电工导 C 第七次实验报告

姓名: 宋士祥　　学号:521030910013　　班级:F2103001

2022 年 11 月 6 日

1　实验概览

介绍了 Flask 框架的相关知识，Flask 框架是一个轻量级别的 Web 框架，利用 Python 编写，用较小的成本便可编写一个网站。

Flask 可以通过使用模板的方式来实现返回相应内容的作用，其中：
（1）模板其实是一个包含响应文本的文件，其中用占位符 (变量) 表示动态部分，告诉模板引擎其具体的值需要从使用的数据中获取。
（2）使用真实值替换变量，再返回最终得到的字符串，这个过程称为“渲染”。
（3）Flask 是使用 Jinja2 这个模板引擎来渲染模板

基于上述内容，本次实验需要我们基于之前的内容完成一个搜索引擎的界面建设。

2　实验环境

本次实验采用所需的实验环境如下：

- Docker 中的`sjtumic/ee208` 镜像
- Python3（使用 VSCode 编译）
- `BeautifulSoup4` 扩展以及`lxml` 扩展。
- `java` 环境及`lucene` 扩展（在 SJTU EE208 中已经给出）。
- Flask 等模板（在 SJTU EE208 中已经给出）。

3　问题重述与代码说明

本次实验要求使用 Flask，结合前面学习的 HTML, Lucene，中文分词等知识点，根据上次实验爬取的网页，建立一个简单的搜索引擎。界面要求含有：

- 标题
- 超链接
- 关键词上下文
- 网址

本次实验分为下几个步骤实现：

3.1 改造 SearchFiles.py

本次实验主要根据 LAB4 中的代码进行实现。在我们的设计中，我们需要返回网页的对应内容，因此我们需要设计一个方式将结果导出。

首先我们删除了所有的**print** 函数，同时新建了一个 html 文件用于存储我们的结果（这一部分在 app.py）。为了防止反复写入覆盖不全面的情况出现，我们采用直接检测该文件是否存在，如有直接删除的方式进行。

我们将整个文件视为一个类，同时将文件视为一整个类，同时使用一个 main 函数作为整个函数每次启动的依据。

而对于返回的文件内容，我们一方面修改了 IndexFiles 使得 contents 也是返回值的一部分，另一方面我们要对返回的内容进行修正使得网页输出为 html 格式。

我们采取的方式是将整个 SearchFiles 视为一个大类，results 和 numDocs 作为输出结果。需要留心的是，我们采用了 lucene 的 Highlighter 用来对关键词进行高亮。相关代码如下：

```
    query = QueryParser("contents", analyzer).parse(command)
    formatter = SimpleHTMLFormatter("<font color='red'>", "</font>")
    fragmentScorer =  QueryScorer(query)
    highlighter = Highlighter(formatter, fragmentScorer)
    fragmenter = SimpleFragmenter(50)
    highlighter.setTextFragmenter(fragmenter)
    scoreDocs = searcher.search(query, 50).scoreDocs
    numDocs = len(scoreDocs)
    for i, scoreDoc in enumerate(scoreDocs):
        doc = searcher.doc(scoreDoc.doc)
        docContent = doc.get("contents")
        hc = highlighter.getBestFragment(analyzer, "contents", docContent)
        if (not hc):
            if(len(docContent)>=50) :
                hc = docContent[0:50]
            else :
                hc = docContent
        try:
            results += [[doc.get("path"), hc, doc.get("url"), doc.get("title")]]
        except:
            i = 0
```

3.2 对 app.py 的修正

app.py 在提供的代码示例中已经给出，但是显然它不符合我们的要求，我们要对它进行修改。

我们将 SearchFiles 作为一个大类，将 numDocs 和 results 作为我们的输出来源。

我们预期的输出 html 为类似百度的页面，我们的返回值也与之类似，对应的结果如下：

```
def showbio():
    search = request.args.get('search')
    SearchFiles.command = search
    SearchFiles.main()
    result = '<p找到>{}个结果。</p>\n'.format(SearchFiles.numDocs)
    for i in SearchFiles.results:
        result += "<p><font size=5><a href = \"{}\">{}</a></font></p>\n<p><font size=4>{
                                                    }</font></p>\n<p><font size=3 color=
                                                    green>{}</font></p>\n".format(i[2],
                                                    i[3],i[1],i[2])
```

```
    try:
        os.remove("templates/result.html")
    except:
        i = 1
    file = open("templates/result.html","w")
    file.write(result)
    file.close()
    if request.method == "POST":
        search = request.form['search']
    return render_template("show_bio.html", search=search)
```

