法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：大数据分析挖掘
  - 新浪微博：ChinaHadoop

分布式爬虫

大纲

- 表单及登录
- 动态页面抓取

表单及登录

HTML 提交数据

- form 表单

```
<form>
        <input type="text" name=”username">
</form>
```

HTML 的标签，由浏览器实现POST方法

- ajax 请求

```
$.(ajax){
}
```

基于ajax技术，异步发送http请求并获得返回数据，然后利用JavaScript对网页进行处理

HTML 表单

<form></form> tag 包围

里面包含了表单的元素：

<select> <textarea> <input> 等

<input> type 包含了多种输入类型，例如：

- <input type="text" name="name"> 文本输入框
- <input type="password" name="password"> 密码输入框
- <input type="submit" value="Submit"> 提交按钮

表单类型: form-data

POST /start.htm HTTP/1.1

Host: 192.168.1.1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Accept-Language: zh-CN,en-US;q=0.8,en;q=0.6

Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

Cache-Control: no-cache

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="name”

chacha

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="password”

chinahadoop

------WebKitFormBoundary7MA4YWxkTrZu0gW--

表单类型

- **form-data**

  http请求中的**multipart/form-data**，它会将表单的数据处理为一条消息，以标签为单元，用分隔符分开。既可以上传键值对，也可以上传文件。当上传的字段是文件时，会有Content-Type来表名文件类型。 由于有boundary隔离，所以multipart/form-data既可以上传文件，也可以上传键值对，它采用了键值对的方式，所以可以上传多个文件。

- **x-www-form-urlencoded**

  application/x-www-from-urlencoded，会将表单内的数据转换为键值对

表单类型: form-data

POST /start.htm HTTP/1.1

Host: 192.168.1.1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Accept-Language: zh-CN,en-US;q=0.8,en;q=0.6

Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

Cache-Control: no-cache

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="name”

chacha

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="password”

chinahadoop

------WebKitFormBoundary7MA4YWxkTrZu0gW--

表单类型: x-www-form-urlencoded

POST /start.htm HTTP/1.1

Host: 192.168.1.1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

'content-type': "application/x-www-form-urlencoded",

Cache-Control: no-cache

name=chacha&password=chinahadoop

ajax 登录

一般以 json 的方式提交数据

```
POST /start.htm HTTP/1.1
Host: 192.168.1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Cache-Control: no-cache

{"name":"chacha", "password":"chinahadoop"}
```

登录

1. 根据 chrome inspector 里检查到的参数，来设置登录方式
   - 最常用的是 x-www-form-urlencoded 和 json 方式，两种方式只是body编码不同
   - 如果是 form-data，按照form-data的方式组合body
   - body 部分经常需要按照特定的方式编码，比如 base64，或者 md5
   - 对于非常复杂的登录协议，利用第5章的动态网页方式登录
2. request = urllib2.Request(url, data, headers = loginheaders)
   - url 是访问的地址
   - data 是 body 部分
   - headers = loginheaders 是HTTP请求的 HEADER 数据

表单登录

```
import urllib2

url = http://jc.lo/dev/login/login.php
headers = {
    'host': "jc.lo",
    'connection': "keep-alive",
    'cache-control': "no-cache",
    'content-type': "application/x-www-form-urlencoded",
    'upgrade-insecure-requests': "1",
    'accept-language': "zh-CN,en-US;q=0.8,en;q=0.6",
}
data = {'name':'caca', "password":'c'}
payload = urllib.urlencode(data)
request = urllib2.Request(url, payload , headers=headers)
response = urllib2.urlopen(request)
```

获取并设置Cookie

登录的核心是为了获得 Cookie

登录成功后，HEADER 会有设置cookie的相关信息

```
Set-Cookie:main[UTMPKEY]=1381959; path=/; domain=.newsmth.net
Set-Cookie:main[UTMPNUM]=14820; path=/; domain=.newsmth.net
Set-Cookie:main[UTMPUSERID]=abc; path=/; domain=.newsmth.net
```

