爬虫简介

为什么要做网络爬虫?

首先请问：都说现在是"大数据时代"，那数据从何而来?

- `企业产生的用户数据`：百度指数、阿里指数、TBI腾讯浏览指数、新浪微博指数
- `数据平台购买数据`：数据堂、国云数据市场、贵阳大数据交易所
- `政府/机构公开的数据`：中华人民共和国国家统计局数据、世界银行公开数据、联合国数据、纳斯达克。
- `数据管理咨询公司`：麦肯锡、埃森哲、艾瑞咨询
- `爬取网络数据`：如果需要的数据市场上没有，或者不愿意购买，那么可以选择招/做一名爬虫工程师，自己动手丰衣足食。拉勾网Python爬虫职位

网络爬虫是什么?

百度百科：网络爬虫

关于Python网络爬虫，我们需要学习的有：

1. Python基础语法学习（基础知识）

2. 对HTML页面的内容抓取（数据抓取）

3. 对HTML页面的数据提取（数据提取）

4. Scrapy框架（第三方框架）

5. 爬虫(Spider)、反爬虫(Anti-Spider)、反反爬虫(Anti-Anti-Spider)之间的斗争...

通用爬虫和聚焦爬虫

根据使用场景，网络爬虫可分为**通用爬虫**和**聚焦爬虫**两种.

通用爬虫

通用网络爬虫 是 搜索引擎抓取系统（Baidu、Google、Yahoo等）的重要组成部分。主要目的是将互联网上的网页下载到本地，形成一个互联网内容的镜像备份。

通用搜索引擎（Search Engine）工作原理

通用网络爬虫 从互联网中搜集网页，采集信息，这些网页信息用于为搜索引擎建立索引从而提供支持，它决定着整个引擎系统的内容是否丰富，信息是否即时，因此其性能的优劣直接影响着搜索引擎的效果。

第一步：抓取网页

搜索引擎网络爬虫的基本工作流程如下：

1. 首先选取一部分的种子URL，将这些URL放入待抓取URL队列；
2. 取出待抓取URL，解析DNS得到主机的IP，并将URL对应的网页下载下来，存储进已下载网页库中，并且将这些URL放进已抓取URL队列。
3. 分析已抓取URL队列中的URL，分析其中的其他URL，并且将URL放入待抓取URL队列，从而进入下一个循环....

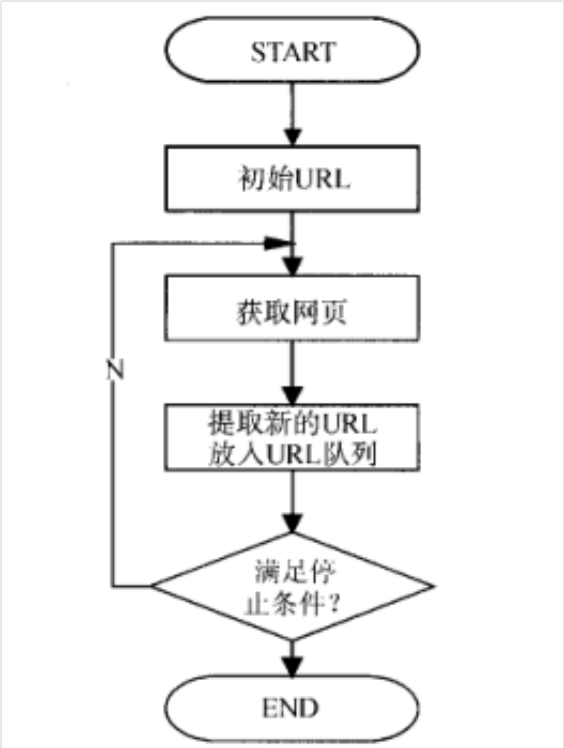


图 1　通用网络爬虫工作流程图

搜索引擎如何获取一个新网站的URL：

1. 新网站向搜索引擎主动提交网址：（如百度http://zhanzhang.baidu.com/linksubmit/url）

2. 在其他网站上设置新网站外链（尽可能处于搜索引擎爬虫爬取范围）

3. 搜索引擎和DNS解析服务商(如DNSPod等）合作，新网站域名将被迅速抓取。

但是搜索引擎蜘蛛的爬行是被输入了一定的规则的，它需要遵从一些命令或文件的内容，如标注为`nofollow`的链接，或者是`Robots`协议。

> Robots协议（也叫爬虫协议、机器人协议等），全称是“网络爬虫排除标准”（Robots Exclusion Protocol），网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取，例如：
>
> 淘宝网：https://www.taobao.com/robots.txt
>
> 腾讯网：http://www.qq.com/robots.txt

