基于SSM框架的牧畜养殖场管理系统的设计与实现

【原文对照报告-大学生版】

报告编号：39e528f9a086cd63　　检测时间：2019-04-23 17:47:37　　检测字数：10,735字

作者名称：任超　　所属单位：

检测范围：

◎ 中文科技期刊论文全文数据库
◎ 中文主要报纸全文数据库
◎ 中国专利特色数据库
◎ 博士/硕士学位论文全文数据库
◎ 中国主要会议论文特色数据库
◎ 港澳台文献资源
◎ 外文特色文献数据全库
◎ 维普优先出版论文全文数据库
◎ 互联网数据资源/互联网文档资源
◎ 高校自建资源库
◎ 图书资源
◎ 古籍文献资源
◎ 个人自建资源库
◎ 年鉴资源
◎ IPUB原创作品

时间范围：1989-01-01至2019-04-23

检测结论：

全文总相似比		复写率		他引率		自引率		专业术语
9.41%	=	8.26%	+	0.00%	+	0.00%	+	1.15%

其他指标：

自写率：90.59%

专业用语：1.15%

高频词：管理,进行,数据,系统,用户

典型相似性：　无

指标说明：

复写率：相似或疑似重复内容占全文的比重

他引率：引用他人的部分占全文的比重，请正确标注引用

自引率：引用自己已发表部分占全文的比重，请正确标注引用

自写率：原创内容占全文的比重

专业用语：公式定理、法律条文、行业用语等占全文的比重

典型相似性：相似或疑似重复内容占互联网资源库的比重，超过60%可以访问

总相似片段：51

期刊：7 博硕：28 外文：0 综合：3 自建库：0 互联网：13

安阳师范学院本科学生毕业论

基于SSM框架的牧畜养殖场管理系统的设计与实现

作　者　　任超

系（院）　　软件学院

专　　业　　软件工程(java方向)

年　　级　　2015 级

学　　号　　150906078

指导教师　　苏静

成 绩

日　　期　　2019年5 月

诚信承诺书

郑重承诺：所呈交的论文是作者个人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得安阳师范学院或其他教育机构的学位或证书所使用过的材料。与作者一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

作者签名：　　日期：

导师签名：　　日期：

院长签名：　　日期：

论文使用授权说明

本人完全了解安阳师范学院有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。保密论文在解密后遵守此规定。

作者签名：　　导师签名：　　日期：

目录

1 概论 1

1.1 论文背景 1

1.2 论文研究的目的和意义 1

1.3 牧畜养殖管理系统概述 2

2 牧畜养殖场管理系统的技术基础 2

2.1 开发与运行环境 2

2.2 总体技术简介和特点 2

2.2.1 简介 2

2.2.2 特点 2

3 牧畜养殖场管理系统需求分析 3
4 牧畜养殖场管理系统总体设计 3
5 牧畜养殖场管理系统的详细设计 5
5.1 前后端分离的系统架构设计 5
5.2 登录页面的实现 5
5.3 注册页面的实现 8
5.4 用户信息修改及密码修改 10
5.5 猪只管理模块的实现 12
5.6 物品管理模块 15
5.7 安全管理模块 18
5.8 数据导出模块 20
5.9 溯源查询模块 22
6 结束语 24
参考文献 24
Abstract: 25
Keywords: Spring B oot ； Angular JS ； Spring Security; Layui 26

基于SSM框架的牧畜养殖场管理系统的设计与实现

任 超
（安阳师范学院 软件学院, 河南 安阳 455000）

摘 要：近年来随着互联网的不断发展，互联网+也越来越过火热，各种各样的传统行业与互联网结合在一起，都取得了不错的成绩，牧畜养殖管理系统的出现，或许可以让我们的畜牧养殖业也享受到互联网所带来的的便利。畜牧养殖管理系统主要是针对畜牧养殖中的养猪业所设计的一个管理系统，系统设计采用了流行的MVC设计模式，后台管理采用了SpringBoot框架来进行搭建，前端页面采用了HTML5、JS、CSS、Layui来进行书写，前后端交互使用了AngularJS实现，使用Maven来进行jar包的版本控制和管理，数据库使用了MySQL数据库，项目通过JAVA语言来进行实现。主要实现了用户中心、猪只管理、物品管理、安全管理、数据导出、溯源查询的功能，可以完成对普通猪场的系统管理，让传统养猪业结合互联网，提升养猪业的管理水平，减少损失，提高收入。

关键字：SpringBoot；AngularJS；SpringSecurity; Layui

1 概论

1.%2 论文背景

在飞速发展的互联网时代，每时每刻都有许多新的互联网产品和技术出现，而让广大的从商群众关注的是如何才能和这个时代接轨，如何才能让自己在这个时代继续生存和发展下去。在这个时候，互联网+出现了，它通过让传统行业和互联网结合，完成了和这个时代的接轨。在各种各样的互联网+发展火热的时候，传统的畜牧养殖业当然也不能落伍，早日完成和互联网的结合，可以早日看到互联网所带来的便利与收益。牧畜养殖系统采用B/S架构进行设计和实现，用户通过浏览器即可使用，无需下载和安装，操作更加简单方便，为系统的使用者带来实质上的帮助。

2.%2 论文研究的目的和意义

牧畜养殖管理系统是针对养猪业所做出的的管理系统，可以帮助养猪人员更好的管理自己的猪场，减少一些因为人为失误所造成的的不必要的损失，减少不必要的支出，增加自己的收益。根据自己实际所学，项目主体采用JAVA语言来进行实现，后端使用了Spring、SpringMVC、MyBatis、SpringBoot、SpringSecurity等框架来进行实现，前端采用HTML5、JS实现异步交互，数据库采用MySQL数据库。因为使用了SpringBoot来进行项目的搭建，使得项目结构更加清晰，极大地减少了配置文件的存在，同时耦合度低、代码可复用性高、可维护性强。前端因为使用了AngularJS框架，使得前端和后端一样实现了MVC架构，极大地降低了系统的耦合性。通过Maven

进行所需jar包的控制，Git进行代码版本的控制，使得开发难度大大降低。同时也使用到了SpringSecurity安全框架和Redis数据存储，让自己对这些技术有了更加深入的了解，积累更多的经验，可以更加从容的面对以后工作中出现的问题。

3.%2 牧畜养殖管理系统概述

牧畜养殖管理系统的开发流程，完全按照软件产品的标准开发流程进行。首先实际去几个猪场养殖地进行调研，了解目前所存在的问题以及具体需要什么样的功能可以更好的管理猪场。了解完具体需求后，开始设计牧畜养殖管理系统的基础架构，首先确定了使用SpringBoot来进行整个项目的构建，后端采用JAVA语言SSM框架来完成系统开发，前端采用HTML5、CSS、JS、Layui完成页面搭建，采用AngularJS框架实现前后端数据的交互，采用MySQL和Redis完成数据的存储，使用Git、Maven完成项目的管理与控制。

根据实际的调查情况和对需求的整理，最终确定了牧畜养殖管理系统的相关功能。此系统的目的就是为了让养猪人员更好的管理自己的猪场，享受互联网所带来的便利，较少损失提高收入。

%1 牧畜养殖场管理系统的技术基础

1

2

1

2

2.1 开发与运行环境

（1） 后端

开发语言：JAVA

开发设备：Windows电脑一台，OS 10及以上版本系统

开发工具：Java SE Development Kit 8 191及以上版本；IntelliJ IDEA 2019.1 x64 及以上版本；apache maven 3.6.0及以上版本

开发工具：；Git version 2.21.0.windows.1 及以上版本

运行环境：apache-tomcat-9.0.17及以上版本；Chrome 74.0.3724.8 及以上版本

（2） 前端

开发语言：HTML5;JavaScript；JQuery

开发设备：Windows电脑一台，OS 10及以上版本系统

开发工具：HBuilderX 1.2.1.20181126.及以上版本

运行环境：Chrome 74.0.3724.8 及以上版本

（3） 数据库

关系型数据库类型版本：MySQL 5.5.27

非关系型数据库类型版本：Redis version 2.8 及以上版本开发设备

开发设备：Windows电脑一台，OS 10及以上版本系统

开发工具：：IntelliJ IDEA 2019.1 x64 及以上版本；Navicat Premium 12.1.17及以上版本

2.2 总体技术简介和特点

2.2.%3 简介

牧畜养殖场管理系统采用目前开发软件首选的前后端分离的系统架构，前后端各自独立开发，互不关联，极大地降低了耦合度，测试功能方便前后端独立测试，产生问题可以及时找出并解决。本系统前端采用HTML5、JavaScript、JQuery、AngularJS进行编写；后端采用SpringBoot、SSM框架进行编写，采用Git、Maven辅助代码开发；数据库采用本地安装的MySQL数据库，使用Navicat进行数据库的视图化管理，使用Redis存储缓存数据；系统使用了SpringSecurity框架来进行安全处理。

3.2.%3 特点

（1） 后端采用SpringBoot+SSM框架，其中SpringBoot框架主要用于整个系统的搭建，SSM即SpringMVC+Spring+MyBatis框架，其

中SpringMVC框架主要用于控制层，负责与前端进行数据交互；Spring框架贯穿整个后端项目，管理着所有的JavaBean，同时是用户业务层，负责详细业务逻辑部分；MyBatis框架主要用于持久层，负责与数据库的交互。

