

目录

前言	1.1
PySpider简介	1.2
PySpider安装与基本用法	1.3
PySpider安装	1.3.1
PySpider基本用法	1.3.2
PySpider的高级用法	1.4
self.crawl	1.4.1
config.json	1.4.2
data目录	1.4.3
phantomjs	1.4.4
PySpider经验与心得	1.5
PySpider的心得	1.5.1
PySpider常见的坑	1.5.2
PySpider案例	1.6
附录	1.7
参考资料	1.7.1

Python爬虫框架： PySpider

- 最新版本： v1.0
- 更新时间： 20200808

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，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

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 admin 艾特 crifan.com，我会尽快删除。谢谢合作。

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。

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-08-08 10:42:51

PySpider简介

PySpider 的基本信息：

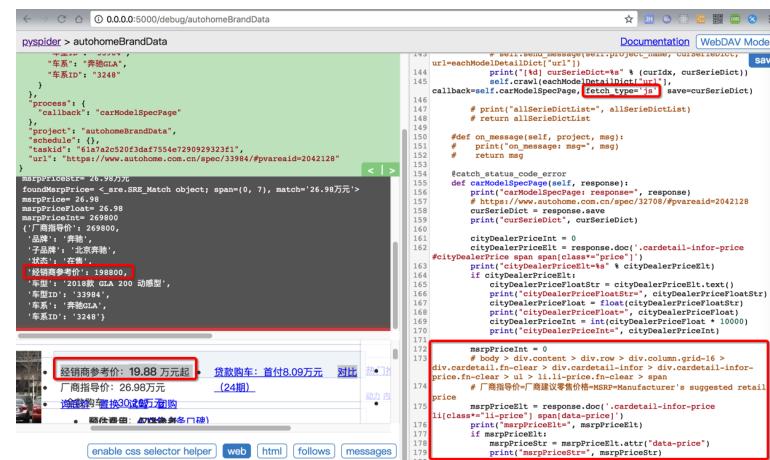
- 是个Python的爬虫框架
 - 最大特点：
 - 带图形界面WebUI的调试
 - 简单易用
 - 同时功能也很强大
 - GitHub
 - <https://github.com/binux/pyspider>
 - 文档：
 - 官方，英文：<http://docs.pyspider.org/>
 - 非官方，中文：<http://www.pyspider.cn/index.html>
 - 作者
 - 网名： Binux
 - 别名：足叉兆虫
 - 博客：[Binuxの杂货铺](#)
 - Github: [binux \(Roy Binux\)](#)

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

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2020-07-30 14:00:04

PySpider安装与基本用法

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-07-30 14:00:04

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
```

即可。

不过，Python的开发中，一般避免不同开发环境的影响，推荐使用虚拟环境工具，比如 pipenv。

此处使用 pipenv 虚拟环境去安装 PySpider：

安装（虚拟环境同时安装）PySpider： `pipenv install pyspider`

即可。

安装期间出现SSL或pycurl错误

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
```

或运行期间出错：

```
|ImportError pycurl libcurl link-time ssl backend (openssl) is different  
|from compile-time ssl backend (none/other)
```

则解决办法是：

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

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)

上述命令要在对应虚拟环境下运行

如果本身你前面是用虚拟环境，比如 `pipenv` 中去安装的pyspider，则上述命令要先去进入虚拟环境：`pipenv shell` 之后再去运行，否则是无效的。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04

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。

注：

- 如果是用虚拟环境安装的PySpider，记得先进去虚拟环境后再运行 PySpider
 - 比如用的 `pipenv`，则是先 `pipenv shell`，再 `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了，此界面一般称为： 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
 - 后续也可以去代码中修改的

