

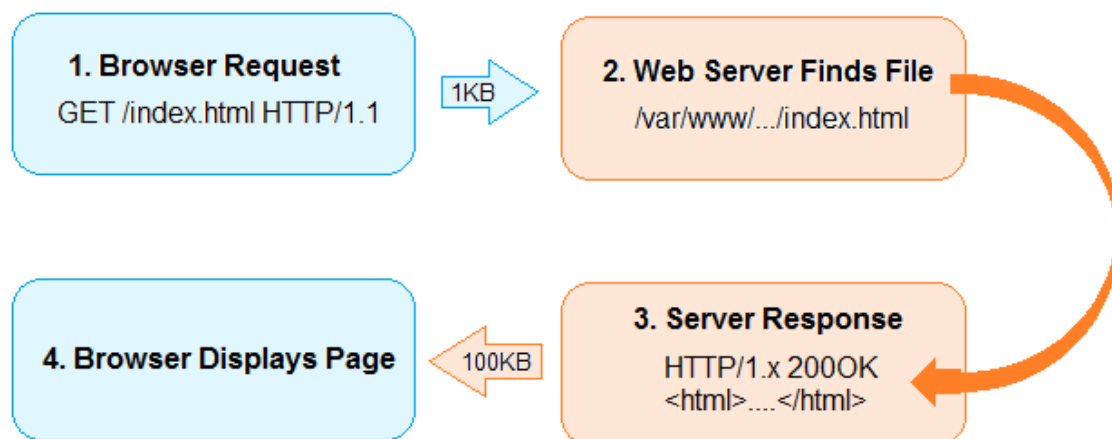
HTTP协议介绍

HTTP(Hypertext Transport Protocol) 超文本传输协议。是在应用层的一种协议，一般而言，在用户空间实现。HTTP 协议最初设计的目的是用来传送和接收 HTML 格式的文档，事实上早期的 HTTP 协议就是仅可以用于传输 HTML 文档，在 HTTP/1.0 版本之后，引入 MIME (多用途互联网邮件扩展MIME, Multipurpose Internet Mail Extensions) 机制后，才支持多媒体数据的处理。在 HTTP/1.0 版本之后，有了保持连接，缓存等功能，使之更加强大。2015年5月，作为互联网标准，正式发布了 HTTP/2 版本。

服务器端实现http协议的软件有：IIS、httpd、nginx、lighttpd、gws。

相关概念

HTTP请求/响应过程



HTTP事务

一次完全的请求及其相对应的响应过程。

HTTP特性

无连接、无状态。

URL

统一资源定位符 (Uniform / Universal Resource Locator), 有时也被俗称为网页地址 (网址)。如同在网络上的门牌, 是因特网上标准的资源的地址 (Address)。

HTTP方法

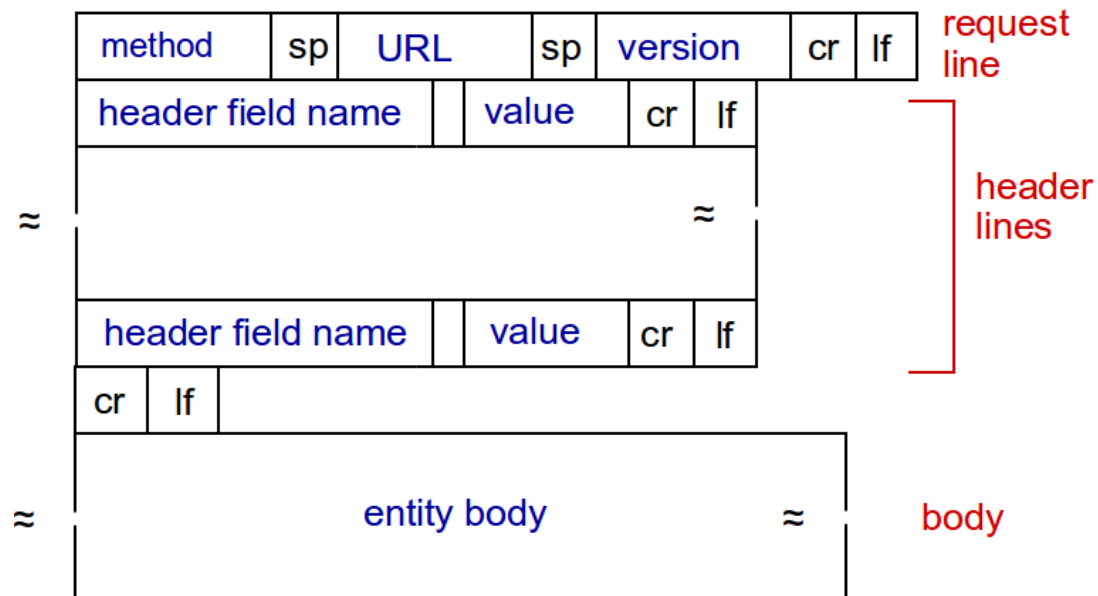
客户端向服务器请求的方式, 常见的有GET、HEAD、PUT、POST、DELETE、OPTIONS、TRACE、

