- **http协议**
  - HTTP1.0的请求方法
  - HTTP1.1新增的请求方法
  - http协议-请求
    - 常用的请求报头
    - http请求-GET
      - Server端可以根据参数名获取值：
    - http请求-POST
      - 混合模式
      - 文本模式
    - http请求-HEAD
    - http请求-options
    - HTTP请求 – PUT DELETE
      - PUT / DELETE使用方法
        - DELETE使用方法
    - HTTP请求 – TRACE ,CONNECT
  - http状态码
    - HTTP状态码的分类
    - 常见的状态码
      - 状态码含义-1xx
      - 状态码含义-2xx
      - 状态码含义-3xx
      - 状态码含义-4xx
      - 状态码含义-5xx
    - 用计算机语言获取HTTP状态码的方法
    - http协议响应头信息(refer location)
      - 常用标准响应头字段
      - 响应头格式
    - http协议中的URL
      - url
      - url编码格式
      - url同源策略

http协议

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。是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。

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）。

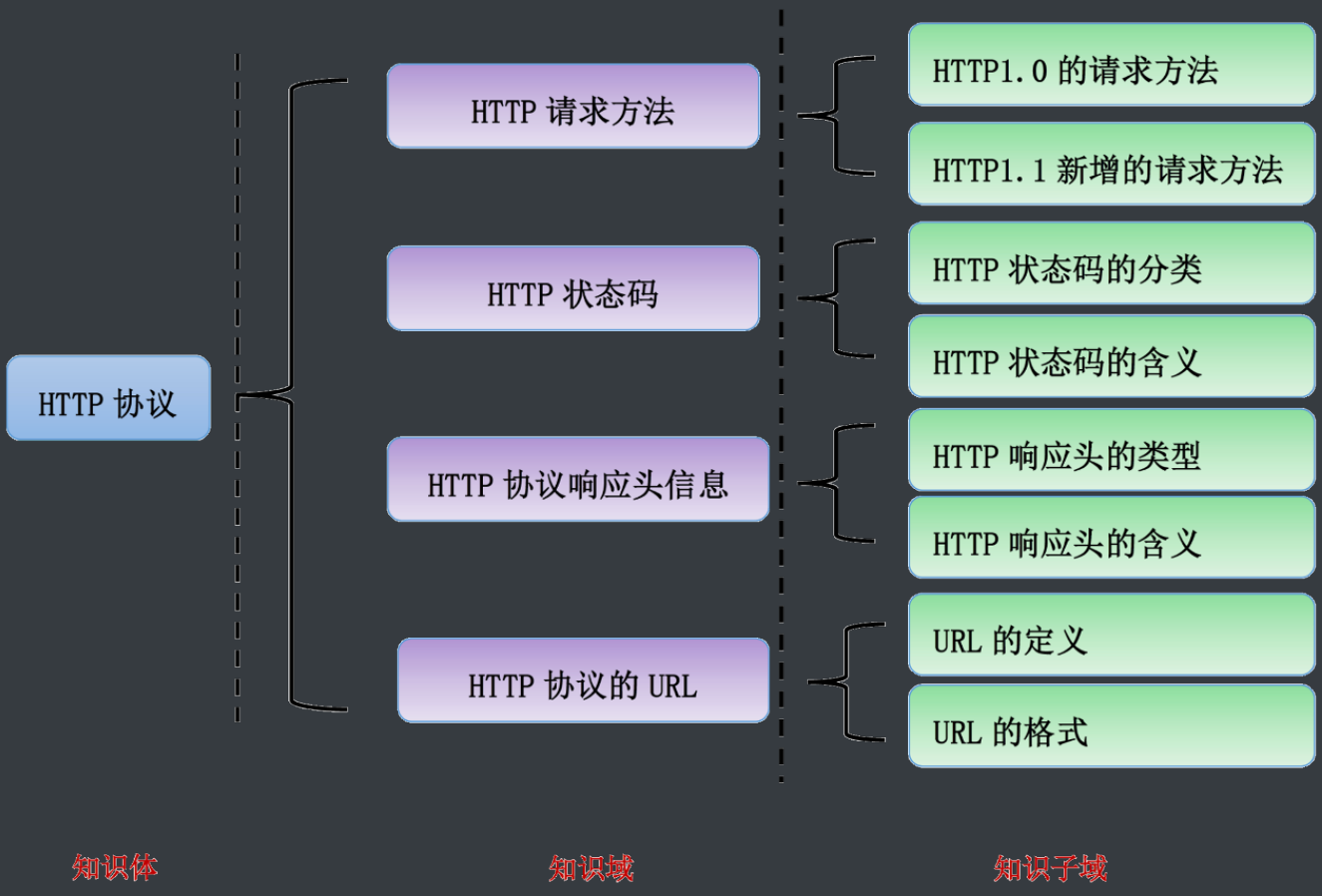
HTTP的发展是由蒂姆·伯纳斯-李于1989年在欧洲核子研究组织（CERN）所发起。HTTP的标准制定由万维网协会（World Wide Web Consortium，W3C）和互联网工程任务组（Internet Engineering Task Force，IETF）进行协调，最终发布了一系列的RFC，其中最著名的是1999年6月公布的 RFC 2616，定义了HTTP协议中现今广泛使用的一个版本——HTTP 1.1。

