# 大神博客：

python爬虫大数据：<https://ask.hellobi.com/blog/seng/category/517>

爬虫系统：<http://mp.weixin.qq.com/s?__biz=MzI2MzQ1NTEwNg==&mid=2247483698&idx=2&sn=cffef134a5470a79567f86d403f22cb7&chksm=eabae9dfddcd60c93b485d1c503030a0563c9fc8706d3fa01e41d5ffd14e998f2bb827d8ad12&mpshare=1&scene=23&srcid=0419WJScIAz2xJJb1T15Irul#rd>

# 环境搭建

## 1 python安装

必须：运行 yum install sqlite-devel（方便后面安装sqlite数据库）

yum install zlib-devel （方便安装setuptools）

yum install gcc libffi-devel python-devel openssl-devel

安装后需要重新编译环境 make&make install

**1.刚开始centos可能会缺少gcc等组件，先安装组件**

yum groupinstall "Development Tools"

**2.下载源码，解压后进入目录**

#下载地址[https://www.python.org](https://www.python.org/" \t "http://www.cnblogs.com/longxiang92/p/_blank)

#解压

tar xvJf \*\*\*.tar.xz#进入目录

cd Python-3.5\*

**3.安装python**

#创建目录

mkdir /usr/python3.5

#运行配置 指定要安装到/usr/python3.5目录下 并共享

./configure --prefix=/usr/python3.5 --enable-shared  
  
make#安装

make install

**4.创建Python3.5软连接**

\*\*在/usr/bin中有python、python2、python2.7三个文件依次指向后者（即都是python2.7.5）

\*\*不建议依照其他网上的方法把默认的python修改为指向3.5版本，这样的话将无法使用firewall防火墙，而且可能还有其他有影响

\*\*建议另外创建一个python3的软连接

cd /usr/bin  
#创建python3.5版本的软连接

ln -s /usr/python3.5/bin/python3 /usr/bin/python3

#查看python版本  
python3 -V

**5.如果运行查看版本失败**

错误信息类似这样 ：error while loading shared libraries: xxxxxxxx: cannot open shared object file: No such file or directory（xxxx为文件名）

进入安装文件夹（解压出来的目录）

#拷贝生成的文件到lib

cp xxxx（libpython3.5m.so.1.0） /usr/local/lib

#修改ldconfig文件

vim /etc/ld.so.conf

#新加一行 /usr/local/lib

运行命令ldconfig

Ldconfig

6、设置别名方便使用

alias py=python3

## 2安装setuptools（easy\_install）

**方法一（推荐）：**

wget --no-check-certificate  https://pypi.python.org/packages/source/s/setuptools/setuptools-19.6.tar.gz#md5=c607dd118eae682c44ed146367a17e26

tar -zxvf setuptools-19.6.tar.gz

cd setuptools-19.6

python3 setup.py build

python3 setup.py install

**方法二（有时不太行）：**

最新版下载地址：<https://pypi.python.org/pypi/ez_setup>

tar -zxvf setuptools-3.6.tar.gz

cd setuptools-3.6

py setup.py install

卸载：easy\_install -mxN networkx

## 3 pip安装

从官网下载最新pip：<https://pypi.python.org/pypi/pip>

tar zvxf 1.5.5 #解压文件

cd pip-1.5.5/

python3 setup.py install

# vim /etc/profile

在文档最后，添加:

export PATH="/usr/local/python2.7/bin:$PATH"

pip查看已安装的包 pip show --files paramiko 或者pip list

更新pip ：python -m pip install --upgrade pip

pip检查哪些包需要更新 pip list --outdated

pip升级包 pip install --upgrade Django

pip卸载包 pip uninstall Django

## 4 安装scrapy

Pip安装 parsel, w3lib, cryptography, pyOpenSSL 如：pip install parsel

安装 lxml 和 twisted 网址：<http://www.lfd.uci.edu/~gohlke/pythonlibs/>

windos下：pip install E:\demo\Twisted-16.4.1-cp35-cp35m-win\_amd64.whl

Twisted要安装zope.interface,可以从下面地址下载

[zope.interface](https://pypi.python.org/packages/source/z/zope.interface/zope.interface-4.0.1.tar.gz):https://pypi.python.org/pypi/zope.interface

[twisted:](http://twistedmatrix.com/Releases/Twisted/12.1/Twisted-12.1.0.tar.bz2)http://twistedmatrix.com/Releases/Twisted

