JAVA SERVER PAGES

Separemos la presentación del contenido
Mis amigos los servlets, JavaBeans, ….

Java Server Pages

- JSP: tecnología de SUN basada en los servlets.
- Permiten ejecución dinámica de contenidos HTML
- Alternativa a ASP, CGI, "Servlets",……
- Cuando hablamos de JSP, detrás siempre hay un servlet.

2

JSP

- Las JSP combinan la potencia de los servlets con una sintaxis específica no mucho más complicada que la de HTML.
- JSP se pueden utilizar junto con servlets, JavaBeans, librerías de etiquetas y clases Java para crear aplicaciones multicapa.

3

JSP & Servlets

- Existe una estrecha relación entre JSP y servlets:
  - JSP son compiladas sobre los servlets la primera vez que se ejecutan.
  - El motor JSP es en si mismo, un motor de servlets que implementa la especificación de JSP.
  - Por tanto, los servlets realizan el trabajo para las JSP.

4

JSP & Servlets

- Se utilizan las JSP porque de algún modo son servlets pero de uso más sencillo.
- Su productividad es mayor que la de los servlets.
- JSP permite una separación "limpia" de las capas:
  - Presentación…… JSP
  - Lógica de negocio……. servlet, JavaBeans, …
  - Datos…… Bases de datos

5

JSP vs. ASP

- Son muy similares.
- JSP se compila 1 vez, ASP cada vez. ASP+
- Lenguaje Script:
  - VBS, otros
  - Java:
    - más potente
    - multiplataforma
    - multithreading
    - gratis
- Cada tecnología en su correcto uso.

6

Sintaxis JSP

- JSP es una combinación de HTML, XML, Java, ….
- Declaraciones:
  - 1) <%! CÓDIGO JAVA %>
    - Se utiliza para declarar variables y métodos fuera del método service() del servlet y por tanto con un ámbito de "application".
  - 2) <% … %>
    - Declarar variables y métodos dentro del service() y por tanto con ámbito de página.

7

Sintaxis JSP

- Expresiones:

  <%= expresión Java %>

  Ej.:

  ```
  <% String outStr="Hola Mundo";%>
  <%=outStr%>
  ```

  - Se realiza una evaluación en el runtime de Java…

    Ej.:

    ```
    <%=MiCalendario.Convert(new java.util.Date())%>
    <%=(10/19)*2.3+2.4%>
    ```

  - Las expresiones no acaban en ";" ya se se pasan directamente a out.println();

8

Sintaxis JSP

- Scriplet:
  - Código Java entre <%.....%> que se inserta directamente en el método service() del servlet.
  - Pueden ser expresiones como las anteriores o algo más complejo.
  - Se ejecuta cada vez que se invoca al JSP.

```
<%
  if (diaSemana.equals("Lunes")) {
     reunionGrupo = true;
     reunión.aviso();
  } else {
     reunionGrupo = false;
  }
%>
```

9

Sintaxis JSP

- Directivas:
  - Las directivas pueden verse como mensajes al motor de JSP.
  - <@ directive attribute="value" %>
  - Existen tres tipos de directivas:
    - Página: permite definir atributos de la página.
    - Include: inserta otro fichero en el JSP que la tiene.
    - Taglib: permite definir etiquetas propias del usuario.

10

Directivas de Página

- Permite manipular algunos atributos de la página.
- Atributos:
  - autoFlush (true|false): da salida a los valores guardados desde que el buffer esté lleno.
  - buffer (none|xkb): espacio del buffer.
  - contentType: cualquier valor válido de mime.
    - <%@ page contentType="text/html" %>
  - errorPage: toma el valor de la URL de una página web no alcanzada.
    - <%@ page errorPage="ErrPage.jsp" %>

11

Directivas de Página

- Más atributos:
  - extends: selecciona a que superclase el servlet compilado debe extender.
  - import: se refiere a que paquete o clase debe importar el servlet compilado.
    - <%@ page import="java.util.Date, xxx.xxx.xxx" %>
  - info: pone el texto que devuelve el servlet en getServletInfo().
  - isErrorPage (true|false): indica si esta página es una página de error de la ejecución de otra.

12

Directivas de Página

- y también …:
  - isThreadSafe (true|false): si “true”, permite al servlet manejar requests simultáneos.
  - session (true|false): si “true” la variable de session debe ser puesta a la sesión actual.

13

Directiva de inclusión

- include:
  - Inserta otro fichero en el fichero JSP que lo contiene.
  - Puede ser un fichero texto, un fichero Javascript , otro JSP o cualquier tipo si está disponible en el container.
    - <%@ include file=“scriptlets/validation.js” %>

14

Directiva “taglib”

- taglib:
  - Permite el uso de etiquetas personalizadas.
  - Tienen 2 atributos en la forma:
    - <%@ taglib uri=“my_uri.tld” prefix=“crcb” %>
  - El primer atributo indica el nombre de la librería.
  - El segundo indica al container el prefijo que utilizaré con esa librería.
  - Antes de utilizar los tags, debemos crear la clase Java que soporte dichos tags y un Tag Library Descriptor.

