SOEN 387: WebBased Enterprised Applications Design

Assignment 1 on Servlets and JSP

Fall 2020, sections F

September 22, 2020

Contents

1 General Information

Date posted: Friday September 22th, 2020.

Date due: Tuesday October 13th, 2020, by 23:59[1].

Weight: 5% of the overall grade.

2 Introduction

This assignment targets: 1) understanding java web technology 2) understanding http server, servlets and jsp, and request and response objects 3) understanding and processing headers, content-types, and data encoding.

3 Ground rules

You are allowed to work on a team of 4 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. See Submission Notes for the details.

ONLY one copy of the assignment is to be submitted by the team leader. Upon submission, you must book an appointment with the marker team and demo the assignment. All members of the team must be present during the demo to receive the credit. Failure to do so may result in zero credit.

This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

[1] see submission notes

4 Your Assignment

In this assignment you implement a small web application that simulates the function of a chat server.

The assignment consists of the following parts:

1) The Chat Business Layer, 2) Creating a Basic Servlet, 3) The Web Front-End, 4) Using Curl, and 5) Source Control.

4.1 The Chat Business Layer

The business layer is to be implemented in a stand alone class library project. The business layer is responsible for chat functionality. This assignment does specify any specific classes to implement. You may use any design or number of classes, however, a single class namely `ChatManager` would be sufficient.

The business layer is to be designed as generic as possible and must not depend on web technology (One may use the very same layer and use it in a console application).

Instead, the business layer is referenced by the web application (see next sections).

Characteristics

- This assignment does not concern security and privacy. All messages are public and users are not authenticated.
- All messages are held in memory. No persistency is required.
- Chat messages are tagged with the username of the person who post them. In case user name is not provided, the message is considered "anonymous".

The functions of the `ChatManager` is as follows:

The Business Functions

1. PostMessage(user, message): where user represents the name of the user and message is a plain text (that may include any character). The message is dated automatically (in UTC) and is returned to the called.

2. ListMessages(date-range): returns all posted messages that are within the date range. If no range is specified, all messages are returned.

3. ClearChat(date-range): removes all chat messages that are within the date range. If no range is specified, all messages are deleted.

4.2 Creating a Basic Servlet

In this part, you are creating a servlet that responds to the following requests:

1. `GET from, to, format`: The servlet responds to a GET request and returns all messages that are dated within the from-to range. The third parameter specifies the format: xml or plain-text. All three parameters are optional. The default format is plain-text. The format of the xml is not specified, you may implement it as you wish. The data is written to the servlet response.

2. `POST user, message`: The servlet response to the POST request, by which the message to be posted is received and processed. The user parameter is optional. The message text is required.

3. `to-be-designed`: This request (to be implemented in POST, GET, or even in DELETE) is used to clear the chat messages. You may your own design to incorporate this option into the previous two, or even in a separate servlet.

Requirements

- The Basic servlet uses ChatManager in the previous section to handle the chat.

- The basic servlet is implemented in a web project that references the business layer, as specified in the previous section. The web project will contact the front-end as well, which is specified in the next section.

- Multiple users in multiple sessions use the chat server, therefore, only one instance of the chat manager must be used by the application.

- the servlet is open to all requests and does not check for authentication.
- To make it little restricted, the servlet looks for "referer" header and in case the referer is not present, it must display an error message and must not proceed with the request.
- in 1, the servlet writes the data into the servlet output stream. Use appropriate `content-disposition` header to specify a proper filename so that the user sees the data as a downloaded file. Make sure the client does not cache the data (see: `expires`).
- In 2 and 3, once the servlet processes the request, it redirects the control to chat front-page (see next session). In case of errors, it passes the error to the front-page in order to be displayed.
- Your encouraged to use Java stream API.

4.3 The Web Front-End

The web front-end mainly consists of a front page in which the current chat messages are displayed as well as it lets the user to post a new message. You may use any front-end framework of your choice. Using a rich user interface is a bonus. The only requirement is the communication with the business layer must strictly be done through the basic servlet, as specified in the previous section.

Functions

The web front-end provides the following functions:

- Lets the user to identify themselves and post a message;
- Lets anyone clear the chat;
- Lets any user download and save the chat as a file (two formats are supported: text, and xml);
- Provides a refresh mechanism via a refresh button or periodically, so that the new messages are displayed to all open windows.

HTML Format

Use XHTML and appropriate DOCTYPE.

Styling

Using CSS, provide two distinct stylesheets to the front-end. The users may switch between the two as they wish. The look-and-feel of the two styles must be relatively different (different color sets, background image, logo, and arrangement (positions) of the links and components. The styling is strictly to be done using CSS and without changing the document structure.

4.4 Using Curl

Using curl, simulate calling the servlet functions from the command line: clear, post, and download as text and xml. Provide the curl commands.

Hint: use curl –v for full response view

During the demo, you will run the commands in conjunction with using the web front-end and explain how they affects the system (see the note on redirect).

4.5 Source Control

Each team may use GitHub or an alternative source control during development phase. Using a source control software is mandatory.

5 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. Your instructor will provide you with more details. It has to be completed by ALL members of the team in one submission file.

Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 4 students at most (including yourself). Any teams of 5 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`.
Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above: `https://moodle.concordia.ca`
Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep a record of your submission confirmation. This is your proof of submission, which you may need should a submission problem arises.

6 Grading Scheme

The Business Layer	10 marks
Servlet POST and DELETE	10 marks
Servlet GET and Download	10 marks
Proper Data-Flow	5 marks
Encoding & Error Handling	10 marks
Front-End	35 marks
CSS Styling	5 marks
Using CURL	10 marks
Git	5 marks
Bonus	10 marks

Total: 100 (+10) marks.

References

1. JSP Tutorial: `https://www.tutorialspoint.com/jsp/index.htm`
2. The Content-Disposition:
   `https://en.wikipedia.org/wiki/MIME#Content-Disposition`
3. The Referer header: `https://en.wikipedia.org/wiki/HTTP_referer`
4. Java Streams: `https://www.baeldung.com/java-8-streams`
5. CSS Tutorial: `https://www.w3schools.com/css/`
6. XHTML DOCTYPE: `https://www.w3schools.com/html/html_xhtml.asp`
7. Scope of JSP Objects:
   `https://javapapers.com/jsp/explain-the-scope-of-jsp-objects/`
8. SimpleDateFormat: `https://dzone.com/articles/java-simpledateformat-guide`