第一章管理信息系统及其开发的认识

管理信息系统（Management Information System，简称MIS）是一种将技术、人员和流程整合起来，以支持组织内部决策制定、问题解决和业务运营的系统。信息系统究竟是什么？从技术的角度看，是从组织内外部收集、储存和分发信息以支持组织的各项职能与决策、沟通、协调、控制、分析以及可视化。以及通过三个基本活动——输入、处理和输出，将原始数据转化为有用的信息。从业务角度看，信息系统是应对公司面临的问题或挑战的解决方案，包含管理、技术和组织三个维度要素。信息系统的管理要素涉及领导力、战略和管理行为等方面的议题；信息系统的技术要素包括计算机硬件、软件、数据管理技术和网络/通信技术（包括互联网）[1]。

1.1 对管理信息系统的认识

MIS旨在收集、处理、存储和分发组织内外的信息，以帮助管理者更好地理解和管理组织的运作。

信息整合： MIS的主要目标之一是整合各种信息资源，包括内部和外部数据，以提供全面的视图，帮助管理者做出更明智的决策。

支持决策制定： MIS为管理层提供及时、准确的信息，以支持决策制定过程。这些决策可能涉及战略规划、资源分配、业务流程优化等方面。

提高效率： 通过自动化和集成业务流程，MIS有助于提高组织的运营效率，减少冗余工作和错误。

改进沟通： MIS促进了组织内外的信息共享和沟通，有助于不同部门之间更好地协作。

风险管理： MIS能够监测和分析业务数据，帮助管理者识别潜在的风险，并采取相应的措施来降低风险。

1.2 对管理信息系统开发的理解

民宿客栈管理系统是典型的MIS，其开发主要包括后台数据库的建立和维护以及前台应用程序的开发两个方面。该系统可以实现对客栈的科学化、规范化、信息化的管理。系统功能主要包括住客管理、客房管理、会员管理等。该系统是根据假日客栈日常管理的实际情况进行设计的，主要目的是为了方便客栈对客房的实际情况进行集中的查询与管理，以提高整个客栈的工作效率。对于管理信息系统的开发，通常涉及以下几个方面：

1.2.1 需求分析

在系统开发的初期，明确定义用户的需求。包括了解组织的业务流程、数据需求、用户期望等。

1.2.2 系统设计

在需求分析的基础上，进行系统设计，确定系统的结构、功能和界面。这一阶段也包括数据库设计、网络设计等方面。

1.2.3 编码和实施

在系统设计完成后，进行编码和实施，即根据设计开发系统的各个模块，并将系统部署到实际运行环境中。

1.2.4 测试

对系统进行全面的测试，包括单元测试、集成测试和系统测试，以确保系统的稳定性和符合用户需求。

1.2.5 部署和维护

MIS能发挥其强大的优势，在数据保存、数据交换等方面均快速可靠，是手工操作所不能完成的。一个管理信息系统的开发必须有一个正确的设计指导思想，通过合理的选择数据结构，编程语言以及开发环境，构成一个完善的网络系统，才能充分发挥管理信息系统优势。

1.3 管理信息系统开发的方法及流程概述

管理信息系统开发方法是指用来指导管理信息系统开发的一系列理论、技术和工具。管理信息系统开发方法有很多种，常见的方法有：

瀑布模型：将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落[2]。 瀑布模型是传统的管理信息系统开发方法，它将开发过程分为一系列的阶段，每个阶段完成后才能进行下一个阶段。优点是简单易行，缺点是灵活性差，不适应需求变化大的项目。

螺旋模型：1988年，巴利·玻姆Barry Boehm正式发表了软件系统开发的“螺旋模型”，它将瀑布模型和快速原型模型结合起来，强调了其他模型所忽视的风险分析，特别适合于大型复杂的系统[3]。 螺旋模型是瀑布模型的改进，它将开发过程分为一系列的迭代，每个迭代都包括需求分析、设计、开发、测试和部署等阶段。优点是灵活性好，适应需求变化大的项目，缺点是开发周期长，成本高。

敏捷开发：是一种面临迅速变化的需求快速开发软件的能力。为了获取这种敏捷性，我们需要使用一些可以提供必要的纪律和反锁的实践[4]。将开发过程分为一系列的短周期，每个周期都包括需求分析、设计、开发、测试和部署等阶段。敏捷开发的优点是开发周期短，成本低，适应需求变化大的项目，缺点是需要开发团队有较强的沟通和协作能力。

管理信息系统开发流程是指管理信息系统开发过程中所需要遵循的一系列步骤。管理信息系统开发流程通常包括：

需求分析：需求分析是开发过程中最重要的一步，它是确定系统功能和性能的基础。需求分析需要收集用户的需求，分析需求的合理性和可行性。

系统设计：系统设计是将需求转化为系统的逻辑和物理结构的过程。系统设计需要确定系统的结构、功能、数据和流程。

系统开发：系统开发是根据系统设计进行系统的编码、测试和部署的过程。系统开发需要开发人员具有一定的技术能力。

系统测试：系统测试是验证系统是否满足需求的过程。系统测试需要测试人员具有一定的测试技能。

系统部署：系统部署是将系统投入使用的过程。系统部署需要考虑系统的安全、可靠性和性能。

管理信息系统的开发方法和流程因项目规模、性质和技术选型而异。

第二章需求分析

需求分析是系统开发的重中之重。对系统后期开发和测试起到了关键性的作用。从软件工程经验教训中得知，需求分析的成功与失败决定了后期系统开发所要花费的成本。好的需求是成功的一半，能够大幅降低系统开发成本。反之，一个失败的不能正确反映客户的需求，将会给后期系统开发和维护带来较大的困难。本系统在进行需求分析时，严格按照客户的实际需求来进行。主要是为了更好的了解用户需求，它是系统设计的起点和系统测试的依据，是否准确的描述了用户的需求直接关系到系统的实现和产品的交互。

