
作者 Andrew_liu (/users/4ee453b72aff) 2014.12.19 23:28*
写了67310字，被849人关注，获得了1081个喜欢
(/users

Python爬虫(七)--Scrapy模拟登录

1. Cookie原理

> HTTP是无状态的面向连接的协议, 为了保持连接状态, 引入了Cookie机制

Cookie是http消息头中的一种属性，包括：

- Cookie名字（Name）Cookie的值（Value）
- Cookie的过期时间（Expires/Max-Age）
- Cookie作用路径（Path）
- Cookie所在域名（Domain），使用Cookie进行安全连接（Secure）。
  前两个参数是Cookie应用的必要条件，另外，还包括Cookie大小（Size，不同浏览器对Cookie个数及大小限制是有差异的）。

更详细的cookie (http://en.wikipedia.org/wiki/HTTP_cookie)

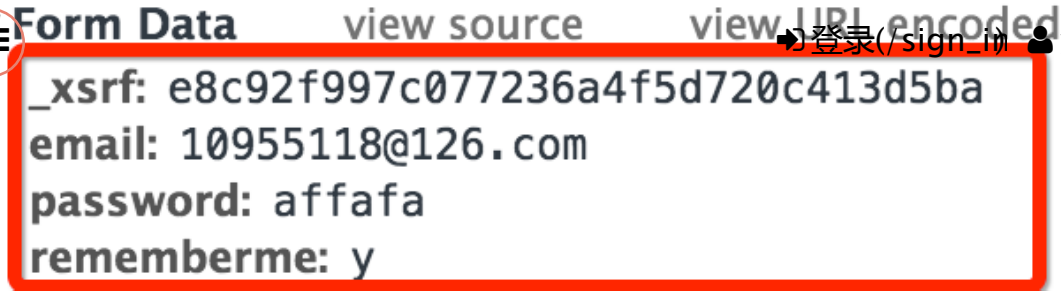
2. 模拟登陆

> 这次主要爬取的网站是知乎 (http://www.zhihu.com)

爬取知乎就需要登陆的, 通过之前的python内建库, 可以很容易的实现表单提交

现在就来看看如何通过Scrapy实现表单提交

首先查看登陆时的表单结果, 依然像前面使用的技巧一样, 故意输错密码, 方面抓到登陆的网页头部和表单(我使用的Chrome自带的开发者工具中的Network功能)



表单截图

查看抓取到的表单可以发现有四个部分:

- 邮箱和密码就是个人登陆的邮箱和密码
- rememberme字段表示是否记住账号
- 第一个字段是 `_xsrf` ,猜测是一种验证机制

> 现在只有 `_xsrf` 不知道，猜想这个验证字段肯定会实现在请求网页的时候发送过来，那么我们查看当前网页的源码(鼠标右键然后查看网页源代码，或者直接用快捷键)

```
<div class="wrapper index-content-wrapper">
<div class="top">
<div class="inner-wrapper">
<div class="form-wrapper" id="js-form-wrapper">
<div class="logo"></div>

<div id="js-sign-flow" class="desk-front sign-flow clearfix">
<div class="view view-signin ">
<form method="post" action="/login" class="zu-side-login-box">
<input type="hidden" name="_xsrf" value="e8c92f997c077236a4f5d720c413d5ba"/>
<div class="title clearfix">
<a class="js-signup signup-switch with-icon" href="#signup">注册<i class="icon-sign icon-sign-arrow"></i></a>
<a class="with-icon return">
<i class="icon-sign icon-sign-arrow-l"></i><span class="js-title-label">登录知乎</span>
</a>
</div>
<div class="email input-wrapper">
<input required type="email" name="email" placeholder="知乎注册邮箱"

>
</div>
<div class="input-wrapper">
<input required type="password" name="password" maxlength="128" placeholder="密码">
</div>
<div class="captcha-holder">
```

