中国科学院大学
University of Chinese Academy of Sciences

计算机与控制学院
School of Computer and Control Engineering

移动互联网技术

第三章　互联网数据获取技术

简单爬虫（一）

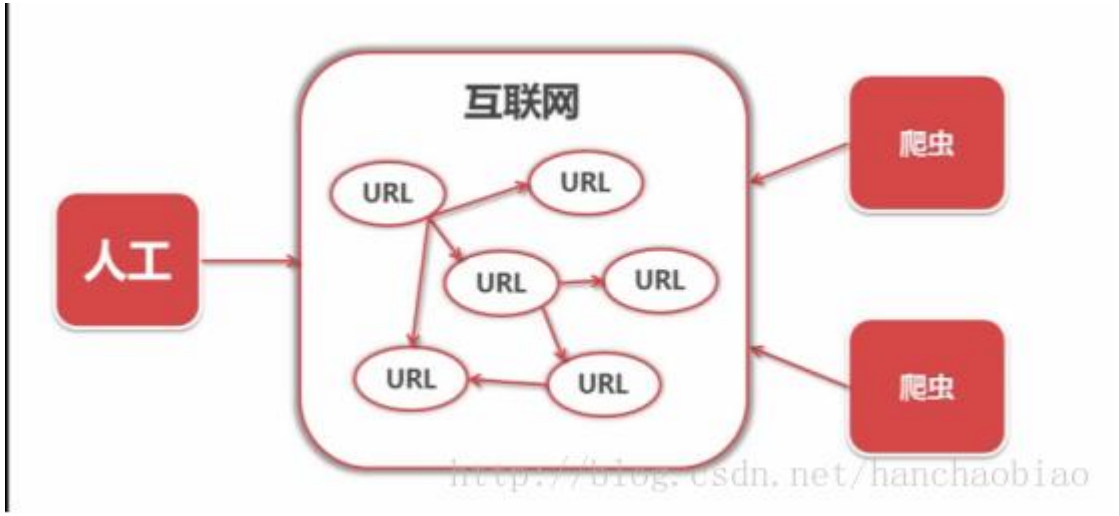
王文杰
wangwj@ucas.ac.cn

本节主要内容

- 爬虫概述
- 简单爬虫框架
- 正则表达式
- URLLib
- 爬虫实例

什么是网络爬虫

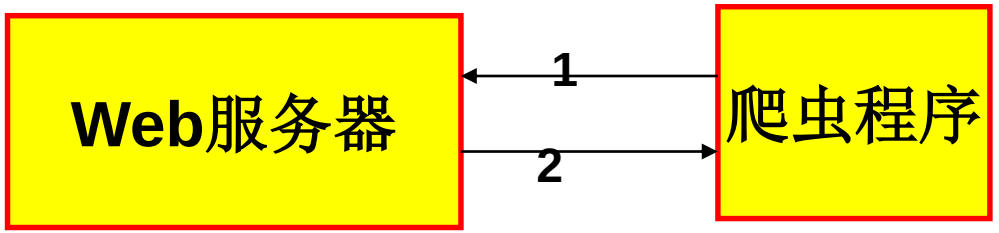
- 网络爬虫(Crawler)，也称为网络蜘蛛(Spider)或网络机器人(Robot)
- 网络爬虫是请求网站并提取数据的自动程序：
  - 它能够自动建立到WEB服务器的网络连接，访问服务器上的某个Web页面或网络资源，获得其内容，并按照页面上的超链接进行更多页面的获取
  - 它可为搜索引擎从Internet网上下载网页，是搜索引擎的重要组成部分。
  - 爬虫本质上是模拟人的浏览点击行为。



网络爬虫如何下载整个网站？

- 基本原理上说，是基于两个步骤：

（1）基本的单页面获取



1 连接指定服务器IP地址和端口，如80、8080；发送URL，即页面或某个文档

2 由服务器执行文件(URL指定)，返回响应数据给爬虫程序

（2）根据超链接，获得整个网站、多个网站上的所有页面

- 假定从一门户网站的首页出发，先下载这个网页，然后通过分析这个网页，可以找到该页面中的所有超链接，也就等于知道了这家门户网站首页所直接连接的全部网页，
- 接下来访问、下载并分析门户网站各个网页，又能找到其他相连的网页。
- 让程序不停地做下去，就能下载整个的互联网。

网络爬虫基本类型

- 通用网络爬虫：
- 聚焦网络爬虫：主题爬虫
- 增量网络爬虫：爬取新增的网络内容
- 深层网络爬虫：直接无法访问的页面可称为深层页面，如需要权限认证的页面。爬取如此这般的爬虫称为深层网络爬虫

传统爬虫的基本流程

- **发起请求**

  通过HTTP库向目标站点发起请求，也就是发送一个Request，请求可以包含额外的header等信息，等待服务器响应

- **获取响应内容**

  如果服务器能正常响应，会得到一个Response，Response的内容便是所要获取的页面内容，类型可能是HTML、Json字符串、二进制数据（图片或者视频）等类型

