**基于WEB的设备信息化管理系统**

# 摘要

本文论述了一个基于SpringBoot、MyBatis、Shiro、Thymeleaf、BootStrap框架，maven架构搭建，并运用MySql数据库连接的设备信息化管理系统网站的数据库设计与搭建，系统使用了JAVA、SQL、XML、HTML、JavaScript等开发语言进行整体设计与开发。

本系统设计根据设备的借记流程，其中包括设备登记，设备查看，设备借出以及设备收回的全部流程，从而更加便捷与准确的记录了设备的全部使用去向。Web系统包括管理后台端与用户展示借记端，其中管理后台包括的主要功能有，管理角色、管理权限、用户管理、通知公告、操作日志、设备管理、设备类型管理、借记管理、借记统计。用户展示端功能主要有设备查看、公告模块、个人中心、借记功能模块。

本系统开发从需求调研、项目设计、原型设计、软件开发、功能测试、项目部署等方面进行全面管理。

**关键词：**web设备信息化管理系统；Java；idea；SpringBoot，HTML

目录

[**基于WEB的设备信息化管理系统** 1](#_Toc64843975)

[摘要 1](#_Toc64843976)

[绪论 1](#_Toc64843977)

[1 开发环境及相关介绍 2](#_Toc64843978)

[1.1开发工具简介 2](#_Toc64843979)

[1.2开发环境及安装过程 2](#_Toc64843980)

[1.3 服务器环境 2](#_Toc64843981)

[2 需求分析及功能介绍 3](#_Toc64843982)

[2.1用户登录 4](#_Toc64843983)

[2.2 用户管理 4](#_Toc64843984)

[2.3 类别管理 4](#_Toc64843985)

[2.4 设备管理 4](#_Toc64843986)

[2.5 借用管理 4](#_Toc64843987)

[2.6 公告管理 4](#_Toc64843988)

[2.7 角色管理 4](#_Toc64843989)

[2.8 权限管理 4](#_Toc64843990)

[2.9 日志管理 4](#_Toc64843991)

[2.10 个人中心 4](#_Toc64843992)

[3 部分功能模块详细设计 5](#_Toc64843993)

[3.1用户登录 5](#_Toc64843994)

[3.2 设备管理 12](#_Toc64843995)

[3.3设备类型管理 13](#_Toc64843996)

[3.3公告模块 14](#_Toc64843997)

[3.4个人中心 16](#_Toc64843998)

# 绪论

设备管理是以设备为研究对象，追求设备综合效率，应用一系列理论、方法，通过一系列技术、经济、组织措施，对设备的物质运动和价值运动进行全过程（从规划、设计、选型、购置、安装、验收、使用、保养、维修、改造、更新直至报废）的科学型管理。

将设备管理迁移至在线设备信息化管理，能更加高效、便捷、准确的行使管理流程。

本软件是浏览器/服务器模式（B/S）模式设计，数据库是运行在服务器端，和软件后台运行在同一个操作系统环境下。后台运行环境基本要求：Linux服务器，CentOS7系统，内存建议在在4G以，上处理器为当前主流CPU环境即可。硬盘存储空间在20G以上（目的是为了保留以后多年存储的数据）。对于网络要求需要联网上传数据或则通过别的途径传递数据。前端运行环境基本要求：对操作系统不做要求，兼容市场主流浏览器，建议使用浏览器新版本，对于网络要求需要联网上传数据或则通过别的途径传递数据。

开发软件运行环境：Java开发语言解决复杂业务开发，效率较高，数据安全，接口处理较快捷等特点。需要安装运行JDK8.0以上。

整体架构搭建使用SpringBoot开源框架，SpringBoot是有Pivotal团队提供固定全新开源框架，其设计的目的是采用简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而是开发人员不在需要定义样板化的配置。通过这种方式，SrpingBoot致力于在蓬勃发展的快速应用开发领域成为领导者，相较于普通的SSM，SSH等成熟框架，其整体配置大大缩小。

本系统采用Shiro权限管理，Shiro是Apache开源安全框架，执行身份验证、授权、密码和会话管理。使用Shiro的易于理解的API可以快速、轻松的移植至任何应用程序，从最小的移动应用程序到最大的网站和企业应用程序，而且相较于Spring Security比较，配置更加便捷，更加轻量级。

数据库使用较为流行的MySql，易于数据库的备份恢复。MySQL是一个关系行数据库管理系统，有瑞典MySQLAB公司开发，现属于Oracle（甲骨文）旗下产品。MYSQL是最流行的关系型数据库之一，在web应用方面，MySQL是最好的RDBMS应用软件之一，相较于Oracle,SQL Server，DB2等数据库，MySQL更加简单方便。

为防止服务器宕机，数据丢失，系统崩溃等情况，建议使用两台服务器作为系统运行服务器，由Ngnix做负载均衡，并经常备份数据库数据，服务器也应经常做线路检查等常规检查，以避免因设备等外部原因导致服务器宕机。

# 1 开发环境及相关介绍

1.1开发工具简介

（1）IDEA 全称 IntelliJ IDEA，是java编程语言开发的集成环境。IntelliJ在业界被公认为最好的java开发工具，尤其在智能代码助手、代码自动提示、重构、JavaEE支持、各类版本工具(git、svn等)、JUnit、CVS整合、代码分析、 创新的GUI设计等方面的功能可以说是超常的。IDEA是JetBrains公司的产品，这家公司总部位于捷克共和国的首都布拉格，开发人员以严谨著称的东欧程序员为主。

