

**本文档详细描述了停车收费系统的设计架构，以及数据库等开发方案的说明。**

**说明书**

**项目软件系统设计**

**缔造小组**

**项目计划书修订版（V1.1）**

目录

[1.软件系统结构 1](#_Toc73737848)

[1.1总体框架 1](#_Toc73737849)

[1.2子模块的设计 1](#_Toc73737850)

[1.2.1模块功能 2](#_Toc73737851)

[1.2.2模块结构 2](#_Toc73737852)

[1.2.3输入输出 4](#_Toc73737853)

[1.2.4处理流程 5](#_Toc73737854)

[1.2.5异常处理 8](#_Toc73737855)

[1.2.6界面原型 8](#_Toc73737856)

[1.3 接口设计 13](#_Toc73737857)

[1.3.1外部接口 13](#_Toc73737858)

[1.3.2内部接口 14](#_Toc73737859)

[2.数据库设计 14](#_Toc73737860)

[2.1数据库说明 14](#_Toc73737861)

[2.2数据库逻辑结构 17](#_Toc73737862)

[3. 软件重用方案 18](#_Toc73737863)

[3.1 重用与移植 18](#_Toc73737864)

[4.设计关键类的重点服务 19](#_Toc73737865)

# 1.软件系统结构

## 1.1总体框架

本停车收费系统系统采用SSM（Spring MVC+Spring+Mybatis）框架开发，是标准的MVC模式，将整个系统划分为View层，Controller层，Service层，DAO层四层。其中，Spring MVC负责请求的转发和视图管理，Spring实现业务对象管理，Mybatis作为数据对象的持久化引擎。整个系统架构运行流程如图1-1所示：



**图1-1 系统架构运行图**

View层：与Controller层结合比较紧密，需要二者结合起来协同工发，主要负责前台Jsp页面的表示。

Controller层：控制器，导入service层，因为service中的方法是我们使用到的，controller通过接收前端传过来的参数进行业务操作，在返回一个指定的路径或者数据表。

Service层：存放业务逻辑处理，也是一些关于数据库处理的操作，但不是直接和数据库打交道，它有接口还有接口的实现方法，在接口的实现方法中需要导入Dao层，Dao层是直接跟数据库打交道的，它也是个接口，只有方法名字，具体实现在mapper.xml文件里，service是供我们使用的方法。

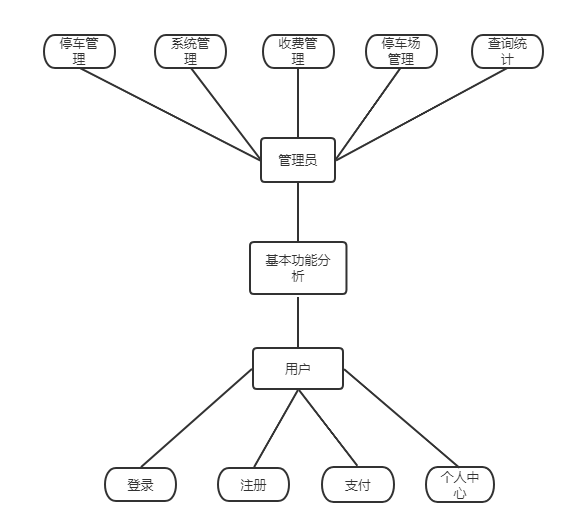
Dao层：对数据库进行数据持久化操作，它的方法语句是直接针对数据库操作的，而service层是针对我们controller，也就是针对我们使用者。service的impl是把mapper和service进行整合的文件。

## 1.2子模块的设计

实质上，停车收费系统的综合性相对较强，复杂程度相对较高，可对现有软件进行充分利用，进行系统设计与规划。构建完善成熟的停车收费系统，其中涉及到以下内容，即前台网页界面、处理程序、MySQL 后台数据库系统等，处理程序其实也就是对用户提交表单与相关操作进行处理，存储在后台数据库的信息有用户数据、订单数据和钱包数据等。

### 1.2.1模块功能

系统按照功能主要分为注册用户、管理员两个模块。注册用户除了基本的停车缴费和个人中心功能外，还有用户注册、用户登录和个人信息设置功能。管理员具有对用户信息、订单信息的管理功能和系统基本设置功能。整个系统模块功能如图1-2所示：