2014年12月，互联网工程任务组（IETF）的Hypertext Transfer Protocol Bis（httpbis）工作小组将HTTP/2标准提议递交至IESG进行讨论，于2015年2月17日被批准。 HTTP/2标准于2015年5月以RFC 7540正式发表，取代HTTP 1.1成为HTTP的实现标准。

HTTP是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP）。通过使用网页浏览器、网络爬虫或者其它的工具，客户端发起一个HTTP请求到服务器上指定端口（默认端口为80）。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如HTML文件和图像。我们称这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理服务器、网关或者隧道（tunnel）。

尽管TCP/IP协议是互联网上最流行的应用，HTTP协议中，并没有规定必须使用它或它支持的层。事实上，HTTP可以在任何互联网协议上，或其他网络上实现。HTTP假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用。因此也就是其在TCP/IP协议族使用TCP作为其传输层。

通常，由HTTP客户端发起一个请求，创建一个到服务器指定端口（默认是80端口）的TCP连接。HTTP服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息、或者其它信息。



HTTP协议包含4个知识域：HTTP请求方法，HTTP状态码，HTTP协议响应头信息，HTTP协议的URL。

HTTP请求方法里会介绍八种请求方法（HTTP1.0 三种 – GET，POST，HEAD。 HTTP1.1新增的五种：OPTIONS，PUT，DELETE，TRACE，CONNECT）。

