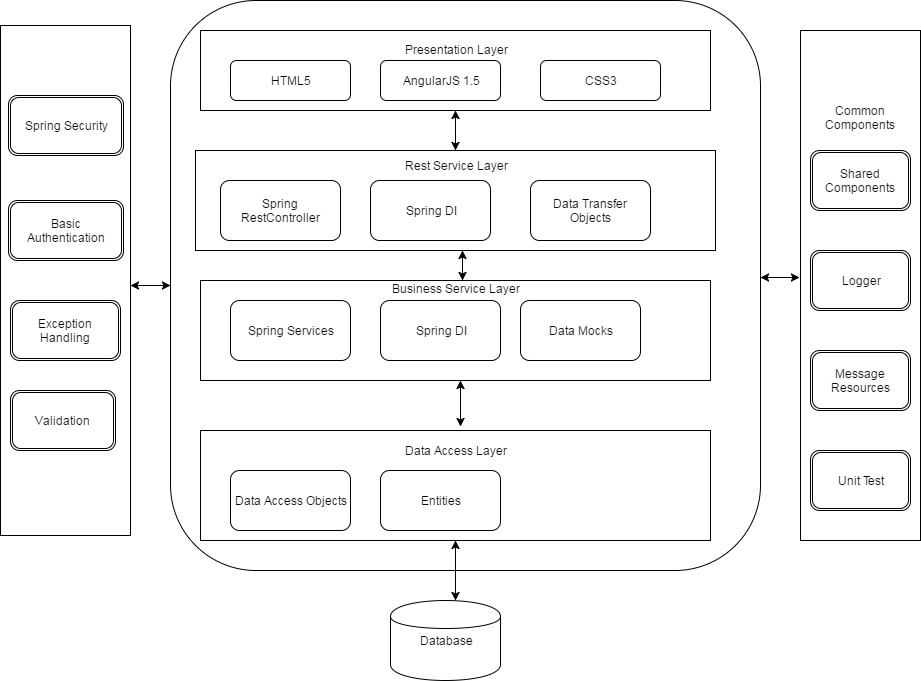
1. Introduction

This document explains the architecture how the application has been visualized and implemented.

It provides step by step instructions on how to deploy this application and the screenshots attached for the reference.

1. System Architecture

This diagram describes the application architecture



1. Technology Stack

The parent pom.xml defines all the libraries and frameworks used to build this application.

The below technology stack is used to develop this application

1. AngularJS 1.5
2. Java 6
3. Spring MVC 4.2.0
4. Spring Rest Service
5. Spring Security 4.2.0
6. Jackson Data binding
7. Junit 4.12
8. Servlet 3.0
9. Tomcat 7 64 bit
10. Eclipse Kepler 64 bit
11. Security

The Sample application is built by using Basic authentication by providing predefined credentials for simplicity such as user name is test and password is test123.

Spring Security has been implemented to ensure the authentication and authorization of user access rights.

The Basic Authentication popup will be displayed once you try to access any resources in the application where user will have to provide the details such as test/test123

Once authenticated, Spring Security will verify the user role to grant access to proceed further.

In this application we have provided the default role to the test user such as ROLE\_USER to access the sample application.

We can find more details about the credentials and the implementation in application-security.xml

Note

The Basic Authentication will be stored by the browser after logged in for the first time.

It is always recommended to open new browser in private mode to test the application every time to ensure the authentication takes place.

1. Deployment

The application is developed as per the requirements given in the requirements document.

We need to unzip the application and extract the testapp.war file.

You can deploy the application war file testapp.war into tomcat deployment folder.

The path of the deployment folder in the tomcat installation is

\apache-tomcat-7.0.77\webapps\testapp.war

After dropping the war file into tomcat, please start the server and verify whether the application is started and ready for testing

The application URL is <http://localhost:8080/testapp/home> and use the port assigned in the tomcat installation for verification.