（2） 前端采用AngularJS框架，该框架具有支持Ajax异步调用，和元素双向绑定的特点。Ajax异步调用可以使用户不刷新页面的情况下与后端进行通信，并根据对应的返回结果进行不同的处理。元素的双向绑定方便开发者对数据的访问，避免了复杂的一层一层爬DOM数调用的访问方式，增加了代码的可读性。

（3） 数据库使用MySQL和Redis，分别存放持久化数据和缓存。

（4） 使用SpringBoot搭建系统框架，在Maven引入jar包的时候更加方便，不用花费大量时间去引入jar包依赖。

（5） 通过SpringBoot引入的Tomcat依赖，使得项目运行时不需要额外的进行Tomcat服务器的部署，直接就可以运行项目，使得测试和运行都变得更加方便。

（6） 引入SpringSecurity安全框架，解决了登录问题和权限控制问题，使得整个项目更加安全、可控。

%1 牧畜养殖场管理系统需求分析

一个管理系统是否真的可用取决于是否管理了需要管理的信息，根据去附近几个养殖场场的实地考察，总结出了以下的需求：

（1） 要有最基本的用户登录、注册和退出功能

（2） 对猪只的管理，包括对种猪的管理，幼猪的管理，肉猪的管理还有生产的管理

（3） 对日常所用物品的管理

（4） 对猪只安全的管理

（5） 数据报表的导出，可以下载对应的excel文件

（6） 对于幼猪和种猪的溯源查询

%1 牧畜养殖场管理系统总体设计

通过对需求的调研结果分析，经过多次讨论和修改，设计出了产品需求文档，根据需求文档设计出了牧畜养殖场管理系统的总体设计图，如图1所示

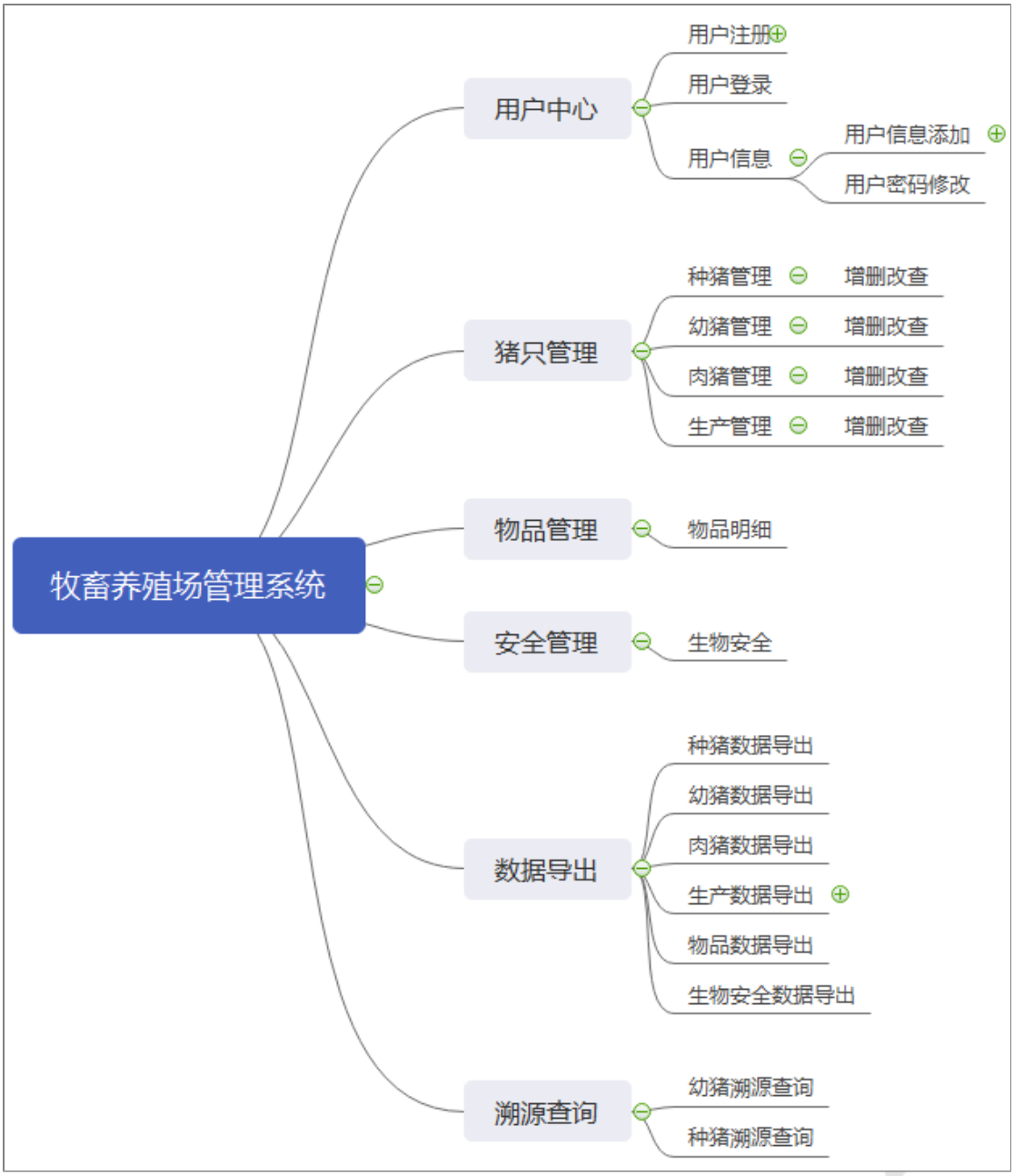

图1 牧畜养殖场管理系统模块划分

牧畜养殖场管理系统主要包括用户中心、猪只管理、物品管理、安全管理、数据导出、溯源查询六个模块，其中猪只管理又分为种猪管理、幼猪管理、肉猪管理、生产管理。这些模块的具体实现逻辑如下：

（1） 用户中心模块，用户中心模块包括了用户注册、用户登录、用户信息；用户注册模块中，用户可以输入用户名、密码等信息，根据输入的手机号发送验证码进行验证后注册；用户登录模块中，用户必须输入用户名和正确的密码才能登录系统；用户信息模块中，用户可以完善自己的个人信息，包括邮箱，所在地等等，还可以对密码进行修改。

（2） 猪只管理模块：猪只管理模块又包括了对种猪、幼猪、肉猪和生产的管理，

（3） 物品管理模块：包括了对物品明细的记录

（4） 安全管理模块：主要包括了对猪只身体安全的管理

（5） 数据导出模块：用户可以将猪只资料等信息下载对应的excel表格进行本地保存查看。

（6） 溯源查询：用户可以通过输入想要查询的猪只耳号，进行对应信息的查询。

%1 牧畜养殖场管理系统的详细设计

1.%2 前后端分离的系统架构设计

牧畜养殖场管理系统采用前后端分离的架构来进行设计，这样做可以让项目的开发变得清晰明了，也可以更好的维护整个系统。前端完全使用静态页面来进行实现，使用了Layui框架可以更快的搭建出静态模板，使用AngularJS框架封装的Ajsx异步请求来和后端进行数据交互，后端只提供可以调用的接口服务即可；因为使用AngularJS，前端同时也使用了MVC的三层架构模式，将调用后端接口的方法分为了Service层和Controller层，页面需要和后端交互，在页面中引入对应的Service和Controller即可，不需要在HTML页面中书写大量的方法，让前端代码的可复用性提高，同时使耦合度降低；后端使用的SpringBoot和SSM框架，也让项目的耦合度极大地降低。

%1.2 登录页面的实现

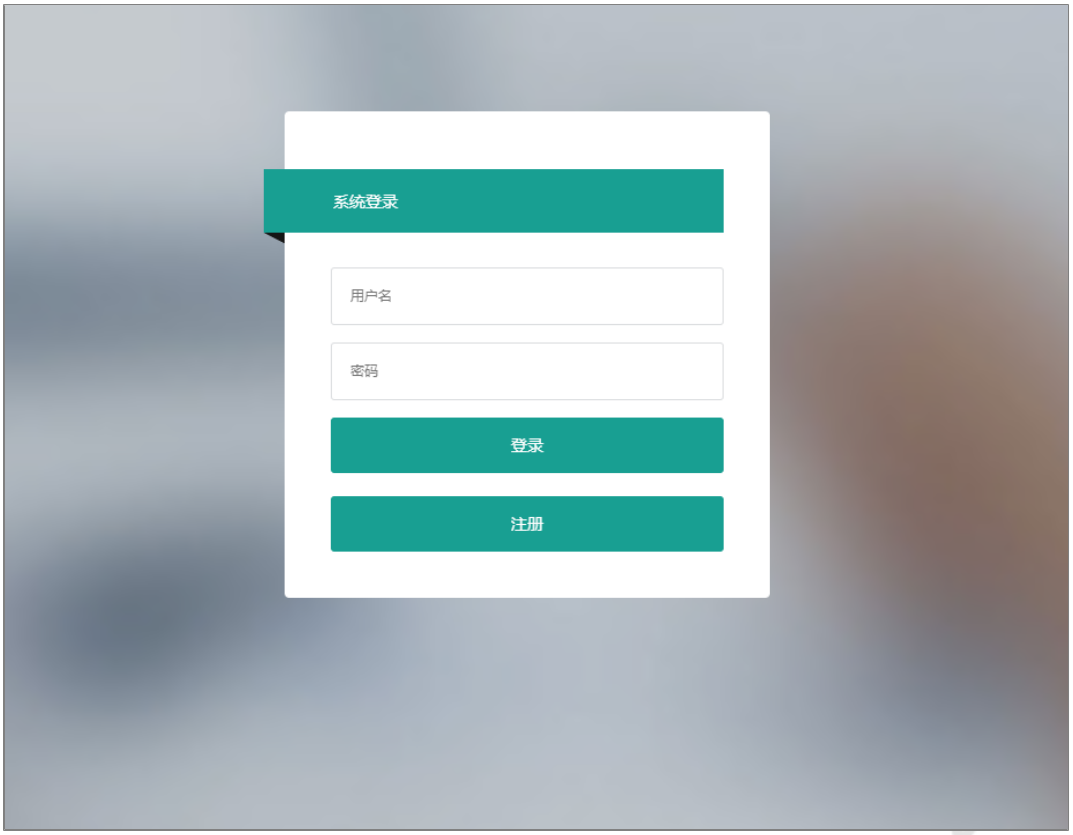

图2 系统登录界面

系统的登录界面如图2所示，当启动项目后，除了注册页面等一些必要的请求外，剩下的无论访问哪个页面都会强制跳转到登录页面，如果有账户和密码，即可登录系统，若没有，可以点击注册跳转到注册页面进行账户的注册。