goby红队专版，owasp, awvs

HTTP状态码中会介绍各种返回的状态码的意义：1xx，2xx，3xx，4xx，5xx各代表什么。

HTTP协议响应头信息会介绍响应头包含的信息，会有何种多余的信息。

HTTP协议的URL会介绍URL结构

HTTP1.0的请求方法

了解HTTP1.0三种请求方法，GET, POST 和 HEAD

掌握GET请求的标准格式

掌握POST请求提交表单，上传文件的方法

了解HEAD请求与GET请求的区别

HTTP1.1新增的请求方法

了解HTTP1.1新增的五种请求方法：OPTIONS，PUT，DELETE，TRACE，CONNECT方法的基本概念

掌握HTTP1.1新增的五种请求的基本方法和产生的请求结果

http协议-请求

http请求由三部分组成，分别是：请求行、消息报头、请求正文

请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，格式如下：

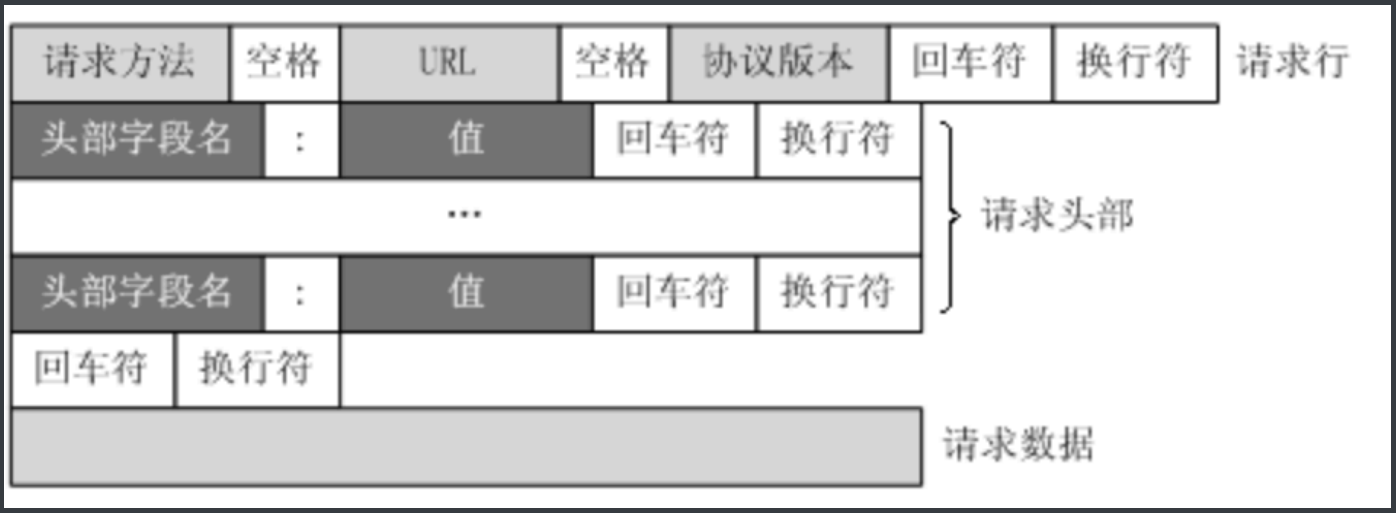
```
Method Request-URI HTTP-Version (CR)(LF)
```

其中 Method表示请求方法(GET,POST,HEAD等），Request-URI是一个统一资源标识符，HTTP-Version表示请求的HTTP协议版本，(CR)(LF)表示回车和换行（除了作为结尾的(CR)(LF)外，不允许出现单独的(CR)或(LF)字符）。

例：

```
GET /get.php?arg1=value1 HTTP/1.1
```

HTTP消息请求报头允许客户端向服务器端传递请求的附加信息以及客户端自身的信息。




```
GET /events HTTP/1.1
Host: hazelshishuaige.club:8000  # 访问的目的地址
Accept: text/event-stream   # 接收的数据类型 image/jpeg。text/plainssss
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36
# 浏览器的指纹信息
Referer: http://hazelshishuaige.club:8000/login?next=%2Fchallenges%3F  #
从哪一个页面访问过来的
Accept-Language: zh-CN,zh;q=0.9  # 可以接收的语言
Cookie: session=942c2275-7738-45eb-aaaa-50 # cookie 身份凭证
Connection: close
```

常用的请求报头

Accept，Accept-Charset，Accept-Encoding，Accept-Language，Authorization，Host（发送请求时，该报头域是必需的），User-Agent 每个类型请求头结束后，会跟上(CR)(LF) 。消息报头和请求正文会隔一行

常用的请求头：

```
Accept:
请求报头域用于指定客户端接受哪些类型的信息。
```

eg:
Accept：image/gif，表明客户端希望接受GIF图象格式的资源；
Accept：text/html，表明客户端希望接受html文本。

Accept-Charset
Accept-Charset请求报头域用于指定客户端接受的字符集。
eg:
Accept-Charset:iso-8859-1,gb2312.如果在请求消息中没有设置这个域，缺省是任何字符集都可以接受。

Accept-Encoding
Accept-Encoding请求报头域类似于Accept，但是它是用于指定可接受的内容编码。
eg:
Accept-Encoding:gzip.deflate.如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受。

Accept-Language
Accept-Language请求报头域类似于Accept，但是它是用于指定一种自然语言。
eg:
Accept-Language:zh-cn.如果请求消息中没有设置这个报头域，服务器假定客户端对各种语言都可以接受。

Authorization
Authorization请求报头域主要用于证明客户端有权查看某个资源。当浏览器访问一个页面时，如果收到服务器的响应代码为401（未授权），可以发送一个包含Authorization请求报头域的请求，要求服务器对其进行验证。

