**汽车维修管理系统的设计与实现**

|  |  |
| --- | --- |
| **学 院：** | 人工智能学院 |
| **专 业：** | 数据科学与大数据技术 |
| **学生姓名：** | 张若琳，葛一儒 |
| **学生学号：** | 1241004028，1241004029 |
| **指导教师：** | 刘宁 |
| **完成日期：** | 2025.7.25 |

汽车维修管理系统的设计与实现

摘 要

随着计算机技术的飞速发展，现代计算机系统已从以计算为核心转向以信息化处理为核心。对于汽车维修领域而言，不仅需要在硬件层面为人们提供汽车维修的实体平台和汽车知识获取环境，更需在软件层面为车辆提供专业的维修服务支持，汽车维修管理系统由此应运而生。 在本课题研究中，通过分析汽车维修管理人员的日常管理现状及用户对维修服务的需求，设计了一套基于SpringMVC的汽车维修管理系统。该系统采用B/S结构，以充分发挥网络信息化优势，集成开发环境为Eclipse，数据库管理系统为MySQL，Web服务器为Tomcat，并结合SpringMVC及Spring3.0框架技术实现。

系统分为专门供用户使用的前台和供汽车维修管理人员操作的后台：首页为系统欢迎界面；管理员可对车辆信息进行管理、查看、修改等操作；用户可查看和修改自身车辆信息，体验日常汽车管理流程并在线查询汽车相关信息。 该系统的实现，能让汽车维修管理人员充分利用信息化处理的优势，提升日常管理效率；用户也能更好地享受信息化带来的便利。系统运行稳定高效，且人机交互友好性显著提升。

目 录

摘 要 I

第1章　项目概述 1

1.1 问题描述 1

1.2 项目目标 1

1.3 项目适用范围 1

1.4 项目应遵守的规范与标准 1

1.5 涉众 1

第2章　需求分析 2

2.1 业务需求 2

2.1.1 业务流程 2

2.1.2 业务对象 4

2.2 功能性需求 4

2.2.1 用例概述 4

2.2.2 用例细化描述 5

2.3 非功能性需求 11

2.4 运行环境 11

第3章　系统设计 13

3.1 技术路线 13

3.2 软件体系结构设计 14

3.2.1 功能结构 14

3.2.2 系统架构 14

3.3 数据库设计 15

3.3.1 概念结构设计 15

3.3.2 表的详细设计 16

第4章　系统实现 29

4.1 核心功能实现 29

4.2 关键技术难点与解决方案 32

4.3 编码规范 33

第5章　系统测试 34

5.1 功能测试 34

5.2 测试总结 36

第6章　结论与展望 38

参考文献 39

第1章　项目概述

**1.1 问题描述**

目前现有的汽车维修管理系统对于用户而言其维修业务流程仍然过于繁琐，对于汽车维修而言其系统安全性并不能保障。同时整套系统所使用的技术相对较为落后，界面不能动态化展示。相比较于其它同类型网站而言不能体现技术先进性。

**1.2 项目目标**

汽车维修管理系统的后台开发目标是以信息管理系统的管理和开发方法，用目前现有的新技术进行系统开发，提供后台管理员高度友好的界面操作以及迅捷的信息处理。而前台的开发目标是以用户的需求作为主导，提供对用户而言非常友好的界面操作环境以及完善系统功能。

本系统主要针对现在汽车维修管理系统中存在的问题，对需要改进的地方进行系统分析，对系统所要实现的功能进行完善，使得其更具有高度界面友好的特性。提出对于系统而言具备可行性的方案，做好系统分析。编译出界面友好，功能齐全的汽车维修管理系统。

**1.3 项目适用范围**

该项目的前台适用于普通用户，后台适用于汽车维修管理人员。

**1.4 项目应遵守的规范与标准**

为了信息传输的安全性，与数据库进行交互的情况下，重要的信息比如“用户密码”，一律用密文传输。

**1.5 涉众**

在本系统中，涉及到的人员有汽车维修管理人员和用户。

对于汽车维修管理人员，其使用习惯是偏向于实用性和安全性。希望该系统能够高度安全，并且能够正常实现汽车维修管理业务的各项功能。

对于用户而言，其更倾向于界面友好性和便捷性。不仅希望系统能动态展示，还希望系统能够做到“一步到位”，其希望能实现的功能除了正常的汽车维修业务以外还希望能够网上维修业务的服务。

