# 使用Scrapy爬取笑话并存储到文件和MySQL

由于项目的需要，必须学习如何使用Scrapy来爬取数据。这篇博客以爬取笑话网的数据为例，说明Scrapy的基本使用。

配套的源码已经上传，可以从<http://download.csdn.net/detail/mrbcy/9764794>下载。

## 安装配置

我的系统是Win10 64位。因为Python3并不能完全支持Scrapy，因此为了完美运行Scrapy，我们使用Python2.7来编写和运行Scrapy。

基本的安装配置可以参看<http://blog.csdn.net/zengsl233/article/details/52166895>

在完成了上面的步骤后再执行一次pip install scrapy，然后再尝试执行一下scrapy，如果出现类似的输出，则下载成功。

Scrapy 1.3.2 - no active project

Usage:

scrapy <command> [options] [args]

Available commands:

bench Run quick benchmark test

commands

fetch Fetch a URL using the Scrapy downloader

genspider Generate new spider using pre-defined templates

runspider Run a self-contained spider (without creating a project)

settings Get settings values

shell Interactive scraping console

startproject Create new project

version Print Scrapy version

view Open URL in browser, as seen by Scrapy

[ more ] More commands available when run from project directory

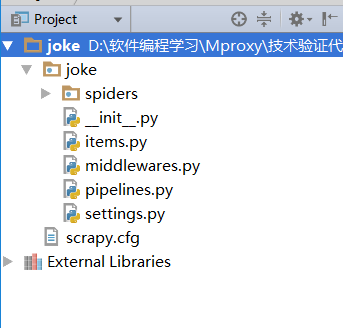
Use "scrapy <command> -h" to see more info about a command

## 创建项目

使用下面的命令创建一个项目：

scrapy startproject joke

创建好的项目结构如下图所示：



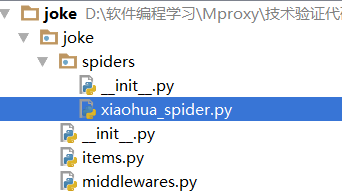
文件说明：

* 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**

## 编写爬虫

在spiders目录中新建 xiaohua\_spider.py 文件。



代码如下：

import scrapy

class XiaoHuarSpider(scrapy.spiders.Spider):

name = "xiaohua"

allowed\_domains = ["xiaohua.com"]

start\_urls = [

"http://xiaohua.com/Index/index/type/1.html",

]

def parse(self, response):

# print(response, type(response))

# from scrapy.http.response.html import HtmlResponse

# print(response.body\_as\_unicode())

current\_url = response.url # 爬取时请求的url

body = response.body # 返回的html

unicode\_body = response.body\_as\_unicode() # 返回的html unicode编码

几个注意要点：

1. 爬虫文件需要定义一个类，并继承scrapy.spiders.Spider
2. 必须定义name，即爬虫名，如果没有name，会报错。因为源码中是这样定义的：
3. 编写函数parse，这里需要注意的是，该函数名不能改变，因为Scrapy源码中默认callback函数的函数名就是parse；
4. 定义需要爬取的url，放在列表中，因为可以爬取多个url，Scrapy源码是一个For循环，从上到下爬取这些url，使用生成器迭代将url发送给下载器下载url的html。

## 运行爬虫

进入joke的目录，使用下面的命令可以启动爬虫。

scrapy crawl xiaohua

没有错误就可以进行下面一步了。

## 爬取起始页的笑话

Scrapy支持使用XPath提取页面的数据，还是非常方便的。关于XPath的教程可以参考：<http://www.w3school.com.cn/xpath/index.asp>

用Chrome分析页面结构，如下图所示。



可以看出使用//p[@class='fonts']/a/text()就可以拿到所有的笑话文本了。

代码如下：

import scrapy

from scrapy.selector import HtmlXPathSelector

class XiaoHuarSpider(scrapy.spiders.Spider):

name = "xiaohua"

allowed\_domains = ["xiaohua.com"]

start\_urls = [

"http://xiaohua.com/Index/index/type/1.html",

]

def parse(self, response):

hxs = HtmlXPathSelector(response)

items = hxs.select('''//p[@class='fonts']/a/text()''')

for item in items:

text = item.extract()

text = text.strip()

print(text)

