

HTTP 3

URLs

Document has "base" URL

Relative and Root Relative URLs in the document

Relative -> absolute URL conversion by the client using base URL

% encoding, '+' for spaces

mailto scheme "mailto:nick@cs.stanford.edu?subject=hello+there"

`hello` -- open in new window

Images

Separate URLs

``.

Width, height optional, but help speed layout on client side

GIF, PNG, JPEG, SVG

Image are very slow, and slowness is the #1 complaint of web users. Use for decoration to fill in around the text content.

Images are fun for the designer, so the error tends towards putting in a lot of images even though the site is worse for the users.

The Picture So Far

Client has URL for document -- base URL

Client request

Server gets request, maps to document root in file system

Response: returns document + meta information, type, size etc. about document

Client gets document, displays according to type

Same mechanism used for HTML, images, PDF, etc. etc.

Client interprets relative URLs in the document according to the base URL

HTTP 1.0 vs. HTTP 1.1

We'll start with HTTP 1.0 which is simplest, and then add in the HTTP 1.1 features

Recall

(previous handout)

Get Request

URL parts

Telnet trick for HTTP

The Empty Request

What about URLs with no path -- e.g. <http://www.cnn.com>

The directory path and file are the empty string.

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.

Here's the new (as of HTTP 1.1 anyway) correct way for the client to deal with this: 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

We'll deal with these later -- GET is by far the most important

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>
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