1. Rest Services

We have developed 2 rest services to provide the details of ATM machines in the Netherlands

They are

1. <http://localhost:8080/testapp/api/locator/atms>

This service is used to return all ATM machine details in the Netherlands

1. <http://localhost:8080/testapp/api/locator/amstelveen/atms>

This service is used to return the ATM machine details based on the city in Netherlands

The application uses the mocked data for simplicity, so you will find only limited data.

The Test data has been provided only for the cities such as amsterdam, amstelveen or others.

We can consume these rest services using any Rest client or Google Chrome.

We need to set the basic authentication details in the client before we access the services.

1. Build the application

The application has been developed using maven build and the version is apache-maven-3.1.0.

We can build the application by triggering mvn clean install in the source directory

The build will create the war file along with the snapshot version and we will have to remove the snapshot version from the generated war file name and keep it as testapp.war

[INFO] Packaging webapp

[INFO] Assembling webapp [testapp] in [C:\Users\C38688\ws\testapp\testapp\target\testapp-0.0.1-SNAPSHOT]

[INFO] Processing war project

[INFO] Copying webapp resources [C:\Users\C38688\ws\testapp\testapp\src\main\webapp]

[INFO] Webapp assembled in [246 msecs]

[INFO] Building war: C:\Users\C38688\ws\testapp\testapp\target\testapp-0.0.1-SNAPSHOT.war

Rename testapp-0.0.1-SNAPSHOT.war as testapp.war and deploy in the server

1. Web Application

After you have deployed the application, you can access the application using this URL

<http://localhost:8080/testapp/home>

The home page will provide you the search functionality where you can enter the predefined search criteria such as amsterdam, amstelveen and others.

It will list down the appropriate search results in the table below based on the search criteria given and the results will be empty if there are no records met for the given criteria

