

摘 要:

科技是第一生产力。从我们的祖先猿人第一次用木头擦出火花,到能够发射火箭探索宇宙,科学技术水平的每一次提升,都或多或少的改善人们的生活水平和生活方式,而智能手机的出现,可以说是创造了一个时代。

过去的几十年里,人们的通信基本都靠功能机、固定电话、电话亭,甚至是使用书信联络,但是现在,几乎人手一部智能手机,手机的使用已经深入到人们生活的方方面面。但事实上,智能手机就是一个集合了百家之长的硬件设备,如果没有软件的支持,它就是一块精致的板砖罢了,它需要软件的支持,不管是系统软件,还是 APP 小程序,人们需要有软件才能与手机交互,智能手机才能称得上智能,它才能帮助人们工作、学习、娱乐。那么一款好用且高效的 APP 就显得很重要了。

我使用 SSM 框架开发的这套考勤管理系统就包含了一款手机 APP,这套系统,旨在帮助学生和教师考勤,改善学生的考勤体验,提升考勤效率。

关键词: 智能手机 SSM 框架 考勤管理系统 APP

Abstract

Science and technology are primary productive forces. From the first time our ancestor ape made sparks with wood, to the ability to launch rockets to explore the universe, every improvement in the level of science and technology has more or less improved people's living standards and lifestyles, and the emergence of smart phones, It can be said that it created an era.

In the past few decades, people's communications have basically relied on feature phones, landlines, telephone booths, and even contact by correspondence. But now, almost every person has a smartphone, and the use of mobile phones has penetrated into all aspects of people's lives. But in fact, a smart phone is a hardware device that gathers hundreds of people. If there is no software support, it is an exquisite brick. It needs software support. Whether it is system software or APP APPllets, people Software is required to interact with mobile phones, and smartphones can be called intelligent, and it can help people work, study, and entertain. So a useful and efficient APP is very important.

The attendance management system I developed using the SSM framework includes a mobile phone APP. This system is designed to help students and teachers attendance and improve attendance efficiency.

Keywords Smartphone SSM Framework Attendance Management System APP

目 录

1 前言	1
2 需求调查和需求分析	2
2.1 需求调查	2
2.2 需求分析	2
2.3 登录/注册/退出	2
2.3.1 教师/学生查询课表.....	2
2.3.2 教师发签到任务.....	2
2.3.3 教师查询看学生签到情况.....	2
2.3.4 教师发通知/作业.....	3
2.3.5 教师查询看自己发的通知.....	3
2.3.6 教师查询看学生的请假情况.....	3
2.3.7 学生签到.....	3
2.3.8 学生请假.....	3
2.3.9 学生查询看自己的请假记录.....	3
2.3.10 学生查询看教师发来的通知.....	3
3 系统模型设计和方案的选择	4
3.1 系统模型设计	4
3.1.1 基础概念模型.....	4
3.1.2 用户操作模型.....	4
3.1.3 服务器模型.....	4
3.1.4 数据库模型.....	4
3.2 方案选择	5
3.2.1 开发方案.....	5
3.2.2 开发环境.....	5

3.2.3 开发导入的部分 jar 包.....	5
4 总体设计	6
4.1 数据库模块	6
4.1.1 用户表.....	6
4.1.2 课表.....	7
4.1.3 通知表.....	8
4.1.4 教师表.....	9
4.1.5 教师发签到任务表.....	10
4.1.6 学生签到表.....	10
4.1.7 学生请假表.....	11
4.2 SSM 框架服务器模块	11
4.2.1 表现层.....	12
4.2.2 业务层.....	12
4.2.3 持久层.....	13
4.2.4 实体类.....	14
4.2.5 拦截器.....	14
4.2.6 定时任务.....	15
4.2.7 配置文件.....	15
4.3 网站模块	16
4.3.1 课表.....	17
4.3.2 服务.....	17
4.3.3 查询通知.....	18
4.3.4 账号.....	19
4.4 安卓应用模块	20
4.4.1 欢迎页.....	20
4.4.2 功能页.....	21
5 详细设计	22
5.1 提要	22
5.1.1 表现层的 View 层.....	22

5.1.2 表现层的控制层.....	22
5.1.3 表现层的实体类.....	23
5.1.4 业务层的接口类和接口实现类.....	23
5.1.5 持久层的 dao 接口类.....	23
5.2 SSM 框架服务器搭建.....	23
5.2.1 applicationContext.xml.....	23
5.2.2 Spring-mvc.xml.....	25
5.2.3 web.xml.....	25
5.2.4 database.properties.....	26
5.2.5 log4j.properties.....	27
5.3 登录功能实现	28
5.4 注册功能实现	29
5.5 退出功能实现	30
5.6 用户查询基本信息功能实现	31
5.7 用户查询课表功能实现	32
5.8 教师发布签到任务	32
5.9 教师查询看学生签到状态的实现	37
5.10 教师发布通知实现	38
5.11 教师查询自己发布的通知实现	39
5.12 教师查询学生请假记录实现	40
5.13 学生提出请假功能实现	42
5.14 学生查询自己请假记录的功能实现	44
5.15 学生签到功能实现	46
5.16 学生查询通知功能实现	49
5.17 服务器定时任务功能实现	50
5.18 服务器拦截器功能实现	50
5.19 APP 端的功能实现.....	51
5.19.1 欢迎页.....	51
5.19.2 功能页.....	53
5.20 网站的功能实现	54

6 成品展示	55
6.1 APP 欢迎页	55
6.2 登录	56
6.3 注册	59
6.4 退出	61
6.5 用户查询基本信息	61
6.6 教师发布签到任务	62
6.7 教师查询学生签到状态	65
6.8 教师发布通知	66
6.9 教师查询自己发布的通知	68
6.10 教师查询学生请假记录	69
6.11 学生提出请假	70
6.12 学生查询自己的请假记录	73
6.13 学生签到	74
6.14 学生查询通知	76
6.15 服务器定时任务	78
6.16 服务器拦截器	78
总 结	79
致 谢	80
参考文献:	81

1 前言

随着时代的进步，科技的发展，手机逐渐成为人们生活中不可或缺的科技产品之一，下至还没有上幼儿园的小孩子，上到两鬓斑白的老人，都会对手机有各种各样的需求，他们或是用于休闲娱乐，或是用来学习，或是用于考勤。

一直以来，大学生上课都要接受考勤。从以前的用纸笔签字，到现在用 APP 点一下就可以签到，毫无疑问，学生的考勤方式越来越方便，也越来越节省时间。时间宝贵，一款好用且高效的考勤系统，无疑会受到学生和老师的欢迎。所以，和传统的考勤方式相比，一款富有多种功能且使用方便的考勤管理系统具有良好的市场和应用前景。

一款好用的学生考勤 APP，是大学生生活中不可缺少的一部分。几乎每名大学生手机中都至少有一个能够用于考勤的 APP，但是并不是所有的 APP 都很好用，所以学生或许会很讨厌用那些 APP 用于考勤，甚至想要卸载它，所以一款好的考勤 APP 或许能改变学生对其的看法，方便师生的使用，提高考勤的效率。

不同手机平台的手机，所支持的考勤 APP 也是各不相同，而现在最流行的手机平台就是 google 的 android 手机平台，所以本课题就致力于开发出一款基于 android 平台的迎合大学生使用习惯的，方便好用的，功能多样的考勤管理系统。

2 需求调查和需求分析

2.1 需求查询

现在的大学生的手机中都有很多的校园 APP，比如我在大学一年级时用过的“觅动校园”，每天早上沿着这个 APP 指定的路线走路锻炼，比如经常用来看课件的“超星学习通”，比如下了但从来没用过的“PU”等。其中有的 APP 是真的难用，不只有我一个人这么想，因为我周围的很多人这这么说，但是为了能够通过一些课程，我们不得不接受那些难用到令人抓狂的 APP，一个好用的 APP 绝对能收获无数学生的青睐。

依据我周围同学的反馈，再结合我学习生活中的经验以及我对考勤的需求，我开发了一套考勤管理系统，用户直接使用其中的考勤管理系统 APP 或网页。

2.2 需求分析

作为一款主打考勤的系统，用户与之交互的主要途径是 APP 或网页，想要使用户能流畅且高效的操作，就需要该套系统的 APP 和网页有如下功能。

2.3 登录/注册/退出

登录、注册、退出，这三个功能是用于需要的最基础的功能，有了这三个功能，每名用户才能称得上是一名独立的用户，自一名用户注册之后，他的信息和操作都将保存在数据库中以备后续操作时作为操作的依据。

2.3.1 教师/学生查询课表

教师和学生都要能查询到自己的课表，以为下一次上课做好准备或者做出学习计划。

2.3.2 教师发签到任务

考勤的重要功能之一，教师要能够向指定班级的学生发出签到任务，一次发布即可，不用口头一个一个念名字，节省上课时间，提升考勤效率。

2.3.3 教师查询看学生签到情况

考勤的重要功能，教师在向指定的班级发出签到任务后，要能看到学生的签到情况，以便为期末考核提供证据。

2.3.4 教师发通知/作业

在教师的教学中，有时会需要向学生发布一些通知，或者布置作业，在这个功能下，教师可以向指定班级发送通知/作业。

2.3.5 教师查询看自己发的通知

教师发布通知之后，当然要有个能查询自己发布的通知的功能。

2.3.6 教师查询看学生的请假情况

学生在校的学习生活中，难免会出现生病或有事不能按时参与上课的情况需要向教师汇报，同时存在有时教师是不能接到学生的请假电话的，而且在得知消息后不做记录的话存在遗忘的可能，这就需要有个功能能够记录学生的请假情况，在学生未能按时签到的情况下，教师可以查询看学生的请假记录，以判断学生是否故意缺课。

2.3.7 学生签到

作为考勤的重要功能之一，学生要能够教师的发布的签到任务，也就是进行签到操作，以证明自己确实按时上课。

2.3.8 学生请假

学生在校的学习生活中，难免会出现生病或有事不能按时参与上课的情况需要向教师汇报，同时存在有时教师是不能接到学生的请假电话的情况，所以学生可以在考勤管理系统的 APP 或者网页上向教师提出请假。

2.3.9 学生查询看自己的请假记录

在学生提出请假后，学生应当能够看到自己的请假记录。

2.3.10 学生查询看教师发来的通知

在教师发布了通知之后，学生应当能够查询看到教师发来的通知，不然的话，教师发布通知将毫无意义，所以应当存在学生查询通知的功能。

3 系统模型设计和方案的选择

3.1 系统模型设计

3.1.1 基础概念模型

为了让用户能够在 APP 端和网页端都能够进行考勤操作，满足功能完整性的同时，降低开发的工作量，方便后期维护，我开发了一个网站，这样用户就可以在网页端实现操作，同时开发一个简单的 APP，能够在 APP 内打开我的网站，并且不会跳转到其他浏览器应用。

由于这套考勤管理系统主要面向手机用户群体，所以网站的各个页面中的各种控件都有放大处理，导航栏置于页面底部，以便用户使用本套系统的 APP 进行操作。用户就可以选择在 APP 上操作，或者在网站操作，同时用户也可使用自己的手机浏览器打开网站进行操作，充分迎合了喜欢 APP 的、喜欢网站的、喜欢浏览器操作的者三种人的喜好，而且简化了开发成本，可谓是一举多得。

3.1.2 用户操作模型

需要填写的操作：用户先在文本框中正确填写数据，点击提交或保存的按钮，数据将上传到服务器。服务器收到用户提交请求后先判断用户操作是否符合其权限级别，若具备权限，继续判断用户提交是否存在空值，若存在空值就让用户跳转到提交失败的页面，若不为空，则根据 APP 或网页传来的数据操作数据库，并让用户跳转到提交成功的页面。

无需填写的操作：用户点击按钮后，对应的请求发送到服务器，服务器判断用户先判断用户操作是否符合其权限级别，若具备权限，则操作数据库，判断用户基本信息，并将数据库数据传给用户，显示在提交成功的页面的对应位置，若不具备权限，则直接将用户界面跳转到失败页面。部分功能不用判断权限，仅检查用户基本信息，再将数据库数据传给用户。

3.1.3 服务器模型

服务器需要能够相应用户操作，能在用户操作后能操作数据库，并将相应数据传回给前端。

3.1.4 数据库模型

使用关系数据库，为服务器提供操作材料。

3.2 方案选择

3.2.1 开发方案

服务器、网页、数据库的操作遵循 SSM (Spring + SpringMVC + MyBatis) 框架，使用 maven 管理开发过程中使用到的 jar 包；使用 c3p0 作为数据库连接池；通过注解简化开发流程。

SSM (Spring+SpringMVC+MyBatis) 框架集由 Spring、MyBatis 两个开源框架整合而成 (SpringMVC 是 Spring 中的部分内容)。常作为数据源较简单的 web 项目的框架。符合本项目轻量级的需求。

因为用户随身携带手机，所以网页样式应尽量满足手机的显示。

APP 使用 WebView, 能够在同一网络下，能够显示网页内容即可。

3.2.2 开发环境

数据库: Mysql 8.0.19

数据库可视化工具: Navicat for MySQL

服务器开发工具: IntelliJ IDEA 2020.1 x64

安卓开发工具: Android Studio 3.3.0.0

java version : 13.0.2

3.2.3 开发导入的部分 jar 包

1. spring 核心依赖: spring-core、spring-beans、spring-context
2. spring dao 依赖: spring-jdbc、spring-tx
3. spring web 依赖: spring-web、spring-webmvc
4. mybatis 和 mybatis 整合进 spring: mybatis、mybatis-spring

