ONLINE

MAP TRANSFORMATION

Exam: Geographic Information Systems
Author: Marco Livraghi
Academic year: 2015/2016
Student Id: 835931

INDEX

1. INTRODUCTION

The goal of the project is to implement a web application for online geographic transformation of vector maps according to a base map used as a reference.
It is based on an algorithm developed in the PhD thesis of eng. Giorgio Zamboni that, given a few pairs of points in two similar map, can calculate many homologous points.
By using the homologous points is possible to apply the best transformation to the goal map in order to align it to the reference one.
The quoted algorithm has been developed some years ago by means of a Java Desktop software.
The main activity for this project was to rethink the user interface of the software as a web application, even in a responsive mode for tablet device.
The technology adopted was Java Spring Framework for the server side and AngularJs for the client application.
Among these it was necessary to use GeoServer, Leaflet and Bootstrap, plus some of the common AngularJS components.

2. Context

The developed application has a common architecture and it only needs a server to be runnable.
It has been developed with Java and AngularJS so it's completely cross-platform and the server does not have specific requirements.
This application needs an instance of Geoserver to communicate with, that allows the user to load own shape files, get maps, apply transformation and publish layers.
It has been tested with version 2.9 of GeoServer, the latest available at the moment.

2.1 Spring Framework

The server module has been developed on top of the Spring framework.
Spring framework is a Java application framework and inversion of control container.
It contains many features for building web application and it can be seen as a replacement of the Enterprise JavaBeans model.
The first version of this framework was released in 2002.
A core part of the Spring Framework is the inversion of control (IoC) container, which provides a way for managing java objects using reflections: this container manages the objects lifecycle and it's responsible for creating, initialize and configure them.
The objects can be configured with XML files or with Java annotations that provide the configuration information needed to create them.
there are two ways of obtaining objects: dependency lookup and dependency injection.
The dependency lookup is a pattern where the caller ask the container objects with a specific name or of a specific type. On the other hand, dependency injection is a pattern where the container passes objects by name to other objects, using constructors or factory methods.

2.2 AngularJS Framework

Front-end module of the application is entirely developed with AngularJS.
Angular is a web application framework developed mainly by Google but still open-source. It addresses many challenges in developing single-page applictions and simplify development and testing of such applications.
The framework adhers at a client-side MVC architectures.
The html pages are enriched with specific custom tag attributes that AngularJS engine reads and interpretes.
Angular is the frontend framework of the MEAN stack, composed of MongoDB, Express, AngularJS and Node.js.
Angular separated two main concepts: user interfaces and application business logic.

For the first one AngularJS belief in declarative programming while the business logic should be written in imperative programming.

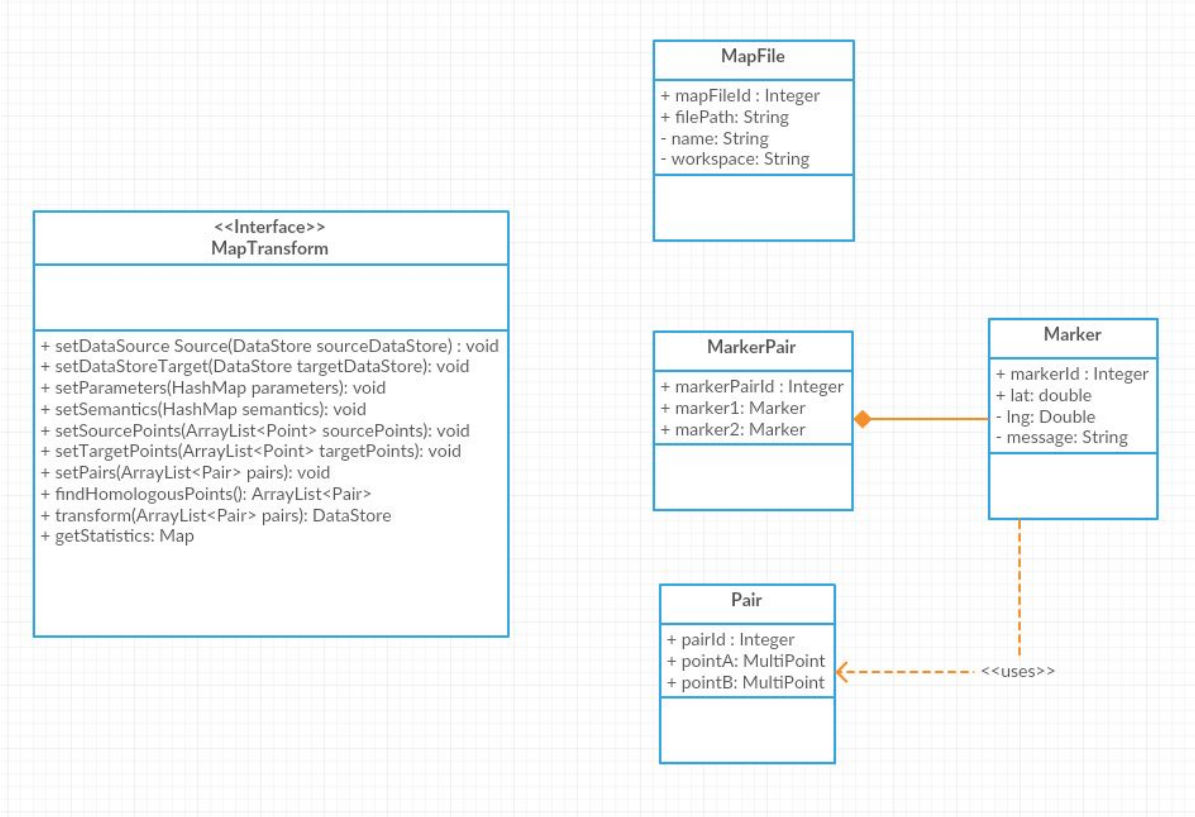
To satisfy this constraint the framework adapts and extends the HTML markup language decoupling the DOM manipulation from the business logic.