2.1 该管理信息系统业务流程分析

该管理信息系统的业务流程分析涉及多个方面，包括住客信息管理、客房信息管理、会员信息管理、数据报表生成、系统设置等。用户希望通过使用该管理系统记录用户、客房等信息，提高管理水平的目的，例如：快速处理日常的业务及相关数据，实时查询各种入住信息；实时录入住宿单、换房单和退房单等情况；实时查询客户消费信息等的具体情况。同时，可以建立客户或会员资料，方便日后建立良好的客户互动关系。也可导出数据等报表。

2.1.1 住客信息管理流程

添加住客： 客户提供必要的个人信息，系统录入住客信息，包括姓名、身份证号、入住日期、离店日期、房

型、房号等。
库存管理： 更新系统中客房的可用数量，避免超售情况的发生。
2.1.2 客房管理流程
包括房间列表信息的显示和房间管理信息的增删改查。管理客栈在在线平台上的信息，包括房型、价格、促销活动等。
客房分配房态管理： 更新系统中客房的入住状态（空房、已入住、未打扫），确保实时反映客房的可用情况。
房务服务： 管理客房清洁、维护等服务，确保客房设施的正常运作。
2.1.3 会员信息管理流程
管理会员信息，实施会员制度，开展促销活动，提高客户忠诚度。会员信息管理流程涉及到对会员数据的收集、存储、更新、查询和删除等操作。
会员注册/加入：通过系统注册用户成为会员，流程：访问注册页面，填写必要的个人信息，如姓名、联系方式、邮箱等。 会员信息更新：随时更新其个人信息，进入会员管理页面，修改需要更新的信息，如地址、联系方式等。
会员等级管理：根据消费记录或其他标准被分配到不同的会员等级（初级、中级、高级）。会员等级信息存储在会员数据库中。。
2.1.4 数据报表生成流程
支持导出房间信息表和会员信息表，点击导出按钮即可完成数据导出到Excel的动作。
2.1.5 系统设置
网站信息维护与管理，系统名称、版权所有、备案号等，以及密码修改，和缓存清理。民宿客栈管理信息系统通过集成这些流程，提高了工作效率，降低了错误率，为客栈管理提供了更好的决策支持。
2.2 系统的可行性分析
民宿客栈管理信息系统的可行性分析是在系统规划阶段进行的，旨在评估系统开发和实施的可行性，包括经济可行性、技术可行性和操作可行性。开发民宿客栈管理系统可能会受到现实生活中的各种各样的限制，比如：开发的时间，技术水平和资源有效分配等等问题。由此可见，在开发系统之前，可行性研究的实行显得十分重要和有必要。可以减少不必要的人力的消耗和财力的损失，减少系统开发的风险投资。这里从以下几个方面进行研究和分析，本系统是否真正的具有可行性、可研究性。以下是这些方面的分析：
2.2.1 技术可行性
主要用IntelliJ IDEA进行本系统的开发与发布，web服务器运用Apache的Tomcat开源免费的服务器，数据库采用免费开源的Mysql数据库。主要开发语言是面向对象的开发语言Java，尽管以C++为基础，但Java是一种更纯粹的面向对象程序设计语言[5]。这个系统主要完成一些简单业务，如住客信息管理、客房信息管理、会员信息管理、数据报表生成、系统设置模块。前端界面采用jsp+css等技术进行页面设计，后台采用java来操作数据库进行增删改查和用Java实现图片上传等操作。以上开源套件的运用完全可以支撑起本系统的开发。
2.2.2 经济可行性
根据技术可行性可以看出，民宿客栈系统主要采用免费开源的软件进行开发，通过这样做，开发成本得到了有效的管理，节约了商用开发工具的费用。采用免费开源的软件还可以根据需求对其进行改造适应自己的需求和业务，从经济角度而言对民宿客栈系统的开发也是可行的。
2.2.3 操作可行性
民宿客栈系统操作简单方便，大多都是通过鼠标和键盘进行操作和管理。而且现在电脑技术的迅速普及，基本每个人都具备基本的电脑操作技巧，因此在用户具备一定的操作技巧的基础之上进行简单的培训就可以使用民宿客栈系统对业务进行全面细致的管理。
根据以上的种种条件，可以判断出民宿客栈管理系统在当今对民宿客栈和发展具有很大的帮助，民宿客栈系统的开发和应用是大势所趋，是非常适合当今时代的发展。通过对这些方面的可行性分析，可以为决策者提供全面的信息，帮助他们决定是否继续推进民宿客栈信息系统的开发和实施。可行性分析有助于确保系统的投资是明智和可行的，并能够为企业带来实际的经济和业务效益。
2.3 系统需求及所要求功能的分析
系统规划解决“是什么”的问题，系统分析主要解决“做什么”的问题。系统分析的主要任务是在系统规划的基础上，通过问题识别、可行性分析等完成系统的逻辑方案设计，其结果是绘制业务流程图和数据流程图。

