第七周实验报告

1. 实验概览

本实验旨在于学习使用 flask 网页代码框架，学习 flask 的代码实现方法，理解 flask 模板的原理和使用方法，学习利用过滤器实现模板内的逻辑，学习实现创建 html 表单内容；基于 flask 实现一个简单的搜索引擎系统。

2. 实验环境

Docker: SJTU-EE208

3. 解决思路

利用之前实验中的多线程代码爬取 1w 条网页（以 sports.163.com 做根节点）

前端方面：

根据提供的示例修改出简单的提交表单 index.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <h1>Search</h1>
    <form action="result">
        <label>Keyword</label>
        <input type="text" name="keyword"><br>
        <input type="submit" name="Search">
    </form>


</html>
```

在用户提交网页信息之后，在 result.html 文件中基于过滤器实现 50 条搜索结果的显示

```
<!DOCTYPE html>
<html>
<head>
    <title>Search</title>
</head>
<body>
    <h1>Search</h1>
    <hr>
    <form action="result">
        <label>Keyword</label>
        <input type="text" name="keyword"><br>
        <input type="submit" name="Search">

    </form>
    <h1>Search for: {{ keyword }}</h1>
    <hr>
    {% for i in range(0,length)%}
    <div>
        <a href = {{urls[i]}}><strong><big><p>{{title[i]}}</p></big></strong></
        a>

        <p>{{con[i]}}</p>

        <p>{{urls[i]}}</p>

        <br/>
        <br/>
    </div>
    {%endfor%}

</body>
</html>
```

利用 flask 实现提交表单和显示表单的部分：

```
app = Flask(__name__)
@app.route('/', methods=['POST', 'GET'])
def bio_data_form():
    if request.method == "POST":
        keyword = request.form['keyword']
        return redirect(url_for('result', keyword=keyword))
    return render_template("bio_form.html")
```

```python
@app.route('/result', methods=['GET'])
def result():
    global titles, urls, con, last_search
    STORE_DIR = "index"
    vm.attachCurrentThread()

    directory = SimpleFSDirectory(File(STORE_DIR).toPath())
    searcher = IndexSearcher(DirectoryReader.open(directory))
    analyzer = SimpleAnalyzer()

    keyword = request.args.get('keyword')
    if keyword == '':
        keyword = last_search
    elif keyword != '':
        last_search = keyword
    keyword = ' '.join(jieba.cut(keyword))

    runs(searcher, analyzer, keyword)
    length = min(len(titles),len(urls))
    length = min(length,len(con))
    del searcher
    return render_template("result.html", keyword=keyword, length=length,
    urls=urls, title=titles, con=con)
```

（其中 runs 函数包装了后端检索过程，具体内容详见后文）

后端方面：

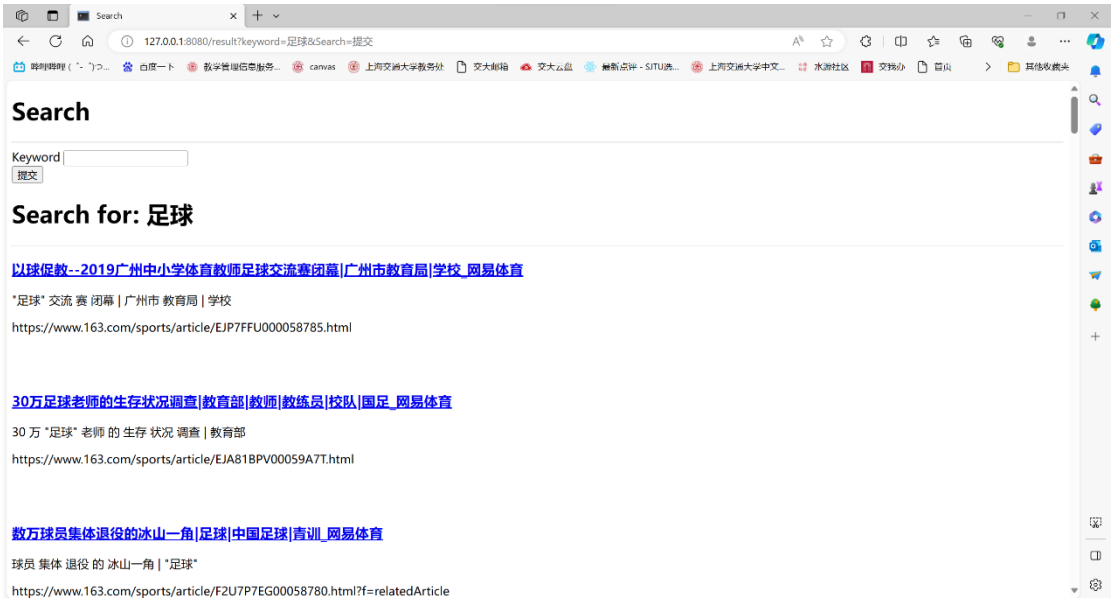
基本基于 lab5-lab6 的实验结果，在原有的代码部分中引入 Luence 库的 Highlighter 类实现关键词显示

```python
formatter = SimpleHTMLFormatter('"','"')
highlighter =  Highlighter(formatter, QueryScorer(query))
highlighter.setTextFragmenter(SimpleFragmenter(25))
tmp = analyzer.tokenStream("contents",contents)
substring = highlighter.getBestFragment(tmp, contents)
```

4. 实验结果

运行 app.py，出现界面：

在搜索框提交文字后：



5. 分析与思考：何为 web3.0

Web 3.0 将重点关注去中心化应用程序，并将大量使用基于区块链的技术。Web 3.0 还将利用机器学习和人工智能（AI）来实现更智能和适应性更强的应用程序，同时这使得语义 web 的概念是 web 3.0 不断发展的定义中的另一个组成部分。

从 Web 1.0 转变为 Web 2.0 花了十多年的时间，预计用 Web 3.0 完全部署和改造 Web 需要同样长的时间。2020 年 Twitter 上的一篇帖子说得最好：Web1 是只读，Web2 是读-写，Web3 将是读-写-自有。

从核心特点来说，web3.0 具有以下特点：

- Web3 是去中心化的：与其说互联网的大部分由中央集权公司管理和拥有，不如说所有权在其架构师和用户之间划分。
- Web3 是无权限的：每个人都可以平等访问 Web3，没有人被排除在外。
- Web3 提供本地支付：它使用加密货币在线消费和汇款，而不是银行和支付处理器的过时基础设施。
- Web3 是不信任的：它不依赖可信的第三方，而是通过激励和经济体系运行。