目录

Python爬虫框架：PySpider

- 最新版本：`v1.4`
- 更新时间：`20210414`

简介

PySpider是一个简单易用且强大的Python主流爬虫框架。此处总结PySpider的安装和基本的使用，以及安装和启动时常见问题，并且给出查找定位元素的PyQuery的基本用法举例，以及一些高级用法，比如self.craw、config.json、data目录、phantomjs，和一些心得和常见的坑，并且给出一些实际的例子供参考。

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_spider_pyspider: Python爬虫框架：PySpider

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

- Python爬虫框架：PySpider book.crifan.com
- Python爬虫框架：PySpider crifan.github.io

离线下载阅读

- Python爬虫框架：PySpider PDF
- Python爬虫框架：PySpider ePub
- Python爬虫框架：PySpider 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

PySpider简介

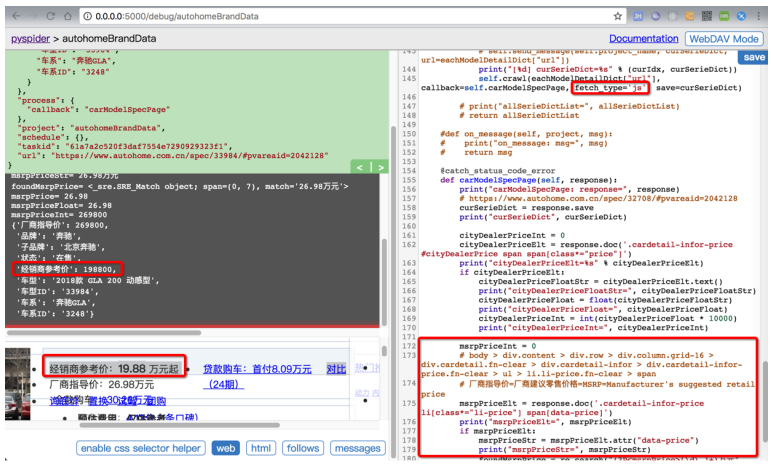
`PySpider` 的基本信息：

- 是个Python的爬虫框架
- 最大特点：
  - 带图形界面WebUI的调试
  - 简单易用
  - 同时功能也很强大
- GitHub
  - https://github.com/binux/pyspider
- 文档：
  - 官方，英文：http://docs.pyspider.org/
  - 非官方，中文：http://www.pyspider.cn/index.html
- 作者
  - 网名：`Binux`
  - 别名：足叉兆虫
  - 博客：Binuxの杂货铺
  - Github: binux (Roy Binux)

`PySpider` 对比 `Scrapy`

对于两个流行的Python的爬虫框架，PySpider和Scrapy，常常会被人拿来对比。

对此，之前简单总结如下：

- PySpider：简单易上手，带图形界面（基于浏览器页面）
  - 一图胜千言：在WebUI中调试爬虫代码
  - 

- Scrapy：可以高级定制化实现更加复杂的控制
  - 一图胜千言：Scrapy一般是在命令行界面中调试页面返回数据：

- 

详见： 【整理】pyspider vs scrapy

PySpider安装与基本用法

概述：

- 安装

```
pip install pyspider
```

- 启动

```
pyspider
```

PySpider安装

下面介绍，如何安装 `PySpider` 。

Mac中安装PySpider

此处以Mac中为例，去介绍如何安装PySpider。

如果只是简单的直接安装的话，则可以去：

```
pip install pyspider
```

安装phantomjs

如果后续用到网页中要允许js的代码，则需要用到无头浏览器 `phantomjs` ，就需要先去安装 `phantomjs`

```
brew cask install phantomjs
```

安装和启动的常见问题

此处整理在安装和启动 `pyspider` 期间，经常遇到的问题和其原因及解决办法。

启动报错：async=True SyntaxError: invalid syntax

- 运行 `pyspider` 报错：

```
  File "/Users/limao/.pyenv/versions/3.8.0/Python.frame
    async=True, get_object=False, no_input=False):
    ^
SyntaxError: invalid syntax
```

- **原因**：PySpider（很久没继续维护了）最新支持版本是 `Python 3.6` ，其把 `async` 作为普通函数参数，是没问题的。
  - 但是 `Python 3.7` 之后把 `async` 改为了**系统保留字**，表示 `异步` ，所以 `asyc` 不能再作为普通函数参数名
    - 所以当前环境是 `Python 3.7+` 时，就会报语法错误了
- **解决办法**：2种思路：
  - （自己）把代码（中这类的语法错误）都改掉
  - 把当前Python版本换成旧版本： `Python 3.6`
    - 此处选择换旧版本 `Python 3.6`
- **步骤**：

Mac中换Python为3.6

此处以 `pyenv` 为例：

先去安装Python 3.6的某个版本：

```
pyenv install 3.6.8
```

再去设置使用 `Python 3.6` ：

- 本地

```
pyenv local 3.6.8
```

- 或全局

```
pyenv global 3.6.8
```

然后重新安装：

```
pip install pyspider
```

之后即可正常运行

```
pyspider
```

详见：

【已解决】Mac中给Python3安装PySpider

推荐用虚拟环境

在Python的开发中，为了避免不同开发环境的互相影响，一般都会用虚拟环境工具，比如 `pipenv` 、 `virtualenv` 。

注意：此处即使用Python虚拟环境，由于前面提到的问题，PySpider也还是要用 `Python 3.6` 的

pipenv中安装PySpider

此处是 `pipenv` ，为了指定使用 `Python 3.6` 则，可以在 `Pipfile` 中加上：

```
[requires]
python_version = "3.6"
```

贴出完整配置：

```
[[source]]
#url = "https://pypi.python.org/simple"
url = "https://pypi.tuna.tsinghua.edu.cn/simple"
verify_ssl = true
name = "pypi"

[packages]
pymysql = "*"

[dev-packages]

[requires]
python_version = "3.6"
```

创建并安装PySpider：

```
pipenv install pyspider
```

或：

先创建虚拟环境，再安装PySpider

```
pipenv install
pipenv shell
pip install pyspider
```

virtualenv中安装PySpider

关于virtualenv：

- 创建虚拟环境：`virtualenv venv`
- 激活激活环境并进入：
  - Mac/Linux：`source venv/bin/activate`
    - 或：
      - `. venv/bin/activate`
    - Win：`venv\Scripts\activate.bat`
- 退出虚拟环境：`deactivate`

在`virtualenv`中安装PySpider：

```
pip install pyspider
```

启动报错：pycurl报ImportError错 或 Curl报ConfigurationError错

`Mac`中，如果安装期间出错：

> __main__.ConfigurationError: Curl is configured to use SSL, but we have not been able to determine which SSL backend it is using

或运行时报错：

```
ImportError: pycurl: libcurl link-time ssl backend (openss
```

- **原因**：此处导入pycurl时，发现libcurl运行时所依赖的ssl的底层是openssl，和当时编译时的版本不匹配
- **解决办法**：重新编译安装，使得版本一致
- **步骤**：

```
pip uninstall -y pycurl
export PYCURL_SSL_LIBRARY=openssl
export LDFLAGS=-L/usr/local/opt/openssl/lib;export CPPFLAGS
```

注意：上述的：

- `/usr/local/opt/openssl` 是你的openssl安装路径
  - 如果你的不是这个路径， 要换成你的实际路径
  - 对应的
    - `/usr/local/opt/openssl/lib` 是 `lib` 库的路径
    - `/usr/local/opt/openssl/include` 是 `include` 头文件的路径

详见：

- 【记录】Mac中安装和运行pyspider
- 【已解决】pipenv虚拟环境中用pip安装pyspider出错：、__main__.ConfigurationError: Curl is configured to use SSL, but we have not been able to determine which SSL backend it is using
- 【已解决】pyspider运行出错：ImportError pycurl libcurl link-time ssl backend (openssl) is different from compile-time ssl backend (none/other)

启动报错：fatal error openssl/ssl.h file not found

如果上面步骤：

`export LDFLAGS=-L/usr/local/opt/openssl/lib;export CPPFLAGS=-I/usr/local/opt/openssl/include;pip install pycurl --compile --no-cache-dir`

报错：

```
    In file included from src/docstrings.c:4:
    src/pycurl.h:165:13: fatal error: 'openssl/ssl.h' file
    #   include <openssl/ssl.h>
    1 error generated.
    error: command 'gcc' failed with exit status 1
```

- **直接原因**：找不到 `openssl/ssl.h`
- **多种可能**
  - 之前没安装过 `openssl`
    - **解决办法**：Mac中去安装：

      ```
      brew install openssl
      ```

        - 然后再重试即可
    - Mac中已安装过 `openssl`
        - 所以此处Mac中是有 `openssl/ssl.h` 的，只是传入的路径不对
        - **解决办法**：找到已安装的 `openssl` 的实际路径，传入正确的路径。
        - **步骤**：

找到已安装的openssl的实际安装路径

```
brew info openssl
```

可以看到有：

`/usr/local/Cellar/openssl@1.1/1.1.1d (7,983 files, 17.9MB)`

其中的：

`/usr/local/Cellar/openssl@1.1/1.1.1d`

就是我们要的，此处openssl的实际安装路径

通过

```
open /usr/local/Cellar/openssl@1.1/1.1.1d
```

确认是有对应的：

`/usr/local/Cellar/openssl@1.1/1.1.1d/include/openssl/ssl.h`

这个文件的。所以传入路径应该改为：

`/usr/local/Cellar/openssl@1.1/1.1.1d/include`

完整命令是：

```
export LDFLAGS=-L/usr/local/opt/openssl/lib;export CPPFLAGS
```

详见：

【已解决】Mac中pip安装pycurl报错：fatal error openssl/ssl.h file not found

启动报错：Error Could not create web server listening on port 25555

- **现象**：运行 `pyspider` 时能看到有错误
    - `Error: Could not create web server listening on port 25555`

- **原因**：之前已启动过 `pyspider`，其内部会默认启动 `phantomjs`，而虽然之前虽然已关闭掉 `pyspider`，但是没有杀掉 `phantomjs` 的进程，导致端口 `25555` 被占用，而报错
- **解决办法**：杀掉端口是 `25555` 的 `phantomjs` 进程即可
- **步骤**：
  - 找 `phantomjs` 进程ID：
    - `ps aux | grep 25555`
  - 杀掉对应进程
    - `kill -9 xxx`
- **举例**

```
✘ limao@xxx ☐ ~/dev/crifan/python/demo_spider ☐ ps aux | g
limao            35620   0.0  0.0  4277272    820 s002  R+
limao            33983   0.0  0.4  6130968  34128 s002  S
 limao@xxx ☐ ~/dev/crifan/python/demo_spider ☐ kill -9 339
```

之后即可正常启动 `pyspider`，且能看到 `phantomjs` 可以正常启动了：

```
phantomjs fetcher running on port 25555
```

启动报错：Deprecated option domaincontroller use http_authenticator.domain_controller instead

启动报错：

```
  File "/Users/limao/.pyenv/versions/3.6.8/lib/python3.6/s
    raise ValueError("Invalid configuration:\n  - " + "\n
ValueError: Invalid configuration:
  - Deprecated option 'domaincontroller': use 'http_authen
```

- **原因**：`wsgidav` 版本兼容问题
- **解决办法**：换兼容的没问题的旧版本 `2.4.1`
- **步骤**：

```
pip install wsgidav==2.4.1
```

启动报错：ImportError cannot import name DispatcherMiddleware

启动报错：

```
  File "/Users/limao/.pyenv/versions/3.6.8/lib/python3.6/s
    from werkzeug.wsgi import DispatcherMiddleware
ImportError: cannot import name 'DispatcherMiddleware'
```

- **原因**：`werkzeug` 版本兼容问题
- **解决办法**：换兼容的没问题的旧版本 `0.16.1`
- **步骤**：

```
pip install werkzeug==0.16.1
```

PySpider基本用法

使用PySpider的基本步骤

下面来介绍一下PySpider的使用的步骤和操作：

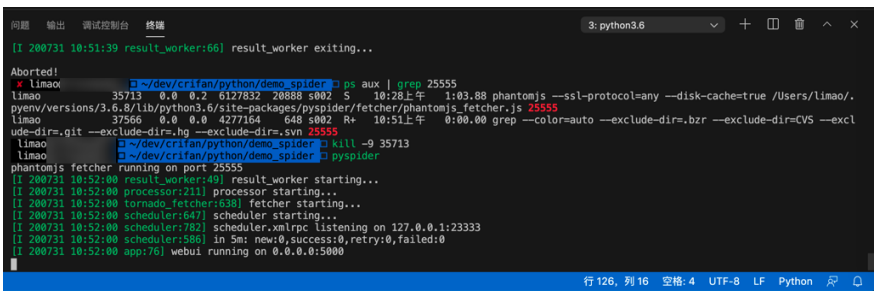
运行PySpider

在某个目录下的终端命令行中输入

```
pyspider
```

即可启动运行，输出举例：

```
□ pyspider
phantomjs fetcher running on port 25555
[I 200731 10:28:35 result_worker:49] result_worker starting
[I 200731 10:28:35 processor:211] processor starting...
[I 200731 10:28:35 tornado_fetcher:638] fetcher starting...
[I 200731 10:28:35 scheduler:647] scheduler starting...
[I 200731 10:28:35 scheduler:782] scheduler.xmlrpc listenir
[I 200731 10:28:35 scheduler:586] in 5m: new:0,success:0,re
[I 200731 10:28:35 app:84] webui exiting...
```



注：

- 如果是用虚拟环境安装的PySpider，记得先进入虚拟环境后再运行PySpider
  - 比如用的 `pipenv` ，则是

    ```
    pipenv shell
    pyspider
    ```

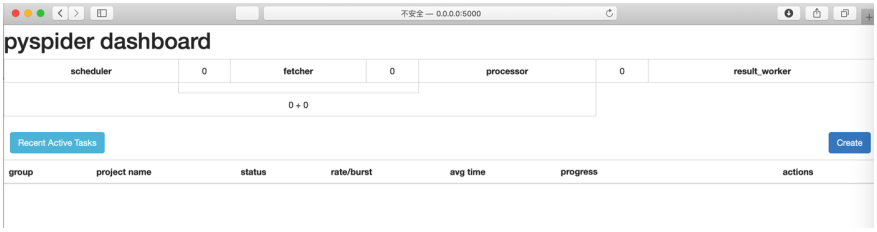
- `pyspider` 等价于 `pyspider all`

进入 `WebUI`

然后去用浏览器打开：

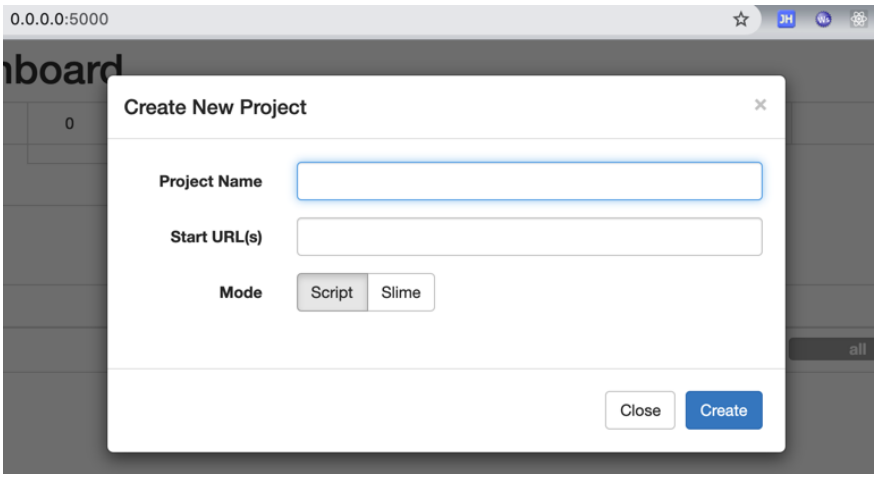
http://0.0.0.0:5000/

即可进入爬虫的 管理界面 = WebUI



新建爬虫项目

点击 Create ，去新建一个爬虫项目



输入：

- 爬虫名称：
- 入口地址：自动生成的代码中，会作为起始要抓取的url
  - 也可以不填
    - 后续可以在代码中修改

然后再点击新建的爬虫项目，进入调试页面

新建出来的项目，默认状态是 TODO

点击新建出来的项目名，直接进入调试界面

然后右边是编写代码的区域

左边是调试的区域，用于执行代码，显示输出信息等用途

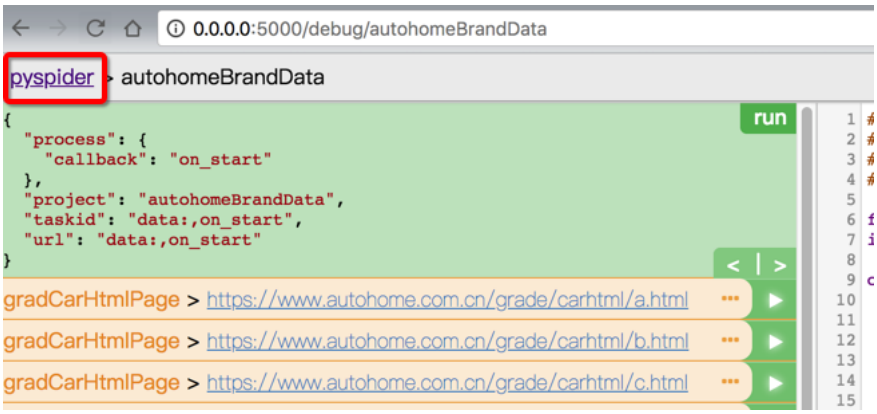
调试爬虫代码

编写代码，调试输出信息，保存代码

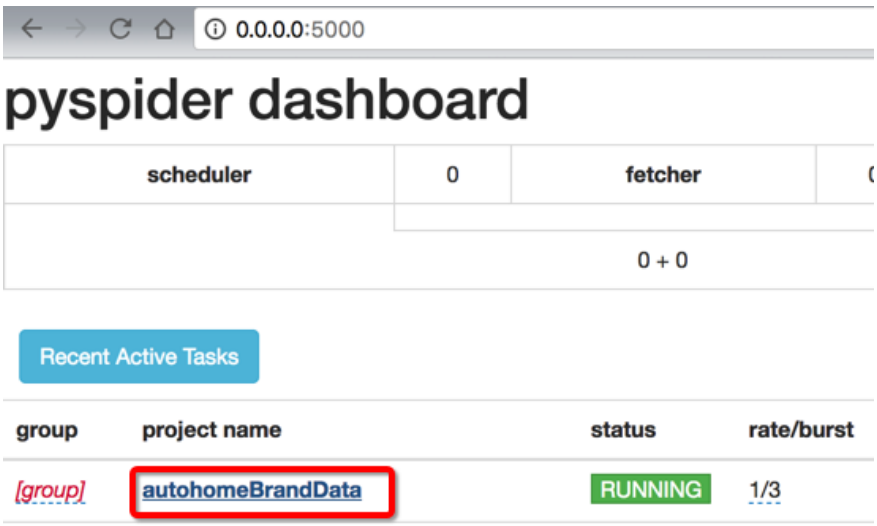
调试代码期间，对于想要返回上一级：

先说之前不熟悉的时候的操作：

之前调试运行时，不知道还有回到上一级，在想要返回上一级时，都直接是点击左上角的项目名字



返回项目列表：



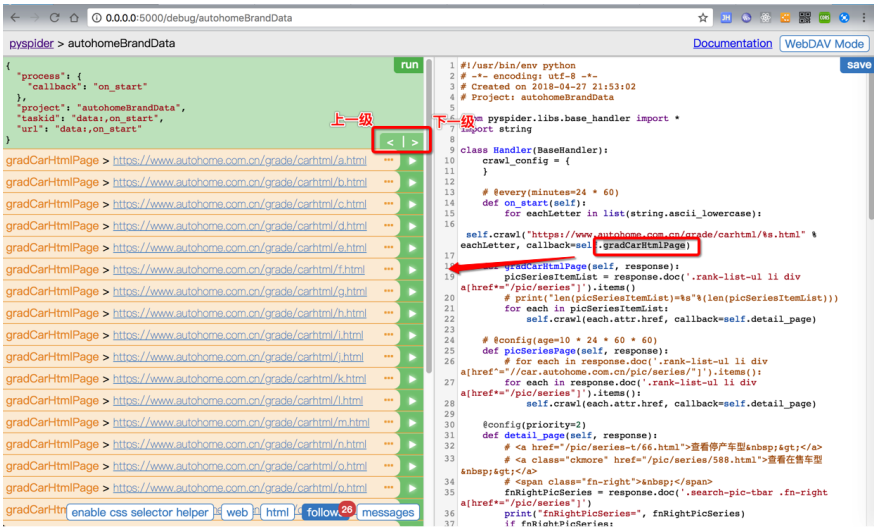
然后重新进去，重新点击Run，直到跑到对应的层级，去继续调试。

再说后来知道了PySpider内置支持这种逻辑操作：

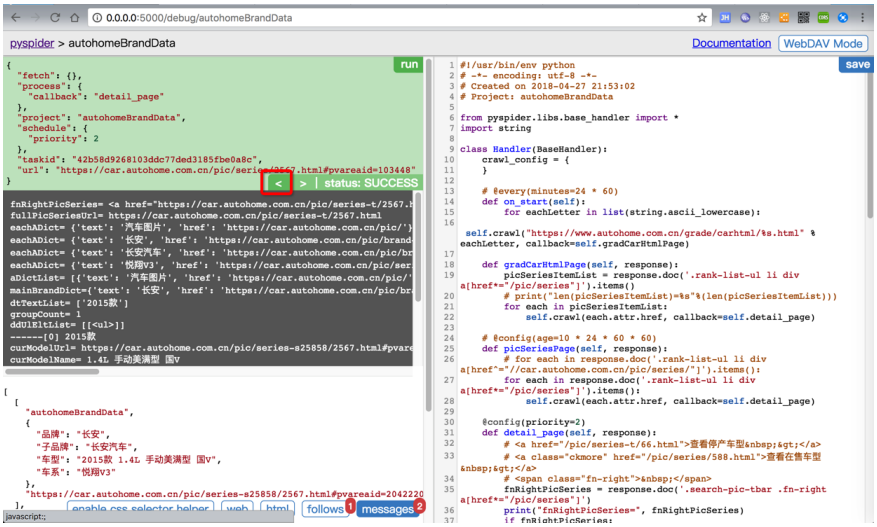
PySpider对在调试期间所需要在上一个连接和下一个连接之间切换的操作，支持的很好：

