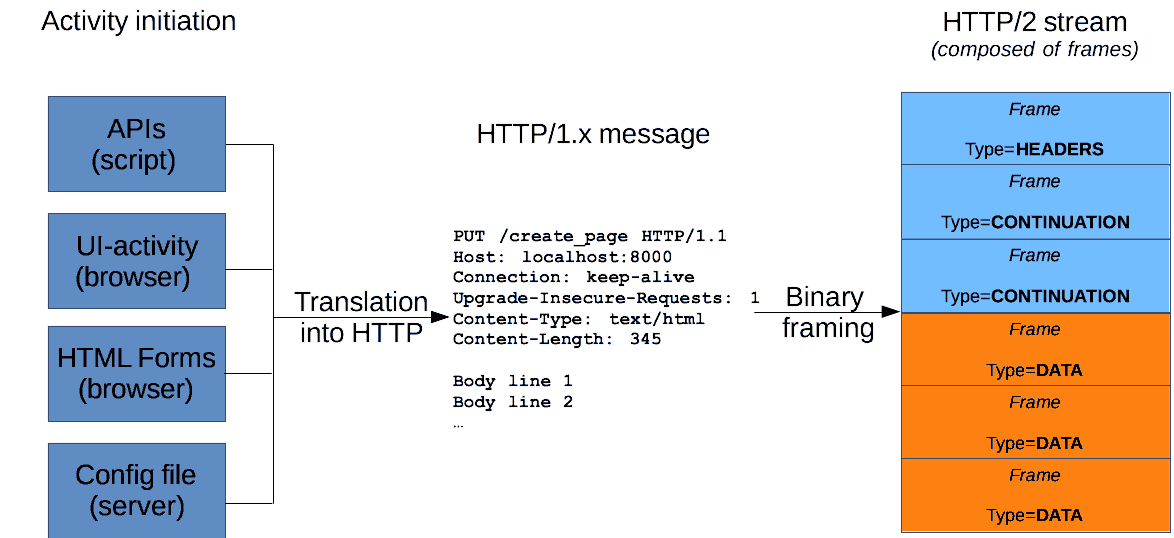
HTTP消息是服务器和客户端之间交换数据的方式。有两种类型的消息︰ 请求--由客户端发送用来触发一个服务器上的动作；响应--来自服务器的应答。

HTTP消息由采用ASCII编码的多行文本构成。在HTTP/1.1及早期版本中，这些消息通过连接公开地发送。在HTTP/2中，为了优化和性能方面的改进，曾经可人工阅读的消息被分到多个HTTP帧中。

Web 开发人员或网站管理员，很少自己手工创建这些原始的HTTP消息︰ 由软件、浏览器、 代理或  服务器完成。他们通过配置文件（用于代理服务器或服务器），API （用于浏览器）或其他接口提供HTTP消息。

