# 昨日回顾

```
1.概述什么是http协议
超文本传输协议
2.爬虫中常用头信息
请求头
User-Agent
Connection
响应头
Content-Type
3.https中涉及到的三种加密方式
 对称密钥加密
 非对称
 证书
 4.requests模块的作用及编码流程
 作用：模拟浏览器发送请求
 编码流程：
 1）指定URL
 2）发送请求
 3）获取响应数据
 4）持久化存储
 5.requests如何进行参数封装,为什么要进行参数封装
```

```
参数封装：
get：  params
post：data
为什么：
在我们发送请求时，服务器需要我们传递参数，如果不传递，则会请求失败
6.简述目前接触到的反爬机制及其反反爬策略
UA检测：加上UA伪装
robots.txt协议：直接无视
动态加载数据
7.什么是动态加载数据
在我们爬取网站上的数据时，有时会遇到通过AJAX发送请求动态获取的数据，需要在开发者工具中定位到XHR进行抓包获取
8.requests模块的响应数据类型
text
json()
content
```

# 聚焦爬虫

通用爬虫：爬取整个页面

聚焦爬虫：爬取页面中指定的数据

## 1 聚焦爬虫的概念:

- 爬取指定的页面数据()

## 2 聚焦爬虫的编码流程:

- 1 指定URL
- 2 发送请求
- 3 获取响应数据
- 4 进行数据解析
- 5 持久化存储

## 3 数据解析的三种方式:

- 正则解析：讲1个案例
- bs4解析：讲1个案例
- xpath解析：讲2-3个案例

## 4 为什么要用到数据解析(原理)

- 1.要进行标签定位
- 2.通过定位的标签取此标签里面的文本和属性值

## 爬取一张图片

```
import requests
import json

url = "https://pic.qiushibaike.com/system/pictures/12220/122203960/medium/L91K2OM72VA2EQ00.jpg"
header = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36'
}

response = requests.get(url=url).content

 with open('t.jpg','wb') as fp:
     fp.write(response)

```

使用urllib中request模块爬取

```
from urllib import request

url = "https://pic.qiushibaike.com/system/pictures/12220/122203960/medium/L91K2OM72VA2EQ00.jpg"

request.urlretrieve(url=url,filename='t.jpeg')

```

## 正则解析爬取图片

```
# 爬取糗事百科里面的糗图
import re
import os
import requests
from urllib import request

url = "https://www.qiushibaike.com/pic/"
headers = {
  "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36"
```

```
}
ex = '<div class="thumb">.*?<img src="(.*?)" alt=".*?</div>'

if not os.path.exists('./qiutu'):
 os.mkdir('./qiutu')

# 发送请求
page_text = requests.get(url=url, headers=headers).text

# 进行数据解析(正则)
url_list = re.findall(ex, page_text, re.S)

# 向图片发送请求并保存到本地
for img_url in url_list:
 img_url = "https:" + img_url
 name = img_url.split("/")[-1]

 request.urlretrieve(url=img_url, filename="./qiutu/" + name)

```

# bs4解析爬取文本

## 解析原理

```
安装:
pip install bs4
pip install lxml # 在进行bs4解析时，需要依赖此lxml库

导包: from bs4 import BeautifulSoup

bs4解析原理:
1. 实例化一个BeautifulSoup对象，同时需要将页面源码数据加载到该对象中
2. 需要通过该对象中的属性和方法进行页面源码的数据提取和解析

 如何实例化一个BeautifulSoup对象:
  本地加载
  f = open()
  soup = BeautifulSoup(f, 'lxml') # f是文件句柄
  网络加载
  soup = BeautifulSoup(page_text, 'lxml') # page_text是通过requests模块发送请求获取到的页面源码数据
```

**bs4对象的属性和方法**

（1）根据标签名查找

- soup.a　获取页面源码中第一个符合要求的标签

（2）获取属性，返回的永远是一个单数

- soup.a.attrs　获取a所有的属性和属性值，返回一个字典
- soup.a.attrs['href']　获取href属性
- soup.a['href']　也可简写为这种形式

（3）获取文本内容

- soup.string: 只可以获取直系标签中的文本内容
- soup.text: 获取标签下的所有文本内容
- soup.get_text(): 获取标签下的所有文本内容

【注意】如果标签还有标签，那么string获取到的结果为None，而其它两个，可以获取文本内容

（4）find()：找到第一个符合要求的标签，返回的永远是一个单数

- soup.find('a') 通过标签名进行数据解析

通过标签属性进行数据解析:

- soup.find('a', title="xxx")
- soup.find('a', alt="xxx")
- soup.find('a', class_="xxx")
- soup.find('a', id_="xxx")

