爬虫案例3(模拟登陆,验证码登陆,cookie登陆)

http://example.webscraping.com/places/default/user/login

模拟登陆的网站

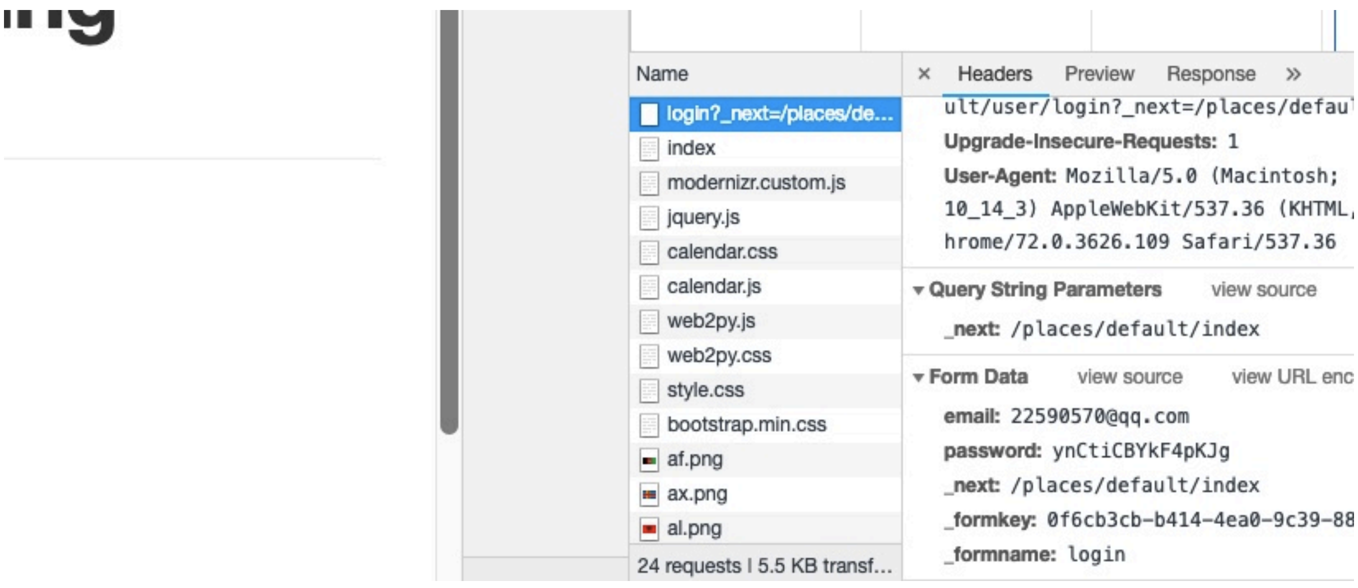
form表单说明下:

method 决定了http请求方法

action 决定了请求的url

enctype 决定了表单数据编码类型

input 决定了表单数据的内容



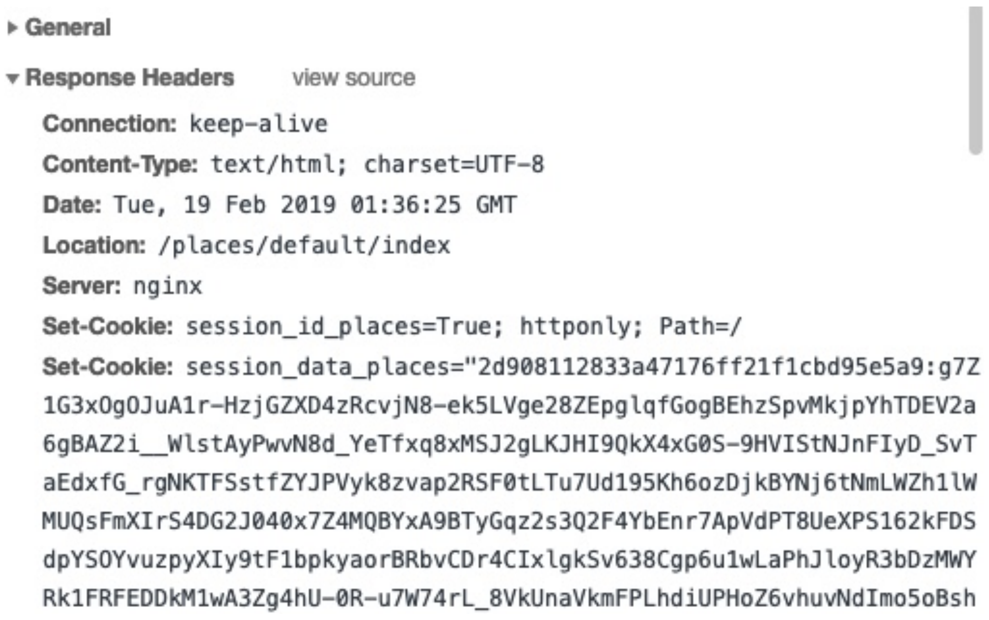
Name | Headers | Preview | Response

login?_next=/places/de...
index
modernizr.custom.js
jquery.js
calendar.css
calendar.js
web2py.js
web2py.css
style.css
bootstrap.min.css
af.png
ax.png
al.png
24 requests | 5.5 KB transf...

```
ult/user/login?_next=/places/defau
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh;
10_14_3) AppleWebKit/537.36 (KHTML,
hrome/72.0.3626.109 Safari/537.36

▼ Query String Parameters    view source
_next: /places/default/index

▼ Form Data    view source    view URL enc
email: 22590570@qq.com
password: ynCtiCBYkF4pKJg
_next: /places/default/index
_formkey: 0f6cb3cb-b414-4ea0-9c39-88
_formname: login
```

以上是请求信息

——————接下来分析响应信息

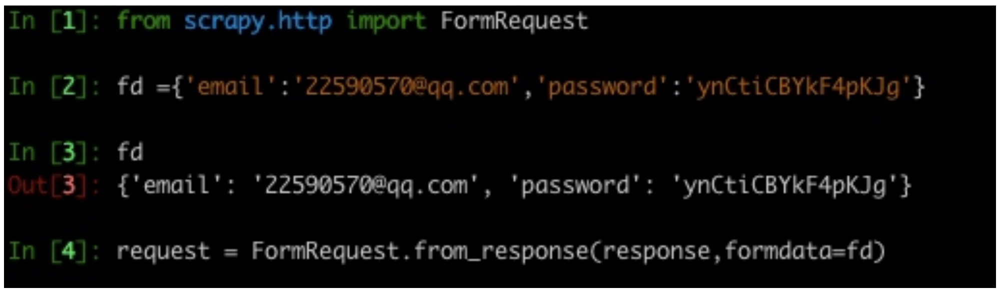


```
Connection: keep-alive
Content-Type: text/html; charset=UTF-8
Date: Tue, 19 Feb 2019 01:36:25 GMT
Location: /places/default/index
Server: nginx
Set-Cookie: session_id_places=True; httponly; Path=/
Set-Cookie: session_data_places="2d908112833a47176ff21f1cbd95e5a9:g7Z
1G3x0g0JuA1r-HzjGZXD4zRcvjN8-ek5LVge28ZEpglqfGogBEhzSpvMkjpYhTDEV2a
6gBAZ2i__WlstAyPwvN8d_YeTfxq8xMSJ2gLKJHI9QkX4xG0S-9HVIStNJnFIyD_SvT
aEdxfG_rgNKTFSstfZYJPVyk8zvap2RSF0tLTu7Ud195Kh6ozDjkBYNj6tNmLWZh1lW
MUQsFmXIrS4DG2J040x7Z4MQBYxA9BTyGqz2s3Q2F4YbEnr7ApVdPT8UeXPS162kFDS
dpYSOYvuzpyXIy9tF1bpkyaorBRbvCDr4CIxlgkSv638Cgp6u1wLaPhJloyR3bDzMWY
Rk1FRFEDDkM1wA3Zg4hU-0R-u7W74rL_8VkUnaVkmFPLhdiUPHoZ6vhuvNdImo5oBsh
```

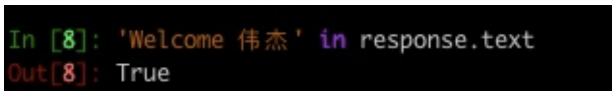
——控制台操作

```
scrapy shell http://example.webscraping.com/places/default/user/login?_next=/places/default/index
```

