Cabeceras y códigos desde servlets

Índice

Veremos a continuación cómo tratar las cabeceras HTTP de una petición y de una respuesta, así como los códigos de estado que emite un servidor Web ante una petición, y las variables CGI a las que podemos acceder.

1. Cabeceras de petición

Cuando se envía una petición HTTP, se pueden enviar, entre otras cosas, unas cabeceras con información sobre el navegador. Para leer estas cabeceras de una petición desde un servlet, se utiliza el método **`getHeader()`** del objeto **`HttpServletRequest`**.

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

El parámetro indica el nombre de la cabecera cuyo valor se quiere obtener. Devuelve el valor de la cabecera, o `null` si la cabecera no ha sido enviada en la petición.

Se tienen otros métodos, como:

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

Con **`getHeaderNames()`** obtendremos todos los nombres de las cabeceras enviadas. Con **`getHeaders()`** obtendremos todos los valores de la cabecera de nombre dado. También hay métodos como **`getIntHeader()`** que devuelve el valor de una cabecera con un tipo de dato específico (entero, en este caso). Los nombres de las cabeceras normalmente no distinguen mayúsculas de minúsculas.

Algunas cabeceras son de uso común, y tienen métodos específicos para obtener sus valores, como:

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

Con **`getCookies()`** obtendremos todas las cookies de la petición (veremos las cookies con más detalle en otro tema). Con **`getContentLength()`** obtenemos el valor de la cabecera `Content-Length`, y con **`getContentType()`** el de la cabecera `Content-Type`.

2. Cabeceras de respuesta

En la respuesta de un servidor web a una petición también pueden aparecer cabeceras que informan sobre el documento servido o sobre el propio servidor. Podemos definir cabeceras de respuesta para enviar cookies, indicar la fecha de modificación, etc. Estas cabeceras deben establecerse ANTES de enviar cualquier documento, o antes de obtener el `PrintWriter` si

es el caso.

Para enviar cabeceras, el método más general es **`setHeader()`** del objeto **`HttpServletResponse`**.

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

Al que se le pasan el nombre de la cabecera y el valor. Hay otros métodos útiles:

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

`setIntHeader()` o **`setDateHeader()`** se utilizan para enviar cabeceras de tipo entero o fecha. Los métodos **`add...()`** se emplean para añadir múltiples valores a una cabecera con el mismo nombre.

Algunas cabeceras tienen métodos específicos de envío, como:

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

Con **`setContentType()`** se establece la cabecera `Content-Type` con el tipo MIME del documento. Con **`setContentLength()`** se indican los bytes enviados. Con **`sendRedirect()`** se selecciona la cabecera `Location`, y con ella se redirige a la página que le digamos. Finalmente, con **`addCookie()`** se establecen cookies (esto último ya lo veremos con más detalle en otro tema). Es recomendable utilizar estos métodos en lugar del método `setHeader()` para la cabecera en cuestión.

3. Variables CGI

Las variables CGI son una forma de recoger información sobre una petición. Algunas se derivan de la línea de petición HTTP y de las cabeceras, otras del propio socket (como el nombre o la IP de quien solicita la petición), y otras de los parámetros de instalación del servidor (como el mapeo de URLs a los paths actuales).

Mostramos a continuación una tabla con las variables CGI, y cómo acceder a ellas desde servlets:

VARIABLE CGI	SIGNIFICADO	ACCESO DESDE SERVLETS
AUTH_TYPE	Tipo de cabecera Authorization (basic o digest)	`request. getAuthType()`
CONTENT_LENGTH	Número de bytes enviados en peticiones POST	`request. getContentLength()`

