WebService REST con Catalyst

Fundamentos de HTTP 1.1

info@covetel.com.ve [1]

[1] Cooperativa Venezolana de Tecnologías Libres R.S.

Fundamentos de HTTP

- Transacciones HTTP
- Peticiones HTTP
- Versiones
- Métodos HTTP
- Probar los métodos
- Respuestas HTTP, códigos de estado

Hypertext Transfer Protocol I

Hypertext Transfer Protocol o **HTTP** (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web.

Fue desarrollado por la W3C y la IETF, esta colaboración culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el **RFC 2616**.

El protocolo HTTP esta orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

El cliente que efectua la petición se le conoce como *User Agent* o **Agente de Usuario** este puede ser un navegador web, un *spider* u otros.

La información transmitida se llama **recurso**.

La información se identifica mediante un *Uniform Resource Locator* (**URL**) o Localizador Uniforme de Recursos, cuyo formato general es:

```
esquema://maquina/directorio/recurso
```

Hypertext Transfer Protocol II

Los recursos transmitidos pueden ser archivos, cadenas de texto, una consulta a una base de datos, o cualquier otro tipo de información.

Stateless, HTTP es un protocolo **sin estado**, esto quiere decir, que no guarda ninguna información sobre conexiones anteriores.

Para mantener información del estado entre transacción y transacción se utilizan las *cookies*, que es información que un servidor puede almacenar en el sistema cliente.

Transacciones HTTP

Una transacción HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato. El encabezado especificará cosas como la acción requerida del servidor, o el tipo de dato retornado, o el código de estado.

Un encabezado es un bloque de datos que precede a la información propiamente dicha.

El servidor puede excluir cualquier encabezado que ya esté procesado, como Authorization, Content-type y Content-length.

Formato de mensajes

Cada transacción HTTP es una comunicación distinta. En cada una de ellas se intercambian mensajes. Según la especificación del protocolo, un mensaje es *la unidad básica de la comunicación HTTP y consiste de una secuencia estructurada de octetos ordenados con formato válido y transmitidos por la conexión.*
Hay dos tipos de mensajes, petición o solicitud (Request) y respuesta (Response), cada uno con su estructura, pero a groso modo el formato de un mensaje genérico sería el siguiente:

- Línea de comienzo: tipo de petición o tipo de respuesta.
- Cero o más líneas de encabezado acabadas en CRLF.
- Separador, que no es más que otro CRLF.
- Cuerpo del mensaje.

Ejemplo de un diálogo HTTP I

- Prepare un servidor HTTP utilizando el módulo `Net::Server::HTTP`

```
$ perl -e 'use Net::Server::HTTP; Net::Server::HTTP->run(port=>3001)'
```

- Si no tiene instalado telnet, por favor instalelo.
- Utilizando telnet, conectese al puerto 3001 en `localhost` para iniciar un diálogo HTTP.

```
$ telnet localhost 3001
```

- Al conectarse exitosamente debe ver el siguiente mensaje:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

- Es hora de enviar nuestra petición.

```
GET / HTTP/1.0
Host: localhost
User-Agent: mi navegador
```

- El servidor responde de la siguiente manera:

Ejemplo de un diálogo HTTP II

```
HTTP/1.0 200 OK
Date: Mon Sep 26 18:09:39 2011 GMT
Connection: close
Server: Net::Server::HTTP/0.99
Content-type: text/html
```

- Una linea en blanco y luego los datos

```
<form method=post action=/bam><input type=text name=foo><input type=submit></form>
<pre>%ENV = (
        'HTTP_HOST' => 'localhost',
        'HTTP_USER_AGENT' => 'mi navegador',
        'NET_SERVER_SOFTWARE' => 'Net::Server::HTTP/0.99',
```

Peticiones HTTP Request

Una petición HTTP, en su formato más básico, tiene la siguiente sintaxis:

```
método  URI  versión
```

El método le indica al servidor que hacer con el URI , por último la versión simplemente indica el número de versión del protocolo que el cliente entiende. Una petición habitual utiliza el método GET para pedirle al servidor que devuelva el URI solicitado:

```
GET /index.html HTTP/1.0
```

Respuesta del Servidor

Generalmente el servidor envía al cliente la siguiente información:

- Un código de estado que indica si la petición fue correcta o no.
- El recurso solicitado.
- Información sobre el recurso que se retorna.

Versiones

0.9 Obsoleta, solo soportaba el método GET.

HTTP/1.0 (mayo de 1996) Esta es la primera revisión del protocolo que especifica su versión en las comunicaciones. Aun se utiliza, sobre todo en servidores proxy.

HTTP/1.1 (junio de 1999) Versión actual, las conexiones persistentes están activadas por defecto (*Keep-Alive*) y funcionan bien con los proxies.

