Introducción a JSP

Que es Java Server Page (JSP)?

Es una interfaz de programación de aplicaciones de servidores Web. En una página jsp se entremezclan bloques de HTML estáticos, y HTML dinámico generados con Java que se ejecutan en el servidor. Una página jsp puede procesar formularios Web, acceder a bases de datos y redireccionar a otras páginas. Las páginas jsp son transformadas a un servlet y después compiladas.

El contenedor JSP proporciona un motor que interpreta y procesa una página JSP como un servlet (en tomcat, dicho motor se llama jasper). Al estar basadas en los servlets, las distintas peticiones a una misma página jsp son atendidas por una única instancia del servlet.

Diferencias entre JSP y Servlet

En JSP, el código de presentación está separado de la lógica del programa, mientras que en un servlet, el código de presentación se compila dentro de la clase.

En una página JSP el código de presentación puede ser actualizado por un diseñador web que no conozca Java.

servlet = clase java (compilada)

Los servlets se encuentran ya compilados, mientras que las páginas JSP se compilan bajo petición, lo que hace que la ejecución del servlet sea algo más rápida (en la primera petición)

Características de JSP

Permiten separar la parte dinámica de la estática en una página web

Las páginas jsp se almacenan en el servidor en archivos con extensión .jsp.

El código JSP es java y se encierra entre: <% y %>, por ejemplo:

<H1>Hora: <%= new java.util.Date() %></H1>

La sintaxis también se puede expresar en formato XML

<jsp:xxx> ... </jsp:xxx>

En una página jsp hay varios objetos implícitos (predefinidos):

request, response, out, session, application, config, pageContext, page y exception

Cada página JSP es compilada automáticamente hacia un servlet por el motor JSP la primera vez que se accede a esa página.

Desde una página JSP se puede llamar a un componente JavaBean donde se puede implementar la lógica del negocio

Funcionamiento de JSP

Una pagina JSP básicamente una página Web con HTML tradicional y código Java incrustado. La extensión de fichero de una página JSP es ".jsp" en vez de ".html" o ".htm", y eso le indica al servidor que esta página requiere un tratamiento especial que se conseguirá con una extensión del servidor o un plug-in.

El servidor web comprueba si la página ha sido solicitada con anterioridad. En ese caso el servlet correspondiente ya estará cargado en memoria. Si no es así, se notifica al motor de jsp y se generará un servlet para la página.

Cuando un cliente solicita una página jsp, se ejecuta en el servidor el código JSP de la página, dando como resultado una página HTML que se fusiona con el HTML original, generando una página HTML de respuesta que será enviada al cliente

Programando con JSP

Tipos de construcciones JSP

Elementos de script:
Permiten especificar código Java
Encerrados entre <%= y %>, <% y %>, <%! y %>

Directivas

Permiten controlar la estructura general del servlet
Encerradas entre <%@ y %>

Comentarios

Permiten insertar comentarios
Encerrados entre <%-- y --%>

Acciones

Permiten especificar los componentes JavaBean que va a utilizar la JSP
Encerradas entre <jsp:xxx> y </jsp:xxx>

Elementos de scrip

Expresiones

Las expresiones permiten insertar valores Java en la salida. La sintaxis es:
<%= expresión Java %> <jsp:expr> ... </jsp:expr>
Por ejemplo:
hora:<%= new java.util.Date() %>
Las expresiones tienen acceso a los objetos implícitos. Por ejemplo:
<%= "Método envío formulario: " + request.getMethod() %>
Ejemplo con XML:
<jsp:expr> "Método envío formulario: " + request.getMethod() </jsp:expr>

Scriptlets:

Permiten insertar código Java directamente en la página JSP.
<% codigo Java; %> <jsp:scriptlet> ... </jsp:scriptlet>
Por ejemplo:
<% i = 7; %>
Los scriplets tienen acceso a los objetos implícitos. Por ejemplo:
<% String str = “hola”; out.println(str); %>
Ejemplo con XML:
<jsp:scriptlet> String str = “hola”; out.println(str); </jsp:scriptlet>

Declaraciones:

Permiten definir las variables o métodos que serán utilizados en la página. No tienen acceso a los objetos implícitos. La sintaxis es:

<%! declaración Java; %> <jsp:decl> ... </jsp:decl>

Por ejemplo:

<%! private int i = 5; %> <%= i %>

Ejemplo con XML:

<jsp:decl> int i = 5; </jsp:decl> <jsp:expr> “i: " + i </jsp:expr>

Directivas

Directiva page:

Permite definir uno o más atributos.