- **解析内容**

  得到的内容可能是HTML，可以用正则表达式，页面解析库进行解析；可能是Json,可以直接转换为Json对象解析；可能是二进制数据，可以做保存或者进一步的处理

- **保存数据**

  保存形式多样，可以存为文本，也可以保存到数据库，或者保存特定格式的文件

本节主要内容

- 爬虫概述
- 简单爬虫框架
- 正则表达式
- URLLib
- 爬虫实例

简单爬虫框架

- **爬虫调度端：**
  - 用来启动、执行、停止爬虫，或者监视爬虫中的运行情况
- **模块URL管理器：**
  - 对将要爬取的URL和已经爬取过的URL这两个数据的管理
- **网页下载器：**
  - 将URL管理器里提供的一个URL对应的网页下载下来，存储为一个字符串，这个字符串会传送给网页解析器进行解析
- **网页解析器：**
  - 一方面会解析出有价值的数据，另一方面，由于每一个页面都有很多指向其它页面的网页，这些URL被解析出来之后，可以补充进URL管理器

URL管理器

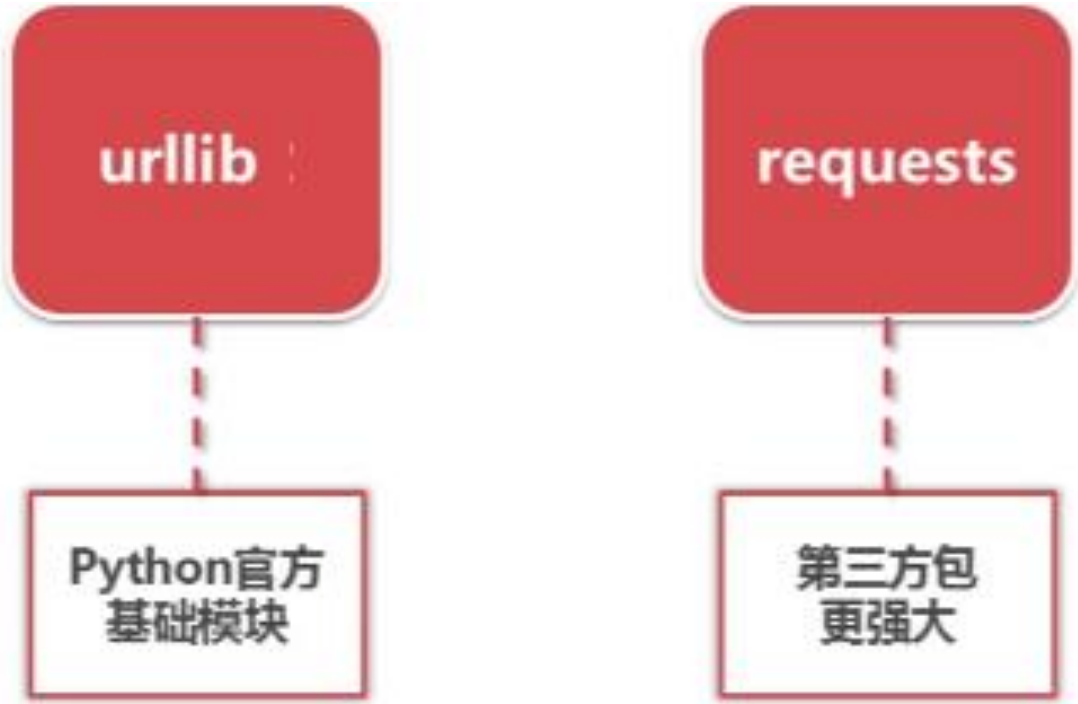
- 管理爬取的URL集合和已经爬取过的URL集合
  - 防止重复抓取和循环抓取

URL管理器

- 实现方式
  - Python内存set集合：set集合支持去重的作用
  - Mysql：url（访问路径）is_crawled(是否访问)
  - Redis/MongoDB:使用Redis性能最好，且Redis中也有set类型，可以去重

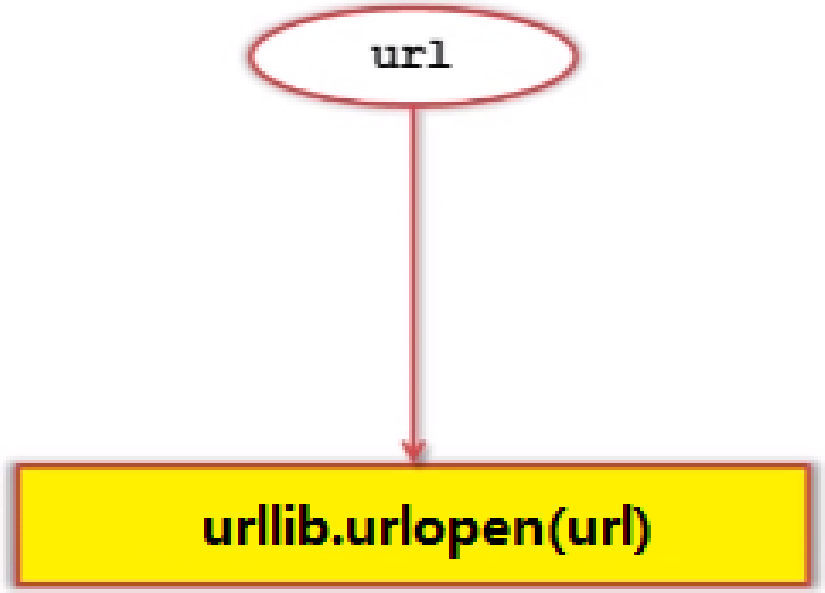
网页下载器

- 将互联网上URL对应的网页下载到本地的工具
- 实现方式：



网页下载器—UrlLib

- 最简单的一种下载方法




```
from urllib import request

#直接请求
response1 = request.urlopen(“http://www.baidu.com”)

#获取状态码，如果是200表示成功
print("状态码:",response1.getcode())

#获取网页内容
cont = response1.read()

#设置编码格式
print(cont.decode("utf8"))
```

