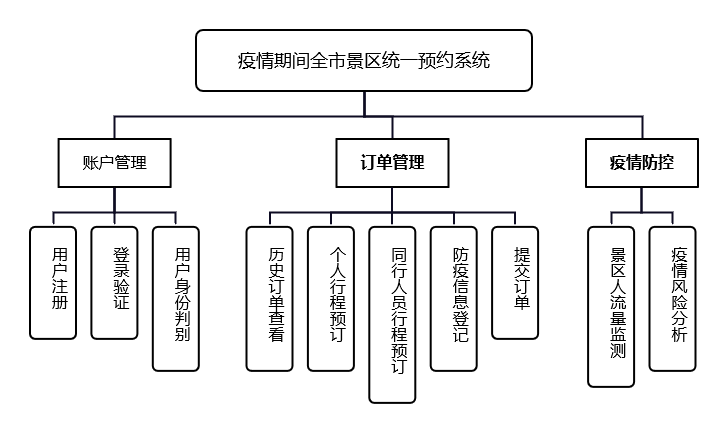
# 景区订票系统详细设计

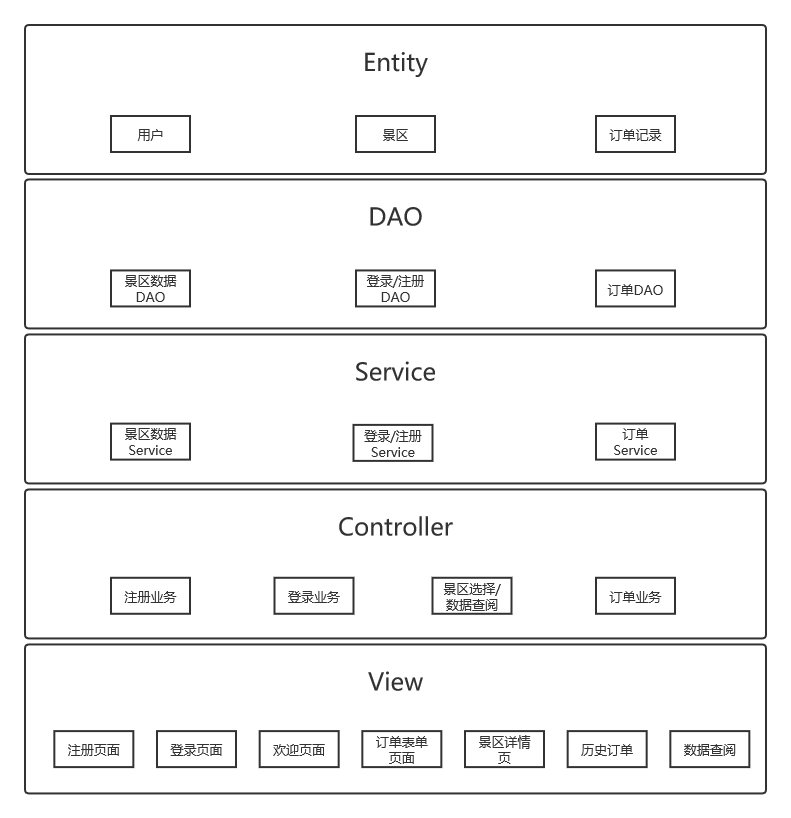
## 系统设计概述

基于需求分析和使用场景构想，疫情期间全市景区统一预约系统主要的业务功能模块划分为账户管理、订单管理、疫情防控3类，为游客和有关部门管理人员这两大用户群体提供便捷的集成化服务。用户通过账户管理实现登录、注册，同时系统根据用户类型（游客或有关部门人员）分配功能页面。游客可以查看自己的订单，也可以选择景点生成新订单并支付、提交。在提交新订单时，游客可以选择几位同行人员共享同一组防疫信息，为自己和同行人员一并预订行程；系统将为每人生成一条订单记录。系统会自动筛查游客上报的防疫信息，若“苏康码”不符合防疫要求则写入记录。系统通过统计景区人流量、游客上报的可疑症状等信息，向有关部门管理人员用户展示数据报、疫情风险分析和提示信息。系统的功能模块示意图如图所示。



## 详细设计概述

疫情期间全市景区统一预约系统采用MVC模式进行软件架构设计，分为Entity、DAO、Service、Controller、View五层，分别承担实体描述、数据库访问、数据库接口函数提供、业务处理和控制、前端用户界面的工作。详细的软件架构设计图如图。



如图所示，五层软件架构符合MVC设计模式，可以分离实体描述、数据库访问、数据函数接口、业务处理和控制、前端用户界面的功能，使得设计更有条理。具体分析如下：

（1）Entity层。本层以Java类的形式对疫情期间全市景区统一预约系统中涉及到的实体进行描述、封装，便于系统访问以实现相关功能。本层的Java类主要包括游客、工作人员、景区、订单信息几个方面。其属性大多数与数据库字段一致，描述这些实体在特定场景下的状态，通过public函数和getter、setter等方式提供对外接口，便于在更高层次中创建相应对象，实现业务处理和管理。

