Day07回顾

selenium+phantomjs/chrome/firefox

- **设置无界面模式（chromedriver | firefox）**

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

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

- **browser执行JS脚本**

```
browser.execute_script(
'window.scrollTo(0,document.body.scrollHeight)'
)
time.sleep(2)
```

- **selenium常用操作**

```
【1】键盘操作
    from selenium.webdriver.common.keys import Keys
    node.send_keys(Keys.SPACE)
    node.send_keys(Keys.CONTROL, 'a')
    node.send_keys(Keys.CONTROL, 'c')
    node.send_keys(Keys.CONTROL, 'v')
    node.send_keys(Keys.ENTER)

【2】鼠标操作
    from selenium.webdriver import ActionChains
    mouse_action = ActionChains(browser)
    mouse_action.move_to_element(node)
    mouse_action.perform()

【3】切换句柄
    all_handles = browser.window_handles
    time.sleep(1)
    browser.switch_to.window(all_handles[1])

【4】iframe子框架
    browser.switch_to.frame(iframe_element)
```

scrapy框架

- **五大组件**

```
【1】引擎（Engine）----------整个框架核心
【2】爬虫程序（Spider）------维护请求队列
【3】调度器（Scheduler）-----获取响应对象
【4】下载器（Downloader）----数据解析提取
【5】管道文件（Pipeline）-----数据入库处理


【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：
    4.1）数据交给Pipeline处理
    4.2）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 ： scrapy startproject 项目名
【2】创建爬虫文件
    2.1）cd 项目文件夹
    2.2）scrapy genspider 爬虫名 域名
【3】运行爬虫
    scrapy crawl 爬虫名
```

- **scrapy项目目录结构**

```
Baidu                    # 项目文件夹
├── 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
```

Day08笔记

scrapy框架

- **全局配置文件settings.py详解**

```
【1】定义User-Agent
   USER_AGENT = 'Mozilla/5.0'

【2】是否遵循robots协议，一般设置为False
   ROBOTSTXT_OBEY = False

【3】最大并发量，默认为16
   CONCURRENT_REQUESTS = 32

【4】下载延迟时间
   DOWNLOAD_DELAY = 1

【5】请求头，此处也可以添加User-Agent
   DEFAULT_REQUEST_HEADERS={}
```

- **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
   8.1) 终端: scrapy crawl 爬虫名
   8.2) pycharm运行
       a> 创建run.py(和scrapy.cfg文件同目录)
         from scrapy import cmdline
             cmdline.execute('scrapy crawl maoyan'.split())
       b> 直接运行 run.py 即可
```

还记得我们抓取的 百度一下,你就知道 吗

- **步骤跟踪**

```
【1】创建项目 'Baidu' 和爬虫文件 'baidu'
   1.1) scrapy startproject Baidu
   1.2) cd Baidu
   1.3) scrapy genspider baidu www.baidu.com

【2】打开爬虫文件: baidu.py
   import scrapy
   class BaiduSpider(scrapy.Spider):
       name = 'baidu'
       allowed_domains = ['www.baidu.com']
       start_urls = ['http://www.baidu.com/']
```

```
        def parse(self,response):
            r_list = response.xpath('/html/head/title/text()').extract()[0]

【3】全局配置文件：settings.py
    ROBOTSTXT_OBEY = False
    DEFAULT_REQUEST_HEADERS = {'User-Agent':'Mozilla/5.0'}

【4】创建文件(和项目目录同路径)：run.py
    from scrapy import cmdline
    cmdline.execute('scrapy crawl baidu'.split())

【5】运行 run.py 启动爬虫
```

瓜子二手车直卖网

- **目标**

```
【1】抓取瓜子二手车官网二手车收据（我要买车）

【2】URL地址：https://www.guazi.com/langfang/buy/o{}/#bread
    URL规律：o1  o2  o3  o4  o5  ... ...

【3】所抓数据
    3.1) 汽车链接
    3.2) 汽车名称
    3.3) 汽车价格
```

实现步骤

- **1-创建项目和爬虫文件**

```
scrapy startproject Car
cd Car
scrapy genspider car www.guazi.com
```

- **2-定义要爬取的数据结构**

```
"""items.py"""
import scrapy

class CarItem(scrapy.Item):
    # 链接、名称、价格
    url = scrapy.Field()
    name = scrapy.Field()
    price = scrapy.Field()
```

- **3-编写爬虫文件（代码实现1）**

```
"""
此方法其实还是一页一页抓取，效率并没有提升，和单线程一样

xpath表达式如下：
【1】基准xpath,匹配所有汽车节点对象列表
    li_list = response.xpath('//ul[@class="carlist clearfix js-top"]/li')

