

# PYTHON程序设计

计算机学院 王纯

## 九 数据获取（爬虫）

- HTML简介
- JSON和Xpath
- Scrapy库
- 静态页面的数据获取
- 动态页面的数据获取

九  
数  
据  
获  
取  
（  
爬  
虫  
）

# HTML简介

- HTML (HyperText Markup Language) 称为超文本标记语言，是一种标识性的语言。它包括一系列标签，通过这些标签来标记要显示的网页中的各个部分。
- 网页文件本身是一种文本文件，通过在文本文件中添加标记符，可以告诉浏览器如何显示其中的内容（如：文字如何处理，画面如何安排，图片如何显示等）。

互联网网站

浏览器



HTTP响应

HTTP请求



HTML文档

# 静态网页

- 在网站设计中，纯粹HTML（标准通用标记语言下的一个应用）格式的网页通常被称为“静态网页”，静态网页是标准的HTML文件，它的文件扩展名是.htm、.html，可以包含文本、图像、声音、FLASH动画、客户端脚本和ActiveX控件及JAVA小程序等。
- 静态网页是网站建设的基础，早期的网站一般都是由静态网页制作的。静态网页是相对于动态网页而言，是指没有后台数据库、不含程序和不可交互的网页。

# 一个简单的网页

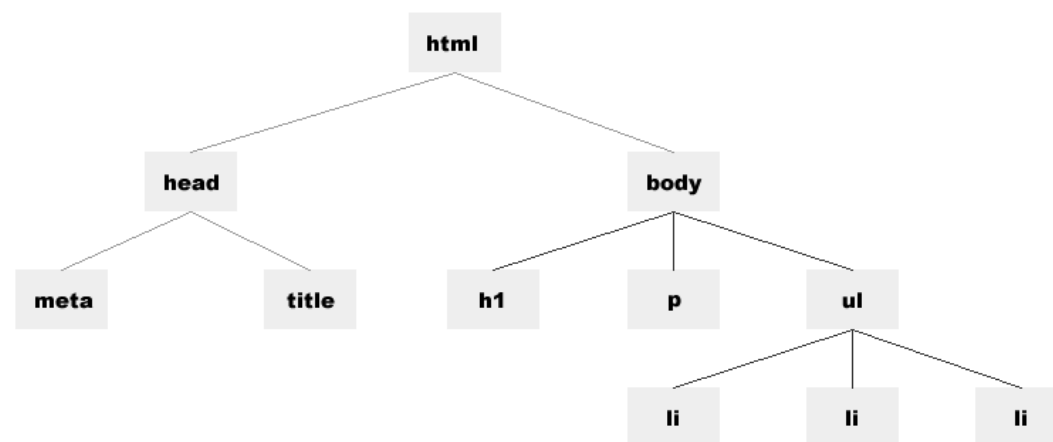
## The Dormouse's story

Once upon a time there were three little sisters; and their names were [Elsie](#), [Lacie](#) and [Tillie](#); and they lived at the bottom of a well.

...

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>
5       The Dormouse's story
6     </title>
7   </head>
8   <body>
9     <p class="title">
10       <b>
11         The Dormouse's story
12       </b>
13     </p>
14     <p class="story">
15       Once upon a time there were three little sisters; and their names were
16       <a class="sister" href="http://example.com/elsie" id="link1">
17         Elsie
18       </a>
19       ,
20       <a class="sister" href="http://example.com/lacie" id="link2">
21         Lacie
22       </a>
23       and
24       <a class="sister" href="http://example.com/tillie" id="link2">
25         Tillie
26       </a>
27       ; and they lived at the bottom of a well.
28     </p>
29     <p class="story">
30       ...
31     </p>
32   </body>
33 </html>
```

## 网页源代码和DOM模型



# 动态网页

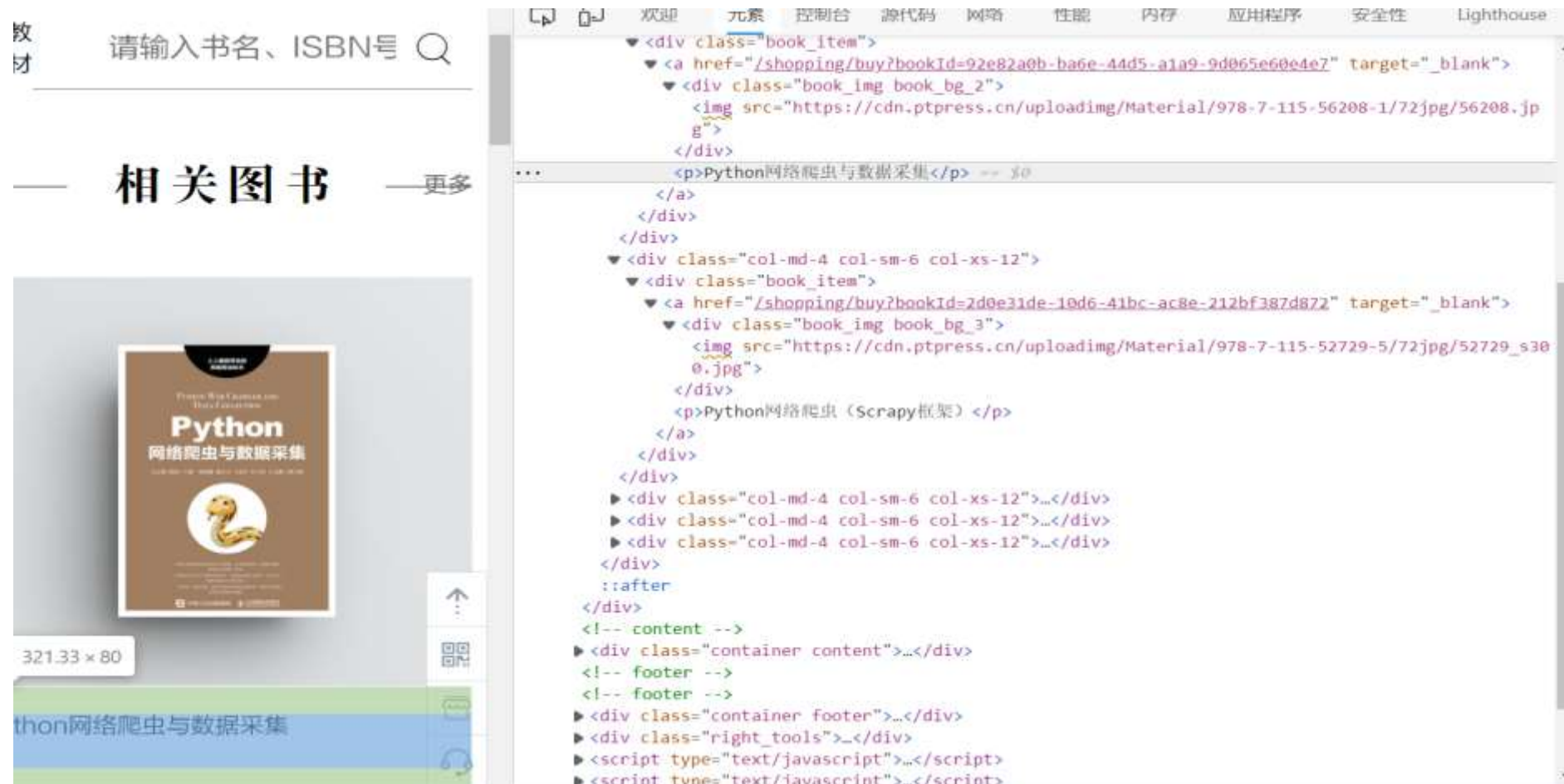
动态网页：无法从HTML源码中直接获取页面元素

- 前端页面与后台数据库联动，动态更新
- 网页内容可能由JavaScript动态生成
- 可能应用了AJAX技术和动态HTML技术（如“查看更多”或手动下拉到网页底端才会加载新内容）

判断静态与动态

- 若存在页面部分信息（特别是网页展示的核心内容）在网页源码中搜索不到，则网页一般为动态的

- 在浏览器中打开网站 “http://www.ptpress.com.cn”，并搜索 “Python网络爬虫”
- 右键点击 “检查” 调出开发者工具，找到 “Python网络爬虫与数据采集” 的HTML信息





- 右键点击“查看网页源代码”，并搜索“Python网络爬虫与数据采集”，发现并无相关内容。



The image shows a web browser window with the source code of a page displayed. A search bar is open in the top right corner, containing the text "Python网络爬虫与数据采集". The search results show "0/0", indicating no matches were found. The source code is HTML, starting with a DOCTYPE declaration and a head section containing meta tags for charset, renderer, http-equiv, and viewport. The title of the page is "人民邮电出版社". The body contains several link tags for stylesheets and a script tag for jQuery.

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="utf-8">
  <meta name="renderer" content="webkit">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>人民邮电出版社</title>

  <link rel="shortcut icon" href="/static/eleBusiness/img/favicon.ico" charset="UTF-8"/>
  <link rel="stylesheet" href="/static/plugins/bootstrap/css/bootstrap.min.css">
  <link rel="stylesheet" href="/static/portal/css/iconfont.css">
  <link rel="stylesheet" href="/static/portal/tools/iconfont.css">
  <link rel="stylesheet" href="/static/portal/css/font.css">
  <link rel="stylesheet" href="/static/portal/css/common.css">
  <link rel="stylesheet" href="/static/portal/css/header.css">
  <link rel="stylesheet" href="/static/portal/css/footer.css?v=1.0">
  <link rel="stylesheet" href="/static/portal/css/compatible.css">

  <script type="text/javascript" src="/static/portal/js/jquery-1.11.3.min.js"></script>

  <script type="text/javascript">
```

# HTML标签

```
<html>
<head>
    <title> 标题 </title>    (浏览器标题)
</head>

<body>
    内容主题    (网页具体内容)
</body>
</html>
```

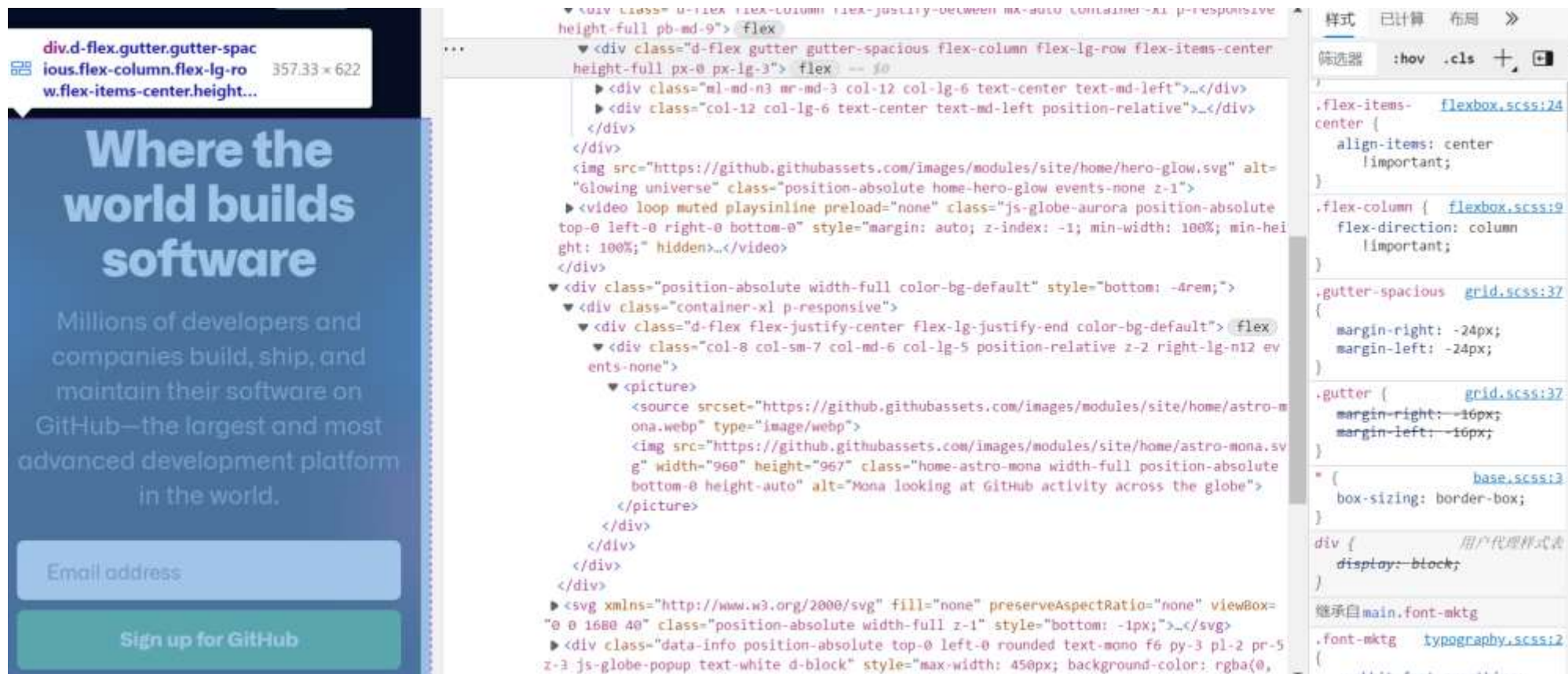
## 常用标签

```
<h1>
<p>
<ul>
<li>
<a href>
<table> <tr> <td>
```

