LIS 900C Webmastering I: the static web page

Lecture 1

Thomas Krichel

2002–05–13

Reading

Fielding, Roy T., James Gettys, Jeffrey C. Mogul, Paul J. Leach, Tim Berners-Lee "Hypertext Transfer Protocol – HTTP/1.1" (1999), RFC 2616

Hypertext Transfer Protocol HTTP

An application-level protocol for *distributed*, *collaborative*, hypermedia information systems.

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP, NNTP, FTP, Gopher, and WAIS protocols. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

history

1990: version 0.9 allows for transfer of raw data.

1996: rfc1945 defines version 1.0. by adding *attribute*:*value* headers.

1999: rfc 2616 adds support for hierarchical proxies, caching, virtual hosts and some support for persistent connections, and is more stringent.

overall operation, client

Client sends request, required items are

- method
- request URI
- protocol version

optional items are

- request modifiers
- client information
- body

overall operation, server

Server sends response, required items are

- status line
  - protocol version
  - success or error code

optional items are

- server information
- body

intermediaries

They come in three flavors

- proxies, i.e. forwarding agents
- gateways, i.e. receiving agents
- tunnels, i.e. relay points that do not change the message

http assumes transport

http assumes that there is a reliable way to transport data from one host on the Internet to another one.

All http requests and responses are separate TCP connections. The default is TCP port 80, but other ports can be used.

http resource identification

identification of resources is assumed through Uniform Resource Identifiers (URI).

As far as http is concerned, URIs are string.

http can use “absolute” and “relative” URIs.

A URL is a special case of a URI.

the absolute http URL

http://*host*[**:***port*][[*abs_path*][**?***query*]]

If *abs_path* is empty, it is **/**. The scheme name **http** and the host name are case-insensitive.

Characters other than those in the "reserved" and "unsafe" sets RFC 2396 are equivalent to their "% *HEX HEX*" encoding.

character sets

A character set is a method used with one of more tables to convert a sequence of binary digits into a sequence of characters.

http shares the same registry as the MIME multimedia email extensions. It is based at the IANA.

The default character set is ISO-8859-1.

Data can be encoded with the methods "gzip", "compress", "deflate", and "identity".

http messages

There are two types of messages.

Requests are sent form the client to the server.

Responses are sent from the server to the client.

The generic format is the same as for email messages:

- start line
- message headers
- empty line
- body

Empty lines before the start line are ignored. A request's start line is called a request-line, the response start line is called a status-line.

message headers

headers have lines that are of the form

field-name: *field-value*

Leading and trailing whitespace at the colon is ignored. Field names are case-insensitive. Order of lines does not matter.

the request line

The request line is of the form

method URI protocol-version

method takes the values "OPTIONS", "GET", "HEAD", "POST", "PUT", "DELETE", "TRACE", "CONNECT".

A valid request line is

```
GET http://openlib.org/home/krichel HTTP/1.1
```

If the URL is relative, the host is transmitted with the "host" header field.

(Why does the host need to know the host name?)

the request headers

Accept:	Accept-Charset:
Accept-Encoding:	Accept-Language:
Authorization:	Expect:
From:	Host:
If-Match:	If-Modified-Since:
If-None-Match:	If-Range:
If-Unmodified-Since:	Max-Forwards:
Proxy-Authorization:	Range:
Referer:	TE:
User-Agent:	

the status line

The status line is a set of lines that are of the form

HTTP-Version Status-Code Reason-Phrase

The status code is a 3-digit number used by the computer and the reason line is a friendly note for a human to read.

the status code classes

1 Informational – Request received, continuing process
2 Success: The action was successfully received, understood, and accepted
3 Redirection – Further action must be taken in order to complete the request
4 Client Error – The request contains bad syntax or cannot be fulfilled
5 Server Error – The server failed to fulfill an apparently valid request

the status code values

100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Time-out
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Large
415	Unsupported Media Type
416	Requested range not satisfiable
417	Expectation Failed
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out
505	HTTP Version not supported

the response headers

Accept-Ranges:	Age:
ETag:	Location:
Proxy-Authenticate:	Retry-After:
Server:	Vary:
WWW-Authenticate:	

the entity headers, common to request and response

Allow:	Content-Encoding:	Content-Language:
Content-Length:	Content-Location:	Content-MD5:
Content-Range:	Content-Type:	Expires:
Last-Modified		

The entity-body (if any) sent with an HTTP request or response is in a format and encoding defined by the entity-header fields.

When an entity-body is included with a message, the data type of that body is determined via the header fields Content-Type and Content-Encoding

The GET and HEAD method

The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

The semantics of the GET method change to a "conditional GET" if the request message includes an If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field.

The semantics of the GET method change to a "partial GET" if the request message includes a Range header field. A partial GET requests that only part of the entity be transferred,

The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.

The POST method

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions:

- Annotation of existing resources;
- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
- Providing a block of data, such as the result of submitting a form, to a data-handling process;
- Extending a database through an append operation.

The PUT & DELETE method

The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server.

The DELETE method requests that the origin server delete the resource identified by the Request-URI.