



Universidad de Huelva

Nuevas Tecnologías de la Programación

Objetivos del tema

Tema 3. Servlets

- 3.1 Introducción
- 3.2 Inicialización
- 3.3 Petición (Request)
- 3.4 Respuesta (Response)
- 3.5 Traza de usuarios




Servlets

- **Ciclo de vida:** Un servlet genérico pasa por tres etapas
 - Inicialization:** Creación e inicialización de los recursos del servlet. Todo servlet implementa la interface `javax.servlet.Servlet`, que define el método `init()`
 - Service:** Atiende las peticiones de los clientes y envía las respuestas. La interface `Servlet` anterior define un método `service()` con dos parámetros:
 - Request:** Petición del cliente al servidor
 - Response:** Respuesta del servidor para el cliente.
 - Destruction:** Borra el objeto servlet cuando no es necesario. La interface `Servlet` define un método `destroy()` con ese objetivo.

Initialization
(Load Resources)

↓

Request → Service
(Accept Requests)

← Response

↓

Destruction
(Unload Resources)

javax.servlet.http.HttpServlet

Initialization
(Load Resources)

↓

Request → Service
(Accept Requests)

← Response

↓

Destruction
(Unload Resources)

doGet()

doPost()

doPut()

doHead()

doDelete()

doTrace()

doOptions()

Universidad de Huelva
Dpto. Ing. Electrónica, Sist. Informáticos y Automática
Curso 2005/2006

Servlets

- Interface **Servlets** (paquete javax.servlet)

Method Summary	
void	destroy() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
ServletConfig	getServletConfig() Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.
java.lang.String	getServletInfo() Returns information about the servlet, such as author, version, and copyright.
void	init(ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
void	service(ServletRequest req, ServletResponse res) Called by the servlet container to allow the servlet to respond to a request.

- **ServletConfig**: Interface para pasar información del Contenedor de Servlets al servlet.
- **GenericServlet**: Clase abstracta que implementa las interfaces Servlet y ServletConfig, para crear Servlet genéricos independientes del protocolo
- **HttpServlet**: Clase abstracta que hereda de GenericServlet, para crear HTTP Servlet

Servlets: Inicialización

Método init(): void init (ServletConfig conf)

Interface ServletConfig – Permite pasar información de inicialización

String getInitParameter(paramName) : Parámetros iniciales (web.xml)
Enumeration getInitParameterNames() : idem etiqueta <init-param>
ServletContext getServletContext() : Contexto del Servlet
String getServletName() : Nombre del servlet

Interface ServletContext – Permite usar información del contexto

String getInitParameter(paramName) : Parámetros iniciales (web.xml)
Enumeration getInitParameterNames() : idem etiqueta <context-param>
RequestDispatcher getRequestDispatcher(ruta): } Devuelve un objeto para delegar
RequestDispatcher getNameDispatcher(nombre): } la petición a otro servlet/jsp/html
void setAttribute(nomb, objeto) : Asocia un atributo al contexto
Object getAttribute(nombr) : Devuelve el valor de un atributo
void removeAttribute(nomb) . Elimina un atributo del contexto

Servlets: Inicialización

- Ejemplo

- Contenido del Servlet

```
public void init(ServletConfig config) throws ServletException {  
    super.init(config);  
    parametroServlet = config.getInitParameter("parServlet");  
    ServletContext context = config.getServletContext();  
    parametroContexto = context.getInitParameter("parContext");  
}
```

- Contenido de descriptor de aplicación (web.xml)

```
<web-app ... >  
...  
<context-param>  
  <param-name>parContext</param-name>  
  <param-value>ValorParametro</param-value>  
</context-param>  
...  
<init-param>  
  <param-name>parServlet</param-name>  
  <param-value>valorPar</param-value>  
</init-param>  
...  
</servlet>
```

Servlets: Petición. Objeto HttpServletRequest

- En este objeto se recibe la petición del cliente. Algunos métodos:

getParameter(NombParam): Devuelve el valor del parámetro especificado o null si el parámetro no existe.