最后安装scrapy：pip install scrapy

打开端口：

iptables -I INPUT -p tcp --doprt 6800 -j ACCEPT

## 5 安装scrapyd

安装scprayd，网址：[https://github.com/scrapy/scrapyd](https://github.com/scrapy/scrapyd" \t "http://blog.csdn.net/xiaoquantouer/article/details/_blank)

安装scrapyd-client，网址：[https://github.com/scrapy/scrapyd-client](https://github.com/scrapy/scrapyd-client" \t "http://blog.csdn.net/xiaoquantouer/article/details/_blank)

用easy\_install安装scrapyd为最新版

Linux下配置文件路径：

/usr/python3.5.3/lib/python3.5/site-packages/scrapyd-1.2.0a1-py3.5.egg/scrapyd

python安装目录下的 Lib\site-packages\scrapyd\default\_scrapyd.conf (Windows)

mkdir eggs

mkdir logs

mkdir items

mkdir dbs

sudo mkdir /etc/scrapyd/

chown -R cdz /etc/scrapyd/

cd /etc/scrapyd/

vi scrapyd.conf 配置如下：  
\* **以下是几个比较重要的参数**：

http\_port 监听端口

bind\_address 监听IP地址

max\_proc 最大进程数（为0指开启等同于cpu的数目的进程）

max\_proc\_per\_cpu 每个cpu开启的进程数，默认：4

debug 是否开启调试模式

eggs\_dir 存储工程egg文件的目录

dbs\_dir 存储工程数据库的目录

logs\_dir 存储日志的目录（要关闭日志则让这个配置为空

logs\_filename = {spider}-{Y}{m}{d}.log

### 调度爬虫

1、添加版本信息

curl http://localhost:6800/addversion.json -F project=myproject -F     version=r23 -F egg=@myproject.egg

2、调度spider执行

curl http://127.0.0.1:6800/schedule.json -d project=bricCrawler -d   spider=somespider

3、取消spider的执行

curl http://localhost:6800/cancel.json -d project=myproject -d      job=6487ec79947edab326d6db28a2d8651

4、列出工程

curl [http://localhost:6800/listprojects.json](http://localhost:6800/listprojects.json？Project=bricCrawler)

5、列出job

curl http://localhost:6800/listjobs.json？Project=bricCrawler

## 6 定时框架apschedule

easy\_install apscheduler

简单例子：<http://stackoverflow.com/questions/29765039/how-to-use-apscheduler-with-scrapy>

详细操作：<http://debugo.com/apscheduler/>

## 7 持久层ORM框架：

tornado+sqlalchemy

Sqlalchemy使用: <http://www.jianshu.com/p/e6bba189fcbd>

更详细的方法详解：<http://www.jb51.net/article/49789.htm>

scrapy结合sqlalchemy+redis：<http://www.code123.cc/1434.html>

sqlchemy详细使用经验：<https://www.keakon.net/2012/12/03/SQLAlchemy%E4%BD%BF%E7%94%A8%E7%BB%8F%E9%AA%8C>

Sqlalchemy链接sqlserver：需要安装pymssql

# pyodbc  
engine = create\_engine('mssql+pyodbc://scott:tiger@mydsn')  
  
# pymssql  
engine = create\_engine('mssql+pymssql://scott:tiger@hostname:port/dbname')

## 8 PDF解析工具

### 1.pdfminer3k

pip install pdfminer3k

网址：<https://pypi.python.org/pypi/pdfminer3k/>

详解：<http://www.thinksaas.cn/topics/0/686/686331.html>

<http://m.php.cn/article/359112.html>

Pdf解密：

PDF是一个加密的PDF，所以无法解析 ，但是如果直接打开PDF却是可以的并没有要求输密码什么的，原因是这个PDF虽然是加过密的，但密码是空，所以就出现了这样的问题。

解决这个的问题的办法是通过qpdf命令来解密文件（要确保已经安装了qpdf），要想在python中调用该命令只需使用call即可：

1 from subprocess import call2 call('qpdf --password=%s --decrypt %s %s' %('', file\_path, new\_file\_path), shell=True)

### 2 pdftables

pip install git+https://github.com/pdftables/python-pdftables-api.git

### 3 tabula-py

只支持单页

试试tabula，读取pdf后可转为pandas dataframe进行后续处理，也可直接输出csv文件。  
python版本的项目主页，有示例代码：[https://github.com/chezou/tabula-py](https://link.zhihu.com/?target=https://github.com/chezou/tabula-py" \t "_blank)  
安装：pip install tabula-py

