
# El Protocolo HTTP

JUAN CARLOS CONDE RAMÍREZ

*WEB-SERVICES*

# Introducción, I

- El *Protocolo de Transferencia de Hipertexto* (HTTP) está diseñado para habilitar la comunicación entre clientes y servidores.

- HTTP funciona como un protocolo de solicitud-respuesta (*request-response*) entre un cliente y un servidor.

- Un navegador Web juega el papel de cliente y una aplicación en una computadora que hospeda un sitio Web juega el rol de servidor.

# Introducción, II

- Ejemplo:
  - Un cliente (navegador) envía una solicitud HTTP al servidor; entonces el servidor devuelve una respuesta al cliente.

- La respuesta contiene información acerca del **estatus de la solicitud** y probablemente también contendrá el **contenido de la solicitud**.

# Un poco de Historia, I

- Desde 1990, el protocolo HTTP es el protocolo más utilizado en Internet.
  - La **versión 0.9** sólo tenía la finalidad de transferir los datos a través de Internet (en particular páginas Web escritas en HTML).
  - La **versión 1.0** del protocolo (la más utilizada) permite la transferencia de mensajes con encabezados que describen el contenido de los mensajes mediante la codificación MIME.
- El tipo MIME (*Multipurpose Internet Mail Extensions*) es un estándar propuesto por los laboratorios **Bell Communications** en 1991 para incluir la posibilidad de insertar documentos (imágenes, sonidos y texto) en un mensaje de correo electrónico.

# Un poco de Historia, II

- Desde entonces, se usa para dar formato tanto a los documentos adjuntos en un mensaje como a los documentos transferidos a través del protocolo HTTP.

- Así, durante una transacción entre un servidor Web y un navegador, el servidor Web envía en primer lugar el tipo MIME del archivo enviado al explorador para que éste sepa cómo se mostrará el documento.

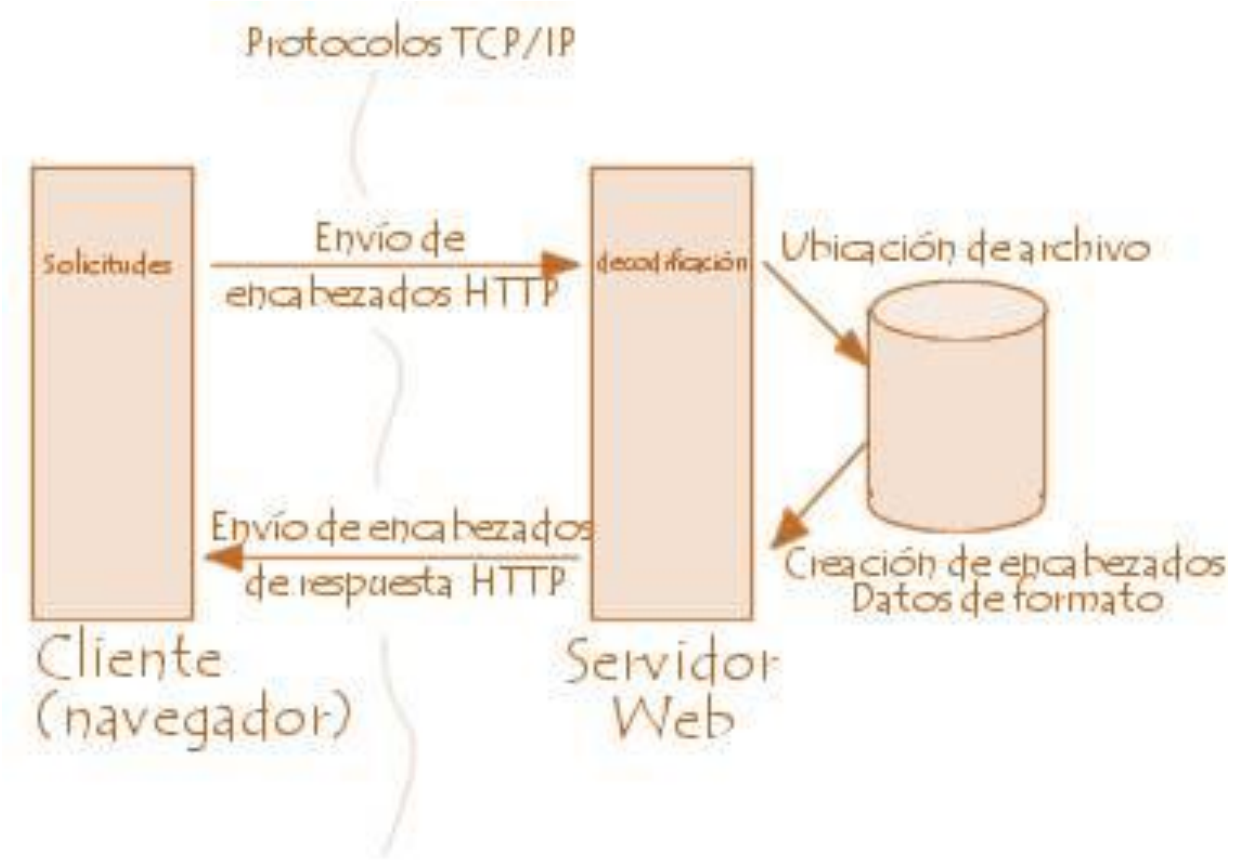
- Un tipo MIME está compuesto de la siguiente manera:

```
Content-type: tipo_mime_principal/subtipo_mime
```

