Day06回顾

多线程写入同一文件

注意使用线程锁

```
from threading import Lock
lock = Lock()
f = open('xxx.txt','a')

lock.acquire()
f.write(string)
lock.release()

f.close()
```

cookie模拟登陆

```
1、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
2、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
3、方法2（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)
```

三个池子

```
1、User-Agent池
2、代理IP池
3、cookie池
```

解析模块汇总

re、lxml+xpath、json

```
# re
import re
pattern = re.compile(r'',re.S)
r_list = pattern.findall(html)

# lxml+xpath
from lxml import etree
parse_html = etree.HTML(html)
r_list = parse_html.xpath('')

# json
# 响应内容由json转为python
html = json.loads(res.text)
# 所抓数据保存到json文件
with open('xxx.json','a') as f:
    json.dump(item_list,f,ensure_ascii=False)
# 或
f = open('xxx.json','a')
json.dump(item_list,f,ensure_ascii=False)
f.close()
```

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
```

- **安装**

```
1、下载、解压
2、添加到系统环境变量
   # windows: 拷贝到Python安装目录的Scripts目录中
   # Linux :  拷贝到/usr/bin目录中
3、Linux中修改权限
   # sudo -i
   # cd /usr/bin/
   # chmod +x phantomjs
```

- **使用流程**

```
from selenium import webdriver

# 1、创建浏览器对象
# 2、输入网址
# 3、查找节点
# 4、做对应操作
# 5、关闭浏览器
```

- **重要知识点**

```
1、browser.page_source
2、browser.page_source.find('')
3、node.send_keys('')
4、node.click()
5、find_element AND find_elements
6、browser.execute_script('javascript')
7、browser.quit()
```

Day07笔记

京东爬虫案例

- 目标

```
1、目标网址 ： https://www.jd.com/
2、抓取目标 ： 商品名称、商品价格、评价数量、商品商家
```

- 实现步骤

1、找节点

```
1、首页搜索框 ： //*[@id="key"]
2、首页搜索按钮   ://*[@id="search"]/div/div[2]/button
3、商品页的 商品信息节点对象列表 ://*[@id="J_goodsList"]/ul/li
```

2、执行JS脚本，获取动态加载数据

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

3、代码实现

```
from selenium import webdriver
import time

```

```python
class JdSpider(object):
    def __init__(self):
        self.browser = webdriver.Chrome()
        self.url = 'https://www.jd.com/'
        self.i = 0

    # 获取商品页面
    def get_page(self):
        self.browser.get(self.url)
        # 找2个节点
        self.browser.find_element_by_xpath('//*[@id="key"]').send_keys('爬虫书籍')
        self.browser.find_element_by_xpath('//*[@id="search"]/div/div[2]/button').click()
        time.sleep(2)

    # 解析页面
    def parse_page(self):
        # 把下拉菜单拉到底部,执行JS脚本
        self.browser.execute_script(
            'window.scrollTo(0,document.body.scrollHeight)'
        )
        time.sleep(2)
        # 匹配所有商品节点对象列表
        li_list = self.browser.find_elements_by_xpath('//*[@id="J_goodsList"]/ul/li')
        item = {}
        for li in li_list:
            item['name'] = li.find_element_by_xpath('.//div[@class="p-name"] |
.//div[@class="p-name p-name-type-2"]').text
            item['price'] = li.find_element_by_xpath('.//div[@class="p-price"]').text
            item['commit'] = li.find_element_by_xpath('.//div[@class="p-commit"]').text
            item['shopname'] = li.find_element_by_xpath('.//div[@class="p-shopnum"]/a |
.//div[@class="p-shopnum"]/span').text
            print(item)
            self.i += 1

    # 主函数
    def main(self):
        self.get_page()
        while True:
            self.parse_page()
            # 判断是否该点击下一页,没有找到说明不是最后一页
            if self.browser.page_source.find('pn-next disabled') == -1:
                self.browser.find_element_by_class_name('pn-next').click()
                time.sleep(2)
            else:
                break
        print(self.i)

if __name__ == '__main__':
    spider = JdSpider()
    spider.main()
```

chromedriver设置无界面模式

