



DECEMBER 6, 2016

JAVA ACADEMY

FINAL PROJECT

JAVIER REYES



Accenture Java Academy

1.- Objectives

For the final project of the Java Academy a spring-boot implementation was required. In this implementation had some requirements.

- Provide a page that says hello to the user when he enters his name as a parameter on the URL.
- Create a page that allows user save their Hobbies through a form.
- Create a search page that shows hobbies per user.
- User information should be saved on session, system file or database.

All the source code required to test the application is available in the at <https://github.com/opas350/Final-Project-Java-Academy>

2.- Project modules

The first function to verify is the one that appears at the root(localhost:8080/)

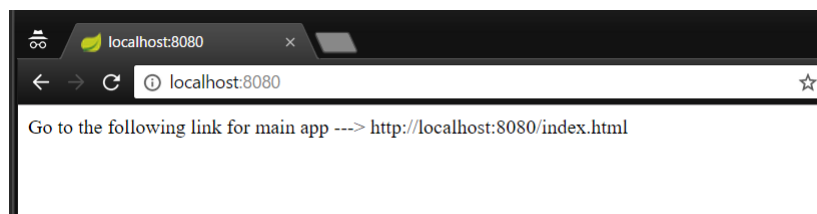


Fig.1 - Message at localhost:8080/

This page just shows the link for the main application, just in case if someone forgot where it should go to test the application.

But first let's try the personalized welcome page on Fig.2, the URL is

`localhost:8080/Hello.html?first=XXXX&last=YYYYY`

Where the X's and Y's replace the first and last name of the user, also is important to mention that last parameter is optional as shown in Fig.3.

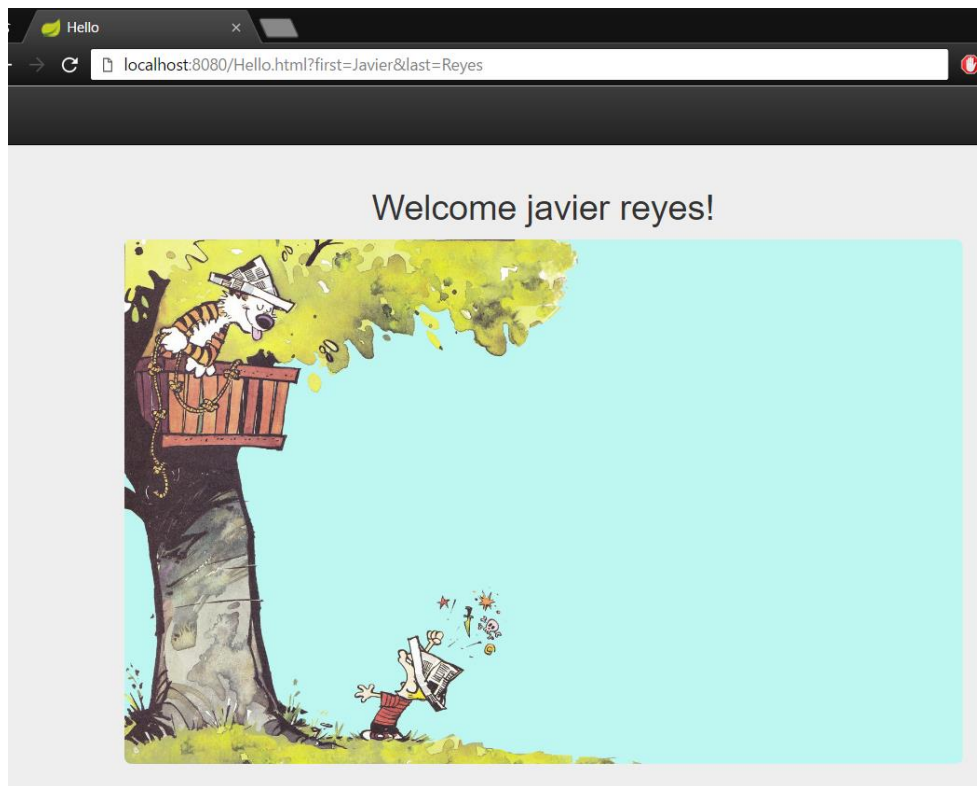


Fig.2 - Personalized Welcome page.

