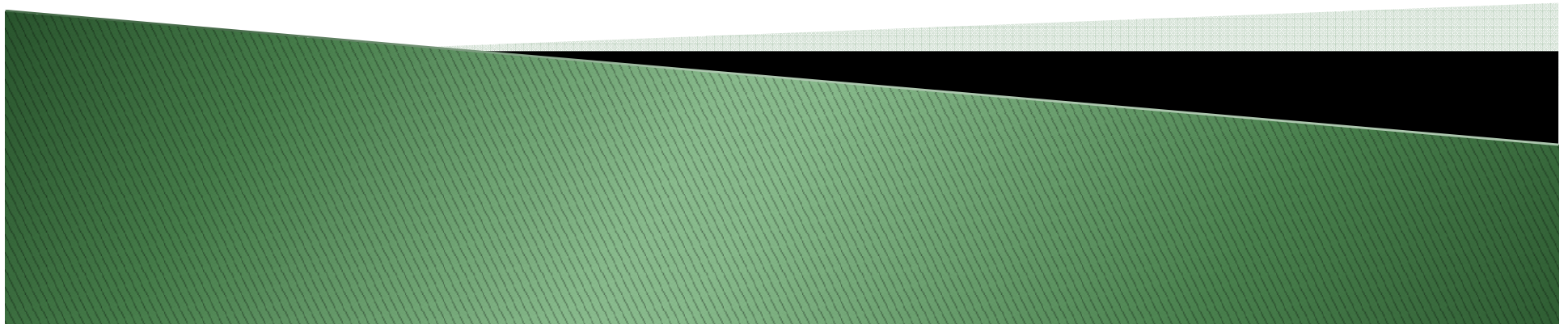


# Unidad 4

## WWW y protocolo HTTP

Despliegue de aplicaciones web  
Curso 2013



# Índice

---

- ▶ WWW.
- ▶ W3C y estándares.
- ▶ Páginas, sitios y aplicaciones web.
- ▶ Componentes y funcionamiento.
- ▶ Clientes Web (Navegadores).
- ▶ URLs y URIs.
- ▶ Servidores Web.

# Índice

---

- ▶ Protocolo HTTP
  - Introducción.
  - Versiones.
  - Funcionamiento.
  - Mensajes HTTP.
  - Métodos de petición.
  - Cabeceras.
  - Códigos de estado y error.
  - Negociación de contenidos. Tipos MIME.

# Índice

---

- *Cookies.*
- Autenticación.
- Sesiones.
- Otras características.
- ▶ Bibliografía.

# WWW (*World Wide Web*)

---

- ▶ Servicio de distribución de información.



- ▶ Acceso a millones de recursos electrónicos y aplicaciones distribuidos en servidores por todo Internet.
- ▶ Identificados y localizados por direcciones (URIs o URLs).
- ▶ Conectados entre sí a través de hiperenlaces (o hipervínculos).

# W3C y estándares web

## ► WWW

- Desarrollada por el CERN (Centro Europeo de Investigación Nuclear) en 1989.

## ► W3C (*World Wide Web Consortium*)

- <http://www.w3.org/>
- <http://www.w3c.es/>
- Comunidad internacional.
- Controla el desarrollo de la WWW.
- Desarrolla estándares web, por ejemplo XHTML, CSS y XML.



**Accesibilidad**

**CSS**

Estándares Web

Independencia de Dispositivo

Internacionalización

**Interacción Multimodal**

Linked Data

Política de Patentes del W3C

Privacidad y P3P

Seguridad

**Servicios Web**

Tecnologías Multimedia

**Tecnologías XML**

**Web Móvil**

**Web Semántica**

**XForms**

**XHTML**

# Páginas, sitios y aplicaciones Web

---

## ► Página web

- Documento hipermedia o conjunto de información electrónica relacionada (texto, audio, imágenes, video, etc.) que normalmente contiene hiperenlaces a otras paginas web o recursos.
- Escrita en lenguajes que son interpretados y/o ejecutados por los navegadores (*XHTML, CSS, Java Script, Flash, ...*)
- Contenidos estático y dinámico.

# Páginas, sitios y aplicaciones Web

---

## ► Sitio web

- Conjunto de paginas web relacionadas y accesibles a partir de un mismo nombre de dominio DNS.
- El conjunto de sitios web de Internet constituyen la WWW.



# Páginas, sitios y aplicaciones Web

---

## ► Aplicación web

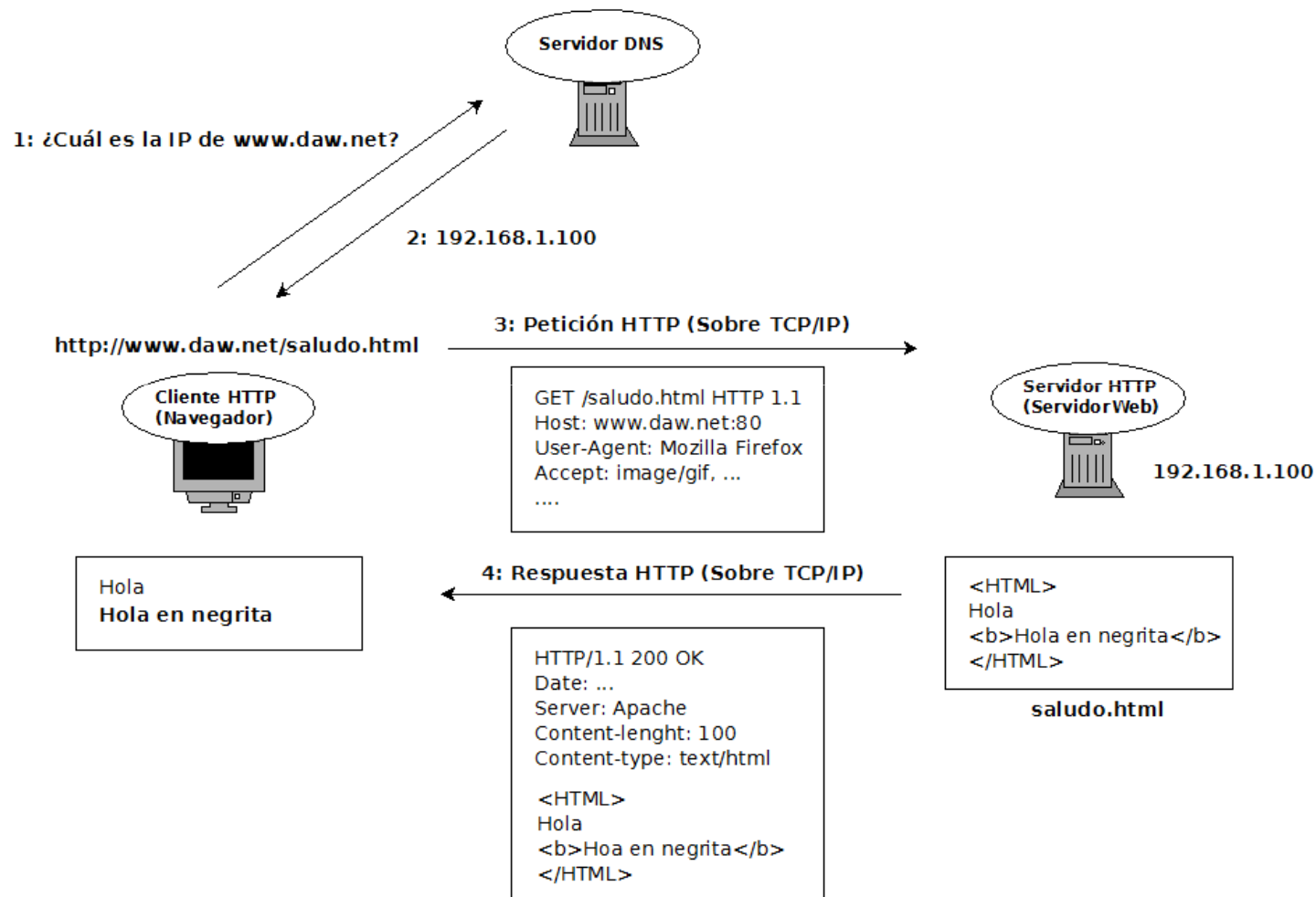
- Aplicación distribuida cuya interfaz de usuario es accesible desde un navegador web.
- El usuario interactúa con un navegador que accede a los servicios y recursos que ofrece un servidor web (Ejemplo: buscador, una tienda electrónica, un cliente de correo web,...).
- Ejemplos: *Gmail, Ebay, Facebook, ...*