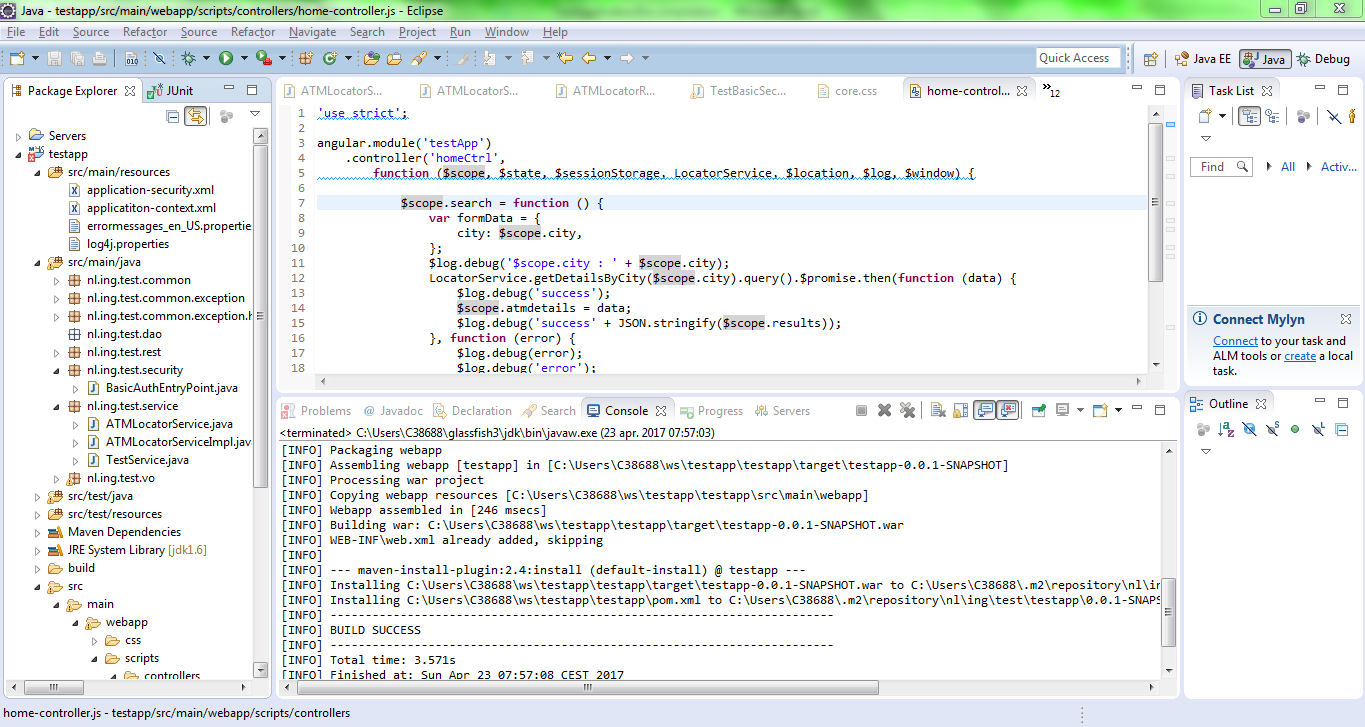
1. Unit Testing

Unit test classes have been written to validate the rest services using RestTemplate framework.

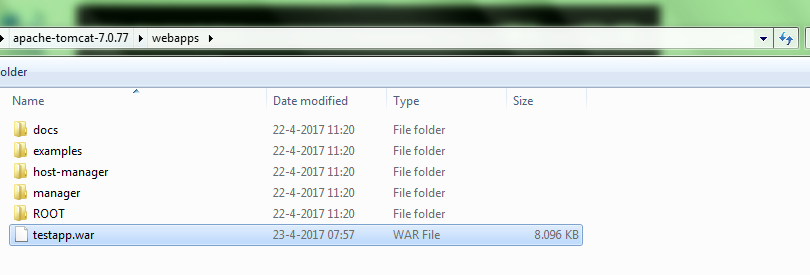
To test the unit test cases, the application should be up and running and the URL of the deployed application needs to be modified accordingly in the test classes.

These unit tests will verify whether the service is running and returning the success status as 200 Successful.

