

专业课程设计II报告

（ 2020 / 2021学年 第 一 学期）

题 目： 网络爬虫设计与实现

|  |  |
| --- | --- |
| **专 业** | **信息安全** |
| **学 生 姓 名** | **王竞誉** |
| **班 级 学 号** | **B17040701** |
| **指 导 教 师** | **王波** |
| **指 导 单 位** | **信息安全系** |
| **日 期** | **2020.12.21-2020.12.31** |

|  |  |  |
| --- | --- | --- |
| **课程目标** | **评价准则** | **计分（每项10分）** |
| 课程目标1：通过课程设计，培养学生综合应用信息安全、计算机技术等领域专业知识的技能。（20分） | 1、能够掌握信息安全的相关基础知识，并能够针对求解的工程问题，收集资料进行合理的分析与设计。 |  |
| 2、从软件的分析、设计到编制调试，结合计算机网络理论知识、编程语言以及程序设计的方法，能够解决一个和信息安全相关的问题。 |  |
| 课程目标2：解决信息安全领域复杂工程问题的实践创新能力。（20分） | 3、通过调研，能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行编程实现。 |  |
| 4、具备一定的人机交互设计意识，人机交互设计合理、友好，操作简便。 |  |
| 课程目标3：文献调研与资料收集能力，问题发现、研究、分析与解决能力。（10分） | 5、具备一定自学能力与探索创新意识，能够充分利用教科书及其资源（如网络等）自学新知识与新技能。 |  |
| 课程目标4：培养工程工具运用能力，能够利用仿真软件或实验系统对信息安全系统进行模拟和预测，并理解仿真软件或实验硬件的局限性。（20分） | 6、能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，理解相关算法，对求解的工程问题进行有效建模和求解。 |  |
| 7、掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并能够进行充分测试。 |  |
| 课程目标5：分组完成一次项目设计与开发的全过程，组内成员通过讨论和交流解决课程设计中的难题，能在实验报告中准确阐述课程设计的内容，能够清晰陈述观点和回答问题。（30分） | 8、组内成员之间有一定的团队合作，互通有无。 |  |
| 9、具备一定的语言表达能力与文字处理能力，能够结合复杂工程问题撰写报告，报告内容和实验数据详实，格式规范。 |  |
| 10、能够正确、完整地回答指导教师关于课题的问询，反映其对课题内容，以及相关的工程基础知识具有较好的理解和掌握。 |  |
| 专业课程设计II能力测评总分 | |  |
| **1.jpeg**  **指导教师： 年 月 日** | | |
|  | | |

**网络爬虫设计与实现**

1. **课题内容和要求**
2. 课题内容

网络爬虫是一个自动提取网页的程序，可以按照事先制定的规则进行浏览，这些规则我们称之为网络爬虫算法。搜索引擎离不开爬虫，比如百度搜索引擎的爬虫叫做百度蜘蛛。百度蜘蛛每天会在海量的互联网信息中进行爬取，爬取优质信息并收录，当用户在百度搜索引擎上检索对应关键词时，百度将对关键词进行分析处理，从收录的网页中找出相关网页，按照一定的排名规则进行排序并将结果展现给用户。在这个过程中，百度蜘蛛起到了至关重要的作用。网络爬虫与网站安全息息相关，向网络爬虫开放网站希望推广的页面，屏蔽比较敏感的页面，对于保持网站的安全运行、保护用户的隐私是极其重要的。

网络爬虫从一个或若干初始网页的URL开始，获得初始网页上的URL，在抓取网页的过程中，不断从当前页面上抽取新的URL放入队列,直到满足系统的一定停止条件。聚焦爬虫的工作流程较为复杂，需要根据一定的网页分析算法过滤与主题无关的链接，保留有用的链接并将其放入等待抓取的URL队列。然后，它将根据一定的搜索策略从队列中选择下一步要抓取的网页URL，并重复上述过程，直到达到系统的某一条件时停止。另外，所有被爬虫抓取的网页将会被系统存贮，进行一定的分析、过滤、去重，并建立索引，以便之后的查询和检索；对于聚焦爬虫来说，这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。

1. 课题要求

了解网络爬虫进行提取网页内容爬取的基本原理，设计实现具有较复杂功能的网络爬虫程序：

（1）设计实现的网络爬虫程序具有一定的实用价值，能选择某一种特定信息进行爬取，例如招聘网站信息、淘宝商品信息、知乎某主题信息、微博信息、学校学生竞赛获奖信息等。

（2）程序具有一定的效率和健壮性，例如，在单机上使用多线程提高效率，也可以考虑使用分布式结构，在多台机器上执行。避免扫描重复的页面，具有一定的查重功能。

（3）有效地进行信息抽取和存储，例如按需要抽取出网页上所有的地址信息“南京市栖霞区文苑路9号”，或者提取出网页上所有竞赛获奖信息，“全国大学生信息安全竞赛一等奖”等等。