查询网页源码

发现我们的猜测是正确的

那么现在就可以来写表单登陆功能了

```
def start_requests(self):
        return [Request("https://www.zhihu.com/login", callback = self.post_login)]  #重写了

    #FormRequeset
    def post_login(self, response):
        print 'Preparing login'
        #下面这句话用于抓取请求网页后返回网页中的_xsrf字段的文字，用于成功提交表单
        xsrf = Selector(response).xpath('//input[@name="_xsrf"]/@value').extract()[0]
        print xsrf
        #FormRequeset.from_response是Scrapy提供的一个函数，用于post表单
        #登陆成功后，会调用after_login回调函数
        return [FormRequest.from_response(response,
                            formdata = {
                            '_xsrf': xsrf,
                            'email': '123456',
                            'password': '123456'
                            },
                            callback = self.after_login
                            )]
```

> 其中主要的功能都在函数的注释中说明

3. Cookie的保存

为了能使用同一个状态持续的爬取网站，就需要保存 `cookie` ，使用cookie保存状态，`Scrapy` 提供了cookie处理的中间件，可以直接拿来使用

CookiesMiddleware (http://doc.scrapy.org/en/0.24/topics/downloader-middleware.html?highlight=cookie#module-scrapy.contrib.downloadermiddleware.cookies)

> 这个cookie中间件保存追踪web服务器发出的cookie，并将这个cookie在接来下的请求的时候进行发送

Scrapy官方的文档中给出了下面的代码范例：

```
for i, url in enumerate(urls):
    yield scrapy.Request("http://www.example.com", meta={'cookiejar': i},
        callback=self.parse_page)

def parse_page(self, response):
    # do some processing
    return scrapy.Request("http://www.example.com/otherpage",
        meta={'cookiejar': response.meta['cookiejar']},
        callback=self.parse_other_page)
```

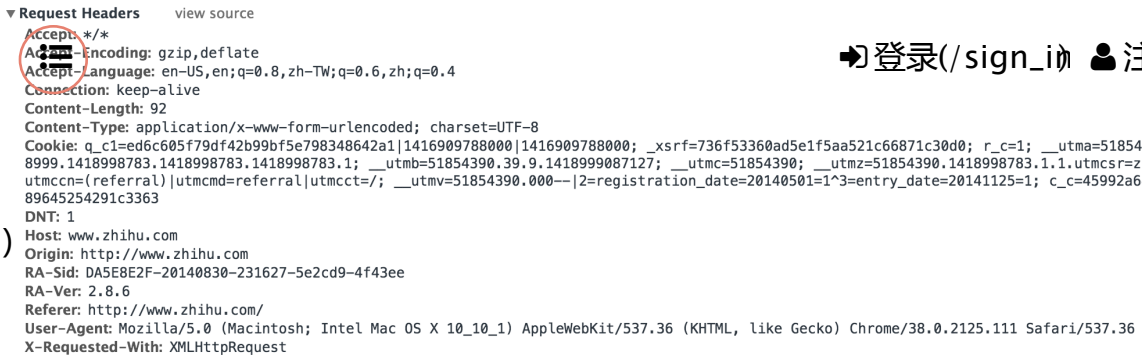
那么可以对我们的爬虫类中方法进行修改，使其追踪cookie

```
    #重写了爬虫类的方法，实现了自定义请求，运行成功后会调用callback回调函数
    def start_requests(self):
        return [Request("https://www.zhihu.com/login", meta = {'cookiejar' : 1}, callback =

    #FormRequeset出问题了
    def post_login(self, response):
        print 'Preparing login'
        #下面这句话用于抓取请求网页后返回网页中的_xsrf字段的文字，用于成功提交表单
        xsrf = Selector(response).xpath('//input[@name="_xsrf"]/@value').extract()[0]
        print xsrf
        #FormRequeset.from_response是Scrapy提供的一个函数，用于post表单
        #登陆成功后，会调用after_login回调函数
        return [FormRequest.from_response(response,   #"http://www.zhihu.com/login",
                            meta = {'cookiejar' : response.meta['cookiejar']}, #注意这里cook
                            headers = self.headers,
                            formdata = {
                            '_xsrf': xsrf,
                            'email': '123456',
                            'password': '123456'
                            },
                            callback = self.after_login,
                            dont_filter = True
                            )]
```