Request和Response

- 浏览器发送消息给网址所在的服务器，这个过程就叫做HTPP Request
- 服务器收到浏览器发送的消息后，能够根据浏览器发送消息的内容，做相应的处理，然后把消息回传给浏览器，这个过程就是HTTP Response
- 浏览器收到服务器的Response信息后，会对信息进行相应的处理，然后展示

- Request中包含什么？
  - 请求方式
  - 请求URL
  - 请求头
  - 请求体：请求是携带的数据

- Response中包含了什么
  - 响应状态
  - 响应头
  - 响应体：包含请求资源的内容，如网页HTMl,图片，二进制数据等

Request部分的HTTP Header

- Accept: text/plain
- Accept-Charset: utf-8
- Accept-Encoding: gzip, deflate
- Accept-Language: en-US
- Connection: keep-alive
- Content-Length: 348
- Content-Type: application/x-www-form-urlencoded
- Date: Tue, 15 Nov 1994 08:12:31 GMT
- Host: en.wikipedia.org:80
- User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0
- Cookie: $Version=1; Skin=new;

Request--请求方式

- 主要有：
  - GET/POST两种类型常用，另外还有HEAD/PUT/DELETE/OPTIONS
  - GET和POST的区别就是：请求的数据GET是在url中，POST则是存放在头部
- GET是从服务器上获取信息，而且是安全的和幂等的
  - 这里所谓安全的意味着该操作用于获取信息而非修改信息。换句话说，GET 请求一般不应产生副作用。就是说，它仅仅是获取资源信息，就像数据库查询一样，不会修改，增加数据，不会影响资源的状态。
  - 幂等的意味着对同一URL的多个请求应该返回同样的结果。
- POST是向服务器传送数据，可用于更新站点
- 可通过登录页面，查看get/post。登录后，找到login进程，里面有post（通过输入错误密码，可以看到该进程）

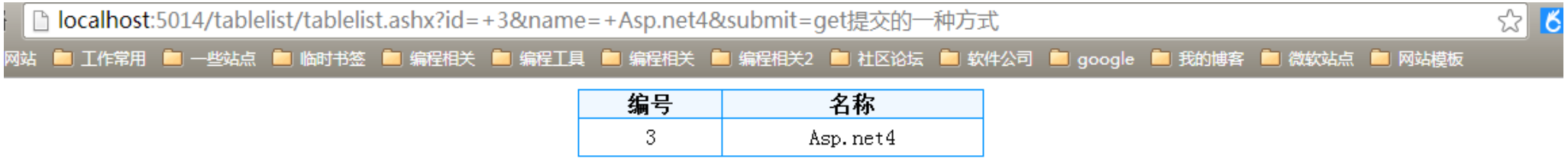
Request--请求方式

- HTTP请求方式中get和post的主要区别

1. GET请求：请求的数据会附加在URL之后，以?分割URL和传输数据，多个参数用&连接。URL的编码格式采用的是ASCII编码，而不是uniclde，即是说所有的非ASCII字符都要编码之后再传输。POST请求：POST请求会把请求的数据放置在HTTP请求包的包体中，需要构造一个表单，通过表单提交得到POST请求。GET请求的数据会暴露在地址栏中，而POST请求则不会。
2. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。

