**南京航空航天大学**

**课 程 设 计 报 告**



**课 程 名 称 计算机网络课程设计**

**学 院 计算机科学与技术 年 级 2016**

**学 生 姓 名 刘鸿搏 学 号 161620222**

**开 课 时 间 2018 至 2019 学年第 一 学期**

|  |  |
| --- | --- |
| **总 成 绩** |  |
| **指导老师** | **夏正友** |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验项目  名 称 | **实验一、多人聊天室** | 成绩 |  |

**一、实验目的**

用Java语言实现基于C/S模式的多人聊天室系统。聊天室分为服务器端和客户端部分，服务器端程序主要负责建立连接接受和发送客户端消息，多个客户端与服务器端成功连接后就可以实现正常的聊天室功能。

1. 使用Java语言完成聊天设计，深入学习使用Java语言。
2. 使用网络编程，掌握基于TCP协议的Socket编程，了解Socket编程的协议约定，掌握简单应用协议的开发。
3. 使用C/S架构，对网络编程有一定的了解。
4. **实验内容与步骤**
5. **需求分析**

使用C/S架构。

服务端创建自己的socket，等待客户端链接，如有连接请求就接受并保存客户端相关信息，如有接收到客户端消息就将消息发送给所有连接到服务器的客户端。

客户端创建自己的socket，填写用户名等信息，创建两个线程，分别用于监听服务器端发来的数据，以及向服务器端发送数据，如果两个线程可以正常建立并启动则说明客户端成功连接到聊天室。

1. **程序框架**

**服务器端：**创建socket，然后进入以下循环：接受客户端链接请求，打印欢迎信息，调用writeToClient过程向客户端发送消息，调用readFromClient从客户端读取消息。

**客户端：**创建socket，填写用户名，创建readThread线程用于接受服务器消息，创建writeThread线程用于发消息给服务器，启动两个线程。

1. **编写代码**

**服务器端：**

package main;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* 基于socket编程的多人聊天室服务器端

\* @author 161620222刘鸿搏

\*

\*/

public class ManyPeopleServer {

private static List<Socket> socketList = new ArrayList<>();

private static final int port = 6666;

private static int count = 0;

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(port);

System.out.println("服务器启动成功。。。。");

while(true){

//服务器端循环接受客户端发来的连接请求，然后接受连接

Socket socket = serverSocket.accept();

//接受连接后加入list

socketList.add(socket);

String msg = "欢迎用户"+(++count)+"号"+socket.getInetAddress()

.getLocalHost().toString().split("/")[1]+"加入聊天室";

writeToClient(msg);

readFromClient();

}

} catch (Exception e) {

System.out.println("【系统提醒】：服务器好像出了点问题");

//e.printStackTrace();

}

}

public static synchronized void writeToClient(String msg)throws Exception{

for(Socket socket:socketList){

// DataOutputStream dataOs = new DataOutputStream(socket.getOutputStream());

PrintWriter writer = new PrintWriter(socket.getOutputStream(),true);

writer.println(msg);

//writer.close();

}

}

public static void readFromClient()throws Exception{

for(Socket socket:socketList){

Thread writeThread = new Thread(){

public void run(){

//开辟一个写的进程来广播数据

try {

// DataInputStream dataIs = new DataInputStream(socket.getInputStream());

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String msg = reader.readLine();

//读取到了就来广播

writeToClient(msg);

} catch (Exception e) {

// TODO Auto-generated catch block

System.out.println("【系统提醒】：一个用户退出了聊天室");

//e.printStackTrace();

}

}

};

writeThread.start();

}

}

}

**客户端：**

package main;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.Scanner;

/\*\*

\* 基于socket编程的多人聊天室客户端

\* @author 161620222刘鸿搏

\*

\*/

public class ManyPeopleClient {

private static final String IP = "127.0.0.1";

private static final int PORT = 6666;

private static String Name = "匿名用户";

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("请输入你的用户名:");

Name = s.next();

try {

Socket socket = new Socket(IP,PORT);

System.out.println(Name+"连接服务器成功！");

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter writer = new PrintWriter(socket.getOutputStream(),true);

Thread readThread = new Thread(){

public void run(){

String msg = "";

while(true){

try {

msg = reader.readLine();

} catch (IOException e) {

// TODO Auto-generated catch block

System.out.println("与服务器断开连接");

//e.printStackTrace();

break;

}

if(null !=msg&&msg!=""){

System.out.println(msg);

}

}

}

};

Thread writeThread = new Thread(){

public void run(){

while(true){

String msg = s.next();

writer.println(Name+":"+msg);

}

}

};

readThread.start();

writeThread.start();

} catch (UnknownHostException e) {

// TODO Auto-generated catch block

System.out.println("找不到服务器:"+IP);

//e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

System.out.println("与服务器断开连接");

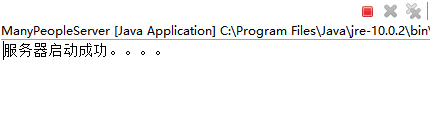
//e.printStackTrace();

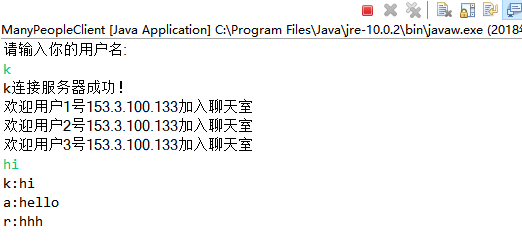
}

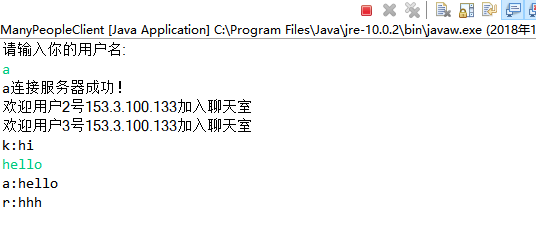
}

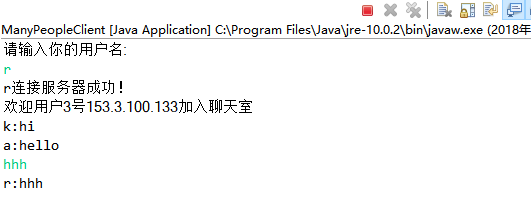
}

1. **实验结果及分析**









|  |  |  |  |
| --- | --- | --- | --- |
| 实验项目  名 称 | **实验二、网络爬虫** | 成绩 |  |

1. **实验目的**

用Python语言实现基于requests, bs4等库的网页数据爬取。实验包括酷狗音乐排行榜数据爬取、招聘网站数据爬取、异步爬取图片、多线程爬虫性能对比

1. 使用Python语言完成爬虫的设计，深入学习使用Python语言。
2. 使用网络编程，掌握基于Python数据挖掘相关库的编程，了解网页数据爬取的相关操作，掌握简单网络爬虫的开发。
3. 掌握网页代码分析和数据存储技术。
4. **实验内容与步骤**
5. **酷狗音乐排行榜爬取**

**网页分析**



需求为爬取歌曲排名、歌手、歌名、时长

打开Chrome开发者工具检查网页相关源码，找到相关代码



**代码编写**

**import** time  
**import** requests  
**from** bs4 **import** BeautifulSoup  
**from** pymongo **import** MongoClient  
client = MongoClient() # mongodb server  
songs = client.kugou\_db.songs # song collection  
headers = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'  
}  
**def get\_info**(url):  
 *'''获取酷狗音乐TOP500信息'''* wb\_data = requests.get(url, headers=headers)  
 soup = BeautifulSoup(wb\_data.text, 'lxml')  
 ranks = soup.select('.pc\_temp\_num') # 排名list  
 # titles = soup.select('.pc\_temp\_songlist > ul > li > a') # 名称list  
 titles = soup.select('.pc\_temp\_songlist > ul > li')  
 song\_times = soup.select('.pc\_temp\_time') # 歌曲时长list  
 **for** rank, title, song\_time **in** zip(ranks, titles, song\_times):  
 data = {  
 'rank': rank.get\_text().strip(),  
 # 'singer': title.get\_text().split('-')[0].strip(),  
 # 'song': title.get\_text().split('-')[1].strip(),  
 'singer': title.get('title').split('-')[0].strip(),  
 'song': title.get('title').split('-')[1].strip(),  
 'time': song\_time.get\_text().strip()  
 }  
 print(data)  
 song\_id = songs.insert\_one(data) # insert db  
 print(song\_id)  
 print('---------------------------------')  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 # 生成需要遍历的url  
 urls = ['http://www.kugou.com/yy/rank/home/{}-8888.html'.format(str(i)) **for** i **in** range(1, 24)]  
 **for** url **in** urls:  
 get\_info(url)  
 # time.sleep(1)

运行结果：

...

{'rank': '302', 'singer': 'T.R.Y.', 'song': '不是因为寂寞才想你', 'time': '4:20'}

<pymongo.results.InsertOneResult object at 0x04A51FD0>

---------------------------------

{'rank': '303', 'singer': '刀郎', 'song': '冲动的惩罚', 'time': '7:48'}

<pymongo.results.InsertOneResult object at 0x050D2A80>

---------------------------------

{'rank': '304', 'singer': '戴荃', 'song': '悟空', 'time': '7:22'}

<pymongo.results.InsertOneResult object at 0x04A51FD0>

---------------------------------

{'rank': '305', 'singer': '高进、小沈阳', 'song': '我的好兄弟', 'time': '4:42'}

<pymongo.results.InsertOneResult object at 0x050D2A80>

---------------------------------

{'rank': '306', 'singer': '林俊杰', 'song': '醉赤壁', 'time': '4:40'}

<pymongo.results.InsertOneResult object at 0x04A51FD0>

---------------------------------

{'rank': '307', 'singer': '曲肖冰', 'song': '静悄悄', 'time': '3:04'}

<pymongo.results.InsertOneResult object at 0x050D2A80>

---------------------------------

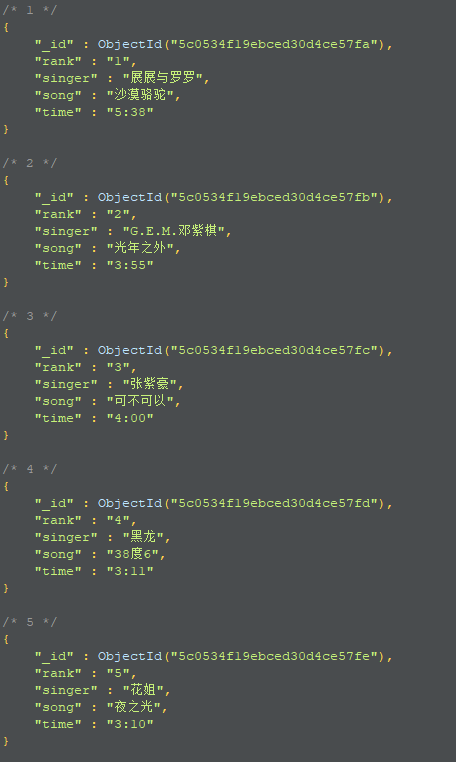
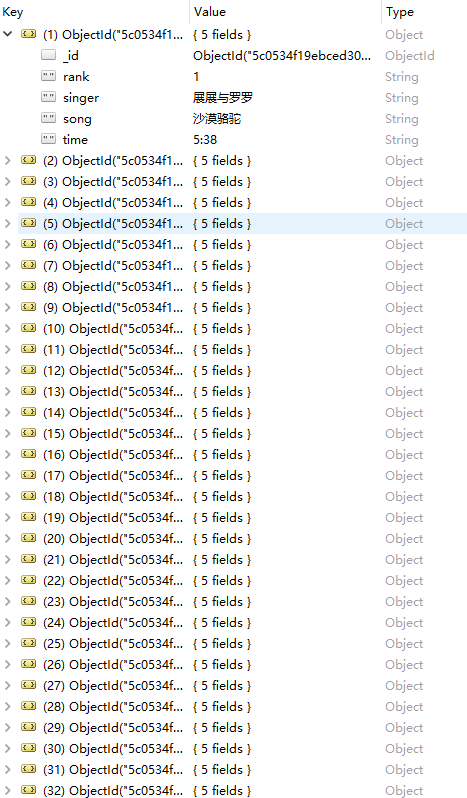
{'rank': '308', 'singer': 'David Guetta、Sia', 'song': 'Flames', 'time': '3:15'}

<pymongo.results.InsertOneResult object at 0x04A51FD0>

...

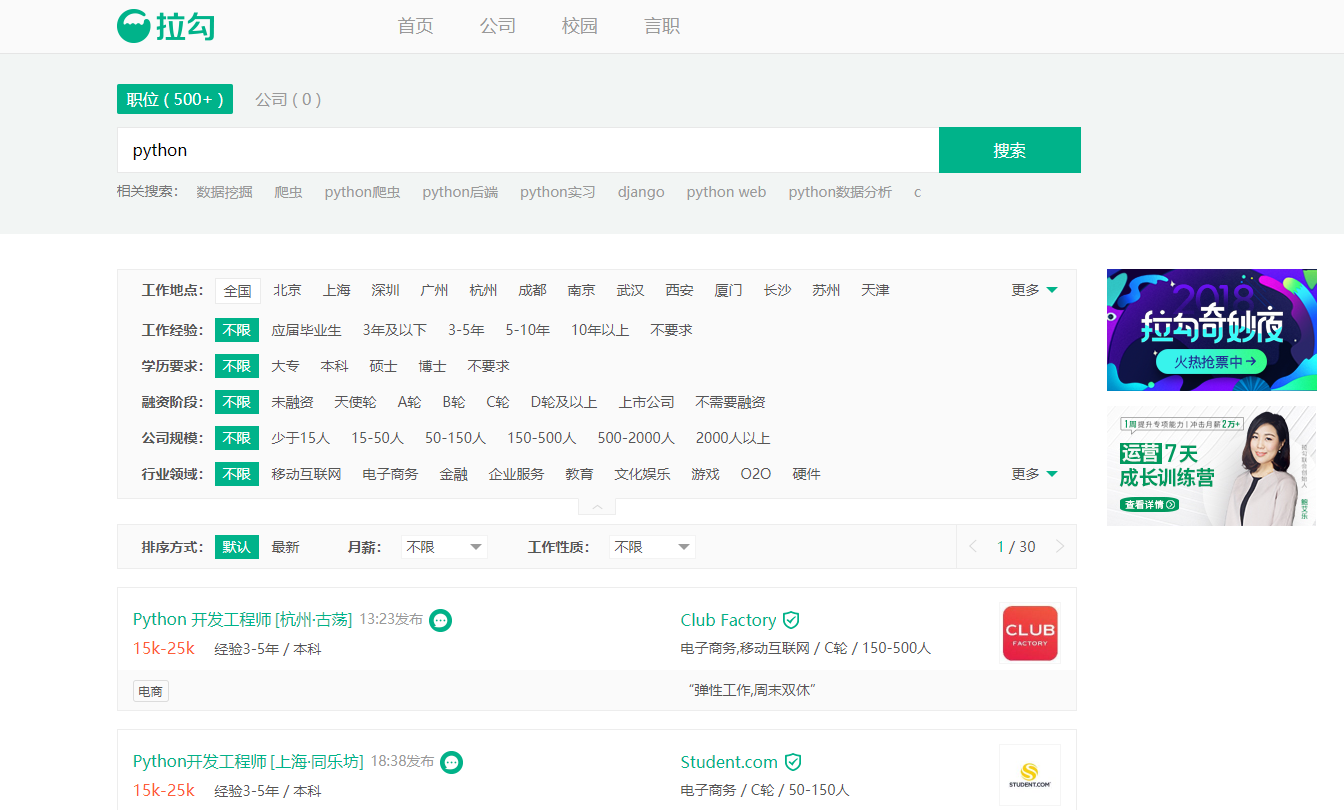
**数据存储**

使用mongdb数据库存储爬取到的信息



1. **招聘网站信息爬取**

**网页分析**

****

需求为爬取歌曲排名、歌手、歌名、时长

打开Chrome开发者工具检查网页相关源码，找到相关代码



**代码编写**

本网站使用cookie，所以在构造请求头时要包含cookie信息。

**import** json  
**import** math  
**import** time  
**import** pymongo  
**import** requests  
client = pymongo.MongoClient('localhost',27017)  
mydb = client['mydb']  
lagou = mydb['lagou']  
headers = {  
 'Accept': 'application/json, text/javascript, \*/\*; q=0.01',  
 'Accept-Encoding': 'gzip, deflate, br',  
 'Accept-Language': 'zh-CN,zh;q=0.8',  
 'Connection': 'keep-alive',  
 'Content-Length': '26',  
 'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8',  
 'Cookie': 'user\_trace\_token=20181204112323-6df9fdbf-f81d-4328-ae53-fd8df4d1b015; SL\_GWPT\_Show\_Hide\_tmp=1; SL\_wptGlobTipTmp=1; JSESSIONID=ABAAABAABEEAAJA820A1AD075588061911650D121ACC7B5; Hm\_lvt\_4233e74dff0ae5bd0a3d81c6ccf756e6=1543893829; sajssdk\_2015\_cross\_new\_user=1; sensorsdata2015jssdkcross=%7B%22distinct\_id%22%3A%22167773e462936b-00b7486db23f01-9393265-2073600-167773e462a1dd%22%2C%22%24device\_id%22%3A%22167773e462936b-00b7486db23f01-9393265-2073600-167773e462a1dd%22%2C%22props%22%3A%7B%22%24latest\_traffic\_source\_type%22%3A%22%E7%9B%B4%E6%8E%A5%E6%B5%81%E9%87%8F%22%2C%22%24latest\_referrer%22%3A%22%22%2C%22%24latest\_referrer\_host%22%3A%22%22%2C%22%24latest\_search\_keyword%22%3A%22%E6%9C%AA%E5%8F%96%E5%88%B0%E5%80%BC\_%E7%9B%B4%E6%8E%A5%E6%89%93%E5%BC%80%22%7D%7D; LGSID=20181204112349-042fbdf6-f774-11e8-89e2-525400f775ce; PRE\_UTM=; PRE\_HOST=; PRE\_SITE=; PRE\_LAND=https%3A%2F%2Fpassport.lagou.com%2Flogin%2Flogin.html%3Fts%3D1543893828599%26serviceId%3Dlagou%26service%3Dhttp%25253A%25252F%25252Fwww.lagou.com%25252Fjobs%25252F%26action%3Dlogin%26signature%3D41E03B5606589A61876B78A36B0FFB4E; LGUID=20181204112349-042fbff1-f774-11e8-89e2-525400f775ce; \_putrc=6D0CECC2C14C62A7123F89F2B170EADC; login=true; unick=%E6%8B%89%E5%8B%BE%E7%94%A8%E6%88%B75932; \_ga=GA1.2.360662032.1543893880; \_gat=1; hasDeliver=0; gate\_login\_token=ca8542d671ee564f2d95ced0125b337da1a0a596c867d7f4e4253694e6d3253d; index\_location\_city=%E6%B1%9F%E8%8B%8F; TG-TRACK-CODE=index\_navigation; X\_HTTP\_TOKEN=a62d43af3f2bbc33e9936fb6250de7d5; LGRID=20181204112649-6f5c8e9c-f774-11e8-8cb7-5254005c3644; Hm\_lpvt\_4233e74dff0ae5bd0a3d81c6ccf756e6=1543894009; SEARCH\_ID=332291ea9c3949cbabfb5fbe5ee2f827',  
 'Host': 'www.lagou.com',  
 'Origin': 'https://www.lagou.com',  
 'Referer': 'https://www.lagou.com/jobs/list\_python?labelWords=&fromSearch=true&suginput=',  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36',  
 'X-Anit-Forge-Code': '0',  
 'X-Anit-Forge-Token': 'None',  
 'X-Requested-With': 'XMLHttpRequest'  
}  
**def get\_page**(url,params):  
 html = requests.post(url,data=params,headers=headers)  
 json\_data = json.loads(html.text)  
 total\_count = json\_data['content']['positionResult']['totalCount']  
 page\_number = math.ceil(total\_count/15) **if** math.ceil(total\_count/15)<30 **else** 30  
 get\_info(url,page\_number)  
**def get\_info**(url, page):  
 **for** pn **in** range(1,page+1):  
 params = {  
 'first':'false',  
 'pn':str(pn),  
 'kd':'Python'  
 }  
 **try**:  
 html = requests.post(url,data=params,headers=headers)  
 json\_data = json.loads(html.text)  
 results = json\_data['content']['positionResult']['result']  
 **for** result **in** results:  
 infos = {  
 'businessZones': result['businessZones'],  
 'city': result['city'],  
 'companyFullName': result['companyFullName'],  
 'companyLabelList': result['companyLabelList'],  
 'companySize': result['companySize'],  
 'district': result['district'],  
 'education': result['education'],  
 'explain': result['explain'],  
 'financeStage': result['financeStage'],  
 'firstType': result['firstType'],  
 'formatCreateTime': result['formatCreateTime'],  
 'gradeDescription': result['gradeDescription'],  
 'imState': result['imState'],  
 'industryField': result['industryField'],  
 'jobNature': result['jobNature'],  
 'positionAdvantage': result['positionAdvantage'],  
 'salary': result['salary'],  
 'secondType': result['secondType'],  
 'workYear': result['workYear']  
 }  
 print('------------------')  
 print(infos)  
 lagou.insert\_one(infos)  
 time.sleep(2)  
 **except** requests.exceptions.ConnectionError:  
 **pass  
if** \_\_name\_\_ == "\_\_main\_\_":  
 url = 'https://www.lagou.com/jobs/positionAjax.json'  
 params = {  
 'first':'true',  
 'pn':'1',  
 'kd':'python'  
 }  
 get\_page(url,params)

运行结果：

...

------------------

{'businessZones': ['车陂', '东圃', '前进'], 'city': '广州', 'companyFullName': '广州回头车信息科技有限公司', 'companyLabelList': ['绩效奖金', '五险一金', '通讯津贴', '带薪年假'], 'companySize': '15-50人', 'district': '天河区', 'education': '大专', 'explain': None, 'financeStage': 'A轮', 'firstType': '开发|测试|运维类', 'formatCreateTime': '2天前发布', 'gradeDescription': None, 'imState': 'today', 'industryField': '生活服务', 'jobNature': '全职', 'positionAdvantage': '周末双休,弹性工作,五险齐全,带薪年假', 'salary': '15k-22k', 'secondType': '后端开发', 'workYear': '3-5年'}

------------------

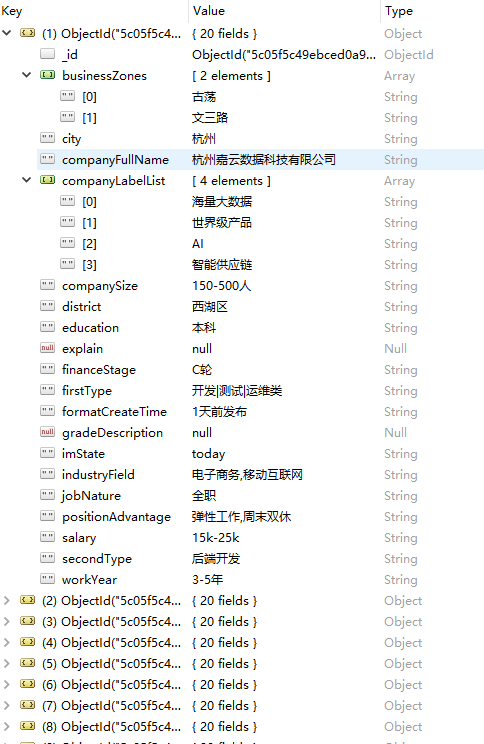
{'businessZones': ['朝外', '朝阳门', '东大桥'], 'city': '北京', 'companyFullName': '北京怡合春天科技有限公司', 'companyLabelList': ['股票期权', '专项奖金', '绩效奖金', '午餐补助'], 'companySize': '50-150人', 'district': '朝阳区', 'education': '本科', 'explain': None, 'financeStage': '不需要融资', 'firstType': '开发|测试|运维类', 'formatCreateTime': '18:37发布', 'gradeDescription': None, 'imState': 'today', 'industryField': '移动互联网,O2O', 'jobNature': '全职', 'positionAdvantage': '薪资高 福利多', 'salary': '15k-30k', 'secondType': '后端开发', 'workYear': '3-5年'}

------------------

{'businessZones': ['长阳路', '平凉路', '周家嘴路'], 'city': '上海', 'companyFullName': '上海敏为信息技术有限公司', 'companyLabelList': ['不差钱', '经纬中国', '核心竞争力', '弹性工作'], 'companySize': '50-150人', 'district': '杨浦区', 'education': '本科', 'explain': None, 'financeStage': 'A轮', 'firstType': '开发|测试|运维类', 'formatCreateTime': '15:10发布', 'gradeDescription': None, 'imState': 'today', 'industryField': '移动互联网,金融', 'jobNature': '全职', 'positionAdvantage': '时间自由,技术先进,区块链', 'salary': '15k-30k', 'secondType': '后端开发', 'workYear': '1-3年'}

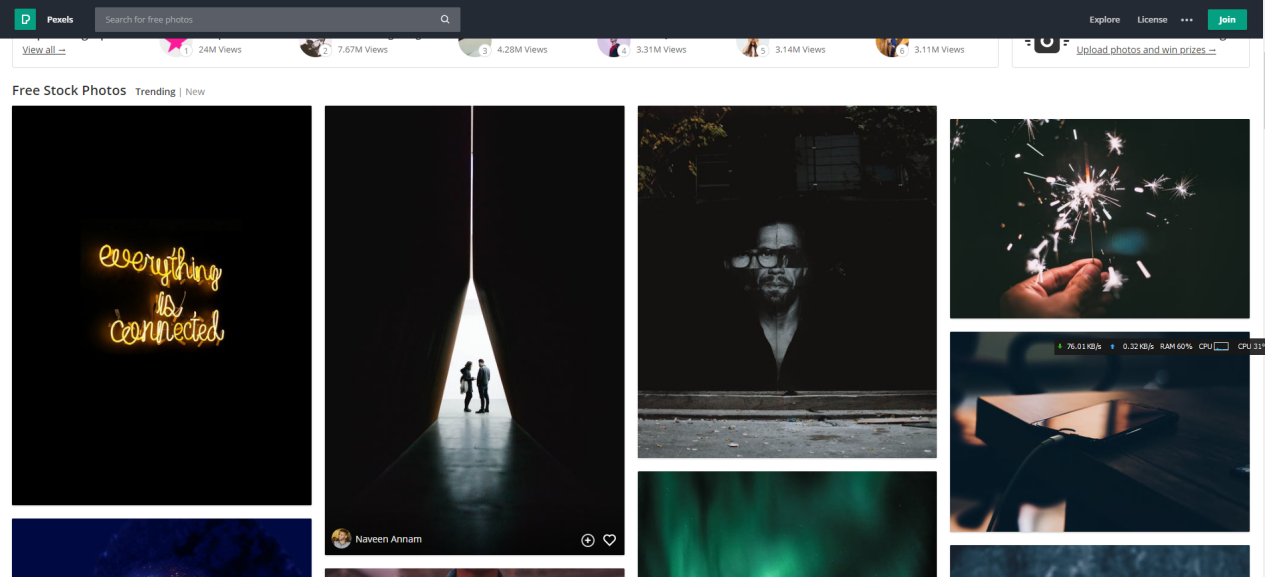
...

**数据存储**



1. **异步爬取图片**

**网页分析**



需求为爬取该网站的图片素材

打开Chrome开发者工具检查网页相关源码，找到相关代码



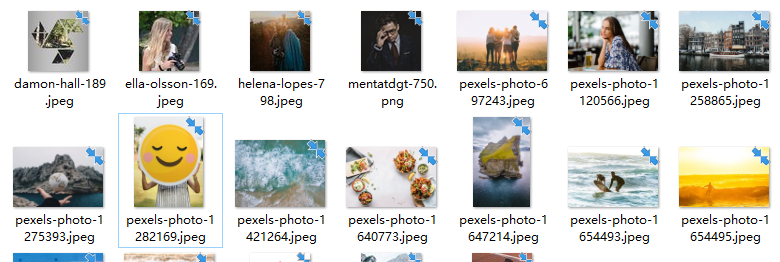
**代码编写**

该网站图片是异步加载的，但是通过改变url参数可以浏览到后面的图片。

爬取异步加载的数据有两种方法：修改url参数、模拟浏览器

**import** requests  
**import** re  
**from** bs4 **import** BeautifulSoup  
headers = {  
 'accept': 'text/javascript, application/javascript, application/ecmascript, application/x-ecmascript, \*/\*; q=0.01',  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'  
}  
urls = ['https://www.pexels.com/?page={}'.format(str(i)) **for** i **in** range(1, 2)]  
photos = []  
**for** url **in** urls:  
 wb\_data = requests.get(url, headers=headers)  
 soup = BeautifulSoup(wb\_data.text, 'lxml')  
 imgs = soup.select('article.photo-item > a > img')  
 **for** img **in** imgs:  
 photo = img.get('src')  
 photos.append(photo)  
 # print(photo)  
path = './yibu-images/'  
**for** item **in** photos:  
 data = requests.get(item, headers=headers)  
 photo\_name = re.findall('\d+\/(.\*?)\?', item)  
 print(photo\_name)  
 **if** photo\_name:  
 fp = open(path + photo\_name[0], 'wb')  
 fp.write(data.content)  
 fp.close()

**数据存储**



1. **多线程爬虫性能对比**

**网页分析**





**代码编写**

**import** re  
**import** time  
**from** multiprocessing **import** Pool  
**import** requests  
headers = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0'  
}  
**def re\_scraper**(url):  
 res = requests.get(url,headers=headers)  
 names = re.findall('<h2>(.\*?)</h2>', res.text, re.S)  
 contents = re.findall('<div class="content">.\*?<span>(.\*?)</span>', res.text, re.S)  
 laughs = re.findall('<i class="number">(\d+)</i>',res.text,re.S)  
 comments = re.findall('<i class="number">(\d+)</i>', res.text, re.S)  
 infos = list()  
 **for** name,content,laugh,comment **in** zip(names,contents,laughs,comments):  
 info = {  
 'name':name,  
 'content':content,  
 'laugh':laugh,  
 'comment':comment  
 }  
 infos.append(info)  
 **return** infos  
**if** \_\_name\_\_ == "\_\_main\_\_":  
 urls = ['https://www.qiushibaike.com/8hr/page/{}/'.format(str(i)) **for** i **in** range(1, 36)]  
 start\_1 = time.time()  
 **for** url **in** urls:  
 re\_scraper(url)  
 end\_1 = time.time()  
 print('串行爬虫耗时:',end\_1 - start\_1)  
 start\_2 = time.time()  
 pool = Pool(processes=2)  
 pool.map(re\_scraper,urls)  
 end\_2 = time.time()  
 print('2进程爬虫耗时:',end\_2 - start\_2)  
 start\_3 = time.time()  
 pool = Pool(processes=4)  
 pool.map(re\_scraper,urls)  
 end\_3 = time.time()  
 print('4进程爬虫耗时:',end\_3 - start\_3)

**运行结果：**

串行爬虫耗时: 9.87952971458435

2进程爬虫耗时: 8.341878652572632

4进程爬虫耗时: 3.208430767059326

1. **实验总结**

网络爬虫可以高效地帮助我们获取网上有用的信息，开发爬虫有以下技巧以满足需求：构造有cookie的请求头来爬取使用cookie的网站；使用模式匹配以及其他分析工具寻找网页代码中有用的信息；使用数据库存储爬取的信息；使用多线程爬虫提高爬取效率。