目录

HTML解析库Python版jQuery：PyQuery

- 最新版本： `v1.0`
- 更新时间： `20210414`

简介

介绍HTML解析领域的Python版的jQuery：PyQuery。先对PyQuery概述，再介绍常见的基本操作，包括如何获取元素的文本值，如何获取带子元素的元素的文本值；获取元素的属性，如何用CSS选择器去选择定位查找元素；对比了find和items的返回结果；整理了CSS选择器的基本语法；给出了具体的实际例子，比如用PySpider抓取汽车之家的品牌和车型车系数据中如何使用PyQuery。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/python_html_parse_pyquery: HTML解析库Python版jQuery：PyQuery

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- HTML解析库Python版jQuery：PyQuery book.crifan.com
- HTML解析库Python版jQuery：PyQuery crifan.github.io

离线下载阅读

- HTML解析库Python版jQuery：PyQuery PDF
- HTML解析库Python版jQuery：PyQuery ePub
- HTML解析库Python版jQuery：PyQuery Mobi

版权说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

PyQuery概述

- 概述
  - 作用
    - `Python` 版的 `jQuery`
      - 注：jQuery是 `JavaScript` 中的库，可以（很方便的）操作 `HTML` （中的元素）
  - 使用
    - PySpider内部也用到了PyQuery
- 官网文档
  - 入口
    - https://pythonhosted.org/pyquery/index.html
  - Quickstart = 快速开始
    - https://pythonhosted.org/pyquery/index.html#quickstart
  - Attributes = 元素属性
    - https://pythonhosted.org/pyquery/attributes.html
  - CSS = 用CSS选择器选择元素
    - https://pythonhosted.org/pyquery/css.html
  - Manipulating = 操作元素
    - https://pythonhosted.org/pyquery/manipulating.html
  - Traversing = 元素遍历
    - https://pythonhosted.org/pyquery/traversing.html
  - API
    - https://pythonhosted.org/pyquery/api.html
  - Scraping = 抓取（页面）
    - https://pythonhosted.org/pyquery/scrap.html
  - pyquery.ajax – PyQuery AJAX extension = AJAX扩展
    - https://pythonhosted.org/pyquery/ajax.html
  - Tips = 提示
    - https://pythonhosted.org/pyquery/tips.html
  - Testing = 测试
    - https://pythonhosted.org/pyquery/testing.html

PyQuery常见操作

获取元素的文本值

举例：html

```
<p class="contributors">
  By
  <a href="/teachers/authors/judy-love.html" target="_self"
  ,
  <a href="/teachers/authors/julie-danneberg.html" target="
</p>
```

写法：

`curHtmlElement.text()`

举例：

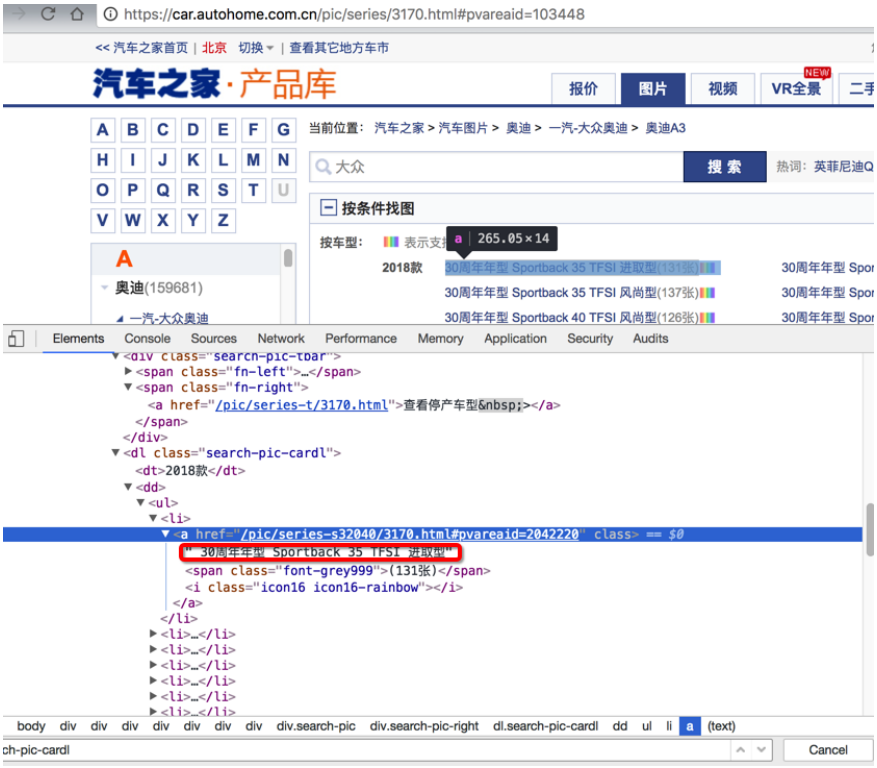
```
authors = []
for eachAuthor in response.doc('p[class="contributors"] a[
    print("eachAuthor=%s", eachAuthor)
    authorText = eachAuthor.text()
```

