

HTTP Part 3

HTTP Request – RFC2616

HTTP – request/response

Today, we'll look at the request and response in more detail

<http://www.w3.org/Protocols/> -- all sorts of HTTP info

<http://www.w3.org/Protocols/rfc2616/rfc2616.txt>

HTTP “One shot” Transaction – “stateless”

"Stateless" – each request/reply pair (HTTP 1.0) uses its own connection. The dialog does not group logically related collections of requests.

The HTTP request/response protocol is "one shot" – the server fulfills the request and closes the connection.

This keeps the server's job simple, but it makes it harder to design a sequence of pages that appear logically connected, since each HTTP request is separate.

The lack of continuity between one HTTP request and the next is one of the harder things to absorb when designing HTTP client/server systems. We'll see this *continuity problem* when we try to build CGI web applications.

HTTP Request

GET Request

Request a resource from the server

GET *path* HTTP/1.0\r\n\r\n

GET /foo.html HTTP/1.0\r\n\r\n

The exact path sent depends on the parts of the URL...

Complex URLs

The usual simple description is that everything after the host is the "path"

The more realistic description divides things after the host into the path, query, and fragment, which is a reasonably accurate model. See the RFC for the full story on URL syntax.

URL Parts

<http://foo.com/a/b/bar.html?hello.there#binky>

scheme or protocol

Letters with a colon (:)

"http"

host

"foo.com"

path

"/a/b/bar.html"

Could separate the directory path "/a/b/" from the file "bar.html".

query

begins with a ?

"hello.there"

fragment

"binky"

URL Query

Begins with a '?' character.

The query may contain almost any characters, so scan for the first '?' and the query is what follows

e.g. `http://foo.com/a/b/bar.html?a=2&pi=22/7`

query is "a=2&pi=22/7"

e.g. `http://foo.com/a/b/bar.html?ref=http://bar.com/binky.html`

query is "ref=http://bar.com/binky.html"

For example, HTML forms can use the query string to send the user data from the form back to the server like this: "`http://form.cgi?name=ronyeh&id=234`".

URL Fragment

Begins with a '#'

`http://foo.com/a/b.html#here`

The fragment is not sent to the server as part of the request.

Instead, the fragment is used by the *client side* to control scrolling.

The fragment name is defined in the target HTML with a name binding:

`` – refer to this spot with the url "`b.html#here`"

Request

The request is basically the path/file/query part of the URL

`http://foo.com/dir/b.html` : request is "`/dir/b.html`"

The query string, if present, is included as part of the request

`http://foo.com/dir/b.html?info=extra&hello` : request is "`/dir/b.html?info=extra&hello`"

Fragment Suffix

A "`#foo`" fragment is **not** sent as part of the request

The "`#`" part of the url is handled on the client side.

`http://foo.com/dir/b.html#bar` : request is "`/dir/b.html`"

Basic GET

telnet *host* 80

Lab exercise: try telneting to port 80 of your favorite web server. Type the GET directive in by hand and watch what happens. On unix, the command looks like...

```
> telnet www.stanford.edu 80
```

Get Options

Some servers are confused if the "GET" is in lowercase

Usually the GET includes optional information on subsequent lines. Newer servers may need an HTTP 1.1 request to work...

```
GET request HTTP/1.0
-- blank line --
```

The Empty Request

What about URLs where the whole path/query section is the empty string? -- e.g.

<http://www.cnn.com>

Could just send the empty string as the request in the HTTP protocol – many servers interpret that as "/", but some treat it as an error.

With HTTP 1.1, there is an official way for the client to deal with this situation. If the request would be the empty string, the client should think of the request as being "/" and send that instead. Relative URLs in the response will be relative to "/".

Other requests: POST, PUT, DELETE

GET and POST are by far the most common.

We'll study POST as part of the CGI section

Other commands, such as PUT and DELETE, allow the client to send or edit data on the server side. They are rarely used.

