**说明书**

**项目软件系统设计**

目录

[1.软件系统总体结构 1](#_Toc74063257)

[2.数据库逻辑结构 2](#_Toc74063258)

[2.1数据库说明 3](#_Toc74063259)

[2.2数据库逻辑结构 4](#_Toc74063260)

[3 重用方案 6](#_Toc74063261)

[4.关键类的重点服务 7](#_Toc74063262)

# 1.软件系统总体结构

本系统采用B/S(Browser/Server)三层体系结构作为系统的总体结构，由浏览器Web服务器和数据库服务器组成，并综合运用HTML，CSS，JavaScript，Spring

Spring MVC,Mybatis等技术，由Web Server统一进行管理和发送，用户通过Web浏览器以HTTP协议向服务器发出请求，并接受和显示服务器提供的Web信息，具体如图1-1所示。

系统总体结构图1-1所示：

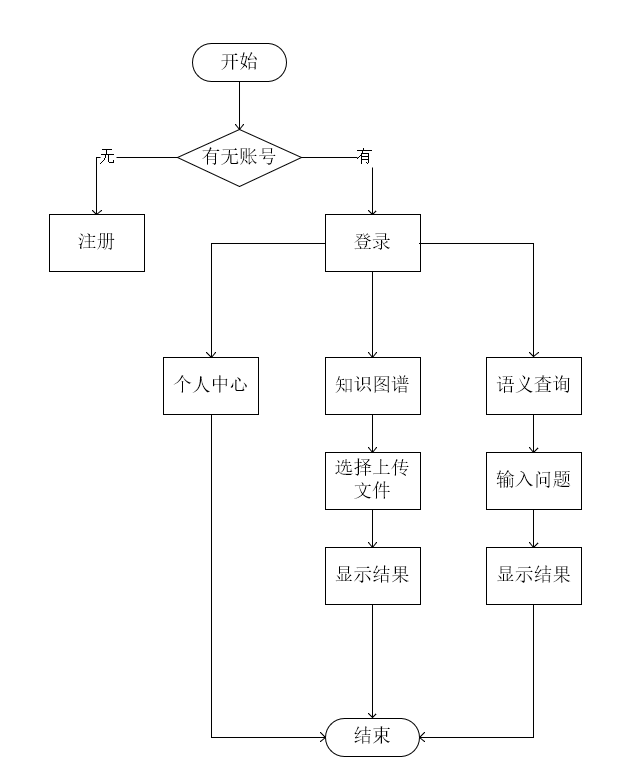
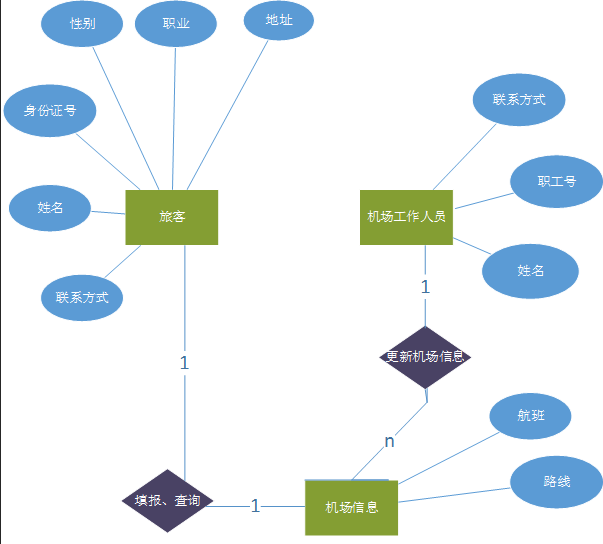


图1-1 系统结构图

# 2.数据库逻辑结构

数据库的设计关系到整个应用系统的运行效率，数据库设计得好，不仅有利于日常数据的维护更新，而且可以提高系统的运行效率，缩短数据查询响应周期，增加网站的流量。合理的数据库设计可以使围绕它支持的Web页面的Java代码简单化，易于实现，并且可以提高数据存储的效率，保证数据的完整一致。校园二手商品交易系统采用MySQL作为后台数据库开发工具。