（2）SQLyog 是一个快速而简洁的图形化管理MYSQL数据库的工具，它能够在任何地点有效地管理你的数据库，由业界著名的Webyog公司出品。

使用SQLyog可以快速直观地让您从世界的任何角落通过网络来维护远端的MySQL数据库。

1.2开发环境及安装过程

(1) JDK的安装。在Oracle官网下载最新JDK。安装后配置环境变量，系统变量→新建 JAVA\_HOME 变量。变量值填写jdk的安装目录。系统变量→寻找 Path 变量→编辑%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;

下载地址：

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

(2) MySql的安装。在https://dev.mysql.com/downloads/installer/下载mysql-installer-web-community-8.0.13.0.msi，点击即可安装。在Navicat里面选择mmap.sql数据库文件，进行数据库的使用。

1.3 服务器环境

(1)Linux系统安装。使用CentOS7系统搭建，CentOS是免费的、开源的、可以重新分发的开源操作系统，CentOS（Community Enterprise Operating System，中文意思是社区企业操作系统）是Linux发行版之一。

CentOS Linux发行版是一个稳定的，可预测的，可管理的和可复现的平台，源于Red Hat Enterprise Linux（RHEL）依照开放源代码（大部分是GPL开源协议）规定释出的源码所编译而成。

# 2 需求分析及功能介绍

为了使本系统更加人性化，便捷化，可操作性强，在参考了大量相关网站之后，制定了本系统的流程图。如图2.1

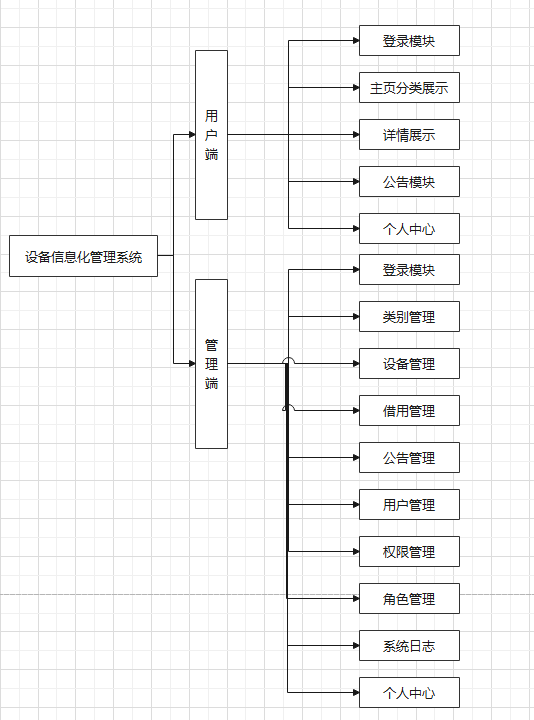


图2.1

系统首次初始化，将分配一名超级管理员，用户名：admin，密码：123456，管理员可使用该账号登陆管理端操作系统。

超级管理员拥有全平台所有权限，可设置普通管理员或者其他管理角色进行功能管理，其中主要功能包括：设置设备类型，添加设备并进行管理，设备借用管理，添加系统公告，权限管理，用户角色管理，系统日志查看，个人信息修改设置，用户管理。

普通用户通过用户名密码登陆后，可进入个人中心修改密码。在用户端登陆后，可进行设备查看，设备借用，查看系统公告等操作。

各项功能具体功能和操作如下。

2.1用户登录

系统管理员可通过初始用户名、密码登陆后台管理系统，登陆失败则提示用户名密码错误。普通用户通过管理员分配的用户名及密码登陆用户端系统，由于本系统为内部设备借用管理系统，使用人员均为指定人群，所以不允许外部注册，所有用户均有管理员统一创建并发放用户名和密码。

2.2 用户管理

管理员登陆后，可在用户管理中添加用户信息，设置用户名和密码。管理员可对用户进行删除操作，以停止该用户继续使用权。可对用户设置用户的角色，用以分配管理员权限等操作。如用户忘记密码，可由管理员将密码重置，管理员重置密码无需验证操作，直接输入新密码即可。

2.3 类别管理

管理员通过类别管理设置设备的所属类别，可对类别进行增、删、改、查操作，其中删除功能可批量进行。

2.4 设备管理

设备管理为系统主功能，管理员通过设备管理对设备进行增、删、改查操作，增添设备时，管理员可设置该设备的名称、类别以及设备数量，数量为0则用户无法继续借用，设备归还则数量恢复。设备报销即逻辑删除，该设备将不可借用。

