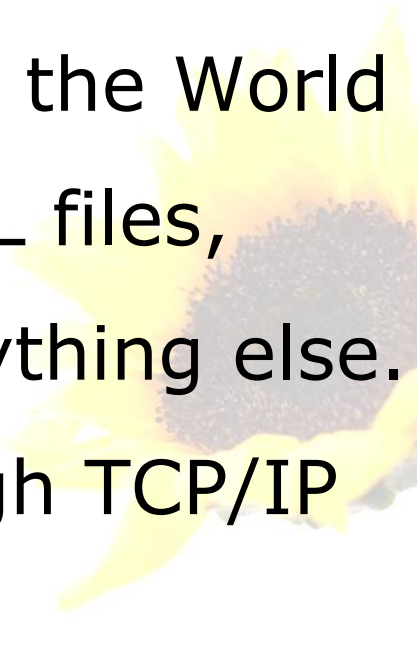


SECTION – B

HTTP

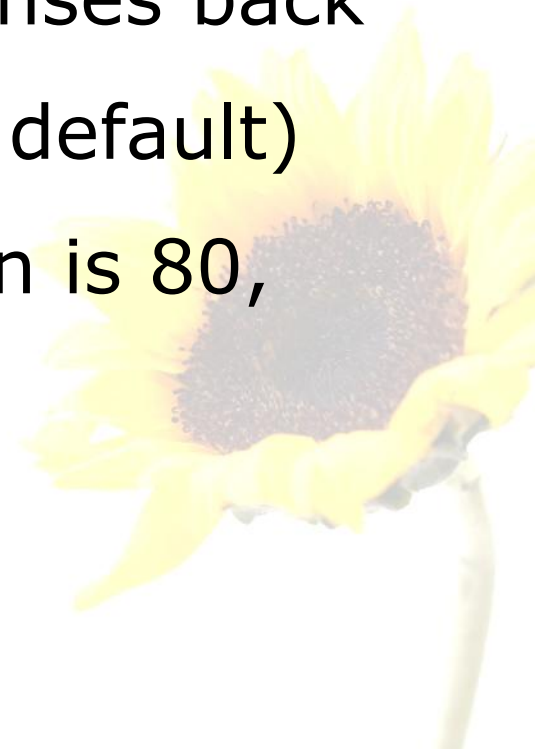


What is HTTP?

- HTTP stands for Hypertext Transfer Protocol. It's the network protocol used to deliver virtually all files and other data (collectively called resources) on the World Wide Web, whether they're HTML files, image files, query results, or anything else. Usually, HTTP takes place through TCP/IP sockets.
- 

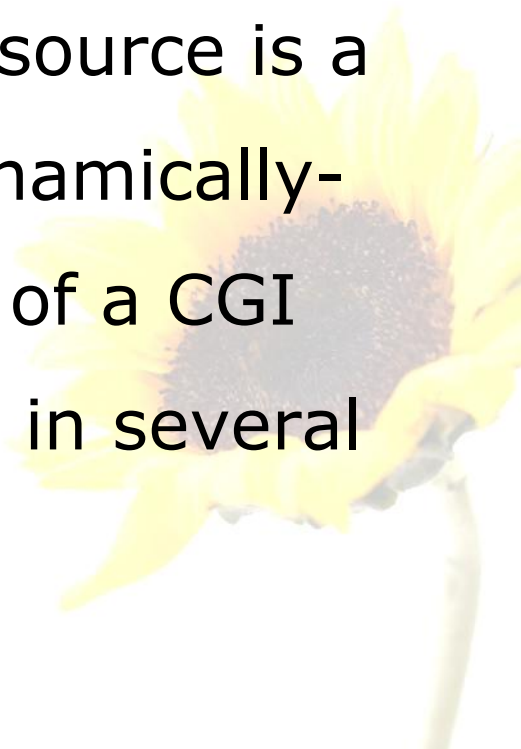
What is HTTP?

- A browser is an HTTP client because it sends requests to an HTTP server (Web server), which then sends responses back to the client. The standard (and default) port for HTTP servers to listen on is 80, though they can use any port.



What are “Resources”?

- HTTP is used to transmit/ resources not just files. A resource is some chunk of information that can be identified by a URL (it’s the R in URL). The most common kind of resource is a file, but a resource may also be dynamically-generated query result, the output of a CGI script, a document that is available in several languages or something else.