此时我们需要把服务器返回的 Cookie 信息，写入到我们后续请求的HEADER 的 Cookie 里

意外？？

当遇到网页登录后，返回302跳转的情况下，urllib2 的 Response 会丢失 Set-Cookie 的信息，导致登录不成功

更通常的情况下，我们需要一个通用的能处理 Cookie 的工具

- 自动处理 Set-Cookie 请求
- 自动处理管理过期 Cookie
- 自动在对应的域下发送特殊 Cookie

使用 urllib2 的插件功能

urllib2 可以绑定一系列的处理对象 handler，hanlder 可以对urllib2 的http 请求提供额外的功能支持。handler 是以绑定的顺序依次调用的。例如 HTTPRedirectHandler 用来处理跳转的情况，ProxyHandler 用于处理代理 。可以开发自定义的 handler （从 BaseHandler继承）

build_opener([*handler*, ...]) # 注册一系列的 handler

```
proxy_handler = urllib2.ProxyHandler({'http': 'http://www.example.com:3128/'})
proxy_auth_handler = urllib2.ProxyBasicAuthHandler()
proxy_auth_handler.add_password('realm', 'host', 'username', 'password')
opener = urllib2.build_opener(proxy_handler, proxy_auth_handler)
# This time, rather than install the OpenerDirector, we use it directly:
opener.open('http://www.example.com/login.html')
```

CookieJar

```
import cookielib

cj = cookielib.CookieJar() #创建 CookieJar 对象
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj)) #注册插件
request = urllib2.Request(url, payload, headers=headers)
response = opener.open(request)

print response.items() #打印headers信息
print response.read() #打印返回的网页
#打印 cookie 内容
for cookie in cj:
    print cookie.name, cookie.value, cookie.domain
```

动态网页

动态网页的使用场景

- 单页模式

  单页模式指的是不需要外部跳转的网页，例如个人设置中心经常就是单页

- 页面交互多的场景

  一部分网页上，有很多的用户交互接口，例如去哪儿的机票选择网页，用户可以反复修改查询的参数

- 内容及模块丰富的网页

  有些网页内容很丰富，一次加载完对服务器压力很大，而且方式延时也会很差；用户往往也不会查看所有内容

动态网页带来的挑战

对于爬虫：

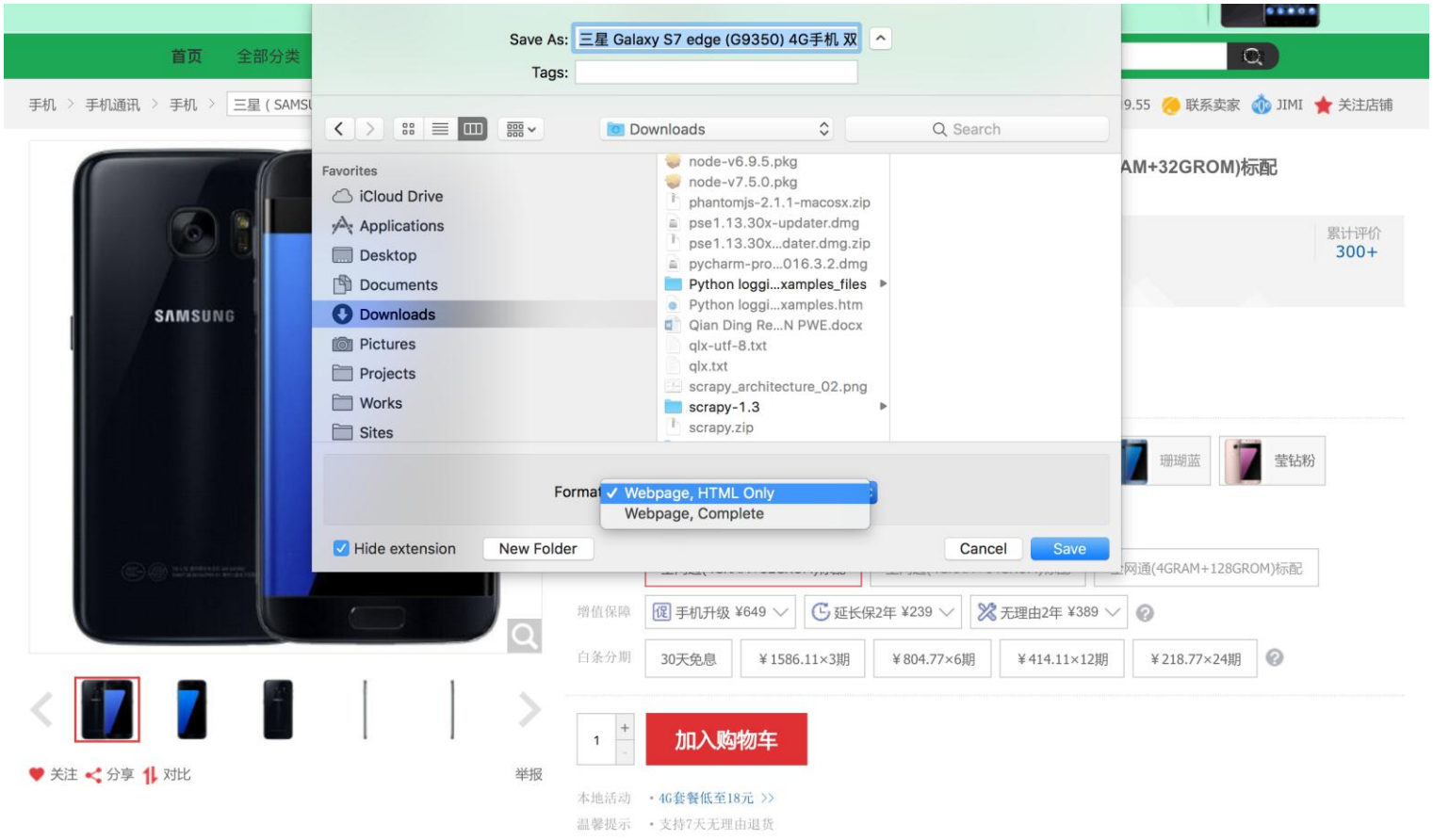
- 简单下载HMTL已经不行了，必须得有一个Web容器来运行HTML的脚本
- 增加了爬取的时间
- 增加了计算机的CPU、内存的资源消耗
- 增加了爬取的不确定性