（5）find_all：找到所有符合要求的标签，返回的永远是一个列表

- soup.find_all('a')
- soup.find_all(['a','b']) 找到所有的a和b标签
- soup.find_all('a', limit=2)　限制前两个

（6）根据选择器选择指定的内容

select选择器，返回的永远是一个列表: soup.select('#feng')

- 常见的选择器：标签选择器(a)、类选择器(.)、id选择器(#)、层级选择器
- 层级选择器：
  - 单层级: div > p > a > .lala　只能是下面一级
  - 多层级: div .dudu #lala .meme .xixi　下面好多级

【注意】select选择器返回永远是列表，需要通过下标提取指定的对象

测试的html页面，test_page.html:

```
<html lang="en">
```

```
<head>
    <meta charset="UTF-8" />
    <title>测试bs4</title>
</head>
<body>
    <div>
        <p>百里守约</p>
    </div>
     <div class="song">
        <p>李清照</p>
        <p>王安石</p>
        <p>苏轼</p>
        <p>柳宗元</p>
        <a href="http://www.song.com/" title="赵匡胤" target="_self">
            <span>this is span</span>
        宋朝是最强大的王朝，不是军队的强大，而是经济很强大，国民都很有钱</a>
        <a href="" class="du">总为浮云能蔽日,长安不见使人愁</a>
        <img src="http://www.baidu.com/meinv.jpg" alt="" />
     </div>
     <div class="song">111111111</div>
     <div class="tang">
        <ul>
            <li><a href="http://www.baidu.com" title="qing">清明时节雨纷纷,路上行人欲断魂,借问酒家何处有,牧童遥指杏花村</a></li>
            <li><a href="http://www.163.com" title="qin">秦时明月汉时关,万里长征人未还,但使龙城飞将在,不教胡马度阴山</a></li>
            <li><a href="http://www.126.com" alt="qi">岐王宅里寻常见,崔九堂前几度闻,正是江南好风景,落花时节又逢君</a></li>
            <li><a href="http://www.sina.com" class="du">杜甫</a></li>
            <li><a href="http://www.dudu.com" class="du">杜牧</a></li>
            <li><b>杜小月</b></li>
            <li><i>度蜜月</i></li>
            <li><a href="http://www.haha.com" id="feng">凤凰台上凤凰游,凤去台空江自流,吴宫花草埋幽径,晋代衣冠成古丘</a></li>
        </ul>
     </div>
</body>
</html>
```

用法举例：

```
from bs4 import BeautifulSoup

```

```
f = open('test_page.html','r',encoding='utf-8')

soup = BeautifulSoup(f,features='lxml')
# print(soup)
# print(soup.a)
# print(soup.a.attrs)
# print(soup.a.attrs['href'])
 # print(soup.a['href'])
 # # print(soup.text) # 获取标签下所有
 # 只能获取当前p标签的文本，p下的子标签文本无法获取
 print(soup.p.string)
 # # print(soup.get_text()) # 获取标签下所有
 # print(soup.find('div',class_ = 'song'))
 # print(soup.find_all('li'))
 # print(soup.find_all(['li','i','a'], limit = 3))
 # print(soup.find_all(['li','i','a']))
 # print(soup.select('.tang > ul > li'))
 # print(soup.select('.tang > ul > li > a'))
 # print(soup.select('.tang a'))
 print(soup.select('.tang i'))
 print(soup.select('.song > p')[2].text)
```

**爬取三国演义里面的数据**

```
# 爬取三国演义里面的数据http://www.shicimingju.com/book/sanguoyanyi.html

import requests
from bs4 import BeautifulSoup

url = "http://www.shicimingju.com/book/sanguoyanyi.html"
headers = {
 "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36"
}

page_text = requests.get(url=url, headers=headers).text

# 实例化一个bs4对象
soup = BeautifulSoup(page_text, 'lxml')
li_list = soup.select('.book-mulu li')

```

```python
f = open('./sanguo.txt', 'w', encoding="utf-8")

# 获取标题页里面的文章标题和详情页的URL
for li in li_list:
 title = li.a.text
 detail_url = "http://www.shicimingju.com" + li.a["href"]

 # 向详情页的URL发送请求，获取文章内容
 detail_page_text = requests.get(url=detail_url, headers=headers).text

 # 重新再实例化一个基于详情页的页面源码数据的soup对象
 detail_soup = BeautifulSoup(detail_page_text, 'lxml')
 content = detail_soup.find("div", class_="chapter_content").get_text()

 f.write(title + ":" + content + "\n")

f.close()

```