系统需求和功能分析是在管理信息系统开发过程中的关键步骤之一。通过对用户需求的深入理解和分析，可以明确系统应该具备的功能和特性。

2.3.1 住客信息管理模块

住客管理模块包括住客入住的添加、删除、修改、查询和住户信息的显示。

2.3.2 客房信息管理模块

客房管理模块包括房间列表信息的显示和房间的添加、删除、修改、查询

客房信息管理，主要有以下几部分组成：

客房类型的增加和删除：豪华大床房、高级大床房、普通大床房、豪华双床房、大床钟点房（四小时）等等。

客房信息的补充和修改：主要指房间号，房间面积，价格以及房间的真实图片等必要的信息。

客房预订的增加和删除：主要是指客户对客栈的预订情况，包括客房的房间号，预订天数，预定时间和客户信息等。

2.3.3 会员信息管理模块

会员管理模块包括会员的增加、删除、修改、查询和会员信息的显示。客户信息的增加修改和删除和查看：主要用于完成对客户信息的修改和删除，当然也可以进行客户信息的增加和查看。

2.3.4 数据报表生成流程

数据表的生成功能模块包括房间信息表的导出和会员信息表的导出。

2.3.5 系统设置

系统设置模块包括网站信息的显示、密码的修改和系统缓存的清除。

第三章系统概要设计

系统概要设计是在系统详细设计之前的一个阶段，主要目的是定义系统的整体结构、模块划分、数据流和处理流程等，为后续的详细设计提供基础。系统概要设计的完成为系统的详细设计和实现提供了指导，同时为团队成员提供了共同的理解框架。设计过程中应该充分考虑系统的可维护性、可扩展性和安全性等方面。

3.1 体系结构设计

体系结构设计是管理信息系统开发过程中的一个关键步骤，涉及到系统整体结构的设计和组织，以确保系统的各个组成部分能够有效地协同工作[6]。

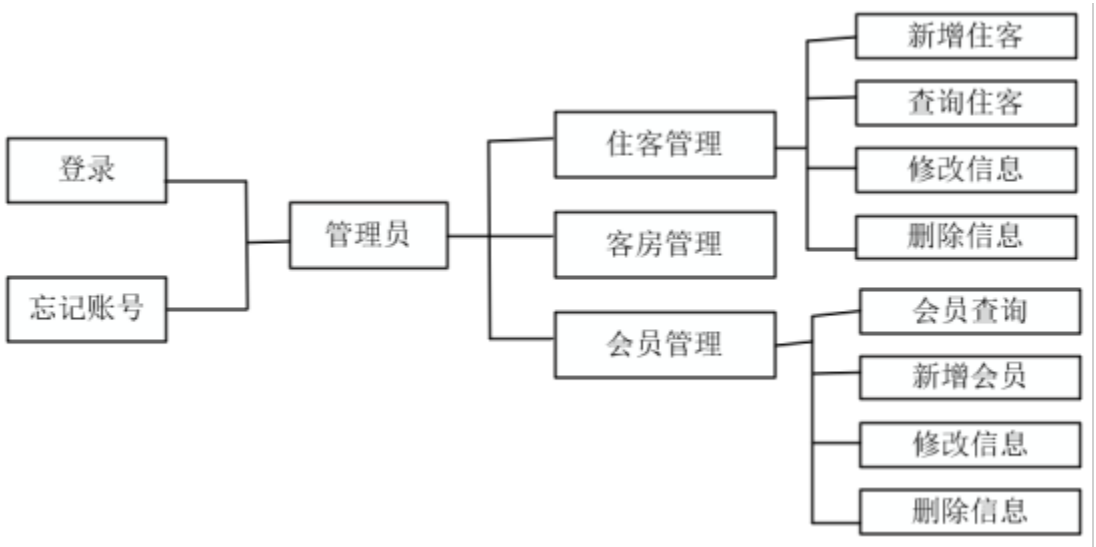


图3-1 系统体系结构

在体系结构设计的过程中，需要充分考虑系统的可用性、可靠性、性能、安全性等方面的需求，以满足用户和业务的实际需求[7]。设计阶段的仔细规划和分析有助于在后续的开发和实施阶段取得成功。

3.2 功能模块设计

功能模块设计（Hierarchy Input-Process-Output，层次输入-处理-输出）是一种结构化的设计方法，通过层次结构清晰地表示系统的输入、处理和输出[8]。

