03 HTTP 请求与响应

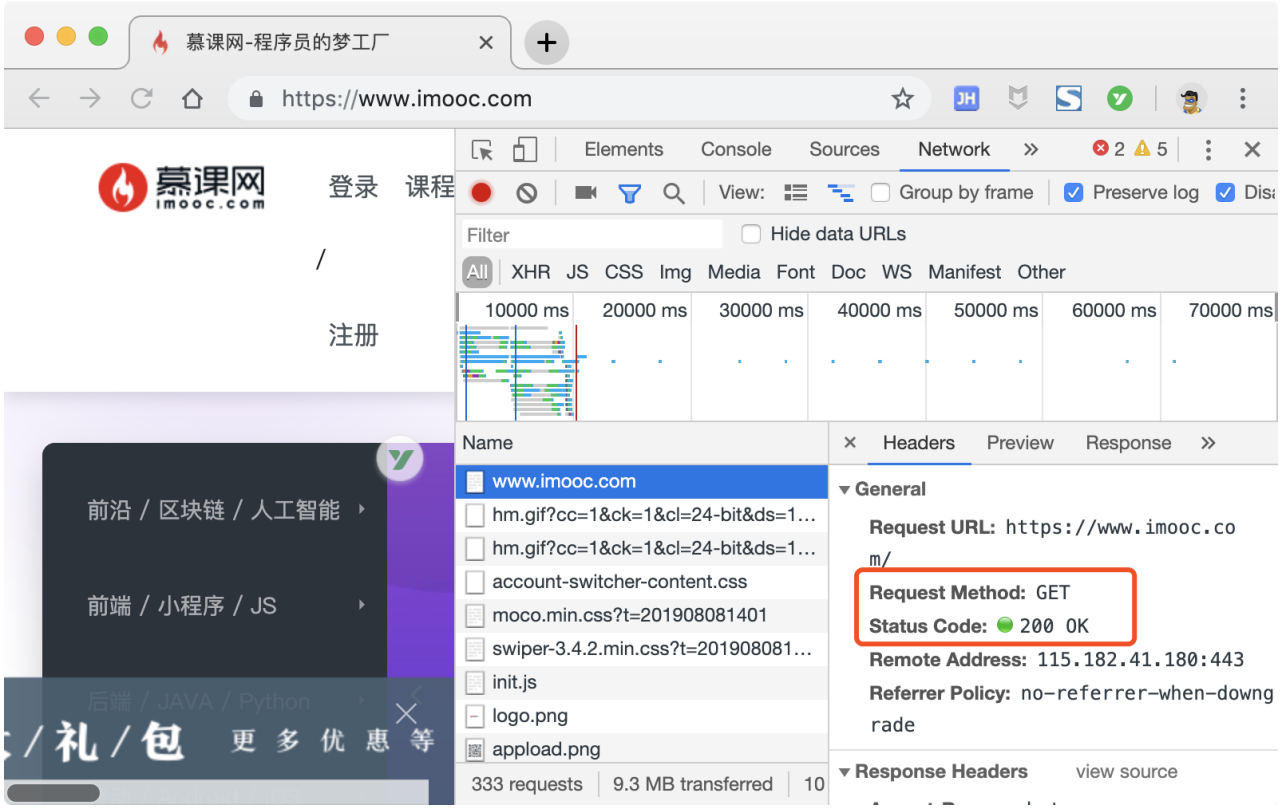
更新时间：2020-06-01 11:00:10

> 生活永远不像我们想像的那样好，但也不会像我们想像的那样糟。——莫泊桑

GET 方法和 **POST** 方法

GET：获取资源

一般来说，GET 请求只用于获取数据。比如我们获取指定页面的信息，并返回响应。



上图是我们访问慕课网的首页，我们可以发现使用的就是 GET 方法。

POST：请求资源：

POST 请求主要是向指定资源提交数据，服务器接收到这些数据进行处理。比如我们在注册页面中填写了一些个人信息，当我们提交这些信息的时候会向服务器发送一个 POST 请求，将信息放在请求体中，服务器收到之后进行相应的处理。

> 虽然 HTTP 协议中的请求方法很多，但是在大部分的时候 GET 方法和 POST 方法已经满足我们的需求了，我在工作中最常用的也是这两种方法，别的方法基本没用过……

GET 和 POST 的区别

说了这么多，那么 GET 方法和 POST 方法到底有什么区别呢？这是一个绕不开的话题，而且这个问题在面试的时候也经常会被问到。也许我们被问到这个问题的时候都能扯上一两句，比如：

- GET 请求的参数是通过 URL 传递的，POST 请求参数是通过 Request Body 传递的；
- GET 允许传递的参数没有 POST 的长度长。

还有很多类似答案，但是事实上是这样的吗？我不得不告诉你一个残酷的事实： GET 和 POST 其实没有任何区别~

HTTP 是一个应用层的协议，它底层使用的是 TCP/IP，所以 GET 和 POST 二者的底层是相同的，二者能做的事情是一模一样的。我们一一**驳斥**上面的几个理由。

- 如果你愿意，GET 请求也可以有 Request body，你可以请求参数放到 Request body 中是完全可以的。同样地，你也可以在 POST 请求的 URL 中加入请求参数。本人刚加入微博的时候，发现很多 POST 请求的 URL 中都包含了很多参数，对此深表疑问，后来查了很多资料才发现，原来是自己一直理解错误了，尴尬~
- GET 和 POST 参数长度问题。这些长度并不是 HTTP 规定的，而是**浏览器**和**服务器**自己的规定，和 HTTP 协议没有一毛钱的关系。