What are “Resources”

- While learning HTTP, it may help to think a resource as similar to a file, but more general. As a practical matter almost all HTTP resources are currently either files or server side script output.



Usage

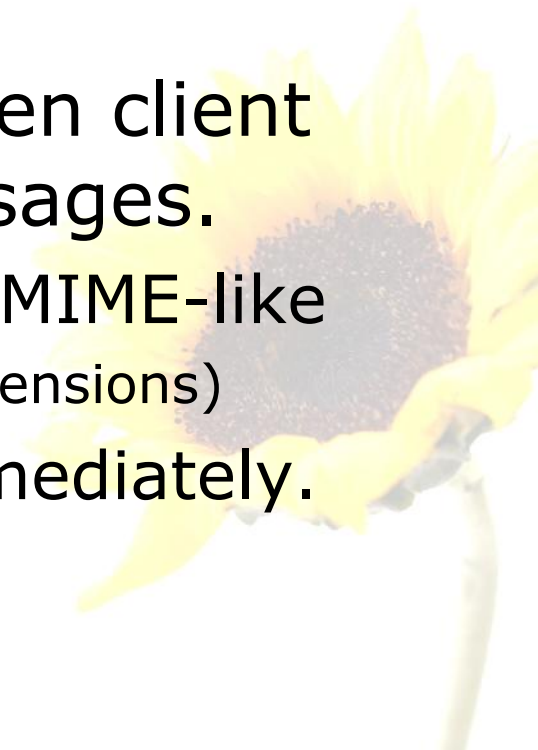
- Used to access data on the WWW
 - Plain text
 - Hypertext (hyper links or links to other pages embedded in them)
 - Audio
 - Video
 - others



Function

Functions like a combination of FTP and SMTP

- FTP as it transfers files and uses the services of TCP.
 - Only one TCP connection
- SMTP as data transferred between client and server looks like SMTP messages.
 - Format of message controlled by MIME-like headers (Multipurpose Internet Mail Extensions)
 - HTTP messages are delivered immediately.



HTTP is very simple

- Client sends a request
- Server sends a response
- Similar to mail request and reply.
- Data in the form of a letter with MIME-like format.



Structure of HTTP Transactions

- Like most network protocols, HTTP uses the client-server model:
- An HTTP client opens a connection and sends a request message to an HTTP server.
- The server then returns a response message, usually containing the source that was requested.



Stateless Protocol

- After delivering the response, the server closes the connection (making HTTP a stateless protocol, i.e. not maintaining any connection information between transactions)
- Cookies are maintained in the client machine.

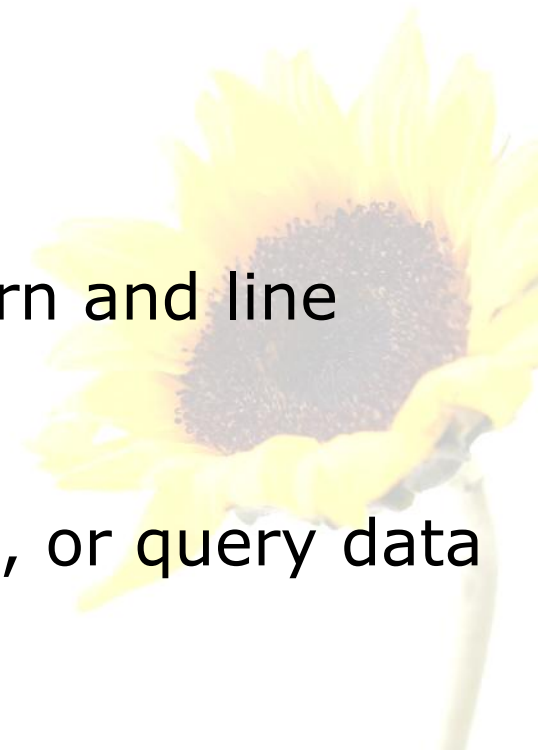


Structure of HTTP Transactions

- The format of the request and response messages are similar , and English-oriented.