```python
from selenium import webdriver

options = webdriver.ChromeOptions()
# 添加无界面参数
options.add_argument('--headless')
browser = webdriver.Chrome(options=options)
browser.get('http://www.baidu.com/')
browser.save_screenshot('baidu.png')
```

selenium - 键盘操作

示例

```python
from selenium.webdriver.common.keys import Keys

browser = webdriver.Chrome()
browser.get('http://www.baidu.com/')
# 1、在搜索框中输入"selenium"
browser.find_element_by_id('kw').send_keys('赵丽颖')
# 2、输入空格
browser.find_element_by_id('kw').send_keys(Keys.SPACE)
# 3、Ctrl+a 模拟全选
browser.find_element_by_id('kw').send_keys(Keys.CONTROL, 'a')
# 4、Ctrl+c 模拟复制
browser.find_element_by_id('kw').send_keys(Keys.CONTROL, 'c')
# 5、Ctrl+v 模拟粘贴
browser.find_element_by_id('kw').send_keys(Keys.CONTROL, 'v')
# 6、输入回车,代替 搜索 按钮
browser.find_element_by_id('kw').send_keys(Keys.ENTER)
```

selenium - 鼠标操作

示例

```python
from selenium import webdriver
# 导入鼠标事件类
from selenium.webdriver import ActionChains

driver = webdriver.Chrome()
driver.get('http://www.baidu.com/')
#输入selenium 搜索
driver.find_element_by_id('kw').send_keys('赵丽颖')
driver.find_element_by_id('su').click()

#移动到 设置，perform()是真正执行操作，必须有
element = driver.find_element_by_name('tj_settingicon')
ActionChains(driver).move_to_element(element).perform()

#单击，弹出的Ajax元素，根据链接节点的文本内容查找
```

```
driver.find_element_by_link_text('高级搜索').click()
```

selenium - 切换页面

- **适用网站**

```
页面中点开链接出现新的页面，但是浏览器对象browser还是之前页面的对象
```

- **应对方案**

```
# 获取当前所有句柄（窗口）
all_handles = browser.window_handles
# 切换browser到新的窗口，获取新窗口的对象
browser.switch_to_window(all_handles[1])
```

- **民政部网站案例**

目标

```
将民政区划代码爬取到数据库中，按照层级关系（分表 -- 省表、市表、县表）
```

数据库中建表

```
# 建库
create database govdb charset utf8;
use govdb;
# 建表
create table province(
p_name varchar(20),
p_code varchar(20)
)charset=utf8;
create table city(
c_name varchar(20),
c_code varchar(20),
c_father_code varchar(20)
)charset=utf8;
create table county(
x_name varchar(20),
x_code varchar(20),
x_father_code varchar(20)
)charset=utf8;
```

思路

```
1、selenium+Chrome打开一级页面，并提取二级页面最新链接
2、增量爬取：和数据库version表中进行比对，确定之前是否爬过（是否有更新）
3、如果没有更新，直接提示用户，无须继续爬取
4、如果有更新，则删除之前表中数据，重新爬取并插入数据库表
5、最终完成后：断开数据库连接，关闭浏览器
```

代码实现

```python
from selenium import webdriver
import time
import pymysql

class GovementSpider(object):
    def __init__(self):
        self.browser = webdriver.Chrome()
        self.one_url = 'http://www.mca.gov.cn/article/sj/xzqh/2019/'
        # 创建数据库相关变量
        self.db = pymysql.connect(
            'localhost','root','123456','govdb',charset='utf8'
        )
        self.cursor = self.db.cursor()
        # 定义3个列表,为了excutemany()
        self.province_list = []
        self.city_list = []
        self.county_list = []