getParameterValues(NombParam): Devuelve un array con los valores del parámetro indicado. Este método es útil para elementos que tienen más de una valor (p.e. checkbox o combobox de un formulario).

getParameterNames(): Devuelve un objeto Enumeration (java.util.Enumeration) de todos los parámetros enviados en la petición.

getInputStream(): Devuelve un objeto ServletInputStream que permite recibir otro tipo de información (como ficheros) desde el cliente.

getHeader(nomb)
getHeaders(nomb)
getHeaderNames()
getIntHeader(nomb)
getDateHeader(nomb)

} Devuelve la información de las cabeceras HTTP del cliente

getCookies(): Devuelve un array de objetos Cookie enviados desde el cliente

getSession(crear): Devuelve un objeto HttpSession, creando una sesión.

Servlets: Petición. Objeto HttpServletRequest

- Lectura de parámetros enviados por el cliente

String **getParameter**(*nombre*) : Devuelve el valor del parámetro *nombre* enviado en la petición.

String[] **getParameterValues**(*nombre*) : Devuelve un array con los valores del parámetro *nombre* enviado en la petición.

Enumeration **getParameterNames**() : Devuelve un objeto Enumeration con los nombres de los parámetros enviados en la petición

Servlets: Petición. Objeto HttpServletRequest

- Lectura de parámetros enviados por el cliente

Ejemplo

```
Enumeration nombresParam = request.getParameterNames();
while(nombresParam.hasMoreElements()) {
    String nombreParam = (String) nombresParam.nextElement();
    out.print("<p>" + nombreParam + ": ");
    String[] valoresParam = request.getParameterValues(nombreParam);
    if (valoresParam.length == 1) {
        String valorParam = valoresParam[0];
        if (valorParam.length() == 0) out.println("<i>Sin valor</i>");
        else out.println(valorParam);
    } else {
        out.println("<ul>");
        for(int i=0; i<valoresParam.length; i++) {
            out.println("<i>" + valoresParam[i] + "</i>");
        }
        out.println("</ul></p>");
    }
}
```



Servlets: Petición. Objeto HttpServletRequest

- Cabeceras HTTP de Petición

String **getHeader**(nombCab) : Devuelve el valor de la cabecera nombCab.

Enumeration **getHeaders**(nombCab) : Devuelve un objeto Enumeration con los valores de la cabecera nombCab.

Enumeration **getHeadersNames**() : Devuelve un objeto Enumeration con los nombres de las cabeceras.

int **getIntHeader**(nombCab) : Devuelve el valor de la cabecera nombCab como un valor entero.

long **getDateHeader**(nombCab) : Devuelve el valor de la cabecera nombCab como un objeto tipo fecha (Date).



Servlets: Petición. Objeto HttpServletRequest

- Cabeceras HTTP de Petición

Accept: Tipo de respuesta que acepta.

Accept-Charset: Conjuntos de caracteres que acepta.

Accept-Encoding: Tipo de codificación acepta.

Accept-Language: Lenguaje de la respuesta que se prefiere.

Authorization: Información de autenticación del cliente

Connection, permite especificar opciones requeridas para una conexión.

Date, representa la fecha y la hora a la que se creó el mensaje.

From: Dirección de correo del usuario.

Host: La máquina y el puerto del recurso pedido.

Max-Forwards, indica el máximo número de elementos por los que pasa.

Proxy-Authorization, permite que el cliente se identifique a un proxy.

Range, establece un rango de bytes del contenido.

Referer, indica la dirección donde obtuvo la URI de la petición.

Transfer-Encoding, indica la codificación aplicada al contenido.

Upgrade, permite al cliente especificar protocolos que soporta.

User-Agent, información sobre el agente que genera la petición.

Via, usado por pasarelas y proxies para indicar los pasos seguidos.