如果包含中文内容需要修改编码格式：  
pd.read\_csv("example.csv", encoding="GB18030")  
参考：<https://www.zhihu.com/question/22016184>

(ps下次提问请先查看万能的stackoverflow：  
<http://stackoverflow.com/questions/17217194/extracting-table-contents-from-a-collection-of-pdf-files/26110587#26110587>

4

<https://blog.pdftron.com/2014/05/25/a-simple-example-of-converting-pdf-to-html/>

# requests使用

## Session的cookie使用

Request.get()每用一次就会建立一个长连接，使用Session则为长连接，建立一次连接，发出多次请求。

session=requests.session()

*#请求post*zhihu\_res=session.post(post\_url,data=post\_url,headers=headers)  
*#保存cookies到文件*session.cookies=cookiejar.LWPCookieJar(filename=**'cookies.txt'**)

*#存储cookie*session.cookies.save()  
*#加载cookie***try**:  
 session.cookies.load(ignore\_discard=**True**)  
**except**:  
 print(**'cookie未能加载'**)

### 从response中获取cookie

*# new\_cookie = response.cookies.get\_dict()***for** \_ **in** response.cookies:  
 **if** \_.name == **'ajaxkey'**:  
 cks = \_.name + **'='** + \_.value

或者注释掉重定向的中间件：

|  |
| --- |
| DOWNLOADER\_MIDDLEWARES\_BASE = {  **'scrapy.contrib.downloadermiddleware.robotstxt.RobotsTxtMiddleware'**: 100,  **'scrapy.contrib.downloadermiddleware.httpauth.HttpAuthMiddleware'**: 300,  **'scrapy.contrib.downloadermiddleware.useragent.UserAgentMiddleware'**: 400,  **'scrapy.contrib.downloadermiddleware.retry.RetryMiddleware'**: 500,  **'scrapy.contrib.downloadermiddleware.defaultheaders.DefaultHeadersMiddleware'**: 550,  *# 'scrapy.contrib.downloadermiddleware.redirect.RedirectMiddleware': 600,* **'scrapy.contrib.downloadermiddleware.cookies.CookiesMiddleware'**: 700,  **'scrapy.contrib.downloadermiddleware.httpproxy.HttpProxyMiddleware'**: 750,  **'scrapy.contrib.downloadermiddleware.httpcompression.HttpCompressionMiddleware'**: 800,  **'scrapy.contrib.downloadermiddleware.stats.DownloaderStats'**: 850,  **'scrapy.contrib.downloadermiddleware.httpcache.HttpCacheMiddleware'**: 900, } |

# scrapy使用

## 命令

调试：Scrapy shell [useragent] <http://www.baidu.com> --nolog

退出：exit()

查看爬虫状态：est()

创建爬虫项目：scrapy startproject name

查看版本：scrapy version

下载网页通过浏览器打开：scrapy view <http://www.baidu.com>

项目命令：

进入爬虫项目：cd projectname

硬件测试命令：scrapy bench

创建爬虫文件：Scrapy genspider

查看爬虫模板：scrapy genspider -l

依据模板创建爬虫：scrapy genspider -t basic filename(爬虫文件名) baidu.com(域名)

检测爬虫：scrapy check filename

运行爬虫文件：Scrapy crawl f1 --nolog(不显示日志信息)

列出当前项目下的爬虫文件:scrapy list

编辑爬虫文件：scrapy edit filname

获取网址：Scrapy parse <http://www.baidu.com>

### 暂停与重启

先在项目目录下创建一个文件夹job\_file，每个爬虫状态再用一个小文件夹来保存，如果想用的爬虫要用重新启动要指定不同的文件夹

Scrapy crawl spider lagou -s JOBDIR=job\_file/001

Ctrl c 来暂停，scrapy会有一些善后动作，不要两次Ctrl+c,会杀死进程，最好采用命令行形式，重新启动只需要再运行上面的命令。

或者在settings中设置：

JOBDIR=’job\_file/001’

## Cookie操作

#请求Cookie

Cookie = response.request.headers.getlist('Cookie')

响应Cookie

Cookie = response.headers.getlist('Set-Cookie')

Settings设置# 自动传递cookies与打印cookies设置

# Disable cookies (enabled by default)

COOKIES\_ENABLED = True

COOKIES\_DEBUG = True

### CookieJar对象：

from scrapy.http.cookies import CookieJar

meta = {'dont\_merge\_cookies': True, 'cookiejar': cookie\_jar}#首次请求

之后的每次请求

Request(login\_url, meta = {'dont\_merge\_cookies': True, 'cookiejar' : response.meta['cookiejar']},

