Day07回顾

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()
# get()方法会等待页面加载完全后才会继续执行下面语句
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)
```

- **browser执行JS脚本**

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

- **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
browser.switch_to.window(all_handles[1])

# 4、iframe子框架
browser.switch_to.iframe(iframe_element)

# 5、Web客户端验证
url = 'http://用户名:密码@正常地址'
```

execjs模块使用

```
# 1、安装
sudo pip3 install pyexecjs

# 2、使用
with open('file.js','r') as f:
    js = f.read()

obj = execjs.compile(js_data)
result = obj.eval('string')
```

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
1、安装依赖包
    1、sudo apt-get install libffi-dev
    2、sudo apt-get install libssl-dev
    3、sudo apt-get install libxml2-dev
    4、sudo apt-get install python3-dev
    5、sudo apt-get install libxslt1-dev
    6、sudo apt-get install zlib1g-dev
    7、sudo pip3 install -I -U service_identity
2、安装scrapy框架
    1、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**

```
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**

```
# 爬虫项目启动
1、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
```

- **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

```

```

代码实现二

```

```

代码实现三

```

```

3. **定义管道文件（pipelines.py）**

```
# -*- 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
```

```python
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: //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**

```

```

4. **管道文件实现数据处理**

```

```

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、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
4、豆瓣电影尝试改为scrapy
```