CONTENT_TYPE	Tipo MIME de los datos adjuntos	`request. getContentType()`
DOCUMENT_ROOT	Path del directorio raíz del servidor web	`getServletContext(). getRealPath("/")`
HTTP_XXX_YYY	Acceso a cabeceras arbitrarias HTTP	`request. getHeader("Xxx-Yyy")`
PATH_INFO	Información de path adjunto a la URL	`request. getPathInfo()`
PATH_TRANSLATED	Path mapeado al path real del servidor	`request. getPathTranslated()`
QUERY_STRING	Datos adjuntos para peticiones GET	`request. getQueryString()`
REMOTE_ADDR	IP del cliente que hizo la petición	`request. getRemoteAddr()`
REMOTE_HOST	Nombre del dominio del cliente que hizo la petición (o IP si no se puede determinar)	`request. getRemoteHost()`
REMOTE_USER	Parte del usuario en la cabecera Authorization (si se suministró)	`request. getRemoteUser`
REQUEST_METHOD	Tipo de petición (GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE)	`request. getMethod()`
SCRIPT_NAME	Path del servlet	`request. getServletPath()`
SERVER_NAME	Nombre del servidor web	`request. getServerName()`
SERVER_PORT	Puerto por el que escucha el servidor	`request. getServerPort()`
SERVER_PROTOCOL	Nombre y versión usada en la línea de petición (HTTP/1.0, HTTP/1.1 ...)	`request. getServerProtocol()`
SERVER_SOFTWARE	Información del servidor web	`getServletContext(). getServerInfo()`

`request` se asume que es un objeto de tipo `HttpServletRequest`. Para obtener

cualquiera de las variables antes mencionadas, sólo hay que llamar al método apropiado desde `doGet()` o `doPost()`.

4. Códigos de estado HTTP

Cuando un servidor web responde a una petición, en la respuesta aparece, entre otras cosas, un código de estado que indica el resultado de la petición, y un mensaje corto descriptivo de dicho código.

El envío de cabeceras de respuesta normalmente se planifica junto con el envío de códigos de estado, ya que muchos de los códigos de estado necesitan tener una cabecera definida. Podemos hacer varias cosas con los servlets manipulando las líneas de estado y las cabeceras de respuesta, como por ejemplo reenviar al usuario a otros lugares, indicar que se requiere un password para acceder a un determinado sitio web, etc.

Para enviar códigos de estado se emplea el método **`setStatus()`** de **`HttpServletResponse`**:

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

Donde se le pasa como parámetro el código del estado. En la clase `HttpServletResponse` tenemos una serie de constantes para referenciar a cada código de estado. Por ejemplo, la constante:

```
HttpServletResponse.SC_NOT_FOUND
```

se corresponde con el código 404, e indica que el documento solicitado no se ha encontrado.

Existen otros métodos para gestión de mensajes de error:

```
void sendError(int codigo, String mensaje)
void sendRedirect(String url)
```

`sendError()` genera una página de error, con código de error igual a `codigo`, y con mensaje de error igual a `mensaje`. Se suele utilizar este método para códigos de error, y `setStatus()` para códigos normales.

`sendRedirect()` genera un error de tipo 302, envía una cabecera `Location` y redirige a la página indicada en `url`. Es mejor que enviar directamente el código, o hacer un `response.setHeader("Location", "http...")`, porque es más cómodo, y porque el servlet genera así una página con el enlace a la nueva dirección, para navegadores que no soporten redirección automática

Si queremos enviar un código en la respuesta, se tiene que especificar antes de obtener el objeto `PrintWriter`.

5. Ejemplos

5.1. Ejemplo de cabeceras de petición

El siguiente servlet muestra los valores de todas las cabeceras HTTP enviadas en la petición. Recorre las cabeceras enviadas y muestra su nombre y valor:

```
package ejemplos;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletCabecerasPeticion
extends HttpServlet
{
   // Metodo para GET

   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
   throws ServletException, IOException
   {
        PrintWriter out = response.getWriter();

        // Mostramos las cabeceras enviadas
        // en la peticion

        out.println ("<HTML>");
        out.println ("<BODY>");
        out.println ("<H1>Cabeceras</H1>");
        out.println ("<BR>");

        Enumeration cabeceras = request.getHeaderNames();

        while (cabeceras.hasMoreElements())
        {
                String nombre = (String)(cabeceras.nextElement());
                out.println ("Nombre: " + nombre +
                   ", Valor: " + request.getHeader(nombre));
                out.println ("<BR><BR>");
        }

        out.println ("</BODY>");
        out.println ("</HTML>");
   }

   // Metodo para POST

   public void doPost(HttpServletRequest request,
```

```
                        HttpServletResponse response)
   throws ServletException, IOException
   {
        doGet(request, response);
   }
}
```

