Day06回顾

多线程爬虫

- **使用流程**

```
# 1、URL队列
q.put(url)
# 2、线程事件函数
while True:
    if not url_queue.empty():
        ...get()、请求、解析
    else:
        break
# 创建并启动线程
t_list = []
for i in range(5):
    t = Thread(target=parse_page)
    t_list.append(t)
    t.start()
# 阻塞等待回收线程
for i in t_list:
    i.join()
```

json模块

- **json转python**

```
变量名 = json.loads(res.text))
```

- **python转json（保存为json文件）**

```
# 保存所抓取数据为json数据
with open(filename,'a') as f:
    json.dump(字典/列表/元组,f,ensure_ascii=False)
```

selenium+phantomjs/chrome/firefox

- **特点**

```
1、简单，无需去详细抓取分析网络数据包，使用真实浏览器
2、需要等待页面元素加载，需要时间，效率低
```

- **使用流程**

```
from selenium import webdriver

# 创建浏览器对象
browser = webdriver.Firefox()
browser.get('https://www.jd.com/')

# 查找节点
node = browser.find_element_by_xpath('')
node.send_keys('')
node.click()

# 获取节点文本内容
content = node.text

# 关闭浏览器
browser.quit()
```

- **设置无界面模式（chromedriver | firefox）**

```
options = webdriver.ChromeOptions()
options.add_argument('--headless')

browser = webdriver.Chrome(options=options)
browser.get(url)
```

京东爬虫

- **执行JS脚本,把进度条拉到最下面**

```
1、js脚本
browser.execute_script(
'window.scrollTo(0,document.body.scrollHeight)'
)
2、利用节点对象的text属性获取当前节点及后代节点的文本内容,想办法处理数据
```

scrapy框架

- 五大组件

```
引擎（Engine）
爬虫程序（Spider）
调度器（Scheduler）
下载器（Downloader）
管道文件（Pipeline）
# 两个中间件
下载器中间件（Downloader Middlewares）
蜘蛛中间件（Spider Middlewares）
```

- 工作流程

```
1、Engine向Spider索要URL,交给Scheduler入队列
2、Scheduler处理后出队列,通过Downloader Middlewares交给Downloader去下载
3、Downloader得到响应后,通过Spider Middlewares交给Spider
4、Spider数据提取：
   1、数据交给Pipeline处理
   2、需要跟进URL,继续交给Scheduler入队列，依次循环
```

- 常用命令

```
# 创建爬虫项目
scrapy startproject 项目名

# 创建爬虫文件
cd 项目文件夹
scrapy genspider 爬虫名 域名

# 运行爬虫
scrapy crawl 爬虫名
```

- scrapy项目目录结构

```
Baidu
├── Baidu               # 项目目录
│   ├── items.py        # 定义数据结构
│   ├── middlewares.py  # 中间件
│   ├── pipelines.py    # 数据处理
│   ├── settings.py     # 全局配置
│   └── spiders
│       ├── baidu.py    # 爬虫文件
└── scrapy.cfg          # 项目基本配置文件
```

- settings.py全局配置

```
1、USER_AGENT = 'Mozilla/5.0'
2、ROBOTSTXT_OBEY = False
3、CONCURRENT_REQUESTS = 32
4、DOWNLOAD_DELAY = 1
5、DEFAULT_REQUEST_HEADERS={}
6、ITEM_PIPELINES={'项目目录名.pipelines.类名':300}
```

Day07笔记

selenium补充

切换页面

1、适用网站

```
页面中点开链接出现新的页面，但是浏览器对象browser还是之前页面的对象
```

2、应对方案

```
# 获取当前所有句柄（窗口）
all_handles = browser.window_handles
# 切换到新的窗口
browser.switch_to_window(all_handles[1])
```

3、民政部网站案例

3.1 目标: 将民政区划代码爬取到数据库中，按照层级关系（分表 -- 省表、市表、县表）

3.2 数据库中建表

```
# 建库
create database govdb charset utf8;
use govdb;
# 建表
create table province(
p_name varchar(20),
p_code varchar(20)
)charset=utf8;
create table city(
c_name varchar(20),
c_code varchar(20),
c_father_code varchar(20)
)charset=utf8;
create table county(
x_name varchar(20),
x_code varchar(20),
x_father_code varchar(20)
)charset=utf8;
```

