

# Day06回顾

## 京东爬虫

- 执行JS脚本,把进度条拉到最下面

```
1 1、js脚本
2 browser.execute_script(
3     'window.scrollTo(0,document.body.scrollHeight)'
4 )
5 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
```

- 设置无界面模式 (chromedriver | firefox)

```
1 options = webdriver.ChromeOptions()
2 options.add_argument('--headless')
3
4 browser = webdriver.Chrome(options=options)
5 browser.get(url)
```

## 多线程爬虫

- 使用流程

```
1 # 1、URL队列
2 q.put(url)
3 # 2、线程事件函数
4 while True:
5     if not url_queue.empty():
6         ...get()、请求、解析
7     else:
8         break
9 # 创建并启动线程
10 t_list = []
11 for i in range(5):
12     t = Thread(target=parse_page)
13     t_list.append(t)
14     t.start()
15 # 阻塞等待回收线程
16 for i in t_list:
```

## json模块

### ■ json转python

```
1 变量名 = json.loads(res.text))
```

### ■ python转json (保存为json文件)

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

## scrapy框架

### ■ 五大组件

```
1 引擎 (Engine)
2 爬虫程序 (Spider)
3 调度器 (Scheduler)
4 下载器 (Downloader)
5 管道文件 (Pipeline)
6  # 两个中间件
7  下载器中间件 (Downloader Middlewares)
8  蜘蛛中间件 (Spider Middlewares)
```

### ■ 工作流程

```
1  1、Engine向Spider索要URL, 交给Scheduler入队列
2  2、Scheduler处理后出队列, 通过Downloader Middlewares交给Downloader去下载
3  3、Downloader得到响应后, 通过Spider Middlewares交给Spider
4  4、Spider数据提取:
5      1、数据交给Pipeline处理
6      2、需要跟进URL, 继续交给Scheduler入队列, 依次循环
```

### ■ 常用命令

```
1 # 创建爬虫项目
2 scrapy startproject 项目名
3
4 # 创建爬虫文件
5 cd 项目文件夹
6 scrapy genspider 爬虫名 域名
7
8 # 运行爬虫
9 scrapy crawl 爬虫名
```

#### ■ scrapy项目目录结构

```
1 Baidu
2 |— Baidu          # 项目目录
3 |   |— items.py    # 定义数据结构
4 |   |— middlewares.py # 中间件
5 |   |— pipelines.py # 数据处理
6 |   |— settings.py  # 全局配置
7 |   |— spiders
8 |       |— baidu.py # 爬虫文件
9 |— scrapy.cfg      # 项目基本配置文件
```

#### ■ settings.py全局配置

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

---

# Day07笔记

## scrapy框架

#### ■ 创建爬虫项目步骤

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

- pycharm运行爬虫项目

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

## 小试牛刀

- 目标

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

### 2. 编写爬虫文件baidu.py, xpath提取数据

```
1 # -*- coding: utf-8 -*-
2 import scrapy
3
4 class BaiduSpider(scrapy.Spider):
5     name = 'baidu'
6     allowed_domains = ['www.baidu.com']
7     start_urls = ['http://www.baidu.com/']
8
9     def parse(self, response):
10         result = response.xpath('/html/head/title/text()').extract_first()
11         print('*'*50)
12         print(result)
13         print('*'*50)
14
```

### 3. 全局配置settings.py

```
1 USER_AGENT = 'Mozilla/5.0'
2 ROBOTSTXT_OBEY = False
3 DEFAULT_REQUEST_HEADERS = {
4     'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
5     'Accept-Language': 'en',
6 }
```

### 4. 创建begin.py (和scrapy.cfg同目录)

```
1 from scrapy import cmdline
2
3 cmdline.execute('scrapy crawl baidu'.split())
```

## 5. 启动爬虫

```
1 直接运行 begin.py 文件即可
```

## 思考运行过程

# 猫眼电影案例

## ■ 目标

```
1 URL: 百度搜索 -> 猫眼电影 -> 榜单 -> top100榜
2 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

```
1 # 创建爬虫项目
2 scrapy startproject Maoyan
3 cd Maoyan
4 # 创建爬虫文件
5 scrapy genspider maoyan maoyan.com
```

### 2. 定义要爬取的数据结构 (items.py)

```
1 name = scrapy.Field()
2 star = scrapy.Field()
3 time = scrapy.Field()
```

### 3. 编写爬虫文件 (maoyan.py)

```
1 1、基准xpath, 匹配每个电影信息节点对象列表
2     dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
3 2、for dd in dd_list:
4     电影名称 = dd.xpath('./a/@title')
5     电影主演 = dd.xpath('./p[@class="star"]/text()')
6     上映时间 = dd.xpath('./p[@class="releasetime"]/text()')
```

## 代码实现一

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