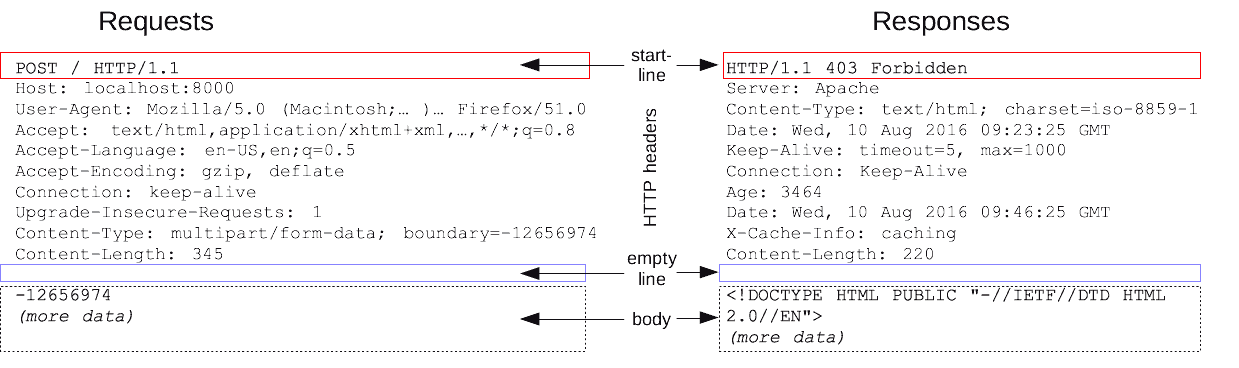


HTTP/2二进制框架机制被设计为不需要改动任何API或配置文件即可应用︰ 它大体上对用户是透明的。

HTTP 请求和响应具有相似的结构，由以下部分组成︰

1. 一行起始行用于描述要执行的请求，或者是对应的状态，成功或失败。这个起始行总是单行的。
2. 一个可选的HTTP头集合指明请求或描述消息正文。
3. 一个空行指示所有关于请求的元数据已经发送完毕。
4. 一个可选的包含请求相关数据的正文 (比如HTML表单内容), 或者响应相关的文档。 正文的大小有起始行的HTTP头来指定。

起始行和  HTTP 消息中的HTTP 头统称为请求头，而其有效负载被称为消息正文。



**HTTP 请求**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Messages$edit#HTTP_%E8%AF%B7%E6%B1%82)

起始行

HTTP请求是由客户端发出的消息，用来使服务器执行动作。*起始行 (start-line)*包含三个元素：

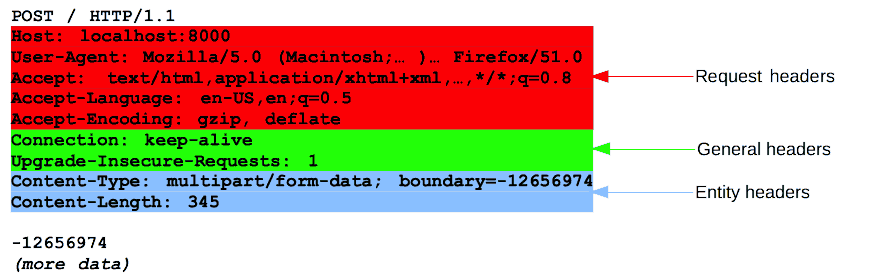
1. 一个 [*HTTP 方法*](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods)，一个动词 (像 [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET), [PUT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/PUT) 或者 [POST](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/POST)) 或者一个名词 (像 [HEAD](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/HEAD) 或者 [OPTIONS](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/OPTIONS)), 描述要执行的动作. 例如, GET 表示要获取资源，POST 表示向服务器推送数据 (创建或修改资源, 或者产生要返回的临时文件)。
2. *请求目标 (request target)，*通常是一个 [URL](https://developer.mozilla.org/en-US/docs/Glossary/URL)，或者是协议、端口和域名的绝对路径，通常以请求的环境为特征。请求的格式因不同的 HTTP 方法而异。它可以是：
   * 一个绝对路径，末尾跟上一个 ' ? ' 和查询字符串。这是最常见的形式，称为 *原始形式 (origin form)*，被 GET，POST，HEAD 和 OPTIONS 方法所使用。  
     POST / HTTP 1.1  
     GET /background.png HTTP/1.0  
     HEAD /test.html?query=alibaba HTTP/1.1  
     OPTIONS /anypage.html HTTP/1.0
   * 一个完整的URL，被称为 *绝对形式 (absolute form)*，主要在 GET 连接到代理时使用。  
     GET http://developer.mozilla.org/en-US/docs/Web/HTTP/Messages HTTP/1.1
   * 由域名和可选端口（以':'为前缀）组成的 URL 的 authority component，称为 *authority form*。 仅在使用 CONNECT 建立 HTTP 隧道时才使用。  
     CONNECT developer.mozilla.org:80 HTTP/1.1
   * *星号形式 (asterisk form)*，一个简单的星号('\*')，配合 OPTIONS 方法使用，代表整个服务器。  
     OPTIONS \* HTTP/1.1
3. *HTTP 版本 (HTTP version*)*，*定义了剩余报文的结构，作为对期望的响应版本的指示符。