点击 `< | >` 的 `<` 或 `>`，则可以 `返回上一级` 或 `进入下一级`

实际效果演示：



想要返回上一级的爬取函数的话，点击 左箭头

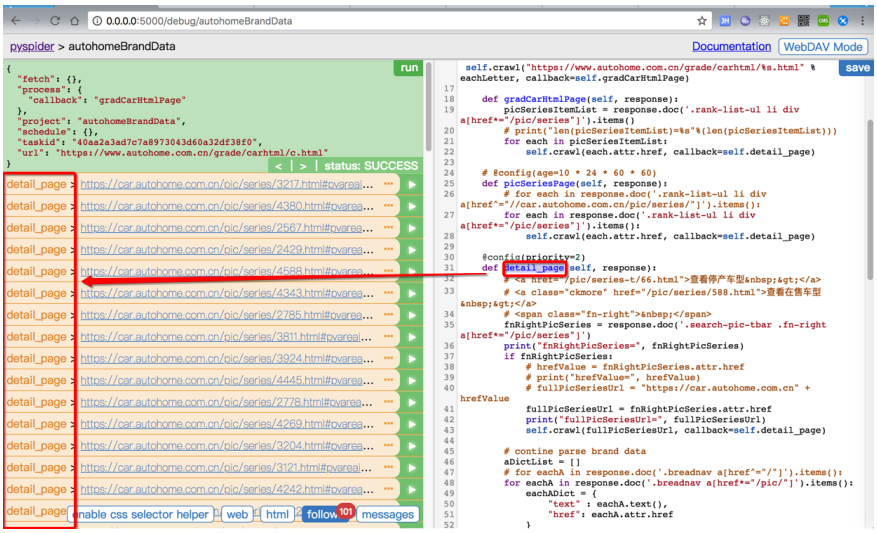


然后再点击Run：

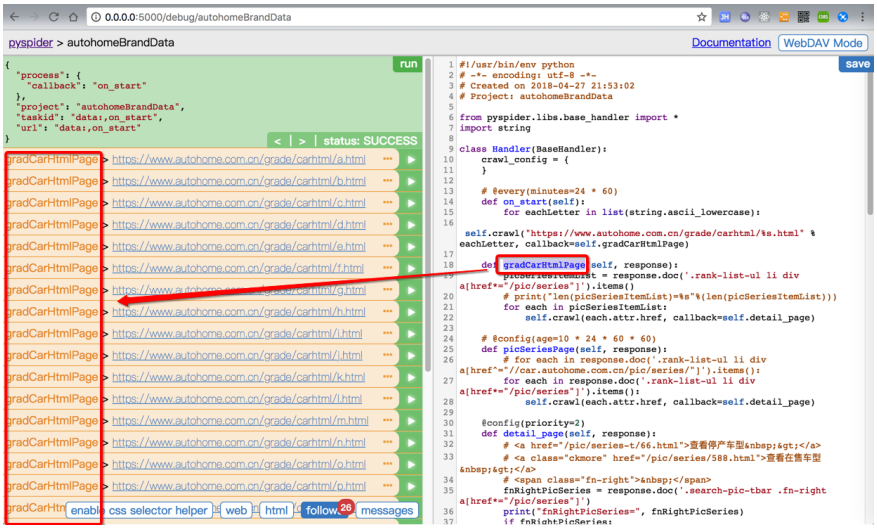
然后就可以返回上一级了。

然后也才注意到，每行的follow的左边开始显示的是：callback函数名

此处的是 `detail_page`



而对应的上一级的结果中，也是上一级的callback：



运行爬虫去爬取数据

调试完毕后，返回项目，status改为DEBUG或RUNNING，点击Run

想要暂停运行：status改为STOP

保存已爬取的数据

当爬取完毕数据，需要保存下来时，可以有多种保存方式：

- mysql数据库
- MongoDB数据库
- CSV或Excel文件

保存到csv或Excel文件

基本思路：确保自己代码中，最后return返回的字段是你要的字段

如何得到CSV文件：在任务运行期间或完毕后，去 `Results` -》点击下载 `CSV` ，即可得到你要的csv格式的数据文件。

结果：PySpider会自动在已有字段中加上额外的 `url` 字段

用VSCode编辑csv文件

- 如果想要去除多余的不需要的 `url` 字段，则可以通过文本编辑器，比如 `VSCode` 去列编辑模式，批量删除，或者查找和替换，都可以实现
- 最后会多余一列，标题是 ...，内容全是 `,{}` ，所以直接用编辑器比如VCScode去替换为空以清空，即可

详见：

【已解决】PySpider如何把json结果数据保存到csv或excel文件中 – 在路上

Excel去打开CSV文件结果乱码

csv文件编码默认为UTF8（是好事，通用的），但是如果用（不论是Mac还是Win中的）excel去打开，结果（估计对于中文系统，都是）会默认以GKB（或GB18030）打开，所以会乱码

解决办法：【已解决】Mac或Win中用Excel打开UTF8编码的csv文件显示乱码

PySpider中查找提取元素

PySpider中内置的用于查找和定位html网页中元素的库是：`PyQuery`

`PyQuery` 算是一个 `css选择器`，模拟 `JS` 领域的 `jQuery`，所以叫做 `PyQuery` 。

具体细节是，PySpider针对html的响应 `response`，默认提供了一个 `doc` 属性，其内置了 `PyQuery` 解析后结果，所以你可以用 `response.doc("your_css_selector")` 去选择你要的html中的内容了。

而具体的your_css_selector的写法，则就变成 `PyQuery` 的写法了。

举例：提取汽车之家车型车系相关数据

下面通过想象的例子来解释， `PyQuery` 的常见用法：

比如想要提取：

```
<ul class="rank-list-ul" 0="">
  <li id="s3170">
...
    <div>
      ...
      <a id="atk_3170" href="//car.autohome.com.cn/pic/seri
    ...
    </div>
  </li>
...
</ul>
```

中的 `href` 的值

- 语言描述可以是：`class` 是 `rank-list-ul` 的 `ul` 元素，下面的 `li`，下面的 `div`，下面的 `a`，且 `href` 值是包含 `/pic/series` 的
- 转换成PyQuery的语法是
    - `.rank-list-ul li div a[href*="/pic/series"]`
        - 也可以换成另外的写法：
            - `ul[class='rank-list-ul'] li div a[href*="/pic/series"]`
            - `ul[class='rank-list-ul'] a[href*="/pic/series"]`

        - 如果你确定此规则不会误匹配其他元素，也可以省略中间的节点的查找
- 对应PySpider的代码是：
    - 获取匹配到的第一个元素

```
firstMatchADoc = response.doc('.rank-list-ul li d
```

    - 获取到所有匹配到的元素

```
for eachADoc in response.doc('.rank-list-ul li di
  print("eachADoc=%s" % eachADoc)
```

以及对于：

```
<dl id="33" olr="6">
  <dt><a href="//car.autohome.com.cn/price/brand-33.html#pv
        src="//car2.autoimg.cn/cardfs/series/g26/M0B/AE/B3/
    <div><a href="//car.autohome.com.cn/price/brand-33.htm
  </dt>
  <dd>

    <div class="h3-tit"><a href="//car.autohome.com.cn/pric
    <ul class="rank-list-ul" 0>

      <li id="s3170">
        <h4><a href="//www.autohome.com.cn/3170/#levelsourc
        <div>指导价：<a class="red" href="//www.autohome.com
        <div><a href="//car.autohome.com.cn/price/series-31
            href="//car.autohome.com.cn/pic/series/3170.htm
            class="js-che168link" href="//www.che168.com/ch
            href="//club.autohome.com.cn/bbs/forum-c-3170-1
            href="//k.autohome.com.cn/3170/#pvareaid=103459

      </li>

      <li id="s692">
        <h4><a href="//www.autohome.com.cn/692/#levelsource
        <div>指导价：<a class="red" href="//www.autohome.com
        <div><a href="//car.autohome.com.cn/price/series-69
            href="//car.autohome.com.cn/pic/series/692.htm
            class="js-che168link" href="//www.che168.com/ch
            href="//club.autohome.com.cn/bbs/forum-c-692-1.
            href="//k.autohome.com.cn/692/#pvareaid=103459

      </li>
...
...
...
```

对应的代码是：

想要从

```
<a href="//car.autohome.com.cn/price/brand-33.html#pvareaid
```

获取brand的logo的 `img` 的代码：

```
brandDoc = response.doc('dl dt')
brandLogoDoc = brandDoc.find('a img')
brandLogoUrl = brandLogoDoc.attr["src"]
```

从：

```
<div><a href="//car.autohome.com.cn/price/brand-33.html#pva
```

中获取brand的name的 `a` 的代码：

```
brandNameDoc = brandDoc.find('div a')
brandName = brandNameDoc.text()
```

从：

```
<div class="h3-tit"><a href="//car.autohome.com.cn/price/br
```

获取merchant的所有的 `a` 的代码：

```
merchantDocGenerator = response.doc("dd div[class='h3-tit']
merchantDocList = list(merchantDocGenerator)
merchantDocLen = len(merchantDocList)
```

注意： `.items()` 返回的是 `generator` ，想要得到 `list` ，需要用 `list(yourGenerator)` 去转换得到

从：

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

获取 `rank-list-ul` 的 `class` 的 `dd` 下面的 `ul` 的merchant的代码：

```
merchantRankDocGenerator = response.doc("dd ul[class='rank-
merchantRankDocList = list(merchantRankDocGenerator)
merchantRankDocListLen = len(merchantRankDocList)
```

以及获取每个元素：

- 属性值：用 `attr`
  - 类型是： `dict`
- 字符串值：用 `text()`
  - 类型是： `str`

举例：

```
for curIdx, merchantItem  in enumerate(merchantDocList):
    merchantName = merchantItem.text()
    merchantItemAttr = merchantItem.attr
    merchantUrl = merchantItemAttr["href"]
```

PyQuery资料

`response.doc` 返回后的 `PyQuery对象` ，之后可以继续用PyQuery去操作

此处列出PyQuery的一些典型的操作函数：

- `PyQuery.filter(selector)`
- `PyQuery.find(selector)`
- `PyQuery.items(selector=None)`
- `PyQuery.siblings(selector=None)`

另外，常见的一些属性来说：

- `PyQuery.text(value=<NoDefault>)` ：当前节点的text文本值
- `PyQuery.html(value=<NoDefault>, **kwargs)` ：当前节点的html值

详见：

- 官网文档
  - pyquery – PyQuery complete API — pyquery 1.2.4 documentation
  - Traversing — pyquery 1.2.4 documentation
  - Attributes — pyquery 1.2.4 documentation
  - CSS — pyquery 1.2.4 documentation
- 独立教程
  - HTML解析库Python版jQuery：PyQuery

PySpider的高级用法

下面介绍PySpider中，除了基本用法之外的，可以算作是高级，稍微更加复杂一些的用法。

self.crawl详解

官方文档

此处要介绍的 `PySpider` 的获取网络请求的函数是： `self.crawl` ，功能很强大。

具体详细的解释，可以参考：

- 官网的英文文档：
  - 比如： crawl参数 params
- 某热心网友整理的 中文文档：
  - self.crawl - pyspider中文文档 - pyspider中文网

给 `GET` 的请求添加 `查询参数`

给 `self.crawl` 中给 `params` 传递对应字典变量， `PySpider` 内部会自动把字典编码为url的查询参数 query string.

官方实例：

```
self.crawl('http://httpbin.org/get', callback=self.callbacl
```

等价于：

```
self.crawl('http://httpbin.org/get?a=123&b=c', callback=se
```

自己之前用的例子有：

```
topSignTopParam = {
    "start": 0,
    "rows": 20
}
self.crawl(TopSignTopUrl,
    callback=self.getMoreUserCallback,
    params=topSignTopParam,
    save={
        "baseUrl": TopSignTopUrl,
        "isNeedCheckNextPage": True,
        "curPageParam": topSignTopParam
    }
)
```

给callback函数加上额外的参数

使用 `self.crawl` 的 `save` 参数即可，然后callback中用 `response.save` 获取传入的值

举例：

```
    def getUserDetail(self, userId):
        self.crawl(UserDetailUrl,
            callback=self.userDetailCallback,
            params={"member_id": userId },
            save=userId
        )

    def userDetailCallback(self, response):
        userId = response.save
        print("userId=%s" % userId)
```

当请求出错时也执行callback回调函数

需要给callback回调函数加上修饰符 `@catch_status_code_error`

举例：

```
    def picSeriesPage(self, response):
      ...
      self.crawl(curSerieDict["url"], callback=self.carMode

    @catch_status_code_error
    def carModelSpecPage(self, response):
        curSerieDict = response.save
        print("curSerieDict=%s", curSerieDict)
        ...
        return curSerieDict
```

独立的配置文件: `config.json`

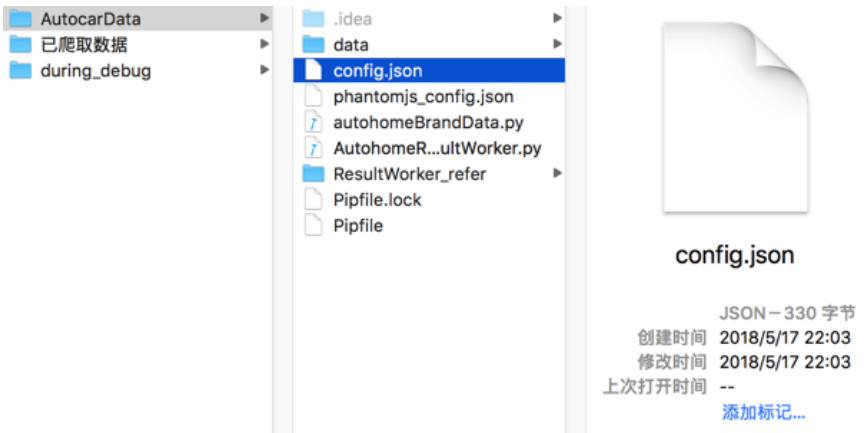
如果需要用到复杂一点的配置，比如 `result worker`，则是需要单独写配置文件。

PySpider的配置文件一般叫做 `config.json`

比如用如下内容:

```
{
  "taskdb":     "mysql+taskdb://root:crifan_mysql@127.0.0.1
  "projectdb":  "mysql+projectdb://root:crifan_mysql@127.0.
  "resultdb":   "mysql+resultdb://root:crifan_mysql@127.0.0
  "result_worker":{
      "result_cls": "AutohomeResultWorker.AutohomeResultWor
   }
}
```

将 `config.json` 保存在 `pyspider` 命令运行所在的当前目录下:



然后去 `-c` 指定配置文件:

```
pyspider -c config.json
```

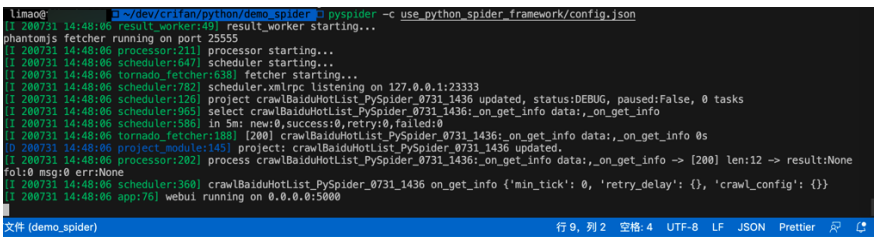
举例

一个配置更简单的 `config.json`

```
{
    "webui": {
        "port": 5000,
        "need-auth": false
    },
    "scheduler": {
        "delete_time": 30
    }
}
```

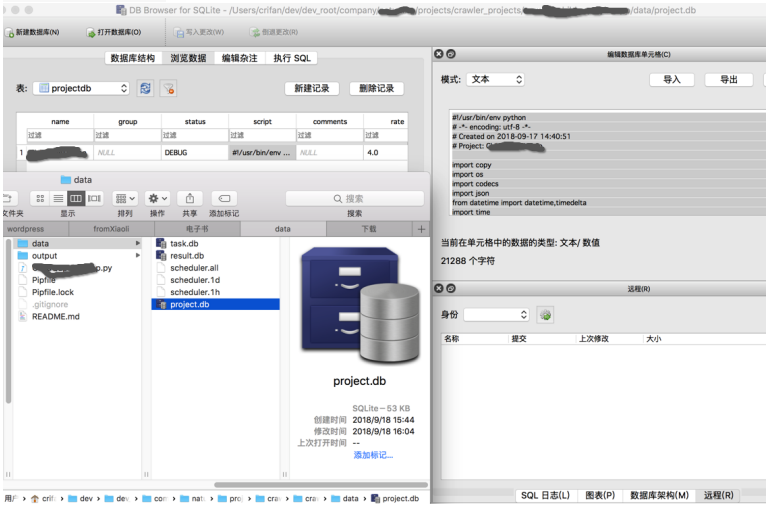
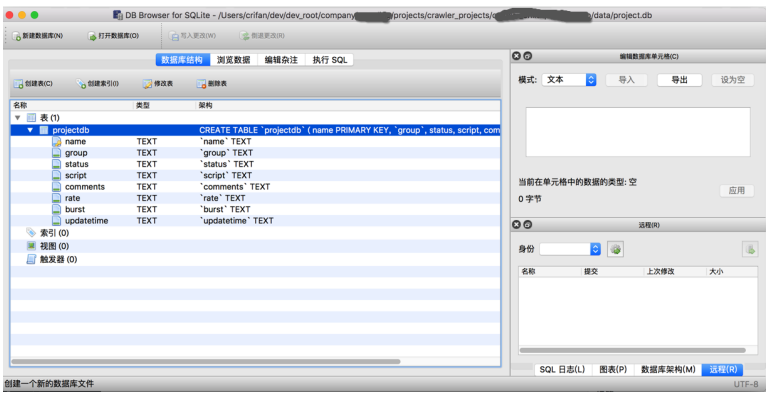
运行调用效果：

```
□ pyspider -c use_python_spider_framework/config.json
[I 200731 14:48:06 result_worker:49] result_worker starting
phantomjs fetcher running on port 25555
[I 200731 14:48:06 processor:211] processor starting...
[I 200731 14:48:06 scheduler:647] scheduler starting...
[I 200731 14:48:06 tornado_fetcher:638] fetcher starting...
[I 200731 14:48:06 scheduler:782] scheduler.xmlrpc listenin
[I 200731 14:48:06 scheduler:126] project crawlBaiduHotList
[I 200731 14:48:06 scheduler:965] select crawlBaiduHotList_
[I 200731 14:48:06 scheduler:586] in 5m: new:0,success:0,re
[I 200731 14:48:06 tornado_fetcher:188] [200] crawlBaiduHot
[D 200731 14:48:06 project_module:145] project: crawlBaiduH
[I 200731 14:48:06 processor:202] process crawlBaiduHotList
[I 200731 14:48:06 scheduler:360] crawlBaiduHotList_PySpide
[I 200731 14:48:06 app:76] webui running on 0.0.0.0:5000
```



PySpider所在目录下的 `data` 目录

在你运行 `pyspider` 后，自动会在命令执行路径下生成 `data` 文件夹，其中包含几个（SQLite）文件：

- `project.db` ：保存了用户的爬虫项目相关信息，包括项目的 `Python代码`
  - 比如用（SQlite）工具去查看，可以看到详细数据
  - 比如Mac中的 `DB Browser for SQLite` 查看的效果：
  - `Python代码` ：
  - 
  - 对应数据库结构字段：
  - 
- `result.db` ：项目运行的结果数据
- `task.db` ：项目相关的任务信息
  - 其中如果开始运行爬虫，还会出现相关的调度信息：
  - `scheduler.all` ， `scheduler.1d` ， `scheduler.1h` ：保存了任务执行后 所有 ， `1天` ， `1小时` 内相关的信息，和WebUI中的 `progress` 中的 `all` ， `1d` ， `1h` 对应：
  - 

指定data目录

用data-path参数

- 方法1：配置 `config.json` 中的 `data-path`

```
{
  "data-path": "/root/xxx/crawler/pyspider/data",
  "webui": {
    "port": 7700,
    "username": "admin",
    "password": "yourPassword",
    "need-auth": true
  },
  "scheduler": {
    "delete_time": 30
  }
}
```

- 方式2：命令行传递参数 `--data-path`

```
--data-path="your_data_folder_path"
```

Phantomjs

如何解决部分页面内部不显示，无法抓取的问题?

折腾【已解决】PySpider中页面部分内容不显示 – 在路上，遇到个问题：

页面中的部分内容不显示，所以无法抓取。

经过研究发现，其实是：

这部分不显示的内容，是原网页中通过后续调用js去生成和获取的，所以可以通过：

给 `self.crawl` 添加

```
fetch_type='js'
```

会使得内部调用 `phantomjs` ，模拟js，渲染生成页面内容。

从而，此处 `PySpider` ，在这种需要显示js加载的页面内容时，可以利用 `phantomjs` 。

用了 phantomis 后又出错： FETCH ERROR HTTP 599 Connection timed out after milliseconds

后续继续运行，加了 `fetch_type='js'` 的代码，去爬取页面数据，结果遇到了：

【未解决】pyspider运行出错：FETCH_ERROR HTTP 599 Connection timed out after milliseconds

尝试了多种办法，都无法解决此问题。

所以目前的情况是：

如果加了 `phantomjs` ，结果在大量爬取页面期间，又会导致出错 `FETCH_ERROR HTTP 599 Connection timed out after milliseconds` ，而暂时找不到解决办法。

给 Phantomjs 添加额外参数

之前折腾过：

【未解决】pyspider中如何给phantomjs传递额外参数 – 在路上

基本上没有实现想要的效果。但是可供参考。

phantomjs中的proxy是什么意思

对于pyspider来说，phantomjs-proxy参数指的是：

你另外所运行的 `phantomjs的实例` = `host:port`

比如：

在一个终端中运行：

```
pyspider phantomjs --port 23450 --auto-restart true
```

然后去另外一个终端中运行pyspider：

```
pyspider -c config.json all
```

其中 `config.json` 包含了：

```
"phantomjs-proxy": "127.0.0.1:23450"
```

就可以使得此处的pyspider在启动时不另外启动phantomjs了。

而是去在需要用到phantomjs时，去连接本地电脑127.0.0.1的23450端口中的phantomjs去处理，去加载页面了。

而对于phantomjs本身来说：

proxy，指的是代理，比如翻墙的代理，等等。

具体相关设置，可以参考：

Command Line Interface | PhantomJS

中的：

- `–proxy=address:port` specifies the proxy server to use (e.g. –proxy=192.168.1.42:8080)
- `–proxy-type=[http|socks5|none]` specifies the type of the proxy server (default is http).
- `–proxy-auth` specifies the authentication information for the proxy, e.g. –proxy-auth=username:password)

详见：【已解决】pyspider中phantomjs中的proxy是什么意思 – 在路上

PySpider经验与心得

折腾了一些PySpider项目，有些经验和心得，整理如下：

PySpider的心得

对于加载更多内容，除了想办法找js或api，也可以换个其他的思路

问题：想要获取单个页面的更多的内容，一般页面都是向下滚动，加载更多。内部往往是js实现，调用额外的api获取更多数据，加载更多数据。

思路：所以一般往往会去研究和抓包，搞清楚调用的api。但是其实有思路多去看看网页中与之相关的其他内容，往往可以通过其他途径，比如另外有个单独的页面，可以获取我所需要的所有的车型车系的数据。就可以避免非要去研究和抓包api了。

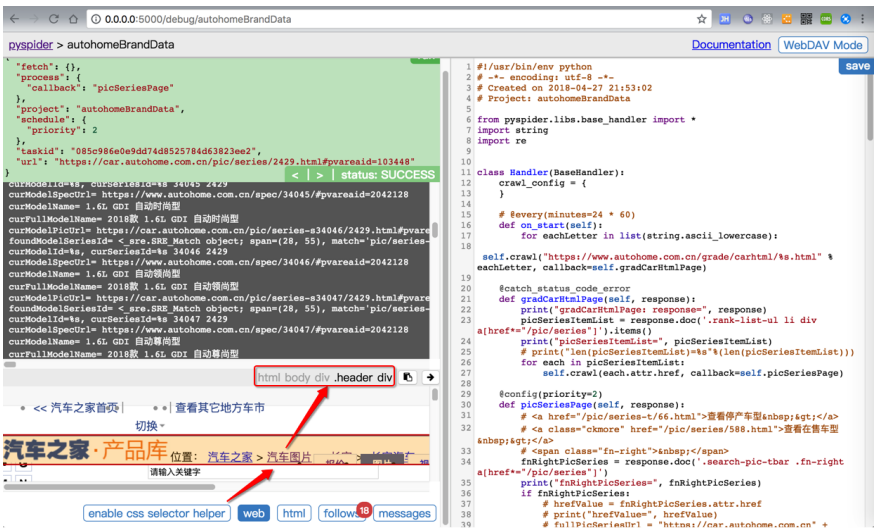
详见：【已解决】pyspider中如何加载汽车之家页面中的更多内容

调试界面中的 `enable css selector helper`

点击web后可以看到html页面内容

再点击 `enable css selector helper` 后

之后点击某个页面元素，则可以直接显示出对应的css的selector



不过话说我个人调试页面期间，很少用到。

都是直接去Chrome浏览器中调试页面，查看html源码，寻找合适的css selector。

发送 POST 请求且传递格式为 application/x-www-form-urlencoded 的 form data 参数

代码：

```
    @config(age=10 * 24 * 60 * 60)
    def index_page(self, response):
        # &lt;ul class="list-user list-user-1" id="list-use
        for each in response.doc('ul[id^="list-user"] li  a
            self.crawl(each.attr.href, callback=self.detai

        maxPageNum = 10
        for curPageIdx in range(maxPageNum):
            curPageNum = curPageIdx + 1
            print("curPageNum=%s" % curPageNum)
            getShowsUrl = "http://xxx/index.php?m=home&amp;
            headerDict = {
                "Content-Type": "application/x-www-form-ur
            }
            dataDict = {
                "counter": curPageNum,
                "order": 1,
                "match_type": 2,
                "match_name": "",
                "act_id": 3
            }
            self.crawl(
                getShowsUrl,
                method="POST",
                headers=headerDict,
                data=dataDict,
                cookies=response.cookies,
                callback=self.parseGetShowsCallback
            )

    def parseGetShowsCallback(self, response):
        print("parseGetShowsCallback: self=%s, response=%s"
        respJson = response.json
        print("respJson=%s" % (respJson))
```

实现了：

- 发送 `POST`
  - 传递header
    - `"Content-Type": "application/x-www-form-urlencoded"`
  - 传递 `data`

        - 一个 `dict` ，包含对应的 `key` 和 `value`
    - 顺带传递了 `cookie`
        - `cookies=response.cookies`
    - 获得返回的 `JSON`
        - `callback` 中用 `response.json`

无法继续爬取时，注意是否是重复url导致的

当发现没有继续爬取后续数据时，记得想想是不是重复url导致的。

比如此处的：

```
POST /selfReadingBookQuery2
{ "offset": 0, "limit":10}
```

和：

```
POST /selfReadingBookQuery2
{ "offset": 10, "limit":10}
```

虽然（json参数）变化了，但是url没变

-> 导致不（重复）爬取

解决办法：让每次的url不同

实现方式：比如给url后面加上 `#hash` 值

举例说明

```
timestampStr = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
curUrlWithHash = curUrl + "#" + timestampStr

self.crawl(curUrlWithHash,
...
```

的：

```
/selfReadingBookQuery2#20190409_162018_413205

/selfReadingBookQuery2#20190409_162117_711811
```

即可实现，每次请求url都不同，就可以继续爬取了。

如果还是不行，或者说，为了更加保险，可以再去加上itag，比如：

```
# add hash value for url to force re-crawl when POST url nc
timestampStr = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
curUrlWithHash = curUrl + "#" + timestampStr

fakeItagForceRecrawl = "%s_%s_%s" % (timestampStr, offset,

self.crawl(curUrlWithHash,
    itag=fakeItagForceRecrawl, # To force re-crawl for next
    method="POST",
```

删除项目

如何清除之前的或正在运行的任务

对于一个写好的爬虫，且已经点击 `Run` 运行，或者运行了一段时间后，主动停止了。

接着想要去删除之前下载的数据，则：

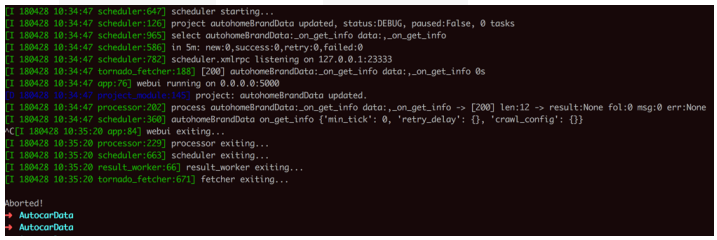
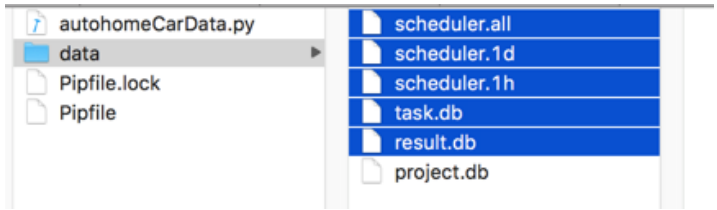
官网的解释是：

设置 `group` 为 `delete` ，以及 `status` 为 `STOP` 后，过了(默认) `24小时` 后，会自动删除该项目所有信息。

但是往往没法满足我们需求：

我不想要等待，只想现在就去：删除掉所有的信息，包括之前已经爬取的数据，之前的调度的任务等等数据。

经过一番研究后，发现了解决方案：

- 先去停止项目
  - WebUI中设置 `status` 为 `STOP`
  - 终端中用 `Control+C` 强制停止 `pyspider` 的运行
    - 
- 再去删除文件： `result.db` 和 `task.db`
  - 如果还有任务相关的 `scheduler.all` ， `scheduler.1d` ， `scheduler.1h` ，则一并删除
    - 

不要轻易在没备份代码情况下删除project.db

注意不要删除，保存了项目（配置和）代码的： `project.db` ，否则代码就没了。（我最开始就这么干过，😂 ）

之后去重新运行pyspider，再去刷新WebUI界面：

http://0.0.0.0:5000/

即可看到干净的项目，没有了之前的任务和数据了。

指定多久之后删除项目，即指定项目删除等待时间

PySpider中的项目，想要删除：

默认逻辑是，status设置为STOP（如果有group，那么group的status也要设置为delete），再等24小时后，才会自动删除

但是往往我们不想要等待那么久

想要指定删除的时间，则有2种方式去设置参数。

举例说明，比如想要 `30秒` 后删除，则可以：

- 文件： `config.json` ：设置 `scheduler` 的 `delete_time` 参数

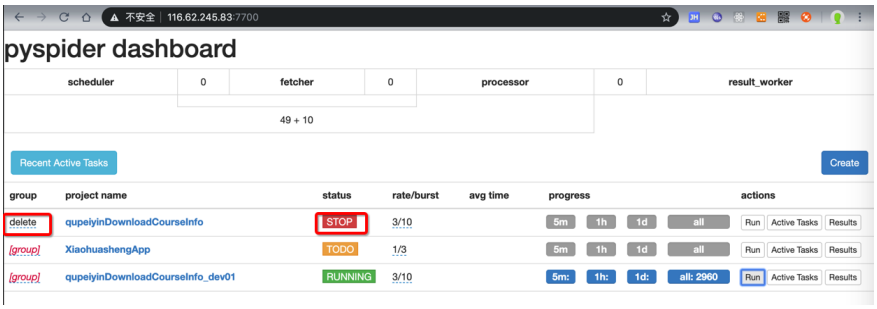
```
"scheduler": {
    "delete_time": 30
}
```

- 命令行传入： `scheduler.DELETE_TIME`

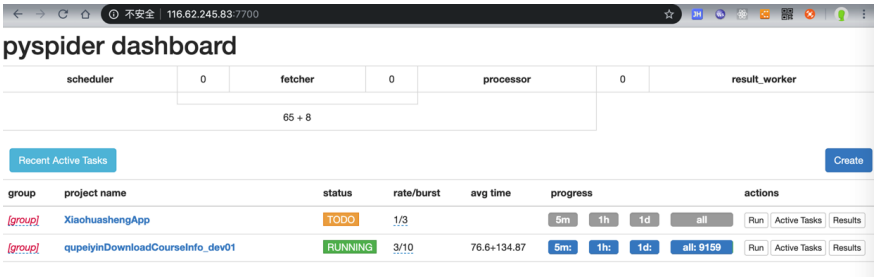
```
pyspider -c config.json scheduler --delete-time=30
```

然后 `WebUI` 中设置

- `status` 是 `STOP`
- `group` 设置为 `delete`



然后过了30秒后，去刷新，该项目就被删除了，看不到了：



PySpider常见的坑

关于折腾PySpider期间，遇到很多或大或小的坑，常见和具体细节相关的坑，已记录到对应部分中了。

此处再继续整理出，其他的一些常见的坑。

`HTTP 599 Operation timed out after milliseconds with out of bytes received`

类似现象：

```
after 120001 milliseconds with 0 bytes receive
after 120000 milliseconds with 1723300 out of 2343850 bytes
```

解释：

-> 意思是：超时了（超过设置的最大超时时间了），但是只下载了总共数据的其中一部分

-> 重点是后半句，意思是可以下载到数据的，只是直到超时都还没下载完全

-> 这种情况的最大可能原因就是：网速太慢

-》所以

解决办法

根本办法：换个更快的网络

比如，我公司是 `1MB/s` 的网络，家里是 `10MB/s` 的网络，换到家里下载，就不会出现这个问题了

临时的规避的缓解的办法：增大延迟 `timeout` （+增大其他容错参数 `connect_timeout`，`retries`）

- 给单个 `self.crawl` 增大参数

```
self.crawl(urlToDownload,
    callback=self.downloadFileCallback,
    connect_timeout=100,
    timeout=600,
    retries=15,
    save=fileInfo)
```

- 或：增大全局参数

```
class Handler(BaseHandler):
    crawl_config = {
        "connect_timeout": 100,
        "timeout": 600,
        "retries": 15,
    }
```

参数含义解释详见官网：

- self.crawl - pyspider

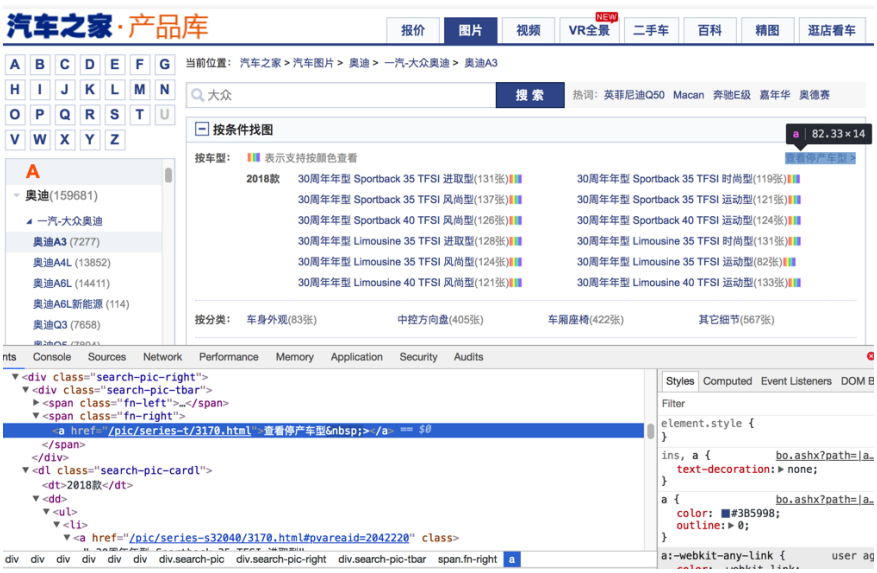
css的选择器不工作

背景：网页中的源码本来是：

```
<a href="//car.autohome.com.cn/pic/series/3170.html#pvarea
```

或者类似的：

```
href="/pic/series-t/3170.html"
```



然后去写css选择器：

```
a[href^="//car.autohome.com.cn/pic/series/"]
```

但是却无法匹配

原因：`PySpider` 内部的css选择器用的是 `PyQuery`，其默认把href的路径，加上了对应的host，所以此时获取到的html实际上变成了：

```
<a href="https://car.autohome.com.cn/pic/series/3170.html#p
```

详见：

response.doc

> Reponse.doc() 返回的就是一个PyQuery的对象 **Links have made as absolute by default**

猜测：估计是为了方便小白用户，所以默认加上了host，但是坑了其他人啊。

解决办法：此处被逼的css选择器写法只能改为：

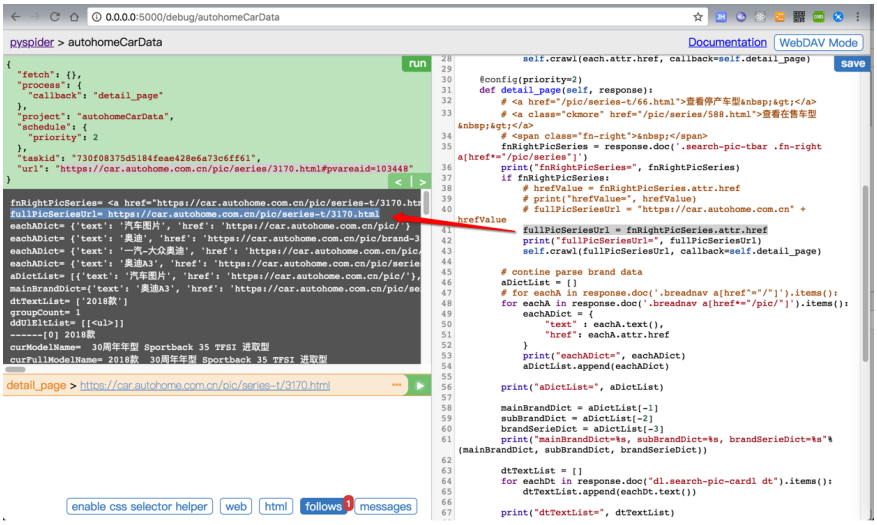
```
a[href*="pic/series/"]
```

或类似的代码：

```
fnRightPicSeries = response.doc('.search-pic-tbar .fn-right
fullPicSeriesUrl = fnRightPicSeries.attr.href
```

已经得到的是，加了host/domain的绝对路径了：

```
fullPicSeriesUrl= https://car.autohome.com.cn/pic/series-t/
```



详见： 【已解决】pyspider中的css选择器不工作 – 在路上

Error Could not create web server listening on port 25555

原因：对应的25555端口被占用了

根本原因：之前的PySpider没有正常的彻底的被关闭，所以残留了。

解决办法：彻底 `kill` 干掉之前的PySpider的进程即可。

举例：

普通Linux类系统，用：

- 找到占了25555端口的进程的id： `ps aux | grep 25555`
- 再去杀掉进程： `kill process_id -9`

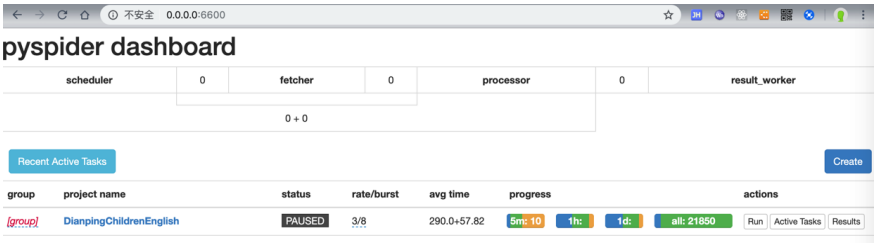
即可。

如果是Mac中，则用 `lsof`

```
➜  AutocarData lsof -i:25555
COMMAND     PID   USER   FD   TYPE             DEVICE SIZE/
phantomjs 46971 crifan   12u  IPv4 0xe4d24cdcaf5e481f
➜  AutocarData kill  46971
```

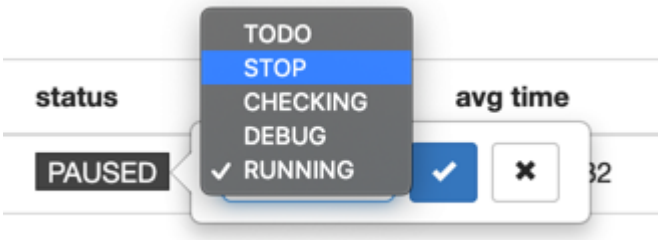
`PAUSED` 后无法立刻继续运行

当PySpider在爬取期间发现太多的url都是 `retry` 重试，则会变成 `PAUSED`



猜测：其内部有比较智能的判断，推测是断网或者网络异常了，所以暂停下载

通过直接把 `PAUSED` 改为 `STOP`



或改为了 `RUNNING`

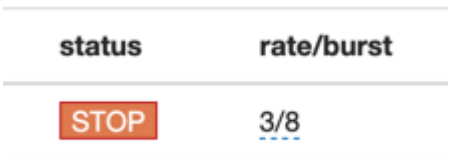
但是刷新一下页面，就还是显示PAUSED

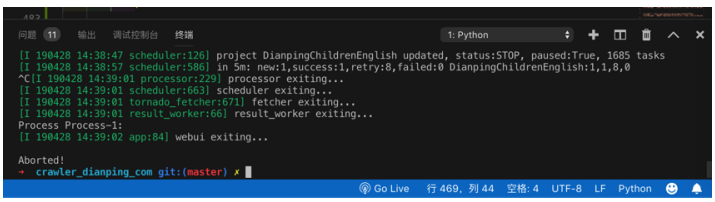
-> 不能立刻开始继续下载

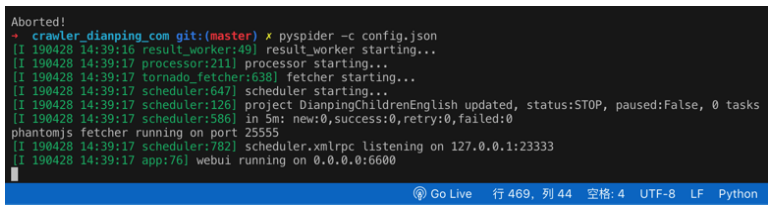
-> 往往要等很长时间之后才能继续下载

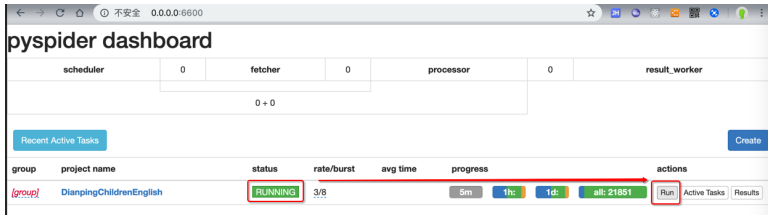
而如果自己想要立刻继续下载，经过研究，可以：

- 先改为STOP

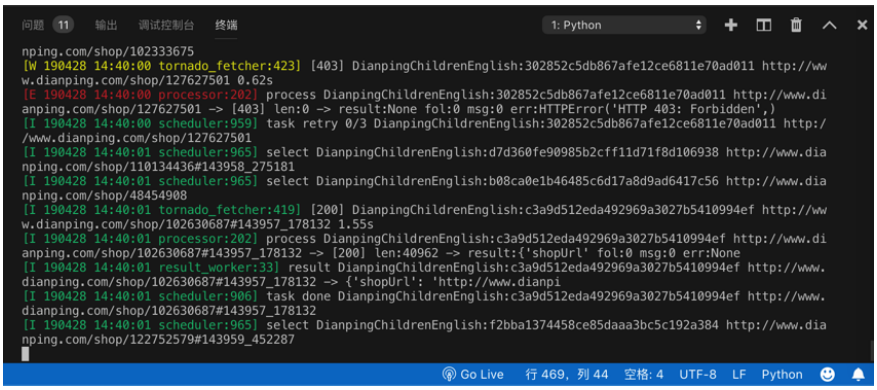


  -
- 停止PySpider
  - Control+C停止
    - 

- 重新运行PySpider
  - 

- 再改为RUNNING，点击Run
  - 

即可立刻继续运行了



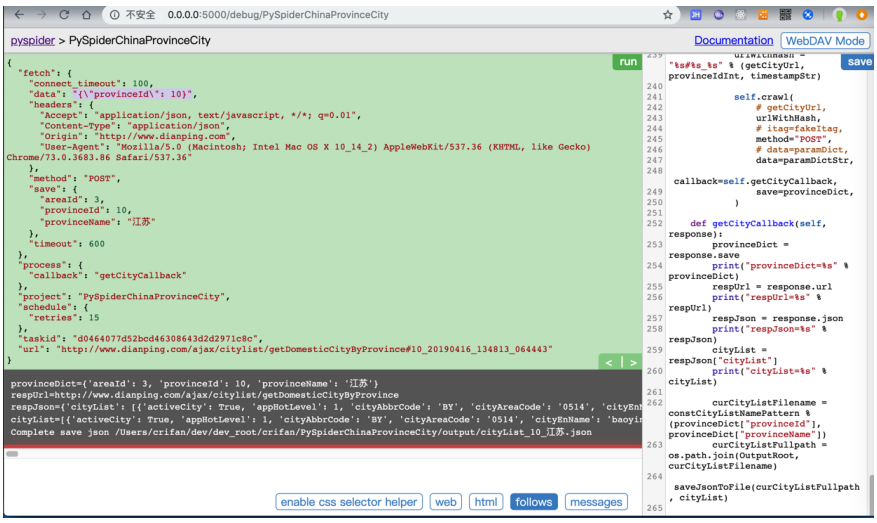
post时data传递dict有时候不行

比如

crifan/PySpiderChinaProvinceCity

```
getCityUrl = "http://www.dianping.com/ajax/citylist/getDome
for eachProvince in provinceList:
    print("eachProvince=%s" % eachProvince)
    provinceIdInt = eachProvince["provinceId"]
    paramDict = {
        "provinceId": provinceIdInt,
    }
    paramDictStr = json.dumps(paramDict)
...
    self.crawl(
        # getCityUrl,
        urlWithHash,
        # itag=fakeItag,
        method="POST",
        # data=paramDict,
        data=paramDictStr,
        callback=self.getCityCallback,
        save=provinceDict,
    )
```

`data` 要传递 `json` 变量 `paramDict` 去 `json.dumps` 后的字符串 `paramDictStr` 才可以：



如果换成 `dict` ：

```
getCityUrl = "http://www.dianping.com/ajax/citylist/getDome
for eachProvince in provinceList:
    print("eachProvince=%s" % eachProvince)
    provinceIdInt = eachProvince["provinceId"]
    paramDict = {
        "provinceId": provinceIdInt,
    }
    paramDictStr = json.dumps(paramDict)
'''
    self.crawl(
        # getCityUrl,
        urlWithHash,
        # itag=fakeItag,
        method="POST",
        data=paramDict,
        # data=paramDictStr,
        callback=self.getCityCallback,
        save=provinceDict,
    )
```

且已经指定了全局配置 `crawl_config` 中的 `headers` 的 `Content-Type` 是 `application/json`

-> 以为 `PySpider` 的 `self.crawl` 会自动把 `dict` 类型的 `data` 去 `json dump` 成字符串，结果却是：

只保留了 `key` 和 `value` ：

```
"data": "provinceId=10",
```

导致报错：

```
    requests.exceptions.HTTPError: HTTP 400: Bad Request
```

![pyspider_http_400_bad_request]

之前还遇到过一个类似的例子：

```
SelfReadingUrl = "http://www.xxxxxxxxx.cn:83/Reading.svc/se

    def on_start(self):
        jValueTemplateSelfReading = "{\"userId\":\"%s\",\"

        paramDictSelfReading = {
            "curUrl": SelfReadingUrl,
            "offset": 0,
            "limit": DefaultPageSize,
            "jValueTemplate": jValueTemplateSelfReading
        }
        self.getBookQuery2(paramDictSelfReading)


    def getBookQuery2(self, curParamDict):
'''
        jValueStr = jValueTemplate % (gUserId, offset, lim
        jcJsonDict = {
            "J": jValueStr,
            "C": 0
        }
        jcJsonDictStr = json.dumps(jcJsonDict)
        ......
        self.crawl(curUrlWithHash,
            itag=fakeItagForceRecrawl, # To force re-crawl
            method="POST",
            # data=jcJsonDict,
            data= jcJsonDictStr,
            # callback=curCallback,
            callback=self.getBookQuery2Callback,
            headers=curHeaders,
            save=curParamDict
        )
```

