09-HTTP报文是什么样子的？

在上一讲里，我们在本机的最小化环境了做了两个HTTP协议的实验，使用Wireshark抓包，弄清楚了HTTP协议基本工作流程，也就是“请求-应答”“一发一收”的模式。

可以看到，HTTP的工作模式是非常简单的，由于TCP/IP协议负责底层的具体传输工作，HTTP协议基本上不用在这方面操心太多。单从这一点上来看，所谓的“超文本传输协议”其实并不怎么管“传输”的事情，有点“名不副实”。

那么HTTP协议的核心部分是什么呢？

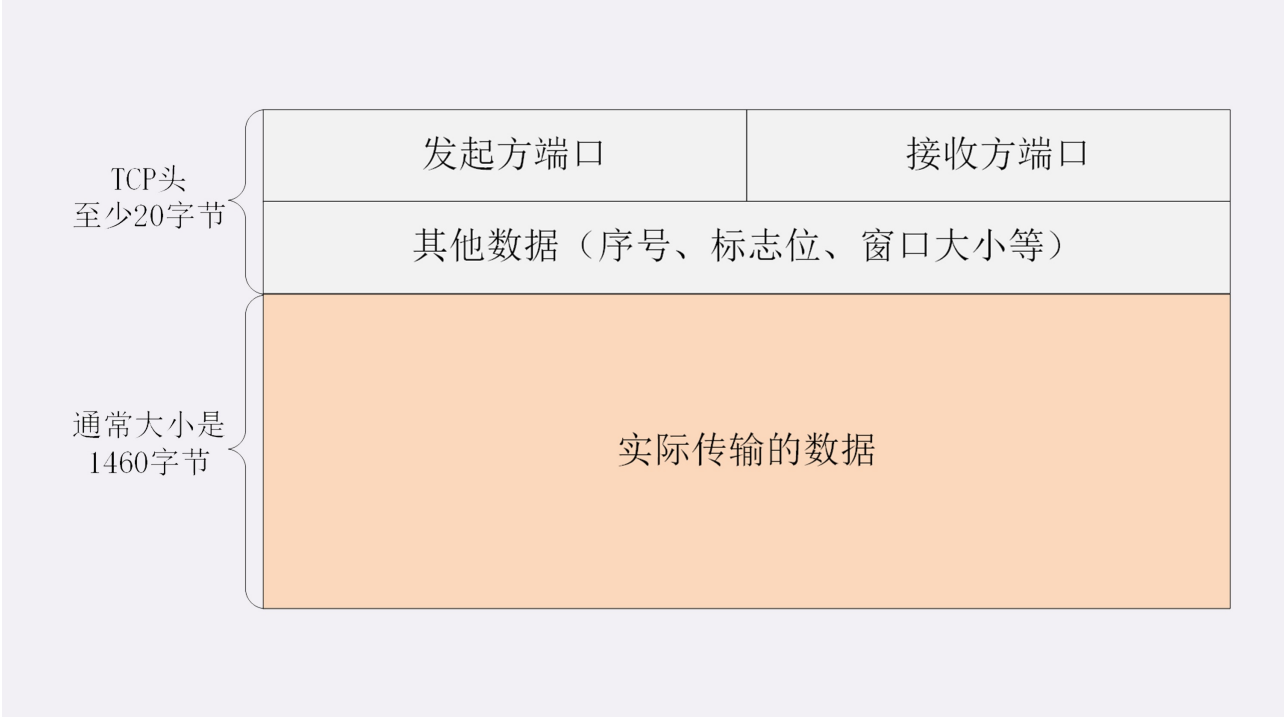
答案就是它传输的报文内容。

HTTP协议在规范文档里详细定义了报文的格式，规定了组成部分，解析规则，还有处理策略，所以可以在TCP/IP层之上实现更灵活丰富的功能，例如连接控制，缓存管理、数据编码、内容协商等等。

报文结构

你也许对TCP/UDP的报文格式有所了解，拿TCP报文来举例，它在实际要传输的数据之前附加了一个20字节的头部数据，存储TCP协议必须的额外信息，例如发送方的端口号、接收方的端口号、包序号、标志位等等。

有了这个附加的TCP头，数据包才能够正确传输，到了目的地后把头部去掉，就可以拿到真正的数据。



HTTP协议也是与TCP/UDP类似，同样也需要在实际传输的数据前附加一些头数据，不过与TCP/UDP不同的是，它是一个“**纯文本**”的协议，所以头数据都是ASCII码的文本，可以很容易地用肉眼阅读，不用借助程序解析也能够看懂。