接下来模拟



```
In [1]: from scrapy.http import FormRequest

In [2]: fd ={'email':'22590570@qq.com','password':'ynCtiCBYkF4pKJg'}

In [3]: fd
Out[3]: {'email': '22590570@qq.com', 'password': 'ynCtiCBYkF4pKJg'}

In [4]: request = FormRequest.from_response(response,formdata=fd)
```



```
In [8]: 'Welcome 伟杰' in response.text
Out[8]: True
```

```
----代码如下
```

```python
# -*- coding: utf-8 -*-
import scrapy
from ..items import LogindemoItem
from scrapy.http import  Request,FormRequest
class LoginSpider(scrapy.Spider):
    name = 'login'
    allowed_domains = ['example.webscraping.com']
    #这个是登陆之后请求的详情页面
    #目的就是主要是为了能够访问登陆之后能够看到的页面
    start_urls = ['http://example.webscraping.com/places/default/user/profile']

    def parse(self, response):
        value = response.css('#auth_user_email__row > td.w2p_fw::text').extract_first()
        loginMM = LogindemoItem()
        loginMM['email_value'] = value
        yield loginMM

    #设置登陆页面的url
    login_url = 'http://example.webscraping.com/places/default/user/login'

    #这个方法是爬虫进入后第一次执行的方法
    def start_requests(self):
        yield Request(self.login_url,callback=self.login)
    #调用login请求登陆
    def login(self,response):
        fd = {'email':'22590570@qq.com','password':'ynCtiCBYkF4pKJg'}
        yield  FormRequest.from_response(response,formdata=fd,callback=self.parse_login)

    #检查是否成功通过判断response里面的内容,调用父类的start_requests方法
    #父类的方法请求的是start_urls的地址
    def parse_login(self,response):
        if 'Welcome 伟杰' in response.text:
            yield  from  super().start_requests()




```

———识别验证码-------

需要用到的东西

OCR 验证码识别库

pytesseract 第三方库

`brew  tesseract`

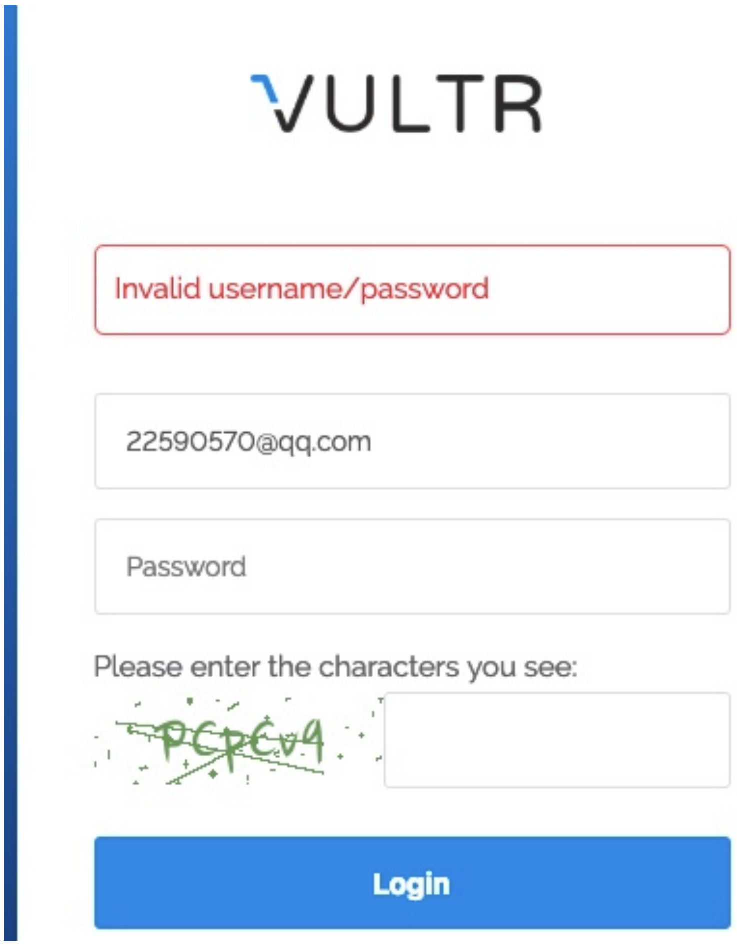
```
sudo pip3 install pillow
```

```
sudo pip3 install pytesseract
```

