Web Architectures - Second assignment

Claudio Facchinetti
<claudio.facchinetti@studenti.unitn.it>

Abstract

The report describes the process of development of the second assignment for the course "Web Architectures" held at the University of Trento in the academic year 2021/2022.

In particoular it is described the problem, the technical solutions that were adopted in order to solve the assignment.

1 Introduction

During lectures it has been discussed of how it is possible to setup a web server using Java technology, and in particular Java Enterprise Edition. Later it has been discussed that it would have been better if the HTML code was not integrated in the Servlet code, introducing the JSP and the MVC pattern and other components that could have been used in order to make our coding much smoother and organised.
In order for us to proof, both to ourselves and to the instructor, that we understood the concepts presented we have been asked to create a web application with the following requirements:

- the application has three type of users: guests, authenticated users and an admin user;
- a guest user can only perform the login operation with a username-password pair becoming an authenticated user;
- an authenticated user can access the list of rooms, create a new room, joining a room and read/write message in/from a room;
- the admin user has can perform any operation an authenticated user can perform while also being able to create new username-password pairs.

This web application had to be developed using the concepts of servlets, JSPs, beans and filters while being compliant with the MVC design pattern.

2 Solution

In this section there are presented all the problems we were asked to face, and the solution(s) adopted to solve them.

2.1 Model-View-Controller (MVC)

The code has been structured in order to be compliant with the MVC pattern; this structure is also visible in the packages that are present in the project.
The `model` package contains the classes that represent the "stored" data, which is the data on top of which the web application operates. This data is stored in the `ServletContext`, as multiple components need to access it.
The `bean` package contains the classes that represent the data the JSPs need in order to render properly. In this case the combination of JSPs and beans they access is the `View` of the MVC pattern.
The `servlet` package contains the classes that are responsible of the business logic, for example adding a new room and producing a message telling whether the operation was successfull or not. Using a more technical language we can say that the servlets receive requests from the views, manage them performing any operation needed and then produce a new view as a response. In this case a servlet it a `Controller` of the MVC pattern.

2.2 Session management

In order to solve the well-known "State problem" of HTTP, I decided to go with the plain `HttpSession` mechanism that JavaEE provides.
In particoular when a user successfully a particular bean is saved in the user session, so that it can be used by the views to know the username and its role; the only thing the user know is a pseudo-random string value which is his/her identifier.
Upon performing the logout operation, the user session is immediately invalidated.

2.3 Access control

Almost every requirement of the project requires some kind of access control mechanism in place; for this project I wen straight with the `Filter` class that were provided by JavaEE.
The first filter is called `AuthenticationFilter` and as the name suggests it is a filter which checks

whether the user is authenticated or not, namely if he/she has a valid session; if the user is not authenticated than it is redirected to the login page to perform the authentication. This filter protects every resource under the `/user` path
The second filter is called `AdminFilter` and it does perform access control checking whether the user is logged in with administrative privileges or not: if this is not the case then the user is redirected to the page showing the list of all rooms. This filter protects every resource under the `/user/admin` path and it is executed after the `AuthenticationFilter`.
The last filter is called `LoginFilter` and it is a convenience filter: if an authenticated user is trying to access the login page, then it is redirected to the page showing the list of available rooms.

2.4 Parameter configuration

In the deployment descriptor there two different parameter which can be configured:

- USERS_FILE: it is the path of the CSV file containing the predefined users (without the admin user). This is defined as a context initialization parameter as multiple servlets need to access it to emulate a database;
- ADMIN_PASSWORD: it is the password of the admin user; this is defined as a initialization parameter of the `LoginServlet` as it is the one responsible for initializing the user map in the `ServletContext`.