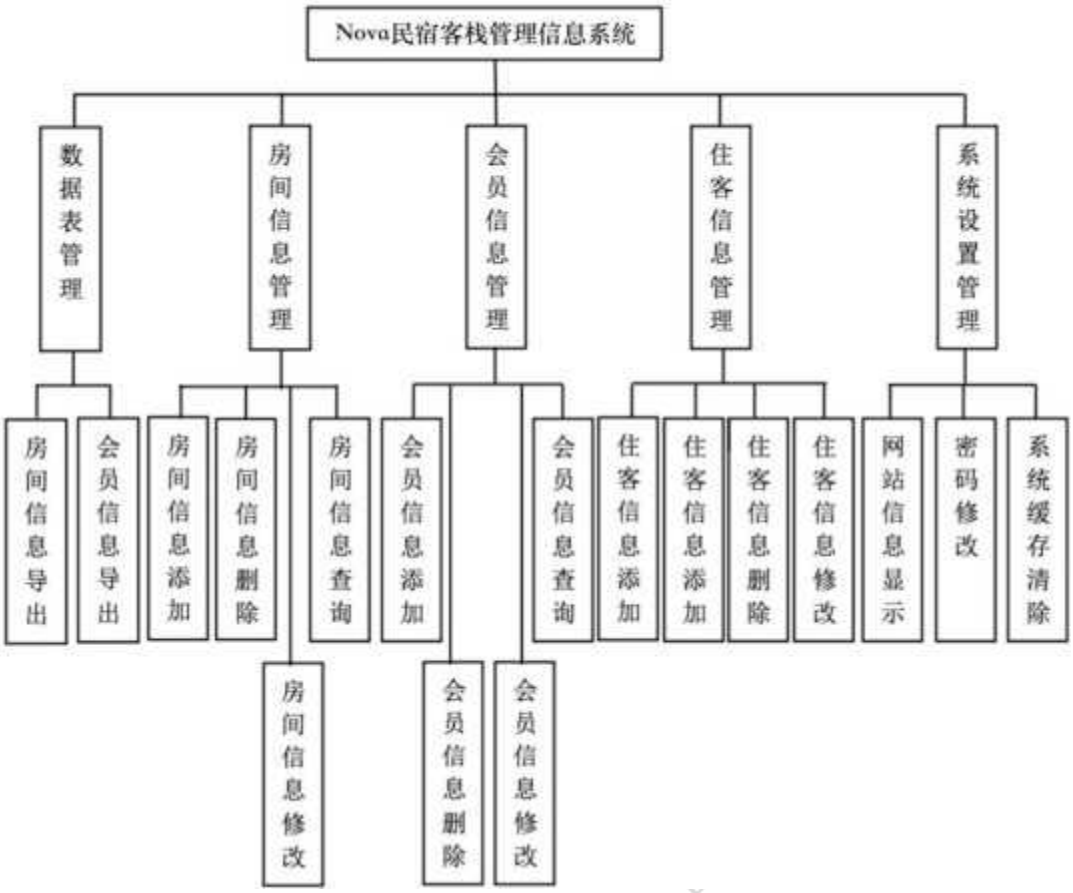


图3-2 Nova系统功能模块

3.3 数据库设计

数据库设计是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求[9]。数据库设计是管理信息系统开发过程中至关重要的一部分。包括概念设计、逻辑设计和物理设计三个阶段。

