Introducción a JSP
Una página JSP se compone de código HTML más:

- Directivas
- Declaraciones
- Expresiones
- Scriptlets

Estos cuatro elementos pueden emplear dos tipos de sintaxis diferentes: JSP estándar o etiquetas XML. Ambas opciones no deben mezclarse en la misma página. Nosotros emplearemos la sintaxis JSP estándar.

Directivas
Aparecen al principio de la página. Contienen información que debe tener en cuenta el contenedor de JSP/Servlet del servidor para procesar la página. Sintaxis:

Estándar JSP: <%@ directiva atributo=valor [atributo=valor] ...%>
XML: <jsp:directive.directiva atributo=valor [atributo=valor] .../>

Ejemplos:
<%@ page errorPage="/paginaError.jsp" %>
<%@ page import="java.util.*"%>
<%@ include file="/WEB-INF/inicio.jsp"%>
<jsp:directive.include file=”/web-inf/proceso.jsp”/>

Declaraciones
Permiten declarar propiedades y métodos. Estas propiedades y métodos serán visibles a lo largo de todo el código de la página. Una declaración supone que la propiedad declarada se convierte en propiedad de la clase Java que dará lugar tras la conversión de la página JSP a servlet por parte del servidor de aplicaciones. Sintaxis:

Estándar JSP: <%! declaración ;%>
XML: <jsp:declaration > declaración </jsp:declaration>

Ejemplos:
```
<%! String nombre=”ninguno” ;%>
<%! public void setNombre (String nom) {
          nombre = nom;
       } %>
<jsp:declaration>
  public String getNombre() {
       return nombre;
  }
</jsp:declaration>
```

Expresiones
Son fragmentos de código Java que se evaluarán antes de crear el código fuente el contenedor web, y que formará parte de la salida generada por la página JSP, dentro del método *_jspService()*. Sintaxis:

Estándar JSP: <%= expresión %>
XML: <jsp:expression> expresión </jsp:expression>
Ejemplo:
<%= new Date() %>

Scriptlets

Son fragmentos de código Java que formarán parte del método *_jspService()* del servlet generado por el contenedor web. Pueden emplearse en cualquier punto de la página. Sintaxis:

Estándar JSP: <% scriptlet %>
XML: <jsp:scriptlet> scriptlet </jsp:scriptlet>

Ejemplos:

```
<%
    out.println("<P><B>Su nombre es:</B>");
    String nombre = request.getParameter("nombreusuario");

    out.print("<P>Nombre: "+nombre);
%>
<jsp:expression>
   StringBuffer sb = new StringBuffer();
   sb.append (“Hola, “);
   sb.append (request.getParameter(“nombre”));
   out.println (sb.toString());
</jsp:expression>
```

Comentarios
Los comentarios en JSP siguen la sintaxis:

```
<!-- comentario -->
```

Con estas nociones de la sintaxis propia de una página JSP ya disponemos de la base para comenzar a diseñar páginas JSP.

El fotograma perdido, versión 1
Partiendo del enunciado base, se estructura en tres páginas. La URL será *http://<servidor>:<puerto>/fotogramaperdidov1/index.html*. La página index.html sólo se emplea como página de bienvenida, y mediante un botón se pasa a verfotograma.jsp. En verfotograma.jsp se carga la imagen del fotograma mediante un archivo almacenado en un directorio denominado fotogramas, haciendo uso de Java. Se muestra junto con la imagen un botón para que pueda verse la respuesta. Al pulsar el botón, se salta a la página respuesta.jsp, que muestra el título de la película a la que pertenece el fotograma de forma estática.

