J2EE: JSP
Java Server Pages

Antonio Espín Herranz

Contenidos

- Introducción.
- Elementos y directivas.
- Sintaxis.
- Objetos implícitos.
- JavaBeans.
- Generar XML con JSPs.

JSPs

- Permiten separar la parte dinámica de tus páginas de la parte HTML.
- El código HTML se escribe tal cual y el código java se encierra entre los símbolos **<% … %>**
- Una JSP cuando se la llama por primera vez se convierte en un Servlet y se compila.

JSPs vs ASPs

- JSP y ASP ofrecen funciones similares. Ambos utilizan etiquetas para permitir código embebido en una página HTML, seguimiento de sesión, y conexión a bases de datos.

- Algunas de las diferencias son:
  - Las páginas ASP están escritas en VBScript y las páginas JSP están escritas en lenguaje Java. Por lo tanto, las páginas JSP son independientes de la plataforma y las páginas ASP no lo son.

  - Las páginas JSP usan tecnología **JavaBeans** como arquitectura de componentes y las páginas ASP usan componentes **ActiveX**.

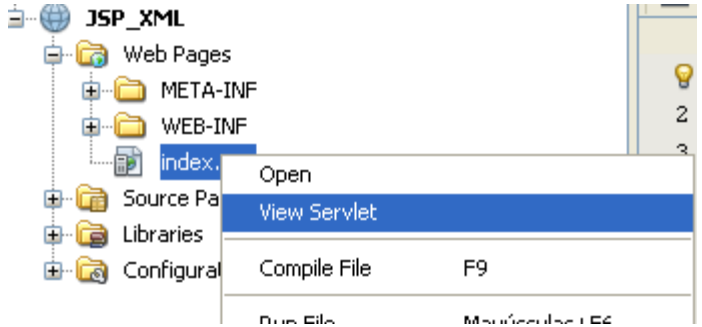
JSPs vs ASPs

- Mas diferencias:
  - **Velocidad y Escalabilidad**: Aunque las páginas ASP son cacheadas, siempre son interpretadas, las páginas JSP son compiladas en Servlets Java y cargadas en memoria la primera vez que se las llama, y son ejecutadas para todas las llamadas siguientes. Esto le da a las páginas JSP la ventaja de la velocidad y escalabilidad sobre las páginas ASP.

  - Las páginas ASP sólo trabajan con Microsoft IIS y Personal Web Server. El uso de páginas ASP requiere un compromiso con los productos de Microsoft, mientras que las páginas JSP no nos imponen ningún servidor web ni sistema operativo. Las páginas JSP se están convirtiendo en un estándar ampliamente soportado.

JSPs

- Cuando la página se convierte en un Servlet, este implementa el método service.
- Por lo tanto la página JSP puede aceptar tanto peticiones GET como POST sin indicar nada.
- Desde Netbeans podemos ver el servlet generado:



JSPs

- A una JSP se la puede llamar desde un enlace o desde el action de un form.

```
<a href="miPagina.JSP">Llamada</a>
<form name="datos" action="miPagina.JSP">
    …
</form>
```

- También se le puede cargar directamente utilizando el objeto RequestDispatcher.

Sintaxis de una JSP

- Hay 3 tipos de construcciones dentro de una JSP (a parte del HTML):
  - **Elementos de Script**: Permiten especificar código java que formará parte del servlet resultante.
  - **Directivas**: Permiten controlar la estructura completa del servlet.
  - **Acciones**: Permiten especificar componentes existentes que serán usados y que controlarán el comportamiento de la página JSP.

Elementos de Script

- **<%=expresión%>** Se evalúa la expresión y se insertan en la salida.
  - Ejemplo: El resultado es:<%=suma%>
  - Nos sirve para imprimir el contenido de variables java.

- **<% código java %>** El código java será insertado en el método service del servlet. Desde un Script podemos acceder a las variables del sistema: request, response, etc. Se pueden mezclar con el código HTML.

- **<%! Código java %>** En este caso los métodos o campos serán insertados en el campo principal de la clase del servlet.

- **<% -- Comentario --%>** Indicamos un comentario dentro de una JSP.

