Arquitectura e Integración de Aplicaciones Empresariales

Cuarta Sesión
Java Server Pages

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Dra. Maricela Bravo
Cubículo H-287-B
mari_clau_18@hotmail.com

JSP: Introducción

- Una tecnología que permite combinar código HTML estático con código generado dinámicamente en un mismo archivo.
- Ventajas:
  - Separación de datos estáticos/dinámicos.
  - Independencia de formato/plataforma.
  - Sencillez (cuando se conoce la tecnología de servlets)

JSP: Introducción

- Comparaciones con otras tecnologías:
  - Vs ASP (Active Server Pages) y ASP.NET.
  - Vs Servlets.
  - Vs JavaScripts
- Los JSP nos permiten separar la parte dinámica de nuestras páginas Web del HTML estático. Simplemente escribimos el HTML regular de la forma normal y encerramos el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con "<%" y terminan con "%>".

JSP: Introducción

- La extensión `.jsp`.
- Aunque el código parezca mas bien HTML, el servidor lo traduce a un servlet en la primera petición.
- 3 elementos en un JSP:
  - Elementos script (scriptlets)
  - Directivas
  - Acciones

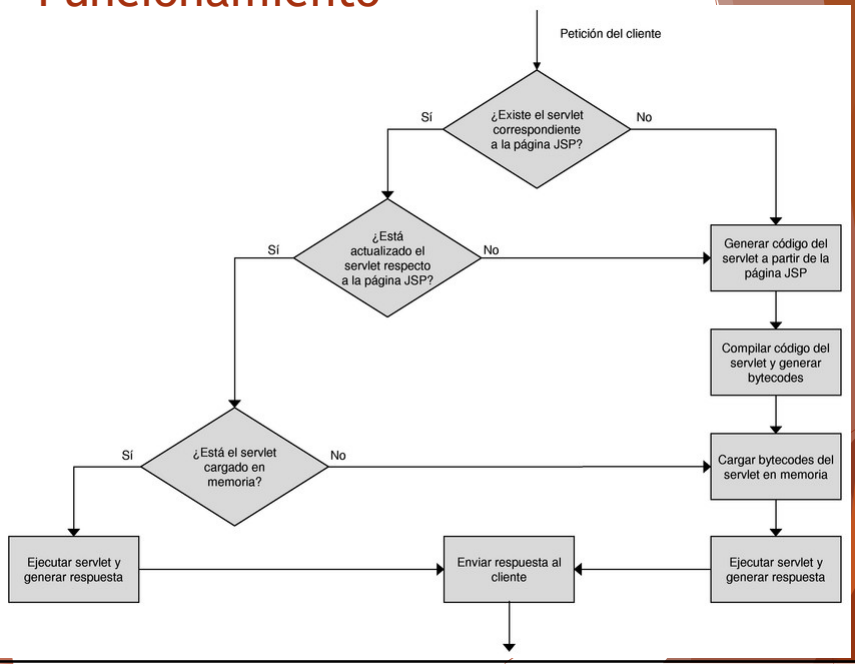
Introducción a JSP

Página JSP → Servlet

- La página JSP se convierte en un servlet
- La conversión la realiza en la máquina servidora el *motor o contenedor JSP*, la primera vez que se solicita la página JSP
- Este servlet generado procesa cualquier petición para esa página JSP
- Si se modifica el código de la página JSP, entonces se regenera y recompila automáticamente el servlet y se recarga la próxima vez que sea solicitada

5

Funcionamiento



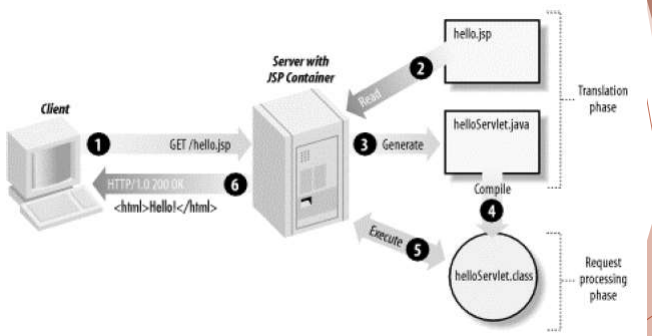
JSP vs Servlet

- Servlets: Java con HTML embebido
- JSPs: HTML con Java embebido
- Semejanzas:
  - JSP son una extensión de Servlets. No aporta funcionalidades nuevas
  - Un JSP compilado es un Servlet
  - Misma función: construir contenido dinámico
- Diferencias:
  - JSP separa más claramente el diseño de la lógica
  - Más sencillo modificar código HTML que miles de sentencias println

