

DAY10

Day09回顾

*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}

```

创建项目流程

```

1 1、 scrapy startproject Tencent
2 2、 cd Tencent
3 3、 scrapy genspider tencent tencent.com
4 4、 items.py(定义爬取数据结构)
5     import scrapy
6     class TencentItem(scrapy.Item):
7         job_name = scrapy.Field()
8 5、 tencent.py (写爬虫文件)
9     import scrapy
10    class TencentSpider(scrapy.Spider):
11        name = 'tencent'
12        allowed_domains = ['tencent.com']
13        start_urls = ['http://tencent.com/']
14        def parse(self, response):
15            xxx
16            yield item
17 6、 pipelines.py(数据处理)
18    class TencentPipeline(object):
19        def process_item(self, item, spider):
20            return item
21 7、 settings.py(全局配置)
22    LOG_LEVEL = ''
23    LOG_FILE = ''
24    FEED_EXPORT_ENCODING = ''
25 8、 终端: scrapy crawl tencent

```

响应对象属性及方法

```
1 # 属性
2 1、response.text : 获取响应内容 - 字符串
3 2、response.body : 获取bytes数据类型
4 3、response.xpath('')
5
6 # response.xpath('')调用方法
7 1、结果 : 列表,元素为选择器对象
8   # <selector xpath='//article' data=''>
9 2、.extract() : 提取文本内容,将列表中所有元素序列化为Unicode字符串
10 3、.extract_first() : 提取列表中第1个文本内容
11 4、.get() : 提取列表中第1个文本内容
```

爬虫项目启动方式

■ 方式一

```
1 从爬虫文件(spider)的start_urls变量中遍历URL地址,把下载器返回的响应对象(response)交给爬虫文件的
  parse()函数处理
2 # start_urls = ['http://www.baidu.com/']
```

■ 方式二

```
1 重写start_requests()方法,从此方法中获取URL,交给指定的callback解析函数处理
2
3 1、去掉start_urls变量
4 2、def start_requests(self):
5     # 生成要爬取的URL地址,利用scrapy.Request()方法交给调度器 **
```

日志级别

```
1 DEBUG < INFO < WARNING < ERROR < CRITICAL
```

数据持久化存储(MySQL、MongoDB)

```

1 1、在setting.py中定义相关变量
2 2、pipelines.py中新建管道类，并导入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 process_item(self,item,spider):
6         # 用于处理抓取的item数据
7     def close_spider(self,spider):
8         # 爬虫结束时执行1次,用于断开数据库连接
9 3、settings.py中添加此管道
10     ITEM_PIPELINES = {'':200}
11
12 # 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
3 # settings.py FEED_EXPORT_ENCODING = 'utf-8'

```

settings.py常用变量

```

1 # 1、设置日志级别
2 LOG_LEVEL = ''
3 # 2、保存到日志文件(不在终端输出)
4 LOG_FILE = ''
5 # 3、设置数据导出编码(主要针对于json文件)
6 FEED_EXPORT_ENCODING = ''
7 # 4、非结构化数据存储路径
8 IMAGES_STORE = '路径'
9 # 5、设置User-Agent
10 USER_AGENT = ''
11 # 6、设置最大并发数(默认为16)
12 CONCURRENT_REQUESTS = 32
13 # 7、下载延迟时间(每隔多长时间请求一个网页)
14 # DOWNLOAD_DELAY 会影响 CONCURRENT_REQUESTS，不能使并发显现
15 # 有CONCURRENT_REQUESTS，没有DOWNLOAD_DELAY：服务器会在同一时间收到大量的请求
16 # 有CONCURRENT_REQUESTS，有DOWNLOAD_DELAY 时，服务器不会在同一时间收到大量的请求
17 DOWNLOAD_DELAY = 3
18 # 8、请求头
19 DEFAULT_REQUEST_HEADERS = {}
20 # 9、添加项目管道
21 ITEM_PIPELINES = {}
22 # 10、添加下载器中间件
23 DOWNLOADER_MIDDLEWARES = {}

```

scrapy.Request() 参数

- 1 | 1、url
- 2 | 2、callback
- 3 | 3、meta : 传递数据, 定义代理

Day10笔记

作业讲解 - 腾讯招聘

■ 1、创建项目+爬虫文件

```
1 scrapy startproject Tencent
2 cd Tencent
3 scrapy genspider tencent hr.tencent.com
4
5 # 一级页面(postId):
6 https://careers.tencent.com/tencentcareer/api/post/Query?
  timestamp=1566266592644&countryId=&cityId=&bgIds=&productId=&categoryId=&parentId=&attr
  Id=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn
7
8 # 二级页面
9 https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1566266695175&postId=
  {}&language=zh-cn
```

■ 2、定义爬取的数据结构

```
1 # 名称+类别+职责+要求+地址+时间
2 job_name = scrapy.Field()
3 job_type = scrapy.Field()
4 job_duty = scrapy.Field()
5 job_require = scrapy.Field()
6 job_address = scrapy.Field()
7 job_time = scrapy.Field()
```

■ 3、爬虫文件