HTTP协议的请求报文和响应报文的结构基本相同，由三大部分组成：

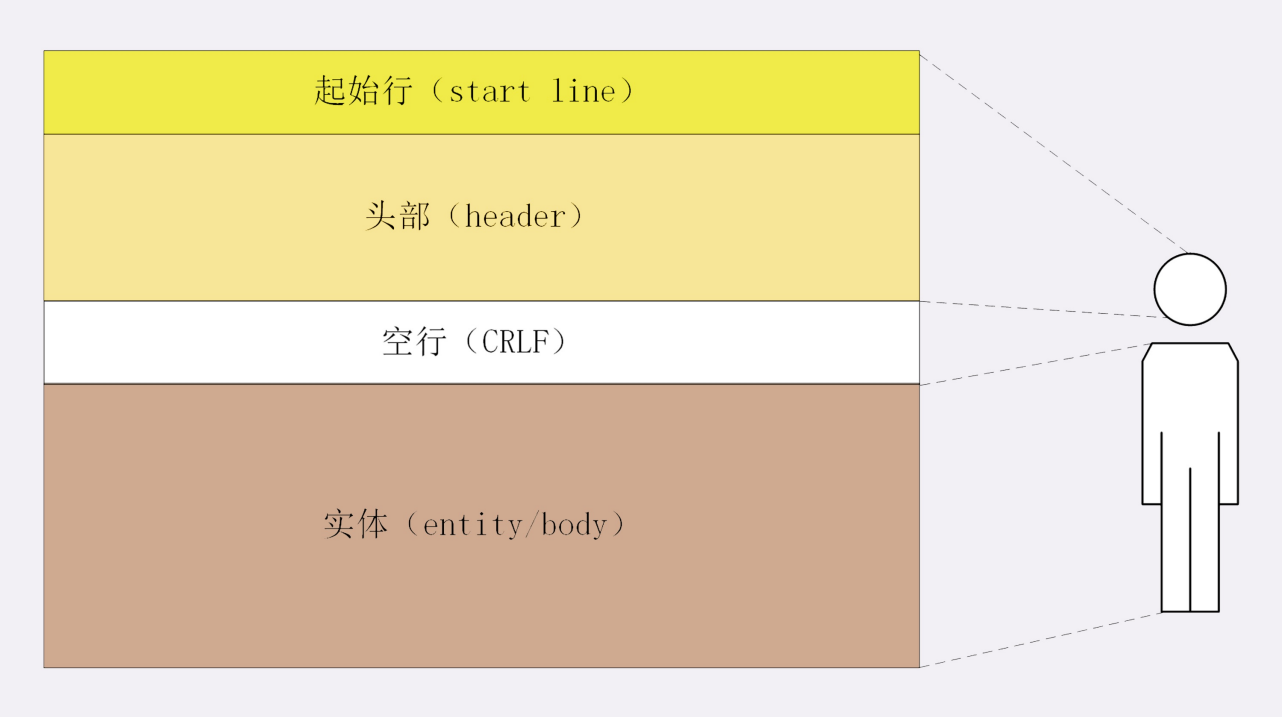
1. 起始行（start line）：描述请求或响应的基本信息；

2. 头部字段集合（header）：使用key-value形式更详细地说明报文；
3. 消息正文（entity）：实际传输的数据，它不一定是纯文本，可以是图片、视频等二进制数据。

这其中前两部分起始行和头部字段经常又合称为“**请求头**”或“**响应头**”，消息正文又称为“**实体**”，但与“**header**”对应，很多时候就直接称为“**body**”。

HTTP协议规定报文必须有header，但可以没有body，而且在header之后必须要有一个“空行”，也就是“CRLF”，十六机制的“0D0A”。

所以，一个完整的HTTP报文就像是下图的这个样子，注意在header和body之间有一个“空行”。



说到这里，我不由得想起了一部老动画片《大头儿子和小头爸爸》，你看，HTTP的报文结构像不像里面的“大头儿子”？

报文里的header就是“大头儿子”的“大头”，空行就是他的“脖子”，而后面的body部分就是他的身体了。

看一下我们之前用Wireshark抓的包吧。

```
GET / HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image
/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
```

在这个浏览器发出的请求报文里，第一行“GET / HTTP/1.1”就是请求行，而后面的“Host”“Connection”等等都属于header，报文的最后是一个空白行结束，没有body。

在很多时候，特别是浏览器发送GET请求的时候都是这样，HTTP报文经常是只有header而没body，相当于只发了一个超级“大头”过来，你可以想象的出来：每时每刻网络上都会有数不清的“大头儿子”在跑来跑去。

不过这个“大头”也不能太大，虽然HTTP协议对header的大小没有做限制，但各个Web服务器都不允许过大的请求头，因为头部太大可能会占用大量的服务器资源，影响运行效率。

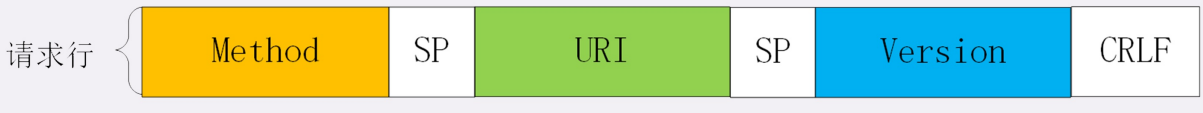
请求行

了解了HTTP报文的基本结构后，我们来看看请求报文里的起始行也就是**请求行**（request line），它简要地描述了**客户端想要如何操作服务器端的资源**。

