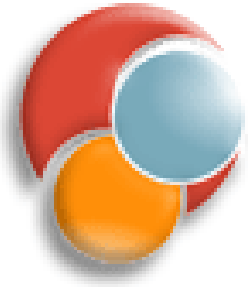

Componentes Web

Sesión 2: Procesamiento de peticiones

Puntos a tratar

- Peticiones
- Respuestas
- Procesamiento de peticiones GET y POST
- Manejo de formularios
- Cabeceras de petición
- Cabeceras de respuesta
- Variables CGI
- Códigos de estado HTTP

ServletRequest y *HttpServletRequest*

- Los objetos *ServletRequest* se emplean para obtener información sobre las peticiones de los clientes
- En concreto, el subtipo *HttpServletRequest* se emplea en las peticiones HTTP
  - Proporciona acceso a los datos de las cabeceras HTTP
  - Proporciona acceso a las cookies
  - Permite ver los parámetros pasados por el usuario
  - ... y todo sin tener que procesar nosotros la petición para obtener los datos (*datos de formulario*)

Métodos útiles

- En esta clase se tienen, entre otros, los métodos:
  - Para obtener nombres y valores de parámetros de una petición (cuidando mayúsculas y minúsculas)

```
Enumeration getParameterNames()
String getParameter(String nombre)
String getParameterValues(String nombre)
```

  - Para obtener los parámetros de una petición GET (devuelve una cadena que deberemos parsear nosotros)

```
String getQueryString()
```

Métodos útiles (II)

- Para obtener los datos enviados con POST, PUT o DELETE

```
BufferedReader getReader()
ServletInputStream getInputStream()
```

- Para obtener el método HTTP, la URI (parte de la URL tras el host, sin contar los datos del formulario) o el protocolo

```
String getMethod()
String getRequestURI()
String getProtocol()
```

ServletResponse y *HttpServletResponse*

- Los objetos *ServletResponse* se emplean para enviar en ellos una respuesta a una petición de un cliente

- En concreto, el subtipo *HttpServletResponse* se emplea para enviar respuestas HTTP

Métodos útiles

- Destacan los métodos para obtener el canal de salida donde escribir la respuesta:

```
Writer getWriter()
ServletOutputStream getOutputStream()
```

- Si se quieren indicar cabeceras, se deben indicar ANTES de obtener estos objetos

Métodos para atender peticiones

- Hemos visto que los métodos *doGet(...)* y *doPost(...)* atienden las peticiones GET y POST:

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
throws ServletException, IOException
{
   ... // Código del método
}

public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
throws ServletException, IOException
{
   ... // Código del método, si es diferente a doGet
   ... // Si no, llamar a doGet y ya está
}
```

Qué hacer con la petición

- Acceder a valores de parámetros

```
String nombreUsuario = request.getParameter("nombre");
```

- Acceder a los parámetros de la petición y procesarlos como queramos

```
String peticion = request.getQueryString();
```

- Obtener un canal de entrada

```
BufferedReader r = request.getReader();
```

  - Esta no es buena idea si tomamos parámetros de peticiones. Se suele usar para transferencia de ficheros

Qué hacer con la respuesta

- Establecer valores de cabecera (antes que nada)

```
response.setContentType(“text/html”);
```

- Obtener el canal de salida por el que enviar la respuesta, y enviar contenido

```
PrintWriter out = response.getWriter();
out.println (“<HTML><BODY>...”);
```

- Redirigir a otra página

```
response.sendRedirect(“http://localhost:8080/pag.html”);
```

Procesamiento secuencial de peticiones

- Los servlets pueden gestionar múltiples peticiones concurrentemente
- Si, por algún recurso compartido u otra causa, queremos que hagan un procesamiento secuencial, podemos:
  - Utilizar bloques de código *synchronized* en el código del servlet necesario
  - Hacer que el servlet implemente la interfaz *SingleThreadModel* (sin métodos)

```
public class MiServlet extends HttpServlet
implements SingleThreadModel
{ ...
```

Manejo de formularios

- Los datos que se envían tras el ? en una petición GET o en una línea aparte en una petición POST se llaman *datos de formulario*.
- Los servlets procesan estos datos de forma automática y extraen la información útil, con métodos como:

```
Enumeration getParameterNames()
String getParameter(String nombre)
String getParameterValues(String nombre)
```

Obtención de cabeceras de petición

- El objeto *HttpServletRequest* de la petición tiene un método *getHeader( )* para acceder a valores de cabeceras

```
String getHeader(String nombre)
```

- Se devuelve el valor de la cabecera *nombre* o *null* si no se ha encontrado la cabecera
- Se tienen otros métodos alternativos como:

```
Enumeration getHeaderNames()
Enumeration getHeaders(String nombre)
int getIntHeader(String nombre)
...
```

Obtención de cabeceras concretas

- Se tienen algunos métodos para obtener cabeceras concretas, como:

```
Cookie[] getCookies()
String getContentLength()
String getContentType()
...
```

Establecer cabeceras de respuesta

- El objeto *HttpServletResponse* tiene un método *setHeader( )* para establecer valores de cabeceras

```
void setHeader(String nombre, String valor)
```

- Se tienen otros métodos alternativos como:

```
void setIntHeader(String nombre, int valor)
void addHeader(String nombre, String valor)
void addIntHeader(String nombre, int valor)
...
```

- Con los métodos *add(...)* podemos añadir múltiples valores a una cabecera con el mismo nombre

Establecer cabeceras concretas

- Se tienen algunos métodos para establecer cabeceras concretas, como:

```
void setContentType(String tipo)
void setContentLength(int tamaño)
void sendRedirect(String url)
void addCookie(Cookie cookie)
...
```

Qué son las variables CGI

- Las variables CGI son una forma de recoger información de una petición
- Se derivan de la línea de la petición, cabeceras, el socket, parámetros del servidor, etc.
- Algunos ejemplos:
  - CONTENT_LENGTH: número de bytes enviados
  - PATH_INFO: información del path junto a la URL
  - REMOTE_ADDR: IP del cliente que hizo la petición
  - REQUEST_METHOD: tipo de petición (GET, POST...)
  - SERVER_PORT: puerto por el que escucha el servidor

Acceso a las variables CGI

- El objeto *HttpServletRequest* y el *ServletContext* del servlet contienen métodos para acceder a las variables CGI:
  - *request.getContentLength()* para CONTENT_LENGTH
  - *request.getPathInfo()* accede a PATH_INFO
  - *request. getRemoteAddr()* accede a REMOTE_ADDR
  - *request. getMethod()* accede a REQUEST_METHOD
  - *request. getServerPort()* accede a SERVER_PORT
  - ... etc

Utilidad de los códigos

- Cuando un servidor envía una respuesta a una petición, envía también un código que indica el resultado de la petición
- El envío de cabeceras se planifica con el envío de códigos, puesto que muchos códigos necesitan una cabecera complementaria (para redirecciones, autentificaciones, etc)

Envío de códigos

- Para enviar un código de estado, el objeto *HttpServletResponse* tiene el método *setStatus( )*

```
void setStatus(int estado)
```

- Se tienen en la clase varias constantes para representar distintos estados

```
setStatus(HttpServletResponse.SC_NOT_FOUND);   //Codigo 404
```

- Algunos métodos se usan para enviar códigos específicos, porque envían también la cabecera necesaria

```
void sendError(int codigo, String mensaje);    //Codigo 4xx o 5xx
void sendRedirect(String url);                 //Codigo 302
```