Elementos de Script

- Dentro de las **expresiones** también podemos utilizar:
- La fecha y la hora es: <%=new Date()%>

- Dentro de estas expresiones podemos utilizar una serie de objetos implícitos que se detallan mas adelante:
- La queryString es: <%=request.getQueryString()%>

Elementos de Script

- Dentro de los bloques de código …
- **<% Código java %>**
- Combinamos código HTML y Java:

```
<table border='1'>
<% for (int i = 0 ; i < 10 ; i++){ %>
    <tr>
    <% for (int j = 0 ; j < 10 ; j++){ %>
        <td><%=relleno%></td>
    <% } %>
    </tr>
<% } %>
</table>
```

Elementos de Script

- **<%! Código Java %>**
- En este caso lo que declaremos dentro de este bloque de código se incrustará dentro de Servlet generado.
- **<%!** private String nombre; **%>**

- Con esta declaración estamos definiendo un atributo en el Servlet.

Directivas en JSPs

En la 1ª línea de la JSP se detalla la directiva **page**.

<%@ page language="java" **errorPage**="url pagina de error"
import="java.sql.*, java.io.*" **%>**

<%@ page contentType="text/html" **%>**

<%@ page session="true" **%>**

Los atributos de las directivas son sensibles a Mayúsculas y Minúsculas.

- Se pueden colocar **varias directivas page**.
- Se detalla el lenguaje, si es necesario se especifica una página de error.
- **Se detallan los paquetes de java** que vamos a usar en la JSP.

Directivas en JSPs

- Algunos de los atributos mas utilizados:
- **import**:
  - pueden aparecer mas de una vez.
- **session**:
  - Por defecto es true, indica que si existe una sesión la página formará parte de esa sesión y no existe se creará una nueva.
- **isErrorPage**:
  - indica que la página es una página de error.

Directivas en JSPs

- **autoflush**=“true | false”:
  - Por defecto es true indica si el buffer debe descargarse cuando esté lleno.
  - Si no se descargara se lanzaría una excepción.
  - Lo normal es dejarlo a true.

- **info**=“message”:
  - Se puede definir un mensaje de texto que se podría recuperar mediante el método getServletInfo.

- **errorPage**=url
  - Se especificaría una página de error, en caso de que hubiera un error, saltaría dicha página.

Directivas en JSPs

- Con la directiva **include** podemos incluir otras páginas.
- **<%@include** file="url relativa"**%>**
- Ejemplo:
  - <%@include file="grabar.jsp%>

- Incluye la página en el momento que la JSP se traduce a un Servlet.

Acciones en JSP

- jsp:include → Permite insertar ficheros en la página.
  - Ejemplo:
    - <jsp:include page="pagina.html" flush="true"/>
    - Inserta la página después de procesar el contenido de esta.

Objetos implícitos

- Como la JSP se compila y se convierte en un Servlet disponemos de una serie de objetos predefinidos (que no necesitamos declararlos dentro del JSP, simplemente los utilizamos).
  - request
  - response
  - out
  - session
  - application
  - config
  - pageContext
  - page

request

- Se corresponde con el objeto HttpServletRequest asociado a la petición. Te permite obtener los parámetros de la petición, mediante getParameter).

- Ejemplo: Podemos recoger los campos de un formulario.

  <%String nombre=request.getParameter(“nombre”); %>

- Para recuperar los parámetros:

  request.getQueryString();

  Los parámetros se envían de la forma:
  nombrePag.jsp**?param1=valor1&param2=valor2**

response

- Se corresponde con el objeto: HttpServletResponse asociado a la respuesta al cliente que realizó la petición.
- Para escribir la respuesta al cliente usaremos el objeto out → es el flujo de salida.

  PrintWriter getWriter() → recupera un Writer.
- Este objeto lo usaremos para la generación de cookies.
- Para redireccionar a otra página →

  response.sendRedirect(“url”);

out

- Se corresponde con el objeto PrintWriter usado para enviar la salida al cliente.

- Ejemplo:

  <% out.println(“<p>El valor de nombre es:” + nombre + “</p>”); %>

- Aunque esto es más cómodo hacerlo así:
- ** El valor de nombre es <%=nombre %> **
- Queda mas claro y se asemeja mas al HTML.

session

- Representa la sesión del Usuario en un JSP/Servlet corresponde al objeto HttpSession.

- Con esto objeto podemos almacenar objetos o variables que sean accesibles desde cualquier JSP de la session.

- Métodos para recuperar y establece variables:
  - Object session.getAttribute(“clave”);
  - void session.setAttribute(“clave”,Object objeto);
  - void session.invalidate(); → Invalida la session.
  - Para activar la session, reflejar el atributo:
    <%page session=“true” %>

application

- Este es el objeto ServletContext que se obtiene mediante getServletConfig().getContext(). Es común para todas las aplicaciones web.

