Python网络爬虫操作流程

基本的操作步骤：

1. 查找域名对应的ip地址
2. 向ip对应的服务器发送请求
3. 服务器响应请求，返回网页内容
4. 浏览器解析网页内容
5. 网络爬虫的步骤：

1、抓取：

1) 基本的抓取步骤

抓取大多数属于get请求，直接从对方服务器上获取数据

方法一 Requests：

import requests

response = requests.get(url)

content = requests.get(url).content

print "response headers:", response.headers

print "content:", content

方法二 Urllib2：

import urllib2

response = urllib2.urlopen(url)

content = urllib2.urlopen(url).read()

print "response headers:", response.headers

print "content:", content

方法三 Httplib2：

import httplib2

http = httplib2.Http()

response\_headers, content = http.request(url, 'GET')

print "response headers:", response\_headers

print "content:", content

对于有参数的地址:

data = {'data1':'XXXXX', 'data2':'XXXXX'}

Requests：data为dict，json

import requests

response = requests.get(url=url, params=data)

Urllib2：data为string

import urllib, urllib2

data = urllib.urlencode(data)

full\_url = url+'?'+data

response = urllib2.urlopen(full\_url)

1. 带有登录情况的爬取：

1、使用表单登录form

这种情况属于post请求，向服务器发送表单数据，服务器将返回的cookie存入本地

data = {'data1':'XXXXX', 'data2':'XXXXX'}

Requests：data为dict，json

import requests

response = requests.post(url=url, data=data)

Urllib2：data为string

import urllib, urllib2

data = urllib.urlencode(data)

req = urllib2.Request(url=url, data=data)

response = urllib2.urlopen(req)

2、使用cookie登录

使用cookie登陆，服务器会认为你是一个已登陆的用户，所以就会返回给你一个已登陆的内容。因此，需要验证码的情况可以使用带验证码登陆的cookie解决

import requests

requests\_session = requests.session()

response = requests\_session.post(url=url\_login, data=data)

若存在验证码，此时采用response = requests\_session.post(url=url\_login, data=data)是不行的，做法应该如下：

response\_captcha = requests\_session.get(url=url\_login, cookies=cookies)

response1 = requests.get(url\_login) # 未登陆

response2 = requests\_session.get(url\_login) # 已登陆，因为之前拿到了Response Cookie！

response3 = requests\_session.get(url\_results) # 已登陆，因为之前拿到了Response Cookie！

1. 对于反爬虫机制的处理
2. 使用代理

适用情况：限制IP地址情况，也可解决由于“频繁点击”而需要输入验证码登陆的情况。

这种情况最好的办法就是维护一个代理IP池，网上有很多免费的代理IP，良莠不齐，可以通过筛选找到能用的。对于“频繁点击”的情况，我们还可以通过限制爬虫访问网站的频率来避免被网站禁掉

proxies = {'http':'http://XX.XX.XX.XX:XXXX'}

Requests：

import requests

response = requests.get(url=url, proxies=proxies)

Urllib2：

import urllib2

proxy\_support = urllib2.ProxyHandler(proxies)

opener = urllib2.build\_opener(proxy\_support, urllib2.HTTPHandler)

urllib2.install\_opener(opener) # 安装opener，此后调用urlopen()时都会使用安装过的opener对象

response = urllib2.urlopen(url)

1. 时间设置

适用情况：限制频率情况。

Requests，Urllib2都可以使用time库的sleep()函数：

import time

time.sleep(1)

1. 伪装成浏览器或者“反盗链”

有些网站会检查你是不是真的浏览器访问，还是机器自动访问的。这种情况，加上User-Agent，表明你是浏览器访问即可。有时还会检查是否带Referer信息还会检查你的Referer是否合法，一般再加上Referer。

headers = {'User-Agent':'XXXXX'} # 伪装成浏览器访问，适用于拒绝爬虫的网站

headers = {'Referer':'XXXXX'}

headers = {'User-Agent':'XXXXX', 'Referer':'XXXXX'}

Requests：

response = requests.get(url=url, headers=headers)