3.3.1 数据库的概念设计

管理员管理住客功能，主要实现管理员对住客的管理，可以对住客进行增删改查，其中住客信息主要包括姓名、性别、身份证号、手机号、入住时间、退房时间、房间类型、房间号。

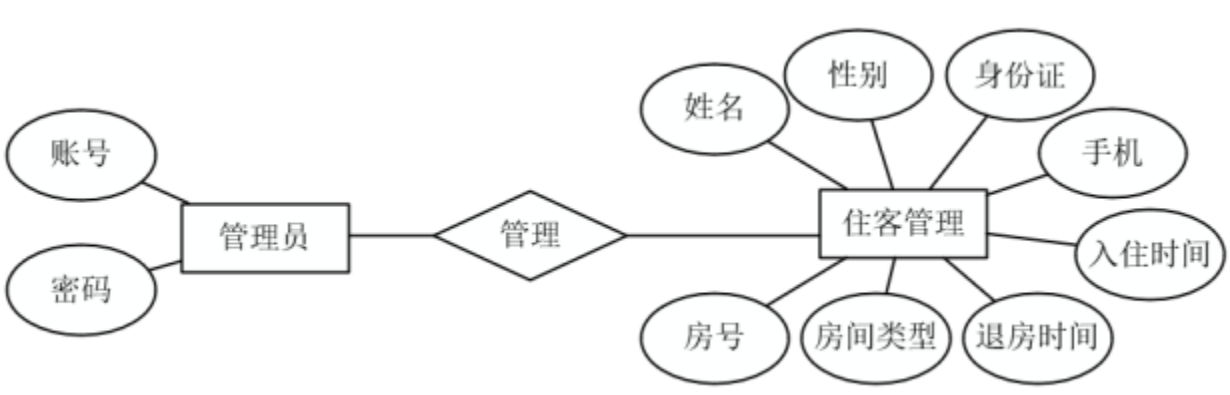


图3-3 会员和客房管理

管理员管理会员功能，主要实现管理员对会员的管理，可以对会员进行增删改查，其中会员信息主要包括姓名、性别、 身份证号、手机号、会员类型、开通时间、到期时间。

管理员管理房间功能，主要实现管理员对房间的管理，可以对房间进行增删改查，其中房间信息主要包括房间号、房间类型、价格、状态、房间描述、图片。

图3-4 会员和客房管理

3.3.2 数据库的逻辑设计

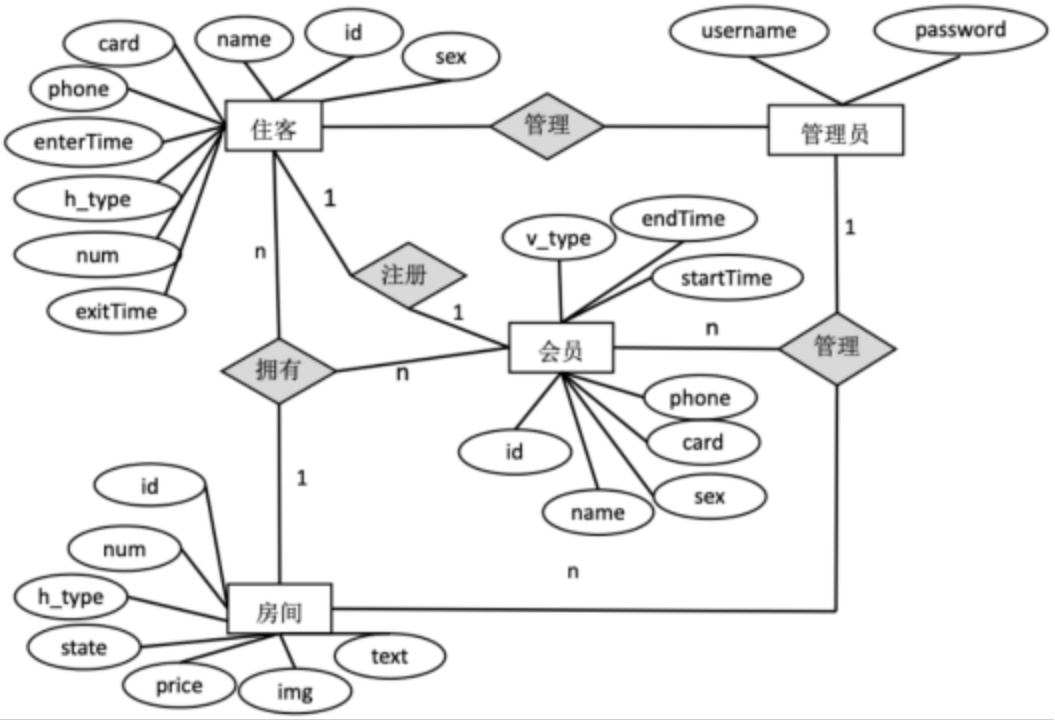
客房管理模块包括房间列表信息的显示和房间的添加、删除、修改、查询

客房信息管理，主要有以下几部分组成：

客房类型的增加和删除：豪华大床房、高级大床房、普通大床房、豪华双床房、大床钟点房（四小时）等等。

客房信息的补充和修改：主要指房间号，房间面积，价格以及房间的真实图片等必要的信息。

客房预订的增加和删除：主要是指客户对客栈的预订情况，包括客房的房间号，预订天数，预定时间和客户信息等。



系统E-R图

3.3.3 数据库的物理设计

根据数据库设计规范化原则以及系统E-R图，设计并定义Nova民宿客栈管理系统所需要的数据库基本数据表如下所示。

表3-1 管理员信息表

列名	数据类型	长度	是否可空	注释
username	varchar	32	否	用户名

图4-13 数据导出界面

第五章系统实现

系统实现是按照预先的设计具体地实现信息系统的过程，数据库的建立和配置是关键的环节[11]。

5.1 数据库的建立和配置

5.1.1 连接 MySQL

输入mysql-uroot-p命令，回车，然后输入MySQL的密码，再回车，连接上MySQL。查看当前的数据库，使用showdatabases;查看当前安装的MySQL中有哪些数据库。

```
show databases;
```

5.1.2 创建数据库

使用 create database 数据库名; 创建数据库。

```
create database novahostel;
```

创建数据库时设置字符编码，使用createdatabase数据库名charactersetutf8;创建数据库并设置数据库的字符编码。

```
create database novahostel character set utf8;
```

MySQL 默认的编码方式 latin1 （单字节编码），通常我们会在数据库中存放中文数据，所以最好把数据库的编码方式设置成 utf-8 ，这样中文才能正常显示。

```
create database novahostel charset utf8;
```

查看和显示数据库的编码方式，使用showcreatedatabase数据库名;显示数据库的创建信息。

```
show create database novahostel;
```

如果没有设置编码格式，可以使用 alter database 数据库名 character set utf8； 修改数据库编码

```
alter database novahostel character set utf8;
```

novahotel 经过修改后，编码方式也变成了 utf-8 。

5.1.2 创建数据表

进入或切换数据库，使用 use 数据库名 进入或切换数据库。

```
use novahostel;
```

创建数据表，查看当前数据库中的表，使用 show tables；查看当前数据库中有哪些表。

```
show tables;
```

创建表：通常使用createtable表名(字段1字段类型,字段2字段类型,字段3字段类型,…)；来创建一张表。如：create table Phone_table(pid INT, name CHAR(20), price INT);

因为我们已经写好了DDL(Data Definition Language) 。文件名为：novahotel.sql存放于Mac系统的首目录developer下。用于创建所有的数据表，使用以下命令直接引入文件即可。

Source developer/nova/novahostel.sql
至此数据库的建立和配置完成。

```
/* NovaHostel DDL Source Server Type : MySQL Source Schema : NovaHostel */ SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0; -- ---------------------------- -- Table structure for admin --
---------------------------- DROP TABLE IF EXISTS `admin`; CREATE TABLE `admin` ( `username` varchar
(10) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '用户名', `password` int(10) NOT
NULL COMMENT '密码', PRIMARY KEY (`username`) USING BTREE ) ENGINE = InnoDB CHARACTER SET = utf8
COLLATE = utf8_general_ci ROW_FORMAT = Dynamic; -- ---------------------------- -- Records of admin
-- ---------------------------- INSERT INTO `admin` VALUES ('admin', 111111); --
---------------------------- -- Table structure for guests -- ---------------------------- DROP
TABLE IF EXISTS `guests`; CREATE TABLE `guests` ( `id` int(10) NOT NULL AUTO_INCREMENT, `name`
varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL, `sex` varchar(10) CHARACTER SET
utf8 COLLATE utf8_general_ci NOT NULL, `card` bigint(20) NOT NULL, `phone` bigint(20) NOT NULL,
`enterTime` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL, `exitTime` varchar(30)
CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL, `h_type` varchar(10) CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL, `num` int(10) NOT NULL, PRIMARY KEY (`id`) USING BTREE, INDEX `num`(`num`)
USING BTREE, INDEX `h_type`(`h_type`) USING BTREE, CONSTRAINT `guests_ibfk_1` FOREIGN KEY (`num`)
REFERENCES `home` (`num`) ON DELETE RESTRICT ON UPDATE RESTRICT, CONSTRAINT `guests_ibfk_2` FOREIGN
KEY (`h_type`) REFERENCES `home` (`h_type`) ON DELETE RESTRICT ON UPDATE RESTRICT ) ENGINE = InnoDB
AUTO_INCREMENT = 8 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic; 图5-1 DDL
```

