目录

[1. 编码问题 2](#_Toc514682784)

[2. 2](#_Toc514682785)

[3. 读写文件熟练 3](#_Toc514682786)

[4. 爬虫 4](#_Toc514682787)

[1.1 网络库 4](#_Toc514682788)

[1.2 解析库 4](#_Toc514682789)

[1.2.1 正则 4](#_Toc514682790)

[1.2.2 XPath的使用 4](#_Toc514682791)

[1.2.3 BeautifulSoup的使用 4](#_Toc514682792)

[1.3 数据库 5](#_Toc514682793)

[1.3.1 MongoDB 5](#_Toc514682794)

[1.3.2 Redis 5](#_Toc514682795)

[1.4 分析Ajax请求 5](#_Toc514682796)

[1.5 框架 5](#_Toc514682797)

[1.6 分布式 7](#_Toc514682798)

[5. 数据库熟练 8](#_Toc514682799)

[6. 机器学习推导，优劣势 8](#_Toc514682800)

# 编码问题

读进IDE用decode(‘utf-8’),写出去用encode(‘utf-8’)

1.1 文本总是unicode,由str类型表示；二进制数据由bytes类型表示

1.2 文件编码：

Python 3的字符是以bytes形式存储，也就是说如果要将字符写入硬盘，就必须对字符进行encode

将str写入文件

* 如果以‘w’模式写入，则要求写入的内容必须是str类型

open(filename,’w’,encoding=’utf-8’)

* 如果以‘wb’形式写入，则要求写入的内容必须是bytes类型。

f=open(filename,'wb')

f.write(text.encode()) #将str类型的text编码成bytes类型，才能完美写入

1.3网络编码

* 网页编码和文件编码方法差不多，如下urlopen下载下来的网页read()且用decoding(‘utf-8’)解码，那就必须以‘w’的方式写入文件。
* 如果只是read()而不用encoding(‘utf-8’)进行编码，一定要以‘wb’方式写入

爬虫 'utf-8' codec can't decode byte 0xbb in position 1054: invalid start byte

如果python中所要处理的字符串中包含中文，那么最好要搞懂所用字符的编码，是gbk/gb2312/gb18030，还是utf-8，否则容易出现乱码，以及此处的语法错误。逐个试。

# 读写文件熟练

# 网络库

requests的使用

req = request.get(url)

str = req.text # 返回字符串文本，可能是html形式,也可能是json形式

如果是HTML格式，用XPath或者其他都行

如果是json格式，用正则处理

# 解析库

## 正则

如果我们有一个字符串”a123b456b”，如果我们想匹配a和最后一个b之间的所有值而非a和第一个出现的b之间的值，可以用?来控制正则贪婪和非贪婪匹配的情况。

可以使用findall配合正则表达式

代码如下：

import re

str = "a123b456b"

pattern = re.compile() #正则,可以添加一个属性re.S，匹配任意字符，包括换行符。可以同时匹配不同的字符，贪婪模式加括号或不加。最终返回一个list，里面是元组。如：</a>.\*?star”>(.\*?)</p>.\*?resfda>(.\*?) ,这里同时匹配了两个，组成元组返回。

* print re.findall(r"a(.+?)b", str)

等价于re.findall(pattern,str)

#输出['123']#?控制只匹配0或1个,所以只会输出和最近的b之间的匹配情况

* print re.findall(r"a(.+)b", str)

#输出['123b456']

* print re.findall(r"a(.\*)b", str)

#输出['123b456']

## XPath的使用

1.按层查找（定位）：/ 从头开始如：/html/head/title

2.提取标签中的文本内容：/逐层标签名/text()

3.提取标签中的属性的值:@ 如：/a/@class

4.寻找所有的标签：//

5.定位：标签[@] 如：//li[@class='dfs']

## BeautifulSoup的使用

html=urlopen(url)

bsObj=BeatuifulSoup(html,'lxml')

find\_all(name,attrs,recursive,text,\*\*kwargs)

可以根据标签名，属性，内容查找文档

1.寻找标签所有信息：bsObj.find\_all('div')

2.提取查到的所有的text='指定值'的文本：find\_all(text='abc')

3.提取指定属性的值所在标签的所有信息:find\_all(attrs={'id':'dfg'})

4.提取文本 bsObj.string

当find\_ll返回一个列表a时，用for i in a: print(i.string)

