



COSC 2206 Internet Tools

The HTTP Protocol

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



What is TCP/IP?

- TCP: Transmission Control Protocol
- IP: Internet Protocol
- These network protocols provide a standard method for sending and receiving messages over the Internet/
- HTTP sits on top of TCP/IP as an application layer protocol that provides client-server communication.



What is HTTP?

- Hypertext Transport Protocol (1.1)
- It is the protocol that web servers and clients use to communicate on the internet.
- The web client is normally a browser on a client machine.
- Web server is a server such as Apache that receives HTTP requests from a client and sends an HTTP response back to a client.



Try this using windows telnet

- Run apache
- telnet localhost 80
- ^] to get prompt (control +])
- ? to get help (there is an open command)
- o localhost 80
- GET /index.html HTTP/1.1
- Host: localhost
- Now press enter to see HTTP response



**This is the
HTTP
request**

HTTP Response

- HTTP/1.1 200 OK
- Date: Tue, 22 Jun 2004 14:20:03 GMT
- Server: Apache/1.3.29 (Win32) PHP/4.3.7
- Last-Modified: Sat, 06 Dec 2003 15:38:57 GMT
- ETag: "0-76-3fd1f811"
- Accept-Ranges: bytes
- Content-Length: 118
- Content-Type: text/html

**HTTP
response
headers**

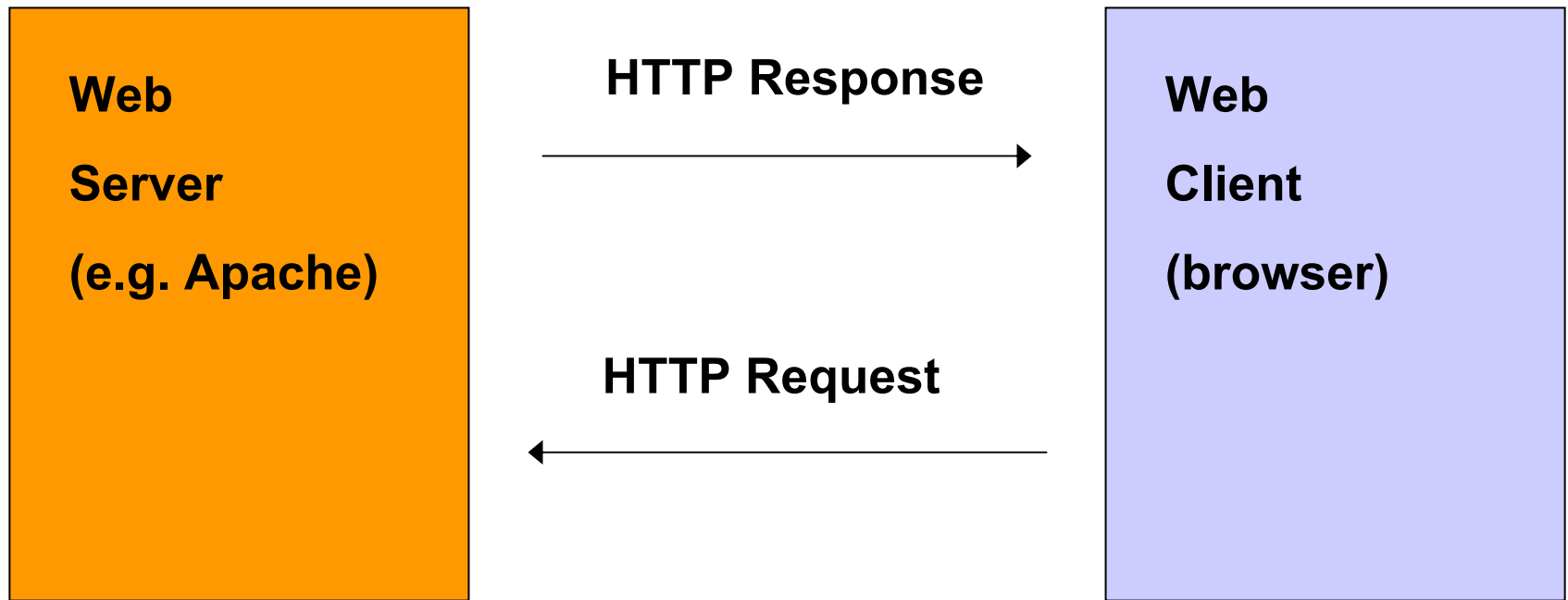
**blank line
is important**

- <html>
- <head><title>My Home Page</title></head>
- <body>
- <h1>My Home Page</h1>
-
- </body>
- </html>

