南京大学

计算机科学与技术系

软件工程实验报告

实验名称：软件设计与实现

学　　号：161220135

姓　　名：吴德亚

指导教师：张天

实验地点：基础实验楼乙 208

实验时间：2019/11/7

一、 实验名称

软件设计与实现实验

二、 实验目的

1. 熟悉 Spring Web 开发框架，了解框架的开发特性并尝试进行 Web 开发
2. 针对工业 App 分级系统根据分配到的功能进行初步实现
3. 完成相应的实验报告

三、 实验要求

1. 配置实验环境，并了解 Spring Web 开发框架的使用；
2. 实现“工业 App 分级系统”的相应功能。
3. 完成实验报告。

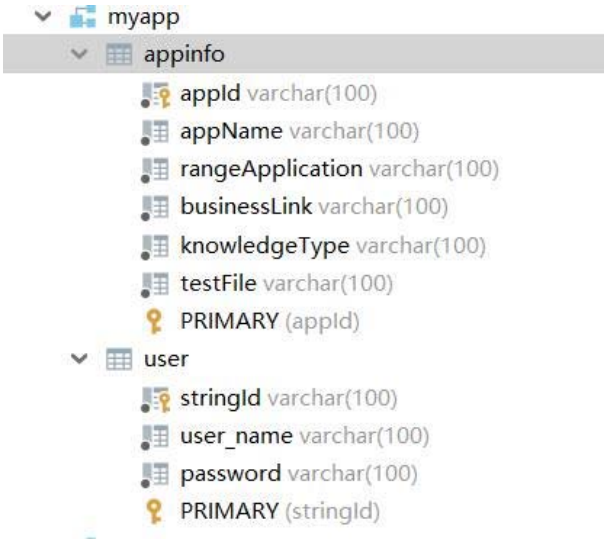
四、 实验环境

1. 软件：JDK 1.8 , Java IDEA, Mysql, Spring(Boot, MVC), Thymeleaf, MyBatis
2. 硬件：笔记本
3. 项目名称：spring-web

五、 实验内容

1. 项目环境搭建

1. 在助教框架代码的基础上进行完善。
2. 新建数据库 myapp，并创建用户表（user）, APP 表（appinfo）并设置关键字。

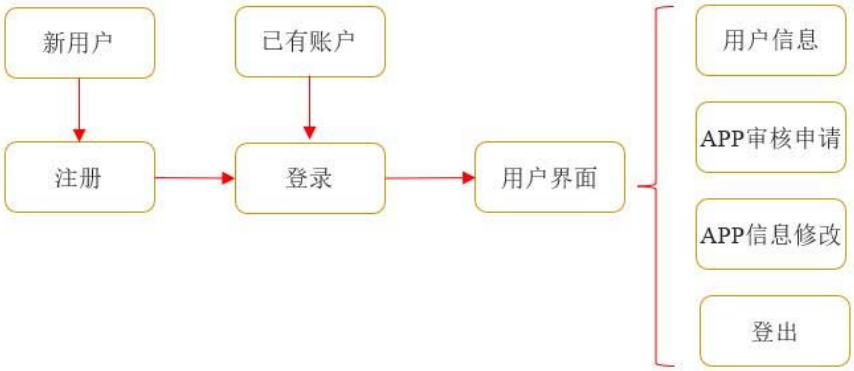


User 表中，从上到下为：用户 ID，用户名，密码，其中用户 ID 为关键字。

APPInfo 表中，从上到下为：APP id，APP 名称，适用范围，业务环节，知识类型，测评文件（只给出一项），其中 APP id 为关键字。

2. 实验效果图

a) 总体逻辑图



1. 如果是未注册用户，则首先需要注册，并进入到登录界面。
2. 登录成功后，有四个选项：用户信息，APP 审核申请，APP 信息修改，登出。
3. 在用户信息中可以查看，修改，注销用户信息。
4. 在 APP 审核申请中，可以填写待审核 APP 具体信息并点击提交。
5. 在 APP 信息修改中，可以查看，修改，删除 APP 信息。
6. 点击登出，回退到登录界面。

b) 帐户管理(登录，注册，登出)