详见官网文档：

https://pythonhosted.org/pyquery/api.html#pyquery.pyquery.PyQuery.text

获取含有子元素的 `a` 的本身的 `text`

页面：



html

```
<a href="/pic/series-s32040/3170.html#pvareaid=2042220" cla
  <span class="font-grey999">(131张)</span>


  <i class="icon16 icon16-rainbow"></i>
</a>
```

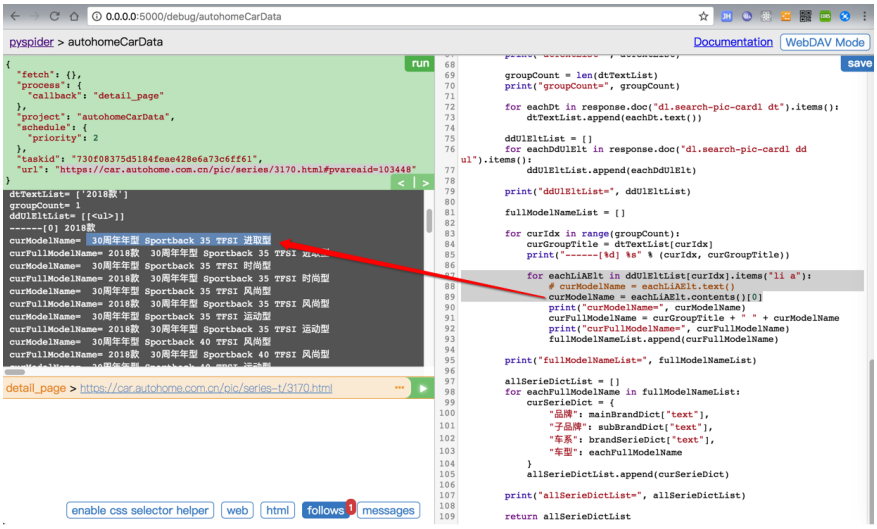
想要：获取a的本身的text

代码：

```
for eachLiAElt in ddUlEltList[curIdx].items("li a"):
    curModelName = eachLiAElt.contents()[0]
```

输出值：`30周年年型 Sportback 35 TFSI 进取型`

效果：



- 官网文档
  - pyquery.pyquery.PyQuery.contents
    - > Return contents (with text nodes)
  - 和之前用过的 `BeautifulSoup` 的 `contents` 是类似的
    - BeautifulSoup contents
      - > tag的 .contents 属性可以将tag的子节点以列表的方式输出

直接用text()会包含子元素的text值

如果直接用 `text()`：

```
for eachLiAElt in ddUlEltList[curIdx].items("li a"):
    curModelName = eachLiAElt.text()
```