扩展的方法有: LOCK、MKCOL、COPY、MOVE

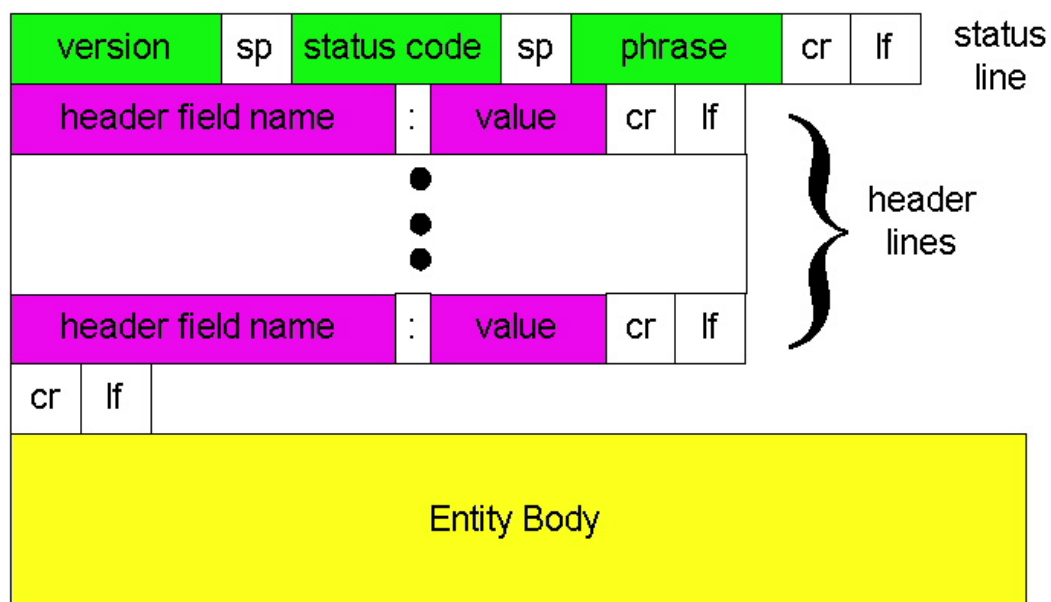
- 1 **GET:** 请求获取一个资源, 需要服务器发送
- 2 **HEAD:** 和GET相似, 但不需要服务器发送资源, 仅传回响应首部
- 3 **POST:** 支持HTML表单提交, 表单中有用户填入数据, 数据发送到服务器, 由服务器存储
- 4 **PUT:** 向服务器写入文档
- 5 **DELETE:** 请求删除url指向的文件
- 6 **OPTIONS:** 探测服务器端对某资源支持的请求方法
- 7 **TRACE:** 跟踪请求经过的防火墙、代理或者网关

HTTP报文

HTTP请求



HTTP响应



HTTP状态码参考

HTTP Status Codes

For great REST services the correct usage of the correct HTTP status code in a response is vital.

1xx – Informational	2xx – Successful	3xx – Redirection	4xx – Client Error	5xx – Server Error
<p>This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line</p> <p>100 – Continue 101 – Switching Protocols 102 – Processing</p>	<p>This class of status code indicates that the client's request was successfully received, understood, and accepted.</p> <p>200 – OK 201 – Created 202 – Accepted 203 – Non-Authoritative Information 204 – No Content 205 – Reset Content 206 – Partial Content 207 – Multi-Status</p>	<p>This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.</p> <p>300 – Multiple Choices 301 – Moved Permanently 302 – Found 303 – See Other 304 – Not Modified 305 – Use Proxy 307 – Temporary Redirect</p>	<p>The 4xx class of status code is intended for cases in which the client seems to have erred.</p> <p>400 – Bad Request 401 – Unauthorised 402 – Payment Required 403 – Forbidden 404 – Not Found 405 – Method Not Allowed 406 – Not Acceptable 407 – Proxy Authentication Required 408 – Request Timeout 409 – Conflict 410 – Gone 411 – Length Required 412 – Precondition Failed 413 – Request Entity Too Large 414 – Request URI Too Long 415 – Unsupported Media Type 416 – Requested Range Not Satisfiable 417 – Expectation Failed 422 – Unprocessable Entity 423 – Locked 424 – Failed Dependency 425 – Unordered Collection 426 – Upgrade Required</p>	<p>Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.</p> <p>500 – Internal Server Error 501 – Not Implemented 502 – Bad Gateway 503 – Service Unavailable 504 – Gateway Timeout 505 – HTTP Version Not Supported 506 – Variant Also Negotiates 507 – Insufficient Storage 510 – Not Extended</p>


Examples of using HTTP Status Codes in REST

201 – When doing a POST to create a new resource it is best to return 201 and not 200.
204 – When deleting a resources it is best to return 204, which indicates it succeeded but there is no body to return.
301 – If a 301 is returned the client should update any cached URI's to point to the new URI.
302 – This is often used for temporary redirect's, however 303 and 307 are better choices.
409 – This provides a great way to deal with conflicts caused by multiple updates.
501 – This implies that the feature will be implemented in the future.

Special Cases

306 – This status code is no longer used. It used to be for switch proxy.
418 – This status code from RFC 2324. However RFC 2324 was submitted as an April Fools' Joke. The message is *I am a teapot*.

Key	Description
Black	HTTP version 1.0
Blue	HTTP version 1.1
Aqua	Extension RFC 2295
Green	Extension RFC 2518
Yellow	Extension RFC 2774
Orange	Extension RFC 2817
Purple	Extension RFC 3648
Red	Extension RFC 4918



BARONE, BUDGE & DOMINICK
Business Technology Solutions

REST is a style of development for exposing data in such a way that it is easy to consume, easy to produce and makes use of HTTP.
REST is not a replacement for SOAP or WS* but an alternative to solve certain problems better.

Architecture

参考

<https://zh.wikipedia.org/wiki/HTTP头字段列表>