基于python汽车之家数据爬取的设计与实现

**爬虫的技术准备：**

Python的基础语言：基础语法、运算符、数据类型、流程控制、函数、对象、

模块（库）、文件操作等

W3C标准：HTML、CSS、JavaScript、JSON

HTTP标准：HTTP的请求过程、请求方式、状态码的含义、头部信息以及Cookie状态管理

爬虫的基本流程：访问url ->解析网页->爬取数据->存储数据（通过不断循环直到满足爬虫停止条件为止）

爬虫的难点主要是分析网页的规律，和应对反爬的一些措施

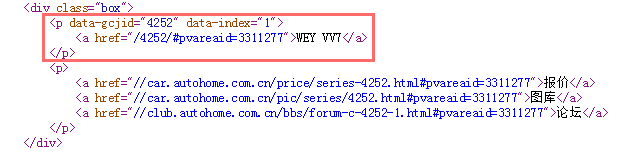
下面通过爬取汽车之家网站示例来展示简单爬虫的实现过程（这里使用的浏览器是chrome浏览器）：

**1.车型数据的爬取**

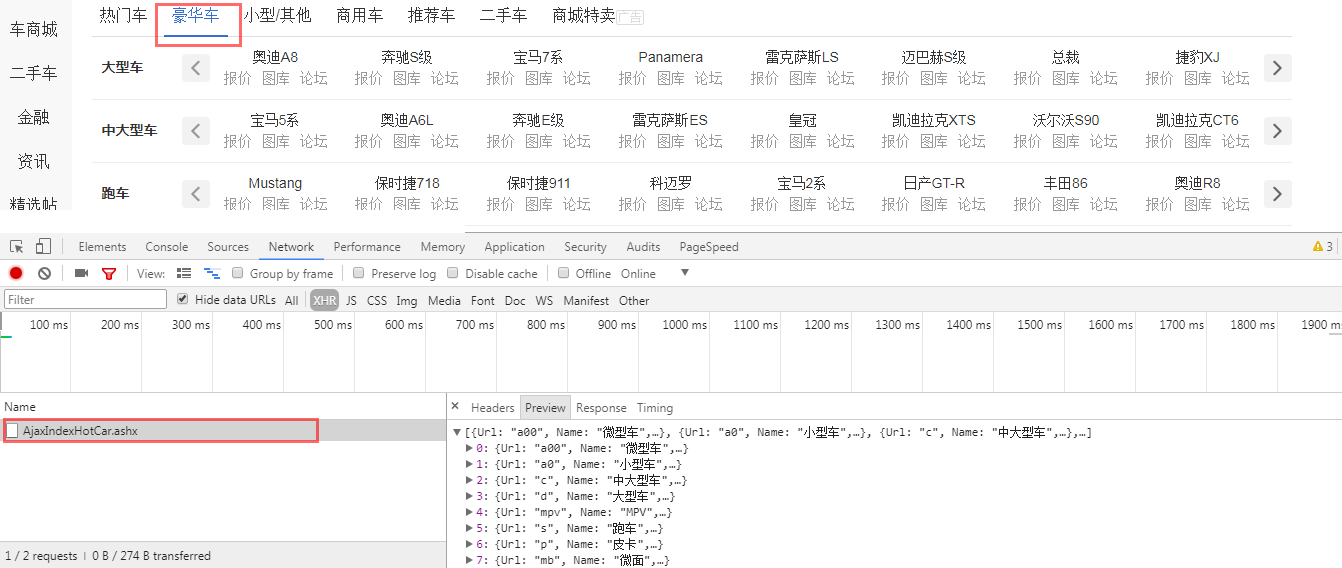
首先分析汽车之家首页车型的栏目，如下图



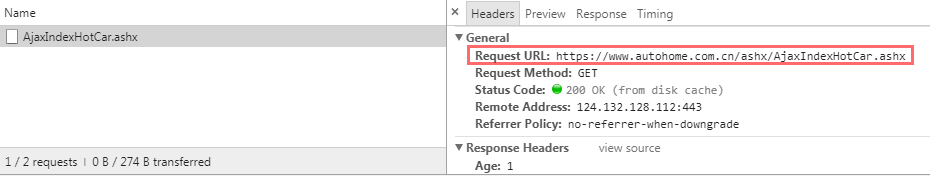
通过右键“查看网页源代码”我们可以容易的找到“热门车”的车型名称和车型id，如下图