    # 获取首页,并提取二级页面链接(虚假链接)
    def get_false_url(self):
        self.browser.get(self.one_url)
        # 提取二级页面链接 + 点击该节点
        td_list = self.browser.find_elements_by_xpath(
            '//td[@class="arlisttd"]/a[contains(@title,"代码")]'
        )
        if td_list:
            # 找节点对象,因为要click()
            two_url_element = td_list[0]
            # 增量爬取,取出链接和数据库version表中做比对
            two_url = two_url_element.get_attribute('href')
            sel = 'select * from version where link=%s'
            self.cursor.execute(sel,[two_url])
            result = self.cursor.fetchall()
            if result:
                print('数据已最新,无需爬取')
            else:
                # 点击
                two_url_element.click()
                time.sleep(5)
                # 切换browser
                all_handles = self.browser.window_handles
                self.browser.switch_to.window(all_handles[1])
                # 数据抓取
                self.get_data()
                # 结束之后把two_url插入到version表中
                ins = 'insert into version values(%s)'
                self.cursor.execute(ins,[two_url])
                self.db.commit()


    # 二级页面中提取行政区划代码
    def get_data(self):
        # 基准xpath
```

```
        tr_list = self.browser.find_elements_by_xpath(
            '//tr[@height="19"]'
        )
        for tr in tr_list:
            code = tr.find_element_by_xpath('./td[2]').text.strip()
            name = tr.find_element_by_xpath('./td[3]').text.strip()
            print(name,code)
            # 判断层级关系,添加到对应的数据库表中(对应表中字段)
            # province: p_name p_code
            # city    : c_name c_code c_father_code
            # county  : x_name x_code x_father_code

            if code[-4:] == '0000':
                self.province_list.append([name, code])
                # 单独判断4个直辖市放到city表中
                if name in ['北京市', '天津市', '上海市', '重庆市']:
                    city = [name, code, code]
                    self.city_list.append(city)

            elif code[-2:] == '00':
                city = [name, code, code[:2] + '0000']
                self.city_list.append(city)

            else:
                # 四个直辖市区县的上一级为: xx0000
                if code[:2] in ['11','12','31','50']:
                    county = [name,code,code[:2]+'0000']
                # 普通省市区县上一级为: xxxx00
                else:
                    county = [name, code, code[:4] + '00']

                self.county_list.append(county)

        # 和for循环同缩进,所有数据爬完后统一excutemany()
        self.insert_mysql()

    def insert_mysql(self):
        # 更新时一定要先删除表记录
        del_province = 'delete from province'
        del_city = 'delete from city'
        del_county = 'delete from county'
        self.cursor.execute(del_province)
        self.cursor.execute(del_city)
        self.cursor.execute(del_county)
        # 插入新的数据
        ins_province = 'insert into province values(%s,%s)'
        ins_city = 'insert into city values(%s,%s,%s)'
        ins_county = 'insert into county values(%s,%s,%s)'
        self.cursor.executemany(ins_province,self.province_list)
        self.cursor.executemany(ins_city,self.city_list)
        self.cursor.executemany(ins_county,self.county_list)
        self.db.commit()
        print('数据抓取完成,成功存入数据库')

    def main(self):
        self.get_false_url()
        # 所有数据处理完成后断开连接
```

```
        self.cursor.close()
        self.db.close()
        # 关闭浏览器
        self.browser.quit()

if __name__ == '__main__':
    spider = GovementSpider()
    spider.main()
```

SQL命令练习

```
# 1. 查询所有省市县信息（多表查询实现）
select province.p_name,city.c_name,county.x_name from province,city,county  where province.p_code=city.c_father_code and  city.c_code=county.x_father_code;
# 2. 查询所有省市县信息（连接查询实现）
select province.p_name,city.c_name,county.x_name from province inner join city on province.p_code=city.c_father_code inner join county on  city.c_code=county.x_father_code;
```

selenium - Web客户端验证

弹窗中的用户名和密码如何输入?