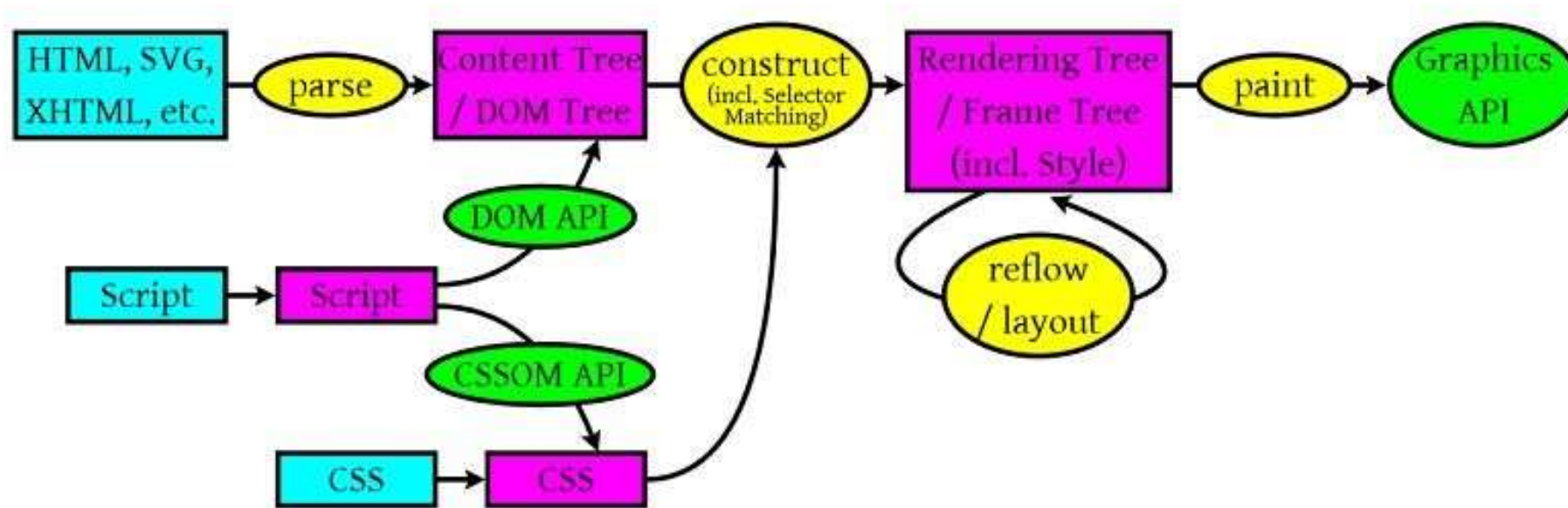
# DIV+CSS

## 层叠式样式表 (Cascading Style Sheet)

提供网页设定样式的功能，使网页能以更精确和结构化的方式显示版面  
HTML定义页面的内容，而CSS处理页面的表现



- 浏览器会解析HTML，产生一个DOM Tree；解析CSS产生CSS规则树；Javascript脚本则通过DOM API和CSSOM API来操作DOM Tree和CSS Rule Tree
- 解析完成后，浏览器引擎会通过DOM Tree 和 CSS Rule Tree 来构造 Rendering Tree；CSS 的 Rule Tree主要是为了完成匹配并把CSS Rule附加上Rendering Tree上的每个Element，也就是DOM结点，即所谓的Frame；然后，计算每个Frame（也就是每个Element）的位置，这又叫layout和reflow过程
- 最后通过调用操作系统Native GUI的API绘制



# JSON

- JSON(JavaScript Object Notation, JS 对象表示法), 是一种轻量级的数据交换格式。它基于 ECMAScript (欧洲计算机协会制定的JS规范) 的一个子集, 采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。易于人阅读和编写, 同时也易于机器解析和生成, 并有效地提升网络传输效率。

- 数据使用键/值（key-value）对表示。
- 使用大括号保存对象，每个名称后面跟着一个 ‘:’（冒号），多个键/值对之间使用,（逗号）隔开。

```
{  
    "id":1,  
    "language":"Python",  
    "edition": "third",  
    "price": 35.50  
}
```

```
{  
    "books": [  
        {  
            "id":1,  
            "language": "Python",  
            "edition": "third",  
            "price": 35.50  
        },  
        {  
            "id":2,  
            "language": "C++",  
            "edition": "second",  
            "price": 29.80  
        }  
    ]  
}
```

- 数组结构

```
<html>
<body>
<h2>JSON Object Creation in JavaScript</h2>
<p>
  ID: <span id="j_id"></span><br>
  Language: <span id="j_language"></span><br>
  Edition: <span id="j_edition"></span><br>
  Price: <span id="j_price"></span><br>
</p>
<script>
  var JSONObject = {
    "id": 1,
    "language": "Python",
    "edition": "third",
    "price": 35.50
  };
  document.getElementById("j_id").innerHTML = JSONObject.id
  document.getElementById("j_language").innerHTML = JSONObject.language
  document.getElementById("j_edition").innerHTML = JSONObject.edition
  document.getElementById("j_price").innerHTML = JSONObject.price
</script>
</body>
</html>
```



## JSON

```
"car" : {  
  "company": Volkswagen,  
  "brand": " Jetta ",  
  "price": 200000  
}
```

## XML

```
<car>  
  <company>Volkswagen</company>  
  <brand>Jetta</brand>  
  <price>200000</price>  
</car>
```



## JSON文件的读写

Python的json模块提供了处理JSON文件格式的方法：

*json.loads()* #读取

*json.dump()* #写入

```
import json

with open('package.json', 'r') as f:
    data = json.load(f)
    #print(data)
    list = data['data']['org_list']
    for item in list:
        print(item['name'], 'offering courses:', item['product_num'])
#构造新的{学校: 课程}的字典
org_data = {item['name']:item['product_num'] for item in list}
with open('orglist.json', 'w') as f:
    json.dump(org_data, f)
    print(org_data)
```

# XPATH

- XPath 是 W3C (World Wide Web Consortium) 的一种标准, 是一种在 XML 文档 (包含HTML文档) 中查找信息的语言。 可以让我们以路径的形式访问 html 网页中的各个元素。
- 使用Xpath可以帮助我们很方便地定位复杂网页中的各个元素, 使得我们有针对性地去获取网页中的特定信息。

根节点

属性

文本

<div>

<ul>

<li class="item-1">li1:<a href="link1.html">text1</a></li>

<li class="item-2">li2:<a href="link2.html">text2</a></li>

<li class="item-3">li3:<a href="link3.html">text3</a></li>

</ul>

</div>

元素

表达式	描述
node name	选取当前节点的所有子节点中， 节点名称为nodename的节点
/	从根节点选取
//	选取所有节点
.	选取当前节点
..	选取当前节点的父节点
@	选取属性



xpath("//@class")

```
<div class="class-0" id="id0"> div0
  <div class="class-1">div1
    <ul>ul1
      <li class="item-1">li1:<a href="link1.html">text1</a></li>
      <li class="item-2">li2:<a href="link2.html">text2</a></li>
      <li class="item-3">li3:<a href="link3.html">text3</a></li>
    </ul>
  </div>
  <div class="class-2">div2
    <ul>ul2
      <li class="item-1">li4:<a href="link4.html">text4</a></li>
      <li class="item-2">li5:<a href="link5.html">text5</a></li>
      <li class="item-3">li6:<a href="link6.html">text6</a></li>
    </ul>
  </div>
</div>
```

在标签后面还可以加上谓词，用于更为精准的选择。  
例如div[1], li[last()], li[price>=80]等

谓词	描述
[1]	选取第一个节点
[last()]	选取最后一个节点
[position()<6]	选取前5个子节点
[@class]	选取含属性名为class的节点
[@class='0']	选取含属性名为class且值为0的节点
[price>80]	选取元素为price且值大于80的节点

xpath('div[1]')

xpath('//div[1]')

```
<div class="class-0" id="id0"> div0
  <div class="class-1">div1
    <ul>
      <li class="item-1">li1:<a href="link1.html">text1</a></li>
      <li class="item-2">li2:<a href="link2.html">text2</a></li>
      <li class="item-3">li3:<a href="link3.html">text3</a></li>
    </ul>
  </div>
  <div class="class-2">div2
    <ul>
      <li class="item-1">li4:<a href="link4.html">text4</a></li>
      <li class="item-2">li5:<a href="link5.html">text5</a></li>
      <li class="item-3">li6:<a href="link6.html">text6</a></li>
    </ul>
  </div>
</div>
```



xpath('//li[1]')

```
<div class="class-0" id="0"> div0
```

```
  <div class="class-1">div1
```

```
    <ul>
```

```
      <li class="item-1">li1:<a href="link1.html">text1</a></li>
```

```
      <li class="item-2">li2:<a href="link2.html">text2</a></li>
```

```
      <li class="item-3">li3:<a href="link3.html">text3</a></li>
```

```
    </ul>
```

```
  </div>
```

```
  <div class="class-2">div2
```

```
    <ul>
```

```
      <li class="item-1">li4:<a href="link4.html">text4</a></li>
```

```
      <li class="item-2">li5:<a href="link5.html">text5</a></li>
```

```
      <li class="item-3">li6:<a href="link6.html">text6</a></li>
```

```
    </ul>
```

```
  </div>
```

```
</div>
```

xpath('//li[last()])')

通配符	描述
//*	选取所有节点
/*	选取当前节点的所有子节点
[@*]	选取含属性的所有节点

使用Chrome浏览器，打开网页后，右键->检查->copy xpath，在网页中精准找到要定位的信息。

# 爬虫工具

- Requests 自动爬取HTML页面 自动网络请求提交
- BeautifulSoup 解析HTML页面
- robots.txt 网络爬虫排除标准
- Scrapy 网络爬虫框架
- Selenium。。。

# 任务

- 抓取北邮主页的组织结构页面数据
- 解析院系及二级单位的名称和网址链接
- 数据以json数据格式保存到一个json文件中

西安办公室

统战部

a108 × 24室

研究生工作部

校长办公室

教务处

人事处

保密处

审计处

组织部

巡察

教师

保密

发展

研究

学生

教材

监察

<li>...</li>

<li>...</li>

<li>...</li>

<li>...</li>

<li>...</li>

<li>

<a href="#" onclick="\_addDynClicks("wburl", 1552820829, 58912)" target="\_blank">纪委监督检查室

</a>

</li>

<li>...</li>

<li>...</li>

<li>

<a href="http://ygb.bupt.edu.cn/" onclick="\_addDynClicks("wburl", 1552820829, 43534)" target="\_blank">研究生工作部</a> == \$0

</li>

<li>...</li>

<li>...</li>

</ul>

</div>

</li>

<li>...</li>

<li>...</li>

<li>...</li>

<li>...</li>

<li>...</li>

</ul>

</div>

</div>

## Requests+bs4实现

- 抓取北邮主页的组织结构页面数据
- 解析院系及二级单位的名称和网址链接
- 数据以json数据格式保存到一个json文件中

```
import requests
from bs4 import BeautifulSoup
import json
# 1. 下载页面
ret = requests.get(url="http://www.bupt.edu.cn/yxjg1.htm")
ret.encoding = ret.apparent_encoding # 指定编码等于原始页面编码
# 2. 解析: 获取想要的指定内容 beautifulsoup
soup = BeautifulSoup(ret.text, 'html.parser') # 使用lxml则速度更快
div = soup.find(name='div', attrs={"class": "content"})
a_list = div.find_all(name='a')
departments = []

for a in a_list:
    if a:
        print ('---->', a.get('href'), a.text)
        dep_dict_item = {}
        dep_dict_item['department'] = a.text
        dep_dict_item['link'] = a.get('href')
        departments.append(dep_dict_item)
print(departments, len(departments))

filename = 'buptpage-with-requests-bs4.json'
with open(filename, "w", encoding="utf-8") as f:
    for dict_item in departments:
        json_str = json.dumps(dict_item, ensure_ascii=False) + "\n"
        f.write(json_str)
```

## Lxml实现

- 抓取北邮主页的组织结构页面数据
- 解析院系及二级单位的名称和网址链接
- 数据以json数据格式保存到一个json文件中

```
from lxml import etree
import json

html = etree.parse("http://www.bupt.edu.cn/yxjg1.htm", etree.HTMLParser(encoding='utf-8'))
xpath_pattern = '/html/body/div[*]/div[*]/div[*]/div/ul/li[*]/div/ul/li[*]/a'
# /html/body/div[2]/div[2]/div[2]/div/ul/li[1]/div/ul/li[1]/a
# /html/body/div[2]/div[2]/div[2]/div/ul/li[6]/div/ul/li[5]/a
result = html.xpath(xpath_pattern)
departments = []

print(f"match xpath {xpath_pattern} is : {result}")
print(len(result))

for i in result:
    print(f"+++++++{i.attrib}, {i.text}")
    dep_dict_item = {}
    dep_dict_item['department'] = i.text
    dep_dict_item['link'] = i.attrib['href']
    departments.append(dep_dict_item)

print(departments, len(departments))

filename = 'buptpage-with-lxml.json'
with open(filename, "w", encoding="utf-8") as f:
    for dict_item in departments:
        json_str = json.dumps(dict_item, ensure_ascii=False) + "\n"
        f.write(json_str)
```

# 什么是框架

- 软件框架是一种提供通用功能的软件抽象，在框架基础上通过有选择地添加用户编写的代码可以构成针对特定应用的软件。框架模式提供了一种构建和部署应用软件的标准方式，有比较好的可重用性，可以包括程序、编译器、程序库、工具集和API等有效贯穿软件开发过程的部件。
- 框架有别于库的主要特点：
  - ✓ 控制的倒置(inversion of control)、缺省行为(default behavior)、可扩展性(extensibility)和不可修改的框架代码 (non-modifiable framework code)
- 好的框架通常产生于有效的重构 (refactoring) 。