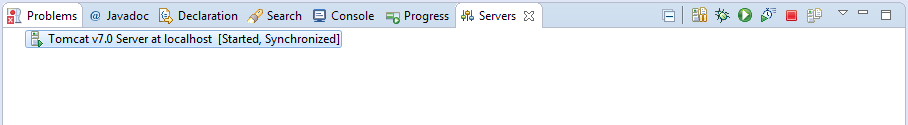
1. Screenshots
2. Application build

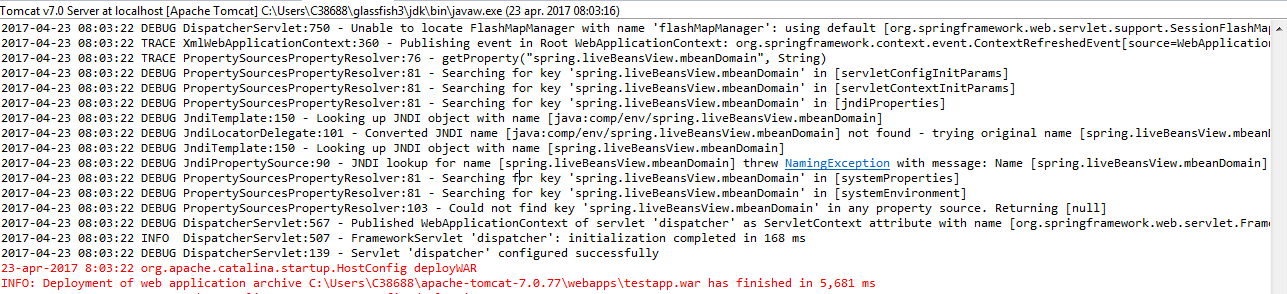


1. Deploy the war file

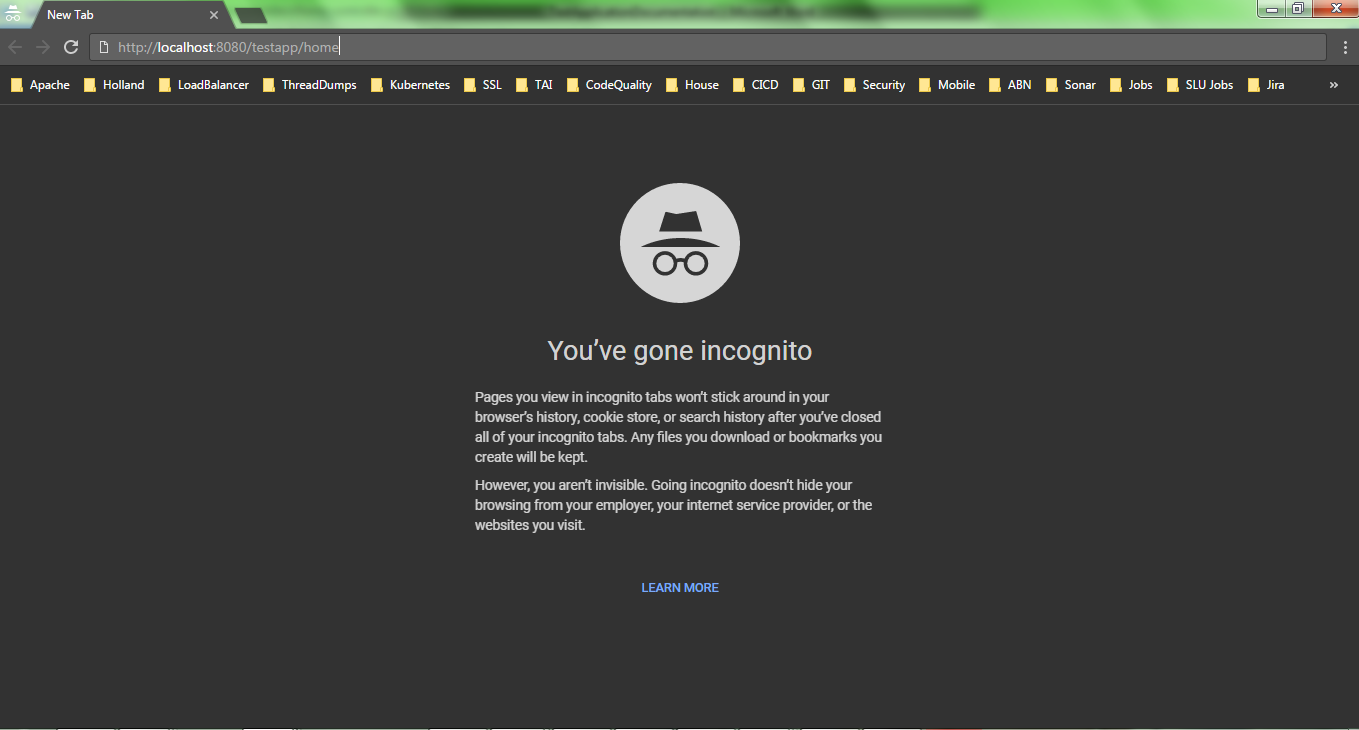


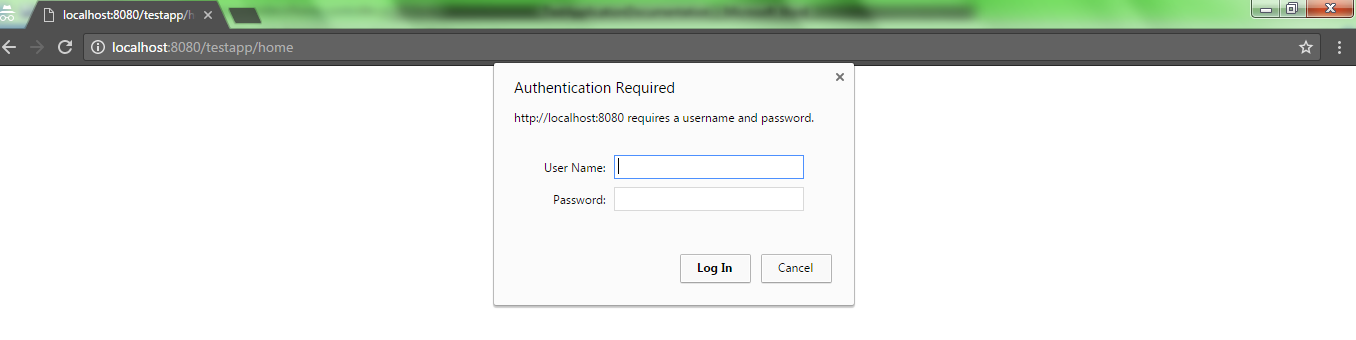
1. Start the server in Eclipse or directly using CLI
2. Deploy the war file

