Diseño básico de páginas JSP

Gestión de errores en páginas JSP

Hasta ahora lo que hemos visto básicamente de las páginas JSP es su estructura, formada por directivas, declaraciones, expresiones y scriptlets. Ahora procederemos a ampliar algunos aspectos fundamentales que formarán parte de la codificación de las páginas JSP en nuestras aplicaciones web. Un apartado importante es la gestión de los errores que se producen en las páginas JSP, entre otras cosas, debido a que en los scriptlets vamos a introducir código Java que puede lanzar excepciones.

Como en todo código java, cuando se produce un error, salta una excepción. Esta excepción podremos capturarla o no. Si no lo hacemos, aparecerá en la salida estándar el resultado de la misma. Capturando las excepciones podremos hacer que ser recupere la aplicación de un error, o al menos, informar adecuadamente de la naturaleza de la misma (causa, cómo evitarla, etc.). Hay varias estrategias para abordar esto en el diseño de páginas JSP. Una es procesar en la misma página la excepción, introduciendo para ello, el correspondiente bloque *try-catch*. En aplicaciones de cierta envergadura, comprobaremos que esta estrategia obliga a generar demasiado código, nada reutilizable, y que su mantenimiento se torna un tanto pesado, debido a que la gestión de las excepciones está esparcido a lo largo de todo el código fuente.

JSP, sin embargo, dispone de un mecanismo para evitar las incomodidades de la estrategia anterior, se trata de la administración de errores mediante páginas JSP de error. Para ello se debe crear la página que se va a mostrar cuando se produzca una situación de error. Esta página incluirá la siguiente directiva:

```
<%@ page isErrorPage=”true” %>
```

Esta directiva indica que se trata de una página de error. El contenedor de servlet proporcionará a esta página un objeto (implícito) de error, se trata de *exception*. Este objeto encapsula el error generado en la página llamada. Para determinar cuál ha sido la página que ha generado el error, y que por tanto, ha *disparado* la ejecución de la página de error, se emplea el siguiente scriptlet:

```
<% String org= request.getParameter(“pagOrigen”); %>
```

La siguiente página JSP permite procesar/informar del error que se comete cuando se ha introducido un error al insertar datos numéricos en un formulario, se introduce una cadena que no puede convertirse a numérico. La página la hemos llamado *errorDatos.jsp*:

```
<%@ page isErrorPage="true" %>
<%@ page contentType="text/html;charset=ISO-8859-15"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
    <title>Error en introducción de datos.</title>
  </head>
  <body>
    <% String pagOrigenError = (String) request.getParameter("pagOrigen"); %>
    <P>Ha ocurrido un error en la p&aacute;gina:<%=pagOrigenError%></P>
    <P>Descripci&oacute;n de la excepci&oacute;n:</P>
    <P> </P>
    <%= exception.toString() %>
  </body>
</html>
```

Y la página que contiene el formulario de introducción de datos deberá incluir la siguiente directiva:

```
<%@ page errorPage=”errorDatos.jsp?pagOrigen=datos.jsp” %>
```

Incluir archivos

Cuando se habla de incluir archivos en una página JSP nos estamos refiriendo a insertar en esa página el contenido de uno o varios archivos. Hay dos mecanismos para realizar esto, uno permite incluir el contenido de un archivo en tiempo de ejecución, el otro en tiempo de compilación. Cada uno tiene sus ventajas y sus inconvenientes.

Cuando se incluye en tiempo de compilación, el contenido se inserta en la página JSP antes de ser compilado el servlet. Si se incluye en tiempo de ejecución, la página JSP no incluirá el archivo hasta después de ser compilado el servlet y recibir la solicitud durante la ejecución de su instancia correspondiente.

Incluir en tiempo de compilación
Se emplea la directiva *include*. Se recomienda emplear este método cuando el contenido del archivo no varía mucho en el tiempo. La sintaxis es la siguiente:

```
<%@ include file="nombrearchivo" %>
```

Suele emplearse este sistema, entre otros, para proveer de una cabecera y un pie comunes a un conjunto de páginas de una aplicación.

Incluir en tiempo de ejecución
Si el contenido del archivo varía y se precisa recoger este cambio de manera inmediata en las páginas JSP, entonces hemos de optar por incluirlo en tiempo de ejecución. La sintaxis es:

```
<jsp:include page="nombrearchivo" flush="true" />
```

Como se observa, esta directiva se emplea con su sintaxis XML. El atributo *flush* obliga a vaciar el búfer de salida del servlet antes de iniciar la inclusión. Es obligatorio. La salida de la página se inserta en la salida de la página JSP contenedora. Sin embargo se debe tener en cuenta que el archivo incluido no puede modificar ninguno de los encabezados HTTP. Con este sistema también se posibilita proporcionar parámetros a la página incluida, útil cuando la página es JSP. Ejemplo:

```
<jsp:include page="pagina.jsp" flush="true">
    <jsp:param name="usuario" value="eduardo"/>
 </jsp:include>
```

Formularios de datos
Los formularios se constituyen en el pilar fundamental de transferencia de datos desde la máquina cliente hacia la aplicación web. Los formularios pueden ser enviados tanto a una página JSP como a un servlet. Cuando el formulario se envía al servidor de aplicaciones, lo primero que deberá realizar la página JSP (o el servlet) es recoger los datos proporcionados por el usuario. Para ello se recurre al objeto *request*, utilizando el método *getParameter()*. En el apartado anterior de *Gestión de errores* se aprecia su uso.