Servlets: Petición. Objeto HttpServletRequest

- Cabeceras HTTP de Petición

Ejemplos de peticiones HTTP:

```
GET /index.html HTTP/1.1
Host: www.unejemplo.com
User-Agent: Mozilla/4.5 [en]
Accept: image/jpeg, image/gif, text/html
Accept-language: es
Accept-Charset: iso-8859-1
```

```
POST /indice.jsp HTTP/1.0
Host: www.unejemplo.com
http://www.unejemplo.com/indice.jsp?
nombre=Fullano+Mengano&OK=1
User-Agent: Mozilla/4.5 [en]
Accept: image/jpeg, image/gif, text/html
Accept-language: en
Accept-Charset: iso-8859-1

nombre=Perico+Palotes&OK=1
```

Servlets: Petición. Objeto HttpServletRequest

- Cabeceras HTTP de Petición

Ejemplos

```
Enumeration nombresEncabezados = request.getHeaderNames();
while(nombresEncabezados.hasMoreElements()) {
    String nombreEncabezado = (String) nombresEncabezados.nextElement();
    out.println("<p>" + nombreEncabezado + ": ";
    out.println(request.getHeader(nombreEncabezado) + "</p>");
}
```

Servlets: Respuesta. Objeto HttpServletResponse

- Este objeto permite enviar la respuesta al cliente. Algunos métodos son:

getWriter(): devuelve un objeto tipo `PrintWriter` para el envío de texto plano.

Un objeto `PrintWriter` dispone de los métodos: `print()`, `println()` o `write()`.

getOutputStream(): devuelve un objeto tipo `ServletOutputStream` que permite enviar un flujo de bytes.

sendRedirect(*url*): La respuesta es la url indicada.

sendError(*error*, *mensaje*): La respuesta es creada con el código error y el mensaje

setContentType(*tipo*): Establece el tipo de respuesta. El más común `text/html`

addHeader(*nomb*, *valor*): Añade la cabecera *nomb* con el valor indicado

containsHeader(*nomb*): Devuelve verdadero si la cabecera *nomb* está definida

setHeader(*nomb*, *valor*): Asigna el valor indicado a la cabecera *nomb*.

setIntHeader(*nomb*, *valorInt*): Asigna el valor entero a la cabecera *nomb*.

setDateHeader(*nomb*, *fecha*): Asigna el valor (*fecha*) a la cabecera *nomb*.

addIntHeader(*nomb*, *valorInt*): Añade la cabecera *nomb*.

addDateHeader(*nomb*, *fecha*): Añade la cabecera *nomb*.

addCookie(*cookie*): Envía una cookie al cliente, Donde *cookie* es un objeto `javax.servlet.http.Cookie` que debe ser creado,

Servlets: Respuesta. Objeto HttpServletResponse

- Estructura de la respuesta

Línea de estado

*(Cabeceras)

CRLF

[Contenido]

```
HTTP/1.1 200 OK Date: Sun, 30 Apr 2006 10:49:37
GMT Server: Apache/1.3.29 (Unix) FrontPage/5.0.2.2635
Last-Modified: Wed, 01 Mar 2006 18:23:35 GMT
Content-Length: 67411
Connection: close
Content-Type: text/html
```

Códigos de estado:

1xx: Informativo. La petición se recibe y sigue el proceso. Esta familia de respuestas indican una respuesta provisional. Este tipo de respuesta está formada por la línea de estado y las cabeceras.

2xx: Éxito. La acción requerida por la petición ha sido recibida, entendida y aceptada.

3xx: Redirección. Para completar la petición se han de tomar más acciones.

4xx: Error del cliente. La petición no es correcta y no se puede llevar a cabo.

5xx: Error del servidor. El servidor falla al atender la petición que aparentemente es correcta.

Servlets: Respuesta. Objeto HttpServletResponse

- Cabeceras HTTP de Respuesta

Server: Información sobre el servidor que maneja las peticiones.

Vary: Indica que hay varias respuestas y el servidor ha escogido una.

Warning: usada para aportar información adicional sobre el estado de la respuesta.

WWW-Authenticate: Información de autenticación.