请求行由三部分构成：

1. 请求方法：是一个动词，如GET/POST，表示对资源的操作；
2. 请求目标：通常是一个URI，标记了请求方法要操作的资源；
3. 版本号：表示报文使用的HTTP协议版本。

这三个部分通常使用空格（space）来分隔，最后要用CRLF换行表示结束。



还是用Wireshark抓包的数据来举例：

```
GET / HTTP/1.1
```

在这个请求行里，“GET”是请求方法，“/”是请求目标，“HTTP/1.1”是版本号，把这三部分连起来，意思就是“服务器你好，我想获取网站根目录下的默认文件，我用的协议版本号是1.1，请不要用1.0或者2.0回复我。”

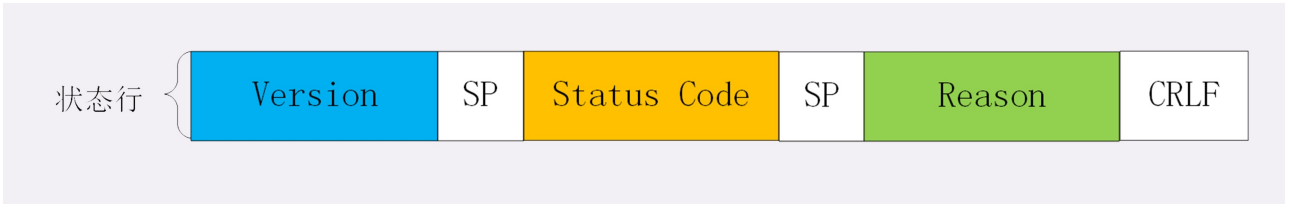
别看请求行就一行，貌似很简单，其实这里面的“讲究”是非常多的，尤其是前面的请求方法和请求目标，组合起来变化多端，后面我还会详细介绍。

状态行

看完了请求行，我们再看响应报文里的起始行，在这里它不叫“响应行”，而是叫“**状态行**”（status line），意思是**服务器响应的状态**。

比起请求行来说，状态行要简单一些，同样也是由三部分构成：

1. 版本号：表示报文使用的HTTP协议版本；
2. 状态码：一个三位数，用代码的形式表示处理的结果，比如200是成功，500是服务器错误；
3. 原因：作为数字状态码补充，是更详细的解释文字，帮助人理解原因。



看一下上一讲里Wireshark抓包里的响应报文，状态行是：

```
HTTP/1.1 200 OK
```

意思就是：“浏览器你好，我已经处理完了你的请求，这个报文使用的协议版本号是1.1，状态码是200，一切OK。”

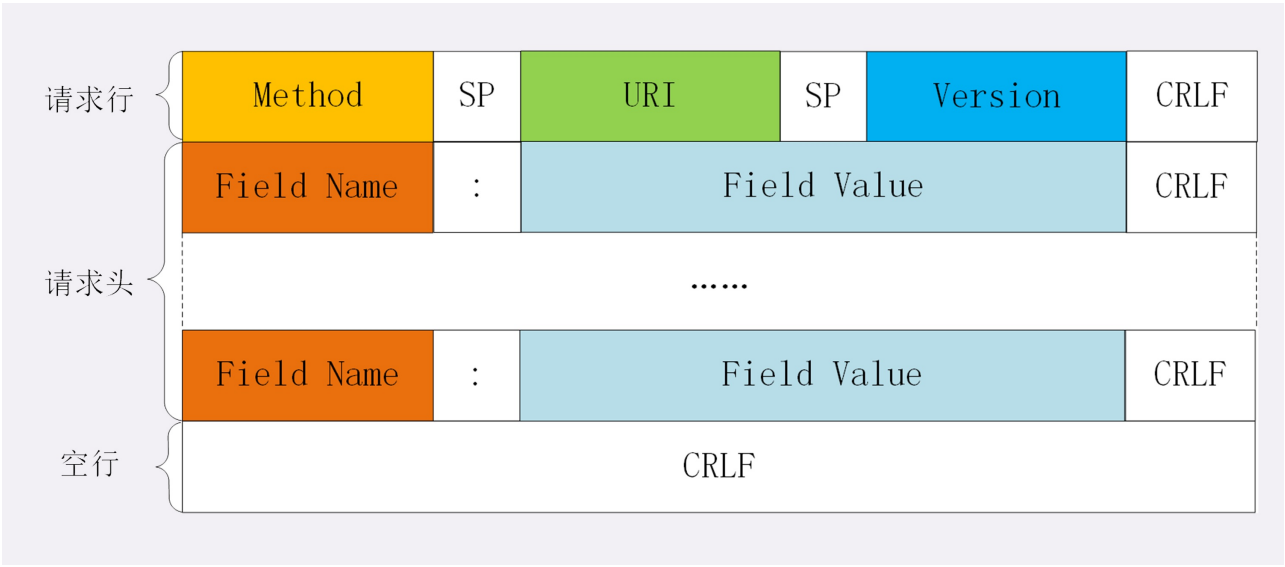
而另一个“GET /favicon.ico HTTP/1.1”的响应报文状态行是：

