Scrapy框架

Scrapy是一个为了爬取网站数据，提取结构性数据而编写的应用框架。 其可以应用在数据挖掘，信息处理或存储历史数据等一系列的程序中。其最初是为了页面抓取 (更确切来说, 网络抓取 )所设计的， 也可以应用在获取API所返回的数据(例如 Amazon Associates Web Services ) 或者通用的网络爬虫。Scrapy用途广泛，可以用于数据挖掘、监测和自动化测试。Scrapy 使用了 Twisted异步网络库来处理网络通讯。整体架构大致如下



Scrapy主要包括了以下组件：

* + **引擎(Scrapy)**  
    *用来处理整个系统的数据流, 触发事务(框架核心)*
  + **调度器(Scheduler)**  
    *用来接受引擎发过来的请求, 压入队列中, 并在引擎再次请求的时候返回. 可以想像成一个URL（抓取网页的网址或者说是链接）的优先队列, 由它来决定下一个要抓取的网址是什么, 同时去除重复的网址*
  + **下载器(Downloader)**  
    *用于下载网页内容, 并将网页内容返回给蜘蛛(Scrapy下载器是建立在twisted这个高效的异步模型上的)*
  + **爬虫(Spiders)**  
    *爬虫是主要干活的, 用于从特定的网页中提取自己需要的信息, 即所谓的实体(Item)。用户也可以从中提取出链接,让Scrapy继续抓取下一个页面*
  + **项目管道(Pipeline)**  
    *负责处理爬虫从网页中抽取的实体，主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。当页面被爬虫解析后，将被发送到项目管道，并经过几个特定的次序处理数据。*
  + **下载器中间件(Downloader Middlewares)**  
    *位于Scrapy引擎和下载器之间的框架，主要是处理Scrapy引擎与下载器之间的请求及响应。*
  + **爬虫中间件(Spider Middlewares)**  
    *介于Scrapy引擎和爬虫之间的框架，主要工作是处理蜘蛛的响应输入和请求输出。*
  + **调度中间件(Scheduler Middewares)**  
    *介于Scrapy引擎和调度之间的中间件，从Scrapy引擎发送到调度的请求和响应。*

Scrapy运行流程大概如下：

1. 引擎从调度器中取出一个链接(URL)用于接下来的抓取
2. 引擎把URL封装成一个请求(Request)传给下载器
3. 下载器把资源下载下来，并封装成应答包(Response)
4. 爬虫解析Response
5. 解析出实体（Item）,则交给实体管道进行进一步的处理
6. 解析出的是链接（URL）,则把URL交给调度器等待抓取

**Scrapy的运作流程**

代码写好，程序开始运行...

* 1 引擎：Hi！Spider, 你要处理哪一个网站？
* 2 Spider：老大要我处理xxxx.com。
* 3 引擎：你把第一个需要处理的URL给我吧。
* 4 Spider：给你，第一个URL是xxxxxxx.com。
* 5 引擎：Hi！调度器，我这有request请求你帮我排序入队一下。
* 6 调度器：好的，正在处理你等一下。
* 7 引擎：Hi！调度器，把你处理好的request请求给我。
* 8 调度器：给你，这是我处理好的request
* 9 引擎：Hi！下载器，你按照老大的下载中间件的设置帮我下载一下这个request请求
* 10 下载器：好的！给你，这是下载好的东西。（如果失败：sorry，这个request下载失败了。然后引擎告诉调度器，这个request下载失败了，你记录一下，我们待会儿再下载）
* 11 引擎：Hi！Spider，这是下载好的东西，并且已经按照老大的下载中间件处理过了，你自己处理一下（注意！这儿responses默认是交给def parse()这个函数处理的）
* 12 Spider：（处理完毕数据之后对于需要跟进的URL），Hi！引擎，我这里有两个结果，这个是我需要跟进的URL，还有这个是我获取到的Item数据。
* 13 引擎：Hi ！管道 我这儿有个item你帮我处理一下！调度器！这是需要跟进URL你帮我处理下。然后从第四步开始循环，直到获取完老大需要全部信息。
* 14 管道调度器：好的，现在就做！