JSP
(Java Server Pages)

Dra. Maricela Bravo

JSP: Introducción



Tipos de Scripting

- **Expresiones**
  - Format: <%= expression %>
  - Evaluado e insertado en la salida del servlet.
- **Scriptlets**
  - Format: <% code %>
  - Insertado en el método –jspService (llamado por servicio)
- **Declaraciones**
  - <%! code %>
  - Insertado en el cuerpo de la clase del servlet, fuera de cualquier método existente

JSP: Expresiones

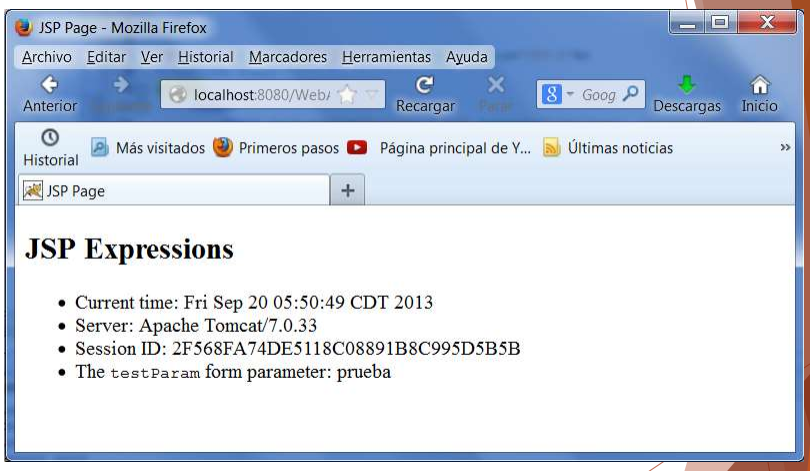
- EXPRESIONES: `<%= expresión %>` Se evalúan y se insertan en la salida.
  - Se tiene acceso a variables:
    - request, el HttpServletRequest
    - response, el HttpServletResponse
    - session, el HttpSession asociado con el request (si existe)
    - out, el PrintWriter (una versión con buffer del tipo JspWriter) usada para enviar la salida al cliente.

```
Hostname: <%= request.getRemoteHost() %>
```

Ejemplo: ¿Como visualizar un contador de visitas en una JSP?

Ejemplo de Expresiones

```
<BODY>
<H2>Ejemplo Expresiones</H2>
<UL>
  <LI>Hora actual: <%= new java.util.Date() %></LI>
  <LI>Server: <%= application.getServerInfo() %></LI>
  <LI>Session ID: <%= session.getId() %></LI>
  <LI>El <CODE>testParam</CODE> formulario de parametros:
     <%= request.getParameter("testParam") %></LI>
</UL>
</BODY>
</HTML>
```



JSP: Scriptlets

- SCRIPTLETS: `<% código %>` que se insertan dentro del método service del servlet.
  - Tienen acceso a las mismas variables que las expresiones.
  - El código dentro de un scriptlet se insertará **exactamente** como está escrito, y cualquier HTML estático (plantilla de texto) anterior o posterior al scriptlet se convierte en sentencias print.

Ejemplo:

```
<html>
<% for ( int i = 0 ; i < 10 ; i ++ )
        out.println("<br>"+i);%>
</html>
```

JSP: Scriptlets

```
<% if (Math.random() < 0.5) { %>
 Ten un <B> bonito </B> dia!
<% } else { %>
  Ten un <B> flojo </B> dia!
<% } %>
```

- Que se traducirá en:

```
if (Math.random() < 0.5) {
   out.println(" Ten un <B> bonito </B>
  dia!");}
else {
   out.println(" Ten un <B> flojo </B> dia!");}
```

Ejemplo Scriplets:

```
<!DOCTYPE html>
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   </head>
   <body>
      <% if (Math.random() < 0.5) { %>
      <H1>Ten un <I> bonito </I> dia!</H1>
      <% } else { %>
      <H1>Ten un <I> flojo </I> dia!</H1>
      <% } %>

   </body>
</html>
```

JSP: Declaraciones

- DECLARACIONES: `<%! codigo %>` que se insertan en el cuerpo de la clase del servlet, fuera de cualquier método existente.
  - Permite insertar métodos, variables...
  - No generan salida alguna. Se usan combinadas con scriptlets.

```
<%! private int accessCount = 0; %>
   Accesos a la página desde el reboot:
<%= ++accessCount %>
```