第二步：数据存储

搜索引擎通过爬虫爬取到的网页，将数据存入原始页面数据库。其中的页面数据与用户浏览器得到的HTML是完全一样的。

搜索引擎蜘蛛在抓取页面时，也做一定的重复内容检测，一旦遇到访问权重很低的网站上有大量抄袭、采集或者复制的内容，很可能就不再爬行。

第三步：预处理

搜索引擎将爬虫抓取回来的页面，进行各种步骤的预处理。

- 提取文字
- 中文分词
- 消除噪音（比如版权声明文字、导航条、广告等……）
- 索引处理
- 链接关系计算
- 特殊文件处理
- ……

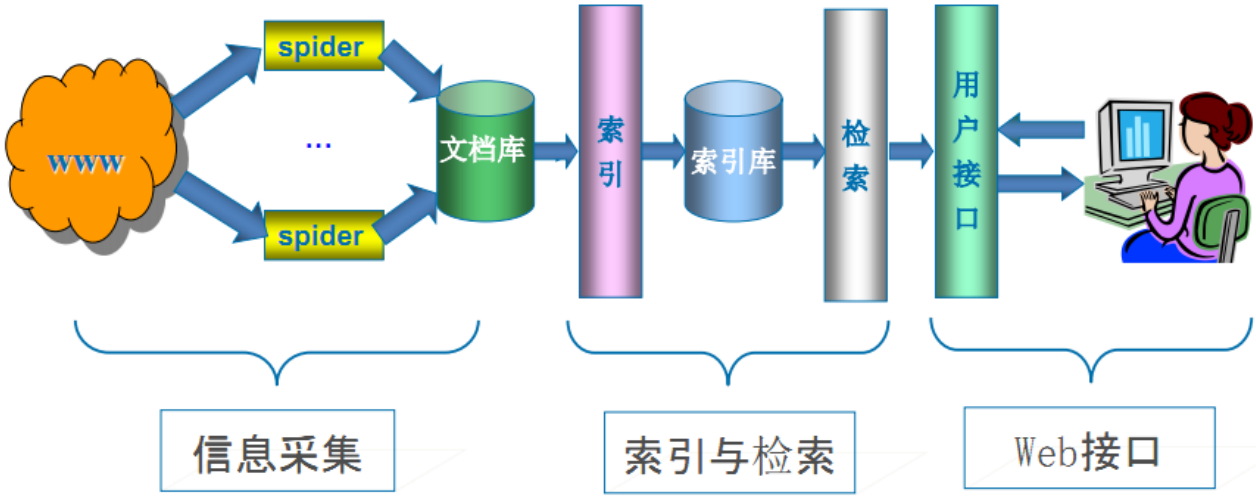
除了HTML文件外，搜索引擎通常还能抓取和索引以文字为基础的多种文件类型，如 PDF、Word、WPS、XLS、PPT、TXT 文件等。我们在搜索结果中也经常会看到这些文件类型。

但搜索引擎还不能处理图片、视频、Flash 这类非文字内容，也不能执行脚本和程序。

第四步：提供检索服务，网站排名

搜索引擎在对信息进行组织和处理后，为用户提供关键字检索服务，将用户检索相关的信息展示给用户。

同时会根据页面的PageRank值（链接的访问量排名）来进行网站排名，这样Rank值高的网站在搜索结果中会排名较前，当然也可以直接使用 Money 购买搜索引擎网站排名，简单粗暴。



课外阅读：[Google搜索引擎的工作原理]