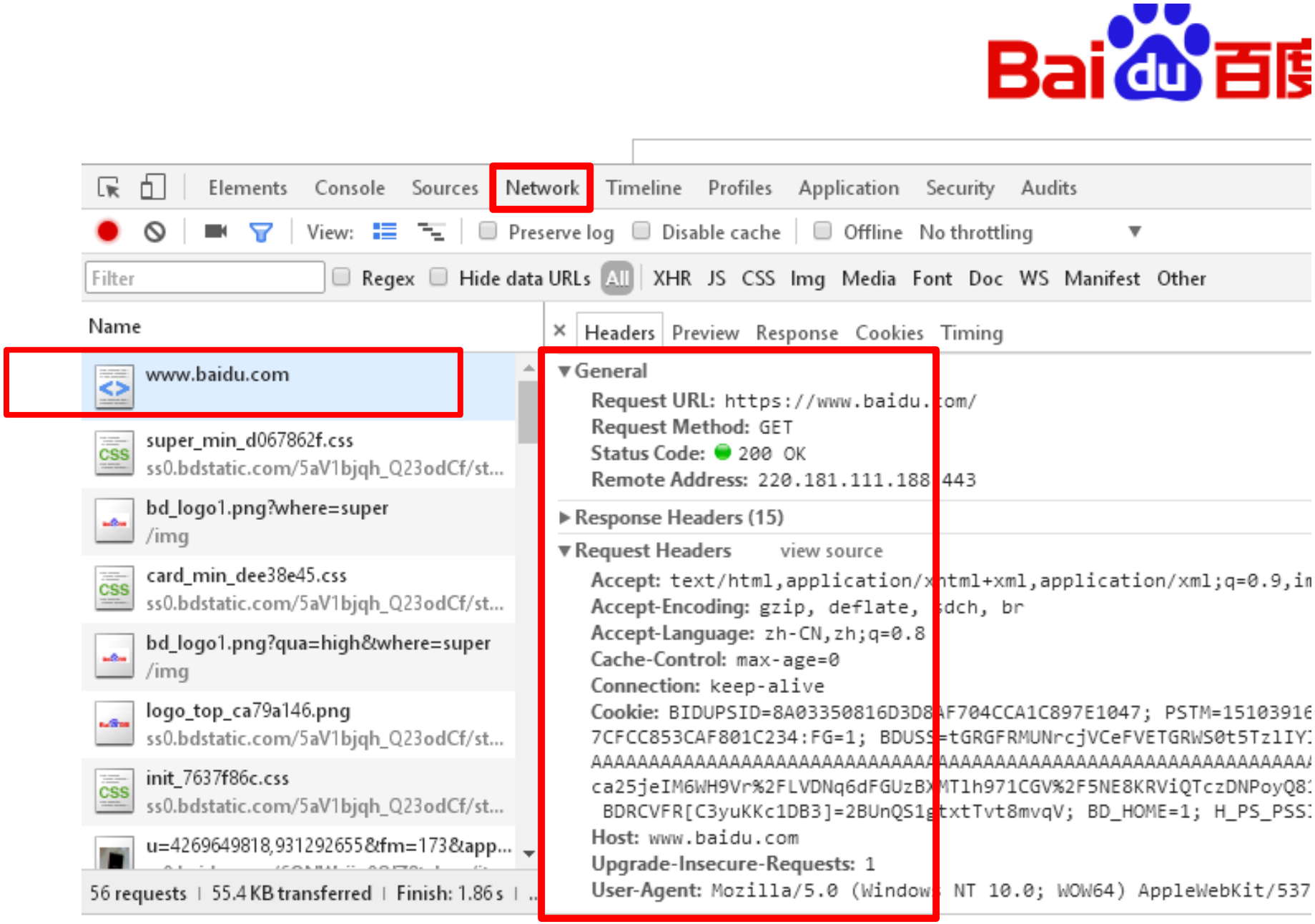
Request--请求方式

3. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。



Request--请求头

- 包含请求时的头部信息，如User-Agent,Host,Cookies等信息
- 下图是请求请求百度时，所有的请求头部信息参数



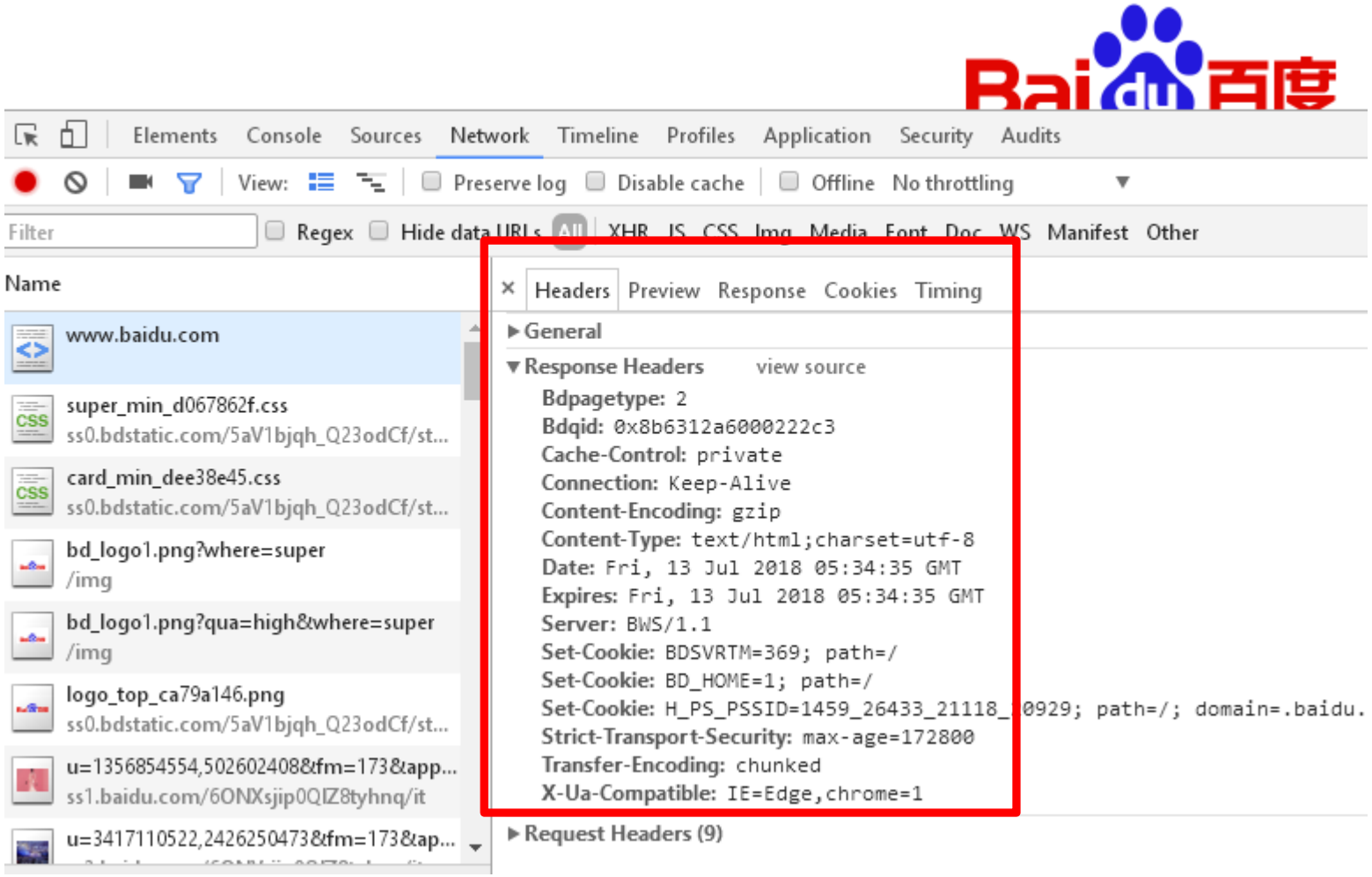
User Agent是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等。

