

Http协议介绍

HTTP协议（HyperText Transfer Protocol，超文本传输协议）是因特网上应用最为广泛的一种网络传输协议，所有的WWW文件都必须遵守这个标准。

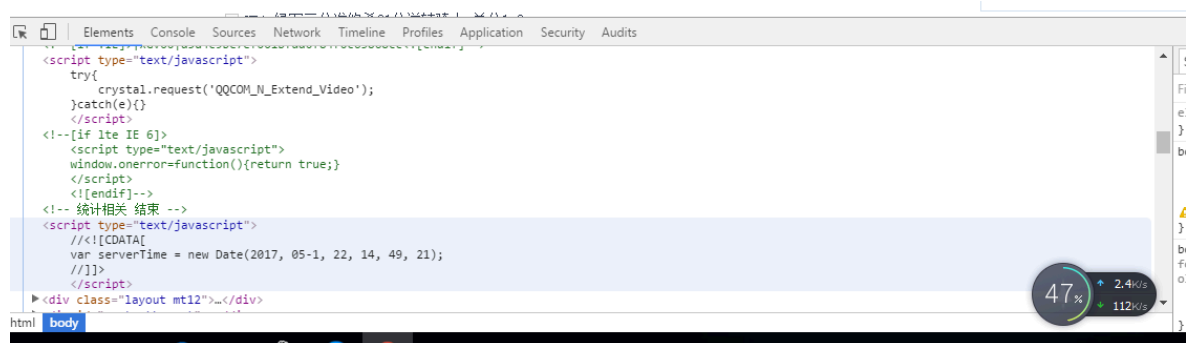
HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