图1-2 ER图

## 2.1数据库说明



图2-1 数据库表

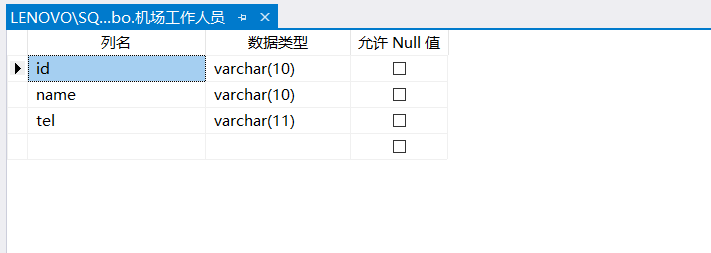


图2-2 机场工作人员表

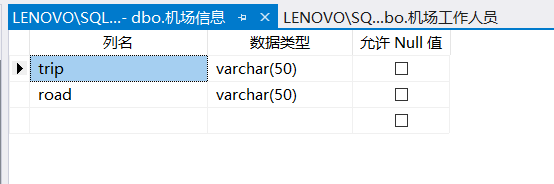


图2-3 机场信息表

概念模型用于信息世界的建模，与具体的DBMS无关。为了把现实世界中的具体事物抽象、组织为某一DBMS支持的数据模型。人们常常首先将现实世界抽象为信息世界，然后再将信息世界转换为机器世界。也就是说，首先把现实世界中的客观对象抽象为某一种信息结构，这种信息结构并不依赖于具体的计算机系统和具体的DBMS，而是概念级的模型，然后再把模型转换为计算机上某一个DBMS支持的数据模型。实际上，概念模型是现实世界到机器世界的一个中间层次。

信息世界中包含的基本概念有实体和联系。

（1）实体

客观存在并可相互区别的事物称为实体。实体可以是具体的人、事、物，也可以是抽象的概念或联系。例如，一个学生、一门课、一个供应商、一个部门、一本书、一位读者等都是实体。

（2）联系

在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常是组成实体的各属性之间的联系。两个实体型之间的联系可分为3类，一对一联系，(1:1)；一对多联系(1:n)；多对多联系(m:n)。

概念模型是对信息世界建模，所以概念模型应该能够方便、准确地表示信息世界中的常用概念。概念模型的表示方法很多，其中最为常用的是P.P.S.Chen于1976年提出的实体，联系方法(Entity-Relationship Approach)简记为E-R表示法)。该方法用E-R图来描述现实世界的概念模型，称为实体-联系模型，简称E-R模型。

根据数据流程分析，绘制校园二手商品交易系统的全局E-R模型如图2-1所示。

根据系统分析的主要实体有：用户、管理员、商品、关注、分类和信息。各个实体具体的描述属性图如下（实体属性在下图中并没有全部给出，因为属性过多的原因）：

## 2.2数据库逻辑结构

数据库逻辑设计主要是把数据库概念设计时设计好的基本E-R图转换为与选用DBMS产品所支持的数据模型相符合的逻辑结构。它包括数据项、记录及记录间的联系、安全性和一致性约束等等。导出的逻辑结构是否与概念模式一致，从功能和性能上是否满足用户的要求，要进行模式评价。

本系统数据库名称为db\_secondhandmarket，数据库中包括：

1）用户表（user）

2）管理员表（admin）

3）商品表（goods）

4）关注表（focus）

5）分类表（catelog）

6）信息表（notice）

各表数据结构如下：

（1）用户表(user)，存储用户信息。

**表 2-1 用户表数据结构**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 用户ID |
| phone | char(11) | N | Y | 手机号 |
| username | varchar(30) | N | Y | 用户名 |
| password | char(32) | N | Y | 密码 |
| create\_at | varchar(20) | N | Y | 创建时间 |
| power | int(10) | N | N | 信用分，普通用户默认为100 |
| last\_login | varchar(20) | N | Y | 最近一次登陆时间 |
| status | tinyint(4) | N | N | 账号是否冻结，默认0未冻结 |