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

PySpider安装

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：

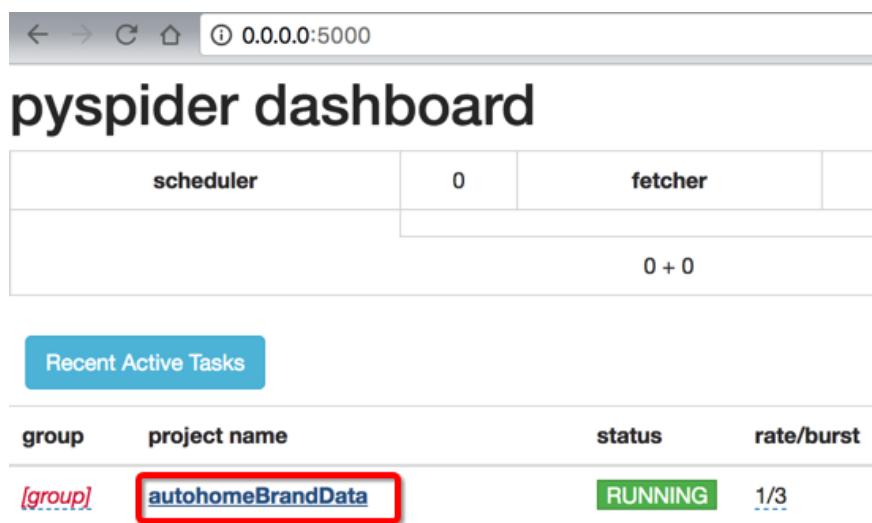
先说之前不熟悉的时候的操作：

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



A screenshot of the PySpider debug interface. The URL in the address bar is 0.0.0.0:5000/debug/autohomeBrandData. The project name 'pyspider' is highlighted with a red box. Below it, the code editor shows a JSON configuration for a task named 'autohomeBrandData'. On the right side, there's a vertical list of numbers from 1 to 15, and a series of orange buttons labeled 'gradCarHtmlPage > https://www.autohome.com.cn/grade/carhtml/a.html', 'gradCarHtmlPage > https://www.autohome.com.cn/grade/carhtml/b.html', and 'gradCarHtmlPage > https://www.autohome.com.cn/grade/carhtml/c.html', each with a '...', '>', and a green 'run' button.

返回项目列表：



A screenshot of the PySpider dashboard. The URL in the address bar is 0.0.0.0:5000. The main title is 'pyspider dashboard'. Below it, there's a table with columns 'scheduler', '0', 'fetcher', and '(0 + 0)'. Underneath the table, there's a section titled 'Recent Active Tasks' with a table:

group	project name	status	rate/burst
[group]	autohomeBrandData	RUNNING	1/3

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：

PySpider安装

想要返回上一级的爬取函数的话，点击 左箭头

0.0.0.0:5000/debug/autohomeBrandData

Documentation [WebDAV Mode]

```
pySpider > autohomeBrandData
```

run

```
{ "fetch": {}, "process": { "callback": "detail_page" }, "project": "autohomeBrandData", "schedule": { "priority": 2 }, "taskid": "42b58d9268103ddc77ded3185be0a8c", "url": "https://car.autohome.com.cn/pic/series-a25858.html#pvareaid=103448" } < > | status: SUCCESS
```

fnRightPicSeries = <a href="https://car.autohome.com.cn/pic/series-t-2567.html" fullSeriesUrl="https://car.autohome.com.cn/pic/series-t-2567.html" eachDict = { "text": "汽车图片", "href": "https://car.autohome.com.cn/pic/" } eachDict = { "text": "长安", "href": "https://car.autohome.com.cn/pic/brand" } eachDict = { "text": "长安汽车", "href": "https://car.autohome.com.cn/pic/brand" } eachDict = { "text": "悦翔", "href": "https://car.autohome.com.cn/pic/series-a25858.html" } eachDict = { "text": "汽车图片", "href": "https://car.autohome.com.cn/pic/brand" } mainEachDict = { "text": "长安", "href": "https://car.autohome.com.cn/pic/brand" } dictTextList = ["2015款"] groupCount = 1 ddList = [[<u></u>]] ----- [0] 2015款 carModelUrl = https://car.autohome.com.cn/pic/series-a25858/2567.html#pvareaid=103448 carModelName = 1.4L 手动美型 国V

