BOLETÍN 1

APLICACIONES DISTRIBUIDAS

ECPaquita74

Ejercicio 1

Para resolver este ejercicio, se crea el formulario *registro.html* que contemple todos los campos, requeridos y opcionales, necesarios para hacer el registro de una actividad. Además, se crea el bean *Actividad* que tiene como atributos los campos que aparecen en el formulario. Obligatoriamente, antes de hacer un submit del registro de la actividad, se han de cumplimentar obligatoriamente los campos para el código (`String`), la fecha (`LocalDate`), el número de plazas (`int`), el nombre (`String`) y el tipo de actividad (una opción del enumerado que se ha creado con el nombre *TipoActividad*). Estos campos estarán señalados en el formulario como un "required field". La descripción y la hora serán opcionales.

Por otro lado, creamos el `ServletGestionarCalendario`. Se puede acceder a este con la url homónima o con */calendario*. El `doPost` realizará el procesamiento del formulario para el registro, usando el contexto de la aplicación (obteniendo el atributo `actividades` con el `getServerConfig().getServerContext()`) en el que guardaremos un mapa con las actividades, indexadas por el código de cada una, que ha de ser único, para así tenerlas disponibles en toda la aplicación. Cuando se registre una actividad, se le indicará al usuario con una nueva pantalla con el mensaje correspondiente. Además, cuando se intente registrar una actividad con el mismo código que una que ya esté registrada, nos saltará un mensaje indicando de que no se ha podido realizar la operación y, tras unos segundos, se volverá a la pantalla del formulario para que pueda volver a intentarlo.

Ejercicio 2 y 4

Ahora, de cara a la resolución de este ejercicio, se realizan las siguientes modificaciones del código anterior. En primer lugar, en el `doGet` del servlet que habíamos creado, tendremos dos posibles caminos:

- Se introducen directamente los parámetros desde la url.
- Se obtiene la ruta del fichero *index.html*, en el que se estructura la petición de consulta de actividades con los campos obligatorios definidos como `mes` y `anyo`.

Para el procesamiento de la información de los parámetros de la petición que se realiza con el formulario, usamos el `doPost` de forma que, en primer lugar, obtengamos, si lo hay, el mapa de actividades de la aplicación. En este comprobamos la existencia de alguna actividad que coincida en mes y año con los datos de la consulta que se realiza. Si hay coincidencias, se muestra una pantalla con una lista (código, nombre) de las actividades encontradas. Si no hay coincidencias, se indica al usuario que no existen actividades con los datos introducidos y se vuelve al menú anterior.

Para el primer camino, la única diferencia respecto a lo que se acaba de describir es que el procesamiento se hace desde el propio `doGet`.

Por otra parte, además de las opciones de consultar actividades que se tratan, se incluye también en el formulario de este ejercicio, en forma de enlace, la opción de acceder al panel de

registro creado para el ejercicio anterior, que seguirá siendo tratado de la misma manera en el `doPost`.

Ejercicio 3

En este ejercicio crearemos un nuevo servlet (`ServletReservar`), accesible con la url con el mismo nombre o con */reservar*. Se ha creado un formulario (*`reserva.html`*) con un parámetro para el código de la actividad en la que se quiere realizar una reserva. Este archivo será abierto en el `doGet` con la ruta correspondiente. El parámetro `codigo` que se introduce es tratado en el `doPost`, de manera que se busque en el contexto la lista de actividades disponibles en la aplicación. Entre estas, si es que las hubiera, se busca la que coincide con el `codigo` de la reserva que quiere hacer el usuario para disminuir su número de plazas una unidad (siempre y cuando este fuera mayor que cero).

En caso de que la reserva se haya realizado con éxito, se indica al usuario con una nueva pantalla y se crea una lista de reservas asociadas a la sesión (cogiendo el atributo con ese nombre de `request.getSession(true)`, si no existe ya previamente) para que la reserva sea añadida al mismo. Por contra, si no existe la actividad (o aún no hay actividades registradas) o el número de plazas está agotado, se indica al usuario a través de una pantalla con el mensaje correspondiente a la causa del error, regresando tras unos segundos al formulario para que, si lo quisiera, pudiera intentar hacer otra reserva distinta.

Por otro lado, si cuando se hace el submit en el formulario de la reserva, el parámetro de la actividad está vacío (no se ha introducido ningún parámetro), se mostrará en una pantalla la lista de las actividades en las que se ha realizado una reserva previamente en la sesión del navegador correspondiente, tomando el atributo de la sesión como hemos hecho previamente. Si no hubiera reservas se le indica al usuario y, posteriormente, se le muestra de nuevo el formulario por si desea intentarlo de nuevo.

Los 3 archivos `*.html` que se adjuntan en la carpeta junto con esta memoria y el proyecto mencionados anteriormente y usados para los formularios se han de encontrar en la siguiente carpeta para que la aplicación funcione como es debido:
.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\ECPaquita74