而不用for i in range(0,len(a)): print(a.string) 很奇怪的...前者还是bsObj，后者只是字符串了，字符串没有string方法

解析json

import json

jsobj = json.loads(response.body)

comment = jsobj['comment']

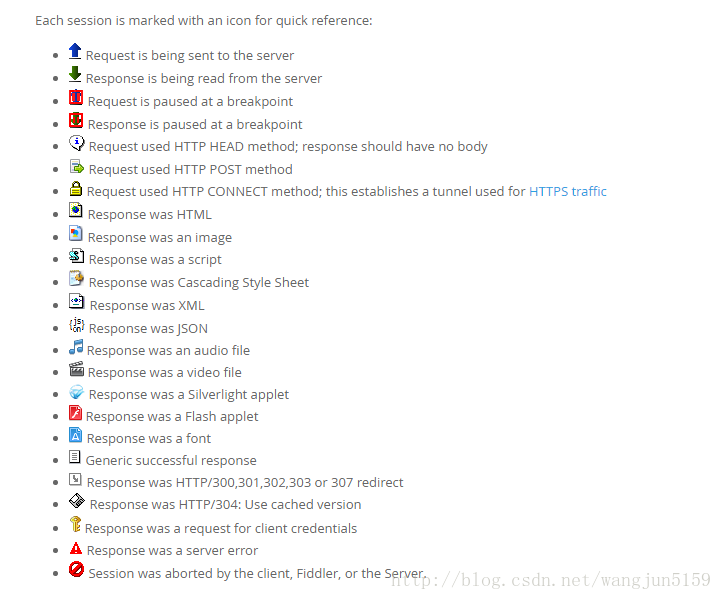
**print**(comment)