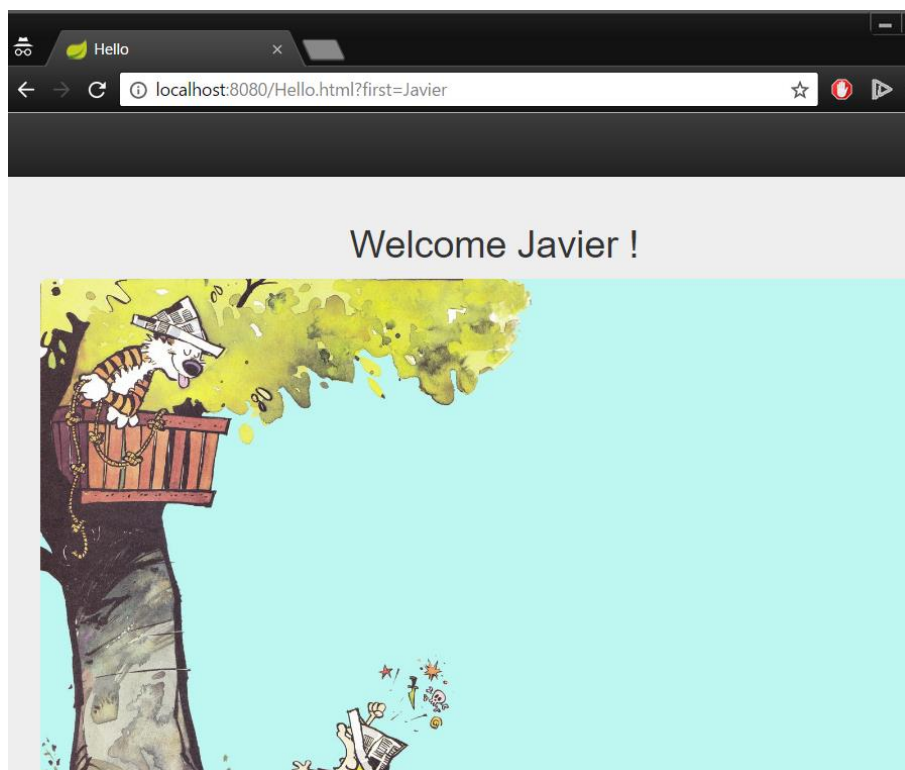


Fig.3 - Personalized page with only one parameter.

Now the main app in which the different views can be accessed at the link `localhost:8080/index.html`

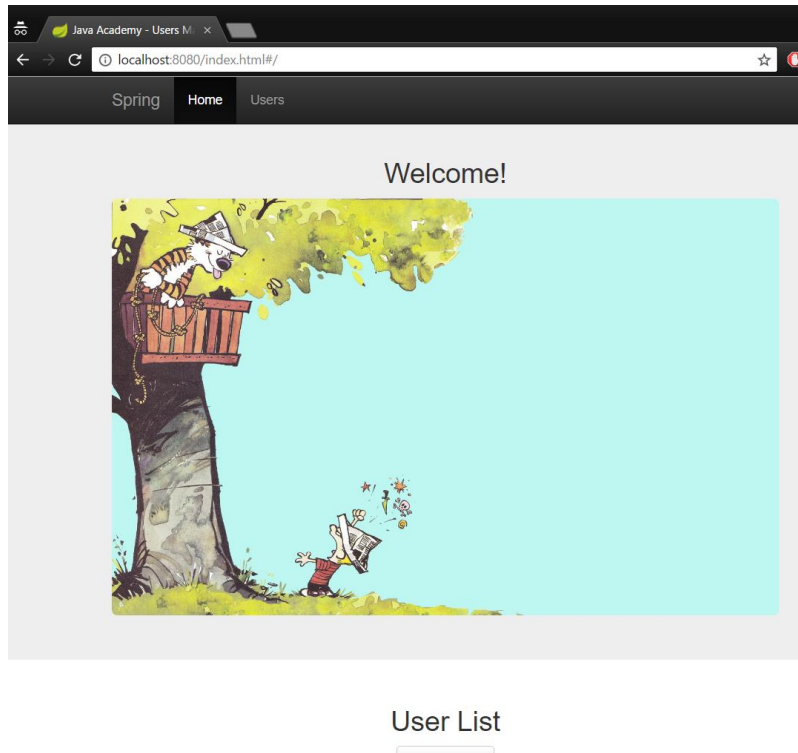


Fig.4 - Main app.

