

# 目录

前言	1.1
爬虫简介	1.2
爬虫的叫法	1.2.1
为何叫爬虫	1.2.2
为何又叫模拟登陆	1.2.2.1
爬虫应用领域	1.2.3
爬虫的核心逻辑	1.3
爬虫的核心流程	1.3.1
爬虫的核心步骤	1.3.2
爬虫的典型实现方式	1.3.3
抓包分析	1.4
抓包分析网页	1.4.1
静态网页	1.4.1.1
动态网页	1.4.1.2
抓包分析app	1.4.2
爬虫框架	1.5
为何需要爬虫框架	1.5.1
常见爬虫框架	1.5.2
如何写爬虫	1.6
用Python写爬虫	1.6.1
用C#写爬虫	1.6.2
用Go写爬虫	1.6.3
用Java写爬虫	1.6.4
用PHP写爬虫	1.6.5
附录	1.7
名词解释	1.7.1
参考资料	1.7.2

# 爬取你要的数据：爬虫技术

- 最新版本: v2.3
- 更新时间: 20200714

## 简介

整理爬虫技术的各种叫法，解释为何叫做爬虫，为何又被叫做模拟登陆，与爬虫有关的一些东西，总结爬虫的核心步骤和阶段，以及每一步的各种细节包括优缺点和其他涉及的内容，继续解释为何要用爬虫框架，总结常见语言的各种爬虫框架，总结实现爬虫的不同方式和爬虫的不同步骤之间的对应关系，用不同语言如何写爬虫，总结爬虫相关名词和概念。

## 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/crawl\\_your\\_data\\_spider\\_technology: 爬取你要的数据：爬虫技术](#)

### 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

- [爬取你要的数据：爬虫技术 book.crifan.com](#)
- [爬取你要的数据：爬虫技术 crifan.github.io](#)

### 离线下载阅读

- [爬取你要的数据：爬虫技术 PDF](#)
- [爬取你要的数据：爬虫技术 ePUB](#)
- [爬取你要的数据：爬虫技术 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](mailto:crifan.com)，我会尽快删除。谢谢合作。

## 鸣谢

为何又叫模拟登陆

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：2020-07-14 18:29:18

# 爬虫简介

爬虫，此处主要指的是，能够从网站的页面或app等数据源中爬取到你所需要的数据的代码程序。

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：2020-07-14 16:49:26

# 爬虫的叫法

爬虫有很多种常见的叫法，整理如下：

- 爬虫
  - 常见英文说法：
    - `crawler` =爬取数据的工具
      - `crawl` 英文原意：爬，爬行
    - `spider` = 蜘蛛 =像蜘蛛捕获昆虫一样你去捕获你要的数据
      - `spider` 英文原意：蜘蛛
    - 为何把爬取数据的工具叫做蜘蛛，见后续的类比解释：[为何叫爬虫](#)
  - `scraper` =刮取到你想要的数据的工具
    - `scrape` 英文原意：刮取
  - `grab` =抓取你要的数据的工具
    - `grab` 英文原意：攫取，夺取
- 爬取数据
  - 常见英文说法：
    - `crawl data` = `crawling data`
    - `scraping data`
    - `grabbing data`
- 爬取网站 = 爬取网页
  - 常见英文说法：
    - `crawl website`
- 模拟登录
  - 常见英文说法：
    - `emulate login`
    - `login emulation`
  - 至于为何也会被叫做 模拟登录，详见后续章节：[为何又叫模拟登陆](#)

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：2020-07-14 16:54:14

## 为何叫爬虫

通过下面类比，来解释为何被叫做爬虫：

对比	现实世界的蜘蛛网	计算机世界的互联网
图解		
	蜘蛛	你  自己
	织网 = 用 蜘蛛网	写 爬虫 代码  = crawler = spider
	捕获 = 抓 自己要的 东西 = 食物 = 昆虫	爬取 = 抓包 自己要的 数据  (并保存)

## 相关说明

- 互联网：是一个包含众多资源的大网络
  - 狹义 上说，主要指的是：
    - Web领域=各种 网站 = 网页
      - 里面有各种（我们想要爬取的）数据
        - 比如想要爬取汽车的车型车系，可以从 汽车之家等网站爬取
  - 广义 上说包含：
    - （上面提到的）各种网站=网页
    - 各种 app
      - 包括各种 Android 和 iOS 中的app软件
      - 比如想要爬取别人的app中的一些数据
        - 比如爬取大众点评app中的商家和用户评论数据
    - 各种 其他渠道 、 终端 的数据和资源
      - 微信公众号
        - 理论上也是属于 网页
      - 小程序
        - 微信小程序
        - 支付宝小程序
    - 等等

## 为何又叫模拟登陆

那 爬虫 为何也会被叫做 模拟登录 呢？

- 对于这种情况：想要爬取很多网站上的数据，需要用户（使用账号和密码等方式）去登录后才能获取到
- 所以要先去 模拟（用户）登录，然后才能 爬取数据
- 而模拟登录的过程，有时候或者经常，比后续的爬取数据更难，更复杂
- 所以此时的爬取全称是 先要模拟用户登录后去爬取数据
- 也就常简称为 模拟登陆
  - 用 模拟登陆 指代 爬虫

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：2020-07-14 16:49:26

# 爬虫应用

## 爬虫应用领域

爬虫的应用领域，从广义上来说，人类用网络能做啥，爬虫就能干啥。

对于爬虫的具体应用领域，借用和爬虫密切相关的 IP代理 技术的应用来说明：

- 定时营销策略
  - 游戏试玩
  - 游戏挂机
  - QQ营销
  - 宝贝收藏
  - 下载补量
    - 网购下单
      - 使用软件自动抢票、抢购手机，提高效率
    - 秒杀抢购
- 数据采集应用
  - 爬虫抓取
  - 页面采集
    - 编写爬虫采集网络公开数据，分析得出全面的市场分析报告
  - 投票点赞
    - 投票发帖
  - 网站注册
  - 效果回访
  - 网站搭建
- 优化推广方案
  - 竞价优化
  - 电商优化
  - 邮件群发
  - 打码投票
  - 论坛发帖
  - 问答推广