- Métodos para recuperar y establece variables:
  - Object application.getAttribute(“clave”);
  - void application.setAttribute(“clave”,Object objeto);

config

- Se refiere al objeto ServletConfig del Servlet.

- Desde esta clase podíamos obtener el contexto del servlet, los parámetros de inicialización, el nombre del Servlet.

pageContext

- Este objeto permite acceder a características del Servidor.
- Objetos implícitos …

- Métodos:
  - forward(String url)
  - getRequest(), getResponse(), getSession().
  - getServletContext(), getServletConfig().

page

- El objeto page hace referencia a la propia página.
- Es un sinónimo de this.
- No se suele utilizar mucho.

Redireccionamiento

Registrar.jsp

```
<% String login, pass, url;
      // CAPTURA LOS PARAMETROS DE UN FORM EN UNA PAGINA HTML.
      login=request.getParameter("NOMBRE");
      pass=request.getParameter("PASS");

      // Recogemos los parametros y los mandamos a otra página:
      url = "redireccionar/registrar2.jsp?login=" + login + "&pass=" + pass;
      response.sendRedirect(url);
%>
```

Registrar2.jsp

```
<% String login, pass;

                out.println("<p>Hemos recibido:</p>");
                out.println("<p>" + request.getQueryString() + "</p>");
%>
```

PRÁCTICAS: JSPs

JavaBeans

- Un javaBean dentro de una página JSP mantiene toda la lógica de nuestra aplicación.

- Puede interactuar con la BD para grabar / recuperar información y oculta todo el código de nuestra página.

JavaBeans

- Hay que tener las **convenciones java** de los JavaBeans.

- Son clases Java que es obligatorio tener un **constructor por defecto** (sin parámetros) y métodos **set / get** para poder acceder a las propiedades.

Uso de JavaBeans en JSPs

- jsp:useBean → Permite cargar un JavaBean en la página para ser usado en la página JSP.
  - Ejemplo:
    - <jsp:useBean id="**name**" scope="session" class="package.class"/>
    - Instancia la clase del Bean y le asigna como id name.

- jsp:setProperty → Establecer un valor a una propiedad del Bean.
  - Ejemplo:
    - <jsp:setProperty name="**name**" property="nomb_prop" value="valor"/>
    - <jsp:setProperty name="**name**" property="nombr_prop" **param="nombre_control_form**"/>

    - *Con el 2º ejemplo podemos cargar el Bean directamente con los campos de un FORM.*

Uso de JavaBean en JSPs

- jsp:getProperty → Recuperar una propiedad del Bean.
  - Ejemplo:
    - <jsp:getProperty name="**name**" property="nomb_prop/>
- jsp:forward → Permite redirigir la petición a otra página.
  - Ejemplo:
    - <jsp:forward page="/dir/ …/error.jsp"/>

Ejemplo de JavaBean

SimpleBean.java

```
package  hall;
public class SimpleBean {
    private String message=“hola”;
    public String getMessage(){
        return(message);
    }
    public void setMessage(String msg){
        this.message=msg;
    }
}
```

Ejemplo de JavaBean

BeanTest.jsp

```
<%@ page language="java" import="hall.*" %>
<jsp:useBean id="test" scope="session" class="hall.SimpleBean" />

<html>
<body>

    <jsp:setProperty name="test" property="message" value="Hola que tal" />
    <h1>mensaje:<I><jsp:getProperty name="test" property="message" /></I></h1>

</body>
</html>

<!- - Si queremos aplicar algún método al bean dentro de la página, será: test.metodo()
     también dentro del Bean tendremos un constructor por defecto:
    public SimpleBean(){ } -- >
```

Prácticas: JAVA BEANS

Generar XML con un JSP

- Indicar dentro del content Type de la página el tipo MIME a devolver: **text/xml**.

- En la cabecera del JSP:

  <%@ page contentType="**text/xml**" %>

- Después se intercalan etiquetas XML con el código Java.

Generar XML con un JSP

- **Ejemplo**:

```
<%@ page contentType="text/xml" %>
<?xml version="1.0" encoding="UTF-8"?>
<personas>
<% for (Persona persona : personas ){ %>
 <persona>
    <nombre><%=persona.getNombre()%></nombre>
    <apellidos><%=persona.getApellidos()%></apellidos>
 </persona>
 <% } %>
 </personas>
```