JAVA A PROFUNDIDAD

El objeto response

Tal como hemos comentado, cuando un servidor web responde (response) a una petición (request) HTTP, la respuesta típicamente consiste de una línea de status, algunas cabeceras de respuesta, una línea en blanco y el documento propiamente (la parte html por ejemplo). Una respuesta típica luce así:

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
   (Blank Line)
<!doctype ...>
<html>
   <head>...</head>
   <body>
      ...
   </body>
</html>
```

La línea de status consiste de la versión de HTTP (HTTP/1.1 en el ejemplo), un código de status (200 en el ejemplo), y un mensaje corto que corresponde al código de status (OK en el ejemplo).

La siguiente lista es un resumen de las cabeceras(headers) HTTP de respuesta más utilizadas. Estas cabeceras son enviadas de vuelta al navegador (browser) desde el servidor web y suelen ser usadas con frecuencia en la programación web. No confundir con las cabeceras de petición, estas son las respuestas del servidor al cliente.

Allow
Esta cabecera especifica los métodos de petición (GET, POST, etc) que el servidor soporta.

Cache-Control
Esta cabecera especifica las circunstancias en las cuales el documento de respuesta puede ser almacenado de manera segura. Puede tener valores **public, private** o **no-cache**. Public significa que el documento es cacheable, Private significa que que el documento es para un solo usuario y solo puede ser almacenado en caches privados (no compartidos) y no-cache significa que el documento no debe ser almacenado en caché nunca.

Connection
Esta cabecera instruye al navegador para determinar si ha de usar conexiones HTTP persistentes o no. El valor **close** instruye al navegador a no usar conexiones HTTP persistentes y **keepalive** significa que utiliza conexiones HTTP persistentes. Las conexiones persistentes, es la idea de utilizar la misma conexión TCP para enviar y recibir múltiples peticiones HTTP/respuestas, en lugar de abrir una nueva para cada par

JAVA A PROFUNDIDAD

petición/respuesta única. El uso de conexiones persistentes es muy importante para mejorar el rendimiento de HTTP. **Nota:** Para más información sobre conexiones persistentes, visite: https://es.wikibooks.org/wiki/HTTP/Conexiones/Conexiones_Persistentes
Content-Disposition Esta cabecera te permite solicitar que el navegador solicite al usuario que guarde la respuesta al disco en un archivo de un nombre dado.
Content-Language Esta cabecera indica el lenguaje con el cual el documento fue escrito. Por ejemplo, en, en-us, ru, es, etc.
Content-Length Esta cabecera indica el número de bytes en la respuesta del servidor. Esta información es necesaria solo si el navegador está usando conexiones HTTP persistentes (keep-alive).
Content-Type Esta cabecera proporciona el tipo MIME (Multipurpose Internet Mail Extension) del documento de respuesta.
Expires Esta cabecera especifica el tiempo en el cual el contenido podrá ser considerado como fuera de tiempo y, por lo tanto, ya no puede ser almacenado en caché.
Last-Modified Esta cabecera indica cuándo se modificó por última vez el documento. El cliente puede entonces almacenar en caché el documento y proporcionar una fecha mediante un encabezado de solicitud if-Modified-Since en solicitudes posteriores.
Location Esta cabecera debería ser incluida con todas las respuestas que tienen un código de status HTTP 300s. Esto notifica al navegador de la dirección del documento. El navegador se reconecta automáticamente a esta ubicación y recupera el nuevo documento.
Refresh Esta cabecera indica cuán pronto el navegador debe solicitar una página actualizada. Puede especificar el tiempo en cantidad de segundos después de lo cual se actualizará una página.

JAVA A PROFUNDIDAD

Retry-After Esta cabecera puede ser usada en conjunto con una respuesta 503 (Service Unavailable) para decir al cliente cuán pronto puede repetirse la petición.
Set-Cookie Esta cabecera especifica una cookie asociado con esta página.

Métodos para manipular cabeceras HTTP en la respuesta del servidor.

Los métodos que siguen a continuación en la lista, pueden ser utilizados para manejar las cabeceras HTTP de respuesta en nuestros Servlets. Estos métodos están disponibles en los objetos HttpServletResponse.

String encodeRedirectURL(String url) Codifica la URL especificada para usar en el método sendRedirect o, si el encoding no es necesario, devuelve la URL sin cambios.
String encodeURL(String url) Codifica la URL especificada incluyendo la sesión ID en ella, si la codificación no es necesaria, devuelve la URL sin cambios.
boolean containsHeader(String name) Devuelve un booleano indicando si existe un determinado header en el objeto response que ha sido enviado.
boolean isCommitted() Devuelve un boolean indicando si la respuesta ha sido entregada.
void addCookie (Cookie cookie) Añade la cookie especificada a la respuesta del servidor.
void addDateHeader (String name, long date) Añade una cabecera al response con un nombre dado y una fecha.

JAVA A PROFUNDIDAD

void addHeader(String name, String value) Añade un header a la respuesta con un nombre y un valor dado.
void addIntHeader(String name, int value) Añade un header al objeto response con un nombre dado y un valor entero.
void flushBuffer() Obliga a cualquier contenido en el buffer a escribirse en el cliente.
void reset() Borra todos los datos que existen en el búfer, así como el código de estado y las cabeceras.
void resetBuffer() Borra el contenido del búfer subyacente en la respuesta, pero sin borrar los encabezados o el código de estado.
void sendError(int sc) Envía un error de respuesta al cliente usando un código de status específico y limpiando el buffer.
void sendError(int sc, String msg) Envía un error de respuesta al cliente usando el estatus especificado y un mensaje.
void sendRedirect(String location) Envía una respuesta de redirección temporal al cliente utilizando la URL de ubicación de redireccionamiento especificada.
void setBufferSize(int size) Especifica el tamaño de buffer preferido para el cuerpo de la respuesta.
void setCharacterEncoding(String charset) Especifica el encoding de caracteres (MIME charset) de la respuesta que se envía al cliente, por ejemplo, a UTF-8

JAVA A PROFUNDIDAD

void setContentLength(int len) Especifica la longitud del cuerpo del contenido de la respuesta, este método establece el valor de la cabecera http **Content-Length.**
void setContentType(String type) Determina el tipo de contenido de la respuesta que se envía al cliente.
void setDateHeader(String name, long date) Especifica un header de respuesta con un nombre dado y un valor de fecha.
void setHeader(String name, String value) Especifica un header de respuesta con un nombre dado y un valor.
void setIntHeader(String name, int value) Especifica un header en la respuesta con un nombre dado y un valor integer.
void setLocale(Locale loc) Especifica la localización de la respuesta, si la respuesta no ha sido entregada todavía.
void setStatus(int sc) Especifica el código de estatus de esta respuesta.

Código de ejemplo

Anteriormente en otros ejemplos durante el curso, hemos utilizado el método setContentType(), sobretodo para especificar que un contenido será html. En el siguiente ejemplo, también usaremos ese método pero además, usaremos el método **setIntHeader()** para establecer una cabecera(header) refresh. Esta cabecera le indicará al cliente(navegador) cada cuántos segundos se debe "refrescar".