在Web应用中，服务器把网页传给浏览器，实际上就是把网页的HTML代码发送给浏览器，让浏览器显示出来。而浏览器和服务器的传输协议是HTTP，所以：

HTML是一种用来定义网页的文本，会HTML，就可以编写网页；

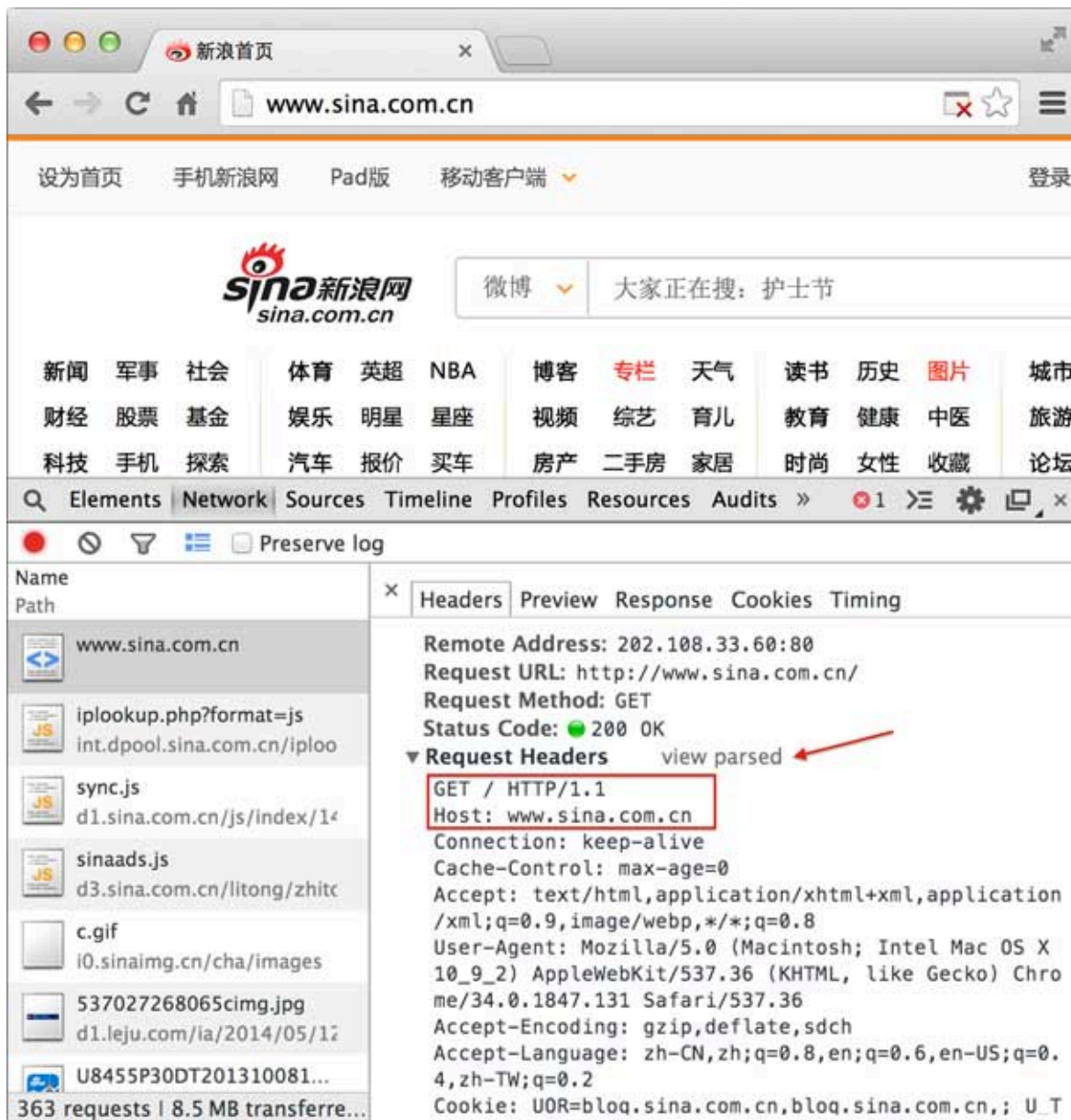
HTTP是在网络上传输HTML的协议，用于浏览器和服务器的通信。

Google Chrome的开发者工具（F12）



Elements显示网页的结构，**Network**显示浏览器和服务器的通信。我们点**Network**，确保第一个小红灯亮着，Chrome就会记录所有浏览器和服务器的通信：

以请求 sina.com.cn 为例,通过代码调试工具看 请求头与响应头



最主要的头两行分析如下，第一行：

GET / HTTP/1.1

GET表示一个读取请求，将从服务器获得网页数据，/表示URL的路径，URL总是以/开头，/就表示首页，最后的HTTP/1.1指示采用的HTTP协议

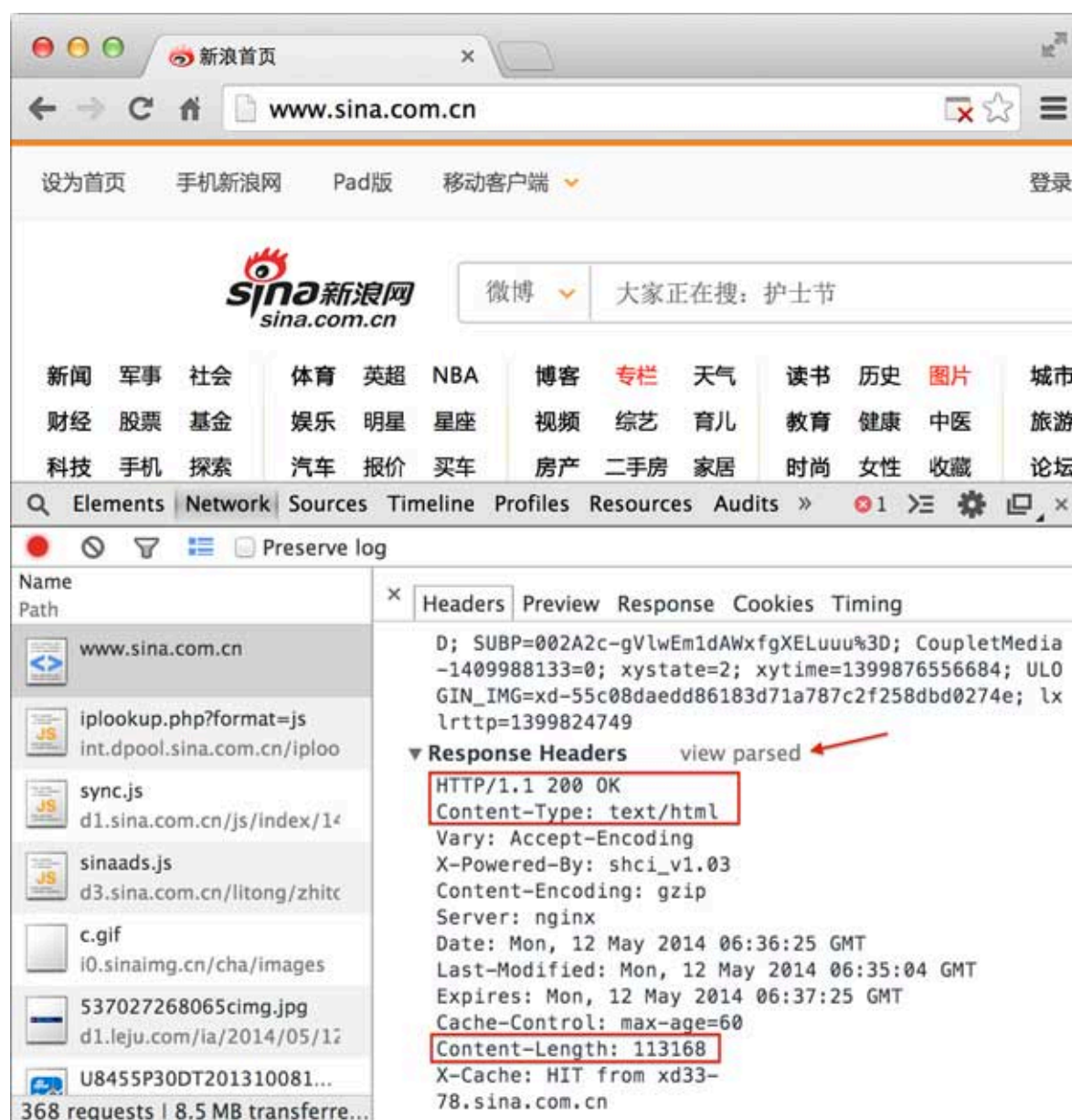
版本是1.1。目前HTTP协议的版本就是1.1，但是大部分服务器也支持1.0版本，主要区别在于1.1版本允许多个HTTP请求复用同一个TCP连接，以加快传输速度。

从第二行开始，每一行都类似于 **Xxx: abcdefg**:

Host: www.sina.com.cn

表示请求的域名是 **www.sina.com.cn**。如果一台服务器有多个网站，服务器就需要通过 **Host** 来区分浏览器请求的是哪个网站。

继续往下找到 **Response Headers**，点击 **view source**，显示服务器返回的原始响应数据：



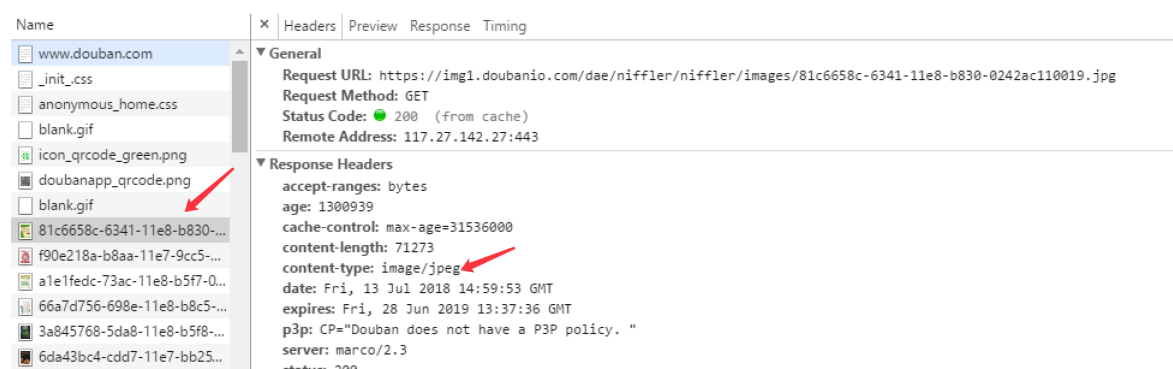
HTTP响应分为Header和Body两部分（Body是可选项），我们在Network中看到的Header最重要的几行如下：

200 OK

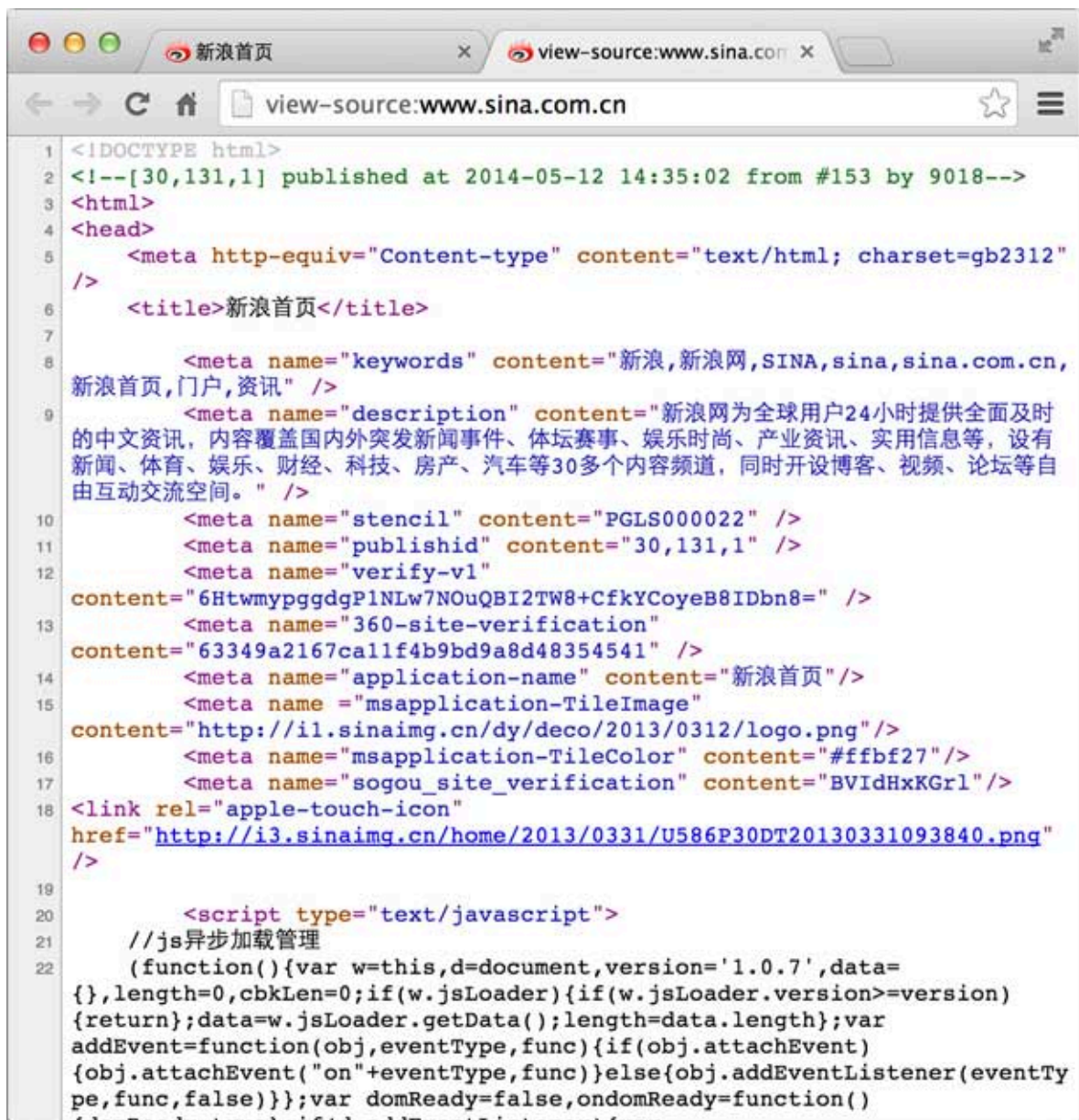
200表示一个成功的响应，后面的OK是说明。失败的响应有404 Not Found：网页不存在，500 Internal Server Error：服务器内部出错，等等。

Content-Type: text/html

Content-Type指示响应的内容，这里是text/html表示HTML网页。请注意，浏览器就是依靠Content-Type来判断响应的内容是网页还是图片，是视频还是音乐。浏览器并不靠URL来判断响应的内容，所以，即使URL是http://example.com/abc.jpg，它也不一定就是图片，我们可以找一个图片看一看它的响应头



