Aplicaciones Web con Servlets y JSP (II)

Jesús Arias Fisteus

Computación Web (2017/18)

uc3m | Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Parte I

Java Server Pages (JSP)

Introducción

- Un servlet no resulta adecuado para presentación (escribir directamente la salida HTML) porque el código HTML está entremezclado dentro del código Java:
  - Incómodo cambiar el código HTML.
  - No facilita la división de tareas entre diseñadores de HTML y programadores.
- Alternativa: escribir directamente el código HTML con pequeños fragmentos de código incrustados.

Introducción

- Una página JSP es una página HTML que incorpora ciertos elementos dinámicos: etiquetas especiales y pequeños fragmentos de código.
  - El código HTML aparece a la salida sin modificaciones.
  - Los elementos dinámicos se evalúan o ejecutan en el servidor en el momento de construcción de la respuesta.

JSP ejemplo

```
1  <%@ page language='java'
2           contentType='text/html;charset=utf-8'%>
3  <%@ page import='java.util.Date' %>
4  <!DOCTYPE html>
5  <html>
6    <head>
7      <title>Hola Mundo</title>
8    </head>
9    <body>
10     <p>Hola, esto es una página JSP.</p>
11     <p>
12       La hora del servidor es <%= new Date() %>
13     </p>
14   </body>
15 </html>
```

JSP ejemplo: documento recibido por el cliente

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Hola Mundo</title>
5    </head>
6    <body>
7      <p>Hola, esto es una página JSP.</p>
8      <p>
9        La hora del servidor es Wed Nov 06 13:25:34 CET 2002
10     </p>
11   </body>
12 </html>
```

JSP como servlet

- Aunque no es estrictamente obligatorio, los contenedores de servlets suelen transformar las páginas JSP en el código fuente de servlets, que después se compilan y ejecutan.

JSP ejemplo transformado en un servlet

```
1  (...)
2  out.write("<!DOCTYPE␣html>\n␣");
3  out.write("<html>\n␣");
4  out.write("␣␣<head>\n␣");
5  out.write("␣␣␣␣<title>Hola␣Mundo</title>\n␣");
6  out.write("␣␣</head>\n␣");
7  out.write("␣␣<body>\n␣␣␣␣<p>Hola,␣esto␣es␣una␣página␣JSP.</p
       >\n␣");
8  out.write("␣␣␣␣<p>La␣hora␣del␣servidor␣es␣");
9  out.print( new Date() );
10 out.write("</p>\n␣");
11 out.write("␣␣</body>\n");
12 out.write("</html>\n");
13 (...)
```

Comentarios en JSP

- Los comentarios JSP no se envían al cliente, al contrario que los comentarios HTML.

```
1 <%-- Esto es un comentario JSP --%>
```

Directivas JSP: *page*

- Todas las páginas JSP deberían incluirla. Atributos habituales:
  - `language`: lenguaje de programación (`java` por defecto).
  - `contenttype`: tipo de contenido de la página (`text/html` por defecto).
  - `isErrorPage`: indica si es una página de error (`false` por defecto).
  - `errorPage`: página a la que dirigirse si ocurre una excepción procesando esta página.

```
1 <%@ page language='java' contentType='text/html'
2          isErrorPage='false' errorPage='error.jsp' %>
```

Otras directivas JSP

- `include`: permite incluir directamente el código de otro fichero en el punto en que aparezca la directiva.
- `import`: permite importar clases Java utilizadas en la página JSP.

```
1 <%@ include file='footer.html' %>
2 <%@ page import='java.util.*' %>
```

Scriptlets

- Permiten incrustar código escrito en otro lenguaje de programación (normalmente Java):
  - <%= `expresión` %>: evalúa la expresión, convierte el resultado a `String` y lo muestra en la página.
  - <% `sentencias` %>: ejecuta las sentencias, sin mostrar nada en la página.