4 总体设计

用户的操作流程应当尽量简洁，用户在使用考勤管理系统网站的流程图如图 4-1 所示。

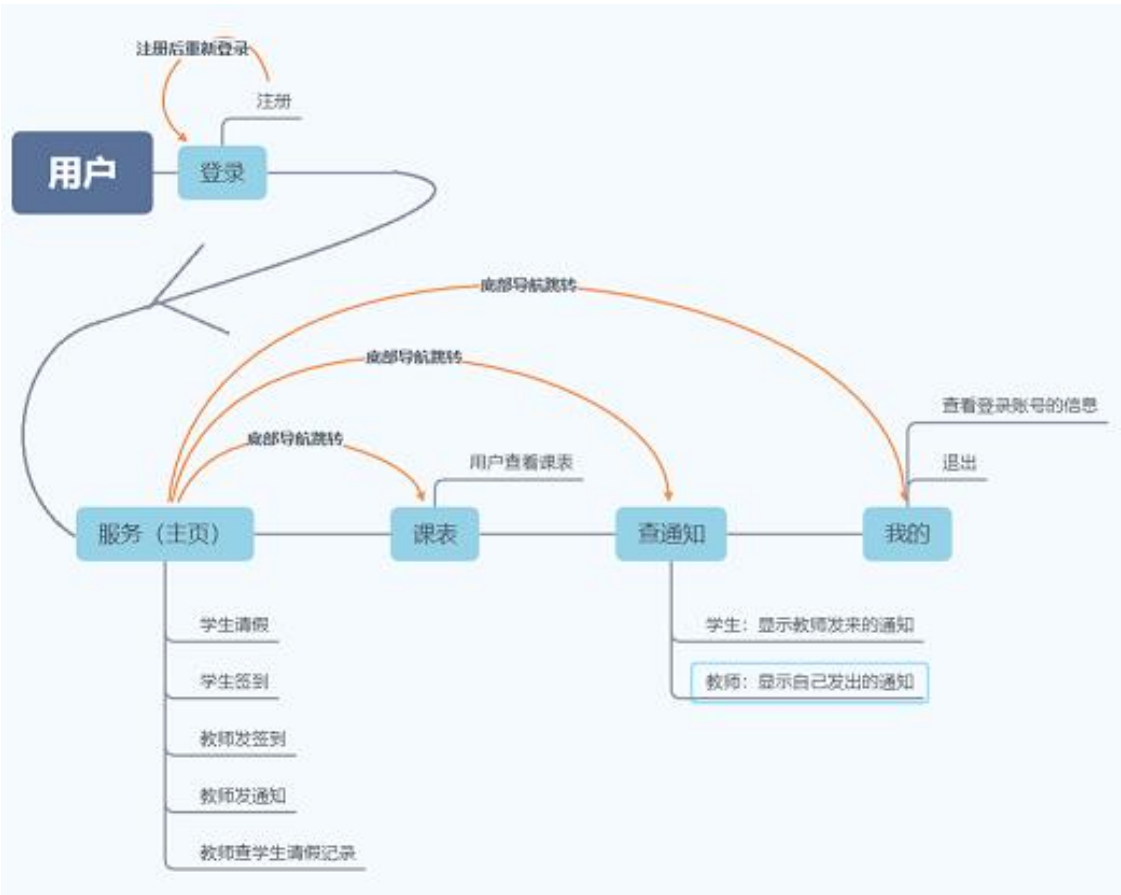


图 4-1 网站使用流程

4.1 数据库模块

考勤管理系统的数据库总体结构如图 4-2 所示。

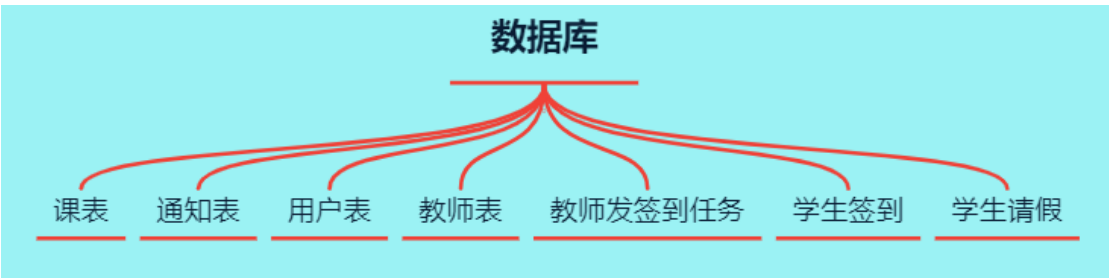


图 4-2 数据库总体结构

4.1.1 用户表

用户表的设计如图 4-3 所示。

名	类型	长度	小数点	不是 null	
id	int	0	0	<input checked="" type="checkbox"/>	 1
username	varchar	255	0	<input checked="" type="checkbox"/>	
password	varchar	255	0	<input checked="" type="checkbox"/>	
level	int	1	0	<input checked="" type="checkbox"/>	
name	varchar	255	0	<input checked="" type="checkbox"/>	
suozaixueyuan	varchar	255	0	<input checked="" type="checkbox"/>	
suozaibanji	varchar	255	0	<input checked="" type="checkbox"/>	
suoshuzhuanye	varchar	255	0	<input checked="" type="checkbox"/>	
qiandaozhuangtai	int	10	0	<input type="checkbox"/>	
kechengmingcheng	varchar	255	0	<input type="checkbox"/>	
teacher	varchar	255	0	<input type="checkbox"/>	
overtime	timestamp	0	0	<input type="checkbox"/>	

图 4-3 数据库用户表设计

表中 username 为用户登录账号；password 为用户登录密码；level 为用户级别，值为 0 时用户为学生，值为 2 时用户为教师，值为 3 时用户为班主任，值为 4 时用户为辅导员，这项用于在服务器中判断用户权限时得到应用；name 为用户真实姓名；suozaixueyuan 为用户所在的学院；suozaibanji 为学生所在的班级；suoshuzhuanye 为用户专业；qiandaozhuangtai 为学生的签到状态，默认为 0，和 kechengmingcheng、teacher、overtime 同为教师发布签到任务时调取学生数据时使用，后三者默认为 null 主键 id 默认自动递增。

4.1.2 课表

课表的设计如图 4-4 所示。


名	类型	长度	不是 null	小数点	
▶ id	int	0	<input checked="" type="checkbox"/>	0	 1
username	varchar	255	<input type="checkbox"/>	0	
name	varchar	255	<input type="checkbox"/>	0	
monday1	varchar	255	<input type="checkbox"/>	0	
monday2	varchar	255	<input type="checkbox"/>	0	
monday3	varchar	255	<input type="checkbox"/>	0	
monday4	varchar	255	<input type="checkbox"/>	0	
monday5	varchar	255	<input type="checkbox"/>	0	
tuesday1	varchar	255	<input type="checkbox"/>	0	
tuesday2	varchar	255	<input type="checkbox"/>	0	
tuesday3	varchar	255	<input type="checkbox"/>	0	
tuesday4	varchar	255	<input type="checkbox"/>	0	
tuesday5	varchar	255	<input type="checkbox"/>	0	
wednesday1	varchar	255	<input type="checkbox"/>	0	
wednesday2	varchar	255	<input type="checkbox"/>	0	
wednesday3	varchar	255	<input type="checkbox"/>	0	
wednesday4	varchar	255	<input type="checkbox"/>	0	
wednesday5	varchar	255	<input type="checkbox"/>	0	
thursday1	varchar	255	<input type="checkbox"/>	0	
thursday2	varchar	255	<input type="checkbox"/>	0	
thursday3	varchar	255	<input type="checkbox"/>	0	
thursday4	varchar	255	<input type="checkbox"/>	0	
thursday5	varchar	255	<input type="checkbox"/>	0	
friday1	varchar	255	<input type="checkbox"/>	0	
friday2	varchar	255	<input type="checkbox"/>	0	
friday3	varchar	255	<input type="checkbox"/>	0	
friday4	varchar	255	<input type="checkbox"/>	0	
friday5	varchar	255	<input type="checkbox"/>	0	

图 4-4 数据库课表设计

课表表用于根据用户名分别存储每名用户每周各个课时端需要上的课的名字，由于大学中几乎 2 节课同为一名教师上同一节课，所以 monday1 指的是星期一上午的 2 节课，后缀 1-4 分别为白天上下午的 2 堂课，5 为 晚上的课，以此类推。主键 id 默认自动递增。

4.1.3 通知表

通知表的设计如图 4-5 所示


名	类型	长度	小数点	不是 null	
id	int	0	0	<input checked="" type="checkbox"/>	 1
title	varchar	255	0	<input checked="" type="checkbox"/>	
zhengwen	varchar	255	0	<input type="checkbox"/>	
starttime	timestamp	6	0	<input type="checkbox"/>	
fabuzhe	varchar	255	0	<input type="checkbox"/>	
fugaibanji	varchar	255	0	<input type="checkbox"/>	

图 4-5 数据库通知表设计

表中 title 为通知标题, zhengwen 为通知正文, starttime 为发出通知的时间, fabuzhe 为发布通知的教师名, fugaibanji 为收到该通知的班级, 班级下的所有学生都将收到该通知。主键 id 默认自动递增。

4.1.4 教师表

教师表的设计如图 4-6 所示

名	类型	长度	小数点	不是 null	
id	int	0	0	<input checked="" type="checkbox"/>	 1
username	varchar	255	0	<input type="checkbox"/>	
name	varchar	255	0	<input type="checkbox"/>	
kechengmingcheng	varchar	255	0	<input type="checkbox"/>	
suozaibanji	varchar	255	0	<input type="checkbox"/>	

图 4-6 数据库教师表设计

表中 username 为教师的登录账号; name 为教师真实姓名; kechengmingcheng 为教师带的课程名称; suozaibanji 为教师所在班级; 主键 id 默认自动递增。

4.1.5 教师发签到任务表

教师发签到任务表的设计如图 4-7 所示


名	类型	长度	小数点	不是 null	
id	int	0	0	<input checked="" type="checkbox"/>	 1
qiandaoma	varchar	50	0	<input type="checkbox"/>	
kechengmingcheng	varchar	255	0	<input type="checkbox"/>	
suozaibanji	varchar	255	0	<input type="checkbox"/>	
starttime	timestamp	6	0	<input type="checkbox"/>	
overtime	timestamp	6	0	<input type="checkbox"/>	
username	varchar	255	0	<input type="checkbox"/>	
name	varchar	255	0	<input type="checkbox"/>	
level	int	0	0	<input type="checkbox"/>	
faqishijian	varchar	255	0	<input type="checkbox"/>	

图 4-7 数据库教师发签到表设计

表中 qiandaoma 为此次新发布签到任务的签到码，学生需提交相同 qiandaoma 才能完成签到；kechengmingcheng 为签到任务发布的课程名称；suozaibanji 为签到任务发布的对象班级；starttime 为签到开始时间；overtime 为签到结束时间；username 为发布者账号；name 为发布者真实姓名；level 为发布者级别；faqishijian 为该签到任务发起时的系统时间；主键 id 默认自动递增。

4.1.6 学生签到表

学生签到表的设计如图 4-8 所示


名	类型	长度	小数点	不是 null	
id	int	0	0	<input checked="" type="checkbox"/>	 1
username	varchar	255	0	<input type="checkbox"/>	
name	varchar	255	0	<input type="checkbox"/>	
qiandaozhuangtai	int	0	0	<input type="checkbox"/>	
kechengmingcheng	varchar	255	0	<input type="checkbox"/>	
suozaibanji	varchar	255	0	<input type="checkbox"/>	
teacher	varchar	255	0	<input type="checkbox"/>	
overtime	timestamp	0	0	<input type="checkbox"/>	
starttime	timestamp	0	0	<input type="checkbox"/>	
qiandaoma	varchar	255	0	<input type="checkbox"/>	
faqishijian	varchar	255	0	<input type="checkbox"/>	

图 4-8 数据库学生签到表设计

表中 username 为学生登录账号；name 为学生真实姓名；qiandaozhuangtai 为

学生签到状态，状态值为 0 则学生未签到，状态值为 1 则学生已经签到，状态值为 2 则该次签到已经作废；kechengmingcheng 为该次签到是为什么课所设；suozaibanji 为该次签到是为哪个班级所设；teacher 为该次签到的发布者的账号名；overtime 为此次签到超时时间；starttime 为此次签到开始时间；qiandaoma 为此次签到任务的密码，学生只有输入正确的签到码才能够完成签到，否则签到失败；faqishijian 为教师发布该签到任务时的系统时间；主键 id 默认自动递增。

4.1.7 学生请假表

学生请假表的设计如下图 4-9 所示

名	类型	长度	小数点	不是 null	
id	int	0	0	<input checked="" type="checkbox"/>	 1
username	varchar	255	0	<input checked="" type="checkbox"/>	
name	varchar	255	0	<input checked="" type="checkbox"/>	
suozaibanji	varchar	255	0	<input checked="" type="checkbox"/>	
suozaixueyuan	varchar	255	0	<input checked="" type="checkbox"/>	
qingjialeixing	varchar	255	0	<input checked="" type="checkbox"/>	
starttime	timestamp	6	0	<input checked="" type="checkbox"/>	
overtime	timestamp	6	0	<input checked="" type="checkbox"/>	
qingjialiyong	varchar	255	0	<input checked="" type="checkbox"/>	

图 4-9 数据库学生请假表设计

表中 username 为学生的账号名称；name 为学生的真实姓名；suozaibanji 为学生在学校所在的班级；suozaixueyuan 为学生在学校所在的学院；qingjialeixing 为学生该次请假的类型，或为事假或为病假等；starttime 为请假离校的开始时间；overtime 为学生应当返校的截止时间；qingjialiyong 为学生该次请假的合理的具理由；主键 id 默认自动递增。

4.2 SSM 框架服务器模块

SSM 框架服务器的基本工作流程：

用户处于表现层的 View 层，在 jsp 页面中操作的请求都会通过 Post 或 Get 方法发送到表现层的控制层，控制层根据用户发送的请求数据调用业务层 Service 的接口，通过接口找到对应的实现类 ServiceImpl，对应的实现类 ServiceImpl 的对应方法再调用持久层 dao 中的方法对数据库进行操作，然后原路 return 回到控制层，再到 View 层反馈给用户。

考勤管理系统的 ssm 框架服务器的结构如图 4-10 所示。

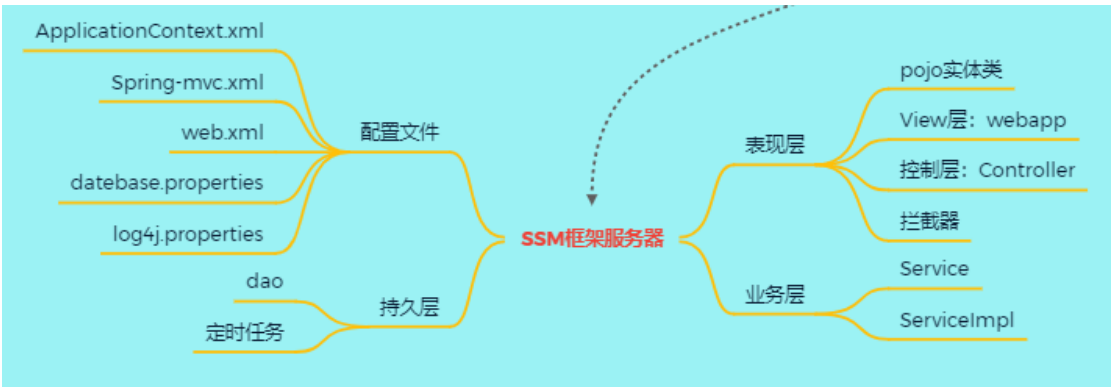


图 4-10 SSM 框架服务器的框架设计

4.2.1 表现层

SSM 框架的表现层由 M（实体类）、V（web 界面）和 C（Controller 控制器）组成，实体类和 View 层的 View 层后面说，先说控制器。

Controller 的目录结构如图 4-11 所示。

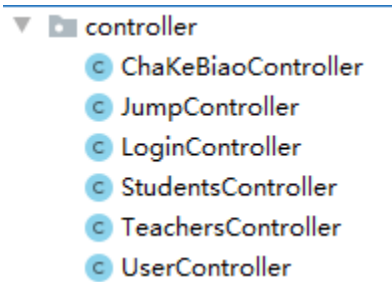


图 4-11 Controller 目录结构

ChaKeBiaoController：用户查询课表时的事务申请发送到这里处理。

JumpController：用户点 View 层按钮时，部分请求发送到这里，JumpController 处理后帮助用户跳转到指定界面。

LoginController：用户的登录、注册功能在这里被处理。

StudentsController：学生的请假、签到等功能在这里被处理。

TeachersController：教师的发通知、发签到任务等功能在这里被处理。

UserController：整合 SSM 框架时测试使用到的控制处理。

4.2.2 业务层

SSM 框架的业务层由 Service 接口类和 ServiceImpl 实现类组成。

业务层的目录结构如图 4-12 所示。

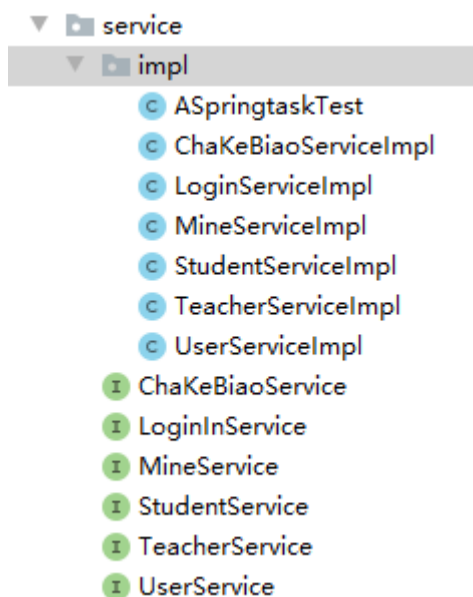


图 4-12 业务层目录结构

以 Service 结尾做文件名的是接口类，对应名称并以 Impl 结尾的是对应接口的实现类，这些实现类会通过继承的方式继承对应名称的接口，并通过对持久层的操作实现其业务层的功能。

ChaKeBiaoServiceImpl 实现查询课表的业务层功能；

LoginServiceImpl 实现用户登录、注册的业务层功能；

MineServiceImpl 实现查询登录的用户信息、退出的业务层功能；

StudentServiceImpl 实现学生签到、请假等业务层功能；

TeacherServiveImpl 实现教师发布签到任务、发布通知的业务层功能；

UserServiceImpl 是 SSM 框架整合时使用的测试类。

ASpringtaskTest 为定时任务类，后面定时任务时会说。

4.2.3 持久层

持久层时数据访问层，其中封装的方法实现对数据库的操作。由于使用了注解的方式把数据库语言写在方法的上面，声明其对应方法为一个对数据库操作的方法，并将其交给 Spring 的 IOC 容器来管，就不需要再在 resource 中创建与 dao 相同目录结构的映射 mapper，由此简化了开发成本，其本质是包含一个个包含数据库操作语言的方法的集合的接口类。