（4）设计程序总体结构，初步实现对某一域名下的所有网页进行爬取，将爬取的URL存放在一个文件中。

**二、需求分析和总体设计**

2.1本课题的主要功能包括：

（1）选择某一特定网址进行爬取

（2）有效地进行信息抽取和存储

（3）设计了完备的数据库存储功能和日志系统

（4）实现对某一域名下的所有网页和关键词进行爬取

（5）实现URL分类：内部、外部和IPC，且能够避免扫描重复的页面

（6）设计了整洁清楚、人机交互友善的主界面建议适当的用例图、序列图或状态图进行描述。

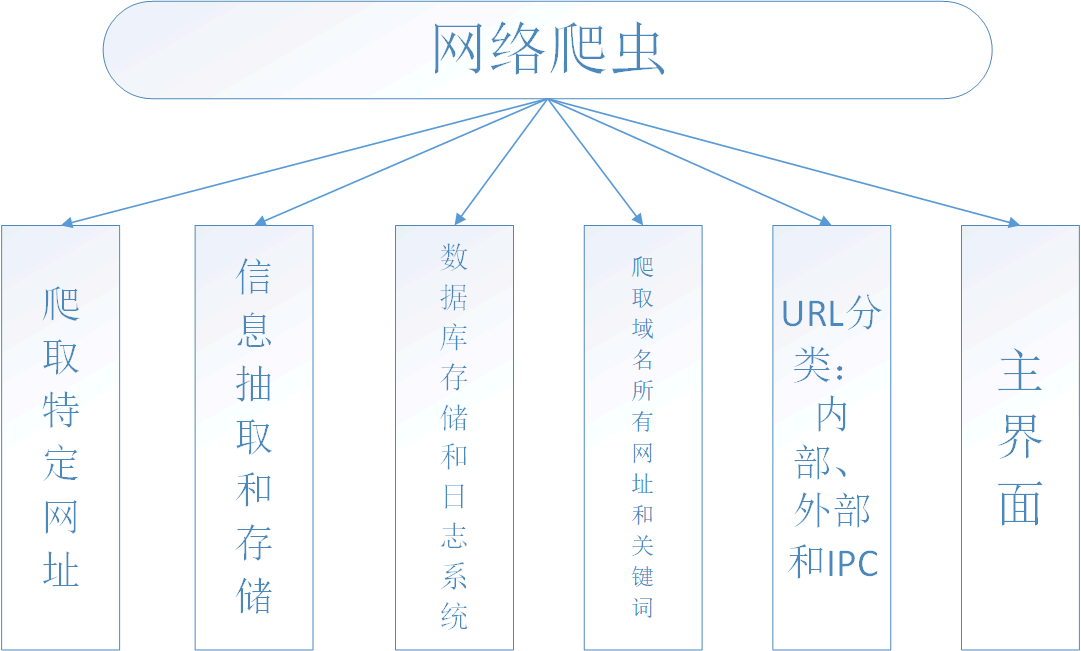


图1 主要功能

2.2 本课题的数据表单设计

以爬取的URL哈希为名建表，SUBURI存放爬取到的域名、链接，LOPCNT默认值为0，LOCATE存放链接类型（内部链接为0、外部链接为1、IPC为2）。如图2所示。

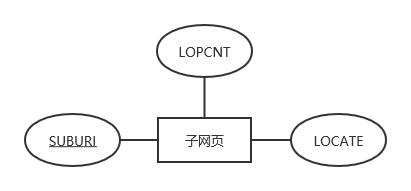


图2 数据表单ER图

2.3 本课题的体系结构设计

这次课题我们使用了浏览器/服务器架构模式（b/s），它的优点是客户端不需要做维护，维护成本低一点。b/s的web应用程序组成有：1、浏览器2、服务器3、http协议4、网页。