3.3 思路

```
1、selenium+Chrome打开一级页面，并提取二级页面最新链接
2、增量爬取：和数据库version表中进行比对，确定之前是否爬过（是否有更新）
3、如果没有更新，直接提示用户，无须继续爬取
4、如果有更新，则删除之前表中数据，重新爬取并插入数据库表
5、最终完成后：断开数据库连接，关闭浏览器
```

3.4 代码实现

```python
from selenium import webdriver
import time
import pymysql

class GovementSpider(object):
    def __init__(self):
        # 创建浏览器对象 + 打开一级页面
        self.options = webdriver.ChromeOptions()
        self.options.add_argument('--headless')
        self.browser = webdriver.Chrome(options=self.options)

        self.one_url = 'http://www.mca.gov.cn/article/sj/xzqh/2019/'
        # 数据库相关变量
        self.db = pymysql.connect('192.168.153.134', 'tiger', '123456', 'govdb', charset='utf8')
        self.cursor = self.db.cursor()
        # 定义数据库中三张表的列表，后续使用executemany方法将数据插入数据库
        self.province_list = []
        self.city_list = []
        self.county_list = []

    # 获取首页，并提取最新二级页面链接
    def get_two_url(self):
        self.browser.get(self.one_url)
        # 提取最新二级页面链接节点 + 点击该节点
        td_list = self.browser.find_elements_by_xpath('//td[@class="arlisttd"]/a[contains(@title,"中华人民共和国县以上行政区划代码")]')
        if td_list:
            two_url_element = td_list[0]
            # 增量爬取数据库核对
            two_url = two_url_element.get_attribute('href')

        sel = 'select * from version'
        self.cursor.execute(sel)
        result = self.cursor.fetchall()
        if result:
            version_url = result[-1][0]
        else:
            version_url = ''

        # 和数据库中url做比对
        if two_url == version_url:
            print('已是最新，无需爬取')
        else:
            two_url_element.click()
            # 获取当前所有句柄 + 将browser切换到新的页面
```

```
            all_handles = self.browser.window_handles
            self.browser.switch_to.window(all_handles[1])
            time.sleep(5)
            self.get_data()
            # 爬取成功后存入数据库
            ins_version = "insert into version values(%s)"
            self.cursor.executemany(ins_version,[two_url])
            self.db.commit()
            # 断开数据库连接
            self.cursor.close()
            self.db.close()

    # 提取真实所需数据
    def get_data(self):
        tr_list = self.browser.find_elements_by_xpath('//tr[@height="19"]')
        print('正在抓取数据，请稍后... ')
        for tr in tr_list:
            code = tr.find_element_by_xpath('./td[2]').text.strip()
            city = tr.find_element_by_xpath('./td[3]').text.strip()
            print(city,code)

            # 判断层级关系，添加到对应列表中，对应数据库中三张表的字段
            if code[2:] == '0000':
                self.province_list.append([city, code, ])
            if code[4:] == '00':
                self.city_list.append([city, code, code[:2] + '0000'])
            else:
                self.county_list.append([city, code, code[:4] + '00'])


        self.insert_mysql()

    # 数据入库
    def insert_mysql(self):
        # 更新时先删除数据库
        del_province = 'delete from province'
        del_city = 'delete from city'
        del_county = 'delete from county'
        self.cursor.execute(del_province)
        self.cursor.execute(del_city)
        self.cursor.execute(del_county)
        # 插入新数据
        ins_province = 'insert into province values(%s,%s)'
        ins_city = 'insert into city values(%s,%s,%s)'
        ins_county = 'insert into county values(%s,%s,%s)'
        print('正在存入数据库，请稍后...')
        self.cursor.executemany(ins_province,self.province_list)
        self.cursor.executemany(ins_city,self.city_list)
        self.cursor.executemany(ins_county,self.county_list)
        self.db.commit()
        print('存入数据库完成')

    def main(self):
        self.get_two_url()

if __name__ == '__main__':
    spider = GovementSpider()
```

```
102     spider.main()
```

