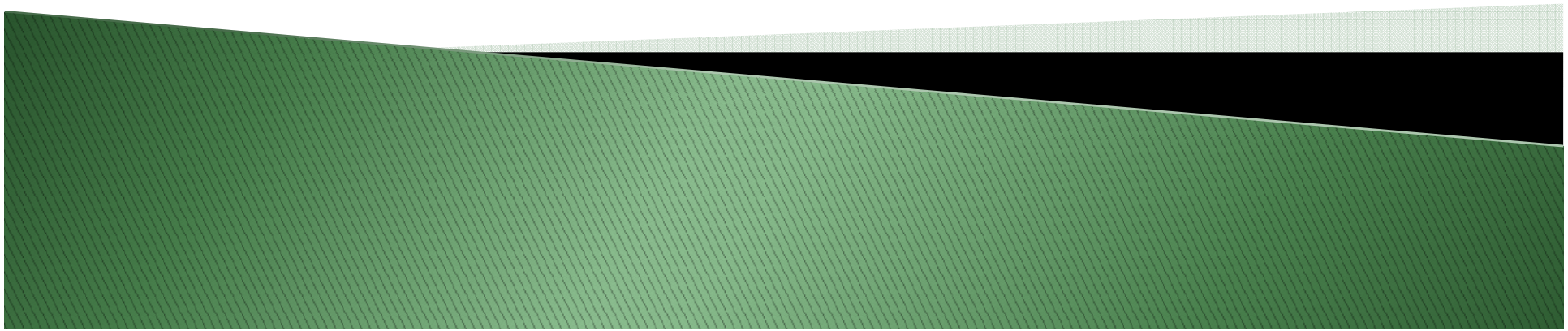
Unidad 6

Java Server Pages

Despliegue de aplicaciones web



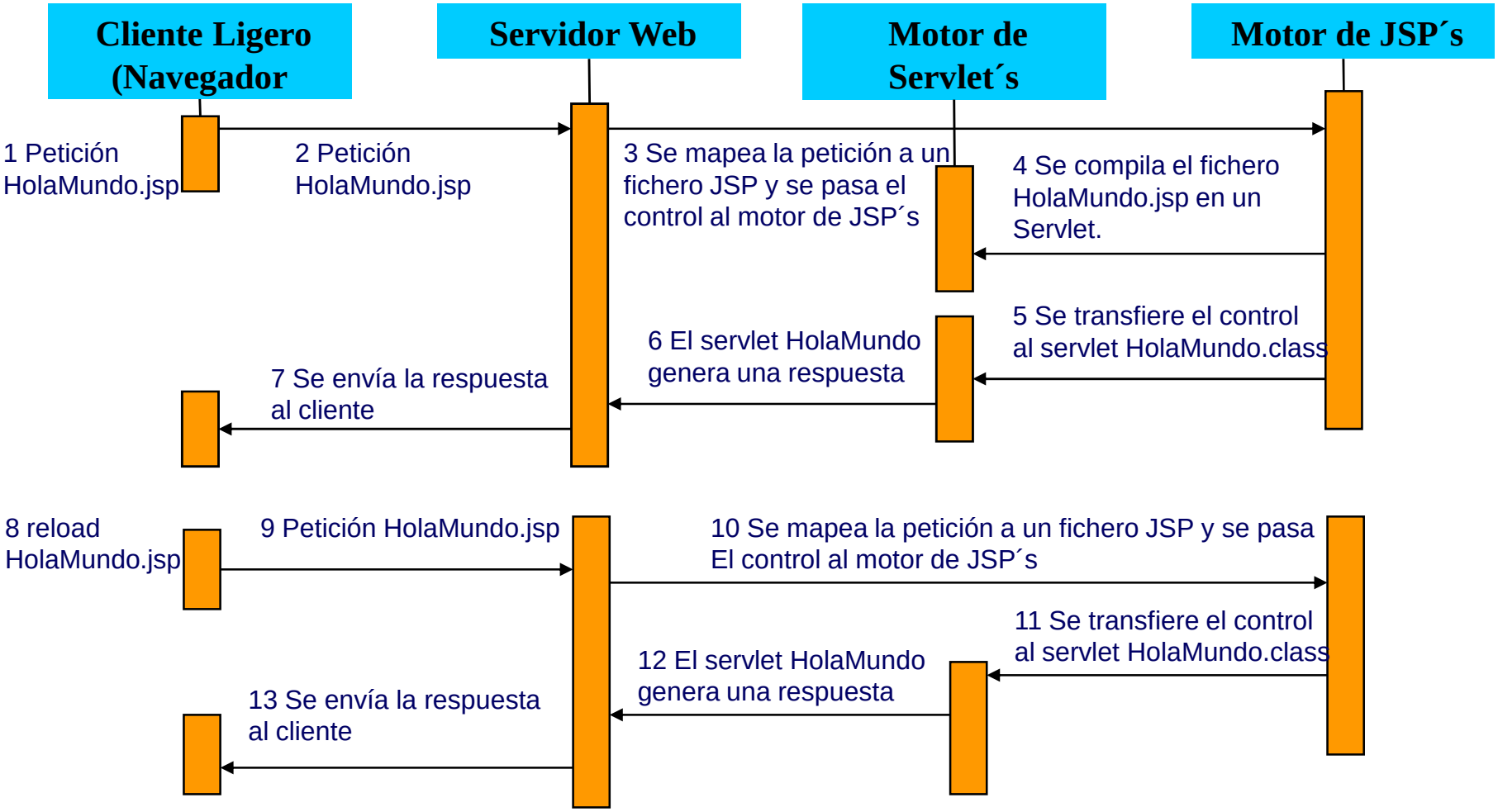
Índice de contenidos

La especificación JSP – ¿Qué es una JSP?

- Las JavaServer Pages (JSP) nos permiten separar la parte dinámica de nuestras páginas Web del código estático. Para ello:
  - Simplemente escribimos el HTML (o XML) regular de la forma normal, usando cualquier herramienta de construcción de paginas Web.
  - Encerramos el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con "<%" y terminan con "%>.

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo Hola mundo  en una JSP</TITLE>
  </HEAD>
  <BODY>
    <% out.println("Hola mundo desde una JSP"); %>
  </BODY>
</HTML>
```

La especificación JSP – Funcionamiento



Elementos JSP

La especificación JSP – Elementos de script JSP y su notación XML

- Los elementos de script nos permiten insertar código Java dentro del servlet que se generará desde la página JSP actual. Hay tres formas:
  1. **Expresiones** de la forma <%= expresión %> que son evaluadas e insertadas en la salida.
  2. **Scriptlets** de la forma <% codigo %> que se insertan dentro del método service() del servlet.
  3. **Declaraciones** de la forma <%! codigo %> que se insertan en el cuerpo de la clase del servlet, fuera de cualquier método existente.
  4. **Directivas** de la forma <%@ directive atributo1="valor1"... atributoN="valorN" %> que afectan a la estructura general del servlet generado.
  5. **Acciones** de la forma <jsp:accion /> que permiten realizar operaciones como acceso a JavaBeans, inclusión de paginas,...

La especificación JSP – Expresiones JSP