【2】遍历后每辆车信息的xpath表达式
    汽车链接：'./a[1]/@href'
    汽车名称：'.//h2[@class="t"]/text()'
    汽车价格：'.//div[@class="t-price"]/p/text()'
"""
# -*- coding: utf-8 -*-
import scrapy
from ..items import CarItem

class CarSpider(scrapy.Spider):
    name = 'car'
    allowed_domains = ['www.guazi.com']
    i = 1
    start_urls = ['https://www.guazi.com/langfang/buy/o1/']

    def parse(self, response):
        # 1.基准xpath,匹配所有汽车节点对象列表
        li_list = response.xpath('//ul[@class="carlist clearfix js-top"]/li')
        # 创建item对象,给items.py中定义的数据结构赋值
        item = CarItem()
        for li in li_list:
            item['url'] = 'https://www.guazi.com/' + li.xpath('./a[1]/@href').get()
            item['name'] = li.xpath('.//h2[@class="t"]/text()').get()
            item['price'] = li.xpath('.//div[@class="t-price"]/p/text()').get()

            yield item

        # 生成下一页的链接,继续交给调度器入队列
        if self.i < 5:
            self.i += 1
            url = 'https://www.guazi.com/langfang/buy/o{}/'.format(self.i)
            # scrapy.Request()是将请求交给调度器入队列的方法
            yield scrapy.Request(url=url,callback=self.parse)
```

- **3-编写爬虫文件（代码实现2）**

```
"""
重写start_requests()方法，效率极高
"""
# -*- coding: utf-8 -*-
import scrapy
from ..items import CarItem

class CarSpider(scrapy.Spider):
    name = 'car2'
    allowed_domains = ['www.guazi.com']
    # 1、去掉 start_urls
    # 2、重写start_requests()方法
    def start_requests(self):
```

```
        """生成所有待爬取的URL地址,统一交给调度器入队列"""
        for i in range(1,5):
            url = 'https://www.guazi.com/langfang/buy/o{}/'.format(i)
            yield scrapy.Request(url=url,callback=self.parse)

    def parse(self, response):
        # 1.基准xpath,匹配所有汽车节点对象列表
        li_list = response.xpath('//ul[@class="carlist clearfix js-top"]/li')
        # 创建item对象,给items.py中定义的数据结构赋值
        item = CarItem()
        for li in li_list:
            item['url'] = 'https://www.guazi.com/' + li.xpath('./a[1]/@href').get()
            item['name'] = li.xpath('.//h2[@class="t"]/text()').get()
            item['price'] = li.xpath('.//div[@class="t-price"]/p/text()').get()

            yield item
```

- **4-管道文件处理数据**

```
"""
pipelines.py处理数据
1、mysql数据库建库建表
create database guazidb charset utf8;
use guazidb;
create table guaziset(
name varchar(200),
price varchar(100),
url varchar(500)
)charset=utf8;
"""
# -*- coding: utf-8 -*-

# 管道1 - 从终端打印输出
class CarPipeline(object):
    def process_item(self, item, spider):
        print(item['name'],item['price'],item['url'])
        return item

# 管道2 - 存入MySQL数据库管道
import pymysql
from .settings import *

class CarMysqlPipeline(object):
    def open_spider(self,spider):
        """爬虫项目启动时只执行1次,一般用于数据库连接"""
        self.db = pymysql.connect(MYSQL_HOST,MYSQL_USER,MYSQL_PWD,MYSQL_DB,charset=CHARSET)
        self.cursor = self.db.cursor()

    def process_item(self,item,spider):
        """处理从爬虫文件传过来的item数据"""
        ins = 'insert into guazitab values(%s,%s,%s)'
        car_li = [item['name'],item['price'],item['url']]
        self.cursor.execute(ins,car_li)
        self.db.commit()

        return item
```

```
    def close_spider(self,spider):
        """爬虫程序结束时只执行1次,一般用于数据库断开"""
        self.cursor.close()
        self.db.close()


# 管道3 - 存入MongoDB管道
import pymongo

class CarMongoPipeline(object):
    def open_spider(self,spider):
        self.conn = pymongo.MongoClient(MONGO_HOST,MONGO_PORT)
        self.db = self.conn[MONGO_DB]
        self.myset = self.db[MONGO_SET]

    def process_item(self,item,spider):
        car_dict = {
            'name' : item['name'],
            'price': item['price'],
            'url'  : item['url']
        }
        self.myset.insert_one(car_dict)
```

- **5-全局配置文件（settings.py）**

