

HTTP 4

Empty Request -- www.stanford.edu

`http://www.stanford.edu`

`http://www.yahoo.com`

The path is the empty string

Some servers accept the empty string as a request and some don't

Special case added to the HTTP protocol: in this case, the client should make the request `"/`, and interpret relative URLs assuming the base URL path is `"/`

Server URL Mapping

The server sees the request path -- can interpret it in ways more complex than just mapping it into the file system

~username

Username -- the server may have a rule by which it maps `~username` or some other pattern into a directory in the file system...

`/~nick/a/x.html --> /users/nick/WWW/a/x.html`

`/class/cs193i/x.html --> /usr/class/cs193i/WWW/x.html`

Directories

A request may map to a directory instead of a file

In URLs, a directory ends with a `"/`

e.g. `http://foo.com/a/b/`

Default file

The server can be programmed to look for a default file in the directory. There may be multiple allowed default files (`index.html`, `default.html`, `index.shtml`, ...).

`/a/b/ --> /a/b/index.html`

Directory listing

The server may be programmed so that, if the default file is not present, the server generates an on the fly a directory listing in HTML and send that back. (This is how our course handouts directory works).

Server control

HTTP does not restrict how the server interprets the request. Whoever sets up the server can program in the mappings however they like. HTTP servers, such as apache, are very configurable.

Server programming features can be used to fix URLs as directories get renamed, files moved around, etc. so old URLs keep working. The most famous is Apache's "mod-rewrite" module which defines URL modification rules for each request.

Certain mappings, such as `directory->index.html`, are defacto standards.

Authoring tip

Either use the directory trick, and refer to directories as /a/b/ throughout the site, or refer to /a/b/index.html throughout the site, just be consistent. Otherwise, you end up with multiple names for the same thing, which confuses the caching and the "visited" color of links. Getting the visited link color correct is a part of site usability.

Client Side

Client unaware

The client is not aware of the index.html or the directory listing or whatever -- they just request /a/b/ and get back some HTML.

The client thinks the URL path is /a/b/

URL relative/absolute conversion is done on the client side

Client relative URL conversion

The client requests /a/

In reality, the server sends back /a/index.html

Suppose there is a "href=c.html in index.html

Notice, the client relative->absolute URL conversion still works, even though the client is unaware of the "index.html" file

base: /a/ + c.html = /a/c.html

The Trailing Slash

Directory URL without the trailing slash

e.g. /class/cs193i -- no trailing slash

No File

There is no file named /class/cs193i, but there is a directory -- the server figures that's what the client meant

302 Moved

Server answers back with a 302 (or 301, 303, ...) result code, basically saying the resource is not available. BUT...

Location: <http://www.stanford.edu/class/cs193i/>

Server includes a location: url field in the header that directs the client to a new url to use. This is a server redirect.

Client Side

All the major browsers automatically re-try the request with the new URL, so the user doesn't notice. However, the "corrected" url is now the one that shows up at the top of the window.

2-Trips

This fixes the problem, but at a small cost -- the client has to make two separate requests.

1. Simple Uses

Fix client naming goof-ups

2. Complex Uses

The client clicks on a link. The server wants to send them to a different page dynamically depending on -- who they are, what server is managing their session, etc.. Custom server programming can use a dynamically computed redirect to bounce the client to a different URL depending on the situation.

Changing URL

You can tell a redirect is happening when you type URL in, but suddenly your browser changes to a different URL because it got a 300 response from the server.

301 Example

```

elaine0:~> telnet www.stanford.edu 80
Trying 171.64.14.235...
Connected to www5.Stanford.EDU (171.64.14.235).
Escape character is '^]'.
GET /class/cs193i HTTP/1.0

HTTP/1.1 301 Moved Permanently
Date: Wed, 24 Apr 2002 19:58:27 GMT
Server: Stronghold/3.0 Apache/1.3.19 RedHat/3014c WebAuth/2.5 (Unix) mod_ssl/2.8.1
OpenSSL/0.9.6 WebAuth/2.5 mod_fastcgi/2.2.10
Location: http://www.stanford.edu/class/cs193i/
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://www.stanford.edu/class/cs193i/">here</A>.<P>
<HR>
<ADDRESS>Stronghold/3.0 Apache/1.3.19 RedHat/3014c WebAuth/2.5 Server at <A
HREF="mailto:webmaster@www.stanford.edu">www.stanford.edu</A> Port 80</ADDRESS>
</BODY></HTML>
Connection closed by foreign host.

```

Suppose did not correct client

Suppose the 300 error was not used, and instead the server guessed what the client wanted, and sent back the appropriate HTML for the directory listing.

1. The client requests "/a/b" where b is really a directory
2. The server figures the client meant "/a/b/" and so sends back the contents of "/a/b/index.html"
3. There's an "c.html" relative URL in the HTML. Remember, the client thinks the base URL path is /a/b. The client does the rel->abs conversion and gets "/a/c.html", when the correct URL was "/a/b/c.html"

Lesson: the client needs to have an accurate understanding of the base URL, including where the "/" falls, so it can do later requests correctly. The 300 redirects are used to correct the client.