Métodos HTTP I

HTTP define 8 métodos (algunas veces referido como "verbos") que indica la acción que desea que se efectúe sobre el recurso identificado.

HEAD Pide una respuesta idéntica a la que correspondería a una petición GET, pero sin el cuerpo de la respuesta. Esto es útil para la recuperación de meta-información escrita en los encabezados de respuesta, sin tener que transportar todo el contenido.

GET Pide una representación del recurso especificado.

```
$ telnet www.google.co.ve 80

GET /images/srpr/logo3w.png HTTP/1.0
```

POST Indica al servidor que se prepare para recibir información del cliente. Suele usarse para enviar información desde formularios.

Métodos HTTP II

PUT Sube, carga o realiza un upload de un recurso especificado (archivo), es el camino más eficiente para subir archivos a un servidor, esto es porque en POST utiliza un mensaje multiparte y el mensaje es decodificado por el servidor. En contraste, el método PUT te permite escribir un archivo en una conexión socket establecida con el servidor.

DELETE Borra el recurso especificado, solo si esta disponible el método para el recurso.

TRACE Este método solicita al servidor que envíe de vuelta en un mensaje de respuesta, en la sección del cuerpo de entidad, toda la data que reciba del mensaje de solicitud. Se utiliza con fines de comprobación y diagnostico.

Métodos HTTP III

OPTIONS Devuelve los métodos HTTP que el servidor soporta para un URL especifico.Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición en lugar de un recurso especifico

CONNECT Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL.

El método HEAD

Es un método utilizado para preguntar información sobre un documento (metadatos), no para obtener el propio documento. HEAD es mucho más rápido que GET, ya que se transfiere mucha menos información. Es generalmente usada por clientes que usan caché para ver si el documento ha cambiado desde la última vez que accedieron a él. Si no cambió, la copia local puede usarse nuevamente; de otra forma, la versión actualizada debe recuperarse con GET.

Este método puede ser usado para testear la validez, accesibilidad y reciente modificación de enlaces, sin tener que descargar los documentos en sí.

El método POST

Es usado para transferir datos del cliente al servidor. Los datos del cuerpo de la solicitud se enviarán al URI especificado. Este URI será una referencia a un manipulador que procesará los datos de alguna forma, típicamente será un programa CGI.

Las respuestas al método POST no serán guardadas en caché salvo que en la propia respuesta se incluyan encabezados de control de caché adecuados.

Probar los métodos

La manera más didáctica para observar la estructura de las peticiones y el uso de los métodos, es con *CouchDB*

Por favor instale *couchdb*, en el sistema operativo Debian GNU/Linux se hace de la siguiente manera:

```
# aptitude install couchdb
```

Vamos a utilizar *Firefox* con la extensión de *Firebug* instalada y accedemos a la dirección `http://localhost:5984/_utils/`

Respuestas HTTP, códigos de estado

La sintaxis de una respuesta HTTP es, en su formato más básico una línea de estado y tiene una sintaxis como la siguiente:

```
versión  código-error  texto-explicativo
```

Incluye la versión de protocolo, un código de éxito o error y el texto explicativo al código anterior. Un ejemplo bastante común es:

```
HTTP/1.1 200 Ok
```

Los posibles códigos de estado se identifican con números de tres cifras y se clasifican en cinco grupos:

- Números del estilo **1XX** que representan mensajes de tipo informativo.
- Números del estilo **2XX** que indican que se completó satisfactoriamente la solicitud del cliente.
- Números del estilo **3XX** que indican que la solicitud fue redirigida.
- Números del estilo **4XX** que indican un error en la solicitud del cliente.
- Números del estilo **5XX** que indican un error en el lado del servidor.

Códigos típicos de estado HTTP

200 OK La solicitud del cliente fue satisfactoria y el servidor ha devuelto la información solicitada.

204 No Content El cuerpo de la respuesta no tiene contenido. Esto puede indicar, por ejemplo, un problema con un CGI que no devuelve datos.

301 Moved Permanently El URI solicitado no está disponible en el servidor. Ha sido movido a otra ubicación. Las solicitudes futuras deberán hacerse a esa ubicación.

400 Bad Request Hay un error de sintaxis en la solicitud del cliente. Por ejemplo mandar una solicitud indicando que el cliente soporta HTTP/1.1 y no enviar el encabezado de Host.

404 Not Found Este es junto con el 200 OK, el código más habitual. Indica que el documento solicitado no está disponible, probablemente el URI haya sido mal escrito.

500 Internal Server Error Este mensaje indica que algo ha ido mal en el servidor, casi siempre tiene que ver con problemas en programas CGI.