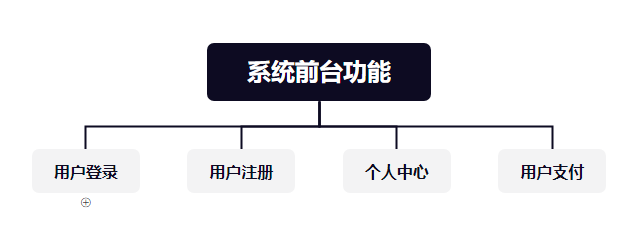


**图1-2 系统模块功能图**

### 1.2.2模块结构

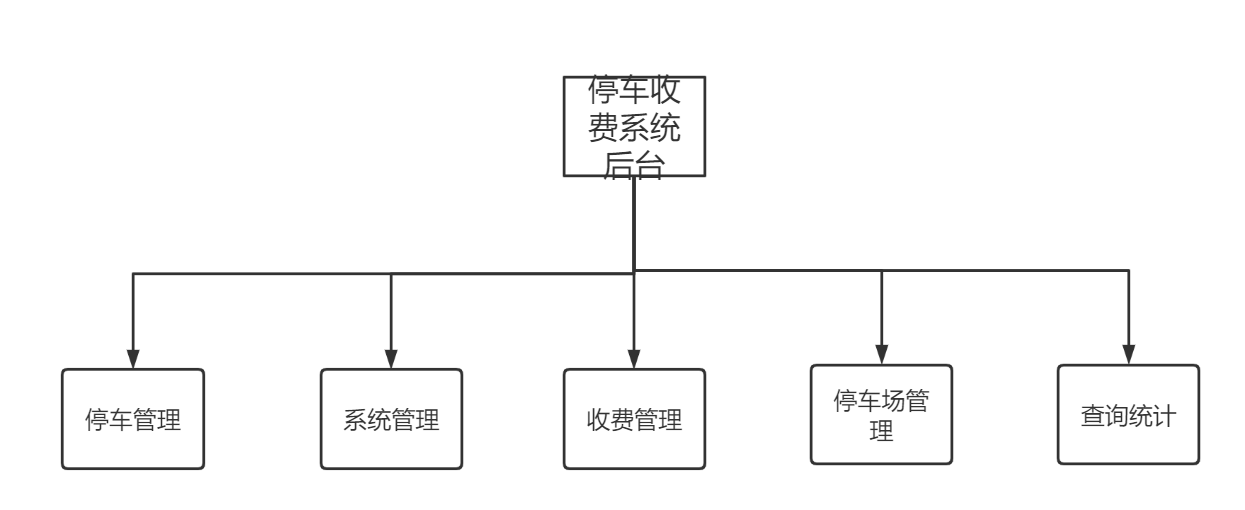
系统总体结构可分为系统前台和系统后台两个功能模块。

前台功能实现以下功能，用户注册、用户登录、个人中心、用户支付等。系统前台功能如图1-3所示：



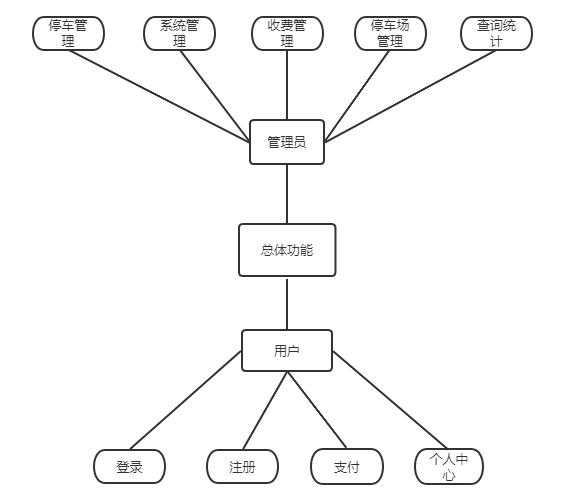
**图1-3 系统前台功能模块结构图**

系统后台功能实现以下功能，停车管理、系统管理、收费管理、停车场管理、查询统计等。系统后台功能如图1-4所示：

****

**图1-4 系统后台功能模块结构图**

系统总体功能结构如图1-5所示：

****

**图1-5 系统总体功能结构图**

### 1.2.3输入输出

输入输出（input/output，I/O）描述的是在计算机上输入输出数据的操作系统、程序或设备。一般的输入输出设备有打印机、硬盘、键盘和鼠标。实际上，有些设备只有输入功能，如键盘和鼠标；有些设备只有输出功能，如打印机；还有些设备具有输入输出2种功能，如硬盘、磁碟和可写性只读光盘（CD-ROM）。