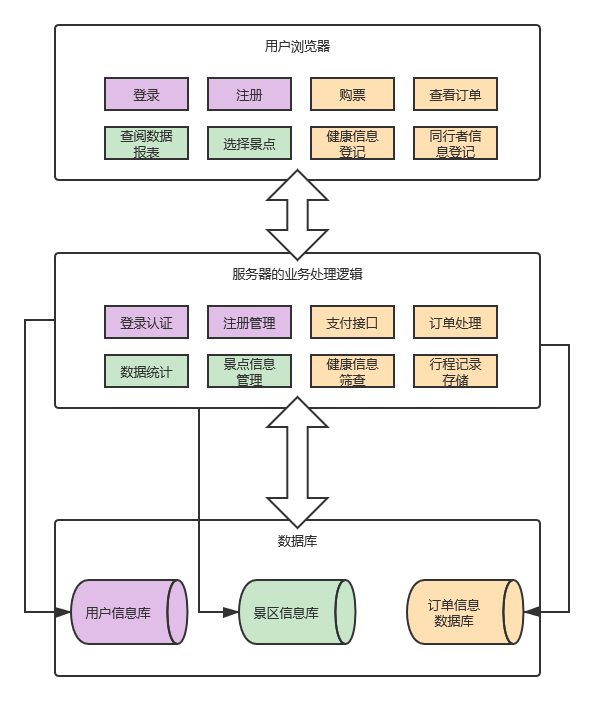
（2）DAO层。本层被用于实现对相应数据库的访问，实现对数据库的增加、删除、修改、查询操作。对应于相应功能的数据库，景区数据DAO实现订单中的景区信息读取、统计数据的读出和写入，登录/注册DAO实现登录信息验证以及注册信息写入，订单DAO记录每一笔订单的相关信息并且实现对订单中每一名游客的个人信息、健康信息的记载，并且用于提交新订单时对相关数据库的操作（以写入为主）。以Java类的形式实现，通过数据库连接工具访问数据库，通过public函数接口、getter和setter等方式允许Service层访问。

（3）Service层。本层以Java类的形式实现，通过访问DAO层的接口与数据库交互，并以函数的形式将这些交互操作封装，以数据访问函数接口的方式向Controller提供数据访问服务。

（4）Controller层。本层可以实现业务处理逻辑的相关功能，从相应的View层接收信息并进行计算、识别、判断，调取相应的Service实现功能，并控制View层页面的更新、跳转。疫情期间全市景区统一预约系统主要有注册、登录、景区选择/数据查阅、订单处理和显示等业务。以Servlet形式实现，通过访问Service提供的接口以获得数据，通过HTTP连接与View层的用户界面交互。

（5）View层。本层以网页（如JSP）实现用户界面，用户通过浏览器进行人机交互，网页通过HTTP连接与Controller层互动，受到Controller控制，并向Controller发送数据、指令。

疫情期间全市景区统一预约系统的用户通过浏览器以HTTP协议连接服务器进行注册、登录、选择景区、购票、健康信息登记、查看数据报表等业务，服务器提供业务接口、管理调度、数据存储等功能。系统的应用架构如图所示。



## 账户管理模块的详细设计

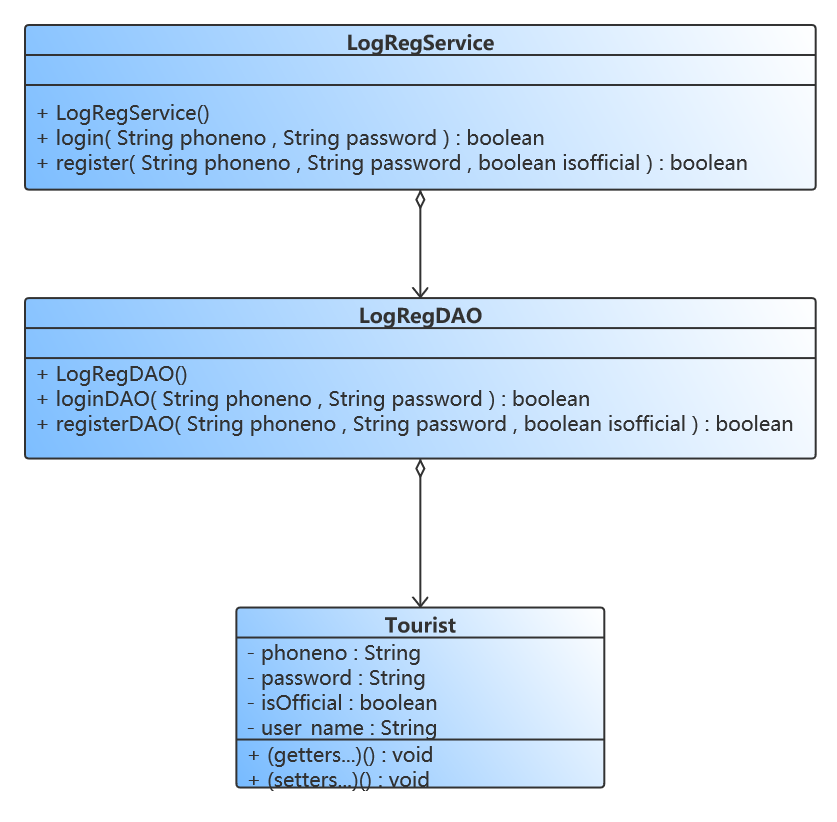
### 登录、注册功能设计

此功能为用户提供账户登录、注册服务。

此功能所需资源如下表所示。

|  |  |  |
| --- | --- | --- |
| MVC架构层 | 资源描述 | 资源名称 |
| View | 登录页面 | login.jsp |
| 注册页面 | register.jsp |
| Controller | 登录业务控制器 | LoginServlet.java |
| 注册业务控制器 | RegisterServlet.java |
| Service | 登录/注册数据服务 | LogRegService.java |
| DAO | 登录/注册DAO | LogRegDAO.java |
| Entity | 用户实体 | User.java |