2.5 JSP

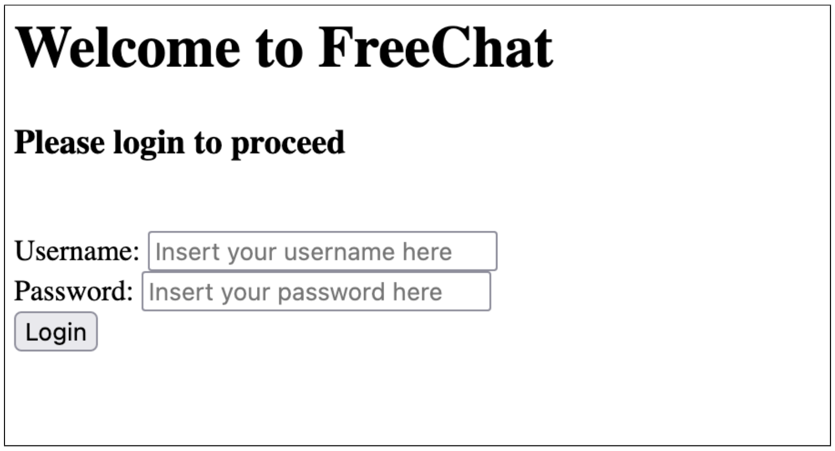
As said before the JSP technology has been used in order to create templates that would have then been "filled" with data coming from the servlets using beans.
In order to have a cleaner code it has been used the `EL` (Expression Language) by importing the `JSTL` Core Library. In would like to point out that it has been used only for presentation logic (like looping over arrays and conditional display) and not for business logic inside the views, which would have broken the MVC pattern.

2.6 Other configurations

In this project there were no error pages implemented, however I did not like the idea of plain Java exceptions returned to the user, like when a JSP page is directly accessed and it requires a bean from the servlet.
In order to solve this minor issue, which was not one of the formal requirements, I decided to setup the page with the list of rooms as error page in the deployment descriptor. In this way when a user tries to access directly a JSP page it will generate an exception and the web server will redirect the user to the "room list" page. For all other cases the error is handled directly by the servlet.

3 Application running

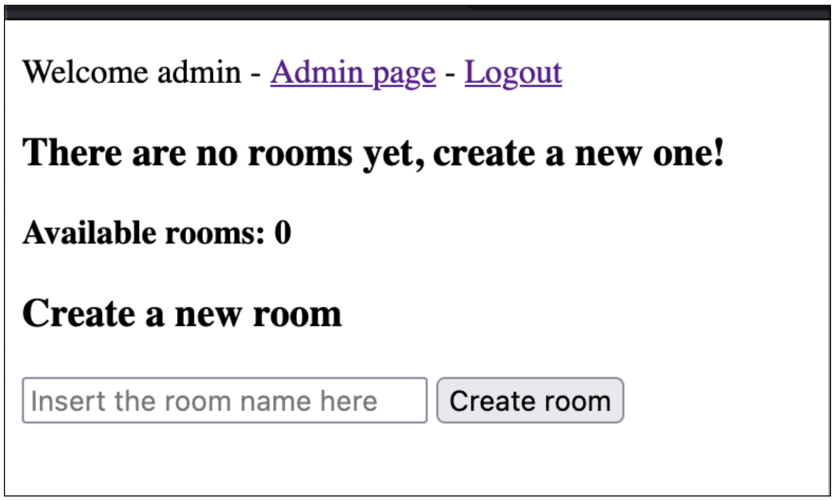
Here there are some screenshots of the application running, showing how it is presented on Mozilla Firefox.


Welcome to FreeChat

Please login to proceed

Username: Insert your username here
Password: Insert your password here
Login

Figure 1: The login page presented


Welcome admin - Admin page - Logout

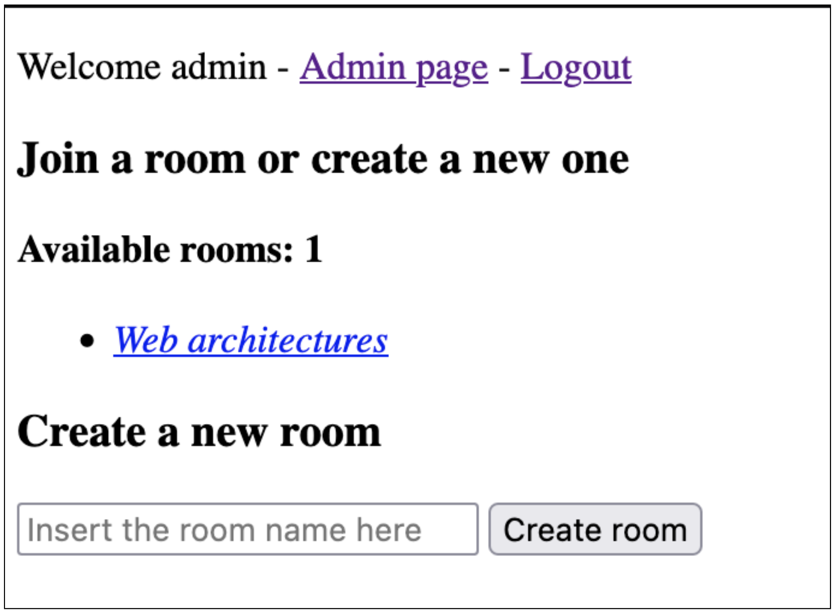
There are no rooms yet, create a new one!