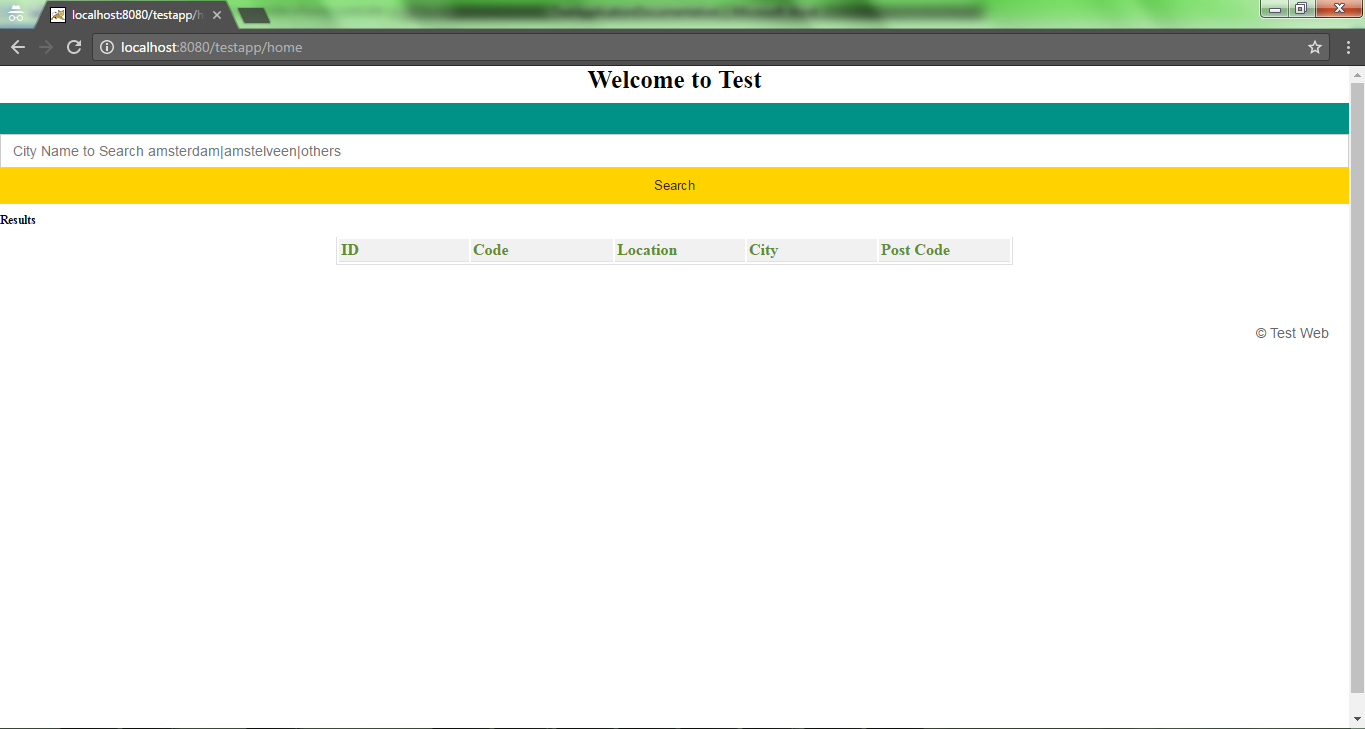


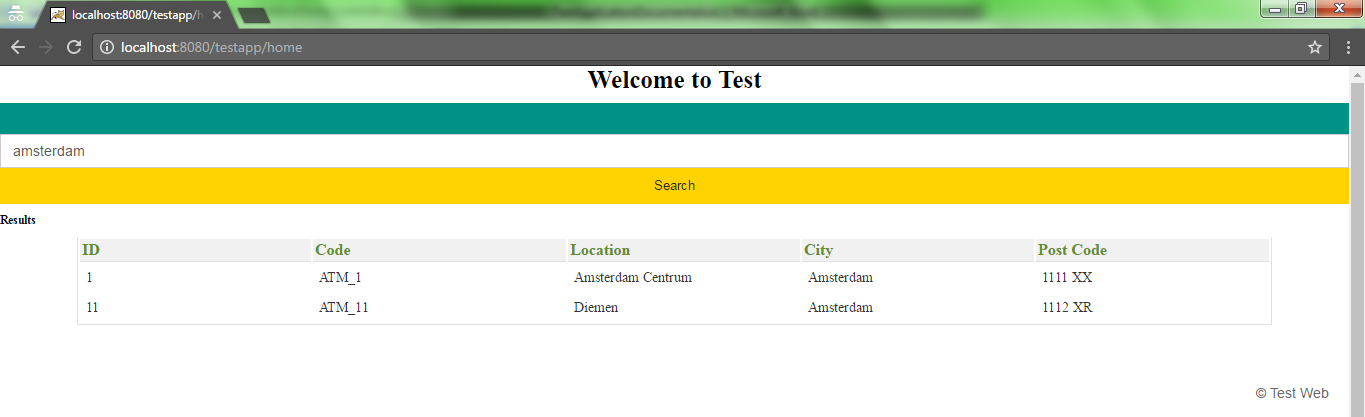


