SOEN 387: WebBased Enterprised Applications Design

Assignment 1 on Servlets and JSP

Fall 2021, sections F

September 17, 2021

Contents

1 General Information

Date posted: Friday September 17th, 2020.

Date due: Tuesday October 19th, 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 3 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.

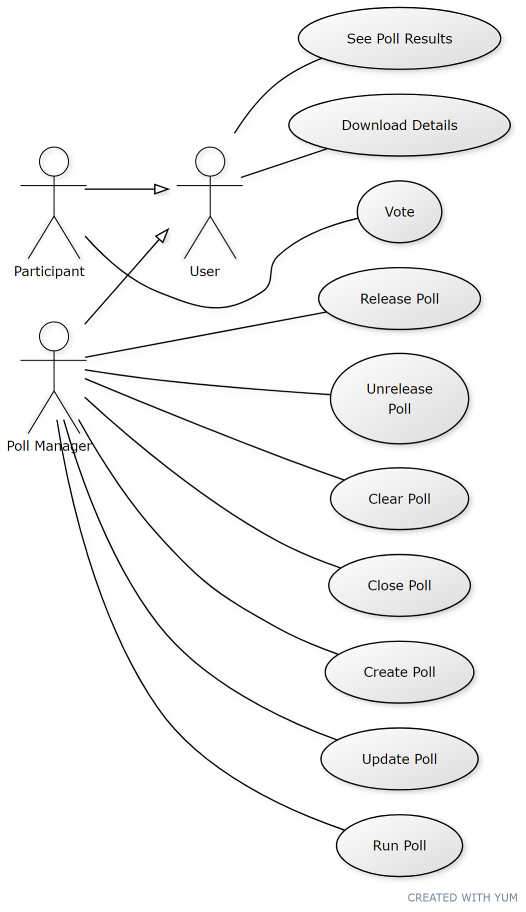
4 System Overview

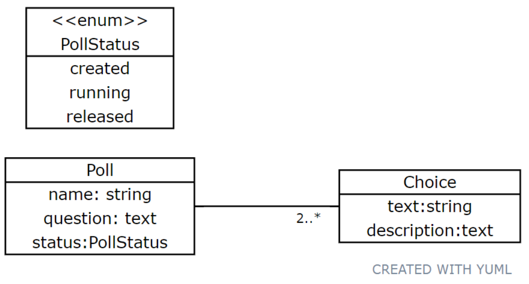
In this assignment you want to create a simple Poll system that allows a poll manager to create, run, and release a poll to a group of participants, as specified in the following:

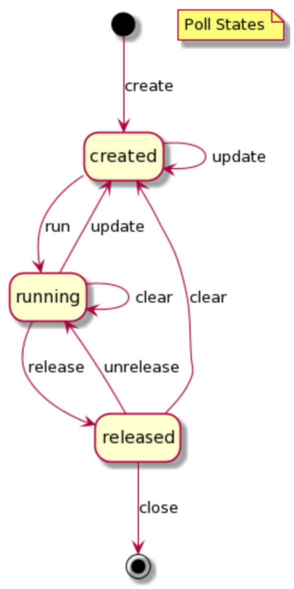
> A poll allows a person to ask one multiple choice question; participants can choose from among pre-defined answers; and only one answer is allowed.

[1]see submission notes

The "partial" conceptual model and the use-cases of the system are as illustrated in the following diagrams:







4.1 System Specifications:

A *Poll* consists of a name/title, a question text (poll subject), an array of two or more choices, and a status. A *Choice* is a text string along with an optional description. A *Participant* selects a single choice, when a poll is running.

A Poll may be in three states (AKA the Poll *Status*): A newly created Poll is in the *created* state. The *Poll Manager* may create a poll. The system allows one poll at a time. The Poll Manager may run a poll, by which the poll is put in the *running* state. While the poll is running, participants may *vote* among the available choices. The Poll Manager may update a running pole, by which, the poll results are *cleared* and the poll is put back at the created state. The Poll Manager may optionally clear a a running poll. The Poll Manager may

decide to stop and release the results, by which the poll will be put at the *released* state. If a poll is released by accident, the Poll Manager may optionally *unrelease* the poll, why which the poll be put back in the running state. The Poll Manager may clear a released poll, by the poll is put back at in the created state. Finally, the Poll Manager may *close* a pole, by which, the poll and its associated data is completely *removed* from the system.
When a poll is released, any user (a *Participant* or the *Poll Manager*) may view or optionally download the results in a *text file*.
All votes are *anonymous*. No user name is entered or recorded. The poll results contains the information about the poll (title, question, and valid choices, as specified in the above) along with the list of captured votes (dated and time-stamped).
A user only has a single vote. However, while the poll is running users may change their votes. Only the most recent received vote is counted towards the poll results.

5 Your Assignment

In this assignment you implement a small web-based poll application. The assignment consists of the following parts: 1) The Poll Business Layer, 2) Creating a Basic Servlet, 3) The Web Front-End, 4) Source Control, 5) UML Class Diagram. See the notes under section 6 for the implementation requirements.

5.1 The Poll Business Layer

The poll business layer is to be implemented in a stand alone java class library project. The business layer is responsible for creating and running the poll. This assignment does not specify any specific classes to be implemented. You may use any design or any number of classes. A sample `PollManager` class may be used as the central facade of (access point to) the business layer.
The business layer is to be designed as generic as possible and must not depend on any web technology (for instance, one may use the very same class library in a console application). The business layer is referenced by the web application (see next sections).

Business Functions

The business functions are listed in the following. The criteria for each business function is specified. The operation may not be allowed, if the criteria for the specific operation is not met, in which case, a proper exception is raised.