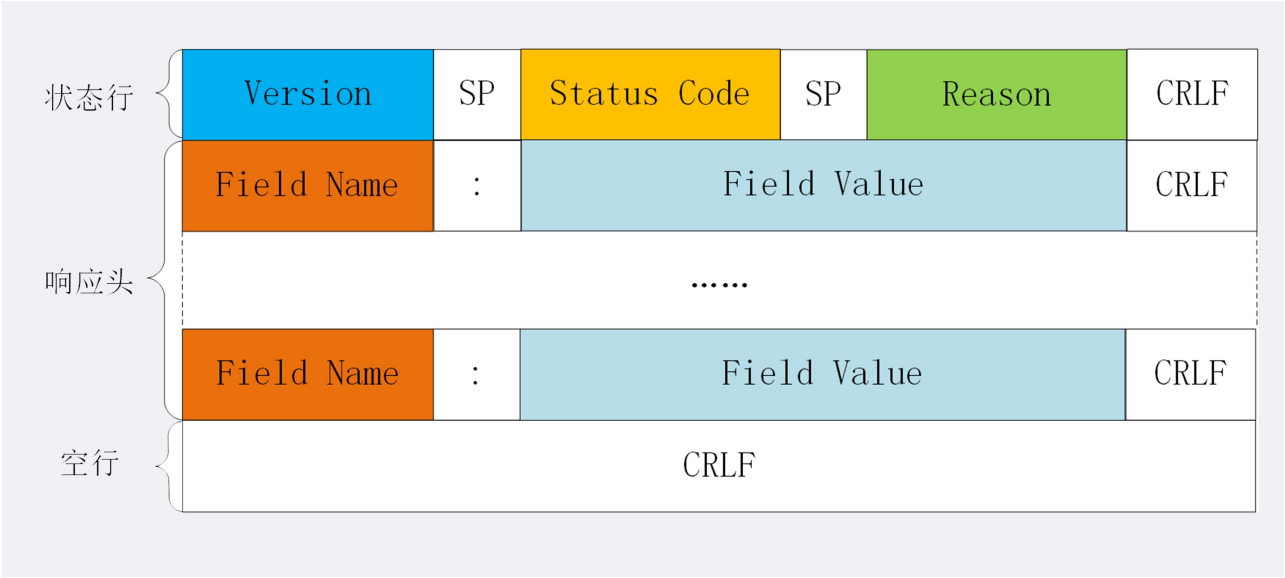
```
HTTP/1.1 404 Not Found
```

翻译成人话就是：“抱歉啊浏览器，刚才你的请求收到了，但我没找到你要的资源，错误代码是404，接下来的事情你就看着办吧。”

头部字段

请求行或状态行再加上头部字段集合就构成了HTTP报文里完整的请求头或响应头，我画了两个示意图，你可以看一下。





请求头和响应头的结构是基本一样的，唯一的区别是起始行，所以我把请求头和响应头里的字段放在一起介绍。

头部字段是key-value的形式，key和value之间用“:”分隔，最后用CRLF换行表示字段结束。比如在“Host: 127.0.0.1”这一行里key就是“Host”，value就是“127.0.0.1”。