持久层的目录结构如图 4-13 所示：

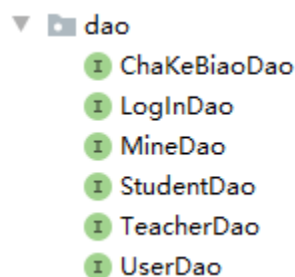


图 4-13 持久层目录结构

ChaKeBiaoDao 包括了用户查询课表时用到的方法；

LoginDao 包括了用户登录、注册时用到的方法；

MineDao 包括了用户查询自己账号基本信息时用到的方法；

StudentDao 包括了学生签到、请假时用到的方法；

TeacherDao 包括了教师发布签到任务、发布通知时用到的方法；

UserDao 包括了测试 SSM 整合时的增删改查询数据库的方法；

4.2.4 实体类

Pojo 是封装实体类的包名，实体类是数据库操作数据库时以及控制层从 View 层拿数据时使用到的工具类，里面定义了 int 类型和 String 类型的属性名，这些属性名和数据库中对应属性表中的属性名相同，以避免 MyBatis 在调用这些属性名时出错；为了在表现层的控制层能够顺利得从 View 层拿到数据，并输出回 View 层，需要为那些属性名提供 set、get、toString 方法。

pojo 实体类的目录结构如图 4-14 所示

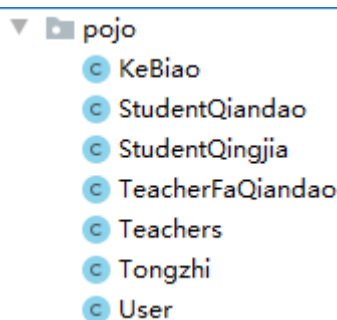


图 4-14 实体类目录结构

实体类 KeBiao、StudentQiandao、StudentQingjia、TeacherFaQiandao、Teachers、Tongzhi、User 内分别是用户查询课表时、学生签到时、学生请假时、教师法签到任务时、教师发的通知、查询教师账号信息时、用户登录注册等时候在 SSM 框架各层使用到的属性；

4.2.5 拦截器

网站中的大部分功能的实现都需要用户登录之后才能后使用，因为考勤管理系统的大部分功能都是对一名用户负责的，所以该套系统就需要一个拦截器，判断用户的操作是否是在登录之后发生的，当然登录操作和注册操作是在用户登录之前发生，所以需要对登录和注册的方法进行过滤。不过在我的系统中功能尚不完善。

拦截器所在的目录如图 4-15 所示

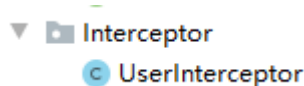


图 4-15 拦截器目录结构

4.2.6 定时任务

在签到任务发布之后，学生应当按时签到，但是会存在有同学不能按时签到的情况发生，就会影响到该生的下一次签到，所以我在系统中加入一个定时任务，服务器启动时、每隔 45 分钟，也就是一节课的时间重置一次学生签到表，把学生的签到状态同一改为 2，意味签到作废。我这样做是因为正常一次签到任务自发布后不会超过 45 分钟，学生应当在上课后不久就应该签到了，教师可以在重置前记下未签到的同学。

定时任务所在的目录如图 4-16 所示

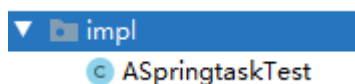


图 4-16 定时任务目录结构

4.2.7 配置文件

整合 SSM 框架服务器需要一些配置文件，配置文件的目录结构如图 4-17 所示。

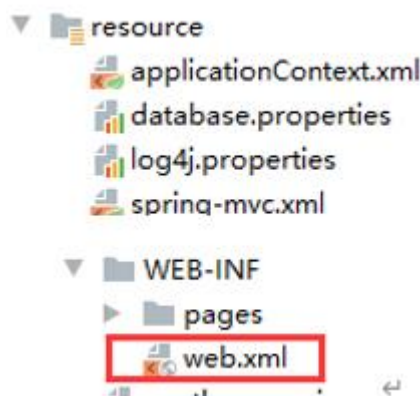


图 4-17 配置文件目录结构

applicationContext.xml:整合 Spring 和 MyBatis;

Spring-mvc.xml 是 SpringMVC 的配置文件;

web.xml 是网页的配置文件;

database.properties 是连数据库时的配置;

log4j.properties 提供了日志输出;

4.3 网站模块

网站是用户直接接触和操作、与服务器和数据库交互的窗口,所以不管用户的操作是否符合常理,网站都应当能够对用户的操作做出相应的反馈,所以我给用户容易操作失误的大部分功能都分别加入了操作成功和失败的页面。页面中的按钮在向服务器发送请求时,使用的大都是 POST 和 GET 方法。

网站由四大部分组成,分别为课表、服务、查询通知、账号,其中由于该网站的操作都需要在用户登录之后才能进行,所以把登录和注册的功能放在了打开网站后出现的第一个页面,要求用户必须先登录才能进行下一步操作,这个页面严格来说并不能算得上是首页,只是为了用户的后续操作不会报出更多错误而这样设置的,同时为了迎合三种不同使用习惯的用户的喜好,网站的控件都比较大,以满足手机用户的使用需求。

网站的页面都存放在目录 webapp 下。

考勤管理系统的网站结构如图 4-18 所示。

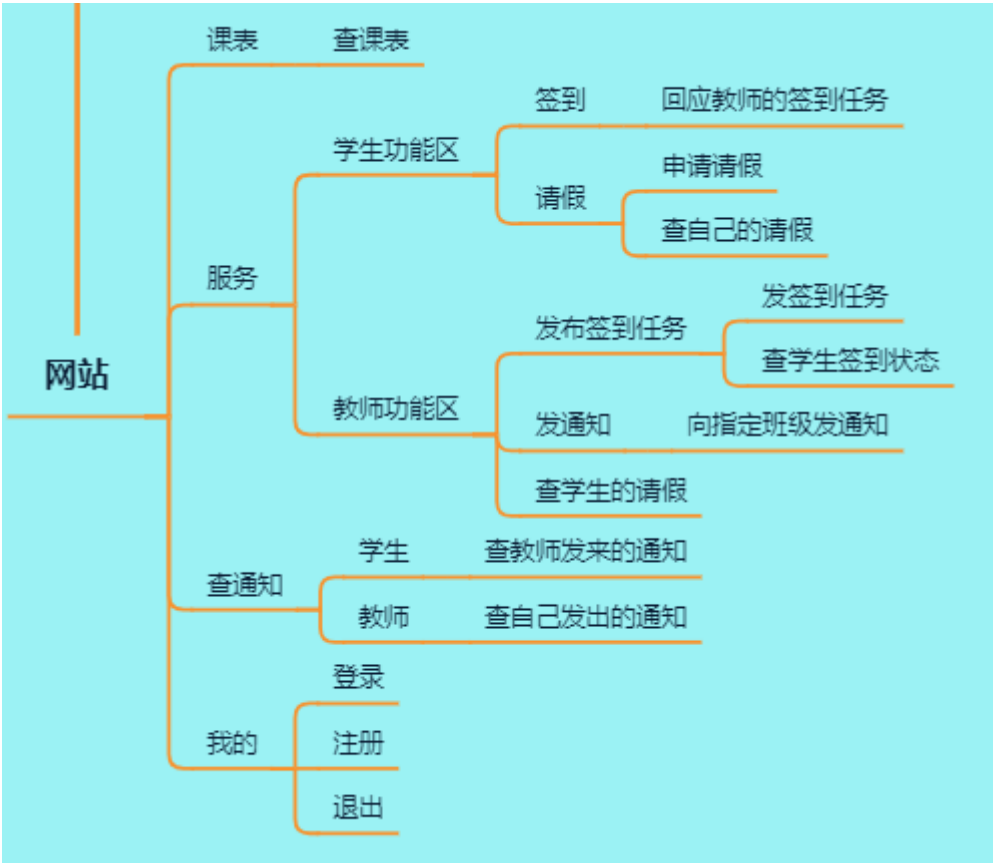


图 4-18 考勤管理系统的网站结构

4.3.1 课表

课表是用户查询自己课表的页面，点击导航栏的菜单按钮后，调用服务器把用户的课表显示在课表页面上。

课表页面的目录结构如图 4-19 所示。

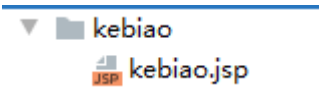


图 4-19 课表页面的目录结构

4.3.2 服务

服务中集合了学生和教师的考勤操作入口，学生可以在这里进行请假、签到操作，教师可以在这里发布签到任务、发布通知、查询看学生的请假情况，为了防止用户操作过程中出现问题，我给大部分功能提供了操作失败和成功的页面，以避免不必要的麻烦，改善用户使用体验。

服务页面的目录结构如图 4-20 所示。

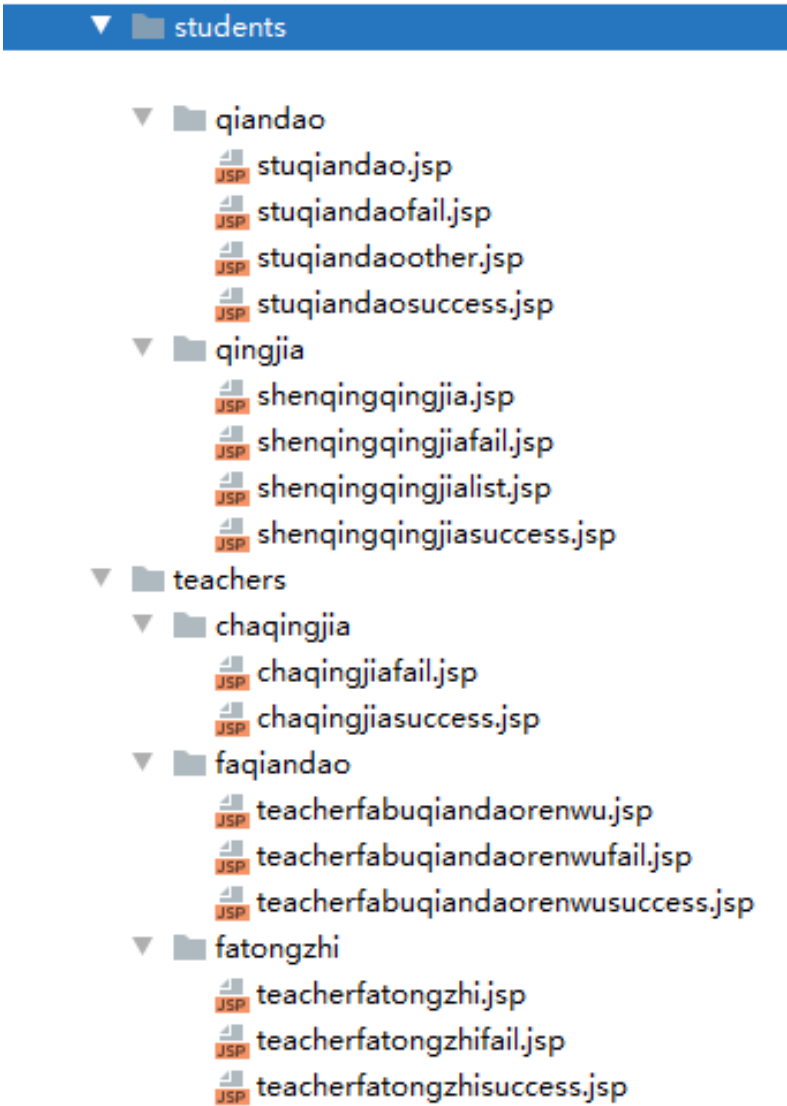


图 4-20 服务页面的目录结构

4.3.3 查询通知

教师可以向指定班级发送通知，也可以布置作业，但更重要的是学生要能查询到这些通知和作业，所以我做了这个页面，学生可以在这个页面查询看教师发来的通知，教师也可以通过这个页面跳转到教师查询看自己发布的通知的页面。

学生查询通知的最终停留页面的目录结构如图 4-21 所示。

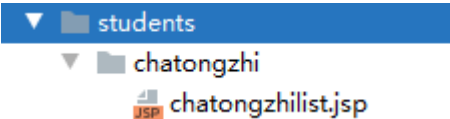


图 4-21 学生查询通知页面的目录结构

教师查询通知的最终停留页面的目录结构如图 4-22 所示。

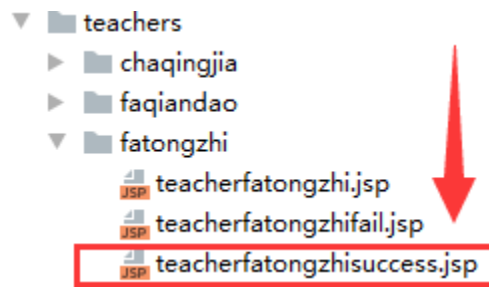


图 4-22 教师查询通知页面的目录结构

4.3.4 账号

考勤管理系统的大部分功能的实现都需要在用户已经登录的前提下才能够实现，因此账号的登录、注册功能在网站打开后的第一个页面出现，退出按钮则放在页面“我的”中，同时页面“我的”中也能显示登录用户的基本信息。

账号功能的目录结构如图 4-23 所示。

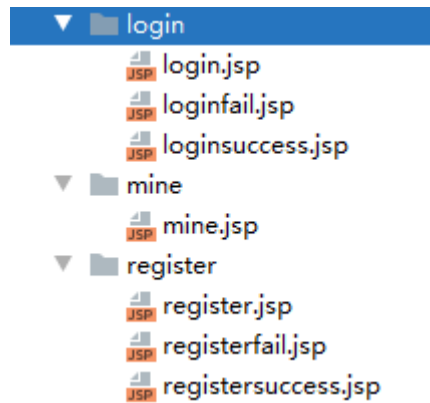


图 4-23 账号功能页面的目录结构

4.4 安卓应用模块

安卓 APP 作为用户在手机上使用考勤管理系统的一个介质，由于只需要能够访问考勤管理系统的网站即可，所以开发起来较为简单，只需要一个欢迎页和一个功能页即可。

安卓用户的操作流程如图 4-24 所示。

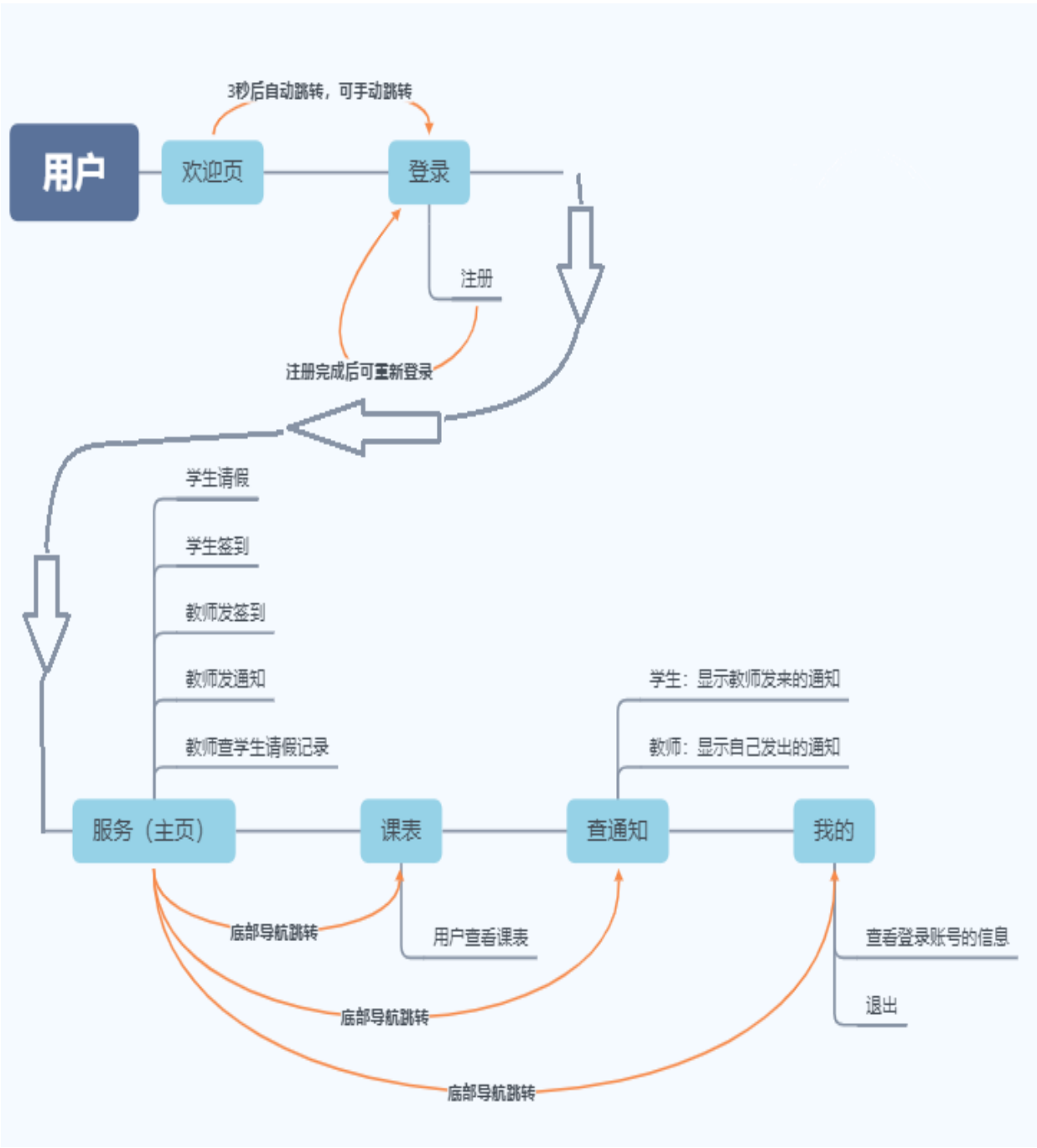


图 4-24 安卓用户的操作流程

4.4.1 欢迎页

安卓应用的欢迎页背景是一张欢迎图片，右下一个按钮可以让用户跳过欢迎页，直接进入登录页面。欢迎页其实是一个定时器任务，三秒钟后自动跳转页面到功能页。

4.4.2 功能页

功能页由控件 WebView 充满，网页需要设计能适应手机宽度，以避免操作过程中页面出现滑动的情况。WebView 设置了网站链接后，只要服务器保持开启，用户从欢迎页跳转到功能页后，就可以看到自动开启的考勤管理系统网站的登录页面，后续操作流程和方法和在网站中一致。