对于网站：

- 为了配合搜索引擎的爬取，与搜索相关的信息会采用静态方式
- 与搜索无关的信息，例如商品的价格、评论，仍然会使用动态加载

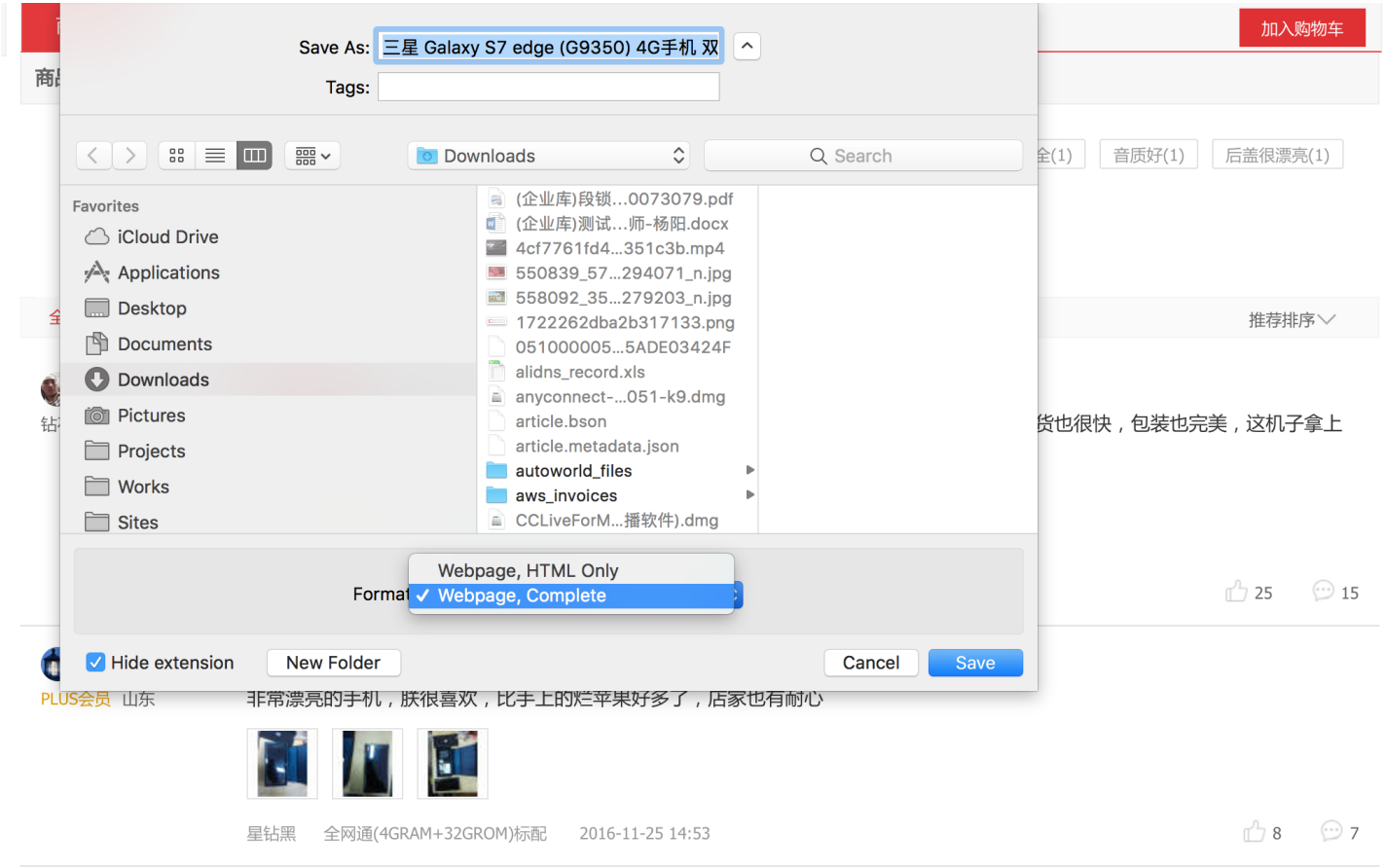
抓取动态网页 – 分析

打开目标网页后，直接右键点击，只保存HTML



分析动态网页 – 分析

把网页滑动到页面的最下面，然后再次保存，这次选择保存完整网页



分析动态网页 – 对比

- 使用 BeyondCompare 或者 SVN 等工具，来对比网页，大致找出动态加载的部分
- 针对要提取的部分，分别查看html only 与 full webpage，找出动态数据的部分
- 记录下它们的 class 或 id，试着用以下代码来提取，如果不能提取，说明是动态的：

```
from lxml import etree

f = open('./s7-full.htm')
c = f.read().decode('gbk')
f.close()

e = etree.HTML(c)
print e.xpath(u'//span[@class="price J-p-10524731933"]')
```

Python Web 引擎

- PyQt PySide：基于QT的python web引擎，需要图形界面的支持，需要安装大量的以来。安装和配置复杂，尤其是安装图形系统，对于服务器来说代价很大
- Selenium：一个自动化的Web测试工具，可以支持包括 Firefox、chrome、PhatomJS、IE 等多种浏览器的连接与测试
- PhantonJs：一个基于Webkit的 Headless 的Web引擎，支持JavaScript

PhantomJS + Selenium

安装

Selenium

pip install selenium

PhantomJS

- PhantomJS 需要先安装 nodejs

```
# yum install nodejs
```

- 为了加速，将NPM的源改为国内的淘宝

```
$ npm install -g cnpm --registry=https://registry.npm.taobao.org
```

- 利用NPM的Package Manager 安装 phantomjs

```
$ npm -g install phantomjs-prebuilt
```

使用 PhantomJS 来加载动态页面

```
# import webdriver from selenium
from selenium import webdriver

# load PhantomJS driver
driver = webdriver.PhantomJS(service_args=['--ignore-ssl-errors=true'])
# set window size, better to fit the whole page in order to
# avoid dynamically loading data
driver.set_window_size(1280, 2400)  # optional
# data page content
driver.get(cur_url)
# use page_source to get html content
content = driver.page_source
```

set_window_size

对于动态网页，有可能存在大量数据是根据视图来动态加载的，PhantomJS 允许客户端设置用来模拟渲染页面的窗口的尺寸，这个尺寸如果设置比较小，我们就不得用 javascript 的 scroll 命令来模拟页面往下滑动的效果以显示更多内容，所以我们可以设置一个相对大的窗口高度来渲染

```
driver.set_window_size(1280, 2400)  # optional
```