*.NET Spring Pytorch tensorflow Flask ....*

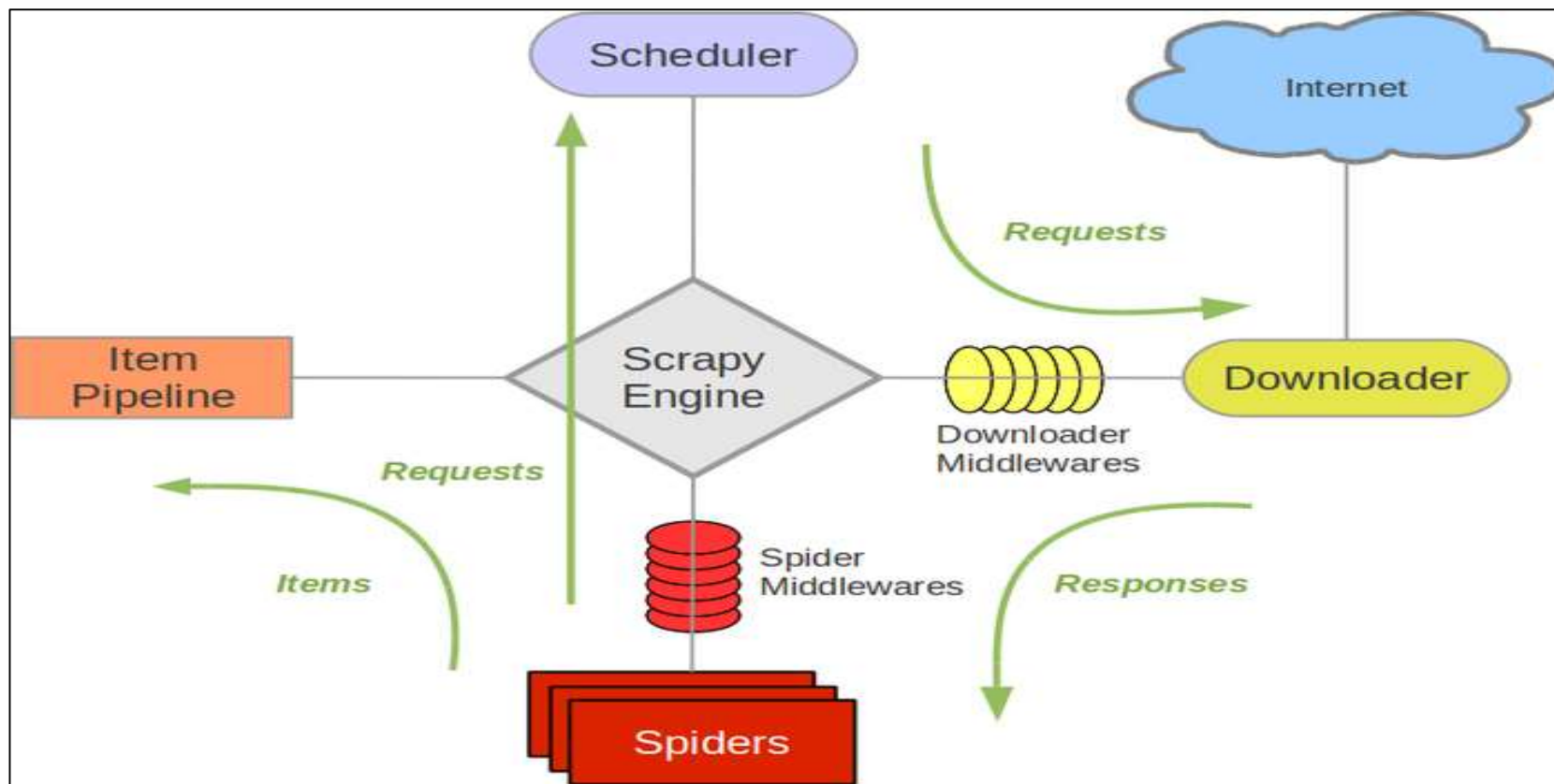


# SCRAPY框架

- Scrapy 是一个功能非常强大的爬虫框架，它不仅能便捷地构建 request请求，还有强大的 selector 能够方便地解析 response响应。
- 可以使用这个工具将爬虫程序工程化、模块化。
- Scrapy是一个基于Twisted的异步处理框架，是纯python实现的爬虫框架，其架构清晰，模块之间的耦合程度低，可扩展性也很强。



## Scrapy的工作过程



# Scrapy的主要模块

模块	说明
Scrapy Engine引擎	整个框架的核心，负责控制数据流在系统中的各个组件之间的传送，并在相应动作发生时触发事件
Scheduler调度器	接受引擎发过来的请求，并将其加入url队列当中，并默认完成去掉重复的url的工作
Downloader 下载器	负责下载 Engine 发送的所有 Requests 请求，并将其获取到的 responses 回传给 Scrapy Engine
Spider 爬虫	负责解析response，从中提取数据赋给Item的各个字段。并将需要继续进一步处理的url提交给引擎，再次进入Scheduler(调度器)
Item Pipeline 项目管道	处理Spider中获取到的Item，并进行后期的处理。例如清理 HTML 数据、验证爬取的数据（检查 item 包含某些字段）、查重（并丢弃）、爬取数据的持久化（写入文件或者存入数据库等）
Downloader Middlewares 下载中间件	是 Engine 和 Downloader 的枢纽。负责处理 Downloader 传递给 Engine 的 responses；它还支持自定义扩展。
Spider Middlewares 爬虫中间件	可以自定扩展和操作引擎和Spider中间通信的功能组件（比如进入Spider的Responses;和从Spider出去的Requests）负责下载 Engine 发送的所有 Requests 请求，并将其获取到的 responses 回传给 Scrapy Engine

## Scrapy的工作过程

- 1、spider爬虫将初始url请求发给引擎；
- 2、引擎将初始请求发给调度器，调度器将该url放入队列；
- 3、调度器回复引擎url已经入队；
- 4、通知下载器进行下载；
- 5、下载器向目标网站发出下载请求；
- 6、获得网页内容；
- 7、下载器通知引擎已经下载完成；
- 8、引擎将response发给spider，spider 解析数据、提取item；
- 9、spider将获取到的数据给到引擎，并通知引擎把新的url给到调度器进入队列，同时把item 数据发送给Item Pipelines进行保存；
- 10、Item Pipelines将提取到的数据加工并保存 到数据库或者文件中
- 11、保存完毕后通知引擎进行下一个url的提取；
- 12、循环1-11步，直到调度器中没有新的url， 结束整个过程。

## Scrapy安装

- scrapy依赖的模块较多，例如wheel、lxml、Twisted、pywin32等，只有这些都成功安装之后，才能安装scrapy。可以使用pip工具依次安装，但这样比较麻烦，推荐使用pycharm来安装更加简单方便。
- 安装好pycharm后，新建一个工程（例如工程名称为myscrapy），打开菜单File->Setting->Project: spider>Project Interpreter，找到右上角的加号符号，选择添加scrapy。安装过程中可能会出现提示，提示缺少Microsoft Visual C++ 14.0。这时可以使用以下网址安装：  
<http://go.microsoft.com/fwlink/?LinkId=691126>。安装好Microsoft Visual Build Tools之后，便可以成功安装 scrapy了。

## 静态页面的获取

- 新建项目 (`scrapy startproject xxx`): 新建一个新的爬虫项目
- 确定目标 (编写`items.py`): 明确你想要抓取的目标
- 制作爬虫 (`spiders/xxspider.py`): 制作爬虫开始爬取网页
- 存储内容 (`pipelines.py`): 设计管道存储爬取内容

## STEP1: 创建一个Scrapy项目

- 先找到安装scrapy的目录：打开cmd命令行，先用cd命令转到该目录下的venv\scripts\，再键入命令：scrapy startproject test1，即可创建一个新的项目。

```
C:\工作目录\北邮\北邮教学\源程序\pythonProject-scrapy-learn-demo\venv\Scripts>scrapy startproject test1
New Scrapy project 'test1', using template directory 'C:\工作目录\北邮\北邮教学\源程序\project1\venv\lib\site-packages\scrapy\templates\project', created in:
  C:\工作目录\北邮\北邮教学\源程序\pythonProject-scrapy-learn-demo\venv\Scripts\test1

You can start your first spider with:
  cd test1
  scrapy genspider example example.com

C:\工作目录\北邮\北邮教学\源程序\pythonProject-scrapy-learn-demo\venv\Scripts>cd test1
```

## STEP1: 创建一个Scrapy项目

### ■ 使用pycharm打开工程test1

需要修改

Item数据文件

管道文件

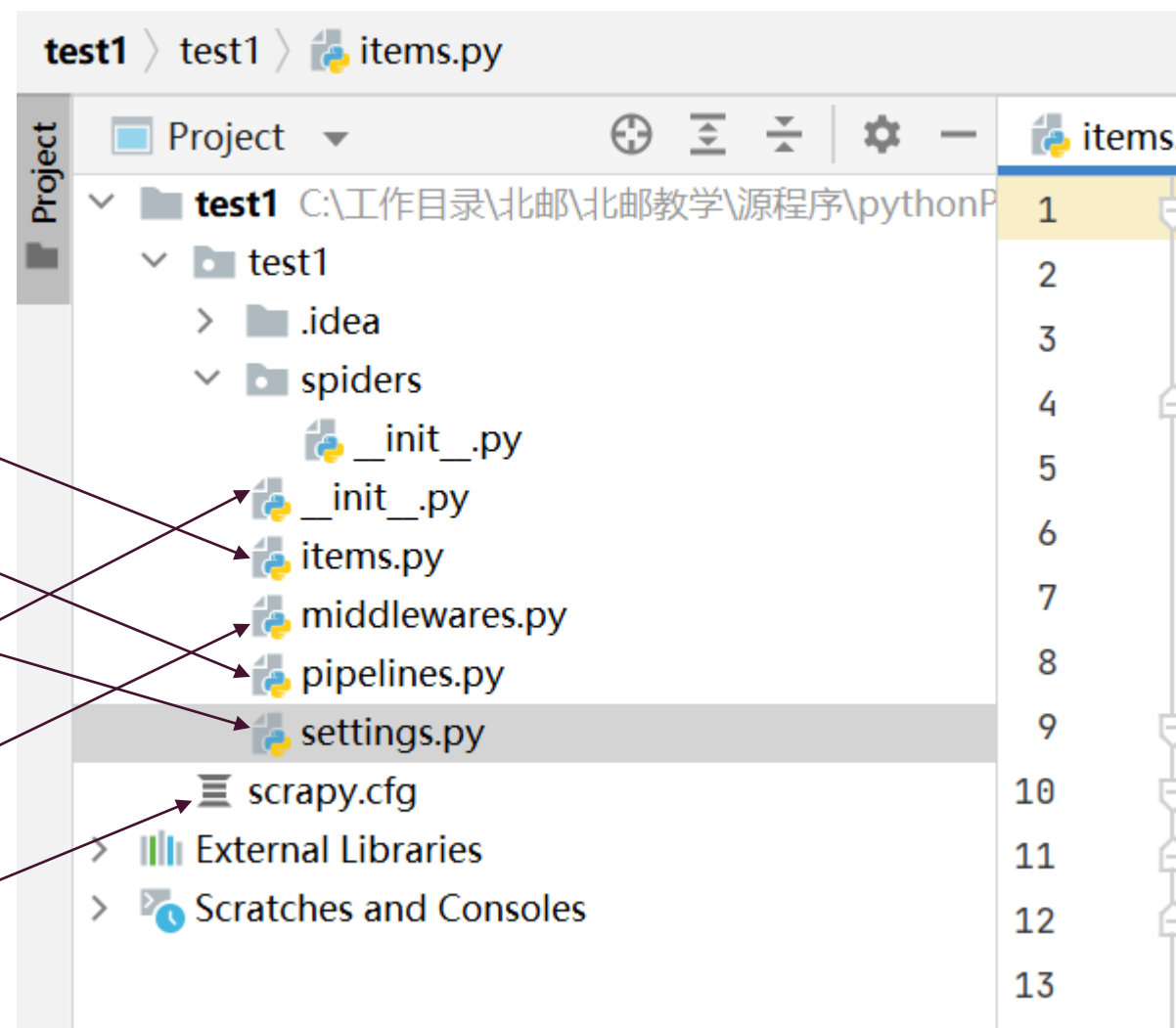
设置文件

不需要修改

初始化文件

中间件文件

项目配置文件



## STEP1: 创建一个Scrapy项目

- 在test1工程之下，新建一个begin.py文件，和scrapy.cfg 在同一级目录下，内容如下：

```
from scrapy import cmdline
```

```
cmdline.execute("scrapy crawl bupt".split())
```

```
#bupt为爬虫的名字，在spider.py中定义
```



## STEP2: 修改items.py文件

```
import scrapy

class MyItem(scrapy.Item):
    # define the fields for your item here like:
    school = scrapy.Field()
    link = scrapy.Field()
```

## STEP3: 新建一个spider.py文件

```
import scrapy
from test1.items import MyItem #从items.py中引入MyItem对象

class mySpider(scrapy.spiders.Spider):
    name = "bupt" #爬虫的名字是bupt
    allowed_domains = ["bupt.edu.cn"] #允许爬取的网站域名
    start_urls = ["https://www.bupt.edu.cn/yxjg1.htm"]
    #初始URL，即爬虫爬取的第一个URL

    def parse(self, response): #解析爬取的内容
        item = MyItem() #生成一个在items.py中定义好的Myitem对象,用于接收爬取的数据
        #/html/body/div[2]/div[2]/div[2]/div/ul/li[1]/div/ul/li[7]/a
        for each in response.xpath("/html/body/div[*]/div[*]/div[*]/div/ul/li[*]/div/ul/li[*]/a"):
            #用xpath来解析html，div标签中的数据，就是我们需要的数据。
            item['school'] = each.xpath("text()").extract() #学院名称在text中
            item['link'] = each.xpath("@href").extract() #学院链接在href中
            if(item['school'] and item['link']): #去掉值为空的数据
                yield(item) #返回item数据给到pipelines模块
```

## STEP4: 修改pipelines文件

```
import json

class MyPipeline(object):
    def open_spider(self, spider):
        try: # 打开json文件
            self.file = open('scrapy-test-bupt.json', "w", encoding="utf-8")
        except Exception as err:
            print(err)

    def process_item(self, item, spider):
        dict_item = dict(item) # 生成字典对象
        json_str = json.dumps(dict_item, ensure_ascii=False) + "\n" # 生成json串
        self.file.write(json_str) # 将json串写入到文件中
        return item

    def close_spider(self, spider):
        self.file.close() # 关闭文件
```

## STEP5: 修改settings文件

- 参数是分配给每个类的整型值，确定了它们运行的顺序， item按数字从低到高的顺序，通过pipeline。
- 通常将这些数字定义在0-1000范围内。

```
BOT_NAME = 'test1'
```

```
SPIDER_MODULES = ['test1.spiders']
```

```
NEWSPIDER_MODULE = 'test1.spiders'
```