5 详细设计

5.1 提要

数据库已在总体设计中阐述明白，网站只是一堆控件的集合罢了，CSS 样式也简单，因此数据库和网站不做过多说明；程序中重复性说较多，故在此写提要，把一些后文无需多讲的内容提出来先讲。

5.1.1 表现层的 View 层

前台页面中的按钮和文本输入框都放在对应<form>标签中，用 action="" 指向 Controller 包中对应注解@RequestMapping("")下的方法，用 method=""表明声明发送请求的方式，我常用 post 和 get。

需要遍历时导入 jstl 标签库，用<c:forEach>标签，需要设置 items="\${ }"，这是是拿到 Controller 中用 Model 保存的数据的办法；用 var="" 相当于是给 items="\${ }"中的名称起了个别名，方便后面读取 items="\${ }"中 Model 保存的数据，如\${kebiao.monday1}就可以把课表数据库中周一第一节课程的课程信息显示出来，<c:forEach>标签可以嵌套在表格中，就可以实现在表格中遍历显示数据，这些数据能组成一张表。

为了能让网页适应手机屏幕尺寸需要在每张网页加入代码如图 5-1 所示。

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0, minimum-scale=0.5, maximum-scale=2.0,  
user-scalable=yes" />
```

图 5-1 网页适应手机屏幕尺寸代码

用 CSS 写控件样式，控件用 id=""引用样式。自此网页设计基本宣告结束，后文将不再提。

5.1.2 表现层的控制层

JumpController 中是部分界面跳转功能的实现；

需要在类名上写注解@Controller, 声明这是一个表现层 Controller 文件，将其交给 Spring 的 IOC 容器管。也可以加注解@RequestMapping(""), 让网页中的控件指向它，再指向其下的方法，中间用 "/" 隔开，就可以调用方法。

在方法前用@Autowired 自动注入业务层接口，也可注入持久层 dao，就可以在

后文调用业务层或持久层的方法。

我的 HttpSession 保存用户的登录账号 username，在各个控制类中都有使用。

5.1.3 表现层的实体类

在实体类中复写对应数据表中各个栏位的名称和类型，并写出 get、set 和 toString 方法，这样后续服务器各层就可以直接用，且不易出现错误。

5.1.4 业务层的接口类和接口实现类

接口类中就是系统各个功能的接口。

接口实现类继承自接口类，类名上写注解@Service 声明这个类属于业务层，就可以交给 Spring 的 IOC 容器管理。

用@Autowired 自动注入持久层接口，接口实现类中就可以调用指定持久层的 dao 中的方法了。

5.1.5 持久层的 dao 接口类

在方法上通过把具体的对数据库的操作语言写在注解中，在业务层中就可以直接调用其方法，注解例子如图 5-2 所示。

```
@Select("select * from user where username=#{username}")  
public User checkusername(@Param("username")String username);
```

图 5-2 持久层的注解例子

注解@Select 对应的就是 mysql 语言中的 select，而@update 则对应 update，其他的同理。

注解@Param(“”)是声明参数，数据库操作语言中使用参数，使用户对数据库的操作更加灵活。

5.2 SSM 框架服务器搭建

搭建服务器框架当然要先创建一个项目，由于我要做 SSM 框架，所以创建了一个 maven 的 web 项目，并在项目文件中的 POM.xml 写入需要的依赖文件，就可以让 maven 帮我下载需要的 jar 包导入项目，接下来只要整合 SSM 即可，下列配置文件的功能都在总体设计中指出便不再赘述。

5.2.1 applicationContext.xml

spring 开启注解的扫描如图 5-3 所示，但是不扫描 Controller 中的类，因为

那是 springmvc 用的，都扫描会发生冲突。

```
<context:component-scan base-package="cn.wf">
  <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
</context:component-scan>
```

图 5-3 spring 开启注解扫描

Spring 整合 MyBatis 框架，关联数据库配置文件如图 5-4 所示。

```
<context:property-placeholder location="classpath:database.properties"/>
```

图 5-4 关联数据库配置文件

配置 c3po 连接池如图 5-5 所示。

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
  <property name="driverClass" value="com.mysql.cj.jdbc.Driver"/>
  <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/wfkaoqinxitong?useSSL:
  <property name="user" value="root"/>
  <property name="password" value="123456"/>
</bean>
```

图 5-5 配置 c3p0 连接池

配置 SqlSessionFactory 工厂如图 5-6 所示。

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

图 5-6 配置 SqlSessionFactory 工厂

配置 dao 接口所在的包如图 5-7 所示，Spring 会自动查询找其下的类。

```
<bean id="mpperScanner" class="org.mybatis.spring.mapper.MapperScannerConfigurer">
  <property name="basePackage" value="cn.wf.dao"/>
</bean>
```

图 5-7 配置 dao 接口所在的包

配置事务管理器如图 5-8 所示。

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

图 5-8 配置 dao 接口所在的包

配置事务通知如图 5-9 所示。

```
<tx:advice id="txAdvice" transaction-manager="transactionManager">
  <tx:attributes>
    <tx:method name="find*" read-only="true"/>
    <tx:method name="*" isolation="DEFAULT"/>
  </tx:attributes>
</tx:advice>
```

图 5-9 配置事务通知

配置 AOP 增强如图 5-10 所示。

```
<aop:config>
  <aop:advisor advice-ref="txAdvice" pointcut="execution(* cn.wf.service.impl.*ServiceImpl.*(..))"/>
</aop:config>
```

图 5-10 配置 AOP 增强

开启对@Scheduled 注解的支持如图 5-11 所示，定时任务 Springtask 要用。

```
<task:annotation-driven/>
```

图 5-11 开启对@Scheduled 注解的支持

5.2.2 Spring-mvc.xml

开启注解扫描，只扫描 Controller 注解如图 5-12 所示。

```
<context:component-scan base-package="cn.wf">
  <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
</context:component-scan>
```

图 5-12 springmvc 开启注解扫描

配置的视图解析器对象，扫描所有包下以 .jsp 结尾的页面，在 controller 中 return 时无需为其写后缀名如图 5-13 所示。

```
<bean id="internalResourceViewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/" />
  <property name="suffix" value=".jsp" />
</bean>
```

图 5-13 配置视图解析器对象

配置拦截器如图 5-14 所示。

```
<mvc:interceptors>
  <mvc:interceptor>
    <mvc:mapping path="/**" />
    <bean class="cn.wf.Interceptor.UserInterceptor" />
  </mvc:interceptor>
</mvc:interceptors>
```

图 5-14 配置拦截器

开启 SpringMVC 注解的支持同时开启 json 格式的支持如图 5-15 所示。

```
<mvc:annotation-driven/>
```

图 5-15 开启 json 格式的支持

5.2.3 web.xml

设置打开网站后的默认界面为登录/注册页面如图 5-16 所示。

```
<welcome-file-list>
  <welcome-file>/login/login.jsp</welcome-file>
</welcome-file-list>
```

图 5-16 设置默认界面

配置 Spring 的监听器如图 5-17 所示。

```
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

图 5-17 配置 Spring 的监听器

设置 spring 的配置文件的的路径如图 5-18 所示

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:applicationContext.xml</param-value>
</context-param>
```

图 5-18 设置 spring 的配置文件的的路径

配置前端控制器如图 5-19 所示

```
<servlet>
  <servlet-name>dispatcherServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <!-- 加载springmvc.xml配置文件-->
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring-mvc.xml</param-value>
  </init-param>
  <!-- 启动服务器就创建该servlet-->
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcherServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

图 5-19 配置前端控制器

解决中文乱码的过滤器如图 5-20 所示

```
<filter>
  <filter-name>characterEncodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>characterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

图 5-20 解决中文乱码的过滤器

5.2.4 database.properties

为 applicationContext.xml 中配置数据库连接池提供参数如图 5-21 所示。

```
jdbc.driver=com.mysql.cj.jdbc.Driver          过长省略  
jdbc.url=jdbc:mysql://localhost:3306/wfkaoqinxitong?use!  
jdbc.username=root      后面跟的功能是1.安全连接; 2.不乱  
jdbc.password=123456    码; 3.mysql8.0以上要设时区
```

图 5-21 数据库连接池提供参数

5.2.5 log4j.properties

log4j 日志配置如图 5-22 所示

```
log4j.rootCategory=ERROR, CONSOLE  
log4j.logger.com.demo.mapper=DEBUG  
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender  
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout  
log4j.appender.CONSOLE.layout.ConversionPattern=%C %d{YYYY-MM-dd hh:mm:ss} %m %n  
log4j.appender.LOGFILE=org.apache.log4j.FileAppender  
log4j.appender.LOGFILE.File=E:/my.log  
log4j.appender.LOGFILE.Append=true  
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout  
log4j.appender.LOGFILE.layout.ConversionPattern=%m %n  
log4j.logger.org.springframework.scheduling = INFO
```

图 5-22 log4j 日志配置

至此，服务器便搭建成功，如图 5-23 所示。

```
[2020-04-30 10:49:31,097] Artifact WF_KQGL_Server:war exploded: Artifact is deployed successfully  
[2020-04-30 10:49:31,097] Artifact WF_KQGL_Server:war exploded: Deploy took 12,952 milliseconds
```

图 5-23 服务器搭建成功

5.3 登录功能实现

登录控制层代码如图 5-24 所示

```
@RequestMapping(value = "/loginin", method = RequestMethod.POST)
public String loginin(User user, HttpSession session) {
    System.out.println("表现层: 有人想要登录了");
    user = loginInService.checkusername(user.getUsername(), user.getPassword());
    if (user != null) {
        // 重定向
        if (user.getLevel() == 0) {
            System.out.println("学生: " + user.getUsername() + " 登录成功");
        } else if (user.getLevel() == 2) {
            System.out.println("教师: " + user.getUsername() + " 登录成功");
        } else if (user.getLevel() == 3) {
            System.out.println("班主任: " + user.getUsername() + " 登录成功");
        } else if (user.getLevel() == 4) {
            System.out.println("辅导员: " + user.getUsername() + " 登录成功");
        }

        session.setAttribute("loginSuccess", user.getUsername());
        System.out.println("用户信息:" + session.getAttribute("loginSuccess"));
        return "login/loginSuccess";
    }
    System.out.println("登录失败! 账号或密码错误!");
    return "login/loginFail";
}
```

图 5-24 登录控制层代码

登录业务层实现类如图 5-25 所示

```
public User checkusername(String username, String password) {
    System.out.println("业务层: 有人登录, 正在验证账号和密码");
    User user = loginDao.checkusername(username);
    //既然已经有了user, 便只需再确认密码是否正确即可
    if (user != null && user.getPassword().equals(password)) {
        return user;
    }
    return null;
}
```

图 5-25 登录业务层实现类

登录持久层代码如图 5-26 所示

```
@Select("select * from user where username=#{username}")
public User checkusername(@Param("username")String username);
```

图 5-26 登录持久层

用户的请求发来后，调用业务层实现类中的 `checkusername()` 方法，以前端发来的用户名和密码为参数，调用持久层中的 `checkusername()` 方法，找数据库中对应用户名的数据记录，如果没有就返回 `null`，此时 `user` 为 `null`；如果存在记录，再验证前端传来的 `password` 是否和数据库中一致，若 `password` 一致，则通过账号密码验证，返回 `user`，此时 `user` 不为空。

判断 `user` 是否为 `null`，若为 `null`，则数据库中没有与用户输入匹配的账号，View 层跳转到登录失败的页面；反之则为用户登录成功，将用户登录的账号的 `username` 保存在名为“`loginsuccess`”的 `Httpsession` 中，以备后续操作时使用，View 层跳转到服务页面，登录功能完毕。

用实体类中准备好的 `getLevel()` 方法，取得 `user` 中保存的用户信息，根据判断结果，在日志中显示。

5.4 注册功能实现

注册控制层如图 5-27 所示

```
@RequestMapping("/register")
public String register(User user) {
    System.out.println("表现层：有人注册");
    // 先判断传来的数据是否为空
    // 如果返回值为false
    if (user.getUsername().equals("") ||
        user.getPassword().equals("") ||
        user.getName().equals("") ||
        user.getSuozaixueyuan().equals("") ||
        user.getSuozaibanji().equals("") ||
        user.getSuoshuzhuanye().equals("")) {
        System.out.println("有一项为空，请重新输入");
        return "register/registerfail";
    } else {
        System.out.println("表现层：调用了业务层方法，正在检查是否已经存在该用户");
        if (loginInService.checkregister(user.getUsername()) == false) {
            System.out.println("表现层：用户不存在，向业务层发送账户数据");
            loginInService.register(user.getUsername(), user.getPassword(),
                user.getLevel(), user.getName(), user.getSuozaixueyuan(),
                user.getSuozaibanji(), user.getSuoshuzhuanye());
            return "register/registerSUCCESS";
        } else {
            System.out.println("用户名已存在，请重新输入");
            return "register/registerfail";
        }
    }
}
```

图 5-27 注册控制层

注册业务层实现类如图 5-28 所示

```
public void register(String username, String password, int level,
                    String name, String suozaixueyuan,
                    String suozaibanji, String suoshuzhuanye) {
    System.out.println("业务层：确认无重名，开始注册");
    loginDao.register(username, password, level, name, suozaixueyuan, suozaibanji, suoshuzhuanye);
    System.out.println("业务层：用户" + username + "注册成功");
}

public boolean checkregister(String username) {
    if (loginDao.checkregister(username) == null) {
        return false;
    } else {
        return true;
    }
}
```

图 5-28 注册业务层实现类

注册持久层如图 5-29 所示

```
//注册新用户
@Select("insert into user (username,password,level,name,suozaixueyuan,suozaibanji,suoshuzhuanye)" +
        " values ({username},{password},{level},{name},{suozaixueyuan},{suozaibanji},{suoshuzhuanye})")
public void register(@Param("username")String username,@Param("password")String password,@Param("level")int level,
                    @Param("name")String name, @Param("suozaixueyuan")String suozaixueyuan,
                    @Param("suozaibanji")String suozaibanji, @Param("suoshuzhuanye")String suoshuzhuanye);

//注册前查询是否有重名
@Select("select username from user where username=#{username}")
public User checkregister(@Param("username")String username);
```

图 5-29 注册持久层

根据前端用户传来的数据，判断用户是否传来空值。如果存在空值就使用户的页面跳转到注册失败页面；如果不存在 null 则继续。

把前端传来的用户名作为参数传给业务层的 checkregister() 方法，让其调用数据库方法 checkregister() 来检查数据库中是否存在相同用户名的记录。

如果不存在相同用户，则把前端传来数据通过业务层的 register() 方法传给持久层的 register() 方法，将新用户的数据保存在对应数据表中，View 层跳转到注册成功页面；否则跳转页面到注册失败页面，注册过程完毕。