At first glance it appears as the previously shown pages, but this view will add as the main app were all the functions implemented can be accessed. By clicking the tab of Users the page will load the document where the added users are shown.

As the Fig.5 shows there are currently no users on the database because this is the first time the application is executed.

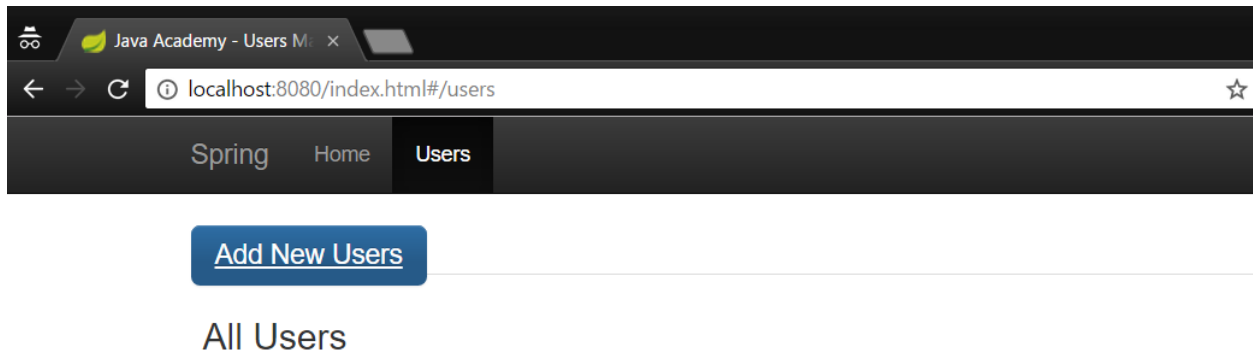


Fig.5 - Users tab.

The next step to test if everything is working properly is to add a new user, after clicking the Add New Users button the view will change as the Fig.6.

A screenshot of the same web browser window, but the URL is now 'localhost:8080/index.html#/users/new'. The 'Users' tab is still active. The form contains the following fields: 'User Name' with the value 'opas350', 'Hobby #1' with 'Read', 'Hobby #2' with 'Listen music', 'Hobby #3' with 'Videogames', 'Hobby #4' with 'None', and a 'Message' text area containing 'I like dogs'. A blue 'Save' button is located at the bottom of the form.

Fig.6 - User information form.

The form for new user will appear, this form will allow the user creation in which he can fill the different fields as he like and even leave some blank, for this test the form was filled with the information as shown on Fig.6 then clicking the button Save.

Then the view will return where all the users are displayed and in this case, it will now show the new user, the view should be like Fig.7.

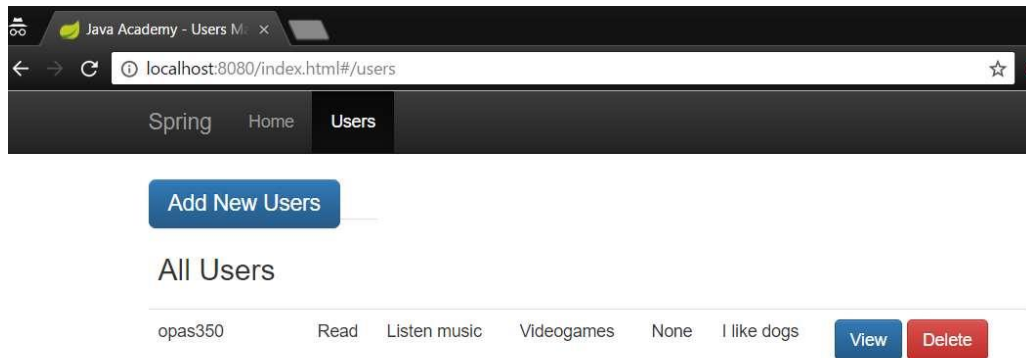


Fig.7 - Users view after adding user.

The next function to test will be to view and update the user information, click the view button and the view should display the user information table now with the labels for each field.