是错误的写法。

因为会把 `span中 的 text`，此处的 `(131张)` 也包括在内

获取元素的属性的值

语法：

- `attr`

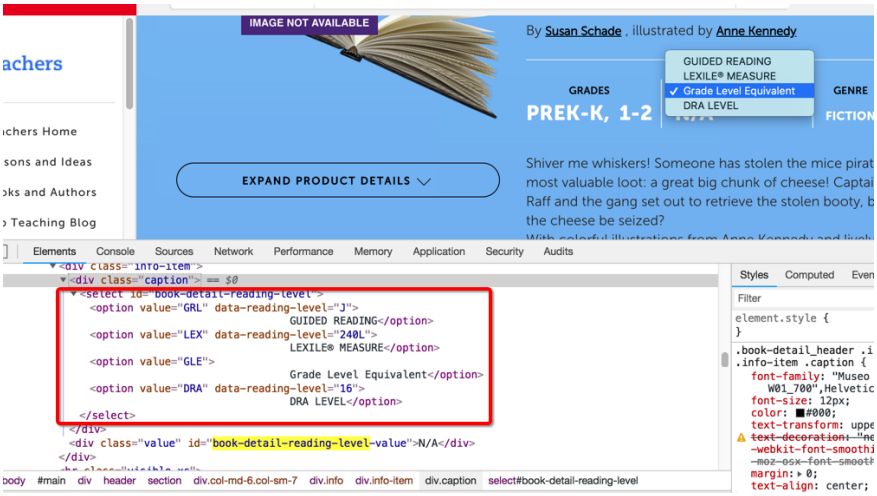
  ```
  htmlElement.attr("property name")
  ```

- 如果 `property name` 是连接在一起的单词这种，比如 `xxx`，`xxx_yyy`，那么可以写成：

  ```
  htmlElement.attr.property_name
  ```

举例1

```
<div class="caption">
    <select id="book-detail-reading-level">
        <option value="GRL" data-reading-level="J">
            GUIDED READING</option>
        <option value="LEX" data-reading-level="240L">
            LEXILE® MEASURE</option>
        <option value="GLE">
            Grade Level Equivalent</option>
        <option value="DRA" data-reading-level="16">
            DRA LEVEL</option>
    </select>
</div>
```



对应代码：

```
readingLevelSelectElement = readingLevelElement.find('selec
print("readingLevelSelectElement=%s" % readingLevelSelectE
guidedReading = readingLevelSelectElement.find('option[valu
lexileMeasure = readingLevelSelectElement.find('option[valu
gradeLevelEquivalent = readingLevelSelectElement.find('opti
draLevel = readingLevelSelectElement.find('option[value="DR
print("guidedReading=%s" % guidedReading)
print("lexileMeasure=%s" % lexileMeasure)
print("gradeLevelEquivalent=%s" % gradeLevelEquivalent)
print("draLevel=%s" % draLevel)
```

即可提取到内容：

```
readingLevelSelectElement=<select id="book-detail-reading-
                                <option value="GRL" data-re
                                        GUIDED READING</opt
                                <option value="LEX" data-re
                                        LEXILE® MEASURE</op
                                <option value="GLE">
                                        Grade Level Equiva
                                <option value="DRA" data-re
                                        DRA LEVEL</option>
                        </select>
guidedReading=J
lexileMeasure=240L
gradeLevelEquivalent=None
draLevel=16
```

举例2

页面：

html:

```
<span data-price="14.59万元" class="fn-left">
                                厂商指导价：14.59万元</span>
```

想要：获取 `span` 的 `data-price` 的属性的值

- 正确写法

```
msrpPriceStr = msrpPriceElt.attr("data-price")
```

- 错误写法：

```
msrpPriceStr = msrpPriceElt.attr.data-price
```

不是用PyQuery.val去获取元素属性值

注意此处不是

pyquery – PyQuery complete API — pyquery 1.2.4 documentation

中的：

```
PyQuery.val(value=<NoDefault>)
```

去获取属性的值的

`CSS` 选择器的写法

`attribute = value` 类的元素定位

例子1

调试看到的 html代码是：

```
<ul class="list-user list-user-1" id="list-user-1">
  <li>
    <a href="/index.php?m=home&c=match_new&a=video&show_id=
```