4. 伪装头部

有时候登陆网站需要进行头部伪装，比如增加防盗链的头部，还有模拟服务器登陆，这些都在前面的爬虫知识中提到过



Headers

为了保险，我们可以在头部中填充更多的字段，如下

```
    headers = {
    "Accept": "*/*",
    "Accept-Encoding": "gzip,deflate",
    "Accept-Language": "en-US,en;q=0.8,zh-TW;q=0.6,zh;q=0.4",
    "Connection": "keep-alive",
    "Content-Type":" application/x-www-form-urlencoded; charset=UTF-8",
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTMl
    "Referer": "http://www.zhihu.com/"
    }
```

在scrapy中 `Request` 和 `FormRequest` 初始化的时候都有一个headers字段，可以自定义头部，这样我们可以添加headers字段

形成最终版的登陆函数

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
from scrapy.contrib.spiders import CrawlSpider, Rule
from scrapy.selector import Selector
from scrapy.contrib.linkextractors.sgml import SgmlLinkExtractor
from scrapy.http import Request, FormRequest
from zhihu.items import ZhihuItem



class ZhihuSipder(CrawlSpider) :
    name = "zhihu"
    allowed_domains = ["www.zhihu.com"]
    start_urls = [
        "http://www.zhihu.com"
    ]
    rules = (
        Rule(SgmlLinkExtractor(allow = ('/question/\d+#.*?', )), callback = 'parse_page', f
        Rule(SgmlLinkExtractor(allow = ('/question/\d+', )), callback = 'parse_page', follo
    )
    headers = {
    "Accept": "*/*",
    "Accept-Encoding": "gzip,deflate",
    "Accept-Language": "en-US,en;q=0.8,zh-TW;q=0.6,zh;q=0.4",
    "Connection": "keep-alive",
    "Content-Type":" application/x-www-form-urlencoded; charset=UTF-8",
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTM
    "Referer": "http://www.zhihu.com/"
    }

    #重写了爬虫类的方法，实现了自定义请求，运行成功后会调用callback回调函数
    def start_requests(self):
        return [Request("https://www.zhihu.com/login", meta = {'cookiejar' : 1}, callback =

    #FormRequeset出问题了
    def post_login(self, response):
        print 'Preparing login'
        #下面这句话用于抓取请求网页后返回网页中的_xsrf字段的文字，用于成功提交表单
        xsrf = Selector(response).xpath('//input[@name="_xsrf"]/@value').extract()[0]
        print xsrf
        #FormRequeset.from_response是Scrapy提供的一个函数，用于post表单
        #登陆成功后，会调用after_login回调函数
        return [FormRequest.from_response(response,   #"http://www.zhihu.com/login",
                            meta = {'cookiejar' : response.meta['cookiejar']},
                            headers = self.headers,  #注意此处的headers
                            formdata = {
                            '_xsrf': xsrf,
                            'email': '1095511864@qq.com',
                            'password': '123456'
```

```
        },
        callback = self.after_login,
        dont_filter = True
        )]

    def after_login(self, response) :
        for url in self.start_urls :
            yield self.make_requests_from_url(url)

    def parse_page(self, response):
        problem = Selector(response)
        item = ZhihuItem()
        item['url'] = response.url
        item['name'] = problem.xpath('//span[@class="name"]/text()').extract()
        print item['name']
        item['title'] = problem.xpath('//h2[@class="zm-item-title zm-editable-content"]/text
        item['description'] = problem.xpath('//div[@class="zm-editable-content"]/text()').e
        item['answer']= problem.xpath('//div[@class=" zm-editable-content clearfix"]/text()
        return item
```

5. Item类和抓取间隔

完整的知乎爬虫代码链接 (https://github.com/Andrew-liu/scrapy_example/tree/master/zhihu)

```
from scrapy.item import Item, Field


class ZhihuItem(Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    url = Field()  #保存抓取问题的url
    title = Field()  #抓取问题的标题
    description = Field()  #抓取问题的描述
    answer = Field()  #抓取问题的答案
    name = Field()  #个人用户的名称
```

设置抓取间隔, 访问由于爬虫的过快抓取, 引发网站的发爬虫机制, 在 `setting.py` 中设置

```
BOT_NAME = 'zhihu'