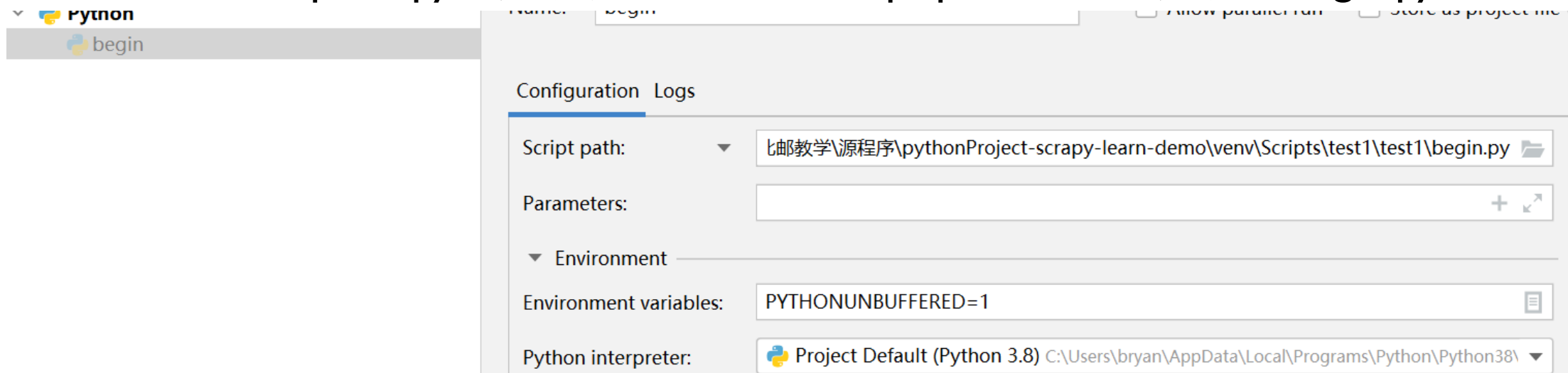
```
ITEM_PIPELINES = {'test1.pipelines.MyPipeline': 300,} #激活管道
```

## STEP6: 运行spider

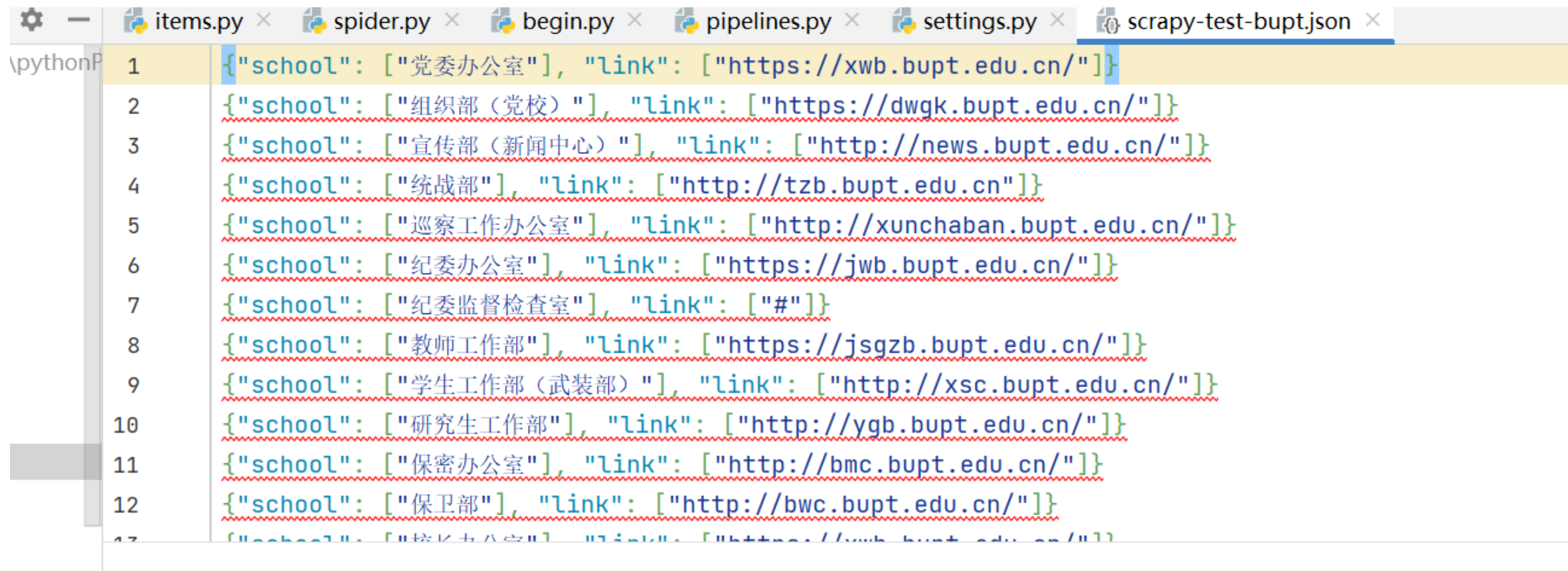
```
5 个目录 359,538,249,728 可用字节

C:\工作目录\北邮\北邮教学\源程序\pythonProject-scrapy-learn-demo\venv\Scripts\test1\test1>scrapy crawl bupt
2021-11-22 17:58:33 [scrapy.utils.log] INFO: Scrapy 2.5.1 started (bot: test1)
2021-11-22 17:58:33 [scrapy.utils.log] INFO: Versions: lxml 4.6.4.0, libxml2 2.9.5, cssselect 1.1.0, parsel 1.6.0, w3lib
1.22.0, Twisted 21.7.0, Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)], pyOpenSSL
L 21.0.0 (OpenSSL 1.1.1f 24 Aug 2021), cryptography 35.0.0, Platform Windows-10-10.0.19041-SP0
2021-11-22 17:58:33 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2021-11-22 17:58:33 [scrapy.crawler] INFO: Overridden settings:
{'BOT_NAME': 'test1',
 'NEWSPIDER_MODULE': 'test1.spiders',
 'ROBOTSTXT_OBEY': True}
```

- 命令行下执行 `scrapy crawl bupt`
- IDE中运行spider.py，并将其运行时的Script path配置项修改为begin.py



## 执行结果



```
items.py x spider.py x begin.py x pipelines.py x settings.py x scrapy-test-bupt.json x
1 {"school": ["党委办公室"], "link": ["https://xwb.bupt.edu.cn/"]}
2 {"school": ["组织部（党校）"], "link": ["https://dwgk.bupt.edu.cn/"]}
3 {"school": ["宣传部（新闻中心）"], "link": ["http://news.bupt.edu.cn/"]}
4 {"school": ["统战部"], "link": ["http://tzb.bupt.edu.cn"]}
5 {"school": ["巡察工作办公室"], "link": ["http://xunchaban.bupt.edu.cn/"]}
6 {"school": ["纪委办公室"], "link": ["https://jwb.bupt.edu.cn/"]}
7 {"school": ["纪委监督检查室"], "link": ["#"]}
8 {"school": ["教师工作部"], "link": ["https://jsgzb.bupt.edu.cn/"]}
9 {"school": ["学生工作部（武装部）"], "link": ["http://xsc.bupt.edu.cn/"]}
10 {"school": ["研究生工作部"], "link": ["http://ygb.bupt.edu.cn/"]}
11 {"school": ["保密办公室"], "link": ["http://bmc.bupt.edu.cn/"]}
12 {"school": ["保卫部"], "link": ["http://bwc.bupt.edu.cn/"]}
13 {"school": ["校长办公室"], "link": ["https://xwb.bupt.edu.cn/"]}
```

# 多次执行

- 可能的需求

- 多个爬虫在一个项目下
- 定时任务。。。

```
1 from scrapy import cmdline
2
3 cmdline.execute("scrapy crawl lianjia-1st".split())
4 cmdline.execute("scrapy crawl lianjia-2nd".split())
```

- Scrapy的限制：cmdline方式（单次执行的推荐方式）一次只能执行一次任务
- 可以通过外部程序或内部机制控制多次启动scrapy
  - CrawlerProcess
  - CrawlerRunner（官方文档推荐方式）
  - multiprocessing

# CrawlerRunner CrawlerProcess

## 运行多个爬虫

- Crawler对象：由crawl方法创建以执行爬虫任务
- reactor：twisted的反应器对象，用来控制任务的异步执行
- 继承关系：两者都是可以控制爬虫任务Crawler的类，CrawlerRunner->CrawlerProcess

```
settings = get_project_settings()
process = CrawlerProcess(settings)
process.crawl(MySpider1)
process.crawl(MySpider2)
process.start()
```

*# the script will block here until all crawling jobs are finished*

```
from scrapy.crawler import CrawlerRunner
from scrapy.utils.log import configure_logging
from twisted.internet import reactor

from test2.spiders.spider2 import firstHandSpider
from test2.spiders.spider import secondHandSpider

configure_logging()
runner = CrawlerRunner()
runner.crawl(firstHandSpider)
runner.crawl(secondHandSpider)
d = runner.join()
d.addBoth(lambda _: reactor.stop())

reactor.run()
```



## multiprocessing

### 进程级实现定时执行爬虫任务

- 仍通过cmdline方式调用爬虫
- 单个爬虫的执行频率
- 多个爬虫间的执行间隔

```
from multiprocessing import Process
from scrapy import cmdline
import time
import logging
# 配置参数即可, 爬虫名称, 运行频率
confs = [
    {
        "spider_name": "lianjia-1st",
        "frequency": 2,
    },
]
def start_spider(spider_name, frequency):
    args = ["scrapy", "crawl", spider_name]
    while True:
        start = time.time()
        p = Process(target=cmdline.execute, args=(args,))
        p.start()
        p.join()
        logging.debug("### use time: %s" % (time.time() - start))
        time.sleep(frequency)

if __name__ == '__main__':
    for conf in confs:
        process = Process(target=start_spider,
                          args=(conf["spider_name"], conf["frequency"]))
        process.start()
        time.sleep(10)
```

## 多个爬虫需要做的修改

- 每个爬虫在spiders目录下编写各自的spider
- items编写各自的item类
- pipelines编写各自的管道类
- settings 添加多个管道名的激活
- 主脚本begin.py（也可以是别的名字）中选用crawler的相关方式替代cmdline方式

# spider

分别编写两个spider，  
初始化数据中增加  
custom\_settings，作用  
是为每个爬虫指定自己  
使用的管道

```
import scrapy
from test2.items import firstHandItem # 从items.py中引入MyItem对象

class firstHandSpider(scrapy.spiders.Spider):
    name = "lianjia-1st" # 爬虫的名字是lianjia
    allowed_domains = ["bj.fang.lianjia.com/"] # 允许爬取的网站域名
    custom_settings = {
        'ITEM_PIPELINES': {'test2.pipelines.firstHandPipeline': 300},
    } # 指定管道
```

```
import scrapy
from test2.items import secondHandItem # 从items.py中引入对应的Item对象
```

```
class secondHandSpider(scrapy.spiders.Spider):
    name = "lianjia-2nd" # 爬虫的名字是lianjia
    allowed_domains = ["bj.lianjia.com/"] # 允许爬取的网站域名
    custom_settings = {
        'ITEM_PIPELINES': {'test2.pipelines.secondHandPipeline': 300},
    } #指定管道
    start_urls = []
```

## items

分别为每个spider定义对应的item类

```
import scrapy
```

```
class secondHandItem(scrapy.Item):  
    # define the fields for your item here like:  
    location = scrapy.Field()  
    house_info = scrapy.Field()  
    price_info = scrapy.Field()
```

```
class firstHandItem(scrapy.Item):
```

## pipelines

- 根据需要分别为每个spider定义对应的管道类，也可以共用
- 多个管道可以实现不同的持久化方式，比如不同格式的文件、数据库等

```
import json

class secondHandPipeline(object):
    def open_spider(self, spider):
        try: # 打开json文件
            self.file = open('scrapy-test-lianjia-2nd.json', "w", encoding="utf-8")
        except Exception as err:
            print(err)

    def process_item(self, item, spider):
        dict_item = dict(item) # 生成字典对象
        json_str = json.dumps(dict_item, ensure_ascii=False) + "\n" # 生成json串
        self.file.write(json_str) # 将json串写入到文件中
        return item

    def close_spider(self, spider):
        self.file.close() # 关闭文件

class firstHandPipeline(object):
    def open_spider(self, spider):
```

## settings

- 激活管道的设置中包含需要使用的管道
  - 预设的优先级数字范围1-1000，越小级别越高
  - 建议选择不同的数值

```
BOT_NAME = 'test2'
```

```
SPIDER_MODULES = ['test2.spiders']
```

```
NEWSPIDER_MODULE = 'test2.spiders'
```

```
ITEM_PIPELINES = {'test2.pipelines.secondHandPipeline': 300, 'test2.pipelines.firstHandPipeline': 300}
```

```
# Crawl responsibly by identifying yourself (and your website) on the user-agent
```

```
USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0'
```

# 主脚本

- 即begin.py，也可以是别的名字，比如myCrawler等等
- 可选方式
  - CrawlerProcess
  - CrawlerRunner

```
from scrapy.crawler import CrawlerRunner
from scrapy.utils.log import configure_logging
from twisted.internet import reactor
```

```
from test2.spiders.spider2 import firstHandSpider
from test2.spiders.spider import secondHandSpider
```

```
configure_logging()
runner = CrawlerRunner()
runner.crawl(firstHandSpider)
runner.crawl(secondHandSpider)
d = runner.join()
d.addBoth(lambda _: reactor.stop())

reactor.run()
```



## 常见问题-人机验证

- 如果爬虫任务执行后发现没有有效的数据存储，则需要进行相应的排查
  - 按爬虫执行步骤逐步排查
  - 可能的问题
    - 捕获网页失败
    - 解析网页错误
    - 管道保存。。。
  - 排查手段
    - 日志查询
    - 断点跟踪
    - 网络抓包。。。





- response返回的内容显示触发了网站的安全防护

The screenshot shows a web browser window with the URL `https://bj.fang.lianjia.com/loupan/nhs1pg1`. The page displays a CAPTCHA message in Chinese, which is highlighted by a red box. The message reads: "您无法访问此网站" (You cannot access this website), "网站开启了安全防护策略, 检测到您最近的操作异常" (The website has enabled security protection strategies, and it has detected abnormal operations in your recent actions), "为保证您的正常访问, 请进行人机身份认证" (To ensure your normal access, please perform human-machine identity authentication), and "若认证成功后短时间内未自动跳转至正常访问页, 请尝试手动刷新当前页面" (If you do not automatically jump to the normal access page within a short time after successful authentication, please try manually refreshing the current page). Below the message is a div with the id "captcha". The browser's developer tools are open, showing the HTML structure and the console. The console shows a warning message: "Protected Attributes" and a self object: `self = [firstHandSpider] <firstHandSpider 'lianjia-1st' at 0x2455fc67ac0>`.