1. 首先是登录界面，需要给出用户 ID 和密码，可以选择新注册和登录。

2. 注册时，需要给出用户 ID，用户名，密码选项，之后跳转到登录界面。

3. 用户界面给出功能：用户信息，APP 审核申请，APP 信息修改，登出。

c) 用户信息修改（查看，修改，注销）

1. 查看用户具体信息

2. 由于用户 ID 唯一确定，所以不能重复也不允许修改（只读）。修改用户名，密码后点击确认即可。

3. 点击注销账户后，则返回登录界面（显示用户不存在）。

d) 企业 APP 分级申请（信息填写，信息修改，删除）

1. 填写待审核 APP 的具体信息，点击提交后即加入数据库中。

2. 点击修改 APP 审核信息即可进行修改。（可以进行删除）

3. 修改完成后显示

六、 实验结果与说明

1. 账户登录，注册，注销，修改等操作。

1) 前端页面截图

2) 后端与数据库交互的表截图为：

```
@Update("update user set user_name = #{name}, password = #{password} where StringId = #{id}")
void updateByID(HelloUser helloUser);//UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值

@Delete("delete from user where StringId = #{id}")
void deleteByID(String id);//DELETE FROM 表名称 WHERE 列名称 = 值
```

通过用户 ID 进行查找、修改、删除等操作。

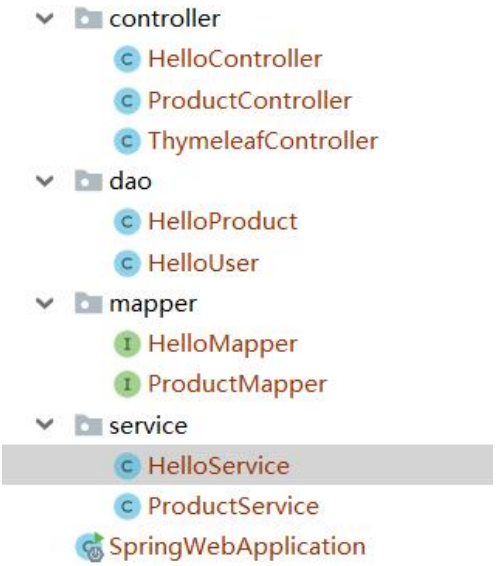
3) 以更新用户数据为例（Update）

```
public void UpdateByID(Map<String, String> params){
    String id = params.get("id");
    HelloUser temp = helloMapper.getOne(id);
    if(params.get("name")!=null)
        temp.setName(params.get("name"));
    if(params.get("password")!=null)
        temp.setPassword((params.get("password")));
    helloMapper.updateByID(temp);
}
```

```
@RequestMapping(value = "/modifyUserInfo", method = RequestMethod.POST) // 修改用户信息
public String ModifyUserInfo(HelloUser user, Model model) {
    Map<String, String> para = new HashMap<>();
    para.put("id", user.getId());
    para.put("name", user.getName());
    para.put("password", user.getPassword());
    helloService.UpdateByID(para);
    return "redirect:modifyUserInfo";
}
```

在之前的页面跳转到 modifyUserInfo.html 后，提交表单被该函数捕获。执行的 hello.service.UpdateByID 函数调用 helloMapper.updateByIdD 函数，而后者的具体实现为@Update 数据库操作。从而完成用户信息的更新。

4）代码框架为：首先是从 SpringWebApplication 开始执行，接收外部的页面请求，在 controller 文件夹下设置对应的请求拦截（POS, GET 等）。实现过程需要使用 dao 文件夹下的对象接收页面数据，之后调用 service 文件夹下的对象执行数据库操作。其中 service 对象的操作是对 Mapper 数据库操作的封装。



2. APP 申请、修改、删除、查看

1) 页面截图

2) 后端与数据库的交互：（部分）包括插入，查找指定 app

```
@Insert("insert into appinfo(appId, appName, rangeApplication, businessLink, knowledgeType," +
        " testFile) values (#{appId}, #{appName}, #{rangeApplication}, " +
        "#{businessLink}, #{knowledgeType}, #{testFile})")
void insert(HelloProduct helloProduct);

@Select("select * from appinfo where appId = #{appId}")
@Results({
        @Result(property = "appId", column = "appId"),
        @Result(property = "appName", column = "appName"),
        @Result(property = "rangeApplication", column = "rangeApplication"),
        @Result(property = "businessLink", column = "businessLink"),
        @Result(property = "knowledgeType", column = "knowledgeType"),
        @Result(property = "testFile", column = "testFile")
})
HelloProduct getOne(String id);
```

