**实验目的**

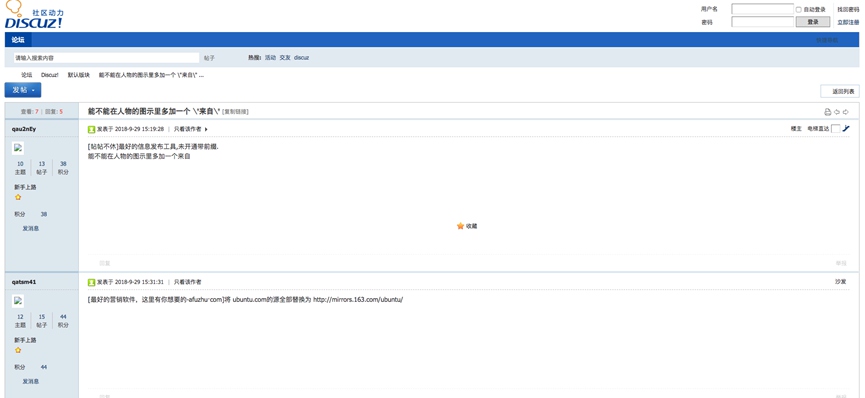
* 要求使用的Python语言编写爬虫代码;
* 要求爬取给定网站的所有发帖数据，包含回帖数据;

**实验目的**

　　discuz的是目前国内知名的开源PHP的社交系统它的基础架构采用PHP + MySQL的实现;！。适用于各种服务器环境的高效论坛系统  
　　。直接访问目标站点的IP即可进入论坛主页论坛的默认模块包含5800 +条主题帖及1700+条回复帖，共计7500+条有效回复内容;包含550+会员。  
　　其中涉及到的信息包含：论坛版块，发帖人，回帖人，发帖人ID，发帖人名称，回帖人ID，回帖人名称，用户头像，发帖内容，回帖内容，发帖ID，回帖ID等。  
　　可以使用操作机直接进行访问其IP：



　　其网站内容如下：



上一步

* Python 3.5.2
* Discuz论坛

**爬虫简介**

* 爬虫简介  
  1.爬虫是以代码的形式去模拟用户使用浏览器去访问网页的URL，并获取网页的所有URL，然后再解析出网页中所有我们需要的内容：



2.如果我们需要爬取的是一个父URL下的所有子URL，我们就需要将父网址下所有的子URL存储为URL列表然后让我们的程序去反复的访问：



* 爬虫分类

**PHP爬虫：** 代码简单，并发处理能力较弱：由于当时PHP没有线程，进程功能，要想实现并发需要借用多路服用模型。

**ř语言爬虫：** 操作简单，功能很差，只适用于小规模的爬取，不适合大规模的爬虫。

**蟒爬虫：** 有着各种爬虫框架，如（Scrapy），方便高效的下载网页;并且支持多线程，进程模型成熟稳定，爬虫是一个典型的多任务处理场景，请求页面时会有较长的延迟，总体来说更多的是等待。多线程或进程会更优化程序效率，提升整个系统下载和分析能力。