[{ "autohomeBrandData": { "品牌": "长安", "子品系": "长安全车", "车型": "2015款 1.4L 手动美型 国V", "车系": "悦翔" }, "url": "https://car.autohome.com.cn/pic/series-a25858/2567.html#pvareaid=2042220 },] enableSeeSelector(hlNav) [web] [html] follows 1 messages 2 javascript;

```
1 #!/usr/bin/env python
2 # -*- encoding: utf-8 -*-
3 # Created on 2018-04-27 21:53:02
4 # Project: autohomeBrandData
5
6 from pyspider.libs.base_handler import *
7 import string
8
9 class Handler(BaseHandler):
10     crawl_config = {
11         ...
12     }
13
14     #every(minutes=24 * 60)
15     def on_start(self):
16         for eachLetter in list(string.ascii_lowercase):
17             self.crawl("https://www.autohome.com.cn/grade/carhtml/%s.html" % eachLetter, callback=self.gradCarHTMLpage)
18
19     def gradCarHTMLpage(self, response):
20         picSeriesItemlist = response.doc().rank.list.ul.li.div[a[href^="#pic/series-"]].items()
21         print("%s(%s)len(picSeriesItemlist)=%s(%slen(picSeriesItemlist))" % (eachLetter, len(picSeriesItemlist)))
22         for each in response.doc().rank.list.ul.li.div[a[href^="#pic/series-"]].items():
23             self.crawl(each.attr.href, callback=self.detail_page)
24
25     #config(priority=10 * 24 * 60 * 60)
26     def picSeriesPage(self, response):
27         for each in response.doc().rank.list.ul.li.div[a[href^="#car.autohome.com.cn/pic/series-11"].items()]:
28             for each in response.doc().rank.list.ul.li.div[a[href^="#self.series-"]].items():
29                 self.crawl(each.attr.href, callback=self.detail_page)
30
31     #config(priority=2)
32     def detail_page(self, response):
33         fnRightPicSeries = response.doc().search-pic-thar .fn-right
34         a[href^="#rightPicSeries-"] pic/series-t-66.html" >查看停产车型<br/></a>
35         <a class="cmor" href="#pic/series/588.html" >查看在售车型<br/></a>
36         span class="fn-right" >右滑屏</span>
37         fnRightPicSeries = response.doc().search-pic-thar .fn-right
38         a[href^="#rightPicSeries-"] fnRightPicSeries
39         if fnRightPicSeries:
```

然后再点击Run：

```
{
    "fetch": {},
    "process": {
        "callback": "gradCarHtmlPage"
    },
    "project": "autohomeBrandData",
    "schedule": {},
    "taskid": "40aa2a3ad7c7a8973043d60a32df38f0",
    "url": "https://www.autohome.com.cn/grade/carhtml/c.html"
}

fnRightPicSeries= <a href="https://car.autohome.com.cn/pic/series-t/2567.html" target="blank">
fullPicSeriesUrl= https://car.autohome.com.cn/pic/series-t/2567.html
eachADict= {'text': '汽车图片', 'href': 'https://car.autohome.com.cn/pic/'}
eachADict= {'text': '长安', 'href': 'https://car.autohome.com.cn/pic/brand/'}
eachADict= {'text': '长安汽车', 'href': 'https://car.autohome.com.cn/pic/brand/car/'}
eachADict= {'text': '悦翔v3', 'href': 'https://car.autohome.com.cn/pic/series-s/25858/2567.html#pvareaid=2042220'}
aDictList= [{'text': '汽车图片', 'href': 'https://car.autohome.com.cn/pic/'}, {"text": "长安", "href": "https://car.autohome.com.cn/pic/brand/"}, {"text": "长安汽车", "href": "https://car.autohome.com.cn/pic/brand/car/"}, {"text": "悦翔v3", "href": "https://car.autohome.com.cn/pic/series-s/25858/2567.html#pvareaid=2042220"}]
mainBrandDict={'text': '长安', 'href': 'https://car.autohome.com.cn/pic/brand/'}
dtTextList= ['2015款']
groupCount= 1
ddUlEltsList= [<ul>]
-----[0] 2015款
curModelUrl= https://car.autohome.com.cn/pic/series-s/25858/2567.html#pvareaid=2042220
curmodelName= 1.4L 手动美满型 国V
}

[
    [
        {
            "autohomeBrandData": {
                "品牌": "长安",
                "子品牌": "长安汽车",
                "车型": "2015款 1.4L 手动美满型 国V",
                "车系": "悦翔v3"
            },
            "url": "https://car.autohome.com.cn/pic/series-s/25858/2567.html#pvareaid=2042220"
        }
    ]
]
```