定义了 HelloProduct 类，用来得到从页面返回的数据，与数据库表项一一对应。

```
public class HelloProduct implements Serializable {
    private String appId;
    private String appName;
    private String rangeApplication;
    private String businessLink;
    private String knowledgeType;
    private String testFile;
```

3) 当用户申请 APP 信息填写页面时，由 appHello 函数负责处理，返回页面

```
// 产品信息提交界面
@RequestMapping(value = "/appsubmit", method = RequestMethod.GET) // 获取用户输入数据
public String appHello(Model model){
    return "appsubmit";
}
```

当用户填写完成时，提交的表单由 receiveAppinfo 函数负责处理。

```java
// 接收产品提交信息，存入数据库
@RequestMapping(value = "/appsubmit", method = RequestMethod.POST) // 获取用户输入数据
public String receiveAppinfo(HelloProduct helloProduct, Model model){
    String appId = helloProduct.getAppId();
    String appName = helloProduct.getAppName();
    String rangeApplication = helloProduct.getRangeApplication();
    String businessLink = helloProduct.getBusinessLink();
    String knowledgeType = helloProduct.getKnowledgeType();
    String testFile = helloProduct.getTestFile();

    HelloProduct temp = productService.getOne(appId);
    if(temp.getAppId() == null) {    // ID可以使用
        Map<String, String> para = new HashMap<>();
        para.put("appId", appId);
        para.put("appName", appName);
        para.put("rangeApplication", rangeApplication);
        para.put("businessLink", businessLink);
        para.put("knowledgeType", knowledgeType);
        para.put("testFile", testFile);
        productService.InsertProduct(para);
        model.addAttribute( s: "alertInfo",  o: "APP审核信息提交成功！");

        currentproduct = new String(appId);

        return "redirect:modifyProductInfo";
    }
```

由 helloProduct 获取表单信息，之后调用 productService 对象的查找功能，判断是否之前已经存在该 appid，若不出在，则将该信息组成 Map 结构，送到 InsertProduct 函数处理，之后跳转到产品信息查看页面。其中，productService 对象中的函数封装了 ProductMapper 对象的函数操作。

3. 实验补充

1. 由于实验中 html 页面使用了下拉框选项，而在用户更新完 app 信息后又需要动态更新到页面中，实现这个功能花费了不少时间。最后的解决办法是：通过 javascript 函数（必须声明为 inline="javascript"，且该方法只能在 html 内嵌代码有用，外部引用的无效）。
2. 在 body 中设置 onload 即页面加载时自动执行，getModel 函数中得到 controller 中对 model 的动态修改值，之后调用函数 selectID 函数，就可以对相应的 select 操作，找出当前为 selected 的一项即可。

```
<script  th:inline="javascript" type="text/javascript" >
    function selectID(optionName, para) { // 下拉框ID,传入参数

        var all_options = document.getElementById(optionName);
        for(i = 0;i < all_options.length;i++) {
            if(all_options.options[i].value == para) {
                all_options[i].selected = true;
                console.log("out" + para);
            }
        }
    }
    function getModel() {
        var rangeApp = [[${html_rangeApplication}]];
        var bussLink = [[${html_businessLink}]];
        var knowType = [[${html_knowledgeType}]];
        console.log(rangeApp + " " + bussLink + " " + knowType);
        selectID("rangeselect", rangeApp);// 下拉框ID, model中修改的参数
        selectID("businessLinkselect", bussLink);
        selectID("knowledgeselect", knowType);
    }
</script>
<body onload='getModel()'>
```

七、 结论

记录完整实验时长、撰写报告时长；以及其他实验感想与建议。

1. 首先是框架代码的理解，spring (boot、mvc)的基本使用，助教给的框架代码有助于理解。
2. 本次实验中，页面只是简单地渲染了一下，而且关于用户 id 和产品 id 并没有关联起来，即一个用户只能在登录状态下申请一个 APP 审核且登出后无法识别该用户申请过哪些 APP，也是本次实验的一个遗憾。
3. 实现了一些用户的登录，注册等其他功能，之前只是做过一些静态网页（通过 js 判断），没有与数据库交互，所以本次实验还是学到了很多东西。
4. 多谢助教关于本次实验问题的解答。

实验评分： _____

指导教师签字： _____

年　　月　　日