等等，我找个网站

https://my.vultr.com/

以这个网站为基础


Invalid username/password

22590570@qq.com

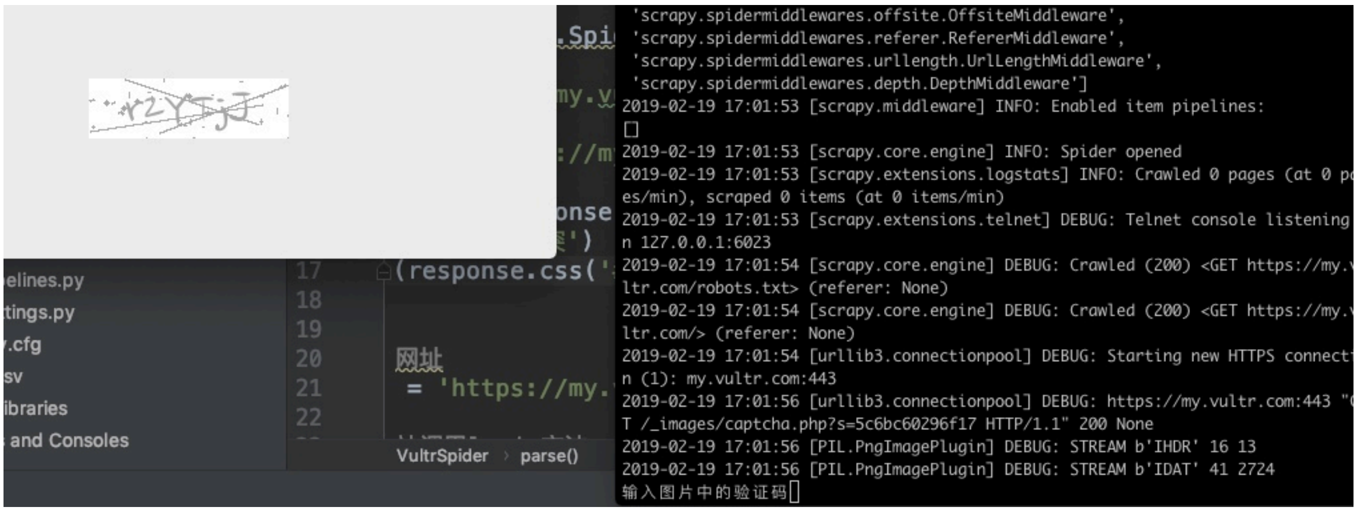
Password

Please enter the characters you see:

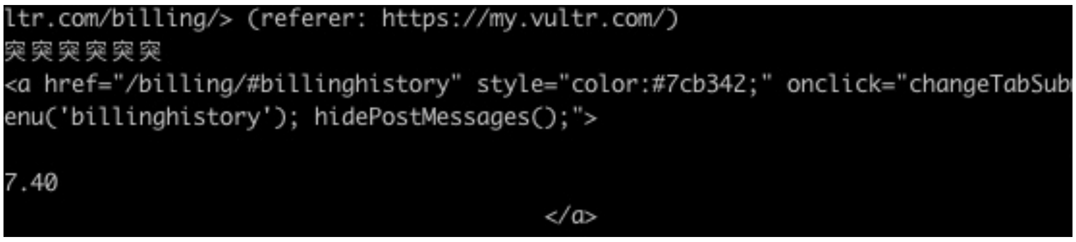
https://my.vultr.com/billing/ 这个是查看个人余额的页面