```
【1】ROBOTSTXT_OBEY = False
【2】DOWNLOAD_DELAY = 2
【3】COOKIES_ENABLED = False
【4】DEFAULT_REQUEST_HEADERS = {
   "Cookie": "此处填写抓包抓取到的Cookie",
   "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.122 Safari/537.36",
  }

【5】ITEM_PIPELINES = {
    'Car.pipelines.CarPipeline': 300,
    #'Car.pipelines.CarMysqlPipeline': 400,
    #'Car.pipelines.CarMongoPipeline': 500,
  }

【6】定义MySQL相关变量
MYSQL_HOST = 'localhost'
MYSQL_USER = 'root'
MYSQL_PWD = '123456'
MYSQL_DB = 'guazidb'
CHARSET = 'utf8'

【7】定义MongoDB相关变量
MONGO_HOST = 'localhost'
MONGO_PORT = 27017
MONGO_DB = 'guazidb'
MONGO_SET = 'guaziset'
```

- **6-运行爬虫（run.py）**

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

数据持久化(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 process_item(self,item,spider):
        """具体处理数据"""
        return item

    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 ,当前管道的process_item()的返回值会作为下一个管道 process_item()的参数
```

知识点汇总

- **节点对象.xpath('')**

```
【1】列表,元素为选择器
    [
        <selector xpath='xxx' data='A'>,
        <selector xpath='xxx' data='B'>
    ]

【2】列表.extract() : 序列化列表中所有选择器为Unicode字符串 ['A','B']

【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) 'A'
```

- **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 }
```

保存为csv、json文件

- **命令格式**

```
"""run.py"""
【1】存入csv文件
    scrapy crawl car -o car.csv

【2】存入json文件
    scrapy crawl car -o car.json

【3】注意：settings.py中设置导出编码 - 主要针对json文件
    FEED_EXPORT_ENCODING = 'utf-8'
```

- **课堂练习**

```
【熟悉整个流程】 ：将猫眼电影案例数据抓取，存入MySQL数据库
```

新浪新闻全站抓取

- **目标**

```
【1】抓取新浪新闻下的所有分类的所有新闻，保存到本地
【2】URL：新浪官网 - 更多 - 导航
    http://news.sina.com.cn/guide/
【3】要求
    将信息保存到scrapy项目目录的 data 文件夹中,并按照分类名称创建子文件夹
```

实现步骤

- **1-创建项目和爬虫文件**

```
scrapy startproject Sina
cd Sina
scrapy genspider sina news.sina.com.cn
```

- **2-定义要抓取的数据结构**

```
"""items.py"""
import scrapy

class SinaItem(scrapy.Item):
    # 大类标题、url 例：新闻、体育、娱乐、财经... ...
    parent_name = scrapy.Field()
    parent_url = scrapy.Field()

    # 小类标题、url 例：体育分类下的 NBA CBA ... ...
    son_name = scrapy.Field()
    son_url = scrapy.Field()

    # 小类目录存储路径
    son_filename = scrapy.Field()

    # 小类下的文章链接、标题、内容
    article_url = scrapy.Field()
    article_head = scrapy.Field()
    article_content = scrapy.Field()
```

- **3-爬虫文件进行数据解析提取**

```
"""sina.py"""
# -*- coding: utf-8 -*-
import scrapy
import os
from ..items import SinaItem

class SinaSpider(scrapy.Spider):
    name = 'sina'
    allowed_domains = ['sina.com.cn']
    # 起始URL地址为导航页的地址
    start_urls = ['http://news.sina.com.cn/guide/']

    def parse(self, response):
```

```
        # 用来存放下一次交给调度器的所有请求,即所有小分类的请求
        son_items = []
        # 基准xpath：获取所有大分类的对象列表
        div_list = response.xpath('//div[@id="tab01"]/div')
        for div in div_list:
            # 1个大分类名称
            parent_name = div.xpath('./h3/a/text()').get()
            parent_url = div.xpath('./h3/a/@href').get()
            if parent_name and parent_url:
                # 1个大分类下面的所有小分类
                li_list = div.xpath('./ul/li')
                for li in li_list:
                    # 继续交给调度器的item对象,确保每个是独立的
                    item = SinaItem()
                    item['son_name'] = li.xpath('./a/text()').get()
                    item['son_url'] = li.xpath('./a/@href').get()
                    item['parent_name'] = parent_name
                    item['parent_url'] = parent_url
                    # 创建对应的文件夹
                    directory =
'./data/{}/{}/'.format(item['parent_name'],item['son_name'])
                    item['son_filename'] = directory
                    if not os.path.exists(directory):
                        os.makedirs(directory)
                    son_items.append(item)