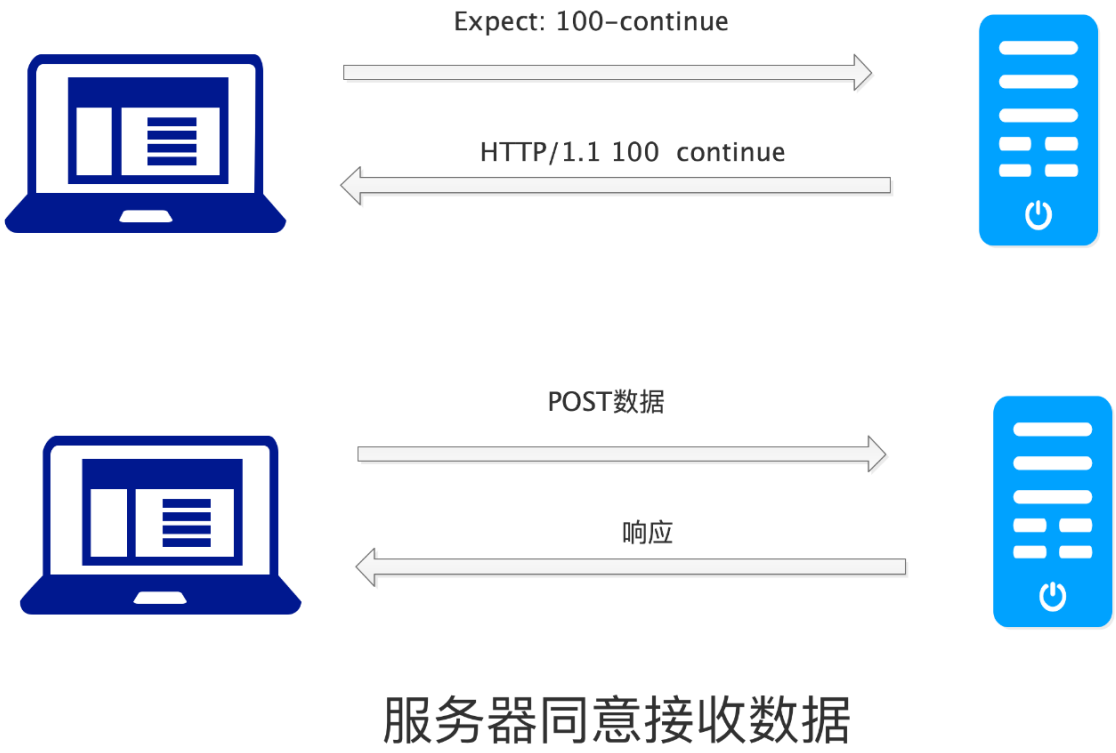
> 浏览器和服务器为了节省内存，所以都会限制我们参数的大小。在 Nginx 中可以通过 large_client_header_buffers 来限制请求头的长度~