enable css selector helper web html follows 1 messages 2

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

```
self.crawl("https://www.autohome.com.cn/grade/carhtml/t/s.html" target="blank")
eachLetter, callback=self.gradCarHtmlPage)
17     def gradCarHtmlPage(self, response):
18         fnRightPicSeries= response.doc('.rank-list-ul li div a[href*='pic/series/']).items()
19         # print(len(picSeriesItemList))
20         for each in picSeriesItemList:
21             self.crawl(each.attr.href, callback=self.detail_page)
22
23     # configuration=10 * 24 * 60 * 60
24     def detail_page(self, response):
25         fnRightPicSeries= response.doc('.rank-list-ul li div a[href*='//car.autohome.com.cn/pic/series/']).items()
26         for each in response.doc('.rank-list-ul li div a[href*='pic/series/']).items():
27             self.crawl(each.attr.href, callback=self.detail_page)
28
29     @config(priority=2)
30     def detail_page(self, response):
31         fnRightPicSeries= response.doc('.rank-list-ul li div a[href*='//car.autohome.com.cn/pic/series/']).items()
32         # print(len(picSeriesItemList))
33         # <a href="/pic/series-t/66.html">查看停产车型<br/></a>
34         # <a class="ckmore" href="/pic/series/588.html">查看在售车型<br/></a>
35         # <span class="fn-right"><br/></span>
36         fnRightPicSeries = response.doc('.search-pic-bar .fn-right a[href*='//car.autohome.com.cn/pic/series/'])
37         print("fnRightPicSeries", fnRightPicSeries)
38         if fnRightPicSeries:
39             # hrefValue = fnRightPicSeries.attr.href
40             # print("hrefValue", hrefValue)
41             # fullPicSeriesUrl = "https://car.autohome.com.cn" +
42             hrefValue
43             fullPicSeriesUrl = fnRightPicSeries.attr.href
44             print("fullPicSeriesUrl", fullPicSeriesUrl)
45             self.crawl(fullPicSeriesUrl, callback=self.detail_page)
46
47         # continue parse brand data
48         # action=parseBrandData
49         # for eachA in response.doc('.breadnav a[href*='/]').items():
50         #     eachADict = {
51             # 'text': eachA.text(),
52             # 'href': eachA.attr.href
53         }
```

而对应的上一级的结果中，也是上一级的callback：

The screenshot shows the PySpider IDE interface. On the left, a list of crawled URLs for 'gradCarHtmlPage' is displayed, each with a red box around it. On the right, the source code for the 'autohomeBrandData' project is shown in a code editor. A red arrow points from one of the URLs in the list to the corresponding line of code in the editor.

```

{
    "process": {
        "callback": "on_start"
    },
    "project": "autohomeBrandData",
    "taskId": "data1:on_start",
    "url": "data1:on_start"
}

```

```

1 #!/usr/bin/env python
2 # -*- encoding: utf-8 -*-
3 # Created on 2016-04-27 21:53:02
4 # Project: autohomeBrandData
5
6 from pyspider.libs.base_handler import *
7 import string
8
9 class Handler(BaseHandler):
10     crawl_config = {
11         ...
12     }
13     # every(minutes=24 * 60)
14     def on_start(self):
15         for eachLetter in list(string.ascii_lowercase):
16             self.crawl("https://www.autohome.com.cn/grade/carhtml/%s.html" % eachLetter, callback=self.gradCarHtmlPage)
17
18     def gradCarHtmlPage(self, response):
19         aList = response.doc('.rank-list-ul li div a[href^="#/pic/series"]').items()
20         print("%s" % len(picSeriesItemList))
21         for each in picSeriesItemList:
22             self.crawl(each.attr.href, callback=self.detail_page)
23
24     # config(page=10 * 24 * 60 * 60)
25     def picSeriesPage(self, response):
26         aList = response.doc('#car.autohome.com.cn/pic/series').items()
27         for each in response.doc('#rank-list-ul li div a[href^="#/car/autohome.com.cn/pic/series/"]').items():
28             self.crawl(each.attr.href, callback=self.detail_page)
29
30     # config(priority=2)
31     def detail_page(self, response):
32         <a href="#">/pic/series-t/66.html</a> 查看停产车型<br/><a href="#">/pic/series/588.html</a> 查看在售车型
33         <span class="fn-right"><br/></span>
34         fnRightPicSeries = response.doc('.search-pic-bar .fn-right a[href^="#/pic/series/"]').items()
35         for each in fnRightPicSeries:
36             print("fnRightPicSeries=", fnRightPicSeries)
37             if fnRightPicSeries:

```

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中选择html中的元素和内容

PySpider中的html的元素的选择，或者说css选择器，默认是用的PyQuery。

确切的说是：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：

想要提取：

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

中的href的值，则：

PyQuery的写法是：.rank-list-ul li div a[href*="/pic/series"]

PySpider的代码是：

```
for each in response.doc('.rank-list-ul li div a[href*="/
  pic/series"]')
```

详见： [【已解决】pyspider中如何写规则去提取网页内容 – 在路上](#)

PyQuery

`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

另外：

- 关于css操作可以参考：[CSS — pyquery 1.2.4 documentation](#)

关于PyQuery，详见另外教程：

[【整理Book】Python心得：PyQuery](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04

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。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04