En este enunciado se plantea por primera vez el diseño de una página JSP con sus propias etiquetas, así como la posibilidad de que la aplicación acceda a un archivo ubicado en el disco local del servidor de aplicaciones. Se comenzará por crear un proyecto con los siguientes datos básicos:

```
Nombre del proyecto: FotogramaPerdidoV1
Servidor destino del proyecto: JBoss 7
Contexto raíz: fotogramaperdidov1
Directorio de contenido web: web
Se debe marcar la opción de generar web.xml
```

Tras crear el proyecto, debe añadirse al directorio *web/WEB-INF* el archivo *jboss-web.xml*, que requiere JBoss para poder desplegar la aplicación con el contexto raíz establecido. Para ello, teniendo seleccionado ese directorio, se accede al menú contextual, se escoge la opción *New->Other->XML->XML File*. Aparecerá un cuadro de diálogo para introducir el nombre del archivo y su ubicación. Hecho esto, se pulsa *Next*, y en el siguiente cuadro de diálogo se deja marcada la tercera opción (*Create XML file from a XML template*). Se pulsa *Next* y se dejan las opciones por defecto, y finalmente, *Finish*.

Tras esto, se abre el archivo *jboss-web.xml* y se le añade el siguiente código:

```
<jboss-web>
      <context-root>fotogramav1</context-root>
</jboss-web>
```

La etiqueta <context-root> establece el contexto raíz de la aplicación para que lo aplique correctamente el servidor JBoss.

Una vez guardados los cambios, se procede a la creación de las tres páginas que se almacenarán en el directorio *web*, y que serán las que marquen la navegación por la aplicación. Comenzamos por crear el archivo *index.html* ubicándolo en ese directorio, siguiendo el procedimiento anterior, pero esta vez escogiendo el tipo HTML. Una vez creado, se abre y se añade el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>El Fotograma Perdido - Versión: 1</title>
</head>
<body>
      <center>
          <h2>El Fotograma Perdido</h2><br><br>
          Pulse en el siguiente botón para visualizar el fotograma.<br><br>
          <form name="VerFotograma" action="verfotograma.jsp">
              <input type="submit" value="Ver" name="Ver" />
          </form>
      </center>
</body>
</html>
```

Como se observa, la página se limita a mostrar un texto y la indicación al usuario para que pulse el botón *Ver* para visualizar el fotograma.

Seguidamente se crea el archivo *verfotograma.jsp*, esta vez, de tipo JSP. Se ubica también en el directorio web de la aplicación y se escogen las opciones por defecto al crearlo. Luego, se abre y se añade el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>El Fotograma Perdido - Versión 1</title>
</head>
<body>
<center>
      <h2>¿Sabrías decir a qué película corresponde el siguiente fotograma?
</h2><br><br>
      <% fotogramas.util.CargaFotograma cargaFotograma; %>
      <% cargaFotograma = new fotogramas.util.CargaFotograma();
         String nombreFotograma = cargaFotograma.leeFotograma();
         //Si falla lo anterior, probar:
         //out.write("<img src=\"fotogramas/"+nombreFotograma+"\"
```

```
alt=\"Fotograma 1\"/>");
      %>
      <img src="fotogramas/<%=nombreFotograma%>" alt="No se encuentra el
archivo"/>
      <form name="Respuesta" action="respuesta.jsp"><br><br>
          Pulsa en el botón siguiente para ver la respuesta<br><br>
          <input type="submit" value="Respuesta" name="Respuesta" />
      </form>
      </center>
</body>
</html>
```

Procedemos a analizar ahora este código. Cuando el usuario pulsa el botón Ver de la página index.html, se solicita la página verfotograma.jsp. Como ya se explicó en párrafos anteriores, esta página es convertida a servlet si es la primera vez que se solicita desde que se instaló la aplicación, en caso contrario, se crea una instancia del servlet si aún no existía. El método _jspService() será el encargado de ejecutar el proceso y su código será el desarrollado en la página JSP. Se ejecuta en el mismo orden en que se escribió el código, de arriba hacia abajo. Por tanto, lo primero que hace el servlet es mostrar al usuario el texto "¿Sabrías decir a qué película pertenece el siguiente fotograma?". Las siguientes líneas se comentan a continuación.

```
<% fotogramas.util.CargaFotograma cargaFotograma; %>
```

Se declara un objeto de la clase *CargaFotograma* en un scriptlet, lo que significa que esa declaración se ubica dentro del propio método *_jspService()*. La clase *CargaFotograma*, que aún no se ha diseñado, tiene como funcionalidad recuperar el nombre del archivo que contiene el fotograma, y pertenece al paquete *fotogramas.util*, que tampoco se ha creado aún.

```
cargaFotograma = new fotogramas.util.CargaFotograma();
```