```

4
5 class MaoyanSpider(scrapy.Spider):
6     # 爬虫名
7     name = 'maoyan'
8     # 允许爬取的域名
9     allowed_domains = ['maoyan.com']
10    offset = 0
11    # 起始的URL地址
12    start_urls = ['https://maoyan.com/board/4?offset=0']
13
14    def parse(self, response):
15
16        # 基准xpath,匹配每个电影信息节点对象列表
17        dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
18        # dd_list : [<element dd at xxx>,<...>]
19        for dd in dd_list:
20            # 创建item对象
21            item = MaoyanItem()
22            # [<selector xpath='' data='霸王别姬'>]
23            # dd.xpath('')结果为[选择器1,选择器2]
24            # .extract() 把[选择器1,选择器2]所有选择器序列化为unicode字符串
25            # .extract_first() : 取第一个字符串
26            item['name'] = dd.xpath('./a/@title').extract_first().strip()
27            item['star'] = dd.xpath('./p[@class="star"]/text()').extract()[0].strip()
28            item['time'] = dd.xpath('./p[@class="releasetime"]/text()').extract()[0]
29
30            yield item
31
32        # 此方法不推荐,效率低
33        self.offset += 10
34        if self.offset <= 90:
35            url = 'https://maoyan.com/board/4?offset={}'.format(str(self.offset))
36
37            yield scrapy.Request(
38                url=url,
39                callback=self.parse
40            )

```

## 代码实现二

```

1 # -*- coding: utf-8 -*-
2 import scrapy
3 from ..items import MaoyanItem
4
5 class MaoyanSpider(scrapy.Spider):
6     # 爬虫名
7     name = 'maoyan2'
8     # 允许爬取的域名
9     allowed_domains = ['maoyan.com']
10    # 起始的URL地址
11    start_urls = ['https://maoyan.com/board/4?offset=0']
12
13    def parse(self, response):
14        for offset in range(0,91,10):
15            url = 'https://maoyan.com/board/4?offset={}'.format(str(offset))
16            # 把地址交给调度器入队列

```

```

17         yield scrapy.Request(
18             url=url,
19             callback=self.parse_html
20         )
21
22     def parse_html(self, response):
23         # 基准xpath,匹配每个电影信息节点对象列表
24         dd_list = response.xpath(
25             '//dl[@class="board-wrapper"]/dd')
26         # dd_list : [<element dd at xxx>,<...>]
27
28         for dd in dd_list:
29             # 创建item对象
30             item = MaoyanItem()
31             # [<selector xpath='' data='霸王别姬'>]
32             # dd.xpath('')结果为[选择器1,选择器2]
33             # .extract() 把[选择器1,选择器2]所有选择器序列化为
34             # unicode字符串
35             # .extract_first() : 取第一个字符串
36             item['name'] = dd.xpath('./a/@title').extract_first().strip()
37             item['star'] = dd.xpath('./p[@class="star"]/text()').extract()[0].strip()
38             item['time'] = dd.xpath('./p[@class="releasetime"]/text()').extract()[0]
39
40         yield item

```

### 代码实现三

```

1  # 重写start_requests()方法, 直接把多个地址都交给调度器去处理
2  # -*- coding: utf-8 -*-
3  import scrapy
4  from ..items import MaoyanItem
5
6  class MaoyanSpider(scrapy.Spider):
7      # 爬虫名
8      name = 'maoyan_requests'
9      # 允许爬取的域名
10     allowed_domains = ['maoyan.com']
11
12     def start_requests(self):
13         for offset in range(0,91,10):
14             url = 'https://maoyan.com/board/4?offset={}'.format(str(offset))
15             # 把地址交给调度器入队列
16             yield scrapy.Request(url=url,callback=self.parse_html )
17
18     def parse_html(self, response):
19         # 基准xpath,匹配每个电影信息节点对象列表
20         dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
21         # dd_list : [<element dd at xxx>,<...>]
22
23         for dd in dd_list:
24             # 创建item对象
25             item = MaoyanItem()
26             # [<selector xpath='' data='霸王别姬'>]
27             # dd.xpath('')结果为[选择器1,选择器2]
28             # .extract() 把[选择器1,选择器2]所有选择器序列化为
29             # unicode字符串

```

```

30         # .extract_first() : 取第一个字符串
31         item['name'] = dd.xpath('./a/@title').get()
32         item['star'] = dd.xpath('./p[@class="star"]/text()').extract()[0].strip()
33         item['time'] = dd.xpath('./p[@class="releasetime"]/text()').extract()[0]
34
35         yield item

```

### 3. 定义管道文件 (pipelines.py)