Response—HTTP Header

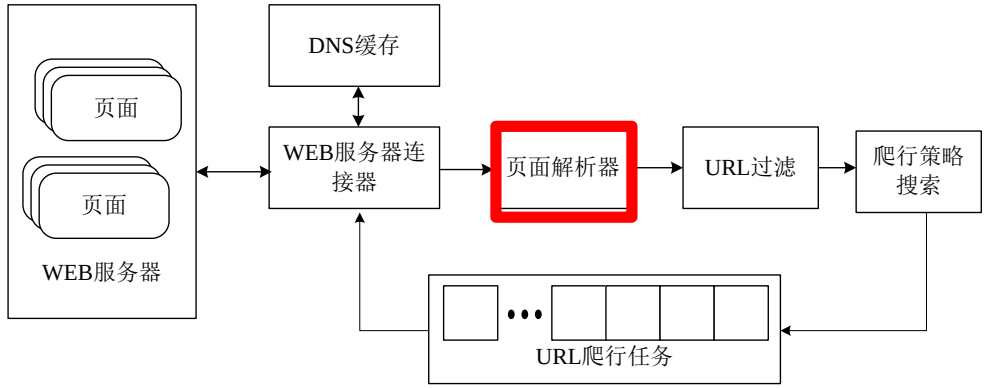
- Accept-Patch: text/example;charset=utf-8
- Cache-Control: max-age=3600
- Content-Encoding: gzip
- Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
- Content-Language: da
- Content-Length: 348
- ETag: "737060cd8c284d8af7ad3082f209582d"
- Expires: Thu, 01 Dec 1994 16:00:00 GMT
- Location: http://www.w3.org/pub/WWW/People.html
- Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1
- Status: 200 OK