Both kinds of messages consist of:

- an initial line.
- Zero or more header lines.
- a blank line (i.e. CRLF(carriage return and line feed) by itself) and
- An optional message body (e.g. a file, or query data or query output)



Structure of HTTP Transactions

- Put another way, the format of an HTTP message is:
- <initial line, different for request vs. response>
- Header1:value1
- Header2:value2
- Header3:value3



Request Messages

Initial line

Headers

Blank line

Body



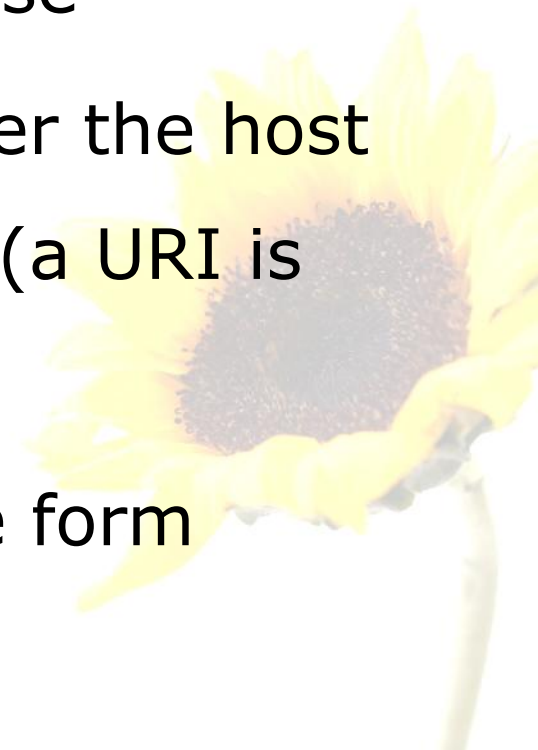
Initial Request Line

- The initial line is different for the request than for the response. A request line has three parts, separated by spaces: a method name, the local path of the requested resource, and the version HTTP being used. A typical request line is:
 - GET/path/to/file/index.html
HTTP/1.0



Initial Request line

- GET is the most common HTTP method: it says “give me this resource”. Other methods include POST and HEAD.
- Method names are always uppercase
- The path is the part of the URL after the host name, also called the request URI (a URI is like URL, but more general)
- The HTTP version always takes the form “HTTP/x.x” uppercase.



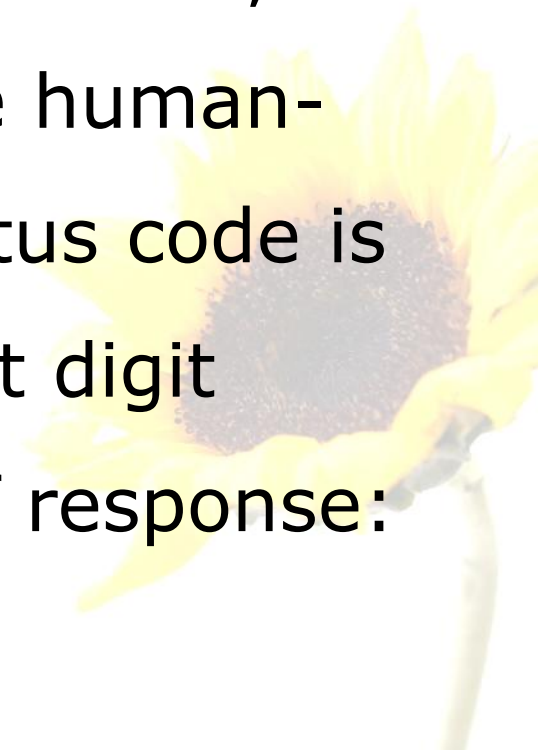
Initial Response Line (Status Line)

- The initial response line, called the status line, also has three parts separated by spaces: the HTTP version, a response status code that gives the result of the request, and an English reason phrase describing the status code. Typical status lines are:
 - HTTP/1.0.200 OK
- Or
 - HTTP/1.0.404 Not Found



Initial Response Line (Status Line)

- The HTTP version is in the same format as in the request line, "HTTP/x.x". The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary. The status code is a three-digit integer, and the first digit identifies the general category of response:



Initial Response Line (Status Line)

- 1xx indicates an informational message only.
- 2xx indicates success of some kind.
- 3xx redirects the client to another URL
- 4xx indicates an error on the client's part.
- 5xx indicates an error on the server's part.



