élogos

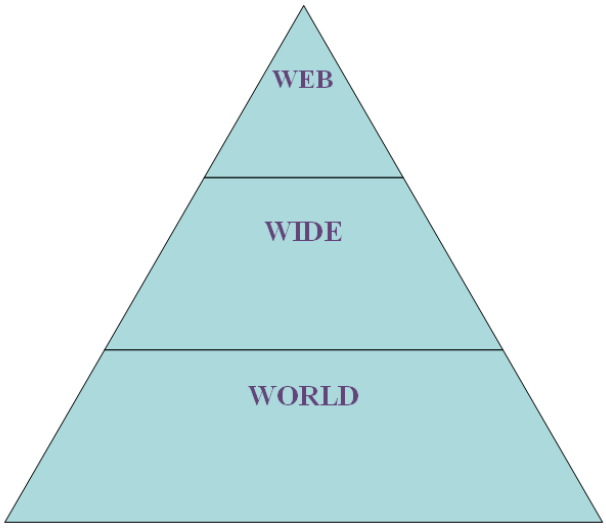
# ARQUITECTURA DE APLICACIONES JAVA EE

## El protocolo HTTP

élogos

# El protocolo HTTP.

### WORLD WIDE WEB




El protocolo HTTP es la base de la Worl Wide Web (WWW). El WWW fue creado en 1980 por Tim Berners Lee, un trabajador del CERN (Centre Européen de Rechenche Nucleaire), como un sistema para el intercambio de documentos.

Las primeras pruebas fueron realizadas con el navegador Mosaic, pero pronto fue adoptado por otros fabricantes como Netscape y Microsoft.

### Conceptos básicos de WWW:

La WWW se basa en las siguientes tecnologías:

HTTP (HiperText Transfer Protocol), es el protocolo de comunicación a nivel de Sockets entre el cliente (navegador) y el servidorWeb que sirve los documentos.

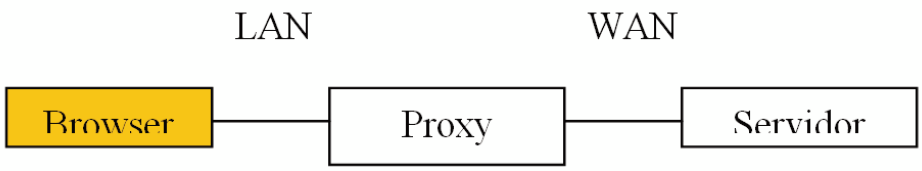
HTML (HyperText MakeUp Language), es un lenguaje de marcas que permite formatear un documento. Todos los navegadores reconocen este lenguaje.

URL (Uniform Resource Locator), es una forma estándar de dar un nombre único de un recurso en Internet (Páginas Web, programas, News, e-mail,...)

Existen muchos servidores Web:

Apache Web server
Lotus Domino
Internet Information Server (ASP)
IBM Http Server

Proxy HTTP




Proxy HTTP es un programa intermedio que se pone entre el Browser y el Servidor Web de tal forma que las peticiones del browser pasarán primeramente al Proxy y no directamente al Servidor Web. El Proxy se encargará de hacer las peticione al Servidor Web. Con este esquema se consiguen dos ventajas:

El acceso es más rápido ya que realiza caché de las peticiones.

Puede actuar como un cortafuegos. Limitando respecto al usuario y al clave, respecto la dirección IP e incluso el acceso a determinadas páginas Web.

Robots de búsqueda, cuawler (gateador), spider (araña), bot (robot), son programas que se entretienen bajándose las paginas web y analizando el contenido de las mismas. Esta información quedará almacenada en una base de datos. Servirá sobre todo para buscadores.

**El Protocolo HTTP**

Documentación del protocolo:

| | |
|---|---|
| RFC 1945 | HTTP 1.0 |
| RFC 2068 | HTTP 1.1 |
| RFC 1738 | URL |

URL (Uniform Resource Locator):

Ejemplo: http://www.sony.uk/~john/index.html
<Protocolo>://<Host>:<Puerto>/<Recurso>?<Parámetros>

<Protocolo> especifica el protocolo utilizado. En el caso de escribir:

mailto: fernando123@studio.com

mailto significa que se utiliza el servicio de mail para acceder al recurso: fernando123@studio.com

Esto no es lo mismo que enviar un e-mail. Solo es una forma de dar nombres, p.e., mailto: news: entre otros.