本文主要从程序的输入输出项目来分析一下该系统的输入输出，该系统程序的输入项目如表1-1所示：

**表 1-1 输入项目表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 输入方式 | 数据来源 |
| 1 | 手机号 | char(11) | 手动输入 | 用户 |
| 2 | 用户名 | varchar(30) | 手动输入 | 用户 |
| 3 | 密码 | char(32) | 手动输入 | 用户 |
| 4 | 微信号码 | varchar(12) | 手动输入 | 用户 |

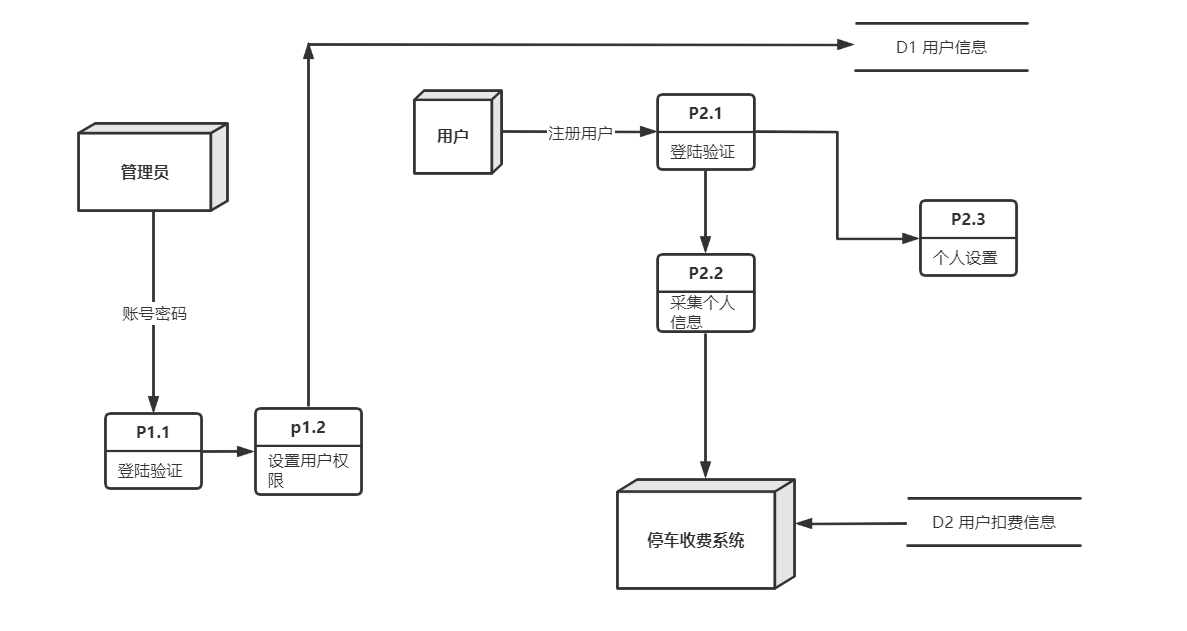
该系统程序的输出项目如表1-2所示：

**表 1-2 输出项目表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 数据来源 |
| 1 | 信用分 | int(10) | 管理员 |
| 2 | 账号是否冻结（默认0未冻结密码） | varchar(30) | 管理员 |
| 3 | 状态（停车1 未停车0） | char(32) | 用户、管理员 |

在数据输入输出的基础上对系统数据流向进行分析就比较容易了。首先，数据流程图（Data Flow Diagram，DFD/Data Flow Chart），是一种能全面地描述系统数据流程的主要工具，它用一组符号来描述整个系统中信息的全貌，综合地反映出信息在系统中的流动、处理和存储情况。