用户可在用户端页面设备管理中查询借用设备列表，可借用设备详情，借用，归还等操作。

2.5 借用管理

管理员可通过借用管理对普通用户的借用申请进行审批，审批通过后，用户方可对设备进行借走使用。

2.6 公告管理

管理员可在公告管理中添加公告，公告讲在用户端首页展示公告列表，普通用户可进行查看。公告为管理员自定义页面展示效果，可调整字体、颜色、插入图片等操作。

2.7 角色管理

管理员可添加用户角色，与权限管理共同使用。

2.8 权限管理

管理员可为角色添加权限，不同权限用户则拥有不同的使用权。

2.9 日志管理

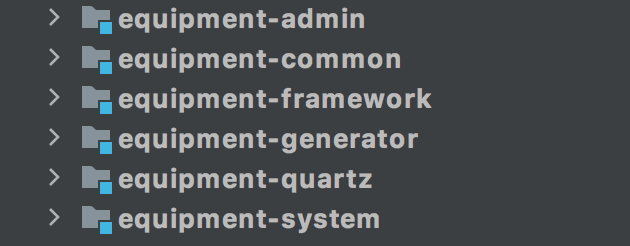
管理员可在日志管理中查看用户的登录日志和操作日志，方便对使用者的监督和管理

2.10 个人中心

管理员通过个人中心可以修改个人的头像、密码等信息。

# 3 部分功能模块详细设计

代码整体分为五个模块，如图所示



其中equipment-admin模块为controller层，即视图层，html的页面也在这里书写。Equipment-system为service层和dao层，即业务逻辑层和持久化层，equipment-common为工具类模块，equipment-framework为权限模块，shiro框架在这里搭建，数据出入也由此模块进行校验。Equipment-generator是自动生成代码模块，此模块书写了自动生成代码的逻辑，由此模块逻辑可快速生成整套预设逻辑代码，方便快速。Equipment-quartz为定时模块，此模块为预留功能模块，如需要定时任务，则可在此模块进行快速引入。

以下为部分功能模块的详细设计。

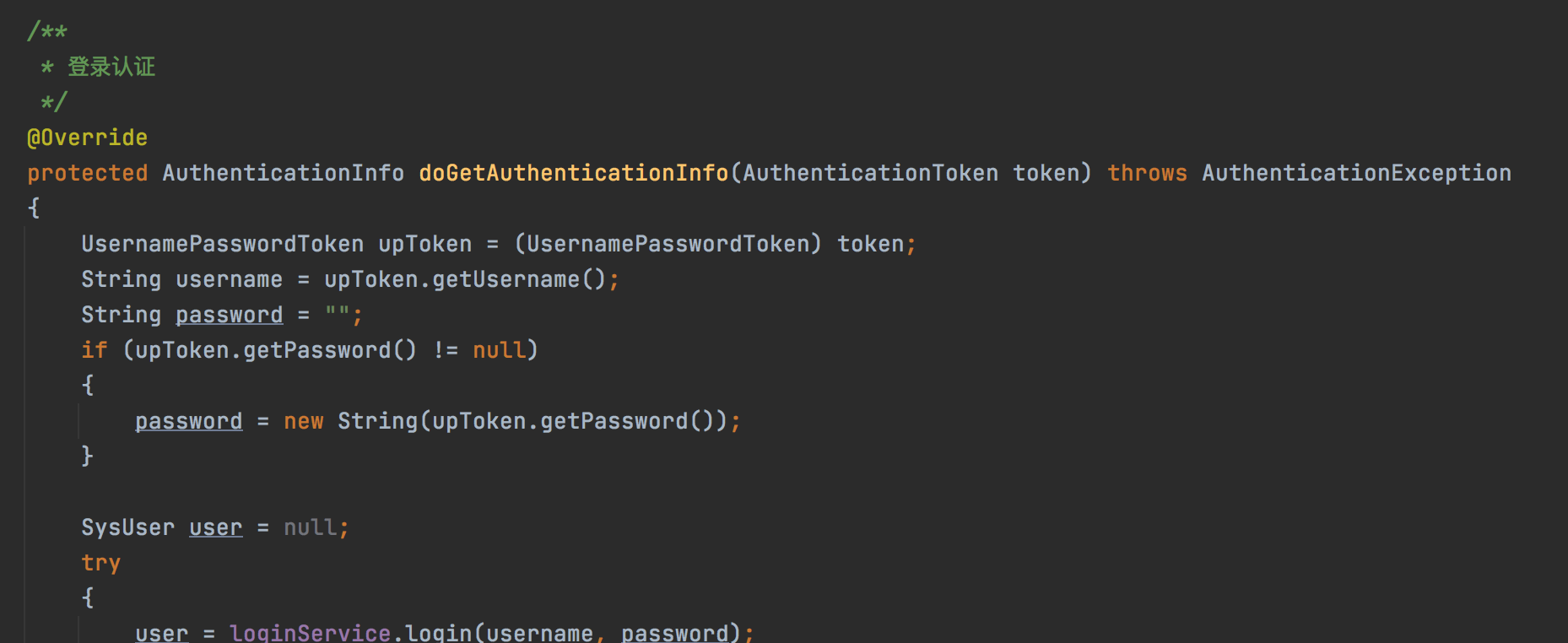
3.1用户登录

1)数据库中user表的信息

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Comment** |
| user\_id | bigint(20) | 用户ID |
| dept\_id | bigint(20) | 部门ID |
| login\_name | varchar(30) | 登录账号 |
| user\_name | varchar(30) | 用户昵称 |
| user\_type | varchar(2) | 用户类型（00系统用户 01注册用户） |
| email | varchar(50) | 用户邮箱 |
| phonenumber | varchar(11) | 手机号码 |
| sex | char(1) | 用户性别（0男 1女 2未知） |
| avatar | varchar(100) | 头像路径 |
| password | varchar(50) | 密码 |
| salt | varchar(20) | 盐加密 |
| status | char(1) | 帐号状态（0正常 1停用） |
| del\_flag | char(1) | 删除标志（0代表存在 2代表删除） |
| login\_ip | varchar(50) | 最后登录IP |
| login\_date | datetime | 最后登录时间 |
| pwd\_update\_date | datetime | 密码最后更新时间 |
| create\_by | varchar(64) | 创建者 |
| create\_time | datetime | 创建时间 |
| update\_by | varchar(64) | 更新者 |
| update\_time | datetime | 更新时间 |
| remark | varchar(500) | 备注 |