Headers

来自请求的 [HTTP headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) 遵循和 HTTP header 相同的基本结构：不区分大小写的字符串，紧跟着的冒号 (':') 和一个结构取决于 header 的值。 整个 header（包括值）由一行组成，这一行可以相当长。

有许多请求头可用，它们可以分为几组：

* *General headers，*例如 [Via](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Via)，适用于整个报文。
* *Request headers，*例如[User-Agent](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/User-Agent)，[Accept-Type](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Accept-Type)，通过进一步的定义(例如 [Accept-Language](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Accept-Language))，或者给定上下文(例如 [Referer](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Referer))，或者进行有条件的限制 (例如 [If-None](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/If-None)) 来修改请求。
* *Entity headers，*例如 [Content-Length](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Length)，适用于请求的 body。显然，如果请求中没有任何 body，则不会发送这样的头文件。



Body

请求的最后一部分是它的 body。不是所有的请求都有一个 body：例如获取资源的请求，GET，HEAD，DELETE 和 OPTIONS，通常它们不需要 body。 有些请求将数据发送到服务器以便更新数据：常见的的情况是 POST 请求（包含 HTML 表单数据）。

Body 大致可分为两类：

* Single-resource bodies，由一个单文件组成。该类型 body 由两个 header 定义： [Content-Type](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type) 和 [Content-Length](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Length).
* [Multiple-resource bodies](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#multipartform-data)，由多部分 body 组成，每一部分包含不同的信息位。通常是和  [HTML Forms](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms) 连系在一起。

**HTTP 响应**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Messages$edit#HTTP_%E5%93%8D%E5%BA%94)

状态行

HTTP 响应的起始行被称作 *状态行* *(status line)*，包含以下信息：

1. *协议版本*，通常为 HTTP/1.1。
2. *状态码 (status code)*，表明请求是成功或失败。常见的状态码是 [200](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/200)，[404](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/404)，或 [302](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/302)。
3. *状态文本 (status text)*。一个简短的，纯粹的信息，通过状态码的文本描述，帮助人们理解该 HTTP 消息。

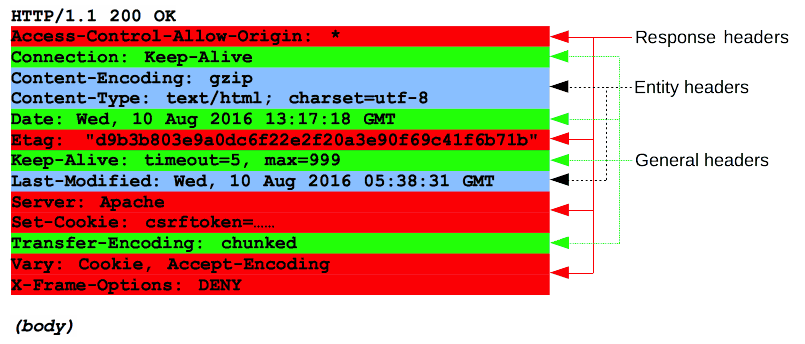
一个典型的状态行看起来像这样：HTTP/1.1 404 Not Found。

Headers

响应的  [HTTP headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) 遵循和任何其它 header 相同的结构：不区分大小写的字符串，紧跟着的冒号 (':') 和一个结构取决于 header 类型的值。 整个 header（包括其值）表现为单行形式。

有许多响应头可用，这些响应头可以分为几组：

* *General headers，*例如 [Via](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Via)，适用于整个报文。
* *Response headers，*例如 [Vary](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Vary) 和 [Accept-Ranges](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Accept-Ranges)，提供其它不符合状态行的关于服务器的信息。
* *Entity headers*，例如 [Content-Length](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Length)，适用于请求的 body。显然，如果请求中没有任何 body，则不会发送这样的头文件。