Urllib2：

import urllib, urllib2

req = urllib2.Request(url=url, headers=headers)

response = urllib2.urlopen(req)

def multi\_session(session, \*arg):

retryTimes = 20

while retryTimes>0:

try:

return session.post(\*arg)

except:

print '.',

retryTimes -= 1

或者

def multi\_open(opener, \*arg):

retryTimes = 20

while retryTimes>0:

try:

return opener.open(\*arg)

except:

print '.',

retryTimes -= 1

1. 多进程抓取
2. 对于Ajax请求的处理

对于“加载更多”情况，使用Ajax来传输很多数据。

1. 自动化测试工具

Selenium是一款自动化测试工具。它能实现操纵浏览器，包括字符填充、鼠标点击、获取元素、页面切换等一系列操作

1. 验证码识别

使用代理，更新IP。

使用cookie登陆。

验证码识别。

可以利用开源的Tesseract-OCR系统进行验证码图片的下载及识别，将识别的字符传到爬虫系统进行模拟登陆。当然也可以将验证码图片上传到打码平台上进行识别。如果不成功，可以再次更新验证码识别，直到成功为止

**爬取有两个需要注意的问题：**

* 如何监控一系列网站的更新情况，也就是说，如何进行增量式爬取？
* 对于海量数据，如何实现分布式爬取？

**分析**

抓取之后就是对抓取的内容进行分析，你需要什么内容，就从中提炼出相关的内容来。

常见的分析工具有[正则表达式](http://deerchao.net/tutorials/regex/regex.htm)，[BeautifulSoup](http://www.crummy.com/software/BeautifulSoup/)，[lxml](http://lxml.de/)等等。

**存储**

分析出我们需要的内容之后，接下来就是存储了。

我们可以选择存入文本文件，也可以选择存入[MySQL](http://www.mysql.com/)或[MongoDB](https://www.mongodb.org/)数据库等。

**存储有两个需要注意的问题：**

* 如何进行网页去重？
* 内容以什么形式存储？

1. 爬虫所用到的框架

Scrapy

Scrapy是一个基于Twisted的开源的Python爬虫框架，在工业中应用非常广泛。

**引擎**，用来处理整个系统的数据流处理，触发事务。

**调度器**，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。

**下载器**，用于下载网页内容，并将网页内容返回给蜘蛛。

**蜘蛛**，蜘蛛是主要干活的，用它来制订特定域名或网页的解析规则。

**项目管道**，负责处理有蜘蛛从网页中抽取的项目，他的主要任务是清晰、验证和存储数据。当页面被蜘蛛解析后，将被发送到项目管道，并经过几个特定的次序处理数据。

**调度中间件**，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。

**下载器中间件**，位于Scrapy引擎和下载器之间的钩子框架，主要是处理Scrapy引擎与下载器之间的请求及响应。

**蜘蛛中间件**，介于Scrapy引擎和蜘蛛之间的钩子框架，主要工作是处理蜘蛛的响应输入和请求输出。

1. Robots协议

Robots协议（也称为爬虫协议、机器人协议等）的全称是“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。