第2章　需求分析

**2.1 业务需求**

**2.1.1 业务流程**

在车辆信息流程中，管理员登录网站后台，汽车维修管理人员受理业务并可以添加车辆信息，其车辆信息流程总体业务如图2.1所示。



图2.1 车辆信息流程总体业务

在维修流程中，用户到网站后，车辆用户可以添加自己的汽车维修信息。

在用户注册总体业务中，用户到汽车维修注册，汽车维修管理人员将根据用户情况进行注册。

**2.2 功能性需求**

**2.2.1 用例概述**

在图2.2高层用例图中抽取了满足汽车维修和用户的基本业务需求的基本用例。由这些用例所显示的从用户使用的前台和汽车维修管理人员使用的后台来完成汽车维修的正常运作。



图2.2高层用例图

**2.2.2 用例细化描述**

（1）管理用户信息用例

汽车维修管理人员可以通过管理用户信息用例对用户信息进行维护，包括对用户信息的增删改查。管理用户信息用例图如图2.3所示。



图2.3管理用户信息用例图

（2）管理维修项目信息用例

汽车维修管理人员可以通过管理维修项目信息用例对维修项目信息进行维护，包括对维修项目信息的增删改查。管理维修项目信息用例图如图2.4所示。



图2.4 管理维修项目信息用例图

（3）管理供应商信息用例

汽车维修管理人员可以通过管理供应商信息用例对维修项目信息进行维护，包括对供应商信息的增删改查。管理供应商信息用例图如图2.5所示。



图2.5 管理供应商信息用例图

**2.3 非功能性需求**

安全性：用户信息具有保密性，用户信息只能由汽车维修管理人员控制更改使用，用户注册也只能在汽车维修管理人员处进行注册登记。

正确性：不允许出现业务错误。

兼容性：可在各种支持HTML5的浏览器中使用。

性能：系统处理业务时页面响应时间不能超过2秒，在繁忙时间最多不能超过6秒。

并发性：允许多个用户用户登录操作。

**2.4 运行环境**

汽车维修管理系统运行需要的硬件环境如下

（1）主机类型

系统主机服务器：DELL PowerEdge R630，四核CPU，8G RAM

数据库服务器：HP DL388 Gen9，四核CPU，8G RAM

用户用PC机：任意可联网设备

（2）网络类型

以太网

（3）存贮器容量

系统主机服务器：200G以上

数据库服务器：1000G以上

（4）设备分布图

汽车维修管理系统运行所需的软件环境如下。

（1）操作系统：

应用服务器：Windows NT Server 2008以上版本

用户用PC机：windows XP和以上

（2）数据库管理系统：

MySQL 5.0.27

（3）网页浏览器：

应用服务器：Google Chrome

用户用PC机：Google Chrome或者Firefox

（4）开发环境：

IDE采用Eclipse Java EE IDE Luna（4.4.1）

（5）数据库管理工具：

Navicat Premium 10.0.9

（6）运行环境：

Tomcat 7.0 + Apache

第3章　系统设计

**3.1 技术路线**

在汽车维修管理系统中，用到的技术路线如下：

B/S(Browser /Service )结构，中文译为浏览器和服务器结构。是目前国内最为常见，而且对于用户而言是最方便的一个常用结构，对以往的客户机/服务器结构（Client/Service）进行更深一层的优化。在这种结构下，用户只需要使用任意浏览器即可进行主要事物的实现和使用。其主要的逻辑在服务器端（Service）完成，而在浏览器部分只处理非常少的事物逻辑处理。可以看作从传统的二层模式结构发展而来的在C/S(Client/Service)结构在Web上的应用的三层结构。该结构适用面非常广，其对系统维护所产生的升级成本和人力时间更低，用户使用成本低，以目前的技术发展情况看，其技术相对来讲易于把握。

Spring MVC是 Spring提供用于构建 Web应用程序的轻量级全功能的 MVC模块开发框架，作为一个一站式轻量级应用开发框架（平台）， Spring抽象了在各类生活场合中的各种共性问题。Spring有其自身独有的设计理念，在Java EE的应用开发中，支持POJO和使用JavaBean的开发方式，使应用可以进行面向接口开发。其支持OO（面向对象）的设计方法。SmartAdmin是一套基于JQuery，Bootstrap构建的UI组件库，是将各种JQuery插件和Bootstrap整合到一起，是一整套基于完整的应用开发UI库。

