

# SPIDER-DAY01

## 1. 网络爬虫概述

### 【1】定义

- 1.1) 网络蜘蛛、网络机器人，抓取网络数据的程序
- 1.2) 其实就是用Python程序模仿人点击浏览器并访问网站，而且模仿的越逼真越好

### 【2】爬取数据的目的

- 2.1) 公司项目的测试数据，公司业务所需数据
- 2.2) 获取大量数据，用来做数据分析

### 【3】企业获取数据方式

- 3.1) 公司自有数据
- 3.2) 第三方数据平台购买(数据堂、贵阳大数据交易所)
- 3.3) 爬虫爬取数据

### 【4】Python做爬虫优势

- 4.1) Python：请求模块、解析模块丰富成熟，强大的Scrapy网络爬虫框架
- 4.2) PHP：对多线程、异步支持不太好
- 4.3) JAVA：代码笨重，代码量大
- 4.4) C/C++：虽然效率高，但是代码成型慢

### 【5】爬虫分类

- 5.1) 通用网络爬虫(搜索引擎使用，遵守robots协议)

robots协议：网站通过robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取，通用网络爬虫需要遵守robots协议(君子协议)

示例：<https://www.baidu.com/robots.txt>

- 5.2) 聚焦网络爬虫：自己写的爬虫程序

### 【6】爬取数据步骤

- 6.1) 确定需要爬取的URL地址
- 6.2) 由请求模块向URL地址发出请求，并得到网站的响应
- 6.3) 从响应内容中提取所需数据
  - a> 所需数据，保存
  - b> 页面中有其他需要继续跟进的URL地址，继续第2步去发请求，如此循环

## 2. 爬虫请求模块

## 2.1 requests模块

### ■ 安装

```
1  【1】 Linux
2      sudo pip3 install requests
3
4  【2】 Windows
5      方法1> cmd命令行 -> python -m pip install requests
6      方法2> 右键管理员进入cmd命令行 : pip install requests
```

## 2.2 常用方法

### ■ requests.get()

```
1  【1】 作用
2      向目标网站发起请求,并获取响应对象
3
4  【2】 参数
5      2.1> url : 需要抓取的URL地址
6      2.2> headers : 请求头
7      2.3> timeout : 超时时间,超过时间会抛出异常
```

### ■ 此生第一个爬虫

```
1  """
2  向京东官网 (https://www.jd.com/) 发请求,获取响应内容
3  """
4  import requests
5
6  resp = requests.get(url='https://www.jd.com/')
7  # 1.text属性: 获取响应内容-字符串
8  html = resp.text
9  print(html)
```

### ■ 响应对象 (res) 属性

```
1  【1】 text      : 字符串
2  【2】 content   : 字节流
3  【3】 status_code : HTTP响应码
4  【4】 url       : 实际数据的URL地址
```

### ■ 重大问题思考

网站如何来判定是人类正常访问还是爬虫程序访问? --检查请求头!!!

```

1 # 请求头 (headers) 中的 User-Agent
2 # 测试案例: 向测试网站http://httpbin.org/get发请求, 查看请求头(User-Agent)
3 import requests
4
5 url = 'http://httpbin.org/get'
6 res = requests.get(url=url)
7 html = res.text
8 print(html)
9 # 请求头中:User-Agent为-> python-requests/2.22.0 那第一个被网站干掉的是谁??? 我们是不是需要
   # 发送请求时重构一下User-Agent??? 添加 headers 参数!!!

```

#### ■ 重大问题解决 - headers参数

```

1 """
2 包装好请求头后,向测试网站发请求,并验证
3 养成好习惯, 发送请求携带请求头, 重构User-Agent
4 """
5 import requests
6
7 url = 'http://httpbin.org/get'
8 headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML,
   like Gecko) Chrome/14.0.835.163 Safari/535.1'}
9 html = requests.get(url=url,headers=headers).text
10 # 在html中确认User-Agent
11 print(html)

```

## 3. 爬虫编码模块

#### ■ urllib.parse模块

```

1 1、标准库模块: urllib.parse
2 2、导入方式:
3 import urllib.parse
4 from urllib import parse

```

#### ■ 作用

```

1 给URL地址中查询参数进行编码
2
3 # 示例
4 编码前: https://www.baidu.com/s?wd=美女
5 编码后: https://www.baidu.com/s?wd=%E7%BE%8E%E5%A5%B3
6

```

### 3.1 urlencode()

#### ■ 作用

```
1 给URL地址中查询参数进行编码，参数类型为字典
2
```

## ■ 使用方法

```
1  # 1、URL地址中 一个查询参数
2  编码前: params = {'wd': '美女'}
3  编码中: params = urllib.parse.urlencode(params)
4  编码后: params结果: 'wd=%E7%BE%8E%E5%A5%B3'
5
6  # 2、URL地址中 多个查询参数
7  编码前: params = {'wd': '美女', 'pn': '50'}
8  编码中: params = urllib.parse.urlencode(params)
9  编码后: params结果: 'wd=%E7%BE%8E%E5%A5%B3&pn=50'
10 发现编码后会自动对多个查询参数间添加 & 符号
11
```