`PySpider` 内部会自动加上当前host，即http前缀的 `http://xxx` ，实际上是：

```
<ul class="list-user list-user-1" id="list-user-1">
  <li>
    <a href="http://xxx/index.php?m=home&c=match_new&a=vi
```

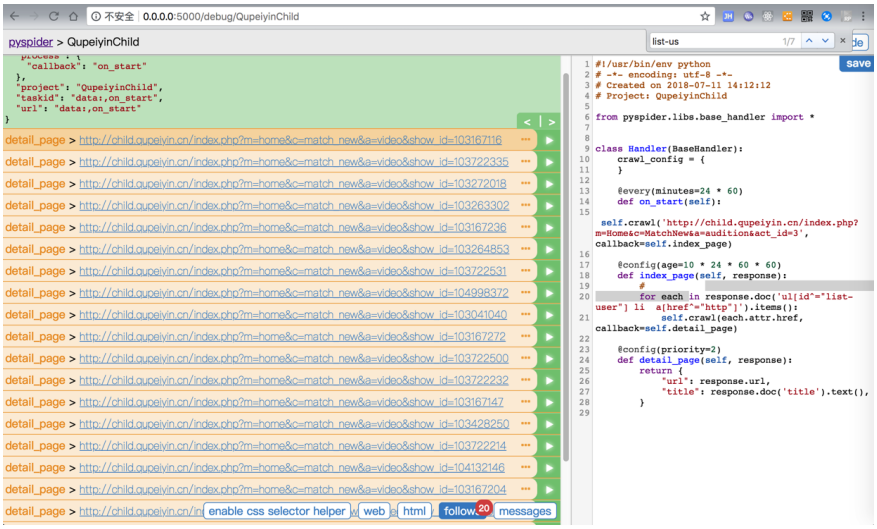
(`PySpider` 中的) `PyQuery` 的写法：

```
response.doc('ul[id^="list-user"] a[href^="http"]')
```

或：

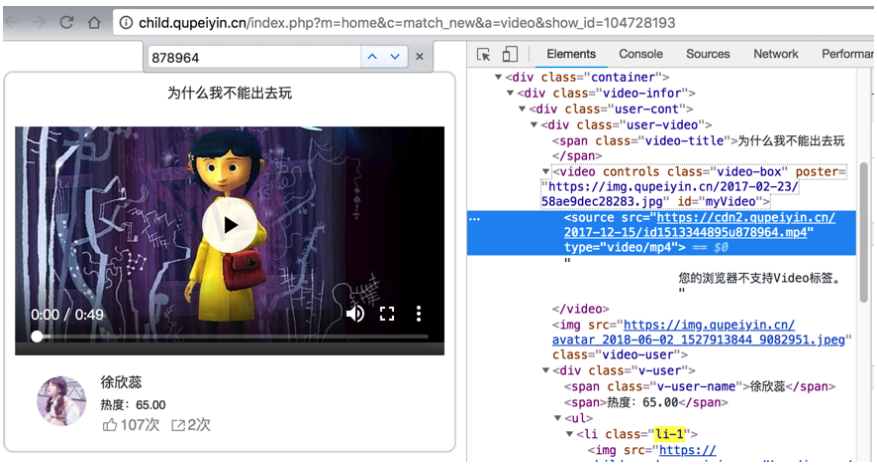
```
response.doc('ul[id^="list-user"] li  a[href^="http"]')
```

（PySpider中的）效果：可以找到元素



例子2

页面：



```
<video controls="" class="video-box" poster="https://img.qu
    <source src="https://cdn2.qupeiyin.cn/2017-12-15/id1513
</video>
```