那说来说去，二者是一样的，那为啥要搞两种方法呢？

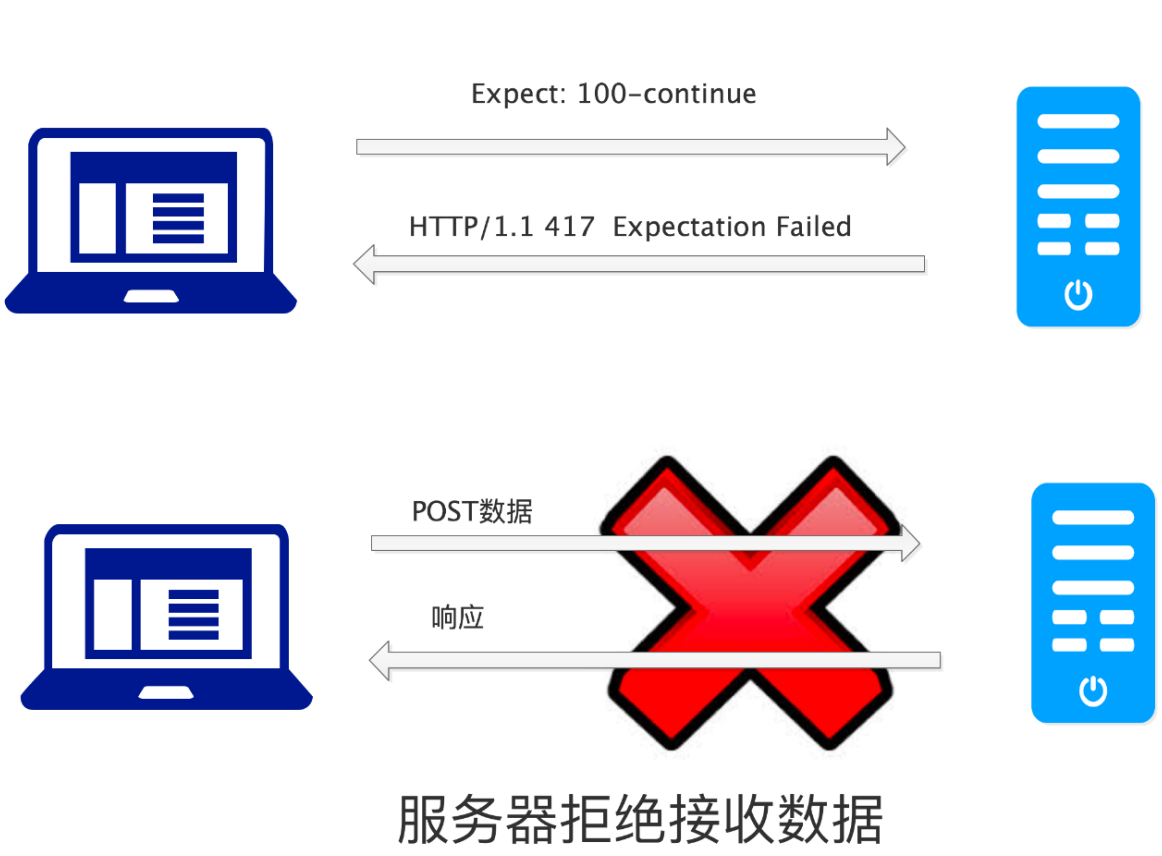
来，我们划重点了：

有一些客户端，比如 CURL 命令，当 POST 的数据大于 1024 字节的时候，会先分为两个步骤：

1. 向服务器发送一个包含 Expect: 100-continue 请求，询问服务器是否愿意接收数据。
2. 服务器返回 100 continue， curl 把真正的 POST 数据发送给服务器。



当然了，并不是所有的服务器都会返回 100-continue，有的会返回 417 Expectation Failed，这是时候就不能继续 POST 了。



> 到这里之后，大家是否明白了 GET 和 POST 的区别了呢?

请求URL

这部分就是我们请求的资源的地址，它配合请求头部的 `Host` 属性共同工作。

协议版本

这部分就是当前请求使用的 `HTTP` 协议的版本信息。上面我们提到过，不同的 `HTTP` 版本的功能是不相同的，所以我们要明确的说明当前请求使用的是哪个版本。

请求头

请求头是个什么东西呢？我们不妨以一个例子说明：

> 放学的时候，王老师对小明说，“小明，明天早上八点，你去学校西门口迎接一位新同学小李，他穿白色T恤，身高180……”

在这里面，**接小李**就是**HTTP的报文**，而**明天早上八点，学校西门口，白色T恤，身高180**这些附加信息就相当于 HTTP 的请求头，这些信息是为了完成工作所添加的额外信息。

HTTP 的请求和响应都包含报文头，报文头分为如下几种：

- 通用首部字段
- 请求首部字段
- 响应首部字段
- 实体首部字段

每一种类型的头部都有很多不同的选项，我们先介绍前两种报文头，后面在学习 `响应报文` 时介绍剩下的两种报文格式。

通用首部

通用首部既可以用在 `Request` 中，也可以用在 `Response` 中。常用的有以下几个：

首部字段	作用
Date	报文的日期相关
Cache-Control	控制缓存
Connection	管理连接

我们这里主要看一下 `Connection` 选项，这个选项是用于管理 `HTTP` 连接的，格式如下：

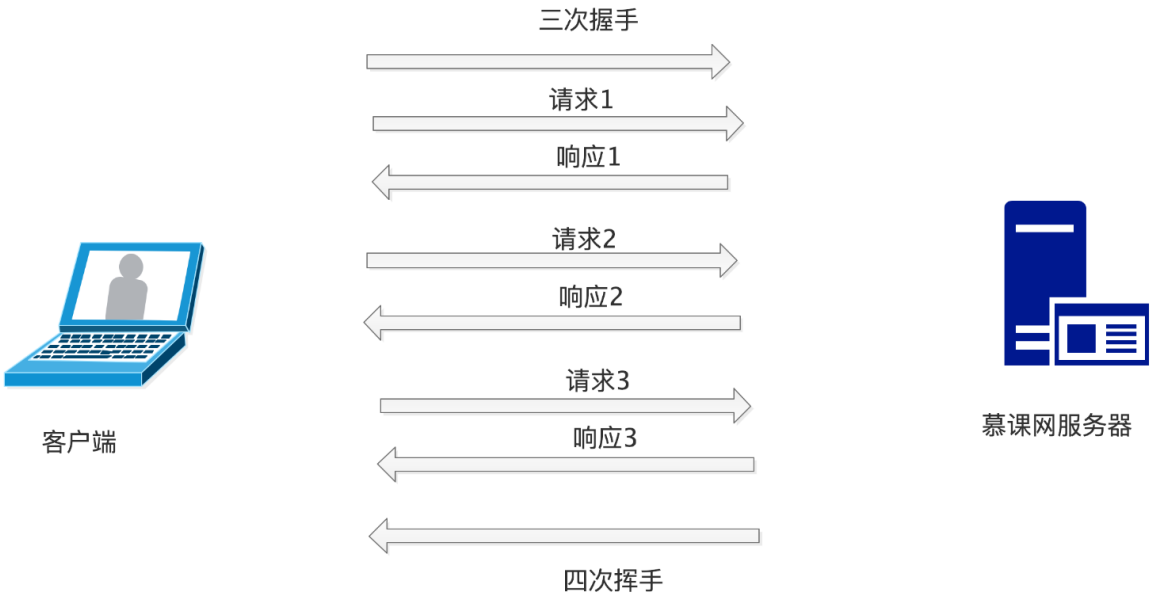
```
Connection: keep-alive
Connection: close
```

keep-alive

HTTP 是基于 Resquest-Response 形式的，客户端每次发送完一个请求之后，等待服务器响应，最后和服务器断开连接，本次请求结束。

由于 HTTP 是基于 TCP/IP 传输的，当 HTTP 发送请求的时候要和服务器经过三次握手建立连接，断开的时候要经过四次挥手进行断开。如果有大量的 HTTP 请求的话，每次都要进过 三次握手 和 四次挥手，就会非常的耗时。