## ■ 拼接URL地址的三种方式

```
1  # url = 'http://www.baidu.com/s?'
2  # params = {'wd': '赵丽颖'}
3  # 问题: 请拼接出完整的URL地址
4  *****
5  params = urllib.parse.urlencode(params)
6  【1】字符串相加
7  【2】字符串格式化 (占位符 %s)
8  【3】format()方法
9  'http://www.baidu.com/s?{}'.format(params)
10
11  【练习】
12  进入瓜子二手车直卖网官网 - 我要买车 - 请使用3种方法拼接前20页的URL地址,从终端打印输出
13  官网地址: https://www.guazi.com/langfang/
14
15  url = 'https://www.guazi.com/bj/buy/o{}/#bread'
16  for o in range(1, 21):
17      page_url = url.format(o)
18      print(page_url)
19
```

## ■ 练习

```
1  """
2  问题: 在百度中输入要搜索的内容,把响应内容保存到本地文件
3  编码方法使用 urlencode()
4  """
5  import requests
6  from urllib import parse
7
8  # 1. 拼接URL地址
9  word = input('请输入搜索关键字:')
10  params = parse.urlencode({'wd': word})
11  url = 'http://www.baidu.com/s?{}'.format(params)
12
13  # 2. 发请求获取响应内容
```

```

14 headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML,
15    like Gecko) Chrome/14.0.835.163 Safari/535.1'}
16
17 # 3. 保存到本地文件
18 filename = '{}.html'.format(word)
19 with open(filename, 'w', encoding='utf-8') as f:
20     f.write(html)
21

```

## 3.2 quote()

### ■ 使用方法

```

1 http://www.baidu.com/s?wd=赵丽颖
2
3 # 对单独的字符串进行编码 - URL地址中的中文字符
4 word = '美女'
5 result = urllib.parse.quote(word)
6 result结果: '%E7%BE%8%E5%A%B3'
7

```

### ■ 练习

```

1 """
2 问题：在百度中输入要搜索的内容，把响应内容保存到本地文件
3 编码方法使用 quote()
4 """
5 import requests
6 from urllib import parse
7
8 # 1. 拼接URL地址
9 word = input('请输入搜索关键字:')
10 params = parse.quote(word)
11 url = 'http://www.baidu.com/s?wd={}'.format(params)
12
13 # 2. 发请求获取响应内容
14 headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML,
15    like Gecko) Chrome/14.0.835.163 Safari/535.1'}
16 html = requests.get(url=url, headers=headers).content.decode('utf-8')
17
18 # 3. 保存到本地文件
19 filename = '{}.html'.format(word)
20 with open(filename, 'w', encoding='utf-8') as f:
21     f.write(html)
22

```

## 3.3 unquote()

```
1 # 将编码后的字符串转为普通的Unicode字符串
2 from urllib import parse
3
4 params = '%E7%BE%8E%E5%A5%B3'
5 result = parse.unquote(params)
6
7 result结果：美女
8
```

## 4. 百度贴吧爬虫

### 4.1 需求

```
1 1、输入贴吧名称：赵丽颖吧
2 2、输入起始页：1
3 3、输入终止页：2
4 4、保存到本地文件：赵丽颖吧_第1页.html、赵丽颖吧_第2页.html
5
```

### 4.2 实现步骤

```
1 【1】查看所抓数据在响应内容中是否存在
2 右键 - 查看网页源码 - 搜索关键字
3
4 【2】查找并分析URL地址规律
5 第1页：http://tieba.baidu.com/f?kw=???&pn=0
6 第2页：http://tieba.baidu.com/f?kw=???&pn=50
7 第n页：pn=(n-1)*50
8
9 【3】发请求获取响应内容
10
11 【4】保存到本地文件
12
```

### 4.3 代码实现

```
1 """
2     1、输入贴吧名称：赵丽颖吧
3     2、输入起始页：1
4     3、输入终止页：2
5     4、保存到本地文件：赵丽颖吧_第1页.html、赵丽颖吧_第2页.html
6 """
7 import requests
```

```

8 from urllib import parse
9 import time
10 import random
11
12 class TiebaSpider:
13     def __init__(self):
14         """定义常用变量"""
15         self.url = 'http://tieba.baidu.com/f?kw={}&pn={}'
16         self.headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1
(KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}
17
18     def get_html(self, url):
19         """请求功能函数"""
20         html = requests.get(url=url, headers=self.headers).content.decode('utf-8')
21
22         return html
23
24     def parse_html(self):
25         """解析功能函数"""
26         pass
27
28     def save_html(self, filename, html):
29         """数据处理函数"""
30         with open(filename, 'w') as f:
31             f.write(html)
32
33     def run(self):
34         """程序入口函数"""
35         name = input('请输入贴吧名:')
36         start = int(input('请输入起始页:'))
37         end = int(input('请输入终止页:'))
38         # 编码
39         params = parse.quote(name)
40         # 拼接多页的URL地址
41         for page in range(start, end + 1):
42             pn = (page - 1) * 50
43             page_url = self.url.format(params, pn)
44             # 请求 + 解析 + 数据处理
45             html = self.get_html(url=page_url)
46             filename = '{}_第{}页.html'.format(name, page)
47             self.save_html(filename, html)
48             # 终端提示
49             print('第%d页抓取完成' % page)
50             # 控制数据抓取的频率
51             time.sleep(random.randint(1, 2))
52
53 if __name__ == '__main__':
54     spider = TiebaSpider()
55     spider.run()
56