其次，它有两个特征：抽象性和概括性。抽象性指的是数据流程图把具体的组织机构、工作场所、物质流都去掉，只剩下信息和数据存储、流动、使用以及加工情况。概括性则是指数据流程图把系统对各种业务的处理过程联系起来考虑，形成一个总体。综合这两个特征对该系统的数据流向进行分析，系统的整体数据流图如图1-6所示：

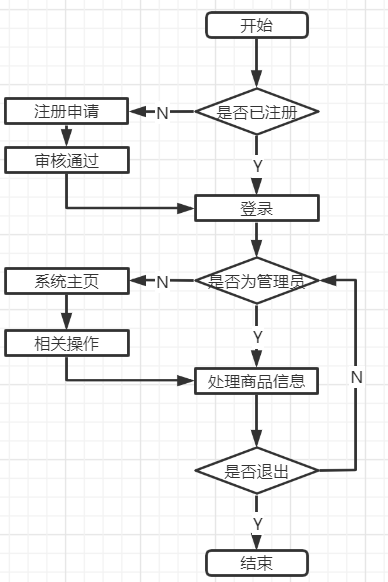


**图1-6 系统数据流图**

### 1.2.4处理流程

（1）登录管理模块

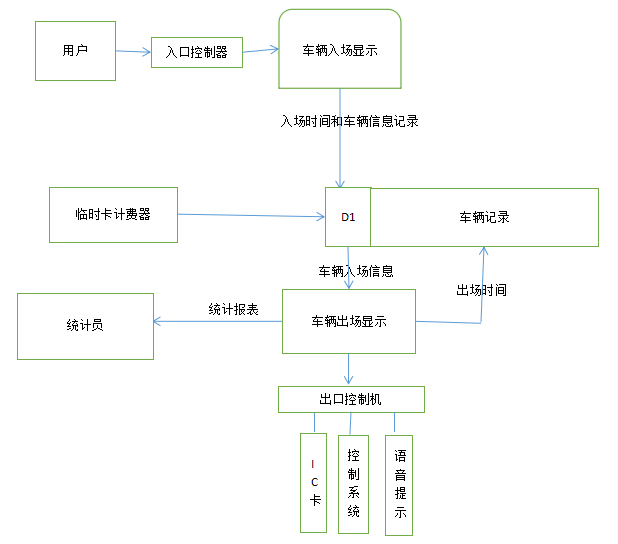
为用户提供用户登录功能；对用户是否注册、是否为管理员进行判断，给系统管理员提供添加、删除用户功能以及查看修改商品信息功能。具体登录管理模块流程如图所示：