Cuando un campo de un formulario puede contener más de un valor (lista de opciones múltiples), puede emplearse el método *getParameterValues()* para devolver una cadena con los valores correspondientes a ese parámetro. La manera en que se debería proceder para la obtención de dichos valores es la siguiente:

```
<% String valores[] = request.getParameterValues("listaOpM");
      for (int i=0; i<valores.length;i++)
            out.println("listaOpM: "+valores[i]);
%>
```

Finalmente, existe otro método que posibilita recabar todos los nombres de los parámetros recibidos, y que es útil cuando se desconocen estos nombres. El método es *getParametersNames()*, y un ejemplo de su uso es el siguiente:

```
<%
```

```
String nombreP;
java.util.Enumeration nombresParams = request.getParameterNames();
while (nombresParams.hashMoreElements()) {
    nombreP = (String) nombresParams.nextElement();
    out.println("El parámetro "+nombreP+" tiene valor: "+request.getParameter(nombreP);
}
%>
```

Sesiones

La etiqueta *<jsp:include>* comentada en apartado anterior ofrecía la posibilidad de hacer referencia al objeto *request* en una página diferente a la página que recibió del contenedor ese objeto. Por tanto, se trata de un mecanismo que permite transferir un objeto *request* entre dos páginas JSP. Existe una etiqueta de función similar, *<jsp:forward>*, que se diferencia de la anterior en que esta última no devuelve el control a la página de origen. Pero sí permite, tras realizar el procesamiento con el objeto *request*, moverlo (transferirlo) a otra página, ya sea JSP o servlet. El hecho de que entre dos páginas pueda enviarse el objeto *request* abre la puerta a un aspecto importante del diseño de las aplicaciones web: el seguimiento de las solicitudes efectuadas por un mismo usuario. Recordemos que HTTP es un protocolo sin mantenimiento del estado, por lo que dos solicitudes efectuadas consecutivamente por un usuario sobre un servidor, se consideran como independientes. Esto hace que sea imposible que una página JSP pueda, a priori, trabajar con los datos proporcionados por el usuario en otra página JSP diferente. Una solución obvia es almacenar los datos en una base de datos, de manera que cualquier página JSP, mire primero en esa base de datos. Sin embargo, determinar qué datos pertenecen a un usuario y cuáles a otro, supone una cierta complejidad de procesamiento.

Para resolver esta problemática, la tecnología de JSP y servlets dispone del concepto de *sesión*. Se puede considerar una sesión como un objeto que encapsula el seguimiento de las solicitudes de un usuario, identificando éstas respecto de las del resto de los usuarios. Para ello, el contenedor web proporciona un objeto, denominado *session*, a las páginas JSP y servlets que lo requieran, que contiene la información asociada a la sesión de un determinado usuario. De esta manera, todas las solicitudes efectuadas por un usuario quedarán confinadas en el marco de una sesión concreta, diferente de las sesiones del resto de los usuarios. Este objeto *session* permitirá el almacenamiento y recuperación de datos escritos en cualquier momento por el usuario que interacciona con la aplicación web.

La primera vez que un usuario solicita una página JSP, se crea la sesión en el servlet (recordemos que las páginas JSP son, al fin y al cabo, servlets). La sesión tendrá un identificador asociado a ella, y será transferido desde el servlet al usuario a través de una *cookie* HTTP. Cada vez que el usuario solicita una página JSP, el identificador de sesión se transfiere con la solicitud, y por lo tanto, se asocia la sesión adecuada con el usuario.

Aunque todos los servidores de aplicaciones proporcionan esta técnica, algunos no crean automáticamente la sesión para el usuario. En este caso se necesitará obtener la sesión llamando al método *getSession()* sobre el objeto *request*. Este método puede recibir un valor booleano que indica si se crea una nueva sesión si no existiera ya una. La sintaxis para obtener el objeto *session* es la siguiente:

```
<% HttpSession session = request.getSession(true); %>
```

El objeto *session* creado posibilita escribir, obtener y eliminar atributos asociados al mismo. Estos atributos pueden ser variables o incluso objetos, tan complejos como se desee. Cada atributo se almacenará con un nombre y tendrá asociado un valor. Para añadir (escribir) un atributo a la sesión se emplea el método *setAttribute()* con la siguiente sintaxis:

```
<% session.setAttribute("<nombreAtributo>",<valor>); %>
```

Y para recuperar el valor:

```
<% <tipoAtributo>atributo = (<tipoAtributo>) session.getAttribute("<nombreAtributo>");%>
```

Por ejemplo, para recuperar el valor de un atributo denominado *nombre* de tipo *String*, se hace lo siguiente:

<% String nombAtributo = (String) session.getAttribute("nombre"); %>

Y para eliminar el atributo considerado en el ejemplo anterior:

<% session.removeAttribute("nombre"); %>

Cada uno de estos métodos puede generar una excepción *IllegalStateException* cuando se llama a cualquiera de ellos en una sesión no válida, por ejemplo, cuando el servlet ha eliminado dicha sesión (que le corresponde efectuarla al motor del servlet, el contenedor web).

Ejemplos de sesiones

Seguidamente voy a mostrar algunos ejemplos básicos de uso de sesiones. Cada ejemplo se puede recopilar en un proyecto. Puede diseñarse en *Eclipse*.

Código de ejemplo nº 1[36]
El siguiente código muestra un ejemplo de uso de sesiones. La aplicación se compone de un formulario, mediante el cual el usuario introduce un identificador, y al enviar el formulario al servidor, éste asocia una sesión al usuario. Entonces el usuario dispondrá de tres enlaces desde los que podrá realizar procesos de altas, bajas y consultas. También se incluye un botón para que el usuario finalice explícitamente la sesión, en cuyo caso se le envía de nuevo a la página del formulario para introducir un nuevo identificador. Los procesos de altas, bajas y consultas no están implementados. En la página que muestra los enlaces a los procesos señalados, se ofrece información diversa asociada a la sesión: identificador de sesión, identificador de usuario, fecha y hora de creación de sesión. Si durante una sesión, el usuario regresa a la página del formulario y vuelve a enviar un nuevo identificador, la aplicación no permitirá crear una nueva sesión, ya que para ello está disponible el botón de fin de sesión. Si se recarga cualquiera de las otras páginas, se aplica el mismo criterio.

ejSesion.html

```
1. <html>
2.   <head>
3.     <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-15"></meta>
4.     <title>Ejemplo 1 de uso de sesiones</title>
5.   </head>
6.   <body>
7.     <DIV align="center">
8.       <STRONG><FONT color="#ff0033">Usando sesiones</FONT></STRONG>
9.     </DIV>
10.    <hr/>
11.    <form action="ejSesion.jsp" method="post">
12.      Introduzca su nombre:
13.      <input type="text" name="usuario"/>
14.      <input type="submit" value="Enviar"/>
15.    </form>
16.  </body>
17.</html>
```

ejSesion.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 1 de uso de sesiones</title>
6.   </head>
7.   <body>
8.     <DIV align="center">
9.       <P>
10.        <STRONG>Ejemplo de uso de sesiones</STRONG>
```

36 Proyecto EjSesion.