UEditor是一款由百度开发的所见即所得富文本Web编辑器，具有轻量，可定制，注重用户体验等特点，开源基于MIT协议，允许自由使用和修改代码。

JQuery是一个快速的、简洁的JavaScript框架，其封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文房操作，时间处理、动画设计和Ajax交互。

Bootstrap，是一个来自Twitter的Web框架。目前很受Web前端开发者们的欢迎，Bootstrap是基于HTML、CSS、JavaScript而成的，以此而形成更加快捷的Web开发方式。其在JQuery的基础上进行了更加人性化和个性化的完善，形成一套Bootstrap独有网站风格。Bootstrap包含了目前非常主流的Web组件，在开发过程中可以通过这些组件快速搭建既实用又美观的网站。许多前端开发者较为熟悉的框架（如WeX5）也是基于Bootstrap进行性能优化而形成的。

**3.2 软件体系结构设计**

**3.2.1 功能结构**

根据需求调研结果确定本系统主要包括以下功能模块，如图3.1所示。

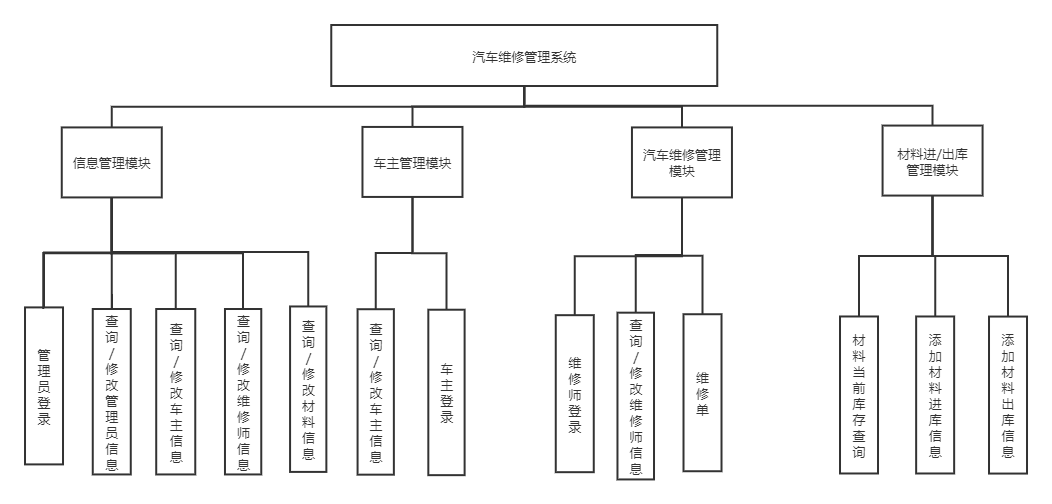


图3.1系统功能结构图

**3.2.2 系统架构**

在系统架构选型上，项目采用了 Java 生态中极具影响力的开源框架Spring 3.0。作为 Java EE 开发领域的标杆性技术，Spring 通过其独特的设计理念和技术实现，有效解决了企业级应用开发中的共性问题。相较于传统 J2EE 架构，Spring 以轻量级、非侵入式的设计风格脱颖而出，Spring 倡导基于普通 Java 对象的开发模式，支持面向接口编程，使应用设计更符合面向对象原则，避免了传统 J2EE 框架对特定接口的强依赖。IoC 容器与依赖注入，通过控制反转容器，Spring 实现了组件间依赖关系的外部化配置，显著降低了代码耦合度，提升了系统可维护性。AOP 切面编程，借助面向切面编程（AOP），Spring 将事务管理、日志记录等横切关注点与业务逻辑分离，简化了企业级应用的复杂性管理。Spring 提供了丰富的基础设施组件（如事务管理、数据访问、安全控制等），通过模块化设计降低了开发成本，并为企业服务实现提供了完整的技术栈支持。

这种架构符合现代软件开发最佳实践，为项目带来了更高的开发效率和代码质量。通过 Spring 技术赋能，保持 Java EE 企业级性特，显著提升代码可测试性和可扩展性。