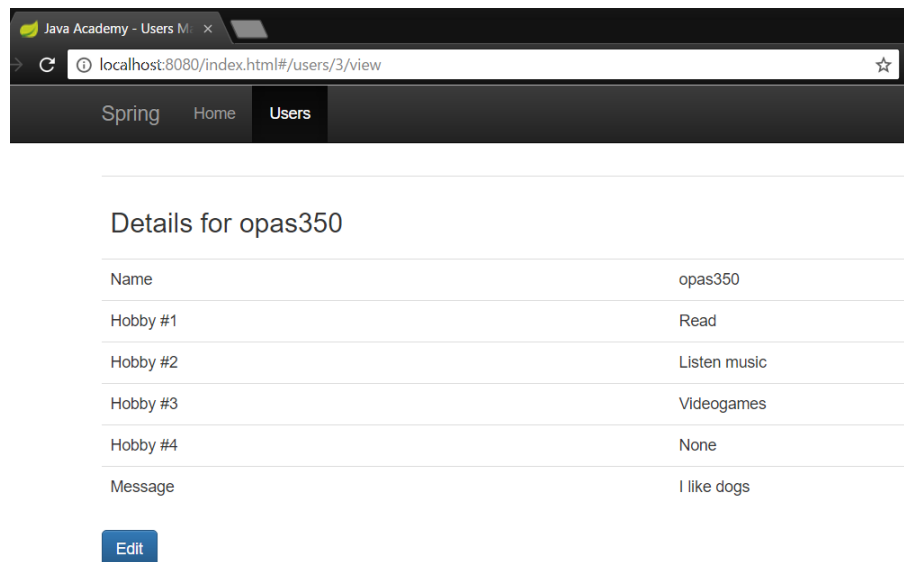


Fig.8 - User information.

After checking the information was correct we proceeded to verify the update/edit information. In this test, we will just update the Hobby#4 parameter as Fig.9.

Java Academy - Users M: x

localhost:8080/index.html#/users/3/edit

Spring Home Users

User Name opas350

Hobby #1 Read

Hobby #2 Listen music

Hobby #3 Videogames

Hobby #4 Cooking

Message I like dogs

Save

Fig.9 – Updating user information.

Java Academy - Users M: x

localhost:8080/index.html#/users

Spring Home Users

Add New Users

All Users

opas350	Read	Listen music	Videogames	Cooking	I like dogs	View	Delete
---------	------	--------------	------------	---------	-------------	------	--------

Fig.10 – Verifying the updated information.

The displayed information should look similar to the figure above, in order to test the delete function another user was added, just as shown on Fig.11.

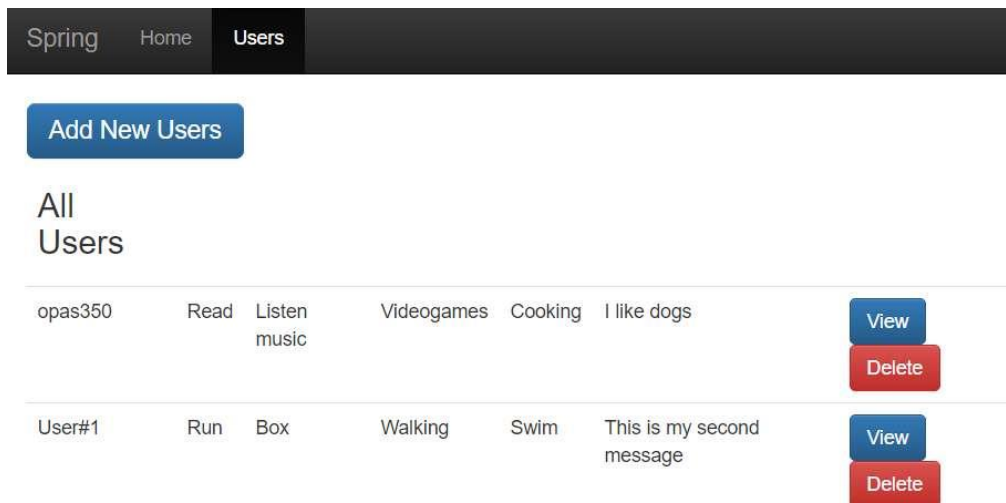


Fig.11 – Dummy user added.

Click the Delete button and a confirmation toast will appear as the Fig.12 shows, after clicking ok, we will be like Fig.7 again.