# 常用方法

## Fiddler

### 图标含义

注意post请求的图标，下图第6个。此时要构造post请求头。先在fiddler构造，再写入python。

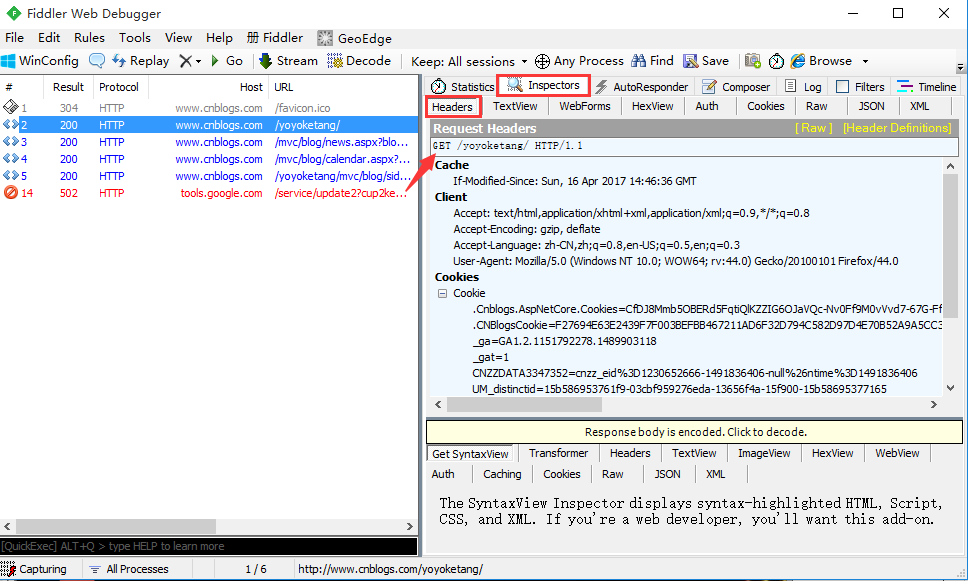


### get请求

1.打开fiddler工具，然后浏览器输入博客首页地址：http://www.cnblogs.com/yoyoketang/

2.点开右侧Inspectors下的Headers区域，查看Request Headers

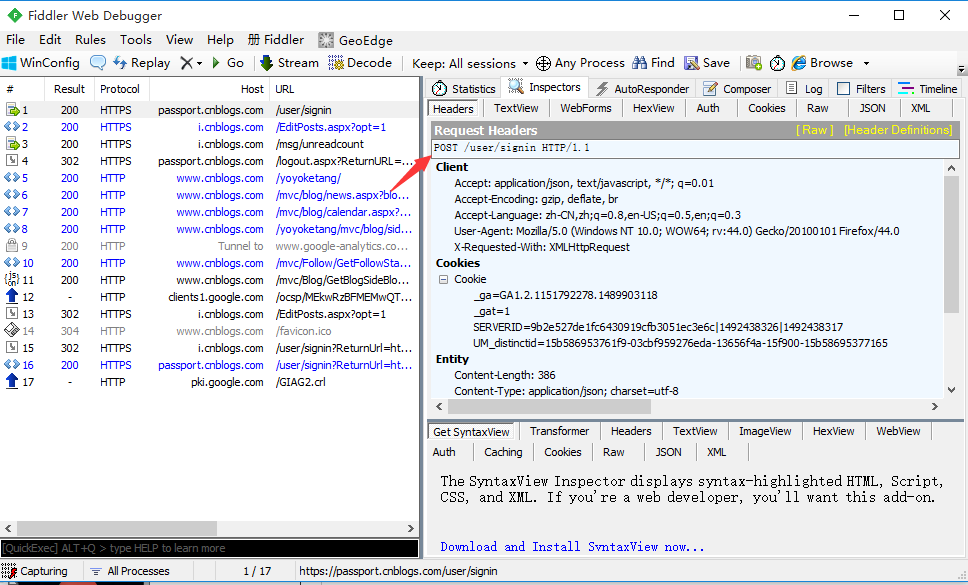
3.Request Headers区域里面的就是请求头信息，可以看到打开博客园首页的是get请求



### post请求

1.打开登录首页：https://passport.cnblogs.com/user/signin

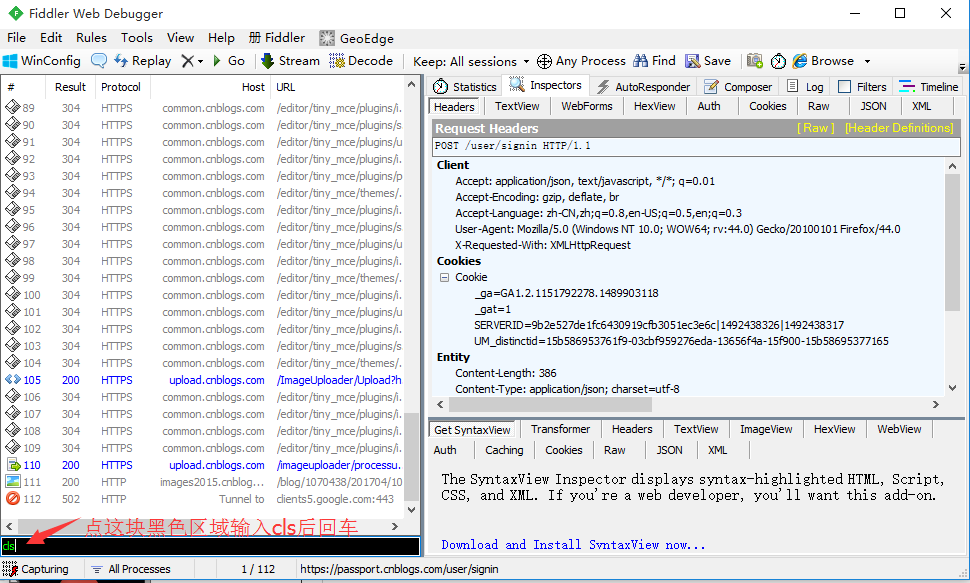
2.输入账号和密码登录成功后，查看fiddler抓包的请求头信息，可以看出是post请求



