

Hypermedia and the Web

Overview of HTTP

HTTP is a request-response protocol.

Versions of HTTP:

Draft Standard (most recent): RFC 2616

Proposed Standard (influential when most current server implementations were performed): RFC 2068

<http://www.ietf.org/rfc/rfc2616.txt>

<http://www.ietf.org/rfc/rfc2068.txt>

Hypertext Transfer Protocol -- HTTP/1.1

Roy Fielding, Jim Gettys, Jeff Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, Tim Berners-Lee

Internet Draft Standard Request for Comments (RFC) 2616, June 1999.

HTTP/1.1 verbs:

GET – read a resource (receive a representation)

HEAD – return only read metadata

POST – submit form data, file upload, tunnel other protocols

PUT – write a resource (submit a representation for persistent storage)

DELETE – remove a resource (exact semantics are unclear: delete, destroy, or just mark for deletion?)

OPTIONS – return methods and capabilities supported by a resource

TRACE – used in protocol debugging, sends received message back in body of response

CONNECT – reserved for use with SSL tunneling

WebDAV verbs:

COPY – copy a resource or resource tree

MOVE – move a resource or resource tree

MKCOL – create a collection (directory/folder)

LOCK – lock a resource (exclusive write lock or shared write lock)

UNLOCK – unlock a resource

PROPFIND – request properties/metadata (perform a directory listing)

PROPPATCH – write properties/metadata

More on these in a few weeks. Described in RFC 2518, currently undergoing revision.

Representational State Transfer (REST):

There are a few core ideas underlying the design of HTTP.

Ship representations of resources, not actual resource. When a server responds to a GET request, what is being sent over the network is an on-the-wire representation of the resource, not the resource itself. Permits computational resources, where the resource itself is source code, but the representation is the HTML output of the resource. Also permits content negotiation, where one of a set of different representations or language translations can be returned. This is a key difference between HTTP and remote filesystem protocols, where the expectation is that you're sending/receiving the actual resource (creates many challenges for authoring, though).

Interface genericity. All resources should support (or have the ability to support) a small set of verbs (the HTTP methods). Hence, almost all HTTP resources support GET, HEAD, OPTIONS, and applications can be developed to depend on this simple interface being uniformly available. Contrast with remote procedure call protocols, where each class of resource can (and usually does) support a different interface.

REST is the name for the architectural style that encompasses these two principles, which were used in the development of HTTP (there is some debate over whether WebDAV is, or is not, RESTful).

Ideas were developed and described in Roy Fielding's PhD dissertation (UC Irvine, 2000).
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

HTTP headers

General headers:

Can appear on request or response:

Date: date and time of message origination

Pragma: send directives to recipient of the message

(Pragma: no-cache – informs proxies in the path not to return a cached copy)

Request headers

Authorization: send authentication credentials

From: provide email address as identification (used by spiders, or agents)

If-Modified-Since: {date} - only send resource if modified since the given date. If resource hasn't changed, return 304 Not Modified response.

Referer: URI of the resource from which the Request-URI was obtained (source page of a link)

User-Agent: information about browser version, client machine operating system:

User-Agent: Mozilla/4.03 (Macintosh; I; 68K, Nav)

Often used by servers to provide browser-specific versions of Web pages.

Response headers:

Location: {url} – used to redirect the request to another location, or to identify where a new resource was created

Server – provides details about the server implementation

Server: Apache/1.x.y mod_perl mod_ssl mod_dav

WWW-Authenticate: used to issue a challenge to the client to provide authentication credentials

Entity headers:

Provides information about the body of the message.

Content-Type: MIME media type of the body

text/...

application/...

Content-Encoding: compression kind, if any

Content-Encoding: x-gzip

Content-Length: length of entity body, in bytes

Expires: after the given time, caches should consider the results to be stale

Last-Modified: the time at which the resource was last changed

Etag: entity tag – a unique identifier for the state of a resource

Response classes:

1xx informational

2xx success

200 OK

201 Created

3xx Redirection

301 Moved Permanently

302 Moved Temporarily

4xx client error

401 Unauthorized

403 Forbidden – often an OS permissions problem

404 Not Found – resource was deleted

5xx server errors