5.5 退出功能实现

退出控制层如图 5-30 所示

```
//退出登录
@RequestMapping("logout")
public String outLogin(HttpSession session) {
    //发现:点浏览器返回键的时候,会自动退出
    //通过session.invalidate()方法来注销当前的session
    System.out.println("用户:" + session.getAttribute(s: "loginsuccess") + " 正在退出");
    session.invalidate();
    System.out.println("用户已经退出");
    return "login/login";
}
```

图 5-30 退出控制层

退出仅需在控制层通过 invalidate() 方法使用户登录时保存在 HttpSession 中的数据作废掉，就可以完成退出功能。

5.6 用户查询基本信息功能实现

查询信息控制层如图 5-31 所示

```
@RequestMapping("mine")
public String mine(Model modelmine, HttpSession session) {
    String username = (String) session.getAttribute(s: "loginsuccess");
    System.out.println(username);
    List<User>list=mineService.mine(username);
    modelmine.addAttribute(s: "mine", list);
    System.out.println("表现层: 查看我的");
    return "mine/mine";
}
```

图 5-31 查询信息控制层

查询信息业务层实现类如图 5-32 所示

```
public List<User> mine(String username){
    return mineDao.mine(username);
}
```

图 5-32 查询信息业务层实现类

查询信息持久层如图 5-33 所示

```
@Select("select * from user where username=#{username}")
public List<User> mine(@Param("username")String username);
```

图 5-33 查询信息持久层

把 HttpSession 的 “loginsuccess” 中保存的用户名 username 作为参数，通过业务层 mineService 的 mine() 方法，调用持久层的 mine() 方法，查询保存在数据表 user 中保存的登录的用户的基本信息，使用 Model 把数据存在 “mine” 中，

并使 View 层将页面跳转到“我的”页面中，页面中会显示对应数据，查询个人基本信息功能完毕。

5.7 用户查询课表功能实现

查询课表控制层如图 5-34 所示

```
@RequestMapping("chakebiao")
public String chakebiao(Model modelchakebiao, HttpSession session) {
    String username = (String) session.getAttribute("loginsuccess");
    String name = mineDao.chaxunyidengluyonghujibenziliao(username).getName();
    List<KeBiao> kebiaolist = chaKeBiaoService.chakebiao(username);
    modelchakebiao.addAttribute("kebiao", kebiaolist);
    System.out.println(modelchakebiao);
    return "kebiao/kebiao";
}
```

图 5-34 查询课表控制层

查询课表业务层实现类如图 5-35 所示

```
//查表
public List<KeBiao> chakebiao(String username) { return chaKeBiaoDao.chakebiao(username); }
```

图 5-35 查询课表业务层实现类

查询课表持久层如图 5-36 所示

```
@Select("select * from kebiao where username=#{username}")
public List<KeBiao> chakebiao(@Param("username")String username);
```

图 5-36 查询课表持久层

以 HttpSession 的“loginsuccess”中保存的用户账号 username 作为参数，通过 chaKeBiaoService 的 chakebiao() 方法，调用持久层的 chakebiao() 方法得到相关数据，并把数据通过 Model 的“课表”保存下来，View 层便可显示对应数据；View 层页面也会跳转到 kebiao 页面，查询课表完毕

5.8 教师发布签到任务

教师发签到任务控制层如图 5-37 所示

```

@RequestMapping("teacherfaqiandao")
public String teacherfaqiandao(TeacherFaQiandao teacherFaQiandao, HttpSession session) {
    System.out.println("表现层: 看一下前台发来了什么东西" + teacherFaQiandao);
    String username = (String) session.getAttribute(s: "loginsuccess");
    int level = mineDao.chaxunyidengluyonghujibenziliao(username).getLevel();
    if (level > 0) {
        System.out.println("表现层: 用户 "+username+" 是教师, 可以发布签到任务");
        if (teacherFaQiandao.getSuozaibanji().equals("") ||
            teacherFaQiandao.getQiandaoma().equals("") ||
            teacherFaQiandao.getStarttime().equals("") ||
            teacherFaQiandao.getOvertime().equals("")) {
            System.out.println("表现层: 新签到任务提交失败, 申请内容为空");
            return "teachers/faqandao/teacherfabuqiandaorenwufail";
        } else {
            SimpleDateFormat df = new SimpleDateFormat(pattern: "yyyy-MM-dd HH:mm:ss");// 设置日期格式
            String faqitime = df.format(new Date());// new Date() 为获取当前系统时间
            System.out.println("表现层: 教师" + username + "发起发布签到任务的申请");

            String name = mineDao.chaxunyidengluyonghujibenziliao(username).getName();
            String suozaibanji = mineDao.chaxunyidengluyonghujibenziliao(username).getSuozaibanji();
            String suozaixueyuan = mineDao.chaxunyidengluyonghujibenziliao(username).getSuozaixueyuan();
            System.out.println("表现层: 正在将新签到任务的参数提交到数据库:" +
                faqitime + "," + suozaibanji + "," + suozaixueyuan);
            teacherService.submitqiandao(teacherFaQiandao.getQiandaoma(),
                teacherFaQiandao.getKechengmingcheng(),
                teacherFaQiandao.getSuozaibanji(),
                teacherFaQiandao.getStarttime(),
                teacherFaQiandao.getOvertime(),
                username, name, level, faqitime);
            teacherService.insertnullqiandaorenwu(level: 0, teacherFaQiandao.getSuozaibanji());
            teacherService.updatestudentqiandao(qiandaozhuangtai: 0, teacherFaQiandao.getKechengmingcheng(),
                username, teacherFaQiandao.getOvertime(),
                teacherFaQiandao.getStarttime(),
                teacherFaQiandao.getQiandaoma(),
                faqitime, teacherFaQiandao.getSuozaibanji());
            System.out.println("表现层: 新签到任务的参数已经提交到数据库");
            return "teachers/faqandao/teacherfabuqiandaorenwusuccess";
        }
    } else {
        System.out.println("表现层: 用户: "+session.getAttribute(s: "loginsuccess")+ "是学生, 不能发布签到任务");
        return "teachers/faqandao/teacherfabuqiandaorenwufail";
    }
}

```

图 5-37 教师发签到任务控制层

教师发签到任务业务层实现类如图 5-38 所示

```

@Select("select * from user where username=#{username}")
public User chaxunyidengluyonghujibenziliao(@Param("username")String username);

@Select("insert into teacherfaqiandao (qiandaoma,kechengmingcheng,suozaibanji," +
    " starttime,overtime,username,name,level,faqishijian)" +
    " values (#{qiandaoma},#{kechengmingcheng},#{suozaibanji},#{starttime}," +
    " #{overtime},#{username},#{name},#{level},#{faqishijian})")
public void submitqiandao(@Param("qiandaoma")String qiandaoma,
    @Param("kechengmingcheng")String kechengmingcheng,
    @Param("suozaibanji")String suozaibanji,
    @Param("starttime") String starttime ,
    @Param("overtime") String overtime,
    @Param("username")String username,
    @Param("name")String name,
    @Param("level")int level,
    @Param("faqishijian")String faqishijian);

@Insert("insert into studentqiandao(username,name,qiandaozhuangtai,suozaibanji)" +
    "select username,name,qiandaozhuangtai,suozaibanji from user " +
    "where level=#{level} and suozaibanji=#{suozaibanji}")
public void insertnullqiandaorenwu(@Param("level")int level,
    @Param("suozaibanji")String suozaibanji);

@Update("update studentqiandao set faqishijian=#{faqishijian} where" +
    " suozaibanji=#{suozaibanji} and teacher is null")
public void updatefaqitime(@Param("faqishijian")String faqishijian,
    @Param("suozaibanji")String suozaibanji);

```

图 5-38 教师发签到任务业务层实现类

教师发签到任务持久层如图 5-39 所示

```

public void submitqiandao(String qiandaoma, String kechengmingcheng, String suozaibanji,
                          String starttime, String overtime, String username, String name,
                          int level,String faqishijian) {
    System.out.println("业务层: 正在把新签到数据写入数据库");
    teacherDao.submitqiandao(qiandaoma, kechengmingcheng, suozaibanji, starttime,
                             overtime, username, name, level,faqishijian);
    System.out.println("业务层: 新签到数据写入数据库成功");
}

public void insertnullqiandaorenwu(int level,String suozaibanji ){

    teacherDao.insertnullqiandaorenwu(level,suozaibanji);
    System.out.println("业务层: 已成批新建新签到任务");
}

public void updatestudentqiandao( int qiandaozhuangtai,
                                  String kechengmingcheng,
                                  String teacher,
                                  String overtime,
                                  String starttime,
                                  String qiandaoma,
                                  String faqishijian,
                                  String suozaibanji
                                  ){
    System.out.println("业务层:更新签到任务空表");
    teacherDao.updatestudentqiandao( qiandaozhuangtai , kechengmingcheng,teacher ,
                                     overtime , starttime, qiandaoma,faqishijian,suozaibanji );
    System.out.println("业务层:更新签到任务空表成功");
}

```

图 5-39 教师发签到任务持久层

用 HttpSession 的 “loginsuccess” 中保存的用户名 username 作为参数，直接使用 mineDao 中的 chaxunyidengluyonghujibenziliao() 方法取得用户的基本资料，并用 getLevel() 方法，得到用户的级别。

判断用户是否为教师，如果用户不是教师，View 层页面跳转到教师发布签到任务失败页面；

如果用户为教师，则继续判断用户从前端传来的数据中是否存在 null 值，如果存在 null 值，View 层页面跳转到教师发布签到任务失败页面；

如果同时满足条件：用户为教师且数据不存在 null 值，那么就可以数据库，成功发起签到任务了；

设置日期格式，把获取教师发布签到任务时的系统时间命名为 faqitime；以

username 作为参数，直接使用 mineDao 的 chaxunyidengluyonghujibenziliao() 方法并分别以 getName() 方法获取发布签到任务的教师的真实姓名；通过业务层 teacherService 的 submitqiandao() 方法，把从前端传来的众多数据和 username、name、level、faqitime 作为参数，调用持久层的 submitqiandao() 方法，在数据表 teacherfaqandao 中插入一条新记录。

再通过业务层 teacherService 的 insertnullqiandaorenwu() 方法，以 0 和前端发来的签到作用班级作为参数，调用持久层 insertnullqiandaorenwu() 方法，把数据表 user 中的 username、name、qiandaozhuangtai、suozaibanji 数据复制插入到数据表 studentqaindao 中，学生签到数据表中还存在 null 值，需要进一步操作：

再次通过业务层 teacherService 的 updatestudentqiandao() 方法将 0 和前端发来的众多一起作为参数，调用持久层的 updatestudentqiandao() 方法修改刚插入学生签到数据表的记录，使其完整。

最后将 View 层跳转到教师发布签到任务成功页面，教师发布签到任务完毕。

5.9 教师查询看学生签到状态的实现

教师查询学生签到状态控制层如图 5-40 所示

```
@RequestMapping("qiandaozhuangtai")
public String qiandaozhuangtai(Model model,yiqiandao,HttpSession session){
    String username= (String) session.getAttribute( s: "Loginsuccess");
    //学生签到表里,教师的username是teacher
    String teacher = username;
    int level = mineDao.chaxunydengluyonghujibenziliao(username).getLevel();
    if (level>0){
        System.out.println("表现层: 是教师, 查询其名下学生的签到状态");

        List<StudentQianDao> yiqiandao = teacherService.qiandaozhuangtai( qiandaozhuangtai: 1,teacher);
        model.yiqiandao.addAttribute( s: "yiqiandao",yiqiandao);
        List<StudentQianDao> weiqiandao = teacherService.qiandaozhuangtai( qiandaozhuangtai: 0, teacher);
        model.yiqiandao.addAttribute( s: "weiqiandao",weiqiandao);
        List<StudentQianDao> qiandaochaoshi = teacherService.qiandaozhuangtai( qiandaozhuangtai: 2, teacher);
        model.yiqiandao.addAttribute( s: "qiandaochaoshi",qiandaochaoshi);
        System.out.println("表现层: 签到状态查询成功");
        return "teachers/faqianDao/teacherfabuqiandaorenwusuccess";
    }else {
        System.out.println("表现层: 不是教师");
    }
    return "teachers/faqianDao/teacherfabuqiandaorenwusuccess";
}
```

图 5-40 教师查询学生签到状态控制层

教师查询学生签到状态业务层实现类如图 5-41 所示

```
public List<StudentQianDao> qiandaozhuangtai(int qiandaozhuangtai, String teacher) {
    System.out.println("业务层: 查询了所有学生的签到状态");
    return teacherDao.qiandaozhuangtai(qiandaozhuangtai, teacher);
}
```

图 5-41 教师查询学生签到状态业务层实现类

教师查询学生签到状态持久层如图 5-42 所示

```
@Select("select * from user where username=#{username}")
public User chaxunydengluyonghujibenziliao(@Param("username")String username);

@Select("select * from studentqianDao where" +
        " qiandaozhuangtai=#{qiandaozhuangtai}" +
        " and teacher = #{teacher}")
public List<StudentQianDao> qiandaozhuangtai(
        @Param("qiandaozhuangtai")int qiandaozhuangtai,
        @Param("teacher")String teacher);
```

图 5-42 教师查询学生签到状态持久层

以 HttpSession 保存的用户名作为参数, 直接调用持久层 mine 的 chaxunydengluyonghujibenziliao() 方法的并通过 getlevel() 得到用户级别;

判断用户是否为教师，如果不是则 View 层页面跳转到教师发布签到任务成功的页面，尽管是成功页面，学生不是教师，所以学生还是查询不了学生的签到情况，只有教师能查询；

用户为教师时，以 0、1、2 分别和用户名 teacher 作为参数，通过 teacherService 调用 teacherDao 的 qiandaozhuangtai() 方法分别查询到已签到、未签到和作废签到，并分别

保存到不同名的 Model 中保存，并跳转到 View 层的对应页面，数据将会在教师查询看学生签到时在遍历，教师查询学生签到情况完毕。

5.10 教师发布通知实现

教师发布通知控制层如图 5-43 所示

```
@RequestMapping("teacherfatongzhi")
public String fatongzhi(Tongzhi tongzhi, HttpSession session) {
    String username = (String) session.getAttribute(s: "loginsuccess");
    int level = mineDao.chaxunyidengluyonghujibenziliao(username).getLevel();
    String name = mineDao.chaxunyidengluyonghujibenziliao(username).getName();
    if (level > 0) {
        System.out.println("表现层: 用户 " + username + " 是教师, 可以发布签到任务");
        if (tongzhi.getTitle().equals("") ||
            tongzhi.getZhengwen().equals("") ||
            tongzhi.getFugaibanji().equals("")) {
            System.out.println("表现层: 新通知发布失败, 存在内容为空");
            return "teachers/fatongzhi/teacherfatongzhifail";
        } else {
            SimpleDateFormat df = new SimpleDateFormat(pattern: "yyyy-MM-dd HH:mm:ss");// 设置日期格式
            String faqitime = df.format(new Date());// new Date()为获取当前系统时间
            teacherService.teacherfatongzhi(tongzhi.getTitle(),
                tongzhi.getZhengwen(),
                faqitime, name,
                tongzhi.getFugaibanji());
            System.out.println("表现层: 教师" + username + "发起发布签到任务的申请");
            return "teachers/fatongzhi/teacherfatongzhisuccess";
        }
    } else {
        System.out.println("表现层: 不是教师, 不能发通知/作业");
        return "teachers/fatongzhi/teacherfatongzhifail";
    }
}
```

图 5-43 教师发布通知控制层

教师发布通知业务层实现类如图 5-44 所示

```
public void teacherfatongzhi(String title, String zhengwen,
    String starttime, String fabuzhe,
    String fugaibanji) {
    teacherDao.teacherfatongzhi(title, zhengwen,
        starttime, fabuzhe, fugaibanji);
}
```

图 5-44 教师发布通知业务层实现类

教师发布通知持久层如图 5-45 所示

```
@Select("select * from user where username=#{username}")
public User chaxunydenglu YonghuJibenZiliao(@Param("username")String username);

@Select("insert into tongzhi (title,zhengwen,starttime,fabuzhe,fugaibanji) " +
        "values (#{title},#{zhengwen},#{starttime},#{fabuzhe},#{fugaibanji})")
public void teacherfatongzhi(@Param("title")String title,
                             @Param("zhengwen")String zhengwen,
                             @Param("starttime")String starttime,
                             @Param("fabuzhe")String fabuzhe,
                             @Param("fugaibanji")String fugaibanji );
```