```
11.       </P>
12.
13.       <%
14.         Usuario datSesionUser;
15.         String usuario = request.getParameter("usuario");
16.         String nuevaSesion = request.getParameter("nuevaSesion");
17.
18.         if (nuevaSesion!=null)
19.           if (nuevaSesion.equals("true"))
20.           {
21.             session.invalidate();
22.       %>
23.             <jsp:forward page="ejSesion.html" />
24.       <%
25.           }
26.         if (usuario != null)
27.         {
28.           if (session.isNew())
29.           {
30.             out.println("Sesión nueva.");
31.             datSesionUser = new Usuario(usuario,session);
32.             session.setAttribute("usuarioSession",usuario);
33.             session.setAttribute("datSesionUser",datSesionUser);
34.           }
35.         }
36.       %>
37.       <P>Sesión actual: <%= session.getId()%>
38.       Usuario: <%= datSesionUser.getIdUsuario()%>
39.       <P>
40.       Fecha y hora de inicio de la sesión: <%=datSesionUser.getFecha() %>
41.       </P>
42.       <P>
43.      Tiempo     máximo     de     inactividad     permitido     por     una     sesión:     <
%=datSesionUser.getSesion().getMaxInactiveInterval()%> segundos.
44.       </P>
45.       <hr/>
46.     </DIV>
47.     <DIV align="center">
48.       <P align="left">Escoja dónde desea acceder:</P>
49.       <DIV align="left">
50.         <OL>
51.           <LI><a href="ejSesionAltas.jsp">Altas</a></LI>
52.           <LI><a href="ejSesionBajas.jsp">Bajas</a></LI>
53.           <LI><a href="ejSesionConsultas.jsp">Consultas</a></LI>
54.           <LI>
55.             <form method="post" action="ejSesion.jsp">
56.               <input type="hidden" name="nuevaSesion" value="true"/>
57.               <input type="submit" value="Finalizar Sesión" name="finsesion"/>
58.             </form>
59.           </LI>
60.         </OL>
61.         <hr/>
62.       </DIV>
63.     </DIV>
64.   </body>
65.</html>
```

ejSesionAltas.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 1 de uso se sesiones - Altas</title>
6.   </head>
7.   <body>
8.     <DIV align="center">
9.       <P>
10.        <STRONG><FONT color="#ff0033">Altas</FONT></STRONG>
11.      </P>
12.      <P>Identificador de sesión: <%=session.getId()%>
```

```
13.       </P>
14.       <P>Usuario: <%=session.getAttribute("usuarioSession")%>
15.       </P>
16.       <P>
17.          Fecha y hora de inicio de sesión:
18.<%=((Usuario)session.getAttribute("datSesionUser")).getFecha()%>
19.       </P>
20.       <hr/>
21.       <P><a href="ejSesion.jsp">Volver</a></P>
22.     </DIV>
23.   </body>
24.</html>
```

ejSesionBajas.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 1 de uso de sesiones - Bajas</title>
6.   </head>
7.   <body>
8.       <DIV align="center">
9.       <P>
10.        <STRONG><FONT color="#ff0033">Bajas</FONT></STRONG>
11.      </P>
12.      <P>Identificador de sesión: <%=session.getId()%>
13.      </P>
14.      <P>Usuario: <%=session.getAttribute("usuarioSession")%>
15.      </P>
16.      <P>
17.        Fecha y hora de inicio de sesión:
18.<%=((Usuario)session.getAttribute("datSesionUser")).getFecha()%>
19.      </P>
20.      <hr/>
21.      <P><P><a href="ejSesion.jsp">Volver</a></P>
22.    </DIV>
23.  </body>
24.</html>
```

ejSesionConsultas.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 1 de uso de sesiones - Consultas</title>
6.   </head>
7.   <body>
8.       <DIV align="center">
9.       <P>
10.        <STRONG><FONT color="#ff0033">Consultas</FONT></STRONG>
11.      </P>
12.      <P>Identificador de sesión: <%=session.getId()%>
13.      </P>
14.      <P>Usuario: <%=session.getAttribute("usuarioSession")%>
15.      </P>
16.      <P>
17.         Fecha y hora de inicio de sesión:
18.<%=((Usuario)session.getAttribute("datSesionUser")).getFecha()%>
19.      </P>
20.      <hr/>
21.      <P><P><a href="ejSesion.jsp">Volver</a></P>
22.    </DIV>
23.  </body>
24.</html>
```

ejsesion.util.Usuario.java

```
1. import javax.servlet.http.HttpSession;
2. import java.util.*;
```

```
3.
4. public class Usuario
5. {
6.   String idUsuario;
7.   String idSesion;
8.   HttpSession sesion;
9.   Date fecha;
10.  public Usuario(String idUsuario, HttpSession sesion)
11.  {
12.    this.idUsuario = idUsuario;
13.    this.idSesion = sesion.getId();
14.    this.sesion = sesion;
15.    this.fecha = new Date();
16.  }
17.
18.  public void setIdUsuario(String idUsuario)
19.  {
20.    this.idUsuario = idUsuario;
21.  }
22.
23.
24.  public String getIdUsuario()
25.  {
26.    return idUsuario;
27.  }
28.
29.
30.  public void setIdSesion(String idSesion)
31.  {
32.    this.idSesion = idSesion;
33.  }
34.
35.
36.  public String getIdSesion()
37.  {
38.    return idSesion;
39.  }
40.
41.
42.  public void setSesion(HttpSession sesion)
43.  {
44.    this.sesion = sesion;
45.  }
46.
47.
48.  public HttpSession getSesion()
49.  {
50.    return sesion;
51.  }
52.
53.
54.  public void setFecha(Date fecha)
55.  {
56.    this.fecha = fecha;
57.  }
58.
59.
60.  public Date getFecha()
61.  {
62.    return fecha;
63.  }
64.}
```