# Componentes y funcionamiento

---

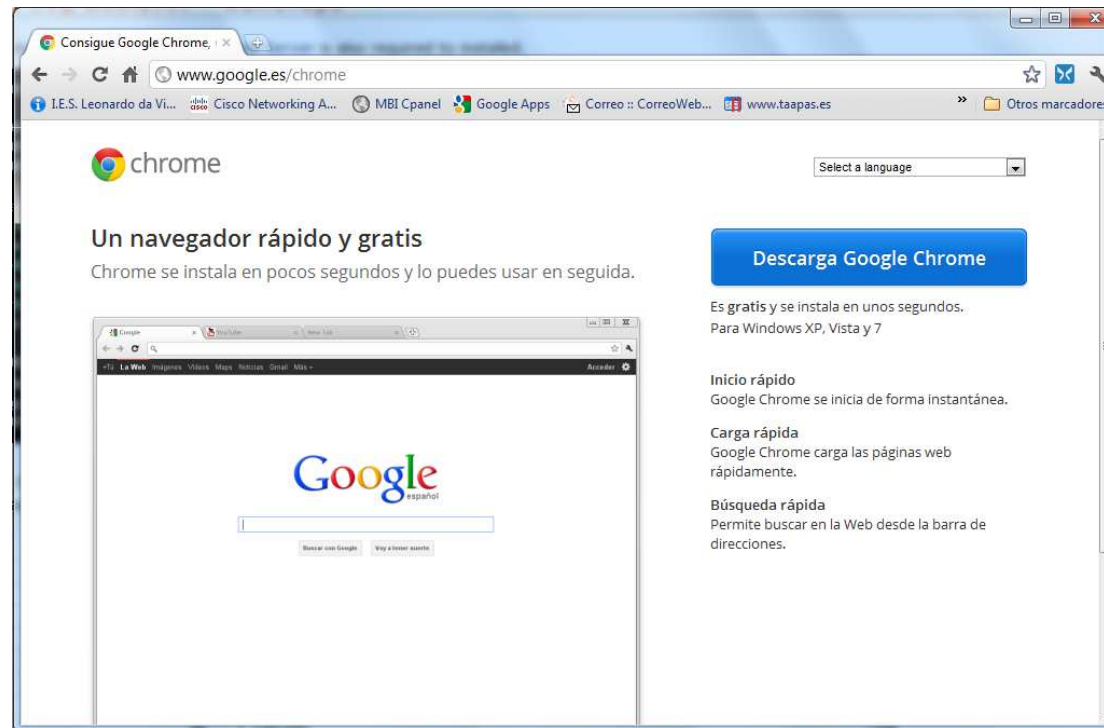
- ▶ Recursos (documentos, vídeos, imágenes, etc.) conectados por hiperenlaces.
- ▶ Clientes web (clientes HTTP o navegadores).
- ▶ Servidores web (o servidores HTTP).
- ▶ Nombres y direcciones (URIs y URLs).
- ▶ Protocolo HTTP.
- ▶ Tecnologías web (XHTML, CSS, XML, Ajax, XPath, etc.).

# Componentes y funcionamiento



# Cientes Web (Navegadores)

- ▶ Programas con los que interactúa el usuario.
- ▶ URIs (o URLs) para acceder a recursos disponibles en la red.



# Clientes Web (Navegadores)

---

- ▶ Clientes de diferentes protocolos.
- ▶ Su función principal es ejercer como clientes HTTP.
- ▶ Mantienen una memoria cache
  - Direcciones a las que han accedido (historial)
  - Recursos procesados
  - Contraseñas introducidas por el usuario en las aplicaciones,
  - ...

