Web Architectures - Delivery 2

Marrocco Simone

October 15, 2022

Contents

1 Introduction

In this assignment we were asked to create a webapp to play a simple "Guess the Flag" game. We were given different tasks to code in order to make the webapp work.

The source code can also be found on github, on https://github.com/Marrocco-Simone/servlet_flag_game

2 Task 1 - Authentication and Registration

2.1 The task

The webapp requires authentication and registration to the server. Authentication should last a defined amount of time.

To do that, at the login page we verify the credentials and then create a new session where we save the username, while for the registration we first create a new set of credentials and then create a similar session.

The credentials are stored in the shared memory of the servlet context. Since it is shared, we need to synchronize the block where we use it. What we save inside it, as the attribute *user* is an array of the class *UserCredentials.java*, which has fields for username and password and is serializable (useful for task 9). It also uses Bean standard, meaning we need to use get and set functions to modify the fields inside (which are private).

Inside the context we also save another attribute *logged*, which contains an array of the class *UserSession.java*. This class contains the fields username and points, which are the same values we save inside the session we give back to the user. It will be useful later for task 8. Like *UserCredentials.java*, it uses Bean standard.

2.2 Login page

2.2.1 login.jsp

The login jsp page is simple: at the start of the file, we take the *error_msg* attribute for the request, and if not null we add it in the page in a paragraph.

We also have a form for the login and a link to the Register page.

2.2.2 Login.java

For the login page, we have two methods, GET and POST. The GET method simply forwards the request to the *login.jsp* file. For the POST method we loop through each set of credentials that we have: if we cannot find the given username or if the password is not correct, we forward back to the jsp page with an error message. The way we do that is simple: the jsp page accepts an *error_msg* attribute.

The place where we have saved the credentials of our users is in the context, so at the start of the POST method we use the function *getUsersFromContext* to get all the credentials saved in our system.

If the credentials are ok, we create a new session in the *setSession* function (or we change the current session, if the user went again to the login page), where we save the username and a point filed initialized at zero (as requested in the task 2). This function also modifies the context: it will be explained later, as it is another task objective.