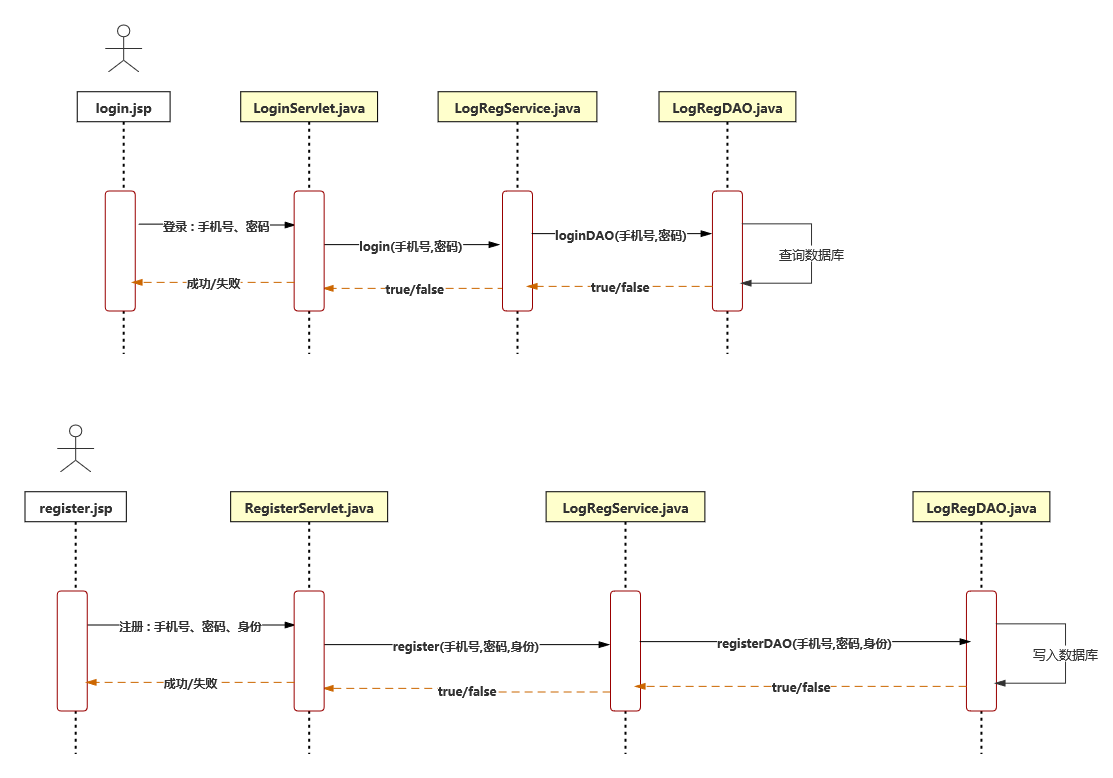
相关类的设计如下图所示。



实现此功能的业务逻辑过程如下图所示。

登录时，用户通过JSP页面填写表单，表单将数据发送到Servlet。Servlet将数据封装进入一个User类的对象，传给Service层并传入DAO层。DAO层取出用户手机号（数据库主键）查询数据库，将查询结果同样封装成User类对象传回Service层。Service层将Servlet发来的对象和DAO查到的对象进行比对，然后向Servlet返回“登录成功”信息或者相应的错误信息，如“用户名不存在”“密码错误”等。

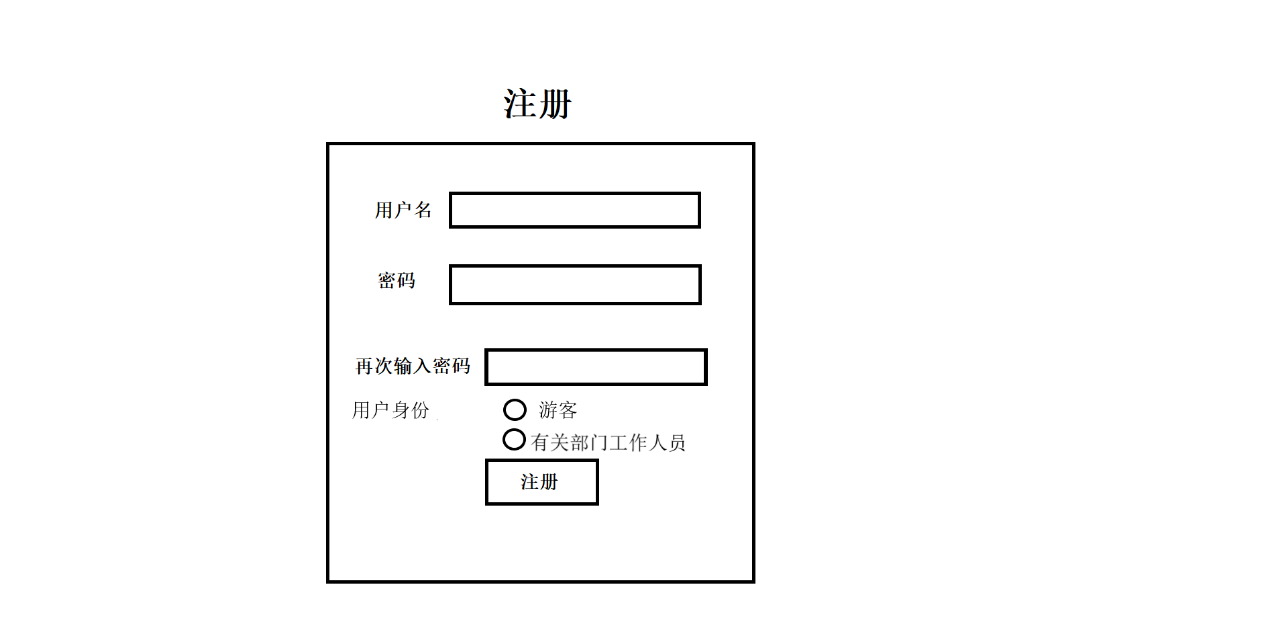
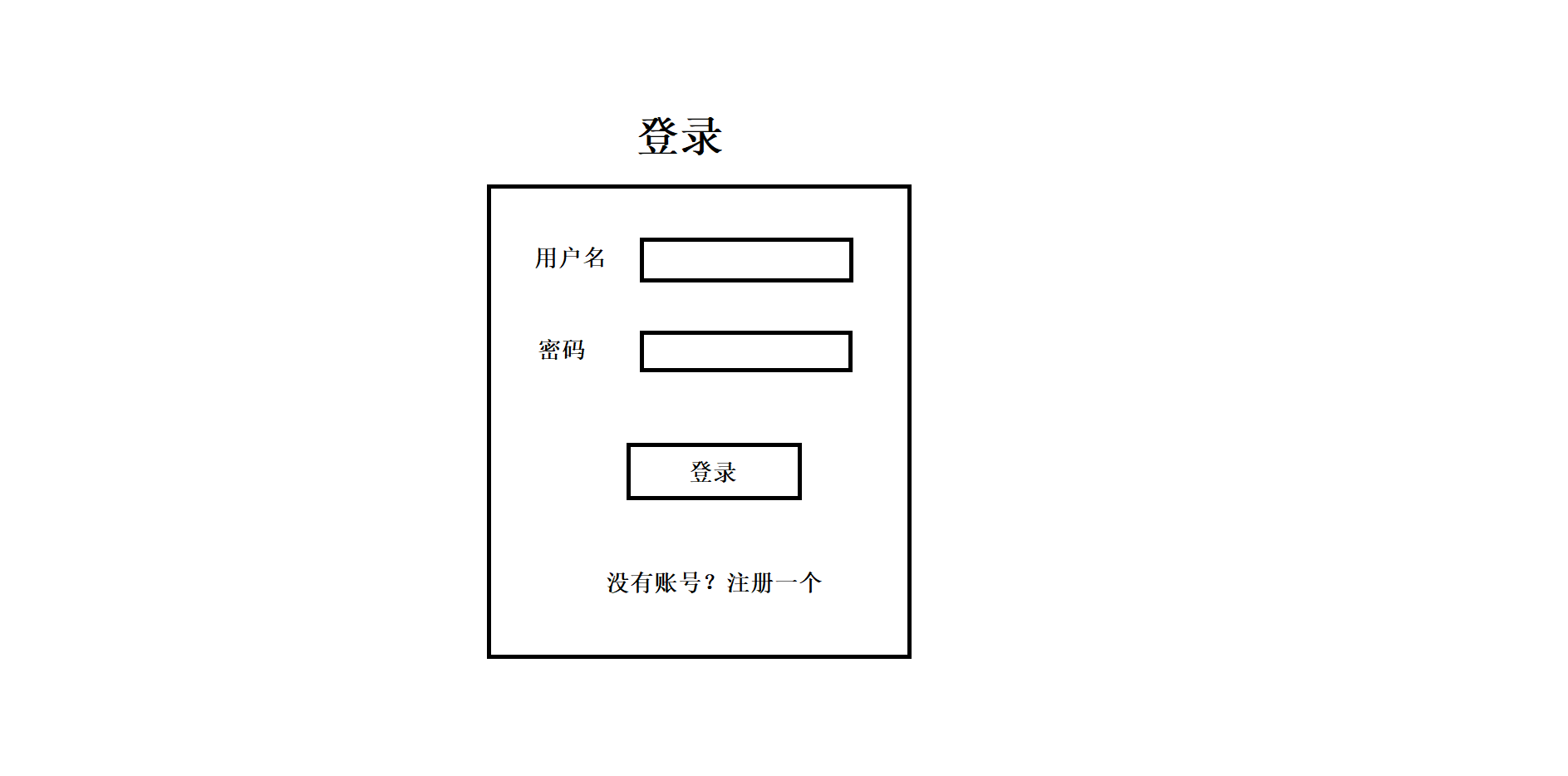
注册时，用户通过JSP页面填写表单，表单将数据发送到Servlet。Servlet首先检查确认密码是否正确，若不正确则通过setParameter方式退回到JSP，JSP接受Parameter后触发弹窗提示，重新注册。若正确，则将注册信息写入User类型对象，发送给Service层，Service调用DAO层执行写入语句，并返回结果。注册成功后弹窗提示，并跳转至登录页面。



#### View层

包含两个jsp组件，登录页面（login.jsp）和注册页面（register.jsp）

本功能所需用户界面的设计示意草图如下图所示。



#### Controller层

登录业务控制器LoginServelet.java



package controller;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.codec.digest.DigestUtils;

import entity.User;

import service.LogRegService;

public class LoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public LoginServlet() {

super();

}

public void destroy() {

super.destroy();

}

public String sha256(String origin) {

String hash="";

try {

hash = DigestUtils.sha256Hex(origin.getBytes("UTF-8"));

} catch (Exception e) {

e.printStackTrace();

}

return hash;

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=utf-8");

User u=new User();

try {

u.setPhoneno(request.getParameter("phone\_no"));

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

u.setPassword(sha256(originalPwd));

boolean isOfficial=false;

if(request.getParameter("is\_official").equals("true")) isOfficial=true;

u.setOfficial(isOfficial);

LogRegService svc=new LogRegService();

String loginRs = svc.login(u);

System.out.println(loginRs);

if(loginRs.equals("suc")) {

request.getSession().setAttribute("user\_info", svc.getUser(u));

if(u.isOfficial())

response.sendRedirect("admin.jsp");

else

response.sendRedirect("welcome.jsp");

}

else {

response.sendRedirect("login.jsp?loginfail="+loginRs);

}

}catch(Exception e) {

e.printStackTrace();

}

}

public void init() throws ServletException {

}

}

注册业务控制器RegisterServelet.java



package controller;

import entity.User;

import service.LogRegService;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.codec.digest.DigestUtils;

public class RegisterServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public RegisterServlet() {

super();

}

public void destroy() {

super.destroy();

}

public String sha256(String origin) {

String hash="";

try {

hash = DigestUtils.sha256Hex(origin.getBytes("UTF-8"));

} catch (Exception e) {

e.printStackTrace();

}

return hash;

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=utf-8");

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

String pwd\_cfm = request.getParameter("password\_confirm").toString();

if(! pwd.equals(pwd\_cfm)) {

response.sendRedirect("register.jsp?failure=psw\_cfm\_err");

return;

}

User u=new User();

try {

u.setPhoneno(request.getParameter("phone\_no"));

u.setUser\_name(request.getParameter("user\_name"));

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

u.setPassword(sha256(originalPwd));

boolean isOfficial = false;

if(request.getParameter("is\_official").equals("true"))

isOfficial = true;

u.setOfficial(isOfficial);

LogRegService svc=new LogRegService();

boolean regRslt = svc.register(u);

if(regRslt) {

response.sendRedirect("register.jsp?failure=no\_failure");

}

else {

response.sendRedirect("register.jsp?failure=reg\_err");

}

}catch(Exception e) {

e.printStackTrace();

}

}

public void init() throws ServletException {

}

}

#### Service层

登录/注册数据服务LogRegServelet.java



package service;

import entity.User;

import java.io.IOException;

import dao.LogRegDAO;

public class LogRegService {

public String login(User u) throws IOException {

System.out.print("Login service requested: ");

u.printInfo();

LogRegDAO dao = new LogRegDAO();

User rslt = dao.loginDAO(u);

if(rslt==null) return "phoneno\_err";

if(! rslt.getPassword().equals(u.getPassword()) ) return "password\_err";

if(rslt.isOfficial()!=u.isOfficial()) return "isofficial\_err";

return "suc";

}

public User getUser(User u) throws IOException {

LogRegDAO dao = new LogRegDAO();

User rslt = dao.loginDAO(u);

return rslt;

}

public boolean register(User u) throws IOException {

System.out.print("Register service requested: ");

u.printInfo();

LogRegDAO dao = new LogRegDAO();

return dao.registerDAO(u);

}

}

#### DAO层

登录/注册DAO LogRegDAO.java



package dao;

import java.io.IOException;

import java.io.InputStream;

import org.apache.ibatis.io.Resources;

import org.apache.ibatis.session.SqlSession;

import org.apache.ibatis.session.SqlSessionFactory;

import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import entity.User;

public class LogRegDAO {

public User loginDAO(User totest) throws IOException{

String resource = "SqlMapConfig.xml";

InputStream in = Resources.getResourceAsStream(resource);

SqlSessionFactory sessionFactory = new SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sessionFactory.openSession();

User rslt = null;

try {

rslt = sqlSession.selectOne("scd.loginDAO", totest);

sqlSession.close();

}catch(Exception e) {

e.printStackTrace();

}

if(rslt!=null) {

rslt.setPhoneno(totest.getPhoneno());

System.out.print("Login DAO: "); rslt.printInfo();

}

return rslt;

}

public boolean registerDAO(User u) throws IOException {

String resource = "SqlMapConfig.xml";

InputStream in = Resources.getResourceAsStream(resource);

SqlSessionFactory sessionFactory = new SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sessionFactory.openSession();

boolean rslt=false;

try {

sqlSession.insert("scd.registerDAO", u);

sqlSession.commit();

sqlSession.close();

}catch(Exception e) {

e.printStackTrace();

System.out.println("Register DAO: "+rslt);

return rslt;

}

rslt=true;

System.out.println("Register DAO: "+rslt);

return rslt;

}

public LogRegDAO(){

}

}

#### Entity层

用户实体User.java



package entity;

public class User {

private String phoneno;

private String user\_name;

private String password;

private boolean isOfficial;

public String getPhoneno() {

return phoneno;

}

public void setPhoneno(String phoneno) {

this.phoneno = phoneno;

}

public String getUser\_name() {

return user\_name;

}

public void setUser\_name(String user\_name) {

this.user\_name = user\_name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public boolean isOfficial() {

return isOfficial;

}

public void setOfficial(boolean isOfficial) {

this.isOfficial = isOfficial;

}

public void printInfo() {

System.out.println("User Entity");

System.out.println("Username: "+user\_name);

System.out.println("Phone No.: "+phoneno);

System.out.println("Password: "+password);

System.out.println("Is official? "+isOfficial);