2)代码中所对应信息

用户通过映射sysuser类对象存取数据，出入信息由shiro进行校验整理，密码加密采用salt加密（盐加密）方式，增加保密强度。



从token中获取用户信息，校验用户的时效性，获取数据库用用户盐信息，并校验密码正确性，从获取信息中校验用户的使用权限，多重验证，保证用户和数据信息安全。

3)相关前台窗口

用户登录如图3-1



图3-1用户登录

管理员查看可修改用户信息如图3-2



图3-2 用户管理

管理员管理用户角色和权限如图3-3和3-4

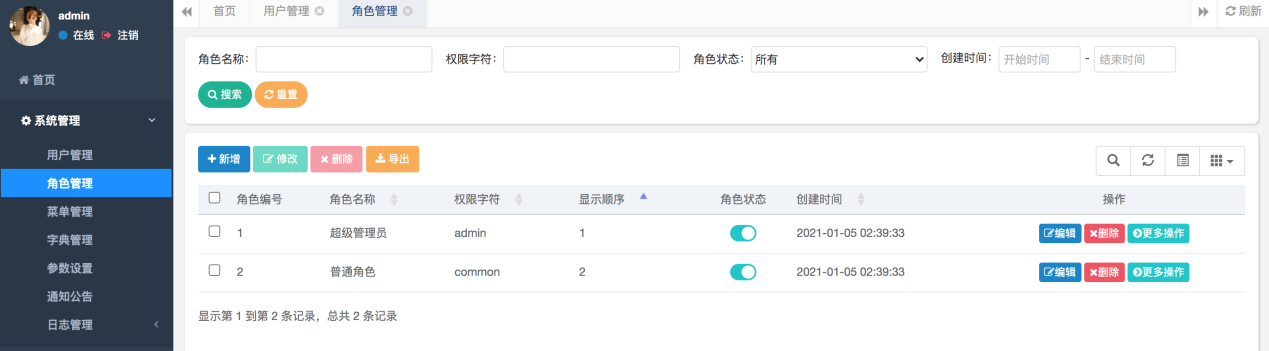


图3-3 角色管理

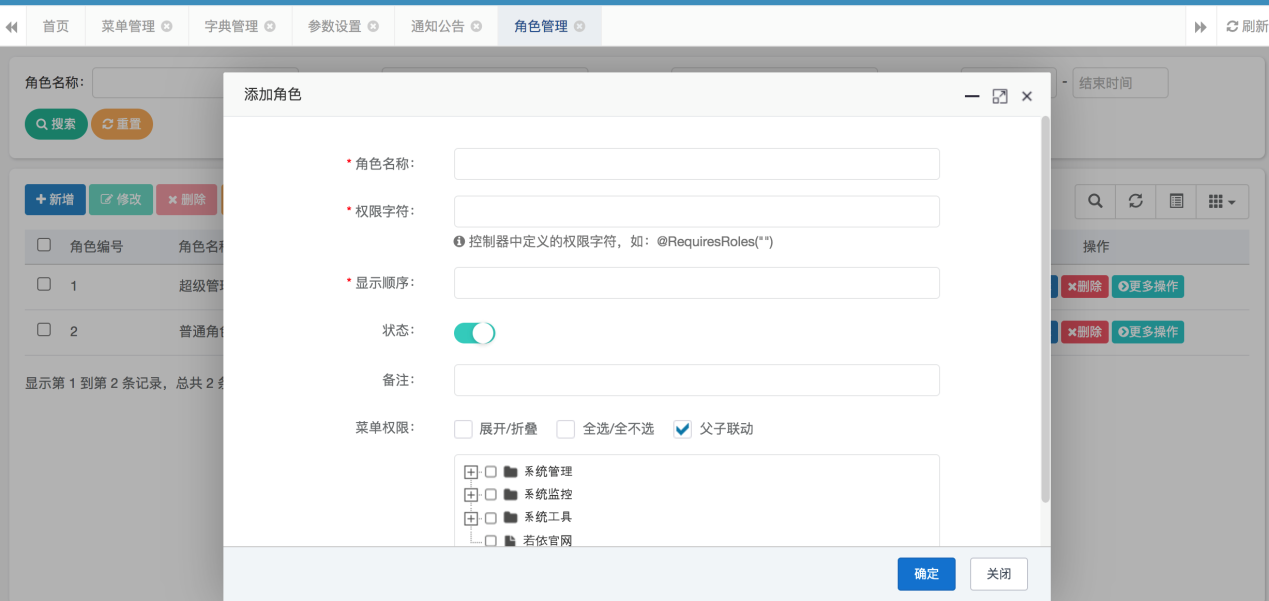


图3-3 新建角色

普通角色主页展示如图3-4

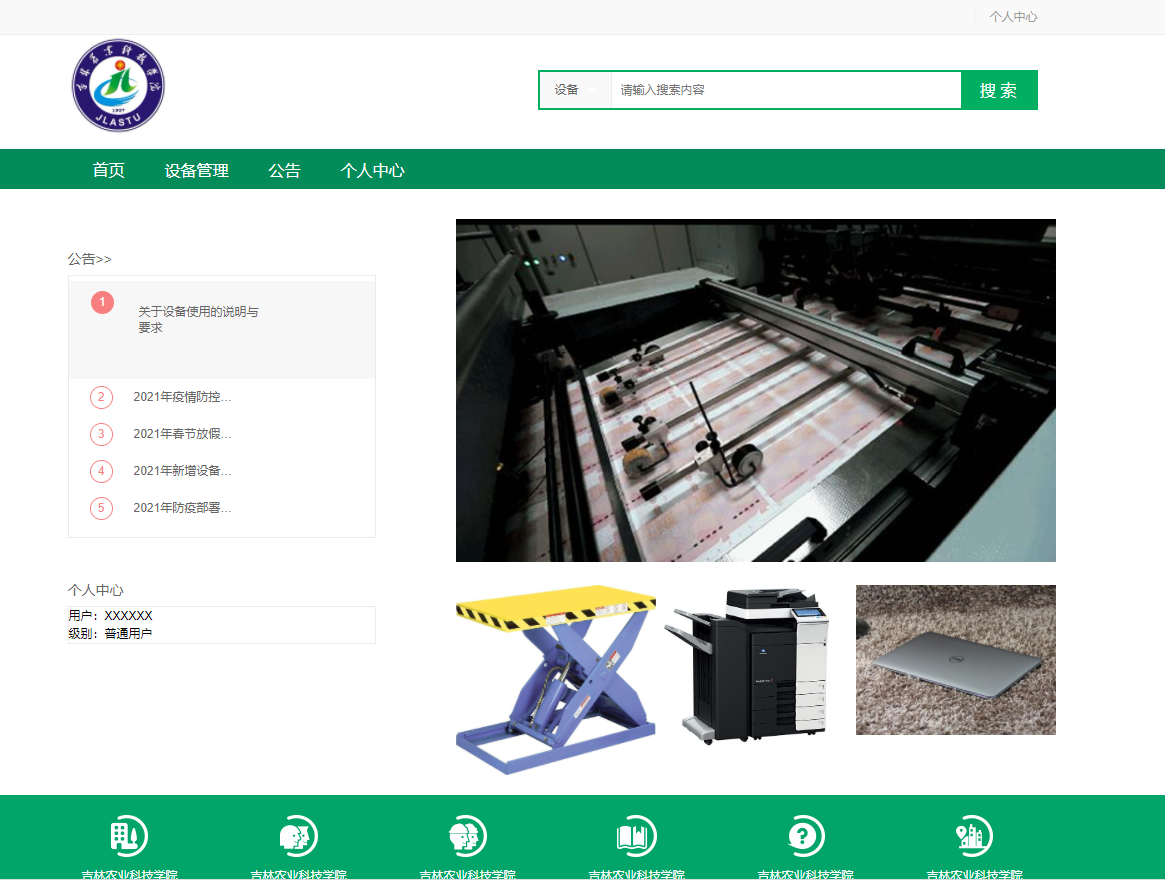


图3-4

附权限验证部分代码逻辑

代码1：

@Bean  
**public** ShiroFilterFactoryBean shiroFilterFactoryBean(SecurityManager securityManager)  
{  
 ShiroFilterFactoryBean shiroFilterFactoryBean = **new** ShiroFilterFactoryBean();  
 *// Shiro的核心安全接口,这个属性是必须的* shiroFilterFactoryBean.setSecurityManager(securityManager);  
 *// 身份认证失败，则跳转到登录页面的配置* shiroFilterFactoryBean.setLoginUrl(**loginUrl**);  
 *// 权限认证失败，则跳转到指定页面* shiroFilterFactoryBean.setUnauthorizedUrl(**unauthorizedUrl**);  
 *// Shiro连接约束配置，即过滤链的定义* LinkedHashMap<String, String> filterChainDefinitionMap = **new** LinkedHashMap<>();  
 *// 对静态资源设置匿名访问* filterChainDefinitionMap.put(**"/favicon.ico\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/equipment.png\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/css/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/docs/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/fonts/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/door/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/door/\*\*/