|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **scu-logo cn** | **文档编号** |  | **2016级** | **公开** |
| **其他** |  |  |  |
|  | **数据库课程设计** | | | |

**《数据库课程设计·期末报告》**

**民宿管理系统**

**学生姓名： 何长鸿**

**学生学号： 2016141482154**

**电子邮件： geekhch@qq.com**

**指导老师： 袁道华**

**报告日期： 2018.12.10**

**章节目录**

[一、 实验目的 1](#_Toc27301)

[二、 开发环境及工具 1](#_Toc28532)

[1. 设备 1](#_Toc21022)

[2. 支撑软件 1](#_Toc651)

[三、 需求分析 2](#_Toc9460)

[1. 需求软件功能说明 2](#_Toc11224)

[2. 对功能的一般性规定 2](#_Toc28994)

[3. 对性能的一般性规定 2](#_Toc25974)

[四、 任务概述 3](#_Toc885)

[1. 目标 3](#_Toc8660)

[2. 系统的特点 3](#_Toc25878)

[3. 假定和约束 3](#_Toc27107)

[五、 设计 4](#_Toc12975)

[1. 开发环境和工具 4](#_Toc18911)

[2. ER模型图 4](#_Toc2565)

[3. 数据表结构定义 4](#_Toc26915)

[六、 功能模块结构图 6](#_Toc6913)

[1. 功能模块介绍 6](#_Toc19952)

[七、 程序功能模块处理流程 7](#_Toc30970)

[1. 客户端功能流程图 7](#_Toc5130)

[2. 服务器端功能流程图 9](#_Toc30530)

[3. 关键功能算法流程 10](#_Toc21908)

[八、 服务器架构 11](#_Toc29472)

[1. 主要代码包和类结构 11](#_Toc16288)

[九、 用户界面 12](#_Toc16036)

[1. 游客 12](#_Toc22591)

[2. 管理员 15](#_Toc15085)

[十、 部分核心代码 18](#_Toc27161)

[1. 服务器后端 18](#_Toc26078)

[2. 分页显示 26](#_Toc27103)

[3. 图片上传实现代码 28](#_Toc32350)

**图目录**

[图 1 ER模型图 4](#_Toc27185)

[图 2 表结构定义 5](#_Toc31909)

[图 3 功能模块 6](#_Toc7044)

[图 4 客户端游客功能模块处理流程 7](#_Toc18694)

[图 5 客户端商家功能操作流程 8](#_Toc21485)

[图 6 服务器请求处理流程 9](#_Toc12926)

[图 7 剩余房间计算控制流程 10](#_Toc27207)

[图 8 项目代码包结构 11](#_Toc4845)

[图 9 用户注册页面 12](#_Toc13469)

[图 10 用户登录页面 13](#_Toc1749)

[图 11 房间浏览界面 13](#_Toc31389)

[图 12 预定房间界面 14](#_Toc4972)

[图 13 输入预定客户信息界面 14](#_Toc27721)

[图 14 确认支付界面 15](#_Toc29423)

[图 15 查看预定订单 15](#_Toc28912)

[图 16 房间分类管理 16](#_Toc31544)

[图 17 房间对象管理 16](#_Toc25181)

[图 18 用户管理 17](#_Toc21139)

[图 19 订单管理 17](#_Toc2126)

**表目录**

[表 1 民宿旅游需求规定 2](#_Toc580)

民宿管理系统设计报告

何长鸿

2016141482154

# 实验目的

开发完成一个小型的民宿管理。完成从需求分析、数据模式设计到编码实现、系统调试的所有流程；通过此一图书管理系统的实现，在实践中掌握数据库系统设计开发的特点、方法和步骤。

# 开发环境及工具

## 设备

本系统采用B/S模式开发，开服务器发语言使用 java

1. 服务器运行于Windows或Linux系统
2. 客户端全平台支持

## 支撑软件

### 服务器

1. MySQL数据库
2. Tomcat 8.5
3. Java SDK1.8

### 客户端

1. 主流浏览器均可，例如Firefox、Chrome、Safari等

# 需求分析

## 需求软件功能说明

|  |  |
| --- | --- |
| 分类 | 功能 |
| 管理 | 1. 房间类型资源上架，修改，统计，SKU管理。 2. 民宿资源展示 3. 房间类型所含属性修改 4. 房间分类图片上传 5. 对订单的管理：查询，修改，删除。 6. 用户信息管理、增删改查 7. 房间信息管理：房间数量、名称、属性修改、原价、活动价 8. 订单信息修改 9. 订单信息多条件组合查询 10. 一键查询已购买超过三天未评论的订单 11. 房间存量同步统计 12. 各种列表清单的导出exel文件和打印 |
| 游客功能 | 1. 预览商家 2. 民宿资源信息 3. 预订下单 4. **空闲房间不足时拒绝购买** 5. 预定订单查看 6. 评论已购买商品 7. 注册 8. 登录 |

表 1民宿旅游需求规定

## 对功能的一般性规定

软件采用可视化图形界面，界面格式统一，界面功能键排版能使用户能较快找到所需功能按钮，设定默认值以防每次操作要求输入太多，统一的错误提示风格。

## 对性能的一般性规定

1. 精度

浮点型数据保存2位以上小数，百分比数据小数点后2位小数。

1. 输入输出要求

信息录入时，数字、字符、时间日期的格式和长度应遵照提示要求，否则不能提交或提示出错。系统输出时，与金额相关的数值都保留2位小数；百分比数值小数点后保留2位小数。

1. 数据管理能力要求

需要管理的问卷

1. 其他专门要求
   * 1. 用户的信息加密、信息认证（登陆访问）
     2. 使用方便
     3. 可维护性、可补充性、易读性、可靠性

# 任务概述

## 目标

本项目旨在开发一款满足用户商户需求的民宿旅游平台。为用户提供更好的房屋以及售后服务，同时为商户提供更多地出租机会。同时营造一个游客们的交流平台，为平台提供一定的人文气息。

## 系统的特点

1. 易用性：人性化设计，面向社会大众使用，简单操作的功能，美观和谐的用户交互界面
2. 实用性：以实际应用为出发点，向旅游大众，名宿商家，旅游管理局做出详细调研，做出最优化设计，满足各方面需求。
3. 高效性：通过互联网技术，使得民宿信息实时更新，让游客随时随地货比三家，选择自己最满意的名宿，提高出行的满意度，通过网络支付手段减少交易流程，使得出行变得高效快捷。
4. 安全性：外部安全性：能够使得旅游局有序管理，用户的意见也能得到第一时间反馈，保证用户出行的安全，做到有记录可循。内部安全性：利用javaweb优秀的封装机制，提供多种安全保护方法，保证了客户资料隐私的安全和完整。
5. 开发性：可以用其他软件，平台完成必要的整合应用。

## 假定和约束

1. 假定I 经费限制

选择Web服务器与内存服务器只能选择内存较小的服务器，则开发完成以后系统的浏览速度不高，并行操作人数也受限。

1. 假定II 开发期限

本项目的开发周期为17个周，由于参与项目开发的学生都是第一次着手开发，能力尚不足，在这规定的开发周期内面临着不能成功开发出一个完整的系统。

# 设计

## 开发环境和工具

1. 使用B/S模式开发
2. MySQL Community 5.5
3. 服务器使用JAVA语言，SSM框架，其中Mybatis数据库配置阿里Druid数据库连接池。