Código de ejemplo nº 2[37]

En este ejemplo, la aplicación muestra un formulario que solicita un identificador de usuario. Si el usuario envía el formulario sin rellenar el campo, la aplicación le muestra de nuevo el formulario, y procederá de esta manera hasta que el usuario envíe algo. Si al realizar el envío se detecta que se va a iniciar una sesión,

37 Proyecto EjSesion2.

entonces la aplicación crea un objeto que encapsula los datos del usuario, y lo añade a la sesión. En caso de que se detectara que la sesión ya existía, se recurrirá a los datos ya almacenados en ella. Si un usuario, una vez iniciada una sesión, intenta de nuevo enviar un identificador, éste será ignorado. Sólo puede contemplarse un nuevo identificador después de que el usuario haya pulsado el botón de *Finalizar Sesión*. Cualquier página puede ser recargada, teniendo en cuenta que esta acción no supone reiniciar sesión. Si un usuario, al conectarse por primera vez, envía el formulario sin rellenar, aunque se haya creado la sesión, la aplicación la eliminará, y retornará la página del formulario al usuario. Se ha desarrollado una clase denominada *Usuario*, que se ha englobado en un paquete denominado *ejsesion2.util*. Se ha llevado a cabo para hacer este proyecto portable a cualquier servidor de aplicaciones, ya que hay algunos que requieren que las clases de una aplicación web estén ubicadas en paquetes. Además, se le ha aplicado a esta clase la interfaz *Serializable*, que aunque no lo requiere, posibilitaría almacenar la información del usuario de forma persistente más allá del tiempo de vida de la sesión. El código es el siguiente:

ejsesion2.html

```
1. <html>
2.   <head>
3.     <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-15"></meta>
4.     <title>Ejemplo 2 de uso de sesiones</title>
5.   </head>
6.   <body>
7.     <DIV align="center">
8.       <STRONG><FONT color="#ff0033">Usando sesiones</FONT></STRONG>
9.     </DIV>
10.    <hr/>
11.    <form action="ejSesion2.jsp" method="post">
12.      Introduzca su nombre:
13.      <input type="text" name="usuario"/>
14.      <input type="submit" value="Enviar"/>
15.    </form>
16.  </body>
17.</html>
```

ejSesion2.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 2 de uso de sesiones</title>
6.   </head>
7.   <body>
8.     <DIV align="center">
9.       <P>
10.        <STRONG>Ejemplo de uso de sesiones</STRONG>
11.      </P>
12.
13.      <%
14.        ejsesion2.util.Usuario datSesionUser;
15.        String nuevaSesion = request.getParameter("nuevaSesion");
16.        String usuario = request.getParameter("usuario");
17.
18.        if (nuevaSesion!=null)
19.          if (nuevaSesion.equals("true"))
20.          {
21.            session.invalidate();
22.      %>
23.            <jsp:forward page="ejSesion2.html" />
24.      <%
25.          }
26.        if (session.getAttribute("datSesionUser")!=null)
27.        {
28.          datSesionUser = (ejsesion2.util.Usuario) session.getAttribute("datSesionUser");
29.        }
30.        else
31.        {
32.          if (usuario!=null)
33.          {
34.            if (usuario.equals(""))
```

```
35.              usuario=null;
36.          }
37.          else
38.            out.println("Usuario null.");
39.        }
40.
41.        if (session.isNew())
42.          if ((usuario != null) && (usuario!=""))
43.          {
44.            out.println("Sesión nueva.");
45.            datSesionUser = new ejsesion2.util.Usuario(usuario,session);
46.            session.setAttribute("usuarioSession",usuario);
47.            session.setAttribute("datSesionUser",datSesionUser);
48.                                                    out.println(((ejsesion2.util.Usuario)
session.getAttribute("datSesionUser")).getIdUsuario());
49.          }
50.          else
51.          {
52.            session.invalidate();
53.      %>
54.            <jsp:forward page="ejSesion2.html" />
55.      <%
56.          }
57.      %>
58.      <P>Sesi&oacute;n actual: <%= session.getId()%>
59.                                              Usuario:       <%=       ((ejsesion2.util.Usuario)
session.getAttribute("datSesionUser")).getIdUsuario()%>
60.      <P>
61.                Fecha   y   hora   de   inicio   de   la   sesión:   <%=((ejsesion2.util.Usuario)
session.getAttribute("datSesionUser")).getFecha() %>
62.      <P>
63.                    Tiempo   máximo   de   inactividad   permitido   por   una   sesión:   <
%=session.getMaxInactiveInterval()%> segundos.
64.      </P>
65.      <hr/>
66.    </DIV>
67.    <DIV align="center">
68.      <P align="left">Escoja dónde desea acceder:</P>
69.      <DIV align="left">
70.        <OL>
71.          <LI><a href="ejSesion2Altas.jsp">Altas</a></LI>
72.          <LI><a href="ejSesion2Bajas.jsp">Bajas</a></LI>
73.          <LI><a href="ejSesion2Consultas.jsp">Consultas</a></LI>
74.          <LI>
75.            <form method="post" action="ejSesion2.jsp">
76.              <input type="hidden" name="nuevaSesion" value="true"/>
77.              <input type="submit" value="Finalizar Sesión" name="finsesion"/>
78.            </form>
79.          </LI>
80.        </OL>
81.        <hr/>
82.      </DIV>
83.    </DIV>
84.  </body>
85.</html>
```

ejSesion2Altas.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 2 de uso se sesiones - Altas</title>
6.   </head>
7.   <body>
8.     <DIV align="center">
9.       <P>
10.        <STRONG><FONT color="#ff0033">Altas</FONT></STRONG>
11.      </P>
12.      <P>Identificador de sesi&oacute;n: <%=session.getId()%>
13.      </P>
```

```
14.       <P>Usuario: <%=session.getAttribute("usuarioSession")%>
15.       </P>
16.       <P>
17.         Fecha y hora de inicio de sesión:
18.<%=((ejsesion2.util.Usuario)session.getAttribute("datSesionUser")).getFecha()%>
19.       </P>
20.       <hr/>
21.       <P><a href="ejSesion2.jsp">Volver</a></P>
22.     </DIV>
23.   </body>
24.</html>
```