        # 大循环结束,items中存放了所有的每个小类请求的 item 对象
        # 发送每个请求到调度器,得到response连同meta数据一起回调函数parse_son_url方法
        for item in son_items:
            yield scrapy.Request(url=item['son_url'],meta=
{'meta_1':item},callback=self.parse_son_url)

    def parse_son_url(self,response):
        """解析小分类函数"""
        meta1_item = response.meta['meta_1']
        # 存放所有新闻页链接的item请求列表
        meta1_items = []
        parent_url = meta1_item['parent_url']
        # 通过观察规律,小分类中的所有新闻连接均以最外层大链接开头
        news_link_list = response.xpath('//a/@href').extract()
        for news_link in news_link_list:
            if news_link.startswith(parent_url) and news_link.endswith('.shtml'):
                # 此item对象是需要继续交给调度器入队列的请求
                item = SinaItem()
                # 把具体的新闻链接发送请求,保存到对应的文件夹下
                item['son_name'] = meta1_item['son_name']
                item['son_url'] = meta1_item['son_url']
                item['parent_name'] = meta1_item['parent_name']
                item['parent_url'] = meta1_item['parent_url']
                item['son_filename'] = meta1_item['son_filename']
                item['article_url'] = news_link
                meta1_items.append(item)

        # 发送每个小类下的新闻链接请求,得到response后和meta数据交给get_content()函数处理
        for item in meta1_items:
            yield scrapy.Request(url=item['article_url'], meta={'meta_2': item},
callback=self.get_content)
```

```
68
69      def get_content(self,response):
70          """获取每个新闻的具体内容"""
71          item = response.meta['meta_2']
72          item['article_head'] = response.xpath('//h1[@class="main-title"]/text() | //span[@class="location"]/h1/text()').get()
73          item['article_content'] = '\n'.join(response.xpath('//div[@class="article"]/p/text() | //div[@class="article clearfix"]/p/text() | //div[@id="artibody"]/p/text()').extract())
74
75          yield item
```

- **4-数据处理**

```
1  """pipelines.py"""
2  # -*- coding: utf-8 -*-
3
4  class SinaPipeline(object):
5      def process_item(self, item, spider):
6          # 文件名使用url地址的中间(即去掉协议和后缀.shtml)
7          filename = item['article_url'][7:-6].replace('/','-')
8          filename_ = item['son_filename'] + filename + '.txt'
9          # 写入本地文件
10         with open(filename_,'w',encoding='utf-8') as f:
11             f.write(item['article_content'])
12
13         return item
```

- **5-全局配置**

```
1  """settings.py"""
2  ROBOTSTXT_OBEY = False
3  DOWNLOAD_DELAY = 1
4  DEFAULT_REQUEST_HEADERS = {
5    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
6    'Accept-Language': 'en',
7    'User-Agent': 'Mozilla/5.0',
8  }
9  ITEM_PIPELINES = {
10    'Sina.pipelines.SinaPipeline': 300,
11 }
```

- **6-运行爬虫**

```
1  """run.py"""
2  from scrapy import cmdline
3  cmdline.execute('scrapy crawl sina'.split())
```

今日作业

```
【1】scrapy框架有哪几大组件？以及各个组件之间是如何工作的？

【2】腾讯招聘尝试改写为scrapy
    2.1) response.text ： 获取页面响应内容
    2.2) scrapy中同样可以使用之前学过的模块,比如果 json模块 等

【3】盗墓笔记小说抓取
    3.1) 目标：抓取盗墓笔记1-8中所有章节的所有小说的具体内容，保存到本地文件
    3.2) http://www.daomubiji.com/
    3.3) 保存路径示例：./novel/盗墓笔记1:七星鲁王宫/七星鲁王_第一章_血尸.txt
                       ./novel/盗墓笔记1:七星鲁王宫/七星鲁王_第二章_五十年后.txt
```

- **盗墓笔记作业提示**

```
【1】一级页面xpath表达式：
    a节点：//li[contains(@id,"menu-item-20")]/a
    title：./text()
    link ：./@href

【2】二级页面xpath表达式
    基准xpath ：//article
    for循环遍历后：
        name=article.xpath('./a/text()').get()
        link=article.xpath('./a/@href').get()

【3】三级页面xpath：
    response.xpath('//article[@class="article-content"]/p/text()').extract()
    # 结果：['p1','p2','p3','']
```