但是，这些通用性搜索引擎也存在着一定的局限性：

1. 通用搜索引擎所返回的结果都是网页，而大多情况下，网页里90%的内容对用户来说都是无用的。
2. 不同领域、不同背景的用户往往具有不同的检索目的和需求，搜索引擎无法提供针对具体某个用户的搜索结果。
3. 万维网数据形式的丰富和网络技术的不断发展，图片、数据库、音频、视频多媒体等不同数据大量出现，通用搜索引擎对这些文件无能为力，不能很好地发现和获取。
4. 通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询，无法准确理解用户的具体需求。

针对这些情况，聚焦爬虫技术得以广泛使用。

聚焦爬虫

聚焦爬虫，是"面向特定主题需求"的一种网络爬虫程序，它与通用搜索引擎爬虫的区别在于： **聚焦爬虫在实施网页抓取时会对内容进行处理筛选，尽量保证只抓取与需求相关的网页信息。**

而我们今后要学习的网络爬虫，就是聚焦爬虫。

HTTP/HTTPS的请求与响应

HTTP与HTTPS

`HTTP协议`（HyperText Transfer Protocol，超文本传输协议）：是一种发布和接收 HTML页面的方法。

`HTTPS`（Hypertext Transfer Protocol over Secure Socket Layer）简单讲是HTTP的安全版，在HTTP下加入SSL层。

`SSL`（Secure Sockets Layer 安全套接层）主要用于Web的安全传输协议，在传输层对网络连接进行加密，保障在Internet上数据传输的安全。

