Servlets y JSP

- Sesión 7: JavaBeans y lenguaje de expresiones

Puntos a tratar

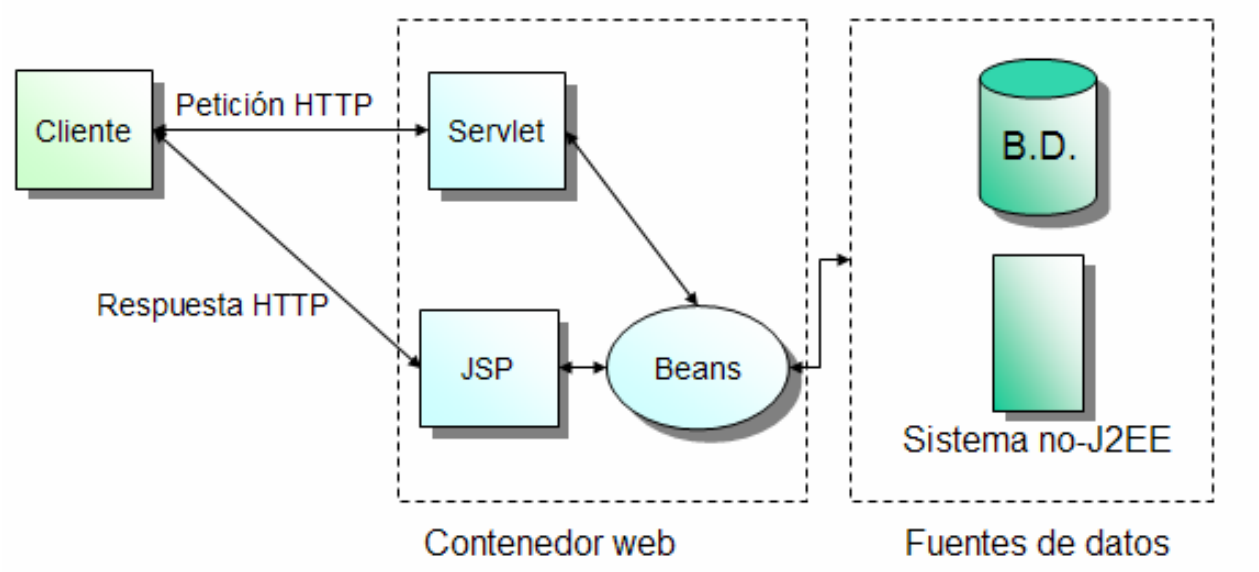
- JavaBeans y JSP
- Uso básico de beans desde JSP
  - Creación
  - Acceso a las propiedades
  - Asignación de las propiedades
- Compartir beans entre JSPs
- Lenguaje de expresiones en JSP 2.0

JavaBeans

- Componente software reutilizable escrito en Java
- La clase del bean cumple una cierta serie de convenciones en cuanto a la forma de acceder y cambiar los valores de sus campos
- Se tiene así un componente software accesible desde otras herramientas mediante etiquetas, sin código
  - JSP: representar datos de la aplicación: usuarios, pedidos, ...
  - IDE: componentes visuales
  - ...

JavaBeans y JSP

- **Ventajas**
  - Evitar sintaxis Java
  - Separar datos de presentación
  - Objetivo final: MVC ⇨ modelo



Características de un *bean*

- Debe tener al menos un constructor sin argumentos
- No debe tener variables miembro de acceso público. El acceso es mediante métodos
- El nombre de los métodos de acceso y modificación debe seguir una convención
  - ⇨ `getNombreVar()` – acceso
  - ⇨ `isNombreVar()` – acceso variables booleanas
  - ⇨ `setNombreVar(tipo-variable)` – modificación

Ejemplo de bean

```
package beans;
public class UsuarioBean {
   private String nombre;
   private boolean varon;
   private int visitas;
   private Date ultimaVisita;

   public UsuarioBean() { ... }

   public String getNombre() { return nombre; }
   public boolean isVaron() { return varon; }
   public int getVisitas() { return visitas; }
   public Date getUltimaVisita() { return ultimaVisita; }

   public void setNombre(String n) { nombre=n; }
   public void setVaron(boolean b) { varon=b; }
   public void setVisitas(int v) { visitas=v; }
   public void setUltimaVisita(Date d) {ultimaVisita=d; }
}
```

Acceso al *bean*

- Acción <jsp:useBean />

```
<jsp:useBean id="nombreBean" class="paquete.Clase"/>
<jsp:useBean id="usuario" class="beans.UsuarioBean" />
```

- Si el *bean* no existía, esto es equivalente a

```
<% beans.UsuarioBean usuario = new beans.UsuarioBean()
%>
```

- La clase debe colocarse en *WEB-INF/classes*, con sus paquetes y subpaquetes (recomendado asignar paquete)

Acceso a las propiedades de un *bean*

- Acción <jsp:getProperty/>

```
<jsp:getProperty name="nomBean" property="nomPropiedad"/>
<jsp:getProperty name="usuario" property="visitas"/>
```

- Equivalente al código Java

```
<%= usuario.getVisitas() %>
```

Asignación de valores

- Acción <jsp:setProperty/>

```
<jsp:setProperty name="nomBean" property= "nomProp"
                 value="valProp"/>
<jsp:setProperty name="usuario" property="visitas"
                 value="1"/>
```

