Assignment 2

A WebApp implements a chat system, as described in the following. Develop the code using an MVC pattern.

Authentication

1) To access the pages, users must authenticate themselves. Unauthenticated users must be directed to a login page, which asks login and password
2) The login page starts the authentication process, asking the user for username and password (do not pass them in a get!). Authentication is performed comparing the provided credentials with the ones contained in a table read from a file (suggestions: use a Map and serialize it). The table must contain (at least) the following pairs:
    - user1 – pw1
    - user2 – pw2

   A special user "admin" has extra capabilities (see below). Its password is defined as a parameter in the web.xml

Banner

3) All pages, except the authentication one, show a banner with the username and a "Log out" button. This button invalidates the authentication, and brings the user back to the login page.

User page

4) User page shows:
    - A list of rooms (at the WebApp start empty): if empty says "Sorry, no rooms are available, but you can create one". If not empty, it says "enter in a room or create a new one" and shows the list of available room names, which are clickable.
    - A form which allows creating a new room, giving it a name (which should suggest what topic the room is about), and goes back to the update user page.

5) Clicking on a room's name in the list, the corresponding room page is called.

Room page

6) A room shows its name, and presents messages on the topic declared in its name, and shows them in reversed chronological order (newest first).

7) Every message is composed by three parts:
    - the name of the user who wrote the message
    - text of the message
    - timestamp of message creation.

8) At the beginning of the page, a form allows adding a new message in the room.

9) Every 15 seconds the page reloads itself automatically, so that new messages (if any) are shown.

10) A button allows manually reloading the page.

11) A link "leave room" allows returning to the user page.

12) It is not requested that rooms and messages are persistent (if the webapp restarts, everything is lost).

Administration page

13) Only the admin user has access to this page, which shows the list of users and allows adding new users (creating a username-password pair). New users have to be saved in the users file mentioned at the point 2.

Suggestions

Start by defining the business objects (e.g. Room, Message) and their collections.

Examine what is the lifespan of data to put them in the right place.

Use a filter to access control.

Define the users file path in a (static) constant, or in a parameter in the web.xml.

For automatic page reloading, see here:
https://stackoverflow.com/questions/8711888/auto-refresh-code-in-html-using-meta-tags

Use two browsers to impersonate two different users and test your webapp.

Warning: spaces in the room names might give you problems if you use them as parameters in the urls. The suggestion is to use string encoding, using a function like this:

```
String getEncodedTitle(String title) {
        String encodedTitle=null;
        try {
            encodedTitle=URLEncoder.encode(title,"UTF-8");
        } catch (UnsupportedEncodingException ex) {
            ex.printStackTrace();
        }
        System.out.println(title+"<->"+encodedTitle);
        return encodedTitle;
 }
```

Delivery

Deliver the (zipped) IntelliJ project.
Deliver a report using the following structure:

Title Page containing date, title, your name

1 Introduction (problem statement, description of the domain)

2 Short description of what you did

3 screen shots of your app running, documenting the various steps

4 Comments and notes (optional: any problems encountered during project development, any other comment)

References (if any)

Delivery has to be delivered by Oct.17, 23:59 on

https://didatticaonline.unitn.it/dol/mod/assign/view.php?id=1001226
Suggestion: do not wait the deadline to deliver!