Built-in DOM selector

Selenium 实现了一系列的类似于 xpath 选择器的方法，使得我们可以直接调用 driver. find_element() 来进行元素的选择，但是这些都是基于Python的实现，执行效率非常低，大约是基于C 的 正则表达式或 lxml 的10倍的时间，因此**不建议使用built-in的选择器，而是采用 lxml 或者 re 对 driver.page_source （html文本）进行操作**

find_element(self, by='id', value=None)
find_element_by_class_name(self, name)
find_element_by_id(self, id_)
find_element_by_css_selector(self, css_selector)

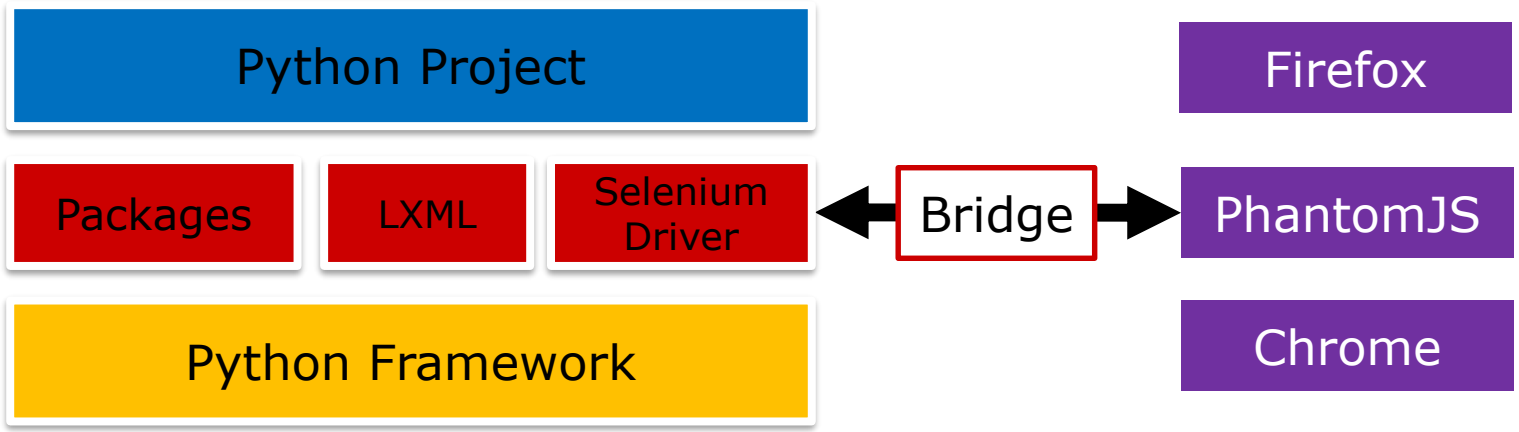
Useful Methods & Properties

Selenium 通过浏览器的驱动，支持大量的HTML及Javascript的操作，常用的可以包括：

- page_source: 获取当前的 html 文本
- title：HTML 的 title
- current_url：当前网页的URL
- get_cookie() & get_cookies()：获取当前的cookie
- delete_cookie() & delete_all_cookies()：删除所有的cookie
- add_cookie()：添加一段cookie
- set_page_load_timeout()：设置网页超时
- execute_script()：同步执行一段javascript命令
- execute_async_script()：异步执行javascript命令

Close and Clear

Selenium 通过内嵌的浏览器 driver 与浏览器进程通信，因此在退出的时候必须调用driver.close() 及 driver.quit() 来退出 PhantomJS，否则 PhantomJS 会一直运行在后台并占用系统资源。



Close and Clear

1. *send_signal is recommended*

   driver.service.process.send_signal(signal.SIGTERM)
2. *driver.close() this is not guaranteed to close PhantomJS*
3. *to assure it's closed, run below command in terminal*

   pgrep phantomjs | xargs kill

提取动态数据

1. 加载的过程中，根据网络环境的优劣，会存在一些延时，因此要多次尝试提取，提取不到不意味着数据不存在或者网络出错
2. 动态页面的元素，所使用的 id 或 class 经常会不止一个，例如京东一件商品的“好评率”，class 包括了 rate 和 percent-con 两种，因此需要对两种情况都进行尝试。更通用的情况，如果一个元素不能找到而 selenium 并没有报网络错误，那么有可能这个元素的class或id有了新的定义，我们需要将找不到的页面及元素信息记录在日志里，使得后续可以分析，找出新的定义并对这一类页面重新提取信息