**Java的爬虫：** 语言笨重，所建立的数据模型的任何修改都会导致代码大量变动。

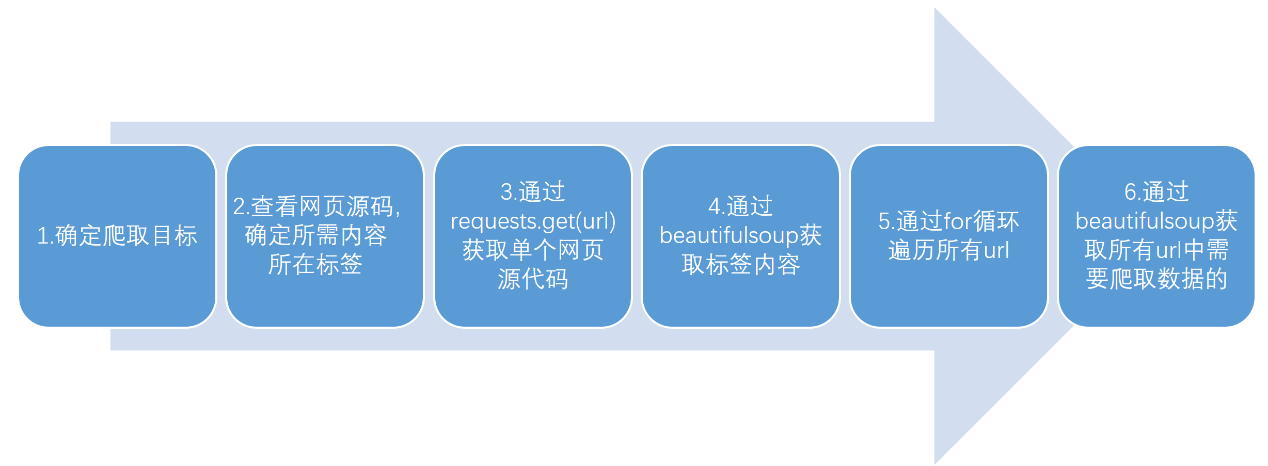
所以我们选择使用更便捷功能更强大的Pyton爬虫。

上一步

**爬虫简介**

* 若想爬取一个网站中的数据我们需要做以下准备工作，如下图所示：

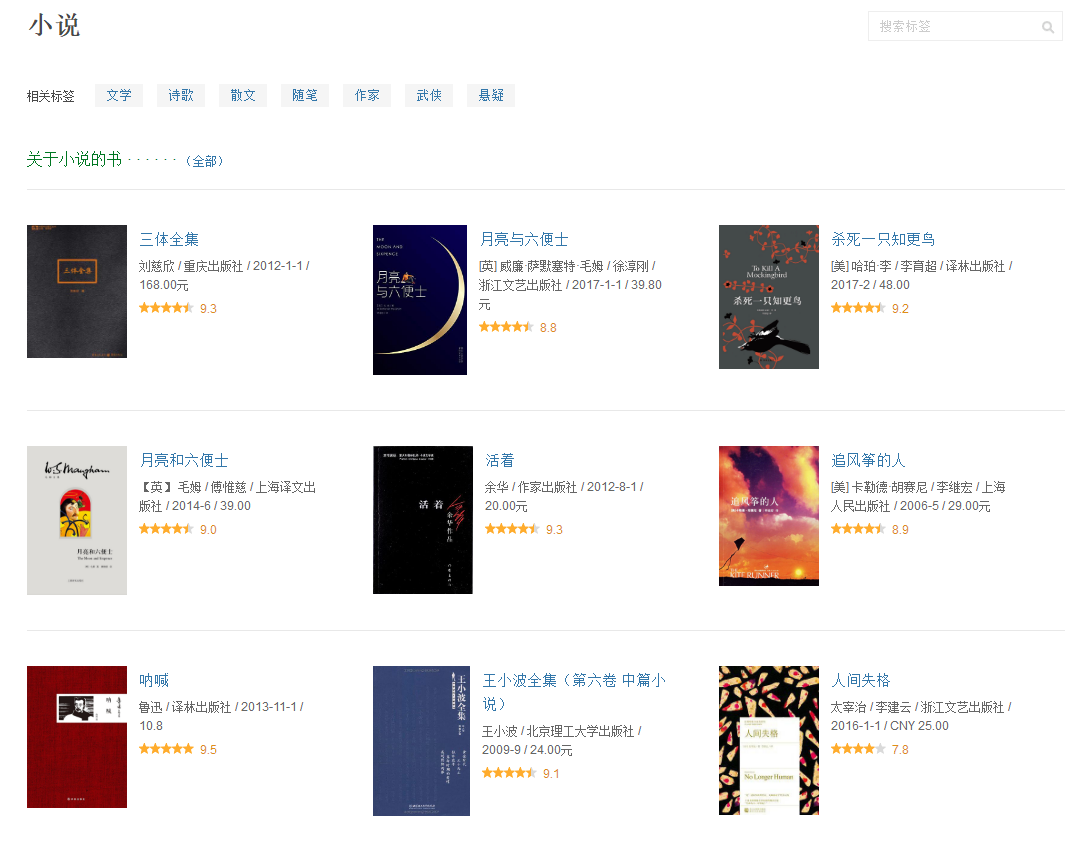
1。确定爬取目标;   
2。查看网页源码，确定所需内容所在HTML哪个标签中;   
3。通过requests.get（）方法获取单个网页源码;   
4。通过beautifulsoup方法获取标签内容;   
5。通过for循环遍历所有URL;   
6。通过beautifulsoup获取所有URL中需要爬取的数据。



