Práctica 6: Nivel Web (JSF con Facelets y MVC)

Objetivo

En las prácticas anteriores hemos empleado servlets para el nivel web. Sin embargo los servlets son una tecnología de bajo nivel de abstracción y en las aplicaciones web orientadas a la presentación se pueden emplear otras tecnologías de más alto nivel de abstracción. En esta práctica vamos a migrar el código anterior a JSF con facelets y empleando MVC.

Proyecto

Al igual que en prácticas anteriores, en lugar de usar un servidor Java EE completo (como JBoss), podemos emplear un servidor de aplicaciones más sencillo (como Tomcat) al que añadir las librerías de JSF.

Para ello en primer lugar creamos un nuevo *Dynamic Web Project*. A continuación desde las propiedades del proyecto accedemos a *Project Facets* y habilitamos JSF 2.2. Al clickar en JSF Eclipse nos indica que se dispone de configuración adicional. Para ello clickamos en *further configuration available*, y realizamos las siguientes acciones:

1. Para elegir la librería de JSF seleccionamos *User Library* y posteriormente haciendo click sobre el icono de descargar, Eclipse descargará por nosotros la implementación de referencia de JSF (Mojarra).
2. Permitimos que incluya la configuración de JSF en el descriptor de despliegue web.xml.

Con esto ya tendremos un proyecto totalmente funcional para desarrollar aplicaciones JSF.

Finalmente incluimos también la librería JAX-RS y modificamos el web.xml como ya hicimos en la práctica anterior. Esto nos permitirá seguir usando REST para servir la lógica de negocio.

Controlador

El controlador encargado de comunicar vista y modelo ya nos viene proporcionado por el propio FacesServlet. Simplemente indicamos que gestione nuestras páginas *.xhtml modificando el web.xml (ya creado en el paso anterior):

1. Mapeamos las páginas *.xhtml al FacesServlet
2. Indicamos como welcome-file la página usuario.xhtml

Práctica 6: Nivel Web (JSF con Facelets y MVC)

Modelo

Como modelo que almacene los datos de la capa de presentación y consuma la lógica de negocio vamos a emplear dos managed beans, uno para dar de alta un nuevo producto en la tienda y otro que devuelva el listado de productos:

1. Creamos una clase ProductBean
   a) Su ámbito será el de la petición HTTP.
   b) Tendrá las propiedades correspondientes a referencia, tipo, nombre y precio (con sus correspondientes getters y setters). Esto puede hacerse extendiendo el bean que ya se creó en la práctica 3.
   c) Sobreescribe el método toString() para visualizar los datos del artículo.
   d) Incluye un método add() que dará de alta el objeto actual en la tienda. Para ello consumirá el servicio REST correspondiente al igual que anteriormente hacíamos desde el servlet de administrador.
2. Creamos una clase ShopBean
   a) Su ámbito será el de la aplicación.
   b) Tendrá la propiedad correspondiente a la lista de productos (con su correspondiente getter). Para ello consumirá el servicio REST correspondiente al igual que anteriormente hacíamos desde el servlet de usuario.

Vista

Vamos a crear tres páginas *.xhtml. Estas páginas JSF usarán Facelets (componentes "ui:") para emplear una plantilla layout.xhtml como se comenta a continuación:

1. templates/layout.xhtml
   a) Contendrá el código común a todas las páginas de la tienda. Por ejemplo: link al CSS, título en head y en body, etc.
   b) Las partes variables las insertará mediante ui:insert.
2. usuario.xhtml
   a) Hará uso de la plantilla anterior (ui:composition) definiendo (ui:define) un título y un contenido.
   b) Contendrá un componente de tipo h:dataTable. Este componente tomará los artículos de la propiedad correspondiente del bean shopBean.

Práctica 6: Nivel Web (JSF con Facelets y MVC)

c) Tened en cuenta que para que se muestre correctamente la tabla hay que incluir los componentes h:column que indiquen cabecera de la columna (f:facet) y valor (h:outputText).

3. administrador.xhtml

   a) Hará uso de la plantilla anterior (ui:composition) definiendo (ui:define) un título y un contenido.

   b) Mostrará un formulario para introducir los datos del nuevo artículo (salvo el código de referencia) empleando los distintos componentes de JSF (h:form, h:outputLabel, h:selectOneMenu, f:selectItem, h:inputText, etc.).

   c) Todos los campos son obligatorios y se debe mostrar un mensaje de error (h:message) en caso de que se dejen en blanco.

   d) Se incluye un componente h:commandButton para dar de alta el nuevo artículo. Para ello su actionListener debe llamar al método add() de productBean.

   e) En caso de que se haya introducido un nuevo artículo, se le notificará al usuario administrador sobre la misma página del formulario. Para eso empleamos un h:panelGroup con los h:outputText que se consideren necesarios para notificar al usuario el artículo dado de alta junto con sus datos. Este componente sólo se debe mostrar si el artículo ha sido dado de alta con éxito, condición que podemos indicar en su atributo rendered.

   f) Finalmente incluimos un h:outputLink que redirija a la página usuario.xhtml para que el administrador pueda comprobar el nuevo listado de artículos de la tienda sin más que hacer click en él.

Probad que al pedir las páginas xhtml de usuario y de administrador el resultado sea el esperado.