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/img/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/ajax/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/js/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/equipment/\*\*"**, **"anon"**);  
 filterChainDefinitionMap.put(**"/captcha/captchaImage\*\*"**, **"anon"**);  
 *// 退出 logout地址，shiro去清除session* filterChainDefinitionMap.put(**"/logout"**, **"logout"**);  
 *// 不需要拦截的访问* filterChainDefinitionMap.put(**"/login"**, **"anon,captchaValidate"**);  
 filterChainDefinitionMap.put(**"/main/index"**, **"anon,captchaValidate"**);  
 *// 注册相关* filterChainDefinitionMap.put(**"/register"**, **"anon,captchaValidate"**);  
 *// 系统权限列表  
 // filterChainDefinitionMap.putAll(SpringUtils.getBean(IMenuService.class).selectPermsAll());* Map<String, Filter> filters = **new** LinkedHashMap<String, Filter>();  
 filters.put(**"onlineSession"**, onlineSessionFilter());  
 filters.put(**"syncOnlineSession"**, syncOnlineSessionFilter());  
 filters.put(**"captchaValidate"**, captchaValidateFilter());  
 filters.put(**"kickout"**, kickoutSessionFilter());  
 *// 注销成功，则跳转到指定页面* filters.put(**"logout"**, logoutFilter());  
 shiroFilterFactoryBean.setFilters(filters);  
  
 *// 所有请求需要认证* filterChainDefinitionMap.put(**"/\*\*"**, **"user,kickout,onlineSession,syncOnlineSession"**);  
 shiroFilterFactoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);  
  
