

Integración de Servlets y JSP

Laboratorio de Aplicaciones Telemáticas

Jesús Arias Fisteus

jaf@it.uc3m.es

Curso 2007/2008

Universidad Carlos III de Madrid

URIs relativas y absolutas



Introducción

- En (X)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.

```
<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.

```
<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).

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

Ejemplo: URIs relativas

<http://www.it.uc3m.es/labttlat/lab8/index.html>

```
<html>
  <head>
    (...)
  </head>
  <body>
    <p>
      <img href="foto.jpg"
            alt="Una foto" />
    </p>
  </body>
</html>
```

<http://www.it.uc3m.es/labttlat/lab8/foto.jpg>

Recomendaciones de diseño

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

Integración de Servlets y JSP



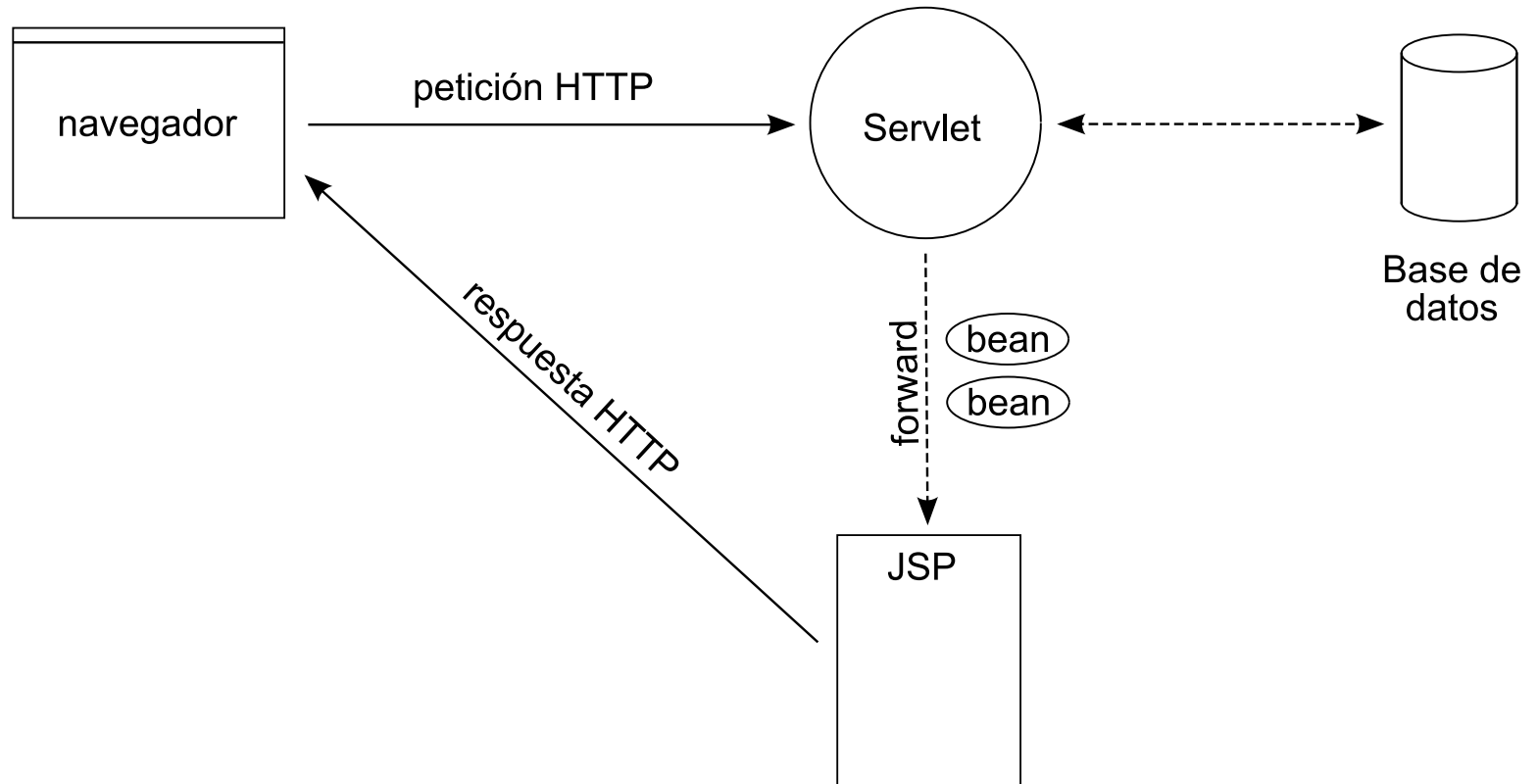
Introducción

- Una aplicación Web realiza tareas de procesamiento y presentación:
 - Los Servlets son adecuados para procesamiento.
 - Las páginas JSP son adecuadas para presentación.
- Una aplicación Web puede combinar Servlets y páginas JSP:
 - Procesamiento de parámetros de la petición: Servlets.
 - Acceso a bases de datos: Servlets.
 - Lógica de la aplicación: Servlets.
 - Presentación (vistas): JSP.