**图1-7 登录管理处理流程图**

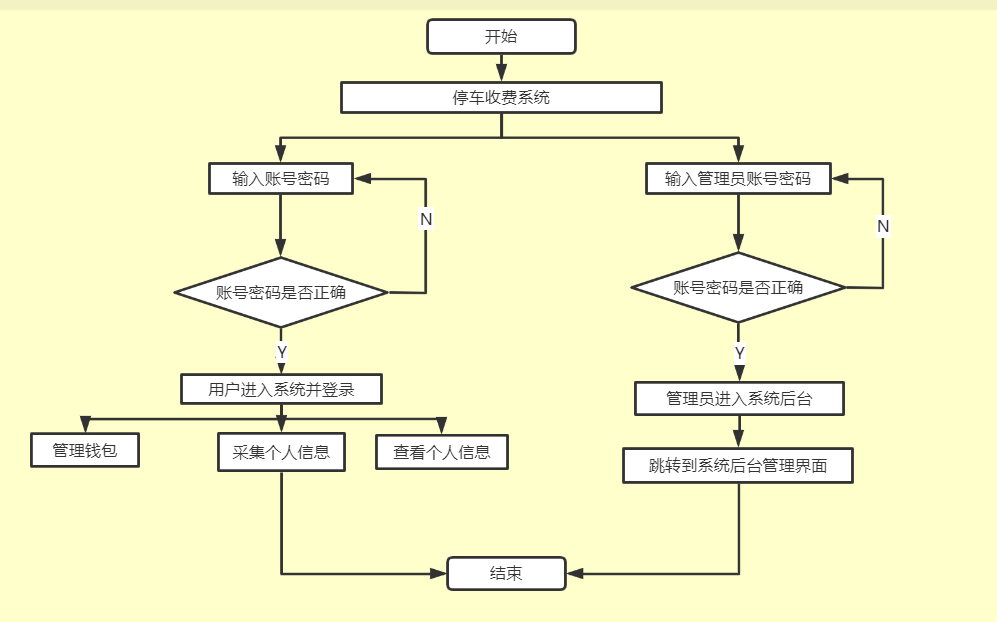
（2）停车收费流程模块

为已登录用户提供发布商品以及个人信息设置等功能。具体商品交易模块流程如图1-8所示：



**图1-8 停车收费流程图**

1. 停车收费系统操作流程



**图1-9 操作流程图**

### 1.2.5异常处理

异常处理（又称为错误处理）是编程语言或计算机硬件里的一种机制，用于处理软件或信息系统中出现的异常状况（即超出程序正常执行流程的某些特殊条件）。其功能提供了处理程序运行时出现的任何意外或异常情况的方法。本系统的异常处理机制如下：

（1）在应用程序遇到异常情况（如被零除情况或内存不足警告）时，就会产生异常。发生异常时，控制流立即跳转到关联的异常处理程序（如果存在）。

（2）如果给定异常没有异常处理程序，则程序将停止执行，并显示一条错误信息。

（3）可能导致异常的操作通过try关键字来执行。

（4）程序可以使用throw关键字显式地引发异常。

（5）异常对象包含有关错误的详细信息，其中包括调用堆栈的状态以及有关错误的文本说明。

（6）即使引发了异常，finally块中的代码也会执行，从而使程序可以释放资源。

（7）异常处理使用try、catch和finally关键字来尝试可能未成功的操作，处理失败，以及在事后清理资源。异常处理通常是防止未知错误产生所采取的处理措施。

采用异常处理机制的好处是开发者不用再绞尽脑汁去考虑各种错误，这为处理某一类错误提供了一个很有效的方法，使项目软件开发效率大大提高。

异常由公共语言运行库(CLR)、第三方库或使用throw关键字的应用程序代码生成。用户也可以用自定义的异常处理类来扩展php内置的异常处理类。以下的代码说明了在内置的异常处理类中，哪些属性和方法在子类中是可访问和可继承的。

### 1.2.6界面原型

界面原型方法是快速地建立可运行、概略的系统，之后不断评价、改进直到获得最终系统。利用原型系统，用户可以爱和开发者一起看到未来交互的软件蓝图、功能和效果，获得较真实的感受，在不断讨论的基础上完善软件未来的设计。不能等到代码写好之后再去验证用户需求，采用原型系统展示用户界面和功能需求。

原型设计与开发不仅要考虑系统核心功能，还要考虑功能的页面排布、用户实际操作中的问题，要提前为用户考虑得当并征求用户意见；原型系统是必须可运行的，可实际使用的系统。本系统界面原型使用墨刀原型开发工具设计实现，

（1）首页界面：



**图1-9 首页界面原型图**

1. 登录界面



**图1-10 登录界面原型图**

（3）注册界面



**图1-11 注册界面原型图**

（4）绑定银行卡界面



**图1-12 绑定银行卡界面**

## 1.3 接口设计

本系统软件的接口设计遵循六大基本原则：分别是：单一职责原则、依赖倒置原则、接口隔离原则、迪米特法则、里氏替换原则、开闭原则。这6个原则是建立稳定的，灵活和健壮设计的基础，但设计的时候还要根据实际情况考虑。

（1）单一职责原则(SRP)：该原则要求我们在设计类或者接口的时候。尤其在设计接口的时候把职责分清楚，通常一个职责不是单一的方法，是一类方法的组合。

（2）依赖倒置原则：为高层模块不应该依赖底层模块，他们都应该依赖抽象。抽象不能依赖细节。细节应该依赖抽象，这个原则要求设计的时候尽量用抽象（抽象类或者接口）把各个模块独立开来，实现解耦，使各模块相对独立。简单来说就是要用面向接口设计。