# Cientes Web (Navegadores)

---

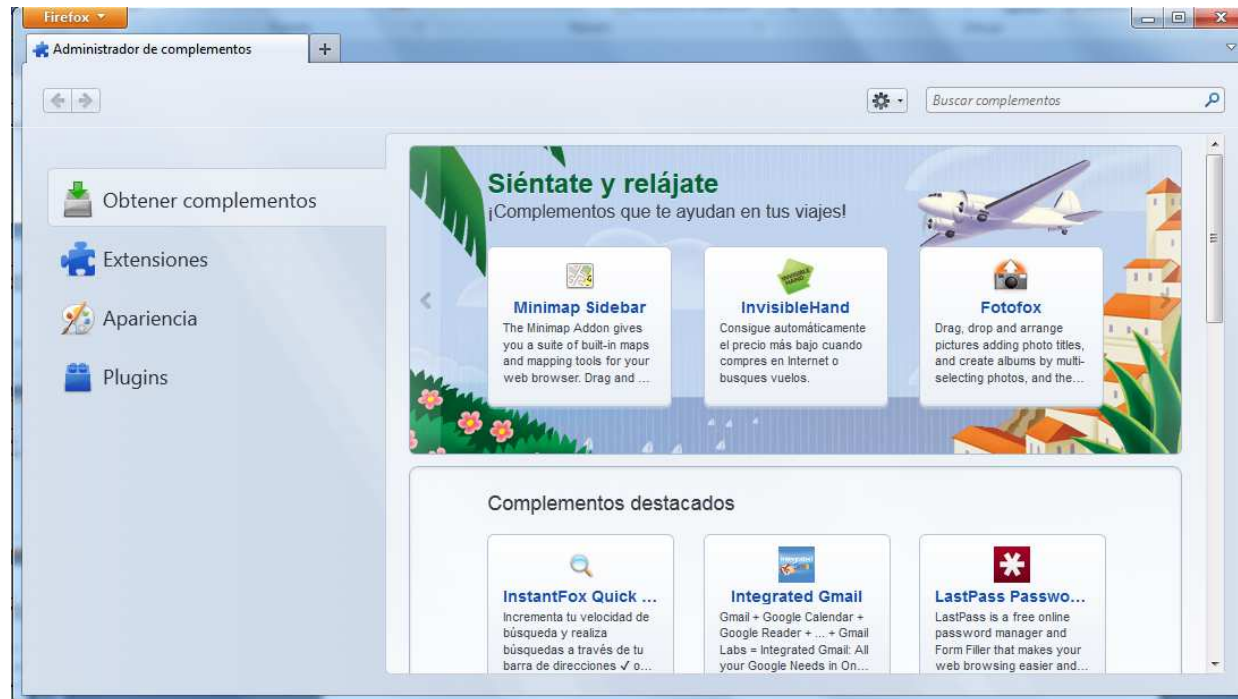
## ► Ejemplos

- *Internet Explorer.*
- *Mozilla Firefox.*
- *Google Chrome .*
- *Chromiun .*
- *Safari.*
- *Opera .*
- Navegadores en modo texto: *Lynx, Links*, etc.
- ...



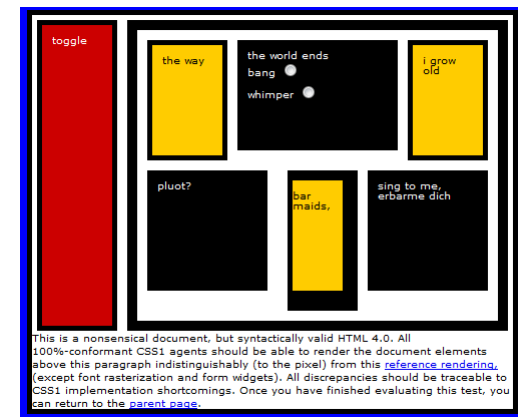
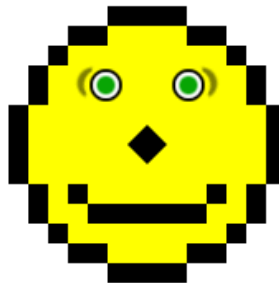
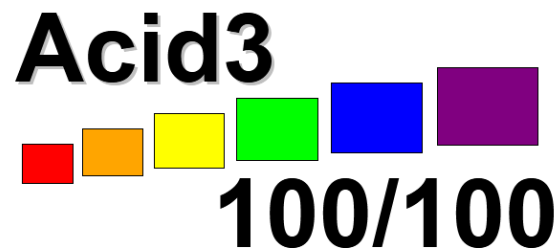
# Cientes Web (Navegadores)

- ▶ Permiten ampliar su funcionalidad con la instalación de plantillas, idiomas, extensiones y complementos



# Cientes Web (Navegadores)

- ▶ Los navegadores, en determinados aspectos no cumplen completamente los estándares.
- ▶ *Acid Tests*
  - <http://www.acidtests.org>.
  - Para evaluar como cumplen los navegadores con los
  - estándares de la W3C .





# URLs y URIs

---

## ► URL (*Uniform Resource Locator*)

- Usado para identificar un recurso en la Web,
- Sintaxis:
  - protocolo://host[:puerto]/ruta-y-nombre.
- Ejemplos
  - <http://192.168.1.100/saludos.html>
  - <http://www.daw.net:8080/datos/practica1.pdf>
  - <ftp://ftp.rediris.es>
  - ...

# URLs y URIs

---

## ► URI (*Uniform Resource Identifier*)

- Mas general que una URL.
- Permiten identificar una parte dentro de un recurso.
- Las URLs son tipos de URIs.
- Sintaxis para HTTP
  - <http://host:puerto/ruta?parametros-petición#parte>
    - parametros-petición
      - Pares de nombre=valor separados por '&'.
- Ejemplo
  - <http://obelix.dae.es/buscarLibros.php?id=2&tema=Historia>

# Servidores Web

---

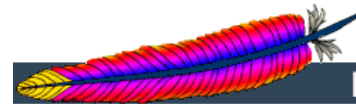
- ▶ **Servidores web o servidores HTTP.**
  - Atienden peticiones HTTP.
  - Procesan e interpretan código escrito en diferentes lenguajes.
  - Envían a los clientes los recursos solicitados.
- ▶ **Múltiples opciones de configuración.**
- ▶ **Arquitectura modular que permite ampliar o quitar funcionalidades fácilmente.**
- ▶ **Peticiones HTTP en el puerto 80/TCP.**

# Servidores Web

---

## ► Ejemplos

- *Apache HTTP server.*
- *IIS de Microsoft*
- *Nginx*
- *Lighttpd*
- *Cherokee*
- ...