HTTP头字段非常灵活，不仅可以使用标准里的Host、Connection等已有头，也可以任意添加自定义头，这就给HTTP协议带来了无限的扩展可能。

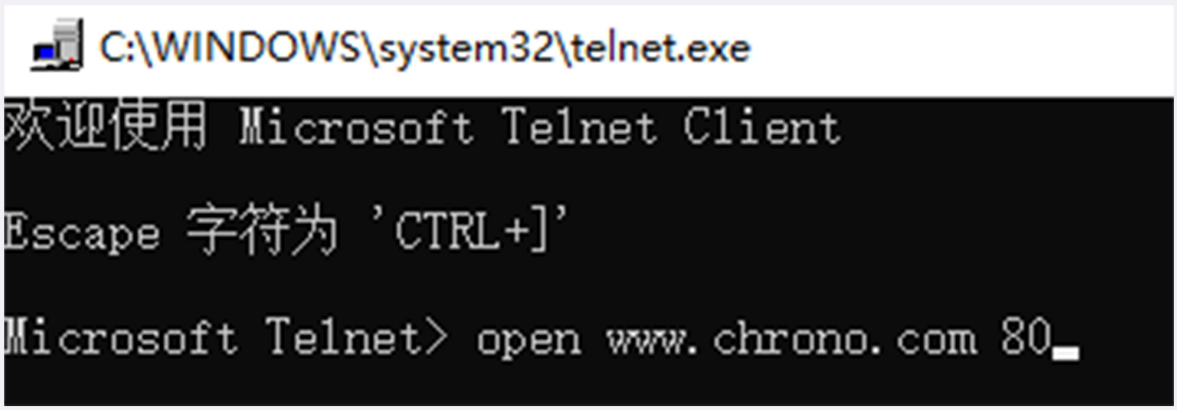
不过使用头字段需要注意下面几点：

1. 字段名不区分大小写，例如“Host”也可以写成“host”，但首字母大写的可读性更好；
2. 字段名里不允许出现空格，可以使用连字符“-”，但不能使用下划线“_”。例如，“test-name”是合法的字段名，而“test name”“test_name”是不正确的字段名；
3. 字段名后面必须紧接着“:”，不能有空格，而“:”后的字段值前可以有多个空格；
4. 字段的顺序是没有意义的，可以任意排列不影响语义；
5. 字段原则上不能重复，除非这个字段本身的语义允许，例如Set-Cookie。

我在实验环境里用Lua编写了一个小服务程序，URI是“/09-1”，效果是输出所有的请求头。

你可以在实验环境里用Telnet连接OpenResty服务器试一下，手动发送HTTP请求头，试验各种正确和错误的情况。

先启动OpenResty服务器，然后用组合键“Win+R”运行telnet，输入命令“open www.chrono.com 80”，就连上了Web服务器。



连接上之后按组合键“CTRL+]”，然后按回车键，就进入了编辑模式。在这个界面里，你可以直接用鼠标右键粘贴文本，敲两下回车后就会发送数据，也就是模拟了一次HTTP请求。

下面是两个最简单的HTTP请求，第一个在“:”后有多个空格，第二个在“:”前有空格。

```
GET /09-1 HTTP/1.1
Host:   www.chrono.com


GET /09-1 HTTP/1.1
Host : www.chrono.com
```

第一个可以正确获取服务器的响应报文，而第二个得到的会是一个“400 Bad Request”，表示请求报文格式有误，服务器无法正确处理：

```
HTTP/1.1 400 Bad Request
Server: openresty/1.15.8.1
Connection: close
```

常用头字段

HTTP协议规定了非常多的头部字段，实现各种各样的功能，但基本上可以分为四大类：

1. 通用字段：在请求头和响应头里都可以出现；

2. 请求字段：仅能出现在请求头里，进一步说明请求信息或者额外的附加条件；
3. 响应字段：仅能出现在响应头里，补充说明响应报文的信息；
4. 实体字段：它实际上属于通用字段，但专门描述body的额外信息。

对HTTP报文的解析和处理实际上主要就是对头字段的处理，理解了头字段也就理解了HTTP报文。

后续的课程中我将会以应用领域为切入点介绍连接管理、缓存控制等头字段，今天先讲几个最基本的头，看完了它们你就应该能够读懂大多数HTTP报文了。

首先要说的是**Host**字段，它属于请求字段，只能出现在请求头里，它同时也是唯一一个HTTP/1.1规范里要求**必须出现**的字段，也就是说，如果请求头里没有Host，那这就是一个错误的报文。

Host字段告诉服务器这个请求应该由哪个主机来处理，当一台计算机上托管了多个虚拟主机的时候，服务器端就需要用Host字段来选择，有点像是一个简单的“路由重定向”。

例如我们的试验环境，在127.0.0.1上有三个虚拟主机：“www.chrono.com”“www.metroid.net”和“origin.io”。那么当使用域名的方式访问时，就必须要用Host字段来区分这三个IP相同但域名不同的网站，否则服务器就会找不到合适的虚拟主机，无法处理。

User-Agent是请求字段，只出现在请求头里。它使用一个字符串来描述发起HTTP请求的客户端，服务器可以依据它来返回最合适此浏览器显示的页面。

但由于历史的原因，User-Agent非常混乱，每个浏览器都自称是“Mozilla”“Chrome”“Safari”，企图使用这个字段来互相“伪装”，导致User-Agent变得越来越长，最终变得毫无意义。