My Home Page



Client/Server communication





Uniform Resource Locator

- Client uses URL to inform server what resource is requested (also a universal resource location). Format is
 - `scheme://host:port/path?queryString#fragment`
- Examples:
 - `http://localhost:8080/index.html`
 - `http://www.cs.laurentian.ca/test.php?name=bob`
- The default port is 80 and isn't specified



Virtual Path

- The path part of a URL is a virtual path and does not need to be a path on the server's file system
- Example
 - `http://localhost:8080/test/servlet/HelloWorld`
- Here the actual directory on the server for the HelloWorld servlet on my computer is
 - `c:/tomcat/webapps/test/WEB-INF/classes`



Absolute Path

- absolute path (absolute URL)
 - completely specifies the path to a resource using a complete URL
- Example
 - `http://www.cs.laurentian.ca/c1046/assign5.html`



Relative Path (no leading /)

- Given a document whose absolute URL is
 - `http://www.cs.laurentian.ca/badams/c2206/test.html`
- Suppose this document contains the links
 - `...`
 - `...`
- Then the absolute URL's are
 - `http://www.cs.laurentian.ca/badams/c2206/test.php`
 - `http://www.cs.laurentian.ca/badams/c2206/jsp/index.html`



Relative Path (leading /)

- Given a document whose absolute URL is
 - `http://www.cs.laurentian.ca/badams/c2206/test.html`
- Suppose this document contains the links
 - `...`
 - `...`
- Then the absolute URL's are
 - `http://www.cs.laurentian.ca/test.php`
 - `http://www.cs.laurentian.ca/jsp/index.html`



Query String (1)

- The query string is an encoded string of name value pairs with the format
 - `?name1=value1&name2=value2& . . .`
- URL encoding is needed to include special characters such as `=`, `#` and `/`. Each special character is encoded as `%HH` where `HH` is the hex representation
- Spaces can be encoded as `+` characters or as `%2B`



Query String (2)

- Encode the characters
 - ; / ? : @ & = + \$,
 - space character: + or %2B
 - delimiters: < > # % "
 - others: { } | \ ^ [] `
- In PHP there is a special function called **rawurlencode** that can do this. Other web languages have similar functions.



Query String (3)

- In a GET request from a client the name-value pairs are sent to the server and are made available to a script such as a PHP script, or to a Java servlet.
- The other way to send values to the server that does not use a query string is using a POST request.
- More on GET and POST later



The HTTP Request

- The client sends information to the server using an HTTP request.
- There are three ways to send the request
 - click on a link in an HTML document that corresponds to a URL on the server:
 - can either retrieve document or execute script
 - type a URL into the browsers location or address field
 - Click on a form submission button



HTTP Request Methods

- The most important methods are
 - GET
 - HEAD
 - POST
- There are several other methods that are not often used:
 - DELETE, OPTION, TRACE, PUT



HTTP GET Method

- Request to retrieve a document or execute a script
- Not supposed to modify data stored on the web server or in an associated database
- Used to return static or dynamic HTML documents and images, results of database queries.
- Can be bookmarked since query string is part of the URL.



HTTP HEAD Method

- Request information about a document such as its last modified date so browser can decide whether to fetch it from server or from cache
- It's like a GET request but no document is sent back by the server.



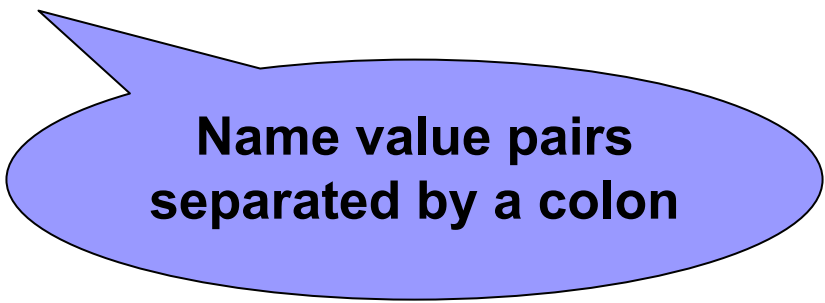
HTTP POST Method

- Used in conjunction with HTML forms to send form data (name-value pairs) to the server.
- After the blank line at end of headers the form data is sent as name-value pairs.
- Use this method if data stored on the server is modified (e.g, rows in database table)
- Cannot be bookmarked.