Initial Response Line (Status Line)

- The most common status codes are”
- 200 OK
 - The request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body.
- 404 Not Found
 - The requested resource doesn't exist.
- 301 Moved Permanently.
- 302 Moved Temporarily.



Initial Response Line (Status Line)

- 303 See Other (HTTP 1.1 only)
 - The resource has moved to another URL (given by the Location: response header), and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.
- 500 Server Error
 - An unexpected server error. The most common cause is a server side script that has bad syntax, fails or otherwise can't run correctly.



Header Lines

- Header lines provide information about the request or response, or about the object sent in the message body.

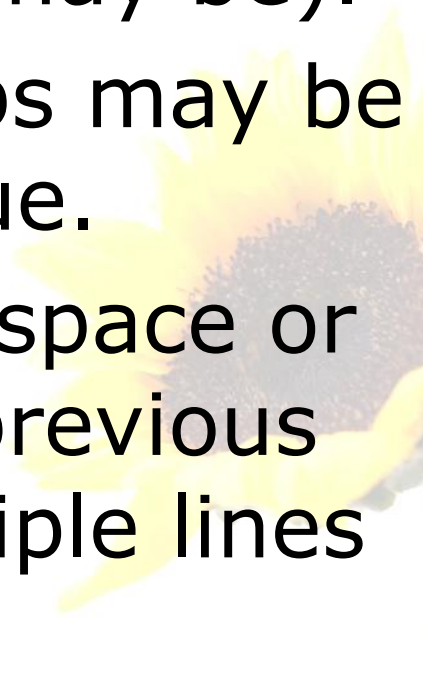


Header Lines

- The header lines are in the usual text header format, which is:
- One line per header
- Of the form “Header-Name: value”
- Ending with CRLF. It’s the same format used for email and news postings.



Header Lines

- As noted above, they should end in CRLF.
 - The header name is not case-sensitive (though the value may be).
 - Any number of spaces or tabs may be between the ":" and the value.
 - Header lines beginning with space or tab are actually part of the previous header line, folded into multiple lines for easy reading.
- 

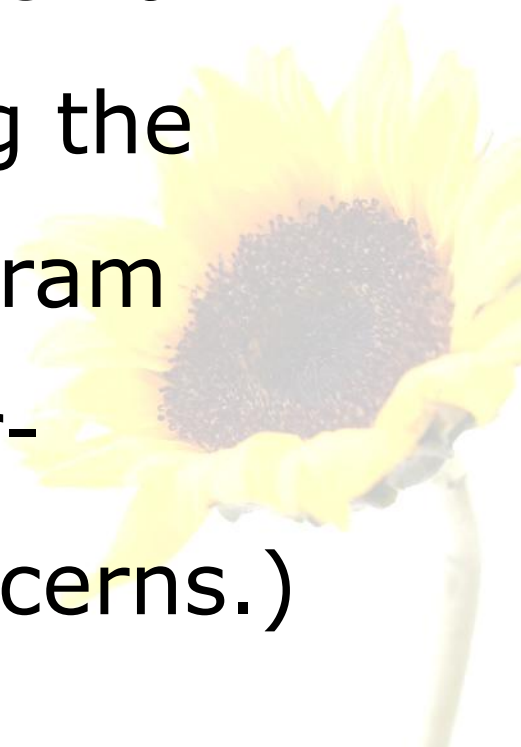
Header Lines

- Thus, the following two headers are equivalent:
 - Header1: some-long-value-1a, some-long-value-1b
 - HEADER1: some-long-value-1a,
– some-long-value-1b.