来走起，新建一个项目 核心代码如下

```python
# -*- coding: utf-8 -*-
import scrapy
import requests
from scrapy.http import Request,FormRequest
from PIL import Image
from io import BytesIO
import  pytesseract

class VultrSpider(scrapy.Spider):
    name = 'vultr'
    allowed_domains = ['my.vultr.com']
    #解析余额的页面
    start_urls = ['https://my.vultr.com/billing/']

    def parse(self, response):
        print('突突突突突突')
        print(response.css('#header2_right > div:nth-child(2) > a').extract_first())


    #设置登录的网址
    login_url = 'https://my.vultr.com/'

    #请求登录网站调用login方法
    def start_requests(self):
        yield  Request(self.login_url,callback=self.login,dont_filter=True)

    #login方法解析验证码和构建表单
    def login(self, response):
        #获取验证码图片url
        captchaUrl = response.css('#loggedout_module > form > div.captcha_container > img::attr(src)').extract_first()
        captchaUrl = response.urljoin(captchaUrl)
        #识别一下咯
        imgresponse = requests.get(captchaUrl)
        BytesIOObj = BytesIO(imgresponse.content)
        #BytesIOObj.write(imgresponse.content)
        img = Image.open(BytesIOObj)
        img = img.convert('L')
        img.show()
        #captcha = pytesseract.image_to_string(img)
        #改为手动输入
        captcha = input('输入图片中的验证码')
        print('验证码'+captcha)
        #构建表单数据
        fd = {'username': '22590570@qq.com', 'password': 'Wwjxly0215', 'captcha': captcha}
        print(fd)
        #因为vultr登录和访问进去后都是用一个网址
        #scrapy默认会过滤重复网页，发起Request添加dont_filter=True，则可以重复请求
        yield FormRequest.from_response(response, formdata=fd, callback=self.parse_login,dont_filter=True)
    #检查是否成功
    def parse_login(self,response):
        if 'Welcome to Vultr.com!' in response.text:
            yield from super().start_requests()
```



```
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2019-02-19 17:01:53 [scrapy.middleware] INFO: Enabled item pipelines:
□
2019-02-19 17:01:53 [scrapy.core.engine] INFO: Spider opened
2019-02-19 17:01:53 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pa
es/min), scraped 0 items (at 0 items/min)
2019-02-19 17:01:53 [scrapy.extensions.telnet] DEBUG: Telnet console listening
n 127.0.0.1:6023
2019-02-19 17:01:54 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://my.v
ltr.com/robots.txt> (referer: None)
2019-02-19 17:01:54 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://my.v
ltr.com/> (referer: None)
2019-02-19 17:01:54 [urllib3.connectionpool] DEBUG: Starting new HTTPS connecti
n (1): my.vultr.com:443
2019-02-19 17:01:56 [urllib3.connectionpool] DEBUG: https://my.vultr.com:443 "G
T /_images/captcha.php?s=5c6bc60296f17 HTTP/1.1" 200 None
2019-02-19 17:01:56 [PIL.PngImagePlugin] DEBUG: STREAM b'IHDR' 16 13
2019-02-19 17:01:56 [PIL.PngImagePlugin] DEBUG: STREAM b'IDAT' 41 2724
输入图片中的验证码
```



```
ltr.com/billing/> (referer: https://my.vultr.com/)
突突突突突突
<a href="/billing/#billinghistory" style="color:#7cb342;" onclick="changeTabSub
enu('billinghistory'); hidePostMessages();">

7.40
                                        </a>
```

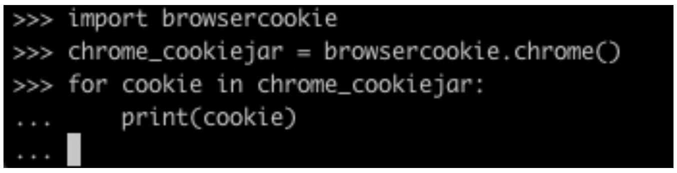
———cookie登录----

有些验证码特别难读，另外一种思路就是：

　　登录网站后，用户信息的cookie会被保存在本地，利用爬虫直接使用cookie发起http请求，就可以绕过表单登录

1.先安装第三方库，获取谷歌和火狐的cookie

pip3 install browsercookie



```
>>> import browsercookie
>>> chrome_cookiejar = browsercookie.chrome()
>>> for cookie in chrome_cookiejar:
...     print(cookie)
...
```

打是打出来了很多cookie信息

（这个思路也可以参考）

https://blog.csdn.net/fox64194167/article/details/81055327

———实现一个自定义的使用cookie中间件

在middlewares中

加入

```
import browsercookie
from scrapy.downloadermiddlewares.cookies import CookiesMiddleware
#实现cookie的中间件
class BrowserCookiesMiddleware(CookiesMiddleware):
    def __init__(self,debug=False):
        super().__init__(debug)
        self.load_browser_cookies()

    #自定义加载浏览器的cookie的方法
    def load_browser_cookies(self):
        jar = self.jars['chrome']
        chrome_cookiejar = browsercookie.chrome()
        for cookie in chrome_cookiejar:
            jar.set_cookie(cookie)
        jar = self.jars['firefox']
        firefox_cookiejar = browsercookie.firefox()
        for cookie in firefox_cookiejar:
            jar.set_cookie(cookie)
```