```
1 |
```

■ 4、管道文件

```

1 create database tencentdb charset utf8;
2 use tencentdb;
3 create table tencenttab(
4 job_name varchar(500),
5 job_type varchar(100),
6 job_duty varchar(1000),
7 job_require varchar(1000),
8 job_address varchar(100),
9 job_time varchar(100)
10 )charset=utf8;

```

管道文件pipelines实现

```

1 class TencentPipeline(object):
2     def process_item(self, item, spider):
3         print(dict(item))
4         return item
5
6 import pymysql
7
8 class TencentMysqlPipeline(object):
9     def open_spider(self, spider):
10        self.db = pymysql.connect(
11            'localhost', 'root', '123456', 'tencentdb', charset='utf8'
12        )
13        self.cursor = self.db.cursor()
14
15    def process_item(self, item, spider):
16        ins='insert into tencenttab values(%s,%s,%s,%s,%s,%s)'
17        L = [
18            item['job_name'],
19            item['job_type'],
20            item['job_duty'],
21            item['job_require'],
22            item['job_address'],
23            item['job_time']
24        ]
25        self.cursor.execute(ins,L)
26        self.db.commit()
27
28        return item
29
30    def close_spider(self, spider):
31        self.cursor.close()
32        self.db.close()

```

■ 5、 settings.py

```

1 # 定义常用变量，添加管道即可

```

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/zjl?ch=beauty&sn={}&listtype=new&temp=1'.format(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 | scrapy startproject So
2 | cd So
3 | scrapy genspider so image.so.com
```

2、定义要爬取的数据结构(items.py)

```
1 | img_link = scrapy.Field()
2 | img_title = scrapy.Field()
```

3、爬虫文件实现图片链接+名字抓取

```
1 |
```

4、管道文件 (pipelines.py)

```
1 |
```

5、设置settings.py

```
1 | IMAGES_STORE = '/home/tarena/images/'
```

6、创建run.py运行爬虫

字符串方法总结

```
1 | 1、strip()
2 | 2、split()
3 | 3、replace(' ','')
4 | 4、''.join()
5 | 5、字符串切片(正向切,反向切) : S[-10:]
```

scrapy shell的使用

■ 基本使用

```
1 # scrapy shell URL地址
2 *1、 request.url      : 请求URL地址
3 *2、 request.headers  : 请求头(字典)
4 *3、 request.meta     : item数据传递, 定义代理(字典)
5
6 4、 response.text     : 字符串
7 5、 response.body     : bytes
8 6、 response.xpath('')
```

■ scrapy.Request()参数

```
1 1、 url
2 2、 callback
3 3、 headers
4 4、 meta : 传递数据, 定义代理
5 5、 dont_filter : 是否忽略域组限制
6 默认False, 检查allowed_domains['']
```

设置中间件(随机User-Agent)

少量User-Agent切换

■ 方法一

```
1 # settings.py
2 USER_AGENT = ''
3 DEFAULT_REQUEST_HEADERS = {}
```

■ 方法二

```
1 # spider
2 yield scrapy.Request(url, callback=函数名, headers={})
```

大量User-Agent切换 (中间件)

■ middlewares.py设置中间件


```

1 1、获取User-Agent
2   # 方法1 : 新建useragents.py,存放大量User-Agent, random模块随机切换
3   # 方法2 : 安装fake_useragent模块(sudo pip3 install fake_useragent)
4       from fake_useragent import UserAgent
5       ua_obj = UserAgent()
6       ua = ua_obj.random
7 2、middlewares.py新建中间件类
8   class RandomUseragentMiddleware(object):
9       def process_request(self, request, spider):
10           ua = UserAgent()
11           request.headers['User-Agent'] = ua.random
12 3、settings.py添加此下载器中间件
13   DOWNLOADER_MIDDLEWARES = {'': 优先级}

```

设置中间件(随机代理)

```

1 class RandomProxyDownloaderMiddleware(object):
2     def process_request(self, request, spider):
3         request.meta['proxy'] = xxx
4
5     def process_exception(self, request, exception, spider):
6         return request

```

Fiddler抓包工具

■ 配置Fiddler

```

1 # 添加证书信任
2 1、Tools - Options - HTTPS
3   勾选 Decrypt Https Traffic 后弹出窗口, 一路确认
4 # 设置只抓取浏览器的数据包
5 2、...from browsers only
6 # 设置监听端口 (默认为8888)
7 3、Tools - Options - Connections
8 # 配置完成后重启Fiddler (重要)
9 4、关闭Fiddler,再打开Fiddler

```

■ 配置浏览器代理

```

1 1、安装Proxy SwitchyOmega插件
2 2、浏览器右上角: SwitchyOmega->选项->新建情景模式->AID1904(名字)->创建
3   输入 : HTTP:// 127.0.0.1 8888
4   点击 : 应用选项
5 3、点击右上角SwitchyOmega可切换代理

```

■ Fiddler常用菜单

```
1 1、Inspector：查看数据包详细内容
2  整体分为请求和响应两部分
3 2、常用菜单
4  Headers：请求头信息
5  WebForms
6    # 1. POST请求Form表单数据：<body>
7    # 2. GET请求查询参数：<QueryString>
8  Raw
9  将整个请求显示为纯文本
```