 **return** shiroFilterFactoryBean;  
}

代码2：

*/\*\*  
 \* 登录  
 \*/***public** SysUser login(String username, String password)  
{  
 *// 验证码校验* **if** (ShiroConstants.***CAPTCHA\_ERROR***.equals(ServletUtils.*getRequest*().getAttribute(ShiroConstants.***CURRENT\_CAPTCHA***)))  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"user.jcaptcha.error"**)));  
 **throw new** CaptchaException();  
 }  
 *// 用户名或密码为空 错误* **if** (StringUtils.*isEmpty*(username) || StringUtils.*isEmpty*(password))  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"not.null"**)));  
 **throw new** UserNotExistsException();  
 }  
 *// 密码如果不在指定范围内 错误* **if** (password.length() < UserConstants.***PASSWORD\_MIN\_LENGTH*** || password.length() > UserConstants.***PASSWORD\_MAX\_LENGTH***)  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"user.password.not.match"**)));  
 **throw new** UserPasswordNotMatchException();  
 }  
  
 *// 用户名不在指定范围内 错误* **if** (username.length() < UserConstants.***USERNAME\_MIN\_LENGTH*** || username.length() > UserConstants.***USERNAME\_MAX\_LENGTH***)  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"user.password.not.match"**)));  
 **throw new** UserPasswordNotMatchException();  
 }  
  
 *// 查询用户信息* SysUser user = **userService**.selectUserByLoginName(username);**if** (user == **null**)  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"user.not.exists"**)));  
 **throw new** UserNotExistsException();  
 }  
   
 **if** (UserStatus.***DELETED***.getCode().equals(user.getDelFlag()))  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"user.password.delete"**)));  
 **throw new** UserDeleteException();  
 }  
   
 **if** (UserStatus.***DISABLE***.getCode().equals(user.getStatus()))  
 {  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_FAIL***, MessageUtils.*message*(**"user.blocked"**, user.getRemark())));  
 **throw new** UserBlockedException();  
 }  
  
 **passwordService**.validate(user, password);  
  
 AsyncManager.*me*().execute(AsyncFactory.*recordLogininfor*(username, Constants.***LOGIN\_SUCCESS***, MessageUtils.*message*(**"user.login.success"**)));  
 recordLoginInfo(user);  
 **return** user;  
}

代码3：

*/\*\*  
 \* 获取会话信息  
 \*  
 \** ***@param sessionId*** *\** ***@return*** *\*/***public** Session getSession(Serializable sessionId)  
{  
 SysUserOnline userOnline = **onlineService**.selectOnlineById(String.*valueOf*(sessionId));  
 **return** StringUtils.*isNull*(userOnline) ? **null** : createSession(userOnline);  
}  
  
**public** Session createSession(SysUserOnline userOnline)  
{  
 OnlineSession onlineSession = **new** OnlineSession();  
 **if** (StringUtils.*isNotNull*(userOnline))  
 {  
 onlineSession.setId(userOnline.getSessionId());  
 onlineSession.setHost(userOnline.getIpaddr());  
 onlineSession.setBrowser(userOnline.getBrowser());  
 onlineSession.setOs(userOnline.getOs());  
 onlineSession.setDeptName(userOnline.getDeptName());  
 onlineSession.setLoginName(userOnline.getLoginName());  
 onlineSession.setStartTimestamp(userOnline.getStartTimestamp());  
 onlineSession.setLastAccessTime(userOnline.getLastAccessTime());  
 onlineSession.setTimeout(userOnline.getExpireTime());  
 }  
 **return** onlineSession;  
}

3.2 设备管理

1)数据库中equipment设备表的信息

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Comment** |
| id | bigint(20) | 设备id |
| name | varchar(30) | 设备名称 |
| type\_id | bigint(20) | 设备类型id |
| model\_num | varchar(30) | 设备型号 |
| quantity | int(10) | 设备数量 |
| flag | int(1) | 设备状态：0正常；1报销 |
| created\_at | datetime |  |
| created\_by | varchar(20) |  |
| updated\_at | datetime |  |
| updated\_by | varchar(20) |  |