其中也是：

给 `data` 参数用了 `json` 去 `dump` 后的字符串变量： `jcJsonDictStr`

而不是 `json` 的 `dict` 变量： `jcJsonDict`

才最终正确获取到数据的。

PySpider案例

下面把一些之前写过的 `PySpider` 的爬虫分享出来，供参考。

汽车之家的品牌等数据

文件：`autohomeBrandData.py`

```
#!/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):
        # <a href="/pic/series-t/66.html">查看停产车型 &
        # <a class="ckmore" href="/pic/series/588.html">查看
        # <span class="fn-right"> </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-
                                                curModelPicU
                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
                    "brandSerieId": curSeriesId,
                    "modelId": curModelId,
                    "model": curFullModelName,
                    "modelStatus": modelStatus
                })

        print("modelDetailDictList=", modelDetailDictList)

        allSerieDictList = []
        for curIdx, eachModelDetailDict in enumerate(modelD
            """
            defined in mysql

                CREATE TABLE `tbl_autohome_car_info` (
                  `id` int(11) unsigned NOT NULL AUTO_INCRE
                  `cityDealerPrice` int(11) unsigned NOT NU
                  `msrpPrice` int(11) unsigned NOT NULL DE
                  `mainBrand` char(20) NOT NULL DEFAULT ''
                  `subBrand` varchar(20) NOT NULL DEFAULT
                  `brandSerie` varchar(20) NOT NULL DEFAULT
                  `brandSerieId` varchar(15) NOT NULL DEFAU
                  `model` varchar(50) NOT NULL DEFAULT '' (
                  `modelId` varchar(15) NOT NULL DEFAULT '
                  `modelStatus` char(5) NOT NULL DEFAULT '
```

```
                `url` varchar(200) NOT NULL DEFAULT '' C
                PRIMARY KEY (`id`)
            ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
        """
        curSerieDict = {
            "url": eachModelDetailDict["url"],
            # "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"]
            "mainBrand": mainBrandDict["text"],
            "subBrand": subBrandDict["text"],
            "brandSerie": brandSerieDict["text"],
            "brandSerieId": eachModelDetailDict["brandS
            "model": eachModelDetailDict["model"],
            "modelId": eachModelDetailDict["modelId"],
            "modelStatus": eachModelDetailDict["modelSt
        }
        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"],
                   callback=self.carModelSpecPage,
                   fetch_type='js',
                   retries=5,
                   connect_timeout=50,
                   timeout=300,
                   save=curSerieDict)

    # 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-infor
    print("cityDealerPriceElt=%s" % cityDealerPriceElt)
    if cityDealerPriceElt:
```

```
            cityDealerPriceFloatStr = cityDealerPriceElt.te
            print("cityDealerPriceFloatStr=", cityDealerPri
            cityDealerPriceFloat = float(cityDealerPriceFlo
            print("cityDealerPriceFloat=", cityDealerPriceF
            cityDealerPriceInt = int(cityDealerPriceFloat *
            print("cityDealerPriceInt=", cityDealerPriceInt

        msrpPriceInt = 0
        # body > div.content > div.row > div.column.grid-16
        # 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<msrpPrice>[\d\.
            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
        curSerieDict["cityDealerPrice"] = cityDealerPriceIn
        curSerieDict["msrpPrice"] = msrpPriceInt

        return curSerieDict
```

文件：`AutohomeResultWorker.py`

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Project: autohomeBrandData
# Function: implement custom result worker for autohome ca
# Author: Crifan Li
# Date: 20180512
# Note:
#   If you want to modify to your mysql and table, you nee
#   (1) change change MysqlDb config to your mysql config
#   (2) change CurrentTableName to your table name
#   (3) change CreateTableSqlTemplate to your sql to creat
#   (4) before use this ResultWorker, run py file to execu
#   (5) if your table field contain more type, edit insert


import pymysql
import pymysql.cursors
from pyspider.result import ResultWorker

CurrentTableName = "tbl_autohome_car_info"
CreateTableSqlTemplate = """CREATE TABLE IF NOT EXISTS `%s`
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT '自
  `cityDealerPrice` int(11) unsigned NOT NULL DEFAULT '0' C
  `msrpPrice` int(11) unsigned NOT NULL DEFAULT '0' COMMENT
  `mainBrand` char(20) NOT NULL DEFAULT '' COMMENT '品牌',
  `subBrand` varchar(20) NOT NULL DEFAULT '' COMMENT '子品牌
  `brandSerie` varchar(20) NOT NULL DEFAULT '' COMMENT '车系
  `brandSerieId` varchar(15) NOT NULL DEFAULT '' COMMENT '
  `model` varchar(50) NOT NULL DEFAULT '' COMMENT '车型',
  `modelId` varchar(15) NOT NULL DEFAULT '' COMMENT '车型ID
  `modelStatus` char(5) NOT NULL DEFAULT '' COMMENT '车型状态
  `url` varchar(200) NOT NULL DEFAULT '' COMMENT '车型url',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;"""

class AutohomeResultWorker(ResultWorker):

    def __init__(self, resultdb, inqueue):
        """init mysql db"""
        print("AutohomeResultWorker init: resultdb=%s, inqu
        ResultWorker.__init__(self, resultdb, inqueue)

        self.mysqlDb = MysqlDb()
        print("self.mysqlDb=%s" % self.mysqlDb)

    def on_result(self, task, result):
        """override pyspider on_result to save data into my
        # assert task['taskid']
        # assert task['project']
        # assert task['url']
```

```
        # assert result
        print("AutohomeResultWorker on_result: task=%s, res
        insertOk = self.mysqlDb.insert(result)
        print("insertOk=%s" % insertOk)

class MysqlDb:
    config = {
        'host': '127.0.0.1',
        'port': 3306,
        'user': 'root',
        'password': 'crifan_mysql',
        'database': 'AutohomeResultdb',
        'charset': "utf8"
    }

    defaultTableName = CurrentTableName
    connection = None

    def __init__(self):
        """init mysql"""
        # 1. connect db first
        if self.connection is None:
            isConnected = self.connect()
            print("Connect mysql return %s" % isConnected)

        # 2. create table for db
        createTableOk = self.createTable(self.defaultTableN
        print("Create table %s return %s" %(self.defaultTab

    def connect(self):
        try:
            self.connection = pymysql.connect(**self.config
            print("connect mysql ok, self.connection=", se
            return True
        except pymysql.Error as err:
            print("Connect mysql with config=", self.config
            return False

    def quoteIdentifier(self, identifier):
        """
            for mysql, it better to quote identifier xxx us
            in case, identifier:
                contain special char, such as space
                or same with system reserved words, like se
        """
        quotedIdentifier = "`%s`" % identifier
        # print("quotedIdentifier=", quotedIdentifier)
        return quotedIdentifier

    def executeSql(self, sqlStr, actionDescription=""):
        print("executeSql: sqlStr=%s, actionDescription=%s"
```

```
        if self.connection is None:
            print("Please connect mysql first before %s" %
            return False

        cursor = self.connection.cursor()
        print("cursor=", cursor)

        try:
            cursor.execute(sqlStr)
            self.connection.commit()
            print("+++ Ok to execute sql %s for %s" % (sql
            return True
        except pymysql.Error as err:
            print("!!! %s when execute sql %s for %s" % (e
            return False

    def createTable(self, newTablename):
        print("createTable: newTablename=", newTablename)

        createTableSql = CreateTableSqlTemplate % (newTable
        print("createTableSql=", createTableSql)

        return self.executeSql(sqlStr=createTableSql, actio

    def dropTable(self, existedTablename):
        print("dropTable: existedTablename=", existedTable

        dropTableSql = "DROP TABLE IF EXISTS %s" % (existe
        print("dropTableSql=", dropTableSql)

        return self.executeSql(sqlStr=dropTableSql, action

    # def insert(self, **valueDict):
    def insert(self, valueDict, tablename=defaultTableName
        """
            inset dict value into mysql table
            makesure the value is dict, and its keys is th
        """
        print("insert: valueDict=%s, tablename=%s" % (valu

        dictKeyList = valueDict.keys()
        dictValueList = valueDict.values()
        print("dictKeyList=", dictKeyList, "dictValueList="

        keyListSql = ", ".join(self.quoteIdentifier(eachKe
        print("keyListSql=", keyListSql)
        # valueListSql = ", ".join(eachValue for eachValue
        valueListSql = ""
        formattedDictValueList = []
        for eachValue in dictValueList:
```

```
            # print("eachValue=", eachValue)
            eachValueInSql = ""
            valueType = type(eachValue)
            # print("valueType=", valueType)
            if valueType is str:
                eachValueInSql = '"%s"' % eachValue
            elif valueType is int:
                eachValueInSql = '%d' % eachValue
            # TODO: add more type formatting if necessary
            print("eachValueInSql=", eachValueInSql)
            formattedDictValueList.append(eachValueInSql)

        valueListSql = ", ".join(eachValue for eachValue in
        print("valueListSql=", valueListSql)

        insertSql = """INSERT INTO %s (%s) VALUES (%s)""" %
        print("insertSql=", insertSql)
        # INSERT INTO tbl_car_info_test (`url`, `mainBrand`

        return self.executeSql(sqlStr=insertSql, actionDesc

    def delete(self, modelId, tablename=defaultTableName):
        """
            delete item from car model id for existing tab
        """
        print("delete: modelId=%s, tablename=%s" % (modelId

        deleteSql = """DELETE FROM %s WHERE modelId = %s"""
        print("deleteSql=", deleteSql)

        return self.executeSql(sqlStr=deleteSql, actionDesc

def testMysqlDb():
    """test mysql"""

    testDropTable = True
    testCreateTable = True
    testInsertValue = True
    testDeleteValue = True

    # 1.test connect mysql
    mysqlObj = MysqlDb()
    print("mysqlObj=", mysqlObj)

    # testTablename = "autohome_car_info"
    # testTablename = "tbl_car_info_test"
    testTablename = CurrentTableName
    print("testTablename=", testTablename)

    if testDropTable:
        # 2. test drop table
```

```
        dropTableOk = mysqlObj.dropTable(testTablename)
        print("dropTable", testTablename, "return", dropTab

    if testCreateTable:
        # 3. test create table
        createTableOk = mysqlObj.createTable(testTablename)
        print("createTable", testTablename, "return", creat

    if testInsertValue:
        # 4. test insert value dict
        valueDict = {
            "url": "https://www.autohome.com.cn/spec/5872/#
            "mainBrand": "宝马", #品牌
            "subBrand": "华晨宝马", #子品牌
            "brandSerie": "宝马3系", #车系
            "brandSerieId": "66", #车系ID
            "model": "2010款 320i 豪华型", #车型
            "modelId": "5872", #车型ID
            "modelStatus": "停售", #车型状态
            "cityDealerPrice": 325000, #经销商参考价
            "msrpPrice": 375000 # 厂商指导价
        }
        print("valueDict=", valueDict)
        insertOk = mysqlObj.insert(valueDict=valueDict, tab
        print("insertOk=", insertOk)

    if testDeleteValue:
        toDeleteModelId = "5872"
        deleteOk = mysqlObj.delete(modelId=toDeleteModelId)
        print("deleteOk=", deleteOk)

def testAutohomeResultWorker():
    """just test for create mysql db is ok or not"""
    autohomeResultWorker = AutohomeResultWorker(None, None)
    print("autohomeResultWorker=%s" % autohomeResultWorker)

if __name__ == '__main__':
    testMysqlDb()
    # testAutohomeResultWorker()
```

配置文件：`config.json`