- `HTTP`的端口号为`80`，
- `HTTPS`的端口号为`443`

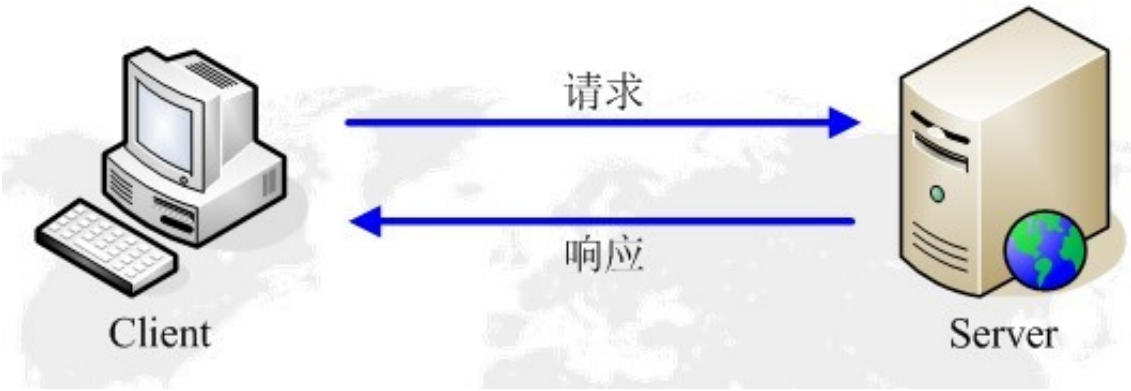
HTTP工作原理

网络爬虫抓取过程可以理解为`模拟浏览器操作的过程`。

浏览器的主要功能是向服务器发出请求，在浏览器窗口中展示您选择的网络资源，HTTP是一套计算机通过网络进行通信的规则。

HTTP的请求与响应

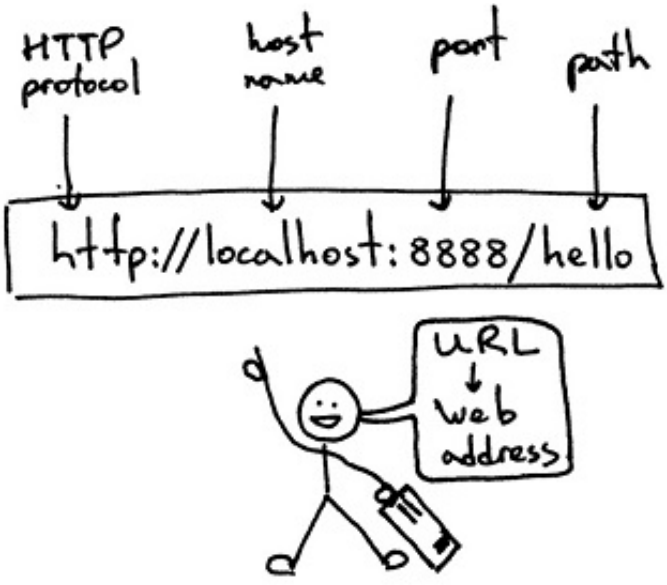
HTTP通信由两部分组成： 客户端请求消息 与 服务器响应消息



浏览器发送HTTP请求的过程：

1. 当用户在浏览器的地址栏中输入一个URL并按回车键之后，浏览器会向HTTP服务器发送HTTP请求。HTTP请求主要分为“Get”和“Post”两种方法。
2. 当我们在浏览器输入URL http://www.baidu.com 的时候，浏览器发送一个Request请求去获取 http://www.baidu.com 的html文件，服务器把Response文件对象发送回给浏览器。
3. 浏览器分析Response中的 HTML，发现其中引用了很多其他文件，比如Images文件，CSS文件，JS文件。 浏览器会自动再次发送Request去获取图片，CSS文件，或者JS文件。
4. 当所有的文件都下载成功后，网页会根据HTML语法结构，完整的显示出来了。

URL（Uniform / Universal Resource Locator的缩写）：统一资源定位符，是用于完整地描述Internet上网页和其他资源的地址的一种标识方法。



基本格式：`scheme://host[:port#]/path/…/[?query-string][#anchor]`

- scheme：协议(例如：http, https, ftp)
- host：服务器的IP地址或者域名
- port：服务器的端口（如果是走协议默认端口，缺省端口80）
- path：访问资源的路径
- query-string：参数，发送给http服务器的数据

127.0.0.1:8000?a=1&b=2

- anchor：锚（跳转到网页的指定锚点位置）

例如：

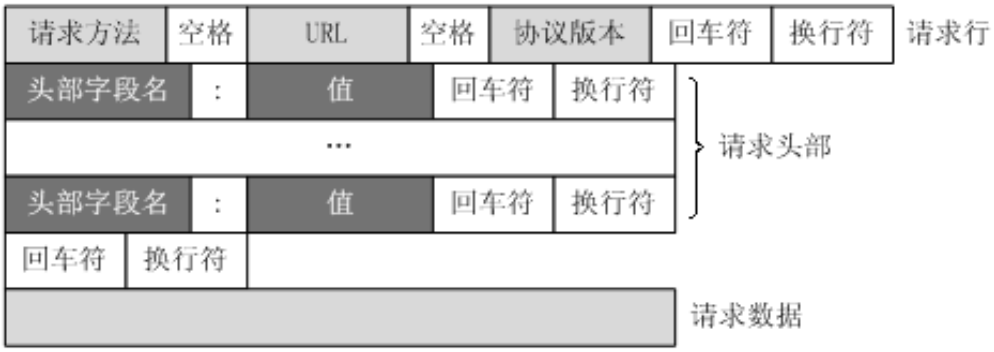
- ftp://192.168.18.18:8080/index
- http://www.baidu.com
- http://item.jd.com/11936238.html#product-detail

客户端HTTP请求

URL只是标识资源的位置，而HTTP是用来提交和获取资源。客户端发送一个HTTP请求到服务器的请求消息，包括以下格式：

```
`请求行`、`请求头部`、`空行`、`请求数据`
```

四个部分组成，下图给出了请求报文的一般格式。



一个典型的HTTP请求示例

```
GET https://www.baidu.com/ HTTP/1.1
Host: www.baidu.com
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/54.0.2840.99 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://www.baidu.com/
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: BAIDUID=04E4001F34EA74AD4601512DD3C41A7B:FG=1; BIDUPSID=04E4001F34EA74AD4601512DD3C41A7B; PSTM=1470329258; MCITY=-343%3A340%3A; BDUSS=nF0MVFiMTVLcUh-Q2MxQ0M3STZGQUZ4N2hBa1FFRkIzUDI3QlBCZjg5cFdOd1pZQVFBQUFBJCQAAAAAAAAAAAEAAADpLvgG0KGyvLrcyfrG-AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFaq3ldWqt5XN; H_PS_PSSID=1447_18240_21105_21386_21454_21409_21554; BD_UPN=12314753; sug=3; sugstore=0; ORIGIN=0; bdime=0; H_PS_645EC=7e2ad3QHl181NSPbFbd7PRUCE1LlufzxrcFmwYin0E6b%2BW8bbTMKHZbDP0g; BDSVRTM=0
```