## ER模型图

在本系统中，涉及的实体性有游客、商家、订单、房间、图片、用户评论等，但为了房间类别关系，去掉大量房间类别重复字段，并且对此形成约束，所以将房间类型抽象为实体型，整个系统数据库ER模型图如下所示:

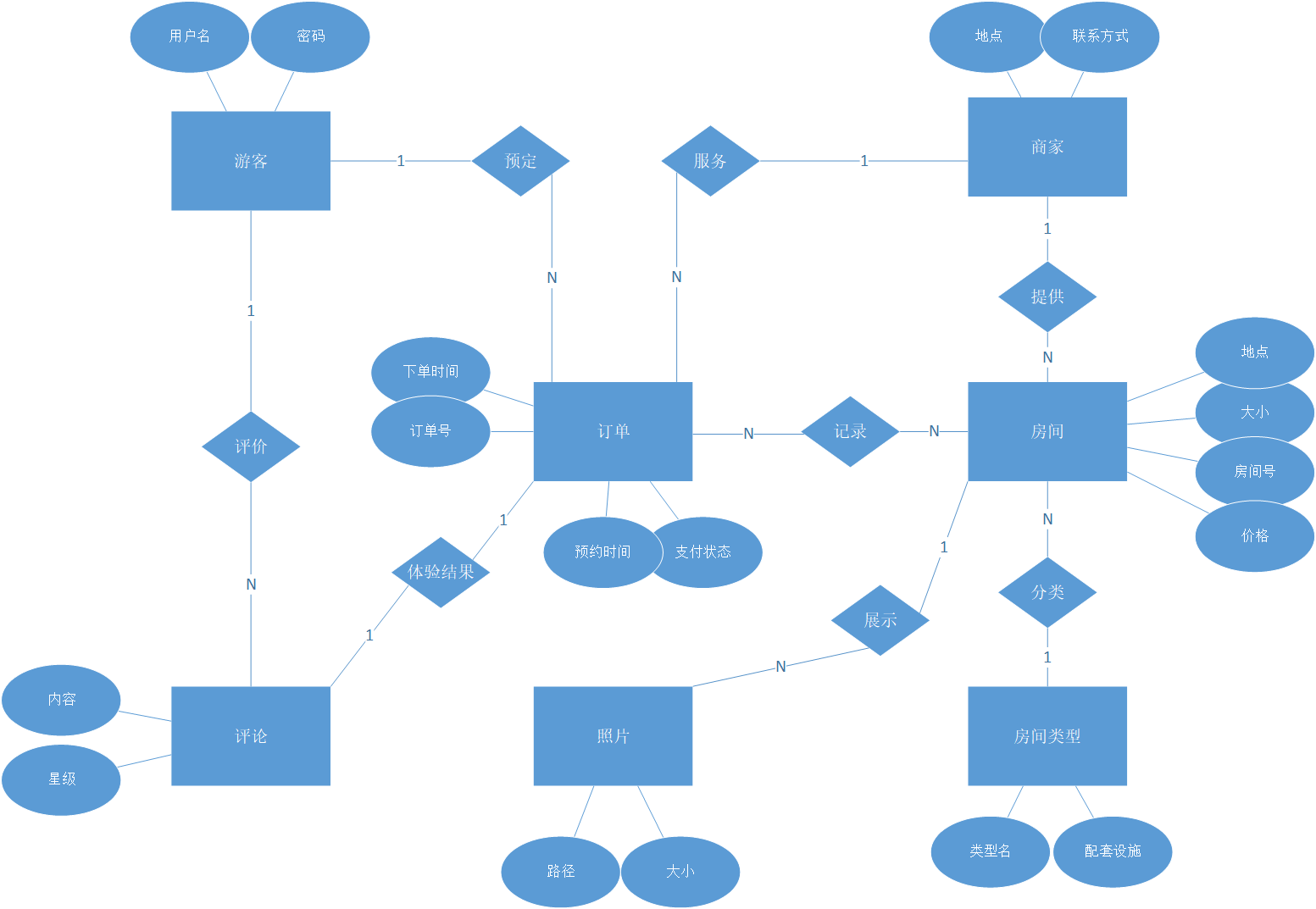


图 1 ER模型图

## 数据表结构定义

在下图中给出了数据表名，字段名、字段类型、数据长度以及数据表说明



图 2 表结构定义

# 功能模块结构图

## 功能模块介绍

本系统分为商家与游客两大部分，使用B/S模式开发，两者功能相互对应。主要功能模块包括用户登录与注销、用户信息管理、房源管理、订单管理、图片库管理、预定等流程。同一个用户账户既可以是游客也可以是商家，但会根据用户发布的房源情况自动判断是否有相关功能的操作权限。