我们中间层的WEB服务器使用了Flask并静态生成HTML，数据层使用了日志和数据库。如图3所示。

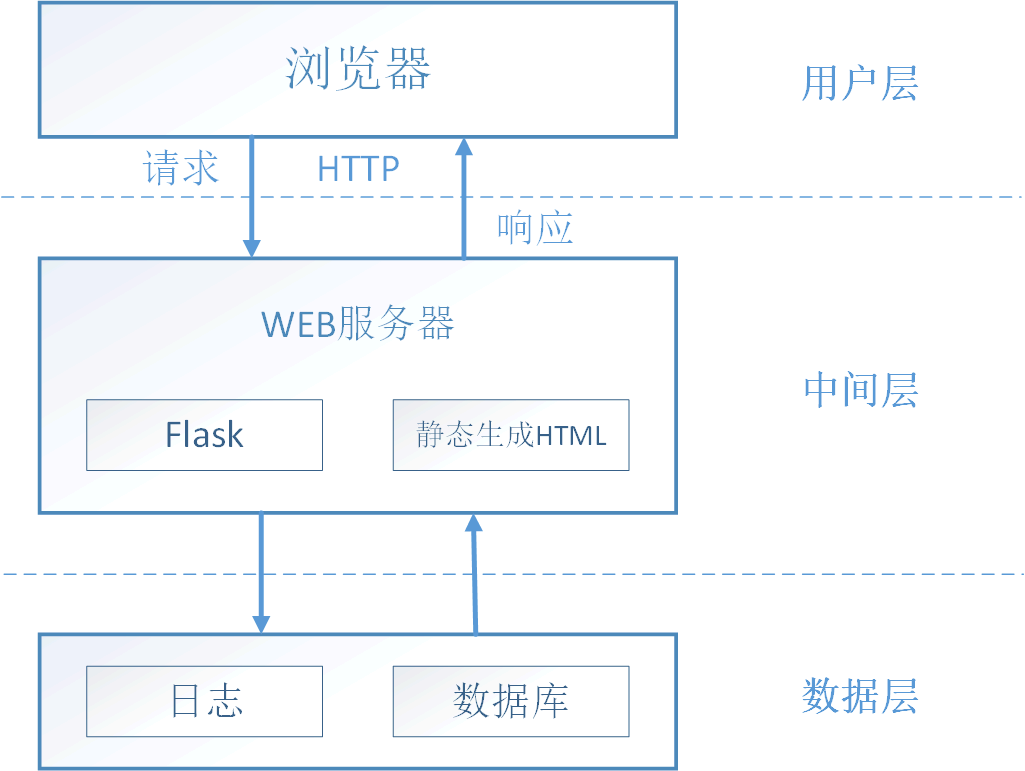


图3 体系结构设计

2.4 本课题的主要功能界面设计

Index界面提供输入框与提交按钮，用户可以在输入框输入需要爬取网页的url链接，通过提交按钮开始程序，如图4。跳转后页面以表格形式显示爬取结果，表格以三列的形式显示，分表列明爬取后得到的连接类型、链接和关键词，如图5。

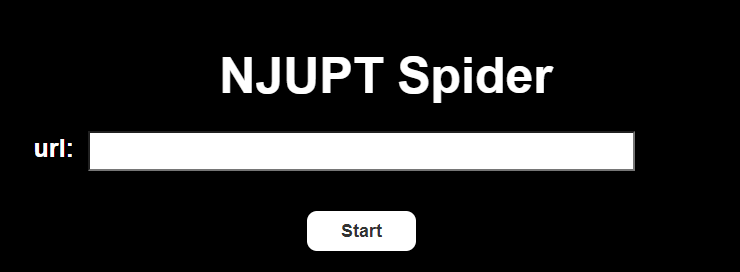


图4 输入界面设计



图5 输出界面设计

**三、相关功能模块详细设计**

1 模块内的数据结构设计

（1）输入页面数据结构

<body>

<form action="" method="post" >

<div id="inBox">

<!-- NJUPT Spider -->

<p>NJUPT Spider</p>

<!-- input框 -->

<div>

<p>url:</p>

<input type="text" name="url" id="inText" class="inEle" >

</div>

<!-- 提交按钮 -->

<input type="submit" value="Start" id="inSub" class="inEle" >

</div>

</form>

</body>

（2）输出页面数据结构

<table class="dataframe" >

#表格头部

<thead>

<tr>

<th>链接类型</th>

<th>链接</th>

<th>关键词</th>

</tr>

</thead>

#表格主体

<tbody>

<tr>

<th>{type}</th>

<th>{link}</th>

<th>{tags}</th>

</tr>

</tbody>

</table>"

2 模块程序流程图设计

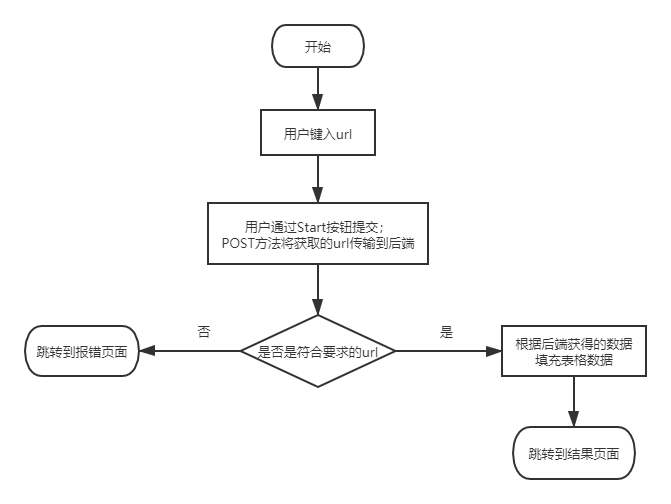


图6 前端部分流程图

**四、部分核心代码**

1.输入页面

（1）index.html头部

<!-- 使用bootstrap框架 -->

{% extends "bootstrap/base.h

tml" %}

{%- block head %}

    <title>NJUPT Spider V0.1</title>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="description" content="">

    <meta name="author" content="NJUPT">

    {%- block styles %}

    <!-- Bootstrap -->

    <link href="{{bootstrap\_find\_resource('css/bootstrap.css', cdn='bootstrap')}}" rel="stylesheet">

      <!-- Bootstrap core CSS -->

{#    <link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">#}

    <!-- bb——这里是通过下载网页上的css文件，保存在static目录下即可-->

    <link href="static/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom styles for this template -->

    <!-- bb——这里是通过下载网页上的css文件，保存在static目录下即可-->

    <link href="static/css/carousel.css" rel="stylesheet">

    <!-- <link href=index.css" rel="stylesheet"> -->

    <!-- <style type="text/css">

        body{

            background-color: black;

        }

    </style> -->

    {%- endblock styles %}

{%- endblock head %}

该部分采用python中的书写方式，设计了html文件的规范，并且引入bootstrap框架，使flask-bootstrap框架可以实现以完成前后端内容的交互，并且flask-bootstrap框架可以便利的搭建可显示的页面http://127.0.0.1:9999/。