```
1 <%-- los siguientes scriptlets son equivalentes --%>
2 <%= user.getName() %>
3 <% out.println(user.getName()); %>
```

Scriptlets: ejemplo

```
1  <table>
2    <tr><th>Product</th><th>Price</th></tr>
3  <%
4    for (Product product: catalog) {
5  %>
6    <tr>
7      <td>
8        <a href='<%= response.encodeURL("view?id="
9                                        + product.getId()) %>'>
10         <%= product.getShortName() %>
11       </a>
12     </td>
13     <td><%= product.getPrice() %> Euros</td>
14   </tr>
15 <% } %>
16 </table>
```

Variables implícitas

- JSP proporciona automáticamente una serie de variables ya declaradas e inicializadas, directamente utilizables por el programador:
  - `application` (`javax.servlet.ServletContext`)
  - `config` (`javax.servlet.ServletConfig`)
  - `out` (`javax.servlet.jsp.JspWriter`)
  - `pageContext` (`javax.servlet.jsp.PageContext`)
  - `request` (`javax.servlet.http.HttpServletRequest`)
  - `response` (`javax.servlet.http.HttpServletResponse`)
  - `session` (`javax.servlet.http.HttpSession`)
- Además, en páginas marcadas con `isErrorPage`:
  - `exception` (`java.lang.Throwable`)

Acciones: jsp:include

- La acción jsp:include invoca al servlet o JSP dado e incluye el resultado de su ejecución en el punto actual del documento JSP desde el cual se incluya.
- El control retorna finalmente a la página inicial.
- Opcionalmente, se pueden pasar parámetros.

```
1 <jsp:include page='header.jsp'>
2   <jsp:param name='title' value='Welcome' />
3 </jsp:include>
```

Java Beans

- Un Java Bean es una clase Java que cumple el siguiente convenio:
  - Contiene *propiedades* (normalmente atributos de instancia privados).
  - El acceso a las propiedades se realiza mediante métodos de acceso *get*, *set* e *is*.
  - Tiene siempre un constructor sin argumentos (aunque podría tener más constructores).

Java Beans: ejemplo

```
1  package foo;
2
3  public class UserInfo implements java.io.Serializable
4  {
5    private String firstName;
6    private boolean registered;
7
8    public String getFirstName() {
9      return firstName;
10   }
11   public void setFirstName(String firstName) {
12     this.firstName = firstName;
13   }
14   public boolean isRegistered() {
15     return registered;
16   }
17   public void setRegistered(boolean registered) {
18     this.registered = registered;
19   }
20 }
```

Java Beans en JSP

- JSP proporciona instrucciones especiales para trabajar más cómodamente con Java Beans.
- La acción `jsp:useBean`:
  - Si el bean aún no existe en el contexto:
    - Declara, crea e inicializa el bean.
    - Crea una referencia al bean con el nombre dado por `id`.
  - Si el bean ya existe en el contexto:
    - Obtiene una referencia al mismo con el nombre dado por `id`.

```
1 <jsp:useBean id='user' class='foo.UserInfo' scope='session'>
2   <jsp:setProperty property='firstName' value='Pepe' />
3 </jsp:useBean>
```

Java Beans en JSP: contextos

- Un bean en JSP se puede almacenar en uno de los siguientes contextos:
  - `page`: asociado a la página JSP y a una petición HTTP concreta, desaparece al finalizar el procesado de la página.
  - `request`: asociado a una petición HTTP concreta, compartida entre todos los JSPs y servlets que atiendan dicha petición, desaparece al finalizar el procesado de la petición.
  - `session`: asociado a la sesión, compartida por todos los JSPs y servlets para todas las peticiones de una misma sesión, desaparece al finalizar la sesión.
  - `application`: asociado al contexto de la aplicación Web, compartido por todos los servlets y JSPs de la aplicación en todas las peticiones.

La acción jsp:getProperty

- Se evalúa al valor de una propiedad de un bean.

```
1 <jsp:getProperty name='user' property='firstName' />
```

La acción jsp:setProperty

- Establece el valor de una propiedad de un bean.
- Convierte, si es necesario, el valor de la propiedad desde una cadena de texto al tipo de datos correspondiente.
- Proporciona un atajo para establecer valores de propiedades a partir de los parámetros de la petición, si ambos tienen el mismo nombre.