**3.3 数据库设计**

**3.3.1 概念结构设计**

针对汽车维修管理系统的功能性需求方面的了解，组织系统概念模型。

**3.3.2 表的详细设计**

通过对汽车维修管理系统在需求分析中的领域对象、业务对象及对象之间的分析，进行了对汽车维修管理系统的数据库建表设计。如表3.1-表3.6 所示。汽车维修管理系统数据库管理部分采用MySQL。

（1）管理人员信息表(admin)

管理人员信息表主要用于保存汽车维修管理人员的信息，主要字段包括：管理人员ID、登录账号、登录密码。

（2）车辆信息表(user)

车辆信息表主要用于保存用户的信息，主要字段包括：用户ID、用户名称、用户登录密码、用户性别、用户电话。

（3）供应商信息表(gys)

供应商信息表主要用于保存供应商的信息，主要字段包括：供应商ID、供应商名称、供应商地址，联系电话。

（4）维修项目出库信息表(chuku)

维修项目出库信息表主要用于保存维修项目出库的信息，主要字段包括：ID、维修项目、维修单号、维修项目、维修项目编号、维修项目名称、维修项目供应商、单价。

（5）维修项目信息表(book\_article)

维修项目信息表主要用于保存维修项目的信息，主要字段包括：维修项目ID、维修项目名称、维修项目编号。

1. 入库信息表(ruku)

入库信息表主要用于保存入库的信息，主要字段包括：ID、维修项目、名称、编号、价格。

第4章　系统实现

**4.1 核心功能实现**

在汽车维修管理系统中，核心业务用例的实现效果如下所示。

1. 登陆注册模块实现

管理员通过点击后台管理，进入页面可以输入账号、密码、验证码进行登录相对应操作，如图5-1所示。

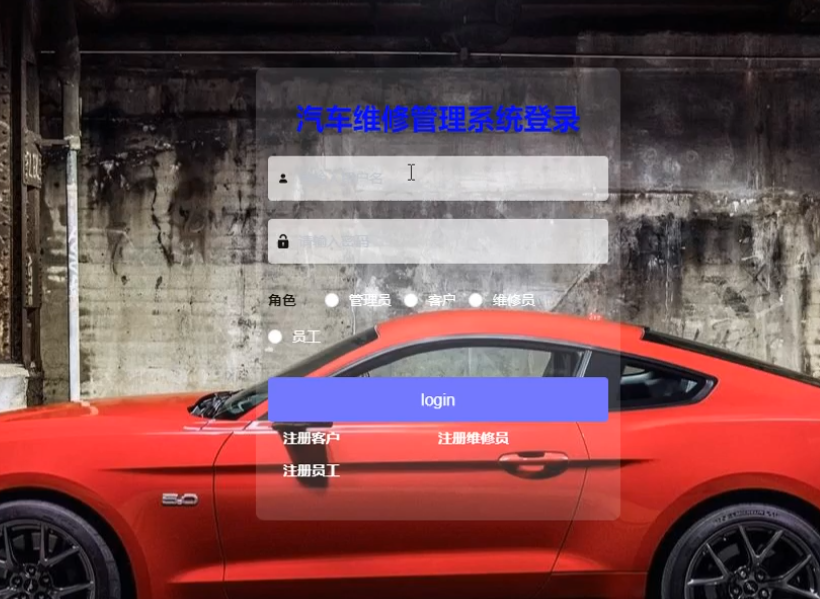


图4.1 登陆注册功能实现效果图

1. 管理人员管理维修项目功能模块

管理人员对维修项目信息进行增删改查操作，采用Ajax发送异步请求将维修项目信息发送到控制层，控制层发送数据到业务逻辑层，业务逻辑层通过调用 DAO层来访问数据库从而执行删除查询更改并返回结果。实现效果如图4.2所示。



图4.2 管理人员管理维修项目实现效果图

该模块控制层主要代码：

/维修项目列表管理页面

@RequestMapping("pinfolist")

public String listPinfo(Model model){

String w="";

if (!"".equals(request.getParameter("name")) && request.getParameter("name") !=null)

w=w+ " and name like '%"+request.getParameter("name")+"%'";

if (!"".equals(request.getParameter("bianhao")) && request.getParameter("bianhao") !=null)

w=w+ " and bianhao like '%"+request.getParameter("bianhao")+"%'";

if (!"".equals(request.getParameter("gys")) && request.getParameter("gys") !=null)

w=w+ " and gys = '"+request.getParameter("gys")+"'";