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.callback)
```

等价于:

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

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.carModelSpecPage)

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

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04

独立的配置文件: config.json

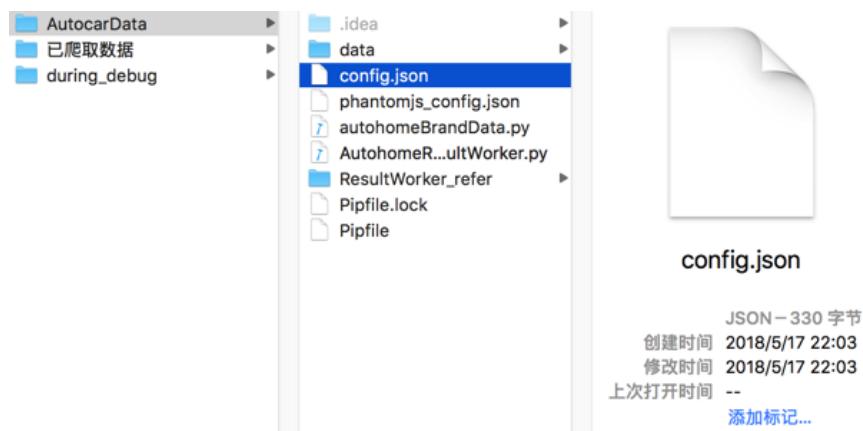
如果需要用到复杂一点的配置，比如 result worker，则是需要单独写配置文件。

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@127.0.0.1:3306/taskdb?charset=utf8",  
    "projectdb": "mysql+projectdb://root:crifan_mysql@127.0.0.1:3306/projectdb?charset=utf8",  
    "resultdb": "mysql+resultdb://root:crifan_mysql@127.0.0.1:3306/resultdb?charset=utf8",  
    "result_worker": {  
        "result_cls": "AutohomeResultWorker.AutohomeResultWorker",  
        "phantomjs": "phantomjs://127.0.0.1:4444"  
    }  
}
```

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



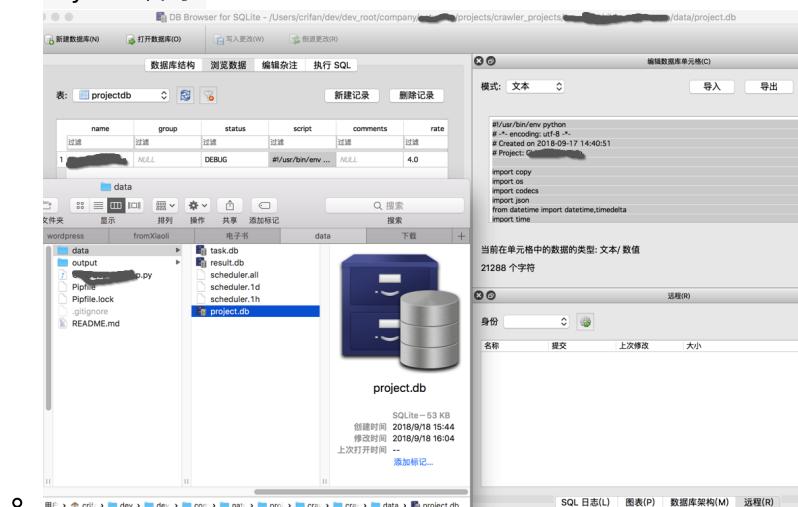
然后去 -c 指定配置文件：

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

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：

The screenshot shows a portion of the PySpider WebUI interface, specifically the 'progress' table. The table has columns for status, rate/burst, avg time, progress, and actions. A row is highlighted with a red box, showing the value '1h: 22020' in the progress column. The 'actions' column contains buttons for Run, Active Tasks, and Results.

status	rate/burst	avg time	progress	actions
TODO	4/10	5m	1h: 22020	Run Active Tasks Results
			1d: all: 1369505	