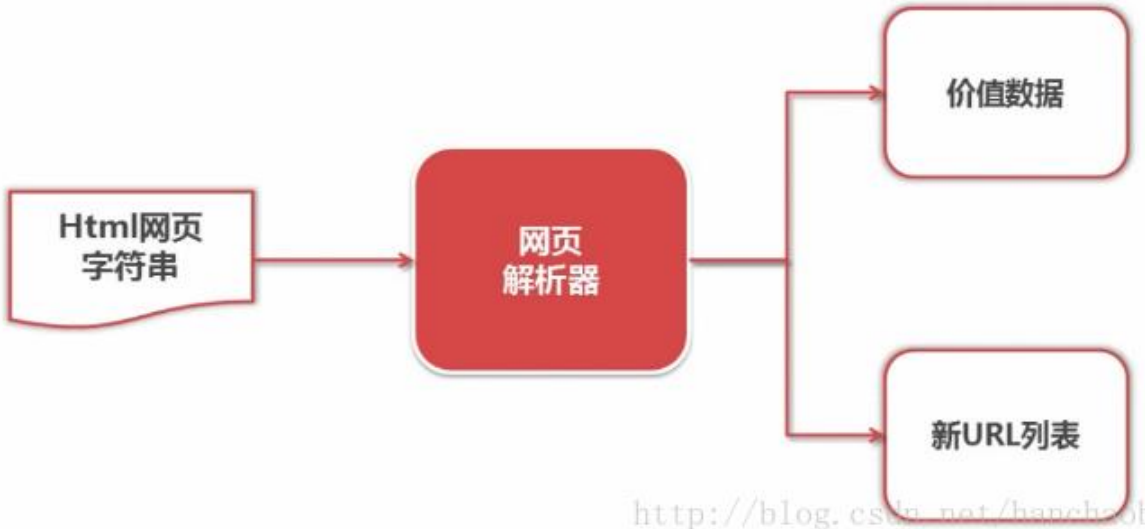
Response--响应头

- 如内容类型，类型的长度，服务器信息，设置Cookie



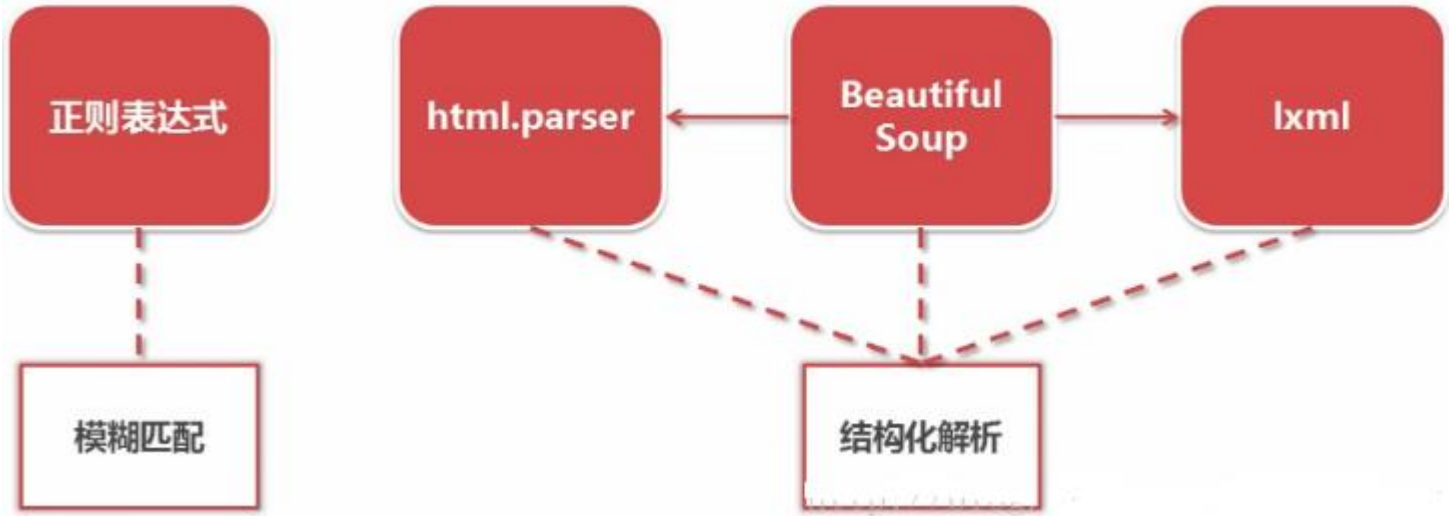
- **页面解析器**
  - 能爬取什么样的数据，即从网页中提取有价值数据的工具
- **网页中数据包括：**
  - 网页文本：如HTML文档，Json格式化文本（AJAX请求的返回）等
  - 图片：获取到的是二进制文件，保存为图片格式
  - 视频：同样是二进制文件
  - 其他：只要请求到的，都可以获取



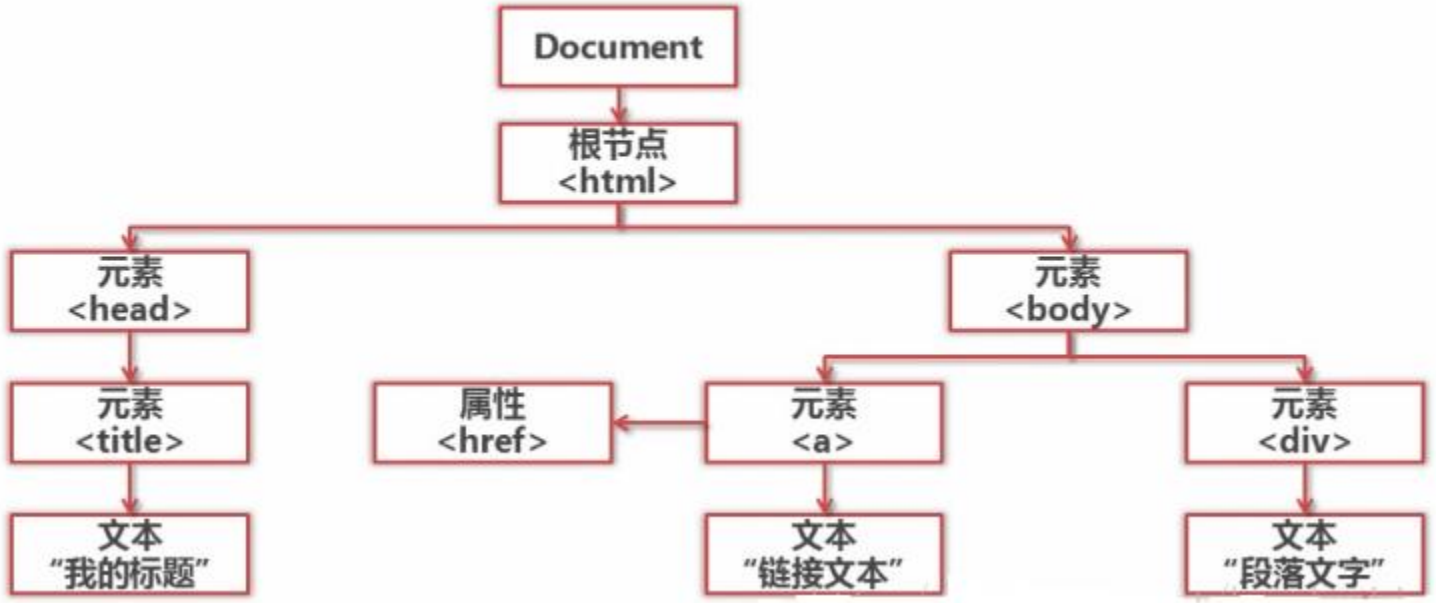


- 如何解析数据？
  - 直接处理：网页比较简单
  - Json解析：ajax
  - 正则表达式处理
  - Html.parser：python自带
  - BeautifulSoup：第三方
  - Lxml：第三方
  - PyQuery解析处理
  - XPath解析处理
- 结构化解析：

  将整个网页文档下载成一颗DOM树，以树的形式对其中元素进行遍历和访问



- 结构化解析-DOM（Document Object Model)树



本节主要内容

- 爬虫概述
- 简单爬虫框架
- 正则表达式
- URLLib
- 爬虫实例

正则表达式

- 什么是正则表达式？
  - 互联网上信息非常多，而我们关注的信息有限。假如我们希望只提取出关 注的数据，此时可以通过一些表达式进行提取，正则表达式就是其中一 种进行数据筛选的表达式。
  - 正则表达式，又称规则表达式（Regular Expression，在代码中常简写为regex、regexp或RE），计算机科学的一个概念。