Example HTTP Request

```
GET index.html HTTP/1.1
Host: localhost
Accept: image/gif, image/jpg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: browser info goes here
```



Name value pairs
separated by a colon



Request Headers

- Accept
 - MIME types the browser will accept
- Accept-Charset
 - the character sets the browser understands
- Accept-Encoding
 - Encodings such as gzip that browser accepts
- Accept-Language
 - languages such as en that browser accepts



Request Headers

- Authorization

- Username/password of browser user

- Connection

- indicates if browser can handle persistent connections for multiple file/image requests

- Content-length

- number of bytes in request content: used only by POST request to give size of post data being sent on server's standard input stream



Request Headers

- Cookie

- Returns name/value pair to server that was set by server on a previous connection.

- Host

- The hostname of the target (required)

- If-Modified-Since

- page should be send only if it has been modified since the specified date



Request Headers

- If-Unmodified-Since
 - opposite of If-Modified-Since (for PUT requests)
- Referer
 - URL that referred user to specified resource
 - The spelling mistake must be made (Use Referer not Referrer)
- User-Agent
 - information on browser making the request



Example HTTP Response

HTTP/1.1 200 OK

status line

Date: date information goes here

Server: Apache/1.3.23 (Unix)

Last-Modified: date info goes here

Content-Length: 141

Content-Type: text/html

blank line
necessary

HTML document index.html goes here



Server Response Status Codes

- 100-199: informational codes
- 200-299: The request was successful
- 300-309: File has moved
 - Location header indicates the new address
 - 301 means that browser will automatically submit a new request for the redirected resource
- 400-499: Client error was made
- 500-599: Server error was made



Common Status Codes

- 200 OK
 - Successful request for a document
 - The document is included in the response
- 204 No Content
- 301 Moved Permanently
- 302 Found (Location header)
- 404 Not Found
- 500 Internal Server Error



MIME Types

- Multi-part internet mail extensions
- Examples:
 - text/plain, text/html
 - image/gif, image/jpg, image/png
 - application/x-gzip
 - application/zip



Response Headers

- Allow
 - request methods such as GET, POST that server supports
- Cache-Control
 - various options for client side caching
- Connection
 - whether to use persistent connections or not



Response Headers

- Content-Encoding
 - gzip, for example
- Content-Language
 - language of the document
- Content-Base
 - base URL used to resolve relative URL's in the document



Response Headers

- Content-Length
 - Length in bytes of the response's content
- Content-Type
 - Media type
 - e.g., text/plain, text/html, or image/jpg
- Expires
 - for a document that changes frequently this tells browser not to use a cached version



Response Headers

- Last-Modified

- when the document was last changed. Useful in caching

- Location

- new document address

- Refresh

- refresh document again after specified number of seconds



Response Headers

- **Retry-After**
 - how soon to repeat document request
- **Server**
 - Name and version of the Web server
- **Set-Cookie**
 - request to have browser set a cookie and return it on future requests