系统会拦截其他的请求而放行需要的请求得益于SpringSecurity框架的引入。SpringSecurity框架是专门针对基于Spring的项目的安全框架，充分利用了Spring IoC，DI（控制反转Inversion of Control ,DI:Dependency Injection 依赖注入）和AOP（面向切面编程）功能。安全框架有两个重要的概念，认证（Authentication）和授权（Anthorization），认证后的用户可以访问当前系统，授权即确定当前用户在当前系统下所拥有的权限，可以访问哪些页面，访问哪些请求等等，SpringSecurity的基本流程图如图3 所示

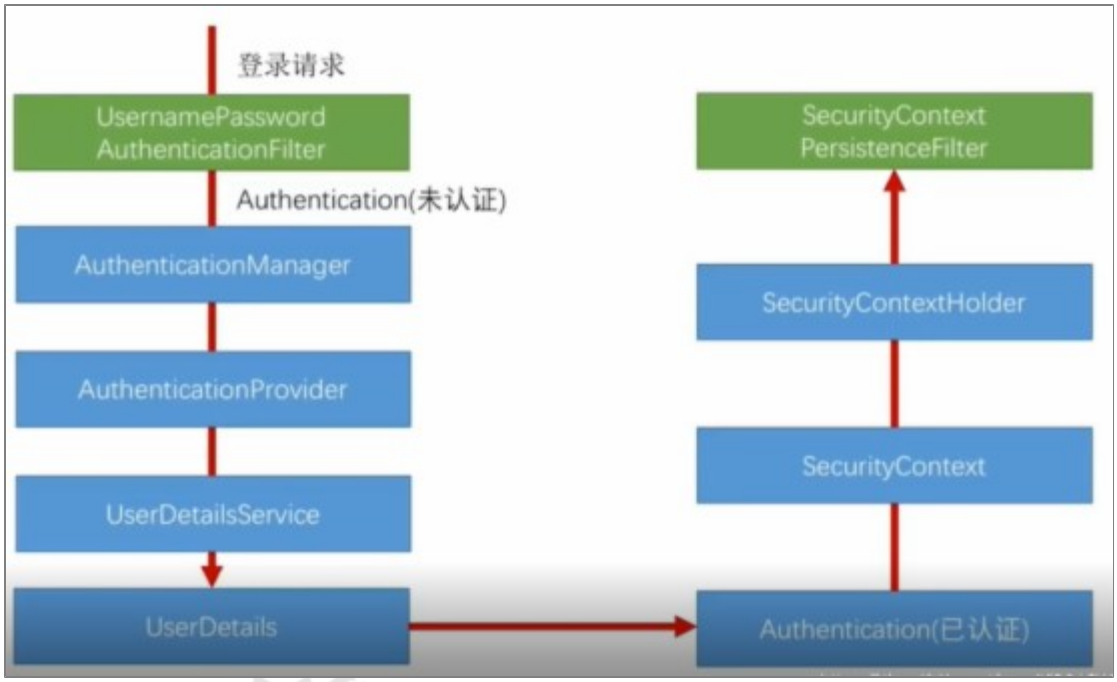

图3 SpringSecurity的认证流程

在项目中SpringSecurity的配置如图4所示：自定义配置类继承WebSecurityConfigurerAdapter，重写configure方法，完成对页面的拦截与控制，重写 configureGlobal，注入自定义认证配置如图5所示，自定义认证配置需要注入继承了UserDetailService 类的对象如图6所示，配置完成后，就可以对用户输入的信息进行查询匹配，之后登陆成功。

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .authorizeRequests()
            .antMatchers("/login.html","/register.html","/css/**",  "/fonts/**", "/images/**", "/
            .anyRequest().authenticated()
            .and()
            .formLogin().loginPage("/login.html")
            .loginProcessingUrl("/login")
            .defaultSuccessUrl("/index.html")
            .permitAll()
            .and()
            .logout().logoutUrl("/logout")
            .logoutSuccessUrl("/login.html")
            .csrf().disable()
            .headers().frameOptions().sameOrigin();
}
```

图4 SpringSecurity配置类

```java
public Authentication authenticate(Authentication authentication) throws AuthenticationException {
    // TODO Auto-generated method stub
    String userName = authentication.getName();// 这个获取表单输入中返回的用户名;
    //System.out.println("用户名: "+userName);
    String password = (String) authentication.getCredentials();// 这个是表单中输入的密码;
    // System.out.println("密码: "+password);
    // 这里构建来判断用户是否存在和密码是否正确
    UserInfo userInfo = (UserInfo) userDetailService.loadUserByUsername(userName); // 这里调用我们的自己
    if (userInfo == null) {
        throw new BadCredentialsException("用户名不存在");
    }
    /* System.out.println("现在输入加密: "+new BCryptPasswordEncoder().encode(password));
    System.out.println("数据库中信息"+userInfo.getPassword());*/
    //System.out.println(DigestUtils.md5DigestAsHex(password.getBytes()));
    if (!userInfo.getPassword().equals( DigestUtils.md5DigestAsHex(password.getBytes()))) {
        throw new BadCredentialsException("密码不正确");
    }
```

图5 自定义AuthenticationProvider对象

```java
@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    // TODO Auto-generated method stub
    UserPo select = registerService.select(username);
    //这里可以可以通过username（登录时输入的用户名）然后到数据库中找到对应的用户信息，并构建成我们自己的UserInfo来
    //这里可以通过数据库来查找到实际的用户信息
    if (null == select) {
        System.out.println("未查询出用户信息");
        return null;
    }else if (username.equals(select.getUsername())) {
        UserInfo userInfo = new UserInfo(select.getUsername(), select.getPassword(), role:"ROLE_ADMIN", accountNo
        return userInfo;
    }
        return null;
    }
```

图6 自定义UserDetailService对象

%1.%2 注册页面的实现

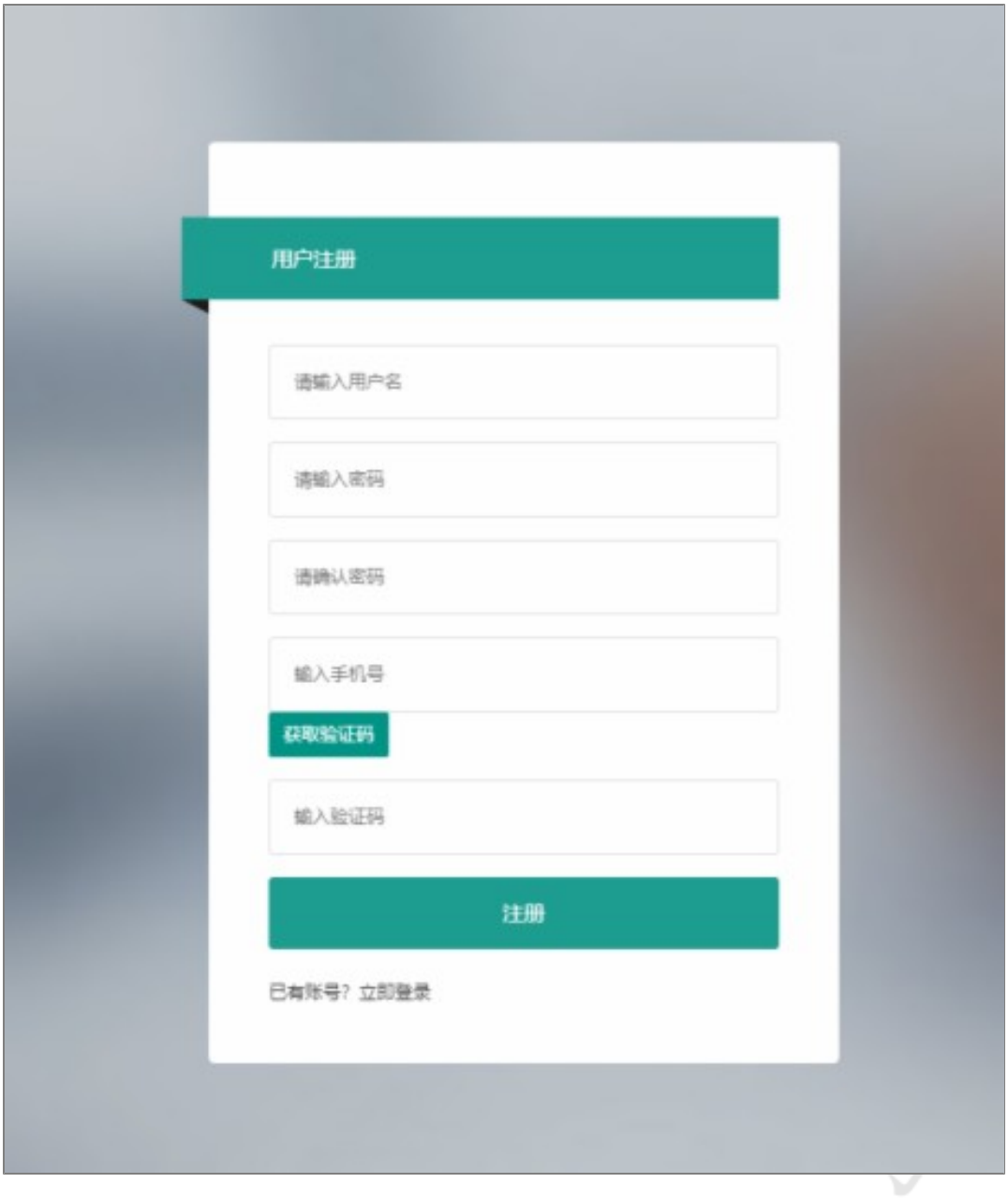

图7 用户注册界面

用户注册界面如图7所示，用户通过输入注册所需的必要信息，完成注册。其中用户名需要保持其唯一性，当输入数据库中已存在的用户名时会弹窗进行提示，如果坚持注册会注册失败，密码输入后会在前台对两次输入的密码进行校验，如果不相同则清空并要求重新输入，输入手机号进行了对手机号的格式校验，格式正确会发送短信验证码成功，否则发送验证码失败，输入正确的验证码后完成用户注册。

发送验证码所利用的是阿里云的短信功能，通过完成对阿里云短信功能的SDK的修改，导入自己的配置信息，完成短信的发送，如图8所示；发送的验证码由后台随机数生成如图9所示，生成后6位随机数后存入Redis缓存中一份，调用发送功能将验证码发送给用户一份；当用户输入验证码后，从Redis中读取之前存入的数据和用户输入的数据进行比较如图10所示，相同则用户注册成功，注册接口如图11所示。进行操作Redis时使用了SpringDataRedis，对Redis的操作进行了封装，使用时只需注入RedisTemplate即可快速方便的操作Redis。