List<Pinfo> cs= pinfoService.list(w);

for(int i=0;i<cs.size();i++)

{

Pinfo pinfo=(Pinfo)cs.get(i);

}

List<Gys> gys= gysService.list("");

// 放入转发参数

model.addAttribute("gys", gys);

int index=0;

if(request.getParameter("index")==null)

{index=1;}

else

{index=Integer.parseInt(request.getParameter("index"));}

int fromIndex = (index - 1) \* Pagesize.size;

int toIndex = Math.min(fromIndex + Pagesize.size, cs.size());

List<Pinfo> cs1 = cs.subList(fromIndex, toIndex);

Pagination p = new Pagination();//创建 分页对象

p.setIndex(index);//设置页数

p.setPageSize(Pagesize.size);

p.setTotle(cs.size());//设置总共的条数

p.setData(cs1);//设置数据

// 放入转发参数

model.addAttribute("cs", p);

model.addAttribute("page", p);

model.addAttribute("dlist", cs);

// 放入转发参数

return "pages/pinfo/pinfolist";

}

@RequestMapping("pinfoadd")

public String addpinfo(Model model){

List<Gys> gys= gysService.list("");

// 放入转发参数

model.addAttribute("gys", gys);

return "pages/pinfo/pinfoadd";

}

//添加维修项目数据

@RequestMapping("pinfoinsert")

public String insertPinfo(Pinfo pinfo,Model model){

String forword=request.getParameter("forword");

pinfoService.insert(pinfo);

model.addAttribute("msg","成功");

model.addAttribute("path",forword);

return "success"; }

//删除维修项目数据

@RequestMapping("pinfodel")

public String deletePinfo(Pinfo pinfo,Model model){

String forword=request.getParameter("forword");

pinfoService.delete(pinfo);

model.addAttribute("msg","删除成功");

model.addAttribute("path",forword);

return "success";

}//提取要修改的维修项目数据

@RequestMapping("pinfoedit")

public String editPinfo(Model model,Pinfo pinfo){

Pinfo c= pinfoService.get(pinfo.getId());

List<Gys> gys= gysService.list("");

// 放入转发参数

model.addAttribute("gys", gys);

model.addAttribute("c", c);

return "pages/pinfo/pinfoedit";

} }

1. 管理人员管理车辆信息功能模块

管理人员对车辆用户进行增删改查操作，采用Ajax发送异步请求将车辆信息发送到控制层，控制层发送数据到业务逻辑层，业务逻辑层通过调用 DAO层来访问数据库从而执行更改添加删除并返回结果。

该模块控制层主要代码：

@RequestMapping("useradd")

public String adduser(Model model){

return "pages/user/useradd";

}

//添加车辆数据

@RequestMapping("userinsert")

public String insertUser(User user,Model model){

String forword=request.getParameter("forword");

userService.insert(user);

model.addAttribute("msg","成功");

model.addAttribute("path",forword);

return "success"; }

//删除车辆数据

@RequestMapping("userdel")

public String deleteUser(User user,Model model){

String forword=request.getParameter("forword");

userService.delete(user);

model.addAttribute("msg","删除成功");

model.addAttribute("path",forword);

return "success";

}//提取要修改的车辆数据

@RequestMapping("useredit")

public String editUser(Model model,User user){

User c= userService.get(user.getId());

model.addAttribute("c", c);

return "pages/user/useredit";

}

//修改的车辆数据

@RequestMapping("userupdate")

public String updateUser(User user,Model model){

String forword=request.getParameter("forword");

userService.update(user);

model.addAttribute("msg","操作成功");

model.addAttribute("path",forword);

return "success";

}

//车辆所有列表页

@RequestMapping("userAll")

public String listUserAll(Model model){

String w="";

if (!"".equals(request.getParameter("name")) && request.getParameter("name") !=null)

w=w+ " and name like '%"+request.getParameter("name")+"%'";

if (!"".equals(request.getParameter("tel")) && request.getParameter("tel") !=null)

w=w+ " and tel like '%"+request.getParameter("tel")+"%'";

if (!"".equals(request.getParameter("chexing")) && request.getParameter("chexing") !=null)

w=w+ " and chexing like '%"+request.getParameter("chexing")+"%'";

if (!"".equals(request.getParameter("chepai")) && request.getParameter("chepai") !=null)

w=w+ " and chepai like '%"+request.getParameter("chepai")+"%'";