2)代码中所对应信息

对应代码实体类对象Equipment.java，通过页面视图的用户操作，完成从Controller到Dao的增删改查功能。

3)相关操作界面如图所示

列表展示如图3-5，增加设备如图3-6

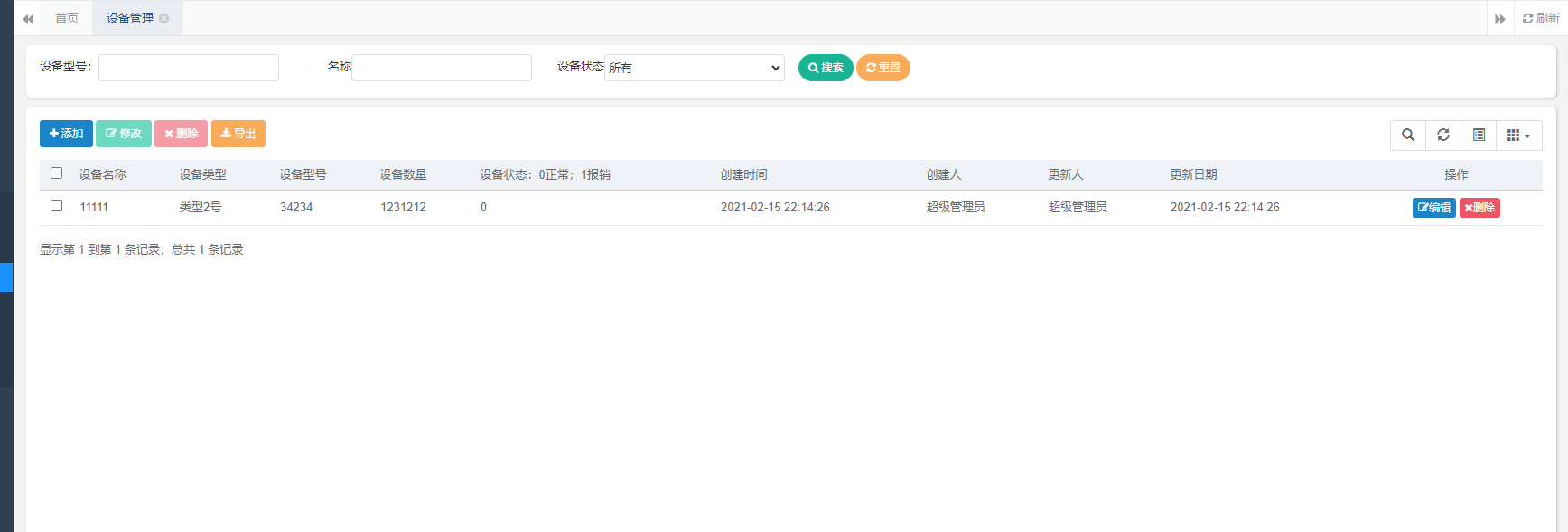


图3-5列表展示

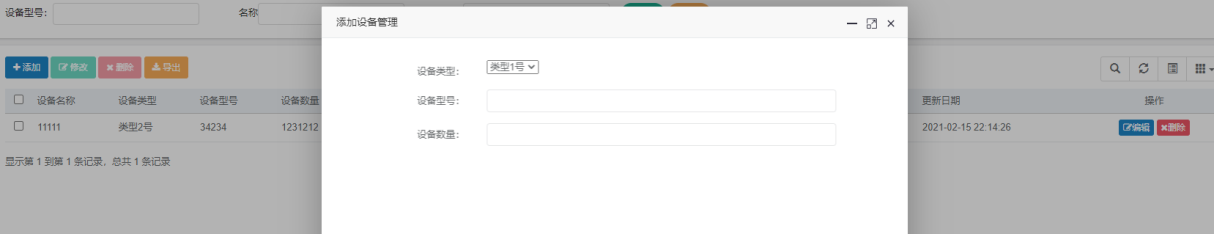


图3.6设备添加

附部分查询SQL

代码1：

<**sql id="selectEquipmentVo"**>  
 select id,name, type\_id, model\_num, quantity, flag, created\_at, created\_by, updated\_at, updated\_by from equipment  
</**sql**>

代码2：

<**select id="selectEquipmentList" parameterType="Equipment" resultMap="EquipmentResult"**>  
 <**include refid="selectEquipmentVo"**/>  
 <**where**>   
 <**if test="typeId != null "**> and type\_id = #{typeId}</**if**>  
 <**if test="name != null and name != ''"**> and name like concat('%', #{name}, '%')</**if**>  
 <**if test="modelNum != null and modelNum != ''"**> and model\_num = #{modelNum}</**if**>  
 <**if test="quantity != null "**> and quantity = #{quantity}</**if**>  
 <**if test="flag != null and flag != -1"**> and flag = #{flag}</**if**>  
 <**if test="createdAt != null "**> and created\_at = #{createdAt}</**if**>  
 <**if test="createdBy != null and createdBy != ''"**> and created\_by = #{createdBy}</**if**>  
 <**if test="updatedAt != null "**> and updated\_at = #{updatedAt}</**if**>  
 <**if test="updatedBy != null and updatedBy != ''"**> and updated\_by = #{updatedBy}</**if**>  
 </**where**>  
</**select**>

代码3：

<**select id="selectEquipmentById" parameterType="Long" resultMap="EquipmentResult"**>  
 <**include refid="selectEquipmentVo"**/>  
 where id = #{id}  
</**select**>

3.3设备类型管理

1)数据库中equipment\_type表的信息

|  |  |  |
| --- | --- | --- |
| Field | Type | Comment |
| id | bigint(20) | 设备类型id |
| type\_name | varchar(50) | 设备类型名称 |
| deleted | int(1) | 0正常；1删除 |
| created\_at | datetime |  |
| created\_by | varchar(20) |  |
| updated\_at | datetime |  |
| updated\_by | varchar(20) |  |