Se puede probar con este formulario, pinchando el botón:

```
<html>
<body>
<form action=
 "/appcab/servlet/ejemplos.ServletCabecerasPeticion">
        <input type="submit" value="Pulsa aqui">
</form>
</body>
</html>
```

5.2. Ejemplo de cabeceras de respuesta

El siguiente servlet espera un parámetro `accion` que puede tomar 4 valores:

- **primos**: El servlet tiene un hilo que está constantemente calculando números primos. Al elegir esta opción se envía una cabecera `Refresh` y recarga el servlet cada 10 segundos, mostrando el último número primo que ha encontrado.
- **redirect**: Utiliza un `sendRedirect()` para cargar la página que se indique como parámetro
- **error**: Utiliza un `sendError()` para mostrar una página de error, con un mensaje de error definido por el usuario, y un código de error a elegir de una lista.
- **codigo**: Envía un código de estado HTTP (con `setStatus()`), a elegir de entre una lista.

```
package ejemplos;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletCabecerasRespuesta
extends HttpServlet implements Runnable
{
   // Ultimo numero primo descubierto
   long primo = 1;
   // Hilo para calcular numeros primos
   Thread t = new Thread(this);

   // Metodo de inicializacion

   public void init()
   {
        t.start();
```

```
}

// Metodo para GET

public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
throws ServletException, IOException
{
     String accion = request.getParameter("accion");

     if (accion.equals("primo"))
     {
             // Buscar el ultimo numero
             // primo y enviarlo

             response.setContentType("text/html");
             response.setHeader("Refresh", "10");
             PrintWriter out = response.getWriter();
             out.println ("<HTML><BODY>");
             out.println ("Primo: " + primo);
             out.println ("</BODY></HTML>");

     } else if (accion.equals("redirect")) {

             // Redirigir a otra pagina

             String url = request.getParameter("url");
             if (url == null)
                     url = "http://www.ua.es";
             response.sendRedirect(url);

     } else if (accion.equals("error")) {

             // Enviar error con sendError()

             int codigo = response.SC_NOT_FOUND;
             try
             {
                codigo = Integer.parseInt
                   (request.getParameter("codigoMensaje"));
             } catch (Exception ex) {
                codigo = response.SC_NOT_FOUND;
             }
             String mensaje = request.getParameter("mensaje");
             if (mensaje == null)
                mensaje = "Error generado";
             response.sendError(codigo, mensaje);

     } else if (accion.equals("codigo")) {

             // Enviar un codigo de error

             int codigo = response.SC_NOT_FOUND;
             try
```

```
                {
                   codigo = Integer.parseInt
                      (request.getParameter("codigo"));
                } catch (Exception ex) {
                   codigo = response.SC_NOT_FOUND;
                }
                response.setStatus(codigo);
        }
   }

   // Metodo para POST

   public void doPost(HttpServletRequest request,
                      HttpServletResponse response)
   throws ServletException, IOException
   {
        doGet(request, response);
   }

   ... el resto del codigo es para el hilo,
   para calcular numeros primos
   Puede consultarse en el fichero fuente,
   aqui se quita por simplificar
}
```