## 爬虫应用举例

### 自动登录=签到脚本

对于需要用户登录的网站或系统，如果用爬虫能够成功模拟登录的话，则往往就可以实现，其他一些人所需要的功能：

- 用来自动登录系统 -》 叫做 自动登录
  - 用途有很多
    - 比如
      - 每天用来签到 -》 爬虫签到 = 爬虫签到脚本
      - 比如
        - 有人弄过 百度贴吧的签到脚本

- 自制BILIBILI弹幕爬取，签到，抢楼等爬虫
- 有人用来用爬虫在各大机场自动签到获取流量
- 每天理财网站登陆签到获取积分
- 浦发信用卡自动签到
- 总之还是那句话
  - 可以用爬虫来干什么
    - 取决于你的想象：你想用来干什么
    - 和你自己的技术水平：你自己能不能搞定，能不能实现
- 模拟用户发布内容
  - 就变成了
    - 自动发帖（脚本）
    - 自动回复（脚本）

## 模拟浏览器操作

既然爬虫可以爬取网页，那么理论上就支持，用来模拟用户去点击网页，去实现模拟用户的操作，用爬虫，也叫做自动化脚本，去模拟浏览器的各种操作。

而对于模拟浏览器操作方面，和爬虫关系很密切，不过又属于不同的领域。

模拟浏览器 = Web Browser Automation，这个领域，又有很多不同的框架，工具，和技术。

比如：

- Selenium
  - Selenium知识总结
- 无头浏览器 = Headless Browser：
  - 含义：没有头的，没有界面的浏览器（内核）
    - -》因为写代码控制和操作浏览器时，往往不需要（像普通用户用浏览器看网页时那样）看界面
    - -》专门用于模拟浏览器行为，用于模拟浏览器，提供接口供你操作浏览器
  - 常见工具
    - Phantomjs

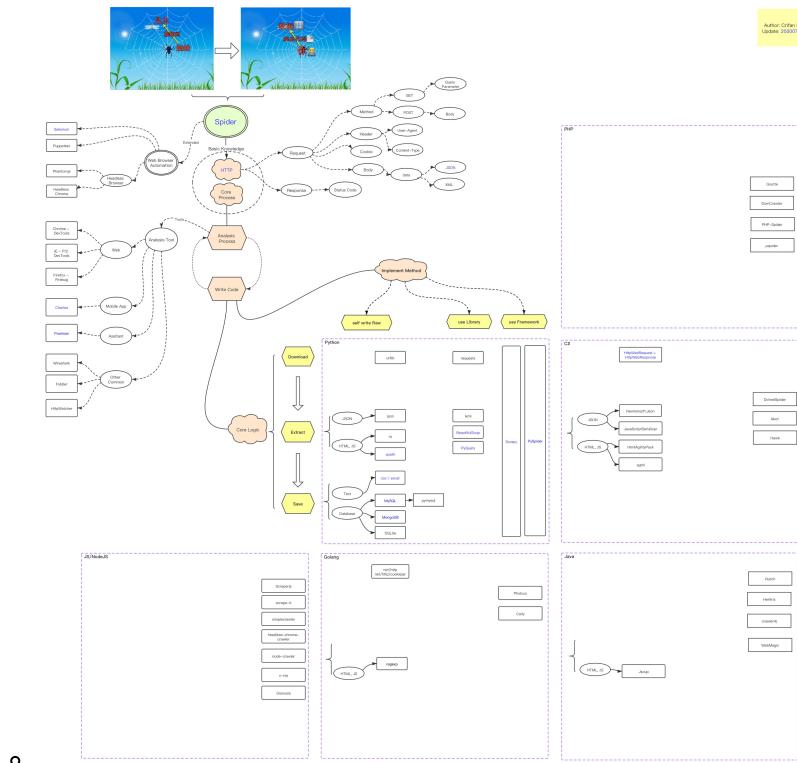
crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by  
Gitbook最后更新：2020-07-14 17:25:17

# 爬虫的核心逻辑

## 整体概览和技术路线图

爬虫的核心逻辑和整体结构和技术路线图，可以通过一图胜千言 来表示：

- 在线浏览
  - Spider Roadmap 20200714 - ProcessOn
- 图



## 核心逻辑和原理

接下来的文字解释，是对上述核心逻辑的详细诠释：

爬虫的最核心的逻辑和原理，主要包含：

- 爬虫的核心 流程
  - 先要 抓包分析
  - 再去 写爬虫代码
- 爬虫的核心 步骤 =核心 功能：写爬虫之前，要搞懂爬虫主要做了哪些事情
  - 下载
  - 提取
  - 保存
- 爬虫的典型 实现方式：然后再去搞懂有哪些方式去实现你的爬虫
  - 裸写 代码
  - 用 库 写

为何又叫模拟登陆

- 用 框架 写

- 再去搞懂：
  - 为何要用框架
  - 以及有哪些语言的哪些 爬虫框架
  - 然后才是选用合适的语言的合适的爬虫框架，去基于框架写爬虫代码

下面详细解释。

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：2020-07-14 17:49:53

## 爬虫的核心流程

如前面所说，爬虫的最核心的流程，其实就是2个阶段：

- 先要 抓包分析
  - 搞清楚，网站中有哪些url网址的网页需要去抓取，app中有哪些页面背后对应着哪些api接口需要去抓取
  - 以及每个url或api中，Request中都需要传递哪些参数，比如GET请求有哪些query parameter，POST有哪些Body的Json参数
  - 才能返回正确的，期望的Response，才能获取到自己要的数据
  - 如此，搞懂要抓取哪些数据，用什么逻辑才能获取到这些数据，是写代码真正实现这些逻辑，获取到真正的数据的前提和基础
- 再去 写爬虫代码
  - 然后才能根据前面已经搞懂的从无到有如何抓取到你要的数据的逻辑
  - 选择合适的方式，是裸写代码，还是用库实现，还是用爬虫框架
  - 去根据对应情况，写代码去下载页面或数据，再去裸写代码找合适的库实现规则去提取要的数据，最终保存数据