5.2 系统实现关键技术分析

系统主要基于SSM框架开发，SSM框架是将Spring框架、SpringMVC框架、MyBatis框架的整合，是标准的MVC模式。以简化在web开发中繁琐、重复的操作，把开发精力专注于业务处理上。它是继SSH之后比较主流的Java EE企业级框架，适用于搭建各种大型的企业级应用系统。标准的SSM框架有四层，分别是DAO层（mapper），SERVICE层，CONTROLLER层和VIEW层。

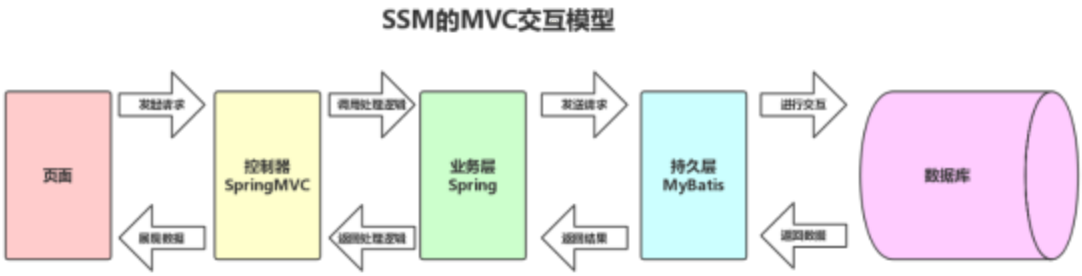


图5-2 SSM交互流程

该管理信息系统实现了代码之间的低耦合，提高了代码的可维护性和可扩展性。使用Spring实现业务对象管理，使用Spring MVC负责请求的转发和视图管理，Mybatis作为数据对象的持久化引擎。 基于约定和配置优先原则，简化了应用程序的开发，同时也减少了冗余代码的编写。而低耦合和高内聚的特性，使得单元测试和集成测试变得更加容易。同时，本系统考虑到实际业务场景需求，加入了事务管理的支持。MyBatis提供了数据库事务的管理，Spring提供了事务管理的支持。

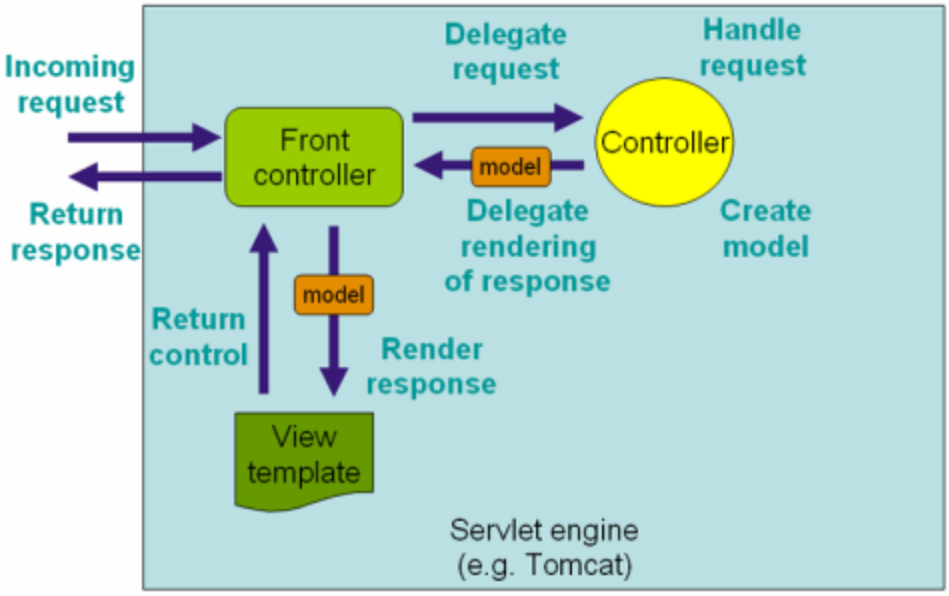


图5-3 Spring MVC 请求处理工作流程

Spring是整个项目中装配bean的大工厂，在配置文件中指定使用特定的参数去调用实体类的构造方法来实例化对象[12]。Spring的核心思想是IOC（控制反转），即不再需要显式地new一个对象，而是让Spring框架来完成这一切。 SpringMVC在项目中拦截用户请求，它的核心Servlet即DispatcherServlet承担中介或是前台这样的职责，将用户请求通过HandlerMapping去匹配Controller，Controller就是具体对应请求所执行的操作。