List<User> cs= userService.list(w);

for(int i=0;i<cs.size();i++)

{

User user=(User)cs.get(i);

}

int index=0;

if(request.getParameter("index")==null)

{index=1;}

else

{index=Integer.parseInt(request.getParameter("index"));}

int fromIndex = (index - 1) \* Pagesize.size;

int toIndex = Math.min(fromIndex + Pagesize.size, cs.size());

List<User> cs1 = cs.subList(fromIndex, toIndex);

Pagination p = new Pagination();//创建 分页对象

p.setIndex(index);//设置页数

p.setPageSize(Pagesize.size);

p.setTotle(cs.size());//设置总共的条数

p.setData(cs1);//设置数据

// 放入转发参数

model.addAttribute("cs", p);

model.addAttribute("page", p);

model.addAttribute("dlist", cs);

// 放入转发参数

return "pages/user/userAll";

} }

**4.2 关键技术难点与解决方案**

1. 对于界面中要输入的大批量文字和含有特殊符号的内容的情况下，经过各种editor的比较，最后决定采用百度编辑器插件UEditor实现大批量文字输入和特殊表情。UEditor可以以高度可视化的编辑方式编辑大批量文字和含有特殊符号的内容。其可操作性和执行效率要比JavaScript的方式要高。
2. 由于登录时汽车维修管理人员和用户使用共同的登录页面，需要登录页面去判断当前登陆账号是用户还是汽车维修管理人员，进而跳转至不同的主页。因此在登录界面中，其控制层的实现方式代码如下所示。

Map<String, Object> map = userService.getUserByName(userName);//检查是否是管理员

Map<String, Object> userLogin = frontReaderService.getReaderByName(userName);//检查是否是用户

// 用户名和密码是否正确,同时检查是用户还是管理员

if (map != null && map.get("password").equals(passwordMD5)) {

HttpSession session = request.getSession(true);

session.setAttribute(Params.SESSION\_ATTR\_KEY\_USER, userName);

retMap.put(Params.RET\_CODE, Params.RET\_CODE\_SUCCESS);

retMap.put(Params.RET\_OBJ, "登录成功");

retMap.put(Params.RET\_MSG, "管理员");

return retMap;

} else {

if(userLogin != null && userLogin.get("readerPassWord").equals(passwordMD5)){

HttpSession session = request.getSession(true);

session.setAttribute(Params.SESSION\_ATTR\_KEY\_USER, userName);

retMap.put(Params.RET\_CODE, Params.RET\_CODE\_SUCCESS);

retMap.put(Params.RET\_OBJ, "登录成功");

retMap.put(Params.RET\_MSG, "用户");

return retMap;

}

**4.3 编码规范**

在该项目的实现阶段，为保证该系统的编码合乎规范，专门制定其编码规范。汽车维修管理系统的编码规范如下表4.1所示。

表4.1 编码规范表

|  |  |
| --- | --- |
| **条目** | **规范内容** |
| 1 | 包名由小写字母构成。类名由一个或者多个单词构成，且单词首字母大写。 |
| 2 | 函数(方法)名由一个或者多个单词构成，第一个单词为动词，首字母小写，后面的单词为名词，且每个单词的首字母大写。 |
| 3 | 常量名全部由大写字母构成，单词间用下划线(\_)分割。 |
| 4 | 变量名由一个或者多个单词构成，第一个单词首字母小写，后面的单词首字母大写。 |
| 5 | 参数命名和变量保持一致。 |
| 6 | Bean命名：去掉表名的下划线同时首字母大写 |
| 7 | DAO接口命名：Bean名+DAO。实现类命名：DAO接口+Impl |
| 8 | Service接口命名：模块名+Service。 实现类命名：Service接口+Impl |
| 9 | Controller类命名：模块名+ Controller |
| 10 | Bean实例及方法调用规范：Controller只允许调用Service，Service只允许调用DAO。  一个DAO对应一个数据库表。  一个Service对应一个业务模块，可以对应多个DAO。  一个Controller对应一个业务模块，可以对应多个Service。 |
| 11 | DAO层方法的命名规则如下：  插入数据：add\*  更新数据：update\*  删除数据：delete\*  查询数据：get\* |
| 12 | Service层方法的命名规则如下：  插入数据：add\*  更新数据：update\*  删除数据：delete\*  查询数据：get\*  是什么：is\*  是否存在：exists\* |
| 13 | jsp文件命名：文件名首字母小写，如果是由多个单词构成，则后面的单词首字母大写。同使用名词+动词的形式。 |

