ATM Search Application

The solution is based on Spring Boot and Java 1.8. IntelliJ IDEA was used as a tool.
It should run on Java 1.8 and Tomcat 7, I've tested it on http://apache.volia.net/tomcat/tomcat-7/v7.0.67/bin/apache-tomcat-7.0.67-windows-x64.zip server under Windows and it worked fine.

Build

To build the application you need to have Maven 4 installed. Build steps are:

1. Unzip the ATM.zip to desired directory, this is your project root (<project_root>) directory.

2. Go to the project root directory, check if the pom.xml file is there.

3. Run 'mvn clean install' command and check the 'target' directory if the 'atm-1.0-SNAPSHOT.war' file created.

4. Deploy the .war file to the Tomcat server. The application's web page should be available at the http://<host:port>/atm-1.0-SNAPSHOT/

or

5. Go to project root directory and run 'mvn spring-boot:run' command. The application web page should be available at the http://localhost:8080/

Configuration

The configuration of application is placed into application.properties file which is under <project_root>/src/main/resources/ directory. There are number of properties to set up: logging level, security properties which are defines authentication for Spring Security and 'atm.locator.url' property which defines the endpoint to request ATM data.

UI

It is built on AngularJs and uses Google Maps API. I guess Angular is sufficient choice if server has REST API.

When you try to access UI you'll be asked for authentication by the browser's default dialog. The user and password from application.properties file should be used, defaults are 'user/password'. This is the only place where the authentication is demanded.

The UI has two sides: left – with the table and right – with the map.

Above left side table there is an input field where user can enter a name of the city to filter ATMs for. Next to the input field there is a button which performs search. If the input field is empty search will return the ING ATM's for all the Netherlands, in other words, ATMs list will remain unfiltered.

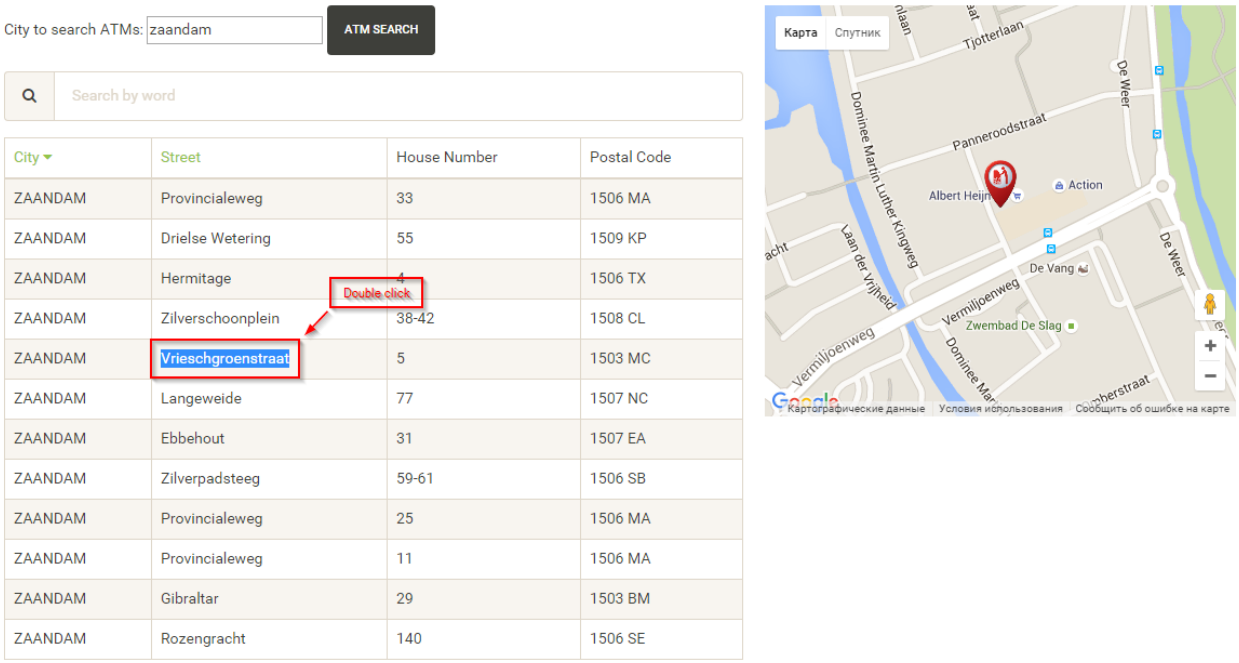
If the search return a result the table will be populated with the ATM' records. As you can see there

are the columns with sorting and the filter field, which filters records by tokens entered.

At the same time markers for the found ATMs should appear on the map at the left.

Double click at the table record will center and zoom map with the clicked ATM. Marker at the map is also shows the details window when clicked.

Please take a look at screenshot:



Server

The server is written in Java and has no database.

I found that amount of data responded from the ING server is not that big and could be processed (actually filtered) in runtime during single user's request. Also, despite the bonus option of using the Camel, I thought not to bring one more dependency to the Spring-based application and used the Spring's RestTemplate to request ING for data.

The application has two layers: controller layer and service layer.

At the Controller's layer you can find the *AtmController* class which exposes the REST API and delegates the requests to the Service's layer.

At the Service's layer there are *IngCommunicationService* (responsible for requesting ING for data) and *AtmService* (responsible for thransforming of DTOs and filtering the data).

There is also *AtmJsonParser* class designed to convert not that valid JSON from ING to DTOs.

The JUnit tests for these services can be found in corresponding packages.