### 存储cookie

#登录成功后，得到要保存的cookies，为CookieJar对象

cookie\_jar = response.meta['cookiejar']

cookie\_jar.extract\_cookies(response, response.request)

可以通过CookieJar.\_cookies方法将cookies转换为字典类型进行存储

Request(url='http://weibo.com',cookies=cookie\_jar,callback=self.logined)

也可以通过遍历的方式将各条cookies分别存储

with open(self.cookie\_file, 'wb+') as f:

for cookie in cookie\_jar:

f.write(str(cookie) + '\n')

### 从文件读取cookie

于登录的cookies的domain都是.weibo.com

with open(self.cookie\_file) as f:

cookiejar = f.read()

p = re.compile('\<Cookie (.\*?) for .weibo.com\/\>')

cookies = re.findall(p, cookiejar)

cookies = (cookie.split('=') for cookie in cookies)

cookie\_jar = dict(cookies)

## User-Agent和IP代理

### Useragent代理插件

fake-useragent(开源的自动切换useragent的库)：

pip install scrapy-fake-useragent

**class** RandomUserAgentMiddlware(object):  
 **def** \_\_init\_\_(self,crawlar):  
 super(RandomUserAgentMiddlware,self).\_\_init\_\_()  
 self.ua=UserAgent()  
 self.ua\_type = crawlar.settings.get(**'RANDOM\_UA\_TYPE'**,**'ramdom'**)  
 @classmethod  
 **def** from\_crawler(cls,crawler):  
 **return** cls(crawler)

**def** process\_request(self,request,spider):  
 **def** get\_ua():  
 **return** getattr(self.ua,self.ua\_type)  
 request.headers.setdefault(**'User-Agent'**,get\_ua())

### Ip代理插件

1 Scrapy-proxies

2 官方插件：scrapy-crawlera

3 免费代理：IPProxyTool

4 Tor工具（洋葱网络，多次转发达到匿名）

## Signals和扩展

### Signals

Middleware就是一种扩展

**scrapy.signals.engine\_started() Scrapy启动时发送信号，有时候会在[spider\_opened](http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/signals.html" \l "std:signal-spider_opened)之后发送；**

**scrapy.signals.engine\_stopped() Scrapy停止时发送信号；**

**scrapy.signals.item\_scraped(item, response, spider) 当item被爬取，并通过所有 [Item Pipeline](http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/item-pipeline.html" \l "topics-item-pipeline) 后(没有被丢弃(dropped)，发送该信号；**

**scrapy.signals.item\_dropped(item, exception, spider) item用完丢弃时；**

**scrapy.signals.spider\_closed(spider,reason) 某个spider被关闭时，reason关闭原因，完成爬取关闭值为'finished'，被引擎关闭值为'cancelled'，引擎关闭值为'shutdown'**

**scrapy.signals.spider\_opened(spider) spieder开始爬取时；**

**scrapy.signals.spider\_idle(spider) spider空闲时：requests正在等待被下载，requests正在被调度，items正在pipline中处理，用途：在 [spider\_idle](http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/signals.html" \l "std:signal-spider_idle) 处理器中调度某些请求来避免spider被关闭；**

**scrapy.signals.spider\_error(failure, response, spider) 抛异常时；**

**scrapy.signals.request\_scheduled(request, spider) 引擎调度一个 [Request](http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/request-response.html" \l "scrapy.http.Request" \o "scrapy.http.Request) 对象用于下载时；**

**scrapy.signals.response\_received(response, request, spider) 引擎从downloader获取到一个新的 [Response](http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/request-response.html" \l "scrapy.http.Response" \o "scrapy.http.Response) 时；**

**scrapy.signals.response\_downloaded(response, request, spider) 当一个 HTTPResponse 被下载时，由downloader发送。**

### 扩展

扩展在扩展类被实例化时加载和激活。 因此，所有扩展的实例化代码必须在类的构造函数(\_\_init\_\_)中执行。

  在 [EXTENSIONS](http://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/settings.html" \l "std:setting-EXTENSIONS) 中，每个扩展都使用一个字符串表示，即扩展类的全Python路径。 比如:

EXTENSIONS **=** {

'scrapy.contrib.corestats.CoreStats': 500,

'scrapy.webservice.WebService': 500,

'scrapy.telnet.TelnetConsole': 500,}

扩展实例：

|  |
| --- |
| **from** scrapy **import** signals  **from** scrapy.exceptions **import** NotConfigured  **class** **SpiderOpenCloseLogging**(object):  **def** **\_\_init\_\_**(self, item\_count):  self**.**item\_count **=** item\_count  self**.**items\_scraped **=** 0  @classmethod  **def** **from\_crawler**(cls, crawler):  *# NotConfigured otherwise*  **if** **not** crawler**.**settings**.**getbool('MYEXT\_ENABLED'):  **raise** NotConfigured  *# get the number of items from settings*  item\_count **=** crawler**.**settings**.**getint('MYEXT\_ITEMCOUNT', 1000)  *# instantiate the extension object*  ext **=** cls(item\_count)  *# connect the extension object to signals*  crawler**.**signals**.**connect(ext**.**spider\_opened, signal**=**signals**.**spider\_opened)  crawler**.**signals**.**connect(ext**.**spider\_closed, signal**=**signals**.**spider\_closed)  crawler**.**signals**.**connect(ext**.**item\_scraped, signal**=**signals**.**item\_scraped)  **return** ext  **def** **spider\_opened**(self, spider):  spider**.**log("opened spider %s" **%** spider**.**name)  **def** **spider\_closed**(self, spider):  spider**.**log("closed spider %s" **%** spider**.**name)  **def** **item\_scraped**(self, item, spider):  self**.**items\_scraped **+=** 1  **if** self**.**items\_scraped **%** self**.**item\_count **==** 0:  spider**.**log("scraped %d items" **%** self**.**items\_scraped) |

## Url去重：

自己实现去重要实现scrapy.core.scheduler的enqueue\_request

方法下的request\_seen方法

## Tips

### 状态数据收集器：

**def** \_\_init\_\_(self):

self.fall\_urls=[]

**def** parse(self, response):  
 *#收集404的url及页面数*  
 **if** response.status == 404:  
 self.start\_urls.append((response.url))  
 self.crawler.stats.inc\_value(**'faild\_url'**)

### 重定向 异常代码：

REDIRECT\_ENABLED = False

或者：

meta = { 'dont\_redirect': True, # 禁止网页重定向

'handle\_httpstatus\_list': [301, 302] # 对哪些异常返回进行处理

}

HTTPERROR\_ALLOWED\_CODES

可以通过判断response.status == 403抛出CloseSpider异常

### 自定义settings：

custom\_settings = {  
 **'ITEM\_PIPELINES'**:{  
 **'bricCrawler.pipelines.intoByDbNamePipLine'**: 100,  
 }  
}

### 自动下载延迟

AutoThrottle 扩展：

Settings中的下载延迟设置 秒为单位

*DOWNLOAD\_DELAY = 3*

*开启自动延迟 默认为false：*

*[AUTOTHROTTLE\_ENABLED](https://doc.scrapy.org/en/latest/topics/autothrottle.html" \l "std:setting-AUTOTHROTTLE_ENABLED)=True*

*初始下载延迟 默认为5秒*

AUTOTHROTTLE\_START\_DELAY

最大下载延迟：默认60秒

AUTOTHROTTLE\_MAX\_DELAY

每个域的并发请求数

[CONCURRENT\_REQUESTS\_PER\_DOMAIN](https://doc.scrapy.org/en/latest/topics/settings.html" \l "std:setting-CONCURRENT_REQUESTS_PER_DOMAIN)

每个ip的并发请求数

[CONCURRENT\_REQUESTS\_PER\_IP](https://doc.scrapy.org/en/latest/topics/settings.html" \l "std:setting-CONCURRENT_REQUESTS_PER_IP)

# Selenium phantomjs

**Scrapy自带一种scrapy-splash，支持分布式**

**Selenium grid也支持分布式**

**Splinter 很好用的纯python的操纵浏览器的，类似与selenium**

PhantomJS是一个基于webkit的JavaScript API。它使用QtWebKit作为它核心浏览器的功能，使用webkit来编译解释执行JavaScript代码。任何你可以在基于webkit浏览器 做的事情，它都能做到。它不仅是个隐形的浏览器，提供了诸如CSS选择器、支持Web标准、DOM操作、JSON、HTML5、Canvas、SVG等， 同时也提供了处理文件I/O的操作，从而使你可以向操作系统读写文件等。PhantomJS的用处可谓非常广泛，诸如前端无界面自动化测试（需要结合 Jasmin）、网络监测、网页截屏等。