Host（发送请求时，该报头域是必需的）
Host请求报头域主要用于指定被请求资源的Internet主机和端口号，它通常从HTTP URL中提取出来的，
eg:
我们在浏览器中输入：http://www.guet.edu.cn/index.html 浏览器发送的请求消息中，就会包含Host请求报头域，如下：
Host：www.guet.edu.cn
此处使用缺省端口号80，若指定了端口号，则变成：Host：www.guet.edu.cn:指定端口号

```
User-Agent
我们上网登陆论坛的时候，往往会看到一些欢迎信息，其中列出了你的操作系统的名称和版本，你所使用的浏览器的名称和版本，这往往让很多人感到很神奇，实际上，服务器应用程序就是从User-Agent这个请求报头域中获取到这些信息。User-Agent请求报头域允许客户端将它的操作系统、浏览器和其它属性告诉服务器。不过，这个报头域不是必需的，如果我们自己编写一个浏览器，不使用User-Agent请求报头域，那么服务器端就无法得知我们的信息了。

xff
X-FORWARDED-FOR:127.0.0.1  #来源IP
```

http请求-GET

GET请求格式:

```
<访问路径>[?<arg1>=<value1>[&<arg2>=<value2>....]]

例子: http://site1.com/get.php?arg1=value1
```

Server端可以根据参数名获取值:

PHP例子:

```
<?php
echo $_GET['arg1'];
?>
```

结果就是输出arg1相应的值

```
curl "http://192.168.0.105/get.php?arg1=value1"

value1
```

GET请求，是可以把数据放在URL中来传递，也可以不包含任何数据，HTTP请求只有请求头，没有请求数据。

请求头中可以不包含Content-Length，例子：

```
GET /get.php?arg1=value1 HTTP/1.1
Host: site1.com
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.12.2) GCDHTTPRequest
```

GET请求可以只有请求的路径：

```
http://www.gooann.com
```

GET请求也可以带需要传递的数据，在访问路径之后带问号（?）+参数=值的方式发送。

```
http://site1.com/get.php?arg1=value1
```

在服务端使用相应的方法（函数）来获取相应的值，比如PHP可以用如下方法：

```
<?php
echo $_GET['arg1'];
?>
```

结果是输出arg1相应的值。

http请求-POST

POST请求是包含数据。请求数据的格式，可以在HTTP头中定义。格式一般会有：

```
表单格式：application/x-www-form-urlencoded
混合格式：multipart/form-data
JSON格式：application/json
XML格式：text/xml
文本：text/plain
```

表单格式：

与get方式类似，是把所有提交数据放在数据区域。POST方式也可以像GET方式在URL带参数，但一般不会这么去使用。表单方式与GET方式类似，只是把数据放在头文件下面的请求正文区域。请求如下：

```
POST /post-form.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 192.168.0.105
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.12.2) GCDHTTPRequest
Content-Length: 11

arg1=value1
```

混合模式

有文件上传时常用的方法。可以接受同时提交不同类型的数据。表单中，可以把类型更改为file就可以上传文件

```
<input type="file" name="file" id="file" />
```

类型后面一般会跟boundary来告知数据区域分隔符。每个数据都可以单独说明数据类型。获取文件时，可以使用相应参数：

PHP代码如下：

```
$_FILES["file"]["name"] - 文件名
$_FILES["file"]["type"] - 类型
$_FILES["file"]["size"] - 文件大小
$_FILES["file"]["tmp_name"] - 临时文件路径
```

混合模式一般是用来传输文件。后面会跟boundary= `__xxxx__` 来进行每个参数的分割。

```
POST /upload_file.php HTTP/1.1
Content-Type: multipart/form-data; boundary=------
WebKitFormBoundaryRTP6hG23yFYrExfg
```

```
Host: 192.168.0.105
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.12.2) GCDHTTPRequest
Content-Length: 101

------WebKitFormBoundaryRTP6hG23yFYrExfg
Content-Disposition: form-data; name="arg1"

value1
------WebKitFormBoundaryRTP6hG23yFYrExfg
Content-Disposition: form-data; name="file"; filename="python.txt"
Content-Type: text/plain

多线程
xxxxx
------WebKitFormBoundaryRTP6hG23yFYrExfg
Content-Disposition: form-data; name="file2"; filename="mails.zip"
Content-Type: application/zip

xxxxx
------WebKitFormBoundaryRTP6hG23yFYrExfg--
```