请求方法

```
GET https://www.baidu.com/ HTTP/1.1
```

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP 0.9：只有基本的文本 GET 功能。

HTTP 1.0：完善的请求/响应模型，并将协议补充完整，定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP 1.1：在 1.0 基础上进行更新，新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

HTTP 2.0（未普及）：请求/响应首部的定义基本没有改变，只是所有首部键必须全部小写，而且请求行要独立为
:method、:scheme、:host、:path这些键值对。

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

HTTP请求主要分为`Get`和`Post`两种方法

- GET是从服务器上获取数据，POST是向服务器传送数据
- GET请求参数显示，都显示在浏览器网址上，HTTP服务器根据该请求所包含URL中的参数来产生响应内容，即“Get”请求的参数是URL的一部分。 例如： http://www.baidu.com/s?wd=Chinese

- POST请求参数在请求体当中，消息长度没有限制而且以隐式的方式进行发送，通常用来向HTTP服务器提交量比较大的数据（比如请求中包含许多参数或者文件上传操作等），请求的参数包含在“Content-Type”消息头里，指明该消息体的媒体类型和编码，

注意：避免使用Get方式提交表单，因为有可能会导致安全问题。 比如说在登陆表单中用Get方式，用户输入的用户名和密码将在地址栏中暴露无遗。

常用的请求报头

1. Host (主机和端口号)

Host：对应网址URL中的Web名称和端口号，用于指定被请求资源的Internet主机和端口号，通常属于URL的一部分。

2. Connection (链接类型)

Connection：表示客户端与服务连接类型

1. Client 发起一个包含 Connection:keep-alive 的请求，HTTP/1.1使用 keep-alive 为默认值。
2. Server收到请求后：
    - 如果 Server 支持 keep-alive，回复一个包含 Connection:keep-alive 的响应，不关闭连接；
    - 如果 Server 不支持 keep-alive，回复一个包含 Connection:close 的响应，关闭连接。
3. 如果client收到包含 Connection:keep-alive 的响应，向同一个连接发送下一个请求，直到一方主动关闭连接。

keep-alive在很多情况下能够重用连接，减少资源消耗，缩短响应时间，比如当浏览器需要多个文件时(比如一个HTML文件和相关的图形文件)，不需要每次都去请求建立连接。

3. Upgrade-Insecure-Requests (升级为HTTPS请求)

Upgrade-Insecure-Requests：升级不安全的请求，意思是会在加载 http 资源时自动替换成 https 请求，让浏览器不再显示https页面中的http请求警报。

HTTPS 是以安全为目标的 HTTP 通道，所以在 HTTPS 承载的页面上不允许出现 HTTP 请求，一旦出现就是提示或报错。

4. User-Agent (浏览器名称)

User-Agent：是客户浏览器的名称，以后会详细讲。

5. Accept (传输文件类型)

Accept：指浏览器或其他客户端可以接受的MIME（Multipurpose Internet Mail Extensions（多用途互联网邮件扩展））文件类型，服务器可以根据它判断并返回适当的文件格式。

举例：

`Accept: */*`：表示什么都可以接收。

`Accept: image/gif`：表明客户端希望接受GIF图像格式的资源；

`Accept: text/html`：表明客户端希望接受html文本。

`Accept: text/html, application/xhtml+xml;q=0.9, image/*;q=0.8`：表示浏览器支持的 MIME 类型分别是 html文本、xhtml和xml文档、所有的图像格式资源。

q是权重系数，范围 0 =< q <= 1，q 值越大，请求越倾向于获得其“;”之前的类型表示的内容。若没有指定q值，则默认为1，按从左到右排序顺序；若被赋值为0，则用于表示浏览器不接受此内容类型。