安装：

安装Selenium 用Pip 安装 如果Pip不能被找到 记得设置环境变量Python/Script

Selenium是一个浏览器虚拟器，可以通过Selenium在各种浏览器上模拟各种行为。python中通过Selenium使用PhantomJS抓取动态网页数据时需要安装以下库：   
1. Beautifulsoup，用于解析网页内容   
2. [Node.js](http://lib.csdn.net/base/nodejs" \o "Node.js知识库" \t "http://blog.csdn.net/jinping_shi/article/details/_blank)   
3. 安装好Node.js之后通过Node.js安装PhantomJS。在Mac终端中输入npm -g install phantomjs即可（Windows下的cmd也是一样）   
4. 安装Selenium ，pip install -U selenium

下载phantomjs好解压后，把文件夹bin中的phantomjs.exe移到python文件夹中的Scripts中，至此，就已经在Win的环境下配置好了环境

如果想使用Chrome来实现解析的，请下载GG浏览器的相应驱动（同时需要安装chrome浏览器），如上图中的第一个程序，第二个程序就是PhantomJS了。

如果想使用Firefox来实现解析，则需要安装Selenium提供的firefox插件，位于python安装目录Lib\site-packages\selenium\webdriver\firefox\webdriver.xpi,请拖入到firefox浏览器中，然后重启浏览器即可

## API简介

### 一. 定位元素方法

        官网地址：[http://selenium-python.readthedocs.org/locating-elements.html](http://selenium-python.readthedocs.org/locating-elements.html" \t "http://blog.csdn.net/eastmount/article/details/_blank)  
        这里有各种策略用于定位网页中的元素(locate elements)，你可以选择最适合的方案，Selenium提供了一下方法来定义一个页面中的元素：

find\_element\_by\_id

find\_element\_by\_name

find\_element\_by\_xpath

find\_element\_by\_link\_text

find\_element\_by\_partial\_link\_text

find\_element\_by\_tag\_name

find\_element\_by\_class\_name

find\_element\_by\_css\_selector

        下面是查找多个元素（这些方法将返回一个列表）：

find\_elements\_by\_name

find\_elements\_by\_xpath

find\_elements\_by\_link\_text

find\_elements\_by\_partial\_link\_text

find\_elements\_by\_tag\_name

find\_elements\_by\_class\_name

find\_elements\_by\_css\_selector

        除了上面给出的公共方法，这里也有两个在页面对象定位器有用的私有方法。这两个私有方法是find\_element和find\_elements。  
        常用方法是通过xpath相对路径进行定位，同时CSS也是比较好的方

### 二. 操作元素方法

        在讲述完定位对象(locate elements)之后我们需要对该已定位对象进行操作，通常所有的操作与页面交互都将通过WebElement接口，常见的操作元素方法如下：

clear 清除元素的内容

send\_keys 模拟按键输入

click 点击元素

submit 提交表单

方法clear()清除输入框默认内容，如“请输入密码”等提示，通过send\_keys("\*\*")输入正确的用户名和密码，最后通过click()点击登录按钮或send\_keys(Keys.RETURN)相当于回车登录，submit()提交表单。  
        PS：如果需要输入中文，防止编码错误使用send\_keys(u"中文用户名")

### 三. WebElement接口获取值

        通过WebElement接口可以获取常用的值，这些值同样非常重要。

size 获取元素的尺寸

text 获取元素的文本

get\_attribute(name) 获取属性值

location 获取元素坐标，先找到要获取的元素，再调用该方法

page\_source 返回页面源码

driver.title 返回页面标题

current\_url 获取当前页面的URL

is\_displayed() 设置该元素是否可见

is\_enabled() 判断元素是否被使用

is\_selected() 判断元素是否被选中

tag\_name 返回元素的tagName

### 四. 鼠标操作

        在现实的自动化[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/eastmount/article/details/_blank)中关于鼠标的操作不仅仅是click()单击操作，还有很多包含在ActionChains类中的操作。如下：

context\_click(elem) 右击鼠标点击元素elem，另存为等行为

double\_click(elem) 双击鼠标点击元素elem，地图web可实现放大功能

drag\_and\_drop(source,target) 拖动鼠标，源元素按下左键移动至目标元素释放

move\_to\_element(elem) 鼠标移动到一个元素上

click\_and\_hold(elem) 按下鼠标左键在一个元素上

perform() 在通过调用该函数执行ActionChains中存储行为

1. action.send\_keys('v') #另存为
2. action.perform()

  效果如下图所示，通过xpath定位到图片位置并右击鼠标，在弹出的菜单中选择“另存为图片”。但是如何点击“另存为对话框”的“保存”按钮是个难点，目前刚学习阶段，境界没到无法解决。原因：  
        WebDriver cannot directly interact with dialog windows this is because dialog windows are the domain of the operating system and not the webpage.