2)代码中所对应信息

EquipmentType.java类

用于数据库中表的数据实体化

通过MyBatis配置xml获取表数据

3)相关前台窗口

文件展示如图3-7



图3.7

增加类型如图3-8

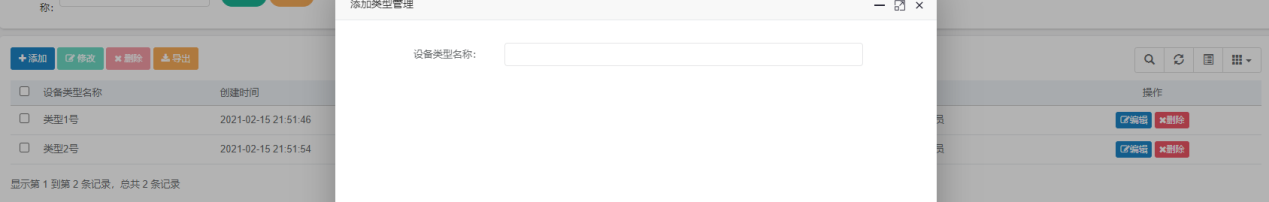


图3-8

用户端可进行借还操作，如图3-9

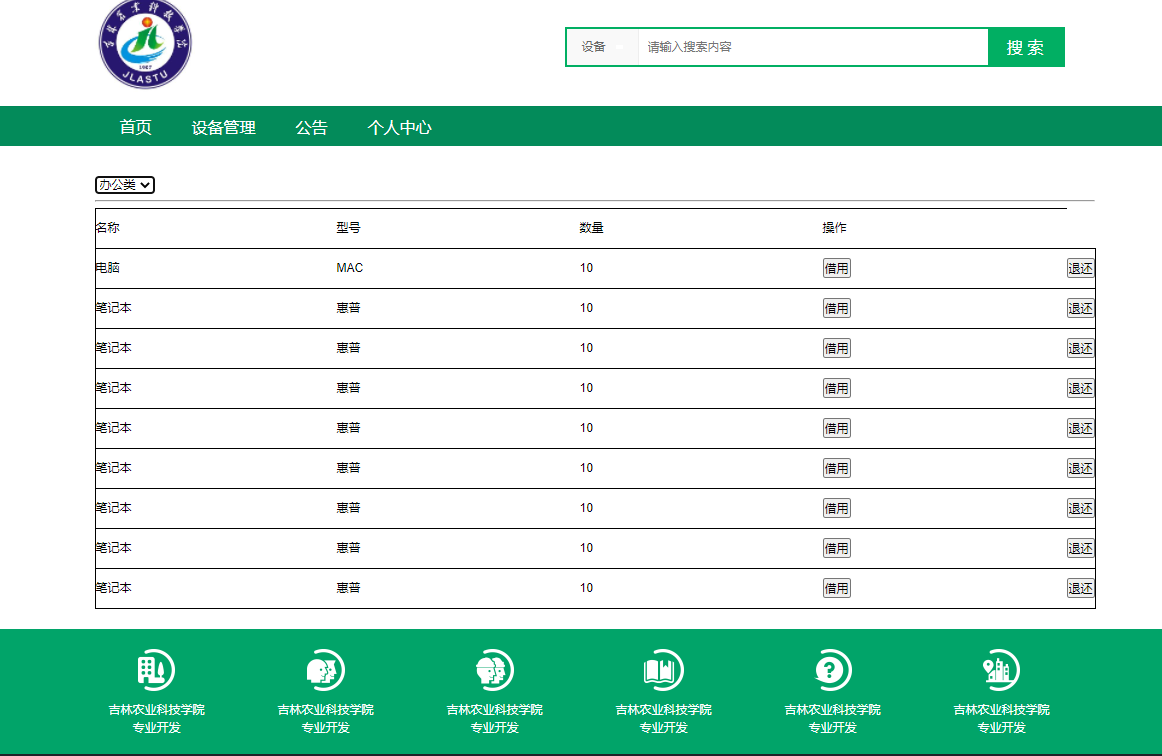


图3-9

3.3公告模块

1)数据库中sys\_notice表的信息

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Comment** |
| notice\_id | int(4) | 公告ID |
| notice\_title | varchar(50) | 公告标题 |
| notice\_type | char(1) | 公告类型（1通知 2公告） |
| notice\_content | varchar(2000) | 公告内容 |
| status | char(1) | 公告状态（0正常 1关闭） |
| create\_by | varchar(64) | 创建者 |
| create\_time | datetime | 创建时间 |
| update\_by | varchar(64) | 更新者 |
| update\_time | datetime | 更新时间 |
| remark | varchar(255) | 备注 |

2)代码中所对应信息

公告模块实体类SysNotice.java，管理员可通过页面操作，增加公告信息，用户端可在页面查看公告。公告栏为富文本编辑器，用户可自定义公告样式，调整字体大小颜色，插入图片等操作。

显示前台页面

3)相关前台窗口

展示如图3-10

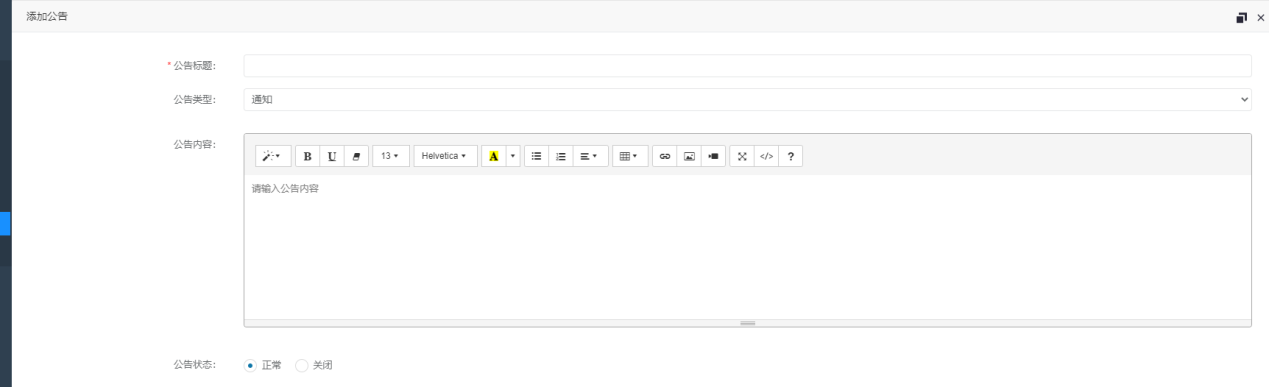


图3-10

用户端可进行可进行借还操作，如图3-11



图3-11

3.4个人中心

1)用户可在个人中心修改用户个人信息，其中包括修改密码，上传头像等操作；