### 如何找出需要的请求

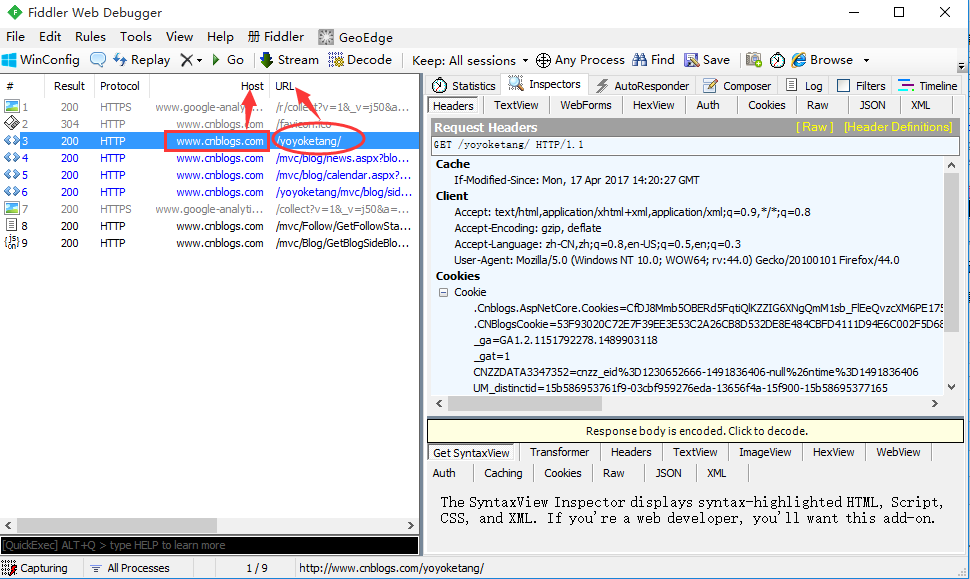
1.打开fiddler后，左边会话框区域刷刷刷的很多请求，那么如何有效的找出自己需要的请求呢？

2.首先第一步：清屏（cls）,在左下角命令行输入cls，清空屏幕



3.第二步在浏览器输入url地址的时候，记住这个地址，如打开博客首页：http://www.cnblogs.com/yoyoketang/

在点击登录按钮的时候，不要做多与的操作了，然后查看fiddler会话框，这时候有好几个请求。



4.如上图，红色框框这个地方就是host地址，红色圈圈地方就是url的路径（yoyoketang），也就是博客首页的地址了，那这个请求就是博客首页的请求了。

### get和post请求参数区别

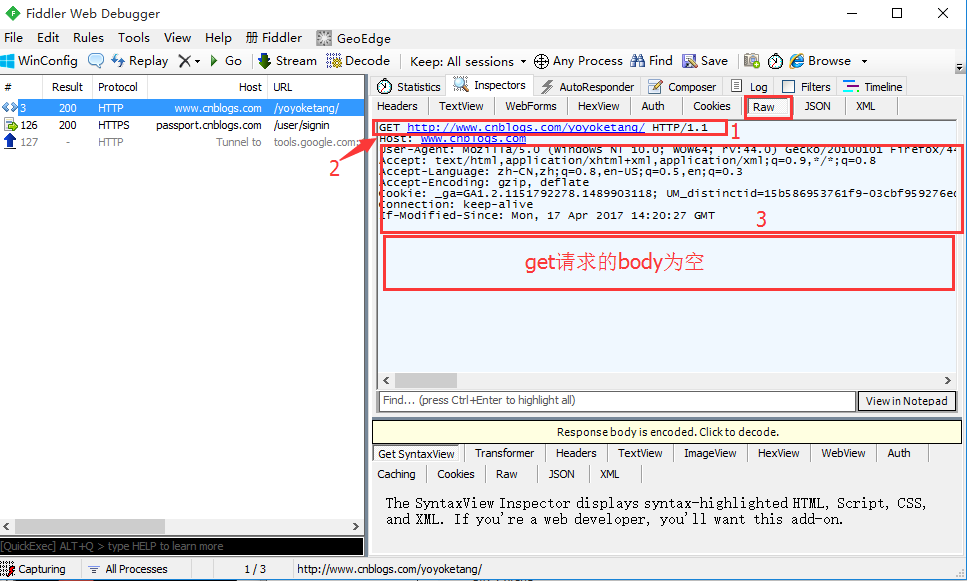
1.关于get和post的功能上区别就不说了，大家自己查资料，这里主要从fiddler抓包的层面查看请求参数上的区别