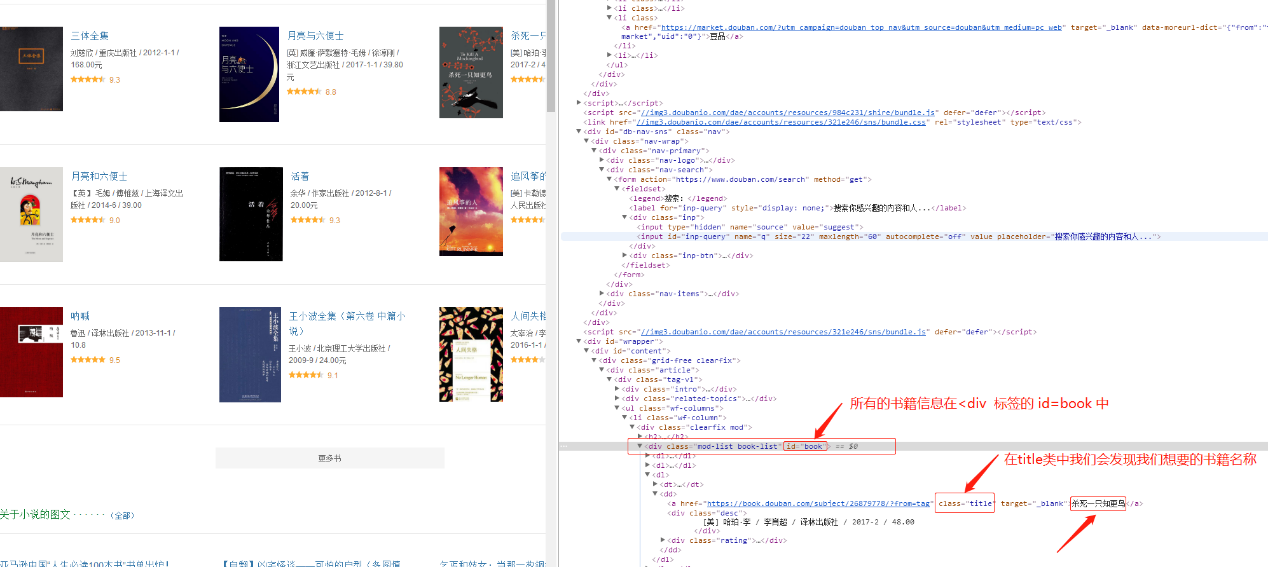
上一步

**爬虫简介**

* 接下来我们用一个小例子去学习我们的Python爬虫：  
  这是我们指定要去爬取数据的URL：豆瓣小说：[http](http://www.douban.com/tag/%E5%B0%8F%E8%AF%B4/?focus=book)：[//www.douban.com/tag/%E5%B0%8F%E8% AF％B4 /？重点=书](http://www.douban.com/tag/%E5%B0%8F%E8%AF%B4/?focus=book)



* 我们可以使用网页自带的审查元素功能可以产看到一下网页源码信息：



* 通过对网页源码的分析之后，我们可以写出一下代码对网页中我们需要的内容进行爬取：

#爬取豆瓣书籍

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

import requests

from bs4 import BeautifulSoup

from urllib import request

url = "http://www.douban.com/tag/%E5%B0%8F%E8%AF%B4/?focus=book" #网页地址

res = request.urlopen(url) #打开连接

response = requests.get(url)#返回http状态响应码

http\_status\_code = response.status\_code

print(http\_status\_code)

soup = BeautifulSoup(res,"html.parser")

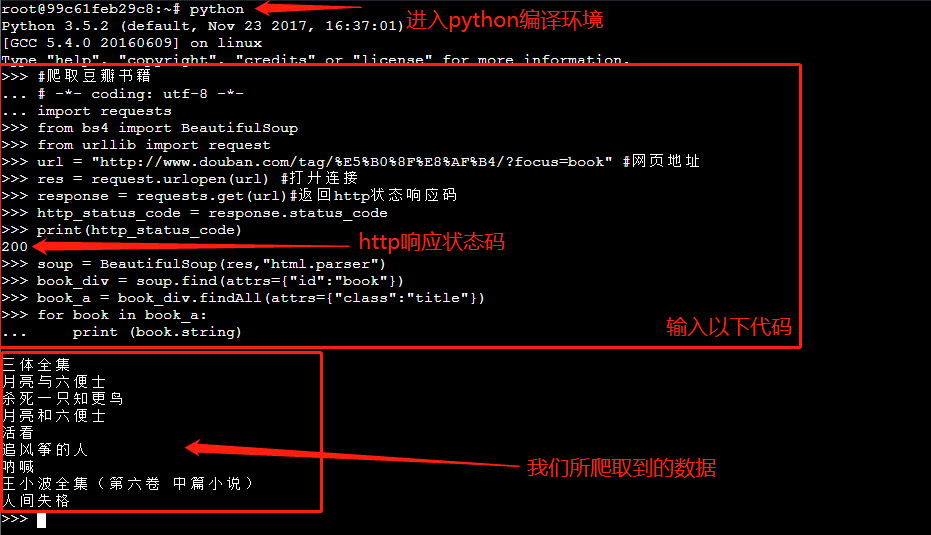
book\_div = soup.find(attrs={"id":"book"})

book\_a = book\_div.findAll(attrs={"class":"title"})

for book in book\_a:

print (book.string)