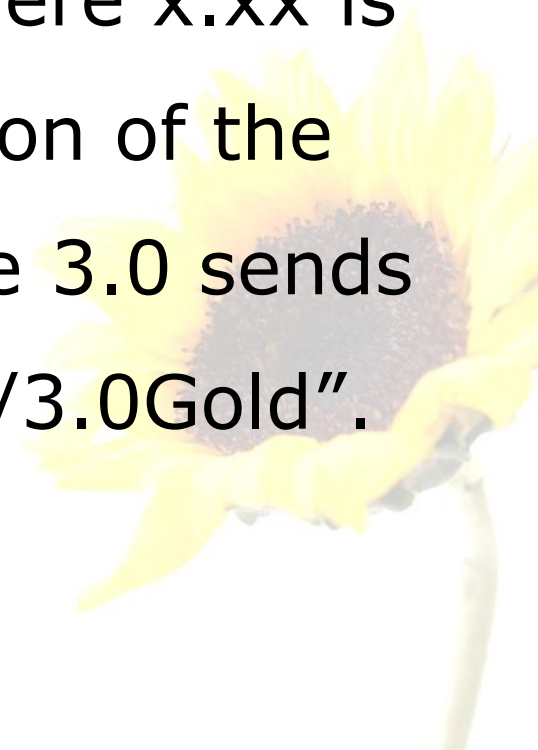


Header Lines

- For Net-politeness, consider including these headers in your requests:
- The From: header gives the email address of whoever's making the request, or running the program doing so. (This must be user-configurable, for privacy concerns.)



- The User-Agent: header identifies the program that's making the request, in the form "Program=name/x.xx", where x.xx is the (mostly) alphanumeric version of the program. For example, Netscape 3.0 sends the header "User-agent: Mozilla/3.0Gold".



Header Lines

- If you're writing servers, consider including these headers in your responses:
- The Server: header is analogous to the User-Agent: header: it identifies the server software in the form "Program-name/x.xx". For example, one beta version of Apache's server returns "Server:Apache/1.2b3-dev".



- The Last-Modified: header gives the modification date of the resource that's being returned. It's used in caching and other bandwidth-saving activities. Use Greenwich Mean Time, in the format
 - Last-Modified: Fri, 31 Dec 1999
23:59:59 GMT



Request Line

- **Request Type**
- **URL**
- **Version**



Request Type

- Categories request messages into several methods.



URL

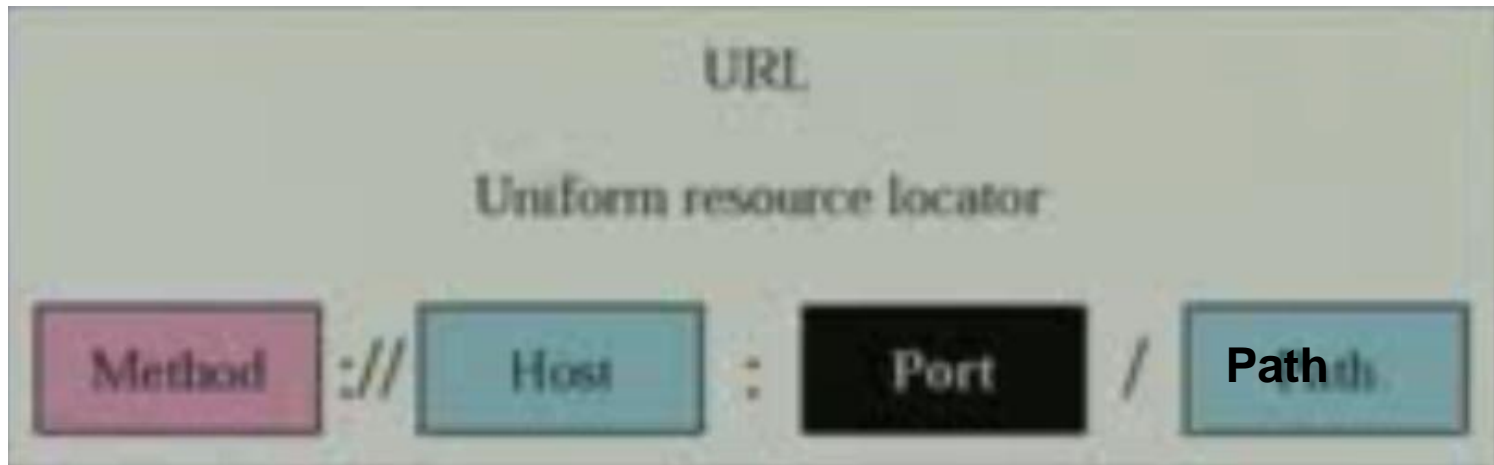
Uniform Resource Locator

Web page/document address

- Method
- Host
- Port Number
- Path



URL



Method

Protocol used to retrieve the document

- Gopher
- FTP
- HTTP
- News
- TELNET



Address

- **Host**
 - **Computer where information is located.**
- **Port Number**
 - **Separated by a colon**
 - **Default is port 80**
- **Path**
 - **Pathname of the file on the server**