<%@ page attributo=“valor” %> <jsp:directive.page attributo=“valor”/>

Los atributos serían

import="paquete.clase"
contentType="MIME-Type"
isThreadSafe="true|false"
session="true|false"
buffer="sizekb|none"
autoFlush="true|false"
extends="package.class"
info="message"
errorPage="url"
isErrorPage="true|false"
nlanguage="java”

Por ejemplo:

<%@ page import=“java.util.*” %>

Ejemplo con XML:

<jsp:directive.page import=“java.util.*”/>

Directiva include:

Permite incluir ficheros en la página JSP. Inserta el fichero en tiempo de compilación. La sintaxis es:

Para no tener redundancia de código.

<%@ include file=“url” %> <jsp:directive.include file=“url”/>

Por ejemplo:

<%@ include file=“pagina.html” %>

Ejemplo con XML:

<jsp:directive.include file=“pagina.html”/>

La primera página JSP

Requisitos para probar JSP

- Cualquier servidor web que soporte Servlets (Ejemplo Apache Tomcat)
- JVM
- IDE de programación Java

Objetos Implícitos

Los objetos implícitos también son llamados variables predefinidas y se utilizan directamente, sin crearlos. A continuación se muestra una tabla con los objetos implícitos que pueden ser utilizados desde las páginas JSP

Objeto	Descripción	Tipo	Ámbito
application	Representa el contexto en el que se ejecutan las JSP	javax.servlet.ServletContext	application
session	Sesión de un cliente	javax.servlet.http.HttpSession	session
request	Extrae datos del cliente	javax.servlet.http.HttpServletRequest	request
response	Envía datos al cliente	javax.servlet.http.HttpServletResponse	page
out	Envía la salida al cliente	javax.servlet.jsp.JspWriter	page
exception	Información último error	java.lang.Throwable	page
config	Representa la configuración del Servlet	javax.servlet.ServletConfig	page
pageContext	Es de donde se obtienen algunos objetos implícitos	javax.servlet.jsp.PageContext	page
page	La propia página JSP	java.lang.Object	page

La mayoría de los objetos implícitos provienen de la clase abstracta javax.servlet.jsp.PageContext. Cuando se compila la JSP, el servlet que se obtiene, contiene la creación de estos objetos implícitos dentro del método _jspService:

```
public void _jspService ( HttpServletRequest request, HttpServletResponse response )
throws IOException, ServletException {
        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        ...
        pageContext = _jspxFactory.getPageContext(this, request, response, "", true, 8192, true);
```

```
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
        ...
}
```

Ejemplos de páginas JSP sencillas

En estos ejemplos se mostrará la utilización de las distintas construcciones JSP vistas anteriormente. Para la realización de las pruebas crearemos un nuevo proyecto Web Dinámico llamado jsp, en el cuál crearemos los estos ejemplos

Ejemplo de Expresiones (Expresiones.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Expresiones JSP</title>
 </head>
 <body>
<H1>Ejemplo de expresiones JSP</H1>  <UL>  <LI>Fecha actual: <%= new java.util.Date()%> <LI>
Nombre del host: <%=request.getRemoteHost()%>  <LI>ID de la sesión: <%=session.getId()%>  <LI>
El parámetro es: <%=request.getParameter("nombre")%>  </UL>
 </body>
 </html>
```

Ejemplo de Scriplets (Color.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Scriptlets JSP</title>
<% String bgColor=request.getParameter("bgColor");
boolean hayColor;
if (bgColor!= null ) hayColor= true ;
else { hayColor=false; bgColor="WHITE"; } %>
</head>
<body BGCOLOR="<%=bgColor%>">
<H1>Ejemplo de scriptlets JSP</H1>
<%
if
```

```
 (hayColor) out.println("Se ha utilizado el color: " + bgColor);
else
 out.println("Se ha utilizado el color por defecto: WHITE"); %>
</body>
</html>
```