```

## 5. 正则解析模块re

## 5.1 使用流程

```
1 r_list=re.findall('正则表达式',html,re.S)
2
```

## 5.2 元字符

元字符	含义
.	任意一个字符（不包括\n）
\d	一个数字
\s	空白字符
\S	非空白字符
[]	包含[]内容
*	出现0次或多次
+	出现1次或多次

### ■ 思考 - 匹配任意一个字符的正则表达式？

```
1 r_list = re.findall('.', html, re.S)
2
```

## 5.3 贪婪与非贪婪

### ■ 贪婪匹配(默认)

```
1 1、在整个表达式匹配成功的前提下,尽可能多的匹配 * + ?
2 2、表示方式: .* .+ .?
3
```

### ■ 非贪婪匹配

```
1 1、在整个表达式匹配成功的前提下,尽可能少的匹配 * + ?
2 2、表示方式: .*? .+? .??
3
```

### ■ 代码示例

```
1 import re
```



```

2
3 html = '''
4 <div><p>九霄龙吟惊天变</p></div>
5 <div><p>风云际会潜水游</p></div>
6 '''
7 # 贪婪匹配
8 p = re.compile('<div><p>.*</p></div>', re.S)
9 r_list = p.findall(html)
10 print(r_list)
11
12 # 非贪婪匹配
13 p = re.compile('<div><p>.*?</p></div>', re.S)
14 r_list = p.findall(html)
15 print(r_list)
16

```

## 5.4 正则分组

### ■ 作用

```

1 在完整的模式中定义子模式，将每个圆括号中子模式匹配出来的结果提取出来
2

```

### ■ 示例代码

```

1 import re
2
3 s = 'A B C D'
4 p1 = re.compile('\w+\s+\w+')
5 print(p1.findall(s))
6 # 分析结果是什么? ? ?
7 # 结果: ['A B', 'C D']
8
9 p2 = re.compile('(\w+)\s+(\w+)')
10 print(p2.findall(s))
11 # 第一步: ['A B', 'C D']
12 # 第二步: ['A', 'C']
13
14 p3 = re.compile('(\w+)\s+(\w+)')
15 print(p3.findall(s))
16 # 第一步: ['A B', 'C D']
17 # 第二步: [('A', 'B'), ('C', 'D')]
18

```

### ■ 分组总结

```

1 1、在网页中,想要什么内容,就加()
2 2、先按整体正则匹配,然后再提取分组()中的内容
3   如果有2个及以上分组(),则结果中以元组形式显示 [(,),(),()]
4 3、最终结果有3种情况
5   情况1: []
6   情况2: ['', '', ''] -- 正则中1个分组时
7   情况3: [(), (), ()] -- 正则中多个分组时
8

```

## ■ 课堂练习

```

1  # 从如下html代码结构中完成如下内容信息的提取:
2  问题1 :
3      [('Tiger', ' Two...'), ('Rabbit', 'Small..')]
4  问题2 :
5      动物名称 : Tiger
6      动物描述 : Two tigers two tigers run fast
7      *****
8      动物名称 : Rabbit
9      动物描述 : Small white rabbit white and white
10

```

## ■ 页面结构如下

```

1  <div class="animal">
2      <p class="name">
3          <a title="Tiger"></a>
4      </p>
5      <p class="content">
6          Two tigers two tigers run fast
7      </p>
8  </div>
9
10 <div class="animal">
11     <p class="name">
12         <a title="Rabbit"></a>
13     </p>
14
15     <p class="content">
16         Small white rabbit white and white
17     </p>
18 </div>
19

```

## ■ 练习答案

```

1  import re
2
3  html = '''<div class="animal">
4      <p class="name">
5          <a title="Tiger"></a>
6      </p>
7
8      <p class="content">

```

```

9         Two tigers two tigers run fast
10     </p>
11 </div>
12
13 <div class="animal">
14     <p class="name">
15         <a title="Rabbit"></a>
16     </p>
17
18     <p class="content">
19         Small white rabbit white and white
20     </p>
21 </div>'''
22
23 p = re.compile('<div class="animal">.*?title="(.*?)".*?content">(.*?)</p>.*?</div>', re.S)
24 r_list = p.findall(html)
25
26 for rt in r_list:
27     print('动物名称:', rt[0].strip())
28     print('动物描述:', rt[1].strip())
29     print('*' * 50)
30

```

## 6. 笔趣阁小说爬虫

### 6.1 项目需求

- 1 【1】 官网地址: <https://www.biqukan.cc/list/>
- 2 选择一个类别, 比如: '玄幻小说'
- 3
- 4 【2】 爬取目标
- 5 '玄幻小说' 类别下前20页的
- 6 2.1》小说名称
- 7 2.2》小说链接
- 8 2.3》小说作者
- 9 2.4》小说描述
- 10