SPIDER_MODULES = ['zhihu.spiders']
NEWSPIDER_MODULE = 'zhihu.spiders'
DOWNLOAD_DELAY = 0.25    #设置下载间隔为250ms
```

抓取结果(只是截取了其中很少一部分)

```
...
 'url': 'http://www.zhihu.com/question/20688855/answer/16577390'}
2014-12-19 23:24:15+0800 [zhihu] DEBUG: Crawled (200) <GET http://www.zhihu.com/question/20
[]
2014-12-19 23:24:15+0800 [zhihu] DEBUG: Scraped from <200 http://www.zhihu.com/question/206
    {'answer': [u'\u9009\u4f1a\u8ba1\u8fd9\u4e2a\u4e13\u4e1a\uff0c\u8003CPA\uff0c\u5165\u8d
                u'\u4e0d\u8fc7\u4f60\u8bf4\u6709\u4eba\u4ece\u5c0f\u5c31\u559c\u6b22\u8d22\u
                u'\u901a\u8fc7\u5b83\uff0c\u4f60\u53ef\u4ee5\u5b66\u4e60\u4efb\u4f55\u4e00\u
                u'\u5982\u679c\u5bf9\u4f1a\u8ba1\u7684\u8ba4\u8bc6\u4ec5\u4ec5\u505c\u7559\u
     'description': [u'\u672c\u4eba\u5b66\u4f1a\u8ba1\u6559\u80b2\u4e13\u4e1a\uff0c\u6df1\u
     'name': [],
     'title': [u'\n\n', u'\n\n'],
     'url': 'http://www.zhihu.com/question/20688855/answer/15861368'}
...
```

6. 存在问题

- Rule 设计不能实现全网站抓取，只是设置了简单的问题的抓取
- Xpath 设置不严谨，需要重新思考
- Unicode 编码应该转换成 UTF-8

推荐拓展阅读 (/sign_in)

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

¥ 打赏支持

蒙面人 (/users/09380fc94312)
2楼 · 2015.04.04 12:29 (/p/b7f41df6202d/comments/235426#comment-235426)
(/users/09380fc94312)

你好，请教如何把unicode转换成utf-8呢

♡ 喜欢(0) 回复

Andrew_liu (/users/4ee453b72aff)
3楼 · 2015.04.05 09:47 (/p/b7f41df6202d/comments/236621#comment-236621)
(/users/4ee453b72aff)

@蒙面人 (/users/09380fc94312) somestr.encode('utf-8') 就行了

♡ 喜欢(0) 回复

哈士奇_银桑 (/users/5f9288d93f31)
4楼 · 2015.10.19 17:49 (/p/b7f41df6202d/comments/763914#comment-763914)
(/users/5f9288d93f31)

请问，如何在简书中插入代码？

♡ 喜欢(0) 回复

> adonisjph (/users/757d0a4045f1)： @哈士奇_银桑 (/users/5f9288d93f31) 使用markdown ''' '''在此之间插入代码即可
> 2016.01.22 16:33 (/p/b7f41df6202d/comments/1295047#comment-1295047) 回复

添加新回复

926621f06141 (/users/926621f06141)
5楼 · 2016.01.12 21:56 (/p/b7f41df6202d/comments/1217937#comment-1217937)
(/users/926621f06141)

顶

♡ 喜欢(0) 回复

登录后发表评论 (/sign_in)

被以下专题收入，发现更多相似内容：

程序员 (/collection/NEt52a)
简书程序员大本营 投稿须知： 1.本专题仅收录与程序有关的文章。 2.请在代码框里贴代码，尽量保证可看性。
5701篇文章 (/collection/NEt52a) · 53492人关注
添加关注 (/sign_in)

干货 (/collection/f16b3d483ec2)
欢迎投稿 将实用的知识共享
2223篇文章 (/collection/f16b3d483ec2) · 17501人关注
添加关注 (/sign_in)

代码改变世界 (/collection/0f5e015fc36c)
Write the Code, Change the World, 改变世界从编程开始, 收集编程相关的干货
1323篇文章 (/collection/0f5e015fc36c) · 8732人关注
添加关注 (/sign_in)