15

Ámbito de las JSP

- El ámbito de los objetos ASP define cuánto tiempo y desde qué JSP los objetos pueden ser utilizados.
  - Objeto session tiene un ámbito que excede el de una página.
  - Objeto application puede ofrecer servicios a un grupo de páginas JSP que forman una aplicación.
- El ámbito de las JSP es función del contexto. Un contexto proporciona un “container” de recursos y una interfaz de comunicación entre ellos:
  - Cualquier información acerca de su servidor, un servlet lo solicita al contexto donde se está ejecutando.
  - Cualquier información que el servidor desee pasar al servlet lo hará a través del contexto.

16

Ámbito de las JSP

- Ambito de "application":
  - Es el ámbito más persistente. Un objeto con este ámbito está disponible a todas las páginas y a todos los clientes en el mismo ServletContext. Se pueden declarar variables globales:
    - getServletContext().setAttribute("complaint", complaint);
- Ambito de "session":
  - Los objetos están disponibles a todas las páginas con el HttpSession actual:
    - HttpSession sess = request.getSession(true);
    - sess= setAttribute("complaint", complaint);

17

Ámbito de las JSP

- Ambito de "request":
  - Los objetos serán sólo accesibles al HttpRequest actual.
    - Request.setAttribute("complaint", complaint);
- Ambito de "page":
  - Este es el valor por defecto. Un servlet no puede utilizarlo para pasarlo a otro servlet, etc…

18

Comentarios en JSP

- HTML:

  <!-- ……. -->

- Java:

```
<%
// comentario de una línea
%>
<%
/*  comentario de
    varias lineas
*/
%>
```

- JSP:
  - <%-- comentario JSP --%>
- Los comentarios no aparecen en el resultado HTML.

19

Caracteres especiales

- Usemos <\% cuando queramos ver <%
- Usemos %\> cuando queramos ver %>
- Usemos \' para poner una comilla en un atributo.
- Usemos \" para poner doble comilla en un atributo.

20

Objetos implícitos

- JSP ofrece un conjunto de objetos implícitos también conocidas como variables predefinidas.

application
config
exception
page
pageContext
out
request
response
session

21

Sentencias <jsp:action>

- Las posibles acciones son:
  - JSP:include
    - permite incluir un fichero cuando la página es solicitada.
  - JSP:forward
    - Redirige la petición a una nueva página.
  - JSP:plugin
    - Genera código específico del navegador.
  - JSP:getProperty
    - Obtiene una propiedad de un Bean.
  - JSP:setProperty
    - Pone una propiedad en un Bean.
  - JSP:useBean
    - Busca o instancia un JavaBean.

22

<jsp:include>

- Esta acción permite incluir un fichero que será evaluado en tiempo de ejecución, más que cuando el servlet es compilado.
- No se pueden utilizar expresiones de JSP.
- Si se permite incluir JSP o servlets, pero se tomará su salida como entrada de este include.
- Suele utilizarse para incluir HTML o JavaScript.
  - <jsp:include page="comun/piedepagina.html" flush="true">

23

<jsp:forward>

- Permite que una JSP pueda redirigirse a un objeto solicitado: JSP, servlet o HTML.
- El destino debe estar en el mismo contexto.

```
<jsp:forward page="error/errorpage.html">
```

```
<jsp:forward page="error/errorpage.html">
  <jsp:param name="tipoerror" value="3">
</jsp:forward>
```

24

<jsp:plugin>

- Nos permite insertar applets que utilizan el java plugin.

```
<jsp:plugin
            type="applet",
            code="package.myClass",
            width="500",
            height="300",
            align="top">
    <jsp:params>
        <jsp:param name="nombre" value="Pepe">
    </jsp:params>
<jsp:plugin
```

25

<jsp:getProperty>

- Nos permite obtener las propiedades de un Bean.
- Se da el valor como una String.

```
<jsp:getProperty name="beanInstanceName" property="propertyName" />
```

26

<jsp:setProperty>

- Nos permite modificar o insertar una valor en una propiedad del Bean.

```
<jsp:setProperty name="beanInstanceName"
                 property="propertyName"
                 param="requestParameterName"
/>
```

alternativa

```
<jsp:setProperty name="beanInstanceName"
                 property="propertyName"
                 value="valordelapropiedad"
/>
```

27

<jsp:useBean>

- Permite la utilización de un Bean desde una JSP.

```
<jsp:useBean
    id="identBean"
    scope="page | request | session | application "
    beandetails
        class : la clase en si.
        type  : para hacer un cast a un tipo diferente de bean.
        beanName : nombre del bean.
>
```

```
<jsp:useBean id="complaint" class="jspintro.ComplaintBean" scope="session"/>
```

28

Ejemplo FormProc.html

```
<FORM METHOD="POST" ACTION="http://localhost:8080/ej/FormProc.jsp">
<input type="hidden" name="status" value="unresolved">
<pre>
  Name: <input type="text" name="offenderName" size=25>
Offense: <input type="text" name="offense" size=25>
</pre>
<P><B>Complaint by:</B> (Check all that apply)<BR>
<input type="checkbox" name="client"> Client<BR>
<input type="checkbox" name="coworker"> Coworker<BR>
<input type="checkbox" name="manager"> Manager</P>
<P><B>Offense number:</B><BR>
<input type="radio" name="offenseNumber" value="1" checked>First<BR>
<input type="radio" name="offenseNumber" value="2">Second<BR>
<input type="radio" name="offenseNumber" value="3">Third</P>
<P><input type="submit" value="Submit Complaint"></P>
</FORM>
```

29

Ejemplo FormProc.jsp

```
<%@ page info="handles FormTest.html input" %>
<HTML>
<HEAD><TITLE>Output</TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<%-- Declaration section (incorrect version) --%>
<%
  int offenseNumber;
  int highestComplaintLevel = 1;
  int numberOfComplaints = 0;
  int seriousness;
  int pageAccesses;
%>
<%-- a scriptlet --%>
<%
  pageAccesses++;
```

30

Ejemplo FormProc.jsp

```
try {
   // get offense number from request
   offenseNumber =
         Integer.parseInt(request.getParameter("offenseNumber"));
  } catch(NumberFormatException nfe) {
   // if unavailable from request, assume it is first offense
   offenseNumber = 1;
  }
  if (request.getParameter("client") != null) {
   highestComplaintLevel = 3;
   numberOfComplaints++;
  }
  if (request.getParameter("manager") != null) {
   highestComplaintLevel = Math.max(highestComplaintLevel, 2);
   numberOfComplaints++;
  }
```

31

Ejemplo FormProc.jsp

```
if (request.getParameter("coworker") != null) {
   highestComplaintLevel = Math.max(highestComplaintLevel, 1);
   numberOfComplaints++;
  }
  if (numberOfComplaints < 1) numberOfComplaints = 1;

  seriousness = offenseNumber * highestComplaintLevel * numberOfComplaints;
%>
<P><I>page accesses: <%= pageAccesses %></I></P>
<P><B>Offender: </B><%= request.getParameter("offenderName") %><BR>
  <B>Offense: </B><%= request.getParameter("offense") %><BR>
  <B>Offense Number: </B><%= offenseNumber %><BR>
  <B>Number Complaining: </B><%= numberOfComplaints %><BR>
  <B>Complaint Level: </B><%= highestComplaintLevel %></P>

<P><B>Severity of offense: </B><%= seriousness %></P>
<P><B>Recommended Action: </B>
```

32

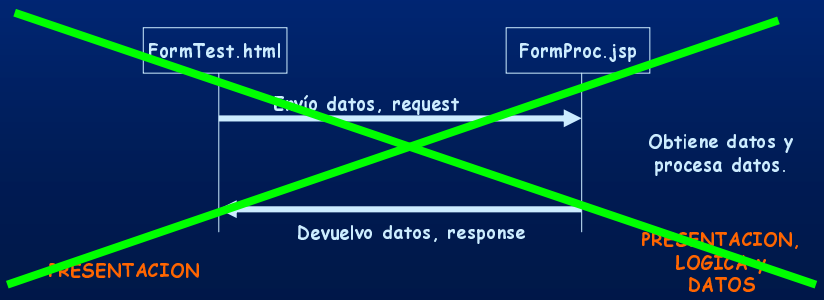
Ejemplo FormProc.jsp

```
<%   if (seriousness < 10) { %>
<FONT COLOR="#000099">WARNING</FONT>
<%   } else if (seriousness < 19) { %>
<FONT COLOR="#009900">REPRIMAND</FONT>
<%   } else { %>
<FONT COLOR="#FF0000">TERMINATION</FONT>
<%
 }
%>
</P>
<HR SIZE=1 NOSHADE>
<P><B>Complaint Status: </B>
   <%= request.getParameter("status") %></P>
</BODY>
</HTML>
```