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。

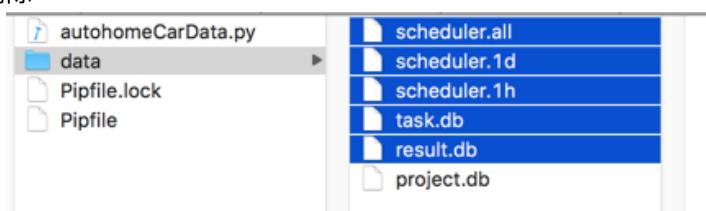
经过一番研究后，发现了解决方案：

- 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

```
180428 10:34:47 scheduler[647] scheduler starting...
180428 10:34:47 scheduler[647] project autohomeBrandData updated, status:DEBUG, paused:False, 0 tasks
180428 10:34:47 scheduler[985] select autohomeBrandData: on_get_info data:., on_get_info
180428 10:34:47 scheduler[586] in Sm: new:0, success:0, retry:0, failed:0
180428 10:34:47 scheduler[782] scheduler.xmlrpc listening on 127.0.0.1:23333
180428 10:34:47 tornado_fetcher[188] [200] autohomeBrandData: on_get_info data:., on_get_info 0s
180428 10:34:47 opp[76] webui running on 0.0.0.0:5000
180428 10:34:47 processor[202] process autohomeBrandData: on_get_info data:., on_get_info
180428 10:34:47 processor[202] process autohomeBrandData: on_get_info data:., on_get_info -> [200] len:12 -> result:None fail:0 msg:0 err:None
180428 10:34:47 scheduler[586] autohomeBrandData on_get_info ['min_tick': 0, 'retry_delay': {}, 'crawl_config': {}]
* [I] 180428 10:35:20 processor[202] processor exiting...
180428 10:35:20 scheduler[663] scheduler exiting...
180428 10:35:20 result_worker[64] result_worker exiting...
180428 10:35:20 tornado_fetcher[671] fetcher exiting...

[Ported]
+ AutoCarData
+ AutoCarData
```

- 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。

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 `phantomjs` 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](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04

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：

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04

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。

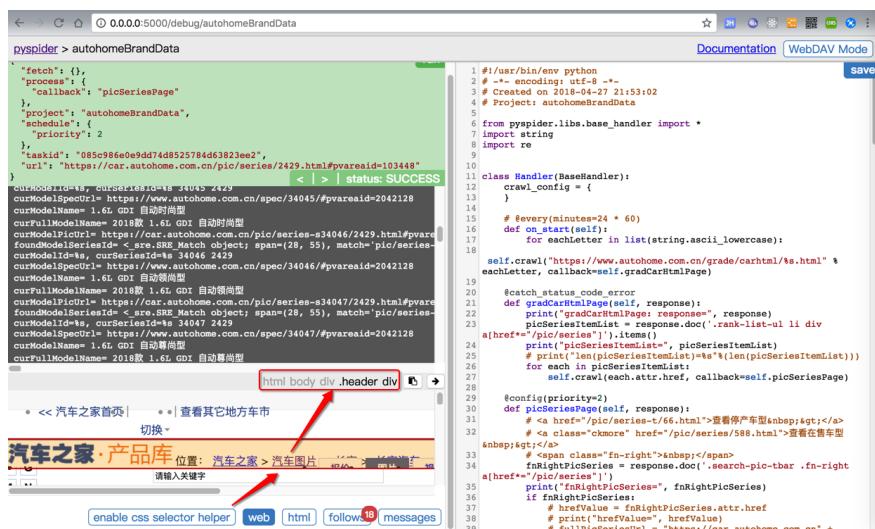
详见： [【已解决】pyspider中如何加载汽车之家页面中的更多内容](#)

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。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新： 2020-07-30 14:00:04

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。

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"
```

The screenshot shows a browser's developer tools with the element selector set to `a`. The element itself is highlighted in blue. In the right-hand panel, the CSS rules for this element are listed, including the rule `a { color: #385998; outline: 0; }`.

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#pvarea:
```

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：

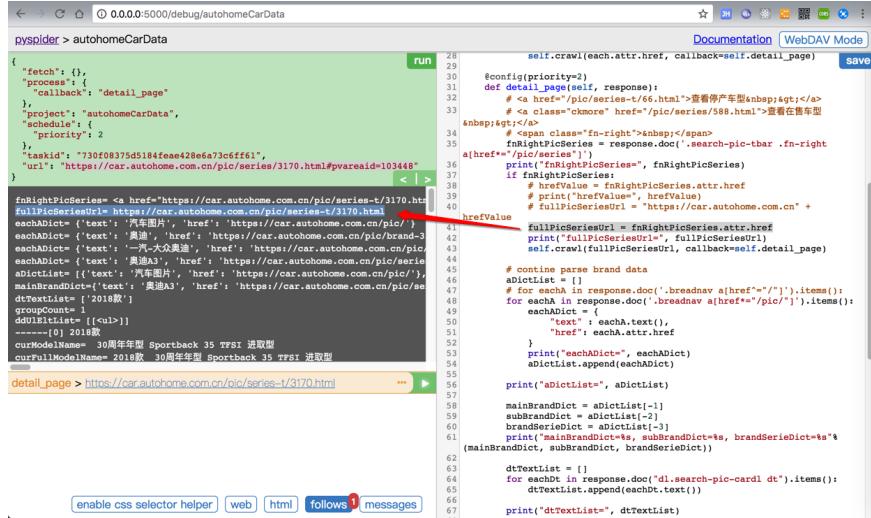
```
a[href*="pic/series/"]
```

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.automhome.com.cn/pic/series-t/
```