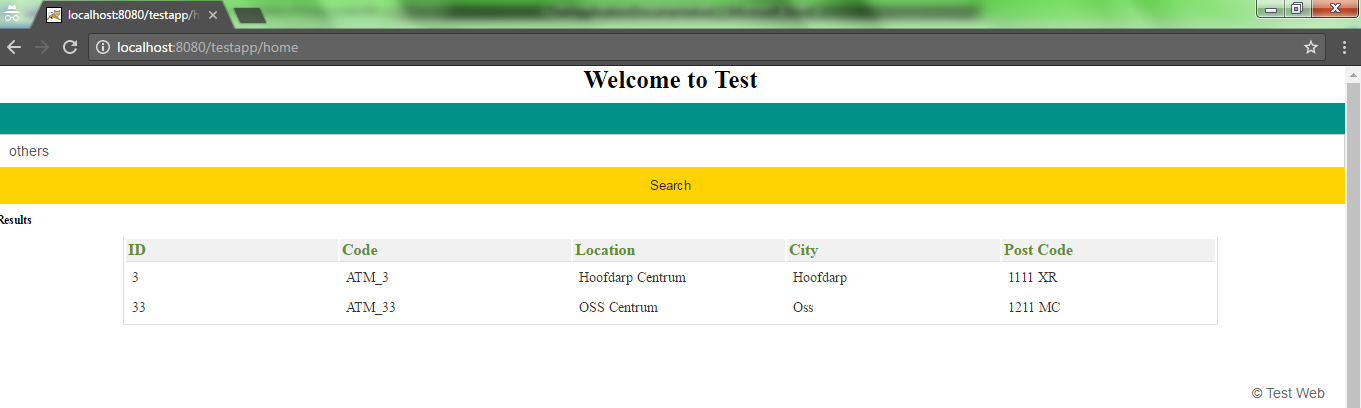
1. Access the application in Chrome Browser In private mode