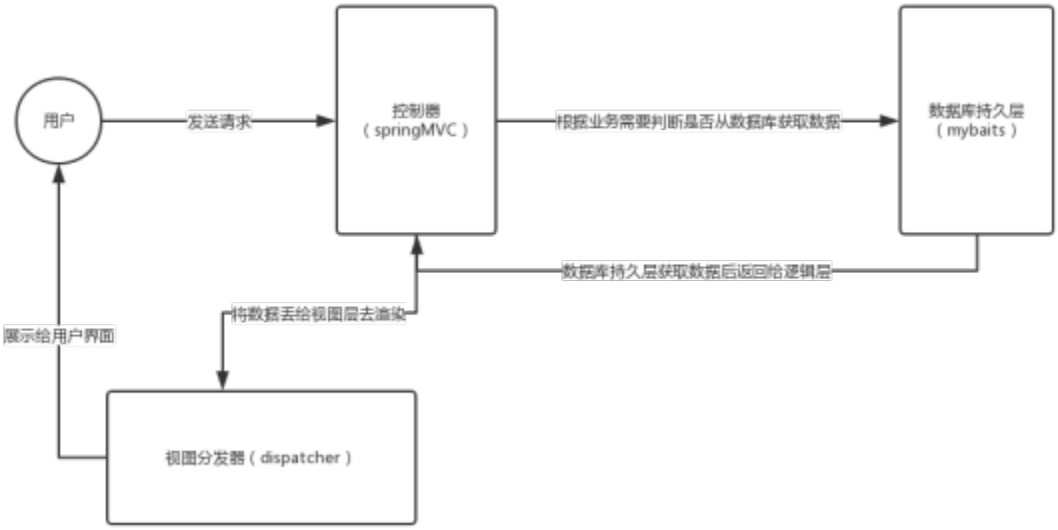


图5-4 请求工作流程

Spring的Web MVC 框架与许多其他 Web MVC 框架一样，是请求驱动的，围绕中央Servlet设计，该Servlet将请求分派到控制器并提供其他促进Web应用程序开发的功能。Spring Web MVC 的请求处理流程DispatcherServlet如下图所示。

Spring MVC是Spring提供的一个强大而灵活的web框架。借助于注解，Spring MVC提供了几乎是POJO的开发模式，使得控制器的开发和测试更加简单。这些控制器一般不直接处理请求，而是将其委托给Spring上下文中的其他bean，通过Spring的依赖注入功能，这些bean被注入到控制器中。

<!-切面 --> <aop:config> <aop:advisor advice-ref="txAdvice" pointcut="execution(* co.nova.hostel.*.*(..))" /> </aop:config> <!-配置Controller扫描 --> <context:component-scan base-package="co.nova.hostel.controller" /> <!-配置注解驱动 --> <mvc:annotation-driven /> <mvc:default-servlet-

handler /> <mvc:resources location="/WEB-INF/views/static/" mapping="/css/"/> <mvc:resources location="/WEB-INF/views/static/" mapping="/js/"/> <mvc:resources location="/WEB-INF/views/static/" mapping="/img/"/> <mvc:resources location="/static/" mapping="/css/"/> <mvc:resources location="/static/" mapping="/js/"/> <mvc:resources location="/static/" mapping="/img/"/> <mvc:resources location="/WEB-INF/views/" mapping="/views/"/> <!-配置视图解析器 --> <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"> <!-前缀 --> <property name="prefix" value="/WEB-INF/views/" /> <!-后缀 --> <property name="suffix" value=".jsp" /> </bean> <!--配置文件上传解析器--> <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"> <property name="maxUploadSize" value="10485760"/> <property name="defaultEncoding" value="utf-8"/> </bean> 图5-5 工程配置代码

MyBatis是对JDBC的封装，它让数据库底层操作变的透明。MyBatis的操作都是围绕一个sqlSessionFactory实例展开的。文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。

页面发送请求给控制器，控制器调用业务层处理逻辑，逻辑层向持久层发送请求，持久层与数据库交互，后将结果返回给业务层，业务层将处理逻辑发送给控制器，控制器再调用视图展现数据。

@Controller @RequestMapping("/guests") public class GuestsController { @Autowired GuestsServiceImpl guestsService; @RequestMapping("/add") public ModelAndView add(Guests guests){ ModelAndView mv = new ModelAndView(); guestsService.addGuests(guests); mv.setViewName("suc_g"); return mv; } @RequestMapping("/delete") public String delete(int id){ guestsService.deleteGuestsById(id); return "redirect:/guests/list"; } @RequestMapping("/list") public ModelAndView list(){ ModelAndView mv = new ModelAndView(); List<Guests> guestsList=guestsService.queryAllGuests(); mv.addObject("list",guestsList); mv.setViewName("guests_list"); return mv; } @RequestMapping("/update") public ModelAndView update1(int id){ ModelAndView mv = new ModelAndView(); Guests guests = guestsService.queryGuestsById(id); mv.addObject("g",guests); mv.setViewName("guests_update"); return mv; } } 图5-6 工程代码

MyBatis是一款优秀的持久层框架，它支持定制化SQL、存储过程以及高级映射。MyBatis避免了几乎所有的JDBC代码和手动设置参数以及获取结果集。MyBatis使用简单的XML或注解来配置和映射原生信息，将接口和Java的POJOs(Plain Ordinary Java Object,普通的 Java对象)映射成数据库中的记录。

该管理系统使用了事务，事务是逻辑上的一组操作，要么都执行，要么都不执行。

事务的特性（ACID）

原子性（Atomicity）： 一个事务（transaction）中的所有操作，或者全部完成，或者全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。即，事务不可分割、不可约简。