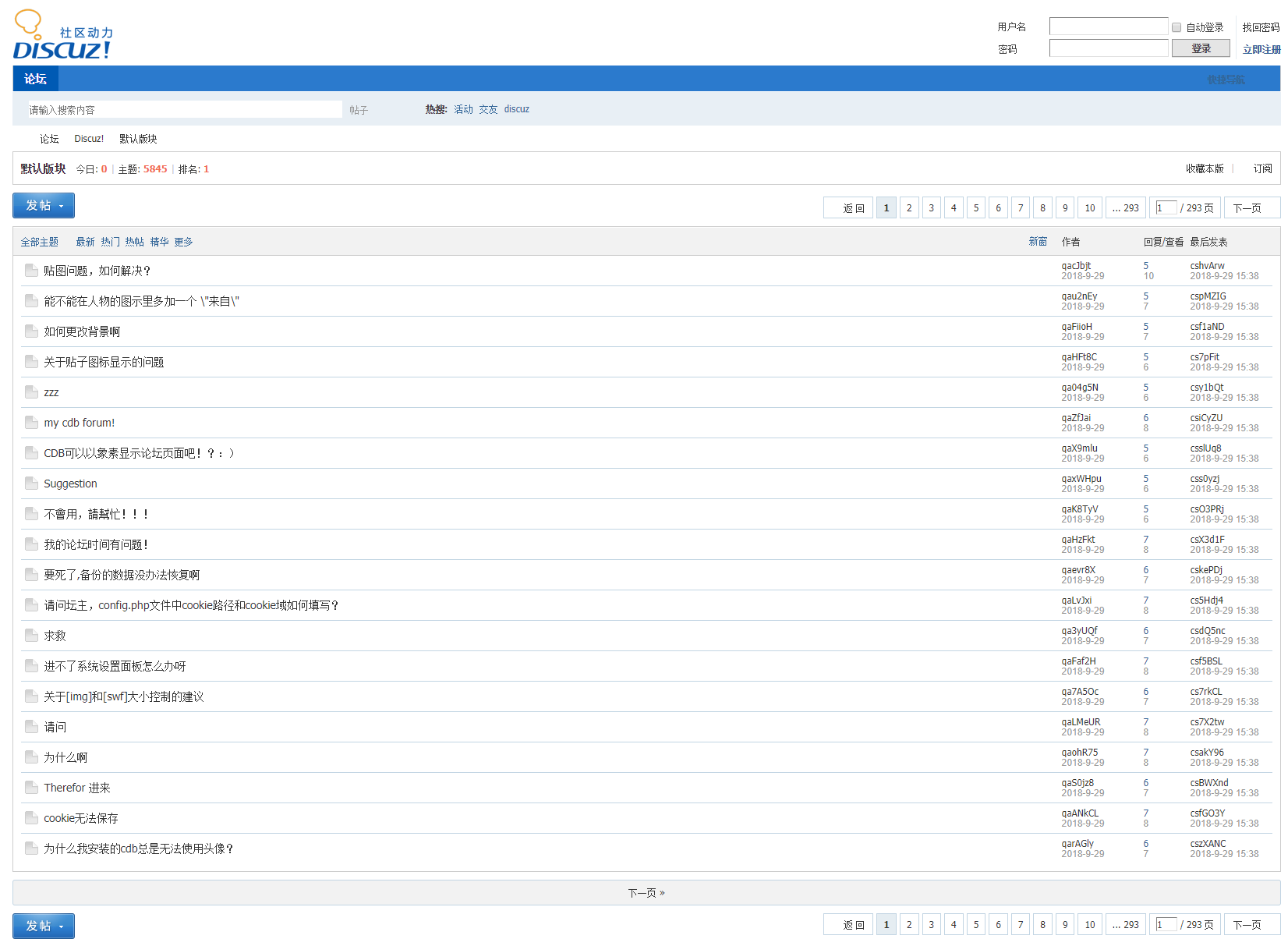
运行结果如下图所示：



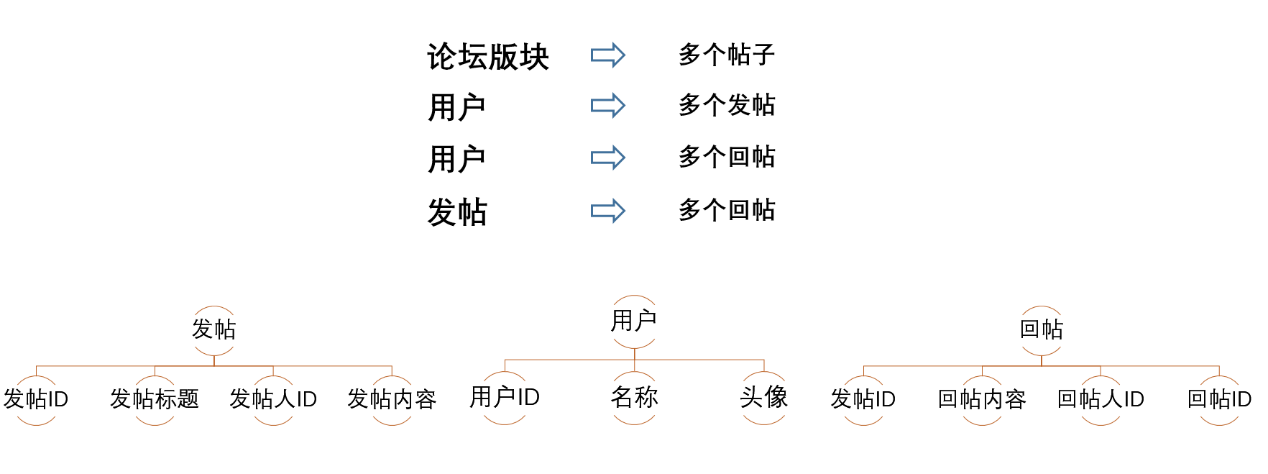
上一步

**爬虫简介**

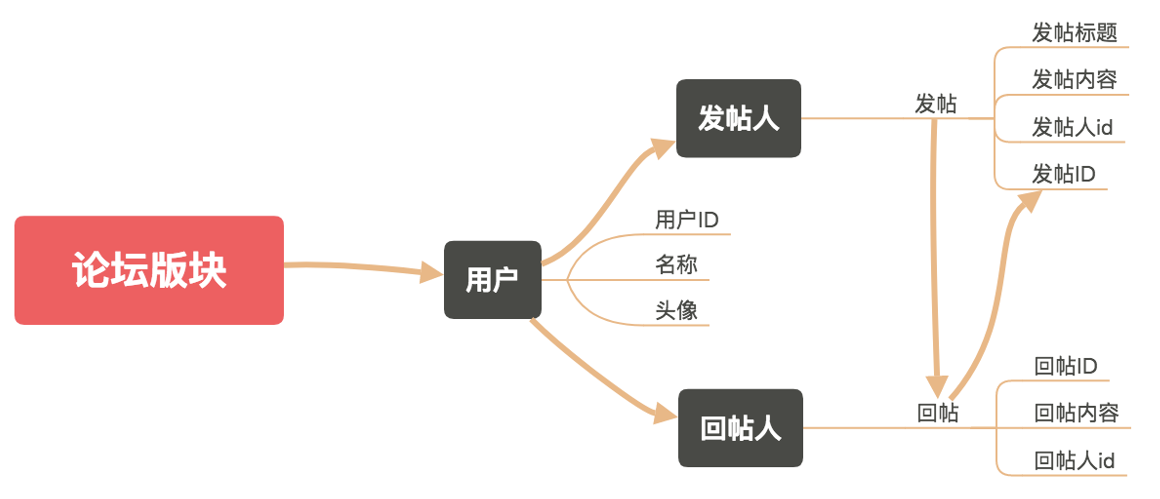
* 直接访问目标站点ip [http://114.112.74.138](http://114.112.74.138/)即可进入论坛主页。论坛的默认模块包含：
* 5800+条主题帖
* 1700+条回复帖
* 7500+条有效回复内容
* 550个会员
* discuz的论坛网页如下图所示：



* 论坛内涉及到的信息包含：论坛版块，发帖人，回帖人，发帖人ID，发帖人名称，回帖人ID，回帖人名称，用户头像，发帖内容，回帖内容，发帖ID，回帖ID等。其关系如下图所示：



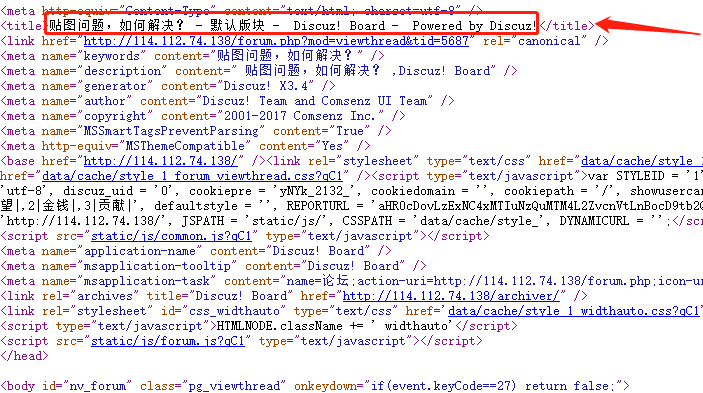
* 其逻辑关系图为：



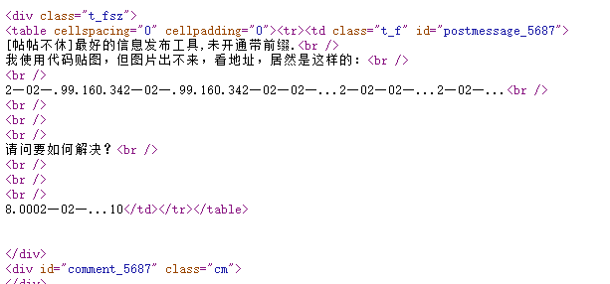
上一步

**爬虫简介**

* 首先我们查看网页源码，然后找到HTML中代码的<title> 标签，其中存放的内容为：



* 接下开我们需要查看一下论坛内帖子所在的位置，网页源码如下图所示：



* 当我们明确了所需要的爬取内容的位置时，我们可以使用以下代码对此论坛进行爬取，分别我们使用了两种方法去爬取我们的<title>标签中的内容，具体代码如下：

#爬取Discuz帖子标题

from bs4 import BeautifulSoup

import requests

for i in range(200):

#使用for循环更改url中的tid

html = requests.get('http://114.112.74.138/forum.php?mod=viewthread&tid='+str(i))

#创建BeautifulSoup对象

soup = BeautifulSoup(html.text,"html.parser")

#第一中爬取标题方法

title1 = soup.title.string

#第二种爬取标题的方法

title2=soup.find(attrs={"id":"thread\_subject"})

print(title1)

print(title2)

* 运行结果如下图所示：



* 爬取页1论坛中的所有帖子信息：

#爬取Discuz帖子内容及连接

from bs4 import BeautifulSoup

import requests

for i in range(200):

html = requests.get('http://114.112.74.138/forum.php?mod=viewthread&tid='+str(i))

soup = BeautifulSoup(html.text,"html.parser")