下面接着去详细解释，如何抓包分析，以及如何写爬虫代码。

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：2020-07-14 16:49:26

## 爬虫的核心步骤

接下来介绍爬虫的原理、过程和步骤，以及相关涉及到的知识。

从原理上来说，写爬虫去爬数据的过程，最核心的就这3步：

- 下载 = download
- 提取 = extract
- 保存 = save

下面详细解释每一步的各种细节：

- 下载 = 下载网页
  - 做了什么：请求网址或api接口，去下载返回
  - 得到什么：html网页或json字符串
  - 涉及到
    - (尤其是新手需要学习) Http基本知识
      - Request
        - Method
          - GET
          - POST
          - 等
        - Header
          - User-Agent
          - Content-Type
          - Accept
          - Authorization
          - 等
        - Cookie
        - Body
          - data
            - json
      - Response
        - Status Code
        - Header
        - Cookie
        - Body
          - data
            - json
    - 教程：主流数据格式：JSON
    - 教程：HTTP知识总结
  - 如果被爬方（网站，app等）
    - 需要用户登录后才能看到数据
    - 用技术绕过限制
      - 模拟登陆
        - 先要抓包分析出登录逻辑
        - 再用代码模拟用户登录
    - 做了一些反爬措施

- 验证码
  - 用技术绕过限制
    - 验证码识别
      - (用第三方) 打码平台
    - IP限制 + 抓取频率 限制
      - 用技术绕过限制
  - 身份限制
    - Http的Headers
    - UA = User-Agent
- 被爬网站所含页面层级很多
  - 抓取策略
    - 深度优先遍历策略
    - 宽度优先遍历策略
    - 反向链接数策略
    - Partial PageRank策略
    - OPIC策略策略
    - 大站优先策略
- 提取数据
  - 做了什么：从（返回的）网页（的 html , js 等）或 json 中提取
  - 得到什么：自己需要的内容
  - 涉及到
    - 字符编码 的问题
      - 如果搞不清编码，就容易出现各种乱码问题
      - 需要学习相关 编码 知识
      - 教程：[字符编码详解与应用](#)
        - 【整理Book】Python心得：字符串和字符编码
        - html 的 meta charset
        - 编码检测
          - Python
          - chardet
    - 如果被爬方做了反爬
      - 数据加密
      - 用技术绕过
        - 找到 解密 的逻辑和方法
        - 教程：[安卓应用的安全和破解](#)
          - 【已解决】尝试破解小花生app安卓apk希望看到api返回的json中的J的解密算法得到明文
- 保存数据
  - 做了什么：把数据保存到对应的地方
  - 得到什么：包含了我们要的特定格式的数据的文件或数据库
  - 保存成不同格式：
    - 文件
      - txt
      - csv / excel
    - 教程：[Python心得：操作CSV和Excel](#)
  - 数据库

为何又叫模拟登陆

- mysql
  - 教程: 主流关系数据库: MySQL
- mongodb
  - 教程: 主流文档型数据库: MongoDB
- sqlite
- 等等

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: 2020-07-14 18:25:47

# 爬虫的典型实现方式

实现爬虫的方式有很多，根据复杂度，可以分成典型的3种方式：

- 裸写 爬虫代码
  - 解释：在自己了解 HTTP 、爬虫等相关背景知识的前提下，用相对比较原始的方式，用内置库实现爬虫全部功能
  - 优点：更贴近和了解底层技术
  - 缺点：要求熟悉底层技术，相对用已有的库，写起来比较复杂
- 用第三方 库 写爬虫代码
  - 解释：用第三方的、更强大、更好用，网络库下载内容，内容提取库提取数据
  - 优点：省心，高效
  - 缺点：
    - 要额外引入库，且要了解如何使用
    - 对于新手，往往是直接用了第三方库后，不了解内部机制
- 用 爬虫框架
  - 用成熟的也更复杂和强大的爬虫框架，让框架帮你做重复工作，自己只需学核心的爬虫逻辑即可爬取到数据
  - 优点：适合更复杂的爬虫任务，充分利用框架的任务调度，url去重等等高级功能
  - 缺点：
    - 很多适合杀鸡用牛刀，比较重，不够轻量级
    - 出了问题，需要熟悉内部机制才容易解决问题

下面详细解释，对于爬虫的核心步骤中，不同实现方式的优缺点和所涉及内容：

- 下载
  - 裸写 爬虫代码
    - 举例
      - Python 的 urllib
      - C# 的 HttpWebRequest + HttpWebResponse
      - [crifanLib.cs之Http](#)
  - 用第三方 库 写爬虫代码
    - 举例
      - Python
      - requests
- 提取
  - 提取数据的方式：
    - 从json中提取想要的内容
      - 用json库，把json字符串转换为json对象（dict，字典）即可
      - 无需（html）解析相关的库
    - 常见的库
      - Python
        - json
      - C#
        - Newtonsoft.Json
        - JavaScriptSerializer

- 从html, js等内容中提取想要的内容
  - 裸写 爬虫代码
    - 正则
      - 应用广泛的超强搜索：正则表达式
      - Python
        - re 模块
        - Python中的正则表达式：re模块详解
    - XPath
      - XPath知识总结
  - 用第三方 库 写爬虫代码
    - Python
      - lxml
        - 【记录】Python中尝试用lxml去解析html – 在路上
      - BeautifulSoup
        - 网页解析利器：BeautifulSoup
      - PyQuery
        - HTML解析库Python版jQuery：PyQuery
      - python-goose
      - 等
    - C#
      - HTML解析
        - HtmlAgilityPack
        - sgml
    - Java
      - Jsoup
  - 保存
    - 裸写 爬虫代码
      - 自己写代码保存到对应文件或数据库中
    - 用 库 写爬虫代码
      - 用库去将数据保存到文件或数据库中
    - 用 爬虫框架
      - 框架内置接口
        - PySpider
          - 用内置接口，自动保存数据到对应数据库中

## 不同实现方式和爬虫不同步骤的对应关系

下面以 Python 语言为例，来解释不用爬虫的实现方式和不同步骤之间的对应关系：

## 为何又叫模拟登陆

	下载 (网页)	提取 (内容)	处理 (逻辑)
自己裸写 Python代码	<code>urllib</code>	<code>re</code>	<code>tx</code>
用各种 Python库组合	<code>requests</code>	<code>BeautifulSoup / lxml</code>	<code>csp</code> <code>y</code>
用框架 PySpider	<code>requests</code> ( PySpider 的 <code>self.crawl</code> )	( PySpider 内置的) <code>PyQuery</code>	( PySpider 内置的) <code>PyQuery</code>