第5章　系统测试

汽车维修管理系统的测试是和系统开发同步进行的，因此在需求阶段测试人员就已经对需求阶段的各个指标进行了相对应的验证测试，在系统开发过程中进行单元测试，在系统开发完成后针对性能、功能性和可用性进行系统测试。限于篇幅所限，下面主要围绕功能测试用例和测试总结两个方面进行说明。

**5.1 测试目的**

在本系统的测试中，主要对各个的功能模块进行测试，在测试的过程之中，为了更好的测试出系统功能的关键功能和不足之处，必须设置出针对系统的各项功能特定的测试方式方法，通过特定的测验，来检查所设计的在线投票系统是否符合开始所设计所有功能，并且在发现出错时对发生错误的地方进行修正，保证系统的运行。。

**5.2** **软件测试的方式**

(1)功能测试：功能测试，顾名思义就是为了测试功能是否存在缺陷，有没有出现链接没有实现，是否出现表单没有提交成功，以及数据库是否能够连上等等，这些都是常规的功能测试内容。

(2) 性能测试：性能测试首先要测试性能是否正常，比如反应速度是否正常，如果客户访问非常慢的话，用户体验将会非常糟糕，这个工作也很重要。

(3)兼容性测试：我们都知道，在系统开发完成以后，要进行多平台的兼容性测试，比如火狐浏览器，谷歌浏览器，微软浏览器，浏览器有多个不同的版本，就需要进行多方面的测试，因为不同的版本可能会出现不同的结果，为了使它们能够在这个系统上面都可以正常使用，就需要进行兼容性测试。

(4) 安全测试：系统测试的最后一步是进行安全性测试，系统开发以后，为了确保运行正常，以及后台的正常使用，确保数据的安全，防止黑客攻击，防止数据库的注入，以及是否存在漏洞，这些都要进行安全性测试，不然一旦数据被黑客攻击，就将对整个系统造成影响，甚至造成数据的泄漏。

## 

## 5.3 测试分析

经过对系统进行有效的测试和修改，保证系统具备较好的稳定性和可用性。测试结束后，在测试总结中通过按等级统计、按类型统计对测试情况进行了客观的统计和分析，具体如图5.1-5.2所示。