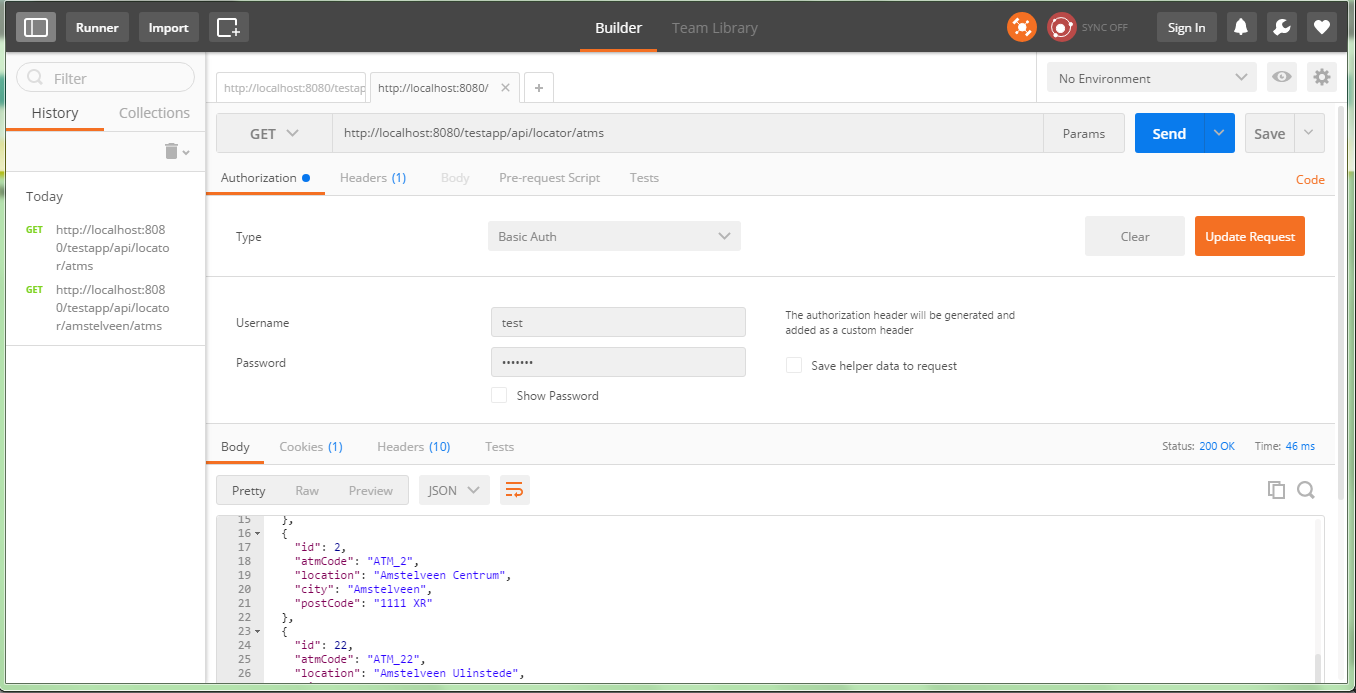


1. Access the rest services using Google Postman

<http://localhost:8080/testapp/api/locator/atms>

Content type : application/json

Basic Auth



<http://localhost:8080/testapp/api/locator/amstelveen/atms>

Content type : application/json

Basic Auth

