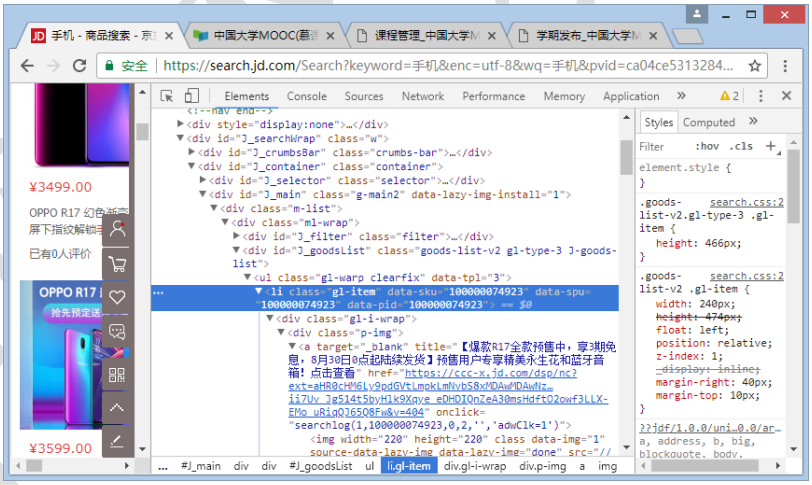
Selenium+Chrome 爬取京东的手机数据

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。

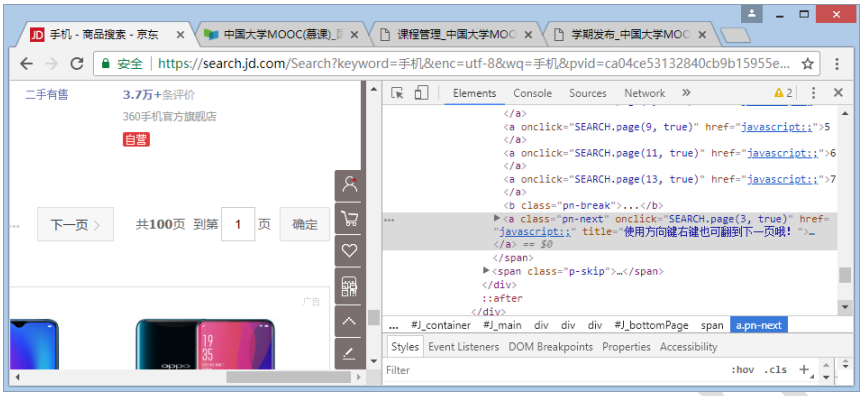
2、数据爬取

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，设计爬取程序 spider.py 如下：

```
import scrapy
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import urllib.request
import threading
import sqlite3
import os
import datetime

class MySpider:
    headers = {"User-Agent": "Mozilla/5.0 (Windows; U; Windows NT 6.0 x64; en-US; rv:1.9pre) Gecko/2008072421 Minefield/3.0.2pre"}
    imagePath="images"

    def startUp(self):
        # Initializing Chrome browser
        chrome_options = Options()
        chrome_options.add_argument('--headless')
        chrome_options.add_argument('--disable-gpu')
        self.web = webdriver.Chrome(chrome_options=chrome_options)
        # Initializing variables
        self.threads=[]
        self.No=0
        # Initializing database
        try:
            self.con = sqlite3.connect("mobiles.db")
```

```
                self.cursor = self.con.cursor()
                try:
                    #如果有 students 表就删除
                    self.cursor.execute("drop table mobiles")
                except:
                    pass
                try:
                    #建立新的 students 表
                    sql = "create table mobiles (mNo varchar(32) primary key,mMark varchar(256),mPrice float,mNote varchar(1024),mFile varchar(32))"
                    self.cursor.execute(sql)
                except:
                    pass
            except Exception as err:
                print(err)
            # Initializing images folder
            try:
                if not os.path.exists(MySpider.imagePath):
                    os.mkdir(MySpider.imagePath)
                images=os.listdir(MySpider.imagePath)
                for img in images:
                    s=os.path.join(MySpider.imagePath,img)
                    os.remove(s)
            except Exception as err:
                print(err)

        def closeUp(self):
            try:
                self.con.commit()
                self.con.close()
            except Exception as err:
                print(err);

        def insertDB(self,mNo,mMark,mPrice,mNote,mFile):
            try:
                sql = "insert into mobiles (mNo,mMark,mPrice,mNote,mFile) values (?,?,?,?,?)"
                self.cursor.execute(sql, (mNo, mMark, mPrice, mNote,mFile))
            except Exception as err:
                print(err)

        def showDB(self):
            try:
                print("%8s %16s %8s %16s %s" % ("No","Mark","Price","Image","Note"))
```

```
                self.cursor.execute("select mNo,mMark,mPrice,mFile,mNote from mobiles
order by mNo")
                rows=self.cursor.fetchall()
                for row in rows:
                    print("%8s %16s %8d %16s %s" % (row[0],row[1],row[2],row[3],row[4]))
            except Exception as err:
                print(err)

        def parseItem(self,selector,url):
            try:
                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)
                    if image:
                        if (image[len(image) - 4] == "."):
                            ext = image[len(image) - 4:]
                        else:
                            ext = ""
                        mFile=no+ext
                        T = threading.Thread(target=self.downloadImage, args=(image,
mFile))
                        T.setDaemon(False)
                        T.start()
                        self.threads.append(T)
                    else:
                        mFile=""
                    self.insertDB(no, mark, float(price), note,mFile)
            except Exception as err:
                print(err)
```

```
        def downloadImage(self,url, mFile):
            try:
                req = urllib.request.Request(url, headers=MySpider.headers)
                data = urllib.request.urlopen(req, timeout=200)
                data = data.read()
                fobj = open(MySpider.imagePath+"\\" +mFile, "wb")
                fobj.write(data)
                fobj.close()
            except Exception as err:
                print(mFile+" "+str(err)+" url="+url)

        def processSpider(self, url):
            print(url)
            try:
                self.web.get(url)
                selector=scrapy.Selector(text=self.web.page_source)
                self.parseItem(selector,url)
                lastPage=selector.xpath("//span[@class='p-num']//a[@class='pn-next
disabled']").extract()
                if not lastPage:

nextPage=self.web.find_element_by_xpath("//span[@class='p-num']//a[@class='pn-next']")
                    nextPage.click()
                    self.processSpider(self.web.current_url)
            except Exception as err:
                print(err)

        def executeSpider(self,url):
            starttime = datetime.datetime.now()
            print("Spider starting......")
            self.startUp()
            self.processSpider(url)
            self.showDB()
            self.closeUp()
            for t in self.threads:
                t.join()
            print("Spider completed......")
            endtime = datetime.datetime.now()
            elapsed=(endtime-starttime).seconds
            print("Total ",elapsed," seconds elapsed")


    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"
    spider=MySpider()
    spider.executeSpider(url)
```

这个程序从第一页入口地址开始，模拟 chrome 浏览器，获取网页文档，爬取手机品牌、价格、说明等数据并存储到数据库，同时启动一个子线程去爬取图像，该图像北存储到 images 的子文件夹中，然后程序获取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转去下一个页面继续爬取手机数据，一直到最后一个页面为止。程序使用一个全程的 threads 列表记录全部的爬取图片的子线程，等待这些线程结束时程序就结束。

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 mobiles.db，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 sqlite3
import flask
app=flask.Flask(__name__,static_folder="images")

@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 = sqlite3.connect("mobiles.db")
        cursor = con.cursor()
        cursor.execute("select mNo,mMark,mPrice,mNote,mFile from mobiles order by mNo LIMIT 10")
        rows = cursor.fetchall()
        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='images/"+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	