提取动态数据

```
# try several times to extract data
while number_tried < constants['MAX_PAGE_TRIED']:
    try:
        price_element = driver.find_element_by_class_name('J-p-%s' % (item_id))
        price = re.findall('\d.*', price_element.get_attribute('innerHTML'))[0]
        item_name_element = driver.find_element_by_class_name('sku-name')
        item_name = item_name_element.text

        # item rating has 2 possible bound class, percent-con and rate
        # try to get rating values with both classes
        perc = re.findall('class="percent-con"', driver.page_source)
        if len(perc) > 0:
            element = driver.find_element_by_class_name('percent-con')
            rating = element.text
        else:
            element = driver.find_element_by_class_name('rate')
            rating = element.find_element_by_tag_name('strong').text
        break
    except selenium.common.exceptions.NoSuchElementException, msgs:
        number_tried += 1
        print msgs[1]
    except Exception, msgs:
        print msgs[1]
```

\ and \\

网页内，href 后面的链接可以有这样三种：

- **href="http://career.taobao.com"**

  http:// 是完整URL，直接跳转（经常外链会是绝对路径，比如引用到了WiKi、百科的一篇文章

- **href="\\detail.taobao.com\iuslkjsd"**

  // 是协议相关的绝对路径，如果现在是 https://xxx 则需要在 //etail.taobao.com 前面加上 https: 前面的协议也可以是 file:// ftp:// 所以这样会比较灵活

- **href="\i\8277375.html"**

  / 是网站的相对路径，需要在前面指明当前url的协议及domain

PhantomJS 配置

--ignore-ssl-errors=[true|false] ignores SSL errors, such as expired or self-signed certificate errors (default is false). Also accepted: [yes|no].

--load-images=[true|false] load all inlined images (default is true). Also accepted: [yes|no].

--disk-cache=[true|false] enables disk cache (at desktop services cache storage location, default is false). Also accepted: [yes|no].

--cookies-file=/path/to/cookies.txt specifies the file name to store the persistent Cookies.

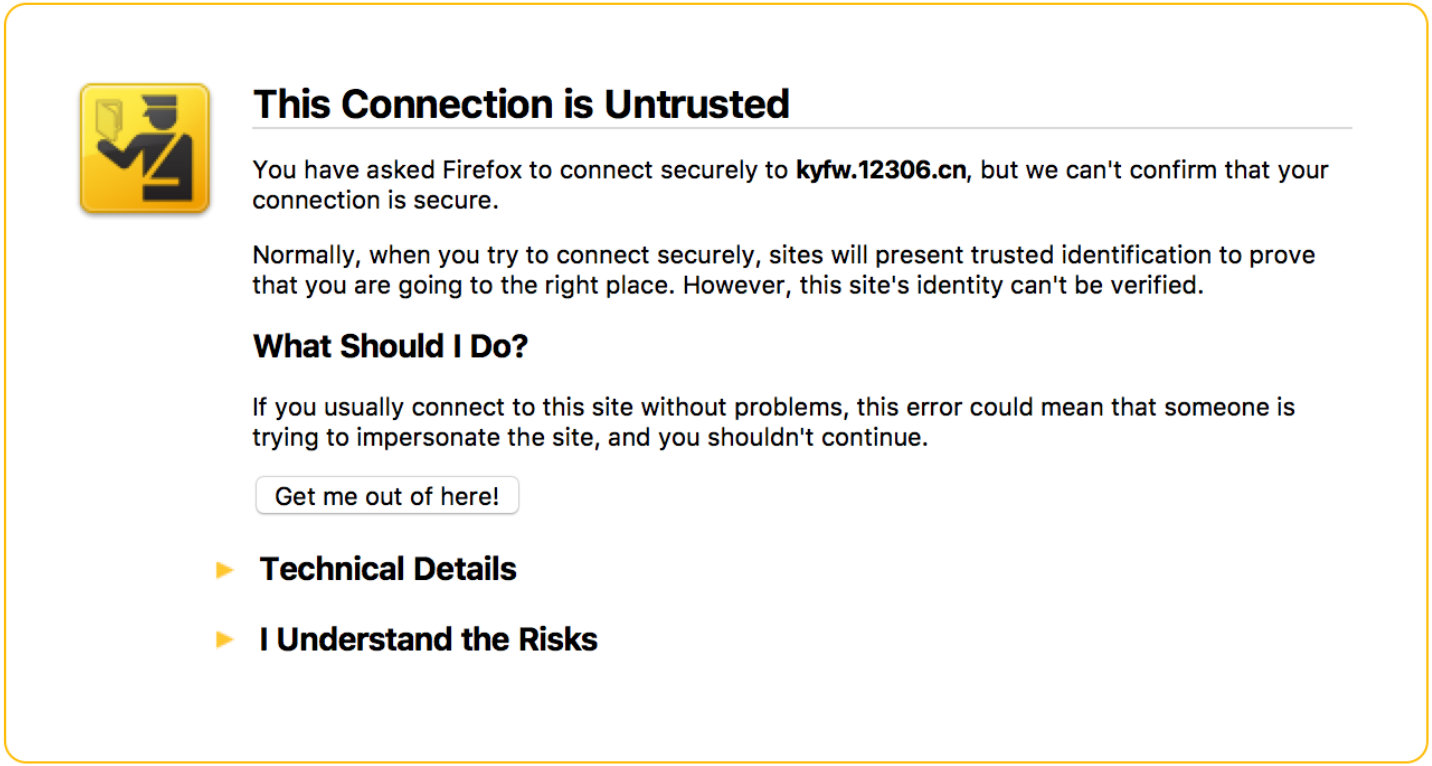
--debug=[true|false] prints additional warning and debug message, default is false. Also accepted: [yes|no].