The geographic library adopted to render data is Leaflet, thanks to its simple interface and adaptability to this context.
Leaflet allows to render maps in an HTML page together with the layer and marker management and the basic functionalities over them like the drag&drop.

3. DIAGRAMS

The domain of the application is not very complex. It consists in a few class the allow to represent all the requirements.

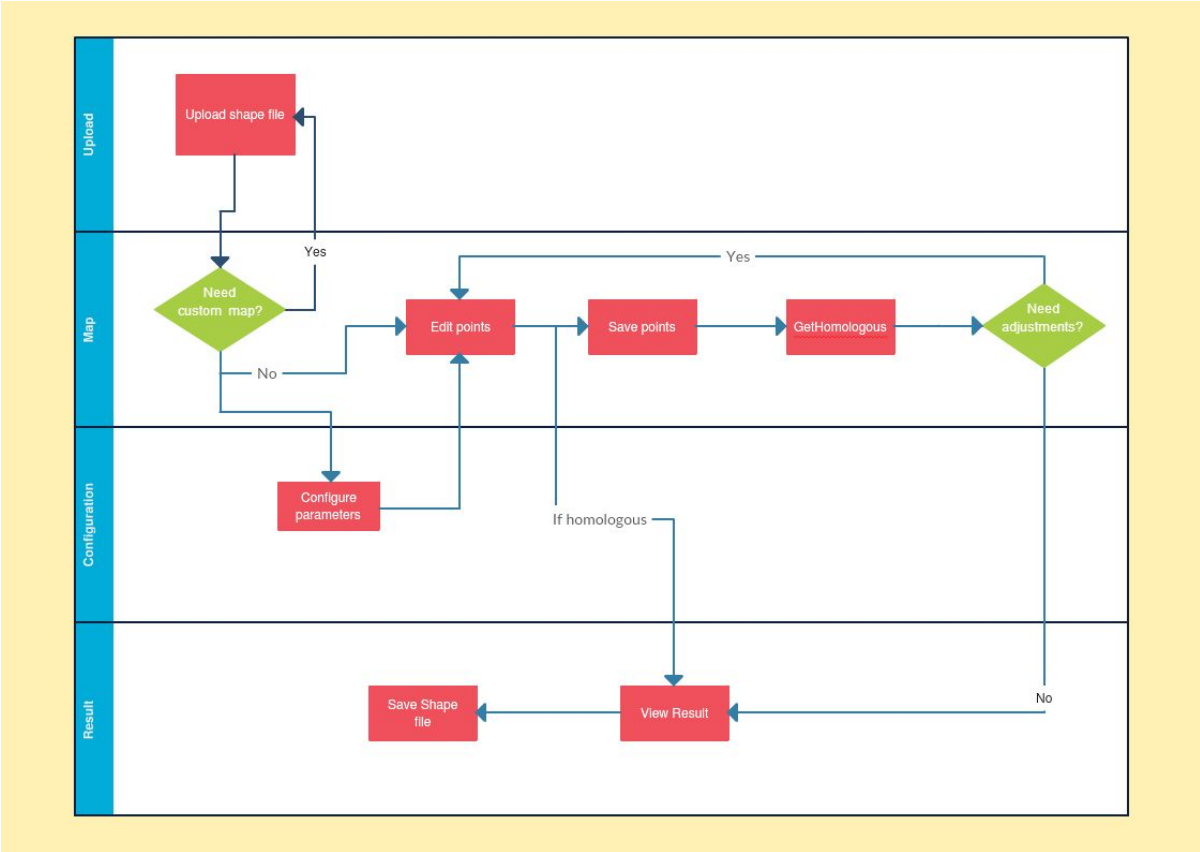
3.1 Class Diagram



- MapFile: this entity represent a map loaded by the user on geoserver by means of its specific shape file. It is stored together with the workspace in order to guarantee the chance in the future to work on different projects.
- MarkerPair: this is the key entity of the domain as it represents a couple of points on the two chosen maps. Both marker inserted by the user and homologous couple points are instance of this class. It's a view class as it is not store on database. The corrispondent class managed by the DBMS is Pair in order to guarantee the full compatibility with the used algorithm.
- Marker: Another view class used to represent a marker in the map, with coordinates and a message to show on tooltip.