文本模式

常见类型为json,xml,plain，这种类型的数据，需要服务端代码自行解析

PHP代码为例：

```
file_get_contents("php://input"); // 可以获取数据区域文本
```

要是接受的是json，使用相关方法（函数）来解析

```
echo json_decode(file_get_contents("php://input"),true)['arg1'];
```

可以根据type（类型）来解开数据。

文本模式，也可以按照文件来接收，使用 `file_get_contents(“php://input”);` 可以免去读取文件里的内容。 `file_get_contents(“php://input”);` 模式不能接收multipart/form-data模式。

下面代码就是PHP写的，当类型为json时，根据json解析

```
<?php
if( $_SERVER['CONTENT_TYPE'] == 'application/json')
{
    echo json_decode(file_get_contents("php://input"),true)['arg1'];
}
?>
```

这时请求数据是：{“arg1”:“value1”} 返回值是：value1

http请求-HEAD

HEAD请求就是返回只有头部数据，数据部分不返回内容。返回的内容基本上与GET，POST的返回头一致

GET模式的返回头：

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
X-Powered-By: ASP.NET
X-UA-Compatible: IE=EmulateIE7
Date: Sun, 20 Aug 2017 07:11:17 GMT
Connection: close
Content-Length: 45511
```

HEAD模式的返回头：

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 45511
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
X-Powered-By: ASP.NET
X-UA-Compatible: IE=EmulateIE7
Date: Sun, 20 Aug 2017 07:11:51 GMT
Connection: close
```

http请求-options

OPTIONS请求，默认情况下会返回允许的请求类型 `http://www.microsoft.com/zh-cn/` 会返回：

```
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type,
Accept
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS
Access-Control-Allow-Credentials: true
```

一般需要跨域的时候需要设置OPTIONS（跨域后面来讲）。使用脚本让浏览器跨域进行请求的时候，会检测OPTIONS，对方服务器是否允许跨域。允许的情况下，才会真正去进行相应请求。当你用浏览器访问 site1.com, 某些脚本操作会提交到site2.com而且附带非浏览器默认的头信息，这个时候浏览器不会直接发出POST请求，先发出OPTIONS请求来判断是否允许发送POST。当对方回复允许了，才能发送POST请求。

只有在特定情况下才会使用。

```
Access-Control-Allow-Headers 定义了允许的附加的headers。
Access-Control-Allow-Methods 定义了跨域访问允许的请求方法
```

列子：

```
var invocation = new XMLHttpRequest();
invocation.open('POST', 'http://www.microsoft.com/zh-cn/', true);
invocation.setRequestHeader('X-Requested-With',
'http://www.microsoft.com');
invocation.send();
```

会先请求：

```
OPTIONS /zh-cn/ HTTP/1.1
Host: www.microsoft.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:54.0)
Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Access-Control-Request-Method: POST
Access-Control-Request-Headers: x-requested-with
Origin: http://192.168.0.105
Connection: close
Cache-Control: max-age=0
```

参考：`http://blog.csdn.net/thc1987/article/details/54572272`

HTTP请求 - PUT DELETE

PUT：在特定目录里上传指定文件，文件名在url中设置。

DELETE：删除特定目录里的文件，文件名在url中设置。

在NGINX中，可以添加如下配置来允许PUT，DELETE

```
location /upload/ {
  dav_methods PUT DELETE;
  root  /usr/share/nginx/html;
}
```

PUT / DELETE使用方法

PUT - 发送包

```
PUT /upload/Untitled HTTP/1.1
Content-Type: application/octet-stream
Host: 192.168.1.64
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.12.6) GCDHTTPRequest
Content-Length: 284

<data>....
```

IISput漏洞，CVE-2017-12615 put漏洞

PUT – 返回包

```
HTTP/1.1 201 Created
Server: nginx/1.12.1
Date: Fri, 01 Sep 2017 13:40:03 GMT
Content-Length: 0
Location: http://192.168.1.64/upload/Untitled
Connection: close
```

DELETE使用方法

DETELE – 发送包

```
DELETE /upload/Untitled HTTP/1.1
Host: 192.168.1.64
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.12.6) GCDHTTPRequest
```

DELETE – 返回包

```
HTTP/1.1 204 No Content
Server: nginx/1.12.1
Date: Fri, 01 Sep 2017 13:51:07 GMT
Connection: close
```

HTTP请求 – TRACE ,CONNECT

HTTP TRACE是让我们的web服务器端将客户端的所有请求信息返回给客户端的方法，该方法多见于debug的需求。

CONNECT是在特定应用走HTTP协议时会用到

proxy可能会用到

某些使用http协议，需要长链接的程序（SSL就使用connect）

http状态码

通过本知识域，我们需要了解：

HTTP状态码的分类

1. 了解HTTP状态码的规范？
2. 了解HTTP状态码的作用？
3. 掌握常见的HTTP状态码？

HTTP状态码的含义

1. 了解HTTP状态码2**,3**,4**,5** 代表的含义？
2. 掌握用计算机语言获取HTTP状态码的方法？

HTTP状态码的分类

HTTP状态码的作用是：Web服务器用来告诉客户端，发生了什么事。

HTTP状态码被分为五大类， 目前我们使用的HTTP协议版本是1.1， 支持以下的状态码。

	已定义范围	分类
1XX	100-101	信息提示
2XX	200-206	成功
3XX	300-307	重定向
4XX	400-417	客户端错误
5XX	500-505	服务端错误

当我们看到状态码，一般只需要看第一位，就大致知道返回状态。

1XX： 信息提示，一般不会出现

2XX： 请求返回正常，可能根据请求状况稍微不同

3XX： 请求重定向到其它资源（有可能是本地）

4XX： 请求的页面错误（不存在，权限不够等）

5XX： 服务端问题（配置，代码等问题）

常见的状态码

`200` OK 服务器成功处理了请求（这个是我们见到最多的）

`301/302` Moved Permanently（重定向）请求的URL已移走。Response中应该包含一个Location URL, 说明资源现在所处的位置

`304` Not Modified（未修改）客户的缓存资源是最新的， 要客户端使用缓存

`404` Not Found 未找到资源

`501` Internal Server Error服务器遇到一个错误，使其无法对请求提供服务

状态码含义-1xx

状态码	状态消息	含义
100	Continue(继续)	收到了请求的起始部分，客户端应该继续请求
101	Switching Protocols（切换协议）	服务器正根据客户端的指示将协议切换成Update Header列出的协议

状态码含义-2xx

状态码	状态消息	含义
200	OK	服务器成功处理了请求（这个是我们见到最多的
201	Created（已创建）	对于那些要服务器创建对象的请求来说，资源已创建完毕。
202	Accepted（已接受）	请求已接受， 但服务器尚未处理
203	Non-Authoritative Information（非权威信息）	服务器已将事务成功处理，只是实体Header包含的信息不是来自原始服务器，而是来自资源的副本。
204	No Content(没有内容)	Response中包含一些Header和一个状态行， 但不包括实体的主题内容（没有response body）
205	Reset Content(重置内容)	另一个主要用于浏览器的代码。意思是浏览器应该重置当前页面上所有的HTML表单。
206	Partial Content（部分内容）	部分请求成功

2XX中200是最常见的。请求正常返回的时候状态码就是200。

201: 一般是PUT请求创建成功之后就会返回201

206状态码代表服务器已经成功处理了部分GET请求（只有发送GET 方法的request, web服务器才可能返回206）。

1. FlashGet, 迅雷或者HTTP下载工具都是使用206状态码来实现断点续传。
2. 将以个大文档分解为多个下载段同时下载 比如，在线看视频。

状态码含义-3xx

状态码	状态消息	含义
300	Multiple Choices（多项选择）	客户端请求了实际指向多个资源的URL。这个代码是和一个选项列表一起返回的，然后用户就可以选择他希望的选项了
301	Moved Permanently（永久移除)	请求的URL已移走。Response中应该包含一个Location URL, 说明资源现在所处的位置
302	Found（已找到）	与状态码301类似。但这里的移除是临时的。 客户端会使用Location中给出的URL，重新发送新的HTTP request
303	See Other（参见其他）	类似302
304	Not Modified（未修改）	客户的缓存资源是最新的， 要客户端使用缓存
305	Use Proxy（使用代理）	必须通过代理访问资源， 代理的地址在Response 的Location中
306	未使用	这个状态码当前没使用
307	Temporary Redirect（临时重定向	类似302

301/302都是重定向。都会在头文件里的Location指定重定向页面

304，是本地有缓存。请求的时候会有If-Modified-Since: <时间>。在这时间之后没有更改，就会返回304

状态码含义-4xx

状态码	状态消息	含义
400	Bad Request（坏请求）	告诉客户端，它发送了一个错误的请求。
401	Unauthorized（未授权）	需要客户端对自己认证
402	Payment Required（要求付款）	这个状态还没被使用， 保留给将来用
403	Forbidden（禁止）	请求被服务器拒绝了
404	Not Found（未找到）	未找到资源
405	Method Not Allowed（不允许使用的方法）	不支持该Request的方法。
406	Not Acceptable（无法接受）	
407	Proxy Authentication Required(要求进行代理认证)	与状态码401类似， 用于需要进行认证的代理服务器
408	Request Timeout（请求超时）	如果客户端完成请求时花费的时间太长， 服务器可以回送这个状态码并关闭连接
409	Conflict（冲突）	发出的请求在资源上造成了一些冲突
410	Gone（消失了）	服务器曾经有这个资源，现在没有了， 与状态码404类似
411	Length Required（要求长度指示）	服务器要求在Request中包含Content-Length。
412	Precondition Failed（先决条件失败）	
413	Request Entity Too Large（请求实体太大）	客户端发送的实体主体部分比服务器能够或者希望处理的要大
414	Request URI Too Long（请求URI	客户端发送的请求所携带的URL超过了服务

	太长）	器能够或者希望处理的长度
415	Unsupported Media Type（不支持的媒体类型）	服务器无法理解或不支持客户端所发送的实体的内容类型
416	Requested Range Not Satisfiable（所请求的范围未得到满足）	
417	Expectation Failed（无法满足期望）	

常见的是403: 没有权限访问， 404: 没有相应的页面。

405是HTTP8中请求中，服务端不支持相应请求时会返回405。

状态码含义-5xx

状态码	状态消息	含义
500	Internal Server Error(内部服务器错误)	服务器遇到一个错误，使其无法为请求提供服务
501	Not Implemented（未实现）	客户端发起的请求超出服务器的能力范围(比如，使用了服务器不支持的请求方法)时，使用此状态码。
502	Bad Gateway（网关故障）	代理使用的服务器遇到了上游的无效响应
503	Service Unavailable（未提供此服务）	服务器目前无法为请求提供服务，但过一段时间就可以恢复服务
504	Gateway Timeout（网关超时）	与状态吗408类似， 但是响应来自网关或代理，此网关或代理在等待另一台服务器的响应时出现了超时
505	HTTP Version Not Supported（不支持的HTTP版本）	服务器收到的请求使用了它不支持的HTTP协议版本。 有些服务器不支持HTTP早期的HTTP协议版本，也不支持太高的协议版本

500: 我们开发网站的时候，当我们的程序出错了时，就会返回500错误。

501: 客户端发起的请求超出服务器的能力范围。按下面尝试：

```
PUT / HTTP/1.1
Host: www.qq.com
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.12.2) GCDHTTPRequest
Content-Length: 0
```

502: 代理服务器连不上上游服务器

用计算机语言获取HTTP状态码的方法

我们可以使用linux里的curl

```
curl -I -m 10 -o /dev/null -s -w "%{http_code}\n" <URL>
```

我们也可以使用Python等语言：

```
import urllib2
urllib2.urlopen('<URL>').code
```

我们可以使用curl：

```
curl -I -m 10 -s -w "%{http_code}\n" "http://baidu.com"

200
```

或者使用简单的脚本语言：Python

```
>>> import urllib2
>>> urllib2.urlopen('http://baidu.com').code

200
```

http协议响应头信息(refer location)