（3）接口隔离原则(ISP)：接口分两种，分别为类实例接口和类接口。这个原则依赖建立在最小的接口之上，依赖自己需要的接口。

（4）迪米特法则(LoD)：也称为最少知识原则。就是一个对象应该对其他对象有最少的了解。这个法则的本质就是解耦，解耦是有限度的不能为了解耦而解耦。

（5）里氏替换原则：（所有引用基类的地方，都能透明地使用其子类的对象。）简单来说就是父类出现的地方，替换为子类不会产出异常。里氏替换原则的好处就是增强程序的健壮性，保持程序的兼容性。

（6）开闭原则：软件的实体如类，模块和函数应该对扩展开发，对修改关闭。这个原则要求我们设计的系统扩展性好，因为需要不会一直不变的，我们需要应对的永远是变化。

### 1.3.1外部接口

（1）用户界面：在界面设计上，应做到简单明了，易于操作，并且要注意到界面的布局，应突出的显示重要以及出错信息。外观上也要做到合理化，考虑到用户多对Windows风格较熟悉，所以该系统尽量向这一方向靠拢。

（2）软件与硬件接口：本系统设有人机操作界面，考虑到操作简单，易于管理方面，主要硬件与接口设备为pc、鼠标、键盘。而软件接口主要以Windows平台为基本平台。

### 1.3.2内部接口

内部接口也被称为嵌套接口，这意味着在另一个接口内声明一个接口。例如，Entry接口声明在Map接口中。由于各模块之间相互独立又彼此关联，系统主要通过函数调用实现各部分连接。

# 2.数据库设计

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。

## 2.1数据库说明

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

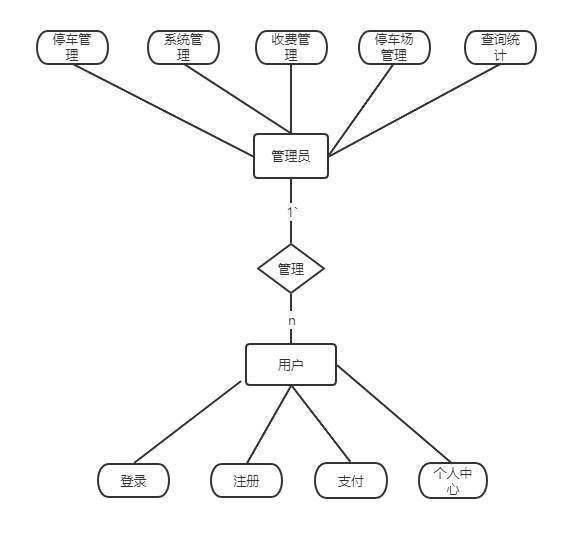
客观存在并可相互区别的事物称为实体。实体可以是具体的人、事、物，也可以是抽象的概念或联系。例如，一个学生、一门课、一个供应商、一个部门、一本书、一位读者等都是实体。

（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。

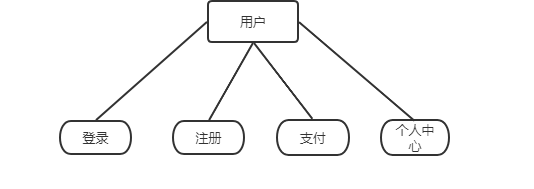
根据数据流程分析，绘制停车收费系统的全局E-R模型如图2-1所示。



**图2-1 系统全局E-R图**

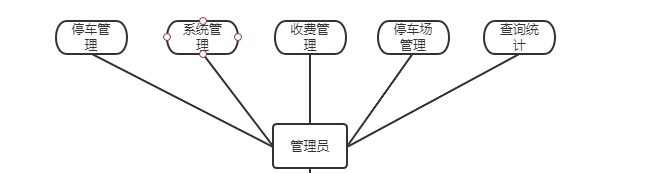
根据系统分析的主要实体有：用户、管理员、商品、关注、分类和信息。各个实体具体的描述属性图如下（实体属性在下图中并没有全部给出，因为属性过多的原因）：