2.get请求的Raw参数查看，主要分三部分：

--第1部分是请求url地址

--第2部分是host地址

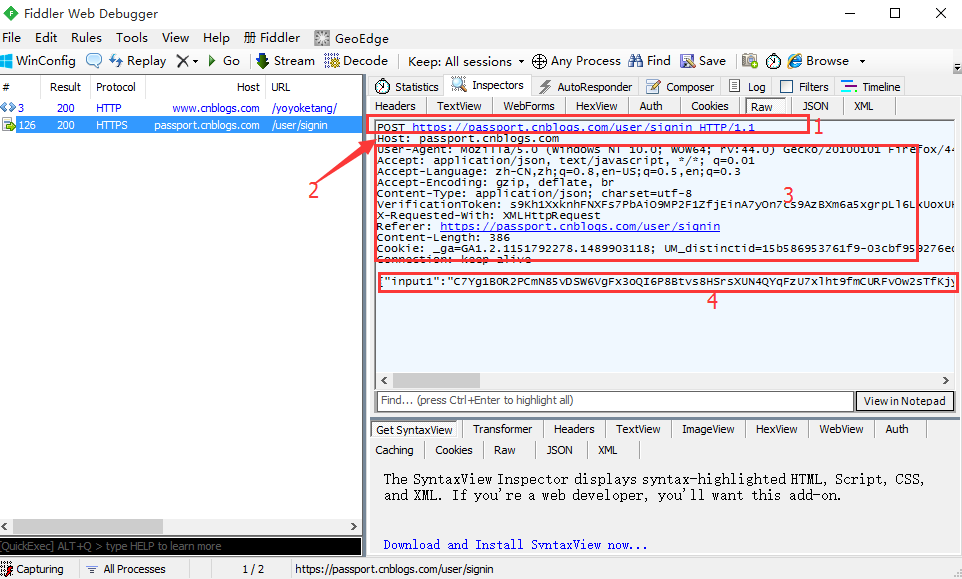
--第3部分是请求头部信息header



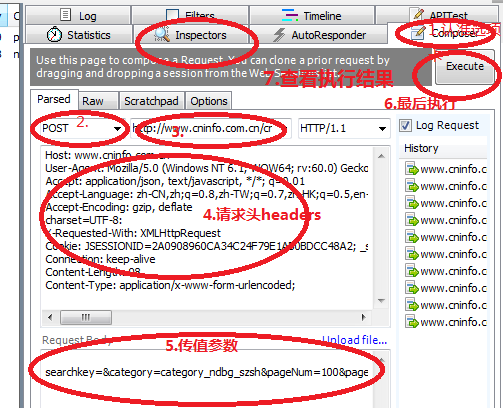
3.再查看博客登录请求的Raw信息，post的信息分四部分。

--前面3块内容都一样，第3部分和第4部分中间会空一行

--第4部分内容就是post请求的请求body（get请求是没body的）



### 模拟post请求



第4部分来自5.1.5的第3部分

第5部分来自5.1.5的第4部分

## 请求头（模拟浏览器）

这部分也可以写入scrapy的setting中，那么接下来的构造请求中就不用写这个参数了。看着用，都行。

**这部分直接复制粘贴上图fiddler的第4部分即可。**

headers = {  
 **'Host'**: **'www.cninfo.com.cn'**,  
 **'User-Agent'**: **'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:60.0) Gecko/20100101 Firefox/60.0'**,  
 **'Accept'**: **'application/json, text/javascript, \*/\*; q=0.01'**,  
 **'Accept-Language'**: **'zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2'**,  
 **'Accept-Encoding'**: **'gzip, deflate'**,  
 **'Referer'**: **' http://www.cninfo.com.cn/cninfo-new/disclosure/szse/showFulltext'**,  
 **'charset'**: **'UTF-8'**,  
 **'X-Requested-With'**: **'XMLHttpRequest'**,  
 **'Cookie'**: **'JSESSIONID=2A0908960CA34C24F79E1AB0BDCC48A2; \_sp\_id.2141=c39867ab-5152-4409-811e-20caebb96dd6.1528111186.3.1528153492.1528127691.c9dc9ea4-97bc-4db5-9662-c40e16fa7209; cninfo\_search\_record\_cookie=000006|%E6%B7%B1%E6%8C%AF%E4%B8%9A%EF%BC%A1|%E6%B7%B1%E5%BA%B7%E4%BD%B3%EF%BC%A1'**,  
 **'Connection'**: **'keep-alive'**,  
 **'Content-Length'**: **'98'**,  
 **'Content-Type'**: **'application/x-www-form-urlencoded'**}

## post请求

使用 **scrapy.FormRequest()**构造。在参数formdata{}里写传递的值，其中的数字都要用’’包起来，当然如例子中的**'stock'**: StkcdSet[i]传值方式也可以。