3.5 SQL命令练习

```
# 1. 查询所有省市县信息（多表查询实现）
select province.p_name,city.c_name,county.x_name from province,city,county  where province.p_code=city.c_father_code and  city.c_code=county.x_father_code;
# 2. 查询所有省市县信息（连接查询实现）
select province.p_name,city.c_name,county.x_name from province inner join city on province.p_code=city.c_father_code inner join county on  city.c_code=county.x_father_code;
```

Web客户端验证

弹窗中的用户名和密码如何输入?

```
不用输入，在URL地址中填入就可以
```

示例: 爬取某一天笔记

```
from selenium import webdriver

url = 'http://tarenacode:code_2013@code.tarena.com.cn/AIDCode/aid1903/12-spider/spider_day07_note.zip'
browser = webdriver.Chrome()
browser.get(url)
```

scrapy框架

- 创建爬虫项目步骤

```
1、新建项目 ： scrapy startproject 项目名
2、cd 项目文件夹
3、新建爬虫文件 ： scrapy genspider 文件名 域名
4、明确目标(items.py)
5、写爬虫程序(文件名.py)
6、管道文件(pipelines.py)
7、全局配置(settings.py)
8、运行爬虫 ： scrapy crawl 爬虫名
```

- pycharm运行爬虫项目

```
1、创建begin.py(和scrapy.cfg文件同目录)
2、begin.py中内容:
    from scrapy import cmdline
    cmdline.execute('scrapy crawl maoyan'.split())
```

小试牛刀

- 目标

```
打开百度首页，把 '百度一下，你就知道' 抓取下来，从终端输出
```

- 实现步骤

1. **创建项目Baidu 和 爬虫文件baidu**

```
1、scrapy startproject Baidu
2、cd Baidu
3、scrapy genspider baidu www.baidu.com
```

2. **编写爬虫文件baidu.py，xpath提取数据**

```python
# -*- coding: utf-8 -*-
import scrapy

class BaiduSpider(scrapy.Spider):
    name = 'baidu'
    allowed_domains = ['www.baidu.com']
    start_urls = ['http://www.baidu.com/']

    def parse(self, response):
        result = response.xpath('/html/head/title/text()').extract_first()
        print('*'*50)
        print(result)
        print('*'*50)

```

3. **全局配置settings.py**

```
USER_AGENT = 'Mozilla/5.0'
ROBOTSTXT_OBEY = False
DEFAULT_REQUEST_HEADERS = {
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
    'Accept-Language': 'en',
}
```

4. **创建begin.py（和scrapy.cfg同目录）**

```
from scrapy import cmdline

cmdline.execute('scrapy crawl baidu'.split())
```

5. **启动爬虫**

```
直接运行 begin.py 文件即可
```

思考运行过程

猫眼电影案例

- 目标

```
URL: 百度搜索 -> 猫眼电影 -> 榜单 -> top100榜
内容:电影名称、电影主演、上映时间
```

- 实现步骤

1. **创建项目和爬虫文件**

```
# 创建爬虫项目
scrapy startproject Maoyan
cd Maoyan
# 创建爬虫文件
scrapy genspider maoyan maoyan.com
```

2. **定义要爬取的数据结构（items.py）**

```
name = scrapy.Field()
star = scrapy.Field()
time = scrapy.Field()
```

3. **编写爬虫文件（maoyan.py）**

```
1、基准xpath,匹配每个电影信息节点对象列表
    dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
2、for dd in dd_list:
    电影名称 = dd.xpath('./a/@title')
    电影主演 = dd.xpath('.//p[@class="star"]/text()')
    上映时间 = dd.xpath('.//p[@class="releasetime"]/text()')
```

代码实现一

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

class MaoyanSpider(scrapy.Spider):
    # 爬虫名
    name = 'maoyan'
    # 允许爬取的域名
    allowed_domains = ['maoyan.com']
    offset = 0
    # 起始的URL地址
    start_urls = ['https://maoyan.com/board/4?offset=0']

