

HTTP 4

Recall

SVG

Web design -- images vs. content

Relative URL Refresher -- Client Side

"Base URL" - the full URL of the enclosing page e.g. `http://foo.com/a/b.html`

Relative URL picks up whatever is missing (scheme, host, ...) from the base URL

`http://foo.com/a/b.html + c.html -> http://foo.com/a/c.html`

In this case, essentially remove the "file" of the URL, after the last slash, and replace it with the file in the relative url (c.html in this case)

Note that the absolute/relative conversion is done on the client side -- what the client thinks the base URL is and what URLs the client sees in the HTML it gets

Empty Request -- www.stanford.edu

`http://www.stanford.edu`

`http://www.yahoo.com`

The file/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 that way

Server URL Mapping

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

~username

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

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

`/class/cs193i/about.html --> /usr/class/cs193i/WWW/about.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, ...) error message 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 server with much more flexibility than a fixed url.

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 were not used, and instead the server added in / where necessary and sent back the HTML

1. Suppose 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. The client does the rel->abs conversion and gets "/a/c.html", when the correct URL was "/a/b/c.html"
 Lesson: the client and the server need to have a consistent understanding of where the "/" are in the URLs. That's why the 300 errors are required to correct the client.

HTTP1.1 Virtual Hosts

Suppose you want to host 20 web sites on one machine
 Hosting a web site is not very demanding, especially for plain web pages. You typically run out of networking bandwidth before you run out of CPU, memory, etc. Therefore, using one machine for many sites is common.
 Want http://foo.com/, http://bar.com/, all on the one machine -- "virtual hosts"
 DNS: multiple names, foo.com, bar.com, www.foo.com, ... can all map to one IP address.

Problem: when a server gets a request "GET / HTTP/1.0" -- which virtual host is it for?

Solution: with HTTP 1.1, the client includes a **host: www.foo.com** header in the request

Alternative: the other way to do this, is "multihoming" -- one machine has multiple IP addresses. Then the server can tell which virtual server is intended by which IP addr the request is addressed to. However, this method is deprecated, since it can use up lots of IP addresses.

Virtual Host 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 / HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Wed, 24 Apr 2002 18:50:04 GMT
Server: Apache/1.3.23 (Darwin)
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

176
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>400 Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
Your browser sent a request that this server could not understand.<P>
client sent HTTP/1.1 request without hostname (see RFC2616 section 14.23): /<P>
<HR>
<ADDRESS>Apache/1.3.23 Server at cslibrary.stanford.edu Port 80</ADDRESS>
</BODY></HTML>

0

Connection closed by foreign host.
```

```

elaine0:~> telnet cslibrary.stanford.edu 80
Trying 171.64.64.168...
Connected to cslibrary.Stanford.EDU (171.64.64.168).
Escape character is '^]'.
GET / HTTP/1.1
host:cslibrary.stanford.edu

```

```

HTTP/1.1 200 OK
Date: Wed, 24 Apr 2002 18:50:15 GMT
Server: Apache/1.3.23 (Darwin)
Last-Modified: Mon, 22 Apr 2002 18:56:54 GMT
ETag: "10cd0-1b34-3cc45cf6"
Accept-Ranges: bytes
Content-Length: 6964
Content-Type: text/html

```

```

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
...

```

HTTP 1.1 Persistent Connections

Problem: typical web page with one body of HTML and 20 little GIF buttons requires 21 separate connections.

TCP and HTTP are not efficient for many little connections -- must bring up the TCP connection many times (cost each time), and TCP is most efficient for longer running connections.

Persistent Connection -- Keep-Alive

Add the following to the request

Connection: keep-alive -- client request to keep connection open

Connection: close -- client request that the server close connection after response (like HTTP 1.0)

With HTTP 1.1, keep-alive is the default

Server indicates if it is closing the connection with

Connection: close -- in the response header (note that the same syntax is used in both the request and response headers)

Persistent Connection

Client can send more requests (GET ...) on the same connection -- avoid the connection setup cost

"Pipelining" -- client can send multiple requests, even before receiving responses.

This can be a great speedup -- avoids the latency*2 cost of a typical request/response system by not waiting for the response.

We say that multiple requests are "in flight" at one time.

Chunked Transfer Encoding

Problem: content-length header is costly for server -- have to wait before sending data

Solution: omit the content-length header, send data back in chunks, each chunk is prefixed by its length in hex (see the above examples)

Server puts **Transfer-encoding: chunked** in the header

End is marked with a chunk length 0