图 5-45 教师发布通知持久层

用 HttpSession 的 “loginsuccess” 中保存的用户名 username 作为参数，直接使用 mineDao 中的 chaxunydenglu YonghuJibenZiliao() 方法取得用户的基本资料，并用 getLevel() 和 getName() 方法，得到用户的级别 level 和真实姓名 name。

判断用户是否为教师，如果不是教师就不能发布通知/作业，并使 View 层跳转到发布通知/作业失败页面，在失败页面可以重新发布通知/作业，或者返回服务页面，或者查询自己已经发布的通知；

如果用户是教师，继续判断教师传来的数据是否存在空值，如果存在空值，那么通知/作业发布失败，View 层跳转到发布通知/作业失败页面；

如果数据中不存在空值，则获取系统时间并修改为合适格式，且命名为 String 类型的 faqitime；将前端发来的参数和经过操作得到的 level、name 和 faqitime 作为参数，通过 teacherService 的 teacherfatongzhi() 方法调用持久层的 teacherfatongzhi() 方法，向数据表 tongzhi 中插入一条新的记录。View 层跳转到发布通知/作业成功页面，教师可在此页面查询看自己发布通知/作业的记录，自此教师发通知实现完毕。

5.11 教师查询自己发布的通知实现

教师查询自己发布的通知控制层如图 5-46 所示

```
//查教师自己发布的通知
@RequestMapping("teacherchazijifadetongzhi")
public String teacherchazijifadetongzhi(Model modeljiaoshichazijifadetongzhi,HttpSession session){
    //获得用户名,需要先强转类型为string
    String username= (String) session.getAttribute( s: "loginsuccess");
    String name= mineDao.chaxunqidengluonghujibenziliao(username).getName();
    //以自己的名字做参数查通知
    List<Tongzhi> list= teacherService.teacherchazijifadetongzhi(name);
    modeljiaoshichazijifadetongzhi.addAttribute( s: "modeljiaoshichazijifadetongzhi",list);
    System.out.println("业务层: 查询了自己发布的通知");
    return "teachers/fatongzhi/teacherfatongzhisuccess";
}
```

图 5-46 教师查询自己发布的通知控制层

教师查询自己发布的通知业务层实现类如图 5-47 所示

```
public List<Tongzhi> teacherchazijifadetongzhi(String fabuzhe){
    return teacherDao.teacherchazijifadetongzhi(fabuzhe);
}
```

图 5-47 教师查询自己发布的通知业务层实现类

教师查询自己发布的通知持久层如图 5-48 所示

```
//持久层实现
@Select("select * from tongzhi where fabuzhe=#{fabuzhe}")
public List<Tongzhi> teacherchazijifadetongzhi(@Param("fabuzhe")String fabuzhe);
```

图 5-48 教师查询自己发布的通知持久层

将 HttpSession 中保存的用户名作为参数,直接调用持久层的 mineDao 的 chaxunqidengluonghujibenziliao().getName() 方法,获取用户真实姓名,

以用户姓名做参数通过 teacherService 的 teacherchazijifadetongzhi() 方法调用到持久层的 teacherchazijifadetongzhi() 方法,并将查询到的数据通过 Model 的 addAttribute() 方法传到前端,View 层页面跳转到教师发通知成功页面,教师查询通知完毕。

备注:学生也可打开教师发布通知成功页面,但是由于学生不能发布通知,所以学生还是不能够查询到自己不能查询到的数据。

5.12 教师查询学生请假记录实现

教师查询学生请假记录控制层如图 5-49 所示

```

@RequestMapping("teacherchaxueshengqingjiajilu")
public String teacherchaxueshengqingjiajilu(Model model, HttpSession session){
    String username=(String)session.getAttribute("loginSuccess");
    String suozaibanji=teacherService.teacherchaziji(username);
    int level=mineDao.chaxunyidengluyonghujibenziliao(username).getLevel();
    System.out.println(suozaibanji);
    System.out.println("表现层：教师查学生请假记录");
    if (level>0){
        List<StudentQingjia> list= teacherService.teacherchaxueshengqingjiajilu(suozaibanji);
        model.addAttribute("chaxueshengqingjia",list);
        System.out.println(list);
        return "teachers/chaqingjia/chaqingjiasuccess";
    }else {
        return "teachers/chaqingjia/chaqingjiafail";
    }
}

```

图 5-49 教师查询学生请假记录控制层

教师查询学生请假记录业务层实现类如图 5-50 所示。

```

public List<StudentQingjia> teacherchaxueshengqingjiajilu(String suozaibanji){
    return teacherDao.teacherchaxueshengqingjiajilu(suozaibanji);
}

public String teacherchaziji(String username){
    return teacherDao.teacherchaziji(username);
}

```

图 5-50 教师查询学生请假记录业务层实现类

教师查询学生请假记录持久层如图 5-51 所示。

```

@Select("select * from user where username=#{username}")
public User chaxunyidengluyonghujibenziliao(@Param("username")String username);

//教师查看学生请假记录
@Select("select * from qingjia where suozaibanji=#{suozaibanji}")
public List<StudentQingjia> teacherchaxueshengqingjiajilu(@Param("suozaibanji")String suozaibanji);

//教师查自己
@Select("select suozaibanji from teachers where username=#{username}")
public String teacherchaziji(String username);

```

图 5-51 教师查询学生请假记录持久层

将 HttpSession 中保存的用户名作为参数，通过 teacherService 的 teacherchaziji() 方法调用持久层 teacherDao 的 teacherchaziji() 方法，得到用户所在班级；直接调用持久层的 mineDao 的 chaxunyidengluyonghujibenziliao().getLevel() 方法，获取用户级别 level；

以用户 level 判断用户是否为教师，如果为教师，则以用户 suozaibanji 为参

数,通过 teacherService 的 teacherchaxueshengqiangjiajilu() 方法,调用持久层 teacherDao 的 teacherchaxueshengqiangjiajilu() 方法,将取得的数据作为数据借由 Model 的 addAttribute() 方法将数据传到对应网页,界面跳转到 chaqingjiasuccess.jsp 中;

如过用户不是教师,则前端页面跳转到 chaqingjiafail.jsp 中;教师查询学生请假记录实现完毕。

5.13 学生提出请假功能实现

学生请假控制层如图 5-52 所示。

```
@RequestMapping("/shenqingqingjia")
public String shenqingqingjia(StudentQingjia studentQingjia, HttpSession session) {
    String username = (String) session.getAttribute(s: "Loginsuccess");
    //只有学生需要申请请假,学生为0,教师1,班主任2,辅导员3
    //用通用mysql查询办法!!!!!!!!!!!!!!达到获取用户level的目的
    int level = mineDao.chaxunydengluyonghujibenziliao(username).getLevel();
    if (level == 0) {
        //从前端返回的数据,默认不为空的,需要取其属性的值,值为空才是空
        if (studentQingjia.getQingjialeixing().equals("") ||
            studentQingjia.getStarttime().equals("") ||
            studentQingjia.getOvertime().equals("") ||
            studentQingjia.getQingjialiyou().equals("")) {
            System.out.println("表现层: 请假申请失败, 申请内容为空");
            return "students/qingjia/shenqingqingjiafail";
        } else {
            System.out.println("表现层: 有人发起请假申请");
            System.out.println("表现层: 确认已经登录的用户名为: " + username);
            //根据用户名查询该用户的基本信息
            GonggongUserShuxingDao.chaxunydengluyonghujibenziliao(username);
            定义一些数据库属性, 属性内容为上面查到的信息
            String name = mineDao.chaxunydengluyonghujibenziliao(username).getName();
            String suozaibanji = mineDao.chaxunydengluyonghujibenziliao(username).getSuozaibanji();
            String suozaixueyuan = mineDao.chaxunydengluyonghujibenziliao(username).getSuozaixueyuan();
            System.out.println("表现层: 正在将请假参数提交到数据库:" + name + ", " + suozaibanji + ", " + suozaixueyuan);
            studentService.submitqingjia(username, name, suozaibanji, suozaixueyuan,
                studentQingjia.getQingjialeixing(),
                studentQingjia.getStarttime(), studentQingjia.getOvertime(),
                studentQingjia.getQingjialiyou());
            System.out.println("表现层: 请假申请的参数已经提交到数据库");
            return "students/qingjia/shenqingqingjiasuccess";
        }
    } else {
        System.out.println("表现层: 用户" + username + "不是学生, 无需申请请假");
        return "students/qingjia/shenqingqingjiafail";
    }
}
```

图 5-52 学生请假控制层

学生请假业务层实现类如图 5-53 所示。

```

public void submitqingjia(String username, String name, String suozaibanji,
                        String suozaixueyuan, String qingjialeixing,
                        String starttime, String overtime, String qingjialiyou) {
    System.out.println("业务层: 正在把请假数据写入数据库");
    studentDao.submitqingjia(username, name, suozaibanji, suozaixueyuan,
                            qingjialeixing, starttime, overtime, qingjialiyou);
    System.out.println("业务层: 请假数据写入数据库成功");
}

```

图 5-53 学生请假业务层实现类

学生请假持久层如图 5-54 所示。

```

@Select("select * from user where username=#{username}")
public User chaxunqidengluoyonghujibenziliao(@Param("username")String username);
//保存一个请假
@Select("insert into qingjia (username,name,suozaibanji,suozaixueyuan," +
        "qingjialeixing,starttime,overtime,qingjialiyou)" +
        " values (#{username},#{name},#{suozaibanji},#{suozaixueyuan}," +
        " #{qingjialeixing},#{starttime},#{overtime},#{qingjialiyou})")
public void submitqingjia(@Param("username")String username,
                        @Param("name")String name,
                        @Param("suozaibanji")String suozaibanji ,
                        @Param("suozaixueyuan")String suozaixueyuan,
                        @Param("qingjialeixing")String qingjialeixing,
                        @Param("starttime") String starttime,
                        @Param("overtime")String overtime,
                        @Param("qingjialiyou")String qingjialiyou);

```

图 5-54 学生请假持久层

将 HttpSession 中保存的用户名 username 作为参数，直接调用 mineDao 的 chaxunqidengluoyonghujibenziliao().getLevel() 方法，得到用户级别；

判断用户是否为教师，如果为教师则继续判断前端发来的数据中是否存在 null 值，若无空值，就直接调用 mineDao 的上述方法的分别查询 getName()、getSuozaibanji()、getSuozaixueyuan() 并分别保存在 name、suozaibanji、suozaixueyuan 中，再以这三个和其他从前端传来的其他参数通过业务层的 StudentService 的 submitqingjia() 方法，调用持久层 studentDao 的 submitqingjia() 方法，向数据表 qingjia 增加一条新的记录，最后将 View 层页面跳转到学生请假成功页面；

若判断用户为学生或从前端传回数据中存在空值，则将 View 层页面跳转到请假失败的页面，在这个页面中可以看到自己请假失败的原因，也可在此页面重新进行请假申请，自此，学生请假功能实现完毕。

5.14 学生查询自己请假记录的功能实现

学生查询自己请假记录控制层如图 5-55 所示。

```
// 5-55 控制层
@RequestMapping("/chakanwodeqingjiajilu")
public String findAllqingjia(Model modelfindAllqingjiajilu, HttpSession session) {
    String username = (String) session.getAttribute(s: "loginsuccess");
    int level = mineDao.chaxunqidengluoyonghujibenziliao(username).getLevel();
    if (level > 0) {
        System.out.println("表现层: 用户" + username + "不是学生, 故没有请假记录, ");
        return "students/qingjia/shenqingqingjiafail";
    } else {
        System.out.println("表现层: 查询所有信息");
        // 调用Service的方法
        // 这里把登录的用户信息的用户名通过HttpSession 的getAttribute传给service的搜索所有用户的方法, 因为类型不一样, 所以强转一下
        List<StudentQingjia> list = studentService.sousuoqingjia((String) session.getAttribute(s: "loginsuccess"));
        // 重定向, 先用sout检查一下list是否由数据
        System.out.println("见擦汗一下list中有咩有"+list);

        modelfindAllqingjiajilu.addAttribute(s: "findAllqingjiajilu", list);
        return "students/qingjia/shenqingqingjialist";
    }
}
}
```

图 5-55 学生查询自己请假记录控制层

学生查询自己请假记录业务层实现类如图 5-56 所示。

```
public List<StudentQingjia> sousuoqingjia(String username) {
    System.out.println("业务层: 查询的所有请假记录");
    return studentDao.sousuoqingjia(username);
}
```

图 5-56 学生查询自己请假记录业务层实现类

学生查询自己请假记录持久层如图 5-57 所示。

```
@Select("select * from user where username=#{username}")
public User chaxunqidengluoyonghujibenziliao(@Param("username")String username);

// 5-57 持久层
@Select("select * from qingjia where username = #{username}")
public List<StudentQingjia> sousuoqingjia(@Param("username")String username);
```

图 5-57 学生查询自己请假记录持久层

将 HttpSession 中保存的用户名 username 作为参数, 直接调用 mineDao 的 chaxunqidengluoyonghujibenziliao().getLevel() 方法, 得到用户级别;

判断用户是否为学生, 如果用户是学生, 则将用户的 username 作为参数通过业务层 studentService 的 sousuoqingjia() 方法, 调用持久层 studentDao 的 sousuoqingjia() 方法, 并将返回的数据通过 Model 保存在“findAllqingjiajilu”

发送到前端页面由前端页面负责显示，View 层的页面跳转到 shenqingqingjialist.jsp。

如果判断用户不是学生，View 层的页面跳转到 shenqingqingjiafail.jsp,将会显示失败原因，自此学生查询自己请假记录实现完毕。

5.15 学生签到功能实现

学生签到控制层如图 5-58 所示。

```
@RequestMapping("xueshengqiandao")
public String xueshengqiandao(TeacherFaQiandao teacherFaQiandao, HttpSession session) {
    String kechengmingcheng = teacherFaQiandao.getKechengmingcheng();
    String username = (String) session.getAttribute(s: "loginsuccess");
    int level = mineDao.chaxunyidengluyonghujibenziliao(username).getLevel();
    if (teacherFaQiandao.getQiandaoma().equals("") ||
        teacherFaQiandao.getKechengmingcheng().equals("")) {
        System.out.println("表现层: 存在空值, 签到失败");
        return "students/qiandao/stuqiandaofail";
    } else {
        if (level < 1) { // 确认是否为学生
            // 时间比较, 系统时间
            SimpleDateFormat df = new SimpleDateFormat(pattern: "yyyy-MM-dd HH:mm:ss");// 设置日期格式
            System.out.println(df.format(new Date())); // new Date() 为获取当前系统时间
            String time = df.format(new Date());
            // 相同类型比较, 返回1则左边大, 返回0相等, 返回-1右边大
            time.compareTo(overtime);
            // 检查 签到码前
            int chageshu = studentService.chageshu(username, kechengmingcheng, qiandaozhuangtai: 0);
            System.out.println(chageshu);
            if (chageshu > 0) { // 确认存在需要签到的任务
                // overtime
                String overtime = studentService.qiandaoyanzheng(username, kechengmingcheng, qiandaozhuangtai:
                if (time.compareTo(overtime) > 0) { // 确认是否超时
                    System.out.println("表现层: 签到超时, 签到失败");
                    return "students/qiandao/stuqiandaofail";
                } else { // 确认未超时
                    int qiandaozhuangtai = studentService.qiandaoyanzheng(username, kechengmingcheng, qiandaozhuangtai:
                    if (qiandaozhuangtai < 1) {
                        qiandaozhuangtai<1 就说明该生还没有签到
                        String qiandaoma = studentService.qiandaoyanzheng(username, kechengmingcheng, qiandaozhuangtai:
                        // 验证签到码, 判断学生提交的签到码是否和教师发布的一致
                        if (qiandaoma.equals(teacherFaQiandao.getQiandaoma())) {
                            studentService.studentqiandao( qiandaozhuangtai: 1, qiandaozhuangtai2: 0, username, kechengming
                            System.out.println("表现层: 签到码正确, 签到成功");
                            return "students/qiandao/stuqiandaosuccess";
                        } else {
                            System.out.println("表现层: 签到码错误, 请重新签到");
                            return "students/qiandao/stuqiandaofail";
                        }
                    } else {
                        System.out.println("表现层: 该生已经签到过了, 无需重复签到");
                        return "students/qiandao/stuqiandaoother";
                    }
                }
            } else {
                System.out.println("表现层: 查到" + chageshu + "个关于这个学生的签到任务");
                return "students/qiandao/stuqiandaoother";
            }
        } else {
            System.out.println("表现层: 不是学生, 无需签到");
            return "students/qiandao/stuqiandaofail";
        }
    }
}
```