**Apache**

HTTP SERVER PROJECT

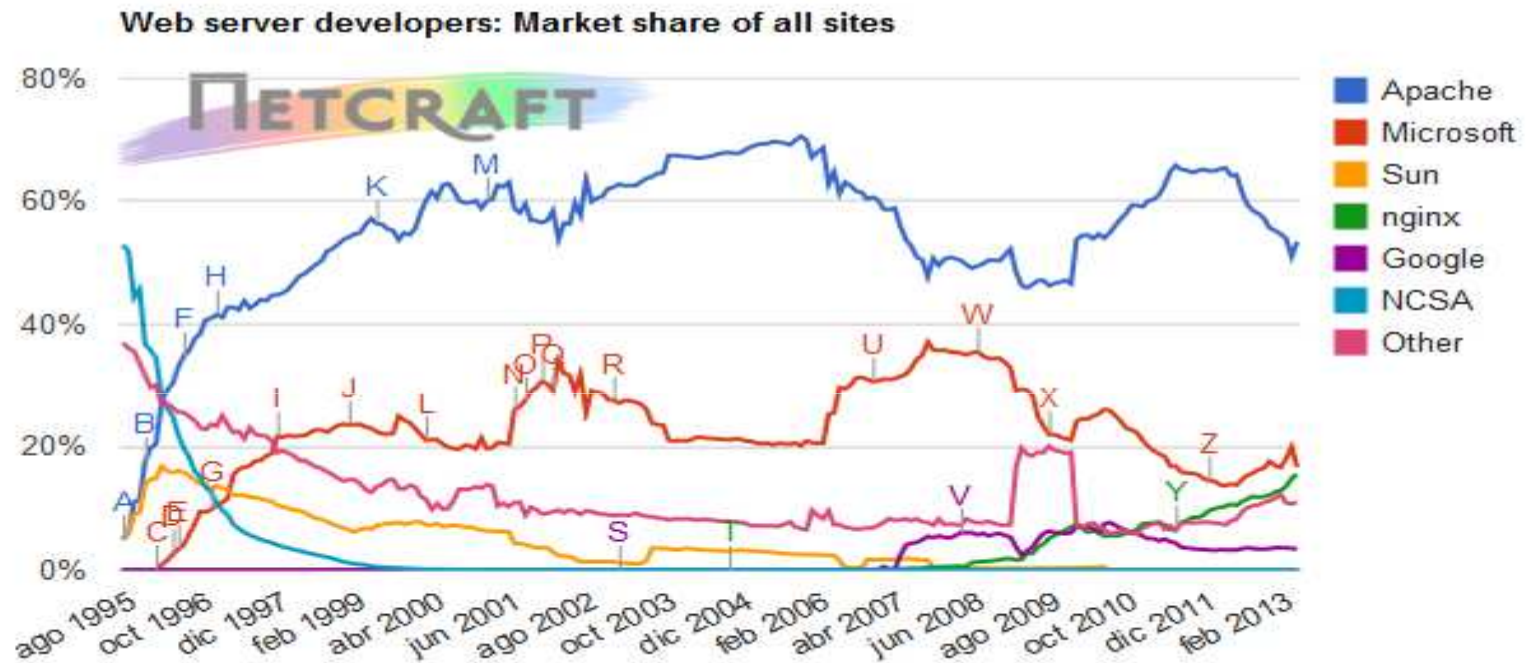


**NGINX**<sup>TM</sup>



**cherokee**

# Servidores Web



Developer	April 2013	Percent	May 2013	Percent	Change
Apache	331,112,893	51.01%	359,441,468	53.42%	2.41
Microsoft	129,516,421	19.95%	112,303,412	16.69%	-3.26
nginx	96,115,847	14.81%	104,411,087	15.52%	0.71
Google	22,707,568	3.50%	23,029,260	3.42%	-0.08

# Protocolo HTTP

## Introducción

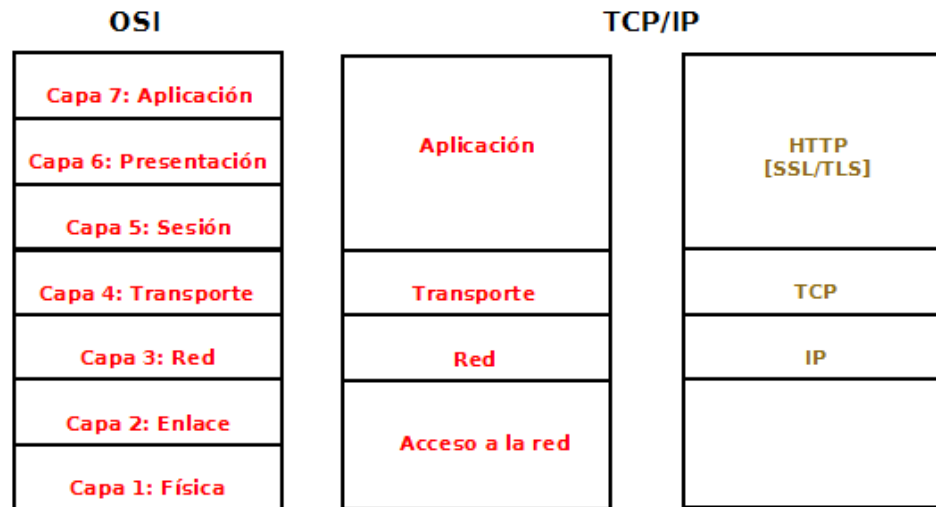
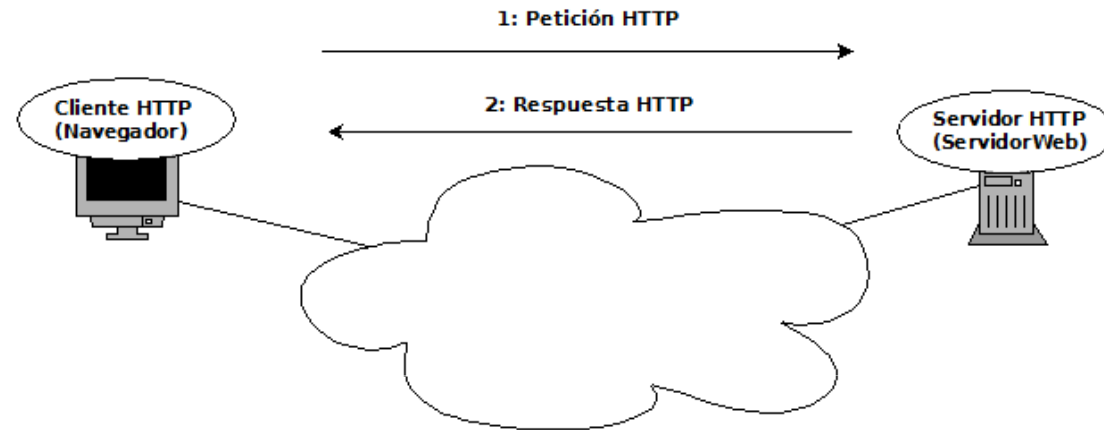
---

- ▶ HTTP (*Hyper Text Transfer Protocol*)
- ▶ Es de facto el protocolo de comunicación en la Web.
- ▶ Protocolo sin estado.
- ▶ Utiliza TCP como protocolo de transporte.
- ▶ Web
  - [http://www.w3.org/standards/techs/http#w3c\\_all](http://www.w3.org/standards/techs/http#w3c_all)