- Una **expresión JSP** se usa para insertar valores Java directamente en la salida. Tiene la siguiente forma:

  ```
  <%= expresión Java %>
  ```

  La expresión Java es evaluada, convertida a un string, e insertada en la página. Esta evaluación se realiza durante la ejecución (cuando se solicita la página) y así tiene total acceso a la información sobre la solicitud. Por ejemplo, esto muestra la fecha y hora, del servidor, en que se solicitó la página:

  ```
  Current time: <%= new java.util.Date() %>
  ```

- **Notación XML:**

  ```
  <jsp:expression>  Expresión Java </jsp:expression>
  ```

La especificación JSP – Ejemplo1: Expresiones

Fichero Expresiones.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Ejemplo1: Expresiones</TITLE>
</HEAD>

<BODY>
<H2>Ejemplos de expresiones JSP</H2>
<UL>
  <LI>Hora actual en el servidor: <%= new java.util.Date() %>
  <LI>Nombre del host: <%= request.getRemoteHost() %>
  <LI>Identificador de sesion: <%= session.getId() %>
  <LI>Valor del parametro testParam:
      <%= request.getParameter("testParam") %>
</UL>
</BODY>
</HTML>
```

La especificación JSP – Scriptlets JSP

- Si queremos hacer algo más complejo que insertar una simple expresión, los **Scriptlets JSP** nos permiten insertar código Java arbitrario dentro del método servlet que será construido al generar la página. Los Scriptlets tienen la siguiente forma:

  <% Código Java %>

  El código dentro de un scriplet se insertará **exactamente** como está escrito, y cualquier HTML estático anterior o posterior al scriptlet se convierte en sentencias **print().** Por tanto no se necesita completar la sentencias Java, y los bloques abiertos pueden afectar al HTML estático fuera de los scriplets.
- **Notación XML**:

  <jsp:scriptlet> Código </jsp:scriptlet>

La especificación JSP – Ejemplo 2: Scriplets JSP

Fichero BGColor.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Color de Fondo</TITLE>
</HEAD>

<%
String bgColor = request.getParameter("bgColor");
boolean hasExplicitColor;
if (bgColor != null) {
  hasExplicitColor = true;
} else {
  hasExplicitColor = false;
  bgColor = "WHITE";
}
%>
<BODY BGCOLOR="<%= bgColor %>">
<H2 ALIGN="CENTER">Color de Fondo</H2>
```

La especificación JSP – Ejemplo 2: Scriplets JSP (continuación)

Fichero BGColor.jsp

```
<%
if (hasExplicitColor) {
  out.println(“Establecido el color de fondo:” +
              bgColor + “.”);
} else {
  out.println(“Usando los coleres de fondo por defecto.”);
}
%>

</BODY>
</HTML>
```

La especificación JSP – Declaraciones JSP.

- Una **declaración** JSP nos permite definir métodos o atributos que serán insertados dentro del cuerpo principal de la clase servlet (fuera del método service() que procesa la petición). Tienen la siguiente forma:

```
<%! Código Java%>
```

- Como las declaraciones no generan ninguna salida, normalmente se usan en conjunción con **expresiones JSP** o **Scriptlets.**

- **Notación XML:**

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

La especificación JSP – Ejemplo 3: Declaraciones JSP.

Fichero ContadorDeAccesos.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Ejemplo3: Declaraciones JSP</TITLE>
</HEAD>

<BODY>
<H1>Declaraciones JSP</H1>

<%! private int accessCount = 0; %>
<H2>Número de accesos al servidor desde que se inicio:
<%= ++accessCount %></H2>

</BODY>
</HTML>
```

La especificación JSP – Comentarios JSP

- **Comentarios ocultos JSP**: Estos comentarios no son visibles en la página generada y tienen la forma:

```
<%-- texto --%>
```

- **Comentarios visibles JSP**: Estos comentarios son visibles en la página generada, incluso se pueden mezclar con expresiones JSP:

```
<!-- texto [<%=expresion JSP%>] -->
```

Directivas JSP

La especificación JSP – Directivas JSP

- Una **directiva** JSP afecta a la estructura general de la clase servlet que resulta de la compilación de la JSP. Normalmente tienen la siguiente forma:

  ```
  <%@ directive atributo1="valor1" atributo2="valor2" ... atributoN="valorN" %>
  ```

- NOTACION XML:

  ```
  <jsp:directive.TIPO_DE_DIRECTIVA atributo1="valor1" atributo2="valor2" ...
      atributoN="valorN“ />
  ```

- Hay tres tipos de directivas:
  1. **page**, que nos permite hacer cosas como importar clases, personalizar la superclase del servlet, establecer el tipo de contenido, etc.
  2. **include**, que nos permite insertar un fichero dentro de la clase servlet en el momento que el fichero JSP es traducido a un servlet.
  3. **taglib**, que nos permite definir etiquetas personalizadas

La especificación JSP – Directiva page

- La directiva **page** dirige al motor Servlet´s sobre la configuración general de la página.
- Nos permite definir uno o mas de los siguientes atributos aplicables a toda la página:

```
<%@ page [import="{paquete.class | paquete.*},..."]
        [session="true|false"]
        [contentType="tipoMime"[;charset=Character-Set]" |
                "text/html;charset=ISO-8859-1"]
        [pageEncoding="characterSet | ISO-8859-1"]
        [errorPage="URL"]
        [isErrorPage="true|false"]
        [isThreadSafe="true|false"]
        [language="java"]
        [extends="paquete.class"]
        [buffer="none"|8kb|sizekb"]
        [autoFlush="true|false"]
        [info="texto"]
%>
```

La especificación JSP – Directiva page. Atributo import

- El atributo **import** nos permite especificar los paquetes que deberían ser importados. Si no se especifica nada en el servlet se importan los siguientes paquetes por defecto:
  - java.lang.*
  - javax.servlet.*
  - javax.servlet.jsp.*
  - javax.servlet.http.*
- Puede aparecer de una de las siguientes formas:

  <%@ page import="paquete.clase"%>

  <%@ page import="paquete.clase1, ..., paquete.claseN"%>
- EL atributo **import** es el único que puede aparecer más de una vez dentro de una JSP.