User-agent: 指定对哪些爬虫生效

Disallow: 指定不允许访问的网址

Allow: 指定允许访问的网址

垂直搜索引擎分布式网络爬虫

垂直搜索引擎中分布式网络爬虫

* 分布式使用redis实现，redis中存储了工程的request，stats信息，能够对各个机器上的爬虫实现集中管理，这样可以 解决爬虫的性能瓶颈，利用redis的高效和易于扩展能够轻松实现高效率下载：当redis存储或者访问速度遇到瓶颈时，可以 通过增大redis集群数和爬虫集群数量改善。
* 底层存储实现了两种方式：
  + 将书名，作者，书籍封面图片文件系统路径，书籍概要，原始网址链接，书籍下载信息，书籍文件系统路径保存到mongodb 中，此时mongodb使用单个服务器,对图片采用图片的url的hash值作为文件名进行存储，同时可以定制生成各种大小尺寸的缩略 图，对文件动态获得文件名，将其下载到本地，存储方式和图片类似，这样在每次下载之前会检查图片和文件是否曾经下载，对 已经下载的不再下载；
  + 将书名，作者，书籍封面图片文件系统路径，书籍概要，原始网址链接，书籍下载信息，书籍保存到mongodb中，此时mongodb 采用mongodb集群进行存储，片键和索引的选择请看代码，文件采用mongodb的gridfs存储,图片仍然存储在文件系统中,在每次下载 之前会检查图片和文件是否曾经下载，对已经下载的不再下载；
* 避免爬虫被禁的策略：
  + 禁用cookie
  + 实现了一个download middleware，不停的变user-aget
  + 实现了一个可以访问google cache中的数据的download middleware(默认禁用)
* 调试策略的实现：
  + 将系统log信息写到文件中
  + 对重要的log信息(eg:drop item,success)采用彩色样式终端打印
* 文件，信息存储：
  + 实现了FilePipeline可以将指定扩展名的文件下载到本地
  + 实现了MongodbWoaiduBookFile可以将文件以gridfs形式存储在mongodb集群中
  + 实现了SingleMongodbPipeline和ShardMongodbPipeline，用来将采集的信息分别以单服务器和集群方式保存到mongodb中
* 访问速度动态控制:
  + 跟据网络延迟，分析出scrapy服务器和网站的响应速度，动态改变网站下载延迟
  + 配置最大并行requests个数，每个域名最大并行请求个数和并行处理items个数
* 爬虫状态查看：
  + 将爬虫stats信息(请求个数，文件下载个数，图片下载个数等)保存到redis中
  + 实现了一个针对分布式的stats collector，并将其结果用graphite以图表形式动态实时显示
* mongodb集群部署：在commands目录下有init\_sharding\_mongodb.py文件，可以方便在本地部署

爬虫基本思路：

发起请求request()，其中带上各种各样的参数 得到一个应答response对象 通过各种工具对应答的内容做各种各样的解析处理,得到我们想要的数据 将数据进行存储(mysql,elastic,mongodb)

* 1. request所带的参数：
* scrapy.Request(url=,meta=,callback=)

请求头，模拟浏览器访问，访问成功，发送下一次的请求

* 1. response应答，解析其中所需要的内容

逐步进行解析:

def parse1(self, response):

Selector(response) 选择

pre\_img\_url = hxs.xpath('//script/text()').extract() 解析正则表达式

Scrapy工程搭建基本思路：

1. 在spider中编写爬虫的主程序
2. 在pipelines中编写爬虫后的数据进行存储
3. 在setting中进行所有的设置
4. 在items中定义爬取的数据的项目

爬虫网页网页加载不完整解决方法

1. 网页中的一些页面是由于js触发的

解决方案：模拟浏览器的JS触发时间,常用的是selenium

在setting.py中设置

1. 抓驱动态网站

<http://scrapy-cookbook.readthedocs.io/zh_CN/latest/scrapy-12.html>

百度动态页面的抓取

<http://www.voidcn.com/article/p-mpeagwtp-bpz.html>

爬取动态页面

<http://www.voidcn.com/article/p-fzxbrexl-w.html>

爬取验证码的数据

<https://blog.csdn.net/u011829453/article/details/77770816>

<http://www.cnblogs.com/chaihy/p/9044574.html>

<https://blog.csdn.net/qq_39122387/article/details/78346785>

模拟登录

<https://piaosanlang.gitbooks.io/spiders/03day/section3.9.html>

分布式爬虫

<https://blog.csdn.net/seven_2016/article/details/72802961>

scrapy\_redis分布式爬虫

<https://blog.csdn.net/qq_25046261/article/details/79748576>

scrapy 反爬虫机制

<http://www.aobosir.com/blog/2016/12/06/python3-large-web-crawler-scrapy-project-Anti-reptile-settings/>

