***Write ups***

Registration Page

Before taking a quiz, a user have to first register and login. To register one can directly click on the register menu or a link to registration page is provided on the login page to create an account.

Login Page

When a user clicks, an exam user is directly redirected  to the login page, if the user is not logged in

## User Home Page

Home page is the landing page of the application from where a user can take any quiz by clicking on the quiz. On the home page, if the user is logged in his name is also shown and a logout link is provided to logout from the application.

## Quiz Screens

On starting a quiz, a user is presented with the first question of the quiz with the next and finish button. Note that the previous button is not shown when the user is on first question and next button is not shown when the user is in the last question.

## Quiz Result

After clicking on the finish button the user is presented with exam results, showing the name of quiz; time when the quiz was started and the number of questions that the user answered correctly.

## How This Quiz Application Works?

Well, it works as you expect it to work. To take a quiz, a user must be logged in. If the user is not logged in, the user will automatically be redirected to the login page, where the user can login. If the user is not already registered to the application, he has to create an account first. After successfully logging in to the application, the user can take any of the quiz by clicking on it. Now, the user will be presented with the first question of the quiz. To move through the quiz questions, the user is provided with the next and previous buttons. The quiz can be finished at any time by clicking on the finish button.

## Creating the Home Page

Home page is pretty straightforward. We have a menu and 8 images displayed in a table format with two rows; each row containing 4 images. On the home page we also make a check, whether the user is logged in or not. If the user is logged in we also display the username and provide a logout link.

## Creating the User Registration Page

There is nothing fancy in the registration page; just an HTML form awaiting the user to provide his name, email and password. Once we get that, we pass this to RegistrationController servlet to create an account.

Note: We are not doing any validation like password should contain 8 characters with at least one uppercase character, one number and special symbol. We will do that in upcoming posts, when we extend this application.

## Getting Database Connection

In this application we have used MySQL database to store user credentials. To get a connection to database we have defined a static method createConnection in DatabaseConnectionFactory class, where all database specific information is stored.

We have just users’ table under quiz database.

## Creating the Login Page

Login page is very much similar to registration page where we are providing two input fields asking user to provide a username and password. Once we get the username and password entered by the user we pass it to LoginController to authenticate user.

## MainController for the Application

It is the MainController where we have written the code to redirect the user to appropriate page according to the incoming request url.

Implementing the Logout Functionality

Once the user clicks on logout, link session is invalidated and all the objects bind in the session are removed.

request.getSession().invalidate();

Storing the Quiz questions

Note that we have stored the questions in separate XML files, not in the database.

## How to Read the Questions Stored in XML File

To read the questions from the XML file we create a document that represents the XML file containing quiz questions. Whenever the user clicks on the next or previous button we call the setQuestion(int i) method, giving the index of question that we want to read and at the same time that question is saved in an ArrayList of QuizQuestion.

## How to Represent a Question?

QuizQuestion is the class that represents a single quiz question; each question will have a number, question statement, options and one correct option index.

Note that since this is a web application, multiple users will be taking exams simultaneously. We have to make sure that one user’s exam does not get into another user’s exam. For example, one user might have just started Java exam and another user is on question 5 of SQL exam; we have to treat them as two separate exams. To do that we will maintain the state of each exam using session.

When the user clicks on start exam button to start the exam, we will create a new instance of exam passing the test type for eg. Java, PHP, CSS etc. So each user will have a different instance of Exam class (that represents an individual exam).

Note that to track the current question in the exam we have currentQuestion property in exam class.

## Handling the Entire Exam

ExamController is the main control from where we control the exam. Here we save user selections (what user have answered for the question) in a Map. ExamController also lets user move through questions by clicking next and previous button, at the back end it is the ExamController which makes the function calls to retrieve questions and store user responses.

## Submitting the Exam and Evaluating Exam Result

When the user clicks on finish button, ExamController calls the calculateResult() method passing the Exam object, calculateResult() compares user responses with correct option for the question and returns how many correct answers a user got.

Note that until now each of our question has 4 options and there is no timer for the quiz. In the upcoming posts we are going to extend this online quiz application and will include the following functionalities :

1. Each quiz can have different number of questions

2. Each question can have different number of options

3. A question can have multiple correct options

4. Implementing a timer for the quiz

5. Maintaining a history of the user; like how many tests a user have taken in the past and his score

6. Randomizing the order of questions presented to the user

7. Giving the user option to review his answers before submitting the test for evaluation

8. A dropdown box to jump to any question in between the test rather then clicking next button multiple times.