Ejemplo de declaraciones y directiva page (Contador.jsp)

```
<%@page import="java.util.*"%>
<%-- Esto en un comentario de JSP --%>
<%!
private int cont=0;
private Date fecha= newDate();
private String host="<I>Sin acceso previo</I>";
%>
<p>Esta página ha sido accedida <b><%= ++cont%></b> veces desde que se inició el servidor.</p>
 <p>El último acceso ha sido desde: <b><%=host%></b> con fecha <b><%=fecha%>
</b></p>
 <%
Host = request.getRemoteHost(); fecha=new Date();
%>
```

Ejemplo de directiva include (Empresa.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-
8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Ejemplo de uso de un contador incluido en un JSP</title>
</head>
<body>
<H1>Ejemplo de uso de un contador incluido en un JSP</H1>
<%@include file="Contador.jsp"%>
</body>
</html>
```

Utilización de Páginas de Error

Excepciones en JSP

La gestión de excepciones en JSP se lleva a cabo a través del objeto Exception. Para manejar las excepciones en las JSP se deben seguir los siguientes pasos:

Escribir un Servlet, JSP u otro componente para que lance excepciones en determinadas condiciones. Por ejemplo desde una JSP podría indicarse como:
public Object metodo() throws NullPointerException { ... }

Escribir una JSP que será la página de error usando la directiva page con isErrorPage="true". Es decir que si ocurre un error en otras JSP que usen el componente, se ejecutará esta página:
<%@ page isErrorPage="true" import="java.util.*" %>

En la página de error usar el objeto exception *para obtener información de la excepción*
<%= exception.toString() %> <% exception.printStackTrace(); %>

En las JSP que usan el componente, indicar qué página se ejecutará si se produce algún error, mediante la directiva page, estableciendo errorPage a la página de error.
<%@ page isThreadSafe="false" import="java.util.*" errorPage="error.jsp" %>

Ejemplo de utilización de páginas de error

A continuación mostramos un ejemplo en el que intervienen dos páginas, la página que puede provocar la excepción o lanzar el error (*Division.jsp)* y la página que saldrá como tratamiento de la excepción (*ErrDiv.jsp)* La página *Division.jsp* podría quedar codificada de la siguiente forma

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Manejo de errores de JSP</title>
</head>
<body >  <%@page errorPage="ErrDiv.jsp"%>
<H1>Ejemplo de manejo de errores en JSP</H1>
<%!
private double toDouble(String value) {
return(Double.valueOf(value).doubleValue()); }
%>
<%
double  op1 = toDouble(request.getParameter("op1"));
double op2= toDouble(request.getParameter("op2"));
double res = op1/op2;
%>
<TABLE border=1>
 <TR>
<TH></TH>
<TH>División</TH>
</TR>
 <TR>
<TH>Operando 1: </TH>
<Td><%=op1%></Td>
```

```
</TR>
 <TR>
<TH>Operando 2: </TH>
<Td><%=op2%></Td>
</TR>
 <TR>
<TH>Resultado: </TH>
<Td><%=res%></Td>
</TR>
 </TABLE>
</body>
 </html>
```

La página de error* ErrDiv.jsp *quedaría como

```
<%@page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
 <title>Página de errores en JSP</title>
</head>
<body >
<%@page isErrorPage="true"%>
<H1>Página de errores en JSP</H1>
<P>Division.jsp ha reportado el siguiente error: <b><%=exception%></b>  </P>
<P>El error que ha ocurrido es:
<pre>  <%exception.printStackTrace(new java.io.PrintWriter(out)); %> </pre>
 </P>
</body>
</html>
```