```
{
  "resultdb":   "mysql+resultdb://root:crifan_mysql@127.0.0
  "result_worker":{
      "result_cls": "AutohomeResultWorker.AutohomeResultWo
   },
  "phantomjs-proxy": "127.0.0.1:23450",
  "phantomjs" : {
    "port": 23450,
    "auto-restart": true,
    "load-images": false,
    "debug": true
  }
}
```

汽车之家的车型详细数据

代码

autohome_20200902.py

- 直接下载：autohome_20200902.py
- 贴出如下

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Created on 2020-09-02 21:22:43
# Project: autohome_20200902

import string
import re
import copy
import json

from lxml import etree

# from bs4 import BeautifulSoup

from pyspider.libs.base_handler import *

AutohomeHost = "https://www.autohome.com.cn"
CarSpecPrefix = "%s/spec" % AutohomeHost # "https://www.aut

class Handler(BaseHandler):
    UserAgent_Mac_Chrome = "Mozilla/5.0 (Macintosh; Intel M
    crawl_config = {
        "headers": {
            "User-Agent": UserAgent_Mac_Chrome,
        }
    }

    def genSpecUrl(self, specId):
        # return "%s/%s" % (CarSpecPrefix, specId)
        return "%s/%s/" % (CarSpecPrefix, specId)

    def genConfigSpecUrl(self, specId):
        configSpecTemplate = "https://car.autohome.com.cn/c
        # https://car.autohome.com.cn/config/spec/43593.htm
        return configSpecTemplate % specId

    def to10KPrice(self, originPrice):
        tenKPrice = ""
        # 19.08 / '19.08' -> '19.08万'
        if isinstance(originPrice, str):
            tenKPrice = "%s万" % originPrice
        elif isinstance(originPrice, float):
            tenKPrice = "%.2f万" % originPrice
        elif isinstance(originPrice, int):
            tenKPrice = "%s.00万" % originPrice

        return tenKPrice


    def extractSpecId(self, specUrl):
```

```
        carSpedId = ""
        # https://www.autohome.com.cn/spec/41511/#pvareaid=
        # https://www.autohome.com.cn/spec/2304/
        foundSpecId = re.search("spec/(?P<specId>\d+)", spe
        print("foundSpecId=%s" % foundSpecId)
        if foundSpecId:
            carSpedId = foundSpecId.group("specId")
            print("carSpedId=%s" % carSpedId)
        return carSpedId

    # @every(minutes=24 * 60)
    def on_start(self):
        # autohomeEntryUrl = "https://www.autohome.com.cn/
        # self.crawl(autohomeEntryUrl, callback=self.carBra
        for eachLetter in list(string.ascii_lowercase):
            letterUpper = eachLetter.upper()
            # # for debug
            # letterUpper = "H"
            print("letterUpper=%s" % letterUpper)
            self.crawl("https://www.autohome.com.cn/grade/
                save={"initials": letterUpper},
                callback=self.gradCarHtmlPage)

    @catch_status_code_error
    def gradCarHtmlPage(self, response):
        print("gradCarHtmlPage: response=", response)

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

        saveDict = response.save
        print("saveDict=", saveDict)
        initials = saveDict["initials"]
        print("initials=", initials)
        respText = response.text
        # print("respText=", respText)

        """
        <dl id="33" olr="6">
            <dt><a href="//car.autohome.com.cn/price/brand-
                    src="//car2.autoimg.cn/cardfs/series/g
                <div><a href="//car.autohome.com.cn/price/
            </dt>
        """
        # brandDoc = response.doc('dl dt')
        # print("brandDoc=%s" % brandDoc)
        # brandListDoc = response.doc('dl[id and orl] dt')
        # dlListDoc = response.doc('dl[id and orl]').items
```

```
        # dlListDoc = response.doc("dl[id*=''][orl*='']").
        # dlListDoc = response.doc("dl[orl*='']").items()
        # dlListDoc = response.doc("dl").items()
        # dlListDoc = response.doc("dl:regex(id, \d+)").ite
        # dlListDoc = response.doc("dl:regex(id,[0-9])").it
        # dlListDoc = response.doc("dl[id]").items()
        dlListDoc = response.doc("dl[olr]").items()
        print("type(dlListDoc)=%s" % type(dlListDoc))
        dlList = list(dlListDoc)
        print("len(dlList)=%s" % len(dlList))
        print("dlList=%s" % dlList)
        for curBrandIdx, eachDlDoc in enumerate(dlList):
            print("%s [%d] %s" % ('#'*30, curBrandIdx, '#'

            dtDoc = eachDlDoc.find("dt")
            # print("dtDoc=%s" % dtDoc)
            # <a href="//car.autohome.com.cn/price/brand-33
            brandLogoDoc = dtDoc.find('a img')
            # print("brandLogoDoc=%s" % brandLogoDoc)
            carBrandLogoUrl = brandLogoDoc.attr["src"]
            print("carBrandLogoUrl=%s" % carBrandLogoUrl)
            # <div><a href="//car.autohome.com.cn/price/bra
            brandADoc = dtDoc.find('div a')
            print("brandADoc=%s" % brandADoc)
            # <a href="https://car.autohome.com.cn/price/br
            carBrandName = brandADoc.text()
            print("carBrandName=%s" % carBrandName)
            carBrandUrl = brandADoc.attr["href"]
            print("carBrandUrl=%s" % carBrandUrl)
            carBrandId = ""
            foundBrandId = re.search("brand-(?P<carBrandId>
            # print("foundBrandId=%s" % foundBrandId)
            if foundBrandId:
                carBrandId = foundBrandId.group("carBrandId
            print("carBrandId=%s" % carBrandId) # 63

            # <div class="h3-tit"><a href="//car.autohome.c
            # merchantDocGenerator = response.doc("dd div[c
            # ddDoc = eachDlDoc.find("dd")
            ddDoc = eachDlDoc.find("dd")
            # print("ddDoc=%s" % ddDoc)

            merchantDocGenerator = ddDoc.items("div[class=
            merchantDocList = list(merchantDocGenerator)
            # print("merchantDocList=%s" % merchantDocList)
            merchantDocLen = len(merchantDocList)
            print("merchantDocLen=%s" % merchantDocLen)

            # <ul class="rank-list-ul" 0>
            # merchantRankDocGenerator = response.doc("dd u
            # merchantRankDocGenerator = response.doc("dd u
```

```
            merchantRankDocGenerator = ddDoc.items("ul[clas
            merchantRankDocList = list(merchantRankDocGener
            # print("merchantRankDocList=%s" % merchantRank
            merchantRankDocListLen = len(merchantRankDocLis
            print("merchantRankDocListLen=%s" % merchantRan

            for curIdx, merchantItem  in enumerate(merchant
            # for curIdx, merchantItem  in enumerate(mercha
                # print("%s" % "="*80)
                print("%s [%d] %s" % ('='*30, curIdx, '='*3
                # print("type(merchantItem)=%s" % type(merc
                # print("[%d] merchantItem=%s" % (curIdx, m
                # print("[%d] merchantItem=%s" % (curIdx, m
                carMerchantName = merchantItem.text()
                print("carMerchantName=%s" % carMerchantNam
                merchantItemAttr = merchantItem.attr
                # print("merchantItemAttr=%s" % merchantIte
                carMerchantUrl = merchantItemAttr["href"]
                print("carMerchantUrl=%s" % carMerchantUrl)

                # curSubBrandDict = {
                #     "brandName": brandName,
                #     "carBrandLogoUrl": carBrandLogoUrl,
                #     "carMerchantName": carMerchantName,
                #     "carMerchantUrl": carMerchantUrl,
                # }
                # self.send_message(self.project_name, curS

                merchantRankDoc = merchantRankDocList[curId
                # print("merchantRankDoc=%s" % merchantRank
                # print("type(merchantRankDoc)=%s" % type(m

                # type(merchantRankDoc)=<class 'lxml.html.H
                # merchantRankHtml = etree.tostring(merchan

                # type(merchantRankDoc)=<class 'pyquery.pyq
                # merchantRankHtml = merchantRankDoc.html()

                # print("merchantRankHtml=%s" % merchantRan

                # <li id="s3170">
                # carSeriesDocGenerator = merchantRankDoc.f
                # carSeriesDocGenerator = merchantRankDoc.f
                carSeriesDocGenerator = merchantRankDoc.ite
                # print("type(carSeriesDocGenerator)=%s" %
                carSeriesDocList = list(carSeriesDocGenerat
                # print("type(carSeriesDocList)=%s" % type(
                # print("carSeriesDocList=%s" % carSeriesDoc
                carSeriesDocListLen = len(carSeriesDocList)
                # print("carSeriesDocListLen=%s" % carSerie
```

```
                for curSeriesIdx, eachCarSeriesDoc in enume
                    print("%s [%d] %s" % ('-'*30, curSeries
                    # print("[%d] eachCarSeriesDoc=%s" % (
                    # print("type(eachCarSeriesDoc)=%s" % t
                    # <h4><a href="//www.autohome.com.cn/3
                    carSeriesInfoDoc = eachCarSeriesDoc.fir
                    # print("type(carSeriesInfoDoc)=%s" % t
                    # print("carSeriesInfoDoc=%s" % carSer
                    carSeriesName = carSeriesInfoDoc.text()
                    print("carSeriesName=%s" % carSeriesNar
                    carSeriesUrl = carSeriesInfoDoc.attr.hr
                    print("carSeriesUrl=%s" % carSeriesUrl)

                    # <div>指导价: <a class="red" href="//ww
                    # 厂商指导价=厂商建议零售价格=MSRP=Manufact
                    # carSeriesMsrpDoc = eachCarSeriesDoc.f
                    carSeriesMsrpDoc = eachCarSeriesDoc.fir
                    # print("carSeriesMsrpDoc=%s" % carSer
                    carSeriesMsrp = carSeriesMsrpDoc.text()
                    print("carSeriesMsrp=%s" % carSeriesMsr
                    carSeriesMsrpUrl = carSeriesMsrpDoc.att
                    print("carSeriesMsrpUrl=%s" % carSeries

                    carSeriesDict = {
                        "carBrandName": carBrandName,
                        "carBrandId": carBrandId,
                        "carBrandLogoUrl": carBrandLogoUrl,
                        "carMerchantName": carMerchantName,
                        "carMerchantUrl": carMerchantUrl,
                        "carSeriesName": carSeriesName,
                        "carSeriesUrl": carSeriesUrl,
                        "carSeriesMsrp": carSeriesMsrp,
                        "carSeriesMsrpUrl": carSeriesMsrpUr
                    }
                    # self.send_message(self.project_name,
                    self.crawl(carSeriesUrl,
                        callback=self.carSeriesDetailPage,
                        save=carSeriesDict,
                    )

    @catch_status_code_error
    def carSeriesDetailPage(self, response):
        print("in carSeriesDetailPage")
        carSeriesDict = response.save
        print("carSeriesDict=%s" % carSeriesDict)

        carModelDict = copy.deepcopy(carSeriesDict)

        carSeriesUrl = response.url
        print("carSeriesUrl=%s" % carSeriesUrl)
```

```
        carSeriesMainImgUrl = ""
        # carSeriesId = ""
        carSeriesLevelId = ""
        carSeriesMsrp = ""
        carSeriesMinPrice = ""
        carSeriesMaxPrice = ""

        # carSeriesUrl=https://www.autohome.com.cn/2123/#le
        foundSeriesId = re.search("www\.autohome\.com\.cn/(
        carSeriesId = foundSeriesId.group("seriesId")
        # carSeriesId = int(carSeriesId)
        print("carSeriesId=%s" % carSeriesId) # 2123
        carModelDict["carSeriesId"] = carSeriesId

        carSeriesHtml = response.text
        print("type(carSeriesHtml)=%s" % type(carSeriesHtm
        # print("carSeriesHtml=%s" % carSeriesHtml)

        foundLevelId = re.search("var\s+levelid\s+=", carSe
        print("foundLevelId=%s" % foundLevelId)
        isNewLayoutHtml = bool(foundLevelId)
        print("isNewLayoutHtml=%s" % isNewLayoutHtml)
        foundShowCityId = re.search("var\s+showCityId\s+=",
        print("foundShowCityId=%s" % foundShowCityId)
        isOldLayoutHtml = bool(foundShowCityId)
        print("isOldLayoutHtml=%s" % isOldLayoutHtml)

        if isOldLayoutHtml:
            # Q开头
            # https://www.autohome.com.cn/grade/carhtml/q.h
            # ->
            # 东风悦达起亚-千里马
            # https://www.autohome.com.cn/142/#levelsource=
            # 其他:
            #
            # 一汽丰田-花冠
            # https://www.autohome.com.cn/109/#levelsource=
            #
            # 昶洋-昶洋 SUV
            # https://www.autohome.com.cn/4550/#levelsource

            """
            <div class="car_detail " id="tab1-2">
                <div class="models">
                <!--年代-->
                    <div class="header">
                        <div class="car_price">
                            <span class="years">2005款</spa
                            <span class="price">指导价（停售）
                            <span class="price">二手车价格：<

            。。。
```

```
                <div class="car_detail current" id="tab1-1">
                    <div class="models">
                        <!--年代-->
                        <div class="header">
                            <div class="car_price">
                                <span class="years">2006款</spa
                                <span class="price">指导价（停售）
                ° ° °
                """
                carDetailDivGenerator = response.doc("div[class
                print("carDetailDivGenerator=%s" % carDetailDiv
                carDetailDivList = list(carDetailDivGenerator)
                print("carDetailDivList=%s" % carDetailDivList)
                for curDivIdx, eachCarDetailDoc in enumerate(ca
                    print("%s [%d] %s" % ('#'*30, curDivIdx, '#
                    if curDivIdx == 0:
                        # use first car model as series: main
                        """
                        <div class="models_info">
                            <dl class='models_pics'>
                                <dt><a href='//car.autohome.com
                                    src='https://car0.autoimg.c
                                    height='180' /></a></dt>
                        """
                        # modelMainImgDocListGenerator = respon
                        # modelMainImgDocList = list(modelMain
                        # firstModelMainImgDoc = modelMainImgDo
                        firstModelMainImgDoc = eachCarDetailDoc
                        firstModelMainImgUrl = firstModelMainIm
                        print("firstModelMainImgUrl=%s" % first
                        carSeriesMainImgUrl = firstModelMainImg
                        print("carSeriesMainImgUrl=%s" % carSer

                        carModelDict["carSeriesMainImgUrl"] = c

                        # <div class="car_price">
                        #   <span class="price">指导价（停售）：<
                        carPriceStrongDocGenerator = eachCarDet
                        print("carPriceStrongDocGenerator=%s" %
                        if carPriceStrongDocGenerator:
                            carPriceStrongDocList = list(carPri
                            print("carPriceStrongDocList=%s" %
                            carPriceStrongDoc = carPriceStrongD
                            print("carPriceStrongDoc=%s" % carP
                            carPriceMinMax = carPriceStrongDoc.
                            print("carPriceMinMax=%s" % carPric
                            if carPriceMinMax:
                                foundMinMax = re.search("(?P<m
                                print("foundMinMax=%s" % foundM
                                if foundMinMax:
```

```
                                minPrice = foundMinMax.grou
                                print("minPrice=%s" % minPr
                                minPriceFloat = float(minPr
                                print("minPriceFloat=%s" %
                                maxPrice = foundMinMax.grou
                                print("maxPrice=%s" % maxPr
                                maxPriceFloat = float(maxPr
                                print("maxPriceFloat=%s" %
                                averageMsrcPrice = (minPric
                                print("averageMsrcPrice=%s"

                                # carSeriesMsrp = "%.2f万"
                                carSeriesMsrp = self.to10KF
                                print("carSeriesMsrp=%s" %
                                # carSeriesMinPrice = "%.2f
                                carSeriesMinPrice = self.to
                                print("carSeriesMinPrice=%s
                                # carSeriesMaxPrice = "%.2f
                                carSeriesMaxPrice = self.to
                                print("carSeriesMaxPrice=%s

                                carModelDict["carSeriesMsrp
                                carModelDict["carSeriesMinP
                                carModelDict["carSeriesMaxP
                    print("")

                self.processSingleCarDetailDiv(carModelDict

            elif isNewLayoutHtml:
                # https://www.autohome.com.cn/3170/#levelsource

                """
                <div class="information-pic">
                    <div class="pic-main">
                    ...
                            <picture>
                                ...
                                <img sizes="380px" width="380"
                                    src="//car2.autoimg.cn/carc
                                    srcset="//car2.autoimg.cn/c
                            </picture>
                """
                mainImgDoc = response.doc("div[class='informati
                print("mainImgDoc=%s" % mainImgDoc)
                carSeriesMainImgUrl = mainImgDoc.attr["src"]
                print("carSeriesMainImgUrl=%s" % carSeriesMainI
                carModelDict["carSeriesMainImgUrl"] = carSeries

                """
                <script type="text/javascript">
                    ...
```

```
                var seriesid = '2123';
                var seriesname='哈弗H6';
                var yearid = '0';
                var brandid = '181';
                var levelid = '17';
                var levelname='紧凑型SUV';
                var fctid = '4';
                var SeriesMinPrice='9.80';
                var SeriesMaxPrice='14.10';
            """

            infoKeyList = [
                "seriesid",
                # "seriesname", # has got
                # "yearid", # no need
                "brandid",
                "levelid",
                "levelname",
                # "fctid", # unknown meaning
                "SeriesMinPrice",
                "SeriesMaxPrice",
            ]
            InfoDict = {}
            for eachInfoKey in infoKeyList:
                curPattern = "var\s+%s\s*=\s*'(?P<infoValue
                print("curPattern=%s" % curPattern)
                foundInfo = re.search(curPattern, carSeries
                print("foundInfo=%s" % foundInfo)
                # if foundInfo:
                infoValue = foundInfo.group("infoValue")
                print("infoValue=%s" % infoValue)
                InfoDict[eachInfoKey] = infoValue
            print("InfoDict=%s" % InfoDict)

            # if "seriesid" in InfoDict:
            carSeriesId = InfoDict["seriesid"] # 2123
            carModelDict["carSeriesId"] = carSeriesId
            # carModelDict["carSeriesName"] = InfoDict["ser
            # if "brandid" in InfoDict:
            carModelDict["carBrandId"] = InfoDict["brandid"
            # if "levelid" in InfoDict:
            carSeriesLevelId = InfoDict["levelid"] # 17
            carModelDict["carSeriesLevelId"] = carSeriesLev
            # if "levelname" in InfoDict:
            carModelDict["carSeriesLevelName"] = InfoDict["
            # if "SeriesMinPrice" in InfoDict:
            carSeriesMinPrice = InfoDict["SeriesMinPrice"]
            carModelDict["carSeriesMinPrice"] = self.to10KF
            # if "SeriesMaxPrice" in InfoDict:
            carSeriesMaxPrice = InfoDict["SeriesMaxPrice"]
            carModelDict["carSeriesMaxPrice"] = self.to10KF
```

```
            """
            <div class="series-list">
            ...
                <li class="more-dropdown">
                    <a href="javascript:void(0);" target="_
                    <ul class="dropdown-con" id="haltList">
                        <li><a href="javascript:void(0);" t
                        ...
                        <li><a href="javascript:void(0);" t
                    </ul>
                </li>
            """
            haltADocGenerator = response.doc("li[class='mor
            print("type(haltADocGenerator)=%s" % type(haltA
            print("haltADocGenerator=%s" % haltADocGenerato
            haltADocList = list(haltADocGenerator)
            print("haltADocList=%s" % haltADocList)
            for curLiIdx, eachHatADoc in enumerate(haltADoc
                print("%s [%d] %s" % ('%'*30, curLiIdx, '%
                self.processSingleHaltA(carModelDict, eachH

            # """
            # <div class="information-summary">
            #     <dl class="information-price">
            #         ...
            #         <dd class="type">
            #             <span class="type__item">紧凑型车<
            # """
            # carLevelDoc = response.doc("div[class='inform
            # print("carLevelDoc=%s" % carLevelDoc)
            # carSeriesLevelName = carLevelDoc.text()
            # print("carSeriesLevelName=%s" % carSeriesLeve
            # carModelDict["carSeriesLevelName"] = carSerie

            carSeriesContentDoc = response.doc("div[class=
            # print("carSeriesContentDoc=%s" % carSeriesCon
            # carSpecWrapDoc = carSeriesContentDoc.find("d
            # carSpecWrapDoc = carSeriesContentDoc.find("d
            carSpecWrapDocGenerator = carSeriesContentDoc.
            print("carSpecWrapDocGenerator=%s" % carSpecWra
            carSpecWrapDocList = list(carSpecWrapDocGenerat
            print("carSpecWrapDocList=%s" % carSpecWrapDocL
            for curSpecWrapIdx, eachSpecWrapDoc in enumerat
                print("%s [%d] %s" % ('#'*30, curSpecWrapId
                self.processSingleSpecWrapDiv(carModelDict,

    def processSingleCarDetailDiv(self, carModelDict, curCa
        print("in processSingleCarDetailDiv")
        curCarModelGroupDict = copy.deepcopy(carModelDict)
```

```
        # <span class="years">2006款</span>
        modelYearDoc = curCarDetailDoc.find("span[class='ye
        print("modelYearDoc=%s" % modelYearDoc)
        carModelYear = modelYearDoc.text()
        print("carModelYear=%s" % carModelYear)
        curCarModelGroupDict["carModelYear"] = carModelYear

        """
        <div class="modelswrap">
            <!-- 信息 start -->
            <div class="models_info">
                <dl class='models_prop'>
                    <dt>发动机: </dt>
                    <dd><span>1.3L</span><span>1.6L</span><
                </dl>
                <dl class='models_prop'>
                    <dt>变速箱: </dt>
                    <dd><span>手动</span><span>自动</span></
                    <dt>车身结构: </dt>
                    <dd><span>三厢</span></dd>
                </dl>
        """
        # modelsPropDdList = curCarDetailDoc.find("div[clas
        modelsPropDdGenerator = curCarDetailDoc.items("div
        print("modelsPropDdGenerator=%s" % modelsPropDdGene
        modelsPropDdList = list(modelsPropDdGenerator)
        print("modelsPropDdList=%s" % modelsPropDdList)
        engineValueDoc = modelsPropDdList[0]
        print("engineValueDoc=%s" % engineValueDoc)
        carModelEngine = engineValueDoc.text()
        print("carModelEngine=%s" % carModelEngine)

        gearBoxValueDoc = modelsPropDdList[1]
        print("gearBoxValueDoc=%s" % gearBoxValueDoc)
        carModelGearBox = gearBoxValueDoc.text()
        print("carModelGearBox=%s" % carModelGearBox)

        bodyStructureValueDoc = modelsPropDdList[2]
        print("bodyStructureValueDoc=%s" % bodyStructureVa
        carModelBodyStructure = bodyStructureValueDoc.text
        print("carModelBodyStructure=%s" % carModelBodyStru

        curCarModelGroupDict["carModelGearBox"] = carModel
        curCarModelGroupDict["carModelDriveType"] = ""
        curCarModelGroupDict["carModelBodyStructure"] = ca

        # curCarModelGroupDict["carModelEnvStandard"] = ""
        # carModelPower = carModelEngine
        # print("carModelPower=%s" % carModelPower)
        # curCarModelGroupDict["carModelPower"] = carModel
        curCarModelGroupDict["carModelEngine"] = carModelEn
```

```
        carModelGroupName = "%s %s %s" % (carModelEngine, c
        print("carModelGroupName=%s" % carModelGroupName)
        curCarModelGroupDict["carModelGroupName"] = carMode

        """
        <table class='models_tab tableline' cellspacing='0
            <tr>
                <td class='name_d'>
                    <div class='name'><a title='2006款 1.6L
                </td>
                <td class='price_d'>
                    <div class='price01'>8.18万</div>
                </td>
        """
        modelsTrDocGenerator = curCarDetailDoc.items("table
        print("modelsTrDocGenerator=%s" % modelsTrDocGenera
        modelsTrDocList = list(modelsTrDocGenerator)
        print("modelsTrDocList=%s" % modelsTrDocList)
        for curTabIdx, eachModelTrDoc in enumerate(modelsTr
            print("%s [%d] %s" % ('='*30, curTabIdx, '='*30
            self.processSingleModelsTr(curCarModelGroupDict

    def processSingleModelsTr(self, curCarModelGroupDict, c
        curTrCarModeDict = copy.deepcopy(curCarModelGroupDi
        # print("curModelTrDoc=%s" % curModelTrDoc)
        nameADoc = curModelTrDoc.find("td[class='name_d'] a
        print("nameADoc=%s" % nameADoc)
        carModelName = nameADoc.text()
        print("carModelName=%s" % carModelName)

        carModelSpecUrl = nameADoc.attr["href"]
        # bug -> wrong url:
        # https://www.autohome.com.cn/142/spec/2304/
        # need repace
        # https://www.autohome.com.cn/142/spec/2304/
        # to
        # https://www.autohome.com.cn/spec/2304/
        foundSpecId = re.search("spec/(?P<specId>\d+)", car
        carModelSpecId = foundSpecId.group("specId")
        print("carModelSpecId=%s" % carModelSpecId) # 2304
        carModelSpecUrl = self.genSpecUrl(carModelSpecId)
        print("carModelSpecUrl=%s" % carModelSpecUrl)

        priceDivDoc = curModelTrDoc.find("td[class='price_d
        print("priceDivDoc=%s" % priceDivDoc)
        carModelMsrp = priceDivDoc.text()
        print("carModelMsrp=%s" % carModelMsrp)

        curTrCarModeDict["carModelName"] = carModelName
        curTrCarModeDict["carModelSpecUrl"] = carModelSpecU
```

```
            curTrCarModeDict["carModelMsrp"] = carModelMsrp

            self.processSingleResult(curTrCarModeDict)

        def processSingleHaltA(self, carModelDict, curHatADoc):
            curHaltCarDict = copy.deepcopy(carModelDict)
            print("curHatADoc=%s" % curHatADoc)
            yearName = curHatADoc.text()
            print("yearName=%s" % yearName)
            yearId = curHatADoc.attr["data-yearid"]
            print("yearId=%s" % yearId)

            # getHaltSpecUrl = "https://www.autohome.com.cn/ash
            carSeriesId = curHaltCarDict["carSeriesId"]
            carSeriesLevelId = curHaltCarDict["carSeriesLevelId
            if carSeriesId and carSeriesLevelId:
                getHaltSpecUrl = "https://www.autohome.com.cn/a
                # https://www.autohome.com.cn/ashx/car/Spec_Lis
                print("getHaltSpecUrl=%s" % getHaltSpecUrl)
                self.crawl(getHaltSpecUrl,
                    callback=self.haltCarSpecCallback,
                    save=curHaltCarDict,
                )

        def processSingleSpecWrapDiv(self, curCarModelDict, cur
            curSpecWrapCarDict = copy.deepcopy(curCarModelDict)
            # print("curSpecWrapDoc=%s" % curSpecWrapDoc)
            """
            <!--即将上市 start-->
            <div class="spec-wrap  active" id="specWrap-1">

                <dl class="halt-spec">
                    <dt>
                        <div class="spec-name">
                            <span>参数配置未公布</span>
                        </div>

            <dl class="halt-spec">
                <dt>
                    <div class="spec-name">
                        <span>1.5升 涡轮增压 169马力 国VI</span>
                    </div>
            """
            # dlDoc = curSpecWrapDoc.find("dl[class='']")
            # dlDoc = curSpecWrapDoc.find("dl")
            dlListDocGenerator = curSpecWrapDoc.items("dl")
            print("dlListDocGenerator=%s" % dlListDocGenerator)
            dlDocList = list(dlListDocGenerator)
            print("dlDocList=%s" % dlDocList)
            for curDlIdx, eachDlDoc in enumerate(dlDocList):
                print("%s [%d] %s" % ('='*30, curDlIdx, '='*30)
```

```
            self.processSingleSpecDl(curSpecWrapCarDict, ea

    def processSingleSpecDl(self, curSpecWrapCarDict, curD
        curDlCarDict = copy.deepcopy(curSpecWrapCarDict)
        # print("curDlDoc=%s" % curDlDoc)
        """
            <dt>
                <div class="spec-name">
                    <span>1.5升 涡轮增压 169马力 国VI</span>
        """
        dtDoc = curDlDoc.find("dt")
        # print("dtDoc=%s" % dtDoc)
        groupSpecNameSpanDoc = dtDoc.find("div[class='spec-
        print("groupSpecNameSpanDoc=%s" % groupSpecNameSpar
        carModelGroupName = ""
        if groupSpecNameSpanDoc:
            carModelGroupName = groupSpecNameSpanDoc.text()
            print("carModelGroupName=%s" % carModelGroupNar

        curDlCarDict["carModelGroupName"] = carModelGroupNa

        # <dd data-sift1="2020款" data-sift2="国VI" data-si
        ddListDoc = curDlDoc.items("dd")
        print("ddListDoc=%s" % ddListDoc)
        for curDdIdx, eachDdDoc in enumerate(ddListDoc):
            print("%s [%d] %s" % ('-'*30, curDdIdx, '-'*30)
            self.processSingleSiftDd(curDlCarDict, eachDdDo

    def processSingleSiftDd(self, curDlCarDict, curDdDoc):
        print("in processSingleSiftDd")
        curDdCarDict = copy.deepcopy(curDlCarDict)

        curDdAttr = curDdDoc.attr
        """
        正常:
            <dd data-sift1="2020款" data-sift2="国VI" data-
                ...
        特殊:
            无sift:
                <dd data-electricspecid="47050">
            电动的 sift位置不同:
                https://www.autohome.com.cn/5240/
                    <dd data-electricspecid="42875" data-si
            混动 sift位置也不同:
                https://www.autohome.com.cn/4460/
                    <dd data-electricspecid="37077" data-si
        """
        # print("curDdAttr=%s" % curDdAttr)
        # carModelDataSift1 = curDdAttr["data-sift1"]
        # print("carModelDataSift1=%s" % carModelDataSift1)
        carModelYear = curDdAttr["data-sift1"]
```

```
        print("carModelYear=%s" % carModelYear)
        carModelDataSift2 = curDdAttr["data-sift2"]
        print("carModelDataSift2=%s" % carModelDataSift2)
        carModelDataSift3 = curDdAttr["data-sift3"]
        print("carModelDataSift3=%s" % carModelDataSift3)
        carModelDataSift4 = curDdAttr["data-sift4"]
        print("carModelDataSift4=%s" % carModelDataSift4)

        curDdCarDict["carModelYear"] = carModelYear
        # curDdCarDict["carModelEnvStandard"] = carModelEnv
        # curDdCarDict["carModelPower"] = carModelPower
        # curDdCarDict["carModelGearBox"] = carModelGearBox
        # curDdCarDict["carModelDataSift1"] = carModelDataS
        curDdCarDict["carModelDataSift2"] = carModelDataSi
        curDdCarDict["carModelDataSift3"] = carModelDataSi
        curDdCarDict["carModelDataSift4"] = carModelDataSi

        """
        <div class="spec-name">
            <div class="name-param">
                <p data-gcjid="41511" id="spec_41511">
                    <a href="/spec/41511/#pvareaid=3454492"
                    <span class="athm-badge athm-badge--gre
                <span class="athm-badge athm-badge--orange"
                <p><span class="type-default">前置前驱</span
            </div>
        </div>
        """
        specNameDoc = curDdDoc.find("div[class='spec-name']
        # print("specNameDoc=%s" % specNameDoc)
        specADoc = specNameDoc.find("p a[class='name']")
        # print("specADoc=%s" % specADoc)
        carModelName = specADoc.text()
        print("carModelName=%s" % carModelName) # 2020款 1.
        carModelSpecUrl = specADoc.attr["href"]
        print("carModelSpecUrl=%s" % carModelSpecUrl) # htt
        typeDefaultListDoc = specNameDoc.items("p span[clas
        print("typeDefaultListDoc=%s" % typeDefaultListDoc)
        typeDefaultList = list(typeDefaultListDoc)
        print("typeDefaultList=%s" % typeDefaultList)
        carModelDriveType = ""
        carModelGearBox = ""
        if typeDefaultList:
            """
            正常:
                <p>
                    <span class="type-default">前置前驱</spa
                    <span class="type-default">7挡双离合</sp
                </p>

            特殊:
```

```
                https://www.autohome.com.cn/4605/
                <p>
                    <span class="type-default">电动</span>
                    <span class="type-default">前置前驱</spa
                    <span class="type-default">AMT（组合10挡
                </p>

                https://www.autohome.com.cn/4460/
                <p>
                    <span class="type-default">电动</span>
                    <span class="type-default">前置四驱</spa
                    <span class="type-default">8挡手自一体</
                </p>

                https://www.autohome.com.cn/5240/
                <p>
                    <span class="type-default">电动</span>
                    <span class="type-default">前置前驱</spa
                    <span class="type-default">电动车单速变速
                </p>
            """
            # spanTypeDefault0 = typeDefaultList[0]
            spanTypeDefault0 = typeDefaultList[-2]
            print("spanTypeDefault0=%s" % spanTypeDefault0)
            carModelDriveType = spanTypeDefault0.text()
            print("carModelDriveType=%s" % carModelDriveTyp
            # 前置前驱
            # 前置四驱
            # spanTypeDefault1 = typeDefaultList[1]
            spanTypeDefault1 = typeDefaultList[-1]
            print("spanTypeDefault1=%s" % spanTypeDefault1)
            carModelGearBox = spanTypeDefault1.text()
            print("carModelGearBox=%s" % carModelGearBox)
            # 7挡双离合
            # AMT（组合10挡）
            # 8挡手自一体
            # 电动车单速变速箱

        curDdCarDict["carModelName"] = carModelName
        if not curDdCarDict["carModelYear"]:
            foundYearType = re.search("(?P<yearType>\d{4}款
            if foundYearType:
                yearType = foundYearType.group("yearType")
                print("yearType=%s" % yearType)
                carModelYear = yearType
                print("extract year=%s from modelName=%s" %
                curDdCarDict["carModelYear"] = carModelYear

        curDdCarDict["carModelSpecUrl"] = carModelSpecUrl
        curDdCarDict["carModelDriveType"] = carModelDriveTy
        curDdCarDict["carModelGearBox"] = carModelGearBox #
```

```
        """
        <div class="spec-guidance">
            <p class="guidance-price">
                <span>10.40万</span>
                <a href="//j.autohome.com.cn/pc/carcounter
            </p>
        </div>

        <div class="spec-guidance">
            <p class="guidance-price">
                <span><span>暂无</span></span>
        """
        specGuidanceDoc = curDdDoc.find("div[class='spec-gu
        # print("specGuidanceDoc=%s" % specGuidanceDoc)
        guidancePriceSpanDoc = specGuidanceDoc.find("p[clas
        # print("guidancePriceSpanDoc=%s" % guidancePriceSp
        carModelMsrp = guidancePriceSpanDoc.text()
        print("carModelMsrp=%s" % carModelMsrp)
        curDdCarDict["carModelMsrp"] = carModelMsrp

        self.processSingleResult(curDdCarDict)

    @catch_status_code_error
    def haltCarSpecCallback(self, response):
        prevCarModelDict = response.save
        carModelDict = copy.deepcopy(prevCarModelDict)
        print("carModelDict=%s" % carModelDict)

        respJson = response.json
        print("respJson=%s" % respJson)

        """
        [
            {
                "name": "1.5升 涡轮增压 169马力",
                "speclist": [
                    {
                        "specid": 36955,
                        "specname": "2019款 红标 1.5GDIT 自动
                        "specstate": 40,
                        "minprice": 102000,
                        "maxprice": 102000,
                        "fueltype": 1,
                        "fueltypedetail": 1,
                        "driveform": "前置前驱",
                        "drivetype": "前驱",
                        "gearbox": "7挡双离合",
                        "evflag": "",
                        "newcarflag": "",
                        "subsidy": "",
```

```
                    "paramisshow": 1,
                    "videoid": 0,
                    "link2sc": "http://www.che168.com/c
                    "price2sc": "7.58万",
                    "price": "10.20万",
                    "syear": 2019
                }, {
                    "specid": 36956,
                    "specname": "2019款 红标 1.5GDIT 自动
                    "specstate": 40,
                    "minprice": 109000,
                    "maxprice": 109000,
                    "fueltype": 1,
                    "fueltypedetail": 1,
                    "driveform": "前置前驱",
                    "drivetype": "前驱",
                    "gearbox": "7挡双离合",
                    "evflag": "",
                    "newcarflag": "",
                    "subsidy": "",
                    "paramisshow": 1,
                    "videoid": 0,
                    "link2sc": "",
                    "price2sc": "",
                    "price": "10.90万",
                    "syear": 2019
                },
                ...
        """
        if respJson:
            for eachModelGroupDict in respJson:
                modelGroupName = eachModelGroupDict["name"]
                modelSpecList = eachModelGroupDict["speclis
                for eachModelDict in modelSpecList:
                    curCarModelDict = copy.deepcopy(carMode

                    carModelYear = "%s款" % eachModelDict["
                    # carModelSpecUrl = "%s/%s" % (CarSpecH
                    carModelSpecUrl = self.genSpecUrl(eachM

                    curCarModelDict["carModelGroupName"] =
                    curCarModelDict["carModelYear"] = carMo
                    # curCarModelDict["carModelEnvStandard"
                    # curCarModelDict["carModelPower"] = ""
                    curCarModelDict["carModelDriveType"] =
                    curCarModelDict["carModelGearBox"] = ea
                    curCarModelDict["carModelName"] = eachM
                    curCarModelDict["carModelSpecUrl"] = ca
                    curCarModelDict["carModelMsrp"] = eachM

                    self.processSingleResult(curCarModelDic
```

```
    @catch_status_code_error
    def processCarSpecConfig(self, curCarModelDict):
        print("in processCarSpecConfig")
        carModelDict = copy.deepcopy(curCarModelDict)
        print("carModelDict=%s" % carModelDict)
        carModelSpecUrl = carModelDict["carModelSpecUrl"]
        print("carModelSpecUrl=%s" % carModelSpecUrl)
        carModelSpecId = self.extractSpecId(carModelSpecUr
        print("carModelSpecId=%s" % carModelSpecId)
        carModelDict["carModelSpecId"] = carModelSpecId # 
        carConfigSpecUrl = self.genConfigSpecUrl(carModelSp
        # https://car.autohome.com.cn/config/spec/43593.htm
        print("carConfigSpecUrl=%s" % carConfigSpecUrl)

        self.crawl(carConfigSpecUrl,
            # fetch_type="js",
            callback=self.carConfigSpecCallback,
            save=carModelDict,
        )

    def getItemFirstValue(self, inputContent, itemIndex):
        print("in getItemFirstValue")
        # firstItemValue = self.extractTrFirstTdValue(input
        firstItemValue = self.extractDictListFirstValue(inp
        return firstItemValue

    def extractDictListFirstValue(self, paramItemDictList,
        """
        [
            ...,
            {
                "id": 1149,
                "name": "能源类型",
                "pnid": "1_-1",
                "valueitems": [{
                    "specid": 39893,
                    "value": "纯电动"
                }, {
                    "specid": 42875,
                    "value": "纯电动"
                }]
            }
            ...,
            {
                "id": 1292,
                "name": "<span class='hs_kw39_configpl'></s
                "pnid": "1_-1",
                "valueitems": [{
                    "specid": 39893,
                    "value": "0.6"
```

```
                }, {
                    "specid": 42875,
                    "value": "0.6"
                }]
            },
            ...
            ,
            {
                "id": 1255,
                "name": "整车<span class='hs_kw36_configpl'
                "pnid": "1_-1",
                "valueitems": [{
                    "specid": 39893,
                    "value": "三<span class='hs_kw7_configp
                }, {
                    "specid": 42875,
                    "value": "三<span class='hs_kw7_configp
                }]
            }
        ]
        """
        paramItemDict = paramItemDictList[itemIndex]
        print("paramItemDict=%s" % paramItemDict)
        firstItemValue = self.extractValueItemsValue(paramI
        print("firstItemValue=%s" % firstItemValue)
        return firstItemValue

    def extractValueItemsValue(self, curItemDict, valueInde
        """
        {
            "id": 0,
            "name": "上市<span class='hs_kw51_configrI'></s
            "pnid": "1_-1",
            "valueitems": [{
                "specid": 1006466,
                "value": "2018.05"
            }, {
                "specid": 1006465,
                "value": "2018.05"
            }, {
                "specid": 1006467,
                "value": "2018.05"
            }]
        }
        """
        valueItemList = curItemDict["valueitems"]
        print("valueItemList=%s" % valueItemList)
        # firstItemDict = valueItemList[0]
        specificItemDict = valueItemList[valueIndex]
        print("specificItemDict=%s" % specificItemDict)
        # specificItemDict={'specid': 43593, 'value': "<spa
```

```
        specificItemValue = specificItemDict["value"]
        # specificItemValue=<span class='hs_kw57_configxt'>
        print("specificItemValue=%s" % specificItemValue)
        return specificItemValue


    # def extractTrFirstTdValue(self, rootDoc, trNumber, is
    def extractTrFirstTdValue(self, rootDoc, trNumber):
        """
        <tr data-pnid="1_-1" id="tr_2">
            <th>
                <div id="1149"><a href="https://car.autohor
                </div>
            </th>
            <td style="background:#F0F3F8;">
                <div>纯电动</div>
            </td>

        <tr data-pnid="1_-1" id="tr_3">
            <th>
                <div id="0">上市<span class="hs_kw40_config
            </th>
            <td style="background:#F0F3F8;">
                <div>2019.11</div>
            </td>
            <td>
                <div>2019.11</div>
            </td>
            <td>
                <div></div>
            </td>
            <td>
                <div></div>
            </td>
        </tr>

        <tr data-pnid="1_-1" id="tr_20" style="background:
            <th>
                <div id="1255"><a href="https://car.autohor
                    class="hs_kw36_configaJ"></span></a></d
            </th>
            <td style="background:#F0F3F8;">
                <div>三<span class="hs_kw7_configaJ"></span
            </td>
            <td>
                <div>三<span class="hs_kw7_configaJ"></span
            </td>
            <td>
                <div></div>
            </td>
            <td>
```

```
                <div></div>
            </td>
        </tr>
        """
        trQuery = "tr[id='tr_%s']" % trNumber
        # print("trQuery=%s" % trQuery)
        trDoc = rootDoc.find(trQuery)
        # print("trDoc=%s" % trDoc)
        tdDocGenerator = trDoc.items("td")
        # print("tdDocGenerator=%s" % tdDocGenerator)
        tdDocList = list(tdDocGenerator)
        # print("tdDocList=%s" % tdDocList)
        firstTdDoc = tdDocList[0]
        # print("firstTdDoc=%s" % firstTdDoc)
        firstTdDivDoc = firstTdDoc.find("div")
        print("firstTdDivDoc=%s" % firstTdDivDoc)
        # if isRespDoc:
        #     respItem = firstTdDivDoc
        # else:
        #     firstItemValue = firstTdDivDoc.text()
        #     respItem = firstItemValue
        # print("respItem=%s" % respItem)
        # return respItem
        respItemHtml = firstTdDivDoc.html()
        print("respItemHtml=%s" % respItemHtml)
        return respItemHtml

    # def extractWholeWarranty(self, firstDivDoc):
    def extractWholeWarranty(self, firstDivHtml):
        print("in extractWholeWarranty")
        carModelWholeWarranty = ""
        # <div>三<span class="hs_kw7_configxv"></span>10<sp
        # print("firstDivDoc=%s" % firstDivDoc)
        # carModelWholeWarranty = firstDivDoc.text() # 三10
        # firstDivHtml = firstDivDoc.html()
        print("firstDivHtml=%s" % firstDivHtml)
        # 三<span class="hs_kw7_configCC"></span>10<span cl
        # carWholeQualityQuarantee = re.sub("[^<>]+(?P<firs
        foundYearDistance = re.search("(?P<warrantyYear>[^<
        if foundYearDistance:
            warrantyYear = foundYearDistance.group("warrant
            distanceNumber = foundYearDistance.group("dista
            distanceUnit = foundYearDistance.group("distanc
            carModelWholeWarranty = "%s年或%s万%s" % (warrar
        else:
            # special:
            # https://car.autohome.com.cn/config/spec/46700
            # <div>三<span class="hs_kw58_configWh"></span>
            # 三<span class="hs_kw58_configOf"></span>
            foundYearNotLimitDistance = re.search("(?P<warr
            print("foundYearNotLimitDistance=%s" % foundYea
```

```
            if foundYearNotLimitDistance:
                warrantyYear = foundYearNotLimitDistance.gr
                print("warrantyYear=%s" % warrantyYear)
                carModelWholeWarranty = "%s年不限公里" % war
        print("carModelWholeWarranty=%s" % carModelWholeWa
        return carModelWholeWarranty

    def getWholeWarranty(self, inputContent, itemIndex):
        # firstDivDoc = self.getItemFirstValue(inputConten
        # print("firstDivDoc=%s" % firstDivDoc)
        # carModelWholeWarranty = self.extractWholeWarrant
        firstDivDocHtml = self.getItemFirstValue(inputConte
        print("firstDivDocHtml=%s" % firstDivDocHtml)
        carModelWholeWarranty = self.extractWholeWarranty(
        return carModelWholeWarranty

    @catch_status_code_error
    def carConfigSpecCallback(self, response):
        print("in carConfigSpecCallback")
        curCarModelDict = response.save
        print("curCarModelDict=%s" % curCarModelDict)
        carModelDict = copy.deepcopy(curCarModelDict)

        configSpecHtml = response.text
        print("configSpecHtml=%s" % configSpecHtml)
        print("")

        # for debug
        return

        # # config json item index - spec table html item
        # ItemIndexDiff = 2

        # isUseSpecTableHtml = True
        # isUseConfigJson = False
        # valueContent = None
        # energyTypeIdx = 2

        # # Method 1: after run js, extract item value fro
        # """
        # <table class="tbcs" id="tab_0" style="width: 932
        #     <tbody>
        #         <tr>
        #             <th class="cstitle" show="1" pid="tab
        #             <h3><span>基本参数</span></h3>
        #             </th>
        #         </tr>
        #         <tr data-pnid="1_-1" id="tr_0">
        # """