    def parse(self, response):
```

```
        # 基准xpath,匹配每个电影信息节点对象列表
        dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
        # dd_list : [<element dd at xxx>,<...>]
        for dd in dd_list:
            # 创建item对象
            item = MaoyanItem()
            # [<selector xpath='' data='霸王别姬'>]
            # dd.xpath('')结果为[选择器1,选择器2]
            # .extract() 把[选择器1,选择器2]所有选择器序列化为unicode字符串
            # .extract_first() : 取第一个字符串
            item['name'] = dd.xpath('./a/@title').extract_first().strip()
            item['star'] = dd.xpath('.//p[@class="star"]/text()').extract()[0].strip()
            item['time'] = dd.xpath('.//p[@class="releasetime"]/text()').extract()[0]

            yield item

        # 此方法不推荐,效率低
        self.offset += 10
        if self.offset <= 90:
            url = 'https://maoyan.com/board/4?offset={}'.format(str(self.offset))

            yield scrapy.Request(
                url=url,
                callback=self.parse
            )
```

代码实现二

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

class MaoyanSpider(scrapy.Spider):
    # 爬虫名
    name = 'maoyan2'
    # 允许爬取的域名
    allowed_domains = ['maoyan.com']
    # 起始的URL地址
    start_urls = ['https://maoyan.com/board/4?offset=0']

    def parse(self, response):
        for offset in range(0,91,10):
            url = 'https://maoyan.com/board/4?offset={}'.format(str(offset))
            # 把地址交给调度器入队列
            yield scrapy.Request(
                url=url,
                callback=self.parse_html
            )

    def parse_html(self,response):
        # 基准xpath,匹配每个电影信息节点对象列表
        dd_list = response.xpath(
            '//dl[@class="board-wrapper"]/dd')
        # dd_list : [<element dd at xxx>,<...>]

```

```
        for dd in dd_list:
            # 创建item对象
            item = MaoyanItem()
            # [<selector xpath='' data='霸王别姬'>]
            # dd.xpath('')结果为[选择器1,选择器2]
            # .extract() 把[选择器1,选择器2]所有选择器序列化为
            # unicode字符串
            # .extract_first() : 取第一个字符串
            item['name'] = dd.xpath('./a/@title').extract_first().strip()
            item['star'] = dd.xpath('.//p[@class="star"]/text()').extract()[0].strip()
            item['time'] = dd.xpath('.//p[@class="releasetime"]/text()').extract()[0]

            yield item
```

代码实现三

```
# 重写start_requests()方法，直接把多个地址都交给调度器去处理
# -*- coding: utf-8 -*-
import scrapy
from ..items import MaoyanItem

class MaoyanSpider(scrapy.Spider):
    # 爬虫名
    name = 'maoyan_requests'
    # 允许爬取的域名
    allowed_domains = ['maoyan.com']

    def start_requests(self):
        for offset in range(0,91,10):
            url = 'https://maoyan.com/board/4?offset={}'.format(str(offset))
            # 把地址交给调度器入队列
            yield scrapy.Request(url=url,callback=self.parse_html )

    def parse_html(self,response):
        # 基准xpath,匹配每个电影信息节点对象列表
        dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
        # dd_list : [<element dd at xxx>,<...>]

        for dd in dd_list:
            # 创建item对象
            item = MaoyanItem()
            # [<selector xpath='' data='霸王别姬'>]
            # dd.xpath('')结果为[选择器1,选择器2]
            # .extract() 把[选择器1,选择器2]所有选择器序列化为
            # unicode字符串
            # .extract_first() : 取第一个字符串
            item['name'] = dd.xpath('./a/@title').get()
            item['star'] = dd.xpath('.//p[@class="star"]/text()').extract()[0].strip()
            item['time'] = dd.xpath('.//p[@class="releasetime"]/text()').extract()[0]

            yield item
```

3. **定义管道文件（pipelines.py）**

```
# -*- coding: utf-8 -*-
```

```python

# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: https://doc.scrapy.org/en/latest/topics/item-pipeline.html
import pymysql
from . import settings

class MaoyanPipeline(object):
    def process_item(self, item, spider):
        print('*' * 50)
        print(dict(item))
        print('*' * 50)

        return item

# 新建管道类,存入mysql
class MaoyanMysqlPipeline(object):
    # 开启爬虫时执行,只执行一次
    def open_spider(self,spider):
        print('我是open_spider函数')
        # 一般用于开启数据库
        self.db = pymysql.connect(
            settings.MYSQL_HOST,
            settings.MYSQL_USER,
            settings.MYSQL_PWD,
            settings.MYSQL_DB,
            charset = 'utf8'
        )
        self.cursor = self.db.cursor()

    def process_item(self,item,spider):
        ins = 'insert into film(name,star,time) ' \
              'values(%s,%s,%s)'
        L = [
            item['name'].strip(),
            item['star'].strip(),
            item['time'].strip()
        ]
        self.cursor.execute(ins,L)
        # 提交到数据库执行
        self.db.commit()
        return item

    # 爬虫结束时,只执行一次
    def close_spider(self,spider):
        # 一般用于断开数据库连接
        print('我是close_spider函数')
        self.cursor.close()
        self.db.close()
```