1.用户实体



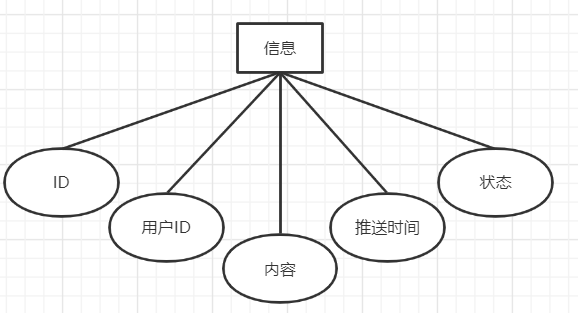
**图 2-2 用户实体图**

2.管理员实体



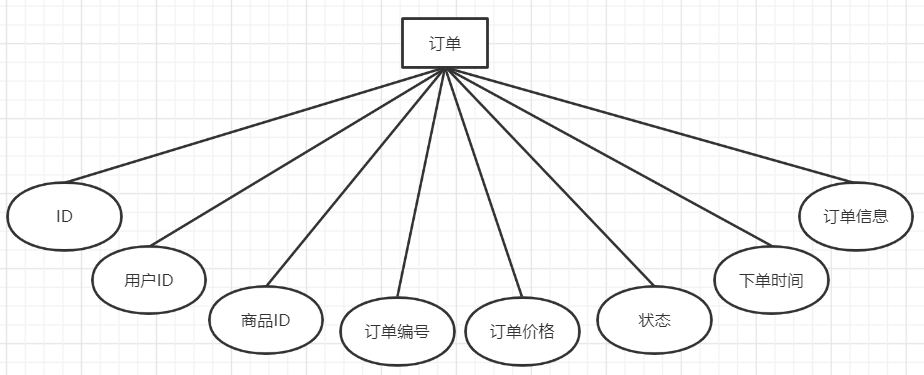
**图 2-3 管理员实体图**

3.信息实体



**图 2-4 信息实体图**

4.信息实体



**图 2-5 订单实体图**

## 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）订单表（focus）

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 |
| QQ | varchar(12) | N | Y | QQ号码 |
| create\_at | varchar(20) | N | Y | 创建时间 |
| goods\_num | int(11) | N | N | 发布过的物品数量 |
| 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 |
| password | varchar(25) | N | N | 密码 |
| phone | bigint(25) | N | N | 手机号 |
| userRole | varchar(25) | N | Y | 角色 |

（3）订单表(focus)，存储订单信息。

**表 2-3 订单表数据结构**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 订单ID |
| user\_id | int(11) | N | Y | 用户ID，外键 |

# 软件重用方案

## 3.1 重用与移植

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eclipse和IDE，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。