The other core interface of the project is MapTransform. It contains some methods to set up the configuration parameters, a method to find homologous points and finally two methods to transform the original map and get statistics of the applied transformation.

3.2 Flow Diagram



This diagram describe the flow that the user should follow in the web application.
First of all he can upload his own shape files to GeoServer and configure the parameters of the algorithm.
Than the two maps can be updated and the user can add his own layers, before selecting a few pairs of points. Once the user is satisfied can save the selected pairs and run the algorithm for researching homologous points.
This points are retrieved and shown in the map, where the user can still apply some changements and discard some of them.
At this point user can run the final transformation to get the result with the associated statistics.

The final transformed map can be saved as shape file and it will be published as a GeoServer layer.

4. IMPLEMENTATION

In the first phase of the implementation the focus was on showing the map. This part has been developed with Leaflet that uses an OpenStreetMap basis plus a layer customizable by the user.
It has been implemented the possibility to load a shape file directly to the configures GeoServer, by means of the Geoserver REST API. The library adopted for this scope is GeoServerManager, a Java library that wrap the geoserver's api.
It is even possible to load on the leaflet map a layer directly from an instance of geoserver, without loading an own file.
The second phase of the developing process was on the configuration parameters. The algorithm uses two kind of parameters:

- A standard set of settings like mean, sigma in order to configure the algorithm
- Semantics

For semantics parameters association lists were developed, that are composed of association between the two compared maps. The maps are queried by means of the JSON format and it is possible to retrieve the attribute's value of a certain map attribute.
After that the user can associate different values of the two maps to help the algorithm in the finding process of homologous points.

The last part of the developing was about implementing a fake mechanism to simulate the algorithm and retrieve a map as a result.

It has been developed even the "snap" mode: when the user submit the points to the server it calculates the closest one in the map, by running a query on the database table and retrieving the point with the lower distance from each one selected by the user.

5. DEVELOPER'S GUIDE

The code is available on a GitHub repository at https://github.com/mrclivraghi/GisProject.

The project has been developed with Eclipse IDE and requires Node.js (https://nodejs.org/en/), a Geoserver instance (http://geoserver.org/) and bower (https://bower.io/).

Another crucial component is Gulp, that can be installed following the detailed guidelines at http://gulpjs.com/.

Repository can be cloned and opened from Eclipse a new Maven project. Once this step is completed it can be necessary to run a "mvn build" from command line or from the IDE. To install the client dependency is sufficient ro run "bower install" from the root of the project.

GeoServer must be configured to accept Cross origin requests. To achieve this it is necessary to add the following filter in the web.xml file of the geoserver webapp.

```
<filter>
  <filter-name>CorsFilter</filter-name>
<filter-class>org.eclipse.jetty.servlets.CrossOriginFilter</filter-class>

</filter>
<filter-mapping>
  <filter-name>CorsFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

The url pattern may be customized based on the developed voluntees.

Properties of the project are defined in src/main/resources/application.properties. Here it is possible to change database setting and geoserver link.

Server application can be easily run with the GisProjectApplication.java in src/main/java. It is the embedded Spring Boot application and will run a tomcat on the port 8080, if not otherwise specified.

There is also the possibility to add the project to a separate server instance. With maven developer can build the related war, with “mvn build” command.

The client application is managed by gulp. Developer need to move to the project folder and run the command “gulp serve”. This will run the client application on the port 3000, accessible via web.
Gulp support a deploy and build system, runnable with “gulp deploy” and “gulp build”. In case of deploy developer needs to properly configure his production environment in /gulp/deploy.js file.

6. FUTURE WORK

The project can be extended with few features.
The most important in order to get the project ready for a production environment is a login mechanism to allow the user saving their work on different projects and retrieve them at any time: the application should be multi tenant.
The statistic of the result transformation may be analyzed in charts and stored in a table in order to be available for future data analysis.
Having used Bootstrap framework the application is almost responsive: it would be interesting to transform it in a mobile application. Ionic framework would be an easy way and it is possible to get the result in a short time.

7. CONCLUSIONS

The implementation has been really helpful to understand in detail all the mechanism that take part in the GIS environment.
By developing a full stack solution it was possible to analyze all the different aspects, from the Leaflet viewer to the Geoserver's REST api and the installation of a Geoserver's instance.
The Geoserver REST api are not well documented, this has been the trickiest part of the project
It would have been interesting appling the real algorithm but it was not possible.