La especificación JSP – Ejemplo 1: Directiva page

Fichero AtributoImport.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Ejemplo1: Directiva page. Atributo import</TITLE>
</HEAD>

<BODY>
<H2>Ejemplo1: Directiva page. Atributo import </H2>
<%@ page import="java.util.*,servletsyjsps.*" %>

<%-- Declaraciones JSP --%>
<%!
private String randomID() {
  int num = (int)(Math.random()*10000000.0);
  return("id" + num);
}

private final String NO_VALUE = "<I>No Value</I>";
%>
```

La especificación JSP – Ejemplo 1: Directiva page (continuación)

Fichero AtributoImport.jsp

```
<%-- Scriptlet JSP --%>
<%
Cookie[] cookies = request.getCookies();
String oldID =
  ServletUtilities.getCookieValue(cookies, "userID", NO_VALUE);
String newID;
if (oldID.equals(NO_VALUE)) {
  newID = randomID();
} else {
  newID = oldID;
}
LongLivedCookie cookie = new LongLivedCookie("userID", newID);
response.addCookie(cookie);
%>

<%-- Expresiones JSP --%>
La última vez que se accedio a esta página fue <%= new Date() %> con un id de
usuario <%= oldID %>.
</BODY>
</HTML>
```

La especificación JSP – Directiva page. Atributo session

- El atributo **session** indica si la página JSP participa o no en una sesión HTTP.
- El atributo puede tomar los valores:

```
<%@page session="true|false" %>
```

  - Un valor **true** (valor por defecto) indica que la variable predefinida session (del tipo HttpSession) debería unirse a la sesión existente si existe una, y si no existe se debería crear una nueva sesión para unirla.
  - Un valor de **false** indica que no se usarán sesiones.

La especificación JSP – Directiva page. Atributo contentType

- El atributo **contentType** especifica el tipo MIME de la salida. El valor por defecto es **text/html** e **ISO-8859-1**.
- Puede aparecer de una de las siguientes formas:

```
<%page contentType="tipoMime"[;charset=Character-Set]" %>
```

- Algunos de tipos MIME más comunes son:

Type	Significado
text/html	Documento HTML
text/plain	Texto plano
text/css	Hoja de estilo
image/GIF	Imagen GIF
image/jpeg	Imagen JPEG
video/mpeg	Video clip MPEG
video/quicktime	Video clip QuickTime

Type	Significado
application/msword	Documento Word
application/pdf	Documento PDF
application/x-gzip	Archivo Gzip
application/ x-java-archive	Fichero JAR
application/zip	Archivo ZIP
application/ postscript	Fichero PostScript

La especificación JSP – Ejemplo 2: Directiva page

Fichero Excel.jsp

```
<%@ page contentType="application/vnd.ms-excel" %>
<%-- Hay que poner tabuladores entre los datos no espacios. --%>
1997    1998    1999    2000    2001 (Anticipated)
12.3    13.4    14.5    15.6    16.7
```

La especificación JSP – Directiva page. Atributo pageEncoding

- El atributo **pageEncoding** especifica el juego de caracteres que usará la página JSP para el response.

- El atributo puede aparecer de las formas:

  <%@ page pageEncoding="*characterSet* | ISO-8859-1" %>

- El valor por defecto es ISO-8859-1

La especificación JSP – Directiva page. Atributos errorPage, isErrorPage

- El atributo **errorPage** especifica una página JSP que se debería procesar si se lanzará cualquier Throwable pero no fuera capturado en la página actual.
  - Si el path empieza con una "/", el path es relativo al directorio raíz de documentos de la aplicación JSP y es resuelto por el servidor Web. Si no, el path es relativo al fichero JSP actual.

- El atributo **isErrorPage** indica si la página actual actúa o no como página de error de otra página JSP.

  El a tributo puede tomar los valores:

  <%@page isErrorPage="true|false" %>

  - Si es **true**, podemos usar el objeto **exception**, que contiene una referencia a la excepción lanzada, en el fichero JSP.
  - Si es **false** (valor por defecto), significa que no podemos usar el objeto **exception** en el fichero JSP.

La especificación JSP – Ejemplo 3: Directiva page

Fichero Velocidad.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Ejemplo 3: Directiva page</TITLE>
</HEAD>
<BODY>
<%@ page errorPage=“VelocidadError.jsp" %>
<%!
// Observa que se lanzará un  excepcion NumberFormatException si el valor es
nulo o
// esta mal formateado
private double toDouble(String value) {
  return(Double.valueOf(value).doubleValue());
}
%>
<%
double espacio = toDouble(request.getParameter(“espacion”));
double tiempo = toDouble(request.getParameter(“tiempo”));
double speed = espacio/tiempo;
%>
```