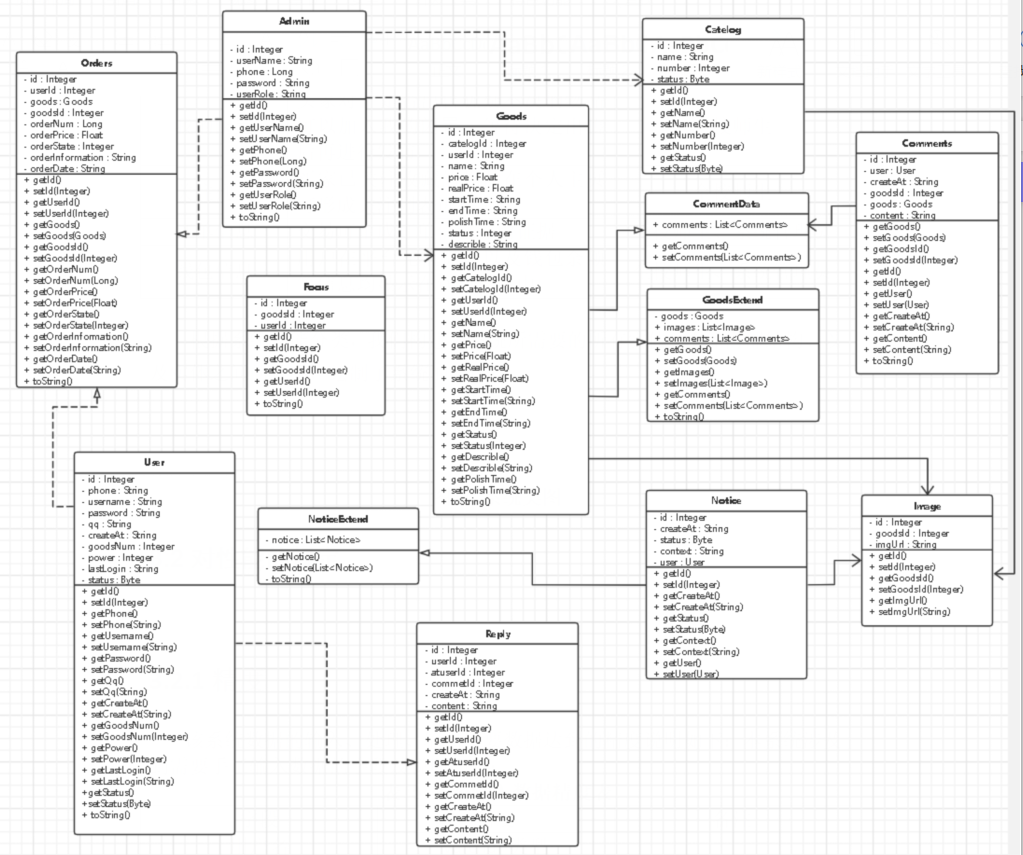
使用重用明显会将开发效率提高一个层次，它不仅提高了开发效率，还保证了应用的风格和质量。特别是对我们这种新手较多的团队开发，采用重用的方法，让有经验的成员负责整个应用的框架，让新手使用重用来创建应用，这非常有利于提高效率，在保证质量的同时也为新手提供一个循序渐进的学习机会，有利于新手的成长。当然使用重用容易，但自己建立重用构件库也是需要代价的，是一个需时间积累的过程，也有一定风险。我们在重用中也遇到了一些问题，一个比较突出的问题是我们的重用主要是在源代码级，因此我们的开发必须在MVC的框架下。

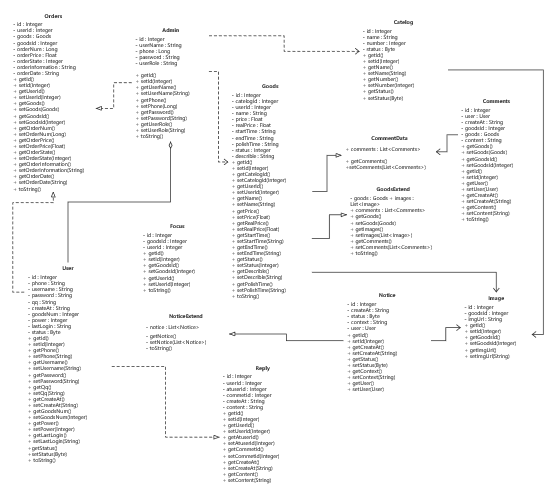
我们的重用主要是在源代码级，通过类的继承来实现。其实可重用的范围是很大的，如设计的重用，测试用列的重用，可运行的代码的重用等。我们想将来扩大重用的粒度，在框架基础上，进一步根据我们单位的软硬件环境定制出一个适用于我们系统的框架，如加进注册功能，数据库连接的功能等。这样，我们可直接重用这个框架，这可以极大地提高软件的开发和维护的效率。最后，我们还设想，将来应有专门的人员管理重用，把重用的维护和应用开发分开，责任明确这样可以更进一步地做好重用工作。

# 4.设计关键类的重点服务

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，停车收费类。基于这三个父类的基础上，继承更行，查询，修改等子类。

设计的类图如图2-9所示：





**图 2-7 类图**

Admin类主要负责对管理员的属性修改等操作服务。

User类主要负责对用户的属性信息操作服务。

Catelog类主要负责对商品交易的操作服务。

CommentData类主要负责对车辆信息查询的反馈操作。

Comments类主要负责对用户评论的操作服务。

Focus类主要负责对用户关注车辆表单的操作服务。

GoodsExtend类是goods类的一个子类，主要负责查询车主的操作服务。

Image类也为Goods的一个关联类，其负责对发布商品交易的操作服务。

Orders类主要负责对的停车用户管理操作服务。

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。