<Puerto> se especifica cuando se va a utilizar un puerto distinto del estándar.
http:// utiliza el puerto 80 como estándar.
ftp:// utiliza el puerto 21 como estándar.

URI (Uniform Resource Identifier)

Es la parte de la consulta que recibe el servidor web.
Siguiendo el ejemplo anterior, es la siguiente parte: /~john/index.html

www.sony.uk
|
DNS ——- IP

Tradicionalmente se le llama www al servidor web, pero realmente las tres w son una dirección IP que podría asociarse con un nombre distinto de servidor, p.ej., neptuno.uam.es

De acuerdo con la RFC 1738.

¿? > %3F
= > %3D

Todos los códigos mayores de 120 hay que codificarlos.

HTTP es un protocolo donde se mandan códigos ASCII, este protocolo se conecta al puerto 80.

Los servidores web con HTTP son stateless, esto significa que el servidor web no mantiene la cuenta del cliente una vez que nos proporciona la página web pedida.

ISO - 8859 - 1

| | |
|---|---|
| GET | > Sirve para pedir una página/fichero al servidor. |
| HEAD | > Envía los datos de la cabecera del fichero (sólo los datos de control). |
| POST | > Envía los datos del fichero. |
| PUT | > Subir un fichero al servidor. |
| DELETE | > Borra un fichero del servidor. |
| TRACE | > Sirve para la depuración, para hacer pruebas. |

La forma de ejecutar estos comandos sería:

GET /pub/docs/rfc-index.html HTTP/1.0
URI (=<Recurso>+<Parámetros>) Versión de HTTP que entiende el browser

POST /cgi-bin/register.pl HTTP/1.0
(.pl = perl; extensión de una página elaborada con lenguaje perl)

Content-Type: application/x-www-form-urlencoded
Content-Length: 65

Los parámetros son pares clave = valor separados por ampersand.
Username = Fernando + Lopez & email = fernando%40centrostudio.com

Browser > Servidor Web > Register.pl

| <VersiónHTTP> | <Código Status> | <Texto Libre> |
|---|---|---|
| HTTP/1.0 | 200 | OK |
| HTTP/1.0 | 404 | NOT FOUND |

**Indicar el formato de los datos enviados**

Accept: sirve para que el cliente diga al servidor qué tipo de datos entiende.
Content-Type: sirve para que el servidor nos diga qué tipo de dato nos ha enviado.

Ejemplo:

Tipos MIME aceptados por el browser del cliente, especificando tipo y subtipo.

Accept: text/plain, text/html, text/*, image/jpeg, image/gif, image/*, */*.
Prefiere texto plano, de no ser así entonces html y así sucesivamente… finalmente acepta cualquier tipo y subtipo (*/*).

Content-Type: text/html.

Cuando el servidor no encuentra en el Accept el tipo especificado en el Content-Type no lo envía al browser del cliente.

**Selección del juego de caracteres.**

ISO – 8859 – 1
Accept-Charset: S-ASCII, ISO – 8859 – 1, UTF – 8.
Content-Type: ext/plain; charset=US-ASCII

**Indicar el idioma.**

Accept-Language: es,en.
Content-Language: en.

Compresión de documentos

Accept-Encoding: gzip, compress (browser)

Se envía esta cabecera desde el browser al servidor para que sepa que es capaz de enteder los formatos gzip y compress, esto quiere decir, ficheros comprimidos con estos formatos.

Content-Encoding: gzip (servidor)

En caso de que el servidor pueda comprimir y enviarlo comprimido en la cabecera Content-Encoding indica el servidor que formato de compresión ha utilizado.
Tamaño del documento enviado

Content-Length: 8135

Es una cabecera que envía el servidor al cliente para indicarle el tamaño del documento en bytes.

