王伟超

wangweichao@tedu.cn

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
```

2. 爬虫请求模块

2.1 requests模块

- **安装**

```
【1】Linux
    sudo pip3 install requests

【2】Windows
    方法1>  cmd命令行 -> python -m pip install requests
    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】作用
    向目标网站发起请求,并获取响应对象

【2】参数
    2.1> url : 需要抓取的URL地址
    2.2> headers : 请求头
    2.3> timeout : 超时时间，超过时间会抛出异常
```

- **此生第一个爬虫**

```
"""
向京东官网（https://www.jd.com/）发请求,获取响应内容
"""
import requests

resp = requests.get(url='https://www.jd.com/')
# 1.text属性：获取响应内容-字符串
html = resp.text
print(html)
```

- **响应对象（res）属性**

```
【1】text        : 字符串
【2】content     : 字节流
【3】status_code : HTTP响应码
【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！！！

```python
# 请求头（headers）中的 User-Agent
# 测试案例：向测试网站http://httpbin.org/get发请求，查看请求头(User-Agent)
import requests

url = 'http://httpbin.org/get'
res = requests.get(url=url)
html = res.text
print(html)
# 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**

```python
"""
包装好请求头后,向测试网站发请求,并验证
养成好习惯，发送请求携带请求头，重构User-Agent
"""
import requests

url = 'http://httpbin.org/get'
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'}
html = requests.get(url=url,headers=headers).text
# 在html中确认User-Agent
print(html)
```

3. URL地址拼接

3.1 拼接URL地址的三种方式

```
【1】字符串相加
【2】字符串格式化（占位符 %s）
【3】format()方法
    'http://www.baidu.com/s?{}'.format(params)

【练习】
    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
    官网地址：https://www.guazi.com/langfang/

url = 'https://www.guazi.com/dachang/buy/o{}/#bread'
for i in range(1, 21):
    page_url = url.format(i)
    print(page_url)
```

3.2 练习

在百度中输入要搜索的内容，把响应内容保存到本地文件

```python
"""
在百度中输入要搜索的内容，把响应内容保存到本地文件
"""
import requests

# 1. 拼接URL地址
word = input('请输入搜索关键字:')
url = 'http://www.baidu.com/s?wd={}'.format(word)

# 2. 发请求获取响应内容
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'}
html = requests.get(url=url, headers=headers).text

# 3. 保存到本地文件
filename = '{}.html'.format(word)
with open(filename, 'w', encoding='utf-8') as f:
    f.write(html)

```

4. 百度贴吧爬虫

4.1 需求

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

```

4.2 实现步骤

```
【1】查看所抓数据在响应内容中是否存在
    右键 - 查看网页源码 - 搜索关键字

【2】查找并分析URL地址规律
    第1页：http://tieba.baidu.com/f?kw=???&pn=0
    第2页：http://tieba.baidu.com/f?kw=???&pn=50
    第n页：pn=(n-1)*50

【3】发请求获取响应内容

【4】保存到本地文件

```

4.3 代码实现

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

class TiebaSpider:
    def __init__(self):
        """定义常用变量"""
        self.url = 'http://tieba.baidu.com/f?kw={}&pn={}'
        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'}

    def get_html(self, url):
        """请求功能函数"""
        html = requests.get(url=url, headers=self.headers).text

        return html

    def parse_html(self):
        """解析功能函数"""
        pass

    def save_html(self, filename, html):
        """数据处理函数"""
        with open(filename, 'w') as f:
            f.write(html)

    def run(self):
        """程序入口函数"""
        name = input('请输入贴吧名:')
        start = int(input('请输入起始页:'))
        end = int(input('请输入终止页:'))
        # 拼接多页的URL地址
        for page in range(start, end + 1):
            pn = (page - 1) * 50
            page_url = self.url.format(name, pn)
            # 请求 + 解析 + 数据处理
            html = self.get_html(url=page_url)
            filename = '{}_第{}页.html'.format(name, page)
            self.save_html(filename, html)
            # 终端提示
            print('第%d页抓取完成' % page)
            # 控制数据抓取的频率
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = TiebaSpider()
    spider.run()
```

```
54
```

5. 正则解析模块re

5.1 使用流程

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

```

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?**

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

```

5.3 贪婪与非贪婪

- **贪婪匹配(默认)**

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

```

- **非贪婪匹配**

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

```

- **代码示例**

```python
import re

html = '''
<div><p>九霄龙吟惊天变</p></div>
<div><p>风云际会潜水游</p></div>
'''
# 贪婪匹配
p = re.compile('<div><p>.*</p></div>',re.S)
r_list = p.findall(html)
print(r_list)

# 非贪婪匹配
p = re.compile('<div><p>.*?</p></div>',re.S)
r_list = p.findall(html)
print(r_list)

```

5.4 正则分组

- **作用**

```
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

```

- **示例代码**

```python
import re

s = 'A B C D'
r1 = re.findall('\w+\s+\w+', s, re.S)
print(r1)
# 分析结果是什么？？？
# 结果：['A B', 'C D']

r2 = re.findall('(\w+)\s+\w+', s, re.S)
print(r2)
# 第一步：['A B', 'C D']
# 第二步：['A', 'C']

r3 = re.findall('(\w+)\s+(\w+)', s, re.S)
print(r3)
# 第一步：['A B', 'C D']
# 第二步：[('A', 'B'), ('C', 'D')]

```

- **分组总结**

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

```

- **课堂练习**

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

```

- **页面结构如下**

```
<div class="animal">
    <p class="name">
            <a title="Tiger"></a>
    </p>
    <p class="content">
            Two tigers two tigers run fast
    </p>
</div>

<div class="animal">
    <p class="name">
            <a title="Rabbit"></a>
    </p>

    <p class="content">
            Small white rabbit white and white
    </p>
</div>

```

- **练习答案**

```
import re

html = '''<div class="animal">
    <p class="name">
        <a title="Tiger"></a>
    </p>

```

```
    <p class="content">
        Two tigers two tigers run fast
    </p>
</div>

<div class="animal">
    <p class="name">
        <a title="Rabbit"></a>
    </p>

    <p class="content">
        Small white rabbit white and white
    </p>
</div>'''

p = re.compile('<div class="animal">.*?title="(.*?)".*?content">(.*?)</p>.*?
</div>',re.S)
r_list = p.findall(html)

for rt in r_list:
    print('动物名称:',rt[0].strip())
    print('动物描述:',rt[1].strip())
    print('*' * 50)

```

6. 笔趣阁小说爬虫

6.1 项目需求

```
【1】官网地址：https://www.biqukan.cc/list/
   选择一个类别，比如：'玄幻小说'

【2】爬取目标
   '玄幻小说'类别下前20页的
   2.1》小说名称
   2.2》小说链接
   2.3》小说作者
   2.4》小说描述

```

6.2 思路流程

```
【1】查看网页源码，确认数据来源
    响应内容中存在所需抓取数据

【2】翻页寻找URL地址规律
    第1页：https://www.biqukan.cc/fenlei1/1.html
    第2页：https://www.biqukan.cc/fenlei1/2.html
    第n页：https://www.biqukan.cc/fenlei1/n.html

【3】编写正则表达式
    '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'

【4】开干吧兄弟
```

6.3 代码实现

```
"""
目标：
    笔趣阁玄幻小说数据抓取
思路：
    1. 确认数据来源 - 右键 查看网页源代码,搜索关键字
    2. 确认静态,观察URL地址规律
    3. 写正则表达式
    4. 写代码
"""

import re
import requests
import time
import random

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        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'}

    def get_html(self, url):
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        self.refunc(html)

    def refunc(self, html):
        """正则解析函数"""
        regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = re.findall(regex, html, re.S)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
```

```
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            print(item)

    def crawl(self):
        for page in range(1, 6):
            page_url = self.url.format(page)
            self.get_html(url=page_url)
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = NovelSpider()
    spider.crawl()

```

7. MySQL数据持久化

7.1 pymysql回顾

- **MySQL建库建表**

```
create database noveldb charset utf8;
use noveldb;
create table novel_tab(
title varchar(100),
href varchar(500),
author varchar(100),
comment varchar(500)
)charset=utf8;

```

- **pymysql示例**

```
import pymysql

db = pymysql.connect('localhost','root','123456','noveldb',charset='utf8')
cursor = db.cursor()

ins = 'insert into novel_tab values(%s,%s,%s,%s)'
novel_li = ['花千骨', 'http://zly.com', '赵丽颖', '小骨的传奇一生']
cursor.execute(ins,novel_li)

db.commit()
cursor.close()
db.close()

```

7.2 笔趣阁数据持久化

```python
"""
1. 在 __init__() 中连接数据库并创建游标对象
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
3. 数据抓取完成后关闭游标及断开数据库连接
"""
import re
import requests
import time
import random
import pymysql

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        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'}
        # 连接数据库
        self.db = pymysql.connect('localhost', 'root', '123456', 'noveldb', charset='utf8')
        self.cur = self.db.cursor()

    def get_html(self, url):
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        self.refunc(html)

    def refunc(self, html):
        """正则解析函数"""
        regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = re.findall(regex, html, re.S)
        for one_novel_info in novel_info_list:
            # 调用数据处理函数
            self.save_to_mysql(one_novel_info)

    def save_to_mysql(self, one_novel_info):
        """将数据存入MySQL数据库"""
        one_novel_li = [
            one_novel_info[1].strip(),
            one_novel_info[0].strip(),
            one_novel_info[2].strip(),
            one_novel_info[3].strip(),
        ]
        ins = 'insert into novel_tab values(%s,%s,%s,%s)'
        self.cur.execute(ins, one_novel_li)
        self.db.commit()
        # 终端打印测试
        print(one_novel_li)

    def crawl(self):
        for page in range(1, 6):
            page_url = self.url.format(page)
            self.get_html(url=page_url)
            time.sleep(random.randint(1, 2))

        # 所有数据抓取完成后断开数据库连接
        self.cur.close()
```

```
        self.db.close()

if __name__ == '__main__':
    spider = NovelSpider()
    spider.crawl()

```

8. 今日作业

```
【1】把百度贴吧案例重写一遍,不要参照课上代码
【2】笔趣阁案例重写一遍,不要参照课上代码
【3】复习任务
    pymysql、MySQL基本命令
    MySQL  ：建库建表普通查询、插入、删除等
    Redis ：  python和redis交互,集合基本操作
【4】猫眼电影top100数据抓取
    url地址: https://maoyan.com/board/4
    所抓数据: 电影名称
              电影主演
              上映时间
    数据处理: 每个电影作为字典打印输出
    # 因频率过高出现滑块验证，则在页面中手动滑动滑块通过验证
```

SPIDER-DAY02

1. CSV数据持久化

1.1 CSV持久化概述

```
【1】作用
   将爬取的数据存放到本地的csv文件中

【2】使用流程
    2.1> 打开csv文件
    2.2> 初始化写入对象
    2.3> 写入数据(参数为列表)

【3】示例代码
    import csv
    with open('sky.csv','w') as f:
        writer = csv.writer(f)
        writer.writerow([])
```

1.2 CSV示例代码

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

1.3 笔趣阁CSV持久化

```
"""
目标:
    笔趣阁玄幻小说数据持久化到CSV
思路:
    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
    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
    3. 数据抓取完成后关闭文件
"""
import re
import requests
import time
import random
import csv

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        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'}
        # 定义csv相关变量
        self.f = open('novel.csv', 'w')
        self.writer = csv.writer(self.f)

    def get_html(self, url):
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        self.refunc(html)

    def refunc(self, html):
        """正则解析函数"""
        regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = re.findall(regex, html, re.S)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            print(item)
            # 将数据存入csv文件
            item_li = [
```

```
                item['title'],
                item['href'],
                item['author'],
                item['comment'],
            ]
            self.writer.writerow(item_li)

    def crawl(self):
        for page in range(1, 6):
            page_url = self.url.format(page)
            self.get_html(url=page_url)
            time.sleep(random.randint(1, 2))

        # 数据抓取完成后关闭文件
        self.f.close()

if __name__ == '__main__':
    spider = NovelSpider()
    spider.crawl()
```

2. MongoDB数据持久化

2.1 MongoDB介绍

```
【1】MongoDB为非关系型数据库,基于key-value方式存储
【2】MongoDB基于磁盘存储,而Redis基于内存
【3】MongoDB数据类型单一,就是JSON文档
    MySQL数据类型:数值类型、字符类型、枚举类型、日期时间类型
    Redis数据类型:字符串、列表、哈希、集合、有序集合
    MongoDB数据类型: JSON文档
【4】和MySQL对比
    MySQL:   库 - 表   - 表记录
    MongoDB: 库 - 集合 - 文档
【5】特性
    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】进入命令行:  mongo
【2】查看所有库:  show dbs
【3】切换库:     use 库名
【4】查看库中集合:show collections  |  show tables
【5】查看集合文档:db.集合名.find().pretty()
【6】统计文档个数:db.集合名.count()
【7】删除集合:   db.集合名.drop()
【8】删除库:     db.dropDatabase()
```

2.3 与Python交互

- **pymongo模块**

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

- **示例代码**

```
"""
库：noveldb
集合：novelset
文档：{'title':'花千骨', 'actor':'美丽的赵丽颖'}
"""
import pymongo

# 创建3个对象：连接对象 库对象 集合对象
conn = pymongo.MongoClient(host='localhost', port=27017)
db = conn['noveldb']
myset = db['novelset']
# 插入文档
myset.insert_one({'title':'花千骨', 'actor':'美丽的赵丽颖'})
```

- **笔趣阁数据持久化**

```
"""
目标：
    笔趣阁玄幻小说数据持久化MongoDB
思路：
    1. __init__()中定义MongoDB相关变量
    2. 将抓取的数据处理成字典，利用insert_one()方法存入数据库
"""
import re
import requests
import time
import random
import pymongo

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        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'}
        # 定义MongoDB相关变量
        self.conn = pymongo.MongoClient('localhost', 27017)
        self.db = self.conn['noveldb']
        self.myset = self.db['novelset']

    def get_html(self, url):
```

```
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        self.refunc(html)

    def refunc(self, html):
        """正则解析函数"""
        regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = re.findall(regex, html, re.S)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            print(item)
            # 将数据存入mongodb数据库
            self.myset.insert_one(item)

    def crawl(self):
        for page in range(1, 6):
            page_url = self.url.format(page)
            self.get_html(url=page_url)
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = NovelSpider()
    spider.crawl()
```

3. 笔趣阁多级页面爬虫

3.1 项目需求

```
【1】爬取地址
   https://www.biqukan.cc/fenlei1/1.html

【2】爬取目标
   '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

【3】爬取分析
   *********一级页面需抓取**********
       1、小说名称
       2、小说详情页链接
       3、小说作者
       4、小说描述

   *********二级页面需抓取**********
```

```
        1、最新更新的章节名称
        2、最新更新的章节链接
```

3.2 项目实现流程

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

3.3 代码实现

```
"""
目标:
    笔趣阁玄幻小说数据抓取
思路:
    1. 确认数据来源 - 右键 查看网页源代码,搜索关键字
    2. 确认静态,观察URL地址规律
    3. 写正则表达式
    4. 写代码
"""
import re
import requests
import time
import random

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        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'}

    def get_html(self, url):
        """功能函数1: 获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        return html

    def refunc(self, regex, html):
        """功能函数2: 正则解析"""
```

```
        r_list = re.findall(regex, html, re.S)

        return r_list

    def crawl(self, first_url):
        """爬虫逻辑函数"""
        # 一级页面开始：小说名称、链接、作者、描述
        first_html = self.get_html(url=first_url)
        first_regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>
(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = self.refunc(regex=first_regex, html=first_html)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            # 获取小说的最新章节名称、链接
            self.get_novel_data(item)

    def get_novel_data(self, item):
        """获取小说最新章节名称、链接"""
        second_html = self.get_html(url=item['href'])
        second_regex = '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
        chapter_list = self.refunc(regex=second_regex, html=second_html)
        for one_chapter_tuple in chapter_list:
            item['chapter_name'] = one_chapter_tuple[1].strip()
            item['chapter_href'] = one_chapter_tuple[0].strip()
            print(item)

    def run(self):
        for page in range(1, 2):
            page_url = self.url.format(page)
            self.crawl(first_url=page_url)
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = NovelSpider()
    spider.run()
```

3.4 练习

```
【1】将小说信息存入'MySQL数据库'
    create database noveldb2 charset utf8;
    use noveldb2;
    create table novel_tab(
    novel_title varchar(200),
    novel_href varchar(300),
    novel_author varchar(200),
    novel_comment varchar(500),
    chapter_name varchar(200),
    chapter_href varchar(300)
    )charset=utf8;
【2】将小说信息存入'MongoDB数据库'
【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】引言
    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，
    例：1. 某电影网站会实时更新一批最近热门的电影
        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
    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?

【2】概念
    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】原理
    1.1》在发送请求之前判断这个URL是不是之前爬取过
        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'
    1.2》在解析内容后判断这部分内容是不是之前爬取过
        适用场景：'页面内容会更新的网站'

【2】实现
    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。
    2.2》对爬取到的网页内容进行唯一标识的制定，然后将该唯一表示存储至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

```python
"""
增量爬虫实现步骤:
    1. 在__init__()中连接redis数据库
    2. md5加密的功能函数
    3. 抓取具体数据之前通过sadd的返回值做判断
       返回值为1：为新更新，说明之前没有抓取过
       返回值为0：无需抓取，之前已经抓取过
"""
import re
import requests
import time
import random
import redis
from hashlib import md5

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        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'}
        # 连接redis
        self.r = redis.Redis(host='localhost', port=6379, db=0)

    def get_html(self, url):
        """功能函数1：获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        return html

    def refunc(self, regex, html):
        """功能函数2：正则解析"""
        r_list = re.findall(regex, html, re.S)

        return r_list

    def md5_href(self, href):
        """功能函数3：生成指纹"""
        m = md5()
        m.update(href.encode())

        return m.hexdigest()

    def crawl(self, first_url):
        """爬虫逻辑函数"""
        # 一级页面开始：小说名称、链接、作者、描述
        first_html = self.get_html(url=first_url)
        first_regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = self.refunc(regex=first_regex, html=first_html)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            # 获取小说的最新章节名称、链接
            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

```
"""
匹配猫眼电影top100: https://maoyan.com/board/4
"""
【1】查找所有的dd节点
    //dd
【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
    //p[@class="name"]/a

【3】获取dl节点下第2个dd节点的电影节点
    //dl[@class="board-wrapper"]/dd[2]
【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
    //p[@class="name"]/a/@href

【注意】
    1> 只要涉及到条件,加 [] :
        //dl[@class="xxx"]   //dl/dd[2]

    2> 只要获取属性值,加 @  :
        //dl[@class="xxx"]   //p/a/@href
```

6.3 xpath语法

- **选取节点**

```
【1】// : 从所有节点中查找(包括子节点和后代节点)
【2】@  : 获取属性值
 2.1> 使用场景1(属性值作为条件)
      //div[@class="movie-item-info"]
 2.2> 使用场景2(直接获取属性值)
      //div[@class="movie-item-info"]/a/img/@src

【3】练习 - 猫眼电影top100
 3.1> 匹配电影名称
     //div[@class="movie-item-info"]/p[1]/a/@title
 3.2> 匹配电影主演
     //div[@class="movie-item-info"]/p[2]/text()
 3.3> 匹配上映时间
     //div[@class="movie-item-info"]/p[3]/text()
 3.4> 匹配电影链接
     //div[@class="movie-item-info"]/p[1]/a/@href
```

- **匹配多路径(或)**

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

- **常用函数**

```
【1】text() : 获取节点的文本内容
    xpath表达式末尾不加 /text() :则得到的结果为节点对象
    xpath表达式末尾加 /text() 或者 /@href : 则得到结果为字符串

【2】contains() : 匹配属性值中包含某些字符串节点
    匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
     //div[contains(@class,"movie-item")]
```

- **终极总结**

```
【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】节点对象: 其他剩余所有情况得到的列表中均为'节点对象'
    [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
    [<element div at xxxa>,<element div at xxxb>]
    [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]
```

- **课堂练习**

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

7. 今日作业

```
【1】正则抓取豆瓣图书top250书籍信息
    地址: https://book.douban.com/top250?icn=index-book250-all
    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

【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
    书籍名称:
    书籍信息:
    书籍评分:
    评论人数:
    书籍描述:
```