### 6.2 思路流程

- 1 【1】 查看网页源码, 确认数据来源
- 2 响应内容中存在所需抓取数据
- 3
- 4 【2】 翻页寻找URL地址规律
- 5 第1页: <https://www.biqukan.cc/fenlei1/1.html>
- 6 第2页: <https://www.biqukan.cc/fenlei1/2.html>
- 7 第n页: <https://www.biqukan.cc/fenlei1/n.html>

```
8
9 【3】编写正则表达式
10 '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
11
12 【4】开干吧兄弟
13
14 【5】排错思路
15 5.1》print(novel_list) 确认是否为空列表
16 5.2》空列表: print(html) 确认响应内容是否正确
17 5.3》响应内容正确,检查正则表达式!!!
18
```

## 6.3 代码实现

```
1 """
2 目标:
3 笔趣阁玄幻小说数据抓取
4 思路:
5 1. 确认数据来源 - 右键 查看网页源代码,搜索关键字
6 2. 确认静态,观察URL地址规律
7 3. 写正则表达式
8 4. 写代码
9 """
10
11 import re
12 import requests
13 import time
14 import random
15
16 class NovelSpider:
17     def __init__(self):
18         self.url = 'https://www.biqukan.cc/fenlei1/{ }.html'
19         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
20
21     def get_html(self, url):
22         html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')
23
24         self.refunc(html)
25
26     def refunc(self, html):
27         """正则解析函数"""
28         regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
29         novel_info_list = re.findall(regex, html, re.S)
30         for one_novel_info_tuple in novel_info_list:
31             item = {}
32             item['title'] = one_novel_info_tuple[1].strip()
33             item['href'] = one_novel_info_tuple[0].strip()
34             item['author'] = one_novel_info_tuple[2].strip()
35             item['comment'] = one_novel_info_tuple[3].strip()
```

```

36         print(item)
37
38     def crawl(self):
39         for page in range(1, 6):
40             page_url = self.url.format(page)
41             self.get_html(url=page_url)
42             time.sleep(random.randint(1, 2))
43
44 if __name__ == '__main__':
45     spider = NovelSpider()
46     spider.crawl()
47

```

## 7. MySQL数据持久化

### 7.1 pymysql回顾

#### ■ MySQL建库建表

```

1 create database noveldb charset utf8;
2 use noveldb;
3 create table novel_tab(
4     title varchar(100),
5     href varchar(500),
6     author varchar(100),
7     comment varchar(500)
8 )charset=utf8;
9

```

#### ■ pymysql示例

```

1 import pymysql
2
3 db = pymysql.connect('localhost','root','123456','noveldb',charset='utf8')
4 cursor = db.cursor()
5
6 ins = 'insert into novel_tab values(%s,%s,%s,%s)'
7 novel_li = ['花千骨', 'http://zly.com', '赵丽颖', '小骨的传奇一生']
8 cursor.execute(ins,novel_li)
9
10 db.commit()
11 cursor.close()
12 db.close()
13

```

### 7.2 笔趣阁数据持久化

```

1  """
2  1. 在 __init__() 中连接数据库并创建游标对象
3  2. 在数据处理函数中将所抓取的数据处理成列表，使用execute()方法写入数据库
4  3. 数据抓取完成后关闭游标及断开数据库连接
5  """
6  import re
7  import requests
8  import time
9  import random
10 import pymysql
11
12 class NovelSpider:
13     def __init__(self):
14         self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
15         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
16         # 连接数据库
17         self.db = pymysql.connect('localhost', 'root', '123456', 'noveldb', charset='utf8')
18         self.cur = self.db.cursor()
19
20     def get_html(self, url):
21         html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')
22
23         self.refunc(html)
24
25     def refunc(self, html):
26         """正则解析函数"""
27         regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
28         novel_info_list = re.findall(regex, html, re.S)
29         for one_novel_info in novel_info_list:
30             # 调用数据处理函数
31             self.save_to_mysql(one_novel_info)
32
33     def save_to_mysql(self, one_novel_info):
34         """将数据存入MySQL数据库"""
35         one_novel_li = [
36             one_novel_info[1].strip(),
37             one_novel_info[0].strip(),
38             one_novel_info[2].strip(),
39             one_novel_info[3].strip(),
40         ]
41         ins = 'insert into novel_tab values(%s,%s,%s,%s)'
42         self.cur.execute(ins, one_novel_li)
43         self.db.commit()
44         # 终端打印测试
45         print(one_novel_li)
46
47     def crawl(self):
48         for page in range(1, 6):
49             page_url = self.url.format(page)
50             self.get_html(url=page_url)
51             time.sleep(random.randint(1, 2))
52
53         # 所有数据抓取完成后断开数据库连接
54         self.cur.close()
55         self.db.close()

```

```
56
57 if __name__ == '__main__':
58     spider = NovelSpider()
59     spider.crawl()
60
```