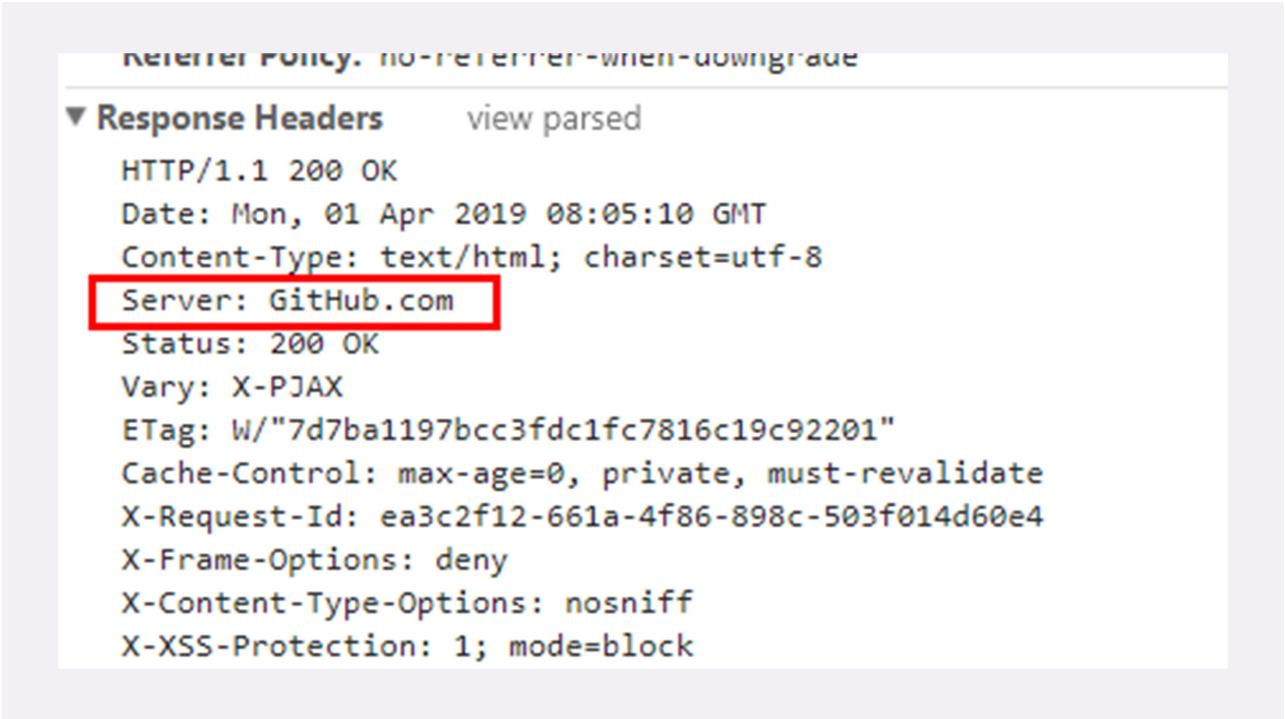
不过有的比较“诚实”的爬虫会在User-Agent里用“spider”标明自己是爬虫，所以可以利用这个字段实现简单的反爬虫策略。

Date字段是一个通用字段，但通常出现在响应头里，表示HTTP报文创建的时间，客户端可以使用这个时间再搭配其他字段决定缓存策略。

Server字段是响应字段，只能出现在响应头里。它告诉客户端当前正在提供Web服务的软件名称和版本号，例如在我们的实验环境里它就是“Server: openresty/1.15.8.1”，即使用的是OpenResty 1.15.8.1。

Server字段也不是必须要出现的，因为这会把服务器的一部分信息暴露给外界，如果这个版本恰好存在bug，那么黑客就有可能利用bug攻陷服务器。所以，有的网站响应头里要么没有这个字段，要么就给出一个完全无关的描述信息。

比如GitHub，它的Server字段里就看不出是使用了Apache还是Nginx，只是显示为“GitHub.com”。



实体字段里要说的一个是**Content-Length**，它表示报文里body的长度，也就是请求头或响应头空行后面数据的长度。服务器看到这个字段，就知道了后续有多少数据，可以直接接收。如果没有这个字段，那么body就是不定长的，需要使用chunked方式分段传输。

小结

今天我们学习了HTTP的报文结构，下面做一个简单小结。

1. HTTP报文结构就像是“大头儿子”，由“起始行+头部+空行+实体”组成，简单地说就是“header+body”；
2. HTTP报文可以没有body，但必须要有header，而且header后也必须要有空行，形象地说就是“大头”必须要带着“脖子”；
3. 请求头由“请求行+头部字段”构成，响应头由“状态行+头部字段”构成；
4. 请求行有三部分：请求方法，请求目标和版本号；
5. 状态行也有三部分：版本号，状态码和原因字符串；
6. 头部字段是key-value的形式，用“:”分隔，不区分大小写，顺序任意，除了规定的标准头，也可以任意添加自定义字段，实现功能扩展；
7. HTTP/1.1里唯一要求必须提供的头字段是Host，它必须出现在请求头里，标记虚拟主机名。

课下作业

1. 如果拼HTTP报文的时候，在头字段后多加了一个CRLF，导致出现了一个空行，会发生什么？
2. 讲头字段时说“:”后的空格可以有多个，那为什么绝大多数情况下都只使用一个空格呢？

欢迎你把自己的答案写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

课外小贴士

01 在 Nginx 里，默认的请求头大小不能超过 8K，但可以用指令“large_client_header_buffers”修改。

02 在 HTTP 报文里用来分隔请求方法、URI 等部分的不一定必须是空格，制表符（tab）也是允许的。

03 早期曾经允许在头部用前导空格实现字段跨行，但现在这种方式已经被 RFC7230 废弃，字段只能放在一行里。

04 默认情况下 Nginx 是不允许头字段里使用“_”的，配置指令“underscores_in_headers on”可以解除限制，但不推荐。

05 与 Server 类似的一个响应头字段是“X-Powered-By”，它是非标准字段，表示服务器使用的编程语言，例如“X-Powered-By: PHP/7.0.22”