Indicar el fabricante del browser y del servidor

User-Agent:

Esta cabecera le sirve al browser para indicar el fabricante del browser. Sobre todo sirve para realizar estadísticas.

Sever:

Sirve para indicar la marca del servidor. Normalmente los servidores no utilizan esta cabecera para evitar los fallos que puedan llegar a tener los servidores.

Indicar la procedencia

From: Fernando García <fgarcia@studio.es>

Sirve para indicarle al servidor la cuenta del usuario que se conecta. Esta cabecera no la utilizan los servidores para evitar que se sepa la cuenta de correo.

Referred: http://www.yahoo.com/search=programacion

Esta cabecera sirve para indicar que la consulta que ha llegado al servidor es gracias a que el usuario ha pinchado en un enlace que estaba en otro servidor. Se utiliza mucho para fines estadísticos.

**Fechas de caducidad y cachés**

Expires: Sun 8 Mar 2008 10:00:00 GMT

Esta cabecera que manda el servidor al browser para indicarle que el documento no caduca hasta la fecha que indica en la cabecera. Le sirve al browser para guardar la página en cache (disco duro) junto a su fecha de caducidad, de tal forma que antes de conectarse al servidor comprobará en el caché la fecha de caducidad y recogerá el documento del disco duro si no se ha superado la fecha de caducidad.

Surge el problema de que la hora del servidor y el cliente no estén sincronizadas. Para que se pueda controlar esto se tiene esta cabecera:

Date: Sun 8 Mar 2008 09:10:46 GMT

Con esta cabecera el browser sabe la fecha y la hora del servidor.

If Modified-Since: Wed 15 Oct 2008 21:30:07 GMT

Sirve para que el browser le diga al servidor que envíe una página solamente si ha cambiado desde la última fecha que se envió la página. Si el documento ha cambiado el servidor manda un código 200 junto con el documento. Si el documento no ha cambiado el servidor manda un código 304 (Not Modified) y además no envía el documento

Last Modified: Wed 15 Oct 2008 21:30:07 GMT

Es una cabecera que envía el Servidor al Cliente para indicarle la última fecha de modificación del fichero. No siempre el servidor mandará esta cabecera.

**Autentificación de usuarios**

En HTTP existe la posibilidad de que el servidor autentifique a los usuarios antes de entregarles un documento. Lo que hace el servidor es enviar el código 401 (Unauthorized) al browser, junto a este código manda la siguiente cabecera:

www-Authenticate:Basic realn=”Alumnos”
———- ——————-
1 2

1.- Método de autentificación
2.- Reino

Cuando el browser recibe esta cabecera con este código saca un diálogo pidiendo el user y el password. Tras escribir el usuario el user y el password el browser manda la petición con la siguiente cabecera:

Authorization: Basic GE48p8H23772
_____
user y password

En esta cabecera enviará el user y la password codificado en BASE64 que el usuario ha introducido en el browser. Si el user y la password son correctos el servidor enviará el documento con el código 200. Si no son correctos el servidor enviará 401. También puede enviar el código 403 indicando que el user y la password son correctos pero la dirección IP no es correcta para conectarse al servidor.

**Las cookies**

Es una de las formas que se han inventado para solucionar el problema de que los servidores web sean stateless (no llevan la cuenta de los clientes conectados). Más tarde cuando se quiso utilizar este protocolo llevando la cuenta de usuario se comprobó que no se podía hacer y una de las formas de solucionar el problema fueron las cookies.

Las cookies sirven para que cuando es la primera vez que se conectan a un servidor, el servidor da un código de tal forma que las próximas veces que se conecten al servidor el browser dará esta cookie al servidor para autentificarse

Con las cookies se pueden hacer estadísticas a partir de las costumbres del usuario a la hora de navegar por los documentos disponibles en el servidor.

La cookie, en un principio, es un código pero tal vez el sitio web tenga un sitio para registrarse de tal forma que la siguiente vez que se conecte sabrán quiénes somos.

Set Cookie:<Nombre>=<Valor>

El servidor manda esta cabecera al browser, para colocar una cookie en el browser.