```
1 <jsp:setProperty name='user' property='firstName'
2    value='<%= request.getParameter("firstName") %>' />
3
4 <%-- si 'firstName' es parámetro de la petición --%>
5 <jsp:setProperty name='user' property='firstName' />
6
7 <%-- todos los parámetros de la petición cuyo nombre
8      coincida con propiedades --%>
9 <jsp:setProperty name='user' property='*' />
```

Algunas referencias

- Marty Hall, *Core Servlets and JavaServer Pages*. Prentice Hall (2000).
- `http://www.oracle.com/technetwork/java/javaee/jsp/`

Parte II

Arquitectura de una aplicación con Servlets y JSP

Patrón Modelo Vista Controlador (MVC)

- Patrón de arquitectura del *software* que divide la aplicación en tres partes:
  - Modelo: almacenamiento y manipulación de los datos de la aplicación.
  - Vista: presentación del modelo que permita al usuario interaccionar con él (interfaz de usuario).
  - Controlador: recibe la entrada del usuario, solicita acciones al modelo y controla la vista.

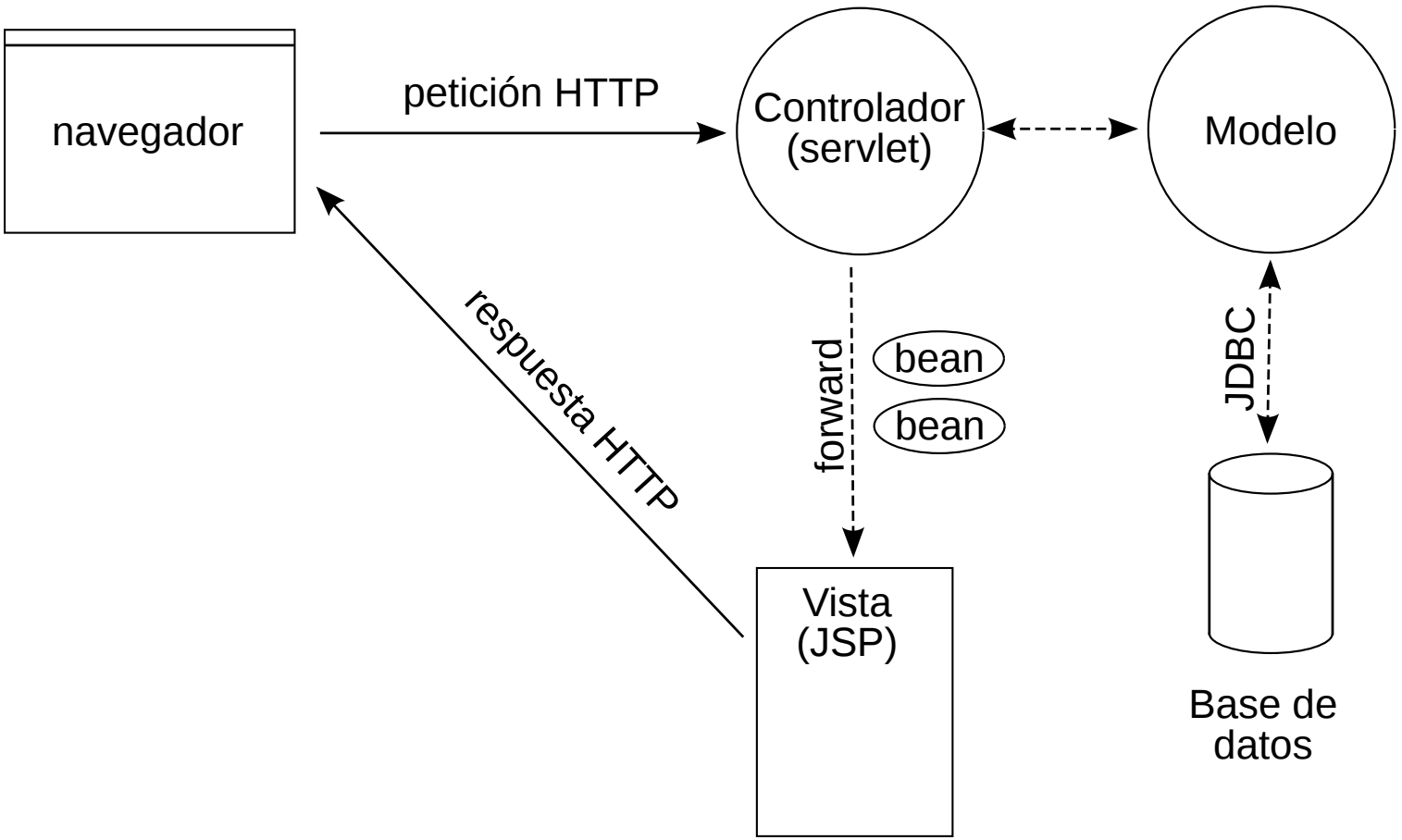
Aplicaciones web Java y MVC

- Patrón MVC en una aplicación web Java:
  - Modelo: gestiona el almacenamiento, recuperación y manipulación de los datos de la aplicación (típicamente mediante base de datos).
  - Vista: produce la salida, normalmente HTML, que se envía al usuario (típicamente JSP).
  - Controlador: recibe las peticiones HTTP, actúa sobre el modelo y controla el funcionamiento de la vista (típicamente servlets).

Modelo de funcionamiento (I)

1. El cliente envía la petición HTTP a un controlador (servlet).
2. El controlador procesa la petición.
   - Si es necesario, interacciona con el modelo.
3. El controlador redirige la petición a una vista (JSP).
   - Si es necesario, añade *beans* como parámetros.
4. La vista lee los parámetros y construye la respuesta HTTP que se envía al cliente.

Modelo de funcionamiento (II)



Mecanismos de redirección de peticiones

- Hay dos formas de redirigir una petición a otro recurso:
  - Redirecciones HTTP (*sendRedirect*):
    - El servidor envía una respuesta al cliente con un código 3xx y la URI a la que este debe enviar la petición.
    - Dos peticiones HTTP: la original y la dirigida a la nueva URI.
  - Redirecciones internas en el servidor (*forward*):
    - Se redirige la petición de un recurso a otro internamente, dentro de la misma aplicación Web.
    - El recurso de la última redirección devuelve al cliente la respuesta HTTP.
    - Una única petición HTTP.

Redirecciones *sendRedirect*

- Fuerza el envío de una respuesta HTTP de redirección al cliente.
- El cliente enviará una nueva petición a la URI recibida en esta respuesta.
- El control retorna al finalizar el método *sendRedirect*, por lo que conviene que sea lo último que se ejecuta.