Ahora se crea un nuevo scriptlet, en el que lo primero que se hace es crear un objeto de la clase cargaFotograma.

```
String nombreFotograma = cargaFotograma.leeFotograma();
```

Seguidamente, se llama al método *leeFotograma()* del objeto creado, que devolverá el nombre del archivo de imagen que contiene el fotograma a visualizar y que se almacenará en la variable de tipo *String nombreFotograma*. Y finaliza el scriptlet.

```
<img src="fotogramas/<%=nombreFotograma%>" alt="No se encuentra el archivo"/>
```

Ahora se muestra una imagen, recuperada mediante una etiqueta <img> de HTML, empleando una expresión JSP que al evaluarse, hará que el servlet la sustituya por el valor de la variable *nombreFotograma*.

```
<form name="Respuesta" action="respuesta.jsp"><br><br>
          Pulsa en el botón siguiente para ver la respuesta<br><br>
          <input type="submit" value="Respuesta" name="Respuesta" />
</form>
```

Tras mostrar la imagen, se ofrece un botón para obtener el título de la película a la que pertenece el fotograma, haciendo una petición sobre la página *respuesta.jsp*.

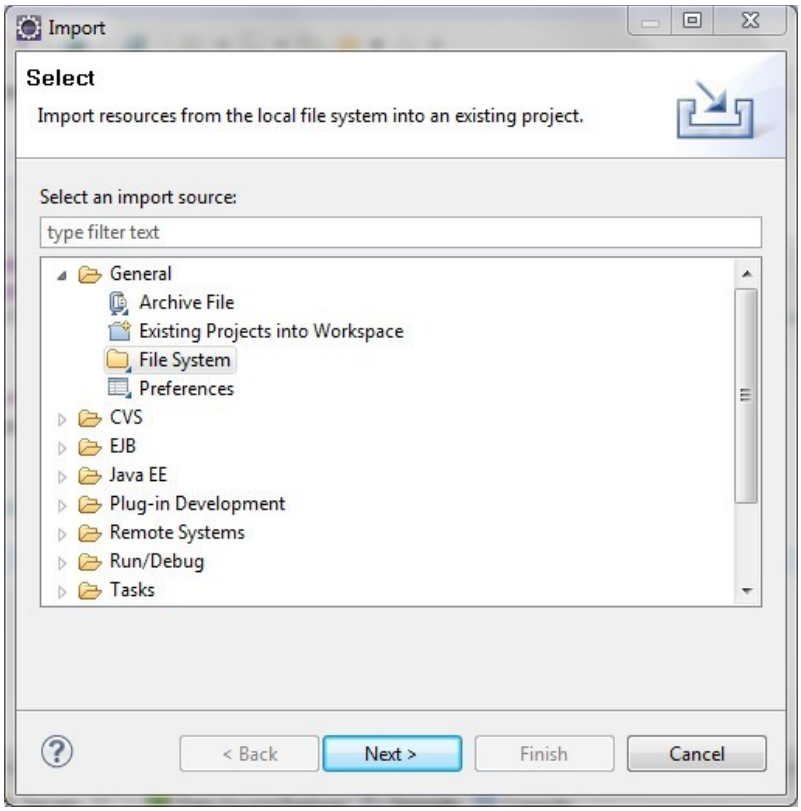
Se procede ahora con la creación de la página *respuesta.jsp*, en el directorio web de la aplicación, abriendo luego el archivo y añadiendo las siguientes líneas:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>El Fotograma Perdido - Versión 1</title>
</head>
<body>
      <center> <h2>2001: Una odisea en el espacio</h2> </center>
</body>
</html>
```

Como se observa, esta página, a pesar de ser de tipo JSP, sólo posee código HTML, y ofrece, de manera estática, el título de la película.

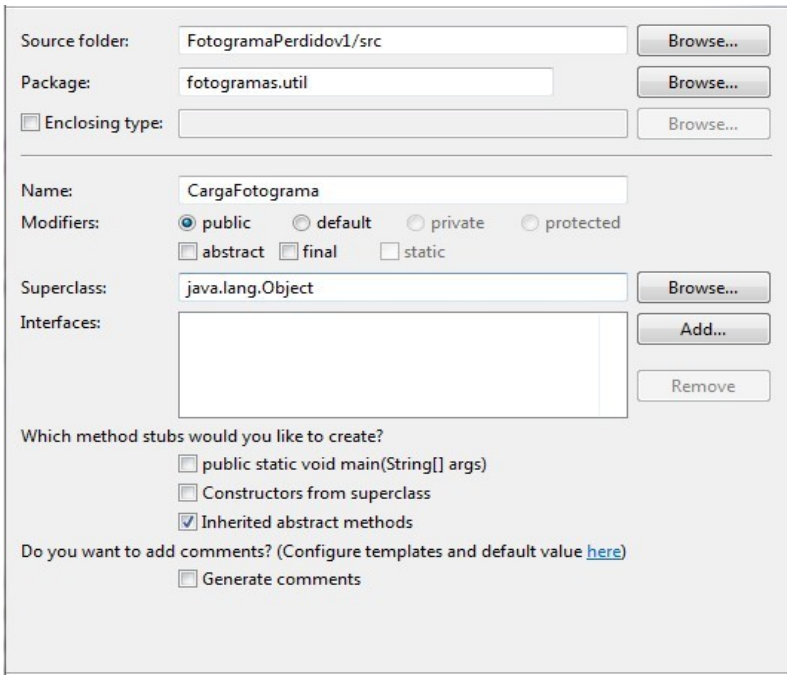
Una vez creadas las páginas web, tanto HTML como JSP, se procede a crear el directorio *fotogramas*, que contendrá las imágenes. En principio sólo habrá una, *f1.jpg*. Se crea el directorio dentro de *web* mediante *New->Folder*. Una vez creado, se selecciona el directorio, se accede de nuevo al menú contextual, y se selecciona la opción *Import*. Aparecerá un cuadro de diálogo en el que habrá que seleccionar *File System*, que permitirá recorrer el sistema de directorios local y poder escoger los archivos que se importarán.