## 8. 今日作业

- 1 【1】把百度贴吧案例重写一遍,不要参照课上代码
- 2 【2】笔趣阁案例重写一遍,不要参照课上代码
- 3 【3】复习任务
- 4     pymysql、MySQL基本命令
- 5     MySQL : 建库建表普通查询、插入、删除等
- 6     Redis : python和redis交互,集合基本操作
- 7 【4】猫眼电影top100数据抓取
- 8     <https://maoyan.com/board/4>
- 9     共10页,抓取 电影名称、主演、上映时间
- 10     数据存入MySQL数据库

# SPIDER-DAY02

## 1. CSV数据持久化

### 1.1 CSV持久化概述

- 1 【1】作用
- 2     将爬取的数据存放到本地的csv文件中
- 3
- 4 【2】使用流程
- 5     2.1> 打开csv文件
- 6     2.2> 初始化写入对象
- 7     2.3> 写入数据(参数为列表)
- 8
- 9 【3】示例代码
- 10     import csv
- 11     with open('sky.csv','w') as f:
- 12         writer = csv.writer(f)
- 13         writer.writerow([])

## 1.2 CSV示例代码

```
1 import csv
2 with open('test.csv','w') as f:
3     writer = csv.writer(f)
4     writer.writerow(['超哥哥','25'])
```

## 1.3 笔趣阁CSV持久化

```
1 """
2 目标:
3     笔趣阁玄幻小说数据持久化到CSV
4 思路:
5     1. 在 __init__() 中打开csv文件, 因为csv文件只需要打开和关闭1次即可
6     2. 在数据处理函数中将所抓取的数据处理成列表, 使用writerow()方法写入
7     3. 数据抓取完成后关闭文件
8 """
9 import re
10 import requests
11 import time
12 import random
13 import csv
14
15 class NovelSpider:
16     def __init__(self):
17         self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
18         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
19         # 定义csv相关变量
20         self.f = open('novel.csv', 'w')
21         self.writer = csv.writer(self.f)
22
23     def get_html(self, url):
24         html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')
25
26         self.refunc(html)
27
28     def refunc(self, html):
29         """正则解析函数"""
30         regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?<(.*?)</p>'
31         novel_info_list = re.findall(regex, html, re.S)
32         for one_novel_info_tuple in novel_info_list:
33             item = {}
34             item['title'] = one_novel_info_tuple[1].strip()
35             item['href'] = one_novel_info_tuple[0].strip()
36             item['author'] = one_novel_info_tuple[2].strip()
37             item['comment'] = one_novel_info_tuple[3].strip()
38             print(item)
39             # 将数据存入csv文件
40             item_li = [
```



```

41         item['title'],
42         item['href'],
43         item['author'],
44         item['comment'],
45     ]
46     self.writer.writerow(item_li)
47
48     def crawl(self):
49         for page in range(1, 6):
50             page_url = self.url.format(page)
51             self.get_html(url=page_url)
52             time.sleep(random.randint(1, 2))
53
54             # 数据抓取完成后关闭文件
55             self.f.close()
56
57 if __name__ == '__main__':
58     spider = NovelSpider()
59     spider.crawl()

```

## 2. MongoDB数据持久化

### 2.1 MongoDB介绍

- 1 【1】 MongoDB为非关系型数据库,基于key-value方式存储
- 2 【2】 MongoDB基于磁盘存储,而Redis基于内存
- 3 【3】 MongoDB数据类型单一,就是JSON文档
- 4 MySQL数据类型:数值类型、字符类型、枚举类型、日期时间类型
- 5 Redis数据类型:字符串、列表、哈希、集合、有序集合
- 6 MongoDB数据类型: JSON文档
- 7 【4】 和MySQL对比
- 8 MySQL: 库 - 表 - 表记录
- 9 MongoDB: 库 - 集合 - 文档
- 10 【5】 特性
- 11 MongoDB无需提前建库建集合,直接使用即可,会自动创建

### 2.2 MongoDB常用命令

- 1 【1】 进入命令行: mongo
- 2 【2】 查看所有库: show dbs
- 3 【3】 切换库: use 库名
- 4 【4】 查看库中集合: show collections | show tables
- 5 【5】 查看集合文档: db.集合名.find().pretty()
- 6 【6】 统计文档个数: db.集合名.count()
- 7 【7】 删除集合: db.集合名.drop()
- 8 【8】 删除库: db.dropDatabase()

## 2.3 与Python交互

### ■ pymongo模块

```
1  【1】 模块名: pymongo
2      sudo pip3 install pymongo
3  【2】 使用流程
4      2.1》创建数据库连接对象
5      2.2》创建库对象(库可以不存在)
6      2.3》创建集合对象(集合可以不存在)
7      2.4》在集合中插入文档
```

### ■ 示例代码

```
1  """
2  库: noveldb
3  集合: novelset
4  文档: {'title': '花千骨', 'actor': '美丽的赵丽颖'}
5  """
6  import pymongo
7
8  # 创建3个对象: 连接对象 库对象 集合对象
9  conn = pymongo.MongoClient(host='localhost', port=27017)
10 db = conn['noveldb']
11 myset = db['novelset']
12 # 插入文档
13 myset.insert_one({'title': '花千骨', 'actor': '美丽的赵丽颖'})
```

