Day08

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)
```

cookie模拟登陆

```
# 适用网站类型
爬取网站页面时需要登录后才能访问，否则获取不到页面的实际响应数据

# 方法1（利用cookie）
    1、先登录成功1次,获取到携带登陆信息的Cookie（处理headers）
    2、利用处理的headers向URL地址发请求

# 方法2(利用requests.get()中cookies参数)
    1、先登录成功1次,获取到cookie,处理为字典
    2、res=requests.get(xxx,cookies=cookies)

# 方法3（利用session会话保持）
    1、实例化session对象
        session = requests.session()
    2、先post : session.post(post_url,data=post_data,headers=headers)
        1、登陆，找到POST地址：form -> action对应地址
        2、定义字典，创建session实例发送请求
            # 字典key ：<input>标签中name的值(email,password)
            # post_data = {'email':'','password':''}
    3、再get : session.get(url,headers=headers)
```

Day08笔记

scrapy框架

- **定义**

```
异步处理框架,可配置和可扩展程度非常高,Python中使用最广泛的爬虫框架
```

- **安装**

```
# Ubuntu安装
sudo pip3 install Scrapy

# Windows安装
cmd命令行(管理员): python -m pip install Scrapy
```

- **Scrapy框架五大组件及工作流程**

```
# 五大组件
1、引擎(Engine)       ：整个框架核心
2、调度器(Scheduler)  ：维护请求队列
3、下载器(Downloader)：获取响应对象
4、爬虫文件(Spider)   ：数据解析提取
```

```
5、项目管道(Pipeline)：数据入库处理

下载器中间件(Downloader Middlewares) ：引擎->下载器,包装请求(随机代理等)
蜘蛛中间件(Spider Middlewares) ：引擎->爬虫文件,可修改响应对象属性

# scrapy爬虫工作流程
1、爬虫启动时由引擎向爬虫程序索要第一个要爬取的URL,交给调度器去入队列
2、调度器处理请求后出队列,通过下载器中间件交给下载器去下载
3、下载器得到响应对象后,通过蜘蛛中间件交给爬虫程序
4、爬虫程序进行数据提取：
    1、数据交给管道文件去入库处理
    2、对于需要继续跟进的URL,再次交给调度器入队列，依次循环
```

- **scrapy常用命令**

```
# 1、创建爬虫项目
scrapy startproject 项目名
# 2、创建爬虫文件
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
```

- **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={}
# 6、项目管道
ITEM_PIPELINES={
```

```
  '项目目录名.pipelines.类名':300
}
```

- **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
/html/head/title/text()
```

- **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

```
# -*- 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run.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 run.py 文件即可
```

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

# https://maoyan.com/board/4?offset=0
```

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):
    name = 'maoyan'
    allowed_domains = ['maoyan.com']
    start_urls = ['https://maoyan.com/board/4?offset=0']
    offset = 0


    def parse(self, response):
        # 给items.py中的类:MaoyanItem(scrapy.Item)实例化
        item = MaoyanItem()

        # 基准xpath
        dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
        # 依次遍历
        for dd in dd_list:
            # 是在给items.py中那些类变量赋值
            item['name'] = dd.xpath('./a/@title').get().strip()
```

```
            item['star'] =
dd.xpath('.//p[@class="star"]/text()').get().strip()
            item['time'] =
dd.xpath('.//p[@class="releasetime"]/text()').get().strip()

            # 把item对象交给管道文件处理
            yield item

        self.offset += 10
        if self.offset <= 91:
            url = 'https://maoyan.com/board/4?offset={}'.format(self.offset)
            # 交给调度器入队列
            yield scrapy.Request(
                url = url,
                callback = self.parse
            )
```

代码实现二

```
import scrapy
from ..items import MaoyanItem

class MaoyanSpider(scrapy.Spider):
    name = 'maoyan3'
    allowed_domains = ['maoyan.com']
    # 去掉start_urls变量

    # 重写start_requests()方法
    def start_requests(self):
        for offset in range(0,91,10):
            url = 'https://maoyan.com/board/4?offset={}'.format(offset)
            yield scrapy.Request(url=url,callback=self.parse)

    def parse(self, response):
        # 给items.py中的类:MaoyanItem(scrapy.Item)实例化
        item = MaoyanItem()

        # 基准xpath
        dd_list = response.xpath('//dl[@class="board-wrapper"]/dd')
        # 依次遍历
        for dd in dd_list:
            # 是在给items.py中那些类变量赋值
            item['name'] = dd.xpath('./a/@title').get().strip()
            item['star'] =
dd.xpath('.//p[@class="star"]/text()').get().strip()
            item['time'] =
dd.xpath('.//p[@class="releasetime"]/text()').get().strip()