        # tbodyDoc = response.doc("table[id='tab_0'] tbody'
        # print("tbodyDoc=%s" % tbodyDoc)
```

```
            # valueContent = tbodyDoc
            # isUseSpecTableHtml = True
            # isUseConfigJson = False
            # energyTypeIdx = 2

            # Method 2: not run js, extract item value from cor
            # get value from config json
            # var config = {"message" ..... "returncode":"0","
            foundConfigJson = re.search("var\s*config\s*=\s*(?P
            print("foundConfigJson=%s" % foundConfigJson)
            if foundConfigJson:
                configJson = foundConfigJson.group("configJson"
                print("configJson=%s" % configJson)
                # configDict = json.loads(configJson, encoding=
                configDict = json.loads(configJson)
                print("configDict=%s" % configDict)

                # if "result" in configDict:
                configResultDict = configDict["result"]
                print("configResultDict=%s" % configResultDict
                # if "paramtypeitems" in configResultDict:
                paramTypeItemDictList = configResultDict["param
                print("paramTypeItemDictList=%s" % paramTypeIte
                # paramTypeItemNum = len(paramTypeItemDictList)
                # print("paramTypeItemNum=%s" % paramTypeItemNu
                basicParamDict = paramTypeItemDictList[0]
                print("basicParamDict=%s" % basicParamDict)
                basicItemDictList = basicParamDict["paramitems"
                print("basicItemDictList=%s" % basicItemDictLis
                # print("type(basicItemDictList)=%s" % type(bas
                # basicItemNum = len(basicItemDictList)
                # print("basicItemNum=%s" % basicItemNum)

                # valueContent = basicItemDictList
                # isUseSpecTableHtml = False
                # isUseConfigJson = True

                # process each basic parameter
                basicItemDictLen = len(basicItemDictList)
                print("basicItemDictLen=%s" % basicItemDictLen
                for curIdx, eachItemDict in enumerate(basicIter
                    print("[%d] eachItemDict=%s" % (curIdx, eac
                    curItemId = eachItemDict["id"]
                    print("curItemId=%s" % curItemId)
                    curItemName = eachItemDict["name"]
                    print("curItemName=%s" % curItemName)
                    curItemFirstValue = self.extractValueItems\
                    print("curItemFirstValue=%s" % curItemFirst

                    curIdNameKeyMapDict = None
                    if curItemId != 0:
```

```
                curIdNameKeyMapDict = self.findMappingD
            else:
                # id = 0
                foundSpan = re.search("<span", curItemN
                print("foundSpan=%s" % foundSpan)
                isSpecialName = bool(foundSpan)
                print("isSpecialName=%s" % isSpecialNar
                if isSpecialName:
                    # id=0 and contain '<span' special
                    foundSuffixHour = re.search("</spar
                    print("foundSuffixHour=%s" % foundS
                    isSpecialSuffixHour = bool(foundSuf
                    print("isSpecialSuffixHour=%s" % is
                    if isSpecialSuffixHour:
                        prevIsQuickCharge = self.isPrev
                        print("prevIsQuickCharge=%s" %
                        if prevIsQuickCharge:
                            # current is MUST 慢充时间(小
                            curIdNameKeyMapDict = {
                                "id": 0,
                                # "name": "<span class=
                                "name": "慢充时间(小时)",
                                "namePattern": "</span>
                                "key": "carModelSlowCha
                            }

                        if not curIdNameKeyMapDict:
                            prevIsActualTestEnduranceMi
                            print("prevIsActualTestEndu
                            if prevIsActualTestEndurand
                                # current is MUST 实测快
                                curIdNameKeyMapDict = {
                                    "id": 0,
                                    # "name": "<span cl
                                    "name": "实测快充时间
                                    "namePattern": "</s
                                    "key": "carModelAct
                                }

                        if not curIdNameKeyMapDict:
                            prevPrevIsActualTestEnduran
                            print("prevPrevIsActualTest
                            if prevPrevIsActualTestEndu
                                # current is MUST 实测慢
                                curIdNameKeyMapDict = {
                                    "id": 0,
                                    # "name": "<span cl
                                    "name": "实测慢充时间
                                    "namePattern": "</s
                                    "key": "carModelAct
                                }
```

```
                        else:
                            curIdNameKeyMapDict = self.find
                    else:
                        curIdNameKeyMapDict = self.findMapp

                print("curIdNameKeyMapDict=%s" % curIdNameK
                if curIdNameKeyMapDict:
                    curItemKey = curIdNameKeyMapDict["key"]
                    print("curItemKey=%s" % curItemKey)
                    # processedItemValue = self.processSpec
                    processedItemValue = self.processSpecia
                    print("processedItemValue=%s" % process
                    carModelDict[curItemKey] = processedIte
                    print("+++ added %s=%s" % (curItemKey,

            print("after extract all item value: carModelDi

        self.saveSingleResult(carModelDict)

        # if isUseConfigJson:
        #     energyTypeIdx += ItemIndexDiff

        # if valueContent:
        #     self.processDiffEneryTypeCar(carModelDict, va
        # else:
        #     self.saveSingleResult(carModelDict)

    # def processSpecialKeyValue(self, itemKey, itemValue,
    def processSpecialKeyValue(self, itemKey, itemValue, re
        print("in processSpecialKeyValue")
        print("itemKey=%s, itemValue=%s" % (itemKey, itemVa
        if itemKey == "carModelWholeWarranty":
            print("process special carModelWholeWarranty")
            # 整车质保
            # 三<span class='hs_kw5_configJS'></span>10<spa
            itemValue = self.extractWholeWarranty(itemValue
            print("itemValue=%s" % itemValue)
        elif itemKey == "carModelBodyStructure":
            print("process special carModelBodyStructure va
            # (1) https://www.autohome.com.cn/spec/46292/#p
            # 5门7座<span class='hs_kw3_configFS'></span>
            # -> 5门7座MPV
            #
            # (2) https://www.autohome.com.cn/spec/1002900/
            # <span class='hs_kw21_configqk'></span>
            # -> 皮卡
            foundSpan = re.search("(?P<bodySpan><span.+?</s
            print("foundSpan=%s" % foundSpan)
            if foundSpan:
                bodySpan = foundSpan.group("bodySpan")
                print("bodySpan=%s" % bodySpan)
```

```
                # extract body structure
                """
                <div class="filtrate-list filtrate-list-co
                    <span class="title">车身结构: </span>
                    <label class="lbTxt" for="PL2$!{1 - 1}
                        <input type="checkbox" class="selec
                        MPV
                    </label>
                </div>
                """
                # soup = BeautifulSoup(curHtml, "html.parse
                # # print("soup=%s" % soup)
                # # bodyStructureSpanSoup = soup.find(text=
                # # # print("bodyStructureSpanSoup=%s" % bo
                # # # emptySoup = bodyStructureSpanSoup.nex
                # # # print("emptySoup=%s" % emptySoup)
                # # # siblingLabelSoup = emptySoup.next_sib
                # # # print("siblingLabelSoup=%s" % sibling
                # # parentDivSoup = bodyStructureSpanSoup.p
                # # print("parentDivSoup=%s" % parentDivSou
                # # inputSoup = parentDivSoup.find("input",
                # carStructSoup = soup.find("input", attrs=
                # print("carStructSoup=%s" % carStructSoup)
                carStructDoc = response.doc("input[name=car
                print("carStructDoc=%s" % carStructDoc)
                bodyStructureValue = carStructDoc.attr["va
                print("bodyStructureValue=%s" % bodyStructu
                itemValue = itemValue.replace(bodySpan, boo
                print("itemValue=%s" % itemValue)

        return itemValue

    def isPrevItemIsQuickCharge(self, curIdx, itemDictList)
        print("in isPrevItemIsQuickCharge")
        print("curIdx=%s" % curIdx)

        prevIsQuickCharge = False

        if curIdx > 0:
            prevIdx = curIdx - 1
            print("prevIdx=%s" % prevIdx)
            prevItemDict = itemDictList[prevIdx]
            print("prevItemDict=%s" % prevItemDict)
            prevItemId = prevItemDict["id"]
            print("prevItemId=%s" % prevItemId)
            prevItemName = prevItemDict["name"]
            print("prevItemName=%s" % prevItemName)
            """
                "id": 1292,
                # "name": "<span class='hs_kw39_configpl'><
                "name": "快充时间(小时)",
```

```
            """
            QuickChargeItemId = 1292
            if prevItemId == QuickChargeItemId:
                prevIsQuickCharge = True

        print("prevIsQuickCharge=%s" % prevIsQuickCharge)
        return prevIsQuickCharge

    def checkIsActualTestEnduranceMileage(self, prevSomeNur
        print("in checkIsActualTestEnduranceMileage")
        print("prevSomeNum=%s, curIdx=%s" % (prevSomeNum, c

        isActualTestEnduranceMileage = False

        minAllowIdx = prevSomeNum - 1

        if curIdx > minAllowIdx:
            prevSomeIdx = curIdx - prevSomeNum
            print("prevSomeIdx=%s" % prevSomeIdx)
            prevSomeItemDict = itemDictList[prevSomeIdx]
            print("prevSomeItemDict=%s" % prevSomeItemDict)
            prevSomeItemId = prevSomeItemDict["id"]
            print("prevSomeItemId=%s" % prevSomeItemId)
            prevSomeItemName = prevSomeItemDict["name"]
            print("prevSomeItemName=%s" % prevSomeItemName)

            if prevSomeItemId == 0:
                """
                    "id": 0,
                    # "name": "<span class='hs_kw22_configp
                    "name": "实测续航里程(km)",
                    "namePattern": "</span>续航里程\(km\)$",
                    "key": "carModelActualTestEnduranceMile
                """
                foundActualTestEnduranceMileage = re.search
                print("foundActualTestEnduranceMileage=%s"
                if foundActualTestEnduranceMileage:
                    isActualTestEnduranceMileage = True

        print("isActualTestEnduranceMileage=%s" % isActualT
        return isActualTestEnduranceMileage

    def isPrevItemIsActualTestEnduranceMileage(self, curIdx
        print("in isPrevItemIsActualTestEnduranceMileage")
        print("curIdx=%s" % curIdx)
        return self.checkIsActualTestEnduranceMileage(1, cu

    def isPrevPrevItemIsActualTestEnduranceMileage(self, cu
        print("in isPrevPrevItemIsActualTestEnduranceMileag
        print("curIdx=%s" % curIdx)
        return self.checkIsActualTestEnduranceMileage(2, cu
```

```
    def findMappingDict(self, itemId=0, itemName=""):
        foundMapDict = None

        paramIdNameKeyMapDict = [
            # 汽油车 参数
            # https://car.autohome.com.cn/config/spec/41572
            # https://car.autohome.com.cn/config/spec/10064
            {
                "id": 1149,
                "name": "能源类型",
                "key": "carEnergyType",
            }, {
                "id": 1311,
                "name": "环保标准",
                "key": "carModelEnvStandard",
            }, {
                "id": 0,
                # "name": "上市<span class='hs_kw51_configv
                "name": "上市时间",
                "namePattern": "^上市",
                "key": "carModelReleaseTime",
            }, {
                "id": 1185,
                # "name": "<span class='hs_kw40_configvR'><
                "name": "最大功率(kW)",
                "key": "carModelMaxPower",
            }, {
                "id": 1186,
                # "name": "<span class='hs_kw40_configvR'><
                "name": "最大扭矩(N·m)",
                "key": "carModelMaxTorque",
            }, {
                "id": 1150,
                "name": "发动机",
                "key": "carModelEngine",
            }, {
                "id": 1245,
                "name": "变速箱",
                "key": "carModelGearBox",
            }, {
                "id": 1148,
                "name": "长*宽*高(mm)",
                "key": "carModelSize",
            }, {
                "id": 1147,
                "name": "车身结构",
                "key": "carModelBodyStructure",
            }, {
                "id": 1246,
                "name": "最高车速(km/h)",
```

```
            "key": "carModelMaxSpeed",
        }, {
            "id": 1250,
            "name": "官方0-100km/h加速(s)",
            "key": "carModelOfficialSpeedupTime",
        }, {
            "id": 1252,
            # "name": "<span class='hs_kw26_configvR'><
            "name": "实测0-100km/h加速(s)",
            "key": "carModelActualTestSpeedupTime",
        }, {
            "id": 1253,
            # "name": "<span class='hs_kw26_configvR'><
            "name": "实测100-0km/h制动(m)",
            "key": "carModelActualTestBrakeDistance",
        }, {
            "id": 1251,
            # "name": "工信部<span class='hs_kw10_config
            "name": "工信部综合油耗(L/100km)",
            "key": "carModelMiitCompositeFuelConsumptio
        }, {
            "id": 1254,
            # "name": "<span class='hs_kw26_configvR'><
            "name": "实测油耗(L/100km)",
            "key": "carModelActualFuelConsumption",
        }, {
            "id": 1255,
            # "name": "整车<span class='hs_kw73_configv
            "name": "整车质保",
            "key": "carModelWholeWarranty",
        },

        # 电动车 参数
        # https://car.autohome.com.cn/config/spec/39893
        # https://car.autohome.com.cn/config/spec/42875
        {
            "id": 1291,
            "name": "工信部纯电续航里程(km)",
            "key": "carModelMiitEnduranceMileagePureEle
        }, {
            "id": 1292,
            # "name": "<span class='hs_kw39_configpl'><
            "name": "快充时间(小时)",
            "key": "carModelQuickCharge",
        # }, {
        #     "id": 0,
        #     # "name": "<span class='hs_kw10_configpl
        #     "name": "慢充时间(小时)",
        #     "namePattern": "</span>\(小时\)$",
        #     "key": "carModelSlowCharge",
        }, {
```

```
                "id": 0,
                # https://car.autohome.com.cn/config/spec/3
                # {'id': 0, 'name': "<span class='hs_kw39_c
                "name": "快充电量百分比",
                "namePattern": "</span>百分比$",
                "key": "carModelQuickChargePercent",
            }, {
                "id": 0,
                "name": "电动机(Ps)",
                "key": "carModelHorsePowerElectric",
            }, {
                "id": 0,
                # "name": "<span class='hs_kw22_configpl'><
                "name": "实测续航里程(km)",
                "namePattern": "</span>续航里程\(km\)$",
                "key": "carModelActualTestEnduranceMileage"
            # }, {
            #     "id": 0,
            #     # "name": "<span class='hs_kw22_configpl
            #     "name": "实测快充时间(小时)",
            #     "namePattern": "</span>\(小时\)$",
            #     "key": "carModelActualTestQuickCharge",
            # }, {
            #     "id": 0,
            #     # "name": "<span class='hs_kw22_configpl
            #     "name": "实测慢充时间(小时)",
            #     "namePattern": "</span>\(小时\)$",
            #     "key": "carModelActualTestSlowCharge",
            }
        ]

        isItemZero = itemId == 0
        print("isItemZero=%s" % isItemZero)

        foundSpan = re.search("<span", itemName)
        print("foundSpan=%s" % foundSpan)
        isSpecialName = bool(foundSpan)
        print("isSpecialName=%s" % isSpecialName)
        isNotSpecialName = not isSpecialName
        print("isNotSpecialName=%s" % isNotSpecialName)

        if not isItemZero:
            for eachMapDict in paramIdNameKeyMapDict:
                eachItemId = eachMapDict["id"]
                if eachItemId == itemId:
                    foundMapDict = eachMapDict
                    break

        if not foundMapDict:
            if itemName and isNotSpecialName:
                for eachMapDict in paramIdNameKeyMapDict:
```

```
                    eachItemName = eachMapDict["name"]
                    if eachItemName == itemName:
                        foundMapDict = eachMapDict
                        break

        if not foundMapDict:
            if (isItemZero and isSpecialName):
                for eachMapDict in paramIdNameKeyMapDict:
                    if "namePattern" in eachMapDict:
                        eachItemNamePattern = eachMapDict["
                        print("eachItemNamePattern=%s" % ea
                        foundMatchName = re.search(eachIter
                        print("foundMatchName=%s" % foundMa
                        if foundMatchName:
                            foundMapDict = eachMapDict
                            break
        print("foundMapDict=%s from id=%s, name=%s" % (four
        return foundMapDict

    def processDiffEneryTypeCar(self, carModelDict, valueCo
        carEnergyType = self.getItemFirstValue(valueConten
        # 纯电动 / 汽油 / 插电式混合动力 / 油电混合
        carModelDict["carEnergyType"] = carEnergyType

        if carEnergyType == "汽油":
            # https://car.autohome.com.cn/config/spec/4359
            # https://car.autohome.com.cn/config/spec/4157

            # self.processGasolineCar(valueContent, carMode

            # https://car.autohome.com.cn/config/spec/1006

            gasolineCarKeyIdxMapDict = {
                "carModelEnvStandard" : 3,
                "carModelReleaseTime" : 4,
                "carModelMaxPower" : 5,
                "carModelMaxTorque" : 6,
                "carModelEngine" : 7,
                "carModelGearBox" : 8,
                "carModelSize" : 9,
                "carModelBodyStructure" : 10,
                "carModelMaxSpeed" : 11,
                "carModelOfficialSpeedupTime" : 12,
                "carModelActualTestSpeedupTime" : 13,
                "carModelActualTestBrakeDistance" : 14,
                "carModelMiitCompositeFuelConsumption" : 15
                "carModelActualFuelConsumption" : 16,
            }
            wholeWarrantyIdx = 17

            if isUseConfigJson:
```

```
                for eachKey in gasolineCarKeyIdxMapDict.key
                    gasolineCarKeyIdxMapDict[eachKey] += It
                wholeWarrantyIdx += ItemIndexDiff

            self.processSingleEneryTypeCar(gasolineCarKeyId

        elif carEnergyType == "纯电动":
            # https://car.autohome.com.cn/config/spec/42875

            # self.processPureElectricCar(valueContent, car

            pureElectricCarKeyIdxMapDict = {
                "carModelReleaseTime": 3,
                "carModelMiitEnduranceMileagePureElectric":
                "carModelQuickCharge": 5,
                "carModelSlowCharge": 6,
                "carModelQuickChargePercent": 7,
                "carModelMaxPower": 8,
                "carModelMaxTorque": 9,
                "carModelHorsePowerElectric": 10,
                "carModelSize": 11,
                "carModelBodyStructure": 12,
                "carModelMaxSpeed": 13,
                "carModelOfficialSpeedupTime": 14,
                "carModelActualTestSpeedupTime": 15,
                "carModelActualTestBrakeDistance": 16,
                "carModelActualTestEnduranceMileage": 17,
                "carModelActualTestQuickCharge": 18,
                "carModelActualTestSlowCharge": 19,
            }
            wholeWarrantyIdx = 20

            if isUseConfigJson:
                for eachKey in pureElectricCarKeyIdxMapDict
                    pureElectricCarKeyIdxMapDict[eachKey] -
                wholeWarrantyIdx += ItemIndexDiff

            self.processSingleEneryTypeCar(pureElectricCarK

        elif carEnergyType == "插电式混合动力":
            # https://car.autohome.com.cn/config/series/440

            # self.processPhevCar(valueContent, carModelDic

            phevCarKeyIdxMapDict = {
                "carModelEnvStandard": 3,
                "carModelReleaseTime": 4,
                "carModelMiitEnduranceMileagePureElectric":
                "carModelQuickCharge": 6,
                "carModelSlowCharge": 7,
                "carModelQuickChargePercent": 8,
```

```
                "carModelMaxPower": 9,
                "carModelMaxTorque": 10,
                "carModelEngine": 11,
                "carModelHorsePowerElectric": 12,
                "carModelGearBox": 13,
                "carModelSize": 14,
                "carModelBodyStructure": 15,
                "carModelMaxSpeed": 16,
                "carModelOfficialSpeedupTime": 17,
                "carModelActualTestSpeedupTime": 18,
                "carModelActualTestBrakeDistance": 19,
                "carModelActualTestEnduranceMileage": 20,
                "carModelActualTestQuickCharge": 21,
                "carModelActualTestSlowCharge": 22,
                "carModelMiitCompositeFuelConsumption": 23,
                "carModelActualFuelConsumption": 24,
            }
            wholeWarrantyIdx = 25

            if isUseConfigJson:
                for eachKey in phevCarKeyIdxMapDict.keys():
                    phevCarKeyIdxMapDict[eachKey] += ItemIn
                wholeWarrantyIdx += ItemIndexDiff

            self.processSingleEneryTypeCar(phevCarKeyIdxMap

        elif carEnergyType == "油电混合":
            # https://car.autohome.com.cn/config/spec/3550

            # self.processHevCar(valueContent, carModelDict

            hevCarKeyIdxMapDict = {
                "carModelEnvStandard": 3,
                "carModelReleaseTime": 4,
                "carModelMaxPower": 5,
                "carModelMaxTorque": 6,
                "carModelEngine": 7,
                "carModelHorsePowerElectric": 8,
                "carModelGearBox": 9,
                "carModelSize": 10,
                "carModelBodyStructure": 11,
                "carModelMaxSpeed": 12,
                "carModelOfficialSpeedupTime": 13,
                "carModelActualTestSpeedupTime": 14,
                "carModelActualTestBrakeDistance": 15,
                "carModelMiitCompositeFuelConsumption": 16,
                "carModelActualFuelConsumption": 17,
            }
            wholeWarrantyIdx = 18

            if isUseConfigJson:
```

```
                for eachKey in hevCarKeyIdxMapDict.keys():
                    hevCarKeyIdxMapDict[eachKey] += ItemInc
                wholeWarrantyIdx += ItemIndexDiff

            self.processSingleEneryTypeCar(hevCarKeyIdxMapD
        else:
            errMsg = "TODO: add support %s!" % carEnergyTyp
            raise Exception(errMsg)

    def processSingleEneryTypeCar(self, keyIdxMapDict, valu
        keyList = keyIdxMapDict.keys()
        keyListLen = len(keyList)
        print("keyListLen=%s" % keyListLen)
        for eachItemKey in keyList:
            print("eachItemKey=%s" % eachItemKey)
            eachItemIndex = keyIdxMapDict[eachItemKey]
            print("eachItemIndex=%s" % eachItemIndex)
            eachItemValue = self.getItemFirstValue(valueCon
            print("eachItemValue=%s" % eachItemValue)
            carModelDict[eachItemKey] = eachItemValue

        # 整车质保
        carModelWholeWarranty = self.getWholeWarranty(value
        print("carModelWholeWarranty=%s" % carModelWholeWar
        carModelDict["carModelWholeWarranty"] = carModelWho

        self.saveSingleResult(carModelDict)

    # def processGasolineCar(self, valueContent, carModelD
    #     # 汽油

    #     # https://car.autohome.com.cn/config/spec/43593.h
    #     # https://car.autohome.com.cn/config/spec/41572.h

    #     # 环保标准
    #     carModelEnvStandard = self.getItemFirstValue(valu
    #     carModelDict["carModelEnvStandard"] = carModelEnv

    #     # 上市时间
    #     carModelReleaseTime = self.getItemFirstValue(valu
    #     carModelDict["carModelReleaseTime"] = carModelRe

    #     # 最大功率(kW)
    #     carModelMaxPower = self.getItemFirstValue(valueCo
    #     carModelDict["carModelMaxPower"] = carModelMaxPow

    #     # 最大扭矩(N·m)
    #     carModelMaxTorque = self.getItemFirstValue(value
    #     carModelDict["carModelMaxTorque"] = carModelMaxTo

    #     # 发动机
```

```
    #     carModelEngine = self.getItemFirstValue(valueCont
    #     carModelDict["carModelEngine"] = carModelEngine

    #     # 变速箱
    #     carModelGearBox = self.getItemFirstValue(valueCo
    #     carModelDict["carModelGearBox"] = carModelGearBo

    #     # 长*宽*高(mm)
    #     carModelSize = self.getItemFirstValue(valueConte
    #     carModelDict["carModelSize"] = carModelSize

    #     # 车身结构
    #     carModelBodyStructure = self.getItemFirstValue(va
    #     carModelDict["carModelBodyStructure"] = carModelB

    #     # 最高车速(km/h)
    #     carModelMaxSpeed = self.getItemFirstValue(valueCo
    #     carModelDict["carModelMaxSpeed"] = carModelMaxSpe

    #     # 官方0-100km/h加速(s)
    #     carModelOfficialSpeedupTime = self.getItemFirstVa
    #     carModelDict["carModelOfficialSpeedupTime"] = car

    #     # 实测0-100km/h加速(s)
    #     carModelActualTestSpeedupTime = self.getItemFirst
    #     carModelDict["carModelActualTestSpeedupTime"] = c

    #     # 实测100-0km/h制动(m)
    #     carModelActualTestBrakeDistance = self.getItemFi
    #     carModelDict["carModelActualTestBrakeDistance"] =

    #     # 工信部综合油耗(L/100km)
    #     carModelMiitCompositeFuelConsumption = self.getIt
    #     carModelDict["carModelMiitCompositeFuelConsumptio

    #     # 实测油耗(L/100km)
    #     carModelActualFuelConsumption = self.getItemFirst
    #     carModelDict["carModelActualFuelConsumption"] = c

    #     self.saveSingleResult(carModelDict)

    # def processPureElectricCar(self, valueContent, carMod
    #     # 纯电动

    #     # https://car.autohome.com.cn/config/spec/42875.h

    #     # 上市时间
    #     carModelReleaseTime = self.getItemFirstValue(valu
    #     carModelDict["carModelReleaseTime"] = carModelRe

    #     # 工信部纯电续航里程(km)
```

```
#     carModelMiitEnduranceMileagePureElectric = self.g
#     carModelDict["carModelMiitEnduranceMileagePureEle

#     # 快充时间(小时)
#     carModelQuickCharge = self.getItemFirstValue(valu
#     carModelDict["carModelQuickCharge"] = carModelQu

#     # 慢充时间(小时)
#     carModelSlowCharge = self.getItemFirstValue(value
#     carModelDict["carModelSlowCharge"] = carModelSlow

#     # 快充电量百分比
#     carModelQuickChargePercent = self.getItemFirstVa
#     carModelDict["carModelQuickChargePercent"] = carM

#     # 最大功率(kW)
#     carModelMaxPower = self.getItemFirstValue(valueCo
#     carModelDict["carModelMaxPower"] = carModelMaxPow

#     # 最大扭矩(N·m)
#     carModelMaxTorque = self.getItemFirstValue(value(
#     carModelDict["carModelMaxTorque"] = carModelMaxTo

#     # 电动机(Ps)
#     carModelHorsePowerElectric = self.getItemFirstVa
#     carModelDict["carModelHorsePowerElectric"] = carM

#     # 长*宽*高(mm)
#     carModelSize = self.getItemFirstValue(valueConter
#     carModelDict["carModelSize"] = carModelSize

#     # 车身结构
#     carModelBodyStructure = self.getItemFirstValue(va
#     carModelDict["carModelBodyStructure"] = carModelB

#     # 最高车速(km/h)
#     carModelMaxSpeed = self.getItemFirstValue(valueCo
#     carModelDict["carModelMaxSpeed"] = carModelMaxSpe

#     # 官方0-100km/h加速(s)
#     carModelOfficialSpeedupTime = self.getItemFirstVa
#     carModelDict["carModelOfficialSpeedupTime"] = car

#     # 实测0-100km/h加速(s)
#     carModelActualTestSpeedupTime = self.getItemFirst
#     carModelDict["carModelActualTestSpeedupTime"] = c

#     # 实测100-0km/h制动(m)
#     carModelActualTestBrakeDistance = self.getItemFir
#     carModelDict["carModelActualTestBrakeDistance"] =
```

```
        #     # 实测续航里程(km)
        #     carModelActualTestEnduranceMileage = self.getIter
        #     carModelDict["carModelActualTestEnduranceMileage'

        #     # 实测快充时间(小时)
        #     carModelActualTestQuickCharge = self.getItemFirst
        #     carModelDict["carModelActualTestQuickCharge"] = c

        #     # 实测慢充时间(小时)
        #     carModelActualTestSlowCharge = self.getItemFirstV
        #     carModelDict["carModelActualTestSlowCharge"] = ca

        #     # 整车质保
        #     carModelWholeWarranty = self.getWholeWarranty(va
        #     carModelDict["carModelWholeWarranty"] = carModelW

        #     self.saveSingleResult(carModelDict)

        # def processPhevCar(self, valueContent, carModelDict):
        #     # 插电式混合动力 = PHEV = Plug-in Hybrid Electric v

        #     # https://car.autohome.com.cn/config/series/4460.

        #     # 环保标准
        #     carModelEnvStandard = self.getItemFirstValue(valu
        #     carModelDict["carModelEnvStandard"] = carModelEnv

        #     # 上市时间
        #     carModelReleaseTime = self.getItemFirstValue(valu
        #     carModelDict["carModelReleaseTime"] = carModelRe

        #     # 工信部纯电续航里程(km)
        #     carModelMiitEnduranceMileagePureElectric = self.g
        #     carModelDict["carModelMiitEnduranceMileagePureEle

        #     # 快充时间(小时)
        #     carModelQuickCharge = self.getItemFirstValue(valu
        #     carModelDict["carModelQuickCharge"] = carModelQui

        #     # 慢充时间(小时)
        #     carModelSlowCharge = self.getItemFirstValue(value
        #     carModelDict["carModelSlowCharge"] = carModelSlow

        #     # 快充电量百分比
        #     carModelQuickChargePercent = self.getItemFirstVal
        #     carModelDict["carModelQuickChargePercent"] = carM

        #     # 最大功率(kW)
        #     carModelMaxPower = self.getItemFirstValue(valueCo
        #     carModelDict["carModelMaxPower"] = carModelMaxPow
```

```
#     # 最大扭矩(N·m)
#     carModelMaxTorque = self.getItemFirstValue(valueC
#     carModelDict["carModelMaxTorque"] = carModelMaxTo

#     # 发动机
#     carModelEngine = self.getItemFirstValue(valueCont
#     carModelDict["carModelEngine"] = carModelEngine

#     # 电动机(Ps)
#     carModelHorsePowerElectric = self.getItemFirstVa
#     carModelDict["carModelHorsePowerElectric"] = carM

#     # 变速箱
#     carModelGearBox = self.getItemFirstValue(valueCon
#     carModelDict["carModelGearBox"] = carModelGearBox

#     # 长*宽*高(mm)
#     carModelSize = self.getItemFirstValue(valueConten
#     carModelDict["carModelSize"] = carModelSize

#     # 车身结构
#     carModelBodyStructure = self.getItemFirstValue(va
#     carModelDict["carModelBodyStructure"] = carModelB

#     # 最高车速(km/h)
#     carModelMaxSpeed = self.getItemFirstValue(valueCo
#     carModelDict["carModelMaxSpeed"] = carModelMaxSpe

#     # 官方0-100km/h加速(s)
#     carModelOfficialSpeedupTime = self.getItemFirstVa
#     carModelDict["carModelOfficialSpeedupTime"] = car

#     # 实测0-100km/h加速(s)
#     carModelActualTestSpeedupTime = self.getItemFirst
#     carModelDict["carModelActualTestSpeedupTime"] = c

#     # 实测100-0km/h制动(m)
#     carModelActualTestBrakeDistance = self.getItemFir
#     carModelDict["carModelActualTestBrakeDistance"] =

#     # 实测续航里程(km)
#     carModelActualTestEnduranceMileage = self.getIten
#     carModelDict["carModelActualTestEnduranceMileage"

#     # 实测快充时间(小时)
#     carModelActualTestQuickCharge = self.getItemFirst
#     carModelDict["carModelActualTestQuickCharge"] = c

#     # 实测慢充时间(小时)
#     carModelActualTestSlowCharge = self.getItemFirstV
#     carModelDict["carModelActualTestSlowCharge"] = ca
```

```
    #     # 工信部综合油耗(L/100km)
    #     carModelMiitCompositeFuelConsumption = self.getIt
    #     carModelDict["carModelMiitCompositeFuelConsumptio

    #     # 实测油耗(L/100km)
    #     carModelActualFuelConsumption = self.getItemFirst
    #     carModelDict["carModelActualFuelConsumption"] = c

    #     # 整车质保
    #     carModelWholeWarranty = self.getWholeWarranty(va
    #     carModelDict["carModelWholeWarranty"] = carModelW

    #     self.saveSingleResult(carModelDict)

    # def processHevCar(self, valueContent, carModelDict):
    #     # 混合电动汽车=HEV=Hybrid Electric Vehicle

    #     # https://car.autohome.com.cn/config/spec/35507.h

    #     # 环保标准
    #     carModelEnvStandard = self.getItemFirstValue(valu
    #     carModelDict["carModelEnvStandard"] = carModelEnv

    #     # 上市时间
    #     carModelReleaseTime = self.getItemFirstValue(valu
    #     carModelDict["carModelReleaseTime"] = carModelRe

    #     # 最大功率(kW)
    #     carModelMaxPower = self.getItemFirstValue(valueCo
    #     carModelDict["carModelMaxPower"] = carModelMaxPow

    #     # 最大扭矩(N·m)
    #     carModelMaxTorque = self.getItemFirstValue(valueC
    #     carModelDict["carModelMaxTorque"] = carModelMaxTo

    #     # 发动机
    #     carModelEngine = self.getItemFirstValue(valueCont
    #     carModelDict["carModelEngine"] = carModelEngine

    #     # 电动机(Ps)
    #     carModelHorsePowerElectric = self.getItemFirstVa
    #     carModelDict["carModelHorsePowerElectric"] = carM

    #     # 变速箱
    #     carModelGearBox = self.getItemFirstValue(valueCon
    #     carModelDict["carModelGearBox"] = carModelGearBox

    #     # 长*宽*高(mm)
    #     carModelSize = self.getItemFirstValue(valueConten
    #     carModelDict["carModelSize"] = carModelSize
```

```
#     # 车身结构
#     carModelBodyStructure = self.getItemFirstValue(va
#     carModelDict["carModelBodyStructure"] = carModelB

#     # 最高车速(km/h)
#     carModelMaxSpeed = self.getItemFirstValue(valueCo
#     carModelDict["carModelMaxSpeed"] = carModelMaxSpe

#     # 官方0-100km/h加速(s)
#     carModelOfficialSpeedupTime = self.getItemFirstVa
#     carModelDict["carModelOfficialSpeedupTime"] = car

#     # 实测0-100km/h加速(s)
#     carModelActualTestSpeedupTime = self.getItemFirst
#     carModelDict["carModelActualTestSpeedupTime"] = c

#     # 实测100-0km/h制动(m)
#     carModelActualTestBrakeDistance = self.getItemFir
#     carModelDict["carModelActualTestBrakeDistance"] =

#     # 工信部综合油耗(L/100km)
#     carModelMiitCompositeFuelConsumption = self.getIt
#     carModelDict["carModelMiitCompositeFuelConsumptio

#     # 实测油耗(L/100km)
#     carModelActualFuelConsumption = self.getItemFirst
#     carModelDict["carModelActualFuelConsumption"] = c

#     # 整车质保
#     carModelWholeWarranty = self.getWholeWarranty(va
#     carModelDict["carModelWholeWarranty"] = carModelW

#     self.saveSingleResult(carModelDict)

def processSingleResult(self, curCarModelDict):
    print("in processSingleResult")
    self.processCarSpecConfig(curCarModelDict)

    # self.saveSingleResult(curCarModelDict)

def saveSingleResult(self, curCarModelDict):
    carModelDict = copy.deepcopy(curCarModelDict)
    # print("before filter: carModelDict=%s" % carMode
    # process
    for curKey, curValue in carModelDict.items():
        print("curKey=%s, curValue=%s" % (curKey, curVa
        if curValue is None:
            carModelDict[curKey] = ""
        elif curValue == "-":
            # '-' -> ''
```

```
                carModelDict[curKey] = ""
            elif "暂无" in curValue:
                # '暂无 暂无 暂无', '暂无' -> ''
                carModelDict[curKey] = ""
        # print("after filter: carModelDict=%s" % carModelD
        """
        插电式混合动力
            https://www.autohome.com.cn/spec/37077/
                {
                    "carBrandId": "33",
                    "carBrandLogoUrl": "https://car2.autoi
                    "carBrandName": "奥迪",
                    "carEnergyType": "插电式混合动力",
                    "carMerchantName": "奥迪(进口)",
                    "carMerchantUrl": "https://car.autohom
                    "carModelActualFuelConsumption": "",
                    "carModelActualTestBrakeDistance": "",
                    "carModelActualTestEnduranceMileage": "
                    "carModelActualTestQuickCharge": "",
                    "carModelActualTestSlowCharge": "",
                    "carModelActualTestSpeedupTime": "",
                    "carModelBodyStructure": "5门5座SUV",
                    "carModelDataSift2": "国V",
                    "carModelDataSift3": "2.0T",
                    "carModelDataSift4": "8挡手自一体",
                    "carModelDriveType": "前置四驱",
                    "carModelEngine": "2.0T 252马力 L4",
                    "carModelEnvStandard": "国V",
                    "carModelGearBox": "8挡手自一体",
                    "carModelGroupName": "2.0升 涡轮增压 252
                    "carModelHorsePowerElectric": "128",
                    "carModelMaxPower": "270",
                    "carModelMaxSpeed": "228",
                    "carModelMaxTorque": "700",
                    "carModelMiitCompositeFuelConsumption":
                    "carModelMiitEnduranceMileagePureElectr
                    "carModelMsrp": "79.08万",
                    "carModelName": "2019款 55 e-tron",
                    "carModelOfficialSpeedupTime": "5.9",
                    "carModelQuickCharge": "2.5",
                    "carModelQuickChargePercent": "",
                    "carModelReleaseTime": "2018.11",
                    "carModelSize": "5071*1968*1716",
                    "carModelSlowCharge": "10.8",
                    "carModelSpecId": "37077",
                    "carModelSpecUrl": "https://www.autohor
                    "carModelWholeWarranty": "三年或10万公里"
                    "carModelYear": "2019款",
                    "carSeriesId": "4460",
                    "carSeriesLevelId": "19",
                    "carSeriesLevelName": "中大型SUV",
```

```
                    "carSeriesMainImgUrl": "https://car3.au
                    "carSeriesMaxPrice": "79.08万",
                    "carSeriesMinPrice": "79.08万",
                    "carSeriesMsrp": "79.08-79.08万",
                    "carSeriesMsrpUrl": "https://www.autoho
                    "carSeriesName": "奥迪Q7新能源",
                    "carSeriesUrl": "https://www.autohome.
                }

        汽油
            https://www.autohome.com.cn/spec/43593/
                {
                    "carBrandId": "33",
                    "carBrandLogoUrl": "https://car2.autoi
                    "carBrandName": "奥迪",
                    "carEnergyType": "汽油",
                    "carMerchantName": "一汽-大众奥迪",
                    "carMerchantUrl": "https://car.autohome
                    "carModelActualFuelConsumption": "",
                    "carModelActualTestBrakeDistance": "",
                    "carModelActualTestSpeedupTime": "",
                    "carModelBodyStructure": "5门5座两厢车",
                    "carModelDataSift2": "国VI",
                    "carModelDataSift3": "1.4T",
                    "carModelDataSift4": "7挡双离合",
                    "carModelDriveType": "前置前驱",
                    "carModelEngine": "1.4T 150马力 L4",
                    "carModelEnvStandard": "国VI",
                    "carModelGearBox": "7挡双离合",
                    "carModelGroupName": "1.4升 涡轮增压 150
                    "carModelMaxPower": "110",
                    "carModelMaxSpeed": "200",
                    "carModelMaxTorque": "250",
                    "carModelMiitCompositeFuelConsumption":
                    "carModelMsrp": "19.32万",
                    "carModelName": "2020款 改款 Sportback
                    "carModelOfficialSpeedupTime": "8.4",
                    "carModelReleaseTime": "2020.04",
                    "carModelSize": "4312*1785*1426",
                    "carModelSpecId": "43593",
                    "carModelSpecUrl": "https://www.autohor
                    "carModelWholeWarranty": "三年或10万公里"
                    "carModelYear": "2020款",
                    "carSeriesId": "3170",
                    "carSeriesLevelId": "3",
                    "carSeriesLevelName": "紧凑型车",
                    "carSeriesMainImgUrl": "https://car3.au
                    "carSeriesMaxPrice": "23.46万",
                    "carSeriesMinPrice": "19.32万",
                    "carSeriesMsrp": "19.32-23.46万",
                    "carSeriesMsrpUrl": "https://www.autoho
```

```
                    "carSeriesName": "奥迪A3",
                    "carSeriesUrl": "https://www.autohome.
                }

            https://car.autohome.com.cn/config/spec/16147.
                {
                    "carBrandId": "91",
                    "carBrandLogoUrl": "https://car3.autoi
                    "carBrandName": "红旗",
                    "carEnergyType": "汽油",
                    "carMerchantName": "一汽红旗",
                    "carMerchantUrl": "https://car.autohom
                    "carModelActualFuelConsumption": "",
                    "carModelActualTestBrakeDistance": "",
                    "carModelActualTestSpeedupTime": "",
                    "carModelBodyStructure": "4门5座三厢车",
                    "carModelDriveType": "后驱",
                    "carModelEngine": "2.0T 204马力 L4",
                    "carModelEnvStandard": "国IV(国V)",
                    "carModelGearBox": "6挡手自一体",
                    "carModelGroupName": "2.0升 涡轮增压 204
                    "carModelMaxPower": "150",
                    "carModelMaxSpeed": "",
                    "carModelMaxTorque": "260",
                    "carModelMiitCompositeFuelConsumption"
                    "carModelMsrp": "37.98万",
                    "carModelName": "2013款 2.0T 尊贵型",
                    "carModelOfficialSpeedupTime": "",
                    "carModelReleaseTime": "2013.05",
                    "carModelSize": "5095*1875*1485",
                    "carModelSpecId": "16147",
                    "carModelSpecUrl": "https://www.autoho
                    "carModelWholeWarranty": "四年或10万公里"
                    "carModelYear": "2013款",
                    "carSeriesId": "2771",
                    "carSeriesLevelId": "5",
                    "carSeriesLevelName": "中大型车",
                    "carSeriesMainImgUrl": "https://car3.au
                    "carSeriesMaxPrice": "31.78万",
                    "carSeriesMinPrice": "25.28万",
                    "carSeriesMsrp": "25.28-31.78万",
                    "carSeriesMsrpUrl": "https://www.autoh
                    "carSeriesName": "红旗H7",
                    "carSeriesUrl": "https://www.autohome.
                }

            https://www.autohome.com.cn/spec/46144/
            {
                "carBrandId": "36",
                "carBrandLogoUrl": "https://car3.autoimg.cn
                "carBrandName": "奔驰",
```

```
                "carEnergyType": "汽油",
                "carMerchantName": "北京奔驰",
                "carMerchantUrl": "https://car.autohome.co
                "carModelActualFuelConsumption": "",
                "carModelActualTestBrakeDistance": "39.01",
                "carModelActualTestEnduranceMileage": "",
                "carModelActualTestQuickCharge": "",
                "carModelActualTestSlowCharge": "",
                "carModelActualTestSpeedupTime": "9.01",
                "carModelBodyStructure": "5门5座SUV",
                "carModelDataSift2": "国VI",
                "carModelDataSift3": "1.3T",
                "carModelDataSift4": "7挡双离合",
                "carModelDriveType": "前置前驱",
                "carModelEngine": "1.3T 163马力 L4",
                "carModelEnvStandard": "国VI",
                "carModelGearBox": "7挡双离合",
                "carModelGroupName": "1.3升 涡轮增压 163马力
                "carModelHorsePowerElectric": "",
                "carModelMaxPower": "120",
                "carModelMaxSpeed": "207",
                "carModelMaxTorque": "250",
                "carModelMiitCompositeFuelConsumption": "6.
                "carModelMiitEnduranceMileagePureElectric":
                "carModelMsrp": "30.38万",
                "carModelName": "2020款 GLA 200",
                "carModelOfficialSpeedupTime": "9",
                "carModelQuickCharge": "",
                "carModelQuickChargePercent": "",
                "carModelReleaseTime": "2020.07",
                "carModelSize": "4417*1834*1610",
                "carModelSlowCharge": "",
                "carModelSpecId": "46144",
                "carModelSpecUrl": "https://www.autohome.co
                "carModelWholeWarranty": "三年不限公里",
                "carModelYear": "2020款",
                "carSeriesId": "3248",
                "carSeriesLevelId": "17",
                "carSeriesLevelName": "紧凑型SUV",
                "carSeriesMainImgUrl": "https://car3.autoi
                "carSeriesMaxPrice": "30.38万",
                "carSeriesMinPrice": "30.38万",
                "carSeriesMsrp": "30.38-30.38万",
                "carSeriesMsrpUrl": "https://www.autohome.c
                "carSeriesName": "奔驰GLA",
                "carSeriesUrl": "https://www.autohome.com.c
            }

        纯电动
            https://www.autohome.com.cn/spec/42875/
                {
```

```
                    "carBrandId": "33",
                    "carBrandLogoUrl": "https://car2.autoi
                    "carBrandName": "奥迪",
                    "carEnergyType": "纯电动",
                    "carMerchantName": "一汽-大众奥迪",
                    "carMerchantUrl": "https://car.autohom
                    "carModelActualTestBrakeDistance": "",
                    "carModelActualTestEnduranceMileage": "
                    "carModelActualTestQuickCharge": "",
                    "carModelActualTestSlowCharge": "",
                    "carModelActualTestSpeedupTime": "",
                    "carModelBodyStructure": "5门5座SUV",
                    "carModelDataSift2": "100KW",
                    "carModelDataSift3": "265公里",
                    "carModelDataSift4": "单速",
                    "carModelDriveType": "前置前驱",
                    "carModelGearBox": "电动车单速变速箱",
                    "carModelGroupName": "电动 136马力",
                    "carModelHorsePowerElectric": "136",
                    "carModelMaxPower": "100",
                    "carModelMaxSpeed": "150",
                    "carModelMaxTorque": "290",
                    "carModelMiitEnduranceMileagePureElectr
                    "carModelMsrp": "22.68万",
                    "carModelName": "2019款 Q2L e-tron 纯电
                    "carModelOfficialSpeedupTime": "",
                    "carModelQuickCharge": "0.6",
                    "carModelQuickChargePercent": "80",
                    "carModelReleaseTime": "2019.11",
                    "carModelSize": "4237*1785*1548",
                    "carModelSlowCharge": "17",
                    "carModelSpecId": "42875",
                    "carModelSpecUrl": "https://www.autoho
                    "carModelWholeWarranty": "三年或10万公里"
                    "carModelYear": "2019款",
                    "carSeriesId": "5240",
                    "carSeriesLevelId": "16",
                    "carSeriesLevelName": "小型SUV",
                    "carSeriesMainImgUrl": "https://car2.au
                    "carSeriesMaxPrice": "23.73万",
                    "carSeriesMinPrice": "22.68万",
                    "carSeriesMsrp": "22.68-23.73万",
                    "carSeriesMsrpUrl": "https://www.autoh
                    "carSeriesName": "奥迪Q2L e-tron",
                    "carSeriesUrl": "https://www.autohome.
                }

            油电混合
                https://car.autohome.com.cn/config/spec/35507.
                    {
                        "carBrandId": "52",
```