ejSesion2Bajas.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 2 de uso de sesiones - Bajas</title>
6.   </head>
7.   <body>
8.       <DIV align="center">
9.       <P>
10.        <STRONG><FONT color="#ff0033">Bajas</FONT></STRONG>
11.      </P>
12.      <P>Identificador de sesión: <%=session.getId()%>
13.      </P>
14.      <P>Usuario: <%=session.getAttribute("usuarioSession")%>
15.      </P>
16.      <P>
17.        Fecha y hora de inicio de sesión:
18.<%=((ejsesion2.util.Usuario)session.getAttribute("datSesionUser")).getFecha()%>
19.      </P>
20.      <hr/>
21.      <P><P><a href="ejSesion2.jsp">Volver</a></P>
22.    </DIV>
23.  </body>
24.</html>
```

ejSesion2Consultas.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Ejemplo 2 de uso de sesiones - Consultas</title>
6.   </head>
7.   <body>
8.       <DIV align="center">
9.       <P>
10.        <STRONG><FONT color="#ff0033">Consultas</FONT></STRONG>
11.      </P>
12.      <P>Identificador de sesión: <%=session.getId()%>
13.      </P>
14.      <P>Usuario: <%=session.getAttribute("usuarioSession")%>
15.      </P>
16.      <P>
17.        Fecha y hora de inicio de sesión:
18.<%=((ejsesion2.util.Usuario)session.getAttribute("datSesionUser")).getFecha()%>
19.      </P>
20.      <hr/>
21.      <P><P><a href="ejSesion2.jsp">Volver</a></P>
22.    </DIV>
23.  </body>
24.</html>
```

ejsesion2.util.Usuario.java

```
1. package ejsesion2.util;
2. import javax.servlet.http.HttpSession;
3. import java.util.*;
```

```
4. import java.io.*;
5.
6. public class Usuario implements Serializable
7. {
8.   String idUsuario;
9.   String idSesion;
10.  HttpSession sesion;
11.  Date fecha;
12.  public Usuario(String idUsuario, HttpSession sesion)
13.  {
14.    this.idUsuario = idUsuario;
15.    this.idSesion = sesion.getId();
16.    this.sesion = sesion;
17.    this.fecha = new Date();
18.  }
19.
20.  public void setIdUsuario(String idUsuario)
21.  {
22.    this.idUsuario = idUsuario;
23.  }
24.
25.
26.  public String getIdUsuario()
27.  {
28.    return idUsuario;
29.  }
30.
31.
32.  public void setIdSesion(String idSesion)
33.  {
34.    this.idSesion = idSesion;
35.  }
36.
37.
38.  public String getIdSesion()
39.  {
40.    return idSesion;
41.  }
42.
43.
44.  public void setSesion(HttpSession sesion)
45.  {
46.    this.sesion = sesion;
47.  }
48.
49.
50.  public HttpSession getSesion()
51.  {
52.    return sesion;
53.  }
54.
55.
56.  public void setFecha(Date fecha)
57.  {
58.    this.fecha = fecha;
59.  }
60.
61.
62.  public Date getFecha()
63.  {
64.    return fecha;
65.  }
66.}
```

Control de la navegación en JSP. Reenvío de peticiones

Normalmente cuando un usuario cursa una petición a un componente de una aplicación web, puede ocurrir que dicho componente no realice todo el procesamiento necesario para esa petición, y que sea necesario que otros componentes internos lo completen. Es más, lo habitual es que la parte de procesamiento interno

se separe de la parte correspondiente a la presentación de los datos al usuario en componentes separados. Esto obliga a que se precise un mecanismo de enlace entre componentes, de manera que al finalizar la ejecución de uno se prosiga el procesamiento en otro. Este mecanismo de enlace de procesamiento entre componentes recibe el nombre de navegación, y la tecnología JSP ofrece una implementación específica para facilitarlo.

JSP ofrece la etiqueta de acción *<jsp:forward>* que permite realizar una petición a un componente JSP o servlet desde otro.

Su sintaxis es la siguiente:

```
<jsp:forward page="<página>" />
```

Donde *<página>* hace referencia al componente sobre el que se desea cursar la petición. Este mecanismo también recibe el nombre de *reenvío* de petición. El componente al que se hace referencia lo es desde el contexto de la página que envía la petición o desde el contexto raíz de la aplicación que contiene al componente que la envía.

Puede emplearse una expresión como nombre de página para que sea evaluada en tiempo de ejecución. Por ejemplo, la siguiente etiqueta de acción es completamente válida:

```
<jsp:forward page="<%=pag%>"/>
```

En este caso, se reenvía la petición a la página que tiene por valor la variable *pag*.