}

}

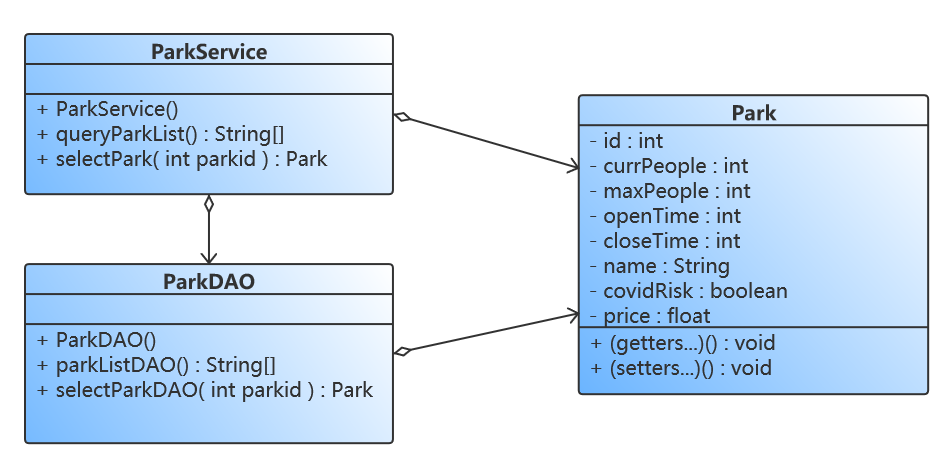
### 用户身份识别功能设计

若登录成功，景区数据Controller将识别用户身份类型，为游客和管理员提供不同功能的服务。游客可进行订单业务操作，有关部门管理员可进行数据查阅等操作。

此功能所需资源如下表所示。

|  |  |  |
| --- | --- | --- |
| **MVC架构层** | **资源描述** | **资源名称** |
| View | 景点预览页面 | welcome.jsp |
| 景点详情页面 | detail.jsp |
| 景区数据大屏 | admin.jsp |
| Controller | 景区业务控制器 | ParkServlet.java |
| Service | 景区数据服务 | ParkService.java |
| DAO | 景区数据DAO | ParkDAO.java |
| Entity | 景区实体 | Park.java |

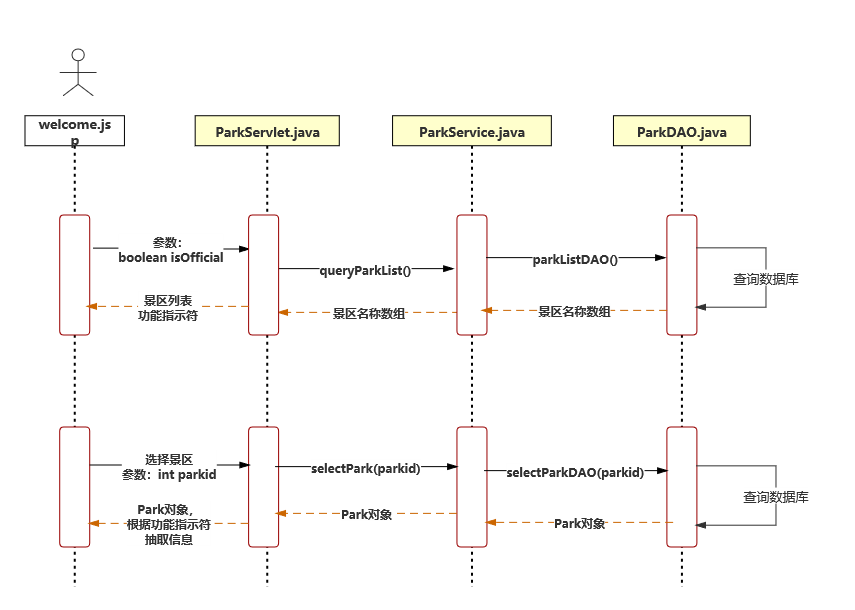
相关类的设计如下图所示。



实现此功能的业务逻辑过程如下图所示。

登录成功后，Servlet通过request.getSession().setAttribute()方法，向当前HTTP会话设置用户属性，属性值即为当前用户的User类型的对象。根据当前用户的isOfficial属性判断当前用户的身份类型。若为游客用户，跳转至景区预览页面；若为管理员用户，跳转至景区数据大屏页面。