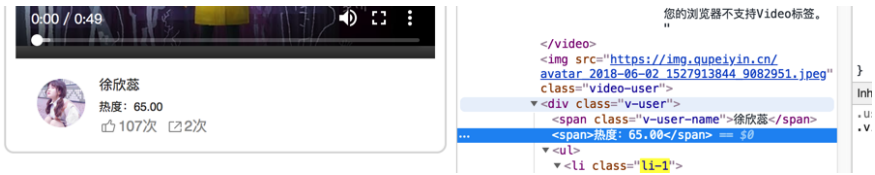
想要：获取src值

代码：

```
# videoUrl = response.doc('video source[src$=".mp4"]')
videoUrl = response.doc('video source[src^="http"]').attr('
```

nth-child 类的元素定位

页面：




```
<div class="v-user">
    <span class="v-user-name">徐欣蕊</span>
    <span>热度：65.00</span>
```

想要定位：`div` 下面第二个 `span`

css选择器的写法：

```
div[class="v-user"] span:nth-child(2)
```

find() vs items()

概述

- `PyQuery.find()`：返回的是 `lxml 的 element 的 generator`
- `PyQuery.items()`：返回的是 `PyQuery 的 generator`

详解

- find
  - 官网文档：pyquery.pyquery.PyQuery.find

    > PyQuery.find(selector)
    >
    > Find elements using selector traversing down from self:
    >
    > ```
    > >> m = ' Whoah!
    > >> d = PyQuery(m)
    > >> d('p').find('em')
    > [, ]
    > >> d('p').eq(1).find('em')
    > []
    > ```

  - -> `find()` 返回的是 `element` 元素
    = `lxml.html.HtmlElement 的 generator`
- items
  - 官网文档：pyquery.pyquery.PyQuery.items

    > PyQuery.items(selector=None)
    >
    > Iter over elements. Return PyQuery objects:
    >
    > ```
    > >> d = PyQuery(' foobar
    > >> [i.text() for i in d.items('span')]
    > ['foo', 'bar']
    > >> [i.text() for i in d('span').items()]
    > ['foo', 'bar']
    > ```

  - -> `items()` 返回的
    是 `PyQuery` = `pyquery.pyquery.PyQuery` 的 `generator`

举例

```
    <ul class="rank-list-ul" 0>

      <li id="s3170">
...
      </li>

      <li id="s692">
...
      </li>
```

find()

如果是 `find()` :

```
carSeriesDocGenerator = merchantRankDoc.find("li[id*='s']"
print("type(carSeriesDocGenerator)=%s" % type(carSeriesDoc
```

则返回的是 `PyQuery`

```
type(carSeriesDocGenerator)=<class 'pyquery.pyquery.PyQuery
```

然后 `generator` 转换成 `list` 后:

```
carSeriesDocList = list(carSeriesDocGenerator)
for curSeriesIdx, eachCarSeriesDoc in enumerate(carSeriesDo
    print("type(eachCarSeriesDoc)=%s" % type(eachCarSeriesD
```

每个元素是: `lxml.html.HtmlElement`

```
type(eachCarSeriesDoc)=<class 'lxml.html.HtmlElement'>
```

items()

如果换成 `items()`

```
carSeriesDocGenerator = merchantRankDoc.items("li[id*='s']"
print("type(carSeriesDocGenerator)=%s" % type(carSeriesDoc
```

则返回的是 `generator` :

```
type(carSeriesDocGenerator)=<class 'generator'>
```

然后 `generator` 转换成 `list` 后:

```
carSeriesDocList = list(carSeriesDocGenerator)
for curSeriesIdx, eachCarSeriesDoc in enumerate(carSeriesDo
    print("type(eachCarSeriesDoc)=%s" % type(eachCarSeriesD
```

每个元素是： `pyquery.pyquery.PyQuery`

```
type(eachCarSeriesDoc)=<class 'pyquery.pyquery.PyQuery'>
```