# Protocolo HTTP

## Introducción

---



# Protocolo HTTP

## Versiones

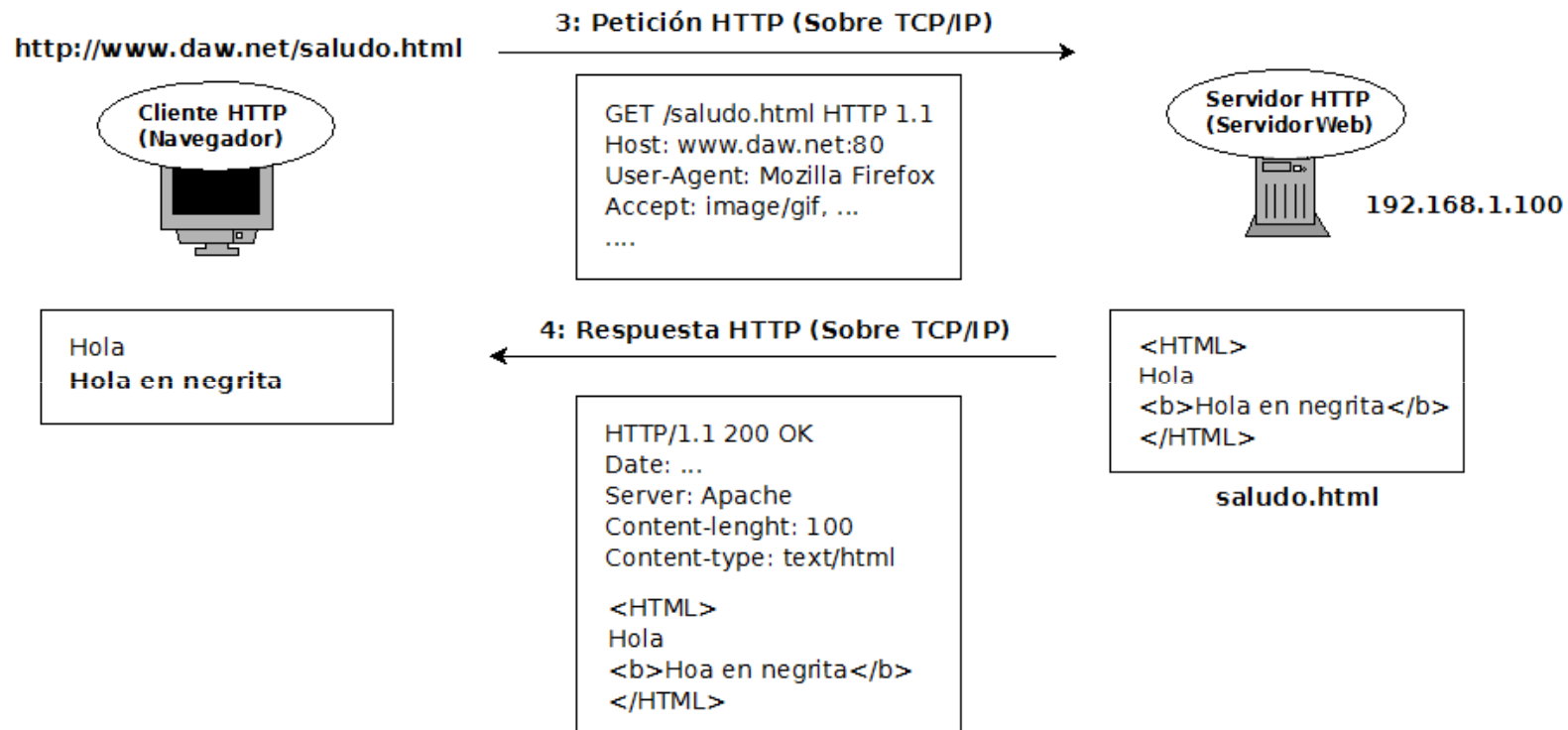
---

- ▶ HTTP/0.9 (Obsoleta).
- ▶ HTTP/1.0.
- ▶ HTTP/1.1 (versión actual) (RFC 2616).
- ▶ HTTP/1.2 (experimental).



# Protocolo HTTP

## Funcionamiento



- 1) El usuario introduce una **URI (o URL)** en la barra de direcciones del navegador o hace clic sobre un hiperenlace.
- 2) El navegador analiza la URL y establece una conexión TCP con el servidor web.
- 3) El navegador envía un **mensaje HTTP de petición** que depende de la URI (o URL).
- 4) El servidor envía un **mensaje de respuesta** que depende de la petición enviada y del estado del servidor.
- 5) Se cierra la conexión TCP.

# Protocolo HTTP

## Mensajes HTTP. Introducción

---

- ▶ Líneas en texto plano (formato ASCII).
- ▶ Dos tipos
  - Mensajes de petición.
  - Mensajes de respuesta.

# Protocolo HTTP

## Mensajes HTTP. Mensajes de petición

---

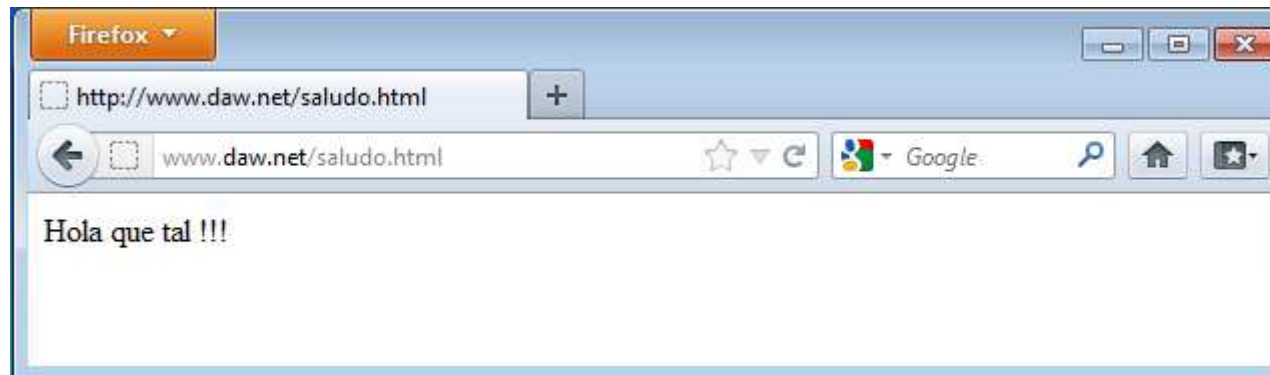
- ▶ Formados por tres partes
  - Línea inicial de petición
    - Método, URL, versión.
  - Líneas/s de cabecera.
  - Cuerpo del mensaje (opcional).      Parámetros o ficheros a enviar al servidor.

```
GET / HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

# Protocolo HTTP

## Mensajes HTTP. Mensajes de petición

---