33

JSP en la arquitectura Multicapa

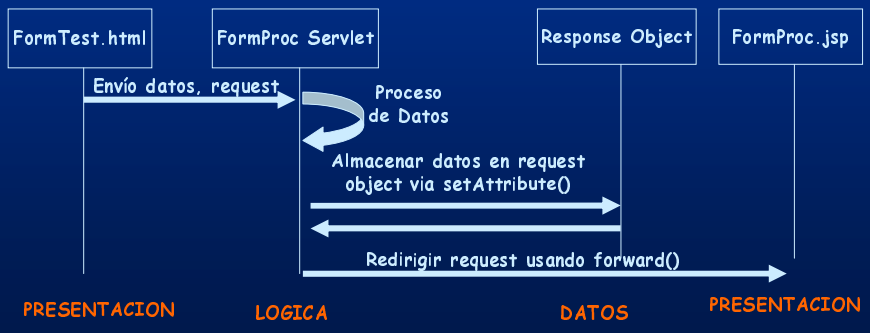
- Buena práctica: mínimo código Java en JSP.
- Separar en tres capas:
  - Datos……. Bases de datos, ficheros.
  - Aplicaciones o reglas de negocio……. servlets, clases, JavaBeans.
  - Lógica de presentación……. JSP y HTML



34

JSP en la arquitectura Multicapa

- Datos……. Bases de datos, ficheros.
- Aplicaciones o reglas de negocio……. servlets, clases, JavaBeans.
- Lógica de presentación……. JSP y HTML



35

JSP en la arquitectura Multicapa

```
req.setAttribute("offenseNumber", offensenumber);
req.setAttribute("highestComplaintLevel", highestcomplaintlevel);
req.setAttribute("seriousness", seriousness);
//……
RequestDistpatcher rd =
   getServletContext().getRequestDispatcher("/FormProc.jsp");

rd.forward(req, res);
```

36

JavaBeans

- Es un componente reutilizable en diferentes aplicaciones Java.
- Basado en la especificación de Sun de JavaBeans:
  - java.sun.com/products/javabeans
- Desde el punto de vista del cliente, un JavaBean es un caja negra. Sabe que puede "pedir" a ese componente pero no sabe como está hecho internamente.
- Los ejemplos que veremos están compuestos de una sola clase, pero los JB pueden tener varias clases.

37

JavaBeans: propiedades simples

- Los JB tienen propiedades a las que se puede acceder a través de métodos bien definidos para las lecturas y modificaciones de las propiedades de un JB.
- La especificación detalla estos aspectos.
- Ejemplo:

```
public class MyBean implements Serializable {
    protected String myProperty="No disponible";
    private int width = 0;
    private int height = 0;
    private boolean threeD= false;
    //…..
```

38

Y segunda parte del ejemplo JavaBeans: propiedades simples

```
public String getMyProperty() {
  return this.myProperty();
}
public void setMyProperty(String myProperty) {
  this.myProperty= myProperty;
}
public String getMyProperty() {
  return this.myProperty();
}
public double getArea() {
  return (double) width* height;
}
public String getInfo() {
  return "Ejemplo de bean";
}
///…
```

39

JavaBeans: propiedades simples

- Los nombres de los beans deben acabar con "Beans".
- "MyBean" extiende Object de forma explícita
- MyBean implementa Serializable y por tanto permite guardar su estado y recuperarlo.
- Variables son protected o private, no public.
  - Son accedidas a través de los métodos get/set.
- Lectura de variables:
  - public <PropertyDataType> get<PropertyName>()
- Escritura de variables:
  - public void set<PropertyName>(<PropertyDataType> <PropertyName>)

40

Otro ejemplo

```
import java.util.*;
public class beanMostrarFecha {
private int contador = 0;
private String cadenaFecha = null;
public beanMostrarFecha() {
  super();
  cadenaFecha = construirCadenaFecha(new GregorianCalendar());
  contador = 0;
}
public String construirCadenaFecha(GregorianCalendar calendario) {
  StringBuffer fecha = new StringBuffer();
  fecha.append(calendario.get(Calendar.DATE));
  fecha.append("/");
  fecha.append(calendario.get((Calendar.MONTH) + 1));
  fecha.append("/");
  fecha.append(calendario.get(Calendar.YEAR));
  return fecha.toString();
}
public int getContador() {
  return contador;
}
public void setContador(int nuevoValor) {
  contador = nuevoValor;
}
public java.lang.String getcadenaFecha() {
  contador++;
  return cadenaFecha;
}
}
```

41

Otro ejemplo

```
<html>
<head>
<title>Mostrar fecha invocando un JavaBean desde una p&aacute;gina
JSP</title>
</head>
<body>
<jsp:useBean id="beanMostrarFecha" class="beanMostrarFecha"
scope="session" />
<h2> La fecha de hoy es:
<jsp:getProperty name="beanMostrarFecha"
property="cadenaFecha"/>
</h2>
<h2>Esta p&aacute;gina ha sido invocada:
<jsp:getProperty name="beanMostrarFecha" property="contador"/>
veces
</h2>
</body>
</html>
```

42