HTTP响应的Body就是HTML源码，我们在菜单栏选择“视图”，“开发者”，“查看网页源码”就可以在浏览器中直接查看HTML源码：



```
1 <!DOCTYPE html>
2 <!--[30,131,1] published at 2014-05-12 14:35:02 from #153 by 9018-->
3 <html>
4 <head>
5   <meta http-equiv="Content-type" content="text/html; charset=gb2312"
6   />
7   <title>新浪首页</title>
8   <meta name="keywords" content="新浪,新浪网,SINA,sina,sina.com.cn,
9   新浪首页,门户,资讯" />
10  <meta name="description" content="新浪网为全球用户24小时提供全面及时
11  的中文资讯,内容覆盖国内外突发新闻事件、体坛赛事、娱乐时尚、产业资讯、实用信息等,设有
12  新闻、体育、娱乐、财经、科技、房产、汽车等30多个内容频道,同时开设博客、视频、论坛等自
13  由互动交流空间。" />
14  <meta name="stencil" content="PGLS000022" />
15  <meta name="publishid" content="30,131,1" />
16  <meta name="verify-v1"
17  content="6HtwmypggdgPlNLw7NOuQBI2TW8+CfkYCoyeB8IDbn8=" />
18  <meta name="360-site-verification"
19  content="63349a2167callf4b9bd9a8d48354541" />
20  <meta name="application-name" content="新浪首页" />
21  <meta name="msapplication-TileImage"
22  content="http://i1.sinaimg.cn/dy/deco/2013/0312/logo.png" />
23  <meta name="msapplication-TileColor" content="#ffbf27" />
24  <meta name="sogou_site_verification" content="BVIdHxKGrl" />
25  <link rel="apple-touch-icon"
26  href="http://i3.sinaimg.cn/home/2013/0331/U586P30DT20130331093840.png"
27  />
28  <script type="text/javascript">
29    //js异步加载管理
30    (function(){var w=this,d=document,version='1.0.7',data=
31    {},length=0,cbkLen=0;if(w.jsLoader){if(w.jsLoader.version>=version)
32    {return};data=w.jsLoader.getData();length=data.length};var
33    addEvent=function(obj,eventType,func){if(obj.attachEvent)
34    {obj.attachEvent("on"+eventType,func)}else{obj.addEventListener(eventTy
35    pe,func,false)}};var domReady=false,ondomReady=function()
```

当浏览器读取到新浪首页的HTML源码后，它会解析HTML，显示页面，然后，根据HTML里面的各种链接，再发送HTTP请求给新浪服务器，拿到相应的图片、视频、Flash、JavaScript脚本、CSS等各种资源，最终显示出一个完整的页面。所以我们在Network下面能看到很多额外的HTTP请求。

HTTP请求

跟踪了新浪的首页，我们来总结一下HTTP请求的流程：

步骤1：浏览器首先向服务器发送HTTP请求，请求包括：

方法：GET还是POST，GET仅请求资源，POST会附带用户数据(如通过表单提交帐号密码)

路径：/full/url/path；

域名：由Host头指定：Host: www.sina.com.cn

以及其他相关的Header；

如果是POST，那么请求还包括一个Body，包含用户数据。

步骤2：服务器向浏览器返回HTTP响应，响应包括：

响应代码：200表示成功，3xx表示重定向，4xx表示客户端发送的请求有错误，5xx表示服务器端处理时发生了错误；

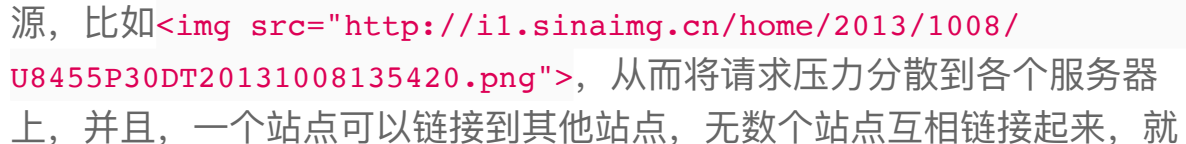
响应类型：由Content-Type指定；

以及其他相关的Header；

通常服务器的HTTP响应会携带内容，也就是有一个Body，包含响应的内容，网页的HTML源码就在Body中。

步骤3：如果浏览器还需要继续向服务器请求其他资源，比如图片，就再次发出HTTP请求，重复步骤1、2。

Web采用的HTTP协议采用了非常简单的请求-响应模式，从而大大简化了开发。当我们编写一个页面时，我们只需要在HTTP请求中把HTML发送出去，不需要考虑如何附带图片、视频等，浏览器如果需要请求图片和视频，它会发送另一个HTTP请求，因此，一个HTTP请求只处理一个资源。