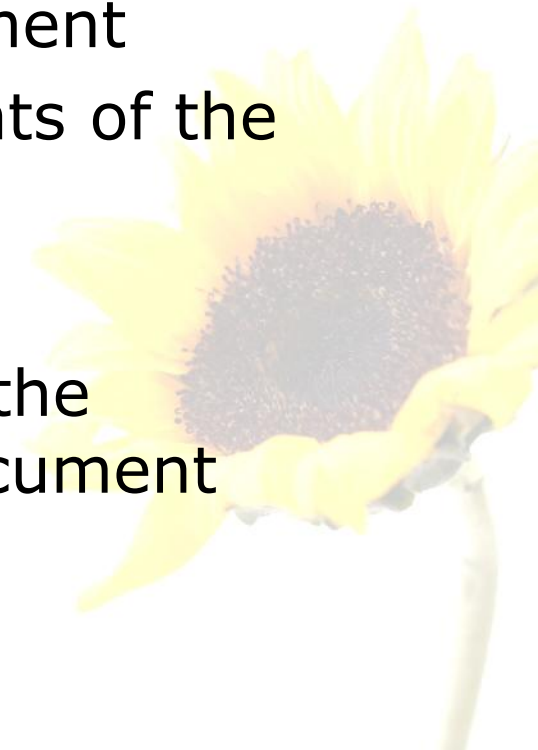
Version

- Current version 1.1
- 1.0 and 0.9 still in use



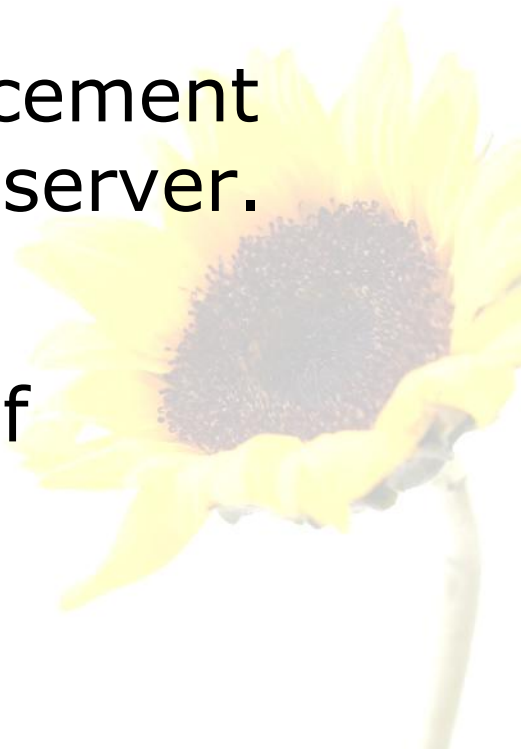
Methods

- actual command or request that a client issues to the server
 - GET
 - client wants to retrieve a document
 - Server responds with the contents of the document
 - HEAD
 - client wants information about the document but not the actual document



Methods

- POST
 - client provides some information to the server
- PUT
 - Client provides a new or replacement document to be stored on the server.
- PATCH
 - Similar to PUT but only a list of differences to be made.



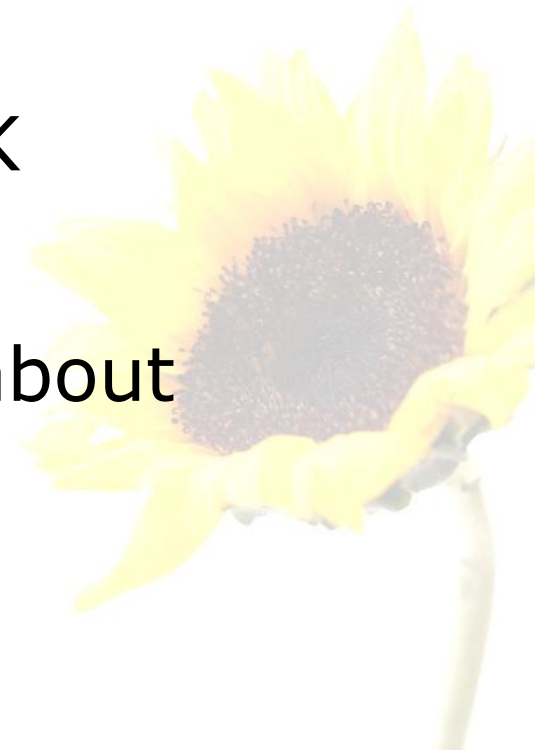
Methods

- COPY
 - copy a file to another location
- MOVE
 - moves a file to another location
- DELETE
 - remove a document on the server