**这部分来自上图fiddler的第5部分，注意表达格式转换**

**yield** scrapy.FormRequest(  
 url = **'http://www.cninfo.com.cn/cninfo-new/announcement/query'**,  
 formdata={  
 **'stock'**: StkcdSet[i],  
 **'category'**: **'category\_ndbg\_szsh'**,  
 **'pageNum'**: **'1'**,  
 **'pageSize'**: **'15'**,  
 **'column'**: **'szse\_main'**,  
 **'tabName'**: **'fulltext'** },  
 callback = self.parse,  
 headers=headers  
)

## get请求

Request(url=,headers=,callback=)

## cookice

# 数据库

## Mysql

python与mysql的坑

1.python连接时，必须指定编码，在pymysql.connect（...,use\_unicode=True, charset="utf8"）

2.mysql中必须指定编码,set names utf8

3.mysql的表类型必须为myisam，不然写不进去。使用命令转换：alter table tablename type=innodb;

4.可以用 show create table tablename 命令看表的类型。

5.关闭数据库，conn.close()必须放在循环外！

6.最最关键的是：别写错变量名,尤其是 values T\_T

7. 记住values的循环写入写法，比较不好理解

sql ="insert into dangdang(title,litle,comment) values('"+title+"','"+link+"','"+comment+"')"

## MongoDB

安装pymongo 库

mangodb已经是一个服务了，不需要过多关注实现细节，关注使用就好。

打开mangodb可视化界面 Robo 3T 即可使用（D盘绿色虫子图标）

在爬虫中使用mangodb

1.在项目目录下新建一个config.py(文件名随意)

写入配置信息:

MONGO\_URL='localhost' # 链接地址

MONGO\_DB = 'xxx' # 数据库名

MONGO\_TABLE = 'xxx' # 表名

写完了

2.回到项目文件

导入刚刚写的配置文件及pymango包

imort pymongo

from config import \*

client = pymongo.MongoClient(MONGO\_URL) # 申明一个mongodb对象，链接到mangodb

db = client[MONGO\_DB] # 使用数据库

这样就可以使用了

3.使用方法,传入一个字典

db[MONGO\_TABLE].insert(dict) # 返回bool值，如果插入成功，既插入数据库，也返回True

## Redis

打开Redis可视化界面 RedisDeskTop （D盘红色盒子图标）

设置了name:localhost Host:localhost 没有设置密码

请看IP池

# 分析Ajax请求

# scrapy框架

## 创建Scrapy（可以爬https的！！！）

1.创建Scrapy工程项目 cmd--f:/.../--输入scrapy startproject ScrapyTest（这是文件名），等待一会

* 2.在1)下创建爬虫

先cd ScrapyTest（这是文件名）

再scrapy genspider -t basic name(爬虫名) 爬取的域名baidu.com(不要http://www)

或者

scrapy genspider -t crawl name(爬虫名) 爬取的域名baidu.com(不要http://www)

两者区别在于basic是普通爬虫模版，crawl是自动爬虫模版

1.先编辑item.py文件，主要用来设置爬取的目标

2.编辑pipelines.py文件，用来后续的处理

3.settings.py文件，用来配置信息

4.最后再编写爬虫（在spiders文件夹下的非\_\_int\_\_.py文件的文件里）

* items.py的处理

1.用content=scrapy.Field()替换pass，可以有多个不同的容器

2.content是容器的名字，随便给。在pipelines.py文件中会用到这个名字

3.scrapy.Field()是固定写法

* pipelines.py文件的处理

1. 直接在def中写处理就可以了，其他的都不用改
2. 处理item属性：视为列表，值为items里面定义的容器名字，使用如：item['content']
3. 在这里写入数据库。也在这里控制录入数据库的长度
4. import pymysql

conn = pymysql.connect(host,user,passwd,db,use\_unicode=True,charset=’utf-8’)

sql=str....

conn.query(sql)

conn.close()

* settings.py文件处理

1.开启pipelines.py文件，去掉67~69的注释，修改第68行xxx.pipelines.XXX中的XXX为pipelines.py文件的def的函数名

