爬虫学习

笔记本： reptile_draft
创建时间： 2021/7/3 9:37 **更新时间：** 2021/7/19 22:03
作者： 134exetj717
URL： about:blank

- 聚焦爬虫：爬取页面中的内容
    - 编码流程：
        - 指定url
        - 发起请求（请求的过程中，content返回的是中文，.json()返回的是json——以上对应于get
        - 获取相应数据
        - 数据解析（对于解析对象soup来说，.text代表其中的内容，包括中文，.contents代表列表
        - 进行持久化存储
- 数据解析分类：
    - 正则
    - bs4
    - xpath(***)
- 数据解析原理概述：
    - 解析的局部的文本内容都会在标签之间或者标签对应的属性中进行存储
    - 1.进行指定标签的定位
    - 2.标签或者标签对应的属性中存储的数据值进行提取（解析）
- 正则解析：

```
re.compile()
re.findall()
```

- bs4进行数据解析
    - 数据解析的原理：
        - 1.标签定位
        - 2.提取标签、标签属性中存储的数据值
    - bs4数据解析的原理：
        - 1.实例化一个BeautifulSoup对象，并且将页面源码数据解析到该对象中
        - 2.通过调用BeautifulSoup对象中的属性或者方法进行标签定位和数据提取
    - 环境安装：
        - pip install bs4
        - pic install lxml
    - 如何实例化BeautifulSoup：
        - from bs4 import BeautifulSoup
        - 对象的实例化
            - 1.将本地的html文档数据加载到该对象当中

```
fp = open('./test.html','r',encoding='utf-8')
soup = BeautifulSoup(fp,'lxml')
```

- 2.将互联网上获取的页面源码加载到该对象当中

```
page_text = response.text
```

```
soup = BeautifulSoup(page_text,'lxml')
```

- 提供的用于数据解析的方法和属性：

```
print(soup.a)          #返回的是 html 中第一次出现的tagName标签
find('tagName'):  等同于soup.div
soup.find():
   -find('tagName')：等同于soup.div()
   -属性定位：
         -soup.find('div',class_/id/attr='song')
   -soup.find_all('tagName')  :返回符合要求的所有标签（列表）
select：
   -select('某种选择器（id,class,标签...选择器）  ‘），返回的是一个列表
   -层级选择器：
      -soup.select('.tang>ul>li>a')：>表示的是一个层级
      -soup.select('.tang>ul a')：空格表示的多个层级
获取标签之间的文本数据：
   -soup.a.text/string/get_text()
   -text/get_text（）：可以获取某一个标签中所有的文本内容
   -string：只可以获取该标签下面的直系文本内容
获取标签中属性值：
   -soup.a['href']  获取 a 中的属性 href
```

- xpath解析：最常用且最便捷高效的一种解析方式。通用性。
  - xpath解析原理：
    1. 实例化一个etree的对象，且需要将被解析的页面源码数据加载到该对象中
    2. 调用etree对象中的xpath方法结合着xpath表达式事先标签的定位和内容的捕获
  - 环境的安装：
    - pip install lxml（解析器）
  - 如何实例化一个etree对象
    1. 将本地的html文档中的源码数据加载到etree对象中：etree.parse(filePath)
    2. 可以将从互联网上获取的源码数据加载到该对象中：etree.HTML('page_text')
    3. xpath（'xpath表达式'）
  - xpath表达式
    1. 实例化对象
    2. / 与 //的区别
    3. 属性定位
    4. 索引定位
    5. 取文本
    6. 取属性

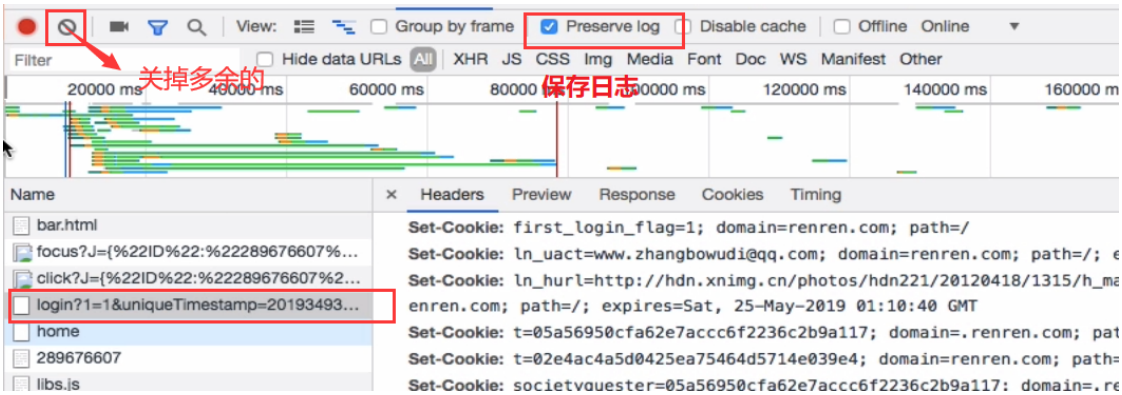
```
from lxml import etree
if __name__ == '__main__':
    # 实例化好了一个etree对象，且将被解析的原码加载到了该对象中
    tree = etree.parse('text.html')
    #第一个/表示从根节点开始寻找
    r = tree.xpath('/html/body/div')
    # //: 表示的是多个层级。可以表示从任意位置开始定位
    r = tree.xpath('html//div')
    r = tree.xpath('//div')
```

```
    # 属性定位
    r = tree.xpath('//div[@class="song"]')
    # 索引定位
    r = tree.xpath('//div[@class="tang"]//li[5]/a/text()')[0]
    # 取文本，- /text()获取的是标签中直系的文本内容，表示下一个内容为空，//text（）标签中非直系的文本内容，表示该内容下的并列内容，./表示从当前开始
    r = tree.xpath('//li[7]//text()')
    #取属性：    /@attrName  ==>img/src
    r =tree.xpath('//div[@class="song"]/img/@src')
```

- 验证码识别
  - 验证码和爬虫之间的爱恨情仇?
  - 反爬机制：验证码。识别验证码图片中的数据，用于模拟登录操作。
- 识别验证码的操作：

  1. 人工肉眼识别
  2. 第三方自动识别

- 云打码：https://www.yundama.com/demo.html，注意（现在网站用不了，
- 模拟登陆：
  - 爬取基于某些用户的用户信息
- 需求：对人人网进行模拟登陆（post请求一般会携带参数，form data）
  - 点击登录按钮之后会发起一个post请求
  - post的请求中会携带登录之前录入的相关的登录信息（用户名，密码，验证码.....）



  - 验证码：每次请求都会发生变化
- 模拟登录人人网

  1. 验证码的识别，获取验证码图片的文字数据
  2. 对post请求进行发送（处理请求参数）
  3. 对响应数据进行持久化存储

```
import requests
from lxml import etree

#1.对验证码图片进行捕获和识别
url
```

- for循环中，我们可以用 zip 来表示一对

```
for (pic_item,pic_name) in zip(src_list,src_name):
```