```
1 // Redirección con URI absoluta
2 response.sendRedirect("http://www.ejemplo.es/");
3
4 // Redirección con URI relativa a la de la petición actual
5 response.sendRedirect("otra.html");
```

Redirecciones *forward*

- Un Servlet o JSP reenvía la petición a otro recurso (Servlet, JSP, HTML) de la misma aplicación Web.
- El cliente no se entera de la redirección (p.e., el navegador muestra la URI original de la petición, no la redirigida).
- El control retorna al finalizar el método *forward*, por lo que conviene que sea lo último que se ejecuta.

Redirecciones *forward*

- *Forward* desde un Servlet:

```
1 RequestDispatcher rd =
2     request.getRequestDispatcher("vista.jsp");
3 rd.forward(request, response);
```

Redirecciones *forward* con parámetros

- El objeto de la petición (*ServletRequest*) de los recursos origen y destino de la redirección es el mismo:
  - Se pueden añadir parámetros como atributos a la petición.

```
1 Noticia nuevaNoticia = (...)
2 request.setAttribute("noticia", nuevaNoticia);
3 RequestDispatcher rd =
4     request.getRequestDispatcher("vista.jsp");
5 rd.forward(request, response);
```

Redirecciones *forward* con parámetros

- Recogida de parámetros desde un Servlet:

```
1 Noticia noticia =
2     (Noticia) request.getAttribute("noticia");
```

- Recogida de parámetros desde un JSP:

```
1 <%-- Alternativa 1: con useBean --%>
2 <jsp:useBean id="noticia" class="beans.Noticia"
3              scope="request" />
4
5 <%-- Alternativa 2: con scriptlet --%>
6 <% Noticia noticia =
7     (Noticia) request.getAttribute("noticia"); %>
```

Parte III

URIs relativas y absolutas

Introducción

- En HTML, en un hiperenlace, imagen, etc. es necesario especificar una URI.
- El navegador necesita la URI completa para seguir el hiperenlace, cargar la imagen, etc.
- Una URI se puede especificar como:
  - URI absoluta.
  - URI relativa a un servidor.
  - URI relativa.

URI absoluta

- Se especifica directamente la URI completa del recurso.
- En HTTP, incluye el identificador de protocolo, servidor, ruta en el servidor y parámetros.
- El navegador simplemente toma la URI.