Por ejemplo, una imagen GIF tiene el siguiente tipo MIME:

```
Content-type: image/gif
```

# Un poco de Historia, III



# Métodos HTTP

- Los métodos HTTP para poder comunicar al cliente con el servidor son:
  - **GET**
  - **POST**
  - **PUT**
  - **HEAD**
  - **DELETE**
  - **PATCH**
  - **OPTIONS**

# El método GET, I

- **GET se utiliza para solicitar datos a partir de un recurso específico.**

- GET es uno de los métodos HTTP más comunes o usados.

- Debe existir una cadena de consulta (nombre/valor) que es enviada la URL de una solicitud GET. Por ejemplo:

```
/test/demo_form.php?name1=value1&name2=value2
```

# El método GET, II

- Algunas otras características del método GET son:
  - Las solicitudes GET pueden ser almacenadas en cache.
  - Las solicitudes GET permanecen en el historial del navegador.
  - Las solicitudes GET pueden ser guardadas en favoritos.
  - Las solicitudes GET nunca deben ser usadas cuando se trabaja con datos confidenciales.
  - Las solicitudes GET tienen restricciones de longitud.
  - Las solicitudes GET deben ser utilizadas únicamente para solicitar datos (no modificar).

# El método POST, I

- **POST es usado para enviar datos a un servidor y crear/actualizar un recurso.**

- Los datos enviados al servidor con POST son almacenados en el cuerpo de la solicitud HTTP:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

- POST también es uno de los métodos HTTP más utilizados.

# El método POST, II

- Algunas otras características del método POST son:
  - Las solicitudes POST nunca son almacenadas en cache.
  - Las solicitudes POST no permanecen en el historial del navegador.
  - Las solicitudes POST no pueden ser guardadas en favoritos.
  - Las solicitudes POST  no tienen restricciones de longitud.

# El método PUT

- **PUT es utilizado para enviar datos a un servidor y crear/actualizar un recurso.**

- La diferencia entre POST y PUT es que las solicitudes PUT son idempotentes.
  - En matemática y lógica, la *idempotencia* es la propiedad para realizar una acción determinada varias veces y aun así conseguir el mismo resultado que se obtendría si se realizara una sola vez.

- En contraste, el uso de una solicitud POST repetidamente tiene efectos secundarios o colaterales al crear el mismo recurso múltiples veces.