**注意！只有当调度器中不存在任何request了，整个程序才会停止，（也就是说，对于下载失败的URL，Scrapy也会重新下载。）**

 安装（建议使用pip进行安装，不容易出错）

[复制代码](javascript:void(0);)

1、更新pip到最新（也可以不提前更新，系统后期会提示如何进行更新）

2、安装wheel（不安装也没事，没影响）

pip install wheel

3、安装lxml(如果没进行第一步这里会在安装好之后提醒更新pip方法)

第一种：pip install lxml

第二种: lxml各版本下载网址：（容易出错）

<https://pypi.python.org/pypi/lxml/4.1.0>; <https://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml>;

lxml版本理解：

<https://blog.csdn.net/loopwastemytime/article/details/79919865>

4、安装pyopenssl

第一种：pip install pyOpenSSL

第二种：https://pypi.python.org/pypi/pyOpenSSL/17.5.0

**5**、安装Twisted（一般而言一定要安装，不然报错缺少c++14.0，或其它版本）

第一种：pip install twisted

第二种：https://www.lfd.uci.edu/~gohlke/pythonlibs/

**6**、安装pywin32

第一种：pip install pywin32

第二种：<https://sourceforge.net/projects/pywin32/files/>；<https://github.com/mhammond/pywin32/releases>

7、安装scrapy(上面的都安装完毕以后才能进行这一步)

pip install scrapy

**xpath（考虑response不是一个网页的情况，例如网址的目的地是一个pdf文件，上网查）**

[**http://www.w3school.com.cn/xpath/xpath\_syntax.asp**](http://www.w3school.com.cn/xpath/xpath_syntax.asp)

[**https://blog.csdn.net/hhtnan/article/details/77509549**](https://blog.csdn.net/hhtnan/article/details/77509549)

**Selector是一个选择器，含有四个基本方法：**

**xpath()：传入xpath表达式，返回该表达式所对应的所有节点的selector list列表。**

**css()：传入css表达式，返回该表达式所对应的所有节点的selector list列表。**

**extract()：序列化该节点为unicode字符串并返回list。**

**re()根据传入的正则表达式对数据进行提取，并返回unicode字符串list列表。**

**response.xpath()==response.selector.xpath()默认情况下已经可以通用，两种情况下已经进行映射，所以完全相同的使用**

**例子：about.txt文件为例**

**>>> response.xpath('//div[@class="toptext"]/strong')**

**[<Selector xpath='//div[@class="toptext"]/strong' data='<strong>DMOZ中文网站分类目录-免费收录各类优秀网站的中文网站目 录.<'>]**

**>>> response.xpath('//div[@class="toptext"]/strong/text()')**

**[<Selector xpath='//div[@class="toptext"]/strong/text()' data='DMOZ中文网站分类目录-免费收录各类优秀网站的中文网站目录.'>]**

**>>> response.xpath('//div[@class="toptext"]/strong').extract()**

**['<strong>DMOZ中文网站分类目录-免费收录各类优秀网站的中文网站目录.</strong>']**

**>>> response.xpath('//div[@class="toptext"]/strong/text()').extract()**

**['DMOZ中文网站分类目录-免费收录各类优秀网站的中文网站目录.']**

**说明了extract()是用来控制返回Selector对象，还是只输出Selector中的data值，含有extract（）用来输出data值，否则输出Selector对象，text()用来控制data数据的格式，当含有text()时，则会将data值中的标签去掉，否则带有标签。（切记只能去掉一层标签）**

**text()用来输出所选节点（就是前面的一个节点）的文本内容**

**response.xpath('//ul[@class="header-nav"]/li/a[@href="http://www.dmozdir.org/User/UserReg.asp"]/text()').extract()**

response.xpath('//ul[@class="header-nav"]/li[**3**]/text()').extract()

response.xpath('//ul[@class="header-nav"]/li[last()-2]').extract()

response.xpath('//ul[@class="header-nav"]/li[position()<3]’).extract()