```
public SendSmsResponse sendSms(String mobile, String template_code, String sign_name, String param) throws C
    System.out.println(mobile+"  "+template_code+"  "+ sign_name+"  "+param);
    String accessKeyId=env.getProperty("accessKeyId");
    String accessKeySecret=env.getProperty("accessKeySecret");
    //可自助调整超时时间
    System.setProperty("sun.net.client.defaultConnectTimeout", "10000");
    System.setProperty("sun.net.client.defaultReadTimeout", "10000");
    //初始化acsClient,暂不支持regionId
    IClientProfile profile = DefaultProfile.getProfile( regionId: "cn-hangzhou", accessKeyId, accessKeySecret);
    DefaultProfile.addEndpoint( endpointName: "cn-hangzhou", regionId: "cn-hangzhou", product, domain);
    IAcsClient acsClient = new DefaultAcsClient(profile);
    //组装请求对象-具体描述见控制台-文档部分内容
    SendSmsRequest request = new SendSmsRequest();
    //必填:待发送手机号
    request.setPhoneNumbers(mobile);
    //必填:短信签名-可在短信控制台中找到
    request.setSignName(sign_name);
    //必填:短信模板-可在短信控制台中找到
    request.setTemplateCode(template_code);
    //可选:模板中的变量替换JSON串,如模板内容为"亲爱的${name},您的验证码为${code}"时,此处的值为
    request.setTemplateParam(param);
```

图8 短信发送

```
@Override
public void createSmsCode(String phone) throws ClientException {
    //生成一个6位随机数
    String smsCode = (long) (Math.random() * 1000000) + "";
    //将验证码放入redis
    redisTemplate.boundHashOps( key: "smsCode").put(phone,smsCode );
    Map map = new HashMap();
    map.put("code", smsCode);
    SendSmsResponse response = smsUtil.sendSms(phone,template_code,sign_name, JSON.toJSONString(map));

    System.out.println("Code=" + response.getCode());
    System.out.println("Message=" + response.getMessage());
    System.out.println("RequestId=" + response.getRequestId());
    System.out.println("BizId=" + response.getBizId());
```

图9 随机数的生成

```
@Override
public boolean checkSmsCode(String phone, String code) {
    String redisCode = (String) redisTemplate.boundHashOps( key: "smsCode").get(phone);
    if (redisCode == null) {
        return false;
    }
    if (!code.equals(redisCode)) {
        return false;
    }
    return true;
}
```

图10 输入验证码的校验

```
public Result addUser(UserPo userRegisterPo, String smsCode) {
    // System.out.println(smsCode);
    boolean b = registerService.checkSmsCode(userRegisterPo.getPhone(), smsCode);
    if (!b) {
        return new Result( result: false, message: "验证码有误");
    }
    try {
        if (null != registerService.select(userRegisterPo.getUsername())) {
            return new Result( result: false, message: "注册失败");
        }else {
            registerService.add(userRegisterPo);
            return new Result( result: true, message: "注册成功");
        }
    } catch (Exception e) {
        e.printStackTrace();
        return new Result( result: false, message: "注册失败");
    }
}
```

图11 用户注册接口

%1.%2 用户信息修改及密码修改

用户信息修改页面如图12所示，用户信息修改主要是对用户在注册时未填写的信息的补充，包括用户的性别，邮箱，所在地等信息的添加。用户信息的修改时的查询是根据用户的用户名来进行查询的，因为使用了SpringSecurity框架，所以可以直接通过SpringSecurity的SecurityContextHolder.getContext().getAuthentication().getName()方法来进行用户名的获取。

密码修改页面如图13所示，进行密码的修改时，需要先输入原密码进行比较，原密码无误后才能输入新的密码进行密码的修改操作，原密码的查询也是通过获取到用户名来进行查询，因为在用户注册时，对密码进行了MD5加密，所以在进行原密码比较的时候也需要先对用户的密码进行加密后才能进行比较，修改密码接口如图14所示。

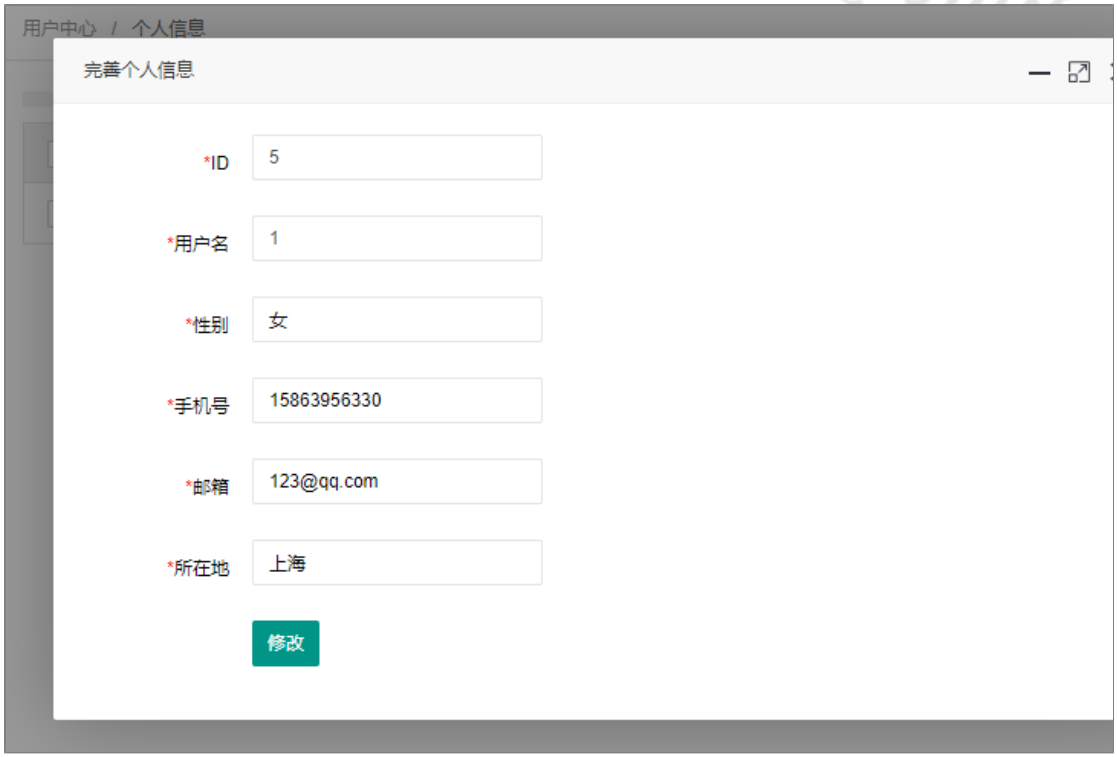

图12 用户信息修改页面

图13 密码修改页面