为了解决这个问题，HTTP引入了 keep-alive 机制。所谓的keep-alive就是客户端和服务器三次握手之后，可以发送多个请求，最后再进行四次挥手。



这样我们就可以节省很多次握手和挥手的步骤，提升了服务器的性能。

HTTP 头部中的 connection 就是完成这个功能的，当我们设置 Connection: keep-alive 的时候，就会保持该链接，直到某个请求的 Connection: close 为止。这样可以在很大程度上提高 HTTP 的性能。

请求首部字段

请求首部字段是发送 HTTP 请求时使用的首部字段，用于补充请求的额外信息，便于服务器理解请求的内容。

请求首部字段有很多内容，我们学习几个常用的字段：

字段名	作用

字段名	作用
Host	请求资源所在的服务器
If-Match	标志位，用于判断资源是否符合要求
If-Modified-Since	当资源比该字段内容更新时，发送资源
If-Unmodified-Since	和 If-Modified-Since 作用相反
Referer	当前请求来源方的地址
User-Agent	客户端的类型，比如 Chrome 浏览器， IE 浏览器, CURL 程序等
Cookie	发送请求时给服务端的一些信息

Host字段

这个字段说明了当前请求的服务器的域名。比如我们访问慕课网首页的时候，我们可以看到浏览器发送的请求中，Host 字段就被设置为了 www.imooc.com 。

```
Host: www.imooc.com
```

If系列

我们这里列出来了三种 If 系列首部字段。HTTP 协议还有其它几种 If 字段，大体功能都类似。从名字中我们可以看出来，这些首部都带有**判断**性质， 如果满足了某种条件才会做什么。这三个选项都是为了让 HTTP 更高效，节省网络带宽。

当服务器发送一个响应的时候，可能会带有一个 ETag 标志，客户端在第二次获取该资源的时候，可以带上 If-Match 字段，它的值就是服务器返回的 ETag，当服务器发现对应的资源发生改变的时候，会将新资源返回，并生成新的 ETag 返回给客户端。如果资源未发生改变，那么服务端会返回 304 Not Modified，这样就节省了带宽。

If-Modified-Since 和 If-Unmodified-Since 也是类似的作用，只不过它们比较的是返回内容的时间戳。

Referer字段

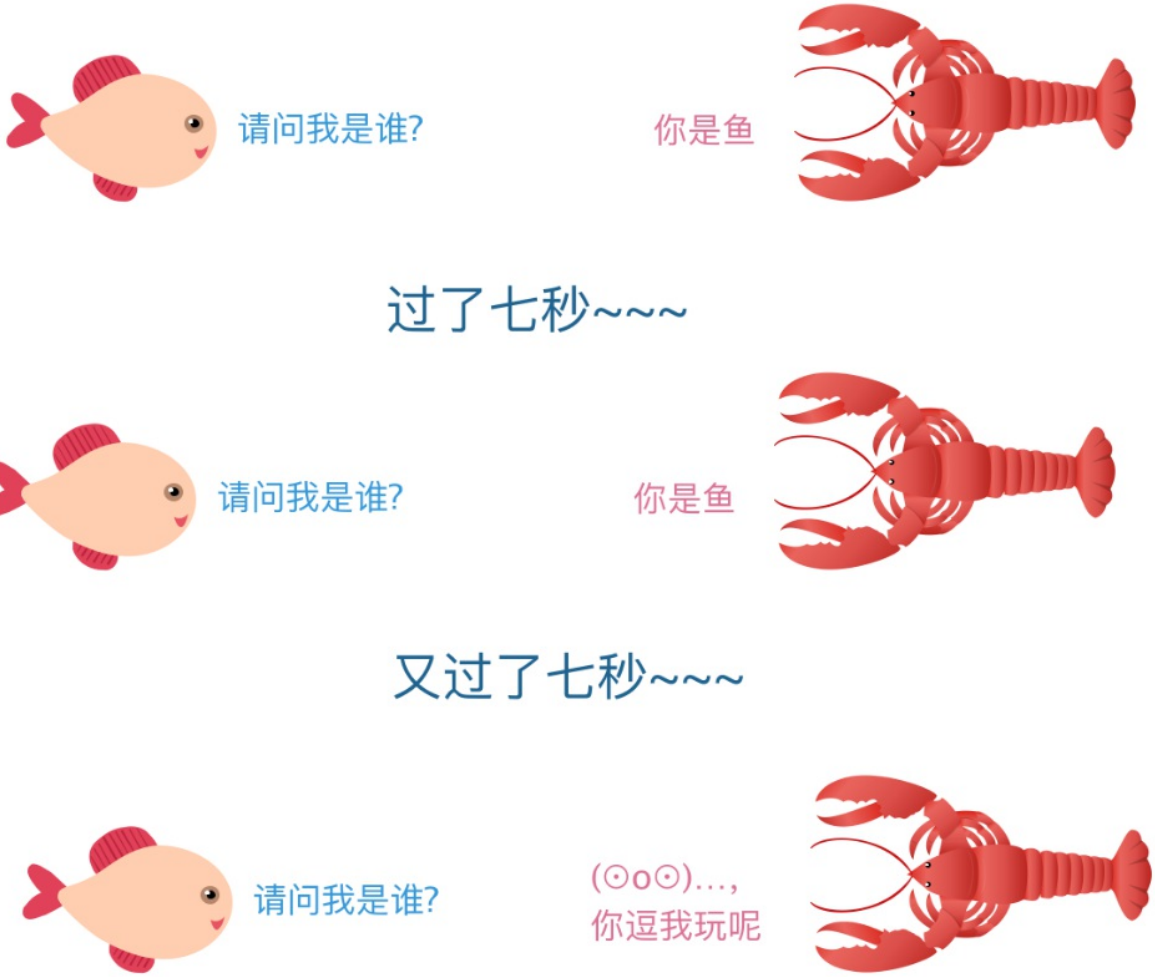
这个字段是为了说明当前是从哪里来的。比如我们从慕课网首页跳转到免费课程的时候，就可以发现请求的 Referer 如下：

```
Referer: https://www.imooc.com/
```