Al pulsar *Next*, se deberá seleccionar a través del botón *Browse* el directorio donde se ubica el archivo *f1.jpg* (puede cargarse cualquier otro archivo de imagen que tenga el alumno en su equipo y que se corresponda con una película, si se trata de una película diferente a la mostrada en el ejemplo, también deberá modificar el título que aparece como respuesta en *respuesta.jsp*). Tras abrir el directorio, se muestra una zona, a la derecha del cuadro de diálogo, en el que se podrá marcar el archivo. Más abajo, en *Into folder*, se especifica la carpeta donde se copiará el archivo (en nuestro caso en *FotogramaPerdidov1/web/fotogramas*). Se pulsa *Finish* y tendremos el archivo *f1.jpg* en *fotogramas*.

Ahora se crea el paquete *fotogramas.util*. Se selecciona el proyecto y se escoge la opción *New->Package*. El nombre será *fotogramas.util* y por defecto se almacena en *Java Resources: src*. Se pulsa *Finish* y aparecerá allí.

Al paquete *fotogramas.util* se le añade la clase *CargaFotograma*. Se hace seleccionando el paquete y accediendo al menú contextual, escogiendo la opción *New->Class*. Se rellena tal como se muestra en la imagen siguiente:



Y se pulsa el botón *Finish*. La clase se habrá creado y llega el momento de añadirle el código siguiente:

```
package fotogramas.util;

import java.io.*;

public class CargaFotograma {

    private String archivoFotogramas;

    public CargaFotograma() {
        archivoFotogramas = "/home/alumno/fotogramas.txt";
    }

    public String leeFotograma() {
        String fotograma;
        try
        {
            // Crear un objeto File que referenciará a un archivo
            File archivo = new File(archivoFotogramas);
            // Se crea un flujo orientado a leer caracteres desde un archivo
            FileReader fr = new FileReader(archivo);
            // Se encadena con un flujo orientado a cadenas de caracteres
            BufferedReader br = new BufferedReader (fr);
            // Se lee una cadena del archivo, que termina en un fin de línea.
            fotograma = br.readLine();

