Revisión del 1º Examén

Sistemas Web

Introducción

Para este ejercicio se va a implementar un pequeño sistema web para poder realizar **tests online**, es decir, una pequeña plataforma de e-learning. El usuario podrá realizar tests, repetirlos y ver una pequeña estadística de sus intentos, por lo que será necesario guardar en una sesión dichos datos. Además el usuario podrá validar el examen para comprobar las preguntas que ha contestado correctamente y viceversa.

Lógica de Negocio

Para implementar la lógica de negocio correspondiente se han implementado hasta 4 cuatro clases Java:

- **Quiz:** Representa el contenedor de preguntas, es decir el test. Para su implementación se ha decidido usar un **HashMap<Integer, Question>** ya que de este modo es más fácil encontrar las preguntas por su id.
- **Question:** Representa una pregunta, es a su vez un contenedor de respuestas.
- **Answer:** Representa una respuesta.

Lectura del Test desde fichero

Para la lectura del test desde fichero se ha usado una clase llamada **QuizLoader** con un sólo método (**loadQuiz**) estático que recibe como parámetro la ruta donde está dicho archivo. El método devuelve a su vez un objeto tipo Quiz.

Clases de Sesión

Se han implementado dos clases que permiten guardar los datos de sesión de usuario:

- **UserAnswer:** Resultado de los datos que se producen cuando un usuario realiza un test.
- **UserResult:** Colección de respuestas del usuario.

Flujo de la aplicación

El flujo de la aplicación es el siguiente:

1. El cliente se conecta a la aplicación la cual le presenta un **index** con un enlaze para comenzar a realizar el test.
2. La apliación presenta al usuario el test, el usuario puede hacer un **submit o ver sus intentos**.
3. Una vez que el usuario a terminado el test, el servidor recibe los datos por **POST** y se encarga de **validar las respuestas**, seteando una **cookie** al navegador con las respuestas seleccionadas por el usuario. El servidor **redirige a la página del test** la cual detecta la cookie y pinta las preguntas correctas de color verde, y las incorrectas de rojo. Además da la posibilidad de repetir el test.

4. Si el usuario escoge repetir el test se redirige a un servlet que **borra la cookie** de preguntas y vuelve a redirigir al test el cual se pinta normalmente.
5. Por último si en cualquier momento el usuario hace click en ver intentos, se mostrará en una página html los datos que hay en sesión acerca de los **intentos del usuario**.

Consideraciones

- Se ha utilizado el objeto nativo de sesión **HTTPSession** de Java para guardar la sesión del usuario mediante el objeto UserResult, de este modo, cuando el usuario pregunta por sus intentos, el servidor contesta mediante el servlet de intentos el cual devuelve un html con sus intentos ordenados por fecha.
- Se ha implementado una pequeña l**ibrería de plantillas** en la clase **TemplateLoader**, para no tener que introducir todo el código html en Strings dentro de las clases Java. De esta manera, cada vez que un usuario solicita realizar el test se cargan dichas plantillas y **se reemplazan los valores** por los correspondientes para el usuario y el test escogido, que en este caso es único.
- Se ha utilizado una **Cookie (correct-questions)** para guardar en el navegador las respuestas del usuario que deben ser corregidas. Así pues, cuando el servlet “view-quiz” recibe una solicitud, primero **pregunta si existe esa Cookie** y muestra la correción del test de acuerdo a dichos datos o un nuevo test. De esta forma **nos ahorramos el tener que duplicar código** o repetir el servlet cuando corregimos el test.