```
@RequestMapping("/updatePassword")
@ResponseBody
public Result updatePassword(PasswordPo passwordPo) {
    String name = SecurityContextHolder.getContext().getAuthentication().getName();
    UserPo userPo = userService.selectByUsername(name);
    if (!userPo.getPassword().equals(DigestUtils.md5DigestAsHex(passwordPo.getOldPassword().getBytes()))) {
        return new Result( result: false,  message: "密码修改失败");
    }
    try {
        userPo.setPassword(DigestUtils.md5DigestAsHex(passwordPo.getPassword().getBytes()));
        userService.modifyPassword(userPo);
        return new Result( result: true,  message: "密码修改成功");
    } catch (Exception e) {
        e.printStackTrace();
        return new Result( result: false,  message: "密码修改失败");
    }
}
```

图14 密码修改的接口

%1.%2 猪只管理模块的实现

猪只管理模块主要包括了对种猪、幼猪、肉猪和生产的管理。

种猪管理模块包括了主要包括了对种猪数据的录入，对录入信息的修改，对录入信息的删除以及批量删除，对已录入信息的条件查询等功能；录入信息时需要保证耳号的唯一性，否则会提示录入信息失败。用户可以通过此模块详细的记录种猪的信息。

幼猪管理模块也包括了对信息的录入、修改、查询和删除等功能。录入信息的时候，需要对种猪的耳号信息进行录入，该耳号必须存在，否则会录入失败。

肉猪管理的信息相对较少，不过也包含了基本的操作。

生产管理主要是针对种猪的生产管理，包括对每个种猪的胎次、产仔数、流产数、存活数等信息的记录，可以详细的看出实际生产情况。

对数据交互类Po的封装均用到了lombok插件，只需要加入@Data、@AllArgsConstructor、@NoArgsConstructor这三个注解，即可省去

构造方法、getter()、setter()等方法书写，让整个数据交互类变得简洁明了。所有的数据交互类都实现了Serializable，保证数据的序列化存储，如图15所示。

在实现查询功能时，分页查询模块的后端使用了Mybatis的PageHelper插件。因为使用了SpringBoot框架，所以需要在启动类中对该插件进行属性的配置如图16所示，否则将会无法使用。使用时仅需调用PageHelper.startPage()方法，传入当前页和页面大小即可实现分页功能，如图17所示。分页的查询结果通过PageResult进行封装，PageResult中包括了当前查询的总记录数（Long类型）和当前页的结果（List）类型。前端页面的分页模块使用了AngularJS的pagination插件，该插件通过配置paginationConf中的currentPage、totalItems、itemsPerPage、perPageOptions属性，即可完成对分页的实现，如图18所示。

前端调用后端接口除了返回基本类型的数据外，额外封装了一个Result结果类来进行数据的返回，如图19所示。该Result类包括了一个Boolean类型的result属性和一个String类型的message属性，前端通过获取这两个属性的值来完成判断和提示的操作。

后端Controller层的接口统一使用了@ResponseBody注解，用来将返回给前端的结果转换成json格式；需要接收实体类的接口使用了@RequestBody属性，用来将前端传过来的对象自动转换成JavaBean对象，如图20所示，极大地简化了开发的难度。

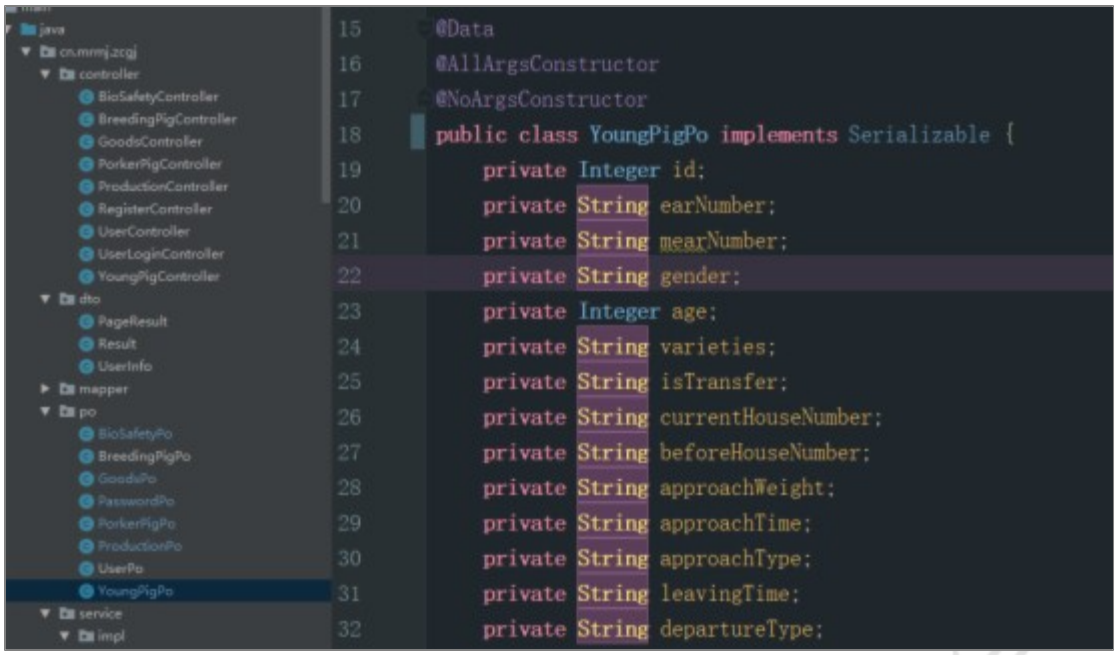

图15 数据交互类Po的实现

```java
/**
 * 配置mybatis的分页插件pageHelper
 */
@Bean
public PageHelper pageHelper() {
    PageHelper pageHelper = new PageHelper();
    Properties properties = new Properties();
    properties.setProperty("offsetAsPageNum", "true");
    properties.setProperty("rowBoundsWithCount", "true");
    properties.setProperty("reasonable", "true");
    //配置mysql数据库的方言
    properties.setProperty("dialect", "mysql");
    pageHelper.setProperties(properties);
    return pageHelper;
}
```

图16 SpringBoot中PageHelper的配置

```java
@Override
public PageResult findYoungPig(YoungPigPo youngPigPo, int pageNum, int pageSize) {
    PageHelper.startPage(pageNum, pageSize);
    Page<YoungPigPo> page = (Page<YoungPigPo>) youngPigMapper.selectByExample(youngPigPo)
    return new PageResult(page.getTotal(),page.getResult());
}
```

图17 查询时PageHelper的调用

```
//分页控件配置
$scope.paginationConf = {
    currentPage: 1,
    totalItems: 10,
    itemsPerPage: 10,
    perPageOptions: [10, 20, 30, 40, 50],
    onChange: function(){
        $scope.reloadList();//重新加载
    }
};
```

图18 前端分页控件配置

```
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Result implements Serializable {
    private boolean result;
    private String message;
}
```

图19 返回的Result结果集