（2）index.html主体部分

{% block body -%}

<body style="background-color: black;">

    <!-- 创建表单，定义表单数据发送方法 -->

    <form action="" method="post" >

        <div id="inBox" style="position: absolute; height:300px; width: 80%; top: 20%;">

            <!-- NJUPT Spider -->

            <p style="font-size: 50px; font-weight: bold; color: white; position: absolute; margin: auto;

left: 37%; right: 30%; top: 10px;">NJUPT Spider</p>

            <!-- input框 -->

            <div style="position: absolute;height: 40px; width: 100%;  top: 100px; ">

                <p style="font-size: 25px; font-weight: bold; left: 20%; margin: 0;position: absolute;

color: white;">url:</p>

                <input type="text"  name="url" id="inText" class="inEle" style="position: absolute;

margin: auto; left: 25%; right: 25%; height: 40px; width: 50%; background-color: white;">

            </div>

            <!-- 提交按钮 -->

            <input type="submit" value="Start" id="inSub" class="inEle" style="position: absolute;

margin: auto; left: 30%; right: 30%; height: 40px; width: 10%;  top: 180px; background-color:

white; border: 0;font-size: large; font-weight: bold; border-radius:10px;" >

        </div>

    </form>

</body>

{%- endblock body %}

主体部分对UI进行布置和格式的设计。主要使用内联式CSS以达到在维持bootstrap框架的同时可以根据自定义调整外观及结构。与后端交互部分使用创建表单并使用“POST”方式，从文本框中获取用户输入的内容并将参数传输到后端。

2.输出页面

def formatTable(result: list):  
 *# 将爬虫返回值（列表嵌套字典）转化为HTML表格* r = "<body style=\"background-color:black;\">\r\n"  
 r += "<p style=\"font-size:50px; color: white; width:50%; left:45%;position:relative;\">RESULT

</p>\r\n"  
   
 *#表格* r += "<table class=\"dataframe\" border=\"0\" style=\"word-break: break-all; color: white;

position: absolute; top: 100px; margin-left:10%; margin-right:10%; \" >\r\n"  
   
 *#表格头部* r += "\t<thead>\r\n"  
 r += "\t\t<tr style=\"height: 40px; margin: 20px; font-size: 18px; font-weight: lighter;\">\r\n"  
 r += "\t\t\t<th style=\"text-align: center;\">链接类型</th>\r\n"  
 r += "\t\t\t<th style=\"text-align: center;\">链接</th>\r\n"  
 r += "\t\t\t<th style=\"text-align: center;\">关键词</th>\r\n"  
 r += "\t\t</tr>\r\n"  
 r += "\t</thead>\r\n"  
   
 *#表格主体* r += "\t<tbody>\r\n"  
 *#获取到返回值后根据格式自动填充表格* for elem in result:  
 r += "\t\t<tr style = \"height: 40px; font-size: 15px; font-weight: thin;\">\r\n"  
 *#根据url\_type判断连接内类型并填入表格* r += "\t\t\t<th style=\"text-align: center;\">{type}</th>\r\n".format(type="内部链接" if elem

["url\_type"] == 0 else "外部链接")  
 *#填入获取到的链接* r += "\t\t\t<th style=\"text-align: center;\">{link}</th>\r\n".format(link=elem["url"])  
 *#填入获取到的关键词* r += "\t\t\t<th style=\"text-align: center;\">{tags}</th>\r\n".format(tags=' '.join(elem["tags"]))  
 r += "\t\t</tr>\r\n"  
 r += "\t</tbody>\r\n"  
 r += "</table>"  
 r +="</body>"  
   
 return r

在python项目的NS\_Utils.py文件中进行输出页面的前端书写，以变量r作为前端代码的返回变量。输出界面以表格为主，表格头部即标题部位固定不变为“连接类型”、“链接”和“关键词”，表格主体部分则由爬虫后获取的数据为填充，以format方式将名为result的list中获得到的数据根据格式要求填入表格对应位置中。

3.前后端交互

*# 引入flask-bootstrap框架*app = Flask(\_\_name\_\_)  
bootstrap = Bootstrap(app)  
  
  
*# 通过methods实现交互*@app.route(**'/'**, methods=[**'GET'**, **'POST'**])  
**def** defaultHandler():  
 **if** request.method == **'GET'**:  
 **return** render\_template(**'index.html'**)  
 **else**:  
 logger = initLog()  
 connector = initDb(logger)  
 A = Spider(request.form[**"url"**], logger, timeout=10.0)  
 SpiderResult = A.takeAction(connector)  
 stopDb(logger, connector)  
 **return** formatTable(SpiderResult)

*# python开始运行时打开flask-bootstrap框架下端口9999的网页***if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 app.run(port=9999, debug=**True**)

后端NS\_main.py文件中import引入flask和bootstrap框架，通过methods中的GET和POST两种方法实现前端用户输入的url可以被后端获取