```
GET /saludo.html HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es;es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 27 Apr 2012 07:40:28 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Fri, 27 Apr 2012 07:40:10 GMT
ETag: "20016-20-4bea436cd2425"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 48
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
.....(.....IT(MU(I.QPTT....H....j. ...|
```

# Protocolo HTTP

## Mensajes HTTP. Mensajes de respuesta

---

- ▶ Formados por tres partes
  - Línea inicial de respuesta (línea de estado)
    - Versión HTTP, código de estado y texto explicativo.
  - Líneas/s de cabecera.
  - Cuerpo del mensaje (opcional). Determinado por el tipo de recurso solicitado.

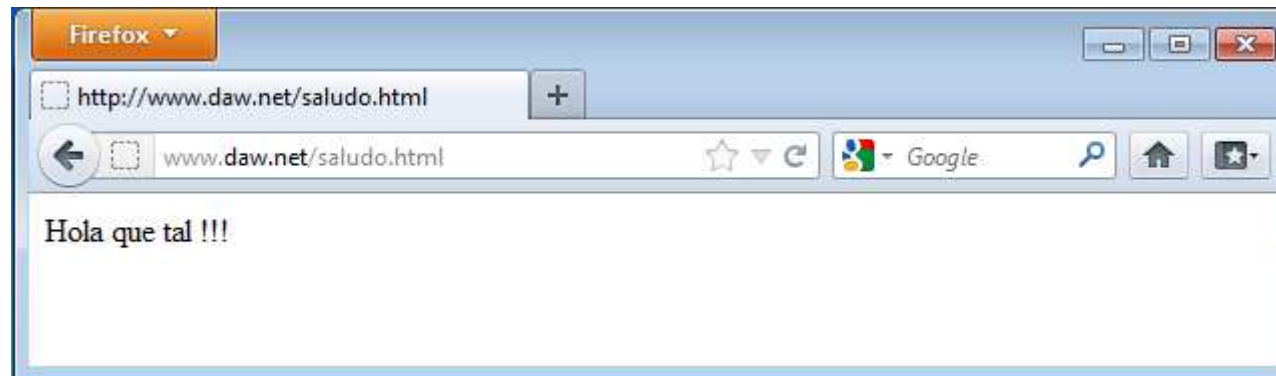
```
HTTP/1.1 200 OK
Date: Fri, 27 Apr 2012 07:47:19 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Fri, 27 Apr 2012 07:47:06 GMT
ETag: "2096e-34-4bea44fa4264c"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 71
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
.....(....I.O....0.SI-N,*....WHIUH,..LNL...K-V(OM...*.....?.4...|
```

# Protocolo HTTP

## Mensajes HTTP. Mensajes de respuesta

---



```
GET /saludo.html HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 27 Apr 2012 07:40:28 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Fri, 27 Apr 2012 07:40:10 GMT
ETag: "20016-20-4bea436cd2425"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 48
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
.....(.....IT(,MU(I.QPTT....H....j. ...|
```

# Protocolo HTTP

## Métodos de petición. Tipos

---

- ▶ Especifican la operación que quiere realizar el cliente en el servidor.
  - GET
  - POST
  - OPTIONS
  - HEAD
  - PUT
  - DELETE
  - TRACE
  - CONNECT
  - PATCH

# Protocolo HTTP

## Métodos de petición. GET

---

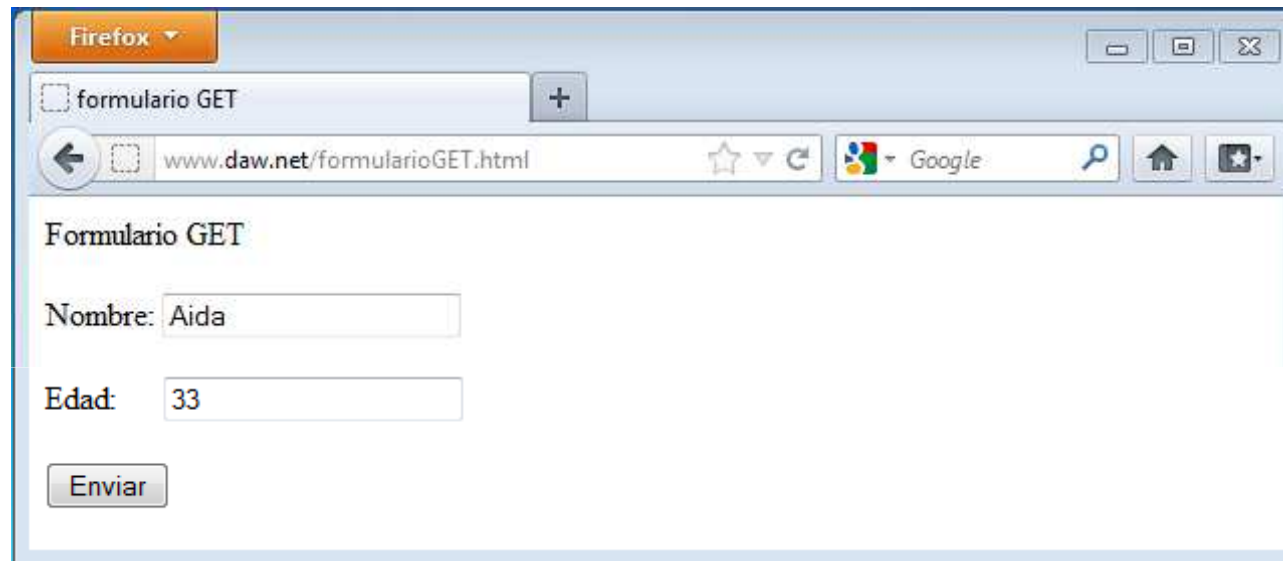
- ▶ Se emplea para obtener cualquier tipo de información del servidor.
- ▶ Se invoca normalmente cuando
  - Se introduce una URL en el navegador.
  - Se pincha sobre un hiperenlace.
  - Se envía un formulario GET.
- ▶ Permite enviar parámetros al servidor en la URI (o URL) (conocidos como *Query String*).
  - <http://datosGET.php?nombre=30&edad=Alicia>



# Protocolo HTTP

## Métodos de petición. GET

---