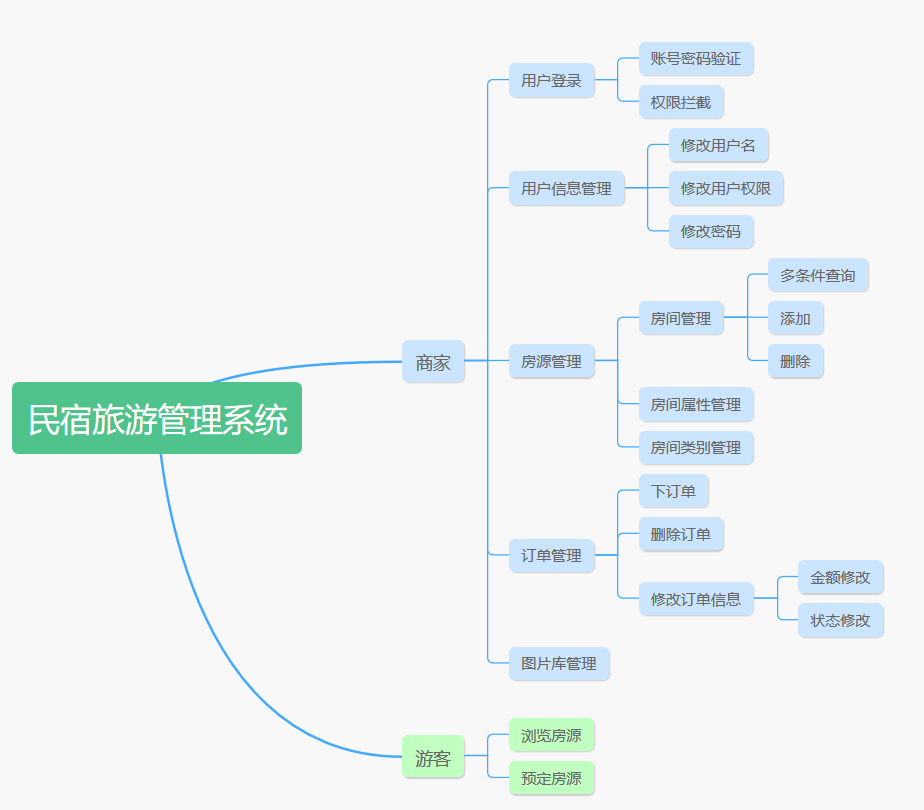


图 3功能模块

# 程序功能模块处理流程

## 客户端功能流程图

### 客户端游客注册及登陆、预定功能流程图

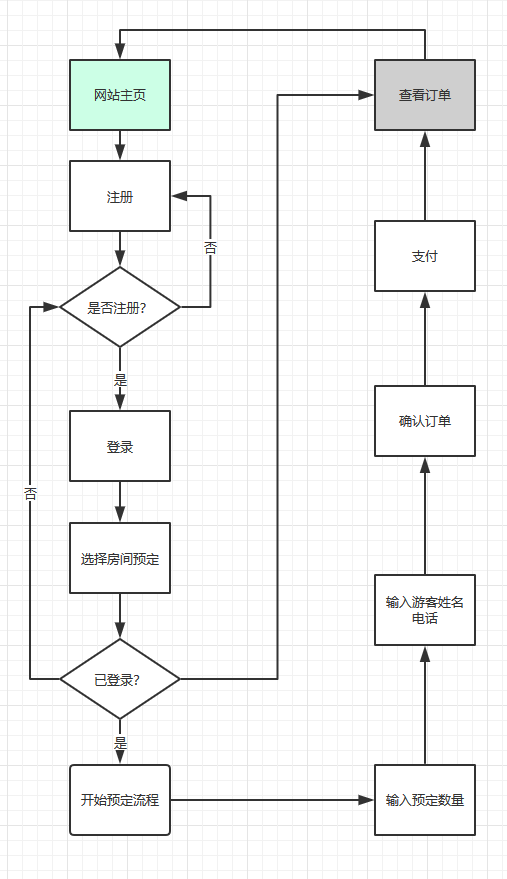


图 4客户端游客功能模块处理流程

### 客户端商家功能流程图

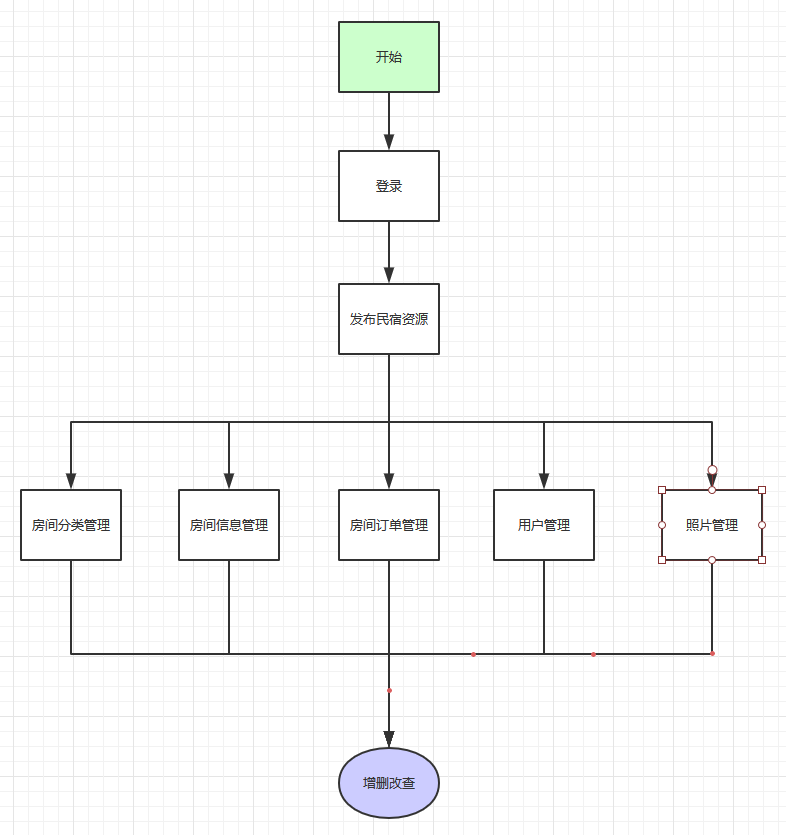


图 5客户端商家功能操作流程

## 服务器端功能流程图

### 请求处理流程

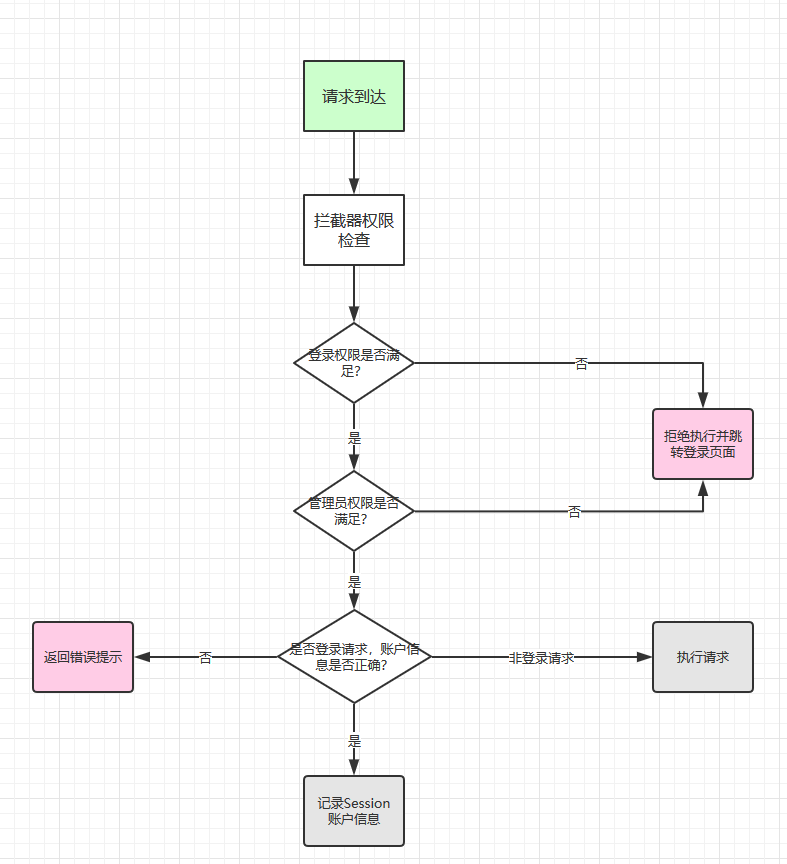


图 6服务器请求处理流程

## 关键功能算法流程

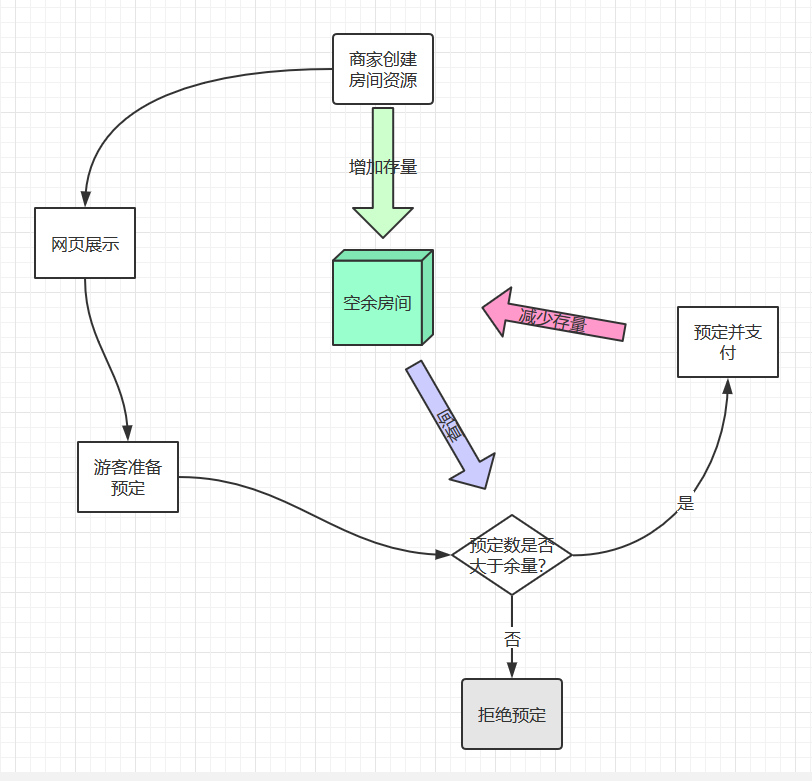


图 7剩余房间计算控制流程

# 服务器架构

## 主要代码包和类结构

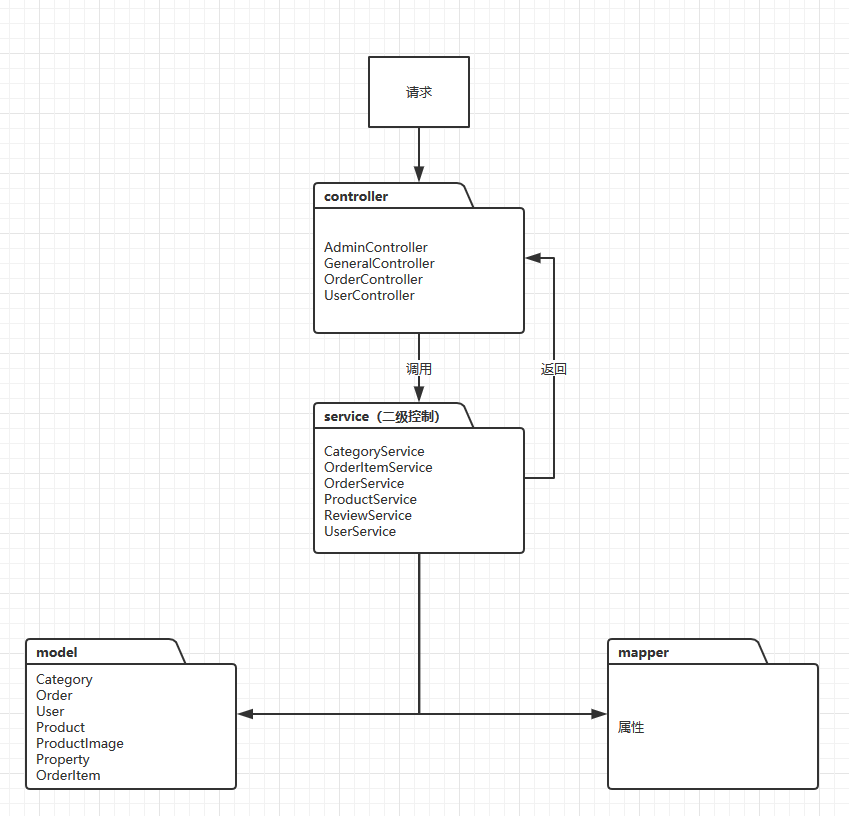


图 8项目代码包结构

# 用户界面

以下使用用户故事的方式来描述软件功能使用流程。

## 游客

1. 打开网站主页
2. 注册并登录账号



图 9用户注册页面

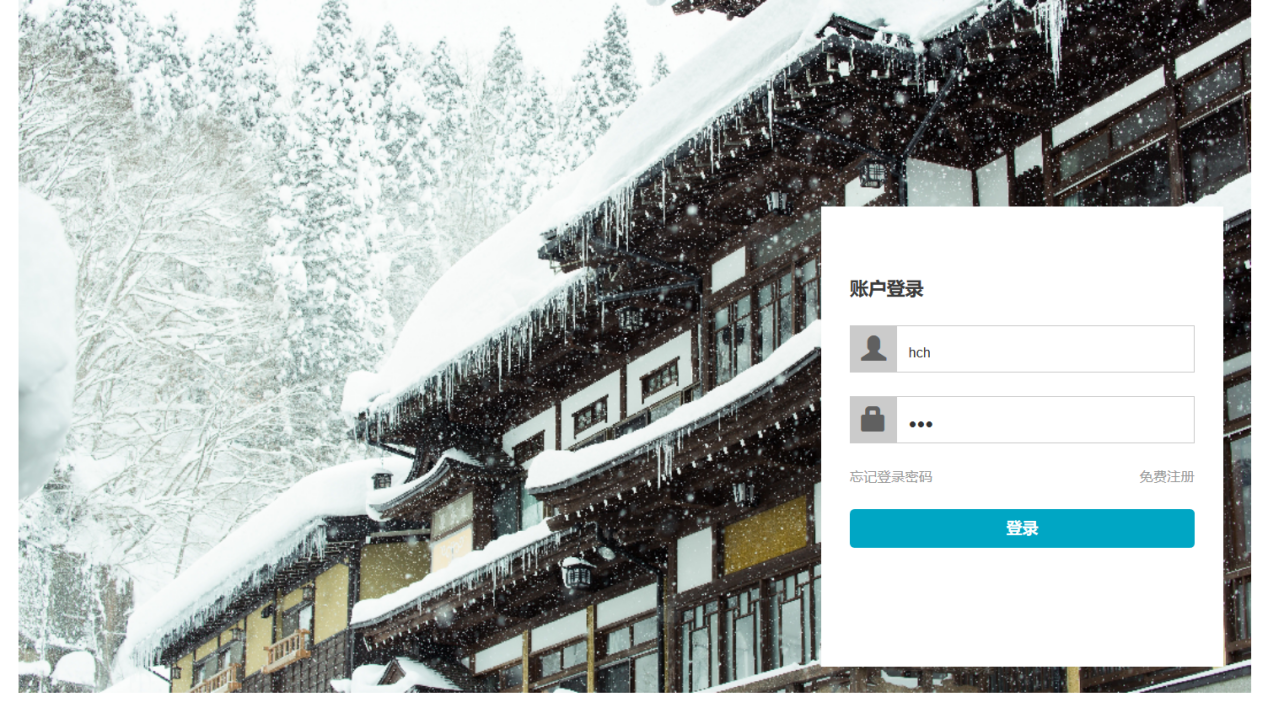


图 10用户登录页面

1. 浏览并选择预定房间

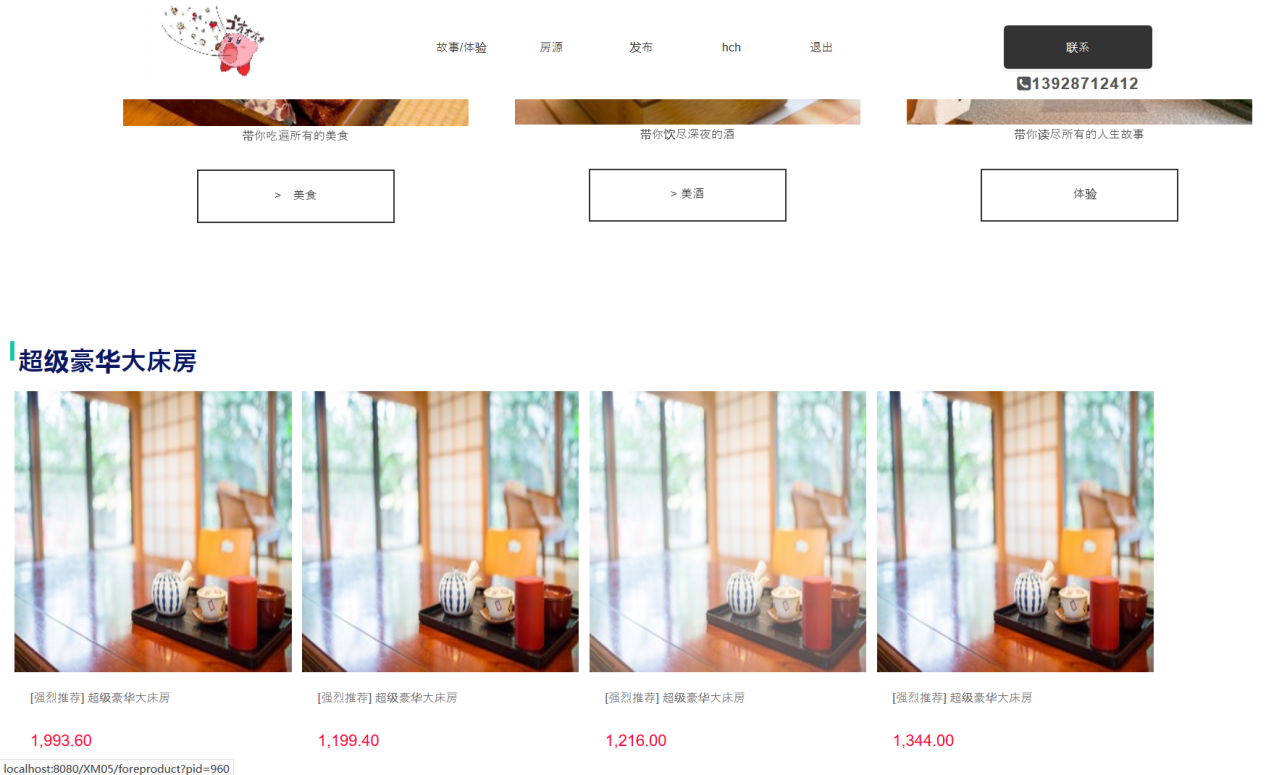


图 11房间浏览界面

1. 如果此时系统检测到用户未登录，则跳转到步骤（2）
2. 输入预定房间数量并确定

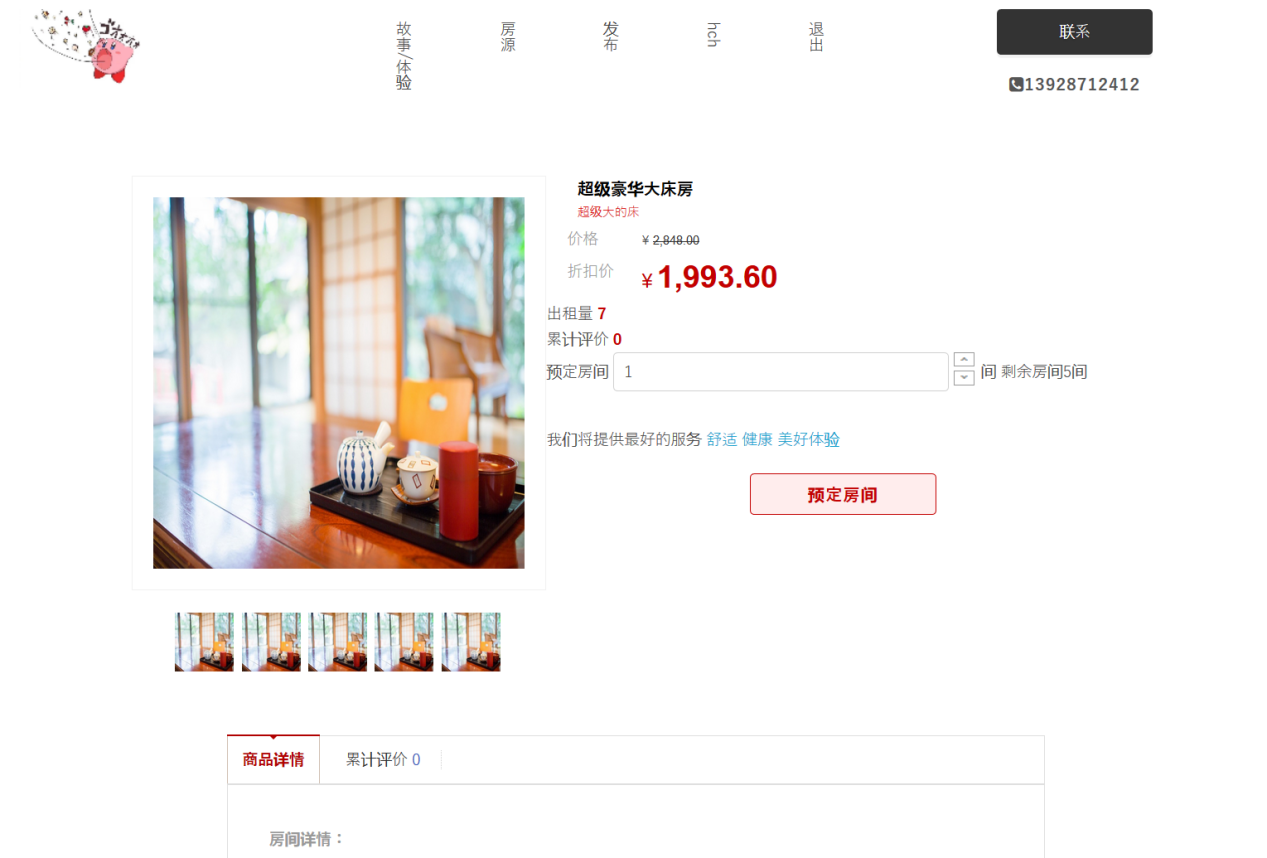


图 12预定房间界面

1. 填入客户信息并确认提交订单

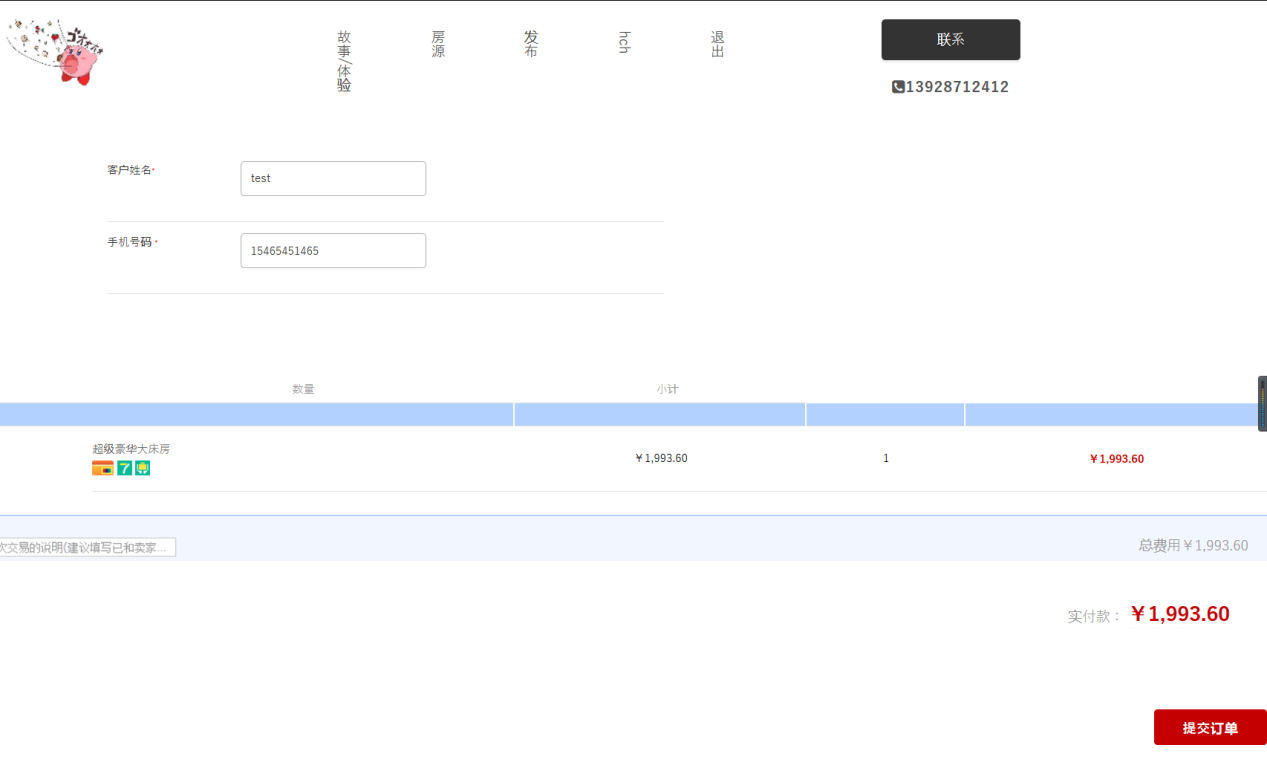


图 13输入预定客户信息界面

1. 确认支付



图 14确认支付界面

1. 查看已预定房间

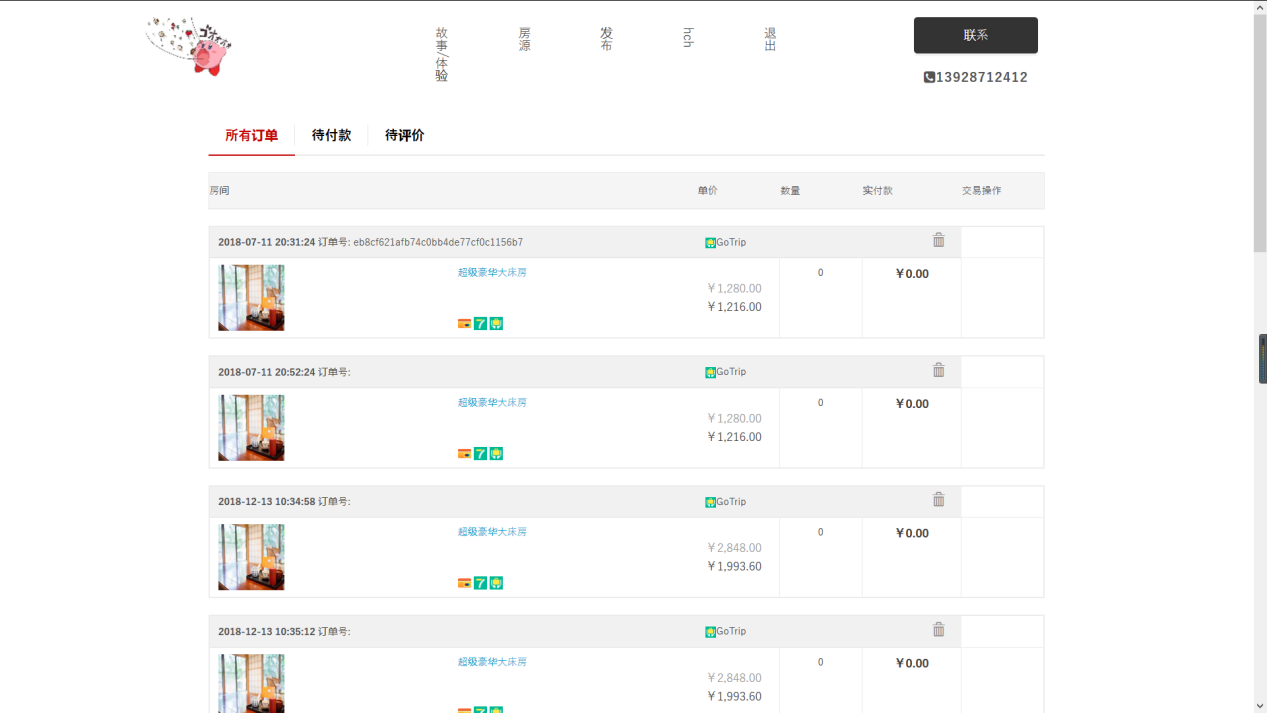


图 15查看预定订单

## 管理员

1. 注册、登录
2. 房源分类管理（增删改）





图 16房间分类管理

1. 某个分类下的房间管理



图 17房间对象管理

1. 用户管理

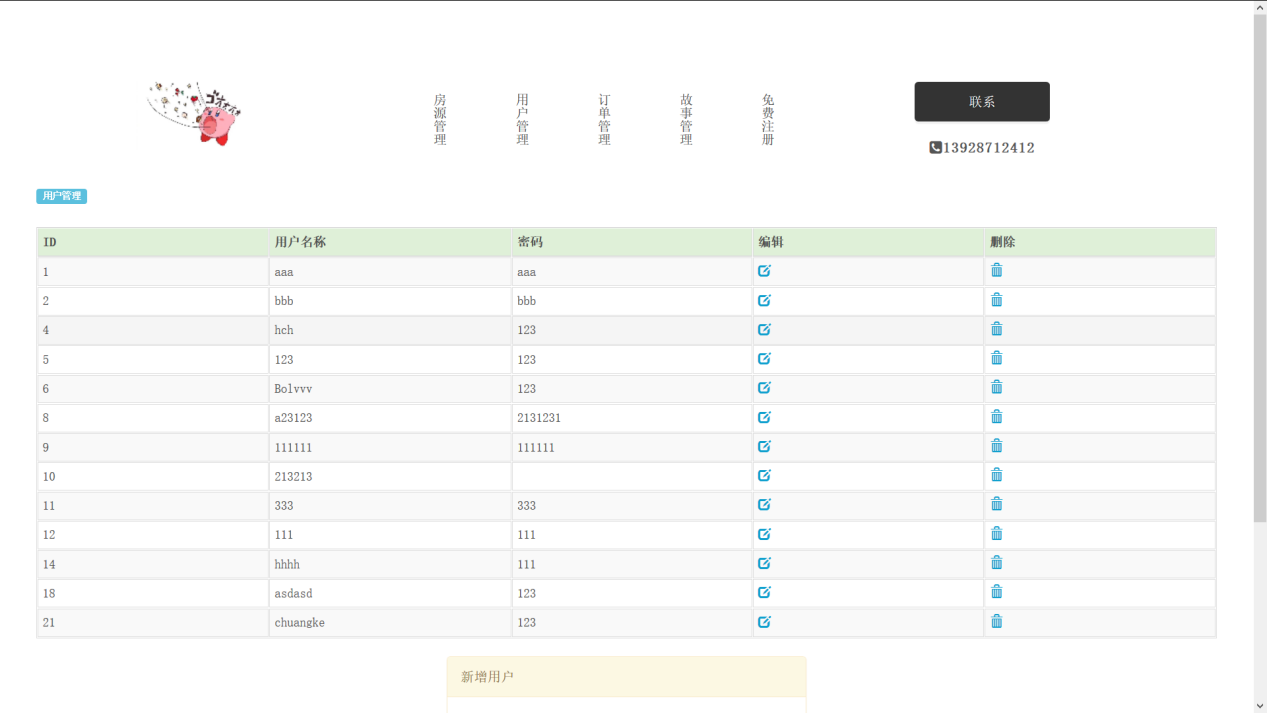




图 18用户管理

1. 订单管理

包括支付状态、金额、订单中预定的房间数量、买家信息、支付时间等数据项。管理功能提供了增删改查等操作。

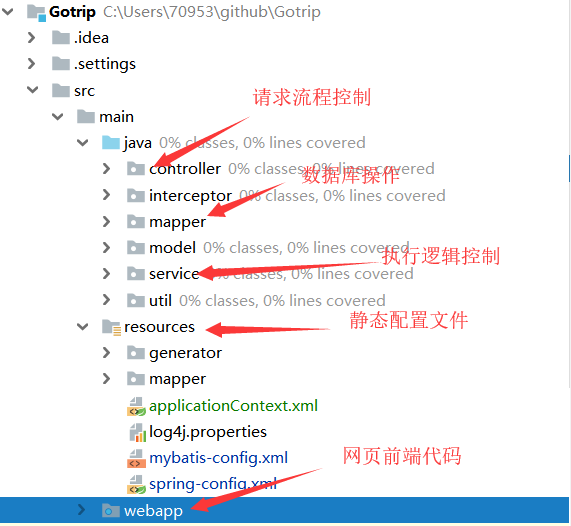


图 19订单管理

# 部分核心代码

## 服务器后端

后端采用JAVA语言，SSM框架编写，主要有model、mapper、service、controller等包结构。一下主要代码中仅列举每个主要包中的一个类的代码。



### 用户登录检查实现代码

|  |
| --- |
| **package** interceptor; **public class** AdminInterceptor **implements** HandlerInterceptor {  **private static** Logger *logger* = Logger.*getLogger*(AdminController.**class**);   @Override  **public boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** IOException {  *logger*.info(**"执行spring拦截器"**);  *//检查当前sessions是否有登录用户* User user = (User) request.getSession().getAttribute(**"user"**);  **if**(user != **null**)  **return true**;  **else** response.sendRedirect(request.getContextPath()+**"/login.jsp"**);  **return false**;  }   @Override  **public void** postHandle(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response, Object handler, ModelAndView modelAndView) **throws** Exception {   *logger*.info(**"postHandle"**);  }   @Override  **public void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception e) **throws** Exception{  *logger*.info(**"spring拦截结束执行"**);  } } |

### 客户预定服务期端处理代码（GeneralController.java）

|  |
| --- |
| **package** controller;  @Controller @RequestMapping(**""**) **public class** GeneralController {  **private static** Logger *logger* = Logger.*getLogger*(OrderController.**class**);  **private** ModelAndView **modelAndView** = **new** ModelAndView();    @Autowired  UserService **userService**;  @Autowired  CategoryService **categoryService**;  @Autowired  OrderItemService **orderItemService**;  @Autowired  OrderService **orderService**;  @Autowired  ProductService **productService**;  @Autowired  ReviewService **reviewService**;   @RequestMapping(value = **"/forelogin"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public** ModelAndView logIn(HttpServletRequest request, HttpServletResponse response,String name, String password) **throws** Exception{  User user = **userService**.getUser(name);  **if**(user != **null** && user.getPassword().equals(password)){  *//登陆成功* request.getSession(**true**).setAttribute(**"user"**,user);  *logger*.info(request.getPathInfo());  **return** home(request);  }  **return null**;  }   @RequestMapping(value = **"/forelogout"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public void** logOut(HttpServletRequest request, HttpServletResponse response) **throws** Exception{  request.getSession().invalidate();  response.sendRedirect(**"home.jsp"**);  }   @RequestMapping(value = **"/foresearch"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public** ModelAndView search(HttpServletRequest request, HttpServletResponse response, String keyword) **throws** Exception{  List<Product> ps = **productService**.listProducts(keyword);   **modelAndView**.addObject(**"ps"**,ps);  **modelAndView**.setViewName(**"/searchResult"**);  **return modelAndView**;  }   @RequestMapping(value = **"/foreproduct"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public** ModelAndView product(Integer pid) **throws** Exception{  Product p = **productService**.getProduct(pid);  List<Propertyvalue> pvs = **productService**.listProprotyValue(p.getId());  **modelAndView**.addObject(pvs);  **modelAndView**.addObject(**"p"**, p);  **modelAndView**.setViewName(**"/product"**);  **return modelAndView**;  }   @RequestMapping(value = **"/forecheckLogin"**, produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public int** check(HttpServletRequest request) **throws** Exception{  User user = (User) request.getSession().getAttribute(**"user"**);  **if**(user != **null**)  **return** 1;  **return** 0;  }   @RequestMapping(value = **"/forebuyone"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public** ModelAndView forebuyone(HttpServletRequest request, Orderitem orderitem) **throws** Exception{  User user = (User) request.getSession().getAttribute(**"user"**);  Map map = **orderItemService**.buyOne(user, orderitem);  request.getSession(**true**).setAttribute(**"order"**,map.get(**"order"**));  request.getSession(**true**).setAttribute(**"pid"**,orderitem.getPid());  Product product = **productService**.getProduct(orderitem.getPid());  product.setSaleCount(product.getSaleCount()+orderitem.getNumber());  **productService**.update(product);  **modelAndView**.addObject(**"ois"**,map.get(**"ois"**));  **modelAndView**.addObject(**"total"**,((Order)map.get(**"order"**)).getTotal());  **modelAndView**.setViewName(**"/buy"**);  **return modelAndView**;  }     @RequestMapping(value = **"/home"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public** ModelAndView home(HttpServletRequest request) **throws** Exception{  List<Category> c = **categoryService**.listTwoCategory();  **for**(Category cs: c){  cs.setProducts(**productService**.listProducts(cs.getId()));  }  **modelAndView**.addObject(**"cs"**,c);  **modelAndView**.setViewName(**"home"**);  **return modelAndView**;  }   @RequestMapping(value = **"/forecreateOrder"**,produces = **"text/html;charset=UTF-8"**)  @ResponseBody  **public** ModelAndView buy(Order order, HttpServletRequest request) **throws** Exception{  */\*\*确定支付\*\*/* Order origin\_order = (Order)request.getSession().getAttribute(**"order"**);  origin\_order.setOrderCode(UUID.*randomUUID*().toString().replace(**"-"**, **""**).toLowerCase());  origin\_order.setReceiver(order.getReceiver());  origin\_order.setMobile(order.getMobile());  origin\_order.setUserMessage(order.getUserMessage());  origin\_order.setStatus(**"waitReview"**);  origin\_order.setPayDate(**new** Date());   **orderService**.update(origin\_order);   **modelAndView**.addObject(**"o"**,origin\_order);  **modelAndView**.setViewName(**"/confirmPay"**);  **return modelAndView**;  }   @RequestMapping(value = **"/foreorderConfirmed"**, produces = **"text/html; charset=UTF-8"**)  @ResponseBody  **public** ModelAndView confirm() **throws** Exception{  **modelAndView**.setViewName(**"orderConfirmed"**);  **return modelAndView**;  }   @RequestMapping(value = **"/forebought"**, produces = **"text/html; charset=UTF-8"**)  @ResponseBody  **public** ModelAndView bought(HttpServletRequest request) **throws** Exception{  **modelAndView**.setViewName(**"bought"**);  User user = (User) request.getSession().getAttribute(**"user"**);  **modelAndView**.addObject(**"os"**, **orderService**.getOrders(user));  **return modelAndView**;  }   @RequestMapping(value = **"/deleteorder"**, produces = **"text/html; charset=UTF-8"**)  @ResponseBody  **public void** deleteorder(HttpServletRequest request,HttpServletResponse response, Integer oid) **throws** Exception{  User user = (User) request.getSession().getAttribute(**"user"**);  **orderService**.deleteOrder(oid);  response.sendRedirect(**"forebought"**);  }   @RequestMapping(value = **"/forestory"**, produces = **"text/html; charset=UTF-8"**)  @ResponseBody  **public** ModelAndView story()**throws** Exception{  **modelAndView**.setViewName(**"story"**);  **return modelAndView**;  }   @RequestMapping(value = **"/forereview"**, produces = **"text/html; charset=UTF-8"**)  @ResponseBody  **public** ModelAndView review(**int** oid, HttpServletRequest request)**throws** Exception{  Product product = **productService**.getProduct((**int**)request.getSession().getAttribute(**"pid"**));  List<Review> reviews = **reviewService**.listReview(product.getId());  **modelAndView**.addObject(**"re"**, reviews);  **modelAndView**.addObject(**"p"**,product);  **modelAndView**.addObject(**"u"**,request.getSession().getAttribute(**"user"**));  **modelAndView**.setViewName(**"review"**);  **return modelAndView**;  }   @RequestMapping(value = **"/foredoreviewed"**, produces = **"text/html; charset=UTF-8"**)  @ResponseBody  **public** ModelAndView reviewed(Review review, HttpServletRequest request)**throws** Exception{  **reviewService**.insert(review);  Product product = **productService**.getProduct(review.getPid());  product.setReviewCount(product.getReviewCount()+1);  **productService**.update(product);  **return** bought(request);  } } |