2.2.3 Screenshots

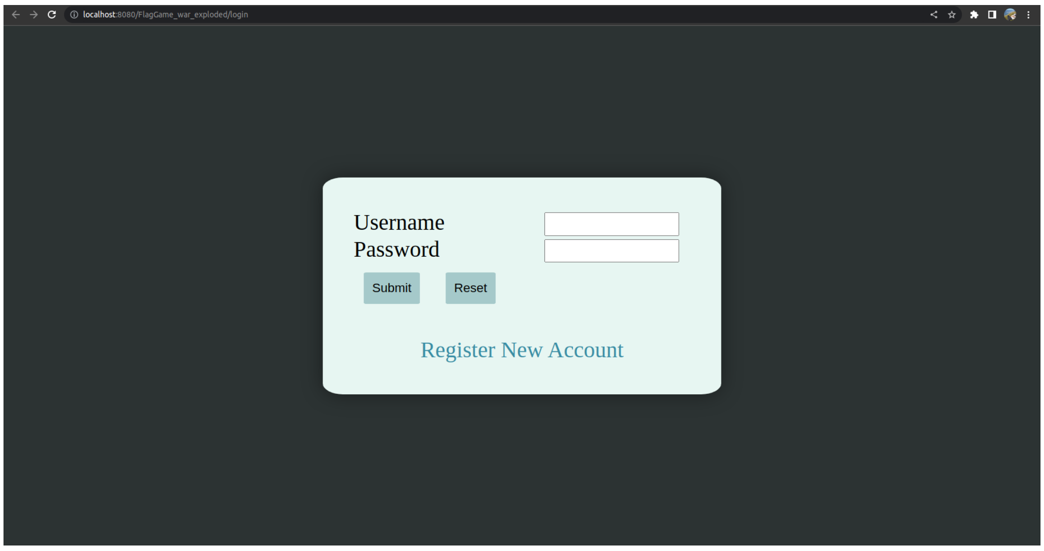


Figure 2.1: Login Page

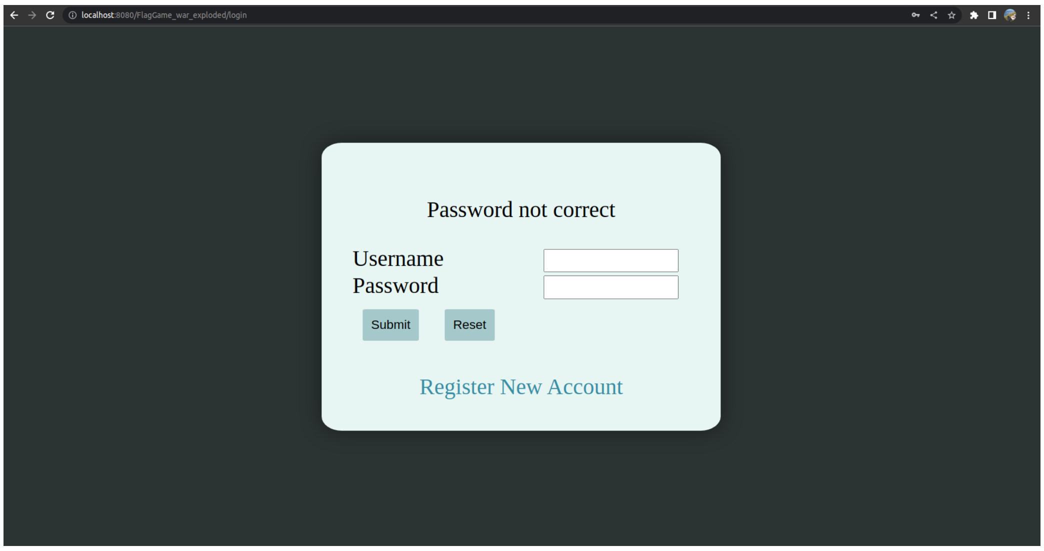


Figure 2.2: Login Page if the password is not correct

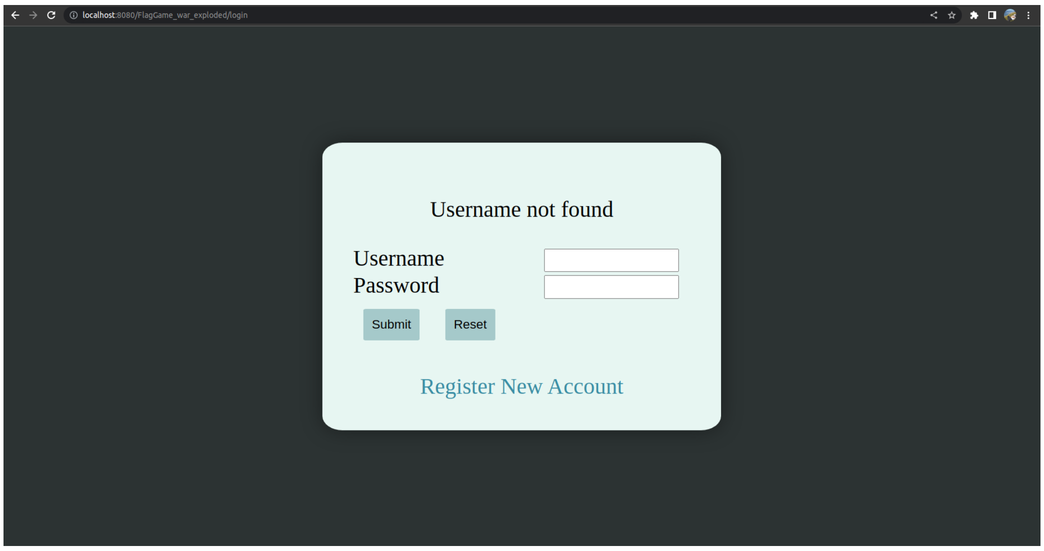


Figure 2.3: Login Page if the username is not found

2.3 Registration Page

2.3.1 register.jsp

Same logic as *login.jsp*, the only difference is that in the form we have a third field.

2.3.2 Register.java

The registration page is very similar: the main difference is that now instead of checking the context for an existing user we add a new one. The checks we do are that the two passwords given are the same and that the username does not already exist.

To get the credentials set and create a new session we use the same functions of *Login.java*, *getUsersFromContext* and *setSession*, so we make these functions static. In particular, at the end of the POST method we modify the context with the new, expanded credentials.

2.3.3 Screenshots

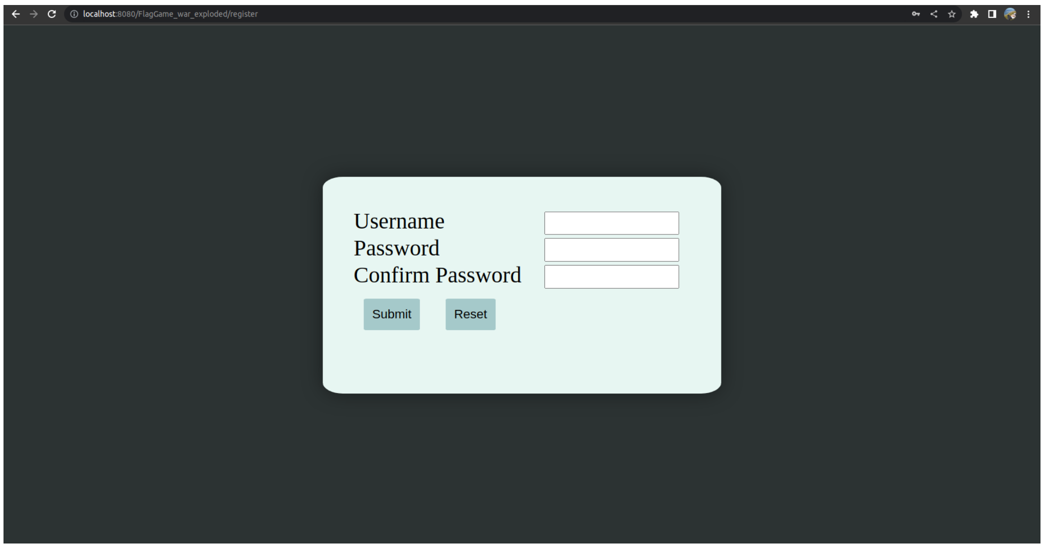
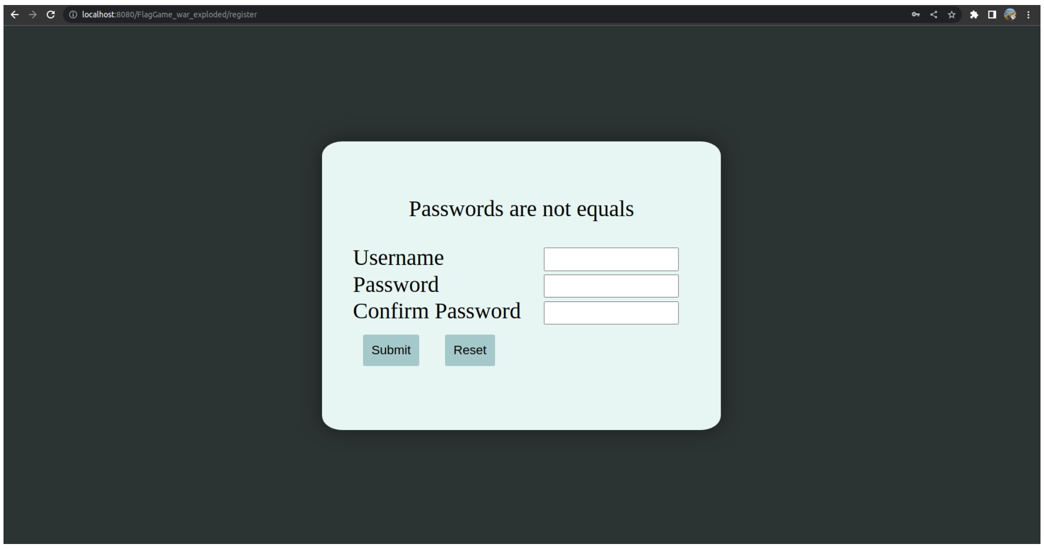