Text：用于标准化地表示的文本信息，文本消息可以是多种字符集和或者多种格式的；Application：用于传输应用程序数据或者二进制数据。详细请点击

6. Referer (页面跳转处)

Referer：表明产生请求的网页来自于哪个URL，用户是从该 Referer页面访问到当前请求的页面。这个属性可以用来跟踪Web请求来自哪个页面，是从什么网站来的等。

有时候遇到下载某网站图片，需要对应的referer，否则无法下载图片，那是因为人家做了防盗链，原理就是根据referer去判断是否是本网站的地址，如果不是，则拒绝，如果是，就可以下载；

7. Accept-Encoding（文件编解码格式）

Accept-Encoding：指出浏览器可以接受的编码方式。编码方式不同于文件格式，它是为了压缩文件并加速文件传递速度。浏览器在接收到Web响应之后先解码，然后再检查文件格式，许多情形下这可以减少大量的下载时间。

举例：Accept-Encoding:gzip;q=1.0, identity; q=0.5, *;q=0

如果有多个Encoding同时匹配, 按照q值顺序排列，本例中按顺序支持gzip, identity压缩编码，支持gzip的浏览器会返回经过gzip编码的HTML页面。 **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。**

8. Accept-Language（语言种类）

Accept-Langeuage：指出浏览器可以接受的语言种类，如en或en-us指英语，zh或者zh-cn指中文，当服务器能够提供一种以上的语言版本时要用到。

9. Accept-Charset（字符编码）

Accept-Charset：指出浏览器可以接受的字符编码。

举例：Accept-Charset:iso-8859-1,gb2312,utf-8

- ISO8859-1：通常叫做Latin-1。Latin-1包括了书写所有西方欧洲语言不可缺少的附加字符，英文浏览器的默认值是ISO-8859-1.
- gb2312：标准简体中文字符集;
- utf-8：UNICODE 的一种变长字符编码，可以解决多种语言文本显示问题，从而实现应用国际化和本地化。

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。

10. Cookie （Cookie）

Cookie：浏览器用这个属性向服务器发送Cookie。Cookie是在浏览器中寄存的小型数据体，它可以记载和服务器相关的用户信息，也可以用来实现会话功能，以后会详细讲。

11. Content-Type (POST数据类型)

Content-Type：POST请求里用来表示的内容类型。

举例：Content-Type = Text/XML; charset=gb2312：

指明该请求的消息体中包含的是纯文本的XML类型的数据，字符编码采用“gb2312”。

服务端HTTP响应

HTTP响应也由四个部分组成，分别是： `状态行`、`消息报头`、`空行`、`响应正文`

```
HTTP/1.1 200 OK
Server: Tengine
Connection: keep-alive
Date: Wed, 30 Nov 2016 07:58:21 GMT
Cache-Control: no-cache
Content-Type: text/html;charset=UTF-8
Keep-Alive: timeout=20
Vary: Accept-Encoding
Pragma: no-cache
X-NWS-LOG-UUID: bd27210a-24e5-4740-8f6c-25dbafa9c395
Content-Length: 180945

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" ....
```

常用的响应报头(了解)

理论上所有的响应头信息都应该是回应请求头的。但是服务端为了效率，安全，还有其他方面的考虑，会添加相对应的响应头信息，从上图可以看到：

1. Cache-Control：must-revalidate, no-cache, private。

这个值告诉客户端，服务端不希望客户端缓存资源，在下次请求资源时，必须要从新请求服务器，不能从缓存副本中获取资源。

- Cache-Control是响应头中很重要的信息，当客户端请求头中包含Cache-Control:max-age=0请求，明确表示不会缓存服务器资源时,Cache-Control作为作为回应信息，通常会返回no-cache，意思就是说，"那就不缓存呗"。
- 当客户端在请求头中没有包含Cache-Control时，服务端往往会定,不同的资源不同的缓存策略，比如说oschina在缓存图片资源的策略就是Cache-Control：max-age=86400,这个意思是，从当前时间开始，在86400秒的时间内，客户端可以直接从缓存副本中读取资源，而不需要