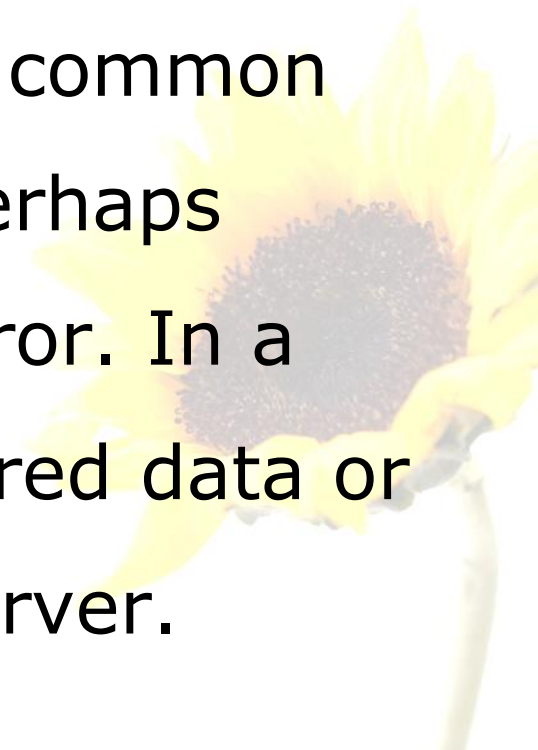
Methods

- LINK
 - creates a link from a document to another location.
- UNLINK
 - delete links created with LINK
- OPTION
 - used by client to ask server about available options



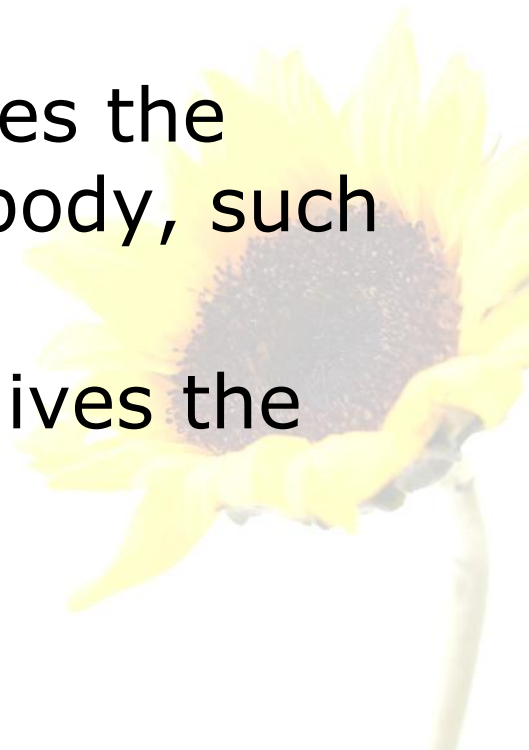
The Message Body

- An HTTP message may have a body of data sent after the header lines. In a response, this is where the requested resource is returned to the client (the most common use of the message body), or perhaps explanatory text if there's an error. In a request, this is where user-entered data or uploaded files are sent to the server.



The Message Body

- If an HTTP message includes a body, there are usually header lines in the message that describe the body. In particular,
 - The Content-Type: header gives the MIME-type of the data in the body, such as text/html or image/gif.
 - The Content-Length: header gives the number of bytes in the body.



Sample HTTP Exchange

- To retrieve the file at the URL
 - <http://www.somehost.com/path/file.html>
- first open a socket to the host www.somehost.com, port 80 (use the default port of 80 because none is specified in the URL)



Sample HTTP Exchange

- Then, send something like the following through the socket:
 - GET/path/file.html HTTP/1.0
 - From: someuser@marshall.com
 - User-Agent: HTTPTool/1.0
 - [blank line here]



Sample HTTP Exchange

- The server should respond with something like the following, sent back through the same socket:
 - HTTP/1.0 200 OK
 - Date: Fri,31 Dec 1999 23:59:59 GMT
 - Content-Type: text/html
 - Content-Length:1354




Sample HTTP Exchange

- `<html>`
- `<body>`
- `<h1> Good Day !! </h1>`
- (more file contents)
- `.`
- `.`
- `.`
- `</body>`
- `</html>`
- After sending the response, the server closes the socket.