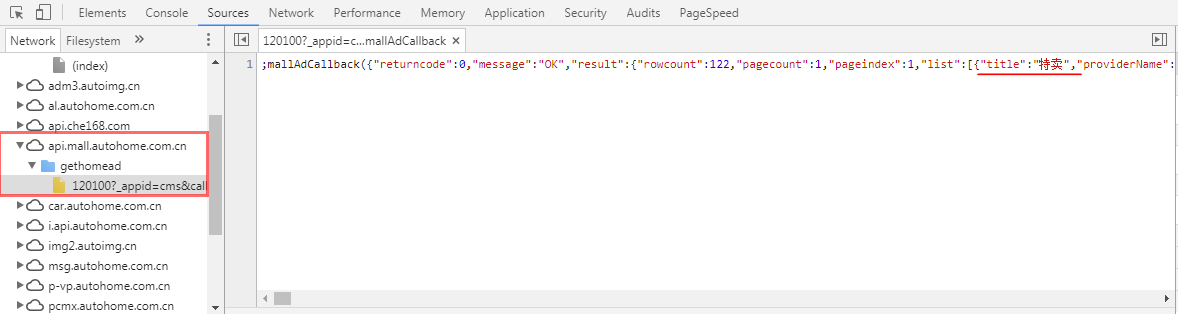
然而，其他例如“豪华车”等标签的车型从源代码中无法直接获取，这时我们可以右键点开“检查模式”，点击“network”，再点击“豪华车”标签，可以看到网页中加载了一个文件，如下图



点击刚加载的那个文件，可以发现这个文件就是用json数据格式存储其它剩下的车型信息，点开它的“header”信息，并记录下这个文件的路径，我们可以通过这个路径爬取剩下的车型信息。如下图



这样车型的基本数据我们就可以得到，当我们想爬取拥有“特卖”标签的车型时，直接从源代码及存储车型文件中无法直接获取，这时我们可以右键“检查”，点击“soure”，这里存储着网页的一些源文件，我们在源文件中发现了存有特卖标签的车型的文件，如下图



我们右键这个文件选择“Open in the new tab”，可以在新的页面中看到这个文件就是“特卖”标签的车型数据，如下图



seriesId相对应的就是车型的id，但这种数据格式并不是我们能处理的数据格式，我们需要把“&callback=mallAdCallback”去除，就得到了标准的json数据格式，把这个链接记录下来，为后面数据的爬取作准备。

如果我们想要爬取车型的基本数据，我们可以点击一个车型的链接，如下图



首先我们观察多个车系首页的链接可以发现，它的url除了中间的一串数字不一样，其它都一样，然后可以进一步发现，这一串数字就是对应该车系的id，这样知道每个车系的id，就能很快的链接到这个页面，从网页源代码中就能爬取到这个车系的基本信息。

经过以上分析后，就可以进行编写爬虫脚本了。

首先爬取一个网页的内容，先获取url对应的响应内容，python提供了urllib2（urllib3）、requests、splinter，这里我使用requests来响应url，

result=requests.get(url)

响应完url后，就要解析网页里的内容，python提供了lxml、beautiful4、re、pyquery等数据解析的库，这里我使用beautiful4来解析数据，

content=BeautifulSoup(result.text, "html.parser")

对于可以直接从网页源代码中获得的数据，数据解析完之后，就可以提取数据，常用的方法有：xpath路径表达式、CSS选择器、正则表达式等。其中，xpath路径表达式、CSS选择器主要用于提取结构化的数据，而正则表达式主要用于提取非结构化的数据。这里我比较常用的是正则表达式，

re\_match=r'<p data-gcjid=.\*? data-index=.\*?>\n<a href="/(.\*?)/#pvareaid=.\*?">(.\*?)</a>\n</p>'

car\_name=re.findall(re\_match,str(content), re.S|re.M)

这样我们就能得到车名和车型的id。

对于获取的是json数据格式，用requests响应后，我们需要用python中的json进行解析，然后再进行数据提取。

car\_list=json.loads(result.decode('GBK'))

这里需要注意先把数据先解码为GBK格式再进行json解析。

通过这些基本的操作就能得到我们需要的内容，剩下的就是一些基本的python数据操作。

**2.车型文章内容的爬取**

爬取车型文章的数据我们先对车型文章的页面进行观察，如下图，



通过观察发现，车型文章的url的组成方式为：https://www.autohome.com.cn/车型id/0/0-0-文章页码-0/

这样我们通过车型id和文章页码就能获取到文章页面，从页面中就可以直接得到文章的标题、url、浏览量等信息。车型的id我们从之前的车型数据爬取就能获得，而文章的页码我们可以从第一页中页面底部获取文章的页数，如下图，