- Equivalente a

```
<% usuario.setVisitas(1) %>
```

- Se puede poner cód. Java (en ocasiones será inevitable)

```
<jsp:setProperty name="usuario" property="ultimaVisita"
value="<%= new java.util.Date() %>"/>
```

Inicializar un bean con valores explícitos

- Por defecto se construye el bean vacío
- Si queremos modificar algún valor, meter <jsp:setProperty/> dentro de <jsp:useBean/>

```
<jsp:useBean id="usuario" class="beans.usuarioBean">
   <b> inicializando datos de usuario </b>
   <jsp:setProperty name="usuario" property="ultimaVisita"
                    value="<%= new java.util.Date() %>"/>
</jsp:useBean>
```

- El código interno SOLO se ejecuta si el bean no existe

Tomar los valores de la petición

- Muy típico: tomar los datos de un formulario HTTP que llame a otra página y meterlos allí en un bean.

```
<html>
<body>
 <form action="main.jsp" method="get">
     Nombre <input type="text" name="nombre">
     <br>
     Sexo:
     varon <input type="radio" name="varon" value="true">
     mujer: <input type="radio" name="varon" value="false">
     <br>
     <input type="submit" value="entrar">
</form>
</body>
</html>
```

Tomar los valores de la petición (II)

- Parámetro `param` de `<jsp:setProperty/>`: toma el valor del parámetro del mismo nombre de la petición HTTP

```
<jsp:useBean id="usuario" class="beans.UsuarioBean"/>
<jsp:setProperty name="usuario" property="nombre"
                 param="nombre"/>
<jsp:setProperty name="usuario" property="varon"
                 param="varon"/>
Buenos días,
<jsp:getProperty name="usuario" property="nombre"/>
```

Tomar los valores de la petición (III)

- Tomar todos los valores de la petición HTTP y meterlos en el bean si coinciden con los campos

```
<jsp:useBean id="usuario" class="beans.UsuarioBean"/>
<jsp:setProperty name="usuario" property="*" />
Buenos días,
<jsp:getProperty name="usuario" property="nombre"/>
```

Compartir beans

- Por defecto, el bean es una “variable” local a la página y a la petición
- Se puede controlar el ámbito con el parámetro `scope` de `<jsp:useBean/>`
  - `page:` por defecto
  - `request:` se conserva aunque se haga un forward o include
  - `session`
  - `application`

```
<jsp:useBean id="usuario" class="beans.UsuarioBean“
                    scope=“session”/>
```

Introducción al EL

- Desde JSP 2.0 se dispone de un lenguaje de expresiones que permite realizar varias tareas, algunas de las cuales se hacían hasta ahora con scriptlets <%=...%>
- Antes, este lenguaje sólo estaba disponible en JSTL, una librería de etiquetas adicionales para JSP

```
<p>La variable edad se ha establecido a ${edad}</p>
<h4>El parametro nombre vale ${param.nombre}</h4>
<c:if test="${persona.edad > 18}">
   ...
</c:if>
```

Atributos y expresiones

- El lenguaje se puede utilizar en atributos de etiquetas JSTL, o dentro de la parte HTML de la página, y se invoca con el elemento *${...}*

```
${expresion}
```

- Podemos utilizar estos elementos:
  - Solos en atributos de etiquetas JSTL

```
<pref:etiq value="${expresion}"/>
```

  - Combinados con otros elementos en atributos JSTL

```
<pref:etiq value="texto${expr1} y ${expr2} texto"/>
```

  - Dentro del contenido HTML de la página

```
<h4>Hola, esto es la variable ${miVar}</h4>
```

Operadores

- Dentro de las expresiones podemos utilizar operadores:
  - *Operadores [ ] y .* son equivalentes:

```
${expr[campo]}=${expr.campo}
```

  - *Operadores aritméticos*: +, -, *, /, div, mod, %
  - *Operadores relacionales*: >, gt, <, lt, >=, ge, <=, le, ==, eq, !=, ne
  - *Operadores lógicos*: &&, and, ||, or, !, not
  - *Operador "empty"*: para indicar si un elemento es nulo o vacío (devuelve *true*) o no (devuelve *false*)

```
<c:if test="${empty A}">
   ...
```

Nombres de variables

- Los nombres de variables que pongamos se evalúan según *pageContext.findAttribute(...)*, buscando la variable en la página, petición, sesión, etc. Si no lo encuentra, devuelve *null*
- Hay algunas palabras reservadas que no podemos usar como nombres: *and, eq, or, lt... etc.*
- Cuando como nombre de la variable utilizamos ciertos nombres implícitos, se devuelve el objeto al que pertenecen:

```
${param.id}
```

  - Devolvería el valor del parámetro *id* de la petición
  - Otros nombres implícitos: *header, sessionScope, cookie...*

Ejemplos

```
// Devuelve la suma de las dos variables
${num1 + num2}

// true si v1 es distinto a v2 y v3 menor que v4
${v1 ne v2 and v3 < v4}

// obtiene el atributo profile de la sesión
${sessionScope.profile}

// obtiene el parámetro password de la petición
${param.password}

// true si el parámetro nombre de la petición está vacío o nulo
${empty param.nombre}
```