Cookie:<Nombre>=<Valor>

Cada vez que el browser se conecta al Servidor, el browser enviará esta cabecera.

Cookies en Java:

Se implementan con el objeto Cookie.

```
Cookie c;
c = new Cookie (“nombre”,”valor”);
resp.addCookie (c);
```

**Conexiones persistentes**

Se introdujo en HTTP versión 1.1 y lo que permite es que el servidor no cierre el Socket una vez que envía el documento al browser, con lo que se puede aprovechar el mismo Socket para pedir más documentos. Para utilizar conexiones persistentes el browser y el servidor deben tener HTTP 1.1

Connection: Keep-Alive

El browser lo envía al servidor. Si el servidor no entiende esta cabecera cerraría el Socket tras enviar el documento. Si lo entiende el Socket no se cierra y el browser manda más peticiones por el Socket al servidor.

En la última petición que realiza el browser se manda la siguiente cabecera para cerrar el socket:

Connection: close

En las conexiones persistentes el único problema que existe es cuando empieza y cuando termina el siguiente documento. La forma de saber cuando termina un documento en las conexiones persistentes es utilizar la siguiente cabecera:

Content-length:650

Esta cabecera indica el tamaño del documento en bytes, el browser contará todos los bytes del documento hasta detectar el último byte.

**Características de los Servidores Web**

Un servidor web es un programa que pone a disposición de los usuarios un trozo del disco duro del servidor donde se van a poner los documentos que se van a ver desde el servidor web.

La forma de acceder a estos documentos será la siguiente:

http://www.vertodo.com/index.html

http://www.vertodo.com/productos/ceniceros.html

```
__ _________. _____________.
 1        2                  3
```

1.- Protocolo.
2.- Nombre DNS del servidor.
3.- URI (Ruta dentro del servidor)

Directorios Virtuales

Se pueden introducir directorios que físicamente no están dentro de la jerarquía de servidor web mediante directorios virtuales.

```
c:\        nuevo
                rebajas.html

           index.html                          actualidad —> c:\nuevo
           productos                           ______-      ______-
                ceniceros.html                   alias        Directorio virtual
           ventas

http://www.vendotodo.com/actualidad/rebajas.html
                          _______
                             alias
```

**Restricciones de acceso**

Los Servidores Web implementan las restricciones a dos niveles:

1.- A nivel de directorio, donde a un directorio se le puede asignar un reino y posteriormente se le indica qué usuarios pueden acceder al reino (realm).

2.- En función de la dirección IP de la que procede la consulta de tal forma que se puede aceptar o rechazar la consulta a partir de la dirección IP de la que procede la consulta.

En vez de utilizar la dirección del IP se puede utilizar el dominio como filtro para aceptar o desechar las consultas. Consiste en que el servidor web pregunta a un servidor DNS por el nombre que le corresponde a una dirección IP, este proceso se denomina Resolución Inversa.

Desvío de la consulta

Con el mantenimiento del Servidor Web se puede llegar a reestructurar toda la información incluida en él. Esto va a producir problemas si los usuarios tienen enlaces a nuestros documentos ya que habrán cambiado de sitio. Existen dos soluciones:

Crear documentos de avisos para los usuarios. Es una solución tediosa ya habrá que mantener nuevos documentos y además ocupan sitio en el servidor web.

Configurar en el servidor web de tal forma que si se pregunta por una página obsoleta sea él mismo el que redireccione al cliente a la nueva página. De tal forma que el servidor mandará al cliente el siguiente mensaje:

HTTP / 1.0 302 Document moved
Location. http://www.ya.com/nuevapos/datos.html

Tiene varias ventajas:

- Ocupa menos espacio
- Los buscadores pueden periódicamente actualizar sus bases de datos.

Los archivos de LOG

Se puede llevar la cuenta de las conexiones que ha habido mediante archivos. LOG, con el fin de sacar estadísticas.

Para almacenar esta información existen muchos formatos. Uno de los más utilizados es el Common Log File (clf), es el más utilizado ya que fue el primero en utilizarse por el primer servidor web denominado NCSA. Existe otro método muy utilizado denominado Extended Log File realizado por el consorcio 3W Consortium. Se diferencia con el primero en que los campos del fichero son variables.