**五、软件测试及其结果分析**

1.对正确输入的url进行测试

（1）测试1，使用url：https://www.sogou.com

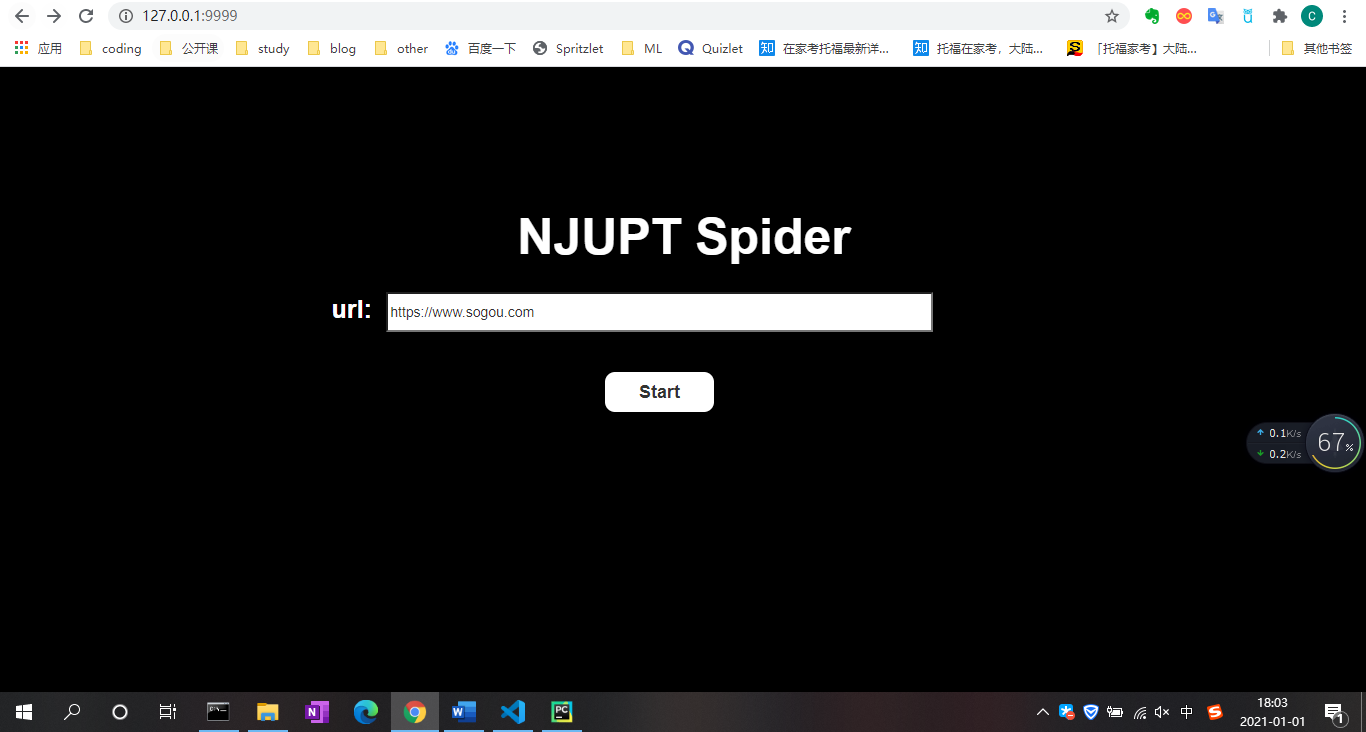


图7 测试1输入界面

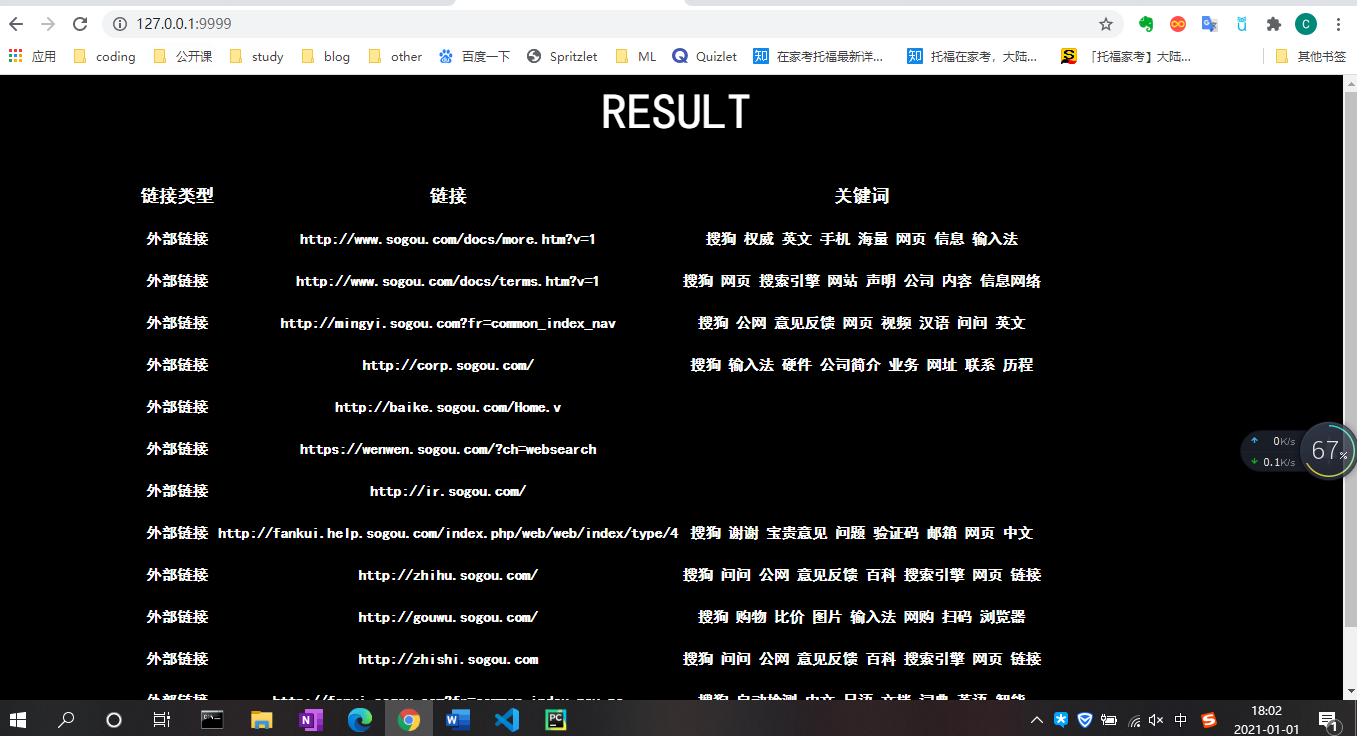


图8 测试1结果界面

（2）测试2，使用url：https://blog.csdn.net/

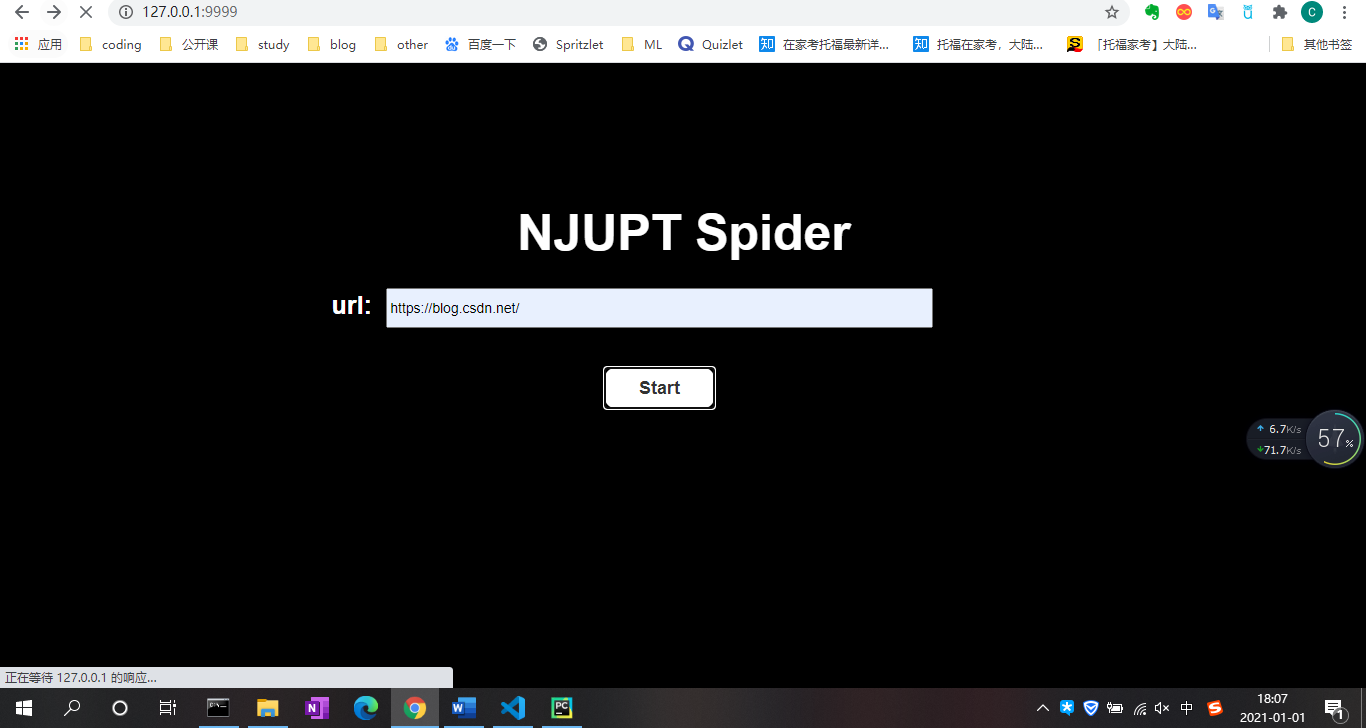


图9 测试2输入界面

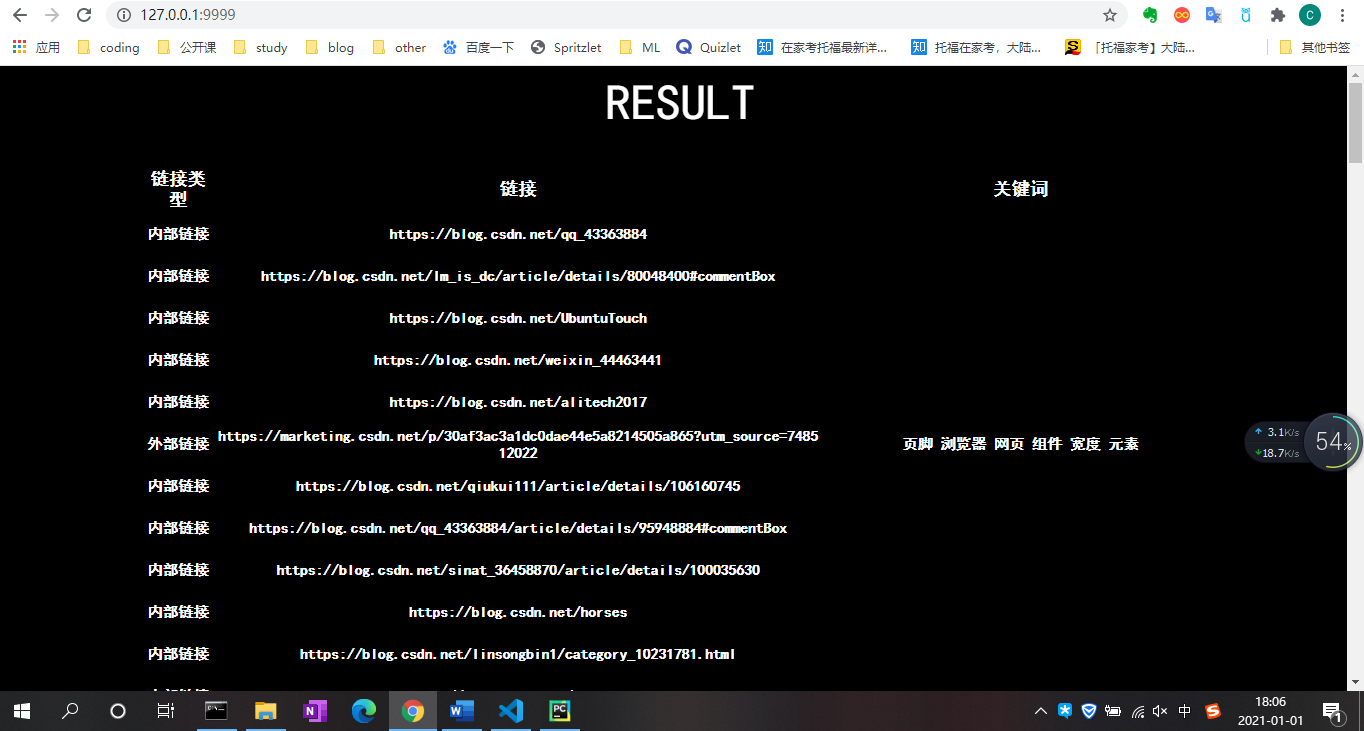


图10 测试2输出界面

2.对错误的url进行测试

（1）对输入格式错误的url进行测试，测试用url：www.baidu.com

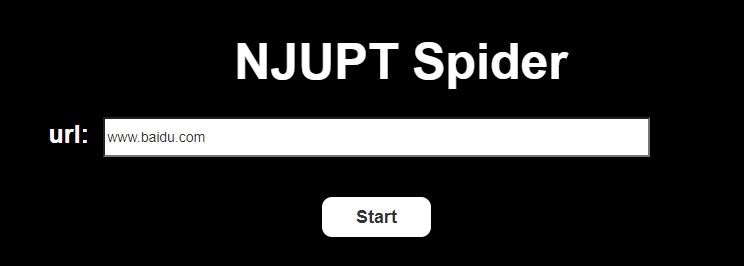


图11 错误的输入格式

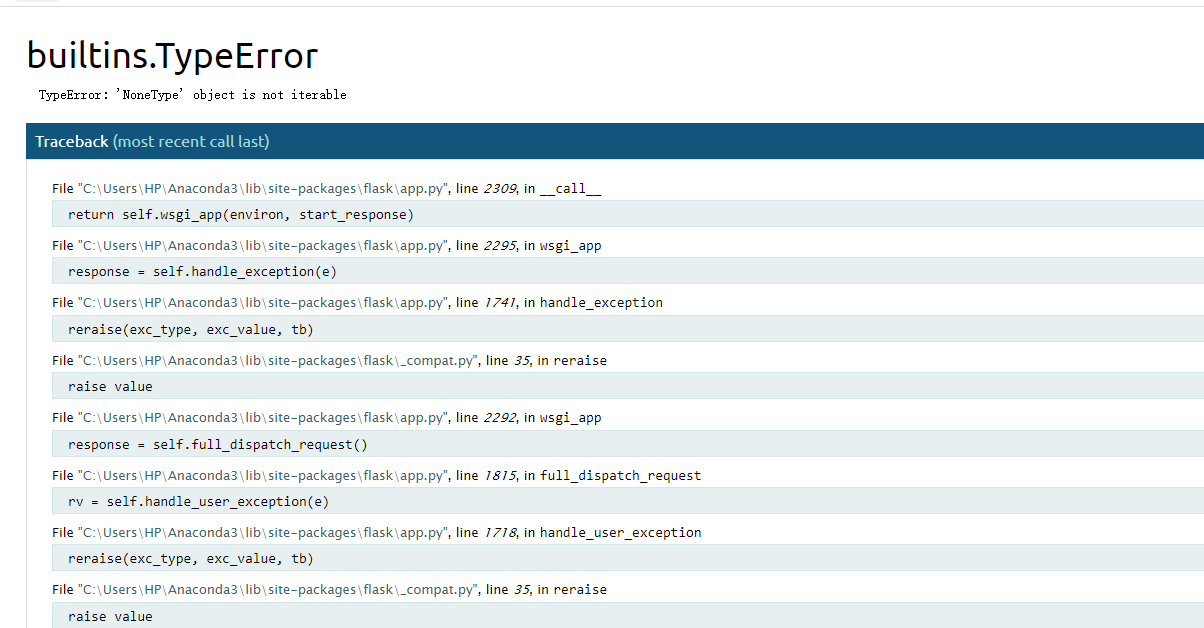