Body

响应的最后一部分是 body。不是所有的响应都有 body：具有状态码 (如 [201](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/201) 或 [204](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/204)) 的响应，通常不会有 body。

Body 大致可分为三类：

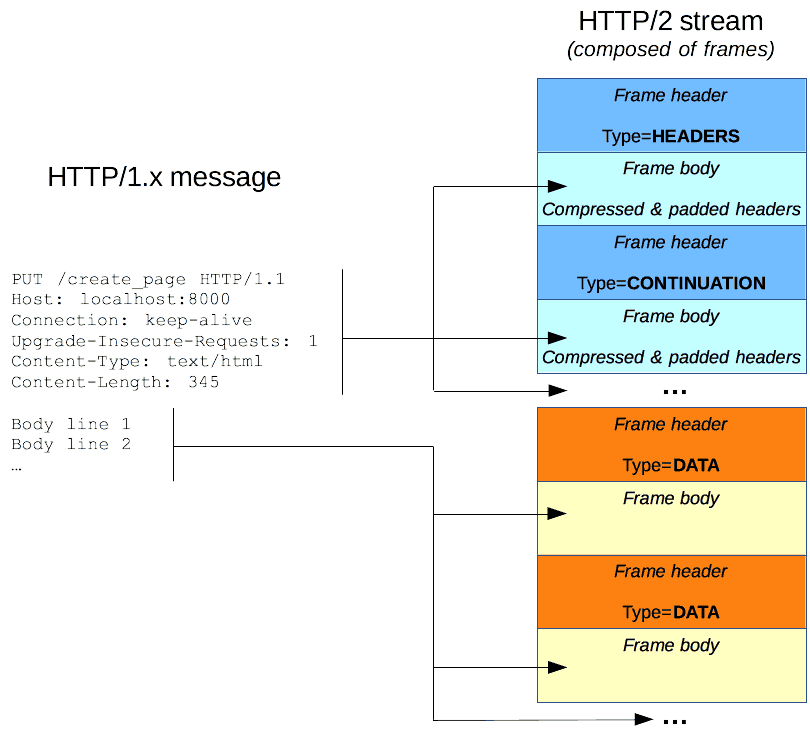
* Single-resource bodies，由**已知**长度的单个文件组成。该类型 body 由两个 header 定义：[Content-Type](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type) 和 [Content-Length](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Length)。
* Single-resource bodies，由**未知**长度的单个文件组成，通过将 [Transfer-Encoding](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Transfer-Encoding) 设置为 chunked 来使用 chunks 编码。
* [Multiple-resource bodies](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#multipartform-data)，由多部分 body 组成，每部分包含不同的信息段。但这是比较少见的。

**HTTP/2 帧**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Messages$edit#HTTP2_%E5%B8%A7)

HTTP/1.x 报文有一些性能上的缺点：

* Header 不像 body，它不会被压缩。
* 两个报文之间的 header 通常非常相似，但它们仍然在连接中重复传输。
* 无法复用。当在同一个服务器打开几个连接时：TCP 热连接比冷连接更加有效。

HTTP/2 引入了一个额外的步骤：它将 HTTP/1.x 消息分成帧并嵌入到流 (stream) 中。数据帧和报头帧分离，这将允许报头压缩。将多个流组合，这是一个被称为 *多路复用 (multiplexing)*的过程，它允许更有效的底层 TCP 连接。



HTTP 帧现在对 Web 开发人员是透明的。在 HTTP/2 中，这是一个在  HTTP/1.1 和底层传输协议之间附加的步骤。Web 开发人员不需要在其使用的 API 中做任何更改来利用 HTTP 帧；当浏览器和服务器都可用时，HTTP/2 将被打开并使用。

**结论**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Messages$edit#%E7%BB%93%E8%AE%BA)

HTTP 报文是使用 HTTP 的关键；它们的结构简单，并且具有高可扩展性。HTTP/2 帧机制是在 HTTP/1.x 语法和底层传输协议之间增加了一个新的中间层，而没有从根本上修改它，即它是建立在经过验证的机制之上。