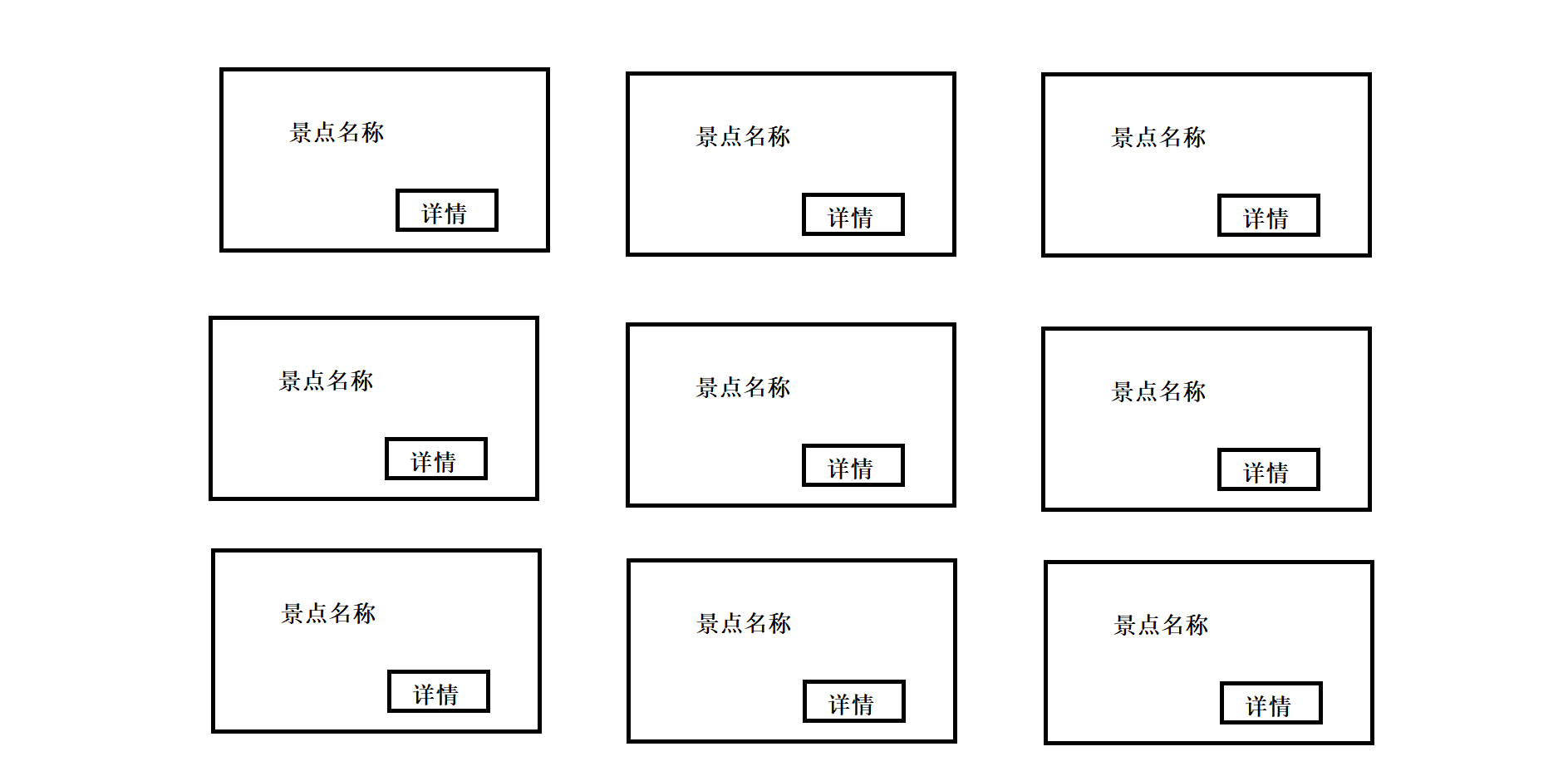
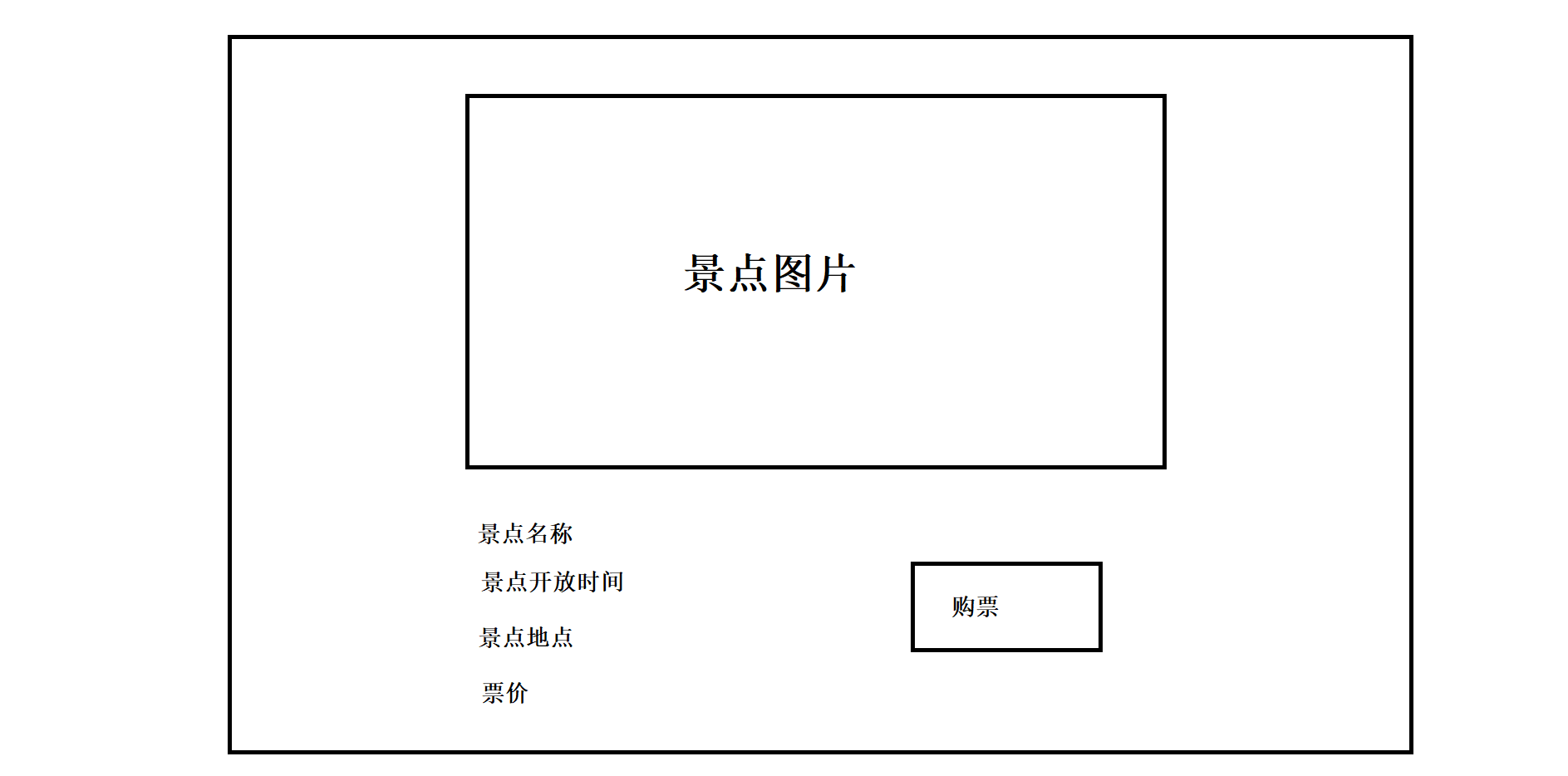
游客在景区预览页面可以看到所有景区的简要信息，包括名称、图片、开放时间、票价、三天内的余票量。每个景区以卡片的形式展示，并配有查看详情按钮。游客可选择任意景区，查看详情按钮将会带着景区编号请求Servlet，调用Service、DAO查询该景区的所有信息，封装进Park对象返回。与之前类似，Servlet将会通过request.getSession().setAttribute()方法，向当前HTTP会话设置景区属性，属性值即为当前选择的景区的Park类型对象，随后跳转至景区详情页detail.jsp。景区详情页从session的属性集中接收这个Park对象，取出其所有详细信息展示在页面上。