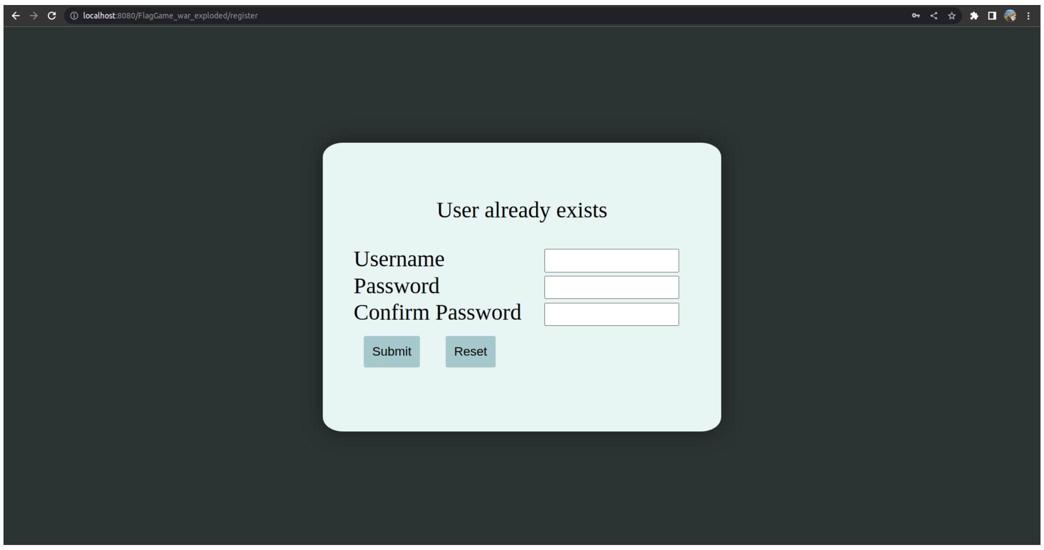
Figure 2.4: Registration Page



Passwords are not equals

Username	
Password	
Confirm Password	

Figure 2.5: Registration Page if the two passwords given are different



User already exists

Username	
Password	
Confirm Password	

Figure 2.6: Registration Page if the username already exists

3 Task 2, 3 and 7 - Starting page

3.1 The task

We should have a starting page that shows the user name in the header, the points of the user and a button to play the game.

If an unauthenticated user tries to access the page, he gets redirected to the login page.

3.2 Main page

3.2.1 main.jsp

This page is just an header for the username, a paragraph for the points and a form with a button to redirect to the game page

3.2.2 Main.java

This page is set up at the starting url , so when the user access the webapp he gets automatically redirected to the login page.

The page itself has only one method, GET, which controls the username saved in the session: if not present, redirects to the login page; if it is the admin, redirects to the admin page (as asked in the task 8); otherwise for a normal user forwards the *main.jsp* page.

We also have a static function to get the user session. This function is used by many pages, and what it does is getting the username and the points from the session; if there is no session or the session does not have an username, it redirects to the login page. In this way we can use just this function to implement the task 7.

3.2.3 Screenshots

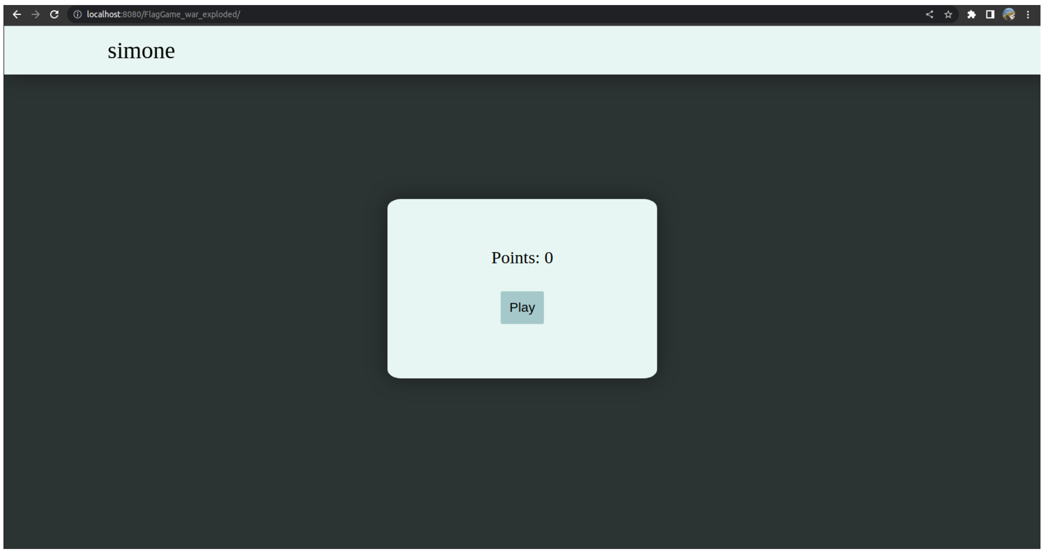


Figure 3.1: Main Page

4 Task 4, 5 and 6 - Flag Game

4.1 The task

The game consists of guessing the capitals of three random flags from a given pool.

The user writes the number on the input text near each flag. This number is taken from the shown list of capital cities. All fields must be filled in to submit a guess.

If the answers are correct, it increases the user points by three, otherwise it decreases them by one. It then returns to the main page.

4.2 Game page

4.2.1 game.jsp

At the start of the page we create a Capitals object: this objects has two fields, *capitals* and *chosen_capitals*, which are both String arrays. At the object creation, the constructor fills *capitals* with city names and shuffle it, then it chooses three random and add them to *chosen_capitals*.

We also get the user with the same method from *Main.java*, so that we can be redirected if we do not have an authentication.

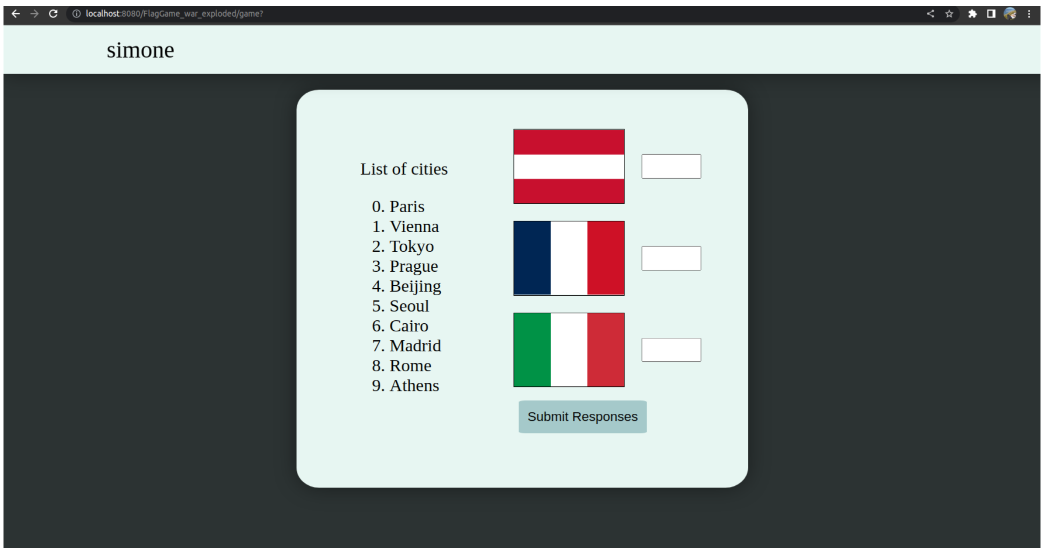
In the page, we show the same header as the main page, then an ordered list that shows *capitals* and a form with an entry for each element of *chosen_capitals*. This form has for each label an image and an input field. The image is saved in the system as *(capital_name).png*, so it is easy to get it from the chosen capital value. The input field is of type number, required and has value *min=0* and *max=(capitals.lenght())*. The name value is taken from a Capitals method *findCapitalId()*, which takes as an input a city name and returns its index in *capitals*. This way, the url that gets the form results has only to compare the field name and field value.

4.2.2 Game.java

The GET method simply redirects to the *game.jsp* page.

The POST method, as sad before, compares the field names and values to determine if the user got all responses correctly. It then updated the user session and the context (as asked for task 8) and redirects to the main page.

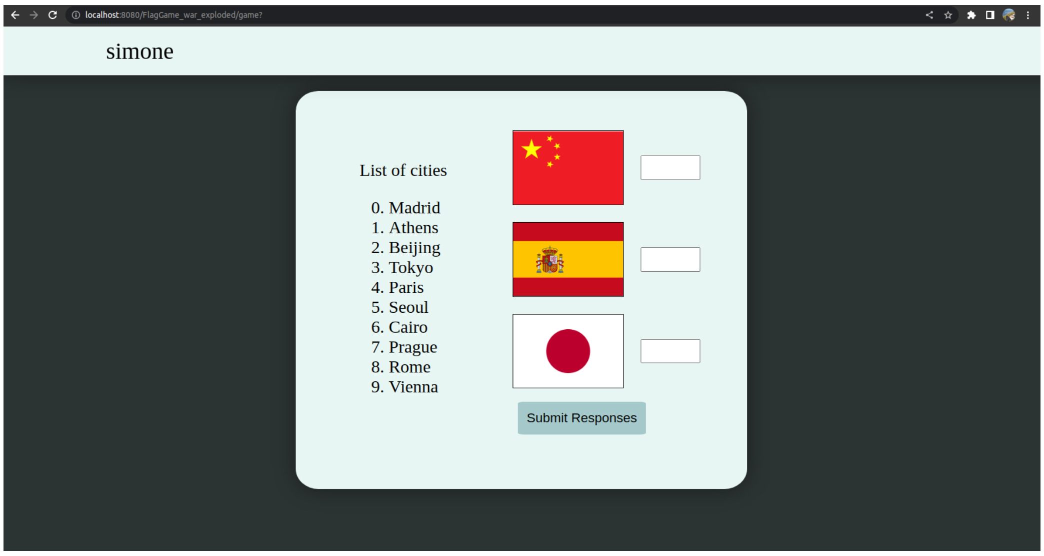
4.2.3 Screenshots



List of cities

0. Paris
1. Vienna
2. Tokyo
3. Prague
4. Beijing
5. Seoul
6. Cairo
7. Madrid
8. Rome
9. Athens

Figure 4.1: Flag Game



List of cities

0. Madrid
1. Athens
2. Beijing
3. Tokyo
4. Paris
5. Seoul
6. Cairo
7. Prague
8. Rome
9. Vienna

Figure 4.2: Flag Game, another example

5 Task 8 - Admin

5.1 The task

The admin (with defined credentials admin, password nimda) should be redirected from the main page to an admin page where he can see the current logged users and their points.

If a normal user tries to access the page, it returns error code 401. (Note: the correct code to return should be 403 Forbidden and not 401 Unauthorized, but we will follow the assignment request).

5.2 Sessions in the Context

Since we do not have a native method to get all the current sessions of our system, we need to save them in the servlet context each time a session is created (login or registration) or updated (points given at the end of a game).

Since the context is shared memory, all the functions that modify it should be in a synchronized block.

5.2.1 Create or update Session

A new session creation is done by the function *Login.setSession()*. Here, we get the context, add the new session and update it. If the session already has an username, it means that the user is doing another login, so we update the session username value. At the same time, we search the context for the old session and delete it, before adding a new one. In this way, we do not have multiple open sessions in the context for an user that changes credentials. (Note: for this reason, during testing accessing first as an user and then as an admin deletes the first session and the admin page would only show the admin as online. It is needed to log in as the user from another browser or from a private/incognito page, which uses different sessions).

A session update is also done by the game page to update the points. In this case, we search in the context for the user session and update the points there too.