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: 2020-07-14 18:27:34

## 抓包分析

- 抓包 = 抓包分析

### 什么是抓包

不论用哪种方式去写爬虫代码，对于 `下载` 来说，具体要请求网站 `url` 是什么，调用什么 `api` 接口，传递什么 `参数`，以及获取到数据后，用什么规则提取出需要的数据等等内容，都需要事先去分析和研究清楚，这个抓取网络请求的数据包的过程，一般叫做：`抓包`

即：

- 下载
  - 需要访问的网页 `url` 地址或 `api` 接口是什么
    - 以及传递什么 `参数`
- 提取
  - 对于返回数据，需要抓取具体哪一部分
    - 对应的数据的 `提取规则` 是什么

提示：

虽然对于爬虫的核心流程是先要抓包分析搞清楚逻辑，然后才能去写爬虫代码，不过实际上很多时候，是边分析，边写代码的。

尤其是对于一些复杂的网站或app来说，往往是分析的同时，也要写一些代码去验证和测试抓取的逻辑是否行得通的。

总之，对于爬虫的流程：

- 逻辑上是：先抓包分析，再写爬虫代码
- 实际上（往往）是：边抓包分析，边写代码

## 抓包的难度

- 普通网页：抓包分析，一般比较简单
- 复杂网站：对于需要登录才能获取到数据，且加了验证码等做了其他反爬措施和手段的网站和app，抓包分析起来，一般都很复杂
  - 复杂网站的抓包分析和破解，往往比（之后的，单纯的）写爬虫去 `下载`+`提取`+`保存`，要难多了

## 抓包常用工具

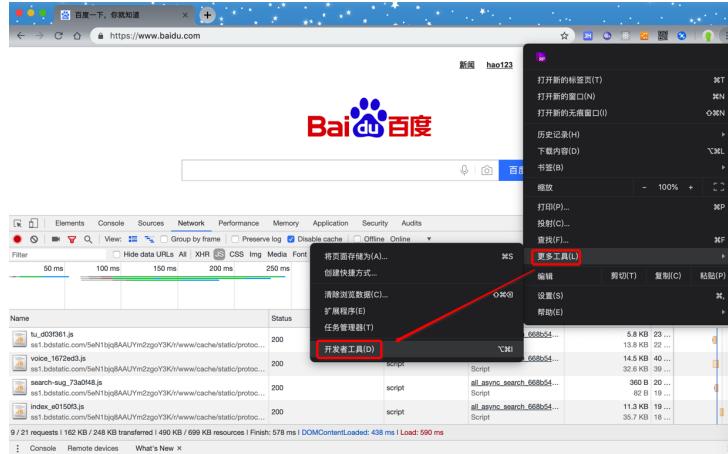
我们要写爬虫去爬取的数据，从数据源的形态分，大概分两类：

- 网站 = 网页 = 网站中的各种网页
- app = app 中内部发出的请求设计的 api 接口

根据要抓取的数据的源不同，常用的一些辅助分析工具有：

- 网站抓包分析

- Chrome 的 开发者工具



- 快捷键:

- Windows: `Ctrl + Shift + I`
    - Mac: `Command + Option + I`

- 如何使用

- 官网资料: [Chrome 开发者工具](#)

- IE 的 F12

- 如何使用

- 【整理】各种浏览器中的开发人员工具Developer Tools: IE9的F12, Chrome的`Ctrl+Shift+J`, Firefox的Firebug
    - 【总结】浏览器中的开发人员工具 (IE9的F12和Chrome的`Ctrl+Shift+I`) -网页分析的利器
    - 【教程】如何利用IE9的F12去分析网站登陆过程中的复杂的 (参数, cookie等) 值 (的来源)
    - 【教程】手把手教你如何利用工具(IE9的F12)去分析模拟登陆网站(百度首页)的内部逻辑过程

- Firefox 的 firebug

- app抓包分析

- Charles

- 教程: [app抓包利器: Charles](#)

- 通用工具

- Wireshark

- Postman

- 用于对于api去设置参数并发送请求测试是否能获取数据

- 教程: [API开发利器: Postman](#)

- Fiddler

- HttpWatcher

## 具体怎么抓包

先要搞清楚自己想要抓取什么数据，然后再去用工具辅助分析出网页或app等数据源中，如何一步步的获取对应数据，找到期间所要依次访问哪些url或api，传递什么参数，最终获取到所要的数据。

下面就来用实际例子来说明如何抓包。

为何又叫模拟登陆

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: 2020-07-14 17:12:06

## 抓包分析网页

对于如何抓包分析网站网页类的内容，下面用具体例子来详细解释。

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：2020-07-14 16:49:26

## 静态网页

### 以抓取汽车之家中车型车系数据为例解释如何抓包

下面就以，想要抓取汽车之家网站中的车型车系数据为例，来解释，如何用抓包工具辅助分析，依次访问哪些页面，之后如何提取，才能得到我们要的数据。

TODO：

用Chrome浏览器分析过程，并截图，添加解释。

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：2020-07-14 18:09:10

# 动态网页

相关内容：

[【教程】手把手教你如何利用工具\(IE9的F12\)去分析模拟登陆网站\(百度首页\)的内部逻辑过程](#)

[【记录】模拟登陆google](#)

[【教程】如何抓取动态网页内容](#)

[【教程】以抓取网易博客帖子中的最近读者信息为例，手把手教你如何抓取动态网页中的内容](#)