### ■ 笔趣阁数据持久化

```
1  """
2  目标:
3      笔趣阁玄幻小说数据持久化MongoDB
4  思路:
5      1. __init__()中定义MongoDB相关变量
6      2. 将抓取的数据处理成字典, 利用insert_one()方法存入数据库
7  """
8  import re
9  import requests
10 import time
11 import random
12 import pymongo
13
14 class NovelSpider:
15     def __init__(self):
16         self.url = 'https://www.biquan.cc/fenlei1/{}.html'
17         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
18         # 定义MongoDB相关变量
19         self.conn = pymongo.MongoClient('localhost', 27017)
20         self.db = self.conn['noveldb']
21         self.myset = self.db['novelset']
22
23     def get_html(self, url):
```

```

24         html = requests.get(url=url, headers=self.headers).content.decode('gbk',
'ignore')
25
26         self.refunc(html)
27
28     def refunc(self, html):
29         """正则解析函数"""
30         regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>
(.*?)</small>.*?>(.*?)</p>'
31         novel_info_list = re.findall(regex, html, re.S)
32         for one_novel_info_tuple in novel_info_list:
33             item = {}
34             item['title'] = one_novel_info_tuple[1].strip()
35             item['href'] = one_novel_info_tuple[0].strip()
36             item['author'] = one_novel_info_tuple[2].strip()
37             item['comment'] = one_novel_info_tuple[3].strip()
38             print(item)
39             # 将数据存入mongodb数据库
40             self.myset.insert_one(item)
41
42     def crawl(self):
43         for page in range(1, 6):
44             page_url = self.url.format(page)
45             self.get_html(url=page_url)
46             time.sleep(random.randint(1, 2))
47
48 if __name__ == '__main__':
49     spider = NovelSpider()
50     spider.crawl()

```

## 3. 笔趣阁多级页面爬虫

### 3.1 项目需求

- 1 **【1】爬取地址**
- 2 `https://www.biqukan.cc/fenlei1/1.html`
- 3
- 4 **【2】爬取目标**
- 5 `'玄幻小说'`分类下所有小说的：小说名称、链接、作者、描述、最新更新章节、最新更新章节链接
- 6
- 7 **【3】爬取分析**
- 8 `*****一级页面需抓取*****`
- 9
  - 1 1、小说名称
  - 10 2、小说详情页链接
  - 11 3、小说作者
  - 12 4、小说描述
  - 13
- 14 `*****二级页面需抓取*****`

15	1、最新更新的章节名称
16	2、最新更新的章节链接

## 3.2 项目实施流程

```

1  【1】确认数据来源 - 响应内容中存在所抓取数据!!!
2  【2】找URL地址规律
3      第1页: https://www.biquan.cc/fenlei1/1.html
4      第2页: https://www.biquan.cc/fenlei1/2.html
5      第n页: https://www.biquan.cc/fenlei1/n.html
6  【3】写正则表达式
7      3.1》一级页面正则表达式
8          '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>
          (.*?)</p>'
9      3.2》二级页面正则表达式
10         '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
11  【4】代码实现

```

## 3.3 代码实现

```

1  """
2  目标:
3      笔趣阁玄幻小说数据抓取
4  思路:
5      1. 确认数据来源 - 右键 查看网页源代码,搜索关键字
6      2. 确认静态,观察URL地址规律
7      3. 写正则表达式
8      4. 写代码
9  """
10 import re
11 import requests
12 import time
13 import random
14
15 class NovelSpider:
16     def __init__(self):
17         self.url = 'https://www.biquan.cc/fenlei1/{}.html'
18         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
19
20     def get_html(self, url):
21         """功能函数1: 获取html"""
22         html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')
23
24         return html
25
26     def refunc(self, regex, html):
27         """功能函数2: 正则解析"""

```

```

28         r_list = re.findall(regex, html, re.S)
29
30         return r_list
31
32     def crawl(self, first_url):
33         """爬虫逻辑函数"""
34         # 一级页面开始：小说名称、链接、作者、描述
35         first_html = self.get_html(url=first_url)
36         first_regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>
(.*?)</small>.*?>(.*?)</p>'
37         novel_info_list = self.refunc(regex=first_regex, html=first_html)
38         for one_novel_info_tuple in novel_info_list:
39             item = {}
40             item['title'] = one_novel_info_tuple[1].strip()
41             item['href'] = one_novel_info_tuple[0].strip()
42             item['author'] = one_novel_info_tuple[2].strip()
43             item['comment'] = one_novel_info_tuple[3].strip()
44             # 获取小说的最新章节名称、链接
45             self.get_novel_data(item)
46
47     def get_novel_data(self, item):
48         """获取小说最新章节名称、链接"""
49         second_html = self.get_html(url=item['href'])
50         second_regex = '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
51         chapter_list = self.refunc(regex=second_regex, html=second_html)
52         for one_chapter_tuple in chapter_list:
53             item['chapter_name'] = one_chapter_tuple[1].strip()
54             item['chapter_href'] = one_chapter_tuple[0].strip()
55             print(item)
56
57     def run(self):
58         for page in range(1, 2):
59             page_url = self.url.format(page)
60             self.crawl(first_url=page_url)
61             time.sleep(random.randint(1, 2))
62
63 if __name__ == '__main__':
64     spider = NovelSpider()
65     spider.run()