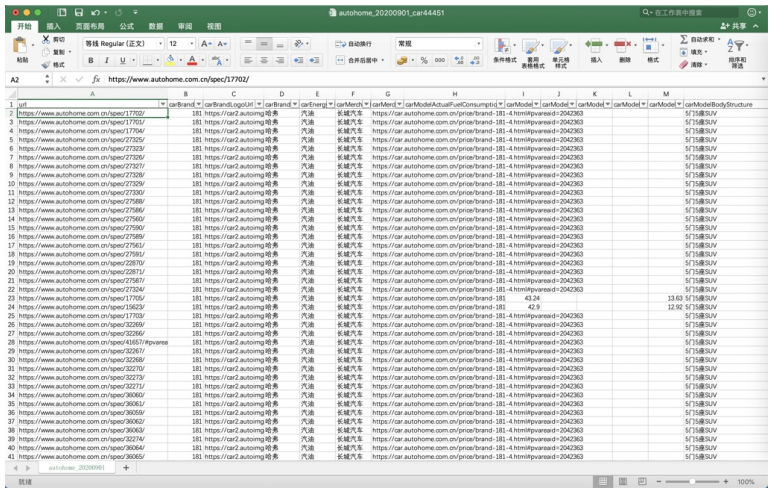
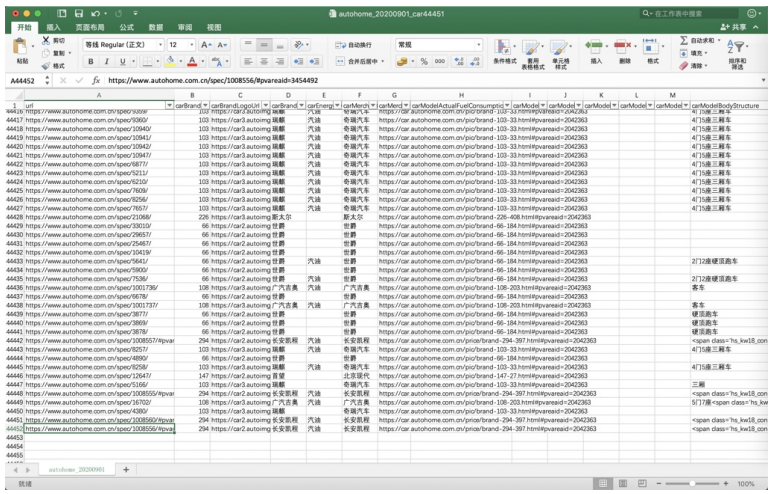
```
                    "carBrandLogoUrl": "https://car2.autoi
                    "carBrandName": "雷克萨斯",
                    "carEnergyType": "油电混合",
                    "carMerchantName": "雷克萨斯",
                    "carMerchantUrl": "https://car.autohom
                    "carModelActualFuelConsumption": "",
                    "carModelActualTestBrakeDistance": "",
                    "carModelActualTestEnduranceMileage": "
                    "carModelActualTestQuickCharge": "",
                    "carModelActualTestSlowCharge": "",
                    "carModelActualTestSpeedupTime": "",
                    "carModelBodyStructure": "5门5座两厢车",
                    "carModelDataSift2": "",
                    "carModelDataSift3": "",
                    "carModelDataSift4": "",
                    "carModelDriveType": "前驱",
                    "carModelEngine": "1.8L 99马力 L4",
                    "carModelEnvStandard": "国IV(国V)",
                    "carModelGearBox": "E-CVT无级变速",
                    "carModelGroupName": "1.8升 99马力",
                    "carModelHorsePowerElectric": "82",
                    "carModelMaxPower": "100",
                    "carModelMaxSpeed": "",
                    "carModelMaxTorque": "",
                    "carModelMiitCompositeFuelConsumption":
                    "carModelMiitEnduranceMileagePureElectr
                    "carModelMsrp": "26.70万",
                    "carModelName": "2018款 CT200h 多彩生活特
                    "carModelOfficialSpeedupTime": "",
                    "carModelQuickCharge": "",
                    "carModelQuickChargePercent": "",
                    "carModelReleaseTime": "2018.08",
                    "carModelSize": "4360*1765*1455",
                    "carModelSlowCharge": "",
                    "carModelSpecId": "35507",
                    "carModelSpecUrl": "https://www.autohor
                    "carModelWholeWarranty": "六年或15万公里"
                    "carModelYear": "2018款",
                    "carSeriesId": "2063",
                    "carSeriesLevelId": "3",
                    "carSeriesLevelName": "紧凑型车",
                    "carSeriesMainImgUrl": "https://car2.au
                    "carSeriesMaxPrice": "28.20万",
                    "carSeriesMinPrice": "21.50万",
                    "carSeriesMsrp": "21.50-28.20万",
                    "carSeriesMsrpUrl": "https://www.autoho
                    "carSeriesName": "雷克萨斯CT",
                    "carSeriesUrl": "https://www.autohome.c
                }
            """
```

```
        carAllKeyList = [
            "carBrandName",
            "carBrandId",
            "carBrandLogoUrl",
            "carMerchantName",
            "carMerchantUrl",
            "carSeriesId",
            "carSeriesName",
            "carSeriesUrl",
            "carSeriesMsrp",
            "carSeriesMsrpUrl",
            "carSeriesMainImgUrl",
            "carSeriesMinPrice",
            "carSeriesMaxPrice",
            "carSeriesLevelName",
            "carSeriesLevelId",
            "carModelName",
            "carModelSpecUrl",
            "carModelSpecId",
            "carModelMsrp",
            "carModelYear",
            "carModelGearBox",
            "carModelDriveType",
            "carModelBodyStructure",
            "carModelEngine",
            "carModelGroupName",
            "carModelDataSift2",
            "carModelDataSift3",
            "carModelDataSift4",
            "carEnergyType",
            "carModelEnvStandard",
            "carModelReleaseTime",
            "carModelMaxPower",
            "carModelMaxTorque",
            "carModelSize",
            "carModelMaxSpeed",
            "carModelOfficialSpeedupTime",
            "carModelActualTestSpeedupTime",
            "carModelActualTestBrakeDistance",
            "carModelMiitCompositeFuelConsumption",
            "carModelActualFuelConsumption",
            "carModelMiitEnduranceMileagePureElectric",
            "carModelQuickCharge",
            "carModelSlowCharge",
            "carModelQuickChargePercent",
            "carModelHorsePowerElectric",
            "carModelActualTestEnduranceMileage",
            "carModelActualTestQuickCharge",
            "carModelActualTestSlowCharge",
            "carModelWholeWarranty",
        ]
```

```
        for eachCarKey in carAllKeyList:
            if eachCarKey not in carModelDict:
                print("found miss key: %s" % eachCarKey)
                carModelDict[eachCarKey] = ""

        carModelSpecUrl = carModelDict["carModelSpecUrl"]
        self.send_message(self.project_name, carModelDict,

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

输出结果

autohome_20200901_car44451.xlsx

- 在线下载：autohome_20200901_car44451.xlsx
- 截图效果
  - 
  - 

百度热榜

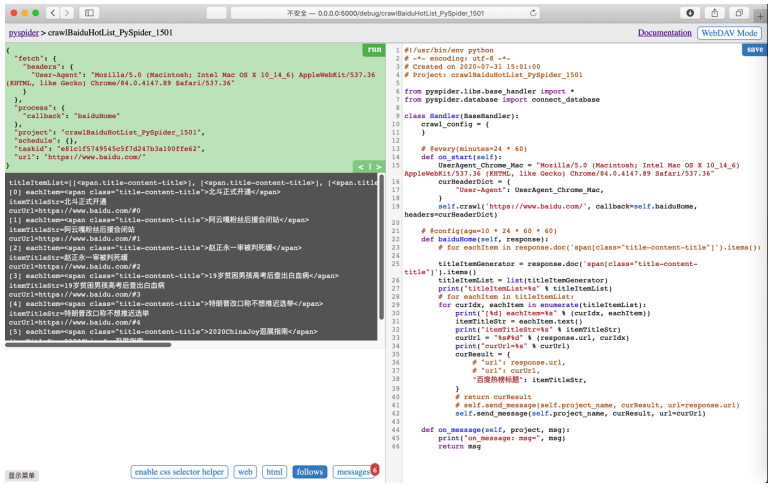
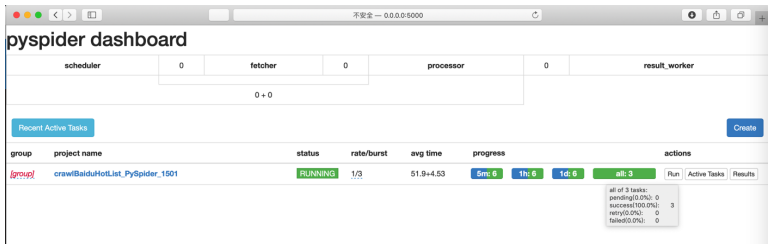
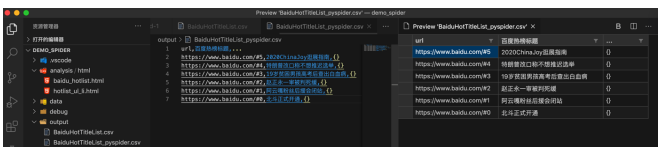
此处以 爬取百度热榜的内容列表 为例，去演示如何使用 `PySpider`

代码 `crawlBaiduHotList_PySpider_1501`

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Created on 2020-07-31 15:01:00
# Project: crawlBaiduHotList_PySpider_1501


from pyspider.libs.base_handler import *
from pyspider.database import connect_database


class Handler(BaseHandler):
    crawl_config = {
    }


    # @every(minutes=24 * 60)
    def on_start(self):
        UserAgent_Chrome_Mac = "Mozilla/5.0 (Macintosh; Int
        curHeaderDict = {
            "User-Agent": UserAgent_Chrome_Mac,
        }
        self.crawl('https://www.baidu.com/', callback=self.

    # @config(age=10 * 24 * 60 * 60)
    def baiduHome(self, response):
        # for eachItem in response.doc('span[class="title-c

        titleItemGenerator = response.doc('span[class="tit
        titleItemList = list(titleItemGenerator)
        print("titleItemList=%s" % titleItemList)
        # for eachItem in titleItemList:
        for curIdx, eachItem in enumerate(titleItemList):
            print("[%d] eachItem=%s" % (curIdx, eachItem))
            itemTitleStr = eachItem.text()
            print("itemTitleStr=%s" % itemTitleStr)
            curUrl = "%s#%d" % (response.url, curIdx)
            print("curUrl=%s" % curUrl)
            curResult = {
                # "url": response.url,
                # "url": curUrl,
                "百度热榜标题": itemTitleStr,
            }
            # return curResult
            # self.send_message(self.project_name, curResu
            self.send_message(self.project_name, curResult,


    def on_message(self, project, msg):
```

```
        print("on_message: msg=", msg)
        return msg
```