Available rooms: 0

Create a new room

Insert the room name here
Create room

Figure 2: Room list with no rooms avalable


Welcome admin - Admin page - Logout

Join a room or create a new one

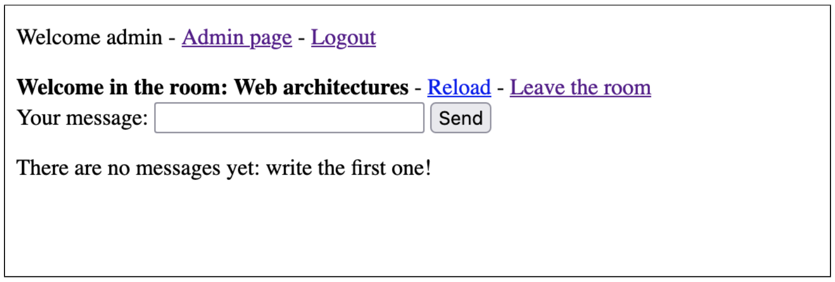
Available rooms: 1

- *Web architectures*

Create a new room

Insert the room name here
Create room

Figure 3: Room list with one room avalable

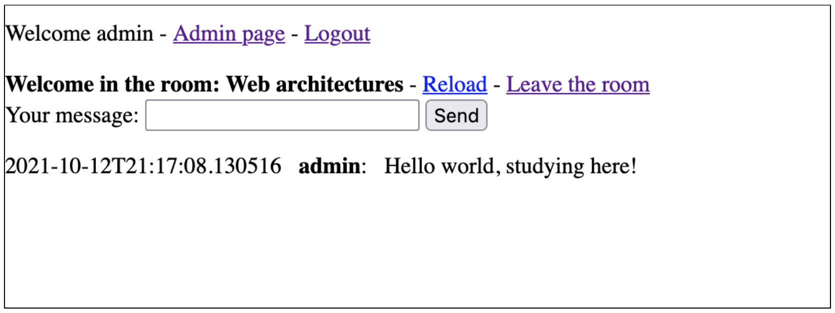

Welcome admin - Admin page - Logout

Welcome in the room: Web architectures - Reload - Leave the room

Your message:

There are no messages yet: write the first one!

Figure 4: Room with no messages

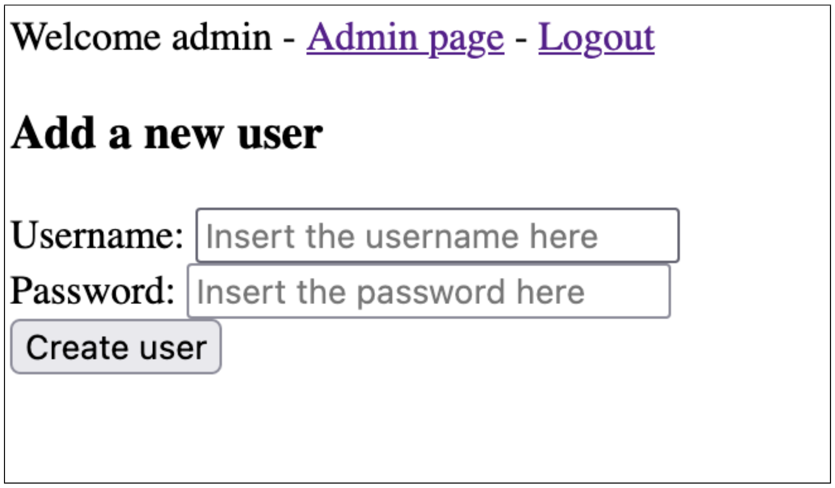

Welcome admin - Admin page - Logout

Welcome in the room: Web architectures - Reload - Leave the room

Your message:

2021-10-12T21:17:08.130516 **admin**: Hello world, studying here!

Figure 5: Room with one messages


Welcome admin - Admin page - Logout

Add a new user

Username: Insert the username here

Password: Insert the password here

Figure 6: Form to insert a new user (admin only)

4 Problems faced

In this section are reported what have been the issues faced during the development of this assignment. The only real issue I faced was in the setting up of the error page, because in the first place I tried setting up the error page only for specific error codes (like Internal server error); I did not manage to make it work so I decided to switch to a generic error page without any error code associated.