Modelo de funcionamiento (I)

1. El cliente envía la petición HTTP a un servlet.
2. El servlet procesa la petición.
 - Si es necesario, se conecta a la base de datos.
3. El servlet redirige la petición a un JSP.
 - Si es necesario, añade *beans* como parámetros.
4. El JSP lee los parámetros y devuelve la respuesta formateada visualmente al usuario.

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.
 - Redirecciones internas en el servidor (*forward*):
 - Se redirige la petición de un recurso a otro dentro de la misma aplicación Web.
 - El recurso de la última redirección devuelve al cliente la respuesta HTTP.
 - La redirección es transparente para el cliente.

Redirecciones *sendRedirect*

- Fuerza el envío de una respuesta HTTP de redirección al cliente.
- El cliente envía la petición a la URI recibida en la respuesta.

```
// Redirección con URI absoluta  
response.sendRedirect("http://www.ejemplo.es/");
```

```
// Redirección con URI relativa a la URI de la petición actual  
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:

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

■ *Forward* desde un JSP:

```
<jsp:forward page="/vista.jsp"/>
```

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.

```
Noticia nuevaNoticia = (...)  
request.setAttribute("noticia", nuevaNoticia);  
RequestDispatcher rd = request.getRequestDispatcher("/vista.jsp");  
rd.forward(request, response);
```

Redirecciones *forward* con parámetros

■ Recogida de parámetros desde un Servlet:

```
Noticia nuevaNoticia = (Noticia) request.getAttribute("noticia");
```

■ Recogida de parámetros desde un JSP:

```
<jsp:useBean id="noticia" class="beans.Noticia"  
            scope="request" />
```

Envío de parámetros de formularios

Envío de parámetros de formularios

- El envío depende del método HTTP y la codificación:
 - Método HTTP:
 - Método GET.
 - Método POST.
 - Codificación:
 - `application/x-www-form-urlencoded`
 - `multipart/form-data`

Envío de parámetros de formularios

- Codificación URL–encoded:
 - Lista de parámetros separados por “&”.
 - Para cada parámetro se especifica nombre “=” valor.
 - Los caracteres especiales (no letras/dígitos ASCII) se codifican en hexadecimal por su código UTF-8.
 - Con método GET o POST.
 - No se usa para campos de tipo *file*.

```
usuario=juan&clave=juanpw&ssid=7fgxc&enviar=enviar  
nombre=juan%201%C3%B3pez%201%C3%B3pez
```

Envío de parámetros de formularios

- Codificación URL–encoded con GET:
 - Los parámetros se codifican en la ruta (*path*) de la petición HTTP.
 - Sólo apto para operaciones idempotentes.

```
GET /jaf/cgi-bin/html2xhtml.cgi?tipo=auto&html=default.html HTTP/1.1
Host: www.ejemplo.es
(...)
```

Envío de parámetros de formularios

- Codificación URL–encoded con POST:
 - Los parámetros se codifican en el cuerpo de la petición HTTP.

```
POST /jaf/cgi-bin/html2xhtml.cgi HTTP/1.1
(...)
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

tipo=auto&html=default.html
```

Envío de parámetros de formularios

- Codificación Multipart (RFC 2388):
 - Datos encapsulados con un mensaje multiparte MIME.
 - Sólo con método POST.
 - Necesario para enviar campos de tipo *file*.
 - No compatible con *HttpServletRequest.getParameter(...)*
 - Es necesario utilizar APIs adicionales desde un Servlet/JSP.

Ejemplo: multipart/form-data

```
POST /jaf/cgi-bin/html2xhtml.cgi HTTP/1.1
(...)
Content-Type: multipart/form-data; boundary=-----2qYzCGdatrpobJh4m5rz50
Content-Length: 972

-----2qYzCGdatrpobJh4m5rz50
Content-Disposition: form-data; name="tipo"

auto
-----2qYzCGdatrpobJh4m5rz50
Content-Disposition: form-data; name="html"; filename="readme.html"
Content-Type: text/html

<html xmlns="http://www.w3.org/1999/xhtml">
(...)
</html>
-----2qYzCGdatrpobJh4m5rz50--
```