PyQuery语法

`CSS Selector = CSS选择器`

PyQuery中选择=定位=查找元素的话，主要是用 `CSSS选择器` 。

此处贴出 `CSS选择器` 的常用的基本的**语法**：

选择器	举例	解释
.class	.intro	选择 class="intr 所有元素
#id	#firstname	选择 id="firstna 所有元素
元素相关		
*	*	选择所有元素
element	p	选择所有 <p> 元
element,element	div,p	选择所有 和所有 <p> 元
element element	div p	选择 元 的所有 <p> 元
element>element	div>p	选择父元素为 < 元素的所有 <p>
element+element	div+p	选择紧接在 <di 素之后的所有 < 素
element1~element2	p~ul	选择前面有 <p> 的每个 元
属性相关		
[attribute]	[target]	选择带有 targe 性所有元素
[attribute=value]	[target=_blank]	选择 target="_blan 所有元素
[attribute~=value]	[title~=flower]	选择 title 属 单词 "flower" 有元素
[attribute|=value]	[lang|=en]	选择 lang 属性 "en" 开头的所
[attribute^=value]	a[src^="https"]	选择其 src 属 "https" 开头的 <a> 元素

选择器	举例	解释
`[attribute$=value]`	`a[src$=".pdf"]`	选择其 `src` 属 `".pdf"` 结尾的 `<a>` 元素
`[attribute*=value]`	`a[src*="abc"]`	选择其 `src` 属 含 `"abc"` 子串 `<a>` 元素
元素的修饰？相关		
`:link`	`a:link`	选择所有未被访 接
`:visited`	`a:visited`	选择所有已被访 接
`:active`	`a:active`	选择活动链接
`:hover`	`a:hover`	选择鼠标指针位 的链接
`:focus`	`input:focus`	选择获得焦点的 `input` 元素
`:before`	`p:before`	在每个 `<p>` 元 容之前插入内容
`:after`	`p:after`	在每个 `<p>` 元 容之后插入内容
`:first-letter`	`p:first-letter`	选择每个 `<p>` 首字母
`:first-line`	`p:first-line`	选择每个 `<p>` 首行
`:first-child`	`p:first-child`	选择属于父元素 个子元素的每个 元素
`:first-of-type`	`p:first-of-type`	选择属于其父元 个 `<p>` 元素的 `<p>` 元素
`:last-of-type`	`p:last-of-type`	选择属于其父元 后 `<p>` 元素的 `<p>` 元素
`:only-of-type`	`p:only-of-type`	选择属于其父元 的 `<p>` 元素的 `<p>` 元素

选择器	举例	解释
`:only-child`	`p:only-child`	选择属于其父元素一子元素的每个元素
`:nth-child(n)`	`p:nth-child(2)`	选择属于其父元素二个子元素的每个 `<p>` 元素
`:nth-last-child(n)`	`p:nth-last-child(2)`	同上，从最后一个素开始计数
`:nth-of-type(n)`	`p:nth-of-type(2)`	选择属于其父元素个 `<p>` 元素的 `<p>` 元素
`:nth-last-of-type(n)`	`p:nth-last-of-type(2)`	同上，但是从最后子元素开始计数
`:last-child`	`p:last-child`	选择属于其父元素一个子元素每个元素
`:lang(language)`	`p:lang(it)`	选择带有以 `"it` 头的 `lang` 属性每个 `<p>` 元素
`:root`	`:root`	选择文档的根元素
`:empty`	`p:empty`	选择没有子元素的 `<p>` 元素（包括节点）
`:target`	`#news:target`	选择当前活动的 `#news` 元素
`:enabled`	`input:enabled`	选择每个启用的 `<input>` 元素
`:disabled`	`input:disabled`	选择每个禁用的 `<input>` 元素
`:checked`	`input:checked`	选择每个被选中的 `<input>` 元素
`:not(selector)`	`:not(p)`	选择非 `<p>` 元素个元素
`::selection`	`::selection`	选择被用户选取的部分

PyQuery实例

下面记录一些实际案例，供参考。

汽车之家品牌车型车系

下面贴出，用PySpider爬取汽车之家的车型车系数据的代码，其中包含 `PyQuery` 相关代码，供参考。

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Created on 2018-04-27 21:53:02
# Project: autohomeBrandData

from pyspider.libs.base_handler import *
import string
import re

class Handler(BaseHandler):
    crawl_config = {
    }

    # @every(minutes=24 * 60)
    def on_start(self):
        for eachLetter in list(string.ascii_lowercase):
            self.crawl("https://www.autohome.com.cn/grade/