一致性（Consistency）： 在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设约束、触发器、级联回滚等。

隔离性（Isolation）： 数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括未提交读（Read uncommitted）、提交读（read committed）、可重复读（repeatable read）和串行化（Serializable）。

持久性（Durability）：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

事务传播行为

事务传播行为是为了解决业务层方法之间互相调用的事务问题[13]。当事务方法被另一个事务方法调用时，必须指定事务应该如何传播。例如：方法可能继续在现有事务中运行，也可能开启一个新事务，并在自己的事务中运行。

@Transactional注解是使用的最多的一个事务传播行为，如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的事务。如：

<tx:advice id="txAdvice" transaction-manager="transactionManager"> <tx:attributes> <!-传播行为 --> <tx:method name="save*" propagation="REQUIRED" /> <tx:method name="insert*" propagation="

REQUIRED" /> <tx:method name="add*" propagation="REQUIRED" /> <tx:method name="create*" propagation="REQUIRED" /> <tx:method name="delete*" propagation="REQUIRED" /> <tx:method name="update*" propagation="REQUIRED" /> <tx:method name="find*" propagation="SUPPORTS" read-only="true" /> <tx:method name="select*" propagation="SUPPORTS" read-only="true" /> <tx:method name="get*" propagation="SUPPORTS" read-only="true" /> <tx:method name="query*" propagation="SUPPORTS" read-only="true" /> </tx:attributes> </tx:advice> 图5-7 事务传播行为配置

如果外部方法没有开启事务的话，Propagation.REQUIRED修饰的内部方法会新开启自己的事务，且开启的事务相互独立，互不干扰。

@RequestMapping("/vip") public void excel_vip(HttpServletResponse response)throws IOException { response.setCharacterEncoding("UTF-8"); VipUsersCons vuc = new VipUsersCons(); List<Vip> vipList=vipService.queryAllVip(); // create excel file HSSFWorkbook wb = new HSSFWorkbook(); //create sheet page HSSFSheet sheet = wb.createSheet(vuc.vipinof); // create title HSSFRow titleRow = sheet.createRow(0); titleRow.createCell(0).setCellValue(vuc.code); titleRow.createCell(1).setCellValue(vuc.name); titleRow.createCell(2).setCellValue(vuc.sex); titleRow.createCell(3).setCellValue(vuc.cards); titleRow.createCell(4).setCellValue(vuc.phone); titleRow.createCell(5).setCellValue(vuc.type); titleRow.createCell(6).setCellValue(vuc.signdate); titleRow.createCell(7).setCellValue(vuc.endate); for(Vip vip:vipList){ HSSFRow dataRow = sheet.createRow(sheet.getLastRowNum()+1); dataRow.createCell(0).setCellValue(vip.getId()); dataRow.createCell(1).setCellValue(vip.getName()); dataRow.createCell(2).setCellValue(vip.getSex()); dataRow.createCell(3).setCellValue(vip.getCard()); dataRow.createCell(4).setCellValue(vip.getPhone()); dataRow.createCell(5).setCellValue(vip.getV_Type()); dataRow.createCell(6).setCellValue(vip.getStartTime()); dataRow.createCell(7).setCellValue(vip.getEndTime()); } // make Excel file name response.setContentType("application/octet-stream;charset=utf-8"); response.setHeader("Content-Disposition", "attachment; filename=" + new String(vuc.viplist.getBytes(),"iso-8859-1") + ".xls"); OutputStream ouputStream = response.getOutputStream(); wb.write(ouputStream); ouputStream.flush(); ouputStream.close(); } 图5-8 Java Control层代码

如果外部方法开启事务并且被Propagation.REQUIRED的话，所有Propagation.REQUIRED修饰的内部方法和外部方法均属于同一事务，只要一个方法回滚，整个事务均回滚。

该管理信息系统的每个业务方法都包括了多个原子性的数据库操作，比如下面的新增会员 add(Vip vip) 方法中就有两个原子性的数据库操作。这些原子性的数据库操作是有依赖的，它们要么都执行，要不就都不执行。事务能否生效数据库引擎是否支持事务是关键。常用的MySQL数据库默认使用支持事务的 innodb引擎。该管理系统也是基于innodb引擎。如果把数据库引擎变为 myisam，那么程序就不再支持事务了。

@RequestMapping("/add") public ModelAndView add(Vip vip){ ModelAndView mv = new ModelAndView(); vipService.addVip(vip); mv.setViewName("suc_v"); return mv; }

第六章系统测试

系统测试主要工作内容是验证和确认。验证是保证软件正确地实现了些特定功能的系列活动，即保证软件做了你所期望的事情。验证确定软件生存周期中的个给定阶段的产品是否达到前阶段确立的需求的过程[14]。

6.1 测试的方法

软件测试，按照测试过程可分为单元测试、集成测试、确认测试及系统测试等；按照测试内容分可分为功能测试、性能测试、界面测试、链接测试等[15]。

单元测试：最初阶段，测试的主要对象就是单元。

集成测试：是在单元测试过程之后来进行，属于测试的关键环节。各个测试结束的模块进行综合分析，组装成比较大的模块进行系统集成分析测试。

确认测试：这一测试过程是发生在集成测试之后的测试过程，用来仿真软件的功能和参数能不能实现使用者的要求。

系统测试：也称为产品测试。确认测试在满足用户的要求之后，系统测试要进行检测实际环境和软件能够很好的结合协调工作。