精选留言：

- 一步 2019-06-17 08:48:31
  1:如果拼 HTTP 报文的时候，在头字段后多加了一个 CRLF，导致出现了一个空行，会发生什么？
  在header 下面第一个空行以后都会被当作body 体

  2:讲头字段时说“:”后的空格可以有多个，那为什么绝大多数情况下都只使用一个空格呢？
  头部多一个空格就会多一个传输的字节，去掉无用的信息，保证传输的头部字节数尽量小 [8赞]

  作者回复2019-06-17 09:51:16
  回答的很好。

- 壹笙☞漂泊 2019-06-17 16:08:42
  答题：
  1、头字段后多了一个CRLF，会被当做body处理
  2、节省资源
  总结：
  HTTP协议的请求报文和相应报文的结构基本相同：
  1、起始行（start line）：描述请求或响应的基本信息
  2、头部字段集合（header）：使用key-value形式更详细的说明报文
  3、消息正文（entity）：实际传输的数据，它不一定是纯文本，可以是图片、视频等二进制数据

  HTTP协议必须有header，可以没有body。而且header之后必须要有一个空行，也就是 “CRLF”，十六进制的“0D0A”

  请求行（请求报文里的起始行）：
  描述了客户端想要如何操作服务器端的资源
  起始行由三部分构成：
  1、请求方法：标识对资源的操作：GET/POST/PUT
  2、请求目标：通常是一个URI，标记了请求方法要操作的资源
  3、版本号：标识报文使用的HTTP协议版本
  以上三部分，通常使用空格分隔，最后用CRLF换行

状态行：（响应报文里的起始行）：
服务器响应的状态
状态行也是由三部分构成：
1、版本号：标识报文使用的HTTP协议版本
2、状态码：三位数，用代码形式标识处理的结果，比如200是成功，500是服务器错误
3、原因：作为数字状态码补充，是更详细的解释文字，帮助人理解原因
以上三部分，通常也使用空格分隔，最后用CRLF换行

头部字段：
请求行或状态行再加上头部字段集合就构成了HTTP报文里完整的请求头或响应头。

头部字段是key-value的形式，用“:”分隔，最后用CRLF换行标识字段结束

头字段，不仅可以使用标准的Host等已有开头，也可以任意添加自定义头

注意：
1.字段名不区分大小写，例如“Host"也可以写成“host”,但首字母大写的可读性更好;
2.字段名里不允许出现空格，可以使用连字符“一”，但不能使用下划线“”。例 如，“test-name”是合法的字段名，而“test name""test_ name' 是不正确的字段名;
3.字段名后面必须紧接着“:”，不能有空格，而“:” 后的字段值前可以有多个空格;
4.字段的顺序是没有意义的，可以任意排列不影响语义;
5.字段原则上不能重复，除非这个字段本身的语义允许，例如Set-Cookie。

常用头字段

基本分为四类：
1.通用字段:在请求头和响应头里都可以出现;
2.请求字段:仅能出现在请求头里，进一步说明请求信息或者额外的附加条件;
3.响应字段:仅能出现在响应头里，补充说明响应报文的信息;
4.实体字段:它实际上属于通用字段，但专门描述body的额外信息。

Host：请求字段，只能出现在请求头。是必须出现的字段
User-Agent：是请求字段，只能出现在请求头里。
Date：是通用字段，通常出现在响应头，标识HTTP报文创建的时间，客户端可以使用这个时间再搭配其他字段决定缓存策略
Server字段是响应字段，只能出现在响应头里。告诉客户端当前正在提供Web服务的软件名称和版本号。
Content-Length：标识报文里body的长度。 [1赞]

- 高翔Gilbert 2019-06-17 08:02:52
  为什么不找人读呢？听起来好吃力 [1赞]

  作者回复2019-06-17 09:12:47
  sorry了，本人非播音科班，听着难受就看文字吧。

- 苏超 2019-06-18 00:28:40
  2. 讲头字段时说“:”后的空格可以有多个，那为什么绝大多数情况下都只使用一个空格呢？
  请问老师，空格可以一个都不加吧，telnet测试也可以正确返回，为什么还要使用一个空格

- -W.LI- 2019-06-17 23:03:14

支持老师原声。赞一个

- Alex 2019-06-17 22:54:09
  老师您好，我初次接触openresty，我在open www.chrono.com这一步操作的时候一直连不上，命令行一直显示"正在连接www.chrono.com",然后过一会就提示"遗失对主机的连接"

- Geek_d4dee7 2019-06-17 20:37:21
  ETag 以下的相应头都不知道意思了 下节课有解释么

- qzmone 2019-06-17 20:04:48
  我也不是很理解这个host字段，比如一个网站的域名解析后的IP是负载均衡的IP，负载均衡后面对应的是web主机集群，那么这个host是什么，浏览器怎么知道虚拟主机的真实IP呢

- 衬衫的价格是19美元 2019-06-17 19:18:15
  所以http请求头是什么数据类型呢？是一个大的数组吗？每个字段都是数组的元素，如果出现空格来就认为头结束了