说明我们是从慕课网的首页跳转而来的，这个字段对于图片防盗链非常有效。

Cookie

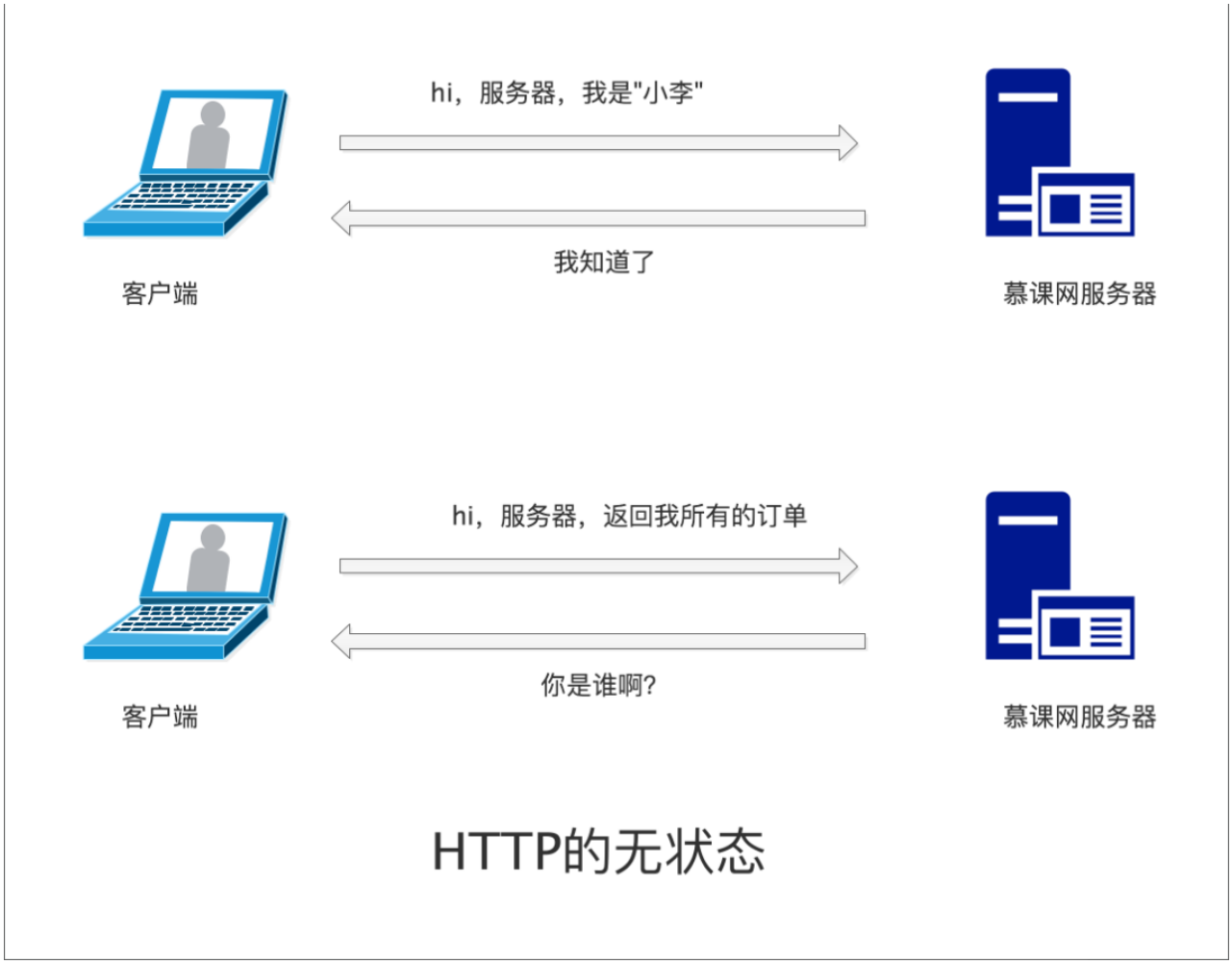
Cookie 相关的问题会经常在面试中被问到，那么什么是 Cookie 呢？在说明这个东西之前，我们先来了解一个概念：HTTP 协议是一个无状态的协议，啥叫无状态协议呢？来看下图：