```
GET /datosGET.php?nombre=Aida&edad=33&Enviar=Enviar HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.daw.net/formularioGET.html
```

# Protocolo HTTP

## Métodos de petición. GET

---

- ▶ Las peticiones GET no envían cuerpo de mensaje.
- ▶ El tamaño de la información enviada estará limitada.
- ▶ No se puede usar para subir archivos o realizar otras operaciones que requieran enviar una gran cantidad de datos al servidor.

# Protocolo HTTP

## Métodos de petición. POST

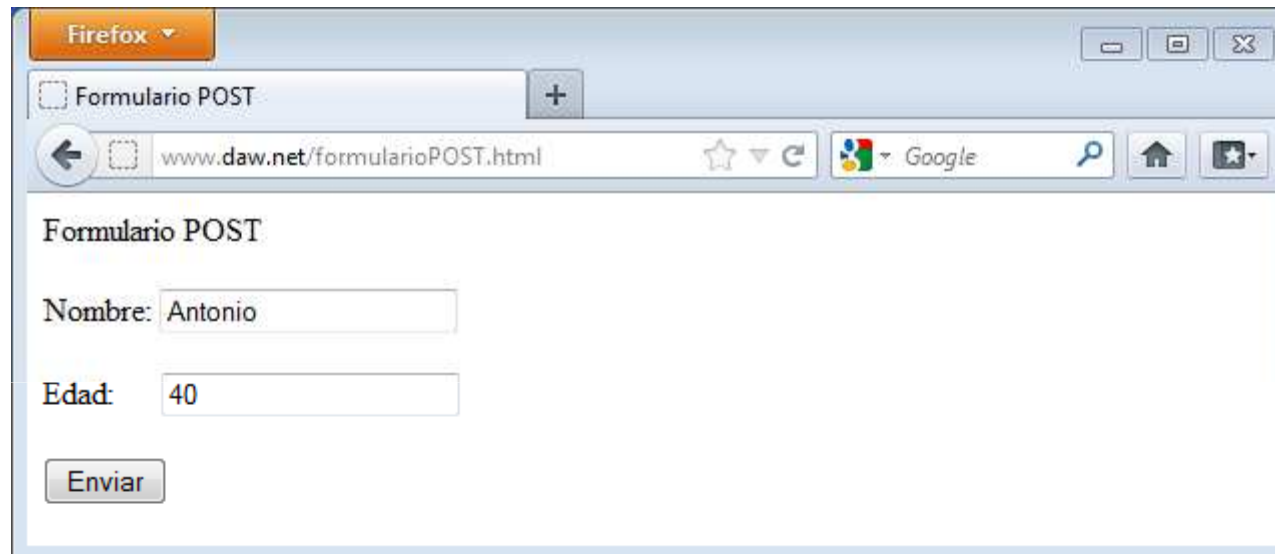
---

- ▶ Para solicitar al servidor que acepte información que se envía adjunta en una petición.
- ▶ Las peticiones POST envían el cuerpo de mensaje.
- ▶ Los parámetros no son visibles por lo tanto en la URL.
- ▶ Se invoca normalmente como consecuencia de enviar un formulario POST.

# Protocolo HTTP

## Métodos de petición. POST

---



Firefox

Formulario POST

Nombre: Antonio

Edad: 40

Enviar

```
POST /datosPOST.php HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.daw.net/formularioPOST.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 36

nombre=Antonio&edad=40&Enviar=Enviar
```

# Protocolo HTTP

## Cabeceras

---

- ▶ Pares de nombre/valor que se pueden incluir en los mensajes de petición y respuesta HTTP.

```
GET / HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

- ▶ Definen información (metadatos) sobre
  - Los datos que se intercambian los clientes y servidores.
  - Los propios clientes y servidores.
  - Sobre la propia transferencia de información.

# Protocolo HTTP

## Cabeceras

---

- ▶ Múltiples tipos de cabeceras
  - Generales (*Date, Transfer-Encoding, ...*).
  - De petición (o de cliente) (*User-Agent, Accept, ...*).
  - De respuesta (o de servidor) (*Server, Age, ...*)
  - De entidad (*Content-Encoding, Content-Language, Content-Type, ...*).
- ▶ Web
  - <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

# Protocolo HTTP

## Códigos de estado y error

---

- ▶ Códigos que envían los servidores en las respuestas HTTP .

```
HTTP/1.1 404 Not Found
Date: Fri, 27 Apr 2012 08:42:18 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 233
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

```
HTTP/1.1 304 Not Modified
Date: Fri, 27 Apr 2012 08:40:48 GMT
Server: Apache/2.2.22 (Ubuntu)
Connection: Keep-Alive
Keep-Alive: timeout=5, max=99
ETag: "2096e-34-4bea44fa4264c"
Vary: Accept-Encoding
```

- ▶ Informan al cliente de cómo ha sido procesada la petición.
- ▶ Se acompañan de un texto explicativo.

# Protocolo HTTP

## Códigos de estado y error

- ▶ Código de 3 dígitos que se clasifican en función del primero.
  - 100 – 199 (Informativo, *Informational*).
  - 200 – 299 (Exito, *Successful*).
  - 300 – 399 (Redireccion, *Redirection*).
  - 400 – 499 (Errores del cliente, *Client Error*).
  - 500 – 599 (Errores en el servidor, *Server Error*).



```
HTTP/1.1 404 Not Found
Date: Fri, 27 Apr 2012 08:42:18 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 233
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```



# Protocolo HTTP

## Negociación de contenidos. Tipos MIME

---

- ▶ HTTP soporta la negociación de contenidos entre cliente y servidor.
  - Tipos de contenido (Tipos MIME)
    - *MIME-Version, Content-Description, Content-Id, Content-Transfer, Content-Type, ...*
  - Lenguaje
    - *Accept-Language*
  - Conjunto de caracteres
    - *Accept-Charset*
  - Codificación/compresión
    - *Accept-Encoding, Content-Encoding*
  - ...

# Protocolo HTTP

## Negociación de contenidos. Tipos MIME

---

- ▶ **MIME (*Multipurpose Internet Mail Extensions*)**
  - Conjunto de especificaciones orientadas a intercambiar, usando protocolos como HTTP y SMTP, todo tipo de recursos (texto, audio, vídeo, imágenes,...) de forma transparente a los usuarios.
  - Tipos y subtipos (*text/html*, *image/gif*, ...) que determinan el contenido de los recursos enviados a través de la red.