        }
        catch(FileNotFoundException fnfe)
        {
            // Excepción que lanza FileReader cuando no encuentra el archivo
            fotograma = "No se ha encontrado el archivo";
        }
        catch (IOException ioe)
        {
```

```
         // Excepción que lanza el método readLine() cuando tiene problemas para leer de la fuente
de datos
            fotograma =  "Error al leer del archivo.";
        }
        catch (Exception exp) {
            fotograma = "No encontrado";
        }
        return fotograma;
    }

}
```

Como se observa en el código, esta clase establece en su constructor el nombre del archivo del que se va a recuperar la imagen del fotograma. En principio se ha colocado en el directorio raíz de la cuenta del usuario *alumno*, pero podía haberse ubicado en cualquier otro. En proyectos posteriores se colocarán en *WEB-INF*, pero el acceso empleará otro mecanismo de codificación.

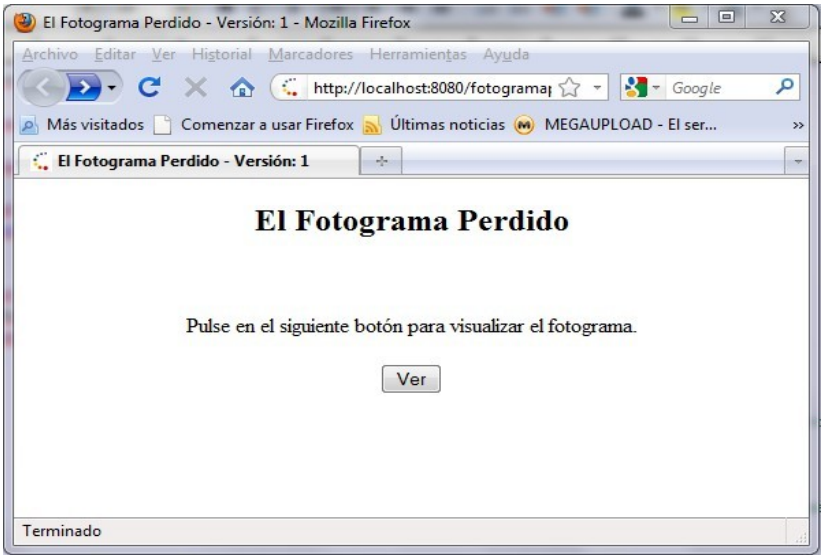
El método *leeFotograma()* se limita a establecer un flujo con el archivo de texto mediante la clase *BufferedReader*, que permite leer líneas de texto. El valor devuelto será el nombre del archivo o bien un mensaje de error.

Por tanto, después de grabar los cambios de la clase *CargaFotograma*, se debe crear el archivo *fotogramas.txt* en el directorio raíz de la cuenta *alumno* conteniendo el nombre completo del archivo de imagen, que en nuestro caso es *f1.jpg*.

Ahora se arranca el servidor *JBoss* si no estaba arrancado, comprobando antes que no hay ningún otro servidor atendiendo por el puerto de *JBoss*. Además, comprobamos que el navegador configurado sea el navegador por defecto del sistema (*Window->Web Browser->Default system web browser*).

Se ejecuta la aplicación con *Run As->Run on Server*, y se prueba.

He aquí el resultado.



Seguidamente se muestra el código generado por el servidor de aplicaciones correspondiente a los servlets asociados a las páginas JSP.

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class verfotograma_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
```

```
  private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

  private static java.util.List _jspx_dependants;

  private javax.el.ExpressionFactory _el_expressionfactory;
  private org.apache.InstanceManager _jsp_instancemanager;

  public Object getDependants() {
    return _jspx_dependants;
  }

  public void _jspInit() {
    _el_expressionfactory =
_jspxFactory.getJspApplicationContext(getServletConfig().getServletContext()).getExpressionFactory()
;
    _jsp_instancemanager =
org.apache.jasper.runtime.InstanceManagerFactory.getInstanceManager(getServletConfig());
  }

  public void _jspDestroy() {
  }

  public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;


    try {
      response.setContentType("text/html; charset=UTF-8");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                  null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;

      out.write("\n");
      out.write("\n");
      out.write("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\" \"http://www.w3.org/TR/html4/loose.dtd\">\n");
      out.write("<html>\n");
      out.write("<head>\n");
      out.write("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
      out.write("<title>El Fotograma Perdido - Versión 1</title>\n");
      out.write("</head>\n");
      out.write("<body>\n");
      out.write("<center>\n");
      out.write("\t<h2>¿Sabrías decir a qué película corresponde el siguiente fotograma?
</h2><br><br>\n");
      out.write("\t");
 fotogramas.util.CargaFotograma cargaFotograma;