```
1 <a href="http://www.it.uc3m.es/labttlat/lab8/">...</a>
```

URI relativa al servidor

- Se especifica de forma absoluta la ruta del recurso (comenzando por "/"), pero no se indica protocolo ni servidor.
- El navegador toma el protocolo y servidor del recurso en el cual está el enlace, imagen, etc.

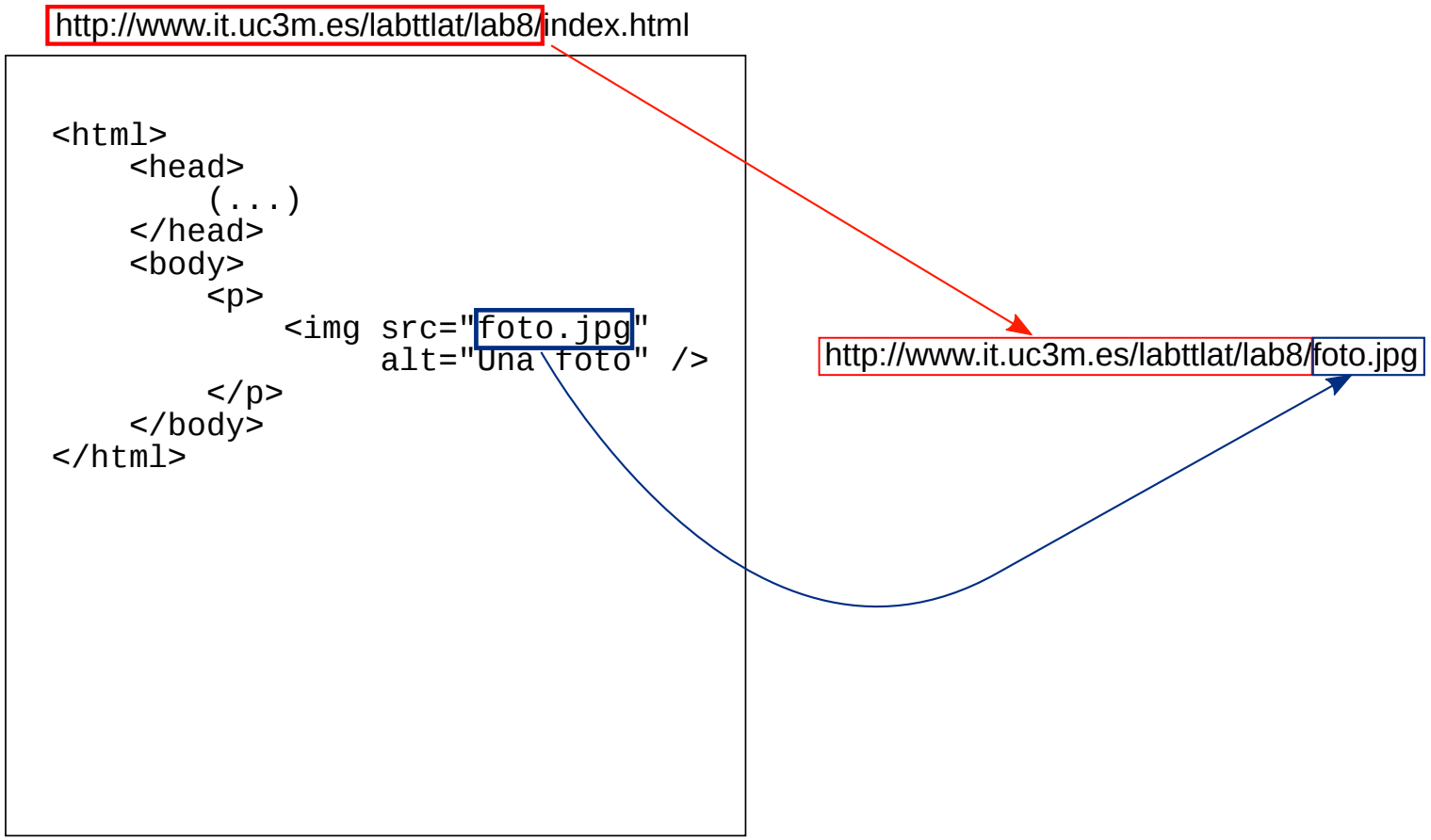
```
1 <a href="/labttlat/lab8/">...</a>
```

URI relativa

- Se especifica sólo la ruta del recurso relativa (no comienza por "/"), pero no se indica protocolo ni servidor, ni parte inicial de la ruta.
- El navegador toma el protocolo, servidor y parte inicial de la ruta del recurso en el cual está el enlace, imagen, etc.
  - Para calcular la ruta, se toma la ruta del recurso actual excepto su último nivel (similar a la forma de nombrar ficheros en un sistema de ficheros).

```
1 <a href="lab8/">...</a>
```

Ejemplo: URIs relativas



Recomendaciones de diseño

- Es recomendable utilizar rutas relativas siempre que sea posible:
  - Permite cambiar la aplicación de servidor o ruta sin necesidad de cambiar ninguna URI en los servlets, JSP, HTML, etc.