<https://blog.csdn.net/zupzng/article/details/80032018>

<https://blog.csdn.net/weixin_35705390/article/details/77802898>

<https://segmentfault.com/a/1190000014333162>

scrapy爬虫性能的优化

<http://so1n.me/2017/08/08/7/>

python解析json格式的数据

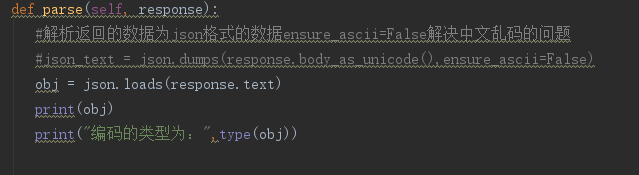
* 1. json.dumps(data) ,将数据转换为josn格式的数据
  2. 将json格式的数转换为python 格式的数据
  3. json数据解析出问题

1. reponse返回的数据为字符串，将其转换为json格式数据

解决方案：① python请求头中请求json格式的数据

② 将返回的数据转换成json格式的，在进行后续的操作

* 1. 提取json格式的数据



* 1. 找到网络中文件类型为html和json类型的请求，
  2. 拿到地址发起请求，观察返回的json数据

Python提取数据

**xpath选择器**

　　xpath即xml路径语言，用来确定xml文档中某个部分的语言，xml文档（html属于xml）是一系列节点构成的树。

基础语法

**/**     选中文档的根   描述一个从根开始的绝对路径(从当前节点选取直接子节点)

**./**表示从当前节点选取 （比如提取了一部分，还需要从提取出来里面进行提取就会使用，如果不加 **.**会从整个文档中开始提取）

**.**      选中当前节点    用来描述相对路径

**..**    选中当前节点的父节点  用来描述相对路径

**ELEMENT**    选中子节点中所有的ELEMENT元素节点

**//ELEMENT**  选中子孙节点中所有的ELEMENT元素节点

**\***      选中所有元素子节点

**text()**   选中所有文本子节点

**@ATTR**  选中名为ATTR的属性节点

**@\***  选中所有的属性节点

**[谓语]**  谓语用来查找某个特定的节点或者包含某个特定值的节点

举例

　　response.xpath('/html/body/div') #选取body下的所有div

　　response.xpath('//a')  #选中文档所有a

　　response.xpath('/html/body//div') #选中body下的所有节点中的div，无论在什么位置

　　response.xpath('//a/text()') #选取所有a的文本

　　response.xpath('/html/div/\*') #选取div的所有元素子节点

　　response.xpath('//div/\*/img') #选取div孙节点的所有img

　　response.xpath('//img/@src') #选取所有img的src属性

　　response.xpath('//a[1]/img/@\*') #选取第一个a下img的所有属性

　　response.xpath('//a[2]') #所有a中的第2个

　　response.xpath('//a[last()]') #所有a中的最后一个  ('/a[last()-1]')#倒第二个   （'//a[position()<=3]'）#使用position函数选中前三个 （'//div[@id]'）#选中所有含有id属性的div  （'//div[@id="song"]'）#选中所有id属性为song的div