Se puede probar con este formulario, eligiendo la acción a realizar, introduciendo los parámetros necesarios en el formulario y pinchando el botón de Enviar Datos:

```
<html>
<body>
<form action=
"/appcab/servlet/ejemplos.ServletCabecerasRespuesta">

<table border="0">

<tr>
<td>
<input type="radio" name="accion" value="primo" selected>
Obtener ultimo numero primo
</td>
<td></td>
<td></td>
</tr>

<tr>
<td>
<input type="radio" name="accion" value="redirect">
Redirigir a una pagina
</td>
<td>
URL:
<input type="text" name="url" value="http://www.ua.es">
</td>
<td></td>
</tr>
```

```
<tr>
<td>
<input type="radio" name="accion" value="error">
Mostrar pagina de error
</td>
<td>
Mensaje:
<input type="text" name="mensaje"
value="Error generado por el usuario">
</td>
<td>
Codigo:
<select name="codigoMensaje">
<option name="codigoMensaje" value="400">400</option>
<option name="codigoMensaje" value="401">401</option>
<option name="codigoMensaje" value="403">403</option>
<option name="codigoMensaje" value="404" selected>404
</option>
</select>
</td>
</tr>

<tr>
<td>
<input type="radio" name="accion" value="codigo">
Enviar codigo de error
</td>
<td>
Codigo:
<select name="codigo">
<option name="codigo" value="200">200</option>
<option name="codigo" value="204">204</option>
<option name="codigo" value="404" selected>404</option>
</select>
</td>
<td></td>
</tr>

</table>

<input type="submit" value="Enviar Datos">

</form>
</body>
</html>
```

5.3. Ejemplo de autentificación

El siguiente servlet emplea las cabeceras de autentificación: envía una cabecera de autentificación si no ha recibido ninguna, o si la que ha recibido no está dentro de un conjunto de `Properties` predefinido, con logins y passwords válidos. En el caso de

introducir un login o password válidos, muestra un mensaje de bienvenida.

Los logins y passwords están en un objeto `Properties`, definido en el método `init()`. Podríamos leer estos datos de un fichero, aunque por simplicidad aquí se definen como constantes de cadena.

Los datos de autentificación se envían codificados, y se emplea un objeto `sun.misc.BASE64Decoder` para descodificarlos y sacar el login y password.

```
package ejemplos;

import java.io.*;
import java.util.*;
import sun.misc.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletPassword extends HttpServlet
{
   // Conjunto de logins y passwords permitidos
   Properties datos = new Properties();

   // Metodo de inicializacion

   public void init()
   {
        datos.setProperty("usuario1", "password1");
        datos.setProperty("usuario2", "password2");
   }

   // Metodo para GET

   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
   throws ServletException, IOException
   {
        response.setContentType("text/html");

        // Comprobamos si hay cabecera
        // de autorizacion

        String autorizacion = request.getHeader("Authorization");

        if (autorizacion == null)
        {
                // Enviamos el codigo 401 y
                // la cabecera para autentificacion

                response.setStatus(response.SC_UNAUTHORIZED);
                response.setHeader("WWW-Authenticate",
                   "BASIC realm=\"privileged-few\"");
        }
        else
```

```
        {
                // Obtenemos los datos del usuario
                // y comparamos con los almacenados

                // Quitamos los 6 primeros caracteres
                // que indican tipo de autentificación
                // (BASIC)

                String datosUsuario =
                    autorizacion.substring(6).trim();

                BASE64Decoder dec = new BASE64Decoder();

                String usuarioPassword = new String
                   (dec.decodeBuffer(datosUsuario));

                int indice = usuarioPassword.indexOf(":");

                String usuario =
                   usuarioPassword.substring(0, indice);

                String password =
                   usuarioPassword.substring(indice + 1);

                String passwordReal =
                   datos.getProperty(usuario);

                if (passwordReal != null &&
                    passwordReal.equals(password))
                {
                        // Mensaje de bienvenida

                        PrintWriter out = response.getWriter();
                        out.println ("<HTML><BODY>");
                        out.println ("OK");
                        out.println ("</BODY></HTML>");
                } else {

                        // Pedir autentificacion

                        response.setStatus
                           (response.SC_UNAUTHORIZED);
                        response.setHeader
                           ("WWW-Authenticate",
                           "BASIC realm=\"privileged-few\"");
                }
        }
}

// Metodo para POST

public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
throws ServletException, IOException
```

```
   {
        doGet(request, response);
   }
}
```

Se puede probar cada ejemplo, respectivamente, con:

```
http://localhost:8080/appcab/inicioCabecerasPeticion.html
http://localhost:8080/appcab/inicioCabecerasRespuesta.html
http://localhost:8080/appcab/servlet/ejemplos.ServletPassword
```

Un ejemplo de login y password válidos para el tercer ejemplo es: login=`usuario1`, password=`password1`.