```
pySpider > autohomeCarData
{
    "fetch": {},
    "process": {
        "callback": "detail_page"
    },
    "project": "autohomeCarData",
    "schedule": [
        {
            "priority": 2
        }
    ],
    "taskId": "730f08375d5184feae428e6a73c6ff61",
    "url": "https://car.automhome.com.cn/pic/series/3170.html#pvareaid=103448"
}

fnRightPicSeries = <a href="https://car.automhome.com.cn/pic/series-t/3170.htm
fullPicSeriesUrl = https://car.automhome.com.cn/pic/series-t/3170.html
eachADict= {'text': '汽车图片', 'href': 'https://car.automhome.com.cn/pic/'}
eachADict= {'text': '奥迪', 'href': 'https://car.automhome.com.cn/pic/brand-3
eachADict= {'text': '奥迪A3', 'href': 'https://car.automhome.com.cn/pic/series
eachADict= {'text': '奥迪A3', 'href': 'https://car.automhome.com.cn/pic/series
mainBrandDict= {'text': '2018款', 'href': 'https://car.automhome.com.cn/pic/se
dtTextList= ['2018款']
groupCount= 1
ddUlList= [[<ul>]
->[<li>]
->[<li>]
curModelName= 30周年车型 Sportback 35 TFSI 进取型
currFullModelName= 2018款_30周年车型_Sportback 35 TFSI_进取型
detail_page > https://car.automhome.com.cn/pic/series-t/3170.html
```

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
```

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved，
powered by Gitbook最后更新：2020-07-30 14:00:04

PySpider案例

下面把一些之前写过的爬虫分享出来，供参考：

爬取汽车之家的品牌车型车型数据

文件： `autohomeBrandData.py`

```

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

from pyparser.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/gradeList.aspx?letter={}&brandId=0".format(eachLetter))

    @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(picSeriesItemList)))
        for each in picSeriesItemList:
            self.crawl(each.attr.href, callback=self.picSeriesPage)

    @config(priority=2)
    def picSeriesPage(self, response):
        # <a href="/pic/series-t/66.html">查看停产车型 </a>
        # <a class="ckmore" href="/pic/series/588.html">查看全部车型</a>
        # <span class="fn-right">&ampnbsp</span>
        fnRightPicSeries = response.doc('.search-pic-tbar .fn-right')
        print("fnRightPicSeries=", fnRightPicSeries)
        if fnRightPicSeries:
            # hrefValue = fnRightPicSeries.attr.href
            # print("hrefValue=", hrefValue)
            # fullPicSeriesUrl = "https://car.autohome.com.cn/pic/series/588.html"
            fullPicSeriesUrl = fnRightPicSeries.attr.href
            print("fullPicSeriesUrl=", fullPicSeriesUrl)
            self.crawl(fullPicSeriesUrl, callback=self.picBrandDataParse)

    # continue 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
        }
        aDictList.append(eachADict)

```

```

        }

        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-card1 dt"):
            dtTextList.append(eachDt.text())

        print("dtTextList=", dtTextList)

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

        for eachDt in response.doc("dl.search-pic-card1 dt"):
            dtTextList.append(eachDt.text())

        ddUlEltList = []
        for eachDdUlElt in response.doc("dl.search-pic-card1 dd ul"):
            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("li"):
                # 1. model name
                # curModelName = eachLiAElt.text()
                curModelName = eachLiAElt.contents()[0]
                curModelName = curModelName.strip()
                print("curModelName=", curModelName)
                curFullmodelName = curGroupTitle + " " + curModelName
                print("curFullmodelName=", curFullmodelName)

                # 2. model id + carSeriesId + spec url
                curModelId = ""
                curSeriesId = ""
                curModelSpecUrl = ""
                modelSpecUrlTemplate = "https://www.autohome.com.cn/specs/
                curModelPicUrl = eachLiAElt.attr.href
                print("curModelPicUrl=", curModelPicUrl)