```

1  # -*- coding: utf-8 -*-
2
3  # Define your item pipelines here
4  #
5  # Don't forget to add your pipeline to the ITEM_PIPELINES setting
6  # See: https://doc.scrapy.org/en/latest/topics/item-pipeline.html
7  import pymysql
8  from . import settings
9
10 class MaoyanPipeline(object):
11     def process_item(self, item, spider):
12         print('*' * 50)
13         print(dict(item))
14         print('*' * 50)
15
16         return item
17
18 # 新建管道类,存入mysql
19 class MaoyanMysqlPipeline(object):
20     # 开启爬虫时执行,只执行一次
21     def open_spider(self, spider):
22         print('我是open_spider函数')
23         # 一般用于开启数据库
24         self.db = pymysql.connect(
25             settings.MYSQL_HOST,
26             settings.MYSQL_USER,
27             settings.MYSQL_PWD,
28             settings.MYSQL_DB,
29             charset = 'utf8'
30         )
31         self.cursor = self.db.cursor()
32
33     def process_item(self, item, spider):
34         ins = 'insert into film(name,star,time) ' \
35             'values(%s,%s,%s)'
36         L = [
37             item['name'].strip(),
38             item['star'].strip(),
39             item['time'].strip()
40         ]
41         self.cursor.execute(ins,L)
42         # 提交到数据库执行
43         self.db.commit()
44         return item
45
46 # 爬虫结束时,只执行一次
47 def close_spider(self, spider):

```



```

48         # 一般用于断开数据库连接
49         print('我是close_spider函数')
50         self.cursor.close()
51         self.db.close()

```

## 5. 全局配置文件 (settings.py)

```

1  USER_AGENT = 'Mozilla/5.0'
2  ROBOTSTXT_OBEY = False
3  DEFAULT_REQUEST_HEADERS = {
4      'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
5      'Accept-Language': 'en',
6  }
7  ITEM_PIPELINES = {
8      'Maoyan.pipelines.MaoyanPipeline': 300,
9  }

```

## 6. 创建并运行文件 (begin.py)

```

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

```

# 知识点汇总

### ■ 节点对象.xpath("")

```

1  1、列表,元素为选择器 ['<selector data='A'>]
2  2、列表.extract() : 序列化列表中所有选择器为Unicode字符串 ['A','B','C']
3  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

```

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

```

### ■ 日志变量及日志级别(settings.py)

```

1  # 日志相关变量
2  LOG_LEVEL = ''
3  LOG_FILE = '文件名.log'
4
5  # 日志级别
6  5 CRITICAL : 严重错误
7  4 ERROR    : 普通错误
8  3 WARNING  : 警告
9  2 INFO     : 一般信息
10 1 DEBUG    : 调试信息
11 # 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 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
2     from ..items import MaoyanItem
3     item = MaoyanItem()
4 2、管道文件 (pipelines.py)
5 3、开启管道 (settings.py)
6     ITEM_PIPELINES = { '项目目录名.pipelines.类名':优先级 }
```

## 数据持久化存储(MongoDB和MySQL)

### 实现步骤

```
1 1、在setting.py中定义相关变量
2 2、pipelines.py中导入settings模块
3     def open_spider(self,spider):
4         # 爬虫开始执行1次,用于数据库连接
5     def close_spider(self,spider):
6         # 爬虫结束时执行1次,用于断开数据库连接
7 3、settings.py中添加此管道
8     ITEM_PIPELINES = {'':200}
9
10 # 注意：process_item() 函数中一定要 return item ***
```

## 保存为csv、json文件

- 命令格式

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

## 盗墓笔记小说抓取案例（三级页面）

- 目标

```
1 # 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
2 1、网址：http://www.daomubiji.com/
```

- 准备工作xpath

```
1 1、一级页面xpath (此处响应做了处理) : //ul[@class="sub-menu"]/li/a/@href
2 2、二级页面xpath: /html/body/section/div[2]/div/article
3 基准xpath : //article
4 3、三级页面xpath: response.xpath('//article[@class="article-content"]//p/text()).extract()
```

## ■ 项目实施

### 1. 创建项目及爬虫文件

```
1 创建项目 : Daomu
2 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)

```
1 import scrapy
2
3 class DaomuItem(scrapy.Item):
4     # 卷名
5     juan_name = scrapy.Field()
6     # 章节数
7     zh_num = scrapy.Field()
8     # 章节名
9     zh_name = scrapy.Field()
10    # 章节链接
11    zh_link = scrapy.Field()
12    # 小说内容
13    zh_content = scrapy.Field()
```

### 3. 爬虫文件实现数据抓取