………………………………………………………………………………………………………………

谓语用来查找某个特定的节点或者包含某个指定的值的节点。谓语被嵌在方括号中。**[@class="header-nav"]，这里括号里的就是所说的谓词，用谓词来对它前面的节点进行修饰，限定，只有满足谓词条件的节点才满足条件，是所要查找的节点。一般充当谓词的有属性，数字（表示顺序）或者一些特定的表示顺序的函数（例如last()函数结果是得到最后一个标签的序号）或其他。**

**response.xpath('//ul[@class="header-nav"]/\*').extract()**

**\*能代表任意一个节点，是一个通配符**

**@\*表示任意一个的属性**

**response.xpath('//div[@class="menu"]/node()').extract()**

**node()能匹配任何类型的节点**

**response.xpath('//div[@class="toptext"]|//div[@class="header"]).extract()**

**“|“表示的是或的关系，与逻辑或一样，一个存在就能查出来一个，两个都存在则查出两个。**

>>> st=response.xpath('//ul[@class="topmenu"]/li/a/span')

>>> info=st.xpath('string(.)')

>>> info

[<Selector xpath='string(.)' data='DmozDir首页'>, <Selector xpath='string(.)' data='提交网站'>, <Selector xpath='string(.)' data='最新收录'>, <Selector xpath='string(.)' data='入站排行榜'>, <Selector xpath='string(.)' data='建站资讯'>, <Selector xpath='string(.)' data='了解本站'>]

st=response.xpath('string(//ul[@class="topmenu"]/li/a/span)').extract()

['DmozDir首页']

**string只能解析匹配到的第一个节点下的值，也就是作用于list时只匹配第一个**

**(string(list)要注意它只能解析第一个，其他的不会解析)**

>>> info=st.xpath('string(.)').extract()

>>> info

['DmozDir首页', '提交网站', '最新收录', '入站排行榜', '建站资讯', '了解本站']

>>> st=response.xpath('//ul[@class="topmenu"]')

>>> info=st.xpath('string(.)')

>>> info

[<Selector xpath='string(.)' data='\r\n\t\t\t\t\tDmozDir首页\r\n\t\t\t\t\t提交网站\r\n\t\t\t\t\t|\r\n\t\t\t'>]

>>> info=st.xpath('string(.)').extract()

>>> info

['\r\n\t\t\t\t\tDmozDir首页\r\n\t\t\t\t\t提交网站\r\n\t\t\t\t\t|\r\n\t\t\t\t\t最新收录\r\n\t\t\t\t\t|\r\n\t\t\t\t\t入站 排行榜\r\n\t\t\t\t\t|\r\n\t\t\t\t\t建站资讯\r\n\t\t\t\t\t|\r\n\t\t\t\t\t了解本站\r\n\t\t\t\t']

**string()函数能够解析当前节点下的字符（主要是文本内容）（当前节点的下的节点的文本内容也可以解析出来，<元素节点>文本内容<元素节点>），text()却只能解析属于当前节点的文本内容（当前节点下的节点的文本内容是不能解析出来的），这一点与string()不太一样**

**例子：**

>>> st=st.extract()

>>> st

['\r\n\t\t\t\t\tDmozDir首页\r\n\t\t\t\t\t提交网站\r\n\t\t\t\t\t|\r\n\t\t\t\t\t最新收录\r\n\t\t\t\t\t|\r\n\t\t\t\t\t入站 排行榜\r\n\t\t\t\t\t|\r\n\t\t\t\t\t建站资讯\r\n\t\t\t\t\t|\r\n\t\t\t\t\t了解本站\r\n\t\t\t\t']

>>> st=st[0]

>>> st

'\r\n\t\t\t\t\tDmozDir首页\r\n\t\t\t\t\t提交网站\r\n\t\t\t\t\t|\r\n\t\t\t\t\t最新收录\r\n\t\t\t\t\t|\r\n\t\t\t\t\t入站排行榜\r\n\t\t\t\t\t|\r\n\t\t\t\t\t建站资讯\r\n\t\t\t\t\t|\r\n\t\t\t\t\t了解本站\r\n\t\t\t\t'