按等级统计bug的结果，如图5.1所示。

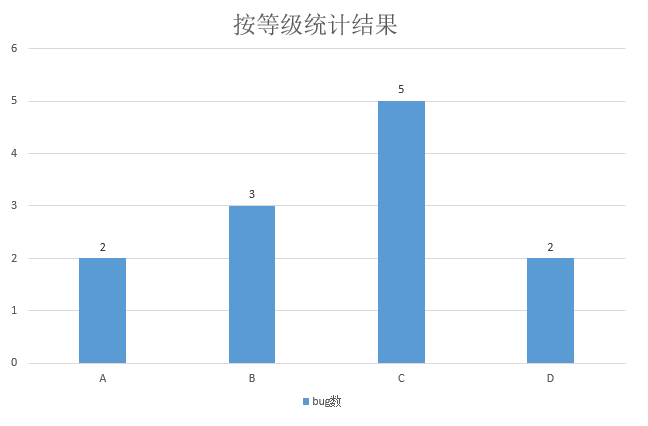


图5.1 按等级统计bug数

按类型统计bug的结果，如图5.2所示。

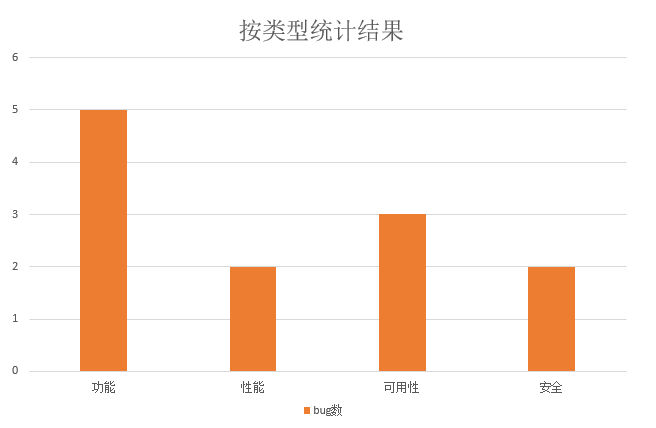


图5.2 按类型统计bug数

经过开发人员对bug进行有效的修改，保证了系统具备较好的稳定性和可用性。经过一系列测试，本软件的各项功能基本实现但是在涉及到一些细节，比如数据唯一性，安全性等存在漏洞，在软件性能方面，数据量过大软件运行会非常吃力，可用性方面不太适用于偏大型的公司数据管理。

**5.4 测试总结**

在开发期间采用的是边开发边测试的方式，使得开发周期和测试周期同步。经检测，在最后的系统测试期间，全部bug数为320个，在开发阶段的bug就有280个。在最后测试阶段的bug则只有40个。通过开发和测试共同进行的方式，最终保证系统能够具有较高的稳定性。同时通过不断的版本控制，使得系统具备较好的可用性。

系统在功能性上由于新技术所造成的不稳定性，导致了新技术所带来的功能性bug较多。但性能bug则很少，另外经过多次改进，新技术造成的错误数大幅度减少。

第6章　结论与展望

这套系统具有很多优点，不仅能完成日常库存管理的，还可以对汽车维修的业务进行扩展。不过在本项目完成的同时，由于时间短，项目紧促，造成了该系统也有很多不足的地方。整个项目仍有改进的空间。具体的改进如下：

1. 优点

本项目的优点在于项目前后台分离，各自负责不同的部分。同时还应用了较为新颖的技术。项目运用了SmartAdmin，使项目能有较高的易用性，其强大的UI库保证了界面的美观程度。页面加载部分使用Ajax方法，使得页面能够动态显示各种信息。在信息传输上使用MD5加密技术，增强信息传输的安全性。

1. 缺点

本项目的缺点在于由于计算的情况完成整套系统的时间显得非常仓促。在需求调研阶段就调研结果而言并不能高度满足于现有系统真正所需要的要求。另外在项目中，并没能很好的完全分离前后台，这一点还有待加强学习。同时，项目中由于SmartAdmin的UI库信息量非常多，其显示的并不一定是最符合其位置的UI样例。

对于汽车维修管理系统的缺点的相对应改善计划是，理清前后台的业务逻辑，使之能够做到前后台分离。而SmartAdmin中我们可以通过加强对JQuery的使用对其进行补充。使得项目能够有更好的易用性。针对SmartAdmin UI库过多的问题，可以对其进行库设计，从而达到快速引用的设计。

通过这几个月的不断努力和学习，最终完成了整个项目。对此有很多的感触。项目中应用到了不少的新技术。有很多都不在学校的课堂中学习到而是通过不断的实践获得的。随着技术的进步，我们还需要不断的学习与探索，会有更好的天地等着我们去实现。

参考文献

[1] 陈韶健．深入实践Spring Boot [M]，机械工业出版社，2016．

[2] 袁方．浅谈网络环境下汽车管理现状及对策[J]，安阳师范学院学报，2016，(4)，154-156．

[3] 岳倩倩，李美莲．汽车管理系统的设计与实现[J]，软件设计开发，2016，(8)，105-107．

[4] 王龙军．基于新浪云计算和Android平台的移动汽车维修研究与应用[J]，广东省对外科技交流中心，2016，(8)，58-60．

[5] 陈英．基于java的汽车管理的现状分析及创新途径探讨[J]，中国中南传媒，2016，(9)，136-137.

[6] 王军．计算机网络环境下的大学汽车维修管理研究[J]，中国宇航出版社，2016，(10)， 115-117．

[7] 周珊．汽车维修电子使用评价与分析[J]，供应商情报导刊，2016，(6)，57-77．

[8] 孙微微．浅析汽车维修管理系统的设计[J]，决策与信息，2016，(9)，239-239．

[9] 贺振．浅议汽车维修管理系统的设计与实现[J]，管理自动化，2016，(9)，262-263．

[10] Shameer Kunjumohamed．Spring Essentials [M]，Packt Publishing，2016．