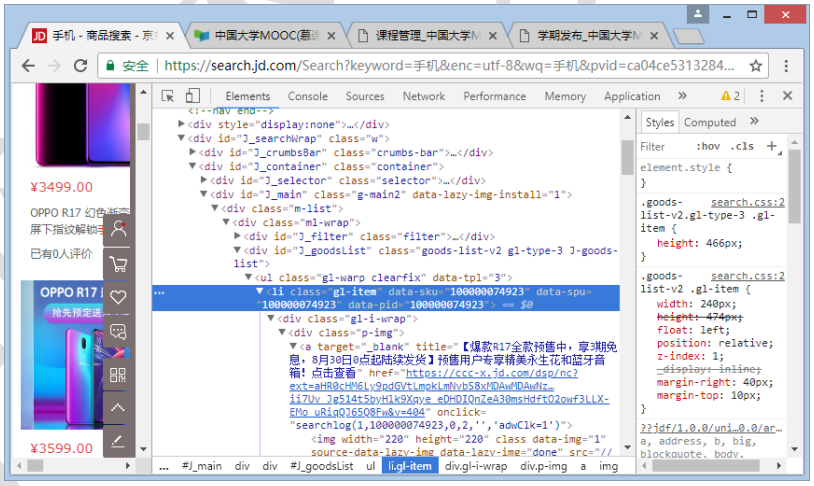
Selenium+Chrome+Scrapy 爬取京东的手机数据

一、网页分析

1、网页浏览

京东网站有很多各种各样的手机买，用 chrome 浏览器进入京东网站输入“手机”，那么慧看到如下的画面：

右键点击第一个手机弹出菜单，选择“检查”可以看到每一个手机的信息是包含在一组<li>的元素中的：



再仔细分析这些<li>都包含在一个 中，而且每个<li>的形式都是<li class="gl-item">，因此分析<li>中的结构就可以找到手机的各种信息，包括图片。

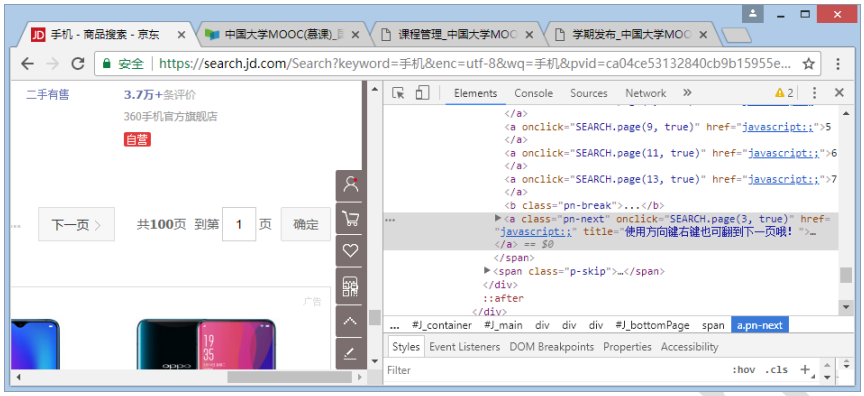
2、数据爬取

设计一个数据库 mobiles.db，它包含 mobiles 表，表中有手机的 mNo,mMark,mPrice,mNote,mFile 等字段，分别表示编号、品牌、价格、说明、图片文件等信息，数据爬起后存储到这个数据库。

3、网页翻页

手机很多，有很多个网页，找到网页的翻页控制，发现翻页不是通过简单的 HTML 代码

控制的，而是通过 JavaScript 控制的，如图：



由此可见要爬取下一个页面的手机就必须获取控制翻页的超级链接元素<a>，并模仿鼠标点击那样去点击该链接，这样就转去下一个页面了。

二、程序设计

根据对网页的分析，设计 scrapy 爬取程序：

(*) items.py

```
import scrapy
class MobileItem(scrapy.Item):
    mNo=scrapy.Field()
    mMark=scrapy.Field()
    mPrice=scrapy.Field()
    mNote=scrapy.Field()
    mFile=scrapy.Field()
```

说明：

mNo、mMark、mPrice、mNote、mFile 分别是手机的编号、型号、价格、描述、图像文件。

(*) settings.py 核心部分

```
DOWNLOADER_MIDDLEWARES = {
    'mobiles.middlewares.MobilesDownloaderMiddleware': 543,
}

ITEM_PIPELINES = {
    'mobiles.pipelines.MobilesPipeline': 300,
}
```

说明：

scrapy 程序采用中间件 MobilesDownloaderMiddleware 实现 Chrome 浏览器的嵌入，当

一个 **Request** 申请时，如果不是图片申请就使用 **Chrome** 浏览器获取网页 **Response**，保证 **JavaScript** 能够执行；如果是图片就不使用 **Chrome** 而是使用默认的方法得到 **Response**。

(*) MySpider.py

```
import scrapy
from mobiles.items import MobileItem
from bs4 import BeautifulSoup
from bs4 import UnicodeDammit
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import pymysql
import os

class MySpider(scrapy.Spider):
    name = "mySpider"

    def clearImages(self):
        try:
            os.mkdir("downloadFiles")
        except:
            pass
        fs=os.listdir("downloadFiles")
        for f in fs:
            os.remove("downloadFiles\\"+f)

    def __init__(self):
        print("MySpider __init__")
        self.clearImages()

        chrome_options = Options()
        chrome_options.add_argument('--headless')
        chrome_options.add_argument('--disable-gpu')
        self.chrome = webdriver.Chrome(chrome_options=chrome_options)
        self.No=0
        try:
            self.con = pymysql.connect(host="127.0.0.1", port=3306, user="root",
passwd="123456", charset="utf8")
            self.cursor = self.con.cursor(pymysql.cursors.DictCursor)
            try:
                self.cursor.execute("create database mydb")
            except:
                pass
            self.con.select_db("mydb")
```

```
                try:
                    self.cursor.execute("drop table mobiles")
                except:
                    pass
                try:
                    sql = """
                        create table mobiles
(
    mNo varchar(8) primary key,
    mMark varchar(512),
    mPrice float,
    mNote varchar(1024),
    mFile varchar(256)
)
                        """
                    self.cursor.execute(sql)
                except:
                    self.cursor.execute("delete from mobiles")
            except Exception as err:
                print(err)


    def close(self, reason):
        self.con.commit()
        self.con.close()
        self.chrome.close()
        print("close")
        print("总共爬取", self.No, "个手机")

    def start_requests(self):
        url = "https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8&wq=%E6%89%8B%E6%9C%BA&pvid=0e1426ee068f4cfe94656dbb0949875f"
        #url="https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8&qrst=1&rt=1&stop=1&vt=2&wq=%E6%89%8B%E6%9C%BA&cid2=653&cid3=655&page=195&s=5820&click=0"
        yield scrapy.Request(url=url,callback=self.parse)

    def parse(self, response):
        print(response.url)
        try:
            dammit = UnicodeDammit(response.body, ["utf-8", "gbk"])
            data = dammit.unicode_markup
            selector=scrapy.Selector(text=data)
            lis=selector.xpath("//div[@id='J_goodsList']//li[@class='gl-item']")
```

```
            for li in lis:
                #We find that the image is either in src or in data-lazy-img attribute
                image=li.xpath(".//div[@class='p-img']//a//img/@src").extract_first()
                if not image:
                    image = li.xpath(".//div[@class='p-img']//a//@data-lazy-
img").extract_first()
                if image:
                    image="http:"+image
                price=li.xpath(".//div[@class='p-price']//i/text()").extract_first()
                note=li.xpath(".//div[@class='p-name p-name-type-2']//em/text()").extract()
                mark=note[0].split(" ")[0]
                self.No=self.No+1
                no=str(self.No)
                while len(no)<6:
                    no="0"+no
                note=",".join(note)

                item=MobileItem()
                item["mNo"]=no
                item["mMark"]=mark
                item["mPrice"]=price
                item["mNote"]=note
                self.cursor.execute("insert into mobiles (mNo) values ('"+no+"')")
                yield item

                if image:
                    request=scrapy.Request(url=image,callback=self.downloadImage)
                    request.meta["mNo"]=no
                    request.meta["image"]=image
                    yield request

            #最后一页时 lastPage 为 None
            lastPage=selector.xpath("//span[@class='p-num']//a[@class='pn-next
disabled']").extract()
            if not lastPage:
                try:
                    nextPage=self.chrome.find_element_by_xpath("//span[@class='p-
num']//a[@class='pn-next']")
                    nextPage.click()
                    url=self.chrome.current_url
                    url = response.urljoin(url)
                    yield scrapy.Request(url=url, callback=self.parse)
                except:
                    pass
```

```
            except Exception as err:
                print(self.chrome.page_source)
                print(err)

    def getFileExt(self,url):
        url=url.strip("/")
        p=url.rfind(".")
        q=url.rfind("/")
        if p>=0 and p>q:
            s=url[p:]
        else:
            s=""
        return s

    def downloadImage(self,response):
        try:
            mNo=response.meta["mNo"]
            image=response.meta["image"]
            fn=mNo+self.getFileExt(image)
            f=open("downloadFiles\\"+fn,"wb")
            f.write(response.body)
            f.close()
            self.cursor.execute("update mobiles set mFile=%s where mNo=%s",(fn,mNo))
        except Exception as err:
            print(err)
```

说明：

在主程序类 MySpider 中初始化时建立数据库与 mobiles 表，建立 Chrome 浏览器的实例，同时清空 downloadFiles 下的所有图像文件。

在一个网页中找到一组数据 mNo、mMark、mPrice、mNote 后就以 mNo 威关键字创建一条数据库记录，然后 yield item 转 pipelines.py 中存储到数据库。如果找到图像就创建一个 Request 请求，这个请求通过 meta["image"]字典标注时图像请求，以便在转中间件时能识别它。图像请求的回调函数是 downloadImage，在该函数保存图像数据，并更新数据库。

(*) pipelines.py

```
class MobilesPipeline(object):
    def process_item(self, item, spider):
        try:
            mNo=item["mNo"]
            mMark=item["mMark"]
            mPrice=item["mPrice"]
            mNote=item["mNote"]
```

```
            sql = "update mobiles set mMark=%s,mPrice=%s,mNote=%s where mNo=%s"
            spider.cursor.execute(sql, (mMark, mPrice, mNote,mNo))
        except Exception as err:
            print(err)
        return item
```

说明：

process_item 中存储数据到数据库。

(*) middlewares.py

```
from scrapy import signals
from scrapy.http import HtmlResponse
import time
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

class MobilesDownloaderMiddleware(object):
    # Not all methods need to be defined. If a method is not defined,
    # scrapy acts as if the downloader middleware does not modify the
    # passed objects.

    @classmethod
    def from_crawler(cls, crawler):
        # This method is used by Scrapy to create your spiders.
        s = cls()
        crawler.signals.connect(s.spider_opened, signal=signals.spider_opened)
        return s

    def process_request(self, request, spider):
        # Called for each request that goes through the downloader
        # middleware.

        # Must either:
        # - return None: continue processing this request
        # - or return a Response object
        # - or return a Request object
        # - or raise IgnoreRequest: process_exception() methods of
        #     installed downloader middleware will be called
        try:
            if "image" in request.meta.keys():
                # 如果是图片请求，例如 request.url="http://.../abc.jpg",那么不需要使用
chrome 加载网页
```

```
                    # 即存在 request.meta["image"]，那么就返回 None，继续这个请求
                    return None
                else:
                    # 如果是网页请求，那么就使用 chrome 获取网页，并返回 response 对象。
                    spider.chrome.get(request.url)
                    wait = WebDriverWait(spider.chrome, 60)
                    locator=(By.XPATH,"//span[@class='p-num']")
                    wait.until(EC.presence_of_element_located(locator))
                    #time.sleep(1)
                    content = spider.chrome.page_source
                    response = HtmlResponse(request.url, encoding='utf-8', body=content,
request=request)
                    return response
            except Exception as err:
                print(err)

    def process_response(self, request, response, spider):
        # Called with the response returned from the downloader.

        # Must either;
        # - return a Response object
        # - return a Request object
        # - or raise IgnoreRequest
        return response

    def process_exception(self, request, exception, spider):
        # Called when a download handler or a process_request()
        # (from other downloader middleware) raises an exception.

        # Must either:
        # - return None: continue processing this exception
        # - return a Response object: stops process_exception() chain
        # - return a Request object: stops process_exception() chain
        pass

    def spider_opened(self, spider):
        spider.logger.info('Spider opened: %s' % spider.name)
```

说明：

这个中间件十分关键，它负责建立网页的 Response 对象。Request 请求通过 meta["image"]字典标注它是否是图像请求。

如果请求 Request 是普通网页，即 meta["image"]为空，那么就使用 Chrome 浏览器获取网页，并由此创建 Response 对象，回调 MySpider 的 parse 函数。

如果 Request 是图片请求，即 meta["image"]不为空，那么就获取图片请求的 Response

对象，之后回调 MySpider 中的 downloadImage 函数。

三、数据爬起

执行该程序，会看到一个个页面被访问，数据被爬取后存储到数据库，总共发现近 200 多个页面，2000 多种手机。

四、结果展示

为了展示爬取的结果，设计程序 show.py 列出 10 条记录，程序如下：

```
import pymysql
import flask
app=flask.Flask(__name__,static_folder="downloadFiles")

@app.route("/")
def show():
    s = "<table border='1' width='800'><tr><th>编号</th><th>品牌</th><th>价格</th><th>说明</th><th>图像</th></tr>"
    try:
        con = pymysql.connect(host="127.0.0.1", port=3306, user="root", passwd="123456", charset="utf8",db="mydb")
        cursor = con.cursor()
        cursor.execute("select mNo,mMark,mPrice,mNote,mFile from mobiles order by mNo LIMIT 10")
        rows = cursor.fetchall()
        print(len(rows))
        for row in rows:
            s=s+"<tr><td>"+row[0]+"</td><td>"+row[1]+"</td><td>"+str(row[2])+"</td><td>"+row[3]+"</td><td>"
            s=s+"<img width='100' height='100' src='downloadFiles/"+row[4]+"'></td></tr>"
        con.close()
    except Exception as err:
        print(err)
    s=s+"</table>"
    return s

if __name__=="__main__":
    app.run()
```

执行该程序并访问 http://127.0.0.1:5000，结果如图：

编号	品牌	价格	说明	图像
000001	华为（HUAWEI）	758.0	华为（HUAWEI）畅享7，黑色 移动全网通（3GB+32GB）	
000002	华为（HUAWEI）	2788.0	华为（HUAWEI）华为 P10 全网通4G智能, 双卡双待 钻雕金（4G+128G）	
000003	Apple	5199.0	Apple 苹果iPhone 7 移动联通电信4G, 红色 256G 标配	