    @catch_status_code_error
    def gradCarHtmlPage(self, response):
        print("gradCarHtmlPage: response=", response)
        picSeriesItemList = response.doc('.rank-list-ul li
        print("picSeriesItemList=", picSeriesItemList)
        # print("len(picSeriesItemList)=%s"%(len(picSeries
        for each in picSeriesItemList:
            self.crawl(each.attr.href, callback=self.picSe

    @config(priority=2)
    def picSeriesPage(self, response):
        # &lt;a href="/pic/series-t/66.html"&gt;查看停产车型&
        # &lt;a class="ckmore" href="/pic/series/588.html"&
        # &lt;span class="fn-right"&gt;&amp;nbsp;&lt;/span&
        fnRightPicSeries = response.doc('.search-pic-tbar
        print("fnRightPicSeries=", fnRightPicSeries)
        if fnRightPicSeries:
            # hrefValue = fnRightPicSeries.attr.href
            # print("hrefValue=", hrefValue)
            # fullPicSeriesUrl = "https://car.autohome.com.
            fullPicSeriesUrl = fnRightPicSeries.attr.href
            print("fullPicSeriesUrl=", fullPicSeriesUrl)
            self.crawl(fullPicSeriesUrl, callback=self.picS

        # contine parse brand data
        aDictList = []
        # for eachA in response.doc('.breadnav a[href^="/"]
        for eachA in response.doc('.breadnav a[href*="/pic/
            eachADict = {
                "text" : eachA.text(),
                "href": eachA.attr.href
            }
```

```
            print("eachADict=", eachADict)
            aDictList.append(eachADict)

        print("aDictList=", aDictList)

        mainBrandDict = aDictList[-3]
        subBrandDict = aDictList[-2]
        brandSerieDict = aDictList[-1]
        print("mainBrandDict=%s, subBrandDict=%s, brandSer

        dtTextList = []
        for eachDt in response.doc("dl.search-pic-cardl dt"
            dtTextList.append(eachDt.text())

        print("dtTextList=", dtTextList)

        groupCount = len(dtTextList)
        print("groupCount=", groupCount)

        for eachDt in response.doc("dl.search-pic-cardl dt"
            dtTextList.append(eachDt.text())

        ddUlEltList = []
        for eachDdUlElt in response.doc("dl.search-pic-car
            ddUlEltList.append(eachDdUlElt)

        print("ddUlEltList=", ddUlEltList)

        modelDetailDictList = []

        for curIdx in range(groupCount):
            curGroupTitle = dtTextList[curIdx]
            print("------[%d] %s" % (curIdx, curGroupTitle

            for eachLiAElt in ddUlEltList[curIdx].items("l
                # 1. model name
                # curModelName = eachLiAElt.text()
                curModelName = eachLiAElt.contents()[0]
                curModelName = curModelName.strip()
                print("curModelName=", curModelName)
                curFullModelName = curGroupTitle + " " + cu
                print("curFullModelName=", curFullModelNam

                # 2. model id + carSeriesId + spec url
                curModelId = ""
                curSeriesId = ""
                curModelSpecUrl = ""
                modelSpecUrlTemplate = "https://www.autoho
                curModelPicUrl = eachLiAElt.attr.href
                print("curModelPicUrl=", curModelPicUrl)
                #https://car.autohome.com.cn/pic/series-s3
```

```
                foundModelSeriesId = re.search("pic/series-
                print("foundModelSeriesId=", foundModelSeri
                if foundModelSeriesId:
                    curModelId = foundModelSeriesId.group('
                    curSeriesId = foundModelSeriesId.group(
                    print("curModelId=%s, curSeriesId=%s",
                    curModelSpecUrl = (modelSpecUrlTemplate
                    print("curModelSpecUrl=", curModelSpecU