图12 提交格式错误的url跳转到的报错页面

（2）对不存在的网址进行测试，测试用url：https://www.baidu

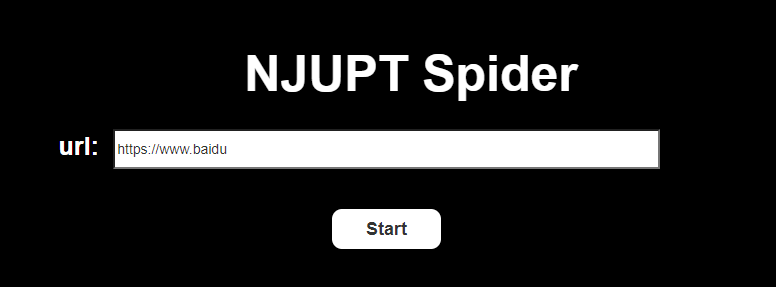


图13 不存在的url

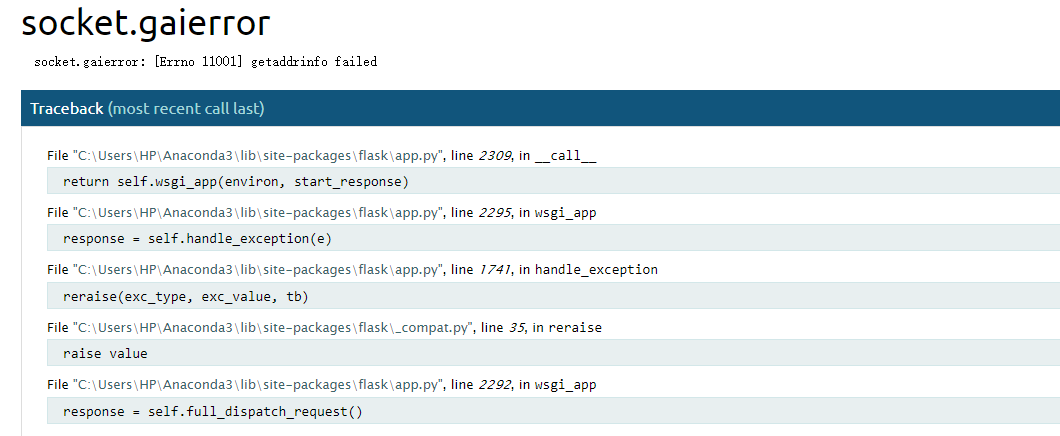


图14 提交不存在的url跳转到的报错页面

3.结果分析

在前端界面中提交格式正确且确实存在的url链接则可以准确获得需要爬取的内容，若提交有错误的url也可以获得报错返回页面，证明前端搭建无误且可以与后端功能进行正常交互。

**六、课题完成过程中遇到的问题及解决方法**

问题1：在设计前端样式并实现的过程中发现，使用index.css文件进行外联式CSS样式自定义时，CSS文件中的样式可以在静态页面演示中显示，但无法在与后端交互后的页面上显示。改为内嵌式CSS之后问题依旧无法解决。