5.2.2 SessionListener.java

If a session ends, we need to remove it from the context. This can be done with an *HttpSessionListener*. We can implement it with the class *SessionListener.java*, where we simply search for the expired session username and delete it from the context.

5.3 Admin page

5.3.1 admin.jsp

The admin jsp page takes as an argument the list of users to show. It contains a simple table with three columns: rank, username and points. The reason for the rank column is that we set up the list of users to be ordered, but this was not required.

This page does not redirect if a simple user tries to access the file directly instead of passing first by the java class. It is not a problem, however, since the page itself does not have the capability to access the context and gather the sensitive data.

5.3.2 unauthorized.jsp

This page simply shows an error message and a link to return to the main page.

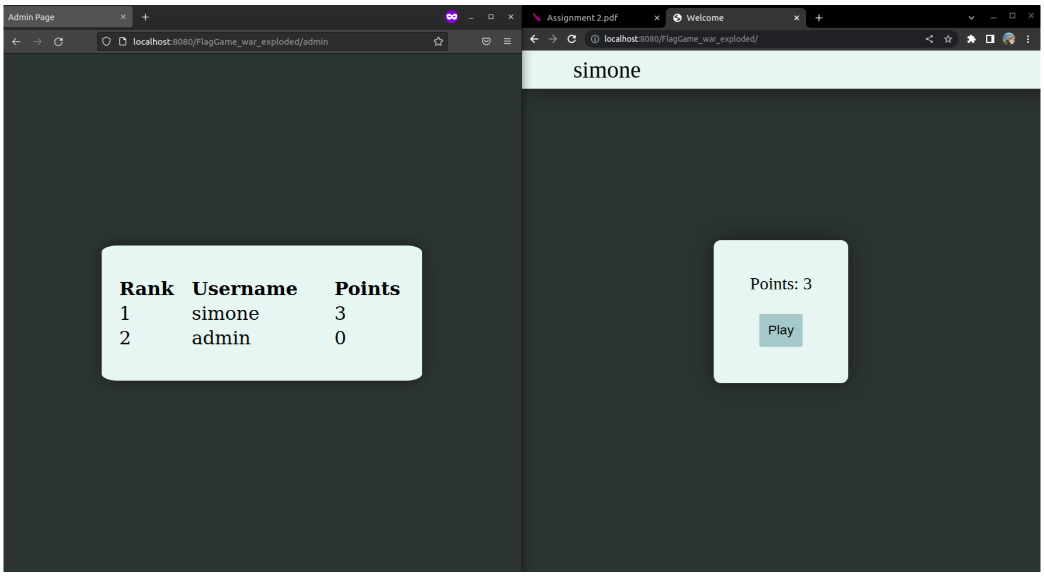
5.3.3 Admin.java

If the user is not authenticated, redirects to the main page with the static method *Main.getUserSession*.

If the username is not the admin one (set up by a static variable *ADMIN_USERNAME* because needed also by *Main.java*), it redirects to *unauthorized.jsp*.

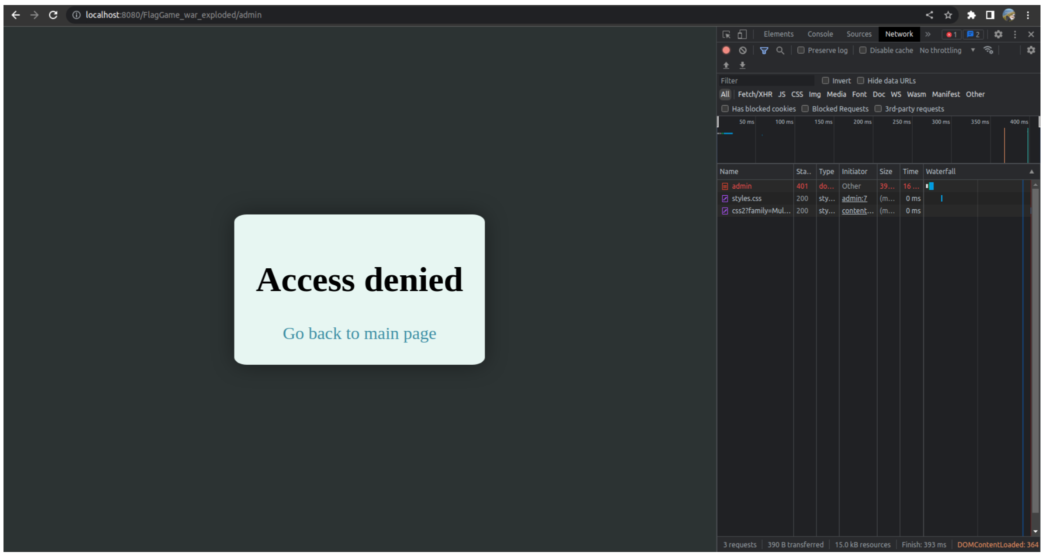
If the user is the admin, it gets all the session saved in the context, sort them and forward them to *admin.jsp*

5.3.4 Screenshots


Admin Page
localhost:8080/FlagGame_war_exploded/admin
Assignment 2.pdf
Welcome
localhost:8080/FlagGame_war_exploded/
simone
Points: 3
Play

Rank	Username	Points
1	simone	3
2	admin	0

Figure 5.1: Admin page and an user main page



Access denied

Go back to main page

Name	Sta..	Type	Initiator	Size	Time
admin	401	do...	Other	39...	16 ...
styles.css	200	sty...	admin:7	(m...	0 ms
css2?family=Mul...	200	sty...	content...	(m...	0 ms

3 requests | 390 B transferred | 15.0 kB resources | Finish: 393 ms | DOMContentLoaded: 364

Figure 5.2: Unauthorized page. The status code is 401

6 Task 9 - Persistent Credentials

6.1 The task

The credentials are mantained as long as the webapp is running. When shutdown, they should be saved to a file and recovered at the next redeploy.