## 继续排查

- 可能的原因
  - 访问频次过高
  - 网站拒绝显式爬虫
  - 。 。 。
- 解决措施
  - 降低访问频率和次数
  - 随机伪装浏览器
    - User-Agent、Cookie
- 重新抓取前先人工通过认证

## 人机认证

### 您无法访问此网站

网站开启了安全防护策略，检测到您最近的操作异常

为保证您的正常访问，请进行人机身份认证

若认证成功后短时间内未自动跳转至正常访问页，请尝试手动刷新当前页面



点击按钮进行验证

IP: 223.70.157.162

UUID: 2f835644-0284-4cf5-a54f-c2186ed560e8

时间: 2021-11-29T02:25:51.540Z

网址: <https://bj.fang.lianjia.com/loupan/nhs1pg1/>

[详细信息](#)

## REQUEST函数常用的参数及其说明

- `class scrapy.http.Request(url[, callback, method='GET', headers, body, cookies, meta, encoding='utf-8', priority=0, dont_filter=False, errback, flags])`

参数名称	说明
url	接收string。表示用于请求的网址。无默认值
callback	接收同一个对象中方法。表示回调用于处理响应的方法，未指定则继续使用parse。无默认值
method	接收string。表示请求的方式。默认为“GET”
headers	接收string, dict, list。表示请求的头信息，string表示单个头信息，list 则表示多个头信息，如果为None，那么将不发送HTTP请求头信息。无默认值
meta	接收dict。表示Request.meta属性的初始值。如果给了该参数，dict将会浅拷贝。无默认值
cookies	接收list, dict。表示请求的cookies。无默认值

# 利用CALLBACK可以实现爬取过程中的多次交互

```
# -*- coding: utf-8 -*-
import scrapy
from quotes_toscrape.items import QuotesToscraperItem

class QuotesSpider(scrapy.Spider):
    name = 'quotes'
    allowed_domains = ['quotes.toscrape.com']
    # start_urls = ['http://quotes.toscrape.com/']
    base_url = 'http://quotes.toscrape.com/'

    page = 1
    first_url = 'http://quotes.toscrape.com/page/{}/'
    start_urls = [first_url.format(page)]

    def parse(self, response):
        node_list = response.xpath('//div[@class="quote"]')
        for node in node_list:
            quote = node.xpath('//span[@class="text"]/text()).extract()[0][1:-1]
            author = node.xpath('//small/text()).extract()[0]
            tags = node_list.xpath('//div[@class="tags"]/a/text()).extract()[0]
            href = self.base_url + node.xpath('//small/following-sibling::a/@href').extract()[0]
            yield scrapy.Request(url=href, meta={'quote': quote, 'author': author, 'tags': tags},
                                callback=self.parse_author, dont_filter=True)
```

```
next_url =
response.xpath('//ul[@class="pager"]/li[@class="next"]/a/@href').extract(
)[0]
if next_url is not None:
    yield scrapy.Request(url=self.base_url + next_url,
                        callback=self.parse)

def parse_author(self, response):
    item = QuotesToscraperItem()
    # 组合信息
    item['quote'] = response.meta['quote']
    item['author'] = response.meta['author']
    item['tags'] = response.meta['tags']
    item['born_date'] = response.xpath('//span[@class="author-born-
date"]/text()).extract()[0]
    item['born_location'] = response.xpath('//span[@class="author-
born-location"]/text()).extract()[0][3:]
    # 去掉前后空格
    item['description'] = response.xpath('//div[@class="author-
description"]/text()).extract()[0].strip()
    yield item
```

## scrapy中yield的使用

1. 因为使用的yield，而不是return。parse函数将会被当做一个生成器使用。scrapy会逐一获取parse方法中生成的结果，并判断该结果是一个什么样的类型；
2. 如果是request则加入爬取队列，如果是item类型则使用pipeline处理，其他类型则返回错误信息。
3. scrapy取到第一部分的request不会立马就去发送这个request，只是把这个request放到队列里，然后接着从生成器里获取；

```
def parse(self, response):
    book_urls = response.xpath('//table[@class="list-item"]//a/@href').extract()

    for book_url in book_urls:
        url = self.base_site + book_url
        yield scrapy.Request(url, callback=self.getInfo)

    # 获取下一页
    next_page_url = self.base_site + \
        response.xpath('//table[@class="page-book"]//a[contains(text(),"下一页")]/@href').extract()[0]
    yield scrapy.Request(next_page_url, callback=self.parse)

def getInfo(self, response):
    item = TextInfoItem()
    # 提取信息
    .....
    yield item
```

4. 取尽第一部分的request，然后再获取第二部分的item，取到item了，就会放到对应的pipeline里处理；
5. parse()方法作为回调函数(callback)赋值给了Request，指定parse()方法来处理这些请求 scrapy.Request(url, callback=self.parse)
6. Request对象经过调度，执行生成 scrapy.http.response()的响应对象，并送回给parse()方法，直到调度器中没有Request（递归的思路）
7. 取尽之后，parse()工作结束，引擎再根据队列和pipelines中的内容去执行相应的操作；
8. 程序在取得各个页面的items前，会先处理完之前所有的request队列里的请求，然后再提取items。
9. 这一切的一切，Scrapy引擎和调度器将负责到底。

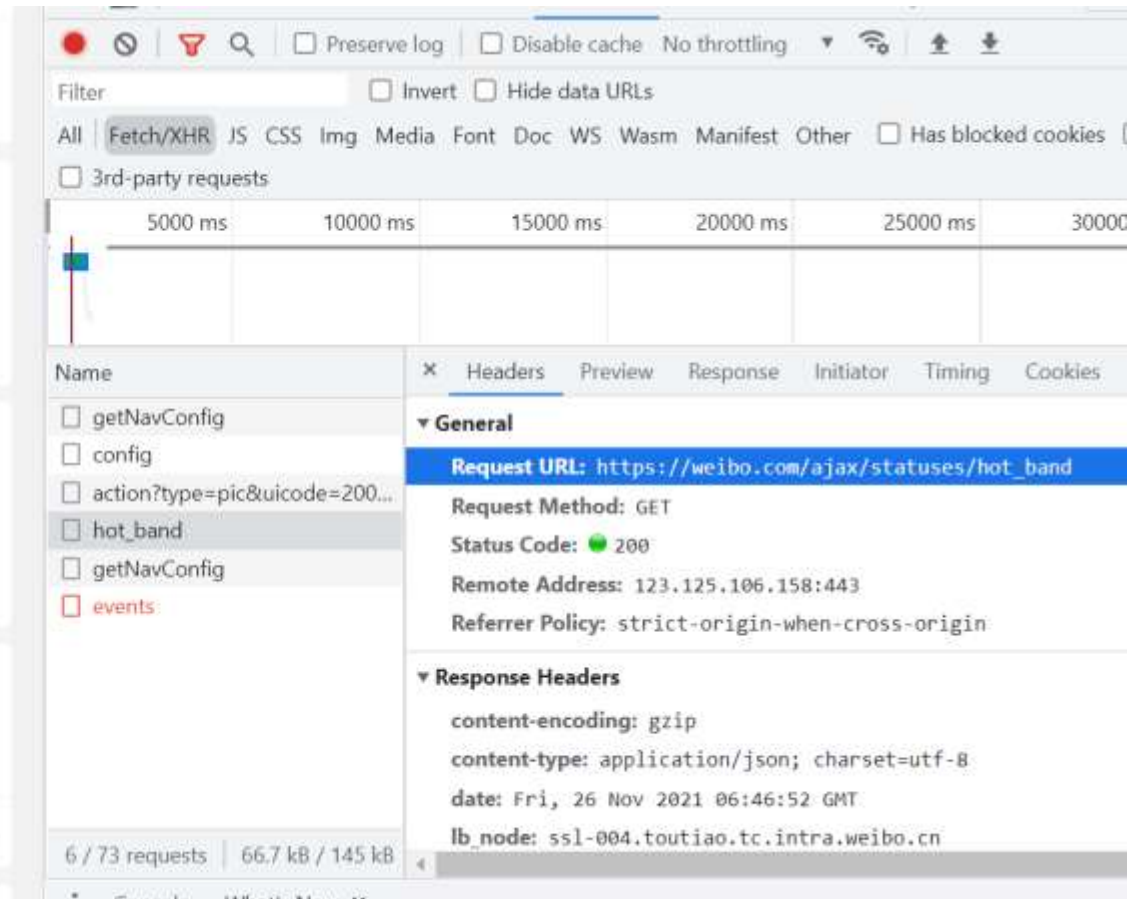
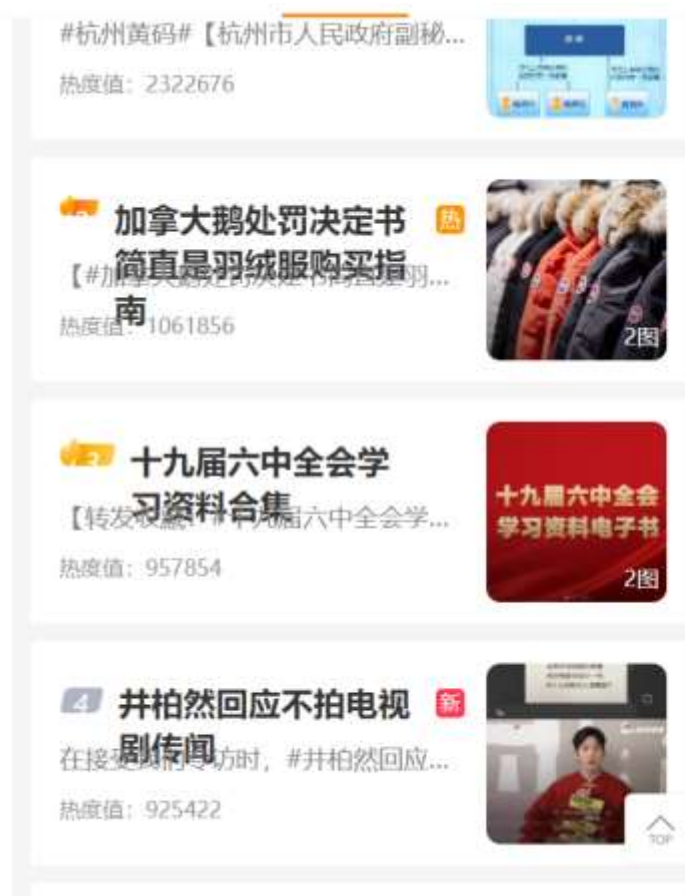
# 动态页面带来的问题

- 需要爬取的页面数据动态加载
  - Ajax
  - JS。。。
- 动态请求后二次得到的响应形式多样
  - html或xml文档
  - json格式的序列化数据。。。
- 更多样的探测验证



## 微博热搜 (未登录)

- Ajax(Asynchronous JavaScript and XML)加载的页面
- 检查-网络-XHR请求，找到返回内容的项目，有的可以直接下载 (json格式)



# 返回json格式数据

request-test.py × request-test-lianjia.py × json-weibo.py × etree-test-xuetang.py × 北京邮电大学.html × package.json ×

test.html

hot.html

g-test.html

gzaixian.html

国大学.html

表达式.py

on-weibo ×

25 page = requests.get('https://weibo.com/ajax/statuses/hot\_band')

26

27 data\_frame = json.loads(page.text)

28 print(data\_frame)

main()

Console

Variables

data\_frame = {dict: 3} {'ok': 1, 'http\_code': 200, 'data': {'band\_list': [{'label\_name': '热', 'star\_word': 0, 'word': '爷爷用原价给我买了四个蛋挞', 'category': '情感', 'num': 2100330, 'subject\_queries': ''}, {'label\_name': '新', 'star\_word': 0, 'word': '日本将宣布禁止所有外国旅客入境', 'category': '国外要闻,旅游', 'num': 1551620, 'realpos': 2, 'flag': 0, 'onboard\_time': 1638120000}, {'label\_name': '新', 'star\_word': 0, 'word': '实现中华民族伟大复兴的康庄大道', 'category': '国内要闻', 'num': 1494729, 'realpos': 3, 'flag': 0, 'onboard\_time': 1638120000}, {'label\_name': '新', 'is\_new': 1, 'star\_word': 0, 'word': '莎头', 'category': '体育', 'num': 1479598, 'realpos': 4, 'flag': 1, 'icon\_desc': '剃'}, {'label\_name': '新', 'is\_new': 1, 'star\_word': 0, 'word': '金鸡奖提名名单', 'category': '影视', 'num': 1336061, 'realpos': 5, 'flag': 8193, 'icon\_desc': '金鸡奖'}, {'label\_name': '新', 'is\_new': 1, 'star\_word': 0, 'word': '王楚钦孙颖莎世乒赛混双夺冠', 'category': '体育', 'num': 912963, 'realpos': 6, 'flag': 0, 'onboard\_time': 1638120000}, {'label\_name': '新', 'star\_word': 0, 'word': '小红书崩了', 'category': '科技', 'num': 874756, 'realpos': 7, 'flag': 0, 'onboard\_time': 1638120000}, {'label\_name': '新', 'is\_new': 1, 'star\_word': 0, 'word': '东北室外冻饺子被喜鹊叼走', 'category': '社会新闻,幽默', 'num': 686669, 'realpos': 8, 'flag': 0, 'onboard\_time': 1638120000}, {'label\_name': '新', 'is\_new': 1, 'star\_word': 0, 'word': '当尤娜遇上魏莱', 'category': '影视', 'num': 655634, 'realpos': 9, 'flag': 1, 'icon\_desc': '当尤娜遇上魏莱'}, {'label\_name': '新', 'star\_word': 0, 'word': '小狗掉50米深洞6年后被救出', 'category': '社会新闻', 'num': 645973, 'realpos': 10, 'flag': 0, 'onboard\_time': 1638120000}, {'label\_name': '新', 'is\_new': 1, 'star\_word': 0, 'word': '杨紫雪中红衣氛围感', 'category': '影视', 'num': 644411, 'realpos': 11, 'flag': 2, 'icon\_desc': '杨紫雪中红衣氛围感'}]}

ODO

Problems

Terminal

Python Packages

Python Console



# B站弹幕

- 弹幕文件统一放在<https://comment.bilibili.com/>上可供下载
- 文件为cid的数字序列，可以通过浏览器->检查->控制台输入cid查看