然后使用下面的命令运行爬虫即可得到输出结果：

scrapy crawl xiaohua --nolog

输出结果如下：

和朋友去吃饭，一哥们喝醉了，跟服务员说：“你猜我用手能起开这瓶啤酒吗？”服务员笑了下，但是没说。那哥们又问：“ 你猜我用 手能起开这瓶啤酒不？”服务员说不信，那哥们说：“不信你TM还不去给我拿起子！”

早上起床，不爱搭理老婆。。。。。老婆跟我说话问我干啥呢！我说死了，又问:那怎么睁着眼睛？我说:死不瞑目！又问:那为啥还喘气 ？我说:咽不下这口气。。。。。。

今天看到的最感人朋友圈：一位大哥真诚地劝大家不要再吃转基因食品了！对孩子伤害很大！他孩子和他做亲子鉴定基因不匹配，就是因为孩子吃转基因食品把基因改变了。这些知识都是他老婆告诉他的…

有一室友特胆小，晚上一人在厕所大号，厕所没灯，我们几个拿着手电筒照着自己的鬼脸，一进厕所那货吓得鬼哭狼嚎的，突然抓起几把屎丢过来！大晚上他在洗手，我们哥几个在洗澡！

物理课上，我心不在焉的趴在桌上，一心却想着去网吧撸几把。这时，老师问：“搬运物体怎么样能省力？”有同学回答：“滚！”“回答正确！滚！”老师大声说…话音刚落，我习惯性从座位上站起来，默默地朝网吧飞奔而去了…身后，老师呐喊:锄小明！你给我回来…

去食堂吃饭，想打点热水回来洗澡，就把桶带上了。排队时，窗口阿姨拿着锅铲冲我这边吆喝：那小伙子，就你，别往后看，吃饭带饭盒可以，带饭桶不行！赶紧换一个！”我……

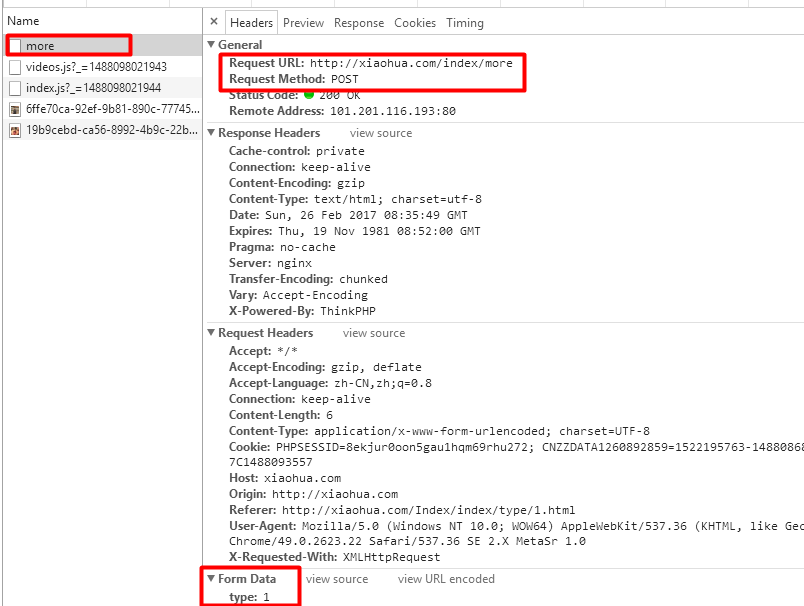
什么叫赌气、我妹买了一辆宝马X6天天嘚瑟，我就买了一辆兰博基尼！她一开我就撞！要不是卖家多送了两块大电池，还真撞不过！

忍不住看完了。。。

来，继续。

## 递归爬取网页

接下来我们需要完成的是继续爬取下一页的笑话。使用Chrome可以看到请求加载下一页笑话的地址是：http://xiaohua.com/index/more,请求方式是POST，还带了一个参数：type=1。如下图所示：