```
1.  // Importamos las librerías requeridas de Java
2.  import java.io.*;
3.  import javax.servlet.*;
4.  import javax.servlet.http.*;
5.  import java.util.*;
6.
7.  // Se hereda de la clase HttpServlet
8.  public class Refresh extends HttpServlet {
9.
10.    // Método para manipular el método GET
```

JAVA A PROFUNDIDAD

```
11.    public void doGet(HttpServletRequest request, HttpServletResponse response)
12.       throws ServletException, IOException {
13.
14.       // Establece el header refresh para que se autocargue cada 5 segundos
15.       response.setIntHeader("Refresh", 5);
16.
17.       // Determina el tipo de contenido de la respuesta.
18.       response.setContentType("text/html");
19.
20.       // Obtenemos la fecha actual
21.       Calendar calendario = new GregorianCalendar();
22.       String am_pm;
23.       int hora = calendario.get(Calendar.HOUR);
24.       int minuto = calendario.get(Calendar.MINUTE);
25.       int segundo = calendario.get(Calendar.SECOND);
26.
27.       if(calendario.get(Calendar.AM_PM) == 0)
28.          am_pm = "AM";
29.       else
30.          am_pm = "PM";
31.
32.       String CT = hora+":"+ minuto +":"+ segundo +" "+ am_pm;
33.
34.       PrintWriter out = response.getWriter();
35.       String title = "Auto Refresh Header Setting";
36.       String docType =
37.          "<!DOCTYPE html>";
38.
39.       out.println(docType +
40.          "<html>\n" +
41.          "<head><title>" + title + "</title></head>\n"+
42.          "<body bgcolor = \"#f0f0f0\">\n" +
43.          "<h1 align = \"center\">" + title + "</h1>\n" +
44.          "<p>Current Time is: " + CT + "</p>\n"
45.       );
46.    }
47.
48.    // Method to handle POST method request.
49.    public void doPost(HttpServletRequest request, HttpServletResponse response)
50.       throws ServletException, IOException {
51.
52.       doGet(request, response);
53.    }
54. }
```