```
@RequestMapping("/searchByExample")
@ResponseBody
public PageResult query(@RequestBody YoungPigPo youngPigPo, int page, int size) {
    return youngPigService.findYoungPig(youngPigPo, page, size);
}
```

图20 配置@ResponseBody和@RequestBody注解

%1.%2 物品管理模块

物品管理模块主要包括物品明细功能，物品明细页面如图21所示。用户可以通过这个功能记录日常的物品购买情况，如图22所示，消费支出，还可以通过对购买的物品通过类型、名称等进行查询，也可以对之前的物品进行删除和修改等操作。

其中删除和修改是根据这条记录的id来进行的。当点击修改操作时，会弹出新的页面，如图23所示，同时前端将这条记录的id传入修改页面，之后先根据传入的id进行这条记录的查询，将查询出的数据回显到页面中进行显示，用户输入新数据后，根据id进行数据的更新。删除操作也是根据用户id来进行的，其中删除包括单条记录的删除和批量删除。批量删除的实现是通过前端获取并记录用户所勾选的记录id，勾选的id添加到集合中，取消勾选后从该集合中删除此id，如图24所示，点击批量删除时将该集合传入后台，后台接收到这个集合后，在Service层对该集合进行遍历删除操作，如图25所示。

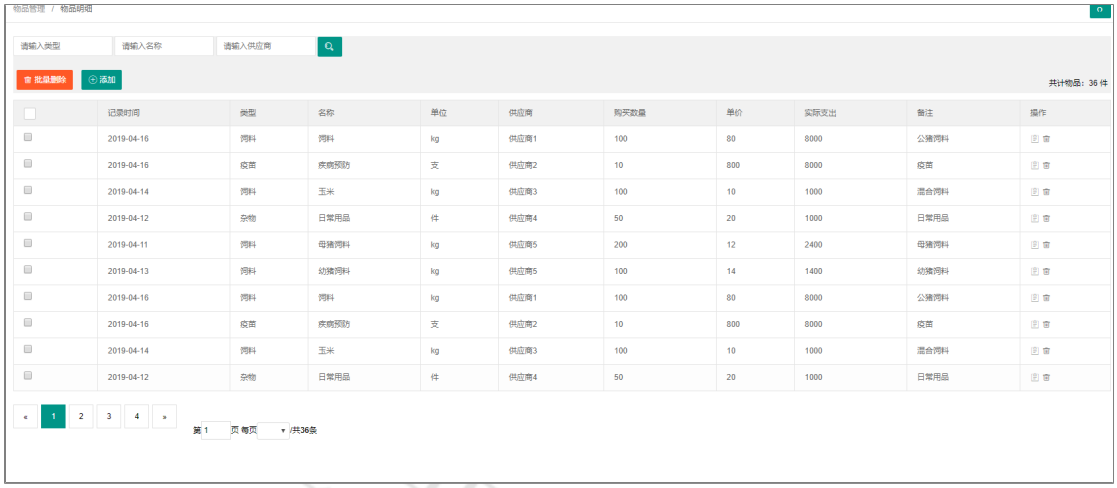

图21 物品明细页面

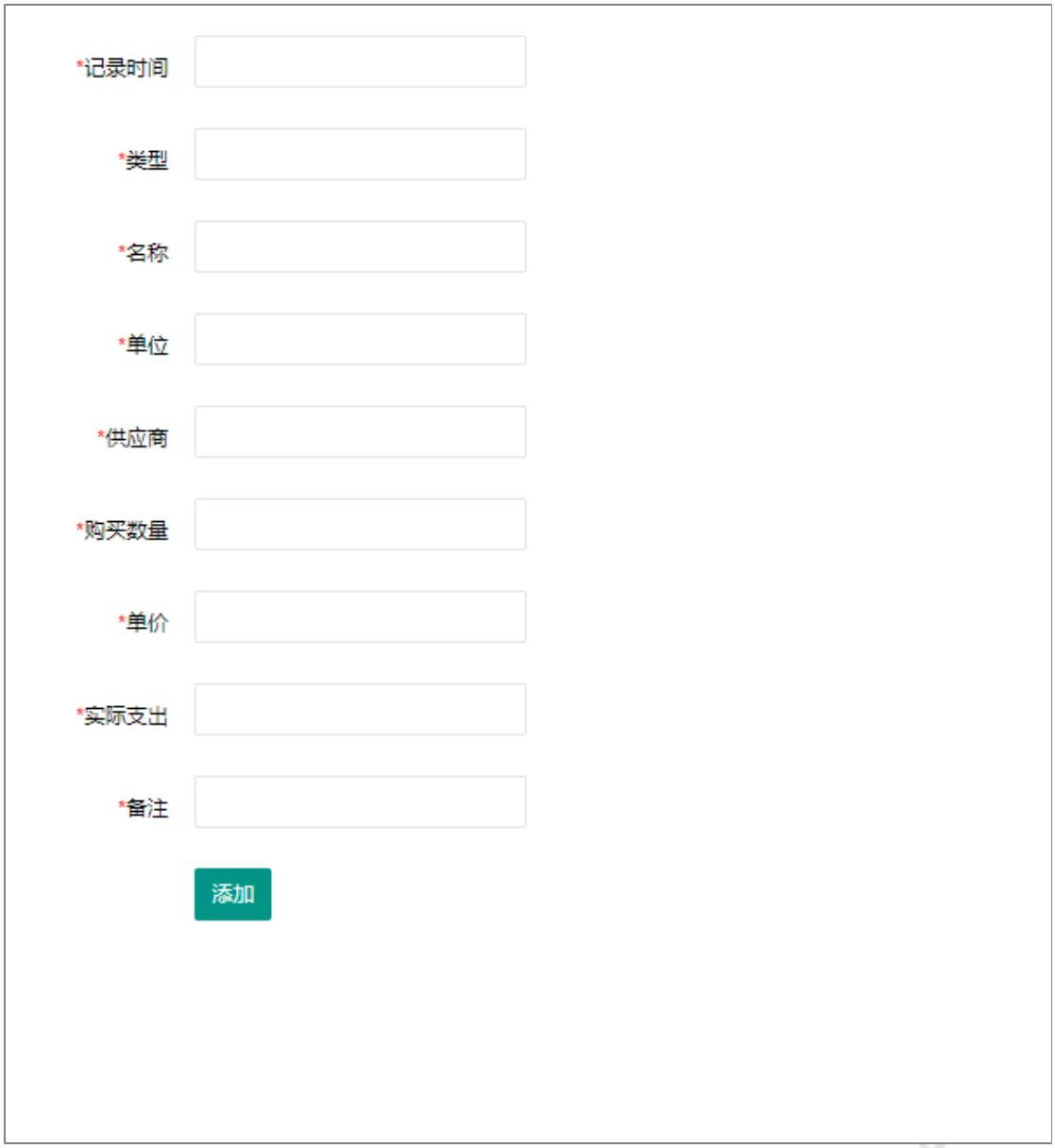

图22 物品添加页面

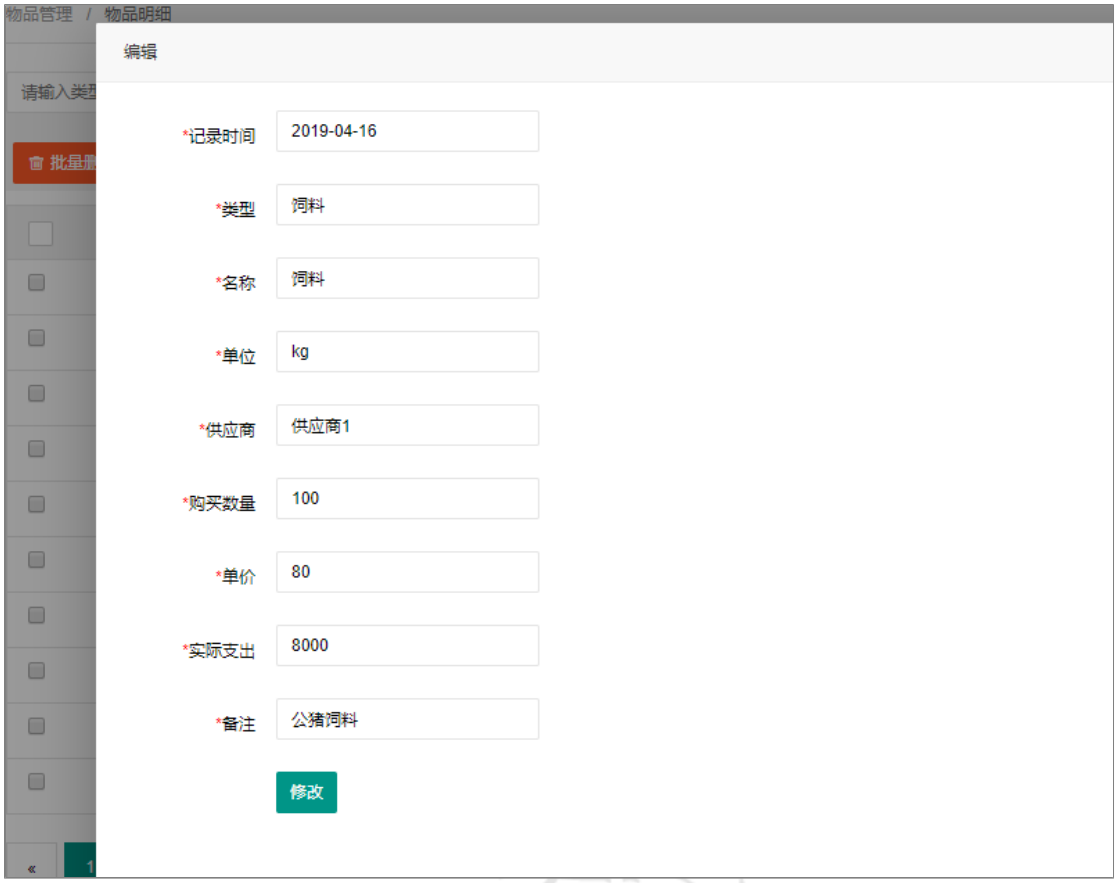

图23 物品修改页面

```
$scope.selectIds=[];//选中的id集合
//更新复选框
$scope.updateSelection=function ($event,id) {
    if($event.target.checked){   //如果被选中，添加到数组中
        $scope.selectIds.push(id);
    }else {
        var idx = $scope.selectIds.indexOf(id);  // 查找的位置
        $scope.selectIds.splice(idx, 1);  //1.移出的位置 2.移出的个数
    }
};
```

图24 前端获取所勾选集合