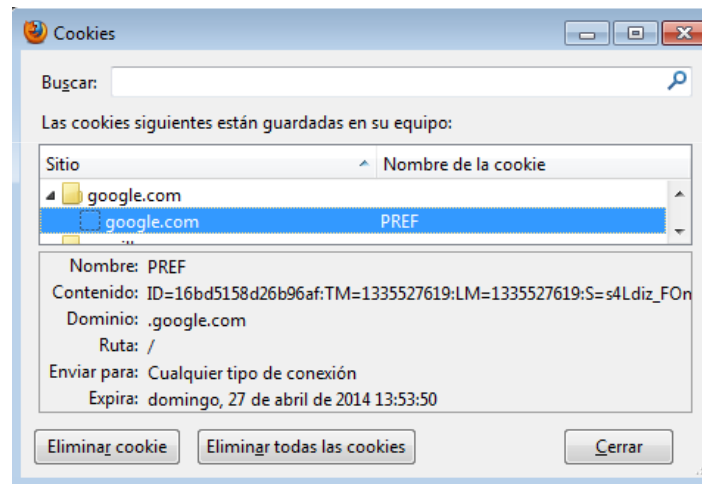
```
HTTP/1.1 200 OK
Date: Fri, 27 Apr 2012 07:47:19 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Fri, 27 Apr 2012 07:47:06 GMT
ETag: "2096e-34-4bea44fa4264c"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 71
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
.....(....I.O....0.sI-N,*....WHIUH,..LNL...K-V(OM...*. ....?.4...|
```

# Protocolo HTTP

## *Cookies*

---

- ▶ Fragmento de información que envía un servidor web en una respuesta HTTP y es almacenada por el navegador.



- ▶ El navegador puede enviar la cookie en solicitudes posteriores al mismo servidor.

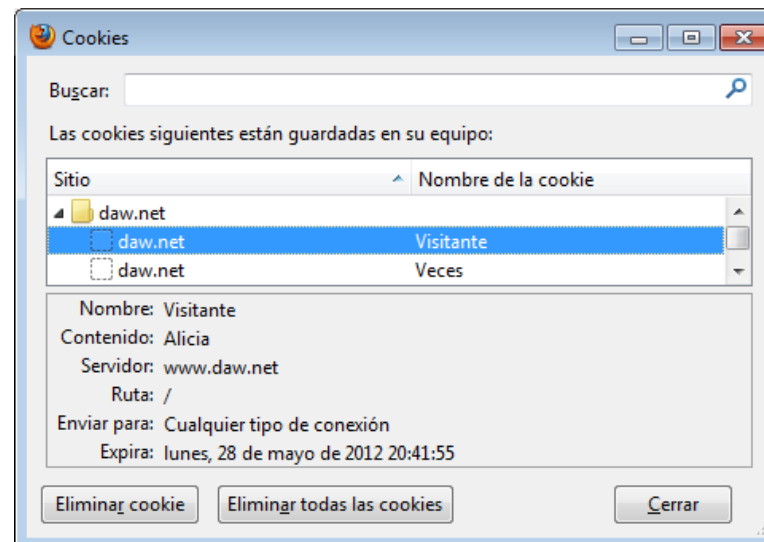
# Protocolo HTTP

## *Cookies*

---

### ► Cabeceras: *Cookies* y *Set-Cookie*

```
GET /cookies.html HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es;es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: Veces=3; Visitante=Alicia
If-Modified-Since: Fri, 27 Apr 2012 08:16:39 GMT
If-None-Match: "20019-975-4bea4b951dbeb"
Cache-Control: max-age=0
```



# Protocolo HTTP

## Autenticación

---

- ▶ Mecanismos de autenticación para controlar el acceso a los recursos que ofrece el servidor.
  - *Basic.*
  - *Digest.*
- ▶ Basados en
  - Código de estado 401.
  - Cabeceras *WWW-Authenticate* y *Authorization*.

# Protocolo HTTP

## Sesiones

---

- ▶ HTTP es un protocolo “sin estado”.
  - Cada transferencia de datos es independiente de la anterior sin ninguna relación entre ellas.
- ▶ Técnicas para mantener la sesión
  - *Cookies*
  - *URL Rewriting*
    - <http://www.daw.net/login.php?sessionid=2a1vJ>
  - Campos ocultos en formularios.
- ▶ APIs de tecnologías
  - *PHP, JavaEE, .NET, ...*

# Protocolo HTTP

## Otras características

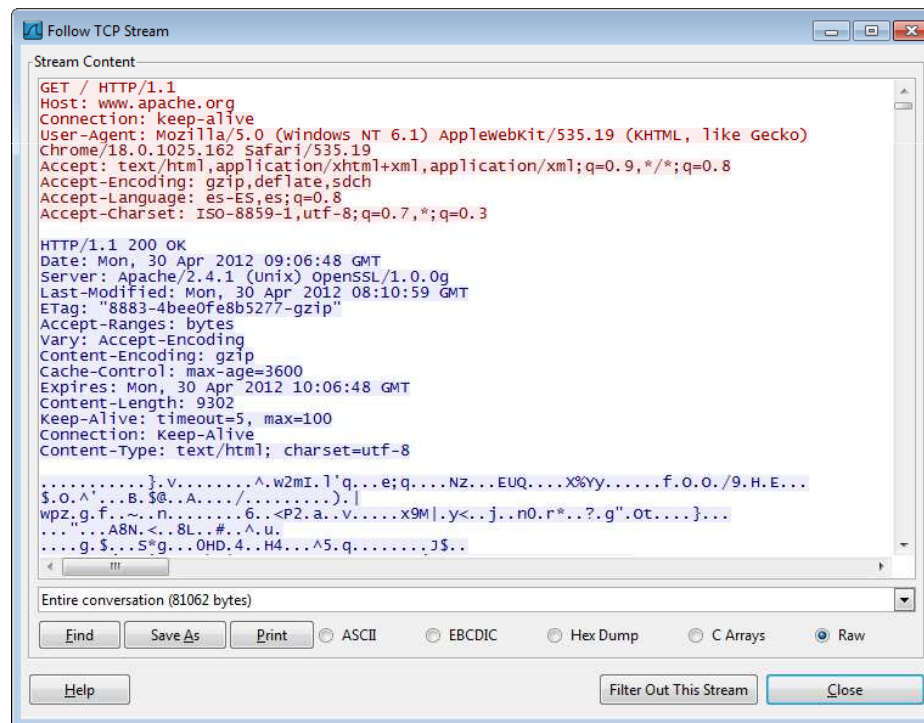
---

- ▶ Almacenamiento en *cache*
  - *Cache-Control, Last-Modified, Expires, Age, Etag, If-Match, IF-Modified-Sinze, ...*
- ▶ Redirecciones
- ▶ Conexiones persistentes
  - *Connection, Content-Length,...*

# Práctica

- Práctica 4.1
  - Protocolo HTTP

63 4.976046 192.168.1.16 140.211.11.131 HTTP 424 GET / HTTP/1.1





# Bibliografía

---

- ▶ Servicios de Red e Internet. Álvaro García Sánchez, Luis Enamorado Sarmiento, Javier Sanz Rodríguez. Editorial Garceta.
- ▶ <http://www.w3c.org>
- ▶ <http://www.w3c.es>