**Los CGI (Common Gateway Interface)**

Los servidores web en un principio servían para servir documentos web a partir de peticiones realizadas desde el browser del cliente.

Posteriormente los servidores web para que pudieran realizar más tareas se crearon los Programas CGI. De tal forma que cuando el browser realizaba una petición, está petición se mandaba a un programa CGI. Este programa generaba un documento web para que el servidor pudiera mandárselo de nuevo al browser.


Entre el Browser y el servidor Web el protocolo utilizado es HTTP mientras que la comunicación entre el servidor Web y el programa CGI está regida por el protocolo CGI.

El programa CGI está realizado en cualquier lenguaje, aunque normalmente se utiliza el C y PERL pero se pueden utilizar otros lenguajes como el Visual Basic, etc.

Envío de comandos del Servidor Web al programa CGI

Para mandar los comandos del Servidor Web al programa CGI se van a utilizar variables de entorno. Una variable de entorno es una variable del S.O. a la cual puede acceder cualquier programa.

La principal utilidad que tienen los CGI es enviar datos introducidos mediante un formulario a un servidor. Pasos para realizarlo:

1.- Crear página HTML:

```
<HTML>
  <BODY>
   <FORM method="get" action="/cgi-bin/register.pl">
     Nombre: <INPUT type="text" name="nombreusuario" size="40">
     Direccion: <INPUT type="text" name="direccion" size="40">
     <INPUT type="submit">
   </FORM>
  </BODY>
</HTML>
```

Envío de los datos al servidor mediante esta consulta:

```
GET /cgi-bin/register.pl?nombreusuario=fernando+lopez+&direccion=C+Abajo HTTP /1.0
    ————————————— ——————————————————— ————————————
          CGI                              PARAMETROS
    SERVER_NAME                      QUERY_STRING
                    (Variables de Entorno)
```

La limitación de la URI es de 240 caracteres, para solucionar este problema existe una forma de enviar los datos con el comando POST.

```
<HTML>
 <BODY>
  <FORM method="post" action="/cgi-bin/register.pl">
   Nombre: <INPUT type="text" name="nombreusuario" size="40">
   Direccion: <INPUT type="text" name="direccion" size="40">
   <INPUT type="submit">
  </FORM>
 </BODY>
</HTML>
```

Envío de los datos al servidor mediante esta consulta:

```
POST /cgi-bin/register.pl HTTP /1.0
Content-Type: application/z-www-form-urlencoded
Content-length: 65

nombreusuario=fernando+lopez&direccion=C+Abajo
  ———————————————————————————————
                PARAMETROS
```

La cabecera Content-Type la recoge en la variable de entorno CONTENT_TYPE.

La cabecera Content-length es recogida en la variable de entorno CONTENT_LENGTH

Para recoger los parámetros en CGI en este caso se recogerá con la entrada estándar, en el caso de Java será System.in.

Tras realizar el tratamiento habrá que devolver los datos al servidor web desde el programa CGI, la forma será la de escribir los datos en la salida estándar. En el caso del Java será System.out.

**Las URL**

La Clase URL

Sirve para representar los recursos.

Se encuentra del paquete java.net:

```
import java.net.*;
```

Constructor:

```
URL (String utl) throws MalformedURLException
```

Para crear el objeto se realizaría de la siguiente forma:

```
URL url;

try {
        url = new URL("http://www.maqfac.com/v.html");
}
catch (MalformedURLException ex) {
        System.out.println(ex);
}
```

Además URL tiene más constructores:

```
URL (String protocol, String host, String url)
        throws MalformedURLException
```

Nos permite dar las diferentes partes de la URL: protocolo, host y nombre de la URL.

Se pueden utilizar la URL para bajar una página Web de un servidor Web, para ello se dispone del siguiente método:

```
InputStream <URL>.openStream() throws IOException
```

```
java PidePagina http://mipc.centrostudio.com/fernando/Default.html
```