TODO：

找个，需要登录的网站，或者是网页内容需要后续执行js才能加载的例子，再去用抓包工具模拟登录，或分析数据是如何加载的。

## 举例：Chrom分析大众点评某页面获取店铺数据后用PySpider实现代码并下载数据

此次通过举例来说明，如何：

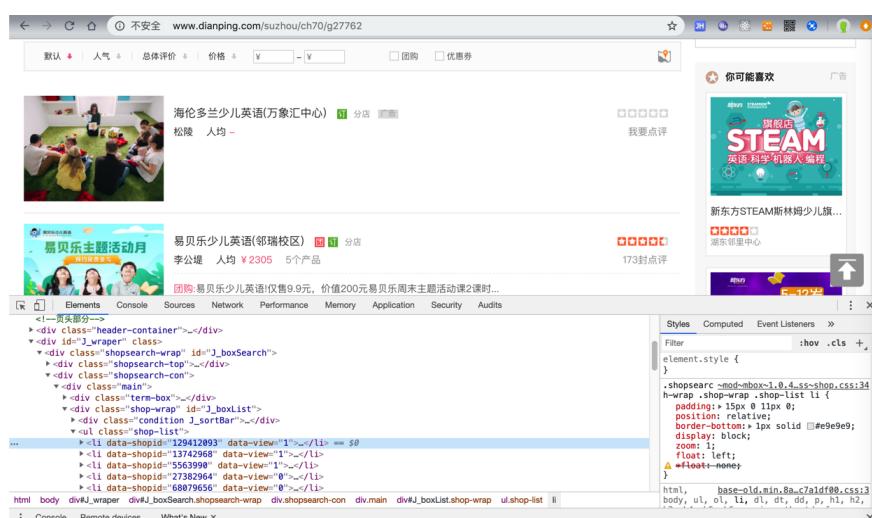
- 用Chrome分析逻辑后
- 再去（PySpider的）代码实现

期间要注意的是：要一点点模拟各种参数，才能获取到数据，否则会出现各种错误

要分析的网址：

<http://www.dianping.com/suzhou/ch70/g27762>

用Chrome打开后，是可以获取到数据的：



The screenshot shows the Google Chrome Developer Tools open on a Dianping shop listing page. The 'Elements' tab is active, displaying the HTML structure of the page. The 'Styles' tab is also visible, showing the CSS rules applied to the elements, including a rule for the shop list items.

但是PySpider中，用代码：

## 为何又叫模拟登陆

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Created on 2019-04-15 14:56:12
# Project: DianpingChilrenEnglish

from pyspider.libs.base_handler import *

import os
import json
import codecs
import base64
import gzip
import copy
import time
import re
import csv
# import datetime
from datetime import datetime, timedelta

#####
# Const
#####

"""
constCityListNamePattern = "cityList_%s_%s.json"
constMainCityFilename = "mainCityWithLevelList.json"

constUserAgentMacChrome = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36"
gHost = "http://www.dianping.com"

CategoryLevel1ParentChild = "ch70" # 全部分类->亲子
CategoryLevel2ChildEnglish = "g27762" # 幼儿教育 -> 幼儿外语

#####
# Project Specific Functions
#####

#####
# Main
#####

class Handler(BaseHandler):
    crawl_config = {
        "connect_timeout": 100,
        "timeout": 600,
        "retries": 15,
        "headers": {
            "User-Agent": constUserAgentMacChrome,
            "Accept": "application/json, text/javascript, */*; q=0.01",
            "Content-Type": "application/json",
            "Origin": "http://www.dianping.com",
            # "X-Requested-With": "XMLHttpRequest",
        }
    }

    def on_start(self):
        # self.init()

        self.realStart()

    def realStart(self):
        """
        # for debug
        """
        eachMainCity={'cityAbbrCode': 'SUZ', 'cityAreaCode': '0512', 'cityEnName': 'suzhou', 'cityName': '苏州', 'cityPinyin': 'suzhou', 'cityRegionCode': '320500', 'citySortOrder': 1, 'cityType': 'main', 'cityUrl': 'http://www.dianping.com/suzhou/ch70/g27762'}
        childEnglishEntryUrl='http://www.dianping.com/suzhou/ch70/g27762'
        childEnglishEntryUrl = "http://www.dianping.com/suzhou/ch70/g27762"
        self.crawl(
```

为何又叫模拟登陆

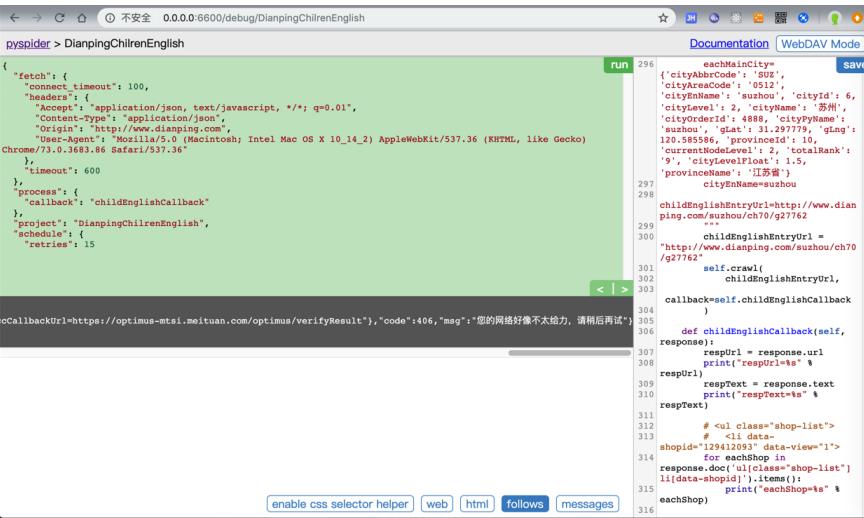
```
        childEnglishEntryUrl,
        callback=self.childEnglishCallback
    )

    def childEnglishCallback(self, response):
        respUrl = response.url
        print("respUrl=%s" % respUrl)
        respText = response.text
        print("respText=%s" % respText)

        # <ul class="shop-list">
        #   <li data-shopid="129412093" data-view="1">
        for eachShop in response.doc('ul[class="shop-list"] li[data-shopid]').
            print("eachShop=%s" % eachShop)
```

结果返回错误信息：

```
respText={"customData": {"requestCode": "c9847c945a1440d49460df748b757250", "veri
```

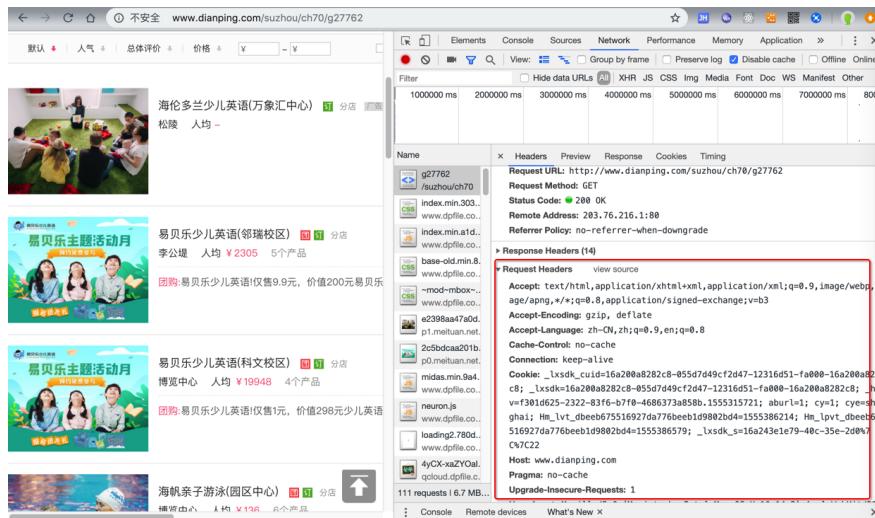