La especificación JSP – Ejemplo 3: Directiva page (continuación)

Fichero Velocidad.jsp

```
<UL>
  <LI>Espacio: <%= furlongs %>.
  <LI>Tiempo: <%= fortnights %>.
  <LI>Velocidad: <%= speed %>.
</UL>

</BODY>
</HTML>
```

La especificación JSP – Ejemplo 3: Directiva page (continuación)

Fichero VelocidadError.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Pagina de error</TITLE>
</HEAD>

<BODY>

<%@ page isErrorPage="true" %>

<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      Error al calcular la velocidad</TABLE>
<P>
Hay un error en la página Velocidad.jsp:
<I><%= exception %></I>.
<PRE><% exception.printStackTrace(new java.io.PrintWriter(out)); %></PRE>

</BODY>
</HTML>
```

La especificación JSP – Directiva page. Atributo extends

- El atributo **extends** indica el nombre, totalmente cualificado, de la superclase del servlet que resulta de compilar la página JSP.

```
<%@ page extends="paquete.class" %>
```

- Sun recomienda que usemos este atributo con cautela, ya puede limitar la habilidad del motor del JSP.

La especificación JSP – Directiva include

- La directiva **include** nos permite incluir ficheros en el momento en que la página JSP es compilada a un servlet. La directiva include tiene la forma:

  <%@ include file="relativeURL" %>

  Observar que la directiva **include** incluye el fichero en el momento de la compilación. Por tanto, si la página incluida cambia después de la compilación, los cambios no se verán reflejados en la página actual hasta que se vuelva a compilar.

La especificación JSP – Ejemplo 1: Directiva include

Fichero Velocidad.jsp

```
<html>
  <head>
    <title>Ejemplo de un include</title>
  </head>
  <body bgcolor="white">
    <font color="blue">
      Fecha y hora actual: <%@ include file="date.jsp" %>
    </font>
  </body>
</html>
```

Fichero date.jsp

```
<%@ page import="java.util.*" %>
<%= (new Date() ).toLocaleString() %>
```

La especificación JSP – Directiva taglib

- Gracias a la directiva **taglib** podemos definir nuestros propios tags JSP.
- A un conjunto de tags JSP lo llamaremos **librería de tags.**
- Para definir un tag necesitamos definir 4 componentes:
  - Una clase que defina el comportamiento del tag.
  - Un fichero TLD (Tag library descriptor) para hacer visible la clase en el servidor.
  - Un fichero web.xml para hacer visible el tag en el servidor.
  - Un fichero JSP que use el tag

Objetos Implícitos

La especificación JSP – Objetos implícitos en una JSP

- Para simplificar el código en las expresiones y scriplets JSP, tenemos 9 variables definidas automáticamente:
    1. **request.**
    2. **response.**
    3. **out.**
    4. **session.**
    5. **application.**
    6. **config.**
    7. **pageContext.**
    8. **page.**
    9. **exception**
- Estas variables reciben el nombre de Objetos implícitos en una JSP.

La especificación JSP – Objetos implícitos en una JSP

1. request:
   Este es el objeto de la clase **HttpServletRequest** asociado con la petición, y nos permite mirar los parámetros de la petición (mediante el método **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, aunque esto casi nunca se lleva a la práctica.

2. response:
   Este es el objeto de la clase **HttpServletResponse** asociado con la respuesta al cliente. Observa que, como el stream de salida (ver out más abajo) 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.

La especificación JSP – Objetos implícitos en una JSP

3. **out**:

   Este es el **PrintWriter** usado para envíar la salida al cliente. Sin embargo, para poder hacer útil el objeto **response** (ver punto anterior), esta 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**. **out** 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**.

4. **session**:

   Este es el objeto de la clase **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. Como veremos, 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.

La especificación JSP – Objetos implícitos en una JSP

5. application:
   Es el objeto de la clase **ServletContext** obtenido mediante:
   `getServletConfig().getContext()`
6. config:
   Es el objeto de la clase **ServletConfig** para la página actual.

7. pageContext:
   Es el objeto de la clase **PageContext**. Este objeto es usado por el motor de Servlet´s para administrar características como páginas de error y los parámetros para hacer include o forward de páginas.

8. Exception:
   Es un objeto de la clase **Throwable**. Solo se crea en el caso en el que usemos la directiva:
   `<%@page ErrorPage="true"%>`

Práctica

- Práctica 6.9
  - Java Server Pages