　　response.xpath('//p[contains(@class,'song')]') #选择class属性中含有‘song’的p元素

　　response.xpath('//div/a | //div/p') 或者，页面中可能是a可能是p

**css选择器**

　　css即层叠样式表，选择器不如xpath强大，原理就是选择的时候翻译成xpath表达式在调用xpath方法。

**\***     选择所有节点

**#container**        选择id为container的节点

**.container**         选择class包含container的节点

**li a**   选择 所有 li 下的所有 a 节点

**ul + p** 选择所有ul后面的第一个p元素

**#container > ul**    选择id为container的第一个ul节点

**a[class]**  选取所有有class属性的a元素

**a[href="http://b.com"]**   含有href="http://b.com"的a元素

**a[href\*='job']**  包含job的a元素

**a[href='^https']** 开头是https的a元素

**a[href$='cn']**结尾是cn的a元素

response.css('div a::text').extract()   所有div下所有a的文本

response.css('div a::attr(href)').extract()          href的值

response.css('div>a:nth-child(1)')  选中每个div的第一个a      > 会设定只在子节点中找，不会到孙节点中

response.css('div:not(#container)')  选取所有id不是container的div

response.css('div:first-child>a:last-child')  第一个div中最后一个a

python正则表达式

1. 正则表达式
2. ‘.’匹配任意一个字符
3. ‘\’转义字符,使后一个字符改变原来的意思
4. ‘\d’ 匹配字符串
5. ‘\D’ 匹配非字符串
6. ‘\s’ 匹配空白字符串
7. ‘\S’ 匹配非空字符串
8. ‘\w’ 匹配单个字符串
9. ‘\W’ 匹配非单个字符串
10. ‘\*’ 匹配前一个字符串一次货无数次
11. ‘+’ 匹配前一个字符串一次货无数次
12. ‘?’ 匹配前一个字符串I或0次
13. {n} 匹配前一个字符串n次
14. {n,m} 匹配前一个字符串n次到m次
15. ‘^’ 匹配字符串开头部分
16. ‘$’ 匹配字符串结尾部分
17. ‘\A’ 仅匹配字符串开头
18. ‘\Z’ 仅匹配字符串末尾
19. ‘|’ 表示左右表达式任意匹配一个
20. ‘()’ and的关系，分组表达式组委一个整体
21. ‘[]’ 或的关系,对应的位置可以匹配字符串的任意字符
22. ‘(?p<name>)’ 别名，除了原有的编号外，在指定一个额外的别名
23. ‘(?p=name)’ 引用别名为name的分组匹配到的字符串
24. ‘(?<=)’ 匹配字符串之前符合表达式规则的字符串
25. ‘(?=)’ 匹配字符串符合表达式的字符
26. ‘(?<!)’ 匹配字符串之前不是表达式规则的字符串
27. 相应的函数
28. Re.compile() 函数根据一个模式字符串和可选的标志参数生成一个正则表达式对象。该对象拥有一系列方法用于正则表达式匹配和替换
29. re.match() 尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match()就返回none。

A = re.compile(r”\d\*”,re.I)

B = A.match(str)

1. re.search() 扫描整个字符串并返回第一个成功的匹配。

Re.search(“([0-9]\*)([a-z]\*)([0-9]\*)”,str).group()

U=[http://k.autohome.com.cn/325/quality/02#pvareaid=#2073101](http://k.autohome.com.cn/325/quality/02#pvareaid=)

Re.search(“\d{2}(?=#)”,u).group()

1. findAll()在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果没有找到匹配的，则返回空列表。

p = re.compile(r'\d+')

print(p.findall('o1n2m3k4'))

执行如果如下：

['1','2','3','4']

import re

tt = "Tina is a good girl , she is cool ,clever, and so on ..."

rr = re.compile(r'\w\*oo\w\*')

print(rr.findall(tt))

print(re.findall(r'(\w)\*oo(\w)'tt)

1. finditer()函数，返回一个顺序访问每一个匹配结果的迭代，找到re匹配的所有子串，并把它们作为一个列表返回

iter = re.finditer(r’\d+’,’12 drumm44ers drumming,11.. 10..’)

for I in iter:

print(i)

print(i.group())

1. re.split() 分字符串

re.split(‘\d+’,’one1two2three4five5’)

1. re.sub() 替换字符串中每一个匹配的子串，返回替换后的字符串

text = ‘JGood is a handsome boy,he is coll,cover,and so on..’

re.sub(‘\s+’,’\_’,text)

1. re.subn() #返回替换此次序

同re.subn()