Es importante tener en cuenta, a diferencia del uso de *<jsp:include>*, que este mecanismo no supone inclusión alguna de código de una página dentro de otra. La página que efectúa la petición, una vez llevada a cabo, finaliza dando lugar al inicio de la ejecución de la página que recibe la petición. Es ésta quien estará en condiciones de responder al usuario (se propaga el objeto flujo de salida) ya que la peticionaria, al finalizar, ya no puede continuar con el procesamiento. Advertir que el objeto *request* se propaga de una a otra, de manera que los parámetros existentes en *request* también se reciben en la página destinataria. Este proceso se realiza en el lado del servidor en su totalidad.

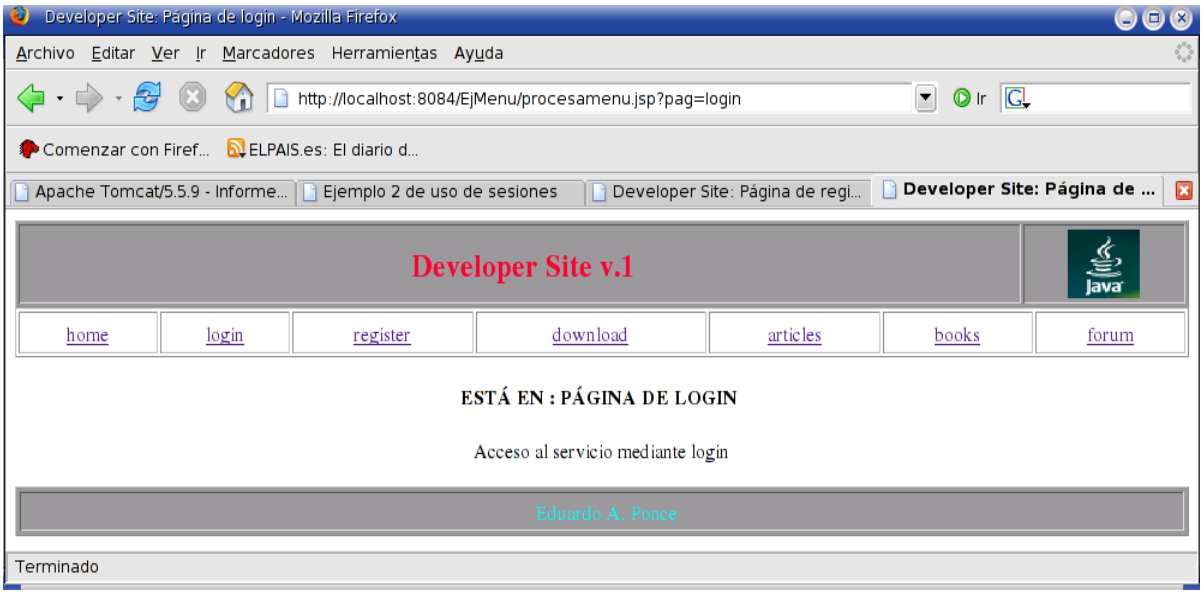
El reenvío de peticiones admite el uso de parámetros. He aquí un ejemplo:

```
<jsp:forward page="WEB-INF/procesapeticion.jsp">
  <jsp:param name="pagina" value="<%=pag%>" />
</jsp:forward>
```

Bajo la etiqueta de acción *<jsp:forward>* se esconde un código Java basado en la funcionalidad que ofrece la clase *RequestDispatcher*. Para ello se emplea el método *forward()* de la clase *PageContext*, método que es abstracto y que cada contenedor web debe implementar debidamente. Este método recibe una cadena, que referencia al recurso sobre el que debe enviarse la petición, y mediante un objeto *RequestDispatcher*, se hace efectiva.

Hay varias maneras de llevar a cabo el reenvío de peticiones mediante la clase *RequestDispatcher*, sin embargo, esto se verá cuando se afronte el diseño de servlets, ya que con JSP este proceso queda subyacente bajo la etiqueta de acción *<jsp:forward>*.

Caso práctico
Se desea diseñar una aplicación web que ofrezca un servicio de acceso a recursos orientados a programadores. Estos recursos, y toda la navegación a través de la aplicación, se efectuará mediante una barra de menú que se ofrecerá en todas las páginas. La página de inicio mostrará el siguiente aspecto:



Cada vez que el usuario haga *clic* sobre una opción de menú, se le enviará a la página asociada a la función representada por la opción. La cabecera y el pie se mantendrán idénticos en toda la aplicación.

Solución

Crearemos el proyecto web *EjMenu*. Hay al menos dos formas básicas de plantear la solución. Una pasa por no hacer uso de reenvío de peticiones (es poco recomendable, sin embargo, es posible) y otra recurrir a la etiqueta de acción *<jsp:forward>*. Nos centraremos en esta última. He aquí el código correspondiente a una primera versión:

home.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.       <title>Developer Site: Aplicación de ejemplo de barra de menú con JSP y reenvío de
peticiones</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ EN : PÁGINA DE INICIO</STRONG> 
13.    </P>
14.    <P>
15.      <center>Bienvenido a Developer Site</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

menu.jsp

```
1.   <jsp:include page="/cabecera.html" flush="true" />
2.     <table cellspacing="2" cellpadding="3" border="1" width="100%" align="center">
3.       <tr>
4.         <td>
5.           <DIV align="center">
6.             <a href="procesamenu.jsp?pag=home">home</a>
7.           </DIV>
8.         </td>
9.         <td>
10.          <DIV align="center">
```

```
11.            <a href="procesamenu.jsp?pag=login">login</a>
12.          </DIV>
13.        </td>
14.        <td>
15.          <DIV align="center">
16.            <a href="procesamenu.jsp?pag=register">register</a>
17.          </DIV>
18.        </td>
19.        <td>
20.          <DIV align="center">
21.            <a href="procesamenu.jsp?pag=downloads">download</a>
22.          </DIV>
23.        </td>
24.        <td>
25.          <DIV align="center">
26.            <a href="procesamenu.jsp?pag=articles">articles</a>
27.          </DIV>
28.        </td>
29.        <td>
30.          <DIV align="center">
31.            <a href="procesamenu.jsp?pag=books">books</a>
32.          </DIV>
33.        </td>
34.        <td>
35.          <DIV align="center">
36.            <a href="procesamenu.jsp?pag=forum">forum</a>
37.          </DIV>
38.        </td>
39.      </tr>
40.    </table>
```