--config specifies JSON-formatted configuration file (see below).

重要的配置-ignore-ssl-errors

--ignore-ssl-errors=[true|false]

一些证书没有获得CA授权（多是自己制作的证书），浏览器会报出证书不受信任，这种情况需要用户交互操作（点击继续或者新人），使用这个命令后，能自动忽略此类错误



重要的配置- load-images

--load-images=[true|false]

网页上一般都存在大量的图片，这些图片对我们第一次执行抓取是没有用的，在这种情况下，选择 --load-images=false 可以不下载这些图片，加快下载速度

重要的配置- config

--config=/path/to/config.json

```
{
    /* Same as: --ignore-ssl-errors=true */
    "ignoreSslErrors": true,
    /* Same as: --max-disk-cache-size=1000 */
    "maxDiskCacheSize": 1000,
    /* Same as: --output-encoding=utf8 */
    "outputEncoding": "utf8”
    /* etc. */
}
```

There are some keys that do not translate directly:

- --disk-cache => diskCacheEnabled
- --load-images => autoLoadImages
- --local-storage-path => offlineStoragePath
- --local-storage-quota => offlineStorageDefaultQuota
- --local-to-remote-url-access => localToRemoteUrlAccessEnabled
- --web-security => webSecurityEnabled
- --debug => printDebugMessages

如何抓取淘宝？

- 淘宝几乎所有信息都是 javascript 动态加载的，包括商品信息、描述、图片、评论等，因此用 urllib2 是不能抓取到淘宝的网页的
- 淘宝的外链包括 tmall.com 与 taobao.com，我们可以分别提取
- 对于有多个相同 class 的属性，利用 xpath 的路径来提取

  //dl[contains(@class, "tm-promo-cur")]//span[@class="tm-price"]

  / 单斜线代表从当前根路径匹配直接的子节点
  // 双斜线代表从当前路径匹配所有节点

  contains 表示属性只要包含一个即可
  @class= 要完整匹配，如果定义的 class="tm-promo-price tm-price" 则 contains 能匹配，而 @class=tm-price 则不能匹配

动态页面的抓取技巧

- Javascript 的执行是需要时间的，为了等待所有的内容都完成加载，需要设置一个等待 time.sleep()
- 使用 --ignore-images=true 来避免加载图片
- 每次初始化 webdriver 每次都会创建一个新的 PhantomJS 的进程，所以 webdriver 是可以重用的，一个 webdriver instance 相当于一个 chrome 的标签页，每次只需调用 get(url) 即可打开一个新的页面
- PhantomJS 运行需要占用比较多的系统资源，所以并发数需要视计算机性能以及实际测试情况而定，建议不要设置太大
- 程序退出后，考虑调用shell来杀掉所有 phantomjs 进程

```
subprocess.call('pgrep phantomjs | xargs kill')
```

疑问

- 问题答疑： http://www.xxwenda.com/
  - 可邀请老师或者其他人回答问题

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop