# 摘 要

随着我国经济的迅猛发展，第三产业在国民经济中扮演着越来越重要的角色。今年我国旅游行业规模不断扩大，因而现代酒店组织日渐庞大，服务项目增多、信息量加大已经成为现代酒店行业的一大特点。作为一个有一定规模的酒店，想要在这样的大趋势下不断发展并取得一定的成果，这就要求酒店必须要有一个适合自身的信息管理平台。这样酒店才能在日渐增大的大趋势下不断提高自身的竞争力，以取得更高的利益。

酒店客房管理系统正是为此而生，本系统是基于JavaEE项目结构设计的。以JSP作为前台页面展示层，动态显示数据；Spring+MyBatis框架作为后台数据处理层，实现数据存储；并且采用了MVC的开发模式，实现模块间的解耦。本系统使用Eclipse IDE开发，依托Tomcat 7.0服务器启动，底层数据库采用MySql数据库。系统主要功能包括操作员登录注册、客户信息管理、订单管理、客房信息管理、账单信息管理。提供操作员和管理员两种登录权限。

这样可以大大减少顾客入住酒店时的等待时间，操作员可以快速的将顾客信息录入系统和高效的管理客房状态。体现了酒店管理系统高效、方便的特点。

关键词：客房管理；JSP；MyBatis；Tomcat；MySql数据库

# Abstract

With the rapid development of China's economy, the third industry plays a more and more important role in the national economy. This year, the scale of China's tourism industry has been expanding, so the organization of Moderno Hotel is growing, the number of service projects increased, the amount of information has become a major feature of the Moderno Hotel industry. As a certain size of the hotel, want to continue to develop and achieve certain results in such a big trend, which requires the hotel must have a suitable information management platform for their own. So that the hotel can continue to improve their competitiveness in the growing trend, in order to obtain a higher interest.

Hotel guest room management system is to this end, the system is based on the structure of the JavaEE project design. To JSP as the front page display layer, dynamic display data; Spring+MyBatis framework as the background data processing layer, data storage; and the use of the MVC development model, to achieve the decoupling between modules. The system uses IDE Eclipse development, relying on the Tomcat 7 server startup, the underlying database using MySql database. The main functions of the system include operator login registration, customer information management, order management, room information management, billing information management. Provide operators and administrators two kinds of login rights.

This can greatly reduce the waiting time of the customer Check Inn Hotel, the operator can quickly enter the customer information system and efficient management of the state of the room. Embodies the hotel management system, efficient and convenient features.

Key words：Room management; MyBatis; Tomcat; MySql Date Base

# 目 录

摘 要 2

Abstract 3

目 录 4

1 系统开发环境及技术简介 5

1.1 系统开发思想和技术 6

1.1.1 Java简介 6

1.1.2 JSP技术简介 6

1.1.3 MVC设计模式简介 7

1.1.4 MyBatis框架介绍 8

1.1.5 Spring框架介绍 9

1.2 系统开发环境 11

1.2.1 IDE介绍 11

1.2.2 Eclipse 11

1.2.3 Tomcat服务器介绍 12

1.2.4 MySql数据库介绍 13

1.3 本章小结 13

2 系统总体设计 14

2.1 系统总体功能模块 14

2.2 系统操作员功能模块 15

2.3 系统用户功能模块 15

2.4 系统业务描述 15

2.5 系统详细流程 16

2.6 本章小结 17

3 系统数据库设计 18

3.1 数据库的配置 18

3.2 数据库逻辑模型 18

3.3 数据库表结构 20

3.4 系统数据库表创建 23

3.5 本章小结 26

4 系统功能实现 27

4.1 系统类图 27

4.1.1 操作员信息实体类 27

4.1.2 用户信息实体类 27

4.1.3 客户信息实体类 28

4.1.4 订单信息实体类 28

4.1.5 登记信息实体类 29

4.1.6 客房类型实体类 29

4.1.7 客房信息实体类 29

4.1.8 预定信息实体类 30

4.1.9 结账信息实体类 30

4.2 系统功能实现的主要框架 30

4.2.1 MyBatis框架 30

4.2.2 Spring框架配置文件 30

4.3 系统功能实现的主要模块 32

4.3.1 系统登录页面 32

4.3.2 信息管理模块 33

4.3.3 订单管理模块 37

4.3.4 客房信息管理 38

4.3.5 账务管理 40

4.4本章小结 42

5 软件测试 43

5.1 软件测试的目的 43

5.2 软件测试的任务 44

5.3 测试环境的搭建 44

5.3.1 搭建安装项目 44

5.3.2 添加项目并运行 45

5.4 本章小结 45

结论 46

参考文献 47

致谢 48

# 1 系统开发环境及技术简介

本章介绍了酒店客房管理系统的开发环境以及使用到的Java编程技术。

## 1.1 系统开发思想和技术

### 1.1.1 Java简介

Java是一门面向对象的编程语言，不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好的实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

Java具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多态性、动态性等特点。Java可以编写桌面应用程序、Web应用程序、分布式系统和嵌入式系统应用程序等。

JDK（Java Development Kit）称为Java开发包或Java开发工具，是一个编写Java的Applet小程序和应用程序的程序开发环境。JDK是整个Java的核心，包括了Java运行环境（Java Runtime Environment），一些Java工具和Java的核心类库（Java API）。不论什么Java应用服务器实质都是内置了某个版本的JDK。主流的JDK是Sun公司发布的JDK，除了Sun之外，还有很多公司和组织都开发了自己的JDK，例如，IBM公司开发的JDK，BEA公司的Jrocket，还有GNU组织开发的JDK 。

JRE是运行环境，JDK是开发环境。因此写Java程序的时候需要JDK，而运行Java程序的时候就需要JRE。而JDK里面已经包含了JRE，因此只要安装了JDK，就可以编辑Java程序，也可以正常运行Java程序。但由于JDK包含了许多与运行无关的内容，占用的空间较大，因此运行普通的Java程序无须安装JDK，而只需要安装JRE即可。

### 1.1.2 JSP技术简介

JSP（Java Server Page）是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP技术优点类似ASP技术，它是在传统的网页HTML文件（\*.html,\*.htm）中插入Java程序段（Scriptlet）和JSP标签（tag），从而形成JSP文件（\*.jsp）。

用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他操作系统上运行。

JSP技术使用Java编程语言编写类XML的tags和scriptlets，来封装产生动态网页的处理逻辑。网页还能通过tags和scriptlets访问存在于服务端的资源的应用逻辑。JSP将网页逻辑与网页设计和网页显示分离，支持可重用的基于组件的设计，使基于Web的应用程序的开发变得迅速和容易。

JSP技术的强势：

1. 一次编写，到处运行。
2. 系统的多平台支持。基本上可以在所有平台上的任意环境中开发，在任意环境中进行系统部署，在任意环境中扩展。
3. 强大的可伸缩性。
4. 多样化和功能强大的开发工具支持。

JSP技术的弱势：

1. 与ASP一样，Java的一些优势正是它致命的问题所在。正是由于为 了跨平台的功能，为了极度的伸缩能力，所以极大的增加了产品的复杂性。
2. Java的运行速度使用class常驻内存来完成的，所以它在一些情况下所使用的内存比起用户数量来说确实是“最低性能价格比”了。从另一方面，它还需要硬盘空间存储一系列的.java文件和.class文件，以及对应版本文件。

### 1.1.3 MVC设计模式简介

MVC即Model-View-Controller，即把一个应用的输入、处理、输出流程按照Model、View、Controller的方式进行分离。这样一个应用被分成三个层——模型层、视图层、控制层。

视图（View）代表用户交互界面，对于Web应用来说，可以概括为HTML界面，但有可能为XHTML、XML和Applet。随着应用的复杂性和规模性，界面的处理也变得具有挑战性。一个应用可能有很多不同的视图，MVC设计模式对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理交予模型（Model）处理。比如一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制和模型。

模型（Model）就是业务流程／状态的处理以及业务规则的制定。业务流程的处理过程对其它层来说是黑箱操作，模型接受视图请求的数据，并返回最终的处理结果。业务模型的设计可以说是MVC最主要的核心。目前流行的EJB模型就是一个典型的应用例子，它从应用技术实现的角度对模型做了进一步的划分，以便充分利用现有的组件，但它不能作为应用设计模型的框架。它仅仅告诉你按这种模型设计就可以利用某些技术组件，从而减少了技术上的苦难。对一个开发者来说，就可以专注于业务模型的设计。MVC设计模式告诉我们，把应用的模型按一定规则抽取出来，抽取的层次很重要，这也是判断开发人员是否优秀的设计依据。抽象与具体不能隔得太远，也不能太近。MVC并没有提供模型的设计方法，而只告诉你应组织管理这些模型，以便于模型的重构和提高重用性。我们可以用对象编程来比喻，MVC定义了一个顶级类，告诉它的子类你只能做这些，但没法限制你能做这些。这点对开发人员非常重要。

业务模型还有一个很重要的模型那就是数据模型。数据模型主要指实体对象的数据保存（持久化）。比如将一张订单保存到数据库，从数据库获取订单。我们可以将这个模型单独列出，所有有关数据库的操作只限制在该模型中。

控制（Controller）可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。划分控制层的作用也很明显，它清楚的告诉你，它就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层并不做任何的数据处理。例如，用户点击一个连接，控制层接受请求后，并不处理业务信息，它只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

### 1.1.4 MyBatis框架介绍

Mybatis本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAO）。

MyBatis应用程序根据XML配置文件创建SqlSessionFactory，SqlSessionFactory在根据配置（配置来源于两个地方，一处是配置文件，一处是Java代码的注解）获取一个SqlSession。SqlSession包含了执行sql所需要的所有方法，可以通过SqlSession实例直接运行映射的sql语句，完成对数据的增删改查和事务提交等，用完之后关闭SqlSession。

MyBatis优点：

1. 简单易学MyBatis本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar文件和配置几个SQL映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。
2. 灵活。Mybatis不会对应用程序或数据库的现有设计强加任何影响。SQL写在XML中，便于统一管理和优化。通过SQL基本上可以实现我们不使用数据访问框架可以实现的所有功能，或许更多。
3. 解除SQL与程序代码的耦合。
4. 提供映射标签，支持对象与数据库的ORM字段关系映射。
5. 提供对象关系映射标签，支持对象关系组件维护。
6. 提供XML标签，支持编写动态SQL。

MyBatis缺点：

1. 编写SQL语句时工作量很大，尤其是字段多、关联表多时，更是如此。
2. SQL语句依赖于数据库，导致数据库移植性差，不能更换数据库。
3. 二级缓存机制不佳。

### 1.1.5 Spring框架介绍

随着软件结构的日益庞大，软件模块化趋势出现，软件开发也需要很多人合作，随即分工出现。如何划分模块，如何定义接口方便分工成为软件工程设计中越来越关注的问题。良好的模块化具有以下优势：可扩展、易验证、易维护、已分工、易理解、代码复用。

优良的模块设计往往遵守“低耦合，高内聚”的原则。而“框架”是对开发中良好设计的总结，把设计中经常使用的代码独立出来，所形成的一种软件工具。用户遵守它的开发原则，就可以实现良好的模块化，避免软件开发中潜在的问题。广义上的框架无处不在，一个常见的例子就是PC硬件体系结构，人们只要按照各自需要的主板、显卡、内存等器件就可以任意的组装成自己想要的电脑。而做主办的厂商不用关心不用关心做显卡的厂商是怎么实现显卡的功能。软件框架也是如此，开发人员只要在Spring框架中填充自己的业务逻辑就能完成一个模块划分清晰的系统。

Spring框架是一个分层架构，由7个定义良好的模块组成，如图1-1所示：

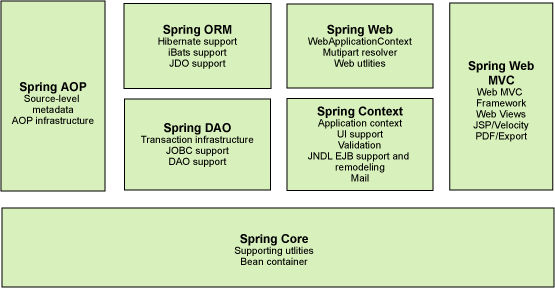


图 1-1 Spring框架的7个模块

1. 核心容器：核心容器提供Spring框架的基本功能。核心容器的主要组件是BeanFactory，它是工厂模式的实现。BeanFactory使用控制反转（IOC）模式将应用程序的配置和依赖性规范与实际的应用程序代码分开。
2. Spring上下文：Spring上下文是一个配置文件，向Spring框架提供上下文信息。
3. Spring AOP：通过配置管理特性，Spring AOP模块直接将面向方面的编程功能集成到了Spring框架中。所以，可以很容易的使Spring框架管理的任何对象支持AOP。
4. Spring DAO：JDBC DAO抽象层提供了有意义的异常层次结构，可用该结构来管理异常处理和不同数据库供应商抛出的错误消息。
5. Spring ORM：Spring框架插入了若干个ORM框架，从而提供了ORM的对象关系工具，其中包括JDO、Hibernate和MyBatis SQL Map。
6. Spring Web模块：Web上下文模块建立在应用程序上下文模块之上，为基于Web的应用程序提供了上下文。
7. Spring MVC框架：MVC框架是一个全功能的构建Web应用程序的MVC实现。通过策略接口，MVC框架成为高度可配置的，MVC容纳了大量视图技术，其中包括JSP、Velocity、iText和POI。

## 1.2 系统开发环境

### 1.2.1 IDE介绍

IDE全称是“Integrated Development Environment”，中文称为“集成开发环境”。是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具。就是继承了代码编写功能、分析功能、编译功能、调试功能等一体化的软件开发服务套。

Java领域常见的IDE：

1. NetBeans,免费开源
2. Jbuilder，收费
3. IntellijIDEA，分社区版和旗舰版
4. Eclipse，免费

### 1.2.2 Eclipse

Eclipse是一个开放源代码、基于Java的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。Eclipse中“万物接插件”。

Eclipse开发环境称为Workbench，它主要由多个视图组成：常用的有编辑窗口视图、Java包结构浏览视图、大纲浏览视图、控制台视图等。如图1-2所示：

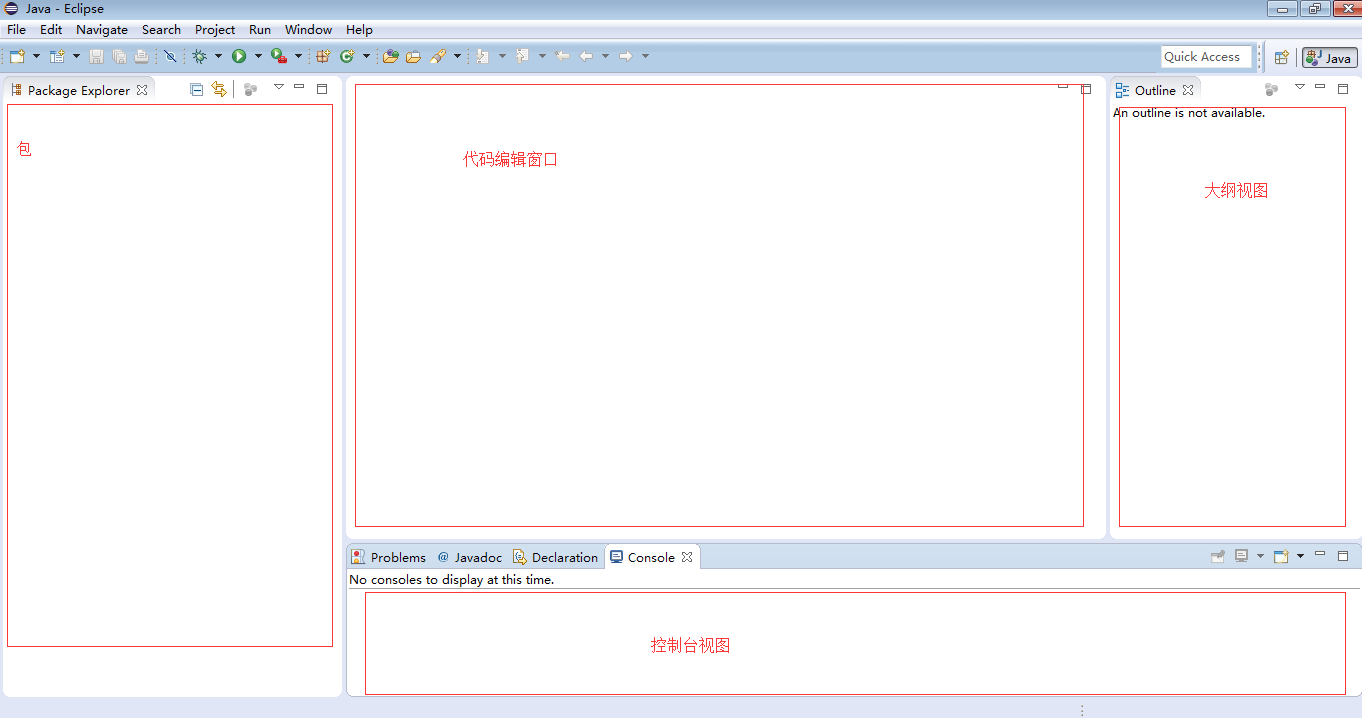


图 1-2 Eclipse Workbench结构

### 1.2.3 Tomcat服务器介绍

Jakarta Tomcat服务器是在sun公司的JSWDK（Java Server Web DevelopmentKit，sun公司推出的小型Servlet／Jsp调试工具）的基础上发展起来的一个优秀的Java Web容器。  
 Tomcat很受广大程序员的喜爱，因为它运行时占用的系统资源小，扩展性好，支持负载平衡与邮件服务等开发应用系统常用的功能；而且它还在不断的改进和完善中，任何一个感兴趣的程序员都可以更改它或在其中加入新的功能。

Tomcat是一个小型的轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试Jsp程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache服务器，可利用它响应对HTML页面的访问请求。实际上Tomcat部分是Apache服务器的扩展，但它是独立运行的，所以当你运行Tomcat时，它实际上作为一个与Apache独立的进程单独运行。

### 1.2.4 MySql数据库介绍

MySql是一个精巧的SQL数据库管理系统，虽然它不是开放源代码的产品，但在某些情况下你可以自由使用。由于它的强大功能、灵活性、丰富的应用编程接口（API）以及精巧的系统结构，受到了广大自由软件爱好者甚至是商业软件用户的青睐，特别是与Apache和PHP／PERL的结合，为建立基于数据库的动态网站提供了强大动力。

MySQL是一个真正的多用户、多线程SQL数据库服务器。SQL（结构化查询语言）是世界上最流行的和标准化的数据库语言。MySQL是一个以客户机／服务器结构的实现，它由一个服务器守护程序mysqld和很多不同的客户程序和库组成。

SQL是一种标准化的语言，它使得存储、更新和存取信息更为容易。例如：你能用SQL语言为一个网站检索产品信息及存储顾客信息，同时MySQL也足够快和灵活以允许你存储记录文件和图像。

MySQL数据库的主要功能只在组织和管理很庞大或复杂的信息和基于WEB的库存查询请求不仅仅为客户提供信息，而且还可以为自己使用数据库提供以下功能：

1. 减少记录编档时间
2. 减小记录检索时间
3. 灵活的查找序列
4. 灵活的输出格式
5. 多个用户同时访问记录

## 1.3 本章小结

本章主要主要介绍了一些酒店客房管理系统开发过程中将会使用的到的开发技术和开发工具。俗话说，工欲善其事，必先利其器。只有真正的掌握了并且可以熟练的运用这些可以带给我们方便的技术和工具，才可以保证我们开发的效率和质量。

# 2 系统总体设计

本章介绍了酒店客房管理系统的总体设计和业务流程。系统总体设计的任务是进行系统模块结构设计。系统模块结构设计的任务是划分子系统，然后确定子系统的模块结构，并画出模块结构图。

酒店客房管理系统采用的是B／S架构。所谓B/S架构就是浏览器和服务器架构。它是随着Internet技术的兴起，对C/S架构（客户端／服务器架构）的一种变化或者改进的多结构。B/S结构是WEB兴起后的一种网络结构模式，WEB浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。

C/S架构的优点就是基于客户机和服务器模式，大部分的运算是在客户机上进行，从而提高一定的效率，但是也因为它在客户机上运行，所以要安装客户端，所以也导致了软件维护和升级的不方便。而B/S架构的优点在于软件维护和升级方便，只要能上网就能升级且在很多地方都能够直接使用。

## 2.1 系统总体功能模块

本系统分系统操作员和系统用户俩个角色。

总功能图如图2-1所示：

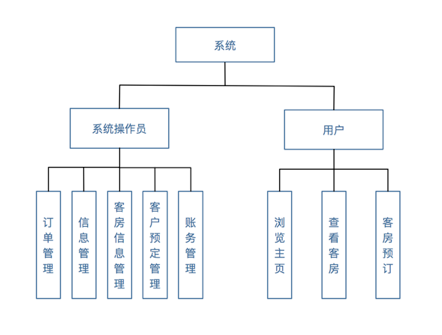


图 2-1 系统功能需求分析

## 2.2 系统操作员功能模块

系统操作员的主要功能有订单管理、用户信息管理、客房信息管理、客户预定管理、账务管理。系统操作员的功能图如图2-2所示：

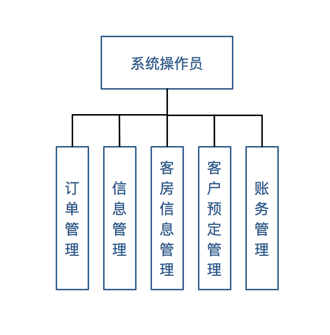


图 2-2 系统操作员功能图

## 2.3 系统用户功能模块

系统用户主要功能有浏览主页、查看客房、客房预订。

系统用户功能图如图2-3所示：

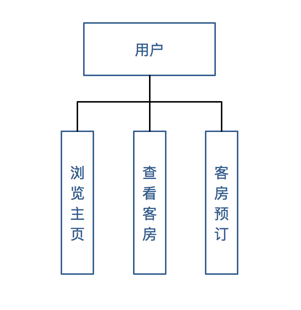


图 2-3 系统用户功能图

## 2.4 系统业务描述

本系统分为系统操作员和系统用户俩个角色。系统操作员的功能有：浏览酒店主页、查看客房、开房、退房、信息录入、注册登录等功能；系统用户的功能有：浏览酒店主页、查看客房、客房预订、注册登录等功能。

用例图如图2-4所示：

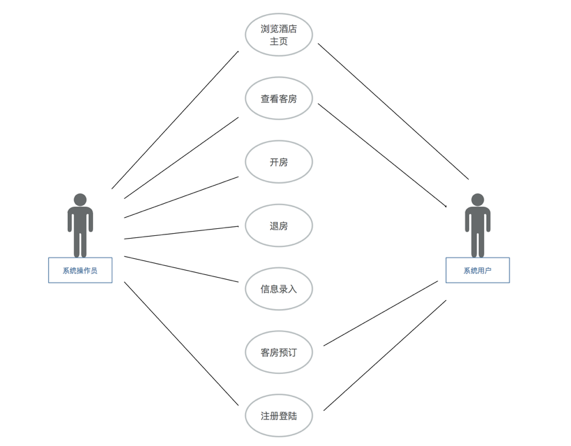


图 2-4 系统用例图

## 2.5 系统详细流程

用户注册流程图如图2-5所示：

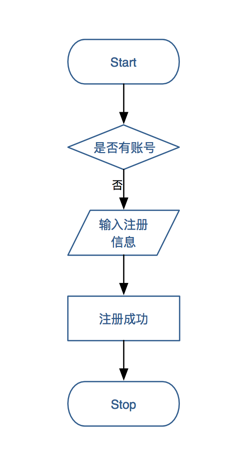


图 2-5 用户注册流程图

用户登录流程图如图2-6所示：

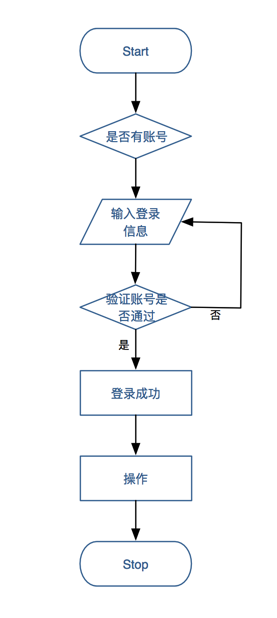


图 2-6 用户登录流程图

## 2.6 本章小结

本章介绍了酒店客房管理系统的总体设计和业务流程。主要目的是为了让用户可以清楚的了解本系统的各功能，以便快速上手使用。

# 3 系统数据库设计

## 3.1 数据库的配置

数据库好比人的大脑的记忆系统，没有了数据库就么有了记忆系统。本系统中数据库的配置信息如图3-1所示：



图 3-1 数据库的配置

## 3.2 数据库逻辑模型

账单信息表（主键：结账信息编号）如图3-2所示：

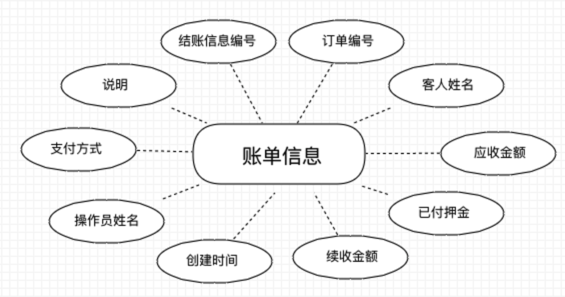


图 3-2 账单信息逻辑图

入住登记信息表（主键：入住登记ID）如图3-3所示：

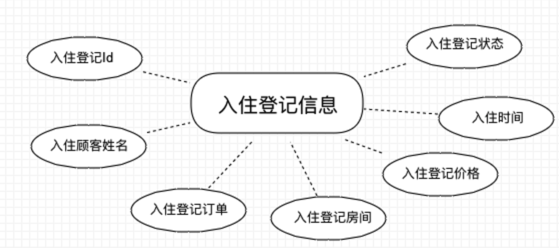


图 3-3 入住登记信息逻辑图

账单信息表（主键：客户ID）如图3-4所示：



图 3-4 账单信息逻辑图

操作员信息表（主键：操作员用户名）如图3-5所示：



图 3-5 操作员信息逻辑图

订单信息表（主键：入住登记订单编号）如图3-6所示：



图 3-6 订单信息逻辑图

客户预订信息表如图3-7所示：

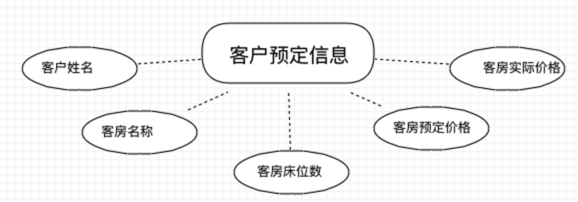


图 3-7 客户预订信息逻辑图

客房信息表（主键：客房编号）如图3-8所示：

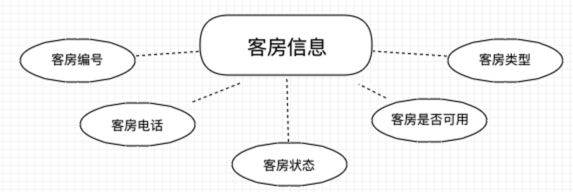


图 3-8 客房信息逻辑图

客房类型表（主键：客房ID）如图3-9所示：

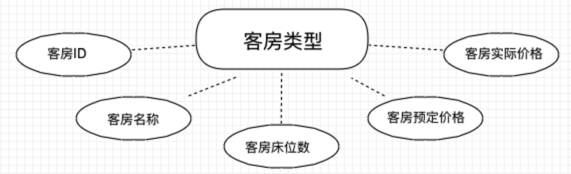


图 3-9 客房类型逻辑图

用户信息表（主键：用户名）如图3-10所示：

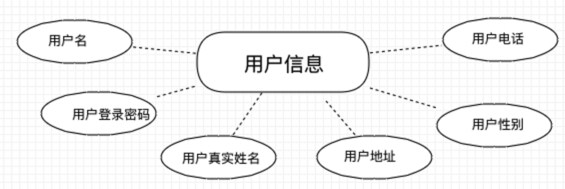


图 3-10 用户信息逻辑图

## 3.3 数据库表结构

结账信息表用来保存系统所有结账的信息。

结账信息表如表3-1所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| bmid | varchar(255) | no |  | 主键，结账信息编号 |
| bmorderid | varchar(255) | yes | 空 | 订单编号 |
| bmguestname | varchar(255) | yes | 空 | 客人姓名 |
| bmtotelrate | varchar(255) | yes | 空 | 应收金额 |
| bmpaidmoney | varchar(255) | yes | 空 | 已付押金 |
| bmreceivmoney | varchar(255) | yes | 空 | 续收金额 |
| bmcreatetime | varchar(255) | yes | 空 | 创建时间 |
| bmoperatorname | varchar(255) | yes | 空 | 操作员姓名 |
| bmpaymenttype | varchar(255) | yes | 空 | 支付方式 |
| bmremark | varchar(255) | yes | 空 | 说明 |

表格 3-1 结账信息表

入住登记信息表用来保存顾客的入住登记信息。

入住登记信息表如表3-2所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| id | varchar(255) | no |  | 入住登记Id |
| name | varchar(255) | yes | 空 | 入住顾客姓名 |
| orderId | varchar(255) | yes | 空 | 入住登记订单 |
| roomId | varchar(255) | yes | 空 | 入住登记房间 |
| price | varchar(255) | yes | 空 | 入住登记价格 |
| inTime | varchar(255) | yes | 空 | 入住时间 |
| state | varchar(255) | yes | 空 | 入住登记状态 |

表格 3-2 入住登记信息表

顾客信息表用来保存顾客入店是的基本信息。

顾客信息表如表3-3所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| id | varchar(255) | no | auto\_increment | 顾客Id |
| name | varchar(255) | yes | 空 | 顾客姓名 |
| cardType | varchar(255) | yes | 空 | 顾客证件类型 |
| cardId | varchar(255) | yes | 空 | 顾客证件号码 |
| country | varchar(255) | yes | 空 | 顾客国籍 |
| address | varchar(255) | yes | 空 | 顾客家庭住址 |
| zip | varchar(255) | yes | 空 | 顾客邮编 |
| phoneNumber | varchar(255) | yes | 空 | 顾客电话 |
| gender | varchar(255) | yes | 空 | 顾客性别 |
| email | varchar(255) | yes | 空 | 顾客邮箱 |
| createTime | varchar(255) | yes | 空 | 顾客信息创建时间 |

表格 3-3 顾客信息表

操作员账号注册信息表用来保存操作员的基本注册信息。

操作员信息表如表3-4所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| username | varchar(255) | no | auto\_increment | 操作员用户名 |
| passwd | varchar(255) | yes | 空 | 操作员密码 |
| privilege | varchar(255) | yes | 空 | 操作员权限 |
| address | varchar(255) | yes | 空 | 操作员住址 |
| name | varchar(255) | yes | 空 | 操作员姓名 |
| phoneNumber | varchar(255) | yes | 空 | 操作员电话 |
| zip | varchar(255) | yes | 空 | 操作员邮编 |
| email | varchar(255) | yes | 空 | 操作员邮箱 |
| createTime | varchar(255) | yes | 空 | 操作员创建时间 |

表格 3-4 操作员注册信息表

订单表保存了顾客入住时的订单信息。

订单表如表3-5所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| id | varchar(255) | no |  | 入住登记订单编号 |
| name | varchar(255) | yes | 空 | 顾客名字 |
| num | varchar(255) | yes | 空 | 入住人数 |
| guestcardtype | varchar(255) | yes | 空 | 顾客证件类型 |
| guestcardnumber | varchar(255) | yes | 空 | 顾客证件号码 |
| discription | varchar(255) | yes | 空 | 备注 |
| state | varchar(255) | yes | 空 | 登记状态 |
| indate | varchar(255) | yes | 空 | 顾客入住时间 |
| preoutdate | varchar(255) | yes | 空 | 顾客预计离开时间 |
| paytype | varchar(255) | yes | 空 | 支付类型 |
| prepay | varchar(255) | yes | 空 | 定金 |
| isreserv | varchar(255) | yes | 空 | 是否有预定 |
| operator | varchar(255) | yes | 空 | 操作员Id |
| guestgender | varchar(255) | yes | 空 | 顾客性别 |
| totalrate | varchar(255) | yes | 空 | 实际金额 |
| orderId | varchar(255) | yes | 空 | 登记单号 |

表格 3-5 订单信息表

顾客预定信息表存储了顾客的预定信息。

顾客预定信息表如表3-6所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| name | varchar(255) | no |  | 顾客姓名 |
| roomname | varchar(255) | yes | 空 | 客房名称 |
| bednumber | varchar(255) | yes | 空 | 客房床位数 |
| preprice | varchar(255) | yes | 空 | 客房预订价格 |
| price | varchar(255) | yes | 空 | 客房实际价格 |

表格 3-6 顾客预定信息表

客房信息表存储了客房的基本信息。

客房信息表如表3-7所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| id | varchar(255) | no |  | 客房编号 |
| phoneNumber | varchar(255) | yes | 空 | 客房电话 |
| state | varchar(255) | yes | 空 | 客房状态 |
| usable | varchar(255) | yes | 空 | 客房是否可用 |
| typeid | varchar(255) | yes | 空 | 客房 |

表格 3-7 客房信息表

客房类型表存储了所有的客房类型信息。

客房类型表如表3-8所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| id | varchar(255) | no |  | 客房Id |
| name | varchar(255) | yes | 空 | 客房名称 |
| bedNumber | varchar(255) | yes | 空 | 客房床位数 |
| preprice | varchar(255) | yes | 空 | 客房预订价格 |
| price | varchar(255) | yes | 空 | 客房实际价格 |

表格 3-8 客房类型表

顾客注册信息表存储了顾客注册网站会员时的基本信息。

顾客注册信息表如表3-9所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 备注 |
| username | varchar(255) | no |  | 登录用户名 |
| passwd | varchar(255) | yes | 空 | 登录密码 |
| name | varchar(255) | yes | 空 | 顾客真实姓名 |
| address | varchar(255) | yes | 空 | 顾客家庭住址 |
| gender | varchar(255) | yes | 空 | 顾客性别 |
| phone | varchar(255) | 空 | 默认值 | 顾客电话 |

表格 3-9 顾客注册信息表

## 3.4 系统数据库表创建

根据系统的功能划分，该系统总共涉及9个实体信息，分别是结账信息、入住登记信息、顾客信息、操作员注册信息、订单信息、顾客预定信息、客房信息、客房类型、顾客注册信息。因此系统共有9张表，分别是：

结账信息表：h\_balancement

建表语句如图3-11所示：

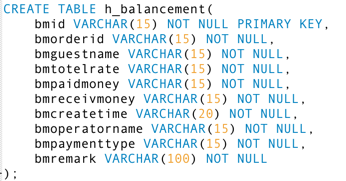


图 3-11 结账信息表建表语句

入住登记信息表：h\_checkinitem

建表语句如图3-12所示：

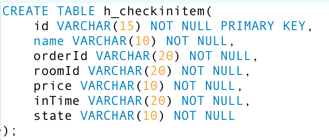


图 3-12 入住登记信息表建表语句

顾客信息表：h\_customer

建表语句如图3-13所示：

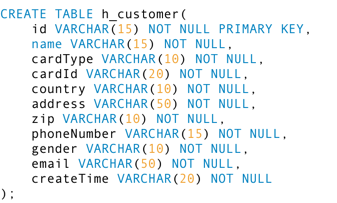


图 3-13 顾客信息表建表语句

操作员注册信息表：h\_operator

建表语句如图3-14所示：

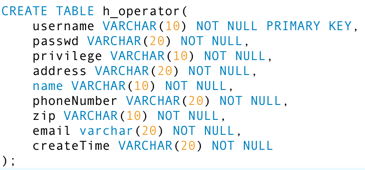


图 3-14 操作员注册信息表建表语句

订单信息表：h\_order

建表语句如图3-15所示：

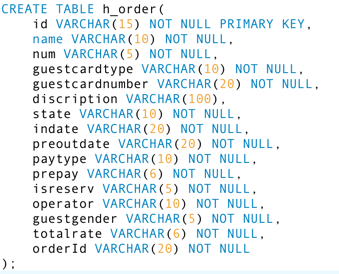


图 3-15 订单信息表建表语句

顾客预定信息表：h\_reserve

建表语句如图3-16所示：

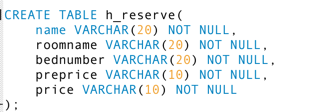


图 3-16 顾客预定信息表建表语句

客房信息表：h\_room

建表语句如图3-17所示：

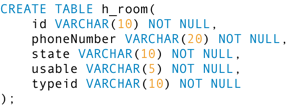


图 3-17 客房信息表建表语句

客房类型信息表：h\_roomtype

建表语句如图3-18所示：

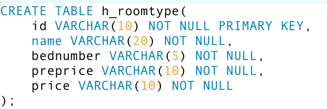


图 3-18 客房类型信息表建表语句

顾客注册信息表：h\_user

建表语句如图3-19所示：

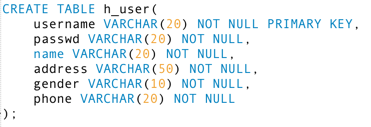


图 3-19 顾客注册信息表建表语句

## 3.5 本章小结

本章主要介绍了酒店客房管理系统的数据库表的设计。可以清楚的看到系统各个功能模块中涉及到的各种信息，更加了解本系统的工作流程。

# 4 系统功能实现

## 4.1 系统类图

类图是（Class diagram）是显示了模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其它类的关系等。类图不显示暂时性信息。

类一般包含三个组成部分。第一个是类名；第二个是属性；第三个是该类提供的方法；类的性质可以放在第四部分；如果类中含有内部类，则会出现第五个组成部分。类名部分是不能省略的，其它组成部分可以省略。

在UML静态机制中类图是一个重点，它不但是设计人员关心的核心，更是实现人员关注的核心。类的命名尽量使用应用领域中的术语，应明确，无歧义，以利于相互交流和理解。类的属性、操作中的可见性使用+、#、-分别表示public、protected、private。

### 4.1.1 操作员信息实体类

操作员信息实体类类图如图4-1所示：

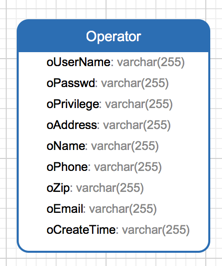


图 4-1 操作员信息实体类

### 4.1.2 用户信息实体类

用户信息实体类类图如图4-2所示：

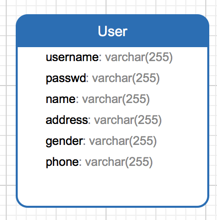


图 4-2 用户信息实体类

### 4.1.3 客户信息实体类

客户信息实体类类题如图4-3所示：

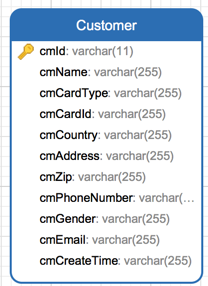


图 4-3 客房信息实体类

### 4.1.4 订单信息实体类

订单信息实体类如图4-4所示：

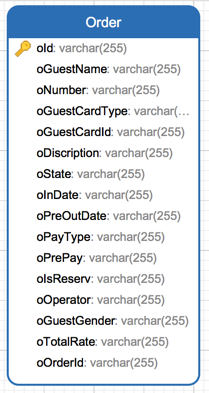


图 4-4 订单信息实体类

### 4.1.5 登记信息实体类

登记信息实体类如图4-5所示：

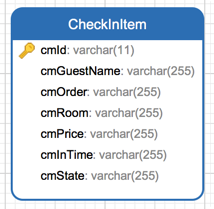


图 4-5 登记信息实体类

### 4.1.6 客房类型实体类

客房类型实体类如图4-6所示：

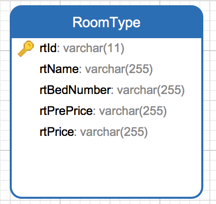


图 4-6 客房类型实体类

### 4.1.7 客房信息实体类

客房信息实体类如图4-7所示：

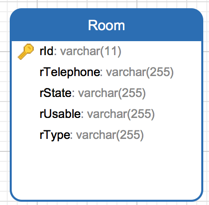


图 4-7 客房信息实体类

### 4.1.8 预定信息实体类

预定信息实体类如图4-8所示：

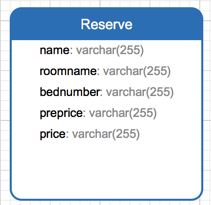


图 4-8 预定信息实体类

### 4.1.9 结账信息实体类

结账信息实体类如图4-9所示：

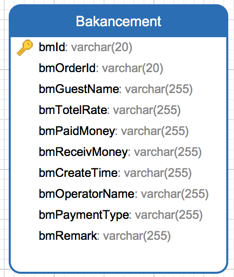


图 4-9 结账信息实体类

## 4.2 系统功能实现的主要框架

### 4.2.1 MyBatis框架

### 4.2.2 Spring框架配置文件

<!-- 数据源 -->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"*></property>

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/jinxDB"*></property>

<property name=*"username"* value=*"jinxin"*></property>

<property name=*"password"* value=*"941300"*></property>

</bean>

这段代码配置了连接数据库所需要的信息。driverClassName是连接MySql数据库使用的JDBC驱动；URL是连接本地MySql数据库的地址；username和password是连接数据库使用的用户名和密码。

<!-- mybatis SqlSessionFactory -->

<bean name=*"SqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

<property name=*"mapperLocations"* value=*"classpath:com/hotel/dao/\*Mapper.xml"*></property>

<property name=*"typeAliasesPackage"* value=*"com.hotel.bean"*></property>

</bean>

这段代码配置了SqlSessionFactory用来创建session对象。在基本的MyBatis中，session工厂可以使用SqlSessionFactoryBuilder来创建。而在MyBatis-spring中则使用SqlSessionFactoryBean来替代。

其中dataSource属性是必须需要配置的，用来指向JDBC的dataSource；一个通用的属性configLocation，它是用来指定MyBatis的XML配置文件路径的；如果基本的MyBatis配置需要改变，那么就需要使用到settings或typeAliases属性。上面配置中typeAliases配置了放置实体类的包，可以让系统自动扫描该包下的所有实体类。

<!-- dao -->

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<property name=*"basePackage"* value=*"com.hotel.dao"*></property>

</bean>

这段代码配置可以自动扫描Mapper接口并生成代理注入到spring中。其中basePackage属性配置了需要扫描的包路径。

<!-- service -->

<bean name=*"customerService"* class=*"com.hotel.service.Impl.CustomerServiceImpl"*>

<property name=*"cm"* ref=*"customerMapper"*></property>

</bean>

<bean name=*"operatorService"* class=*"com.hotel.service.Impl.OperatorServiceImpl"*>

<property name=*"om"* ref=*"operatorMapper"*></property>

</bean>

<bean name=*"orderService"* class=*"com.hotel.service.Impl.OrderServiceImpl"*>

<property name=*"oom"* ref=*"orderMapper"*></property>

</bean>

<bean name=*"checkinService"* class=*"com.hotel.service.Impl.CheckInItemServiceImpl"*>

<property name=*"cs"* ref=*"checkInItemMapper"*></property>

</bean>

<bean name=*"balancementService"* class=*"com.hotel.service.Impl.BalancementServiceImpl"*>

<property name=*"bm"* ref=*"balancementMapper"*></property>

</bean>

<bean name=*"roomService"* class=*"com.hotel.service.Impl.RoomServiceImpl"*>

<property name=*"rm"* ref=*"roomMapper"*></property>

</bean>

<bean name=*"roomTypeService"* class=*"com.hotel.service.Impl.RoomTypeServiceImpl"*>

<property name=*"rtm"* ref=*"roomTypeMapper"*></property>

</bean>

<bean name=*"userService"* class=*"com.hotel.service.Impl.UserServiceImpl"*>

<property name=*"um"* ref=*"userMapper"*></property>

</bean>

<bean name=*"reserveService"* class=*"com.hotel.service.Impl.ReserveServiceImpl"*>

<property name=*"rm"* ref=*"reserveMapper"*></property>

</bean>

这段代码配置了系统使用到的所有的业务层代码。

<!-- 事务管理 -->

<bean name=*"tranManager"*

class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

<tx:advice id=*"tranInterceptor"* transaction-manager=*"tranManager"*>

<tx:attributes>

<tx:method name=*"\*"* propagation=*"REQUIRED"* isolation=*"DEFAULT"* rollback-for=*"Exception"*/>

</tx:attributes>

</tx:advice>

这段代码配置了数据库的事务管理。针对所有的方法，事务传播行为为REQUIRED，事务隔离级别为DEFAULT。

## 4.3 系统功能实现的主要模块

### 4.3.1 系统登录页面

酒店客房管理系统共有两种登录权限。一是系统操作员登陆，二是用户登录。

系统操作员登录界面如图4-10所示：



图 4-10 管理员登录界面

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

String name = request.getParameter("username");

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

**try** {

Operator operator = os.operatorLogin(name, password);

HttpSession session = request.getSession();

session.setAttribute("operator", operator);

response.sendRedirect("roomViewServlet");

} **catch** (Exception e) {

e.printStackTrace();

request.setAttribute("message", e.getMessage());

request.getRequestDispatcher("operatorlogin.jsp").forward(request, response);

}

}

当用户在登录界面输入用户名密码并点击登陆后，后台Servlet会通过request对象调用getParameter方法将用户输入的用户名密码获取到后台，并通过获取到的用户名密码调用service中的operatorLogin方法，在operatorLogin方法中会判断当前用户是否存在或密码是否正确。如果正确，会将该用户保存到session域中并且跳转到相应的操作页面，否则在界面弹出用户登录失败的弹窗。

### 4.3.2 信息管理模块

#### 4.3.2.1 顾客信息登记功能

顾客信息登记页面如图4-11所示：



图 4-11 顾客信息登记页面

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

String cmId = CreateID.*getCustomerId*();

String cmName = request.getParameter("cmname");

String cmCardType = request.getParameter("cmcardtype");

String cmCardId = request.getParameter("cmcardid");

String cmCountry = request.getParameter("cmcountry");

String cmAddress = request.getParameter("province") + request.getParameter("city")

+ request.getParameter("cmaddress");

String cmZip = request.getParameter("cmzip");

String cmPhoneNumber = request.getParameter("cmphonenumber");

String cmGender = request.getParameter("cmgender");

String cmEmail = request.getParameter("cmemail");

String cmCreateTime = request.getParameter("cmcreatetime");

Customer customer = **new** Customer(cmId, cmName, cmCardType, cmCardId, cmCountry, cmAddress, cmZip, cmPhoneNumber,

cmGender, cmEmail, cmCreateTime);

cs.saveCustomer(customer);

response.sendRedirect("customerViewServlet");

}

顾客信息登记用于将顾客的个人基本信息进行保存。实现顾客信息登记功能，首先从前台页面输入框中输入信息，每个输入框都有对应的name值，在后台用request对象的getParameter()方法传入前台输入框对应的name值获取到前台页面输入的值，封装成Customer对象。通过CustomerService对象调用saveCustomer()方法将对象中的值持久化到数据库中。

#### 4.3.2.2 查看顾客信息功能

查看顾客信息功能如图4-12所示：



图 4-12 查看顾客信息功能

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

HttpSession session = request.getSession();

List<Customer> allCustomer = cs.getAllCustomer();

session.setAttribute("allCustomer", allCustomer);

response.sendRedirect("customerview.jsp");

}

查看顾客信息功能可以查看当前所有入住过本酒店的顾客的基本信息。实现查看顾客信息功能，通过在前台页面点击查看顾客信息选项调用后台servlet。在servlet 中，通过getAllCustomer()方法在数据库中查询到所有的顾客信息，并获取页面的Session域对象，调用setAttribute()方法将查询到的顾客信息存到session中。在前台jsp页面中使用EL表达式动态显示到页面中。同时在该页面增加了对顾客信息的删除功能。

#### 4.3.2.3 入住信息登记功能

入住信息登记功能如图4-13所示：



图 4-13 入住信息登记功能

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

String cmId = CreateID.*getCheckInItemId*();

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

String orderId = os.getOrderByName(cmGuestName).

get(os.getOrderByName(cmGuestName).size() - 1).getoId();

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

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

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

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

CheckInItem c = **new** CheckInItem(cmId, cmGuestName, orderId, cmRoom, cmPrice, cmInTime, cmState);

rs.setRoomState(cmRoom, cmState, "否");

cs.saveCheckinItem(c);

response.sendRedirect("checkInItemViewServlet");

}

入住信息登记功能用于登记当前顾客的基本入住信息，比如入住房间、入住时间等。实现入住信息登记功能，首先通过request对象的getParameter()方法获取到前台页面输入的值，封装成CheckInItem对象，获取CheckInItemService实例调用saveCheckInItem()方法将对象中的值持久化到数据库中并同时将客房状态修改为当前状态。

#### 4.3.2.4 查看信息登记功能

查看信息登记功能如图4-14所示：



图 4-14 查看信息登记功能

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

HttpSession session = request.getSession();

List<CheckInItem> allCheckInItem = cs.getAllCheckInItem();

session.setAttribute("allCheckInItem", allCheckInItem);

response.sendRedirect("checkinitemview.jsp");

}

查看信息登记功能用于查看当前所有的入住情况。实现查看顾客入住登记功能，通过在前台页面点击查看顾客信息选项调用后台servlet。在servlet 中，通过getAllCheckInItem()方法在数据库中查询到所有的顾客信息，并获取页面的Session域对象，调用setAttribute()方法将查询到的登记信息存到session中。在前台jsp页面中使用EL表达式获取session中的登记信息，动态显示到页面中。同时在该页面增加了对登记信息的删除功能。

### 4.3.3 订单管理模块

#### 4.3.3.1 查看订单功能

查看订单功能如图4-15所示：



图 4-15 查看订单功能

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

HttpSession session = request.getSession();

List<Order> allOrder = os.getAllOrder();

session.setAttribute("allOrder", allOrder);

response.sendRedirect("orderview.jsp");

}

查看订单功能用于查看所有的订单信息，同时可以对订单信息进行删除。实现查看订单功能，通过获取OrderService的实例来调用getAllOrder()方法，从数据库中查找到信息并封装为List对象，然后获取该页面的Session域对象，调用setAttribute()方法将获取到的List对象存到session中。在前台通过EL表达式获取Session域中保存的订单信息并显示在网页上。

#### 4.3.3.2 订单填写功能

订单填写功能如图4-16所示：



图 4-16 订单填写

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

String orderId = CreateID.*getOrderId*();

String oGuestName = request.getParameter("oguestname");

String oNumber = request.getParameter("onumber");

String oGuestCardType = request.getParameter("oguestcardtype");

String oCheckTypt = request.getParameter("ochecktype");

String oInDate = request.getParameter("oindate");

String oPreOutDate = request.getParameter("opreoutdate");

String oPayType = request.getParameter("opaytype");

String oCardId = request.getParameter("ocardid");

String oPrePay = request.getParameter("oprepay");

String oIsReserv = request.getParameter("oisreserv");

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

String oTotalRate = request.getParameter("ototalrate");

String oOrderId = request.getParameter("oorderid");

String oDiscription = request.getParameter("odiscription");

String oGuestGender = request.getParameter("oguestgender");

Order order = **new** Order(orderId, oGuestName, oNumber, oGuestCardType, oCardId, oDiscription, oCheckTypt,

oInDate, oPreOutDate, oPayType, oPrePay, oIsReserv, operator, oGuestGender, oTotalRate, oOrderId);

oos.saveOrder(order);

response.sendRedirect("orderViewServlet");

}

订单填写功能用于登记当前顾客的订单信息，比如顾客姓名、入住人数、证件号码、入住时间、离开时间等。实现订单填写功能，首先通过request对象的getParameter()方法获取到前台页面输入的值，封装成Order对象，获取OrderService实例调用saveOrder()方法将对象中的值持久化到数据库中。

### 4.3.4 客房信息管理

#### 4.3.4.1 客房添加

客房添加功能如图4-17所示：

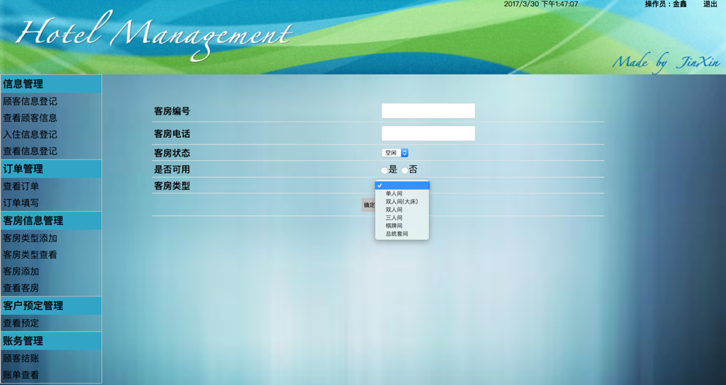


图 4-17 客房添加

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

String rId = request.getParameter("rid");

String rTelephone = request.getParameter("rtelephone");

String rState = request.getParameter("rstate");

String rUsable = request.getParameter("rusable");

String rType = request.getParameter("rtype");

Room room = **new** Room(rId, rTelephone, rState, rUsable, rType);

rs.saveRoom(room);

response.sendRedirect("roomViewServlet");

}

客房添加功能用于添加新的客房。实现客房添加功能，首先通过request对象的getParameter()方法获取到前台页面输入的值，封装成Room对象，获取RoomService实例调用saveOrder()方法将对象中的值持久化到数据库中。客房类型的选项是通过使用AJAX异步查询从数据库中读取客房类型并显示到前台页面中。

#### 4.3.4.2 客房查看

客房查看功能如图4-18所示：



图 4-18 客房查看功能

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

HttpSession session = request.getSession();

List<Room> allRoom = rs.getAllRoom();

session.setAttribute("allRoom", allRoom);

response.sendRedirect("roomstate.jsp");

}

查看订单功能用于查看所有的订单信息，同时可以对订单信息进行删除。实现查看订单功能，通过获取RoomService的实例来调用getAllRoom()方法，从数据库中查找到信息并封装为List对象，然后获取该页面的Session域对象，调用setAttribute()方法将获取到的List对象存到session中。在前台通过EL表达式获取Session域中保存的客房信息并显示在网页上。

### 4.3.5 账务管理

#### 4.3.5.1 顾客结账

顾客结账功能如图4-19所示：



图 4-19 顾客结账

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

String roomId = cs.getAllCheckInItem().get(cs.getAllCheckInItem().size() - 1).getCmRoom();

String bmId = CreateID.*getBalancementId*();

String bmGuestName = request.getParameter("bmguestname");

Order order = os.getOrderByName(bmGuestName).get(os.getOrderByName(bmGuestName).size() - 1);

String bmTotelRate = request.getParameter("bmtotelrate");

String bmOrderId = order.getoId();

String bmPaidMoney = order.getoPrePay();

String bmReceivMoney = request.getParameter("bmreceivmoney");

String bmCreateTime = request.getParameter("bmcreatetime");

String bmOperatorId = request.getParameter("bmoperator");

String bmPaymentType = request.getParameter("bmpaymenttype");

String bmRemark = request.getParameter("bmremark");

Balancement b = **new** Balancement(bmId, bmOrderId, bmGuestName, bmTotelRate, bmPaidMoney, bmReceivMoney,

bmCreateTime, bmOperatorId, bmPaymentType, bmRemark);

bs.saveBalancement(b);

rs.setRoomState(roomId, "空闲", "是");

response.sendRedirect("balancementViewServlet");