```
@Override
public void delete(Long[] ids) {
    for (Long id : ids) {
        youngPigMapper.deleteByIds(id);
    }
}
```

图25 Service层的遍历删除操作

%1.%2 安全管理模块

安全管理模块主要是对生物安全的管理，记录猪只的体检情况、免疫情况、疫苗接种类型、消毒情况等数据，如图26所示，根据唯一标识耳号来进行查询和区分不同的猪只，完成对猪只生物安全的管理。

在添加数据时输入耳号后，如果鼠标离开输入框，会自动调用根据耳号查询的方法，返回Result结果集，如果该耳号数据库中已存在，会出现弹窗提示，如图27所示，不存在即可继续输入其他项，其中待输入项前有红色*号标识的为必填项，如果不填将不允许数据的提交，如图28所示。添加已存在的耳号弹窗方法的Controller层如图29所示，Service层如图30所示，耳号弹窗的后端接口如图31所示

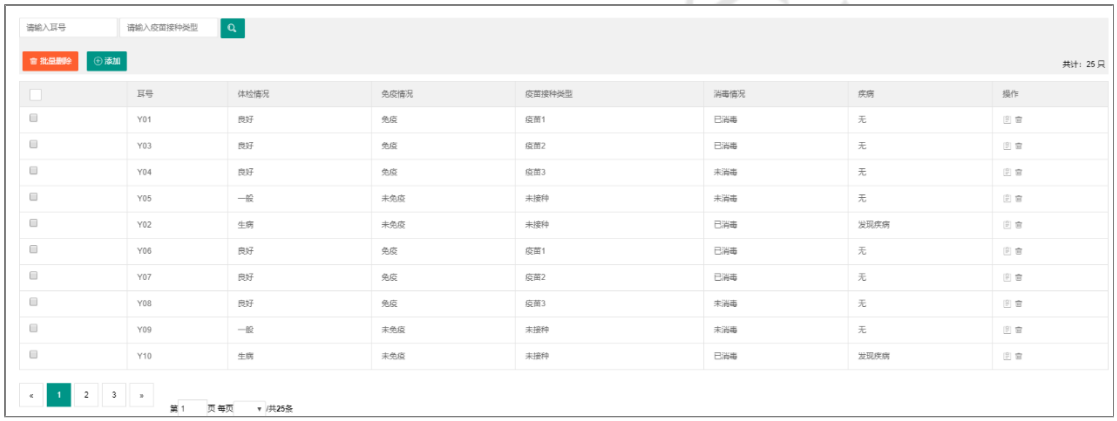

图26 生物安全界面

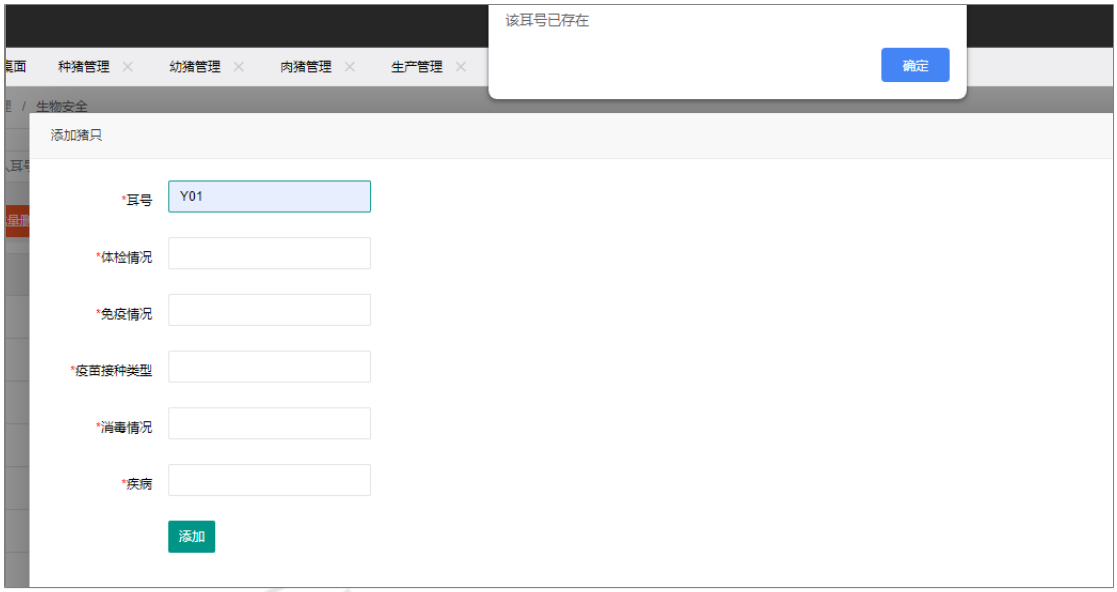

图27 添加已存在耳号弹窗提示

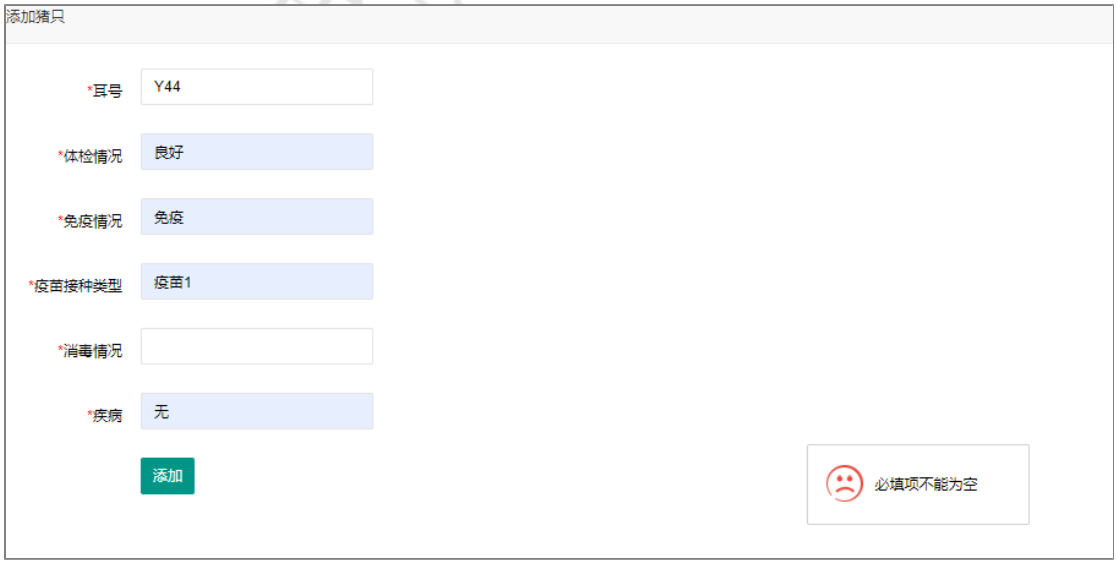

图28 未添加必填项禁止数据提交

```
$scope.selectByEarNum=function (earNumber) {
    bioSafetyService.selectByEarNum(earNumber).success(
        function (response) {
            if (response.result){
                alert("该耳号已存在");
            }
        });
};
```

图29 添加已存在耳号弹窗提示前端controller实现

```
this.selectByEarNum=function (earNumber) {
    return $http.post('../bioSafety/selectByEarNum?earNumber='+earNumber);
};
```

图30 添加已存在耳号弹窗提示前端service实现

```
@RequestMapping("/selectByEarNum")
@ResponseBody
public Result selectByEarNum(String earNumber){
    BioSafetyPo bioSafetyPo= bioSafetyService.selectByEarNum(earNumber);
    if (bioSafetyPo != null) {
        return new Result( result: true,  message: "耳号已存在");
    } else {
        return new Result( result: false,  message: "正常");
    }
}
```

图31 添加已存在耳号弹窗提示后端接口实现

%1.%2 数据导出模块

数据导出模块主要实现了数据的Excel格式导出。包括种猪数据导出，幼猪数据导出，肉猪数据导出，生产数据导出，物品数据导出，生物安全数据导出，如图32所示。

数据导出并未使用JAVA的插件在后台实现导出，而是将数据传入前端，在前端生成报表进行导出。因为前端使用了layui进行了页面的搭建设计，所以数据导出所使用的前端插件也是以Layui为基础设计的。layui_exts插件，具有强大的SheetJS功能，依赖XLSX.js和FileSaver，可以出色的完成数据报表的导出。

在需要实现导出功能的页面调用Layui的方法手动加载Excel插件，如图33所示，通过Ajax请求后端接口返回数据，如图34所示，后端依旧返回json格式的数据即可。通过执行梳理函数excel.filterExportData()，将后端给的数据顺序顺序和映射关系对应，如图35所示；通过执行data.unshift()函数，为导出的报表添加表头；最后调用excel.exportExcel()方法，如图36所示，传入导出报表的数据、报表的名称和报表的格式，即可完成报表的导出。

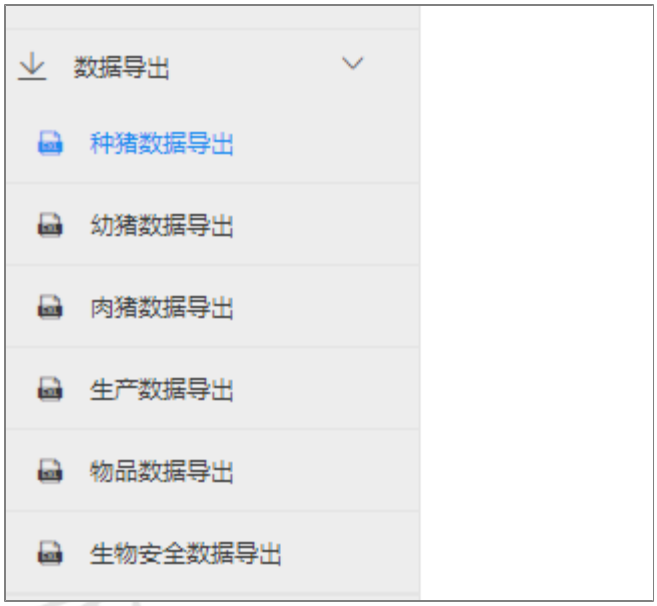

图32 数据报表导出的前端页面

```
<!--layui插件引入-->
<link rel="stylesheet" type="text/css" href="../lib/layui/css/layui.css">
<script type="text/javascript" src="../lib/layui/layui.js"></script>
<!--手动加载 excel 插件-->
<script type="text/javascript">
    layui.config({
        base: '../layui_exts/'
    }).extend({
        excel: 'excel'
    });
</script>
```

图33 数据报表导出页面引入并加载插件

```
<!--导出方法-->
<script>
    var Export = function exportApi() {
        layui.use(['jquery', 'excel', 'layer'], function() {
            var $ = layui.jquery;
            var layer = layui.layer;
            var excel = layui.excel;
            $.ajax({
                url: '../youngPig/findAll'
                ,dataType: 'json'
                ,success: function(response) {
                    var data = response.rows;
                    console.log(data);
```

图34 Ajax调用后端接口返回json数据