# El método HEAD

- **HEAD es muy parecido a GET, pero sin una respuesta (cuerpo de respuesta).**

- En otras palabras, si con GET se devuelve una lista de usuarios, entonces con HEAD se hará la misma solicitud pero sin retornar la lista de usuarios.

- La solicitudes HEAD resultan útiles para comprobar qué devolverá una solicitud GET antes de realizarla – por ejemplo, antes de descargar un archivo grande o un cuerpo respuesta extenso.

# Métodos DELETE y OPTIONS

- **El método DELETE elimina el recurso especificado.**
- **El método OPTIONS describe las opciones de comunicación para el recurso objetivo.**
- A continuación una tabla comparativa entre los métodos más utilizados, GET y POST, que puede resultar de utilidad en el futuro…

# GET vs POST

| Rasgo | GET | POST |
|---|---|---|
| Botones Atrás/Recargar | Inofensivo | Los datos se volverán a enviar (el navegador debe alertar al usuario de que los datos están a punto de volver a enviarse) |
| Favoritos (marcadores) | Puede ser marcado | No puede ser marcado |
| Almacenamiento en cache | Se almacena | No se almacena |
| Tipo de codificación | `application/x-www-form-urlencoded` `multipart/form-data` | Usa codificación *multipart* para datos binarios. |
| Historial | Los parámetros se mantienen en el historial del navegador | Los parámetros no son guardados en el historial del navegador |

# GET vs POST

| Rasgo | GET | POST |
|---|---|---|
| Restricciones de longitud | Sí, al enviar datos, el método GET agrega los datos a la URL; y la longitud de una URL es limitada (la longitud máxima de la URL es de 2048 caracteres) | Sin restricciones |
| Restricciones de tipo de datos | Sólo se permiten caracteres ASCII | Sin restricciones. Se permiten datos binarios |
| Seguridad | GET es menos seguro en comparación con POST porque los datos enviados son parte de la URL<br>¡Nunca uses GET cuando envíes contraseñas u otra información confidencial! | POST es un poco más seguro que GET porque los parámetros no se almacenan en el historial del navegador o en los registros del servidor web |
| Visibilidad | Los datos en la URL son visibles para todos | Los datos no se muestran en la URL |

# Solicitudes HTTP (*requests*), I

- Una solicitud HTTP es un conjunto de líneas que el navegador envía al servidor incluye:
  - **línea de solicitud**: es una línea que especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizada. La línea está formada por tres elementos que deben estar separados por un espacio:
    - el método
    - la dirección URL
    - la versión del protocolo utilizada por el cliente (por lo general, HTTP/1.0)

# Solicitudes HTTP (*requests*), II

- **campos del encabezado**: es un conjunto de líneas opcionales que permiten aportar información adicional sobre la solicitud y/o el cliente (navegador, sistema operativo, etc.). Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

- **cuerpo**: es un conjunto de líneas opcionales que deben estar separadas de las líneas precedentes por una línea en blanco y, por ejemplo, permiten que se envíen datos por un comando POST durante la transmisión de datos al servidor utilizando un formulario.

# Solicitudes HTTP (*requests*), III

- Por lo tanto, una solicitud HTTP posee la siguiente sintaxis (<crlf> significa retorno de carro y avance de línea):

1: MÉTODO URL VERSIÓN<crlf>
2: ENCABEZADO: Valor<crlf>
3: . . . ENCABEZADO: Valor<crlf>
4: Línea en blanco <crlf>
5: CUERPO DE LA SOLICITUD

- A continuación se encuentra un ejemplo de una solicitud HTTP:

GET http://es.kioskea.net HTTP/1.0
Accept : Text/html
If-Modified-Since : Saturday,15-January-2000 14:37:11 GMT