6.2 Servlet Context Listener

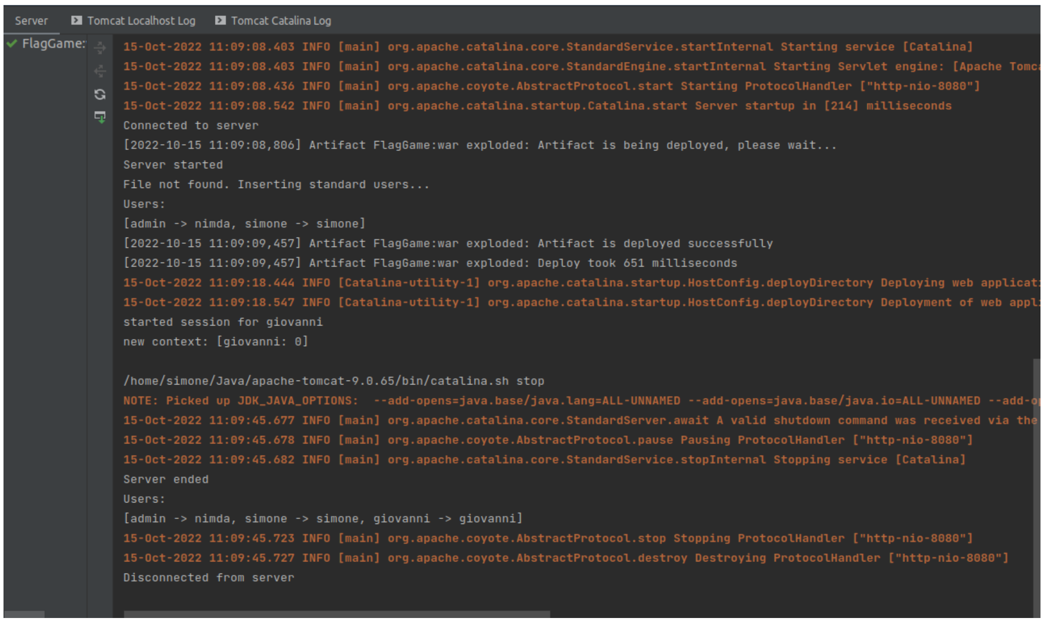
We can execute actions when the servlet is turned on or off with the servlet context listener, which uses two methods: *contextInitialized* and *contextDestroyed*.

6.2.1 ContextListener.java

When the servlet is turned on, we read the data from the file. If the file does not exist or the file cannot be opened, we create a new context with the admin and another starting user; otherwise we add to the context the users we read.

When the servlet is turned off, we save all the user credentials saved in the context inside the file.

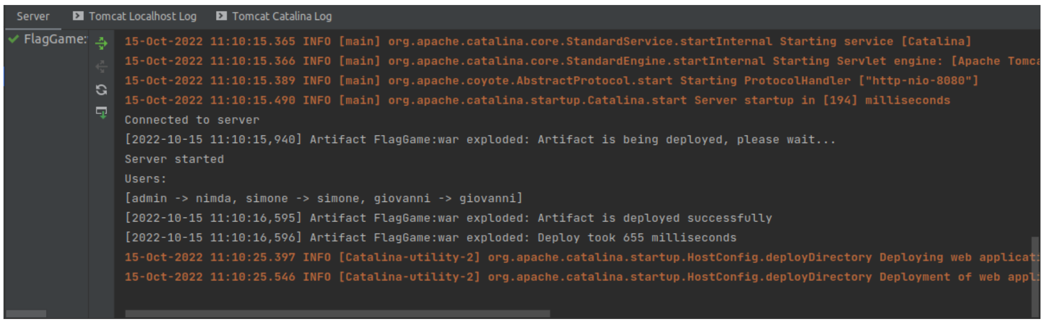
6.2.2 Screenshots



```
15-Oct-2022 11:09:08.403 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
15-Oct-2022 11:09:08.403 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache Tomca
15-Oct-2022 11:09:08.436 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
15-Oct-2022 11:09:08.542 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [214] milliseconds
Connected to server
[2022-10-15 11:09:08,806] Artifact FlagGame:war exploded: Artifact is being deployed, please wait...
Server started
File not found. Inserting standard users...
Users:
[admin -> nimda, simone -> simone]
[2022-10-15 11:09:09,457] Artifact FlagGame:war exploded: Artifact is deployed successfully
[2022-10-15 11:09:09,457] Artifact FlagGame:war exploded: Deploy took 651 milliseconds
15-Oct-2022 11:09:18.444 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicat
15-Oct-2022 11:09:18.547 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web appl
started session for giovanni
new context: [giovanni: 0]

/home/simone/Java/apache-tomcat-9.0.65/bin/catalina.sh stop
NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-o
15-Oct-2022 11:09:45.677 INFO [main] org.apache.catalina.core.StandardServer.await A valid shutdown command was received via the
15-Oct-2022 11:09:45.678 INFO [main] org.apache.coyote.AbstractProtocol.pause Pausing ProtocolHandler ["http-nio-8080"]
15-Oct-2022 11:09:45.682 INFO [main] org.apache.catalina.core.StandardService.stopInternal Stopping service [Catalina]
Server ended
Users:
[admin -> nimda, simone -> simone, giovanni -> giovanni]
15-Oct-2022 11:09:45.723 INFO [main] org.apache.coyote.AbstractProtocol.stop Stopping ProtocolHandler ["http-nio-8080"]
15-Oct-2022 11:09:45.727 INFO [main] org.apache.coyote.AbstractProtocol.destroy Destroying ProtocolHandler ["http-nio-8080"]
Disconnected from server
```

Figure 6.1: When the file is not found, a standard context is created. We then create a new User, and we can see how it also get saved at the server shutdown



```
15-Oct-2022 11:10:15.365 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
15-Oct-2022 11:10:15.366 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache Tomca
15-Oct-2022 11:10:15.389 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
15-Oct-2022 11:10:15.490 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [194] milliseconds
Connected to server
[2022-10-15 11:10:15,940] Artifact FlagGame:war exploded: Artifact is being deployed, please wait...
Server started
Users:
[admin -> nimda, simone -> simone, giovanni -> giovanni]
[2022-10-15 11:10:16,595] Artifact FlagGame:war exploded: Artifact is deployed successfully
[2022-10-15 11:10:16,596] Artifact FlagGame:war exploded: Deploy took 655 milliseconds
15-Oct-2022 11:10:25.397 INFO [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicat
15-Oct-2022 11:10:25.546 INFO [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web appl
```