HTTP Response

HTTP Response

The server responds with a header section which describes the document, followed by a blank line, followed by the document data.

```
HTTP/1.0 200 OK
```

The very first line of the response has the form VERSION CODE REASON

e.g. HTTP/1.0 200 OK

e.g. HTTP/1.1 404 Not Found

VERSION = the version of HTTP which the server is speaking

CODE = a numeric code which compactly indicates how the request is going

REASON = a human-readable statement of the CODE

Spaces follow the VERSION and CODE

Content-Length: size

Content-Length: *length* – the size in bytes of the document

The number of bytes to read after the blank line

Some HTTP1.1 variants get rid of this field, since it means the server cannot send any data until it knows how many bytes there are.

Content-Type: mime-type

Content-Type: *MIME-type*

Indicates the MIME type of the content to come – required

HTTP 1.0 Example

e.g. retrieve the HTML at <http://cslibrary.stanford.edu/test.html>

There's a blank line after the request, and a blank line that separates the header from the body in the response.

```

elaine0:~> telnet cslibrary.stanford.edu 80
Trying 171.64.64.168...
Connected to cslibrary.Stanford.EDU (171.64.64.168).
Escape character is '^]'.
GET /test.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 22 Apr 2002 18:59:37 GMT
Server: Apache/1.3.23 (Darwin)
Last-Modified: Mon, 22 Apr 2002 18:58:01 GMT
ETag: "115b1-be-3cc45d39"
Accept-Ranges: bytes
Content-Length: 190
Connection: close
Content-Type: text/html

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <title>Test</title>
</head>
<body bgcolor="#FFFFFF">

<h1>
Test</h1>
Just a little test doc.

</body>
</html>
Connection closed by foreign host.
elaine0:~>

```

Status Codes

200 = OK

3xx = Minor client error— document in another location

301 = Moved permanently. The Location: field of the header gives the correct URL.

302 = Moved temporarily – the server may suggest the correct url using a location: in the header

304 = Not Modified. The request had If-Modified-Since: field, but the document has not been modified, so the client should use their cached copy.

400 = Real client error

bad request, document not found, not allowed

400 = syntax error

401 = Unauthorized

403 = Forbidden – "permission denied"

404 = Not found!!!

500 = Server error

service not implemented, service not available

503 = Service unavailable. The server is temporarily not able to provide the service. The header may contain a Retry-After: field to indicate when the client might give it another shot.

Not-Found Example

```
elaine0:~> telnet cslibrary.stanford.edu 80
Trying 171.64.64.168...
Connected to cslibrary.Stanford.EDU (171.64.64.168).
Escape character is '^]'.
GET /nosuchthing.html HTTP/1.0
```

```
HTTP/1.1 404 Not Found
Date: Mon, 22 Apr 2002 19:02:40 GMT
Server: Apache/1.3.23 (Darwin)
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /nosuchthing.html was not found on this server.<P>
<HR>
<ADDRESS>Apache/1.3.23 Server at cslibrary.stanford.edu Port 80</ADDRESS>
</BODY></HTML>s
Connection closed by foreign host.
```

MIME Types

Multipurpose Internet Mail Extensions – a standard which predates HTTP for identifying different types of data for inclusion in email messages.

content-type/sub-type [*; aux-info*]

- text/html
- text/plain
- text/plain ; charset = us-ascii
- multipart/mixed ; boundary = SpecialBoundaryString
- application/pdf
- application/postscript
- audio/basic – .au audio
- image/jpeg
- video/quicktime
- "x-.." = experimental

MIME Type Dynamics

The server determines the MIME type of the document

It may use the file extension (.html, .pdf) or some other scheme

The client sees the type in the HTTP response header, and so knows what to look for after the blank line

The data may be binary instead of text (GIF, JPEG for example)

The browser may use plug-ins to handle some MIME types – the browser plug-in architecture is keyed on the MIME type of the incoming data e.g. the PDF browser plugin looks for the MIME type application/pdf