```
{
    "fetch": {
        "connect_timeout": 100,
        "headers": {
            "Accept": "application/json, text/javascript, */*; q=0.01",
            "Content-Type": "application/json",
            "Origin": "http://www.dianping.com",
            "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36"
        },
        "timeout": 600
    },
    "process": {
        "callback": "childEnglishCallback"
    },
    "project": "DianpingChildrenEnglish",
    "schedule": {
        "retries": 15
    }
}

cCallbackUrl=https://optimus-mtsi.meituan.com/optimus/verifyResult","code":406,"msg":`您的网络好像不太给力，请稍后再试`}
```

```
296         eachShop['cityName'],
297         {'cityAbbCode': 'suz',
298         'cityAreaCode': '0512',
299         'cityEnName': 'suzhou', 'cityId': 6,
300         'cityLevel': 2, 'cityName': '苏州',
301         'cityLevelFloat': 4.5, 'cityNameEn':
302         'suzhou', 'lat': 31.297779, 'long':
303         120.585586, 'provincId': 10,
304         'currentNodeLevel': 2, 'totalRank':
305         9, 'cityLevelFloat': 1.5,
306         'provincName': '江苏省',
307         'cityEnName': 'suzhou'
308     }
309     childEnglishEntryUrl=http://www.dian
310     ping.com/suzhou/ch70/g27762
311     ...
312     childEnglishEntryUrl =
313     "http://www.dianping.com/suzhou/ch70
314     /g27762"
315     self.crawl(
316         childEnglishEntryUrl,
317         callback=self.childEnglishCallback
318     )
319     def childEnglishCallback(self,
320         response):
321         respUrl = response.url
322         print("respUrl=%s" %
323             respUrl)
324         respText = response.text
325         print("respText=%s" %
326             respText)
327
328         # <ul class="shop-list">
329         #   <li data-
330         for eachShop in
331         response.doc('ul[class="shop-list"] li[data-shopid]').
332         items():
333             print("eachShop=%s" %
334                 eachShop)
```

然后去参考Chromet中看到的参数：



对于各种header，想办法一个个加上去试试，最终发现是：

给 Accept 加上 text/html 相关类型：

为何又叫模拟登陆

```
# "Accept": "application/json, text/javascript, */*; q=0.01",
"Accept": "application/json, text/javascript, text/html,application/xhtml+xml,
```

就可以获取到数据了：

The screenshot shows the PySpider web interface. At the top, it displays the URL `0.0.0.0:6600/debug/DianpingChilrenEnglish`. Below the URL is the configuration code for the spider:

```
{
  "fetch": {
    "connect_timeout": 100,
    "headers": {
      "Accept": "application/json, text/javascript, text/html,application/xhtml+xml;q=0.9,image/webp,image/apng,*/*; q=0.8,application/signed-exchange;v=b3",
      "Content-Type": "application/json",
      "Origin": "http://www.dianping.com",
      "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36"
    },
    "timeout": 600
  },
  "process": {
    "callback": "childEnglishCallback"
  },
  "project": "DianpingChilrenEnglish",
  "schedule": {
    "retries": 15
  },
  "taskid": "ca9d080e79697c52fcffcf9ff51efcc8",
  "url": "http://www.dianping.com/suzhou/ch70/g27762"
}
```

Below the configuration is the raw HTML response from the server:

```
respUrl=http://www.dianping.com/suzhou/ch70/g27762
respText=
<!DOCTYPE html>
<html>
<head>
<title>苏州幼儿外语,苏州少儿英语(第1页)-苏州幼教-大众点评网</title>
<link rel="icon" type="image/x-icon" href="/www.dpfile.com/app/pc-common/dp_favicon.a4af753914321c8e82e4"/>
<link rel="shortcut icon" type="image/x-icon" href="/www.dpfile.com/app/pc-common/dp_favicon.a4af753914321"/>
<link rel="search" type="application/opensearchdescription+xml" href="/opensearch.xml" title="餐馆搜索"/>
<link rel="alternate" href="/rss/search/6/27762/0/70/" type="application/rss+xml" title="RSS"/>
<meta name="Description" content="苏州幼儿外语,苏州少儿英语" />
<meta http-equiv="mobile-agent" content="format=html5; url=https://m.dianping.com/shoplist/1/c/27762">
<meta name="location" content="province=江苏;city=苏州" />
<!--基础优化-->
```

At the bottom of the interface, there are several buttons: `enable css selector helper`, `web`, `html`, `follows`, and `messages`.

-» 还是之前提到的那个逻辑：想办法参考 Chrome 调试看到的所有的重要参数：

- `url` 中的 `query string`
- `header` 中有价值的部分
- 甚至相关的 `cookie`, `session`, `localStorage` 等内容

都加上后，多数情况下，就可以获取到对应的返回的数据了。

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：2020-07-14 18:16:41

## 抓包分析app

相关内容：

[app抓包利器：Charles](#)

[【记录】Mac中用Charles去抓包Android中的app家长通中的绘本](#)

[【基本解决】爬取app数据少儿流利说](#)

[【记录】爬取小花生app中自主阅读馆和亲子阅读馆中的有音频的绘本数据](#)

[【整理】Mac中用Charles抓包iOS或Android手机app中包括https的数据](#)

TODO：

用Charles去抓包某app，并给出详细过程。

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：2020-07-14 16:49:26

## 爬虫框架

而上面的三个步骤：下载+提取+保存，其中包含很多通用的，重复的逻辑和操作，所以有些人开发出来，独立的爬虫框架，方便我们去实现爬虫。

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：2020-07-14 16:49:26

# 为何需要爬虫框架

接着来解释，为何要用爬虫框架：

- 框架帮你把大部分重复的工作都实现了
  - 做了哪些通用的事情
    - 下载
      - 网络异常时 自动重试 retry
      - 还可以设置
        - 最大 重试次数：最多重试几次
        - 如果还是不行，才视为下载失败
        - 重试间隔：两次重试之间的间隔时间
      - 好处：不用担心偶尔某次网络有问题，就导致下载失败了，因为还可以自动重试
      - 对比：自己裸写代码，就要考虑这种异常情况，导致自己爬虫代码臃肿和逻辑复杂
        - 花了太多精力在和爬取数据关系不大的方面，不值得，效率低
    - 下载 进程的管理
      - 同时发出多个url请求去下载内容
      - 有专门的进程管理和调度策略
      - 好处：能同时并发多个请求
      - 对比：自己裸写代码去下载，往往同一时刻只能有一个请求
        - 否则就要花很多精力去实现并发
    - url去重
      - 前后（不同页面，不同场景下）发出的多个url中是否有重复的
      - 如果有，则自动忽略掉，去掉，去除重复=去重
    - 提取
      - 做了啥
        - 内置常用的内容提取的库
          - PySpider 集成 PyQuery
          - Scrapy 集成选择器，支持： xpath、css、re
        - 同时支持可选的第三方的库
          - Scrapy 也支持用 BeautifulSoup 提取内容
      - 好处：不用额外安装和使用这些库
      - 对比：自己裸写代码就要考虑选用哪些合适的库去提取内容
    - 保存
      - 做了啥
        - 集成各种保存数据的接口和框架
          - PySpider
            - 自带默认保存为 sqlite 中
            - 可以从界面中导出 csv 或 json
          - 其他数据库接口
            - mysql
            - mongodb
            - 等等

- 好处：可以方便的选择保存数据的方式，无需过多操心细节
- 对比：自己裸写代码，还要安装不同的数据库的库，再手动写（sql等）代码去保存数据
- 还带了很多额外的好用功能
  - PySpider
    - 带 UI界面， 调试非常方便
    - 支持网页内容是 执行js 后才生成的
    - 通过集成第三方 phantomjs

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：2020-07-14 16:49:26

# 常见爬虫框架

- Python
  - PySpider
    - Python爬虫框架：PySpider
  - Scrapy
    - 主流Python框架：Scrapy
  - 其他
    - Grab
    - Portia
    - newspaper
    - ruia
    - Cola
    - Sasila
- Java
  - Nutch
    - Nutch是一个基于Apache的Lucene，类似Google的完整网络搜索引擎解决方案，基于Hadoop的分布式处理模型保证了系统的性能，类似Eclipse的插件机制保证了系统的可客户化，而且很容易集成到自己的应用之中。
  - Heritrix
    - Heritrix是一个开源，可扩展的web爬虫项目。Heritrix设计成严格按照robots.txt文件的排除指示和META robots标签。
  - crawler4j
    - crawler4j is an open source web crawler for Java which provides a simple interface for crawling the Web. Using it, you can setup a multi-threaded web crawler in few minutes
  - WebMagic
    - 国人黄亿华先生的良心大作。无须配置、便于二次开发的爬虫框架，它提供简单灵活的API，只需少量代码即可实现一个爬虫
- Golang
  - Pholcus
  - Colly
- NodeJS
  - headless-chrome-crawler
- C#
  - abot

# 如何写爬虫

而不同语言去实现爬虫，其方便程度、难度、逻辑，又有一些区别，此处再来详细解释。

## 旧教程

之前写过一些旧教程，需要的也可以参考：

- 详解抓取网站，模拟登陆，抓取动态网页的原理和实现（Python，C#等）
- 如何用Python，C#等语言去实现抓取静态网页+抓取动态网页+模拟登陆网站  
– 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：2020-07-14 18:01:44

为何又叫模拟登陆

# 用Python写爬虫

爬虫领域中，相对最方便和好用的编程语言（和库、框架等），要属： Python

具体如何做，详见独立教程：

[如何用Python写爬虫](#)

以及其他一些相关内容：

[【教程】模拟登陆网站 之 Python版（内含两种版本的完整的可运行的代码）](#)

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： 2020-07-14 17:15:50