Figure 6.2: At the next redeploy, the three users get correctly recovered

7 Code snippets

Below some snippets of code to better understand the project

7.1 Login.java code

```java
public static synchronized List<UserCredentials> getUsersFromContext(ServletContext context) {
    Object usersAttribute =  context.getAttribute("users");
    @SuppressWarnings("unchecked")
    List<UserCredentials> userCredentials = (List<UserCredentials>) usersAttribute;
    return userCredentials;
}
```

Code 1: method to get the user credentials from the context

```java
public static synchronized void setSession(HttpServletRequest req, String username, ServletContext context){
    System.out.println("started session for " + username);

    Object loggedAttribute =  context.getAttribute("logged");
    @SuppressWarnings("unchecked")
    List<UserSession> logged = (ArrayList<UserSession>) loggedAttribute;
    if (logged == null) logged = new ArrayList<>();

    HttpSession session = req.getSession();

    // if the user already had a session
    String old_username = (String) session.getAttribute("username");
    if (old_username != null) {
        System.out.println("old login to delete: " + old_username);
        UserSession old_session = new UserSession(old_username, 0);
        logged.remove(old_session);
    }

    session.setAttribute("username", username);
    session.setAttribute("points", 0);
    UserSession new_session = new UserSession(username, 0);

    logged.add(new_session);
    System.out.println("new context: " + logged);
    System.out.println();
    context.setAttribute("logged", logged);
}
```

Code 2: method to set the user session

```java
protected void sendLoginForm(HttpServletRequest req,
   HttpServletResponse res, String error_msg) throws
   IOException, ServletException {
      req.setAttribute("error_msg", error_msg);
      req.getRequestDispatcher("jsp/login.jsp").forward(req,
    res);
}

@Override
protected void doPost(HttpServletRequest req,
   HttpServletResponse res) throws IOException,
   ServletException {
    String username = req.getParameter("username");
    String password = req.getParameter("password");

    List<UserCredentials> userCredentials;
    synchronized (getServletContext()) {
        ServletContext context = getServletContext();
        userCredentials = getUsersFromContext(context);
    }
    Iterator<UserCredentials> iter = userCredentials.
   iterator();
    boolean found = false;
    while (iter.hasNext()) {
        UserCredentials user = iter.next();
        if (user.getUsername().equals(username)) {
          found = true;
          if (!user.getPassword().equals(password)) {
              // password not correct
              sendLoginForm(req, res, "Password not correct"
   );
              return;
          }
          // account verified
          break;
        }
    }
    if (!found) {
        // username not found
        sendLoginForm(req, res, "Username not found");
        return;
    }

    synchronized (getServletContext()) {
        ServletContext context = getServletContext();
        setSession(req, username, context);
    }

    res.sendRedirect(req.getContextPath());
}
```

Code 3: POST method for login page

7.2 Register.java code

```java
protected void sendRegisterForm(HttpServletRequest req, HttpServletResponse res, String error_msg) throws IOException, ServletException {
    req.setAttribute("error_msg", error_msg);
    req.getRequestDispatcher("jsp/register.jsp").forward(req, res);
}

@Override
protected void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException {
    String username = req.getParameter("username");
    String password = req.getParameter("password");
    String confirm_password = req.getParameter("confirm_password");

    if(!password.equals(confirm_password)) {
        sendRegisterForm(req, res, "Passwords are not equals");
        return;
    }

    List<UserCredentials> userCredentials;
    synchronized (getServletContext()) {
        ServletContext context = getServletContext();
        userCredentials = Login.getUsersFromContext(context);
    }

    // check the user does not already exist
    for (UserCredentials user : userCredentials) {
        if (user.getUsername().equals(username)) {
            sendRegisterForm(req, res, "User already exists");
            return;
        }
    }

    userCredentials.add(new UserCredentials(username,
   password));
    synchronized (getServletContext()) {
        ServletContext context = getServletContext();
        context.setAttribute("users", userCredentials);
        Login.setSession(req, username, context);
    }

    res.sendRedirect(req.getContextPath());
}
```

Code 4: POST method for registration page

7.3 Main.java code

```java
public static synchronized UserSession getUserSession(
   HttpServletRequest req, HttpServletResponse res) throws
   IOException {
    HttpSession session = req.getSession();
    String username = (String) session.getAttribute("
   username");
    if(username == null) {
        res.sendRedirect("login");
        return null;
    }
    int points = (int) session.getAttribute("points");
    return new UserSession(username, points);
}
```

Code 5: method to get the user from the session

```java
@Override
public void doGet(HttpServletRequest req,
   HttpServletResponse res) throws IOException,
   ServletException {
    UserSession user = getUserSession(req, res);
    if (user == null) return;
    if (user.getUsername().equals(Admin.ADMIN_USERNAME)) {
        res.sendRedirect("admin");
        return;
    }

    req.getRequestDispatcher("jsp/main.jsp").forward(req,
   res);
}
```

Code 6: GET method for main page

7.4 Game code

```java
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException {
    UserSession user = Main.getUserSession(req, res);
    if (user == null) return;

    Enumeration<String> param_names = req.getParameterNames();
    boolean guessed_all = true;
    while(param_names.hasMoreElements()){
        String param_name = param_names.nextElement();
        String param_value = req.getParameter(param_name);
        if(!param_value.equals(param_name)){
            guessed_all = false;
            break;
        }
    }

    int added_points = 0;
    if(guessed_all) added_points = 3;
    else if(user.getPoints() > 0) added_points = -1;
    user.setPoints(user.getPoints() + added_points);

    synchronized (getServletContext()) {
        ServletContext context = getServletContext();
        Object loggedAttribute =  context.getAttribute("logged");
        @SuppressWarnings("unchecked")
        List<UserSession> logged = (ArrayList<UserSession>) loggedAttribute;

        for (UserSession logged_user: logged) {
            if(logged_user.getUsername().equals(user.getUsername())) {
                logged_user.setPoints(user.getPoints());
                break;
            }
        }

        System.out.println(user.getUsername() + " made " + added_points + " points");
        System.out.println("new context: " + logged);
        System.out.println();
        context.setAttribute("logged", logged);
    }

    HttpSession session = req.getSession();
    session.setAttribute("points", user.getPoints());
    res.sendRedirect(req.getContextPath());
}
```