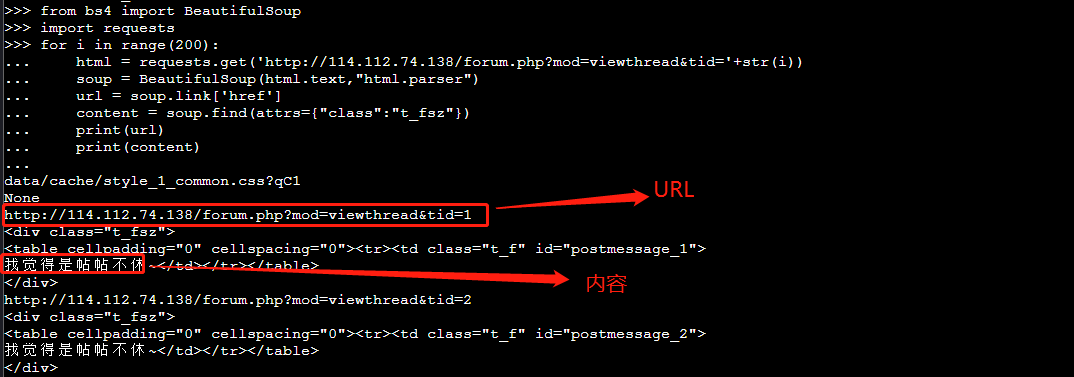
url = soup.link['href']

content = soup.find(attrs={"class":"t\_fsz"})

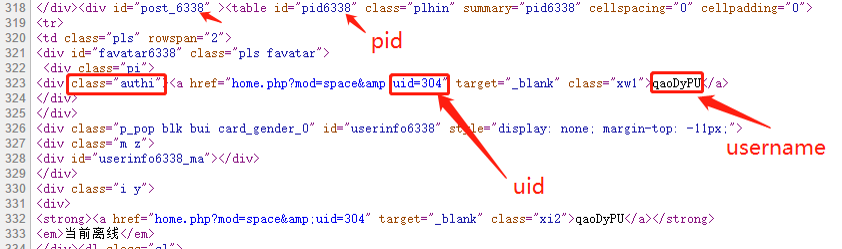
print(url)

print(content)

* 运行结果如下图所示：



* 查看源码中的PID和UID位置



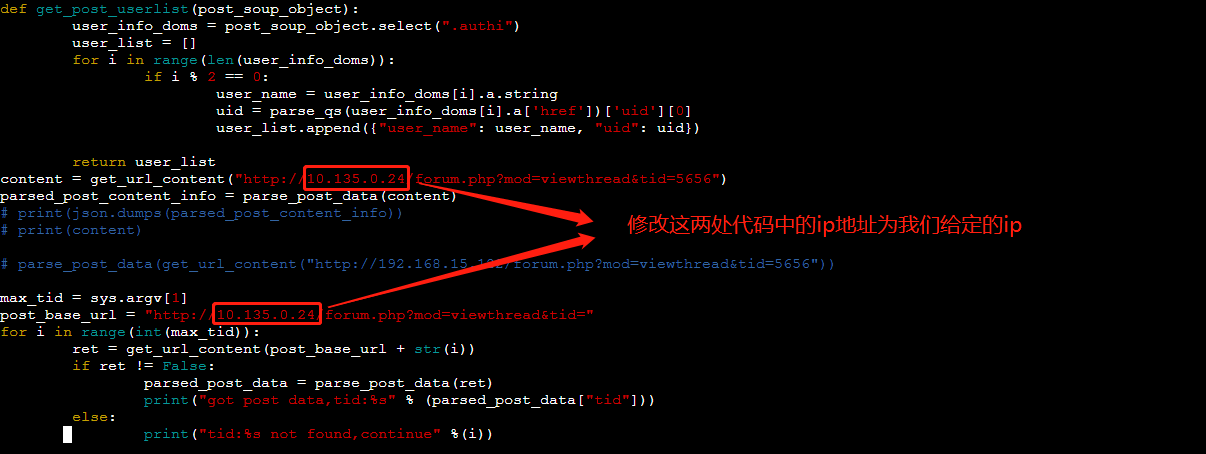
* 查看源码中贴子ID和评论ID位置



上一步

**爬虫简介**

* 首先我们使用vim的修改我们main.py代码中的IP地址为我们给定页1论坛服务器的IP地址：



import requests

##引入request库， requests是python实现的简单易用的HTTP库，使用起来比urllib简洁很多

from bs4 import BeautifulSoup

##BS4本身是一种对描述语言进行封装的函数操作模块，通过提供面向对象的操作方式将文档对象中的各种节点、标签、属性、内容等等都封装成了python中对象的属性，在查询操作过程中，通过调用指定的函数直接进行数据匹配检索操作，非常的简单非常的灵活。

from urllib.parse import urlparse

##使用urlparse包中的urlparse方法来解析url

from urllib.parse import parse\_qs

##主要用于分析URL中query组件的参数，返回一个key-value对应的字典格式

import sys

##这句语句告诉Python，我们想要使用这个模块。sys模块包含了与Python解释器和它的环境有关的函数。

def get\_url\_content(url):

response = requests.get(url)

##请求访问目标网站

if response.status\_code == 200:

##判断请求状态码（状态），返回值为200正常。

if "抱歉，指定的主题不存在或已被删除或正在被审核" in response.text:

return False

else:

return response.text

else:

return False

##判断"抱歉，指定的主题不存在或已被删除或正在被审核"是否在response.text网页源码（文本形式）中(是，返回False，否，以文本形式返回网页源码)

def parse\_post\_data(html\_text):

soup\_object = BeautifulSoup(html\_text, "html5lib")

##解析网页源码

title = soup\_object.title.string

##获取title并转换成string类型

url = soup\_object.link['href']

##获取href后的链接里的内容

parsed\_url = urlparse(url)

##将url解析成6个部分（协议、位置、路径、参数、查询、片段）

query\_string\_object = parse\_qs(parsed\_url.query)

##获取解析后元组中的query项

tid = query\_string\_object['tid'][0]

##获取解析后元组中第一个tid，代表着发帖

user\_list = get\_post\_userlist(soup\_object)

content\_list = get\_post\_content\_list(soup\_object)

for i in range(len(content\_list)):

content\_list[i]["user\_info"] = user\_list[i]

##获取发帖用户

post\_content\_info = {

"title" : title,

"url": url,

"tid": tid,

"author": user\_list[0],

"content": content\_list[0]["content"],

"comments": content\_list[1:]

}

return post\_content\_info

##将抓取的数据以字典的形式保存

def get\_post\_content\_list(post\_soup\_object):

content\_object\_list = post\_soup\_object.select('.t\_f')

#选择器。源码中标有.t\_f的为获取对象

content\_list = []

#定义一个空列表

for i in range(len(content\_object\_list)):

postmessage\_id = content\_object\_list[i]['id']

##获取发帖id

tid = postmessage\_id.split("\_")[1]

##获取发帖ID，以"\_"进行切分，取第二个元素

content = content\_object\_list[i].string

##将下角标为i的数据转换为string类型

content\_list.append({"tid": tid, "content": content})

return content\_list

##向列表中添加元素

def get\_post\_userlist(post\_soup\_object):

user\_info\_doms = post\_soup\_object.select(".authi")

##选择器，可以获得多条也可以获得单条数据

user\_list = []

##定义一个用户空列表

for i in range(len(user\_info\_doms)):

if i % 2 == 0:

user\_name = user\_info\_doms[i].a.string

##将下角标为i的数据转换为string类型

uid = parse\_qs(user\_info\_doms[i].a['href'])['uid'][0]

user\_list.append({"user\_name": user\_name, "uid": uid})

##向列表中添加元素

return user\_list

content = get\_url\_content("http://10.135.0.8/forum.php?mod=viewthread&tid=5656")

##调用get\_url\_content方法，请求"http://10.135.0.8/forum.php?mod=viewthread&tid=5656"，以文本形式返回网页源码

parsed\_post\_content\_info = parse\_post\_data(content)

#调用parse\_post\_data方法

# print(json.dumps(parsed\_post\_content\_info))

# print(content)

#parse\_post\_data(get\_url\_content("http://192.168.15.122/forum.php?mod=viewthread&tid=5656"))

max\_tid = sys.argv[1]

##sys.argv实现从程序外部向程序传递参数

post\_base\_url = "http://10.135.0.8/forum.php?mod=viewthread&tid="

for i in range(int(max\_tid)):

ret = get\_url\_content(post\_base\_url + str(i))

#调用get\_url\_content方法请求目标网站

if ret != False:

parsed\_post\_data = parse\_post\_data(ret)

print("got post data,tid:%s" % (parsed\_post\_data["tid"]))

else:

print("tid:%s not found,continue" %(i))

##判断ret是否不等于False<html>

<body>

<!--StartFragment--><pre style="margin-top: 0px; margin-right: 0px; margin-bottom: 0px !important; margin-left: 0px; padding: 16px; box-sizing: border-box; font: 400 14px/1.6 Consolas, &quot;Liberation Mono&quot;, Menlo, Courier, monospace; overflow-wrap: normal; overflow: auto; background-color: rgb(247, 247, 247); border-radius: 3px; color: rgb(36, 41, 46); letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; text-decoration-style: initial; text-decoration-color: initial;"><code style="margin: 0px; padding: 0px; box-sizing: border-box; font-family: Consolas, &quot;Liberation Mono&quot;, Menlo, Courier, monospace; font-size: 16px; background: transparent; border-radius: 3px; word-break: normal; white-space: pre; border: 0px; display: inline; overflow: initial; line-height: inherit; overflow-wrap: normal; max-width: none;">import requests

##引入request库， requests是python实现的简单易用的HTTP库，使用起来比urllib简洁很多

from bs4 import BeautifulSoup

##BS4本身是一种对描述语言进行封装的函数操作模块，通过提供面向对象的操作方式将文档对象中的各种节点、标签、属性、内容等等都封装成了python中对象的属性，在查询操作过程中，通过调用指定的函数直接进行数据匹配检索操作，非常的简单非常的灵活。

from urllib.parse import urlparse

##使用urlparse包中的urlparse方法来解析url

from urllib.parse import parse\_qs

##主要用于分析URL中query组件的参数，返回一个key-value对应的字典格式

import sys

##这句语句告诉Python，我们想要使用这个模块。sys模块包含了与Python解释器和它的环境有关的函数。

def get\_url\_content(url):

response = requests.get(url)