正则表达式

- 原子：
  - 原子是正则表达式中最基本的组成单位，每个正则表达式中至少要包含 一个原子。
  - 常见的原子类型有：
    - a普通字符作为原子
    - b非打印字符作为原子，如:\n，\t
    - c通用字符作为原子
    - d原子表

原义字符

- 特殊字符转义：
  - 在表达式中用到的一些元字符不再表示原来的字面意义，如果要匹配这些有特殊意义的元字符，必须使用“\”将这些字符转义为原义字符
  - “\”的作用是将下一个字符标记为特殊字符、或一个向后引用、或一个八进制转义符。例如，“\\n”匹配\n。“\n”匹配换行符。序列“\\”匹配“\”而“\(”则匹配“(”。即相当于多种编程语言中都有的“转义字符”的概念。

定位符

- 正则表达式由一些普通字符和一些特殊字符（元字符metacharacters）组成。普通字符包括大小写的字母和数字，而元字符则具有特殊的含义。
- 在正则表达式中，有一些用于验证文本的定位符：
  - “^”：匹配目标字符串的开始位置，^只有出现在表达式的最前面才具有定位符的作用
  - “$”：匹配目标字符串的结尾位置，$只有出现在表达式的最后面才具有定位符的作用
  - “\b”：匹配一个单词边界，也就是指单词和空格间的位置（即正则表达式的“匹配”有两种概念，一种是匹配字符，一种是匹配位置，这里的\b就是匹配位置的）。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”。
  - “\B”：匹配非单词边界。“er\B”能匹配“verb”中的“er”，但不能匹配“never”中的“er”。

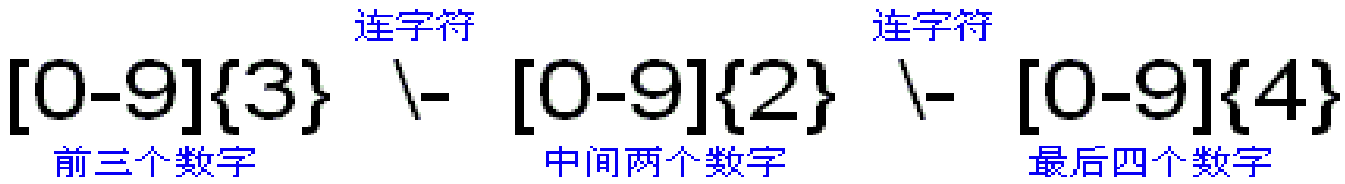
“否”符号

- “^”符号称为“否”符号。如果用在方括号内，“^”表示不想要匹配的字符。
- 例如，[^X]正则表达式匹配所有单词，但以“X”字母开头的单词除外。

[xyz]	字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”。
[^xyz]	负值字符集合。匹配未包含的任意字符。例如，“[^abc]”可以匹配“plain”中的“plin”。
[a-z]	字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符。 注意:只有连字符在字符组内部时,并且出现在两个字符之间时,才能表示字符的范围; 如果出字符组的开头,则只能表示连字符本身.
[^a-z]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符。

例子

- 假设我们要在文本文件中搜索美国的社会安全号码。这个号码的格式是999-99-9999
- 在正则表达式中，连字符（“-”）有着特殊的意义，它表示一个范围，比如从0到9。
- 匹配社会安全号码中的连字符号时，它的前面要加上一个转义字符“\”。



- 假设进行搜索的时候，你希望连字符号可以出现，也可以不出现——即，999-99-9999和999999999都属于正确的格式。

其他符号-1

\s	匹配任何不可见字符，包括空格、制表符、换页符等等。等价于[\f\n\r\t\v]。
\S	匹配任何可见字符。等价于[^ \f\n\r\t\v]。

- 对于日期：“June 26, 1951”，匹配该日期的正则表达式为：

其他符号-2

\w	匹配包括下划线的任何单词字符。类似但不等价于“[A-Za-z0-9_]”，这里的"单词"字符使用Unicode字符集。
\W	匹配任何非单词字符。等价于“[^A-Za-z0-9_]”。
\d	匹配一个数字字符。等价于[0-9]
\D	匹配一个非数字字符。等价于[^0-9]

验证15位、17位、18位身份证号码?

- 15位数字(\d{15})
- 18位数字(\d{18})
- 18位(前17位数字，后可为字母x) (\d{17}[x])

正则表达式

- 元字符：
  - 所谓的元字符，就是正则表达式中具有一些特殊含义的字符，比如重复 N 次前面的字符等

.	匹配除换行符 \n之外的任何单字符。要匹配 .，请使用转义标识“ \ ” “t.n”，它匹配“tan”、“ten”、“tin”和“ton”，还匹配“t#n”、“tpn”等
[]	只有方括号里面指定的字符才参与匹配。 也就是说，正则表达式“t[aeio]n”只匹配“tan”、“Ten”、“tin”和“ton”。但“Toon”不匹配，因为在方括号之内你只能匹配单个字符 .方括号表示可能出现的单个字符
x| y	与 x 或 y 匹配。例如，“z| food”与“z”或“food”匹配 句点和方括号只匹配单个字符，“|”可以匹配多个字符。“t(a|e|i|o|oo)n”正则表达式。这里不能使用方扩号，因为方括号只允许匹配单个字符；这里必须使用圆括号“()”。可以匹配 tan、ten、tin、ton、toon等