1	出现时间点	模式	字体	颜色	发送时间	弹幕池	用户ID	rowID	undef	content
2	472.467	1	25	16777215	1.64E+09	0 b975ff37	5.82E+16	10		所以同时的概念是
3	491.508	1	25	16777215	1.64E+09	0 b3bd5a69	5.82E+16	10		我不李姐
4	505.643	5	25	41194	1.64E+09	0 2ab39062	5.82E+16	10		很好的诠释了，万
5	333.894	1	25	16777215	1.64E+09	0 ef803c50	5.82E+16	10		因为光速比我们的
6	492.703	1	25	16777215	1.64E+09	0 d1e2f87d	5.82E+16	10		初中生听懂的视频
7	310.059	1	25	16777215	1.64E+09	0 8cc9276f	5.82E+16	10		特征值与特征向量
8	358.206	1	25	16777215	1.64E+09	0 4.27E+35	5.82E+16	10		时间是有密度的
9	72.019	1	25	16777215	1.64E+09	0 9.00E+18	5.82E+16	10		结果是烟花刚刚放
10	483.826	1	25	16777215	1.64E+09	0 efd138be	5.82E+16	10		还在说“他在试图
11	461.059	1	25	16777215	1.64E+09	0 98be7dc2	5.82E+16	10		空间与时间有一致
12	537.659	1	25	16777215	1.64E+09	0 d46aa757	5.82E+16	10		在速度变化的瞬间
13	146.837	1	25	16777215	1.64E+09	0 4c04fb84	5.82E+16	10		光猫，哈哈哈
14	540.095	1	25	16777215	1.64E+09	0 e58ec728	5.82E+16	10		转向，导致地球日
15	420.394	1	25	16777215	1.64E+09	0 b4a96753	5.82E+16	10		我的天呐，作为文
16	18.539	1	25	16777215	1.64E+09	0 ec81b540	5.82E+16	10		好耶！3b1b的风
17	162.747	1	25	16777215	1.64E+09	0 e72ff1d4	5.82E+16	10		大早上居然也会跟
18	11.458	1	25	16777215	1.64E+09	0 4d2e81e4	5.82E+16	10		我懂了！！



## 读取弹幕

- 通过cid  
直接获取  
弹幕文件
- 解析xml  
文档
- 存入csv  
文件

```
import requests

cid = 450189304 # 弹幕数据文件id

url = 'https://comment.bilibili.com/{}.xml'.format(cid) # 组合url
rq = requests.get(url)

from bs4 import BeautifulSoup
rq.encoding = 'utf-8' # 为获取到的网页指定编码方式
soup = BeautifulSoup(rq.text, 'lxml') # 解析网页数据
tmp = soup.select('d') # 定位弹幕数据标签
danmu = [i.text for i in tmp] # 获取弹幕文本
data = [i.get('p').split(',') for i in tmp] # 获取弹幕数据其他信息

import pandas as pd
data = pd.DataFrame(data) # 转为数据框
data.columns = ['出现时间点', '模式', '字体', '颜色', '发送时间', '弹幕池', '用户ID', 'rowID', 'undef']
data['content'] = danmu # 添加弹幕文本数据
data.to_csv('content.csv', index=None, encoding='utf-8-sig', mode='w') # 保存数据
```

## 降低分析网页难度的方式

- 通过仿真浏览器访问更简便的获取页面信息
  - selenium、phantomJS
- 主要用作测试，也能规避http交互分析的一些难点和复杂度
- 缺点是速度较慢



# SELENIUM使用：基本概念

- 如果想抓取页面中的链接里面的内容怎么办？包括怎么抓取需要登录的网页的信息？
- selenium模块可以直接控制浏览器，实际点击链接，填写登录信息，几乎就像是有一个人在与页面交互。用途之一，可以解决上面的问题。pip install selenium。
- 安装之后，还需要安装对应浏览器的driver程序。selenium2默认自己已经内置了火狐浏览器的driver。selenium3.0后，火狐提供了geckodriver作为driver程序，也可以使用chrome等其他浏览器。
  - 建议浏览器使用火狐浏览器：<http://getfirefox.com> 或chrome浏览器
  - 下载geckodriver需要访问github，使用国内镜像（选择对应操作系统的版本）：  
<https://github.com.cnpmjs.org/mozilla/geckodriver/releases/>
  - chromedriver放在venv\Scripts目录下，下载[chromedriver.storage.googleapis.com/index.html](http://chromedriver.storage.googleapis.com/index.html)
  - 无论用哪种浏览器仿真，注意driver版本需要和浏览器版本匹配

# SELENIUM使用：CHROMEDRIVER下载

- 通过镜像方式访问下载页面，选择匹配版本，选择对应的操作系统版本。
- 下载之后，将解压后得到的“chromedriver.exe”文件放入Python项目虚拟环境路径的Scripts下面。  
如：C:\工作目录\北邮\code\python\project1\venv\Scripts





# SELENIUM使用：打开网页

- 导入selenium模块。
- 使用webdriver方法创建driver对象。
- 用get方法打开输入网址的网页。  
`https://bbs.byrcn/index`
- Python自动调用谷歌浏览器，并打开了输入的网页（北邮人论坛）。

```
from selenium import webdriver # 导入selenium中的webdriver包
url = 'https://bbs.byrcn/index' # 北邮人首页网址
page = webdriver.Chrome() # 选择浏览器
page.get(url) # 获取页面
```



## SELENIUM使用：查找元素

- **`find_element()`**方法。返回一个WebElement对象，代表页面中匹配查询的第一个元素。

`find_element(by,value)`

- **`find_elements()`**方法。返回WebElement对象的列表，包含页面中所有匹配的元素。

`find_elements(by,value)`

#第1个参数，可传入下面取值；第2个参数即为对应的内容

'**class name**' #value为类的名字

'**css selector**' #value为CSS选择器

'**id**' #value为id的值

'**link text**' #value为完整匹配给定的text的值

'**partial link text**' #value为部分匹配text的值

'**tag name**' #value为标签的取值

'**xpath**' #value为xpath表达式的值

`user_list=byr_driver.find_element('class name','c-total')`

#根据class name进行选择;user\_list.text即为北邮人当前总在线人数

`user_list=byr_driver.find_element('css selector','aside[id="bot_info"]>span')`

#根据class selector进行选择；user\_list[0].text即为当前总在线人数

## SELENIUM使用：查看元素信息

对于查找到的元素，  
可以查看其相关信息。

```
#比如，假设已经用上一页的方法选中一个对象user_list
print(user_list.text) #该元素内的文本
print(user_list.tag_name) #输出标签名，例如 'a'表示<a>元素
print(user_list.get_attribute('name')) #输出该元素 name 属性的值
print(user_list.is_displayed()) #如该元素可见，返回True，否则False
print(user_list.is_enabled()) #对于输入元素，如果该元素启用，返回True，否则
返回 False
print(user_list.is_selected()) #对于复选框或单选框元素，如果该元素被选中，选
择 True，否则返回 False
print(user_list.location) #一个字典，包含键'x'和'y'，表示该元素在页面上的位置。
如{'x': 628, 'y': 502}
```



## SELENIUM使用：执行其他操作

对于查找到的元素，可以对其进行操作。

#比如，假设已经用上一页的方法选中一个对象`html_obj`

`html_obj.clear()` #对于文本字段或文本区域元素，清除其中输入的文本

`html_obj.click()` #这个方法可以用于链接跳转，选择单选按钮，点击提交按钮，或者触发该元素被鼠标点击时发生的任何事情

`html_obj.send_keys()` #对于文本字段的<input>或<textarea>元素，然后调用 `send_keys()`方法，如给用户名、口令等元素输入信息

`html_obj.submit()` #在提交按钮或表单的任何元素上调用 `submit()`方法，都等同于点击该元素所在表单的 Submit提交按钮，向server端发送填好的表单

# SELENIUM使用：发送特殊键和浏览器其他操作

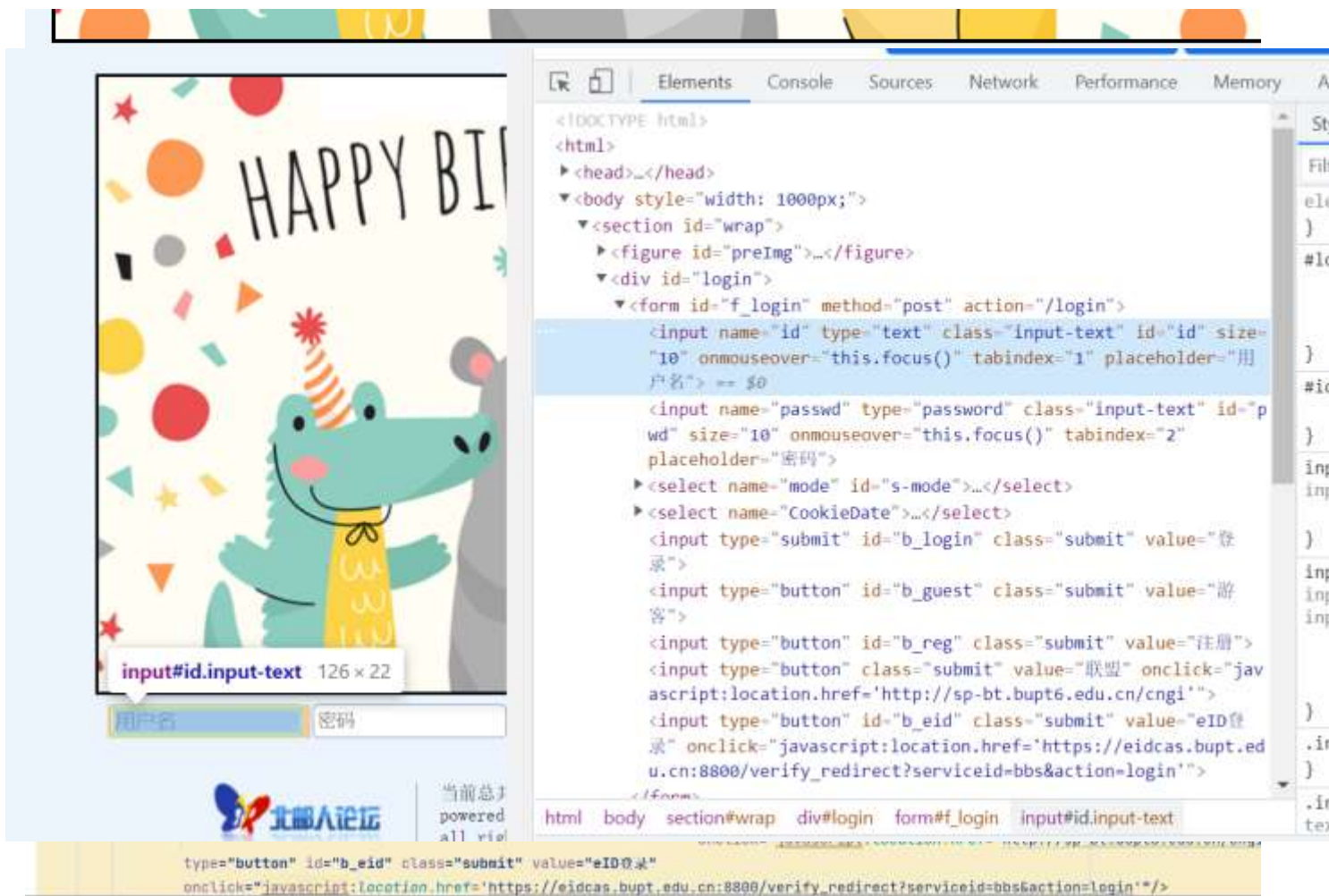
## 输入特殊键

- 可以发送一些特殊的键值。先导入后，即可像右侧一样发送特殊键。***from selenium.webdriver.common.keys import Keys***
- 执行等效浏览器工具栏的操作。
  - `.back()` #点击“返回”按钮
  - `.forward()` #点击“前进”按钮
  - `.refresh()` #点击“刷新”按钮
  - `.quit()` 点击 # “关闭窗口”按钮

***Keys.DOWN, Keys.UP, Keys.LEFT, Keys.RIGHT*** #箭头键  
***Keys.ENTER, Keys.RETURN*** #回车和换行键  
***Keys.HOME, Keys.END,***  
***Keys.PAGE\_DOWN, Keys.PAGE\_UP*** #Home 键、End 键、PageUp 键和 Page Down 键  
***Keys.ESCAPE, Keys.BACK\_SPACE, Keys.DELETE*** #Esc、Backspace 和字母键  
***Keys.F1, Keys.F2, ..., Keys.F12*** #键盘顶部的 F1到 F12键  
***Keys.TAB*** #Tab 键

# 典型抓取场景：登录北邮人抓取当日十大

- 手动打开北邮人首页，查看源代码，观察一下用户名密码在什么地方。
- 在id为'login'的div里面，有一个id为'f\_login'的form表单，然后有一个name为'id'的input输入框（用户名），有一个name为'passwd'的input输入框（密码），有一个id为'b\_login'的input（登录）。