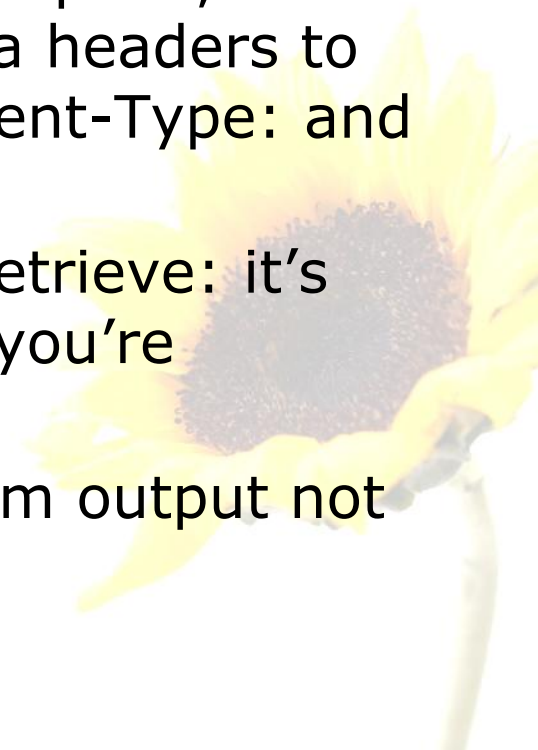


The HEAD Method

- A HEAD request is just like GET request, except it asks the server to return the response headers only, and not the actual resource (i.e. no message body). This is useful to check characteristics of a resource without actually downloading it, thus saving bandwidth. Use HEAD when you don't actually need a file's contents.
 - The response to a HEAD request must never contain a message body, just the status line and headers.
- 

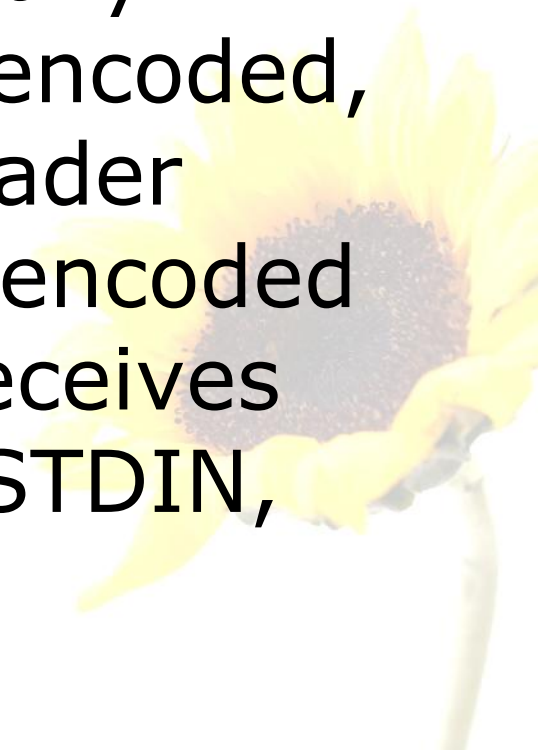
The POST Method

- A POST request is used to send data to the server to be processed in some way, like by a CGI script. A POST request is different from a GET request in the following ways:
 - There's a block of data sent with the request, in the message body. There are usually extra headers to describe this message body, like Content-Type: and Content-Length:.
 - The request URI is not a resource to retrieve: it's usually a program to handle the data you're sending.
 - The HTTP response is normally program output not a static file.



The POST Method

- The most common use of POST, by far, is to submit HTML form data to CGI scripts. In this case, the Content-type: header is usually application/x-www-form-urlencoded, and the Content-Length: header gives the length of the URL-encoded form data. The CGI script receives the message body through STDIN, and decodes it.



The POST Method

- Here's a typical form submission, using POST:
 - POST/path/script.cgi HTTP/1.0
 - From: frog@marshall.com
 - User-Agent: HTTPTool/1.0
 - Content-Type: application/x-www-form-urlencoded
 - Content-Length: 32
 - Home=Cosby&favorite+flavor=flies