2.22行，True改为False

3.19行，设置模拟浏览器的请求头

* 爬虫文件的处理（basic模版）

1. 导入items.py文件中的方法,from items所在文件夹名.items import 方法名如：FirstItem
2. 在def parse()中写
3. 先创建FirstItem对象：item=FirstItem().然后就可以使用在FirstItem定义的属性了，如：item['content']作为容器
4. 先运行start\_url,然后返回一个response()【即我们不用管请求了】
5. 运行后爬取的内容保存在response()中，所以可以直接使用item['content']=response.xpath("").extract()进行提取信息并拿出来
6. extract()与extract\_first()类似find与findall()
7. 使用yield来返回，而不用return。固定写yield item，然后在pipelines.py文件就可以使用item了
8. **翻页**设置，同样在parse()函数里写。构造需要翻页的url（可以用for来循环控制），然后 yield Request(url=url,callback=parse()),构成一个递归。
9. 保存item,命令行输入 scrapy crawl 爬虫名 -o 爬虫名.json 这样运行完好就会在父目录下保存一个item的json文件，或csv文件
10. 模拟浏览器from scrapy.http import Request

先注释掉start\_urls

再写下面的代码

def start\_reqeust(self):

ua = {“User-Agent”:...}

yield Request(start\_url(首次爬取的地址),header = ua)

1. 使用 def start\_requests(self):进行首次爬取
2. 在7）下使用 header={"User-Agent":浏览器头}
3. yield Request('url',headers=header)

* 自动化爬虫文件的处理（crawl模版）

1.导入items.py文件中的方法,from items所在文件夹名.items import 方法名如：FirstItem【如果创建时含有了QsautoItem,那就不用导入了，因为已经导入了】

2.与basic不同的地方。rules的Rule中，allow=r"xxx"写链接提取的规律，只需写需要爬取页面的url中共有的内容（不含域名）callback为回调函数（一般为parse\_item）

2.在def parse\_item()中写

3.创建FirstItem【QsautoItem】对象：i=FirstItem【QsautoItem】.然后就可以使用在FirstItem定义的属性了，如：i['content']作为容器

3.运行后爬取的内容保存在response()中，所以可以直接使用i['content']=response.xpath("").extract()进行提取信息并拿出来

4.使用yield来返回，而不用return。写yield i，然后在pipelines.py文件就可以使用item了

5.模拟浏览器from scrapy.http import Request

6.注释掉start\_urls

7.使用 def start\_requests(self):进行首次爬取

8.在7）下使用 header={"User-Agent":浏览器头}

yield Request('url',headers=header)

## 运行Scrapy

scrapy crawl 爬虫名 --nolog

* 用yield格式化返回结果

# 分布式

## Scrapy-Redis分布式搭建

Scrapy-Redis分布式框架库：<https://github.com/rmax/scrapy-redis> 就可以完成分布式搭建了

使用时修改以下设置setting:

ITEM\_PIPELINES = {

'scrapy\_redis.pipelines.RedisPipeline': 300

}

# Enables scheduling storing requests queue in redis.

SCHEDULER = "scrapy\_redis.scheduler.Scheduler"

# Ensure all spiders share same duplicates filter through redis.

DUPEFILTER\_CLASS = "scrapy\_redis.dupefilter.RFPDupeFilter"

## IP池

先在后台开启代理池 python run.py

再写获取代理的方法：

proxy = None # 使用全局变量。如果不用，会不停地传代理，很繁琐。直接用到被封，再换

先声明代理的url,因为代理的ip显示在网页上：

proxy\_pool\_url = 'http://127.0.0.1:5000/get'# 一般是这个

def get\_proxy():

try:

response = requests.get(proxy\_pool\_url)

if response.status\_code =='200':

return response.text

return None

except ConnectionError:

return

max\_coount=5

def get\_html(url,count=1): # countq请求次数

global proxy

try:

if proxy: #判断是否用代理

proxies = {

'http':'http://'+proxy

}

response = requests.get(url,all\_redirects=Flase,headers=headers,proxies = proxies)

else:

response = requests.get(url,all\_redirects=Flase,headers=headers)

if response.status\_code =='200':

return response.text

if response.status\_code =='302':

proxy=get\_proxy()

if proxy:

return get\_html(url)

else:

return None

except ConnectionError as e:

print('Error ',e.args)

proxy=get\_proxy()

count+=1

return get\_html(url,count)

## Cookies池

# 反爬处理

遇到302状态码，证明ip被封，切换代理

先后台运行代理池，再写函数，通过代理池提供的接口获取代理