效果和结果

- 调试
  - 
- 运行
  - 
- 点击 `Results` -> Results结果
  - 
- 点击 `CSV` -> 导出CSV
  - 
    - CSV预览
    - 

更多细节

详见：

【已解决】用Python爬虫框架PySpider实现爬虫爬取百度热榜内容列表

附录

下面列出相关参考资料。

参考资料

- 【未解决】用Python爬取汽车之家的车型车系详细数据
- 【已解决】用Python爬虫框架PySpider实现爬虫爬取百度热榜内容列表
- 【已解决】PySpider中如何在单个页面返回多个结果保存到自带的Results页面中的列表中
- 【已解决】PySpider抓包百度热榜标题列表结果
- 【已解决】Mac中安装phantomjs
- 【已解决】Mac中启动PySpider
- 【已解决】Mac中pip安装pycurl报错：fatal error openssl/ssl.h file not found
- 【已解决】Mac中给Python3安装PySpider
- 【已解决】CentOS7中安装全局的PySpider并配置和运行
- 【已解决】CentOS中全局的PySpider中如何指定data目录位置
- 【未解决】PySpider运行批量下载时报错：HTTP 599 Operation timed out after milliseconds with out of bytes received
- 【已解决】PySpider无法继续爬取剩余绘本数据
- 【已解决】PySpider中如何更改默认5000端口
-
- 【已解决】写Python爬虫爬取汽车之家品牌车系车型数据 – 在路上
- 【无法解决】PySpider的部署运行而非调试界面上RUN运行
- HTML解析库Python版jQuery：PyQuery
- pyspider是开源强大的python爬虫系统 - pyspider中文网
- 【已解决】PySpider中保存数据到mysql
- 【已解决】PySpider中如何清空之前运行的数据和正在运行的任务 – 在路上
- 【未解决】pyspider中如何给phantomjs传递额外参数 – 在路上
- 【已解决】PySpider中页面部分内容不显示 – 在路上
- 【未解决】pyspider运行出错：FETCH_ERROR HTTP 599 Connection timed out after milliseconds
- 【记录】Mac中安装和运行pyspider
- 【整理】pyspider vs scrapy
- 【已解决】pyspider中phantomjs中的proxy是什么意思 – 在路上
- 【已解决】pyspider中运行result_worker出错：ModuleNotFoundError No module named mysql
- 【已解决】PySpider中传递参数给下一级且当下一级失败时也可以执行
- 【已解决】pyspider中出错：TypeError __init__() got an unexpected keyword argument resultdb – 在路上
- 【已解决】pyspider运行出错：ImportError pycurl libcurl link-time ssl backend (openssl) is different from compile-time ssl backend

(none/other)

- 【已解决】pyspider中pymysql中insert失败且except部分代码没有执行
- 【已解决】pyspider运行出错：Error Could not create web server listening on port 25555 – 在路上
- 【已解决】pyspider中的css选择器不工作 – 在路上
- 【已解决】pyspider中如何写规则去提取网页内容 – 在路上
- 【已解决】PySpider如何把json结果数据保存到csv或excel文件中 – 在路上
- 【已解决】PySpider中如何单个页面返回多个json数据结果 – 在路上
- 【已解决】Mac或Win中用Excel打开UTF8编码的csv文件显示乱码
- 【已解决】pyspider中如何加载汽车之家页面中的更多内容 – 在路上
- 【已解决】PySpider中如何发送POST请求且传递格式为application/x-www-form-urlencoded的form data参数 – 在路上
- 【已解决】PySpider中如何强制让重复的url地址继续爬取 – 在路上
-
- 安装pyspider遇到的坑（python3.6）_盛夏88688的博客-CSDN博客_python 3.6 with报错use async with instead
- How to delete a spider and the data? · Issue #380 · binux/pyspider
- Frequently Asked Questions - pyspider
- python爬虫 - crawl 连接网页超时，HTTP 599 - SegmentFault 思否
-