Ejercicios de herramientas para aplicaciones web

Índice

1. Definir un servidor web en WebTools

Vamos a ir construyendo y probando una aplicación web compleja utilizando WebTools. Para ello, lo primero que tenemos que hacer es tener nuestro servidor web configurado para poderlo utilizar desde WebTools.

Este primer ejercicio consiste en que definas un nuevo servidor Tomcat 5.5 en la vista *Servers* de la perspectiva J2EE (si no lo tienes ya creado), siguiendo los pasos que se explican en la parte de teoría.

Importante: En los siguientes ejercicios iremos creando poco a poco un proyecto de aplicación web, y añadiendo funcionalidades. En cada paso, se te pedirá que incorpores al proyecto determinados ficheros que se te proporcionan en la plantilla. Procura copiar SOLO los ficheros que se te indican, y no los demás, hasta que se te requieran, para evitar que la aplicación funcione incorrectamente.

2. Creación y prueba de un proyecto web

A continuación, crearemos un proyecto web dinámico (*Dynamic Web Project*), con el nombre **servd-web-sesion4**. En el asistente para crearlo, pondremos como *Context Root* (paso 3 del asistente), **sw04**, y dejaremos que nos cree las carpetas por defecto de `src` y `WebContent`.

Una vez lo tenemos creado, vamos a ir copiando en los lugares respectivos los siguientes ficheros que tenemos en la plantilla:

- Crea un paquete llamado **`es.ua.jtech.servdweb.sesion4`** en la carpeta **`src`**
- Copia en ese paquete los ficheros Java **`ConversorServlet`**, **`BDServlet`** y **`LoggingServlet`** de la plantilla. Son dos servlets que tendrás que cargar más adelante.
- Copia el fichero **`index.htm`** en **`WebContent`**
- Copia el fichero **`web.xml`** de la plantilla en la carpeta **`WebContent/WEB-INF`**
- Copia los ficheros JAR (**`commons-logging-1.1.jar`** y **`log4j-1.2.8.jar`**) en la carpeta **`WebContent/WEB-INF/lib`**, y el fichero **`mysql-connector-java-5.0.3-bin.jar`** en la carpeta **`common/lib`** de Tomcat

Posiblemente tendrás que añadir al *Build Path* del proyecto la librería `commons-logging-1.1.jar` que hemos copiado antes, para que compile el servlet `LoggingServlet`.

Una vez lo tengas todo en su sitio y no haya errores, despliega la aplicación en el servidor de prueba. Para ello, ve con el botón derecho del ratón sobre el proyecto, y eliges *Run - Run As - Run on Server*.

Debería aparecer la página `index.htm` en un navegador, con el conversor de euros a pesetas visto en otros ejemplos.

3. Configuración de un pooling de conexiones en una aplicación web

Sobre la aplicación web que tenemos ya creada del ejercicio anterior, vamos a hacer las siguientes modificaciones:

- Crea una carpeta **`bd`** en la carpeta raíz del proyecto
- Copia en esta carpeta los ficheros **`prueba.sql`** y **`build.xml`** de la carpeta **`bd`** de la plantilla (¡OJO! No copies aún el fichero `build.xml` de la raíz de la plantilla. Ese se te pedirá más adelante).
- Ejecuta el script `build.xml` que acabas de copiar, para que instale la base de datos en el servidor MySQL de tu máquina
- Crea un fichero **`context.xml`** en la carpeta **`WebContent/META-INF`** del proyecto, con los elementos necesarios para configurar el pooling. Además, añade en el fichero **`web.xml`** de **`WebContent/WEB-INF`** una etiqueta **`resource-ref`** con las subetiquetas necesarias para referenciar al pooling creado en el fichero `context.xml` anterior, tal y como se ha explicado en el apartado de teoría. Llama al pooling (atributo `name`), **`jdbc/prueba`**.
- Vuelve a ejecutar la aplicación web sobre el servidor, y carga la página **`http://localhost:8080/sw04/paginabd`**. Debería mostrar un listado de nombres y descripciones que existen en la base de datos `prueba` que acabas de instalar.

4. Configuración del logging en una aplicación web

La clase `LoggingServlet` que instalaste en el segundo ejercicio es un servlet que saca unos mensajes de log, pero como la aplicación no está configurada, no podemos tener control sobre estos mensajes. Se pide:

- Crea una carpeta de fuentes llamada **`resources`** en el proyecto web.
- Crea en ella dos ficheros, **`commons-logging.properties`** y **`log4j.properties`**, para que, utilizando Log4J, saque por pantalla los mensajes de log de nivel INFO en adelante, y con el siguiente formato:

  Prioridad del mensaje - Texto del mensaje - Fecha y hora (formato dd/MM/aa hh:mm) -

salto de línea

Para probar el servlet, accede a la dirección
`http://localhost:8080/sw04/logging`

5. (*) Pruebas en aplicaciones web con Cactus

Finalmente, vamos a incorporar un servlet y su caso de prueba (ambos ya implementados) a nuestra aplicación, y a configurar Cactus para que podamos ejecutar el caso de prueba. Para ello, los pasos a seguir son:

- Copia la clase **`PruebaServlet`** de la plantilla al paquete de servlets que ya tienes en **`src`**
- Crea una carpeta de fuentes llamada **`test`**, y copia en ella la clase **`PruebaServletTest`** de la plantilla. Este será el caso de prueba del servlet que acabas de copiar.
- Copia todos los ficheros JAR de la carpeta **`cactus`** de la plantilla a la carpeta **`WebContent/WEB-INF/lib`**. Son las librerías que necesitarás para poder ejecutar las pruebas de Cactus. Añádelas también al *Build Path* del proyecto, para poder compilar y ejecutar las pruebas.
- Crea un fichero **`cactus.properties`** en la carpeta **`resources`** que indique la ruta de contexto de la aplicación a probar:
  ```
  cactus.contextURL=http://localhost:8080/sw04
  ```
- Añade al fichero **`Web-Content/WEB-INF/web.xml`** las líneas necesarias para definir los redirectores que usará Cactus:

```
<filter>
        <filter-name>FilterRedirector</filter-name>
<filter-class>org.apache.cactus.server.FilterTestRedirector</filter-class>
</filter>
<filter-mapping>
        <filter-name>FilterRedirector</filter-name>
        <url-pattern>/FilterRedirector</url-pattern>
</filter-mapping>
<servlet>
        <servlet-name>ServletRedirector</servlet-name>
<servlet-class>org.apache.cactus.server.ServletTestRedirector</servlet-class>
</servlet>
<servlet>
        <servlet-name>JspRedirector</servlet-name>
        <jsp-file>/jspRedirector.jsp</jsp-file>
</servlet>
<servlet-mapping>
        <servlet-name>ServletRedirector</servlet-name>
        <url-pattern>/ServletRedirector</url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
        <servlet-name>JspRedirector</servlet-name>
        <url-pattern>/JspRedirector</url-pattern>
</servlet-mapping>
```

- Detén el servidor en la vista *Servers*. Ejecuta ahora de nuevo la aplicación web (haz un *Run on Server*) para rearrancarlo todo.
- Ejecuta la clase `PruebaServletTest` como un test de JUnit (*Run As - JUnit test*) y comprueba que las pruebas son satisfactorias.

6. PARA ENTREGAR

Como entrega de esta sesión, empaqueta en un fichero ZIP el proyecto web dinámico que has creado, con todas las funcionalidades que hayas ido añadiendo a lo largo de todos estos ejercicios.