>>> st=st.replace("\r\n\t\t\t\t","")

>>> st

'\tDmozDir首页\t提交网站\t|\t最新收录\t|\t入站排行榜\t|\t建站资讯\t|\t了解本站'

>>> st=st.replace("\t","")

>>> st

'DmozDir首页提交网站|最新收录|入站排行榜|建站资讯|了解本站'

>>> st=st.replace("|","")

>>> st

'DmozDir首页提交网站最新收录入站排行榜建站资讯了解本站'

>>> st=st.extract()

>>> st

['\r\n\t\t\t\t\tDmozDir首页\r\n\t\t\t\t\t提交网站\r\n\t\t\t\t\t|\r\n\t\t\t\t\t最新收录\r\n\t\t\t\t\t|\r\n\t\t\t\t\t入站 排行榜\r\n\t\t\t\t\t|\r\n\t\t\t\t\t建站资讯\r\n\t\t\t\t\t|\r\n\t\t\t\t\t了解本站\r\n\t\t\t\t']

>>> st=st[0]

>>> st

'\r\n\t\t\t\t\tDmozDir首页\r\n\t\t\t\t\t提交网站\r\n\t\t\t\t\t|\r\n\t\t\t\t\t最新收录\r\n\t\t\t\t\t|\r\n\t\t\t\t\t入站排行榜\r\n\t\t\t\t\t|\r\n\t\t\t\t\t建站资讯\r\n\t\t\t\t\t|\r\n\t\t\t\t\t了解本站\r\n\t\t\t\t'

>>> st=st.replace("\r\n\t\t\t\t","")

>>> st

'\tDmozDir首页\t提交网站\t|\t最新收录\t|\t入站排行榜\t|\t建站资讯\t|\t了解本站'

>>> st=st.replace("\t","")

>>> st

'DmozDir首页提交网站|最新收录|入站排行榜|建站资讯|了解本站'

>>> st=st.replace("|","")

>>> st

'DmozDir首页提交网站最新收录入站排行榜建站资讯了解本站'

多层爬取（看实例dcpq）

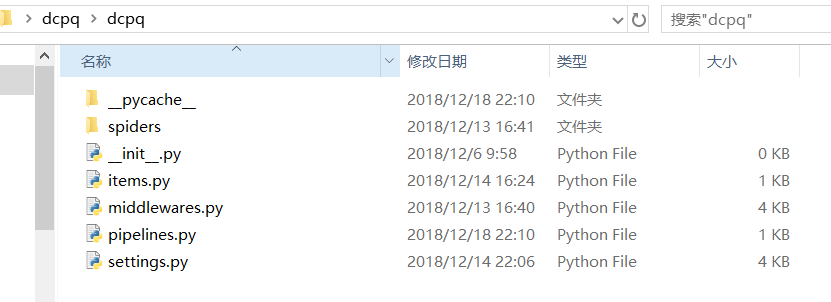
1、创建工程

|  |  |
| --- | --- |
| 1 | scrapy startproject dcpq |

2、创建爬虫程序

|  |  |
| --- | --- |
| 1  2 | cd dcpq  scrapy genspider appendix appendix.com |

 3、自动创建目录及文件



4、文件说明：

* scrapy.cfg  项目的配置信息，主要为Scrapy命令行工具提供一个基础的配置信息。（真正爬虫相关的配置信息在settings.py文件中）
* items.py    设置数据存储模板，用于结构化数据，如：Django的Model
* pipelines    数据处理行为，如：一般结构化的数据持久化
* settings.py 配置文件，如：递归的层数、并发数，延迟下载等
* spiders      爬虫目录，如：创建文件，编写爬虫规则

注意：一般创建爬虫文件时，以网站域名命名

5、设置数据存储模板

items.py

 import scrapy

class DcpqItem(scrapy.Item):