```
data = excel.filterExportData(data, {
    earNumber: 'earNumber'
    ,mearNumber: 'mearNumber'
    ,gender: 'gender'
    ,age: 'age'
    ,varieties: 'varieties'
    ,isTransfer: 'isTransfer'
    ,currentHouseNumber: 'currentHouseNumber'
    ,beforeHouseNumber: 'beforeHouseNumber'
    ,approachWeight: 'approachWeight'
    ,approachTime: 'approachTime'
    ,approachType: 'approachType'
    ,leavingTime: 'leavingTime'
    ,departureType: 'departureType'
});
```

图35 后端数据顺序映射

```
data.unshift({earNumber:'耳号',mearNumber:'母猪耳号',gender:'性别',
    age:'日龄',varieties:'品种',isTransfer:'是否转舍',currentHouseNumber:'现在舍号',
    beforeHouseNumber:'转前舍号',approachWeight:'进场重量',approachTime:'进场时间',approachType
    leavingTime:'离场时间',departureType:'离场类型'});

var timestart = Date.now();
excel.exportExcel({
    sheet1: data
}, '幼猪数据.xlsx', 'xlsx');
var timeend = Date.now();
```

图36 添加表头并执行导出方法

%1.%2 溯源查询模块

溯源查询模块包括了幼猪溯源查询和种猪溯源查询，如图37所示，主要实现的是根据输入的耳号，从数据库中查询出所有和这个耳号相关的数据并返回。其中种猪溯源查询还会把对应的幼猪查询出来并显示。

将对应的AngularJS、 service和controller引入页面，如图38所示，在body元素上添加ng-app和ng-controller引入服务，在点击按钮上添加ng-click()方法，调用controller中所写好的服务即可完成数据的查询，如图39所示。种猪溯源查询用时执行了两个方法来完成种猪和幼猪信息的查询，最后将返回的结果集映射绑定在对应的值上，如图40所示，页面就可以将数据显示出来，如图41所示。

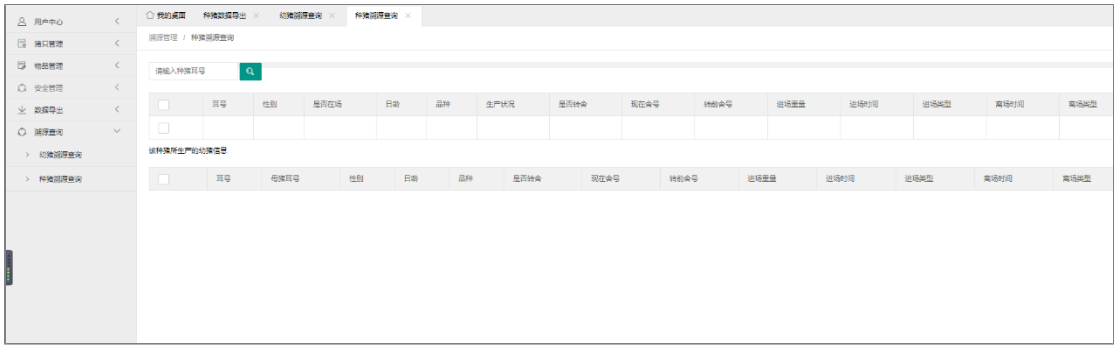

图37 溯源查询的前端页面

```
<!--分页组件-->
<script src="../js/angularjs/pagination.js"></script>
<link rel="stylesheet" href="../js/angularjs/pagination.css">

<script type="text/javascript" src="../js/base.js"></script>
<script type="text/javascript" src="../js/service/sourceService.js"></script>
<script type="text/javascript" src="../js/controller/sourceController.js"></script>



</head>

<body ng-app="zcgj" ng-controller="sourceController" >
  <div class="x-nav">
```

图38 引入对应的JS

```
<input type="text" name="username"  placeholder="请输入种猪耳号" autocomplete="true" class="layui-input"  ng-model="searchEntity.bearNumber" ng

<button class="layui-btn"  lay-submit="" lay-filter="sreach" ng-click="selectBreeding(searchEntity.bearNumber)" ><i class="layui-icon">&#xe615;
```

图39 在按钮上添加调用

```
<tbody>
<tr ng-repeat="ventity in list">
  <td>
    <div class="layui-unselect header layui-form-checkbox" lay-skin="primary"><i class="layui-icon">&#xe605;</i></div>
  </td>
  <td>{{ventity.earNumber}}</td>
  <td>{{ventity.mearNumber}}</td>
  <td>{{ventity.gender}}</td>
  <td>{{ventity.age}}</td>
  <td>{{ventity.varieties}}</td>
  <td>{{ventity.isTransfer}}</td>
  <td>{{ventity.currentHouseNumber}}</td>
  <td>{{ventity.beforeHouseNumber}}</td>
  <td>{{ventity.approachWeight}}</td>
  <td>{{ventity.approachTime}}</td>
  <td>{{ventity.approachType}}</td>
  <td>{{ventity.leavingTime}}</td>
  <td>{{ventity.departureType}}</td>
</tr>
```

图40 数据的绑定

图41 页面查询出数据的显示

%1 结束语

通过这次系统的开发，使我对软件开发的流程有了更加清晰全面的认识，同时也丰富了自己的知识体系，增强了动手实践的能力，使自己的编程习惯和编程经验得到了不小的提升。从前端到后端再到数据库，让我对软件工程的结构体系有了更加深刻的认知。在开发的过程中，自己学习到了前后端分离的技术架构，学习到了SpringBoot为软件开发所带来的便利，学习并了解了许多优秀的前端框架和开发工具，让自己的知识水平获得了进一步的提升。

项目中所使用的一些技术和方法都是自己在网上或者实习过程中所学习到的，完成这个项目后，让我对这些知识有了更加深刻的了解，认识到了什么事都要自己手动去实现一次，才是真的会用了，才能将这些知识真的变成自己的东西。这种学习方式不仅可以提高自己的编程效率，同时也可以提高自己对新技术的学习。通过这次开发让我认识到了目前越来越多的行业正在向互联网+转型，我身为一名软件工程专业的学生，身为一名软件开发者，正处于这个时代的浪潮，而自己要想在这个时代有所作为，就要继续去学习更多的知识，通过自己的努力，让自己变得更加优秀，让自己对社会的进步尽一份微薄之力。

致谢

转眼间大学的时光就到了尾声，而我们也该要对自己的学生时代说一句再见了。大学是一个令人敬仰的地方，大学是一个学习知识的圣地，在这四年中，我在同学和老师的帮助下学习到了许多的专业知识，同时自学了许多的其他的知识，开拓了自己的视野，丰富了自己的知识储备。从大一开始，苏静老师就邀请我和其他同学一起加入了学习团队，而正是因为这个学习团队，让我的大学生活变得充满目标，让我从中学习到了很多很多的知识，为以后的实习奠定了扎实的基础。苏静老师在百忙之中抽空对我们团队的指导，也让我们受益匪浅。

由衷的感谢我的毕业论文指导老师苏静老师，苏静老师负责任的督促和定期的检查，让我的毕业设计有了保障，对我的指导也让我对毕业设计有了更加全面的认识，对于完成我的毕业设计给予了很大的帮助。

感谢大学里的每一位指导过我的老师，感谢你们的悉心指导，希望在以后的日子里，老师们的工作生活一帆风顺，而我也会在接下来的生活中继续努力，为了自己的梦想奋斗，体验精彩的人生。

参考文献

[1] 杨家炜. 基于Spring Boot的web设计与实现[J]. 轻工科技, 2016(7):86-89.

[2] Li J , Peng C . jQuery-based Ajax general interactive architecture[C]// Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on. IEEE, 2012.

[3] 孟庆利, 王朝军, 徐利, et al. 中小规模猪场管理细节分析[C]// 2014.

[4] BradDayley, 戴利, 王肖峰, et al. AngularJS开发秘籍[M]. 清华大学出版社, 2015.

[5] 丁振凡. 基于Spring Security的Web资源访问控制[J]. 宜春学院学报, 2012, 34(8):71-74.

[6] （美）凯 S.霍斯特曼. Java核心技术 卷I：基础知识（原书第10版）[M].北京：机械工业出版社，2016-09-01.

[7] （美）凯 S.霍斯特曼. Java核心技术 卷II：高级特性（原书第10版）[M].北京：机械工业出版社，2017-09-01.

[8] [美] Nicholas C.Zakas 著；李松峰，曹力 译，《JAVA Script 高级程序设计》，人民邮电出版社，2012-03-01
[9] 阳小兰，罗明. 基于Spring+SpringMVC+MyBatis网上论坛的设计与实现[J]. 黑龙江科技信息，2016(36):279-280.

Design and implementation of livestock farm management system based on SSM framework

RENChao
(School of Software Engineering, Anyang Normal University, Anyang, Henan 455000)

Abstract: In recent years, with the continuous development of the Internet, "Internet +" has become more and more popular. A variety of traditional industries have been combined with the Internet and achieved good results. The emergence of livestock breeding management system may enable our livestock breeding industry to enjoy the convenience brought by the Internet. Animal husbandry management system mainly for animal husbandry and aquaculture in pig breeding by design of a management system, the system design using the popular MVC design pattern, background management SpringBoot framework is adopted to build, the front page USES HTML, JS, CSS, Layui for writing, front and back end interaction use AngularJS implementation, using Maven jars of version control and management, database using the MySQL database, project implemented by the JAVA language. It mainly realizes the functions of user center, pig management, goods management, safety management, data export, and traceability query, which can complete the system management of ordinary pig farms, and combine the traditional pig industry with the Internet to improve the management level of the pig industry, reduce losses and improve income.

Keywords: SpringBoot; AngularJS; SpringSecurity; Layui