----接下来试一下模拟知乎中的cookie访问

1.创建项目：

scrapy startproject browser_cookie

```python
# -*- coding: utf-8 -*-

# Define here the models for your spider middleware
#
# See documentation in:
# https://doc.scrapy.org/en/latest/topics/spider-middleware.html

from scrapy import signals
import browsercookie
from scrapy.downloadermiddlewares.cookies import CookiesMiddleware


class BrowserCookieSpiderMiddleware(object):
    # Not all methods need to be defined. If a method is not defined,
    # scrapy acts as if the spider middleware does not modify the
    # passed objects.

    @classmethod
    def from_crawler(cls, crawler):
        # This method is used by Scrapy to create your spiders.
        s = cls()
        crawler.signals.connect(s.spider_opened, signal=signals.spider_opened)
        return s

    def process_spider_input(self, response, spider):
        # Called for each response that goes through the spider
        # middleware and into the spider.

        # Should return None or raise an exception.
        return None

    def process_spider_output(self, response, result, spider):
        # Called with the results returned from the Spider, after
        # it has processed the response.

        # Must return an iterable of Request, dict or Item objects.
        for i in result:
            yield i

    def process_spider_exception(self, response, exception, spider):
        # Called when a spider or process_spider_input() method
        # (from other spider middleware) raises an exception.

        # Should return either None or an iterable of Response, dict
        # or Item objects.
        pass

    def process_start_requests(self, start_requests, spider):
        # Called with the start requests of the spider, and works
        # similarly to the process_spider_output() method, except
        # that it doesn’t have a response associated.

        # Must return only requests (not items).
        for r in start_requests:
            yield r

    def spider_opened(self, spider):
        spider.logger.info('Spider opened: %s' % spider.name)


class BrowserCookieDownloaderMiddleware(object):
    # Not all methods need to be defined. If a method is not defined,
    # scrapy acts as if the downloader middleware does not modify the
    # passed objects.

    @classmethod
    def from_crawler(cls, crawler):
        # This method is used by Scrapy to create your spiders.
        s = cls()
        crawler.signals.connect(s.spider_opened, signal=signals.spider_opened)
        return s

    def process_request(self, request, spider):
        # Called for each request that goes through the downloader
        # middleware.

        # Must either:
        # - return None: continue processing this request
        # - or return a Response object
        # - or return a Request object
        # - or raise IgnoreRequest: process_exception() methods of
        #   installed downloader middleware will be called
        return None

    def process_response(self, request, response, spider):
        # Called with the response returned from the downloader.

        # Must either;
        # - return a Response object
        # - return a Request object
        # - or raise IgnoreRequest
        return response

    def process_exception(self, request, exception, spider):
        # Called when a download handler or a process_request()
        # (from other downloader middleware) raises an exception.

        # Must either:
        # - return None: continue processing this exception
        # - return a Response object: stops process_exception() chain
        # - return a Request object: stops process_exception() chain
        pass

    def spider_opened(self, spider):
        spider.logger.info('Spider opened: %s' % spider.name)


#实现cookie的中间件
```

```python
class BrowserCookiesMiddleware(CookiesMiddleware):
    def __init__(self,debug=False):
        super().__init__(debug)
        self.load_browser_cookies()

    #自定义加载浏览器的cookie的方法
    def load_browser_cookies(self):
        jar = self.jars['chrome']
        chrome_cookiejar = browsercookie.chrome()
        for cookie in chrome_cookiejar:
            jar.set_cookie(cookie)
        jar = self.jars['firefox']
        firefox_cookiejar = browsercookie.firefox()
        for cookie in firefox_cookiejar:
            jar.set_cookie(cookie)
```

2.在setting.py中添加一下配置

伪装浏览器

http://blog.51cto.com/laoyinga/2046970 随机

```python
ROBOTSTXT_OBEY = False
```

https://blog.csdn.net/you_are_my_dream/article/details/60479699 关于上面设置false的说明

```python
USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36'
DOWNLOADER_MIDDLEWARES = {
    'scrapy.downloadermiddlewares.cookies.CookiesMiddleware':None,
    'browser_cookie.middlewares.BrowserCookiesMiddleware': 5,
}
```

3.构建代码

```python
# -*- coding: utf-8 -*-
import scrapy


class ZhihuSpider(scrapy.Spider):
    name = 'zhihu'
    allowed_domains = ['www.zhihu.com']
    start_urls = ['https://www.zhihu.com/settings/account']

    def parse(self, response):
        print(response.css('#SettingsMain > div > div:nth-child(1) > div > div > div > div > h2'))

    #重写请求
    def start_requests(self):
        url = 'https://www.zhihu.com/settings/account'
        yield scrapy.Request(url,meta={'cookiejar':'chrome'},callback=self.parse)

```