图 5-58 学生签到控制层

学生签到业务层实现类如图 5-59 所示。

```
public StudentQiandao studentqiandao(int qiandaozhuangtai, int qiandaozhuangtai2,
                                     String username, String kechengmingcheng) {
    System.out.println("业务层: 学生要签到, 改0为1");
    return studentDao.studentqiandao(qiandaozhuangtai, qiandaozhuangtai2, username, kechengmingcheng);
}
public int chageshu(String username, String kechengmingcheng, int qiandaozhuangtai) {
    System.out.println("业务层: 查0的个数");
    return studentDao.chageshu(username, kechengmingcheng, qiandaozhuangtai);
}
public StudentQiandao qiandaoyanzheng(String username, String kechengmingcheng, int qiandaozhuangtai) {
    System.out.println("业务层: 签到前验证");
    return studentDao.qiandaoyanzheng(username, kechengmingcheng, qiandaozhuangtai);
}
```

图 5-59 学生签到业务层实现类

学生签到持久层如图 5-60 所示。

```
//签到 改0为1
@Select("update studentqiandao set qiandaozhuangtai=#{qiandaozhuangtai}" +
        " where qiandaozhuangtai=#{qiandaozhuangtai2} and " +
        "username = #{username} and kechengmingcheng=#{kechengmingcheng}")
public StudentQiandao studentqiandao(@Param("qiandaozhuangtai")int qiandaozhuangtai,
                                     @Param("qiandaozhuangtai2")int qiandaozhuangtai2,
                                     @Param("username")String username,
                                     @Param("kechengmingcheng")String kechengmingcheng);

//当签到状态为0的个数为0时, 执行qiandaoyanzheng() .getQiandaozhuangtai()会报空指针异常
//那么我先查0个数, 如果不为0, 那么再执行qiandaoyanzheng()
@Select("select count(*) from studentqiandao where username=#{username}" +
        " and kechengmingcheng=#{kechengmingcheng} and qiandaozhuangtai=#{qiandaozhuangtai}")
public int chageshu(@Param("username")String username,
                   @Param("kechengmingcheng")String kechengmingcheng,
                   @Param("qiandaozhuangtai")int qiandaozhuangtai);

//签到前验证签到码和签到时间
//在学生签到表中, 每名学生的每
! @Select("select * from studentqiandao where username=#{username}" +
        " and kechengmingcheng=#{kechengmingcheng} " +
        "and qiandaozhuangtai=#{qiandaozhuangtai}")
public StudentQiandao qiandaoyanzheng(@Param("username")String username,
                                     @Param("kechengmingcheng")String kechengmingcheng,
                                     @Param("qiandaozhuangtai")int qiandaozhuangtai
                                     );

//自动查询已经登录的用户的基本资料
@Select("select * from user where username=#{username}")
public User chaxunyidengluyonghujibenziliao(@Param("username")String username);
```

图 5-60 学生签到持久层

通过 getKechengmingcheng() 方法获得前端输入的课程名称并保存为 String 类型的 kechengmingcheng;

通过 session 的 getAttribute() 方法获得保存在 HttpSession 的

“loginSuccess”中的用户名 username，保存为 String 类型的 username；

以 username 为参数，直接调用持久层 mineDao 的 chaxunyidenglujibenziliao().getLevel();得到用户的级别 Level，保存为 int 类型的 level；

先判断用户传来的数据中是否存在空值，若存在空值则令 View 层页面跳转到学生签到失败页面 stuqiandaofail.jsp；

如果用户传来数据中不存在 null，继续判断用户是否为学生，若用户不是学生，则 View 层跳转页面到 stuqiandaofail 页面，提示用户不是学生，无需签到；

若为学生则获取当前时间，设置为合适格式，以 username 和 kechengmingcheng 和 0 作为参数，通过业务层 studentService 的 chashu() 方法

调用持久层 studentDao 的 chashu() 方法，取得学生签到表中希望请假的这名用户需要请假的记录的个数，保存为 int 类型的 chageshu；

判断是否存在需要签到的任务，若不存在签到任务，View 层页面跳转为 stuqiandaooother 页面，页面中会提示用户不存在签到任务；

否则存在签到任务，通过业务层 studentService 的 qiandaoyanzheng() 方法，调用持久层 studentDao 的 qiandaoyanzheng() 方法定位到

这条签到任务，再以.getOvertime() 获取该次签到任务的超时时间并保存为 String 类型的 overtime，使用 compareTo() 方法比较当前

系统时间和 overtime，若得到的结果小于 0 则说明签到没有超时；

签到没有超时就通过业务层 studentService 的相同办法得到签到状态 qiandaozhuangtai 保存为 int 类型的 qiandaozhuangtai，

判断签到状态是否为未签到（状态为 0 则为未签到，为 1 则为已签到），若用户状态为已签到，则 View 层页面跳转为 stuqiandaooother 页面，

页面中会提示用户已经完成签到无需重复签到；

若用户未签到，则用上述办法再得到签到码 qiandaoma，验证学生输入的签到码和数据库中签到码是否一致，若不一致则签到码错误需重新签到，

View 层页面跳转为签到失败页面，用户可在页面中继续签到；若签到码正确，则 View 层页面跳转为签到成功页面，学生签到功能设计完毕。

5.16 学生查询通知功能实现

学生查询通知控制层如图 5-61 所示。

```
@RequestMapping("studentchatongzhi")
public String studentchatongzhi(Model modelstudentchatongzhi, HttpSession session) {
    //通过HttpSession查登录账号的用户名
    String username = (String) session.getAttribute(s: "loginsuccess");
    //查用户名的User表信息,拿到班级数据,顺便查以下level
    String suozaibanji = mineDao.chaxunyidengluyonghujibenziliao(username).getSuozaibanji();
    int level = mineDao.chaxunyidengluyonghujibenziliao(username).getLevel();
    if (level > 0) {
        System.out.println("用户是教师, 跳转到教师看通知的页面");
        return "teachers/fatongzhi/teacherfatongzhisuccess";
    } else {
        System.out.println("用户是学生继续进行查通知");
        //把所在班级作为参数查通知输出为一个list
        List<Tongzhi> changguitungzhi = studentService.studentchatongzhi(suozaibanji);
        //把list中的数据传入model容器中,就像是个内存,可以保存
        modelstudentchatongzhi.addAttribute(s: "modelstudentchatongzhi", changguitungzhi);
        //用return 跳转到chatongzhilist.jsp页
        return "students/chatongzhi/chatongzhilist";
    }
}
```

图 5-61 学生查询通知控制层

学生查询通知业务层实现类如图 5-62 所示。

```
public List<Tongzhi> studentchatongzhi(String fugaibanji){
    return studentDao.studentchatongzhi(fugaibanji);
}
```

图 5-62 学生查询通知业务层实现类

学生查询通知持久层如图 5-63 所示。

```
@Select("select * from user where username=#{username}")
public User chaxunyidengluyonghujibenziliao(@Param("username")String username);

@Select("select * from tongzhi where fugaibanji =#{fugaibanji}")
public List<Tongzhi> studentchatongzhi(@Param("fugaibanji")String fugaibanji);
```

图 5-63 学生查询通知持久层

通过 HttpSession 的 getAttribute() 方法, 得到用户登录时保存的用户名 username, 保存为 String 类型的 username; 用持久层的 mineDao 的方法得到用户所在班级和级别分别保存为 String 类型的 suozaibanji 和 int 类型的 level;

判断用户是否为学生, 若用户不是学生, 则为其跳转到教师查询通知的页面;

若用户为学生, 则以学生所在班级为参数, 通过业务层的 studentService 的 studentchatongzhi() 方法, 调用持久层 studentDao 的 studentchatongzhi() 方法, 找到关于该学生所在班级的所有通知, 并保存了 List 类型的列表中, 再保存到 Model

中，前端就可以遍历通知记录，学生查询通知功能实现完毕。

5.17 服务器定时任务功能实现

如果在教师发布新的签到任务之前，学生并没有按时签到，那么在新签到任务发布后，学生签到表中就会存在 2 个需要签到的任务出现，系统就会出错，所以我需要能自动将过期签到任务重置的定时任务源码如图 5-64 所示。

```
@Component
public class ASpringtaskTest {
    @Autowired
    private StudentDao studentDao;

    //initialDelay: 服务启动后, 多少毫秒启动该定时任务
    //fixedDelay: 每隔多长时间执行一次定时任务
    @Scheduled(initialDelay = 1 * 1000, fixedDelay = 2700 * 1000)
    public void updateqiandaozhuangtai() {
        SimpleDateFormat df = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");//设置日期格式
        String time = df.format(new Date());// new Date()为获取当前系统时间
        System.out.println(" -----定时任务 每45分钟一次-----")
        System.out.println(" |
        System.out.println(" |
        System.out.println(" | "+"当前系统时间为"+time+"
        System.out.println(" | 已经将学生签到表的数据更新
        System.out.println(" | -----")
        studentDao.updateqian( qiandaozhuangtai: 2);

    }

    @Update("update studentqiandao set qiandaozhuangtai=#{qiandaozhuangtai}")
    public void updateqian(int qiandaozhuangtai);
}
```

图 5-64 定时任务源码

类名上注解@Component 以确保该定时任务能被 spring 扫描到，自动注入持久层接口类

方法上注解@Scheduled 声明该方法为计时器方法，服务器启动后，1 秒后启动该定时任务，每隔 45 分钟启动一次该定时任务。

该定时任务就是显示一下系统时间，并且通过持久层 studentDao 的 updateqian() 方法，将数据表中签到状态全部改为 2，以达到避免出错的目的，自此定时任务功能实现。

5.18 服务器拦截器功能实现

由于该套系统几乎所有的操作都要在用户登录的前提下才能正常执行，因此需要一个拦截器，判断用户是否登录，拦截器如图 5-65 所示。

```

public class UserInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object o) throws Exception {
        // 获取请求的uri:去除http://localhost:8080这部分剩下的
        String uri = request.getRequestURI();
        // UTL:除了login.jsp是可以公开访问的,其他的URL都进行拦截控制
        // 请求的url内 login"的位置,返回的是一个数字,代表出现的位置,-1表示不存在。>=0即存在,即是login的请求
        if (uri.indexOf("/login")>0||uri.indexOf("/register")>0) {
            System.out.println("-----");
            System.out.println("拦截器:检测前端请求为登录或注册,拦截通过");
            return true;
        }
        // 获取session
        // request.getSession()可以帮你得到HttpSession类型的对象
        HttpSession session = request.getSession();
        // User user = (User) session.getAttribute("USER_SESSION");
        // 判断session中是否有用户数据,如果有,则返回true,继续向下执行
        if (session != null) {
            System.out.println("-----");
            System.out.println("拦截器:该用户已经登录,通过拦截,用户为:"+session.getAttribute("login"));
            return true;
        }
        System.out.println("-----");
        // 不符合条件的给出提示信息,并转发到登录页面
        // request.setAttribute("msg", "您还没有登录,请先登录!");
        // request.getRequestDispatcher("/login.jsp").forward(request, response);
        System.out.println("检测到没有登录,需要先登录");
        return false;
    }
}

```

图 5-65 拦截器

该类继承自 HandlerInterceptor。

获取请求的 uri:如果请求由登录或注册页面发出,则 return true 放行;

通过 HttpSession 是否为 null 判断一个请求发出前用户是否登录,,若已经登录,则 return true 放行,否则 return false 拦截拦截器功能实现完毕。

5.19 APP 端的功能实现

APP 由 2 部分组成,欢迎页和功能页,打开 APP 后先看到的是欢迎页,三秒后自动跳转到功能页,也可手动跳过;功能页是里是一个 WebView 控件,用来显示考勤管理系统的网站。

5.19.1 欢迎页

欢迎页 Xml 布局如图 5-66 所示。

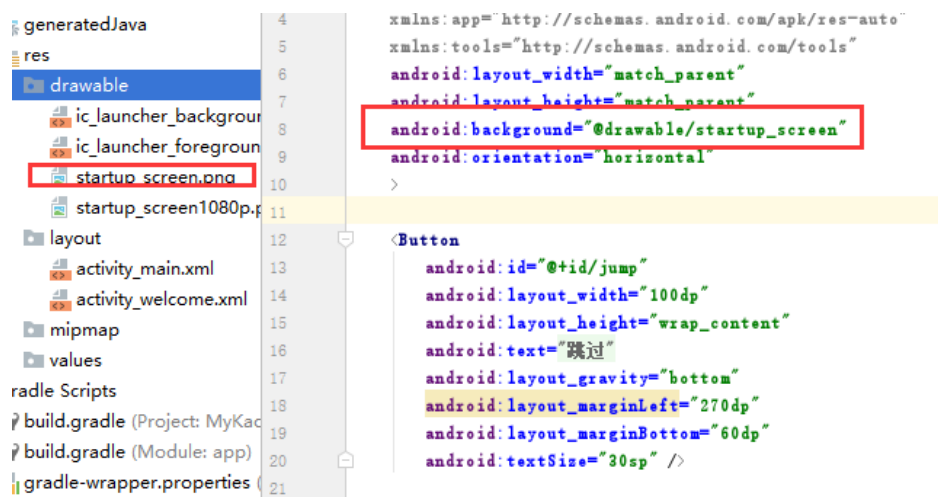


图 5-66 欢迎页 Xml 布局

欢迎页 Java 实现如图 5-67 所示。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_welcome);

    getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
    startMainActivity(); //启动下面写的一个方法
    jump = (Button) findViewById(R.id.jump);
    jump.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent zhudong = new Intent(packageContext, WelcomeActivity.this, MainActivity.class);
            startActivity(zhudong);
        }
    });
}

private void startMainActivity() {
    TimerTask delayTask = new TimerTask() {
        @Override
        public void run() {
            Intent zidong = new Intent(packageContext, WelcomeActivity.this, MainActivity.class);
            startActivity(zidong);
            WelcomeActivity.this.finish();
        }
    };
    Timer timer = new Timer();
    timer.schedule(delayTask, delay: 3000); //延时3秒执行 run 里面的操作
}
```

图 5-67 欢迎页 Java 实现

欢迎页的布局是以一张图片为背景，一个按钮为跳过功能按钮，其功能需要在 WelcomeActivity 中实现。

WelcomeActivity 中，用 getWindow().addFlags() 的方法，让欢迎页在 APP 打

开时能够达到覆盖全屏的效果;为跳过按钮添加点击监听,当按钮被点击后,从当前页面跳转到功能页。

欢迎页也可在 3 秒倒计时后自动跳转到功能页,仅需在写一个方法 `startMainActivity()`,其中新建一个 `Timer` 定时器,使其每隔三秒执行一次 `run()` 方法中的操作,而 `run()` 方法中的操作则是将页面从欢迎页跳转到功能页,使用 `finish()` 方法关闭 `WelcomeActivity`,可以保证定时器效果不再出现,接下来就可以操作功能页中, `WebView` 的网页了。

为了让 APP 能先显示欢迎页,需要修改 `AndroidManifest.xml` 默认打开的主活动,配置文件配置如图 5-68 所示。

```
<activity android:name=".WelcomeActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"></action>
        <category android:name="android.intent.category.LAUNCHER"></category>
    </intent-filter>
</activity>
<activity android:name=".MainActivity" />
```

图 5-68 配置文件配置

5.19.2 功能页

功能页 `Xml` 布局如图 5-69 所示。

```
<WebView
    android:id="@+id/activity_main_webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"></WebView>
```

图 5-69 功能页 `Xml` 布局

功能页 `Java` 实现如图 5-70 所示。

```
public class MainActivity extends AppCompatActivity {
    private WebView webView;
    @Override
} protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    webView=(WebView)findViewById(R.id.activity_main_webview);
    //这句可以让界面跳转时，不自动打开手机默认浏览器，让程序只在webview中运行
    webView.setWebViewClient(new WebViewClient());
    webView.loadUrl("http://192.168.0.121:8080/WF_IQGL_Server_war_exploded/");
} }
}
```

图 5-70 功能页 Java 实现

功能页中只有一个控件 WebView。

为了保证对 WebView 中的网页操作时不会出现自动打开手机默认浏览器，让程序只在本 APP 的 WebView 中运行，保证 WebView 不跳转的代码需要加入代码如图 5-71 所示。