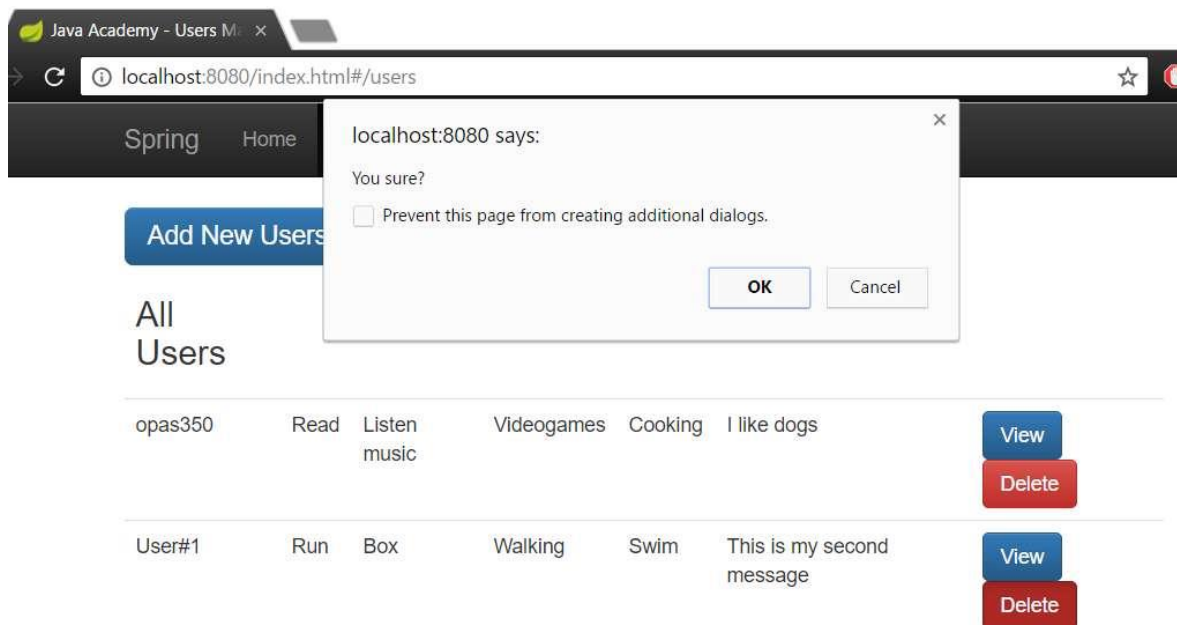
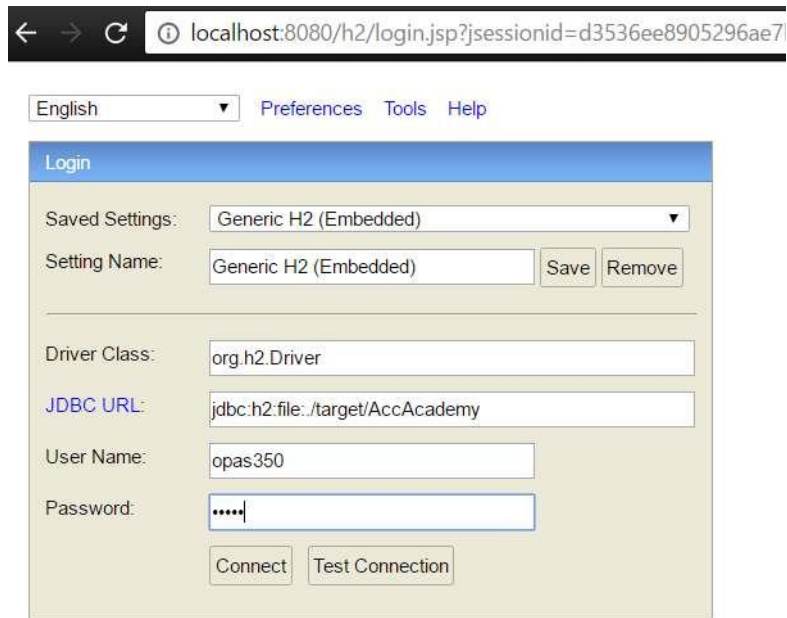


Fig.12 – Deleting user.