3.3 对虚拟机的启动问题的修正

经过助教和同学的提醒，我们发现存在 java 虚拟机的启动问题。如反复启动，会导致 java 虚拟机已启动而重复启动的问题，我们直接采取 ppt 上所述的方法：

```
# SearchFiles.py
    try:
        vm_env = lucene.initVM(vmargs=['-Djava.awt.headless=true'])
        print ('lucene', lucene.VERSION)
    except:
        vm_env = lucene.getVMEnv()
    vm_env.attachCurrentThread()

# app.py
@app.before_first_request
@app.route('/showbio', methods=['POST','GET'])
def showbio():
    ...
```

3.4 界面的设计

我们设计了两个页面用于展示，分别为`bio_from.html` 和`show_bio.html`，分别用于主页与搜索用。

对于主页，我们定义了一个搜索框用于搜索，并将搜索文件名定义为 search 用于搜索，具体如下：

```
    <div class = "top">
        <img src = "../static/LOGO.png/" href="/" alt="电院" width=800>
    <h1 style="text-align: center;">Search Engine for SJTU EE208</h1>
    <form action="showbio">
        <div class="search">
        <input type="text" name="search"><button>搜索</button><br>
        </div>
    </form>
    </div>
```

其中LOGO.png 为界面美化部分的内容。

而对于搜索页，我们采取了与之类似的方式，也是标题 + 搜索框。我们添加了展示搜索内容的部分。对于搜索内容我们直接导入已经写好的search.html 即可，对应代码如下：

```
<h1>Search Engine for SJTU EE208</h1>
<hr>
<h2>搜索:{{ search }}</h2>
    <hr>
    <form action="showbio">
        <div class="search">
            <input type="text" name="search"><button>搜索</button><br>
            </div>
    {% include 'result.html' %}
```

3.5 界面的美化

本次实验并未要求界面美化，但笔者考虑到之后的大作业需求，笔者仍对界面进行了一定程度的美化。

我们首先考虑背景的设置，我们希望背景稍微好看一点。笔者采用了使用渐变色的方式（这是我假期在招生组打工时参考某招考网站的背景改的，原网址现已不存在）。这一部分需要在前面写好 CSS 头文件来设置网页的具体格式，同时需要在 body 部分设置网页背景以适配。

同时，我们也对搜索框进行了改良，搜索框的具体源代码来自：https://www.cnblogs.com/smile-xin/p/11390319.html。

我们在网页还加入了尾注和标题图片。尾注需要提前设置格式以保证美观。

原 CSS 文件过长，此处不做展示。

4 结果展示

5 注意

本次实验采用 8080 接口，可能无法成功加载。如 8080 接口加载失败，可能说明接口被转发至 8081 接口，可尝试访问 8081 接口。

6 问题讨论与拓展思考

6.1 JVM 问题

我们知道，初始代码出现了 JVM 的相关问题，这是由于第一次运行 SearchFiles 时 JVM 已经启动，再次启动则会引发 bug。我们采用的形式按照 ppt 所言，如果未启动 JVM 虚拟机则启动虚拟机，而启动虚拟机后就直接提取虚拟机的信息，运行另一个线程即可。

6.2 关于图片的插入

本次实验为了界面美化，提取了交大和电院的 LOGO 作为搜索引擎的标题。但是笔者一开始插入图片时，图片无法成功加载。这是因为，HTML 文件夹是动态的，这样会导致其无法正常加载。因此我们需要在根文件新建一个文件夹来储存...

6.3 端口的转发问题

我们发现 8080 端口被自动转发到 8081 端口。

通过查询相关资料，我们知道，8080 端口一般是连接代理服务器的端口，而 8081 端口为虚拟访问端口。这也就是为什么访问 8080 时无效而访问 8081 时有效。