5. **全局配置文件（settings.py）**

```
USER_AGENT = 'Mozilla/5.0'
ROBOTSTXT_OBEY = False
DEFAULT_REQUEST_HEADERS = {
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
    'Accept-Language': 'en',
}
ITEM_PIPELINES = {
   'Maoyan.pipelines.MaoyanPipeline': 300,
}
```

6. **创建并运行文件（begin.py）**

```
from scrapy import cmdline
cmdline.execute('scrapy crawl maoyan'.split())
```

知识点汇总

- **节点对象.xpath('')**

```
1、列表,元素为选择器 ['<selector data='A'>]
2、列表.extract() ：序列化列表中所有选择器为Unicode字符串 ['A','B','C']
3、列表.extract_first() 或者 get() :获取列表中第1个序列化的元素(字符串)
```

- **pipelines.py中必须由1个函数叫process_item**

```
def process_item(self,item,spider):
    return item ( * 此处必须返回 item )
```

- **日志变量及日志级别(settings.py)**

```
# 日志相关变量
LOG_LEVEL = ''
LOG_FILE = '文件名.log'

# 日志级别
5 CRITICAL ：严重错误
4 ERROR    ：普通错误
3 WARNING  ：警告
2 INFO     ：一般信息
1 DEBUG    ：调试信息
# 注意：只显示当前级别的日志和比当前级别日志更严重的
```

- **管道文件使用**

```
1、在爬虫文件中为items.py中类做实例化，用爬下来的数据给对象赋值
    from ..items import MaoyanItem
    item = MaoyanItem()
2、管道文件（pipelines.py）
3、开启管道（settings.py）
    ITEM_PIPELINES = { '项目目录名.pipelines.类名':优先级 }
```

数据持久化存储(MySQL)

实现步骤

```
1、在setting.py中定义相关变量
2、pipelines.py中导入settings模块
    def open_spider(self,spider):
        # 爬虫开始执行1次,用于数据库连接
    def close_spider(self,spider):
        # 爬虫结束时执行1次,用于断开数据库连接
3、settings.py中添加此管道
    ITEM_PIPELINES = {'':200}

# 注意 ： process_item() 函数中一定要 return item ***
```

练习

把猫眼电影数据存储到MySQL数据库中

保存为csv、json文件

- 命令格式

```
scrapy crawl maoyan -o maoyan.csv
scrapy crawl maoyan -o maoyan.json
```

盗墓笔记小说抓取案例（三级页面）

- 目标

```
# 抓取目标网站中盗墓笔记1-8中所有章节的所有小说的具体内容，保存到本地文件
1、网址 ： http://www.daomubiji.com/
```

- 准备工作xpath

```
1、一级页面xpath（此处响应做了处理）：//ul[@class="sub-menu"]/li/a/@href
2、二级页面xpath：/html/body/section/div[2]/div/article
  基准xpath ：//article
3、三级页面xpath：response.xpath('//article[@class="article-content"]//p/text()').extract()
```

- 项目实现

1. **创建项目及爬虫文件**

```
创建项目 ：Daomu
创建爬虫 ：daomu  www.daomubiji.com
```

2. 定义要爬取的数据结构（把数据交给管道）

```
import scrapy

class DaomuItem(scrapy.Item):
    # 卷名
    juan_name = scrapy.Field()
    # 章节数
    zh_num = scrapy.Field()
    # 章节名
    zh_name = scrapy.Field()
    # 章节链接
    zh_link = scrapy.Field()
    # 小说内容
    zh_content = scrapy.Field()
```