Content-Language: Idioma del contenido.

Content-Length: Tamaño del contenido del mensaje.

Content-Location: Localización del recurso que da el contenido del mensaje.

Content-Type: Indica el tipo de contenido.

Last-Modified: Indica la fecha de la última modificación.

Refresh : Indica cuándo debería pedir el navegador una página actualizada

Set-Cookie : Indica la cookie asociada a la página.

Servlets: Respuesta. Objeto HttpServletResponse

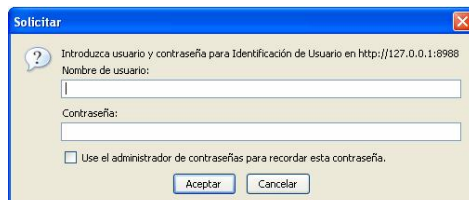
- Cabeceras HTTP de Respuesta. Ejemplo

Authorization : Cabecera de petición en la que se envía el nombre de usuario y la clave, codificado BASE64, si el usuario se ha identificado.

```
String a = request.getHeader("Authorization");
```

WWW-Authenticate : Cabecera de respuesta que fuerza al navegador a solicitar identificación de usuario. Va asociada a una respuesta con código 401 (no autorizado)

```
response.setHeader("WWW-Authenticate", "BASIC realm=\"Identificación de Usuario\"");  
response.sendError(401, "<h2>Página restringida a usuarios registrados</h2>");
```






Servlets: Respuesta. Objeto HttpServletResponse

- Cabeceras HTTP de Respuesta. Ejemplo

```
private void identif(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException{
    String a = request.getHeader("Authorization");
    if (a==null) {
        response.setHeader("WWW-Authenticate",
            "BASIC realm=\"Identificación de Usuario\"");
        response.sendError(401, "<h2>Página restringida</h2>");
    }
    else {
        String auth = a.substring(6).trim();

        BASE64Decoder decoder = new BASE64Decoder(); //sun.misc.BASE64Decoder;
        String usuarioClave = new String(decoder.decodeBuffer(auth));
        int indice = usuarioClave.indexOf(":");
        name = usuarioClave.substring(0, indice);
        clave = usuarioClave.substring(indice+1);
    }
}
```



Servlets: Ejercicio

- Crear un Servlet para Identificación de usuarios
 - Solicitará identificación del usuario mediante cabecera.
 - Para comprobar la identificación del usuario leerá dos parámetros iniciales (usuario y clave) que debe coincidir con la indicada por el usuario.
 - Si la identificación es correcta se mostrará la información solicitada.
 - En otro caso se mostrará una página que aclare el problema de identificación.



Servlets: Delegar una petición


- Los Servlets pueden delegar las peticiones a otros Servlets/JSP/HTML. Esto es necesario para implementaciones del “Model 2”.

Para ello:

- Recoger el contexto del Servlet.
`getServletContext()` : Devuelve el contexto del Servlet (`ServletContext`)
- Invocar el método:
`getRequestDispatcher(recurso)` : Devuelve un objeto `RequestDispatcher`
- Redirigir la respuesta al recurso, mediante el método:
`forward (petición, respuesta)`.

Ejemplo:

```
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher("/shoppingCar.jsp");
rd.forward( request, response);
```



Servlets: Ejercicio

- Crear un Servlet para Identificación de usuarios
 - Solicitará identificación del usuario mediante cabecera.
 - Para comprobar la identificación del usuario leerá dos parámetros iniciales (usuario y clave) que debe coincidir con la indicada por el usuario.
 - Si la identificación es correcta se mostrará la información solicitada. (*)
 - En otro caso se mostrará una página que aclare el problema de identificación.
- Ampliar el ejemplo anterior incluyendo un segundo Servlet que muestre la información deseada.