##请求访问目标网站

if response.status\_code == 200:

##判断请求状态码（状态），返回值为200正常。

if "抱歉，指定的主题不存在或已被删除或正在被审核" in response.text:

return False

else:

return response.text

else:

return False

##判断"抱歉，指定的主题不存在或已被删除或正在被审核"是否在response.text网页源码（文本形式）中(是，返回False，否，以文本形式返回网页源码)

def parse\_post\_data(html\_text):

soup\_object = BeautifulSoup(html\_text, "html5lib")

##解析网页源码

title = soup\_object.title.string

##获取title并转换成string类型

url = soup\_object.link['href']

##获取href后的链接里的内容

parsed\_url = urlparse(url)

##将url解析成6个部分（协议、位置、路径、参数、查询、片段）

query\_string\_object = parse\_qs(parsed\_url.query)

##获取解析后元组中的query项

tid = query\_string\_object['tid'][0]

##获取解析后元组中第一个tid，代表着发帖

user\_list = get\_post\_userlist(soup\_object)

content\_list = get\_post\_content\_list(soup\_object)

for i in range(len(content\_list)):

content\_list[i]["user\_info"] = user\_list[i]

##获取发帖用户

post\_content\_info = {

"title" : title,

"url": url,

"tid": tid,

"author": user\_list[0],

"content": content\_list[0]["content"],

"comments": content\_list[1:]

}

return post\_content\_info

##将抓取的数据以字典的形式保存

def get\_post\_content\_list(post\_soup\_object):

content\_object\_list = post\_soup\_object.select('.t\_f')

#选择器。源码中标有.t\_f的为获取对象

content\_list = []

#定义一个空列表

for i in range(len(content\_object\_list)):

postmessage\_id = content\_object\_list[i]['id']

##获取发帖id

tid = postmessage\_id.split("\_")[1]

##获取发帖ID，以"\_"进行切分，取第二个元素

content = content\_object\_list[i].string

##将下角标为i的数据转换为string类型

content\_list.append({"tid": tid, "content": content})

return content\_list

##向列表中添加元素

def get\_post\_userlist(post\_soup\_object):

user\_info\_doms = post\_soup\_object.select(".authi")

##选择器，可以获得多条也可以获得单条数据

user\_list = []

##定义一个用户空列表

for i in range(len(user\_info\_doms)):

if i % 2 == 0:

user\_name = user\_info\_doms[i].a.string

##将下角标为i的数据转换为string类型

uid = parse\_qs(user\_info\_doms[i].a['href'])['uid'][0]

user\_list.append({"user\_name": user\_name, "uid": uid})

##向列表中添加元素

return user\_list

content = get\_url\_content("http://10.135.0.8/forum.php?mod=viewthread&amp;tid=5656")

##调用get\_url\_content方法，请求"http://10.135.0.8/forum.php?mod=viewthread&amp;tid=5656"，以文本形式返回网页源码

parsed\_post\_content\_info = parse\_post\_data(content)

#调用parse\_post\_data方法

# print(json.dumps(parsed\_post\_content\_info))

# print(content)

#parse\_post\_data(get\_url\_content("http://192.168.15.122/forum.php?mod=viewthread&amp;tid=5656"))

max\_tid = sys.argv[1]

##sys.argv实现从程序外部向程序传递参数

post\_base\_url = "http://10.135.0.8/forum.php?mod=viewthread&amp;tid="

for i in range(int(max\_tid)):

ret = get\_url\_content(post\_base\_url + str(i))

#调用get\_url\_content方法请求目标网站

if ret != False:

parsed\_post\_data = parse\_post\_data(ret)

print("got post data,tid:%s" % (parsed\_post\_data["tid"]))

else:

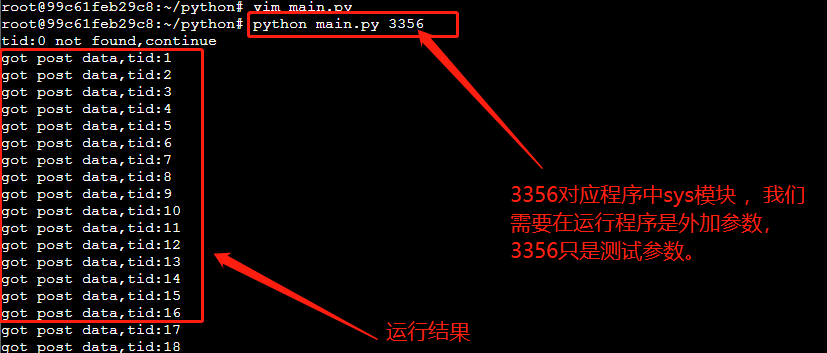
print("tid:%s not found,continue" %(i))

##判断ret是否不等于False</code></pre><!--EndFragment-->

</body>

</html>

* 此次爬虫代码我们放置在root用户的根目录下的python文件夹内，运行方法使用python main.py 3356   
  运行结果如下图所示：



上一步