### 对User表的数据库操作（UserMapper.java）

本系统采用mybaatis自动生成基本数据库操作代码，并且被封装在xml文件中，因此，仅对部分复杂数据库操作才需要重写SQL语句，并且以注解方式写在Mapper的接口中。

|  |
| --- |
| **package** mapper;  **import** java.util.List; **import** model.User; **import** model.UserExample; **import** org.apache.ibatis.annotations.Param; **import** org.apache.ibatis.annotations.Select; **import** org.springframework.stereotype.Repository;  @Repository **public interface** UserMapper {  **long** countByExample(UserExample example);   **int** deleteByExample(UserExample example);   **int** deleteByPrimaryKey(Integer id);   **int** insert(User record);   **int** insertSelective(User record);   List<User> selectByExample(UserExample example);   @Select(**"select \* from User"**)  List<User> selectAll();   User selectByPrimaryKey(Integer id);   *//使用注解方式执行查询代码* @Select(**"select \* from User where name = #{name}"**)  User selectByName(String name);   **int** updateByExampleSelective(@Param(**"record"**) User record, @Param(**"example"**) UserExample example);   **int** updateByExample(@Param(**"record"**) User record, @Param(**"example"**) UserExample example);   **int** updateByPrimaryKeySelective(User record);   **int** updateByPrimaryKey(User record); } |

### User模型类，用于Spring框架的注入（User.java）

|  |
| --- |
| **package** service;  **import** mapper.\*; **import** model.User; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Service;  **import** java.util.List;  @Service **public class** UserService **implements** UserInt {   @Autowired()  ProductMapper **productMapper**;   @Autowired  ProductimageMapper **productimageMapper**;  @Autowired  OrderitemMapper **orderitemMapper**;  @Autowired  OrderMapper **orderMapper**;  @Autowired  ReviewMapper **reviewMapper**;  @Autowired  UserMapper **userMapper**;   *//根据用户名查询用户* @Override  **public** User getUser(String name){  **return userMapper**.selectByName(name);  }    *//根据用户id查询用户* @Override  **public** User getUser(**int** id){  **return userMapper**.selectByPrimaryKey(id);  }    *//查询所有用户* @Override  **public** List<User> listAll(){  **return userMapper**.selectAll();  }    *//通过id删除用户* @Override  **public void** deleteByPrimaryKey(Integer id) {  **userMapper**.deleteByPrimaryKey(id);  }    *//新增用户* @Override  **public void** insert(User user) {  **userMapper**.insert(user);  }    *//更新用户信息* @Override  **public void** update(User user) {  **userMapper**.updateByPrimaryKey(user);  } } |

## 分页显示

|  |
| --- |
| **package** util;  **public class** Page {  **private int start**;  **private int count**;  **private int total**;  **private** String **param**;  **public int** getStart() {  **return start**;  }  **public void** setStart(**int** start) {  **this**.**start** = start;  }  **public int** getCount() {  **return count**;  }  **public void** setCount(**int** count) {  **this**.**count** = count;  }  **public** Page(**int** start, **int** count) {  **super**();  **this**.**start** = start;  **this**.**count** = count;  }    **public boolean** isHasPreviouse(){  **return start** != 0;   }  **public boolean** isHasNext(){  **return start** != getLast();  }    **public int** getTotalPage(){  **int** totalPage;  *// 假设总数是50，是能够被5整除的，那么就有10页* **if** (0 == **total** % **count**)  totalPage = **total** /**count**;  *// 假设总数是51，不能够被5整除的，那么就有11页* **else** totalPage = **total** / **count** + 1;    **if**(0==totalPage)  totalPage = 1;  **return** totalPage;    }    **public int** getLast(){  **int** last;  *// 假设总数是50，是能够被5整除的，那么最后一页的开始就是45* **if** (0 == **total** % **count**)  last = **total** - **count**;  *// 假设总数是51，不能够被5整除的，那么最后一页的开始就是50* **else** last = **total** - **total** % **count**;    last = last<0?0:last;  **return** last;  }    **public int** getTotal() {  **return total**;  }  **public void** setTotal(**int** total) {  **this**.**total** = total;  }  **public** String getParam() {  **return param**;  }  **public void** setParam(String param) {  **this**.**param** = param;  }   } |

## 图片上传实现代码

|  |
| --- |
| *//添加分类 //特别注意，由于本操作涉及图片保存，更换运行环境是需要重新配置路径* @RequestMapping(value = **"/admin\_category\_add"**,produces = **"text/html;charset=UTF-8"**) @ResponseBody **public** ModelAndView admin\_category\_add(@RequestParam(**"name"**) String name, @RequestParam(**"filepath"**) MultipartFile filepath) **throws** IOException {  Category category = **new** Category();  category.setName(name);  Integer cid = categoryService.insert(category);  String path1 = **"C:\\Users\\70953\\github\\Gotrip\\src\\main\\webapp\\img\\category\\"**+cid.toString()+**".jpg"**;  String path2 = **"C:\\Users\\70953\\github\\Gotrip\\target\\Gotrip\\img\\category\\"**+cid.toString()+**".jpg"**;  filepath.transferTo(**new** File(path1));  FileUtils.copyFile(**new** File(path1),**new** File(path2));  mav = listCategory();  **return** mav; } |