- 使用selenium，打开网页，输入用户名、密码，点击登录按钮。

name为‘id’的input输入框（用户名）；name为‘passwd’的input输入框（密码）；id为‘b\_login’的input（登录）按钮



```
import time
from selenium import webdriver # 导入selenium中的webdriver包

url = 'https://bbs.byrcn/index' #北邮人首页网址
byr_page = webdriver.Chrome() #选择浏览器
byr_page.get(url) #获取页面

#用户名输入
user_name = byr_page.find_element_by_css_selector('input[name="id"]')
user_name.send_keys('bupt2021123')
time.sleep(1)
#密码输入
user_pwd = byr_page.find_element_by_css_selector('input[name="passwd"]')
user_pwd.send_keys('bupt2021')
time.sleep(1)
#点击提交按钮
login_button = byr_page.find_element_by_css_selector('input[id="b_login"]')
login_button.submit()
time.sleep(10)

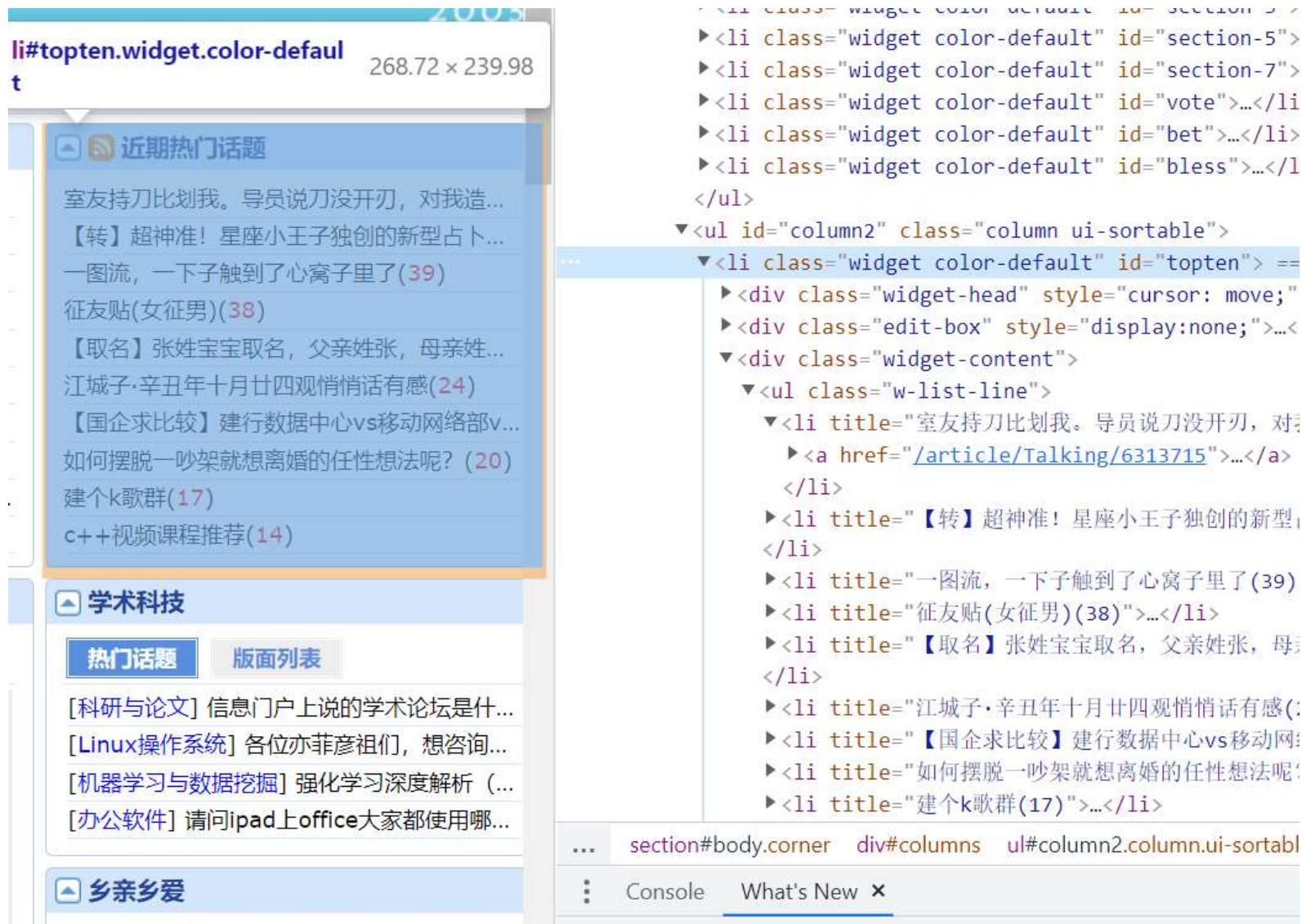
byr_page.close()
```





检查一下近期热门话题的文档路径。

在一个id为'topten'的li对象里，第3个div的ul里面，有10个li对象，它们的title属性就是我们要获取的值。



The screenshot displays a web browser interface with a forum page. The page features a sidebar with categories: '近期热门话题' (Recent Popular Topics), '学术科技' (Academic Technology), and '乡亲乡爱' (Home Love). The '近期热门话题' section is expanded, showing a list of topics with their titles and counts. The developer tools on the right show the DOM tree, with the 'li#topten.widget.color-default' element selected. The DOM tree structure is as follows:

```
li#topten.widget.color-default
  <div class="widget-head">
    <div class="edit-box" style="display:none">
    <div class="widget-content">
      <ul class="w-list-line">
        <li title="室友持刀比划我。导师说刀没开刃，对我造...">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="【转】超神准！星座小王子独创的新型占卜...">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="一图流，一下子触到了心窝子里了(39)">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="征友贴(女征男)(38)">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="【取名】张姓宝宝取名，父亲姓张，母亲姓...">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="江城子·辛丑年十月廿四观悄悄话有感(24)">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="【国企求比较】建行数据中心vs移动网络v...">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="如何摆脱一吵架就想离婚的任性想法呢？(20)">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="建个k歌群(17)">
          <a href="/article/Talking/6313715">...</a>
        </li>
        <li title="c++视频课程推荐(14)">
          <a href="/article/Talking/6313715">...</a>
        </li>
      </ul>
    </div>
  </div>
```

使用路径检查确定的CSS选择器定位到话题内容的元素，调用find\_elements获取到需要的数据

```
login_button = byr_page.find_element_by_css_selector('input[id="b_login"]')
login_button.submit()
time.sleep(10)
#获取近期热门话题并输出
topten_list = byr_page.find_elements_by_css_selector('li[id="topten"]>div:nth-child(3)>ul>li')
print([each.get_attribute('title') for each in topten_list])
```

Selenium x

```
['室友持刀比划我。导员说刀没开刃，对我造不成伤害。（原标题：(64)', '【转】超神准！星座小王子独创的新型占卜、來一起  
試玩一下！(41)', '一图流，一下子触到了心窝子里了(39)', '征友贴(女征男)(38)', '【取名】张姓宝宝取名，父亲姓张  
，母亲姓德（少数民族姓氏）(34)', '江城子·辛丑年十月廿四观悄悄话有感(25)', '【国企求比较】建行数据中心vs移动网络  
部vs工银瑞信(22)', '如何摆脱一吵架就想离婚的任性想法呢？(20)', '建个k歌群(19)', 'c++视频课程推荐(15)']
```

```
> byr_page = {WebDriver}
> login_button = {WebElement}
> topten_list = {list: 10}
01 url = {str} 'https://bbs
> user_name = {WebElement}
> user_pwd = {WebElement}
```

# 典型抓取场景：抓取指定微博新帖

抓取新京报微博的3条最新帖子。

找到新京报的微博地址：

<https://weibo.com/u/1644114654>

查看帖子特点，越新的越靠上。



可以发现：帖子内容都是位于class等于"detail\_wbtext\_4CRf9"的div里面。





```
url = 'https://weibo.com/u/1644114654' #新京报新浪微博页面
weibo_page = webdriver.Chrome() #选择浏览器
weibo_page.get(url) #获取页面
```

```
time.sleep(10)
#获取新闻并输出头条三条
news_list = weibo_page.find_elements_by_css_selector('div[class="detail_wbtext_4CRf9"]')
for each in range(3):
    print(f'第{each+1}个帖子' + news_list[each].text + '\n\n')
```

#### Python Console

第1个帖子【#云南困深洞6年的小狗被救助者领养#】11月29日，云南曲靖一只小狗困在深洞6年后被救起引发关注。11月14日，义务救援队看到村民刘伍兵的求助信息后对被困小狗进行救援。#救助村民讲述小狗被困深洞6年#称，目前小狗情况正常，已做消毒处理，后续会打疫苗，自己已领养了这只小狗。我们视频暖心闻的微博视频

第2个帖子【“20年合同”是否存在利益关联，并不影响公众对这这起食物中毒事件的追问】近日，河南省封丘县“30余名师生餐后集体呕吐腹泻”“校长痛哭称换不动送餐公司”成为网民热议话题。校长一哭，不仅公开了当地有关各方在学生餐供给上的激烈竞争关系，也不可避免地扯出了之前承包合同的枝枝节节。

就目前而 ...展开

第3个帖子【#广州一民办学校将停办千余学生无学可上# 千名外来工子女：不愿再当留守儿童】有家长报料孩子就读的黄埔崇德实验学校将面临停办，且附近因旧改无合适的转学学校，担心孩子上学问题。据悉，黄埔崇德实验学校是一所九年一贯制民办学校，学生大多为外来务工人员的子女。记者联系到该校，校方表示目前学 ...展开

1) 文本中“...展开”的内容，查找到该类元素对象，对它发送点击，就相当于展开了，就能抓取到全文。

2) weibo有反扒机制，不要频繁打开，会被封禁，如果被封了换个IP或者等待一下。或者模拟登录，再搜索和抓取，会安全一点。

3) 网页分为静态和动态两种，类似北邮人首页的那种就是静态页面；类似论坛里面的网页和微博的这种就是动态网页。静态网页用常规方法都可以；动态网页，简单就用selenium，复杂的话先抓包研究网络访问数据，找到里面的真正数据链接，再抓取。

## 典型抓取场景：抓取指定公众号的文章

- 抓取微信公众号的文章，比较复杂，需要使用抓包工具进行数据包分析。我们使用selenium实现简单的抓取。
- 场景：抓取北京邮电大学微信公众号的5篇文章。
- 问题：公众号的历史文章页面，不能在电脑上直接打开。



- 选择一篇北京邮电大学的最新推文，用浏览器打开，把地址拷贝出来，作为初始访问的页面：  
首批新文科！北邮申报项目全部获批！ <https://mp.weixin.qq.com/s/rKuUWocWBUCeMwOUtsNchQ>
- 观察一下页面结构，每篇推文最后面，有“精彩回顾”会放置本篇之前的两期文章。
- 检查页面元素发现，文本为“精彩回顾”的strong标签，往上回退3层之后的第1个a标签，就是我们要找的下一篇文章。a标签里面，包含了文章标题和链接地址。

**105 × 26.67**

精彩回顾:

听乔建永校长谈：5G已来，6G还有多远



加强针来啦！现场直击~



```
order-box; ">
▼ <section style="box-sizing: border-box;" powered-
by="xiumi.us">
  ▶ <section style="display: inline-block;vertical-al
ign: top;width: 20%;box-sizing: border-box;">...
  </section>
  ▼ <section style="display: inline-block;vertical-al
ign: top;width: 80%;letter-spacing: 0px;padding: 0px
5px;box-sizing: border-box;">
    ▼ <section style="color: rgb(75, 85, 107);box-siz
ing: border-box;" powered-by="xiumi.us">
      ▶ <p style="white-space: normal;margin: 0px;pad
ding: 0px;box-sizing: border-box;">...</p>
      ▼ <p style="white-space: normal;margin: 0px;pad
ding: 0px;box-sizing: border-box;">
        ▼ <span style="font-size: 21px;box-sizing: bor
der-box;">
          <strong style="box-sizing: border-box;">精
彩回顾: </strong> == $0
        </span>
      </p>
    </section>
    ▼ <a title="https://mp.weixin.qq.com/s/?__biz=MjM5
OTg4ODIxNA==&mid=2662503969&idx=1&sn=82fd2a3e5c6d2
=cat=...&scene=0" href="https://mp.weixin.qq.com/s/rKuUWocWBUCeMwOUtsNchQ">
```

观察一下页面代码，文本为“精彩回顾”的strong标签，往上回退3层之后的第1个a标签，就是我们要找的下一篇文章。a标签里面，包含了文章标题和链接地址。通过xpath定位

```
import time
from selenium import webdriver # 导入selenium中的webdriver包

url = 'https://mp.weixin.qq.com/s/rKuUWocWBUCeMwOUtsNchQ' #北邮微信页面
bupt_wechat_page = webdriver.Chrome() #选择浏览器
bupt_wechat_page.get(url) #获取页面

#第一篇直接存入
news_list_dic = {1: '首批新文科！北邮申报项目全部获批！'}
for i in range(2, 7):
    time.sleep(5)
    next_news = bupt_wechat_page.find_element_by_xpath('//strong[text()="精彩回顾："]/../../following-sibling::a[1]')
    news_list_dic[i] = next_news.text
    bupt_wechat_page.get(next_news.get_attribute('href'))

for k,v in news_list_dic.items():
    print(f'第{k}篇 ' + f'文章标题为: {v}')

bupt_wechat_page.close()
```



1) 右边就是输出的结果。如果需要，可以在此基础上，在每页加载后，把url、发布时间、帖子文本等信息都提取出来。

2) 为什么取了5篇，请看右边第6篇——竟然没有“精彩回顾”.....只有这一篇没有...遇到这种情况，特殊处理...

tViaSelenium x

### Python Console

第1篇 文章标题为: 首批新文科! 北邮申报项目全部获批!

第2篇 文章标题为: 听乔建永校长谈: 5G已来, 6G还有多远

第3篇 文章标题为: 加强针来啦! 现场直击~

第4篇 文章标题为: 热议 | 北京邮电大学师生认真学习领会党的十九届六中全会精神

第5篇 文章标题为: 听说这里很酷! 一起去看看吧~

第6篇 文章标题为: 我为师生办实事 | 一封感谢信

>>>

来源: 离退休工作处

排版: 梅皓

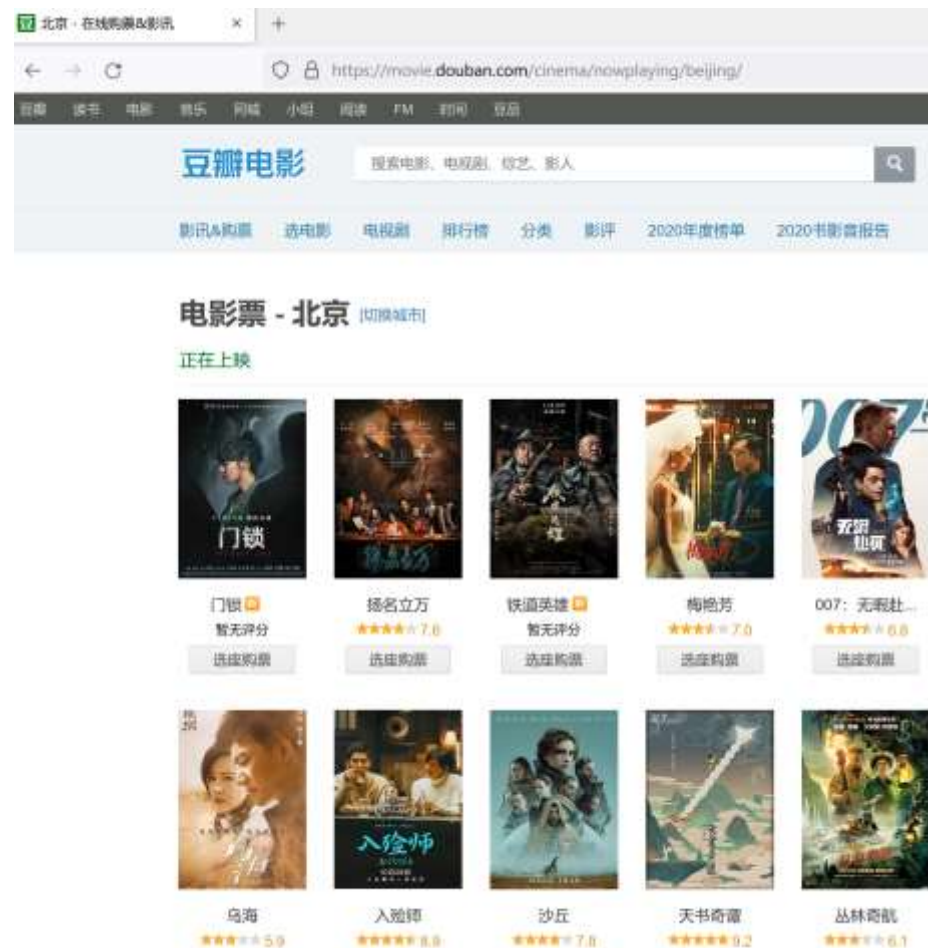
责编: 董思捷

邮苑初雪 如你所愿!