9) \*?,+?,??,{m,n}?    前面的\*,+,?等都是贪婪匹配，也就是尽可能匹配，后面加?号使其变成惰性匹配

python.xpath提取数据遇到的问题总结

1. 提取td标签的文本遇到问题：用xpath [@class=]/text()方法进行提取

BeautifulSoup抓取网页的内容

1. 获取对象 soup = BeautifuLSoup() type(soup)
2. Dir(soup) #获取soup的方法和属性
3. Soup.title #输出title标签
4. Soup.title.name #输出title名字
5. 获取标签的内容
   1. Soup.title.string #获取文本的内容
   2. Soup.title.get\_text()
   3. Soup.title.text
6. 通过上下级关系，获取对象
   1. soup.title.parent #获取父级对象
   2. Soup.p.children
7. 取货标签的属性
   1. a.attrs #获取所有的属性
   2. a.id
   3. a.get(‘id’)
   4. a[‘id’]
   5. a[‘class’]
   6. a.get(‘class’)
   7. a[‘href’]
   8. a.get(‘href’)
8. 获取标签,UI想
   1. Soup.find(‘p’)
   2. Soup.find\_all(‘P’) #返回list
   3. Soup.find(‘p’,{‘’:’’})
   4. Soup.find\_all(‘p’,{‘’:’’})

Python将数据进行保存

1. 将数据保存在json文件中

Open(‘demo.json’,’bw’) as file

Data = json.dump(response.text)

File.write(data)

1. 将数据保存在mongodb中

import pymongo

class MongoPipeline(object):

collection\_name = 'scrapy\_items'

def \_\_init\_\_(self, mongo\_uri, mongo\_db):

self.mongo\_uri = mongo\_uri

self.mongo\_db = mongo\_db

@classmethod

def from\_crawler(cls, crawler):

return cls(

mongo\_uri=crawler.settings.get('MONGO\_URI'),

mongo\_db=crawler.settings.get('MONGO\_DATABASE', 'items')

)

def open\_spider(self, spider):

self.client = pymongo.MongoClient(self.mongo\_uri)

self.db = self.client[self.mongo\_db]

def close\_spider(self, spider):

self.client.close()

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

self.db[self.collection\_name].insert(dict(item))

return item

1. 配置文件中注册ITEM\_PIPLINES
2. 请求和响应Request对象和Response对象
3. 传递参数的时候使用response.meta[]
4. 模拟登录认证进行爬取

import scrapy

class LoginSpider(scrapy.Spider):

name = 'example.com'

start\_urls = ['http://www.example.com/users/login.php']

def parse(self, response):

return scrapy.FormRequest.from\_response(

response,

formdata={'username': 'john', 'password': 'secret'},

callback=self.after\_login

)

def after\_login(self, response):

# check login succeed before going on

if "authentication failed" in response.body:

self.logger.error("Login failed")

return

# continue scraping with authenticated session...

Python内置的服务

1、日志的配置  
LOG\_ENABLED = true

LOG\_ENCODING = "utf-8"

LOG\_LEVEL = logging.INFO

LOG\_FILE = "log/spider.log"

LOG\_STDOUT = True

LOG\_FORMAT = "%(asctime)s [%(name)s] %(levelname)s: %(message)s"

LOG\_DATEFORMAT = "%Y-%m-%d %H:%M:%S"

2、发送email邮件，基于[Twisted non-blocking IO](http://twistedmatrix.com/documents/current/core/howto/defer-intro.html)实现，只需几个简单配置即可。

1）初始化 mailer = MailSender.from\_settings(settings)

2）发送不包含附件 mailer.send(to=["someone@example.com"], subject="Some subject", body="Some body", cc=["another@example.com"])

3）配置文件 MAIL\_FROM = 'scrapy@localhost'

MAIL\_HOST = 'localhost'

MAIL\_PORT = 25

MAIL\_USER = ""

MAIL\_PASS = ""

MAIL\_TLS = False

MAIL\_SSL = False

* 1. 同一个进程运行多个spider

import scrapy

from scrapy.crawler import CrawlerProcess

class MySpider1(scrapy.Spider):

# Your first spider definition

class MySpider2(scrapy.Spider):

# Your second spider definition

process = CrawlerProcess()

process.crawl(MySpider1)

process.crawl(MySpider2)

process.start() # the script will block here until all crawling jobs are finished