cabecera.html

```
1.   <table cellspacing="2" cellpadding="3" border="1" width="100%" align="center"
bgcolor="#999999">
2.       <tr>
3.         <td width="86%" align="center">
4.           <P> 
5.             <FONT color="#ff0033" size="5"><STRONG>Developer Site v.1</STRONG></FONT>
6.           </P>
7.         </td>
8.         <td width="14%" align="center">
9.           <img src="grafico.jpg" width="60" height="51"/>
10.        </td>
11.      </tr>
12.    </table>
```

pie.html

```
1.          <table   cellspacing="3"   cellpadding="2"   border="1"   width="100%"   align="center"
bgcolor="#999999">
2.       <tr>
3.         <td align="center"> 
4.         <FONT color="#00ffff">Eduardo A. Ponce</FONT></td>
5.       </tr>
6.     </table>
```

procesamenu.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Procesa opción de menú</title>
6.   </head>
7.   <body>
8.   <%
9.     String pag = request.getParameter("pag");
10.    pag = pag+".jsp";
11.  %>
12.  <jsp:forward page="<%=pag%>"/>
```

```
13.   </body>
14.</html>
```

articles.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Developer Site: Página de articles</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ EN : PÁGINA DE ARTICLES</STRONG> 
13.    </P>
14.    <P>
15.      <center>Acceso a artículos</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

books.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Developer Site: Página de books</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ EN : PÁGINA DE BOOKS</STRONG> 
13.    </P>
14.    <P>
15.      <center>Acceso a libros</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

downloads.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Developer Site: Página de download</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ EN : PÁGINA DE DOWNLOAD</STRONG> 
13.    </P>
14.    <P>
15.      <center>Proceso de downloads</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

forum.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Developer Site: Página de forum</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ EN : PÁGINA DE FORUM</STRONG> 
13.    </P>
14.    <P>
15.      <center>Acceso al foro</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

login.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Developer Site: Página de login</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ EN : PÁGINA DE LOGIN</STRONG> 
13.    </P>
14.    <P>
15.      <center>Acceso al servicio mediante login</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

register.jsp

```
1. <%@ page contentType="text/html;charset=ISO-8859-15"%>
2. <html>
3.   <head>
4.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
5.     <title>Developer Site: Página de registro</title>
6.   </head>
7.   <body>
8.     <P>
9.       <jsp:include page="menu.jsp" flush="true"/>
10.    </P>
11.    <P align="center">
12.      <STRONG>ESTÁ; EN : PÁGINA DE REGISTRO</STRONG> 
13.    </P>
14.    <P>
15.      <center>Proceso de registro</center>
16.    </P>
17.    <jsp:include page="pie.html" flush="true" />
18.  </body>
19.</html>
```

Puede observarse que todos los archivos están ubicados en el directorio raíz de la aplicación. La página de inicio es *home.jsp* que muestra tanto el menú (que incluye en su código la propia cabecera) como el pie. El usuario, accederá a cada opción del menú haciendo *clic* sobre ella. Se comprueba que el sistema de menú ha sido implementado mediante código HTML, mediante enlaces que establecen una petición que incorpora

un parámetro identificativo de la acción asociada, de forma que todas las peticiones se envían sobre la misma página, *procesapeticion.jsp*, determinando ésta última, en base al valor del parámetro *pag*, la página que debe mostrarse. En este mecanismo el procesamiento de una opción de menú lleva aparejada la ejecución de dos páginas JSP. La primera se encarga de recibir la petición del usuario, y determinar qué página debe encargarse del procesamiento, y la segunda es precisamente la página escogida para ello. Podemos establecer que la página *procesamenu.jsp* es la encargada de controlar la navegación en la aplicación, mientras que las páginas llamadas por ella, se especializan en ejecutar la funcionalidad asignada para cada caso. La página *procesamenu.jsp* es muy simple, y en el caso que nos ocupa incluso podríamos haber prescindido de ella, sin embargo, en una aplicación real, es habitual que esta página se encargue de realizar uno o varios procesos preliminares que serán comunes a todas las páginas encargadas de procesar cada opción de menú, y que por optimización del diseño, se recomienda no transferirlos a ellas.

Aunque la página *procesamenu.jsp* toma como nombre base de la página JSP a la que va a reenviar la petición el valor del parámetro asociado a la opción del menú escogida por el usuario, puede establecer un mecanismo de independencia entre dicho valor y el nombre real de la página. He aquí una posible forma de llevar a cabo esto en *procesamenu.jsp*:

```
<%@ page contentType="text/html;charset=ISO-8859-15"%>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
  <title>Procesa opción de menú</title>
 </head>
 <body>
 <%
  String pag = request.getParameter("pag");
  String paginareal;
  if (pag.equals(“login”))
       paginareal = “procesalogin.jsp”;
  if (pag.equals(“books”))
       paginareal = “procesabooks.jsp”;
  ...
 %>
 <jsp:forward page="<%=paginareal%>"/>
 </body>
</html>
```

Otra manera más adecuada de afrontar esta aplicación web (y en general, todas ellas), es ubicando todas las páginas que no son accesibles directamente por el usuario, en el directorio *WEB-INF* de la aplicación, ya que todos los componentes o recursos ubicados en él están ocultos para los usuarios, ya que éstos no pueden hacer una petición directa sobre ellos[38]. Normalmente todas las páginas encargadas exclusivamente de realizar procesamiento, sin mostrar datos al usuario, pueden ubicarse en dicho directorio.

Parámetros de inicialización. Nivel servlet. Nivel aplicación.

Son parámetros accesibles por las páginas JSP, declarados e inicializados en el archivo descriptor de despliegue del módulo web (*web.xml*). Sus valores pueden ser modificados en estos archivos sin necesidad de recompilación, aunque sí se requiere redespliegue.

Suelen emplearse para almacenar información necesaria para inicializar datos de la aplicación o del servlet, así como para establecer información de configuración (por ejemplo, relativa a los servidores de bases de datos con los que se conectará la aplicación).

Si se va a definir un parámetro de inicio a *nivel de aplicación*, se empleará, en el archivo *web.xml*, la etiqueta <*context-param*> para definirlo y darle un valor. Estos valores son siempre tomados como cadenas de texto. He aquí un ejemplo:

```
<context-param>
  <param-name>nombreAp</param-name>
  <param-value>Ejemplo enésimo</param-value>
```

38 El servidor de aplicaciones indicará que el recurso no existe o no lo localiza.

</context-param>

En este caso de establece como nombre del parámetro *nombreAp* , mediante la etiqueta *<param-name>*, y como valor, *Ejemplo enésimo*, con la etiqueta *<param-value>*. Este parámetro estará disponible para todos los componentes de la aplicación.

Si el parámetro se especifica a *nivel de un determinado servlet*, entonces se recurre a la etiqueta *<init-param>*, dentro del contexto de configuración del servlet en *web.xml*. Hasta ahora no hemos entrado en detalle en este aspecto, limitándonos siempre a hacer uso de los objetos que implícitamente nos proporcionaba el contenedor web del servidor de aplicaciones, sin embargo, hay ciertos aspectos de la configuración de las aplicaciones que pueden reflejarse en el archivo *web.xml* y que atañe a los servlets, tanto para los asociados a las páginas JSP (que son generados por el contenedor web) como a los escritos por el programador. Para analizar este aspecto se ofrece el siguiente código tomado de un caso concreto:

```
<servlet>
   <servlet-name>multiplebienvenida</servlet-name>
   <jsp-file>multiplebienvenida.jsp</jsp-file>
   <init-param>
     <param-name>veces</param-name>
     <param-value>10</param-value>
   </init-param>
   <init-param>
     <param-name>mensaje</param-name>
     <param-value>Holaaaaaaaa</param-value>
   </init-param>
</servlet>
<servlet-mapping>
   <servlet-name>multiplebienvenida</servlet-name>
   <url-pattern>/bienvenida</url-pattern>
</servlet-mapping>
```

Se observa que la etiqueta para establecer la información relativa a la configuración de un servlet es *<servlet>*. Dentro de esta etiqueta se incluyen otras, que permiten, entre otras cosas, establecer los parámetros de inicio del servlet. Por otra parte, normalmente, cuando en una aplicación web se envían peticiones a componentes que son servlets o páginas JSP, se suele hacer empleando una referencia lógica del componente, en lugar de hacer mención expresa del archivo físico involucrado. Por ejemplo, si una aplicación puede recibir una petición a una página JSP denominada *proceso.jsp*, puede emplearse la URL *http://<servidor>:<puerto>/<contextoRaíz>/proceso.jsp* o bien, *enmascarar* el nombre del archivo *proceso.jsp* por el de *ejecutar*, por ejemplo. Entonces la URL quedaría como *http://<servidor>:<puerto>/<contextoRaíz>/servlet/ejecutar*[39], de manera que no se sabe el nombre físico del archivo. Esto posibilita ocultar al usuario el nombre de los archivos que componen la aplicación, pero además tiene otras ventajas. Una de ellas es que de esta manera pueden emplearse identificadores para las páginas y servlets más comprensibles dentro del contexto funcional de la aplicación, y por otra, es que esta asociación entre nombre lógico y nombre físico de archivo se establece a nivel de archivo descriptor, de forma que puede variarse los nombres de los archivos, sin necesidad de modificar los lógicos, y por tanto, el código de las páginas que hacen referencia a ellos.

Para conseguir esta separación entre nombres lógicos y físicos se emplean las etiquetas *<servlet-name>* y *<jsp-file>* si la asociación es con una página JSP[40], y *<servlet-name>* y *<servlet-class>* si es con una clase de tipo servlet diseñada por el programador. En el caso que nos ocupa, se establece como nombre de servlet *multiplebienvenida* y se asocia con la página JSP *multiplebienvenida.jsp*. Desde este momento, se

39 Es preciso anteceder *servlet* al nombre lógico, ya que la llamada a un servlet, salvo que se trate de un mapeo (que se abordará más adelante) se hace siempre indicando al servidor de aplicaciones en la URL que el componente es un servlet. Esta sintaxis es siempre la misma en todas las aplicaciones J2EE, y se hace siempre poniendo *servlet* tras el nombre de la aplicación, como si se hiciera referencia a un directorio.

40 Obsérvese que se hace mención a la página JSP y no al nombre del servlet que genera a partir de ella el contenedor WEB.

puede hacer una petición con la URL *http://<servidor>:<puerto>/<contextoRaíz>/multiplebienvenida*. El servidor, al recibir la petición entenderá que se desea ejecutar el servlet generado a partir de la página *multiplebienvenida.jsp*. Este mecanismo es además el que posibilita que puedan establecerse parámetros de inicio a nivel de servlet.

En el ejemplo que se trata, al servlet *multiplebienvenida* se le configuran dos parámetros de inicio, *veces* con valor 10 y *mensaje* con valor *Holaaaaaaaa*. Para cada uno de ellos, se emplean las etiquetas <*param-name*> y <*param-value*> dentro de <*init-param*>.

Finalmente, dado un servlet, pueden establecerse para él otras posibles formas de llamarlo dentro de una petición, a este mecanismo se le denomina *mapeado*. Así, al servlet *multiplebienvenida* se le asigna la cadena */bienvenida* para que pueda realizarse la petición a él con la URL *http://<servidor>:<puerto>/<contextoRaíz>/bienvenida*. La barra "/" que precede a */bienvenida* se emplea para denotar que la cadena se emplea precedida inmediatamente por el contexto raíz de la aplicación.