Ejemplo Declaraciones:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!--
Ejemplo de Declaración JSP.
-->
<HTML>
<HEAD>
<TITLE>Uso de declaraciones JSP</TITLE>
<LINK REL=STYLESHEET
    HREF="JSP-Styles.css"
    TYPE="text/css">
</HEAD>
<BODY>
<H1>Declaraciones JSP</H1>
<%! private int accessCount = 0; %>
<H2>Accesos a la página desde el inicio:
<%= ++accessCount %></H2>
</BODY>
</HTML>
```

Ejemplo: Scriplets

```
<HTML>
<HEAD> <TITLE>Prueba de color</TITLE></HEAD>
<%
String bgColor = request.getParameter("bgColor");
if ((bgColor == null) || (bgColor.trim().equals(""))) {
  bgColor = "WHITE";
} %>
<BODY BGCOLOR="<%= bgColor %>">
<H2 ALIGN="CENTER">Probando el color "<%= bgColor %>".</H2>
<BR>
<FORM>
  Color: <INPUT TYPE="TEXT" NAME="bgColor"><BR>
  <INPUT TYPE="SUBMIT" VALUE="Probar el Color">
</FORM>
</BODY>
</HTML>
```

Ejemplo: Scriplets

```
public class RanUtilities {
  public static int randomInt(int range) {
    return(1 + ((int)(Math.random() * range)));
  }
  public static void main(String[] args) {
    int range = 10;
    try {
      range = Integer.parseInt(args[0]);
    } catch(Exception e) {

    }
    for(int i=0; i<100; i++) {
      System.out.println(randomInt(range));
    }
  }
}
```

Ejemplo 5: Scriplets, cont…

```
<BODY>
<H1>Random List</H1>
<UL>
<%
int numEntries = coreservlets.RanUtilities.randomInt(10);
for(int i=0; i<numEntries; i++) {
%>
<LI><%= coreservlets.RanUtilities.randomInt(10) %>
<% } %>
</UL>
</BODY>
```

Ejercicio

- Realiza dos JSP:
  1. El primero que genere una lista de cinco números aleatorios entre el 1 y el 10. Usar expresiones.
  2. El segundo que genere una lista entre 1 a 10 números (seleccionados aleatoriamente), cada uno de los cuales sea un número al azar entre el 1 y el 10. Usar un scriplet.

JSP: Declaraciones

- Como con los scriptlet, si queremos usar los caracteres "%>", ponemos "%\>".
- El equivalente XML de <%! Código %> es:

```
<jsp:declaration>
   Código
</jsp:declaration>
```

JSP: Directivas

- Afectan a la estructura general de la clase servlet. Normalmente tienen la siguiente forma:

```
<%@ directive attribute="value" %>
```

- También podemos combinar múltiples selecciones de atributos para una sola directiva:

```
<%@ directive   attribute1="value1"
        attribute2="value2"
        ...
        attributeN="valueN" %>
```

JSP: Directivas

- PAGE:
  - **import**="package.class" o import="package.class1,...,package.classN". Esto permite especificar los paquetes que deberían ser importados. El atributo import es el único que puede aparecer múltiples veces.
  - **ContentType** = "MIME-Type" o contentType = "MIME-Type; charset = Character-Set" Esto especifica el tipo MIME de la salida. El valor por defecto es text/html. Tiene el mismo valor que el scriptlet usando "response.setContentType".
  - isThreadSafe="true|false". Un valor de true (por defecto) indica un procesamiento del servlet normal, donde múltiples peticiones pueden procesarse simultáneamente con un sólo ejemplar del servlet, bajo la suposición que del autor sincroniza los recursos compartidos. Un valor de false indica que el servlet debería implementar SingleThreadModel.

JSP: Directivas

- session="true|false". Un valor de true (por defecto) indica que la variable predefinida session (del tipo HttpSession) debería unirse a la sesión existente si existe una, si no existe se debería crear una nueva sesión para unirla. Un valor de false indica que no se usarán sesiones, y los intentos de acceder a la variable session resultarán en errores en el momento en que la página JSP sea traducida a un servlet.
- buffer="sizekb|none". Esto especifica el tamaño del buffer para el JspWriter out. El valor por defecto es específico del servidor y debería ser de al menos 8kb.
- autoflush="true|false". Un valor de true (por defecto) indica que el buffer debería descargase cuando esté lleno. Un valor de false, raramente utilizado, indica que se debe lanzar una excepción cuando el buffer se sobrecargue. Un valor de false es ilegal cuando usamos buffer="none".

JSP: Directivas

- **extends**="package.class". Esto indica la superclase del servlet que se va a generar. Debemos usarla con extrema precaución, ya que el servidor podría utilizar una superclase personalizada.
- info="message". Define un string que puede usarse para ser recuperado mediante el método getServletInfo.
- **errorPage**="url". Especifica una página JSP que se debería procesar si se lanzará cualquier Throwable pero no fuera capturado en la página actual.
- **isErrorPage**="true|false". Indica si la página actual actúa o no como página de error de otra página JSP. El valor por defecto es false.
- **language**="java". En algunos momentos, esto está pensado para especificar el lenguaje a utilizar. Por ahora, no debemos precuparnos por él ya que java es tanto el valor por defecto como la única opción legal.

JSP: Directivas

- INCLUDE: Permite incluir archivos en el momento en que la página JSP es traducida a un servlet.