HTTP协议同时具备极强的扩展性，虽然浏览器请求的是<http://www.sina.com.cn/>的首页，但是新浪在HTML中可以链入其他服务器的资源，比如，从而将请求压力分散到各个服务器上，并且，一个站点可以链接到其他站点，无数个站点互相链接起来，就

形成了World Wide Web，简称WWW。

HTTP请求方法

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP1.0定义了三种请求方法：GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

7	OPTI ONS	允许客户端查看服务器的性能。
8	TRA CE	回显服务器收到的请求，主要用于测试或诊断。

HTTP状态码

HTTP状态码

当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

HTTP状态码的英文为HTTP Status Code。

下面是常见的HTTP状态码：

- 200 - 请求成功
- 301 - 资源（网页等）被永久转移到其它URL
- 404 - 请求的资源（网页等）不存在
- 500 - 内部服务器错误

HTTP状态码分类

HTTP状态码由三个十进制数字组成，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

HTTP状态码分类

分类	分类描述
----	------

1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

400 Bad Request

- 1、语义有误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求。
- 2、请求参数有误。

403 Forbidden

服务器已经理解请求，但是拒绝执行它。与401响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。如果这不是一个 HEAD 请求，而且服务器希望能够讲清楚为何请求不能被执行，那么就应该在实体内描述拒绝的原因。当然服务器也可以返回一个404响应，假如它不希望让客户端获得任何信息。

404 Not Found (请求资源不存在)

请求失败，请求所希望得到的资源未被在服务器上发现。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况的话，应当使用410状态码来告知旧资源因为某些内部的配置机制问题，已经永久的不可用，而且没有任何可以跳转的地址。404这个状态码被广泛应用于当服务器不想揭示到底为何请求被拒绝或者没有其他适合的响应可用的情况下。出现这个错误的最有可能的原因是服务器端没有这个页面。

500 Internal Server Error

服务器遇到了一个未曾预料的状态，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器端的源代码出现错误时出现。

502 Bad Gateway

作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。

503 Service Unavailable

由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。如果能够预计延迟时间，那么响应中可以包含一个 `Retry-After` 头用以标明这个延迟时间。如果没有给出这个 `Retry-After` 信息，那么客户端应当以处理500响应的方式处理它。

注意：503状态码的存在并不意味着服务器在过载的时候必须使用它。某些服务器只不过是希望拒绝客户端的连接。

504 Gateway Timeout

作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应。

注意：某些代理服务器在DNS查询超时时会返回400或者500错误

505 HTTP Version Not Supported

服务器不支持，或者拒绝支持在请求中使用的 HTTP 版本。这暗示着服务器不能或不愿使用与客户端相同的版本。响应中应当包含一个描述了为何版本不被支持以及服务器支持哪些协议的实体。

通过requests模块请求URL 并通过传入headers 为了让这个请求更是人类用户而不是爬虫

HTTP的请求头是每次向网络服务器发送请求的时候 传递的一组属性和配置信息 HTTP 定义了十几种古怪的请求头信息 不过大多数都不常用 下面的为常用的请求头部信息

普通浏览器的请求头

```
▼ Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8
Connection: keep-alive
Host: www.whatismybrowser.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
```

所以通过请求头信息就可以区分哪些是真实的用户哪些是爬虫

请求头可以通过 `urllib2` 或 `request` 模块自己定义 虽然网站可能会对HTTP请求头的每个属性进行检查 但是最重要的可能被检查的参数信息是 `User-Agent` 所以无论做什么爬虫项目 建议都要修改 `User-Agent` 属性 将其设置成不容易引起怀疑的内容 如果是一些警觉性比较高的网站 可能还会检查 比如 `Accept-Lanuage` 这些属性 请将这些属性妥善配置

代码:

```
import requests
```

```
url = 'http://www.whatismybrowser.com'
#headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36'}
response = requests.get(url, headers=headers)
print response.status_code, response.text
```

##通过以上代码 我们成功修改了请求头部信息