（2）管理员表（admin），存储管理员信息。

**表 2-2 管理员表数据结构**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 管理员ID |
| username | varchar(25) | N | Y | 账号 |

# 3 重用方案

从广义的角度来说，本系统在以下三个层次进行了重用，知识的重用，方法和标准的重用以及软件成分的重用。

知识的重用是如软件工程等相关知识的重用，应用相同的知识设计规划、建设不同的软件产品。方法和标准的重用是比如本产品是利用面向对象方法设计又或者是采用国家标准局制定的软件开发文档规范。

软件成分的重用进一步我们划分为三个级别，即代码重用，设计结果重用和分析结果重用。而本软件产品的重用主要在源代码级，这与我们的开发工具有很大的关系，我们选用的开发工具是pycharm，采用spring MVC框架。利用该框架可以快速开发出一个程序的框架，是一个灵活性较强的框架。

我们的重用设计是建立在MVC框架之上。我们分析可能重用部分的实例有两种方式，一种是继承类库中的构件要用到的基本功能的类。主要是一些界面元素，如菜单活框、列表框，这些构件在很多模块中都要用到，且处理逻辑大致相同，并进行扩充其处理逻辑，如增加输入合法性检测等，这样我们在使用这些经过扩充的构件时不必每个都去重复那些处理逻辑，大大提高了效率，而且，这些构件都是经过测试或其他人使用过的，质量也有保证。

另一种方式是我们的构件在类库找不到相似的类，我们将从头创建自己的类，但为了将自己创建的类，如用户权限处理排序打印预览等，也纳入MVC的框架，我们自己创建的类都继承框架的一个抽象类，这样做的好处是把自己创建的类纳入Controller类的层次化管理，且这也是为实现虚拟函数和动态联编方便。我们也为每个件的功能和接口建立了文档，供应用开发中使用。这样我们就在MVC的框架下，建立了我们自己的一个重用构件库。最后是在应用中重用构件库中的构件。当用户需求变化时，我们能用构件库快速重构我们的应用。

使用重用明显会将开发效率提高一个层次，它不仅提高了开发效率，还保证了应用的风格和质量。特别是对我们这种新手较多的团队开发，采用重用的方法，让有经验的成员负责整个应用的框架，让新手使用重用来创建应用，这非常有利于提高效率，在保证质量的同时也为新手提供一个循序渐进的学习机会，有利于新手的成长。当然使用重用容易，但自己建立重用构件库也是需要代价的，是一个需时间积累的过程，也有一定风险。我们在重用中也遇到了一些问题，一个比较突出的问题是我们的重用主要是在源代码级，因此我们的开发必须在MVC的框架下。

我们的重用主要是在源代码级，通过类的继承来实现。其实可重用的范围是很大的，如设计的重用，测试用列的重用，可运行的代码的重用等。我们想将来扩大重用的粒度，在框架基础上，进一步根据我们单位的软硬件环境定制出一个适用于我们系统的框架，如加进注册功能，数据库连接的功能等。这样，我们可直接重用这个框架，这可以极大地提高软件的开发和维护的效率。最后，我们还设想，将来应有专门的人员管理重用，把重用的维护和应用开发分开，责任明确这样可以更进一步地做好重用工作。

# 4.关键类的重点服务

本系统主要设计了三个大的关键类:admin,user,goods,即管理员类，用户类，基于这三个父类的基础上，继承更行，查询，修改等子类。

Admin类主要负责对管理员的属性修改等操作服务，其属性包括用户名，密码，手机号等。

User类主要负责对用户的属性信息操作服务。

Comments类主要负责对用户评论的操作服务。

以上为本系统设计关键类的对应重点服务，遵循软件工程中软件设计的模块化原则，力求尽量达到“高内聚，低耦合”的程度，降低类的复杂度，且使方法属性单一，逻辑关系简单。这样可以提高类的可读性，降低功能模块之间的关联性。