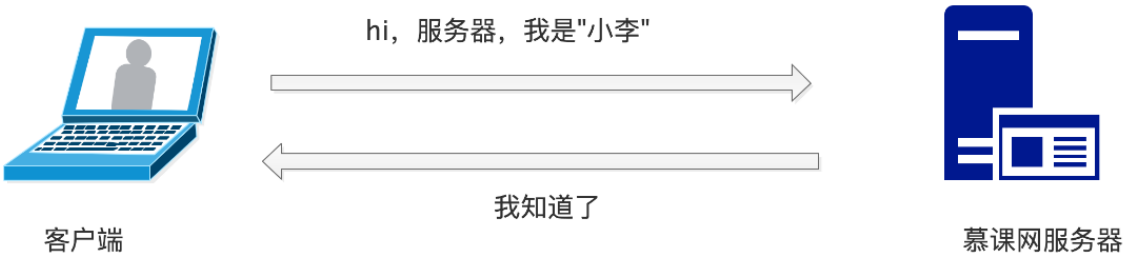
> 传说，鱼只有七秒的记忆，所以它可以每天非常欢乐的在水里游啊游，游啊游~

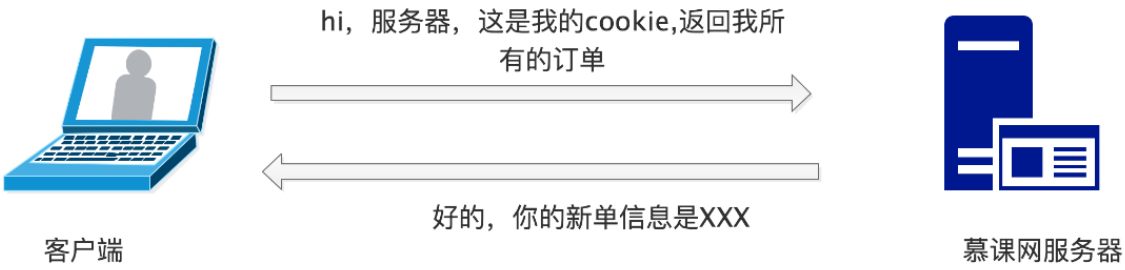
如果把鱼比作 HTTP 协议的话，那么它应该是世界上最**快乐**的协议了，因为 HTTP 连一点**记忆**都没有~~ 。HTTP 的无状态指的是它不会保存任何状态信息，每个请求都是独立的，和其它请求没有任何关系。

比如，我们在第一个页面登陆了慕课网，当我们打开第二个慕课网页面的时候，HTTP并不会记住我们已经登录了慕课网。



为了解决这个问题，HTTP 引入了 cookie 和 session 机制。每次发出 HTTP 请求的时候，都会把 cookie 带上，那么第二次请求通过状态信息就可以知道当前用户是谁了。





HTTP的无状态

但是由于 cookie 是保存在客户端的信息，所以很容易被篡改，因此 HTTP 引入了 session 机制。session 是保存在服务端的信息，起到的作用和 cookie 类似，都是用于保存一些状态信息。

到这里之后，我们再把 Cookie 的概念告诉你，是不是很容易理解了呢？

> An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with the next request to the same server. Typically, it's used to tell if two requests came from the same browser — keeping a user logged-in, for example. It remembers stateful information for the stateless HTTP protocol.
>
> —以上是来自 MDN 对于 Cookie 的解释

在这里我用我那撇脚的英语水平给大家翻译一下：

> Cookie 是服务端发送给客户端的一段数据。客户端可以保存该数据，并在后续请求中带上该数据。通常来说，Cookie 用于用户登录等操作。Cookie 使得无状态的 HTTP 变得有状态了。

> 大概就是这么个意思吧~~

请求体

这里面就是我们真正要做的事情了，这里面的内容就是每个请求自定义的~~。比如我们在注册的时候，通常在点击注册按钮时，会发送一个 POST 请求，请求体里面就包含了我们填写的姓名、密码、邮箱等个人信息。