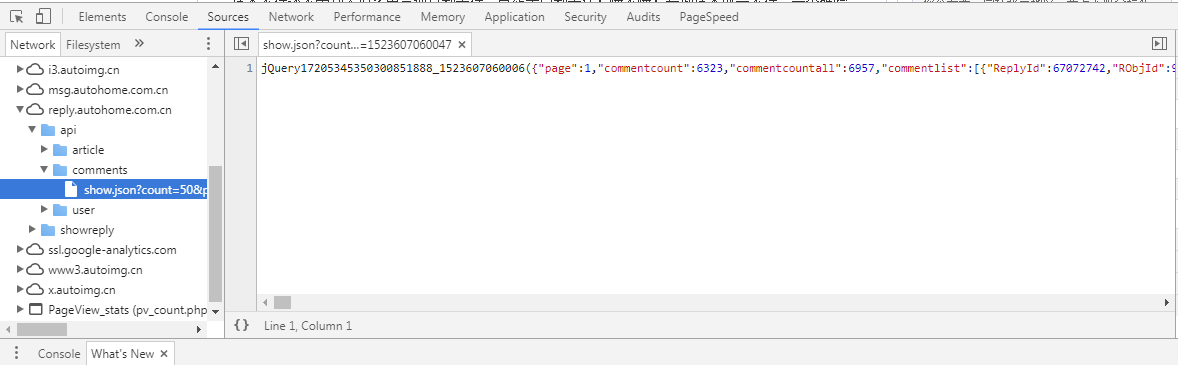
因为不能直接得到页码的最大值，具体获取页码最大值的方式为：

1）用正则匹配出页面中的页码数字

2）找到页码数字中最大值

这样我们有了车型id和文章的总页数，就可以通过循环的方式，经过“车型数据爬取”中提到的:响应url->解析内容->匹配->存储数据。这一基本过程就能获取文章的基本信息，如：标题，url，浏览量，评论量等。

当我们要获取文章评论时，点击到文章评论网页中进行观察发现，网页源代码中并没有评论的信息，但我们从源文件发现存储评论的文件，如下图



观察评论文件的链接可以发现它的组成方式：https://reply.autohome.com.cn/api/comments/show.json?count=50&page=评论页码 &id=文章id&appid=1

这里我们需要得到文章id和评论总页码，这样就可以循环得到评论的json数据格式，通过响应url->json解析->提取数据->存储数据的过程就能得到每篇文章的评论内容。

文章id在每篇文章的详情页面就能匹配到，而评论的页码，我采用的是利用评论第一页的json数据获取评论的总数，然后除以每页50条评论得到评论的总页数。

这样文章的基本信息和评论就都能获取到了。

**3.车型口碑数据的爬取**

首先观察口碑页面发现，口碑页面的url的组成：https://k.autohome.com.cn/车型id/index\_口碑页码.html?pvareaid=2099118#dataList，车型id在“车型数据的爬取”中可以得到，口碑页码可以从口碑的首页中的底部直接匹配到，如下图



然后就是循环链接，通过响应url->解析内容->匹配->存储数据，爬取到全部口碑的链接。

最后循环口碑链接，匹配出需要的数据，口碑评论的爬取同文章评论的爬取类似。若是在网页源代码中不能直接得到的数据，试着在源文件中找找是否存有这些数据。

**4.车型论坛数据的爬取**

首先观察论坛页面发现，论坛页面的url的组成：https://club.autohome.com.cn/bbs/forum-c-车型id-论坛页码.html?qaType=-1#pvareaid=101061，车型id在“车型数据的爬取”中可以得到，论坛页码可以从论坛首页中的底部直接匹配到，如下图



然后就是循环链接，通过响应url->解析内容->匹配->存储数据，爬取到全部论坛的链接和论坛的一些数据。

再对论坛链接进行循环的方式爬取论坛内容和回复，论坛回复的爬取跟之前有些不一样，因为没有相对应的json格式文件进行爬取，所以我们得直接从网页源代码中直接获取，如下图



由于回复的标签有许多种，我们不能通过一个正则匹配表达式进行直接匹配，为了解决这个问题，我采用的方法是先对红框内的整体进行匹配，再对红框里的内容进行一一匹配，这样能解决回复楼层中标签不一的问题。