```
不用输入，在URL地址中填入就可以
```

示例: 爬取某一天笔记

```
from selenium import webdriver

url = 'http://tarenacode:code_2013@code.tarena.com.cn/AIDCode/aid1904/15-spider/spider_day06_note.zip'
browser = webdriver.Chrome()
browser.get(url)
```

selenium - iframe子框架

特点

```
网页中嵌套了网页，先切换到iframe子框架，然后再执行其他操作
```

方法

```
browser.switch_to.iframe(iframe_element)
```

示例 - 登录qq邮箱

```
from selenium import webdriver
```

```python
import time

driver = webdriver.Chrome()
driver.get('https://mail.qq.com/')

# 切换到iframe子框架
login_frame = driver.find_element_by_id('login_frame')
driver.switch_to.frame(login_frame)

# 用户名+密码+登录
driver.find_element_by_id('u').send_keys('2621470058')
driver.find_element_by_id('p').send_keys('密码')
driver.find_element_by_id('login_button').click()

# 预留页面记载时间
time.sleep(5)

# 提取数据
ele = driver.find_element_by_id('useralias')
print(ele.text)
```

百度翻译破解案例

目标

```
破解百度翻译接口，抓取翻译结果数据
```

实现步骤

- **1、F12抓包,找到json的地址,观察查询参数**

```
1、POST地址: https://fanyi.baidu.com/v2transapi
2、Form表单数据（多次抓取在变的字段）
   from: zh
   to: en
   sign: 54706.276099  #这个是如何生成的?
   token: a927248ae7146c842bb4a94457ca35ee # 基本固定,但也想办法获取
```

- **2、抓取相关JS文件**

```
右上角 - 搜索 - sign: - 找到具体JS文件(index_c8a141d.js) - 格式化输出
```

3、在JS中寻找sign的生成代码

```
1、在格式化输出的JS代码中搜索: sign: 找到如下JS代码: sign: m(a),
2、通过设置断点，找到m(a)函数的位置，即生成sign的具体函数
   # 1. a 为要翻译的单词
   # 2. 鼠标移动到 m(a) 位置处，点击可进入具体m(a)函数代码块
```

4、生成sign的m(a)函数具体代码如下(在一个大的define中)

```
function a(r) {
        if (Array.isArray(r)) {
            for (var o = 0, t = Array(r.length); o < r.length; o++)
                t[o] = r[o];
            return t
        }
        return Array.from(r)
    }
function n(r, o) {
    for (var t = 0; t < o.length - 2; t += 3) {
        var a = o.charAt(t + 2);
        a = a >= "a" ? a.charCodeAt(0) - 87 : Number(a),
            a = "+" === o.charAt(t + 1) ? r >>> a : r << a,
            r = "+" === o.charAt(t) ? r + a & 4294967295 : r ^ a
    }
    return r
}
function e(r) {
    var o = r.match(/[\uD800-\uDBFF][\uDC00-\uDFFF]/g);
    if (null === o) {
        var t = r.length;
        t > 30 && (r = "" + r.substr(0, 10) + r.substr(Math.floor(t / 2) - 5, 10) +
r.substr(-10, 10))
    } else {
        for (var e = r.split(/[\uD800-\uDBFF][\uDC00-\uDFFF]/), C = 0, h = e.length, f = []; h
> C; C++)
            "" !== e[C] && f.push.apply(f, a(e[C].split(""))),
                C !== h - 1 && f.push(o[C]);
        var g = f.length;
        g > 30 && (r = f.slice(0, 10).join("") + f.slice(Math.floor(g / 2) - 5, Math.floor(g /
2) + 5).join("") + f.slice(-10).join(""))
    }
//    var u = void 0
//    , l = "" + String.fromCharCode(103) + String.fromCharCode(116) +
String.fromCharCode(107);
//    u = null !== i ? i : (i = window[l] || "") || "";
//  断点调试,然后从网页源码中找到 window.gtk的值
    var u = '320305.131321201'