该部分推荐参考资料：  
            [selenium 右键下载图片，结合sikuli - tobecrazy](http://www.cnblogs.com/tobecrazy/p/4523314.html" \t "http://blog.csdn.net/eastmount/article/details/_blank)

<http://www.cnblogs.com/tobecrazy/p/4523314.html>  
            [Selenium WebDriver 中鼠标和键盘事件分析及扩展](http://www.ibm.com/developerworks/cn/java/j-lo-keyboard/" \t "http://blog.csdn.net/eastmount/article/details/_blank)

<http://www.ibm.com/developerworks/cn/java/j-lo-keyboard/>  
            [Selenium Windows Save/Open Open Dialouge - StackOver](http://stackoverflow.com/questions/15576430/selenium-windows-save-open-open-dialouge" \t "http://blog.csdn.net/eastmount/article/details/_blank)

<http://stackoverflow.com/questions/15576430/selenium-windows-save-open-open-dialouge>

### 五. 键盘操作

        参考：[http://selenium-python.readthedocs.org/api.html](http://selenium-python.readthedocs.org/api.html" \t "http://blog.csdn.net/eastmount/article/details/_blank)  
        前面讲述了鼠标操作，现在讲述键盘操作。在webdriver的Keys类中提供了键盘所有的按键操作，当然也包括一些常见的组合键操作如Ctrl+A(全选)、Ctrl+C(复制)、Ctrl+V(粘贴)。更多键参考官方文档对应的编码。

send\_keys(Keys.ENTER) 按下回车键

send\_keys(Keys.TAB) 按下Tab制表键

send\_keys(Keys.SPACE) 按下空格键space

send\_keys(Kyes.ESCAPE) 按下回退键Esc

send\_keys(Keys.BACK\_SPACE) 按下删除键BackSpace

send\_keys(Keys.SHIFT) 按下shift键

send\_keys(Keys.CONTROL) 按下Ctrl键

send\_keys(Keys.ARROW\_DOWN) 按下鼠标光标向下按键

send\_keys(Keys.CONTROL,'a') 组合键全选Ctrl+A

send\_keys(Keys.CONTROL,'c') 组合键复制Ctrl+C

send\_keys(Keys.CONTROL,'x') 组合键剪切Ctrl+X

send\_keys(Keys.CONTROL,'v') 组合键粘贴Ctrl+V

## PhantomJs集成scrapy

Github上的ptantomjs继承scrapy的中间件：Scrapy-phantomjs-downloader

异步使用phantomjs

Spider中：

**from** selenium **import** webdriver  
**from** scrapy.xlib.pydispatch **import** dispatcher  
**from** scrapy **import** signals

**def** \_\_init\_\_(self):  
 super(JobboleSpider,self).\_\_init\_\_()  
 *#设置chrome不显示界面，只能在linux下使用* **from** pyvirtualdisplay **import** Display  
 display = Display(visible=0, size=(800,800))  
 display.start()  
 *#使用信号机制，爬虫停止时关闭chrome* self.browser = webdriver.Chrome(executable\_path=**''**)  
 dispatcher.connect(self.spider\_colsed,signals

.spider\_closed)

**def** spider\_colsed(self,spider):  
 *#当爬虫退出时退出* print(**'spider close'**)  
 self.browser.quit()

Middleware中：

**from** scrapy.http **import** HtmlResponse  
*#通过selenium请求动态网页***class** JsPageMiddleware(object):  
 **def** process\_request(self,request,spider):  
 **if** spider == **"jobbole"**:  
 spider.browser.get(request.url)  
 **import** time  
 time.sleep(3)  
 print(**'访问{0}'**.format(request.url))  
 **return** HtmlResponse(url=spider.browser.current\_url,body=spider.browser.page\_source,encoding=**'utf-8'**)

## 浏览器窗口操作：

Chrome不显示界面：

只在linux下有用：

*#设置chrome不显示界面，只能在linux下使用*

*Pip install pyvirtualdisplay***from** pyvirtualdisplay **import** Display  
display = Display(visible=0, size=(800,800))  
display.start()

browser = webdriver.Chrome(executable\_path=**''**)

### 最大化：

browser.maximize\_window()

