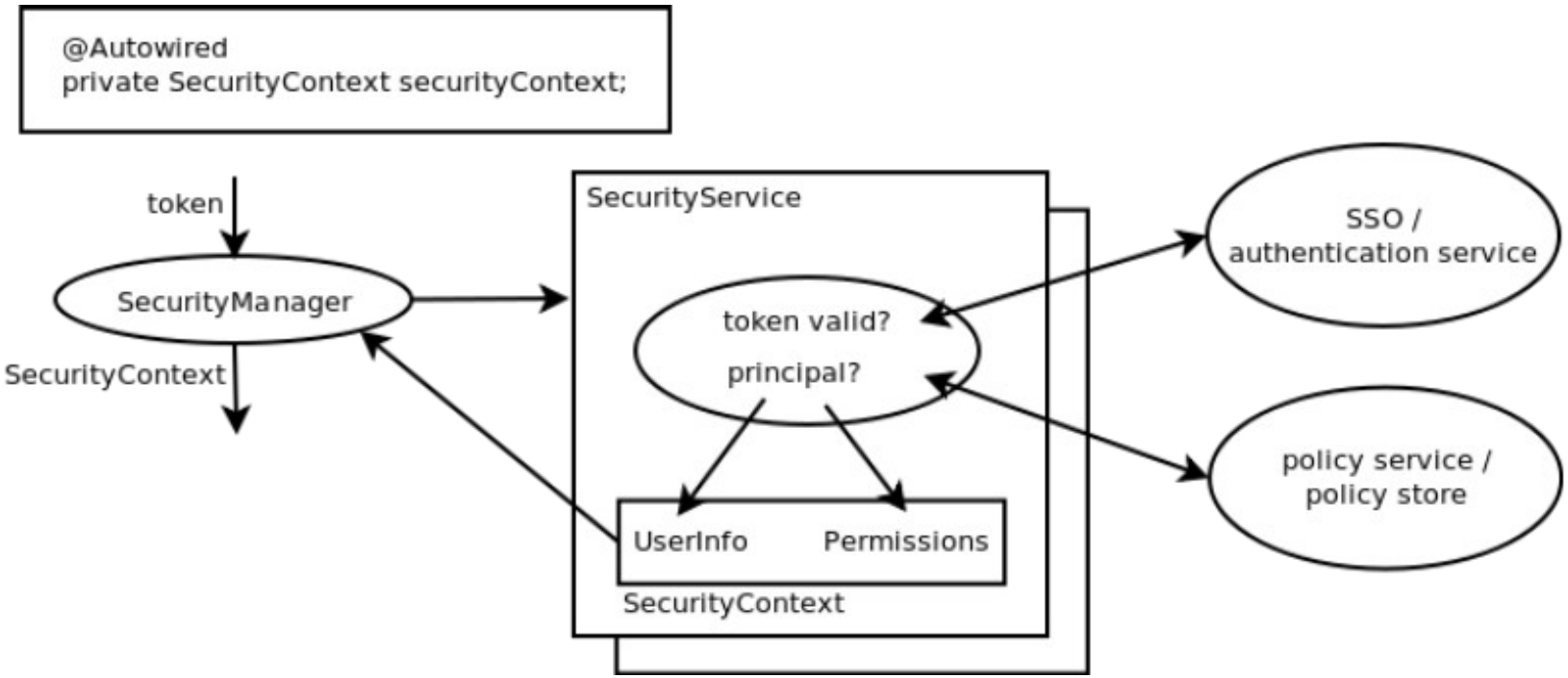
Authentication

- Option A: Use Spring \o/
- Option B: Different system
  - Login is external / Single Sign On
  - Application does not know credentials



Security Context

- Token-based authorization
- Allows access to policies
- Server-side filtering & authorization



Authorization – Layer Policies

- Area (CRUD)
- Individual features (CRUD)
- Individual feature attributes (CRUD)
- Custom application policies

- How?
  - Spring: Implement 1 interface & configure app to use it

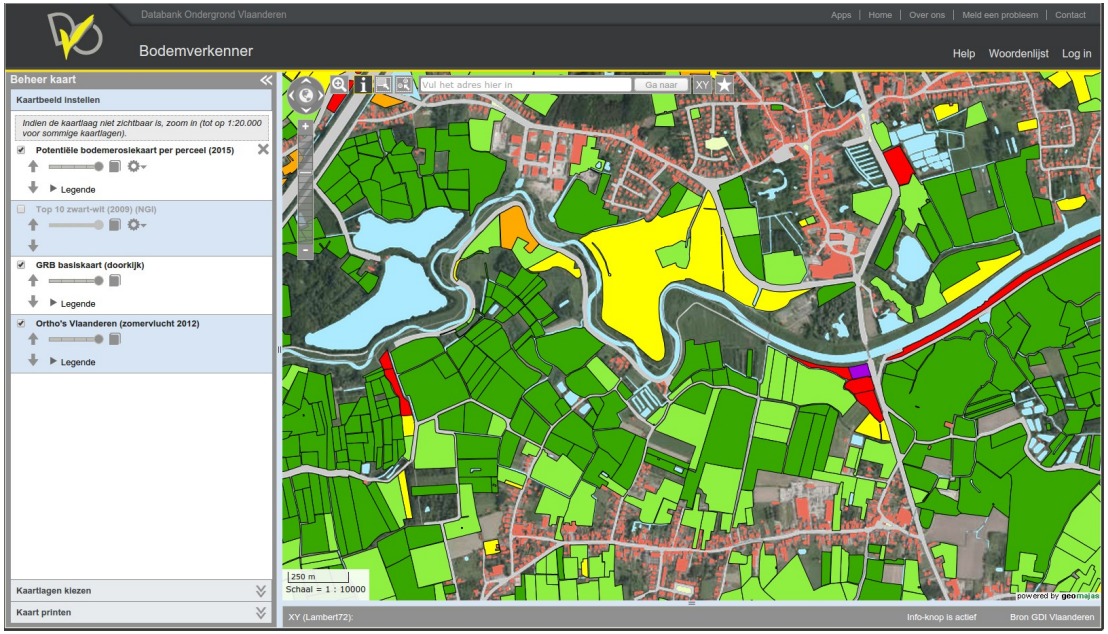
Authorization – code

```
public interface FeatureAuthorization extends BaseAuthorization {


boolean isFeatureVisible(String layerId, InternalFeature feature);

boolean isFeatureUpdateAuthorized(String layerId, InternalFeature feature);

boolean isFeatureUpdateAuthorized(String layerId, InternalFeature
orgFeature, InternalFeature newFeature);

boolean isFeatureDeleteAuthorized(String layerId, InternalFeature feature);

boolean isFeatureCreateAuthorized(String layerId, InternalFeature feature);
}
```

Demo

Demo

- Example application:
  - BodemVerkenner (Soil Explorer)
  - https://www.dov.vlaanderen.be/bodemverkenner



Q&A

Questions?

- Mail me: jan.demoerloose@geosparc.com
- Twitter: @geomajas

- www.geomajas.org
- www.geosparc.com