```

### 3.4 练习

```
1  【1】将小说信息存入'MySQL数据库'
2      create database noveldb2 charset utf8;
3      use noveldb2;
4      create table novel_tab(
5          novel_title varchar(200),
6          novel_href varchar(300),
7          novel_author varchar(200),
8          novel_comment varchar(500),
9          chapter_name varchar(200),
10         chapter_href varchar(300)
11     )charset=utf8;
12  【2】将小说信息存入'MongoDB数据库'
13  【3】将小说信息存入'novel_info.csv文件'
```

## 4. 增量爬虫

### 4.1 增量爬虫概述

```
1  【1】引言
2      当我们在浏览相关网页的时候会发现，某些网站定时会在原有网页数据的基础上更新一批数据，
3      例：1. 某电影网站会实时更新一批最近热门的电影
4          2. 小说网站会根据作者创作的进度实时更新最新的章节数据等等
5      当我们在爬虫的过程中遇到时，我们是否需要只爬取网站中最近更新的数据，而不每次都做全量爬虫呢？
6
7  【2】概念
8      通过爬虫程序监测某网站数据更新的情况，以便可以爬取到该网站更新出的新数据
```

### 4.2 增量爬虫实现

```
1  【1】原理
2      1.1 在发送请求之前判断这个URL是不是之前爬取过
3          适用场景：'不断有新页面出现的网站，比如说小说的新章节，每天的最新新闻等'
4      1.2 在解析内容后判断这部分内容是不是之前爬取过
5          适用场景：'页面内容会更新的网站'
6
7  【2】实现
8      2.1 将爬取过程中产生的url进行存储，存储在redis的set中。当下次进行数据爬取时，首先对即将要发起的请求对应的url在存储的url的set中做判断，如果存在则不进行请求，否则才进行请求。
9      2.2 对爬取到的网页内容进行唯一标识的制定，然后将该唯一标识存储至redis的set中。当下次爬取到网页数据的时候，在进行持久化存储之前，首先可以先判断该数据的唯一标识在redis的set中是否存在，在决定是否进行持久化存储
```

### 4.3 笔趣阁增量爬虫

```

1  """
2  增量爬虫实现步骤:
3      1. 在__init__()中连接redis数据库
4      2. md5加密的功能函数
5      3. 抓取具体数据之前通过sadd的返回值做判断
6          返回值为1: 为新更新, 说明之前没有抓取过
7          返回值为0: 无需抓取, 之前已经抓取过
8  """
9  import re
10 import requests
11 import time
12 import random
13 import redis
14 from hashlib import md5
15
16 class NovelSpider:
17     def __init__(self):
18         self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
19         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
20         # 连接redis
21         self.r = redis.Redis(host='localhost', port=6379, db=0)
22
23     def get_html(self, url):
24         """功能函数1: 获取html"""
25         html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')
26
27         return html
28
29     def refunc(self, regex, html):
30         """功能函数2: 正则解析"""
31         r_list = re.findall(regex, html, re.S)
32
33         return r_list
34
35     def md5_href(self, href):
36         """功能函数3: 生成指纹"""
37         m = md5()
38         m.update(href.encode())
39
40         return m.hexdigest()
41
42     def crawl(self, first_url):
43         """爬虫逻辑函数"""
44         # 一级页面开始: 小说名称、链接、作者、描述
45         first_html = self.get_html(url=first_url)
46         first_regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>
(.*?)</small>.*?>(.*?)</p>'
47         novel_info_list = self.refunc(regex=first_regex, html=first_html)
48         for one_novel_info_tuple in novel_info_list:
49             item = {}
50             item['title'] = one_novel_info_tuple[1].strip()
51             item['href'] = one_novel_info_tuple[0].strip()
52             item['author'] = one_novel_info_tuple[2].strip()
53             item['comment'] = one_novel_info_tuple[3].strip()
54             # 获取小说的最新章节名称、链接
55             self.get_novel_data(item)

```

```

56
57 def get_novel_data(self, item):
58     """获取小说最新章节名称、链接"""
59     second_html = self.get_html(url=item['href'])
60     second_regex = '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
61     chapter_list = self.refunc(regex=second_regex, html=second_html)
62     for one_chapter_tuple in chapter_list:
63         item['chapter_name'] = one_chapter_tuple[1].strip()
64         item['chapter_href'] = one_chapter_tuple[0].strip()
65         print(item)
66         finger = self.md5_href(item['chapter_href'])
67         if self.r.sadd('novel:spiders', finger) == 1:
68             print('章节有更新, 开始抓取... ..')
69         else:
70             print('章节未更新, 跳过此章节')
71
72     def run(self):
73         for page in range(1, 2):
74             page_url = self.url.format(page)
75             self.crawl(first_url=page_url)
76             time.sleep(random.randint(1, 2))
77
78 if __name__ == '__main__':
79     spider = NovelSpider()
80     spider.run()