### 固定宽高：

browser.set\_window\_size(480, 800) #参数数字为像素点

### 浏览器前进后退：

browser.back()

browser.forward()

## 常用操作：

### 最大化。截图：

browser.get\_screenshot\_as\_file(**'login1.png'**)

### Cookies 操作：

获取：

cookie = [item[**"name"**] + **"="** + item[**"value"**] **for** item **in** browser.get\_cookies()]  
cookiestr = **';'**.join(item **for** item **in** cookie)

设置cookie：driver.add\_cookie(cookie)

browser.get(url)*# 删除原来的cookie*browser.delete\_all\_cookies()  
*# 携带cookie打开*browser.add\_cookie({**'name'**:**'ABC'**,**'value'**:**'DEF'**})  
*# 通过js新打开一个窗口*

newwindow=**'window.open("https://www.baidu.com");'**browser.execute\_script(newwindow)  
input(**"查看效果"**)  
browser.quit()

### 抛异常前等待：

Java： driver.manage().timeouts().implicitlyWait(5,TimeUnit.***SECONDS***);

Python： browser.set\_page\_load\_timeout(2)

driver**.**implicitly\_wait(10) driver**.**get("http://somedomain/url\_that\_delays\_loading")

### 设置请求头：

Phantomjs:  
**from** selenium.webdriver.common.desired\_capabilities **import** DesiredCapabilities  
dcap = dict(DesiredCapabilities.PHANTOMJS)  
dcap[**"phantomjs.page.settings.userAgent"**] = (  
**"Mozilla/5.0 (Linux; A..."**)  
driver = webdriver.PhantomJS(executable\_path=**''**

,sdesired\_capabilities=dcap)

Chrome:options = webdriver.ChromeOptions()*# 进入浏览器设置*options.add\_argument(**'lang=zh\_CN.UTF-8'**)*# 设置中文*  
options.add\_argument(**'user-agent="Mozilla/5.0 (iPod; U; CPU iPhone OS 2\_1.."'**)*# 更换头部*  
browser = webdriver.Chrome(chrome\_options=options)

### Chrome不加载图片：

options = webdriver.ChromeOptions()  
prefs = {  
 **'profile.default\_content\_setting\_values'**: {  
 **'images'**: 2  
 }  
}

或者：

prefs={**'profile.managed\_default\_content\_settings.images'**:2}  
options.add\_experimental\_option(**'prefs'**, prefs)  
browser = webdriver.Chrome(chrome\_options=options)

### 执行JS

下拉滚动：browser.execute\_script(**"windows.scrollTo(0,document.body.scrollHeight); var lenOfPage=document.body.scrollHeight; return lenOfPage"**)

## 大神博客

<http://www.cnblogs.com/gooseeker/p/5646444.html>

Selenium常见操作：<http://blog.csdn.net/eastmount/article/details/48108259>

书籍《selenium2 python自动化测试》 作者：虫师

# 常用技巧

### Unicode编码：

data=json.dumps(intoDB)  
logging.info(**'传送数据{}'**.format(data.encode(**'raw-unicode-escape'**)))

### 时间减一天：

yesterday=self.today-datetime.timedelta(days=1)

例：

now = datetime.datetime.now()-datetime.timedelta(days=1)

day1 = now.strftime(**'%Y%m%d'**)

### Xpath中使用正则：

xpath(  
 **r'//div[@class="xh-nei-text"]/div/div[contains(strong/text(), "金乡：")]/following-sibling::div[2]/text()'**,  
 namespaces={**"re"**: **"http://exslt.org/regular-expressions"**})

### requests处理gb2312编码的网页：

response = requests.get(**'http://www.51garlic.com/jiage/kwsj1/'**)  
response.encoding = **'gb2312'**text = response.text

### Python处理gb2312编码：

1.

import chardet

......

        content\_type = chardet.detect(html\_content)

        #print(content\_type['encoding'])

        if content\_type['encoding'] != "UTF-8":

            html\_content = html\_content.decode(content\_type['encoding'])

        html\_content = html\_content.encode("utf-8")

        open(filename,"wb").write(html\_content)

2.

page=response.body.decode(response.encoding)

res=Selector(text=page)

res.xpath()

### 时间格式化为带中文：

today = datetime.datetime.now().strftime(**'%Y{y}%m{m}%d{d}'**)

.format(y=**'年'**, m=**'月'**, d=**'日'**)

### Scrapy处理json

json\_body=json.loads(response.body\_as\_unicode())

### 中文转url编码：

**from** urllib.parse **import** quote

quote(keyword)