1. CreatePoll(name, question, choices): Create a new poll; operation is not allowed if there is currently a poll in the system.
2. UpdatePoll(name, question, choices): Updates a poll; A poll pay be updated if it is newly created or running. Updating a poll, will clear its results and puts the poll back in the created state. The update process may update any of the poll fields: the poll name, the question, and its valid answers.
3. ClearPoll(): clears the results of a running or released poll. A running poll remains in the running state, while a released poll will be put back into the created state.
4. ClosePoll(): closes the current poll. Only a released poll may be closed.
5. RunPoll(): runs current poll. Only a created poll may be run.
6. ReleasePoll(): releases a running poll.
7. UnreleasePoll(): unreleased a released poll. The unreleased maintains the current answers and changes the poll status to running.
8. Vote(participant, choice): allows an user to vote. The vote is allowed while the poll is running. The participant object keeps the anonymous identity of the user[2].
9. GetPollResults(): returns the poll results in a `HashTable`, whose keys and values are the choices and the total count per choice. Only the results of a released poll may be retrieved.
10. DownloadPollDetails(PrintWriter output, ref String filename): writes the poll details to the output. The filename will be set to "`«POLLNAME»-«DATETIME».txt`" where `«POLLNAME»` is replaced with the name of the poll and `«DATETIME»` is the timestamp at which the poll is released. Only the details of a released poll may be extracted.

[2]You may use a session id, or a unique identifier per session

Error Handling

In case any of the above operations is not allowed (i.e. creating a new poll while a poll is running), a custom exception is raised by the business layer that provides the reason of the error to the caller. Be specific as possible.

User Management

For simplicity, the business layer does not maintain any user management. All above business functions are accessible via method calls in Java. See user management under section 5.3.

Data Persistency

This assignment does not use any persistency later. No database is used in this implementation. All poll data are held in memory.

5.2 Creating a Basic Servlet

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

1. `GET name, format`: The servlet responds to a GET request and returns the poll details as a file. 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`). The `GET` request verifies if the name matches the currently released poll name and the format string is set to "text". If either of the conditions are not met or the business layer throws an exception, the exception text is printed to the server output stream and no file is returned. The server may optionally redirects the user to a GUI error page, in which the error content is displayed.

2. `POST choice`: The servlet response to the POST request, by which the selected choice by the current user is recorded. Only a valid choice is accepted. In case the choice is invalid or no poll is running, an error message is to be displayed (similar to the error

in the `GET` method). Use `HttpSession` to maintain the user session. All user's answer in an active session may be counted as one (see 4.1).

The servlet is open to all requests and does not check for authentication.

5.3 The Web Front-End

The web front-end consist of the following page/pagelets:

1. A front page for the anonymous users to participate in a poll (if available), view poll results, download results as a text file (via the servlet in 5.2).

2. A front page for the poll manager to manage the poll.

User Management

The user management responsibility is to be implemented in the Web-UI layer, by which a **pre-defined pass-code** is to be entered by the poll manager. There is no username.
All other users are anonymous. No user id or password is required to access the system. A web session is essentially representing an anonymous user.

View Poll Results

Use Google charts (see 8) to display the results. You may use a bar chart, a pie chart, or other chart of your choice. The chart essentially displays the choices and total number of votes per choice.

HTML and CSS

Use XHTML and appropriate DOCTYPE. The styling is strictly to be done using CSS and without changing the document structure.

5.4 Source Control

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

5.5 UML Class Diagram

After the project is done, draw the complete UML class diagram of the Business Layer and include it in your submission.

6 Implementation Platform and Requirements

The following non-functional requirements must be met:

1. The implementation consists of a *front-end* and a *back-end*. While you are allowed to used any technology or framework in the implementation of the front-end, the back-end must strictly be implemented in java. Section 5.1 provides detailed specifications on how the business layer must be implemented using a pure java class library. No other languages or platform is allowed.

2. Section 5.2 specifies that a basic servlet must be implemented to handle the file download. While the rest of the back-end may be implemented using any framework or technology, the servlet POST method must be implemented as specified. Collecting user votes is strictly to be implemented via the servlet `POST` method.

3. You may use any technology / framework to implemented the front-end. Using rich javascript-based frameworks are welcome. however, see notes under section 4.1 regarding the session management. An active user may submit multiple votes, among which the most recent one is only counted.

4. User Google charts is mandatory (see section 5.3).

7 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.

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 3 students at most (including yourself). Any teams of 4 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 should be submitted for the tasks that require them. Archive all files 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.

8 Grading Scheme

The Business Layer	20 marks
The Web-UI	25 marks
The Servlet	15 marks
File Download	10 marks
Proper Error Handling	10 marks
CSS Styling	5 marks
Git	5 marks
UML Class Diagram	10 marks

Total: 100 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. CSS Tutorial: `https://www.w3schools.com/css/`
4. XHTML DOCTYPE: `https://www.w3schools.com/html/html_xhtml.asp`
5. Scope of JSP Objects:

   `https://javapapers.com/jsp/explain-the-scope-of-jsp-objects/`
6. Java Date Time: `https://www.w3schools.com/java/java_date.asp`
7. SimpleDateFormat: `https://dzone.com/articles/java-simpledateformat-guide`
8. Google Charts:
   - `https://developers.google.com/chart/interactive/docs/gallery/piechart`
   - `https://developers.google.com/chart/interactive/docs/gallery/barchart`
9. Online UML tools:
   - `https://yuml.me/diagram/scruffy/usecase/draw`
   - `https://yuml.me/diagram/scruffy/class/draw`
   - `https://plantuml.com/state-diagram`