Code 7: POST method for game page

```
<%
  Capitals cap = new Capitals();
  UserSession userCredentials = Main.getUserSession(request,
    response);
  if (userCredentials == null) return;
%>
<html>
<head>
  <title>Flag Game</title>
  <link rel="stylesheet" href="styles.css"/>
</head>
<body class="game-body">
  <header><%=userCredentials.getUsername()%></header>

  <div class="capitals-list">
      <p>List of cities</p>
      <ol start="0">
          <% for(String capital: cap.getCapitals()) { %>
          <li><%=capital%></li>
          <% } %>
      </ol>
  </div>

  <form action="game" method="POST" class="game-form">
      <% for(String capital: cap.getChosenCapitals()){ %>
          <div class="game-input">
              <label for="<%=capital%>">
                  <img
                      src="flags/<%=capital%>.png"
                      width="150"
                      height="100"
                      alt="Refresh the page"
                  />
              </label>
              <input
                  id="<%=capital%>"
                  name="<%=cap.findCapitalId(capital)%>"
                  type="number"
                  required min="0"
                  max="<%=cap.getNCapitals()%>"
              />
          </div>
      <% } %>
      <button type="submit">Submit Responses</button>
  </form>
</body>
```

```
</html>
```
Code 8: game.jsp code

7.5 SessionListener.java code

```java
@WebListener
public class SessionListener implements HttpSessionListener {
    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        HttpSession old_session = se.getSession();
        String username = (String) old_session.getAttribute("username");
        System.out.println("Finished session for " + username);
        UserSession old_user_session = new UserSession(username, 0);

        synchronized (old_session.getServletContext()) {
            ServletContext context = old_session.getServletContext();
            Object loggedAttribute =  context.getAttribute("logged");
            @SuppressWarnings("unchecked")
            List<UserSession> logged = (ArrayList<UserSession>) loggedAttribute;
            logged.remove(old_user_session);
            System.out.println("new context: " + logged);
            System.out.println();
            context.setAttribute("logged", logged);
        }
    }
}
```
Code 9: listener of session changes

7.6 Admin.java code

```java
public List<UserSession> getLoggedUser() {
    synchronized (getServletContext()) {
        ServletContext context = getServletContext();
        Object loggedAttribute = context.getAttribute("logged");
        @SuppressWarnings("unchecked")
        List<UserSession> logged = (ArrayList<UserSession>) loggedAttribute;
        Collections.sort(logged);
        return logged;
    }
}
```

Code 10: method to get the logged users in the context

```
@Override
protected void doGet(HttpServletRequest req,
    HttpServletResponse res) throws IOException,
    ServletException {
     UserSession user = Main.getUserSession(req, res);
     if (user == null) return;
     if (!user.getUsername().equals(ADMIN_USERNAME)) {
          res.setStatus(401);
          req.getRequestDispatcher("jsp/unauthorized.jsp").
    forward(req, res);
          return;
     }

     List<UserSession> logged = getLoggedUser();
     req.setAttribute("logged", logged);
     req.getRequestDispatcher("jsp/admin.jsp").forward(req,
    res);
}
```

Code 11: GET method for admin page

7.7 ContextListener.java code

```
@WebListener
public class ContextListener implements
    ServletContextListener {
     String FILENAME = "Users.txt";

     /** if there is an error reading FILENAME, initialize
    the servlet as new with standard users */
     public void startNewServer(List<UserCredentials>
    userCredentials, String error_msg) {
          System.out.println(error_msg + ". Inserting standard
     users...");
          userCredentials.add(new UserCredentials(Admin.
    ADMIN_USERNAME, "nimda"));
          userCredentials.add(new UserCredentials("simone", "
    simone"));
     }

     @Override
     public void contextInitialized(ServletContextEvent
    contextEvent) {
          System.out.println("Server started");
          List<UserCredentials> userCredentials = new
    ArrayList<>();
```

```java

     try {
         FileInputStream f = new FileInputStream(FILENAME);
         ObjectInputStream o = new ObjectInputStream(f);

         // Read objects
         while (f.available() > 0) {
             UserCredentials user = (UserCredentials) o.readObject();
             userCredentials.add(user);
         }

         o.close();
         f.close();
     } catch (FileNotFoundException e) {
         startNewServer(userCredentials, "File not found");
     } catch (IOException e) {
         startNewServer(userCredentials, "Error initializing stream");
     } catch (ClassNotFoundException e) {
         startNewServer(userCredentials, "File has not correct format");
     }
     System.out.println("Users:\n" + userCredentials);

     synchronized (contextEvent.getServletContext()) {
         ServletContext context = contextEvent.getServletContext();
         context.setAttribute("users", userCredentials);
     }
 }

 @Override
 public void contextDestroyed(ServletContextEvent contextEvent) {
     System.out.println("Server ended");

     List<UserCredentials> userCredentials;
     synchronized (contextEvent.getServletContext()) {
         ServletContext context = contextEvent.getServletContext();
         userCredentials = Login.getUsersFromContext(context);
     }
     System.out.println("Users:\n" + userCredentials);

     try {
            FileOutputStream f = new FileOutputStream(FILENAME);
            ObjectOutputStream o = new ObjectOutputStream(f);

            for (UserCredentials user : userCredentials) {
                o.writeObject(user);
            }

            o.close();
            f.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
        } catch (IOException e) {
            System.out.println("Error initializing stream");
        }
    }
}
```
Code 12: listener of context changes