在解析完成之后只要使用下面的语句就可以递归爬取网页了。

yield Request(url, callback=self.parse)

看了官方的文档：<http://scrapy-chs.readthedocs.io/zh_CN/latest/topics/request-response.html>。因为我们要发送一个带参数的POST请求，使用FormRequest应该更好。

代码如下：

import scrapy

from scrapy import FormRequest

from scrapy.selector import HtmlXPathSelector

class XiaoHuarSpider(scrapy.spiders.Spider):

name = "xiaohua"

allowed\_domains = ["xiaohua.com"]

start\_urls = [

"http://xiaohua.com/Index/index/type/1.html",

]

def parse(self, response):

hxs = HtmlXPathSelector(response)

items = hxs.select('''//p[@class='fonts']/a/text()''')

for item in items:

text = item.extract()

text = text.strip() + '\n'

print(text)

# recursively get more jokes

more\_url = 'http://xiaohua.com/index/more'

return FormRequest(url=more\_url,

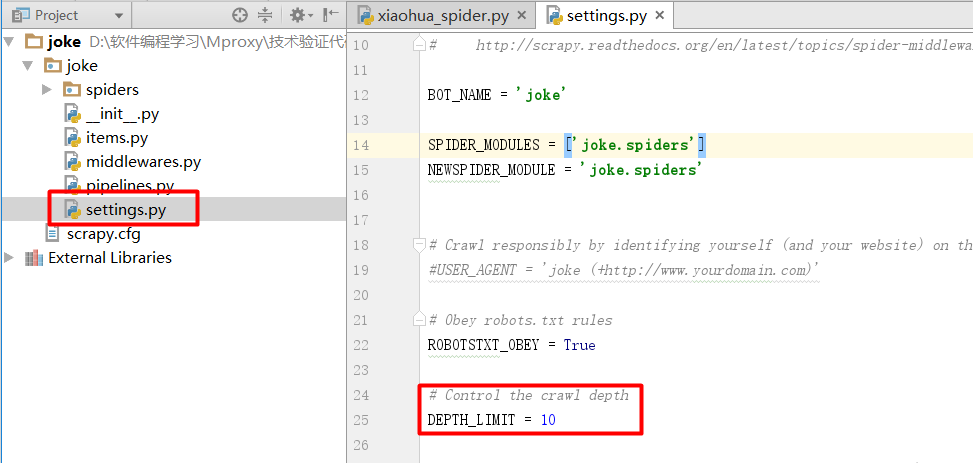
formdata={'type': '1'},

dont\_filter=True,

callback=self.parse)

然后还有一项工作，我们需要配置一个参数用于控制爬虫爬取的深度（就是加载more的次数）不然我们永远也结束不了了。

在settings.py文件中配置DEPTH\_LIMIT参数即可。如下图所示：

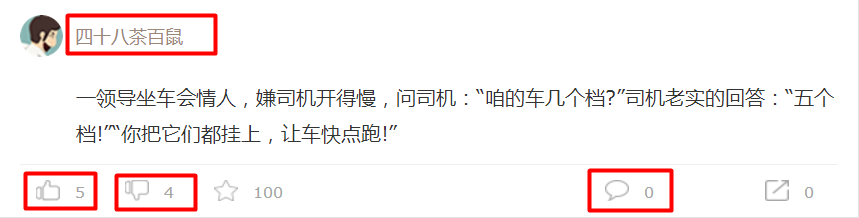


然后使用下面的命令运行爬虫即可：

scrapy crawl xiaohua --nolog

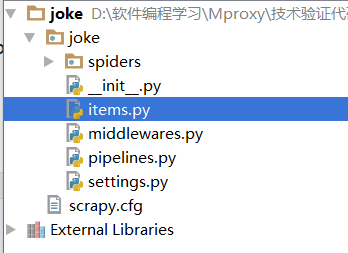
## 格式化处理

现在我们要做更多的事。之前我们拿到的只有笑话的文本数据，但是从页面上来看还有更多的内容可以拿。



如图所示，接下来可以获取发笑话的用户名，赞数，踩数，评论数。这样我们就可以拿到最好笑的笑话，或者说可以看那个用户发的笑话最受欢迎，重点的看他发的笑话。

修改items.py:



代码如下：

# -\*- coding: utf-8 -\*-

# Define here the models for your scraped items

#

# See documentation in:

# http://doc.scrapy.org/en/latest/topics/items.html

import scrapy

class JokeItem(scrapy.Item):

# define the fields for your item here like:

# name = scrapy.Field()

user\_name = scrapy.Field()

up\_vote\_num = scrapy.Field()

down\_vote\_num = scrapy.Field()

comment\_num = scrapy.Field()

joke\_text = scrapy.Field()

然后对Spider的代码进行一些修改：

import scrapy

from scrapy import FormRequest

from scrapy.selector import HtmlXPathSelector

from joke.items import JokeItem

class XiaoHuarSpider(scrapy.spiders.Spider):

name = "xiaohua"

allowed\_domains = ["xiaohua.com"]

start\_urls = [

"http://xiaohua.com/Index/index/type/1.html",

]

def parse(self, response):

items = response.xpath('''//div[@class='one-cont']''')

for item\_selector in items:

# go on to get elements

user\_name = item\_selector.xpath('''.//div[@class='one-cont-font clearfix']/i/text()''').extract\_first()

up\_vote\_num = item\_selector.xpath('''.//li[@class='active zan range']/span/text()''').extract\_first()

down\_vote\_num = item\_selector.xpath('''.//li[@class='range cai']/span/text()''').extract\_first()

comment\_num = item\_selector.xpath('''.//li[@class='range jxi']/span/text()''').extract\_first()

joke\_text = item\_selector.xpath('''.//p[@class='fonts']/a/text()''').extract\_first()

joke\_text = joke\_text.strip() + '\n'

joke\_id = item\_selector.xpath('''.//p[@class='fonts']/a/@href''').extract\_first()

joke\_id = joke\_id.replace('/Index/pinlun/id/','')

joke\_id = joke\_id.replace('/type/1.html','')

# data\_list = [joke\_id,user\_name,up\_vote\_num,down\_vote\_num,comment\_num]

# print('\t'.join(data\_list))

# print(joke\_text)

item = JokeItem()

item['joke\_id'] = joke\_id

item['user\_name'] = user\_name

item['up\_vote\_num'] = up\_vote\_num

item['down\_vote\_num'] = down\_vote\_num

item['comment\_num'] = comment\_num

item['joke\_text'] = joke\_text

yield item

# recursively get more jokes

more\_url = 'http://xiaohua.com/index/more'

yield FormRequest(url=more\_url,

dont\_filter=True,

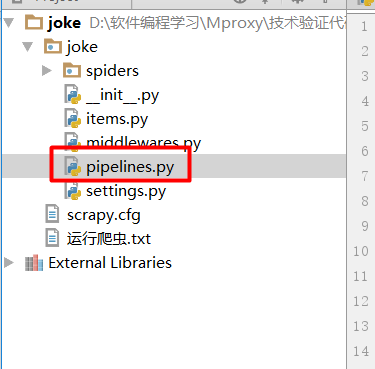
formdata={'type': '1'},

callback=self.parse)

这边使用xpath的方法也根据官方文档做了一些修改，可参考：<http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/selectors.html>

调用yield item之后Scrapy就会把Item交给pipelines的类来处理。

修改pipelines.py。



我们先把爬到的笑话存到一个文件里。代码如下：

# -\*- coding: utf-8 -\*-

# Define your item pipelines here

#

# Don't forget to add your pipeline to the ITEM\_PIPELINES setting

# See: http://doc.scrapy.org/en/latest/topics/item-pipeline.html

class JokePipeline(object):

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

return item

class JokeFilePipeLine(object):

def \_\_init\_\_(self):

self.file = open('d:/jokes', 'wb')