Servlets: Traza de usuarios

- HTTP es un protocolo sin estado

Posibilidades:

- 1) Reescritura de url's
- 2) Formulario con campos ocultos
- 3) Cookies
- 4) Sesiones



Servlets: Traza de usuarios. Cookies

- Cookies
 - Son pares nombre, valor con una caducidad establecida y propias de un dominio.
 - Son almacenadas en el cliente y son enviadas, como cabeceras, en cada petición de este a ese dominio del servidor.
 - En Java se incluye la clase `Cookie`, cuyos métodos principales son:
 - constructor*. `Cookie(nombre, valor)` : crea la cookie nombre=valor
 - `getName()` : Devuelve el nombre.
 - `getValue()` : Devuelve el valor.
 - `setDomain(dom)` : Establece el dominio al que debe enviarse.
 - `getDomain()` : Devuelve el dominio.
 - `setMaxAge(tpo)` : Establece el tiempo, en segundo, en el que expirará.
 - `getMaxAge()` : Devuelve el número de segundos de validez de la cookie.
 - `setPath(ruta)` : Establece la ruta a la que debe enviarse.
 - `getPath()` : Devuelve la ruta.
 - `setSecure(ok)` : Indica si la cookie debe enviarse sólo bajo un protocolo seguro.
 - `getSecure()` : Devuelve verdadero si la cookie debe enviarse bajo protocolo seguro.
 - `setComment(comentario)` : Especifica un comentario sobre el objetivo de la cookie
 - `getComment()` : Devuelve el comentario sobre el objetivo de la cookie.

Servlets: Traza de usuarios. Cookies

- Cookies: Uso en Servlets

- Crear la Cookie

```
Cookie miCookie = new Cookie( "nombre", "valor" );  
// Usar el resto de métodos para dominio, ruta, comentario, seguridad, etc.
```

- Añadirla a la respuesta: HttpServletResponse

```
response.addCookie( miCookie );
```

- Leer un Cookie: HttpServletRequest

```
Cookie c[ ] = request.getCookies();  
for( int i=0; c.length ; i++ )  
    out.println("Cookie: " + c[ i ].getName() +" vale " + c[i].getValue() );
```

Servlets: Traza de usuarios: Sesiones

- Sesiones

- Java dispone del objeto HttpSession accesible desde HttpServletRequest
- Utiliza la funcionalidad de la cookies, pero permiten almacenar pares atributo, valor que son válidos en una sesión (hasta que el usuario cierre el navegador)

- HttpSession

Object **getAttribute(String name)** : Devuelve el atributo name

void **setAttribute(String name, Object value)** : Asigna un atributo a la sesión.

void **removeAttribute(String name)** : Elimina un atributo de la sesión.

Enumeration **getAttributeNames()** : Devuelve un Enumeration de los atributos

long **getCreationTime()** : Devuelve el tiempo en que la sesion fue creada, en milisegundos, transcurrido 1/1/1970 GMT.

String **getId()** : Devuelve el identificador de la sesion.

long **getLastAccessedTime()** : Devuelve el tiempo trasncurrido desde que el cliente envió la última petición medido en milisegundos desde 1/1/1970.

int **getMaxInactiveInterval()** : Devuelve el periodo, en segundos, que el contenedor de servlets mantendrá abierta la sesion.

ServletContext **getServletContext()** : Devuelve el contexto de la sesion.

void **invalidate()** : Invalida la sesión, eliminando todos los atributos.

boolean **isNew()** : Devuelve true si el cliente no tenía abierta una sesion.

void **setMaxInactiveInterval(int interval)** : Establece el tiempo de validez de la sesión.



Servlets: Traza de usuarios: Sesiones

- Sesiones: Uso con Servlets
 - Activar la sesion desde HttpServletRequest
`sesion = request.getSession(true);`
 - Uso de atributos en una sesión: HttpSession
`sesion.setAttribute("dni", new String("11.111.111"));`
`out.println("Su dni es " + sesion.getAttribute("dni"));`
`sesion.removeAttribute("dni");`
 - Invalidar una sesión
`sesion.invalidate();`
 - Mostrar el identificador de una sesión:
`out.println("ID de la sesión actual: " + sesion.getID());`