```
1 # -*- coding: utf-8 -*-
2 import scrapy
3 from ..items import DaomuItem
4
5 class DaomuSpider(scrapy.Spider):
6     name = 'daomu'
7     allowed_domains = ['www.daomubiji.com']
8     start_urls = ['http://www.daomubiji.com/']
9
10    # 解析一级页面,提取 盗墓笔记1 2 3 ... 链接
11    def parse(self, response):
12        one_link_list = response.xpath('//ul[@class="sub-menu"]/li/a/@href').extract()
13        print(one_link_list)
14        # 把链接交给调度器入队列
15        for one_link in one_link_list:
16            yield scrapy.Request(url=one_link, callback=self.parse_two_link, dont_filter=True)
17
18    # 解析二级页面
19    def parse_two_link(self, response):
20        # 基准xpath,匹配所有章节对象列表
21        article_list = response.xpath('/html/body/section/div[2]/div/article')
22        # 依次获取每个章节信息
23        for article in article_list:
24            # 创建item对象
```

```

25         item = DaomuItem()
26         info = article.xpath('./a/text()').extract_first().split()
27         # info : ['七星鲁王', '第一章', '血尸']
28         item['juan_name'] = info[0]
29         item['zh_num'] = info[1]
30         item['zh_name'] = info[2]
31         item['zh_link'] = article.xpath('./a/@href').extract_first()
32         # 把章节链接交给调度器
33         yield scrapy.Request(
34             url=item['zh_link'],
35             # 把item传递到下一个解析函数
36             meta={'item':item},
37             callback=self.parse_three_link,
38             dont_filter=True
39         )
40
41     # 解析三级页面
42     def parse_three_link(self, response):
43         item = response.meta['item']
44         # 获取小说内容
45         item['zh_content'] = '\n'.join(response.xpath(
46             '//article[@class="article-content"]//p/text()')
47             .extract())
48
49         yield item
50
51         # '\n'.join(['第一段', '第二段', '第三段'])

```

#### 4. 管道文件实现数据处理

```

1  # -*- coding: utf-8 -*-
2
3  # Define your item pipelines here
4  #
5  # Don't forget to add your pipeline to the ITEM_PIPELINES setting
6  # See: https://doc.scrapy.org/en/latest/topics/item-pipeline.html
7
8
9  class DaomuPipeline(object):
10     def process_item(self, item, spider):
11         filename = '/home/tarena/aid1902/{-}{-}{-}.txt'.format(
12             item['juan_name'],
13             item['zh_num'],
14             item['zh_name']
15         )
16
17         f = open(filename, 'w')
18         f.write(item['zh_content'])
19         f.close()
20         return item

```

## 图片管道(360图片抓取案例)

## ■ 目标

```
1 | www.so.com -> 图片 -> 美女
```

## ■ 抓取网络数据包

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

## ■ 项目实施

### 1. 创建爬虫项目和爬虫文件

```
1 |
```

### 2. 定义要爬取的数据结构

```
1 |
```

### 3. 爬虫文件实现图片链接抓取

```
1 | # -*- coding: utf-8 -*-
2 | import scrapy
3 | import json
4 | from ..items import SoItem
5 |
6 | class SoSpider(scrapy.Spider):
7 |     name = 'so'
8 |     allowed_domains = ['image.so.com']
9 |
10 |    # 重写Spider类中的start_requests方法
11 |    # 爬虫程序启动时执行此方法,不去找start_urls
12 |    def start_requests(self):
13 |        for page in range(5):
14 |            url = 'http://image.so.com/zj?ch=beauty&sn=
15 |            {}&listtype=new&temp=1'.format(str(page*30))
16 |            # 把url地址入队列
17 |            yield scrapy.Request(
18 |                url = url,
19 |                callback = self.parse_img
20 |            )
21 |
22 |    def parse_img(self, response):
23 |        html = json.loads(response.text)
24 |
25 |        for img in html['list']:
26 |            item = SoItem()
27 |            # 图片链接
28 |            item['img_link'] = img['qimg_url']
```

28

29       yield item

#### 4. 管道文件 (pipelines.py)

```
1  from scrapy.pipelines.images import ImagesPipeline
2  import scrapy
3
4  class SoPipeline(ImagesPipeline):
5      # 重写get_media_requests方法
6      def get_media_requests(self, item, info):
7          yield scrapy.Request(item['img_link'])
```

#### 5. 设置settings.py

1

#### 6. 创建begin.py运行爬虫

1

## 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
- 4     response.text : 获取页面响应内容