User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

# Encabezados de Solicitud

| Encabezado | Descripción |
|---|---|
| Accept | Tipo de contenido aceptado por el navegador (por ejemplo, texto/html). Consulte Tipos de MIME |
| Accept-Charset | Juego de caracteres que el navegador espera |
| Accept-Encoding | Codificación de datos que el navegador acepta |
| Accept-Language | Idioma que el navegador espera (de forma predeterminada, inglés) |
| Authorization | Identificación del navegador en el servidor |
| Content-Encoding | Tipo de codificación para el cuerpo de la solicitud |
| Content-Language | Tipo de idioma en el cuerpo de la solicitud |
| Content-Length | Extensión del cuerpo de la solicitud |
| Content-Type | Tipo de contenido del cuerpo de la solicitud (por ejemplo, texto/html). Consulte Tipos de MIME |
| Date | Fecha en que comienza la transferencia de datos |
| Forwarded | Utilizado por equipos intermediarios entre el navegador y el servidor |
| From | Permite especificar la dirección de correo electrónico del cliente \| Permite especificar que debe enviarse el documento si ha sido modificado desde una fecha en particular |
| Link | Vínculo entre dos direcciones URL |
| Orig-URL | Dirección URL donde se originó la solicitud |
| Referer | Dirección URL desde la cual se realizó la solicitud |
| User-Agent | Cadena con información sobre el cliente, por ejemplo, el nombre y la versión del navegador y el sistema operativo |

# Respuestas HTTP (*responses*), I

- Una respuesta HTTP es un conjunto de líneas que el servidor envía al navegador está constituida por:

  - **línea de estado**: es una línea que especifica la versión del protocolo utilizada y el estado de la solicitud en proceso mediante un texto explicativo y un código. La línea está compuesta por tres elementos que deben estar separados por un espacio:
    - la versión del protocolo utilizada
    - el código de estado
    - el significado del código

  - **campos del encabezado**: es un conjunto de líneas opcionales que permiten aportar información adicional sobre la respuesta y/o el servidor. Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

# Respuestas HTTP (*responses*), II

- El cuerpo de la respuesta contiene el documento solicitado. Por lo tanto, una respuesta HTTP posee la siguiente sintaxis (<crlf> significa retorno de carro y avance de línea):

```
1: VERSIÓN-HTTP CÓDIGO EXPLICACIÓN <crlf>
2: ENCABEZADO: Valor<crlf>
3: . . . ENCABEZADO: Valor<crlf>
4: Línea en blanco <crlf>
5: CUERPO DE LA RESPUESTA
```