    for (var d = u.split("."), m = Number(d[0]) || 0, s = Number(d[1]) || 0, S = [], c = 0, v
= 0; v < r.length; v++) {
        var A = r.charCodeAt(v);
        128 > A ? S[c++] = A : (2048 > A ? S[c++] = A >> 6 | 192 : (55296 === (64512 & A) && v
+ 1 < r.length && 56320 === (64512 & r.charCodeAt(v + 1)) ? (A = 65536 + ((1023 & A) << 10) +
(1023 & r.charCodeAt(++v)),
            S[c++] = A >> 18 | 240,
            S[c++] = A >> 12 & 63 | 128) : S[c++] = A >> 12 | 224,
                                                                S[c++] = A >> 6 & 63 |
128),
                                S[c++] = 63 & A | 128)
    }
```

```
    for (var p = m, F = "" + String.fromCharCode(43) + String.fromCharCode(45) +
String.fromCharCode(97) + ("" + String.fromCharCode(94) + String.fromCharCode(43) +
String.fromCharCode(54)), D = "" + String.fromCharCode(43) + String.fromCharCode(45) +
String.fromCharCode(51) + ("" + String.fromCharCode(94) + String.fromCharCode(43) +
String.fromCharCode(98)) + ("" + String.fromCharCode(43) + String.fromCharCode(45) +
String.fromCharCode(102)), b = 0; b < S.length; b++)
        p += S[b],
            p = n(p, F);
    return p = n(p, D),
        p ^= s,
        0 > p && (p = (2147483647 & p) + 2147483648),
        p %= 1e6,
        p.toString() + "." + (p ^ m)
}
```

- **5、直接将代码写入本地js文件,利用pyexecjs模块执行js代码进行调试**

```
import execjs

with open('node.js','r') as f:
    js_data = f.read()
# 创建对象
exec_object = execjs.compile(js_data)
sign = exec_object.eval('e("hello")')
print(sign)
```

- **获取token**

```
# 在js中
token: window.common.token
# 在响应中想办法获取此值
token_url = 'https://fanyi.baidu.com/?aldtype=16047'
regex: "token: '(.*?)'"
```

- **具体代码实现**

```
import requests
import re
import execjs

class BaiduTranslateSpider(object):
    def __init__(self):
        self.token_url = 'https://fanyi.baidu.com/?aldtype=16047'
        self.post_url = 'https://fanyi.baidu.com/v2transapi'
        self.headers = {
            'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,appl
ication/signed-exchange;v=b3',
            # 'accept-encoding': 'gzip, deflate, br',
            'accept-language': 'zh-CN,zh;q=0.9',
            'cache-control': 'no-cache',
```

