Práctica 3: Páginas Web Dinámicas

Objetivo

En esta práctica se introducirá el concepto de páginas dinámicas, tanto en el lado del cliente mediante javascript como en el lado del servidor mediante servlets corriendo sobre un servidor de aplicaciones.

Scripting del Lado del Cliente

Javascript

Vamos a hacer más robusta la página de la práctica anterior de administración de la tienda. Para ello en el HTML incluiremos código javascript que se ejecutará en el lado del cliente antes de enviar los datos para comprobar si todo es correcto.

1. Dentro de la sección "<head>" incluid un "<script>" con una función que se deberá ejecutar al hacer click en el botón enviar del formulario.
2. La función javascript comprobará que los campos de descripción y de precio no estén vacíos. Si falta alguno de los campos deberá resaltar el fondo del mismo en color amarillo y devolver "false".
3. A continuación comprobará que el campo de precio sea un número. Si no es así, mostrará un mensaje de alerta en una ventana y devolverá "false".
4. Finalmente si todo es correcto devolverá "true".

Scripting del Lado del Servidor

Servidor de Aplicaciones

En asignaturas anteriores hemos utilizado como entorno de desarrollo para Java la plataforma Oracle/Sun, concretamente el IDE NetBeans y el servidor de aplicaciones GlassFish. Sin embargo hay otros entornos para Java y concretamente una combinación para Java EE muy habitual en el mundo empresarial es emplear el IDE Eclipse (IBM) y el servidor de aplicaciones JBoss (RedHat).

En el caso de que no sea necesaria toda la funcionalidad de Java EE se puede recurrir a otros servidores de aplicaciones más ligeros, como Tomcat (Apache) o Jetty (Eclipse). Precisamente éste es el caso de la práctica actual con servlets, por lo que de momento emplearemos el servidor de aplicaciones Tomcat y dejaremos JBoss para las prácticas posteriores.

Práctica 3: Páginas Web Dinámicas

En primer lugar descargaremos[1] el tarball con la distribución binaria de Tomcat 7 y posteriormente lo debemos descomprimir en el directorio que consideréis. Con esto ya estará totalmente operativo el servidor de aplicaciones.

El IDE Eclipse ya está instalado de la práctica 1 y, como veremos a continuación, bastará con indicarle la ruta a la carpeta donde hemos descomprimido Tomcat para que el propio Eclipse se encargue de arrancarlo y publicar en él nuestros servlets.

Aplicación Web

Mediante servlets vamos a reemplazar las páginas HTML anteriores por una aplicación web que generará dinámicamente el código HTML en el lado del servidor. Para ello inicialmente partiremos de las páginas HTML estáticas que incluiremos en nuestro entorno Eclipse+Tomcat.

1. Desde Eclipse cread un "Dynamic Web Project" y en el target runtime dad a "New Runtime..." para registrar el servidor Tomcat que hemos descargado. Se os pedirá que introduzcáis la ruta donde lo hayáis descomprimido. El resto de opciones que se muestran al crear el proyecto podéis dejarlas con sus valores por defecto.
2. Tras completar este paso se habrá creado en la izquierda del IDE una estructura en forma de árbol con los componentes de vuestro proyecto. En concreto en "Java Resources/src" será donde se almacenarán los ficheros Java de código fuente que se vayan creando, y en "WebContent" estarán las páginas HTML y binarios Java que Eclipse publicará en el servidor de aplicaciones.
3. Copiad en WebContent las páginas HTML del usuario y administrador de la tienda y el fichero CSS. Si lo hacéis desde fuera de Eclipse acordaos de posteriormente ejecutar "Refresh" haciendo click con el derecho sobre vuestro proyecto de Eclipse.
4. Ejecutad el proyecto con "Run As/Run on Server" seleccionando Tomcat. Esto lo que hace es publicar vuestro WebContent en Tomcat (usando como contexto el nombre de vuestro proyecto Eclipse) y arrancar el servidor en el puerto 8080. Comprobad que vuestras páginas HTML siguen estando accesibles y funcionando.
5. A partir de este momento cada vez que cambiéis algún fichero del proyecto el propio Eclipse se encargará de republicarlo en el servidor. Esto puede comprobarse en la pestaña "Servers" que aparece en la perspectiva Java EE. En ocasiones el cambio realizado requiere reiniciar el propio servidor de aplicaciones, cosa que puede realizarse desde la misma pestaña "Servers" cuando Eclipse indique "restart".

1 http://tomcat.apache.org/download-70.cgi

Práctica 3: Páginas Web Dinámicas

Hasta aquí no hay diferencia con el primer apartado de la práctica de scripting del lado del cliente, así que vamos a empezar a añadir el código dinámico del lado del servidor.

Servlet de Administrador

La página del administrador la vamos a comunicar con un servlet. Para ello:

1. Haciendo click con el derecho sobre vuestro proyecto dad a "New/Servlet" invocando así el asistente de Eclipse para la creación de servlets. El servlet deberá atender las peticiones por POST, indicando un mensaje de error en caso de que se le llame por GET.
2. En la página HTML de administrador actualizad el código del formulario para que al hacer click envíe los datos por POST a nuestro servlet.
3. En caso de que se invoque el servlet por POST significará que se nos está llamando a través del formulario anterior, y podremos leer los valores introducidos por el usuario. Con estos valores podríamos realizar la operación que queramos, que en nuestro caso va a ser simplemente generar una nueva página con un mensaje similar al siguiente:

   Artículo *Blade Runner* dado de alta en la categoría *DVD* con un precio de *30* euros.
4. Finalmente haced que el artículo dado de alta se guarde en un fichero datos.csv donde cada línea contenga un producto de la tienda con los siguientes campos separados por comas: referencia (automático), categoría, artículo, precio. Si el fichero no existe habrá que crearlo.

Servlet de Usuario

Finalmente reemplazad la página HTML del usuario por otro servlet que muestre de forma dinámica el listado de artículos (datos.csv) que previamente han sido dados del alta por el administrador.

Patrones de Diseño

Como la aplicación aún es muy simple, no se puede apreciar claramente las ventajas de los patrones de arquitectura, pero sí resulta de utilidad separar el modelo de datos y la lógica de negocio en un bean que sea utilizado por ambos servlets.