- A continuación se encuentra un ejemplo de una respuesta HTTP:

```
HTTP/1.0 200 OK
Date: Sat, 15 Jan 2000 14:37:12 GMT
Server : Microsoft-IIS/2.0
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```

# Encabezados de Respuesta

| Encabezado | Descripción |
|---|---|
| Content-Encoding | Tipo de codificación para el cuerpo de la respuesta |
| Content-Language | Tipo de idioma en el cuerpo de la respuesta |
| Content-Length | Extensión del cuerpo de la respuesta |
| Content-Type | Tipo de contenido del cuerpo de la respuesta (por ejemplo, texto/html). Consulte Tipos de MIME |
| Date | Fecha en que comienza la transferencia de datos |
| Expires | Fecha límite de uso de los datos |
| Forwarded | Utilizado por equipos intermediarios entre el navegador y el servidor |
| Locaton | Redireccionamiento a una nueva dirección URL asociada con el documento |
| Server | Características del servidor que envió la respuesta |

# Códigos de Respuesta

- Son los códigos que se ven cuando el navegador no puede mostrar la página solicitada.

- El código de respuesta está formado por tres dígitos:
  - El 1º indica el estado
  - El 2º y el 3º explican la naturaleza exacta del error

# Códigos de Respuesta: 10x y 20x

| Código | Mensaje | Descripción |
|---|---|---|
| 10x | Mensaje de información | Estos códigos no se utilizan en la versión 1.0 del protocolo |
| 20x | Éxito | Estos códigos indican la correcta ejecución de la transacción |
| 200 | OK | La solicitud se llevó a cabo de manera correcta |
| 201 | CREATED | Sigue a un comando POST e indica el éxito, la parte restante del cuerpo indica la dirección URL donde se ubicará el documento creado recientemente |
| 202 | ACCEPTED | La solicitud ha sido aceptada, pero el procedimiento que sigue no se ha llevado a cabo |
| 203 | PARTIAL INFORMATION | Cuando se recibe este código en respuesta a un comando de GET indica que la respuesta no está completa |
| 204 | NO RESPONSE | El servidor ha recibido la solicitud, pero no hay información de respuesta |
| 205 | RESET CONTENT | El servidor le indica al navegador que borre el contenido en los campos de un formulario |
| 206 | PARTIAL CONTENT | Es una respuesta a una solicitud que consiste en el encabezado range. El servidor debe indicar el encabezado content-Range |

# Códigos de Respuesta: 30x

| Código | Mensaje | Descripción |
|---|---|---|
| 30x | Redirección | Estos códigos indican que el recurso ya no se encuentra en la ubicación especificada |
| 301 | MOVED | Los datos solicitados han sido transferidos a una nueva dirección |
| 302 | FOUND | Los datos solicitados se encuentran en una nueva dirección URL, pero, no obstante, pueden haber sido trasladados |
| 303 | METHOD | Significa que el cliente debe intentarlo con una nueva dirección; es preferible que intente con otro método en vez de GET |
| 304 | NOT MODIFIED | Si el cliente llevó a cabo un comando GET condicional (con la solicitud relativa a si el documento ha sido modificado desde la última vez) y el documento no ha sido modificado, este código se envía como respuesta |

# Códigos de Respuesta: 40x

| Código | Mensaje | Descripción |
|---|---|---|
| 40x | Error debido al cliente | Estos códigos indican que la solicitud es incorrecta |
| 400 | BAD REQUEST | La sintaxis de la solicitud se encuentra formulada de manera errónea o es imposible de responder |
| 401 | UNAUTHORIZED | Los parámetros del mensaje aportan las especificaciones de formularios de autorización que se admiten. El cliente debe reformular la solicitud con los datos de autorización correctos |
| 402 | PAYMENT REQUIRED | El cliente debe reformular la solicitud con los datos de pago correctos |
| 403 | FORBIDDEN | El acceso al recurso simplemente se deniega |
| 404 | NOT FOUND | Un clásico. El servidor no halló nada en la dirección especificada. Se ha abandonado sin dejar una dirección para redireccionar... :) |

# Códigos de Respuesta: 50x

| Código | Mensaje | Descripción |
|---|---|---|
| 50x | Error debido al servidor | Estos códigos indican que existe un error interno en el servidor |
| 500 | INTERNAL ERROR | El servidor encontró una condición inesperada que le impide seguir con la solicitud (una de esas cosas que les suceden a los servidores...) |
| 501 | NOT IMPLEMENTED | El servidor no admite el servicio solicitado (no puede saberlo todo...) |
| 502 | BAD GATEWAY | El servidor que actúa como una puerta de enlace o proxy ha recibido una respuesta no válida del servidor al que intenta acceder |
| 503 | SERVICE UNAVAILABLE | El servidor no puede responder en ese momento debido a que se encuentra congestionado (todas las líneas de comunicación se encuentran congestionadas, inténtelo de nuevo más adelante) |
| 504 | GATEWAY TIMEOUT | La respuesta del servidor ha llevado demasiado tiempo en relación al tiempo de espera que la puerta de enlace podía admitir (excedió el tiempo asignado...) |

# Documentación de Origen

- Para obtener más información sobre el protocolo HTTP, consulte la RFC (*Request for Comments*) 1945, que explica el protocolo en detalle:

  - RFC 1945 - Protocolo de transferencia de hipertexto - HTTP/1.0

  - RFC 2616 - Protocolo de transferencia de hipertexto - HTTP/1.1