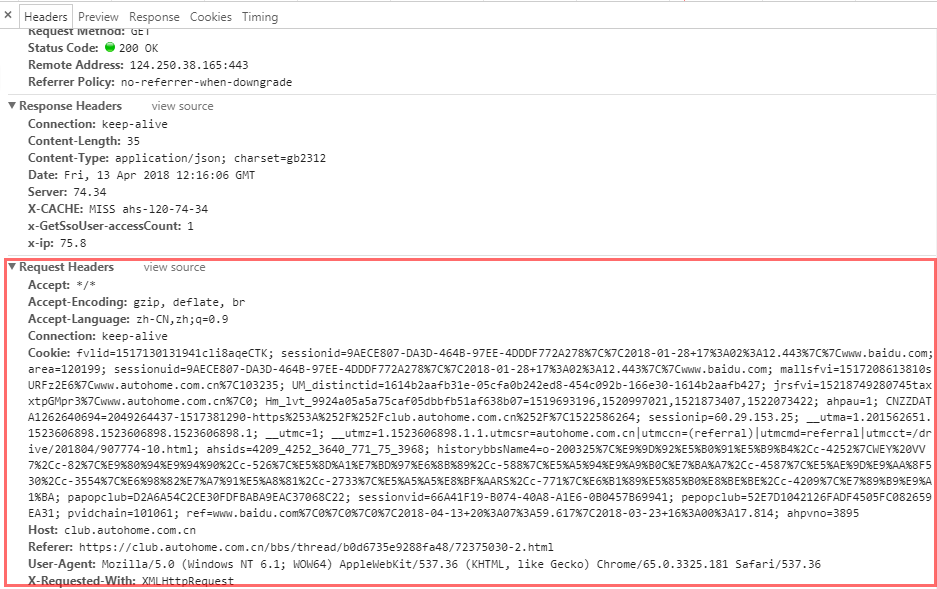
最后循环帖子链接https://club.autohome.com.cn/bbs/thread/b0d6735e9288fa48/72375030-页码.html中的页码，对回复的信息进行爬取，页码如下图，



这样论坛的数据也就爬取完成了。

**简单的反反爬虫技巧：**

1）模拟浏览器的头部进行响应url，浏览器的头部可以从网页中查看，如下图，



通过设置浏览器头部和cookie，就可以让服务器认为是人在访问，而不是机器人，当然不同的浏览器有着不同的头部，可以每次响应url时不断的变换头部达到混淆效果。

headers = {"User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36","Accept": "application/json, text/javascript, \*/\*; q=0.01"}

result=requests.get(url,headers=headers)

2）虽然设置了不同的浏览器的头部，但访问网站的ip地址却是唯一的，服务器会检测用户的ip地址，若一个ip短时间内不断响应的话，就会被当作爬虫屏蔽掉，所以可以通过代理ip的方式不断切换ip地址解决这一效果，但这要保证代理ip可用并且是高匿的，否则真正的ip还是会被检测出来。因此我们需要许多的代理ip，构建一个ip池来供我们使用。

proxies={'代理ip类型': '代理ip'}

result=requests.get(url, proxies= proxies)

3）长时间内不断的响应一个服务器，有可能会造成服务器的瘫痪，也不利于数据的爬取。所以最好每响应一次就间隔一段时间再进行爬取，以此来减轻服务器的压力，也防止被反爬虫识别。

time.sleep（时间s）

**解决论坛帖子内容字体缺失的问题：**

当我们对帖子帖主进行爬取的时候，发现帖子内容里有些字体是缺失的，如下图，



一些字符是由span标签替代，当我们正常浏览网页时没有问题，查看网页源代码时就发现了这个问题。通过观察发现，这些字符是被重新编码过了，防止一些爬虫爬取到内容。

解决方案：

1）找到这些字符重新编码的文件

通过查看和搜索网页源代码，我们可以发现字符的文件，如下图



通过访问这个链接，可以下载该字体文件，利用字体查看器（FontCreator）打开，如下图，



从图中我们可以知道，字符和新的字符编码，通过观察多个字体文件可知，缺失字符的数量和顺序是不变的，所以我们可以事先定义好缺失字符的列表

wordList = ['一', '七', '三', '上', '下', '不', '九', '了', '二', '五', '低', '八','六', '十', '的', '着', '近', '远', '长', '右', '呢', '和', '四', '地', '坏','多', '大', '好', '小', '少', '短', '矮', '高', '左', '很', '得', '是', '更']

2）让字符与编码一一对应还原回去

python中有专门接替ttf文件的库TTFont，通过以下代码可以解析ttf文件，并构建与汉字匹配的编码列表。

font = TTFont('autohome.ttf')

uniList = font['cmap'].tables[0].ttFont.getGlyphOrder()

utfList=[]

for uni in uniList[1:]:

utfList.append((r'\u'+uni[3:].lower()).encode('utf-8').decode("unicode\_escape"))

通过帖子内容的编码可以找到在编码列表的位置，再通过编码的位置对应上相应的汉字，从而替换到帖子内容的编码，这样就可以解决帖子内容不全的问题。