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

line = "%s\t%s\t%s\t%s\t%s\t%s\n" % (item['joke\_id'],

item['user\_name'],

item['up\_vote\_num'],

item['down\_vote\_num'],

item['comment\_num'],

item['joke\_text'])

self.file.write(line.encode("utf-8"))

return item

然后修改一下settings.py，指定使用我们新写的pipeline。

# Configure item pipelines

# See http://scrapy.readthedocs.org/en/latest/topics/item-pipeline.html

ITEM\_PIPELINES = {

'joke.pipelines.JokePipeline': None,

'joke.pipelines.JokeFilePipeLine': 300

}

后面的None是指不使用这个pipeline，300是一个优先级，数字越大，优先级越高。

运行后即可在文件中看到爬取的笑话。



## 将笑话写入MySQL

### 配置Python和MySQL的连接

详细步骤参看<http://www.cnblogs.com/orchid/archive/2013/03/26/2982037.html>

### 创建表和存储过程

使用下面的代码创建表：

CREATE TABLE jokes(

id VARCHAR(255) PRIMARY KEY,

userName VARCHAR(255),

upVoteNum INT,

downVoteNum INT,

commentNum INT,

jokeText TEXT)

使用下面的代码创建储存过程：

DELIMITER $$

CREATE

PROCEDURE `jokedb`.`addJoke`(IN jokeId VARCHAR(255),IN userName VARCHAR(255),IN upVoteNum INT,IN downVoteNum INT,IN commentNum INT,IN jokeText TEXT)

BEGIN

SET @jokeId = jokeId;

SET @upVoteNum = upVoteNum;

SET @downVoteNum = downVoteNum;

SET @commentNum = commentNum;

SET @jokeText = jokeText;

SET @userName = userName;

SET @existsFlag='';

SELECT id INTO @existsFlag FROM jokes WHERE id = @jokeId LIMIT 1;

IF @existsFlag = '' THEN

SET @insertSql = CONCAT('INSERT INTO jokes VALUES(?,?,?,?,?,?)');

PREPARE stmtinsert FROM @insertSql;

EXECUTE stmtinsert USING @jokeId,@userName,@upVoteNum,@downVoteNum,@commentNum,@jokeText;

DEALLOCATE PREPARE stmtinsert;

END IF;

END$$

DELIMITER ;

### 调用MySQL的存储过程测试代码

# -\*- coding: utf-8 -\*-

import MySQLdb

conn = MySQLdb.connect(host='localhost',user='root',passwd='sorry',db='jokedb',charset="utf8")

cur =conn.cursor()

cur.callproc('addJoke',('1000','张三',0,100,100,'测试啊啊啊啊'))

cur.close()

conn.commit()

conn.close()

这样虽然可以实现写入MySQL，但是没运行一次都要重新建立连接，非常耗费资源。一个比较好的方法是使用数据库连接池。

使用下面的代码来安装DBUtils:

pip install DBUtils

然后修改代码：

# -\*- coding: utf-8 -\*-

import MySQLdb

from DBUtils.PooledDB import PooledDB

pool = PooledDB(MySQLdb,5,host='localhost',user='root',passwd='sorry',db='jokedb',port=3306,charset="utf8") #5为连接池里的最少连接数

conn = pool.connection()

cur =conn.cursor()

cur.callproc('addJoke',('1000','张三',0,100,100,'测试啊啊啊啊'))

cur.close()

conn.commit()

conn.close()

### 编写pipelines代码

class JokeMySqlPipeLine(object):

def \_\_init\_\_(self):

self.pool = PooledDB(MySQLdb,5,host='localhost',user='root',passwd='sorry',db='jokedb',port=3306,charset="utf8")

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

conn = self.pool.connection()

cur = conn.cursor()

cur.callproc('addJoke', (item['joke\_id'], item['user\_name'], int(item['up\_vote\_num']),

int(item['down\_vote\_num']), int(item['comment\_num']), item['joke\_text']))

cur.close()

conn.commit()

conn.close()

然后把这个pipelines配上去。

# Configure item pipelines

# See http://scrapy.readthedocs.org/en/latest/topics/item-pipeline.html

ITEM\_PIPELINES = {

'joke.pipelines.JokePipeline': None,

'joke.pipelines.JokeFilePipeLine': None,

'joke.pipelines.JokeMySqlPipeLine': 300

}

运行爬虫，即可在MySQL中看到爬取到的笑话：



## 小优化

这一部分内容来自<http://blog.csdn.net/u012150179/article/details/35774323>

### download\_delay

设置这个值可以使爬虫在爬完一个页面后停一下，避免太过集中的访问被服务器封掉。

可以在settings.py中设置，也可以在Spider中设置，我在Spider中设置了。

class XiaoHuarSpider(scrapy.spiders.Spider):

name = "xiaohua"

allowed\_domains = ["xiaohua.com"]

download\_delay = 2

......