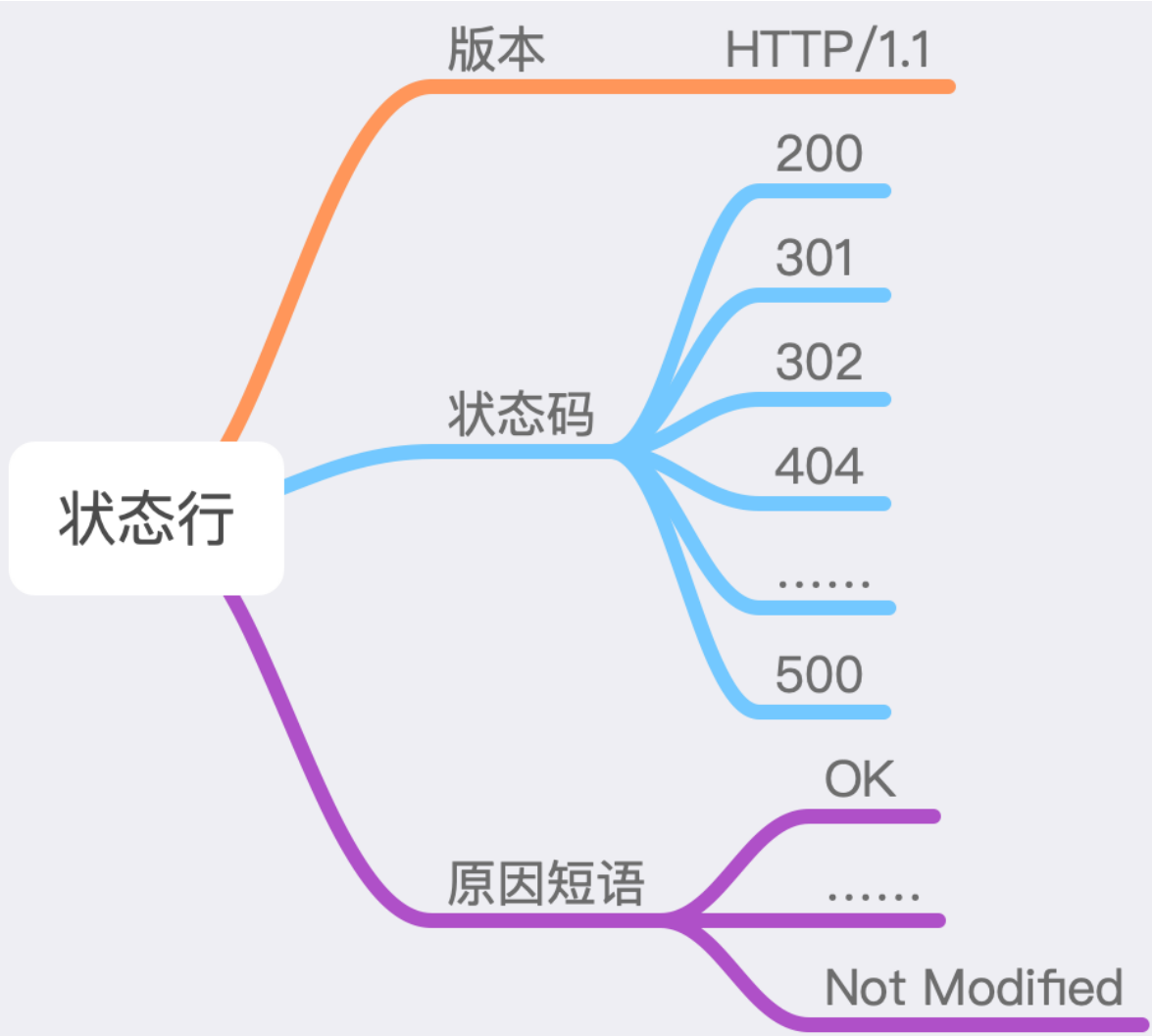
HTTP 响应报文

HTTP响应报文格式

可以看到，响应报文同样分为三部分：

- 状态行
- 响应头
- 相应内容

状态行

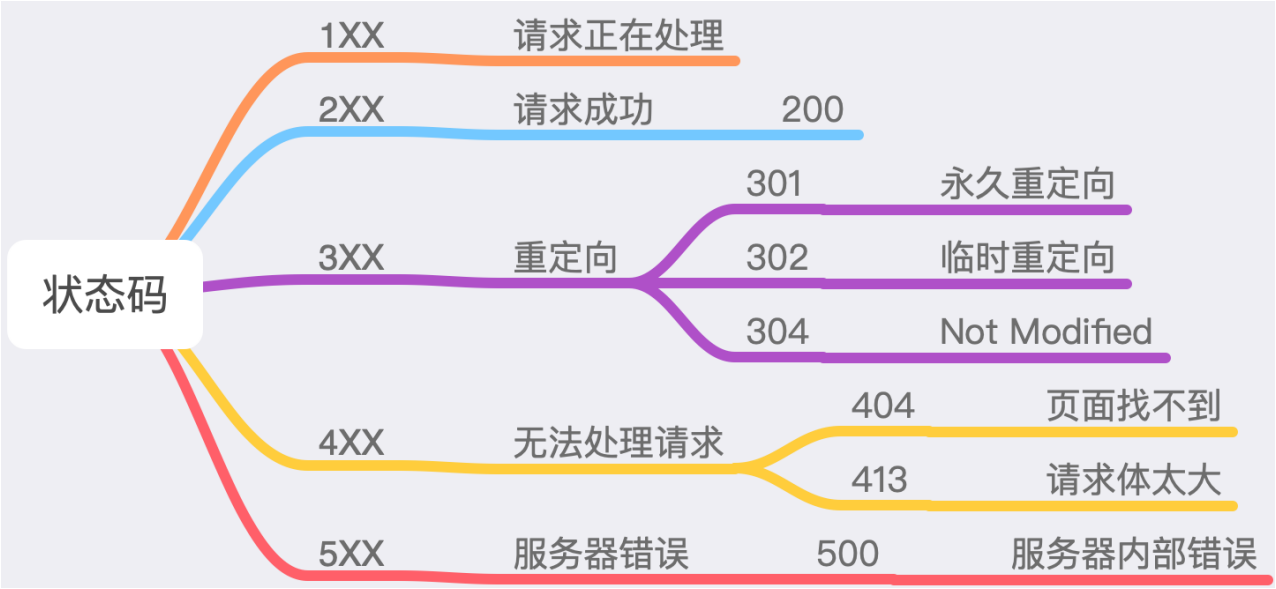


状态行分为三部分，分别是HTTP版本，状态码以及原因短语。

HTTP版本

当前使用的 HTTP 版本号，比如 HTTP/1.1

状态码和原因短语



状态码是一个数字，是为**计算机**设计的，而原因短语是当前状态码的一种可读文本，是为**人**设计的。

HTTP 协议中的状态码分为五大部分，大概60多种，下面我们列举几个会经常用到的：

- 200 : 这个是我们经常遇到的，表示当前的请求成功了；
- 301 : 表示当前请求的资源已经被永久的移动了了另外一个位置，响应的 Location 头部应该包含资源的新地址，客户端应该去另一个地址获取这个资源；
- 302 ： 表示当前请求的资源被临时的移动到了另外一个位置，响应的 Location 头部应该包含资源的新地址；
- 304 ： 如果请求头里面包含类似 If-Modified-Since 这样的选项，若服务器发现当前请求的资源不符合 If-Modified-Since 要求，那么就会返回 304 ,表示当前的资源没有发生改变，不用重新请求；
- 404 ： 这个应该是大家最熟悉的了，表示当前的资源不存在；
- 413 ： 当 POST 的数据太大的时候， Nginx 就会返回这个状态码，响应的原因短语是 Request Entity Too Large ；
- 500 ： 这个错误表示当前的服务器发生了错误，比如我们的代码有 bug 等。

