from bs4 import BeautifulSoup

from urllib.request import urlopen

import requests

u=requests.get(r'http://www.pythonscraping.com/pages/warandpeace.html')

infor=BeautifulSoup(u.text,'lxml')

使用infor.div.h1直接找到网页中的第一个div标签的第一个h1

### find()和findAll()

### findAll(tag , tagattr , recursive , text , limit , keywords)

tag标签名，字符串类型或字符串列表

tagattr 标签的属性，字典类型 {‘color’ : ’red’ , ’green’}

recursive 是否递归查找，布尔类型，默认为True

text 使用text去匹配相应的标签，找出具有此文本的标签

limit 设置抓取前几项匹配的标签，当limit=1时，findAll函数等于find函数

子标签和后代标签（子标签就是父标签的儿子，后代标签就是儿子孙子等等）

子标签 html.children()

后代标签 html.descendants()

兄弟标签

html.next\_siblings()只调用后面的兄弟标签，不调用自己，不调用前面的兄弟标签

html.privious\_siblings()只调用前面的兄弟标签，不调用自己，不调用后面的兄弟标签

html.next\_sibling()和html.privious\_sibling()之调用单个标签

### 正则表达式

|  |  |
| --- | --- |
| ^ | 匹配字符串的开头 |
| $ | 匹配字符串的末尾。 |
| . | 匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符。 |
| [...] | 用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k' |
| [^...] | 不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符。 |
| re\* | 匹配0个或多个的表达式。 |
| re+ | 匹配1个或多个的表达式。 |
| re? | 匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式 |
| re{ n} | 精确匹配 n 个前面表达式。例如， **o{2}** 不能匹配 "Bob" 中的 "o"，但是能匹配 "food" 中的两个 o。 |
| re{ n,} | 匹配 n 个前面表达式。例如， o{2,} 不能匹配"Bob"中的"o"，但能匹配 "foooood"中的所有 o。"o{1,}" 等价于 "o+"。"o{0,}" 则等价于 "o\*"。 |
| re{ n, m} | 匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式 |
| a| b | 匹配a或b |
| (re) | 匹配括号内的表达式，也表示一个组 |
| (?imx) | 正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域。 |
| (?-imx) | 正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域。 |
| (?: re) | 类似 (...), 但是不表示一个组 |
| (?imx: re) | 在括号中使用i, m, 或 x 可选标志 |
| (?-imx: re) | 在括号中不使用i, m, 或 x 可选标志 |
| (?#...) | 注释. |
| (?= re) | 前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所含表达式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。 |
| (?! re) | 前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功 |
| (?> re) | 匹配的独立模式，省去回溯。 |
| \w | 匹配字母数字及下划线 |
| \W | 匹配非字母数字及下划线 |
| \s | 匹配任意空白字符，等价于 [\t\n\r\f]. |
| \S | 匹配任意非空字符 |
| \d | 匹配任意数字，等价于 [0-9]. |
| \D | 匹配任意非数字 |
| \A | 匹配字符串开始 |
| \Z | 匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。 |
| \z | 匹配字符串结束 |
| \G | 匹配最后匹配完成的位置。 |
| \b | 匹配一个单词边界，也就是指单词和空格间的位置。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'。 |
| \n, \t, 等. | 匹配一个换行符。匹配一个制表符。等 |
| \1...\9 | 匹配第n个分组的内容。 |
| \10 | 匹配第n个分组的内容，如果它经匹配。否则指的是八进制字符码的表达式。 |

### 爬去Json数据

import json

dic=Json.loads(’json字符串’) 把json字符串转化为python的字典

Json.dump(dic) 把python字典转换成json

json.dump(dic,ensure\_ascii=false,indent=2) #ensure\_ascii能够让中文显示成中文

#indent=2让下一行在上一行的基础上空2格

headers中的referer可能有用

### Requests

Requests.get(url，header={})

Url：请求访问url,返回一个response的类

Header： 请求头，可以伪装为浏览器，请求头有用的包括referer和User-Agent

Timeout：整数类型，超时的秒数，超过时间则报错

Response类

成员：

Content：内容，一般为二进制类型，response.content.decode()（如不行，可使用response.content.decode(‘gbk’)）可以解码

Text：得到网页的html网页字符串，出现乱码使用response.encoding=’utf-8’

Status\_code:请求状态码,详细见 网页：

<https://blog.csdn.net/helloxiaozhe/article/details/82623918>

|  |  |
| --- | --- |
| **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**（200为常见）** |
| 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 |

Url:真正访问的url地址

Requests：发送请求的对象，response.request.headers请求头等信息

Headers：响应头

Requests.post(url,data=None,json=None)

Url:网络连接

Data：数据 ，字典格式

返回response对象

### Retrying模块

From retrying import retry

@retry(stop\_max\_attempt\_number=3)

Def func1():

Print()

Raise ValueError()

### 处理带有cookies的请求

1. 在header中添加cookies

Header={‘user-Agent’ : ’...’ , ‘cookies’ : ‘...’ }

Requests.get(url,header=header)

1. Cookie可以拆分成一个字典

Cookie={‘a’ : ‘b’ ,’c’ : ‘d’}

Requests.get(url,cookie=cookie)

1. 先发送post请求，获取cookie，带有cookie的请求登录页面

Session=requests.session() #session的函数跟requests一样

Session.post(url,data,headers) #服务器设置在本地的cookie会保存在session中

Session.get(url) #会带上之前保存在session中的cookie

Url为登录页面登录输入框的链接，data是用户名和密码，dict类型

### Xpath

使用时应保证所用路径和网页源代码上的一致

1. /html/head/ /可以定位标签
2. //li //可以定位所有的li标签
3. /html/body/div[@class=’class’] 选取属性class的值为class的div
4. /html/body/a/@href @href获取href属性的值
5. /html/body/a/text() 获取标签a的文本
6. /html/body/a//text() //获取a下面的所有文本
7. ./ ./表示当前节点，跟linux一样

Xpath和lxml的使用

From lxml import etree

Element=etree.HTML(‘html字符串’)

Element.xpath(‘xpath的字符串’)