解决方法：查阅资料后可知，由于引入了flask-bootstrap框架且bootstrap框架具有一定的优先级，所以自定义的外联式及内嵌式CSS无法覆盖bootstrap框架中的样式。若需使用自定义的CSS样式则需要使用内联式CSS即行内式CSS进行style定义。

问题2：在对输出页面的输出表格进行设计时，需要使表格中的内容居中显示，但在<table>及<tr>标签中进行style定义”text-align: center”都无法改变表格中内容的对齐方式。

解决方法：查阅资料后得知，”text-align: center”语句只有在<th>标签中使用才能使表格中文字内容居中。

问题3：在调整页面中UI的位置时发现，定义了元素的位置距离后元素依旧无法改变位置。

解决方法：检查代码后发现，若需要对未知的定义生效，需要先定义当前元素与其所处容器的position属性，而该元素的容器没有定义position属性。

**七、总结**

本次程序设计的题目是网络爬虫设计与实现，在分析了题目的需求后，我们很快地意识到，最适合这个设计题目且最高效、便捷的程序语言就是python，而根据题目中的要求，最后的产品应当是一个前后端相结合、能够使用户进行简洁的人机交互的产品，于是在小组内的商讨之后，我作为对前端较有经验的组员负责起了前端设计与实现以及一部分前后端交互的工作。

在思考前端的实现过程中，我考虑了两种实现可视化的方法，其一是使用PyQt实现GUI界面，另一种是使用web实现界面。考虑到通常大部分用户喜爱的搜索类产品都是以web的形式呈现且PyQt实现的GUI界面相对于web页面来说较为单调，我最后决定使用后者来完成可视化页面。

在实现的过程中，我充分的锻炼了前端代码的能力，同时也在实践中发现了自己从前未知的知识盲点。在我不是很熟悉的前后端交互领域，我也通过这次程序设计查找了相关资料进行学习，并且最终完成了程序的需求。

在调试的过程中，我使用了静态测试和动态测试相结合的方法，也在这个过程中发现了两者间的差异。

除此之外，在整个程序设计的过程中，我与小组的成员们也能做到及时高校的沟通，我们的团队合作能力都得到了了长足的进步。

总之，通过这次程序设计，我完成项目以及各个方面的能力都得到了发展与加强，我也收获了宝贵的经验。