响应头

我们在 请求头 部分说了 通用首部字段 和 请求首部字段 ， 下面我们看一下剩余的两种首部字段。

响应首部字段

响应首部字段是服务器向客户端返回的报文中使用的字段，该字段也有很多，我们讲几个比较常用的。

字段名	作用
Accept-Ranges	服务器用来表示自身支持的范围请求
Location	配合 301 和 302 状态码使用，表示资源位置发生了改变
Etag	资源标识

Accept-Ranges

该字段表示服务器自身是否支持 范围请求 ，它的值用于表示范围请求的单位。

格式：

```
1) Accept-Ranges: bytes
2) Accept-Ranges: none
```

- none

  不支持任何范围请求单位，由于其等同于没有返回此头部，因此很少使用。不过一些浏览器，比如IE9，会依据该头部去禁用或者移除下载管理器的暂停按钮。

- bytes

  范围请求的单位是 bytes （字节）。

Etag

服务器对返回的内容会计算一个数值，比如返回文件的`MD5`，这个值标识了当前返回的内容。
客户端根据这个值配合`If-Match`请求头使用，可以有效的降低网络带宽。

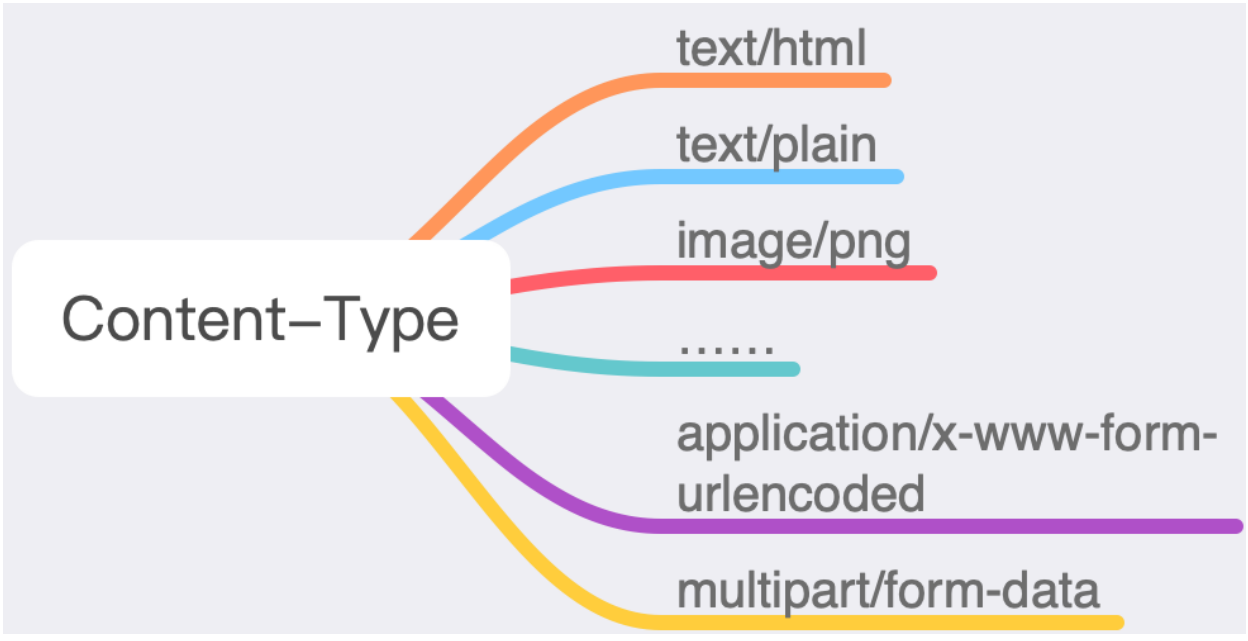
实体首部字段

实体首部字段可以用于请求报文，也可以用于响应报文，用于表示`实体`的一些相关特性。

字段名	作用
Content-Length	表示实体的大小，单位是字节
Content-Range	用于范围请求
Content-Type	实体的类型
Last-Modified	最后修改时间

Content-Type

表示实体的类型，这个类型非常非常的多，



但是我们常用的也就那么几个

- application/x-www-form-urlencoded

  默认的`GET`和`POST`编码方式，所有的数据都会变成键值对的形式，如 `key1=value1&key2=value2`。

- multipart/form-data

  如果我们上传资源的时候，那么必须使用这种格式。

Content-Length

这个字段的值代表的是实体的长度，以字节表示。

Content-Range

这个字段是配合 Range 请求使用的，适用于断点续传，下载等功能。
比如在下载的时候，可以使用多个进程，分别下载文件的一部分，到最后合并成一个文件，加快下载速度。

> 这个实体是非常费解的，英文叫做Entity，我的个人理解它就是请求或返回的body数据

响应body

这部分就是响应的主体部分哈~~

总结

HTTP 协议的内容很多，这两篇文章只是讲了一些比较基础的东西，知道了这些内容之后，后面我们在学习Nginx的时候会更加容易。

> 作为当前互联网的基础，HTTP协议是非常庞大的，我们只能简单的说一下比较常用的一些功能，如果你想了解更多的关于HTTP协议相关内容，可以参考HTTP RFC文档，《HTTP权威指南》、《图解HTTP》等书籍。

}

02 HTTP 初窥门径

04 与新朋友 Nginx 友好交流