- bywuu 2019-06-17 12:20:11
  如果http请求报文可以比喻成大头儿子，而对于多数带实体内容的响应报文则是小头爸爸，那么。。。中间人攻击是不是隔壁老王呢？？？

  作者回复2019-06-17 13:14:52
  这我倒没想到，同学们真是发散思维啊。

- bywuu 2019-06-17 12:18:28
  问题一：没什么实际影响，就是报文多了一个空行呗
  问题二：为了减少头部的大小，因为不允许太大的头部文件出现，节省空间，爱护环境，从我做起

  作者回复2019-06-17 13:16:08
  第二个回答很对。

  第一个就不对了，因为多了一个空行，就变更了报文的结构，两个空行，那么下面的就都认为是body了。

  可参考其他同学的回答。

- 鸟人 2019-06-17 11:15:37
  1:如果拼 HTTP 报文的时候，在头字段后多加了一个 CRLF，导致出现了一个空行，会发生什么？
  有可能导致crlf注入

  作者回复2019-06-17 12:52:10
  √

- 古夜 2019-06-17 09:53:35
  1.报错或者自动去一行
  2.空格也是有占数据长度的

  作者回复2019-06-17 10:17:07
  第一个可参考其他同学的回答，需要再仔细理解一下http的报文格式。

- 佳佳大魔王 2019-06-17 09:49:23
  老师，telnet一直连接不上，总是显示遗失对主机的连接（之前我的端口冲突，我把80改成了90，把443该为了444，在浏览器时加冒号和90可以连接，这里使用 open www. chrono. com 90出现的问题）

  作者回复2019-06-17 10:21:06
  连接后需要ctrl+]，然后回车，进入编辑界面，手写HTTP请求，不然长时间无数据就会断开连接。

- 石维康 2019-06-17 09:31:50
  试着回答一下本节留的思考题
  1.如果拼 HTTP 报文的时候，在头字段后多加了一个 CRLF,导致出现了一个空行，会发生什么？
  一个换行就是CRLF,那么会导致收到的body少两个字节吧

  2.讲头字段时说“:”后的空格可以有多个，那为什么绝大多数情况下都只使用一个空格呢？
  方便人类阅读，并且为了节省网络传输带宽

  作者回复2019-06-17 09:53:27
  第一个，因为多了个换行，那么就相当于报文头结束了，那么后面的就都是body了。

- 无名 2019-06-17 09:06:20
  rfc有规定GET 请求参数的具体要求吗？开发时碰到个问题，就是GET请求的参数为数组。
  客户端传递的格式是：?arr%5B%5D=value0&arr%5B%5D=value1&arr%5B%5D=value2。
  服务端的web服务器为tomcat，报不符合RFC规范。
  后来看了源码发现tomcat支持的格式为：
  1、?arr=value0&arr=value1&arr=value2
  2、?arr%5B0%5D=value0&arr%5B1%5D=value1&arr%5B2%5D=value2

  但我查RFC文档时确没有找到关于GET参数传递数组的规定。

  作者回复2019-06-17 09:55:35
  query参数不支持数组，都是key=value的形式。

  Tomcat没用过，所以不好回答。但一个key对应多个value的形式是可以的，服务器可以把它理解成数组的形式。

- 一步 2019-06-17 08:28:21
  文中说 http 的头部不能使用下划线，感觉是有问题的，就拿 nginx 来说吧，虽然nginx 默认是忽略下划线的头部的，但是可以设置 underscores_in_headers on; 来获取下划线的头部

  对于常用的 web application 服务器，下划线的头部好像是可以直接获取到的，不用配置什么

  老师你说的不能使用下划线是 RFC规范吗？

  作者回复2019-06-17 09:51:57
  是的，RFC有规定，但现实中也有部分不遵守。

- lucy 2019-06-17 07:55:37
  在抓网页时，经常会遇到favicon.ico,请问这是什么东西

  作者回复2019-06-17 09:13:16
  网站的小图标，显示在标签页上用的，非http标准。

- lucy 2019-06-17 07:51:19
  我觉得http请求报文可以比喻成大头儿子，而对于多数带实体内容的响应报文则是小头爸爸

  作者回复2019-06-17 09:14:09
  这个说法不错。

- WL 2019-06-17 07:36:16
  请问一下老师HOST头部的value是虚拟主机, 虚拟主机的意思具体是什么? 在请求行中有URL不是就可以看到主机地址或者域名了吗, 为什么HTTP1.1规定HOST头部是必须的呢?

  作者回复2019-06-17 09:16:21
  host字段是域名，虚拟主机对应的是“真实主机”。

  很久以前都是一个ip对应一个主机，就是“真实主机”，但这样太浪费资源，所以就出现了一个ip上有多台主机，这个就是“虚拟主机”。

  你看http报文，uri里是没有域名的，只有path，所以http/1.1要求必须有host字段，这样服务器就可以在一个ip上区分多个虚拟主机。