      out.write('\n');
      out.write('      ');
 cargaFotograma = new fotogramas.util.CargaFotograma();
          String nombreFotograma = cargaFotograma.leeFotograma();
          //Si falla lo anterior, probar:
          //out.write("<img src=\"fotogramas/"+nombreFotograma+"\" alt=\"Fotograma 1\"/>");

      out.write("\n");
      out.write("\t<img src=\"fotogramas/");
      out.print(nombreFotograma);
```

```
      out.write("\" alt=\"No se encuentra el archivo\"/>\n");
      out.write("\t<form name=\"Respuesta\" action=\"respuesta.jsp\"><br><br>\n");
      out.write("\t    Pulsa en el botón siguiente para ver la respuesta<br><br>\n");
      out.write("\t    <input type=\"submit\" value=\"Respuesta\" name=\"Respuesta\" />\n");
      out.write("\t</form>\n");
      out.write("\t</center>\n");
      out.write("</body>\n");
      out.write("</html>");
    } catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          try { out.clearBuffer(); } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class respuesta_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

  private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

  private static java.util.List _jspx_dependants;

  private javax.el.ExpressionFactory _el_expressionfactory;
  private org.apache.InstanceManager _jsp_instancemanager;

  public Object getDependants() {
    return _jspx_dependants;
  }

  public void _jspInit() {
    _el_expressionfactory =
_jspxFactory.getJspApplicationContext(getServletConfig().getServletContext()).getExpressionFactory()
;
    _jsp_instancemanager =
org.apache.jasper.runtime.InstanceManagerFactory.getInstanceManager(getServletConfig());
  }

  public void _jspDestroy() {
  }

  public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;


    try {
      response.setContentType("text/html; charset=UTF-8");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                  null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
```

```
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;

      out.write("\n");
      out.write("\n");
      out.write("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\" \"http://www.w3.org/TR/html4/loose.dtd\">\n");
      out.write("<html>\n");
      out.write("<head>\n");
      out.write("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
      out.write("<title>El Fotograma Perdido - Versión 1</title>\n");
      out.write("</head>\n");
      out.write("<body>\n");
      out.write("\t<center> <h2>2001: Una odisea en el espacio</h2> </center>\n");
      out.write("</body>\n");
      out.write("</html>");
    } catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          try { out.clearBuffer(); } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

El código fuente de la página enviada al usuario al solicitar *verfotograma.jsp* es el siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>El Fotograma Perdido - Versión 1</title>
</head>
<body>
<center>
        <h2>¿Sabrías decir a qué película corresponde el siguiente fotograma?</h2><br><br>


        <img src="fotogramas/f1.jpg" alt="No se encuentra el archivo"/>
        <form name="Respuesta" action="respuesta.jsp"><br><br>


            Pulsa en el botón siguiente para ver la respuesta<br><br>
            <input type="submit" value="Respuesta" name="Respuesta" />
        </form>
        </center>
</body>
</html>
```

Llegados a este punto, y antes de abordar las versiones posteriores del fotograma perdido, se hará un recordatorio de un grupo de conceptos que se trataron en el curso anterior, y que son imprescindibles para acometer lo que aún queda por desarrollar.