4、分布式爬虫 如果你有很多个spider，最简单的方式就是启动多个Scrapyd实例，然后将spider分布到各个机器上面。如果你想多个机器运行同一个spider，可以将url分片后交给每个机器上面的spider。比如你把URL分成3份

5、防止被封策略:

1) 使用user agent池。也就是每次发送的时候随机从池中选择不一样的浏览器头信息，防止暴露爬虫身份

2) 禁止Cookie，某些网站会通过Cookie识别用户身份，禁用后使得服务器无法识别爬虫轨迹

3) 设置download\_delay下载延迟，数字设置为5秒，越大越安全

4) 如果有可能的话尽量使用[Google cache](http://www.googleguide.com/cached_pages.html)获取网页，而不是直接访问

5) 使用一个轮转IP池，例如免费的[Tor project](https://www.torproject.org/)或者是付费的[ProxyMesh](http://proxymesh.com/)

6) 使用大型分布式下载器，这样就能完全避免被封了，只需要关注怎样解析页面就行。一个例子就是[Crawlera](http://scrapinghub.com/crawlera)

同时开启多个爬虫multi-scrapy

1在spiders 文件下建立多个蜘蛛(spiders)

匹配规则

rules = (Rule(LinkExtractor(allow=r"/tag/((\d+)|([\u4e00-\u9fa5]+)|(\w+))$")),  
 Rule(LinkExtractor(allow=r"/tag/((\d+)|([\u4e00-\u9fa5]+)|(\w+))\?start=\d+\&type=T$")),  
 Rule(LinkExtractor(allow=r"/subject/\d+/reviews$")),  
 Rule(LinkExtractor(allow=r"/subject/\d+/reviews\?start=\d+$")),  
 Rule(LinkExtractor(allow=r"/subject/\d+/$"), callback="parse\_video", follow=True),  
 Rule(LinkExtractor(allow=r"/review/\d+/$"), callback="parse\_review", follow=True),  
 )

2在items中建立多个实体数据模型

3 在piplines中进行存储的时候对其进行相应的判断执行相应的方法

Python scrapy部署

1 scrapyd开源方案

* + - 1. scrapy cloud方案

3、scrapyd 发布执行scrapy 工具

1）使用scrapyd-deploy 进行发布

安装发布工具 pip install scrapyd-client

Copy scrapyd-deploy工具到爬虫目录下

修改爬虫的scrapy.cfg文件 将url地址打开

deploy:127表示把爬虫发布到名为127的爬虫服务器上。

http://127.0.0.1:6800/daemonstatus.json

2、获取项目列表  
http://127.0.0.1:6800/listprojects.json

3、获取项目下已发布的爬虫列表

http://127.0.0.1:6800/listspiders.json?project=myproject

4、获取项目下已发布的爬虫版本列表

http://127.0.0.1:6800/listversions.json?project=myproject

5、获取爬虫运行状态

http://127.0.0.1:6800/listjobs.json?project=myproject  
 6、启动服务器上某一爬虫（必须是已发布到服务器的爬虫）  
http://localhost:6800/schedule.json （post方式，data={"project":myproject,"spider":myspider}）

7、删除某一版本爬虫

http://127.0.0.1:6800/delversion.json （post方式，data={"project":myproject,"version":myversion}）

8、删除某一工程，包括该工程下的各版本爬虫

http://127.0.0.1:6800/delproject.json（post方式，data={"project":myproject}）

发布爬虫

scrapyd-deploy <target> -p <project> --version <version>

企业级scrapy架构

<http://www.dataguru.cn/article-5327-1.html>

<http://baijiahao.baidu.com/s?id=1587199502093266832&wfr=spider&for=pc>

<https://juejin.im/entry/5acb0cbdf265da239e4e5f5c>

实现带验证码的网络爬虫

问题：Unknown column '1a' in 'field list'

Char型的数据需要加 ’”’括起来

问题：html类型数据保存数据库出现问题’\t\n’不能保存

item['tender\_content'] = "".join(content.split())

去掉内容中的’\t\n’

问题：pymysql.err.ProgrammingError: (1064,

使用pymysql.escape\_string()方法