SOEN 387: WebBased Enterprised Applications Design Assignment 2 on Layered Architecture, Data Formats, and using Databases

Fall 2020, sections F

October 20, 2020

Contents

1 General Information **2**

2 Introduction **2**

2.1 Overview 2

3 Ground rules **2**

4 Your Assignment **3**

4.1 Message Board Business Layer 3

4.2 The [optional] Persistence Layer 5

4.3 The File Download Servlet 5

4.4 The Web Front-End 6

4.5 Source Control 6

4.6 Documentation 7

5 What to Submit **7**

6 Grading Scheme **8**

1 General Information

Date posted: Tuesday October 20th, 2020.

Date due: Wednesday November 11th, 2020, by 23:59[1].

Weight: 8% of the overall grade.

2 Introduction

This assignment targets implementing web applications with an emphasis on

1) layered architecture, 2) handling data and file formats, and 3) using databases.

2.1 Overview

In this assignment you are implementing a simplified "Message Board System". A message board is an online discussion site where people can hold conversations in the form of posted messages. The messages are normally longer than chat messages and may have attachments.

Note: The detailed description of the functionality of the system is explained in section 4. While the topic of this assignment is slightly different from **assignment 1**, you may still re-use the skeleton and most components you created in that assignment. If you wish to do so, make sure you perform a proper refactoring, so that the requirements of the message board system are fully addressed.

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

[1] see submission notes

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.

4 Your Assignment

In this assignment you implement a small web application that simulates a simplified "Message Board System".
The assignment consists of the following parts:

1) The Message Board Business Layer, 2) The [optional] Persistence Layer, 3) The File Download Servlet, 4) The Web Front-End, 5) Source Control, and 6) Documentation.
It is recommended that you start the project by working on the detailed design of the "Message Board Business Layer".

4.1 Message Board Business Layer

The functionality of the Message Board System is outlined in the following. Similarly to the previous assignment, the business layer is to be implemented in a stand alone class library project. This business layer will be responsible for the message board system functionality and should not have any dependency on any web technology. This assignment does not specify any specific classes to implement. You may use any design or any number of classes to implement the business layer (see 2).

Business Functions

The functions of the Message Board System are outlined in the following. The system it to be implement as a "prototype" and as a result not all features of a message board system are available (e.g. user registration). The options to those functions must be present in the system; however, when the user wishes to select such unimplemented features, a proper

error message must be given to the user indicating that the feature is not implemented at the moment.

Users and Authentication

The user management component is in a prototype state and is not fully functional. The system currently uses a set of pre-defined users with pre-set passwords. In future, users will be able to **sign-up**, **modify** their accounts, and **change their password**.

Note: All passwords must be hashed and not kept in plain text. To do so, you may use any hashing algorithm such as **MD5** or **SHA** (see 8, 9).

User Posts

Users, once authenticated, are able to do the following:

1. Create a Post (automatically time-stamped by the system)
2. Delete a Post
3. Update a Post (once updated, the post is marked as updated)
4. Search for Posts, based on the user who posted it, a date range, as well as a lists of hash-tags; all of which are optional. The result is sorted in a reverse chronological order.
5. View recent Posts, by which the most recent posts are displayed, in a reverse chronological order. The number of posts to display must be **configurable** (to be read form a configuration file), and by default is set to 10.

Note: A post is a free text message that may optionally contain **hash-tags**. The hash-tags are textual words within or after the message text, prefixed with a pound sign (‘#’).

File Attachments

A post may optionally have a file attachment. The attachment may be **downloaded** by the user who views the post. The original user may optionally **delete** or **replace** the attachment in the post, by which the post is marked **updated**.

Note: Direct access to the attachment files must be **restricted**; the web user may not be able to access them via a **back-door**.

Access Level

All posts and attachments are **publicly** accessible by all **authenticated** users. Unauthorized users will **NOT** be able to make or see any posts.

4.2 The [optional] Persistence Layer

The data persistency may be implemented within the Business layer or in a completely separated layer. The following data persistency must be supported:

1. Readonly access to the users data, stored in a restricted file, in either **XML** or **JSON** format. Users information contain: user-id, full name, email address, and hashed password.
2. Full access to the posts data: to be stored in the database. Posts information contain: post title, user who posted it, the date and time of the post, last modified date, and the message text with hashtags.
3. Attachments: to be stored in a "restricted" file system or in a BLOBsupported database table. The following information are implicitly available for each attachment: the original file name, the file size, the mediatype, and the post id which the attachment belongs to.

BONUS: Additional marks are given to implementing file attachments in BLOBs.

4.3 The File Download Servlet

The File Download Servlet provides the donwload / view access to the attachments. The servlet receives the post id as well as the attachment id as its arguments and returns the file content in the response. In case of an error, the error text, or an generic error message (or image) may be displayed. You may optionally redirect the user to an error page (see: jsp exception). Errors may be caused by accessing the servlet without authentication, or perhpas by passing invalid ids, etc.

Use appropriate `content-type` in case of success or error. The content type must match the file content (i.e. attaching jpeg vs. gif or png) 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`).

Note: The Download Servlet must **strictly** call appropriate methods in the Business layer to retrieve the post attachment and must **NOT** directly access the file. The Business layer is responsible to return the **file data** in binary format (i.e. `byte array`). You may optionally provide a method that receives an **output stream** as its argument, by which the method reads the file and writes the content into the given output stream.

4.4 The Web Front-End

The web front-end mainly consists of a front page that enables the user to see the recent posts, search for posts, and create, update, or delete a post with options for attachments. You may use any front-end framework of your choice. However the **business layer** must strictly be implemented as a **java class library**.

Functions

The web front-end provides the following functions:

- The web application always check for the user session. If the user is not authenticated, the system must let the user logs in to the system first.
- Once authenticated the user goes to the front page by which the most recent posts are displayed. User has the option to sign out.
- A search option must be provided by which the user will be able to search by date, author, and hash-tags.

4.5 Source Control

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

4.6 Documentation

As part of the submission, each team must provide the following list or artifacts:

- A **README.md** file that contains the team information, the description of the application, and the release notes (instructions on how to install the software components);
- The complete **Use-Case Diagram** of the system;
- The **Class Diagram** of the system (illustrating the classes and associations in the business layer as well as the data access layer);
- The **Entity Relationship Model** (see 7) in the form of an ERD or a UML diagram;
- A **Sequence Diagram** for a file-download scenario, illustrating all classes that are involved;
- A **Sequence Diagram** for a create post scenario, illustrating all classes that are involved.

5 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. 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

System Functionality	20 marks
The Business Layer	15 marks
The Download Servlet	15 marks
Encoding & Error Handling	10 marks
Front-End	25 marks
Source Control	5 marks
Documentation	10 marks
DAL and BLOBs	10 marks

Total: 100 (+10) marks.

References

1. Message Board: https://en.wikipedia.org/wiki/Internet_forum
2. Martin Fowler, "Patterns of Enterprise Application Architecture (EAA)", ch. 2: Organizing Domain Logic, 2002, ISBN-10: 0321127420, ISBN-13: 978-0321127426
3. JSP exception:

   https://www.tutorialspoint.com/jsp/jsp_exception_handling.htm
4. The Content-Disposition:

   https://en.wikipedia.org/wiki/MIME#Content-Disposition
5. Java Date Time: https://www.w3schools.com/java/java_date.asp
6. SimpleDateFormat: https://dzone.com/articles/java-simpledateformat-guide
7. The Entity Relationship Model:

   https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
8. Java support for MD5: https://www.baeldung.com/java-md5
9. Java support for SHA: https://www.baeldung.com/sha-256-hashing-java