### 写爬虫的套路

1. Url

知道网页的url地址的规律，构造url的列表

1. 发送请求
2. 提取数据

发送请求返回的是json数据，则使用json模块

返回的是html字符串，则使用lxml模块配合xpath定位数据，提取数据

4.保存

### python2中的urllib和urllin2库和python3的urllib库的对应关系

|  |  |
| --- | --- |
| Python 2 name | Python 3 name |
| urllib.urlretrieve() | urllib.request.urlretrieve() |
| urllib.urlcleanup() | urllib.request.urlcleanup() |
| urllib.quote() | urllib.parse.quote() |
| urllib.quote\_plus() | urllib.parse.quote\_plus() |
| urllib.unquote() | urllib.parse.unquote() |
| urllib.unquote\_plus() | urllib.parse.unquote\_plus() |
| urllib.urlencode() | urllib.parse.urlencode() |
| urllib.pathname2url() | urllib.request.pathname2url() |
| urllib.url2pathname() | urllib.request.url2pathname() |
| urllib.getproxies() | urllib.request.getproxies() |
| urllib.URLopener | urllib.request.URLopener |
| urllib.FancyURLopener | urllib.request.FancyURLopener |
| urllib.ContentTooShortError | urllib.error.ContentTooShortError |
| urllib2.urlopen() | urllib.request.urlopen() |
| urllib2.install\_opener() | urllib.request.install\_opener() |
| urllib2.build\_opener() | urllib.request.build\_opener() |
| urllib2.URLError | urllib.error.URLError |
| urllib2.HTTPError | urllib.error.HTTPError |
| urllib2.Request | urllib.request.Request |
| urllib2.OpenerDirector | urllib.request.OpenerDirector |
| urllib2.BaseHandler | urllib.request.BaseHandler |
| urllib2.HTTPDefaultErrorHandler | urllib.request.HTTPDefaultErrorHandler |
| urllib2.HTTPRedirectHandler | urllib.request.HTTPRedirectHandler |
| urllib2.HTTPCookieProcessor | urllib.request.HTTPCookieProcessor |
| urllib2.ProxyHandler | urllib.request.ProxyHandler |
| urllib2.HTTPPasswordMgr | urllib.request.HTTPPasswordMgr |
| urllib2.HTTPPasswordMgrWithDefaultRealm | urllib.request.HTTPPasswordMgrWithDefaultRealm |
| urllib2.AbstractBasicAuthHandler | urllib.request.AbstractBasicAuthHandler |
| urllib2.HTTPBasicAuthHandler | urllib.request.HTTPBasicAuthHandler |
| urllib2.ProxyBasicAuthHandler | urllib.request.ProxyBasicAuthHandler |
| urllib2.AbstractDigestAuthHandler | urllib.request.AbstractDigestAuthHandler |
| urllib2.HTTPDigestAuthHandler | urllib.request.HTTPDigestAuthHandler |
| urllib2.ProxyDigestAuthHandler | urllib.request.ProxyDigestAuthHandler |
| urllib2.HTTPHandler | urllib.request.HTTPHandler |
| urllib2.HTTPSHandler | urllib.request.HTTPSHandler |
| urllib2.FileHandler | urllib.request.FileHandler |
| urllib2.FTPHandler | urllib.request.FTPHandler |
| urllib2.CacheFTPHandler | urllib.request.CacheFTPHandler |
| urllib2.UnknownHandler | urllib.request.UnknownHandler |

### Scrapy框架

Spiders

Engine

Scheduler

Item Pipelines

Downloader

调度

调度

调度

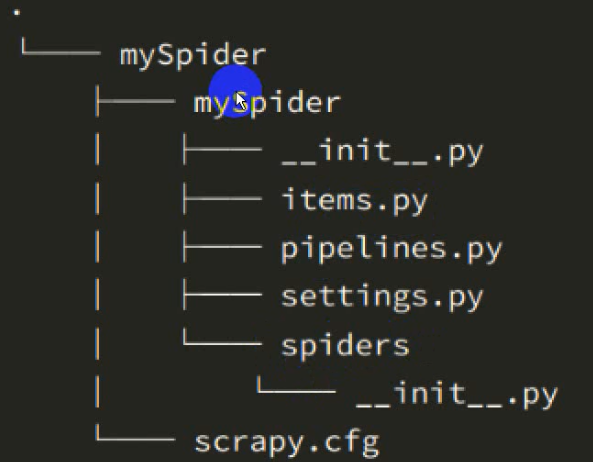
调度

downloader middleware

可修改

流程：

1. 在目录中创建一个scrapy 使用命令 scrapy startproject ITcast
2. 在ITcast\ITcast\spider中开始爬虫 命令：scrapy genspider itcast “http://www.baidu.com” itcast为爬虫生成的文件名，爬虫的一些信息在里面；网址是爬虫获取的范围
3. 执行爬虫，命令scrapy crawl



#### 各个文件的作用：

Scrapy.cfg：项目的配置文件

Myspider/项目的python模块，将会从这里引用代码

Myspider/items.py ：项目的目标文件

Myspider/pipelines.py：项目的管道文件

Myspider/setting.py：项目的设置文件

Myspider/spiders/：储存爬虫代码目录

#### Setting.py

USER\_AGENT

ROBOTSXT\_OBEY =True 遵循网站的协议 有些反爬虫的就不能用了 所以一般是设置False

CONCURRENT\_REQUESTS=32 请求的并发量

DOWNLOAD\_DELAY=3 下载延迟3s

COOKIES\_ENABLED=False 是否启用cookies

TLNETCONSOL\_ENABLED=False 是否启用监控

DEFAULT\_REQUEST\_HEADER={} 默认请求报头