3. **爬虫文件实现数据抓取**

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

class DaomuSpider(scrapy.Spider):
    name = 'daomu'
    allowed_domains = ['www.daomubiji.com']
    start_urls = ['http://www.daomubiji.com/']

    # 解析一级页面,提取 盗墓笔记1 2 3 ... 链接
    def parse(self, response):
        one_link_list = response.xpath('//ul[@class="sub-menu"]/li/a/@href').extract()
        print(one_link_list)
        # 把链接交给调度器入队列
        for one_link in one_link_list:
            yield scrapy.Request(url=one_link,callback=self.parse_two_link,dont_filter=True)

    # 解析二级页面
    def parse_two_link(self,response):
        # 基准xpath,匹配所有章节对象列表
        article_list = response.xpath('/html/body/section/div[2]/div/article')
        # 依次获取每个章节信息
```

```
        for article in article_list:
            # 创建item对象
            item = DaomuItem()
            info = article.xpath('./a/text()').extract_first().split()
            # info : ['七星鲁王','第一章','血尸']
            item['juan_name'] = info[0]
            item['zh_num'] = info[1]
            item['zh_name'] = info[2]
            item['zh_link'] = article.xpath('./a/@href').extract_first()
            # 把章节链接交给调度器
            yield scrapy.Request(
                url=item['zh_link'],
                # 把item传递到下一个解析函数
                meta={'item':item},
                callback=self.parse_three_link,
                dont_filter=True
            )

    # 解析三级页面
    def parse_three_link(self,response):
        item = response.meta['item']
        # 获取小说内容
        item['zh_content'] = '\n'.join(response.xpath(
          '//article[@class="article-content"]//p/text()'
        ).extract())

        yield item

        # '\n'.join(['第一段','第二段','第三段'])
```

4. **管道文件实现数据处理**

```
# -*- coding: utf-8 -*-

# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: https://doc.scrapy.org/en/latest/topics/item-pipeline.html


class DaomuPipeline(object):
    def process_item(self, item, spider):
        filename = '/home/tarena/aid1902/{}-{}-{}.txt'.format(
            item['juan_name'],
            item['zh_num'],
            item['zh_name']
        )

        f = open(filename,'w')
        f.write(item['zh_content'])
        f.close()
        return item
```

图片管道(360图片抓取案例)

- 目标

```
www.so.com -> 图片 -> 美女
```

- 抓取网络数据包

```
2、F12抓包,抓取到json地址 和 查询参数(QueryString)
      url = 'http://image.so.com/zj?ch=beauty&sn={}&listtype=new&temp=1'.format(str(sn))
      ch: beauty
      sn: 90
      listtype: new
      temp: 1
```

- 项目实现

1. 创建爬虫项目和爬虫文件

```

```

2. 定义要爬取的数据结构

```

```

3. 爬虫文件实现图片链接抓取

```
# -*- coding: utf-8 -*-
import scrapy
import json
from ..items import SoItem

class SoSpider(scrapy.Spider):
    name = 'so'
    allowed_domains = ['image.so.com']

    # 重写Spider类中的start_requests方法
    # 爬虫程序启动时执行此方法,不去找start_urls
    def start_requests(self):
        for page in range(5):
            url = 'http://image.so.com/zj?ch=beauty&sn=
{}&listtype=new&temp=1'.format(str(page*30))
            # 把url地址入队列
            yield scrapy.Request(
                url = url,
                callback = self.parse_img
            )

    def parse_img(self, response):
        html = json.loads(response.text)

        for img in html['list']:
```

```
            item = SoItem()
            # 图片链接
            item['img_link'] = img['qhimg_url']

            yield item
```

4. **管道文件（pipelines.py）**

```
from scrapy.pipelines.images import ImagesPipeline
import scrapy

class SoPipeline(ImagesPipeline):
    # 重写get_media_requests方法
    def get_media_requests(self, item, info):
        yield scrapy.Request(item['img_link'])
```

5. **设置settings.py**

```

```

6. **创建bigin.py运行爬虫**

```

```

今日作业

```
1、把今天内容过一遍
2、Daomu错误调一下(看规律,做条件判断)
3、腾讯招聘尝试改写为scrapy
   response.text ： 获取页面响应内容
```