# 典型抓取场景：抓取豆瓣正在上映的高分电影（I）

- 抓取豆瓣当前正在上映的打分最高的三部影片。
- 豆瓣北京正在上映的URL为：  
<https://movie.douban.com/cinema/nowplaying/beijing/>



- 抓取豆瓣正在上映的打分最高的三部影片
- 检查页面元素，在id为nowplaying的div标签下面的第2个div下面的ul中的每个li对象是一部电影。
  - data-title="扬名立万"
  - data-score="7.6"
  - data-director="刘循子墨"
  - data-actors="尹正 / 邓家佳 / 喻恩泰"
- 显示全部影片按钮，对应的事class为“more”的div



```
<script src="//img3.doubanio.com/dae/accounts/resour
defer="defer"></script>
▼ <div id="wrapper">
  ▼ <div id="content">
    <h1>北京 - 在线购票&影讯</h1>
    ▼ <div class="grid-16-8 clearfix">
      ▼ <div class="article">
        ▶ <script id="db-tmpl-subject-tip" type="text/
        ▶ <div id="hd">_</div>
        ▶ <div id="dale_movie_nowplaying_top_left" ad-
        type="CPD" data-type="DoubanRender" data-versi
        ▼ <div id="nowplaying"> == $0
          ▶ <div class="mod-hd">_</div>
          ▼ <div class="mod-bd">
            ▼ <ul class="lists">
              ▼ <li id="35422807" class="list-item" da
              score="7.6" data-star="40" data-release="
              data-region="中国大陆" data-director="刘循
              佳 / 喻恩泰" data-category="nowplaying" da
              "True" data-votecount="238786" data-subje
              ▼ <ul class>
                ▼ <li class="poster">
                  ▼ <a href="https://movie.douban.com
                  ng_poster" class="ticket-btn" target
                  r">
                    
```

通过CSS选择器定位，完成爬取

```
#点击全部电影按钮
all_button = douban_page.find_element_by_css_selector('div[class="more"]')
all_button.click()
time.sleep(5) #等一会儿
movie_list = douban_page.find_elements_by_css_selector('div[id="nowplaying"]>div:nth-child(2)>ul>li')

results_all = []
for each in range(len(movie_list)):
    r_dic = {}
    r_dic['电影名称'] = movie_list[each].get_attribute('data-title')
    r_dic['豆瓣评分'] = movie_list[each].get_attribute('data-score')
    r_dic['导演'] = movie_list[each].get_attribute('data-director')
    r_dic['演员'] = movie_list[each].get_attribute('data-actors')
    results_all.append(r_dic)

results_all = sorted(results_all, key=lambda x : x['豆瓣评分'], reverse=True)

for i in range(5):
    print(results_all[i])
```

```
matViaSelenium x doubanViaSelenium x
{'电影名称': '天书奇谭', '豆瓣评分': '9.2', '导演': '王树忱 钱运达', '演员': '丁建华 / 毕克 / 苏秀'}
{'电影名称': '入殓师', '豆瓣评分': '8.9', '导演': '泷田洋二郎', '演员': '本木雅弘 / 广末凉子 / 山崎努'}
{'电影名称': '南太平洋之旅', '豆瓣评分': '8.4', '导演': '格雷戈·迈吉里弗雷 Stephen Judson', '演员': '凯特·布兰切特 / Ferdiel Ballamu / Menas Mambasar'}
{'电影名称': '永不消逝的电波', '豆瓣评分': '8.0', '导演': '王苹', '演员': '孙道临 / 袁霞 / 王心刚'}
{'电影名称': '沙丘', '豆瓣评分': '7.8', '导演': '丹尼斯·维伦纽瓦', '演员': '蒂莫西·柴勒梅德 / 丽贝卡·弗格森 / 奥斯卡·伊萨克'}

>>>
```

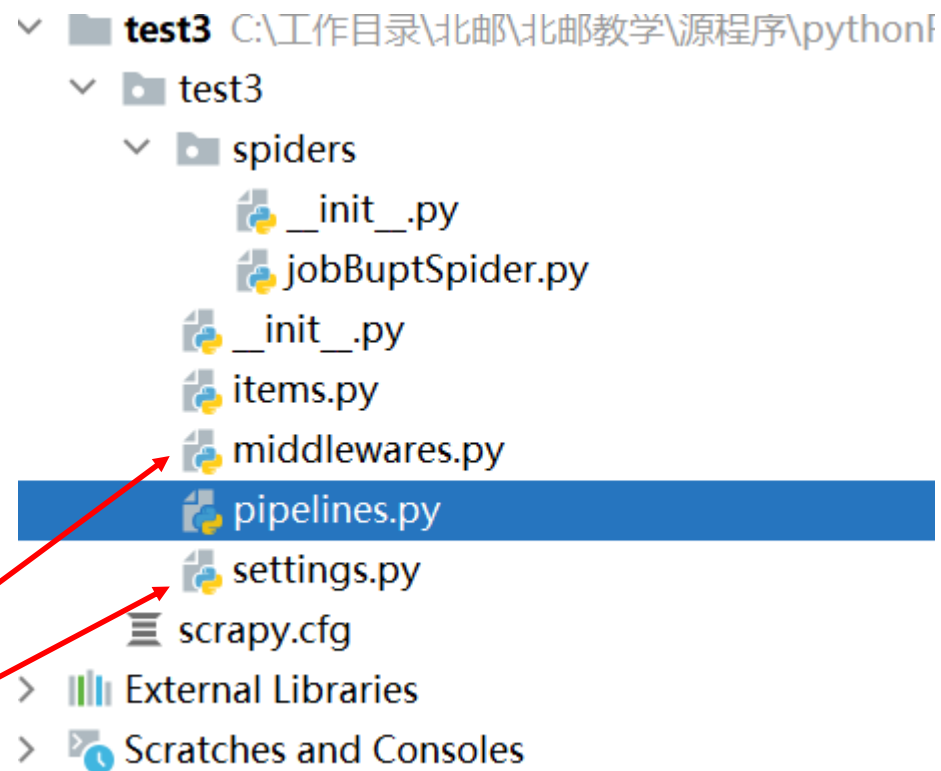


# SCRAPY+SELENIUM

Scrapy对接Selenium进行抓取，采用Downloader Middleware来实现。在Middleware里面的process\_request()方法里对每个抓取请求进行处理，启动浏览器并进行页面渲染，再将渲染后的结果构造一个HtmlResponse对象返回。

注：爬取多级网页时request方法的dont\_filter置为True，避免去重过滤。

主要修改的文件



## 修改 middlewares.py 增加使用selenium 进行网页获取的类 SeleniumMiddlewa re，替代原有下载 中间件

```
# -*- coding: utf-8 -*-
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.chrome.options import Options
from scrapy.http import HtmlResponse
from logging import getLogger

class SeleniumMiddleware():
    def __init__(self, timeout=None, weibo_username=None, weibo_password=None):
        self.logger = getLogger(__name__)
        self.timeout = timeout
        self.weibo_username = weibo_username
        self.weibo_password = weibo_password
        self.chrome_options = Options()
        self.chrome_options.add_argument('--headless')
        self.chrome_options.add_argument('--disable-gpu')
        self.browser = webdriver.Chrome(chrome_options=self.chrome_options)
        self.browser.set_page_load_timeout(self.timeout)
        self.wait = WebDriverWait(self.browser, self.timeout)
```

```
# 淘宝登录
self.browser.get(
    'https://login.taobao.com/member/login.jhtml?redirectURL=http%3A%2F%2Fs.taobao.com%2Fsearch%3Fq%3DiPad')
# 切换到密码登录
login_swith = self.wait.until(EC.element_to_be_clickable((By.CLASS_NAME, 'login-switch')))
login_swith.click()
# 点击微博登录
weibo_login = self.wait.until(EC.presence_of_element_located((By.CLASS_NAME, 'weibo-login')))
weibo_login.click()
# 输入微博用户名
username_input = self.wait.until(EC.presence_of_element_located((By.NAME, 'username')))
username_input.send_keys(self.weibo_username)
# 输入微博密码
password_input = self.browser.find_element_by_name('password')
password_input.send_keys(self.weibo_password)
# 提交
weibo_submit = self.browser.find_element_by_class_name('W_btn_g')
weibo_submit.send_keys(Keys.ENTER)

def __del__(self):
    self.browser.close()
```

```
@classmethod
```

```
def from_crawler(cls, crawler):
```

```
    return cls(timeout=crawler.settings.get('SELENIUM_TIMEOUT'),
               weibo_username=crawler.settings.get('WEIBO_USERNAME'),
               weibo_password=crawler.settings.get('WEIBO_PASSWORD'))
```

```
def process_request(self, request, spider):
```

```
    """
```

```
    用PhantomJS抓取页面
```

```
    :param request: Request对象
```

```
    :param spider: Spider对象
```

```
    :return: HtmlResponse
```

```
    """
```

```
self.logger.debug('Chrome headless is Starting')
```

```
page = request.meta.get('page', 1)
```

```
try:
```

```
    # 不是第一页，跳转翻页
```

```
        if page > 1:
```

```
            # 跳转页面输入框
```

```
                input = self.wait.until(
                    EC.presence_of_element_located((By.CSS_SELECTOR, '#mainsrp-pager div.form > input')))
```

```
            # 跳转确定
```

```
                submit = self.wait.until(
                    EC.element_to_be_clickable((By.CSS_SELECTOR, '#mainsrp-pager div.form > span.btn.J_Submit')))
```

```
input.clear()
# 输入需要跳转的页面
input.send_keys(page)
submit.click()
# 等待页面跳转
self.wait.until(
    EC.text_to_be_present_in_element((By.CSS_SELECTOR, '#mainsrp-pager li.item.active > span'), str(page)))
# 下拉界面到最下方
self.browser.execute_script('window.scrollTo(0, document.body.scrollHeight)')
# 等待商品页面加载完毕
self.wait.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, '.m-itemlist .items .item'))))
return HtmlResponse(url=request.url, body=self.browser.page_source, request=request, encoding='utf-8',
                    status=200)
except TimeoutException:
    return HtmlResponse(url=request.url, request=request, status=500)
```

在settings.py里，我们设置调用刚才定义的SeleniumMiddleware

```
DOWNLOADER_MIDDLEWARES = {
    'test3.middlewares.SeleniumMiddleware': 543,
}
```

爬虫项目的名称

# ROBOTS协议

- 网络爬虫引发的问题
  - 对网络性能的影响
  - 法律风险
  - 隐私泄露
- Robots协议
  - 来源审查：判断User-Agent进行限制 检查来访HTTP协议头的User-Agent域，只响应浏览器或友好爬虫的访问
  - 发布公告：Robots协议 告知所有爬虫网站的爬取策略，要求爬虫遵守

- 示例：百度的Robots协议文件  
<https://www.baidu.com/robots.txt>
- \*代表所有
- /代表根目录
- ?代表含问号的路径

```
User-agent: Baiduspider
Disallow: /baidu
Disallow: /s?
Disallow: /ulink?
Disallow: /link?
Disallow: /home/news/data/
Disallow: /bh
```

```
User-agent: Googlebot
Disallow: /baidu
Disallow: /s?
Disallow: /shifen/
Disallow: /homepage/
Disallow: /cpro
Disallow: /ulink?
Disallow: /link?
Disallow: /home/news/data/
Disallow: /bh
```

...

## 常见的反爬虫攻防

- 封锁间隔时间：settings设置 DOWNLOAD\_DELAY，延长间隔
- 封锁cookies：settings设置COOKIES\_ENABLED=False，禁用cookies
- 封锁user-agent：middlewares中增加UserAgents数据，即浏览器伪装
- 封锁IP：middlewares中增加PROXIES数据，即IP池



## 作业

- 爬取学堂在线的合作院校页面内容学堂在线 - 精品在线课程学习平台 ([xuetaangx.com](http://xuetaangx.com))要求将开课院校的学校名称和对应的课程数量，保存到一个json文件中。例如 “清华大学,308”
- 爬取链家官网新房的数据（爬取3-5页即可） 起始页面地址：北京楼盘|新开盘楼盘|房价信息\_北京楼盘(北京链家新房) ([lianjia.com](http://lianjia.com))要求将楼盘名称、房间数、平米数保存到json文件中

# 作业

- 编写Python程序，方式不限，完成如下任务：
  - 编写网页抓取程序，抓取北邮人论坛的求职信息板块中，日期为“昨天”“0点至23点59分59秒”的所有帖子的标题。（最多向前翻5页；如果某些昨天的帖子超出在5页之外，就不抓了）
  - 抓取完成后，将上述列表中的数据写入一个csv文件，第1列为序号，第2列为标题名字。该CSV文件的名字为“BYR-JOB-YYYY-MM-DD”。
  - 爬取网页频次不宜过高

# 作业

## ■ 任务要求

- 酷狗榜单酷狗飙升榜\_排行榜\_乐库频道\_酷狗网 (kugou.com) 为定向爬取对象。
- 通过每个榜单链接获取每个榜单可显示的歌曲（一般为22首）以及其后10页
- 每个榜单存储到一个单独的csv文件，文件名为kugou-XX榜.csv，内容包括rank,singer,title,times

- HTML简介
- JSON和Xpath
- Scrapy库
- 静态页面的数据获取
- 动态页面的数据获取

九  
数  
据  
获  
取  
（  
爬  
虫  
）



# 谢谢