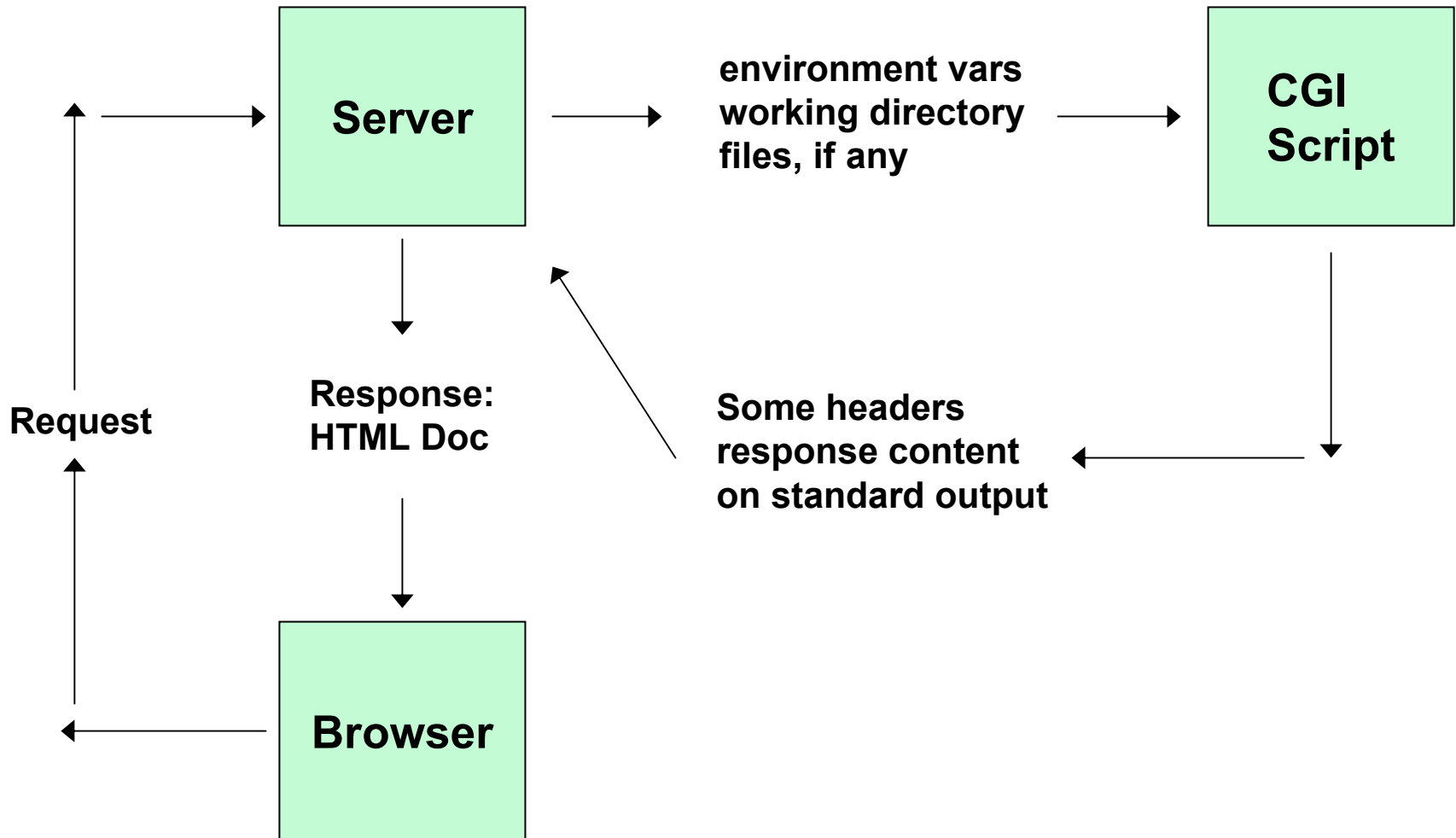


Response Headers

- WWW-Authenticate
 - Specifies the authorization scheme and the realm

- Common Gateway Interface
- The original way for servers to provide dynamic content by executing external server-side scripts (e.g. Perl)
- Server provides script with an environment
- Script provides the dynamic page
- Inefficient since a new process is started on the server for each client request

CGI





CGI Environment Variables (1)

- **DOCUMENT_ROOT**

- Root directory of your web server

- **HTTP_COOKIE**

- Client's cookie, if one has been set

- **HTTP_HOST**

- The host name of the web server

- **HTTP_REFERER**

- URL of page that called your script



CGI Environment Variables (2)

- **HTTP_USER_AGENT**

- Client's browser type string

- **HTTPS**

- indicates if script is invoked by a secure server

- **PATH**

- Path to your server

- **QUERY_STRING**

- string of name value pairs sent by client



CGI Environment Variables (3)

- **REMOTE_ADDR**
 - IP address of the client
- **REMOTE_HOST**
 - IP address or hostname of the client
- **REMOTE_PORT**
 - Port tht client is connected to
- **REMOTE_USER**
 - client's user name if applicable



CGI Environment Variables (4)

- **REQUEST_METHOD**
 - GET or POST
- **REQUEST_URI**
 - Document path relative to document root
- **SCRIPT_FILENAME**
 - Full path name of the script
- **SCRIPT_NAME**
 - path relative to document root



CGI Environment Variables (5)

- **SERVER_ADMIN**

- email address of server's web master

- **SERVER_NAME**

- The URL of the server

- **SERVER_PORT**

- Port number on which server is listening

- **SERVER_SOFTWARE**

- String describing the server software and version