通过本知识域，我们需要了解：

HTTP响应头的含义

1. 了解常见的HTTP响应头
2. 掌握HTTP响应头的作用

HTTP响应头的类型

1. 了解HTTP响应头的名称
2. 掌握HTTP响应头的格式

常用标准响应头字段

Access：服务器支持哪些请求方法（如GET、POST等）。

Content-Encoding：文档的编码（Encode）方法。

Content-Length：表示内容长度。

Content-Type：表示后面的文档属于什么MIME类型。

Date：当前的GMT时间。

Expires：应该在什么时候认为文档已经过期，从而不再缓存它

Last-Modified：文档的最后改动时间。

Location：表示客户应当到哪里去提取文档。

Refresh：表示浏览器应该在多少时间之后刷新文档，以秒计。

Server：服务器名字。

Set-Cookie：设置和页面关联的Cookie。

WWW-Authenticate：标识访问请求实体的身份验证方案

更多的，请参考RFC2612

响应头格式

```
HTTP/1.1 302 Found
Date: Sun, 20 Aug 2017 13:38:54 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16
Location: https://www.aqzhi.com/
Content-Length: 206
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"><html><head><title>302
Found</title></head><body><h1>Found</h1><p>The document has moved <a
href="https://www.aqzhi.com/">here</a>.</p></body></html>
```

第一行是 http版本，状态码，状态码说明。

第二行开始就是头信息各个值。key：value形式。

头信息结束后，空一行，然后就是返回数据

http协议中的URL

通过本知识域，我们需要了解：

URL的基本构成

1. 了解URL的基本概念

2. 了解URL的结构
3. 掌握URL的编码格式

url

URL是统一资源定位符，是互联网上标准资源的地址

URL包含

1. 协议
2. 用户名:密码
3. 主机 - 子域名.域名.顶级域名（或IP）
4. 端口号
5. 目录/文件名.文件后缀
6. 参数=值
7. 标志

格式：

```
协议://用户名:密码@子域名.域名.顶级域名:端口号/目录/文件名.文件后缀?参数=值#标志
http://admin:password@www.baidu.com:8801/1/1.php?url=a
http://www.baidu.com/ls/l.php
```

相对URL：

```
/目录/文件名.文件后缀?参数=值#标志
```

URL是UniformResourceLocation的缩写,译为“统一资源定位符”。可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。互联网上的每个文件都有一个唯一的URL，它包含的信息指出文件的位置以及浏览器应该怎么处 理它。

URL中协议包含如下：

```
http--超文本传输协议资源
https--用安全套接字层传送的超文本传输协议 443
ftp--文件传输协议
mailto--电子邮件地址
ldap--轻型目录访问协议搜索
file--当地电脑或网上分享的文件
news--Usenet新闻组
gopher--Gopher协议
telnet--Telnet协议
```

url编码格式

只有字母和数字[0-9a-zA-Z]、一些特殊符号“$-_.+!*‘(),”[不包括双引号]、以及某些保留字，才可以不经过编码直接用于URL。

编码格式为16进制，每两个16进制前加百分号（%）

```
例：“你好”的utf-8码为：\xe4\xbd\xa0\xe5\xa5\xbd

“你好”的URL – utf-8格式编码为： %E4%BD%A0%E5%A5%BD
```

URL编码根据服务端解析方式进行转码。

url同源策略

URL格式中，协议，主机，端口三部分相同，才能算是同源。

浏览器设置里，默认情况下只有同源的内容才能相互操作。

```
打开site1.com

创建iframe，打开site2.com

site1.com的js访问site2.com的内容

这个时候会报错，不是同源不能进行相关操作。
```

```
<iframe>标签用于定义了一个独立加载文档的内嵌框架。
```

```
A:<iframe id="mainIframe" name="mainIframe" src="/main.html"
frameborder="0" scrolling="auto" ></iframe>
B:<iframe id="mainIframe" name="mainIframe" src="http://www.baidu.com"
frameborder="0" scrolling="auto" ></iframe>
```

使用A时，因为同域，父页面可以对子页面进行改写,反之亦然。
使用B时，不同域，父页面没有权限改动子页面,但可以实现页面的跳转

任意网页：

```
var f1 = document.createElement('iframe')
document.body.appendChild(f1)
f1.src="不同源"
f1.contentDocument
```

上面操作后会报错

```
f1.src="同源"
f1.contentDocument
```

如上操作后就可以正常查看iframe内容。