            # 把item对象交给管道文件处理
```

```
            yield item
```

4、定义管道文件（pipelines.py）

```
class MaoyanPipeline(object):
    # item: 从爬虫文件maoyan.py中yield的item数据
    def process_item(self, item, spider):
        print(item['name'],item['time'],item['star'])

        return item


import pymysql
from .settings import *

# 自定义管道 - MySQL数据库
class MaoyanMysqlPipeline(object):
    # 爬虫项目开始运行时执行此函数
    def open_spider(self,spider):
        print('我是open_spider函数输出')
        # 一般用于建立数据库连接
        self.db = pymysql.connect(
            host = MYSQL_HOST,
            user = MYSQL_USER,
            password = MYSQL_PWD,
            database = MYSQL_DB,
            charset = MYSQL_CHAR
        )
        self.cursor = self.db.cursor()

    def process_item(self,item,spider):
        ins = 'insert into filmtab values(%s,%s,%s)'
        # 因为execute()的第二个参数为列表
        L = [
            item['name'],item['star'],item['time']
        ]
        self.cursor.execute(ins,L)
        self.db.commit()

        return item

    # 爬虫项目结束时执行此函数
    def close_spider(self,spider):
        print('我是close_spider函数输出')
        # 一般用于断开数据库连接
        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,
   'Maoyan.pipelines.MaoyanMysqlPipeline':200,
}
# 定义MySQL相关变量
MYSQL_HOST = '127.0.0.1'
MYSQL_USER = 'root'
MYSQL_PWD = '123456'
MYSQL_DB = 'maoyandb'
MYSQL_CHAR = 'utf8'
```

6. **创建并运行文件（run.py）**

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

练习

```
添加管道，将抓取数据存入到MongoDB数据库中
库：maoyandb
集合：maoyanset
```

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 }
```

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 process_item(self,item,spider):
        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 ***
```

保存为csv、json文件

- **命令格式**

```
scrapy crawl maoyan -o maoyan.csv
scrapy crawl maoyan -o maoyan.json
# settings.py中设置导出编码 - 主要针对json文件
FEED_EXPORT_ENCODING = 'utf-8'
```

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:
a节点: //li[contains(@id,"menu-item-20")]/a
title: ./text()
link : ./@href

2、二级页面
  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','']
```

- **项目实现**

1、创建项目及爬虫文件

```
1、创建项目 ： scrapy startproject Daomu
2、创建爬虫 ：
   1、cd Daomu
   2、scrapy genspider daomu www.daomubiji.com
```

2、定义要爬取的数据结构 - items.py

```
import scrapy


class DaomuItem(scrapy.Item):
    # 确定pipelines处理数据时需要哪些数据
    # 1. 一级页面标题 - 创建文件夹需要
    title = scrapy.Field()
    # 2. 二级页面标题 - 创建文件需要
    name = scrapy.Field()
    # 3. 小说内容
    content = scrapy.Field()
```

3、爬虫文件实现数据抓取 - daomu.py

```
# -*- 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 - 链接+title
    def parse(self, response):
        # 基准xpath
        a_list = response.xpath('//li[contains(@id,"menu-item-20")]/a')
        for a in a_list:
            item = DaomuItem()
            item['title'] = a.xpath('./text()').get()
            link = a.xpath('./@href').get()
            # 扔给调度器入队列
            yield scrapy.Request(
                url=link,
                # 不同解析函数之间传递数据
                meta={'item':item},
                callback=self.parse_two_page
            )

    # 解析二级页面函数 : 名称(七星鲁王 第一章 血尸)+链接
    def parse_two_page(self,response):
        # 获取item
        item = response.meta['item']
        article_list = response.xpath('//article')
        for article in article_list:
            name = article.xpath('./a/text()').get()
            two_link = article.xpath('./a/@href').get()
            # 继续交给调度器入队列
```

```
            yield scrapy.Request(
                url=two_link,
                meta={'item':item,'name':name},
                callback=self.parse_three_page
            )

    # 解析三级页面：小说内容
    def parse_three_page(self,response):
        item = response.meta['item']
        item['name'] = response.meta['name']
        # p_list: ['段落1','段落2','段落3']
        p_list = response.xpath('//article[@class="article-
content"]//p/text()').extract()
        content = '\n'.join(p_list)
        item['content'] = content

        yield item
```

4、管道文件实现数据处理 - pipelines.py

```
import os

class DaomuPipeline(object):
    def process_item(self, item, spider):
        # item['title']:  盗墓笔记1:七星鲁王宫
        # item['name']:   七星鲁王 第一章 血尸
        # item['content']: 具体小说内容
        # directory: /home/tarena/novel/盗墓笔记1：七星鲁王宫/
        directory = '/home/tarena/novel/{}/'.format(item['title'])
        if not os.path.exists(directory):
            os.makedirs(directory)

        filename = directory + item['name'] + '.txt'
        with open(filename,'w') as f:
            f.write(item['content'])

        return item
```

5、全局配置 - setting.py

6、运行文件 - run.py

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
   response.text ： 获取页面响应内容
3、豆瓣电影尝试改为scrapy
```