```
<%@ include file="url relativa" %>
```

- Los contenidos del archivoincluido son analizados como texto normal JSP y así pueden incluir HTML estático, elementos de script, directivas y acciones.
- Uso: Barras de navegación.

JSP: Directivas

- La síntaxis XML para definir directivas es:

```
<jsp:directive.TipoDirectiva atributo=valor />
```

Por ejemplo, el equivalente XML de:

```
<%@ page import="java.util.*" %>
```

es:

```
<jsp:directive.page import="java.util.*" />
```

JSP: Variables predefinidas

- **REQUEST:** Este es el HttpServletRequest asociado con la petición,.
- Nos permite leer los parámetros de la petición (mediante `getParameter`), el tipo de petición (GET, POST, HEAD, etc.), y las cabeceras HTTP entrantes (cookies, Referer, etc.). Estrictamente hablando, se permite que la petición sea una subclase de ServletRequest distinta de HttpServletRequest, si el protocolo de la petición es distinto del HTTP. Esto casi nunca se lleva a la práctica.

JSP: Variables predefinidas

- **RESPONSE:** Este es el HttpServletResponse asociado con la respuesta al cliente. Como el stream de salida tiene un buffer, **es** legal seleccionar los códigos de estado y cabeceras de respuesta, aunque no está permitido en los servlets normales una vez que la salida ha sido enviada al cliente.

JSP: Variables predefinidas

- **OUT:** Este es el PrintWriter usado para envíar la salida al cliente. Sin embargo, para poder hacer útil el objeto response, ésta es una versión con buffer de PrintWriter llamada JspWriter. Podemos ajustar el tamaño del buffer, o incluso desactivar el buffer, usando el atributo buffer de la directiva page. Se usa casi exclusivamente en scriptlets ya que las expresiones JSP obtienen un lugar en el stream de salida, y por eso raramente se refieren explícitamente a out.

JSP: Variables predefinidas

- **SESSION:** Este es el objeto HttpSession asociado con la petición. Las sesiones se crean automáticamente, por esto esta variable se une incluso si no hubiera una sesión de referencia entrante. La única excepción es usar el atributo session de la directiva page para desactivar las sesiones, en cuyo caso los intentos de referenciar la variable session causarán un error en el momento de traducir la página JSP a un servlet.

JSP: Variables predefinidas

- **APPLICATION:** El ServletContext obtenido mediante `getServletConfig().getContext()`.
- **CONFIG:** El objeto ServletConfig.
- **PAGECONTEXT:** JSP presenta una nueva clase llamada PageContext para encapsular características de uso específicas del servidor como JspWriters de alto rendimiento. La idea es que, si tenemos acceso a ellas a través de esta clase en vez directamente, nuestro código seguirá funcionando en motores servlet/JSP "normales".

JSP: Variables predefinidas

- **PAGE:** Esto es sólo un sinónimo de `this`, y no es muy útil en Java. Fue creado como situación para el día que el los lenguajes de script puedan incluir otros lenguajes distintos de Java.

JSP: Acciones

- Usan construcciones de síntaxis XML para controlar el comportamiento del motor de Servlets. Podemos insertar un archivo dinámicamente, reutilizar componentes JavaBeans, reenviar al usuario a otra página, etc.

JSP: Acciones: include

a) **ACCION jsp:include** nos permite insertar archivos en una página que está siendo generada. La síntaxis se parece a esto:

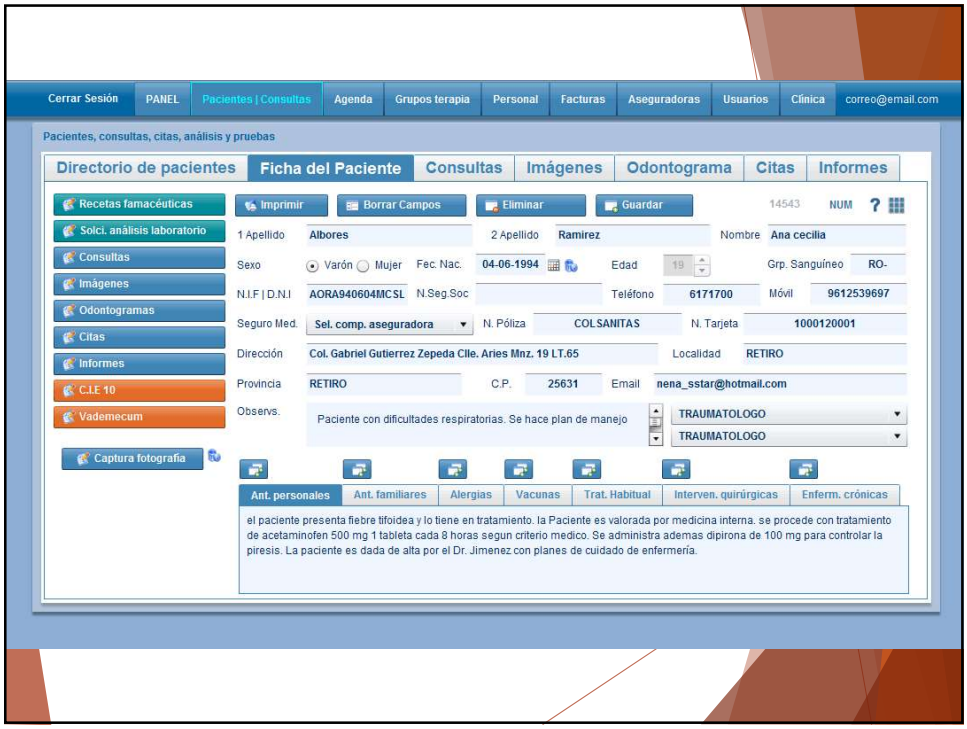
```
<jsp:include page="relative URL" flush="true" />
```

Al contrario que la directiva include, que inserta el fichero en el momento de la conversión a un Servlet, inserta el fichero cuando la página es solicitada.

Segunda aplicación TOMCAT

- Crear index.jsp con:

```
Hola mundo! A contar! :D<br>
<%
for ( int i = 0 ; i < 10 ; i ++ )
{
   %>Hola mundo!!!!!! saludo número ...<%=i%><br>
   <%
   }
%>
```


Cerrar Sesión
PANEL
Pacientes | Consultas
Agenda
Grupos terapia
Personal
Facturas
Aseguradoras
Usuarios
Clínica
correo@email.com
Pacientes, consultas, citas, análisis y pruebas
Directorio de pacientes
Ficha del Paciente
Consultas
Imágenes
Odontograma
Citas
Informes
Recetas famacéuticas
Solci. análisis laboratorio
Consultas
Imágenes
Odontogramas
Citas
Informes
C.I.E 10
Vademecum
Captura fotografía
Imprimir
Borrar Campos
Eliminar
Guardar
14543
NUM
1 Apellido
Albores
2 Apellido
Ramirez
Nombre
Ana cecilia
Sexo
Varón
Mujer
Fec. Nac.
04-06-1994
Edad
19
Grp. Sanguíneo
RO-
N.I.F | D.N.I
AORA940604MCSL
N.Seg.Soc
Teléfono
6171700
Móvil
9612539697
Seguro Med.
Sel. comp. aseguradora
N. Póliza
COLSANITAS
N. Tarjeta
1000120001
Dirección
Col. Gabriel Gutierrez Zepeda Clle. Aries Mnz. 19 LT.65
Localidad
RETIRO
Provincia
RETIRO
C.P.
25631
Email
nena_sstar@hotmail.com
Observs.
Paciente con dificultades respiratorias. Se hace plan de manejo
TRAUMATOLOGO
TRAUMATOLOGO
Ant. personales
Ant. familiares
Alergias
Vacunas
Trat. Habitual
Interven. quirúrgicas
Enferm. crónicas
el paciente presenta fiebre tifoidea y lo tiene en tratamiento. la Paciente es valorada por medicina interna. se procede con tratamiento de acetaminofen 500 mg 1 tableta cada 8 horas segun criterio medico. Se administra ademas dipirona de 100 mg para controlar la piresis. La paciente es dada de alta por el Dr. Jimenez con planes de cuidado de enfermería.