```
webView.setWebViewClient(new WebViewClient());←
```

图 5-71 保证 WebView 不跳转的代码

使用 webview 的 loadUrl() 方法，指定其链接的网址就可以打开考勤管理系统的网站了。

当然想要能联网，需要为 APP 提供网络权限后才能实现，在 AndroidManifest.xml 中加入网络权限以保证 APP 能正常运行，网络权限如图 5-72 所示。

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

图 5-72 网络权限

5.20 网站的功能实现

在提要中已经说明，网站中都是按钮、文本输入框和表格带有简单的样式，用到的技术重复率高，数量大，因此省略。

6 成品展示

由于 App 的使用覆盖了网站的使用，所以成品展示以 APP 为主。

6.1 APP 欢迎页

APP 欢迎页如图 6-1 所示。



图 6-1 APP 欢迎页

6.2 登录

登录页面如图 6-2 所示。



欢迎登录考勤管理系统

请输入用户名

请输入密码

登录

注册

图 6-2 登录页面

登录失败页面如图 6-3 所示。



图 6-3 登录失败页面

登录成功服务页面如图 6-4 所示。



图 6-4 登录成功服务页面

6.3 注册

注册页面如图 6-5 所示。

注册

用户名

密码

身份：学生填0 教师填1 班主任填2 辅导员填3

身份

姓名

学院

班级

专业

保存

返回登录

图 6-5 注册页面

注册成功页面如图 6-6 所示。



图 6-6 注册成功页面

6.4 退出

退出页面位置如图 6-7 所示。



图 6-7 退出页面位置

6.5 用户查询基本信息

用户查询基本信息如图 6-8 所示。



图 6-8 用户查询基本信息

6.6 教师发布签到任务

教师发布签到任务如图 6-9 所示。

发布签到任务

作用班级

签到码

课程名称

时间写法例子：20200424.2222，年月日、时间以"."连接

开始时间

结束时间

提交

返回服务

查询学生的签到状态

图 6-9 教师发布签到任务

教师发布签到任务成功如图 6-10 所示。

发布成功/查学生签到状态

返回服务

查询学生签到状态

以下为已签到同学

学号	姓名	level	专业	班级
----	----	-------	----	----

以下为未签到同学

学号	姓名	level	专业	班级
----	----	-------	----	----

以下为作废记录

学号	姓名	level	专业	班级
----	----	-------	----	----

图 6-10 教师发布签到任务成功

6.7 教师查询学生签到状态

教师查询学生签到状态如图 6-11 所示。

发布成功/查学生签到状态

返回服务

查询学生签到状态

以下为已签到同学

学号	姓名	level	专业	班级
----	----	-------	----	----

以下为未签到同学

学号	姓名	level	专业	班级
166308105	张三	0	安卓基础	17移动应用开发3班
176303301	赵六	0	安卓基础	17移动应用开发3班
176303302	王凯	0	安卓基础	17移动应用开发3班
176303327	王启年	0	安卓基础	17移动应用开发3班
176303322	张无忌	0	安卓基础	17移动应用开发3班
176303323	杨过	0	安卓基础	17移动应用开发3班
100	学生1	0	安卓基础	17移动应用开发3班

以下为作废记录

图 6-11 教师查询学生签到状态

6.8 教师发布通知

教师发布通知如图 6-12 所示。

发通知/作业

标题

正文

覆盖班级

提交

返回服务

查我发布的通知

图 6-12 教师发布通知

发布通知失败如图 6-13 所示。

发布通知失败

发布通知失败，可能的原因如下：

1.您不是教师，不能够发通知

2.存在空值，请重新发布

您可以重新发布通知

标题

正文

覆盖班级

提交

返回服务

查我发布的通知

图 6-13 发布通知失败

6.9 教师查询自己发布的通知

教师查询自己发布的通知如图 6-14 所示。

发布通知成功/查通知

返回服务

查我发布的通知

发布时间	发布者	覆盖班级	标题	正文
2020-04-15 02:08:54	辅导员1	17移动应用开发3班	助学金	关于助学金
2020-04-15 02:08:54	辅导员1	17移动应用开发3班	奖学金	关于奖学金
2020-04-15 02:08:54	辅导员1	17移动应用开发3班	疫情	关于疫情
2020-04-28 20:35:06	辅导员1	17移动应用开发3班	123	7544564564
2020-04-28 20:44:13	辅导员1	17移动应用开发3班	123	测试
2020-04-28 20:50:21	辅导员1	17移动应用开发3班	一个app的测试	一个app的测试
2020-04-28 20:59:25	辅导员1	17移动应用开发3班	疫情期间如何	一个ap

课表 服务 **通知** 我的

图 6-14 教师查询自己发布的通知

6.10 教师查询学生请假记录

教师查询学生请假记录如图 6-15 所示。

查学生请假记录成功							
返回服务							
学工号	姓名	所在班级	所在学院	请假类型	开始时间	结束时间	请假理由
12	熊二	17移动应用开发3班	计算机工程学院	事假	2020-04-24 14:30:00	2020-04-24 15:30:00	测试理由
166308105	张三	17移动应用开发3班	计算机工程学院	事假	2020-04-24 14:30:00	2020-04-24 15:30:00	测试理由
100	学生1	17移动应用开发3班	计算机工程学院	事假	2020-04-24 14:30:00	2020-04-24 15:30:00	测试理由

图 6-15 教师查询学生请假记录

6.11 学生提出请假

学生请假如图 6-16 所示。

学生请假

请假类型

时间写法例子：20200424.2222，年月日、时间以"."连接

开始时间

结束时间

请假理由

保存

查看请假记录

返回服务

图 6-16 学生请假

学生请假失败如图 6-17 所示。

请假失败

返回服务

访问不成功

1. 学生请假填入空值，无法上传数据，请重新填写

2. 教师申请请假：您是教师不用请假

3. 教师查询请假记录，您是教师，不用请假，故没有请假记录

重新申请

请假类型

时间写法例子：20200424.2222，年月日、时间以"."连接

开始时间

结束时间

请假理由

保存

查看请假记录

图 6-17 学生请假失败

学生请假成功如图 6-18 所示。



图 6-18 学生请假成功

6.12 学生查询自己的请假记录

学生查询自己的请假记录如图 6-19 所示。

请假记录							
返回服务							
查看请假记录							
学工号	姓名	所在班级	所在学院	请假类型	开始时间	结束时间	请假理由
100	学生1	17移动应用开发3班	计算机工程学院	事假	2020-04-24 14:30:00	2020-04-24 15:30:00	测试理由
100	学生1	17移动应用开发3班	计算机工程学院	测试录视频	2020-12-20 22:22:00	2020-04-29 22:22:00	测试理由
100	学生1	17移动应用开发3班	计算机工程学院	事假	2020-12-20 22:22:00	2020-04-29 22:22:00	测试理由
100	学生1	17移动应用开发3班	计算机工程学院	w'w'w'w	2020-12-20 22:22:00	2020-04-29 22:22:00	测试理由
100	学生	17移动应用开	计算机工程学	生病了去	2020-12-20	2020-04-29	测试

图 6-19 学生查询自己的请假记录

6.13 学生签到

学生签到如图 6-20 所示。



The image shows a web interface for student sign-in. It features a teal header bar with the text "签到" (Sign-in). Below the header, there are two input fields: the first is labeled "签到码" (Sign-in Code) and the second is labeled "课程名称" (Course Name). Below these fields are two buttons: "提交签到" (Submit Sign-in) and "返回服务" (Return Service).

签到

签到码

课程名称

提交签到

返回服务

图 6-20 学生签到

学生签到失败如图 6-21 所示。

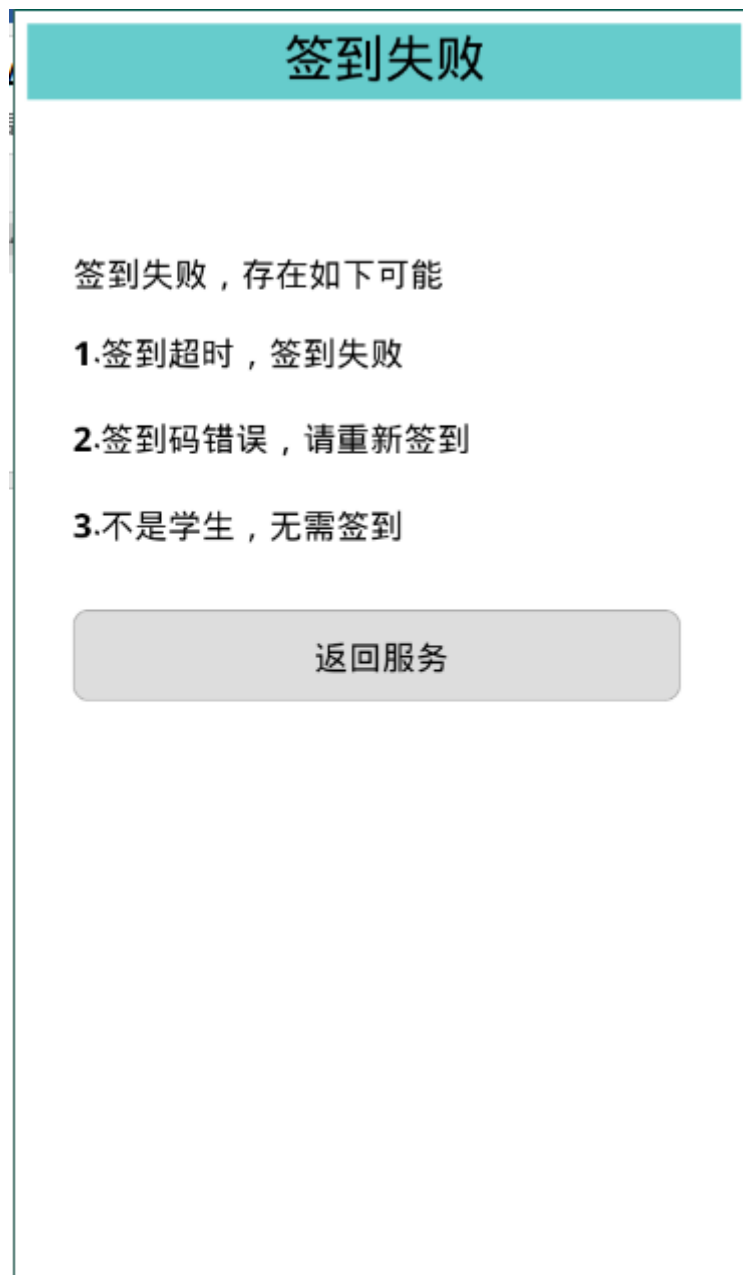


图 6-21 学生签到失败

学生签到成功如图 6-22 所示。



图 6-22 学生签到成功

6.14 学生查询通知

学生查询通知如图 6-23 所示。

通知				
查通知				
发布时间	发布者	覆盖班级	标题	正文
2020-04-15 02:08:54	辅导员1	17移动应用 开发3班	助学金	关于助学金
2020-04-15 02:08:54	辅导员1	17移动应用 开发3班	奖学金	关于奖学金
2020-04-15 02:08:54	辅导员1	17移动应用 开发3班	疫情	关于疫情
2020-04-26 20:29:46	班主任1	17移动应用 开发3班	开学	关于开学
2020-04-26 20:32:21	班主任1	17移动应用 开发3班	作业	关于作业
2020-04-28 20:35:06	辅导员1	17移动应用 开发3班	123	7544564564
2020-04-28 20:44:13	辅导员1	17移动应用 开发3班	123	测试
2020-04-28 20:50:21	辅导员1	17移动应用 开发3班	一个app 的测试	一个app的 测试
2020-04-28	辅导	17移动应用	疫情期间	一个ap
课表		服务	通知	我的

图 6-23 学生查询通知

6.15 服务器定时任务

服务器定时任务如图 6-24 所示。

```
-----定时任务 每45分钟一次-----
|
|                                当前系统时间为2020-04-30 10:49:30
|                                已经将学生签到表的数据更新
|
|-----|
[2020-04-30 10:49:31,097] Artifact WF_KQGL_Server:war exploded: Artifact is deployed successfully
[2020-04-30 10:49:31,097] Artifact WF_KQGL_Server:war exploded: Deploy took 12,952 milliseconds
```

图 6-24 服务器定时任务

6.16 服务器拦截器

服务器拦截器如图 6-25 所示。

```
拦截器:检测前端请求为登录或注册, 拦截通过
表现层: 有人想要登录了
业务层: 有人登录, 正在验证账号和密码
辅导员: 123 登录成功
用户信息:123

-----

拦截器:该用户已经登录,通过拦截,用户为:123
{kebiao=[KeBiao{username='123', name='辅

-----

拦截器:该用户已经登录,通过拦截,用户为:123
用户是教师, 跳转到教师看通知的页面

-----

拦截器:该用户已经登录,通过拦截,用户为:123
用户是教师, 跳转到教师看通知的页面

-----

拦截器:该用户已经登录,通过拦截,用户为:123
123跳转回登录成功界面
```

图 6-25 服务器拦截器

总 结

一个小小的考勤管理系统便花费了我不少时间，然而社会需要的优秀的系统比我的要复杂无数倍，而没有竞争力的系统终将被淘汰出局。独木难支，想要能够开发出一套优秀的系统，需要很多人的不懈努力，为了将来能参与开发优秀系统，我在毕业后仍需不断学习，毕竟摩尔定律仍是事实，技术的革新是很快的。

骐骥千里,非一日之功。踏入社会后，我当戒骄戒躁，不当急功近利，毕竟没有原始积累，哪怕是因机缘巧合取得成就，也会因基础不扎实，而体验到社会的毒打，毕竟站的越高摔得越惨；且德不配位自然而然地会受到同事的冷言，只有身怀足够的实力，才能服众。

路在脚下，走便是。

致 谢

三年时间不经意间就过去了，在这三年里，我由幼稚缓慢成长，在周围人的见证下，我由对移动应用开发这一方面从门外汉到慢慢入行。(感谢不分先后)

感谢我的父母，让我能接收这三年的教育。让我能有机会看见更广阔的天，让我有机会接触更精彩的世界。

要感谢我的班主任，他脾气很好，经常耐心地为我们的解惑，也常常走到我们的课桌前帮助我们解决编程中遇到的 Bug；他虽然平时很忙，但是每次学校有事，他都会亲自开班会，事事为我们考虑，以我们为重。

感谢我教我安卓的赵老师，她是为我们授课时间最长的教师，她脾气很好，上课时常常为我们解惑，也常常走到我们的课桌前帮助我们解决编程中遇到的 Bug。

感谢我的各科老师为我大学生涯给予的授业解惑，你们的阅历比我多，所以有时你们的一句话或者行动都会给我带来不一样的体会，感谢你们。

祝大家身体健康，家庭幸福美满，财运亨通。

参考文献:

- [1] 赵永杰, 马宝龙, 包国强. 基于 SSM 书香驿站平台的设计与实现[J]. 农家参谋, 2020(06):188.
- [2] 郭玉芝, 周太宇. 基于 SSM 框架的高校学生平时成绩管理系统的设计与实现[J]. 现代信息技术, 2019, 3(23):17-19.
- [3] 甘启宏, 崔亚强, 王皓, 余淇, 冯鸟东. 基于 SSM 框架的高校教学设备管理系统设计与实现[J]. 软件导刊, 2020, 19(02):44-47.
- [4] 马梓昂, 贾克斌. 基于 Web 的高性能智能快递柜管理系统[J]. 计算机应用与软件, 2020, 37(04):1-5+47.
- [5] 彭兵. 浅谈 MVC 设计模式在 JSP 程序中的应用[J]. 信息与电脑(理论版), 2019(11):104-105.
- [6] 冯涛, 李朋, 张金芳, 孙晓磊, 李亚娟. 高校学生考勤管理系统开发中的关键技术研究[J]. 河北水利电力学院学报, 2019(04):32-39.
- [7] 段修亮, 赵霞. 小型企业人事管理系统的设计与实现[J]. 科技风, 2020(01):76.
- [8] 郑银环. 智慧课堂学生考勤管理研究与设计[J]. 现代信息技术, 2019, 3(09):1-4+7.
- [9] 李惠, 杨加林. 基于移动终端的高校学生考勤管理系统的设计与实现[J]. 福建电脑, 2018, 34(07):114-115.