向服务器请求。

2. Connection：keep-alive

这个字段作为回应客户端的Connection：keep-alive，告诉客户端服务器的tcp连接也是一个长连接，客户端可以继续使用这个tcp连接发送http请求。

3. Content-Encoding:gzip

告诉客户端，服务端发送的资源是采用gzip编码的，客户端看到这个信息后，应该采用gzip对资源进行解码。

4. Content-Type：text/html;charset=UTF-8

告诉客户端，资源文件的类型，还有字符编码，客户端通过utf-8对资源进行解码，然后对资源进行html解析。通常我们会看到有些网站是乱码的，往往就是服务器端没有返回正确的编码。

5. Date：Sun, 21 Sep 2016 06:18:21 GMT

这个是服务端发送资源时的服务器时间，GMT是格林尼治所在地的标准时间。http协议中发送的时间都是GMT的，这主要是解决在互联网上，不同时区在相互请求资源的时候，时间混乱问题。

6. Expires:Sun, 1 Jan 2000 01:00:00 GMT

这个响应头也是跟缓存有关的，告诉客户端在这个时间前，可以直接访问缓存副本，很显然这个值会存在问题，因为客户端和服务器的时间不一定会都是相同的，如果时间不同就会导致问题。所以这个响应头是没有Cache-Control：max-age=*这个响应头准确的，因为max-age=date中的date是个相对时间，不仅更好理解，也更准确。

7. Pragma:no-cache

这个含义与Cache-Control等同。

8.Server：Tengine/1.4.6

这个是服务器和相对应的版本，只是告诉客户端服务器的信息。

9. Transfer-Encoding：chunked

这个响应头告诉客户端，服务器发送的资源的方式是分块发送的。一般分块发送的资源都是服务器动态生成的，在发送时还不知道发送资源的大小，所以采用分块发送，每一块都是独立的，独立的块都能标示自己的长度，最后一块是0长度的，当客户端读到这个0长度的块时，就可以确定资源已经传输完了。

10. Vary: Accept-Encoding

告诉缓存服务器，缓存压缩文件和非压缩文件两个版本，现在这个字段用处并不大，因为现在的浏览器都是支持压缩的。

响应状态码

响应状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值。

常见状态码：

- 100~199：表示服务器成功接收部分请求，要求客户端继续提交其余请求才能完成整个处理过程。
- 200~299：表示服务器成功接收请求并已完成整个处理过程。常用200（OK 请求成功）。
- 300~399：为完成请求，客户需进一步细化请求。例如：请求的资源已经移动一个新地址、常用302（所请求的页面已经临时转移至新的url）、307和304（使用缓存资源）。
- 400~499：客户端的请求有错误，常用404（服务器无法找到被请求的页面）、403（服务器拒绝访问，权限不够）。

- 500~599：服务器端出现错误，常用500（请求未完成。服务器遇到不可预知的情况）。

上下文管理器

回顾一下，之前我们学习的文件操作，一般都是三步:

1) 打开文件； 2) 进行读或写操作； 3）当2)完成后，关闭文件。

```
f = open("file_name.txt", "w")
f.write(content)
# 或者
# f.read()
f.close()
```

问题来了：在文件读写的时候，因为各种原因(譬如，计算机内存不足)，是不一定会成功的，即上面的程序是存在漏洞的，文件读写失败，但是却没关闭文件，极可能造成数据丢失。

为此，python引入了异常机制，解决方案如下:

```
try:
    f = open("file_name.txt", "w")
    f.write(content)
    # 或者
    # f.read()
finally:
    f.close()
```

问题是解决了，但是，代码变得复杂了，既不美观，操作也变得繁琐了，为此，python引入了**上下文管理器**即保证了IO操作的安全性(自动关闭资源)，又能简化操作。

```python
with open("file_name.txt", "w") as f:
    f.write(content)
    # 或者
    # f.read()
```

爬虫、反爬、反反爬

爬虫: 使用代码模拟浏览器向服务器发起请求，获取响应。