}

顾客结账功能用于记录顾客的结帐信息，比如顾客姓名、应收金额、操作员等。实现顾客结账功能，首先通过request对象的getParameter()方法获取到前台页面输入的值，封装成Balancement对象，获取BalancementService实例调用saveBalancement()方法将对象中的值持久化到数据库中并同时将客房状态修改为空闲状态。

#### 4.3.5.2 账单查看

账单查看功能如图4-20所示：



图 4-20 账单查看

代码如下：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

HttpSession session = request.getSession();

List<Balancement> allBalancement = bs.getAllBalancement();

session.setAttribute("allBalancement", allBalancement);

response.sendRedirect("billview.jsp");

}

账单查看功能用于查看所有的账单信息，同时可以对账单信息进行删除。实现查看账单功能，通过获取BalancementService的实例来调用getAllBalacnement()方法，从数据库中查找到信息并封装为List对象，然后获取该页面的Session域对象，调用setAttribute()方法将获取到的List对象存到session中。在前台通过EL表达式获取Session域中保存的账单信息并显示在网页上。

## 4.4 本章小结

本章主要介绍了酒店客房管理系统的功能实现代码，完成了需求中所需要实现的功能，同时在数据库中创建了数据表，对数据库进行连接。

# 5 软件测试

软件测试（Software Testing），描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。换句话说，软件测试是一种实际输出和预期输出间的审核或者比较过程。软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。软件测试分为单元测试、集成测试、系统测试、验收测试。

主要的测试方法有黑盒测试、白盒测试和灰盒测试。

黑盒测试：指的是把被测的软件看作是一个黑盒子，我们不去关心盒子里面的结构是什么样子的，只关心软件的输入数据和输出结果。它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当的接收输入数据而产生正常的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。

白盒测试：指的是把盒子盖子打开，去研究里面的源代码和程序结构。它是按照程序的内部结构测试程序，通过测试来检测程序内部动作是否按照设计规格说明书的规定正常运行，检测程序中的每条通路是否都能按预定要求正确工作。

灰盒测试：介于黑盒测试和白盒测试之间。可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现，但这种关注不像白盒那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态，有时候输出是正确，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。

## 5.1 软件测试的目的

软件测试是为了证明获取系统在可接受风险范围内可用的信心，尝试在非正常情况和条件下的功能和特性，保证一个工作产品是完整的且可用或可被集成的。同时发现代码缺陷、错误和系统的不足并且定义系统能力和局限性，提供组件、工作产品和系统的质量信息。

## 5.2 软件测试的任务

1.发现代码中存在BUG。

2.确保系统可以满足用户的需求。

3.评估软件的质量。

## 5.3 测试环境的搭建

### 5.3.1 搭建安装项目

要部署一个项目，首先要将Tomcat服务器集成到Eclipse中：

如图5-1现将Tomcat在计算机本地的路径添加到Eclipse中，

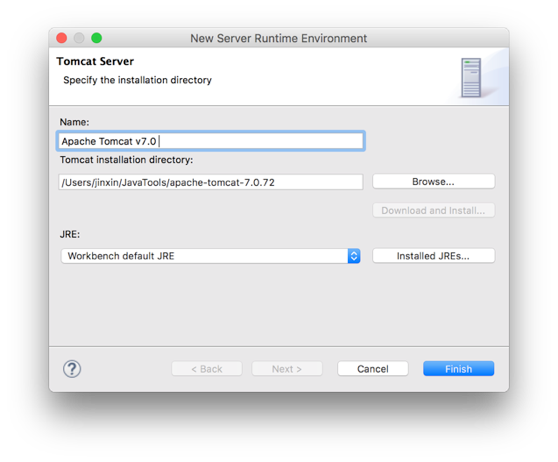


图 5-1 添加Tomcat到Eclipse

然后如图5-2选择添加服务器选项，选择Tomcat v7.0 Server完成添加服务器。

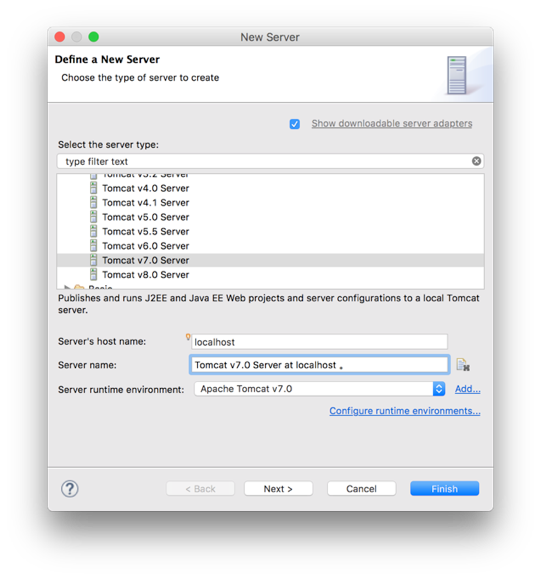


图 5-2 添加服务器

### 5.3.2 添加项目并运行

如图5-3所示向Tomcat服务器中添加项目

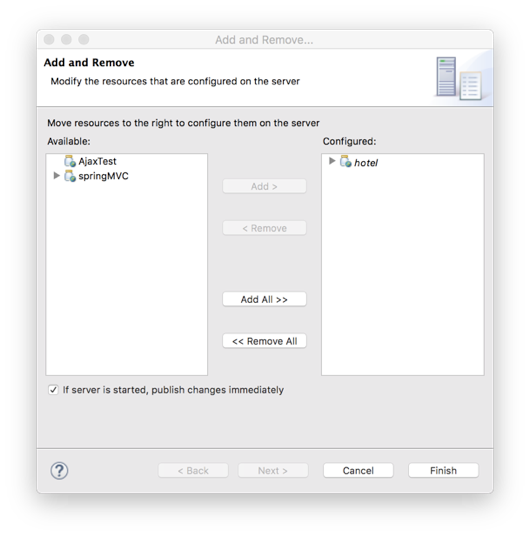


图 5-3 添加项目输出

向Tomcat服务器中添加好项目之后，点击运行按钮，会看到控制台输出日志信息，如果没有保存的话，说明项目运行成功。

## 5.4 本章小结

本章主要介绍了对酒店客房管理系统的测试过程。采用了黑盒测试和白盒测试的方法。首先使用白盒测试的方法，主要测试系统是否能正常登录，是否可以正常的产看各类信息；然后使用黑盒测试的方法，看各个功能是否都已实现。

# 结论

随着答辩日的临近，酒店客房管理系统的开发也已经结束，系统已经基本成型，需求所要求的功能也已基本实现，本系统实现了用户模块和系统管理员模块的功能，用户可以在线查看客房状态和预订客房，系统管理员可以管理用户信息和客房信息。

系统主要采用了瀑布型的开发模式，将功能实现与设计分开，采用需求分析、软件设计、程序编写、软件测试的步骤进行开发。这种开发模式的优势是可以在编写实现代码之前就确定系统需要实现哪些功能，各模块之间的关系是什么。所以，可以加快系统功能实现代码的编写速度，快速有效的实现系统功能。

系统采用了MVC的设计模式，可以在不影响其它功能代码的情况下增加其它功能，同时也有效的实现了代码复用，防止编写重复的代码。

当然，在系统中还有很多不足之处。比如在登记完客户信息之后创建订单时，需要再次填写部分的客户信息，会做一定的重复操作，在日后的改进中会做成自动从数据库中读取所需的用户信息，避免一些重复性的操作。

# 参考文献

1. 张敏. 酒店客房信息管理系统[D]. 山东大学, 2010
2. Bryan Basham, Kathy Sierra, Bert Bates. Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam. America: O’Reilly Media, Inc.2008.3.
3. Baron 王小东. 高性能MYSQL. 北京:电子工业出版社. 2010.2.
4. 作：埃克尔 译：陈昊鹏. Java编程思想. 机械工业出版社. 2007.6.
5. 文斌. 软件工程与软件文档写作[M]. 清华大学出版社. 2005
6. 作：(英)福勒 译：王怀民、周斌. 企业应用架构模式. 机械工业出版社.2010.4.