# define the fields for your item here like:

# name = scrapy.Field()

type = scrapy.Field()

title = scrapy.Field()

pubTime = scrapy.Field()

picture = scrapy.Field()

content = scrapy.Field()

url=scrapy.Field()

pass

6、编写爬虫

appendix.py

# -\*- coding: gbk -\*-

import scrapy

from scrapy import Request

from dcpq.items import DcpqItem

class AppendixSpider(scrapy.Spider):

name = "appendix"

allowed\_domains = ['ches.org.cn']

start\_urls = ['http://www.ches.org.cn/ches/slkp/slkpsy/']

def parse(self, response):

title\_list = response.xpath("/html/body/div[5]/div/div[1]/div[2]/ul/li/a/p/text()").extract()

url\_list = response.xpath("/html/body/div[5]/div/div[1]/div[2]/ul/li/a/@href").extract()

for i, j in zip(title\_list, url\_list):

dcpq = DcpqItem()

dcpq['type'] = i

url ='http://www.ches.org.cn/ches' + j[5:] + '/'

yield scrapy.Request(url, callback=self.title\_parse, meta={'item': dcpq, 'url': url})

print(i, ':', url)

def title\_parse(self, response):

dcpq = response.meta['item']

dcpqurl = response.meta['url']

title\_list = response.xpath("/html/body/div/div/div[4]/div[1]/div/div[1]/div/div/div/div/div[1]/h5/a/text()").extract()

url\_list = response.xpath("/html/body/div/div/div[4]/div[1]/div/div[1]/div/div/div/div/div[1]/h5/a/@href").extract()

time\_list = response.xpath("/html/body/div/div/div[4]/div[1]/div/div[1]/div/div/div/div/div[2]/h6/text()").extract()

for title, url, time in zip(title\_list, url\_list, time\_list):

dcpq['title'] = title

dcpq['pubTime'] = time

url = dcpqurl + url[2:]

dcpq["url"]=url

print(title, ':', time, ':', url)

if url[len(url)-3:]!="pdf":

yield scrapy.Request(url, callback=self.content\_parse, meta={'item': dcpq})

def content\_parse(self, response):

dcpq = response.meta['item']

content = response.xpath('string(//div[@class="row"]/div/div/h3[@style="line-height:40px;"])').extract()

if content!=[]:

content=content[0]

print(content)

dcpq['content'] = content

yield dcpq

**下载器下载后会返回一个response对象，这个对象会返回给parse（）函数进行处理，当scrapy检测到parse函数要当作生成器使用时，就会以生成器的方式调用parse即：**

**for n in parse(self, response): pass，每次执行for循环就相当于执行next（）函数，执行parse函数，当执行到yield是，parse函数就暂停，并返回yield后面的对象，即执行多少次据返回多少次，（及不断地向项目管道输送数据，返回一个 输送一个，然后pipelines.py处理一个，直到送完，处理完。）**

具体执行请参照：<https://blog.csdn.net/heheyanyanjun/article/details/79199378>

7、设置配置文件

settings.py增加如下内容

ITEM\_PIPELINES = {'dcpq.pipelines.DcpqPipeline':1000}

USER\_AGENT='Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.146 Safari/537.36'

8、编写数据处理脚本

　　pipelines.py

 class DcpqPipeline(object):

def process\_item(self, item, spider):

with open("my\_dcpq.txt",'a') as fp:

fp.write(item["type"]+item["title"]+item["pubTime"]+item["url"]+item["content"]+'\n')

9、执行爬虫

|  |  |
| --- | --- |
| 1  2 | cd dcpq  scrapy crawl appendix --nolog |

10、结果



参考：<https://www.cnblogs.com/kongzhagen/p/6549053.html>

<http://python.jobbole.com/82294/>

<https://www.jianshu.com/p/8d353c7cf606>

<https://blog.csdn.net/weixin_37947156/article/details/74974208>

<https://www.cnblogs.com/thunderLL/p/6551641.html>

<http://www.runoob.com/w3cnote/scrapy-detail.html>

<http://www.bubuko.com/infodetail-2611517.html>

<https://blog.csdn.net/heheyanyanjun/article/details/79199378>

<https://blog.csdn.net/on_the_road_2018/article/details/80985481>