正则表达式

*	匹配前面的子表达式任意次。例如，zo*能匹配“z”，也能匹配“zo”以及“zoo”。*等价于{0,}。
+	匹配前面的子表达式一次或多次(大于等于1次）。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。+等价于{1,}。
?	匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”。?等价于{0,1}。
{*n*}	*n*是一个非负整数。匹配确定的*n*次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个o。
{*n*,}	*n*是一个非负整数。至少匹配*n*次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“fooooood”中的所有o。“o{1,}”等价于“o+”。“o{0,}”则等价于“o*”。
{*n*,*m*}	*m*和*n*均为非负整数，其中*n*<=*m*。最少匹配*n*次且最多匹配*m*次。例如，“o{1,3}”将匹配“fooooood”中的前三个o为一组，后三个o为一组。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格。

正则表达式

- 模式修正符：
  - 所谓的模式修正符，即可以在不改变正则表达式的情况下，通过模式修正符改变正则表达式的含义，从而实现一些匹配结果的调整等功能

正则表达式

- 贪婪模式与懒惰模式：
  - 贪婪模式的核心点就是尽可能多的匹配，
  - 而懒惰模式的核心点就是尽可 能少的匹配。

正则表达式

- 正则表达式函数
  - 正则表达式函数有
    - re.match()函数
    - re.search()函数
    - 全局匹配函数：compile().findall()
    - re.sub()函数

小结

- 句点.表示单个字符
- 方括号[]表示可能出现字符集里的单个字符
- 括号()可表示多个字符,还有分组功能
- 大括号{}表示次数
- 转义字符“\”
- 一般正则表达式是由通配符和固定字符组成
- ^”符号称为“否”符号
- \d \D \w \W \s \S等快捷符号

正则表达式例子

- 匹配.com或.cn网址
- 打开当当上的一个网页，抓取该网页上图书的价格，并将结果写入文件。

Python库—URLLib

- Pathon内置库，无需安装
  - **Source code:** Lib/urllib/

- **URLLIB：含有四个模块的、用于处理URL的包**
  - **Urllib.request：请求模块，模拟HTTP请求，打开和读取URL**
  - **Urllib.error：异常处理模块**
  - **Urllib.parse：工具模块，提供URL解析处理方法**
  - **Urllin.robotparser：robot.txt解析模块，判断网站是否可爬**

URLLib—urlopen

- 可从指定的 URL 地址获取网页数据
- 相比Python2变化
  - Python2
    - import urllib2
    - response = urllib2.urlopen('http://www.baidu.com')
  - Python3
    - import urllib.request
    - response = urllib.request.urlopen('http://www.baidu.com')

Urllib基础

- 主要包括：
  - urlretrieve(url,filename)：一次性将某网页直接爬到本地存储
  - urlcleanup()：将urlretrieve产生的缓存清除掉
  - info()：返回爬取的环境信息
  - getcode()：
  - geturl()：

超时设置和自动模拟HTTP请求

- urlopen一般常用的有三个参数，它的参数如下：
  urllib.requeset.urlopen(url,data,timeout)
- 第一个参数url即为URL，第二个参数data是访问URL时要传送的数据，第三个timeout是设置超时时间。
  - Timeout：在某些网络情况不好或者服务器端异常的情况会出现请求慢的情况，或者请求异常，所以这个时候我们需要给请求设置一个超时时间，而不是让程序一直在等待结果。
  - Data：**用来上传递数据**，添加data参数的时候就是以post请求方式请求，如果没有data参数就是get请求方式

爬虫的异常处理

- 异常处理概述：
  - 爬虫在运行的过程中，很多时候都会遇到这样或那样的异常。
  - 如果没有异常处理，爬虫遇到异常时就会直接崩溃停止运行，下次再次运行时， 又会重头开始，所以，要开发一个具有顽强生命力的爬虫，必须要进行 异常处理。

爬虫的异常处理

- URLError与HTTPError
  - 两者都是异常处理的类，
  - HTTPError是URLError的子类，HTTPError有异常状态码与异常原因，URLError没有异常状态码，
    - URLError里只有一个属性：reason,即抓异常的时候只能打印错误信息，
    - HTTPError里有三个属性：code,reason,headers，即抓异常的时候可以获得code,reson，headers三个信息
  - 所以，在处理的时候，不能使用URLError直接代替HTTPError。如果要代替，必须要判断是否有状态码属性。

爬虫的浏览器伪装技术

- 浏览器伪装我们一般通过报头进行
- 由于urlopen()对于一些HTTP的高级功能不支持，所以，如果要修改报头，可以使用urllib.request.build_opener()进行，当然，也可以 使用urllib.request.Request()下的add_header()实现浏览器的模拟。

新闻爬虫

- 需求：
  - 将新浪新闻首页（http://news.sina.com.cn/）所有新闻都爬到 本地。
- 思路：
  - 先爬首页，通过正则获取所有新闻链接，然后依次爬各新闻，并存储到本地。

爬虫防屏蔽手段之代理服务器

- 什么是代理服务器
  - 所谓代理服务器，是一个处于我们与互联网中间的服务器，如果使用代 理服务器，我们浏览信息的时候，先向代理服务器发出请求，然后由代 理服务器向互联网获取信息，再返回给我们。
- 使用代理服务器进行信息爬取，可以很好的解决IP限制的问题。

图片爬虫

- 什么是图片爬虫
  - 所谓图片爬虫，即是从互联网中自动把对方服务器上的图片爬下来的爬 虫程序。
- 如何爬取淘宝的图片？

技术不会停下脚步，学习永无止境。