```

```

# https://car.autohome.com.cn/pic/series-s:
foundModelSeriesId = re.search("pic/series-s",
                                curModelPic)
print("foundModelSeriesId=", foundModelSeriesId)
if foundModelSeriesId:
    curModelId = foundModelSeriesId.group(1)
    curSeriesId = foundModelSeriesId.group(2)
    print("curModelId=%s, curSeriesId=%s",
          curModelSpecUrl = (modelSpecUrlTemplate % curSeriesId))
    print("curModelSpecUrl=", curModelSpecUrl)

# 3. model status
modelStatus = "在售"
foundStopSale = eachLiAElt.find('i[class*="icon"]')
if foundStopSale:
    modelStatus = "停售"
else:
    foundWseason = eachLiAElt.find('i[class*="icon"]')
    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(modelDetailDictList):
    print("modelDetailDictList[%d] = %s" % (curIdx, eachModelDetailDict))
    print("modelDetailDictList[%d].keys() = %s" % (curIdx, eachModelDetailDict.keys()))
    print("modelDetailDictList[%d].values() = %s" % (curIdx, eachModelDetailDict.values()))
    print("modelDetailDictList[%d].items() = %s" % (curIdx, eachModelDetailDict.items()))

    # insert into mysql
    # define in mysql
    # create table
    CREATE TABLE `tbl_autohome_car_info` (
        `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
        `cityDealerPrice` int(11) unsigned NOT NULL DEFAULT '0',
        `msrpPrice` int(11) unsigned NOT NULL DEFAULT '0',
        `mainBrand` char(20) NOT NULL DEFAULT '',
        `subBrand` varchar(20) NOT NULL DEFAULT '',
        `brandSerie` varchar(20) NOT NULL DEFAULT '',
        `brandSerieId` varchar(15) NOT NULL DEFAULT '',
        `model` varchar(50) NOT NULL DEFAULT '',
        `modelId` varchar(15) NOT NULL DEFAULT '',
        `modelStatus` char(5) NOT NULL DEFAULT ''
    )

```

```

        `url` varchar(200) NOT NULL DEFAULT '' (
    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["brandSerieId"],
    "model": eachModelDetailDict["model"],
    "modelId": eachModelDetailDict["modelId"],
    "modelStatus": eachModelDetailDict["modelStatus"]
}
allSerieDictList.append(curSerieDict)
# print("before send_message: [%d] curSerieDict=%s" % (curIdx, curSerieDict))
# self.send_message(self.project_name, curSerieDict)
print("[%d] curSerieDict=%s" % (curIdx, curSerieDict))
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.automobile.com.cn/spec/32708/#pvareaid=10000000000000000000000000000000
    curSerieDict = response.save
    print("curSerieDict", curSerieDict)

    cityDealerPriceInt = 0
    cityDealerPriceElt = response.doc('.cardetail-info')
    print("cityDealerPriceElt=%s" % cityDealerPriceElt)
    if cityDealerPriceElt:

```

文件: AutohomeResultWorker.py

```

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

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 '自增ID',
    `cityDealerPrice` int(11) unsigned NOT NULL DEFAULT '0' COMMENT '城市经销商价',
    `msrpPrice` int(11) unsigned NOT NULL DEFAULT '0' COMMENT '厂商指导价',
    `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 '车系ID',
    `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, inqueue=%s" % (resultdb, inqueue))
        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 mysql db"""
        # assert task['taskid']
        # assert task['project']
        # assert task['url']

```

```

# assert result
print("AutohomeResultWorker on_result: task=%s, res=%s" % (task, result))
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.defaultTableName)
    print("Create table %s return %s" %(self.defaultTableName, createTableOk))

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

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

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

```

```

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

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

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

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

        createTableSql = CreateTableSqlTemplate % (newTablename)
        print("createTableSql=%s", createTableSql)

        return self.executeSql(sqlStr=createTableSql, action="CREATE")

    def dropTable(self, existedTablename):
        print("dropTable: existedTablename=%s", existedTablename)

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

        return self.executeSql(sqlStr=dropTableSql, action="DROP")

    # 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 the column name
        """
        print("insert: valueDict=%s, tablename=%s" % (valueDict,
                                                       tablename))

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

        keyListSql = ", ".join(self.quoteIdentifier(eachKey)
        print("keyListSql=%s", keyListSql)
        # valueListSql = ", ".join(eachValue for eachValue in
        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="insert")

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

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

    return self.executeSql(sqlStr=deleteSql, actionDesc="delete")

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", dropTableOk)

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

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, tableName=testTablename)
    print("insertOk=", insertOk)

if testDeleteValue:
    toDeleteModelId = "5872"
    deleteOk = mysqlObj.delete(modelId=toDeleteModelId, tableName=testTablename)
    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@127.0.0.1:3306/test?charset=utf8",  
    "result_worker": {  
        "result_cls": "AutohomeResultWorker.AutohomeResultWorker",  
    },  
    "phantomjs-proxy": "127.0.0.1:23450",  
    "phantomjs": {  
        "port": 23450,  
        "auto-restart": true,  
        "load-images": false,  
        "debug": true  
    }  
}
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-07-30 14:00:04

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2020-07-30 14:00:04

参考资料

- [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](#)

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2020-07-30 14:00:04