3.- DB Console

The database is created using SQL and managed by H2, this means that the H2 console is available for the user if the direct manipulation is required or desired.

The link for the console is <http://localhost:8080/h2/>



The screenshot shows a web browser window with the address bar displaying `localhost:8080/h2/login.jsp?jsessionid=d3536ee8905296ae7`. The page has a navigation bar with a language dropdown set to "English", and links for "Preferences", "Tools", and "Help". The main content area is titled "Login" and contains the following fields and buttons:

- Saved Settings:** A dropdown menu showing "Generic H2 (Embedded)".
- Setting Name:** A text input field containing "Generic H2 (Embedded)", with "Save" and "Remove" buttons to its right.
- Driver Class:** A text input field containing "org.h2.Driver".
- JDBC URL:** A text input field containing "jdbc:h2:file:./target/AccAcademy".
- User Name:** A text input field containing "opas350".
- Password:** A text input field with masked characters ".....".
- Buttons:** "Connect" and "Test Connection" buttons at the bottom.

Fig.13. Login configuration.

To connect with the console, enter the information displayed on Fig.13. username: opas350 and password: 12345, these parameters were configured on the application.properties file. Once on the console the user can manipulate the DB using SQL commands.

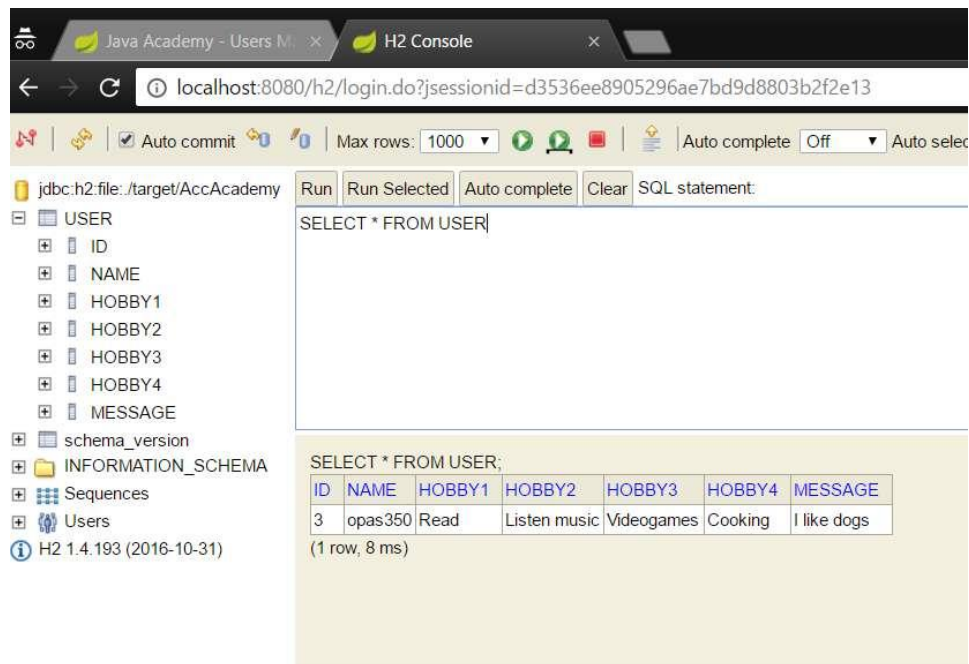


Fig.14 – Example of H2 Use.

The DB implemented stores all the information and is even available after the user exits the application, and since the file is stored in on the target folder is available is the spring instance is initialized later but is not recompiled.

4.- Further work

This project was my first approach to different tools like maven and spring, so there is so much more that could be improved or implemented, but for lack of time and experience there wasn't more for now, but I expect to continue working on this first example to make it better.

The first thing that I'll be implementing is this guide to the main project, to make it more accessible.

5.- References

As I said this was my first time using and integrating a project of this kind, so this could not be possible without some online references, courses and samples likes the one enlisted below.

- Spring-boot app: <https://app.pluralsight.com/library/courses/spring-boot-first-application/table-of-contents>
- Fundamentals of Angular: <https://app.pluralsight.com/library/courses/angularjs-get-started/table-of-contents>
- Spring Fundamentals: <https://app.pluralsight.com/library/courses/spring-fundamentals/table-of-contents>