```

## 5. Chrome浏览器插件

- 1 **【1】在线安装**
- 2 1.1> 下载插件 - google访问助手
- 3 1.2> 安装插件 - google访问助手: Chrome浏览器-设置-更多工具-扩展程序-开发者模式-拖拽(解压后的插件)
- 4 1.3> 在线安装其他插件 - 打开google访问助手 - google应用商店 - 搜索插件 - 添加即可
- 5
- 6 **【2】爬虫常用插件**
- 7 2.1》google-access-helper : 谷歌访问助手,可访问 谷歌应用商店
- 8 2.2》Xpath Helper: 轻松获取HTML元素的XPath路径
- 9 打开/关闭: Ctrl + Shift + x
- 10 2.3》JsonView: 格式化输出json格式数据
- 11 2.4》Proxy SwitchyOmega: Chrome浏览器中的代理管理扩展程序

## 6. xpath解析

### 6.1 xpath定义

- 1 XPath即为XML路径语言, 它是一种用来确定XML文档中某部分位置的语言, 同样适用于HTML文档的检索



## 6.2 匹配演示

```
1  """
2  匹配猫眼电影top100: https://maoyan.com/board/4
3  """
4  【1】查找所有的dd节点
5      //dd
6  【2】获取所有电影的名称的a节点：所有class属性值为name的a节点
7      //p[@class="name"]/a
8
9  【3】获取d1节点下第2个dd节点的电影节点
10     //d1[@class="board-wrapper"]/dd[2]
11 【4】获取所有电影详情页链接：获取每个电影的a节点的href的属性值
12     //p[@class="name"]/a/@href
13
14 【注意】
15     1> 只要涉及到条件,加 [] :
16         //d1[@class="xxx"]    //d1/dd[2]
17
18     2> 只要获取属性值,加 @ :
19         //d1[@class="xxx"]    //p/a/@href
```

## 6.3 xpath语法

### ■ 选取节点

```
1  【1】 // : 从所有节点中查找 (包括子节点和后代节点)
2  【2】 @ : 获取属性值
3      2.1> 使用场景1 (属性值作为条件)
4          //div[@class="movie-item-info"]
5      2.2> 使用场景2 (直接获取属性值)
6          //div[@class="movie-item-info"]/a/img/@src
7
8  【3】 练习 - 猫眼电影top100
9      3.1> 匹配电影名称
10         //div[@class="movie-item-info"]/p[1]/a/@title
11      3.2> 匹配电影主演
12         //div[@class="movie-item-info"]/p[2]/text()
13      3.3> 匹配上映时间
14         //div[@class="movie-item-info"]/p[3]/text()
15      3.4> 匹配电影链接
16         //div[@class="movie-item-info"]/p[1]/a/@href
```

### ■ 匹配多路径 (或)

```
1  xpath表达式1 | xpath表达式2 | xpath表达式3
```

### ■ 常用函数

```
1 【1】 text() : 获取节点的文本内容
2   xpath表达式末尾不加 /text() :则得到的结果为节点对象
3   xpath表达式末尾加 /text() 或者 /@href : 则得到结果为字符串
4
5 【2】 contains() : 匹配属性值中包含某些字符串节点
6   匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
7   //div[contains(@class,"movie-item")]
```

## ■ 终极总结

```
1 【1】 字符串: xpath表达式的末尾为: /text() 、 /@href 得到的列表中为 '字符串'
2
3 【2】 节点对象: 其他剩余所有情况得到的列表中均为 '节点对象'
4   [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
5   [<element div at xxxa>,<element div at xxxb>]
6   [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]
```

## ■ 课堂练习

```
1 【1】 匹配汽车之家-二手车,所有汽车的链接 :
2   //li[@class="cards-li list-photo-li"]/a[1]/@href
3   //a[@class="carinfo"]/@href
4 【2】 匹配汽车之家-汽车详情页中,汽车的
5   2.1)名称: //div[@class="car-box"]/h3/text()
6   2.2)里程: //ul/li[1]/h4/text()
7   2.3)时间: //ul/li[2]/h4/text()
8   2.4)挡位+排量: //ul/li[3]/h4/text()
9   2.5)所在地: //ul/li[4]/h4/text()
10  2.6)价格: //div[@class="brand-price-item"]/span[@class="price"]/text()
```

# 7. 今日作业

```
1 【1】 正则抓取豆瓣图书top250书籍信息
2   地址: https://book.douban.com/top250?icn=index-book250-all
3   抓取目标: 书籍名称、书籍信息、书籍评分、书籍评论人数、书籍描述
4
5 【2】 使用xpath helper在页面中匹配豆瓣图书top250的信息, 写出对应的xpath表达式
6   书籍名称:
7   书籍信息:
8   书籍评分:
9   评论人数:
10  书籍描述:
```