                # 3. model status
                modelStatus = "在售"
                foundStopSale = eachLiAElt.find('i[class*="
                if foundStopSale:
                    modelStatus = "停售"
                else:
                    foundWseason = eachLiAElt.find('i[class
                    if foundWseason:
                        modelStatus = "未上市"

                modelDetailDictList.append({
                    "url": curModelSpecUrl,
                    "车系ID": curSeriesId,
                    "车型ID": curModelId,
                    "车型": curFullModelName,
                    "状态": modelStatus
                })
            print("modelDetailDictList=", modelDetailDictList)

            allSerieDictList = []
            for curIdx, eachModelDetailDict in enumerate(modelD
                curSerieDict = {
                    "品牌": mainBrandDict["text"],
                    "子品牌": subBrandDict["text"],
                    "车系": brandSerieDict["text"],
                    "车系ID": eachModelDetailDict["车系ID"],
                    "车型": eachModelDetailDict["车型"],
                    "车型ID": eachModelDetailDict["车型ID"],
                    "状态": eachModelDetailDict["状态"]
                }
                allSerieDictList.append(curSerieDict)
                # print("before send_message: [%d] curSerieDict
                # self.send_message(self.project_name, curSerie
                print("[%d] curSerieDict=%s" % (curIdx, curSeri
                self.crawl(eachModelDetailDict["url"], callbac

            # print("allSerieDictList=", allSerieDictList)
            # return allSerieDictList

    #def on_message(self, project, msg):
    #    print("on_message: msg=", msg)
    #    return msg
```

```
    @catch_status_code_error
    def carModelSpecPage(self, response):
        print("carModelSpecPage: response=", response)
        # https://www.autohome.com.cn/spec/32708/#pvareaid=
        curSerieDict = response.save
        print("curSerieDict", curSerieDict)

        # cityDealerPriceInt = 0
        # cityDealerPriceElt = response.doc('.cardetail-int
        # print("cityDealerPriceElt=%s" % cityDealerPriceE
        # if cityDealerPriceElt:
        #     cityDealerPriceFloatStr = cityDealerPriceElt.
        #     print("cityDealerPriceFloatStr=", cityDealerP
        #     cityDealerPriceFloat = float(cityDealerPriceF
        #     print("cityDealerPriceFloat=", cityDealerPric
        #     cityDealerPriceInt = int(cityDealerPriceFloat
        #     print("cityDealerPriceInt=", cityDealerPrice

        msrpPriceInt = 0
        # body &gt; div.content &gt; div.row &gt; div.colum
        # 厂商指导价=厂商建议零售价格=MSRP=Manufacturer's sugge
        msrpPriceElt = response.doc('.cardetail-infor-price
        print("msrpPriceElt=", msrpPriceElt)
        if msrpPriceElt:
            msrpPriceStr = msrpPriceElt.attr("data-price")
            print("msrpPriceStr=", msrpPriceStr)
            foundMsrpPrice = re.search("(?P&lt;msrpPrice&gt
            print("foundMsrpPrice=", foundMsrpPrice)
            if foundMsrpPrice:
                msrpPrice = foundMsrpPrice.group("msrpPrice
                print("msrpPrice=", msrpPrice)
                msrpPriceFloat = float(msrpPrice)
                print("msrpPriceFloat=", msrpPriceFloat)
                msrpPriceInt = int(msrpPriceFloat * 10000)
                print("msrpPriceInt=", msrpPriceInt)

        # curSerieDict["经销商参考价"] = cityDealerPriceInt
        curSerieDict["厂商指导价"] = msrpPriceInt

        return curSerieDict
```

详见：

【已解决】写Python爬虫爬取汽车之家品牌车系车型数据

附录

下面列出相关参考资料。

参考资料

- 【已解决】PySpider中获取PyQuery获取到节点的子元素
- 【已解决】写Python爬虫爬取汽车之家品牌车系车型数据
- 【已解决】PySpider中PyQuery选择div后面的第二个span元素
-
- CSS Selectors Reference
- CSS selectors - CSS: Cascading Style Sheets | MDN
- CSS 选择器参考手册
-