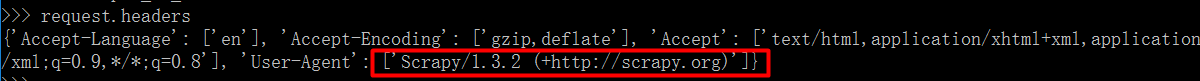
### 使用user agent池

所谓的user agent，是指包含浏览器信息、操作系统信息等的一个字符串，也称之为一种特殊的网络协议。服务器通过它判断当前访问对象是浏览器、邮件客户端还是网络爬虫。在request.headers可以查看user agent。如下，使用scrapy shell查看：

scrapy shell http://blog.csdn.net/u012150179/article/details/34486677

进而输入如下，可得到uesr agent信息：

request.headers



由此得到,scrapy本身是使用Scrapy/1.3.2来表明自己身份的。这也就暴露了自己是爬虫的信息。

首先编写自己的UserAgentMiddle中间件，新建rotate\_useragent.py,代码如下：

建立user agent池（user\_agent\_list）并在每次发送request之前从agent池中随机选取一项设置request的User\_Agent。编写的UserAgent中间件的基类为UserAgentMiddle。

# -\*-coding:utf-8-\*-

from scrapy import log

"""避免被ban策略之一：使用useragent池。

使用注意：需在settings.py中进行相应的设置。

"""

import random

from scrapy.contrib.downloadermiddleware.useragent import UserAgentMiddleware

class RotateUserAgentMiddleware(UserAgentMiddleware):

def \_\_init\_\_(self, user\_agent=''):

self.user\_agent = user\_agent

def process\_request(self, request, spider):

ua = random.choice(self.user\_agent\_list)

if ua:

# Show current useragent

print "\*\*\*\*\*\*\*\*Current UserAgent:%s\*\*\*\*\*\*\*\*\*\*\*\*" %ua

# do the log

# log.msg('Current UserAgent: '+ua, level='INFO')

request.headers.setdefault('User-Agent', ua)

# the default user\_agent\_list composes chrome,I E,firefox,Mozilla,opera,netscape

# for more user agent strings,you can find it in http://www.useragentstring.com/pages/useragentstring.php

user\_agent\_list = [

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 "

"(KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1",

"Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 "

"(KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.6 "

"(KHTML, like Gecko) Chrome/20.0.1092.0 Safari/536.6",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 "

"(KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6",

"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.1 "

"(KHTML, like Gecko) Chrome/19.77.34.5 Safari/537.1",

"Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/536.5 "

"(KHTML, like Gecko) Chrome/19.0.1084.9 Safari/536.5",

"Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536.5 "

"(KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",

"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",

"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_8\_0) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 "

"(KHTML, like Gecko) Chrome/19.0.1061.0 Safari/536.3",

"Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/535.24 "

"(KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24",

"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.24 "

"(KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24"

]

除此之外，要在settings.py(配置文件)中禁用默认的useragent并启用重新实现的User Agent。配置方法如下：

#取消默认的useragent,使用新的useragent

DOWNLOADER\_MIDDLEWARES = {

'scrapy.contrib.downloadermiddleware.useragent.UserAgentMiddleware' : None,

'joke.rotate\_useragent.RotateUserAgentMiddleware' :400

}

## 总结

算是初步的体验了一下Scrapy，感觉还是很强大的。这篇博客也为项目打下了一定的基础。后续还要再好好学一下Scrapy。

## 参考资料

<http://python.jobbole.com/86405/>

<http://blog.csdn.net/u012150179/article/details/35774323>