```
            'cookie': 'BAIDUID=52920E829C1F64EE98183B703F4E37A9:FG=1;
BIDUPSID=52920E829C1F64EE98183B703F4E37A9; PSTM=1562657403;
to_lang_often=%5B%7B%22value%22%3A%22en%22%2C%22text%22%3A%22%u82F1%u8BED%22%7D%2C%7B%22val
ue%22%3A%22zh%22%2C%22text%22%3A%22%u4E2D%u6587%22%7D%5D; REALTIME_TRANS_SWITCH=1;
FANYI_WORD_SWITCH=1; HISTORY_SWITCH=1; SOUND_SPD_SWITCH=1; SOUND_PREFER_SWITCH=1; delPer=0;
BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; BCLID=6890774803653935935;
BDSFRCVID=4XAsJeCCxG3DLCbwbJrKDGwjNA0UN_I3KhXZ3J;
H_BDCLCKID_SF=tRk8oIDaJCvSe6r1MtQ_M4F_qxby26nUQ5neaJ5n0-
nnhnL4W46bqJKFLtozKMoI3C7fotJJ5nololIRy6CKjjb-jaDqJ5n3bTnjstcS2RREHJrg-
trSMDCShGRGWlO9WDTm_D_KfxnkOnc6qJj0-jjXqqo8K5Ljaa5n-
pPKKRAaqD04bPbZL4DdMa7HLtAO3mkjbnczfn02OP5P5lJ_e-4syPRG2xRnWIvrKfA-
b4ncjRcTehoM3xI8LNj405OTt2LEoDPMJKIbMI_rMbbfhKC3hqJfaI62aKDs_RCMBhcqEIL4eJOIb6_w5gcq0T_Httj
tXR0atn7ZSMbSj4Qo5pK95p38bxnDK2rQLb5zah5nhMJS3j7JDMP0-4rJhxby523i5J6vQpnJ8hQ3DRoWXPIqbN7P-
p5Z5mAqKl0MLIOkbC_6j5DWDTvLeU7J-n8XbI60XRj85-
ohHJrFMtQ_q4tehHRMBUo9WDTm_DoTttt5fUj6qJj855jXqqo8KMtHJaFf-pPKKRAashnzWjrkqqOQ5pj-
WnQr3mkjbn5yfn02OpjPX6joht4syPRG2xRnWIvrKfA-
b4ncjRcTehoM3xI8LNj405OTt2LEoC0XtIDhMDvPMCTSMt_HMxrKetJyaR0JhpjbWJ5TEPnjDUOdLPDW-
46HBM3xbKQw5CJGBf7zhpvdWhC5y6ISKx-_J68Dtf5; ZD_ENTRY=baidu; PSINO=2;
H_PS_PSSID=26525_1444_21095_29578_29521_28518_29098_29568_28830_29221_26350_29459;
locale=zh; Hm_lvt_64ecd82404c51e03dc91cb9e8c025574=1563426293,1563996067;
from_lang_often=%5B%7B%22value%22%3A%22zh%22%2C%22text%22%3A%22%u4E2D%u6587%22%7D%2C%7B%22v
alue%22%3A%22en%22%2C%22text%22%3A%22%u82F1%u8BED%22%7D%5D;
Hm_lpvt_64ecd82404c51e03dc91cb9e8c025574=1563999768;
yjs_js_security_passport=2706b5b03983b8fa12fe756b8e4a08b98fb43022_1563999769_js',
            'pragma': 'no-cache',
            'upgrade-insecure-requests': '1',
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/75.0.3770.142 Safari/537.36',
        }

    # 获取token
    def get_token(self):
        token_url = 'https://fanyi.baidu.com/?aldtype=16047'
        # 定义请求头
        r = requests.get(self.token_url,headers=self.headers)
        token = re.findall(r"token: '(.*?)'",r.text)
        window_gtk = re.findall(r"window.*?gtk = '(.*?)';</script>",r.text)
        if token:
            return token[0],window_gtk[0]

    # 获取sign
    def get_sign(self,word):
        with open('百度翻译.js','r') as f:
            js_data = f.read()

        exec_object = execjs.compile(js_data)
        sign = exec_object.eval('e("{}")'.format(word))

        return sign

    # 主函数
    def main(self,word,fro,to):
        token,gtk = self.get_token()
        sign = self.get_sign(word)
        # 找到form表单数据如下,sign和token需要想办法获取
        form_data = {
            'from': fro,
```

```
            'to': to,
            'query': word,
            'transtype': 'realtime',
            'simple_means_flag': '3',
            'sign': sign,
            'token': token
        }
        r = requests.post(self.post_url,data=form_data,headers=self.headers)
        print(r.json()['trans_result']['data'][0]['dst'])

if __name__ == '__main__':
    spider = BaiduTranslateSpider()
    menu = '1. 英译汉 2. 汉译英'
    choice = input('1. 英译汉 2. 汉译英 : ')
    word = input('请输入要翻译的单词:')
    if choice == '1':
        fro,to = 'en','zh'
    elif choice == '2':
        fro,to = 'zh','en'

    spider.main(word,fro,to)
```

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**

```
# -*- 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

今日作业

1、熟记如下问题

```
1、scrapy框架有哪几大组件?
2、各个组件之间是如何工作的?
```

2、Windows安装scrapy

```
Windows ： python -m pip install Scrapy
# Error: Microsoft Visual C++ 14.0 is required
# 解决 ： 下载安装 Microsoft Visual C++ 14.0
```