## 用C#写爬虫

之前写的相关内容：

- [【教程】抓取网并提取网页中所需要的信息 之 C#版](#)
- [【教程】模拟登陆网站 之 C#版（内含两种版本的完整的可运行的代码）](#)

其中也把C#的常用网络功能，整理出独立函数，并发布，且加了说明：

[crifanLib.cs之Http - 详解crifan的C#库：crifanLib.cs](#)

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：2020-07-14 17:58:18

为何又叫模拟登陆

## 用Go写爬虫

之前写的相关内容：

[【记录】用go语言实现模拟登陆百度](#)

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：2020-07-14 16:49:26

为何又叫模拟登陆

# 用Java写爬虫

之前写的相关内容：

[【教程】模拟登陆百度之Java代码版](#)

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：2020-07-14 16:49:26

## 用PHP写爬虫

之前找的一点资料：

[【整理】和PHP的HTTP,网页抓取,网络爬虫相关的库,框架,资料 – 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：2020-07-14 16:49:26

## 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: 2020-07-14 16:49:26

## 名词解释

### 页面爬取策略=爬虫策略

在爬虫系统中,等待抓取URL队列是很重要的组成部分,等待抓取URL队列中的URL的顺序排列方式也是一个很重要的问题,因为这会决定到先抓取哪个页面,后抓取哪个页面.而决定这些URL排列顺序的方法,叫做抓取策略.下面主要介绍几种常见的抓取策略:

- 深度优先遍历策略
  - 深度优先遍历策略是指网络爬虫会从起始页开始,一个链接一个链接跟踪下去,直到处理完这条线路之后才会转入下一个起始页,继续跟踪链接.遍历的路径为: A-F-G ,E-H-I ,B ,C, D
- 宽度优先遍历策略
  - 宽度优先遍历策略的基本思路就是,将新下载网页中发现的链接直接放入待抓取URL队列的末尾.也就是说网络爬虫会优先抓取起始网页中链接的所有网页,所有网页都抓取完之后,再选择其中的一个链接网页,继续抓取在此网页中链接的所有网页.它的路径可以这样写:A-B-C-D-E-F ,G ,H, I
- 反向链接数策略
  - 反向链接数是指一个网页被其他网页链接指向的数量,同时反向链接数也是表示一个网页的内容受到其他人的推荐的程度.因此,很多时候搜索引擎的抓取系统会使用这个指标来评价网页的重要程度,从而决定不同网页的抓取先后顺序.
  - 而然在真实的网络环境中,由于许多广告链接、作弊链接等等的存在,反向链接数不能完全等同于重要程度.因此,许多的搜索引擎往往考虑一些可靠的反向链接数.
- OPIC策略策略
  - 这种算法实际上也是对网络页面进行一个重要性的打分.在算法开始前,会给所有页面一个相同的初始现金 (cash) .当下载了某个页面P之后,将P的现金分摊给所有从P中分析出的链接,并且将P的现金清空.对于待抓取URL队列中的所有页面按照现金数进行排序
- 大站优先策略
  - 对于待抓取URL队列中的所有网页,根据所属的网站进行分类.对于待下载页面数多的网站,优先下载.这个策略也因此叫做大站优先策略

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: 2020-07-14 16:49:26

## 参考资料

- 【未解决】搞懂IP代理池相关概念和逻辑
- 
- [crifanLib.cs之Http](#)
- [HTTP知识总结](#)
- [app抓包利器：Charles](#)
- [JSON详解](#)
- [主流数据格式：JSON](#)
- 【已解决】C#中解析Json字符串
- 【记录】Python中尝试用lxml去解析html
- [Python爬虫框架：PySpider](#)
- [Scrapy](#)
- [Selenium知识总结](#)
- [主流Python框架：Scrapy](#)
- 【记录】C#中的HTML解析 – 在路上
- [如何用Python写爬虫](#)
- [XPath知识总结](#)
- 【整理】和PHP的HTTP,网页抓取,网络爬虫相关的库,框架,资料 – 在路上
- 【教程】抓取网并提取网页中所需要的信息 之 C#版
- 【教程】模拟登陆网站 之 C#版 (内含两种版本的完整的可运行的代码)
- 【教程】模拟登陆网站 之 Python版 (内含两种版本的完整的可运行的代码)
- 【记录】用go语言实现模拟登陆百度
- 【教程】模拟登陆百度之Java代码版
- [Python心得：操作CSV和Excel](#)
- [主流关系数据库：MySQL](#)
- [主流文档型数据库：MongoDB](#)
- [Python中的正则表达式：re模块详解](#)
- 【整理】Mac中用Charles抓包iOS或Android手机app中包括https的数据
- 【记录】模拟登陆google
- 【教程】如何抓取动态网页内容
- 【教程】以抓取网易博客帖子中的最近读者信息为例，手把手教你如何抓取动态网页中的内容
- 【经验总结】Http, 网页访问, HttpRequest, HttpResponse相关的知识 – 在路上
- 如何用Python, C#等语言去实现抓取静态网页+抓取动态网页+模拟登陆网站 – 在路上
- [字符编码详解与应用](#)
- [安卓应用的安全和破解](#)
- 
- [Grab](#)
- [python-goose](#)
- [PySpider](#)
- [Portia](#)
- [newspaper](#)
- [ruia](#)

- [Cola](#)
- [Sasila](#)
- [Nutch](#)
- [Heritrix](#)
- [crawler4j](#)
- [WebMagic](#)
- [Colly](#)
- [Pholcus](#)
- [headless-chrome-crawler](#)
- [scrapy中的提取正文的方法-python,爬虫,scrapy研究-51CTO博客](#)
- [基于Python的Scrapy爬虫入门：页面提取 SegmentFault](#)
- [Scrapy定向爬虫教程\(二\)——提取网页内容 - 春华秋实 - CSDN博客](#)
- [Scrapy笔记04- Selector详解 | 飞污熊](#)
- [Scrapy爬虫抓取网站数据 | ShinChan's Blog](#)
- [Scrapy爬虫入门教程十二 Link Extractors \(链接提取器\) - inke的博客 - CSDN博客](#)
- [基于WebMagic的CSDN博客爬虫 - zhuqihui的专栏 - CSDN博客](#)
- [Heritrix与Nutch对比 - 爱专集](#)
- [Nutch、heritrix、crawler4j优缺点 - CSDN博客](#)
- [爬虫用哪个好？ - 知乎](#)
- [作为基础服务的数据采集，发展到哪个阶段了？搜狐科技搜狐网](#)
- [Python3网络爬虫\(四\)：使用User Agent和代理IP隐藏身份 - Jack-Cui - CSDN博客](#)
- [Python 爬虫一些常用的UA\(user-agent\) - abe\\_abd的博客 - CSDN博客](#)
- [如何评价可以自动更换 User-Agent 的爬虫设计？ - 知乎](#)
- [DarkSand/Sasila: 一个灵活、友好的爬虫框架](#)
- [Python有哪些常见的、好用的爬虫框架？ - 知乎](#)
- [8个最高效的Python爬虫框架，你用过几个？ - 个人文章 - SegmentFault 思否](#)
- [爬虫的几种抓取策略 | 阿布云 - 因为专业·所以简单](#)
- [【爬虫工程师招聘】智慧芽爬虫工程师招聘-BOSS直聘](#)
- [【数据采集招聘】智慧芽数据采集招聘-BOSS直聘](#)
- [【高级爬虫工程师招聘】智慧芽高级爬虫工程师招聘-BOSS直聘](#)
- [如何对知乎内容进行爬虫？ - 知乎](#)
- [用爬虫在各大机场自动签到获取流量](#)
- [每天理财网站登陆签到获取积分](#)
- [浦发信用卡自动签到](#)
- [自制BILIBILI弹幕爬取，签到，抢楼等爬虫](#)
- [Selenium](#)
- [梦见蜘蛛网\\_国学易经](#)
- [Python爬虫原理 - Python开发之路 - 博客园](#)
- [蜻蜓代理 - 企业级高质量代理ip平台](#)
- [讯代理-爬虫代理-HTTP代理-代理服务器](#)
- [BruceDone/awesome-crawler: A collection of awesome web crawler,spider in different languages](#)
-