#### View层

包含三个jsp组件，景点预览页面welcome.jsp、景点详情页面detail.jsp、景点数据大屏admin.jsp。

本功能所需用户界面的设计示意草图如下图所示。

#### Controller层

景区业务控制器ParkServelet.java



package controller;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import entity.Park;

import service.ParkService;

public class ParkServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public ParkServlet() {

super();

}

public void destroy() {

super.destroy();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

int park\_id = Integer.parseInt(request.getParameter("park\_id"));

Park pk = new Park();

ParkService svc=new ParkService();

pk=svc.selectOnePark(park\_id);

request.getSession().setAttribute("park\_selected", pk);

response.sendRedirect("detail.jsp");

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

public void init() throws ServletException {

}

}

#### Service层

景区数据服务ParkService.java



package service;

import java.util.ArrayList;

import dao.ParkDAO;

import entity.Park;

public class ParkService {

public ArrayList<Park> getParkSet(){

ParkDAO dao = new ParkDAO();

ArrayList<Park> rslt = dao.parkSetDAO();

return rslt;

}

public Park selectOnePark(int id) {

ParkDAO dao = new ParkDAO();

return dao.oneParkDAO(id);

}

public boolean parkReserved(int parkid, int num, int datediff) {

ParkDAO dao = new ParkDAO();

return dao.parkReservedDAO(parkid, num, datediff);

}

public boolean declareCovidRisk(int pkid) {

ParkDAO dao = new ParkDAO();

return dao.covidRiskDAO(pkid);

}

}

#### DAO层

景区数据DAO ParkDAO.java



package dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import entity.Park;

public class ParkDAO {

static final String JDBC\_DRIVER="org.postgresql.Driver";

static final String DB\_URL="jdbc:postgresql://127.0.0.1:5432/software\_course\_design";

static final String USER="chenrz";

static final String PASS="jsnj130601\*";

public ArrayList<Park> parkSetDAO(){

try{Class.forName(JDBC\_DRIVER);}catch(ClassNotFoundException e){e.printStackTrace();}

ArrayList<Park> rslt=new ArrayList<Park>();

try{

Connection conn=DriverManager.getConnection(DB\_URL,USER,PASS);

Statement statement=conn.createStatement();

String sql="SELECT \* FROM parks ORDER BY park\_id ASC;";

ResultSet rs=statement.executeQuery(sql);

while(rs.next()){

Park tmp=new Park();

tmp.setPark\_id(rs.getInt("park\_id"));

tmp.setPark\_name(rs.getString("park\_name"));

tmp.setPrice(rs.getFloat("price"));

tmp.setOpen\_time(rs.getInt("open\_time"));

tmp.setClose\_time(rs.getInt("close\_time"));

tmp.setPark\_address(rs.getString("park\_address"));

tmp.setContact(rs.getString("contact"));

tmp.setMax\_num(rs.getInt("max\_num"));

tmp.setVisitor\_0(rs.getInt("visitor\_0"));

tmp.setVisitor\_1(rs.getInt("visitor\_1"));

tmp.setVisitor\_2(rs.getInt("visitor\_2"));

tmp.setCovid\_risk(rs.getBoolean("covid\_risk"));

tmp.setDescription(rs.getString("description"));

rslt.add(tmp);

}

rs.close();

statement.close();

conn.close();

}

catch(SQLException e){e.printStackTrace();}

return rslt;

}

public Park oneParkDAO(int id) {

try{Class.forName(JDBC\_DRIVER);}catch(ClassNotFoundException e){e.printStackTrace();}

Park tmp=new Park();

try{

Connection conn=DriverManager.getConnection(DB\_URL,USER,PASS);

Statement statement=conn.createStatement();

String sql="SELECT \* FROM parks WHERE park\_id="+id+";";

ResultSet rs=statement.executeQuery(sql);

while(rs.next()){

tmp.setPark\_id(rs.getInt("park\_id"));

tmp.setPark\_name(rs.getString("park\_name"));

tmp.setPrice(rs.getFloat("price"));

tmp.setOpen\_time(rs.getInt("open\_time"));

tmp.setClose\_time(rs.getInt("close\_time"));

tmp.setPark\_address(rs.getString("park\_address"));

tmp.setContact(rs.getString("contact"));

tmp.setMax\_num(rs.getInt("max\_num"));

tmp.setVisitor\_0(rs.getInt("visitor\_0"));

tmp.setVisitor\_1(rs.getInt("visitor\_1"));

tmp.setVisitor\_2(rs.getInt("visitor\_2"));

tmp.setCovid\_risk(rs.getBoolean("covid\_risk"));

tmp.setDescription(rs.getString("description"));

}

rs.close();

statement.close();

conn.close();

}

catch(SQLException e){e.printStackTrace();}

return tmp;

}

public boolean parkReservedDAO(int parkid, int num, int datediff) {

try{Class.forName(JDBC\_DRIVER);}catch(ClassNotFoundException e){e.printStackTrace();}

int max\_num=0, v=0;

int isSuc = 0;

try{

Connection conn=DriverManager.getConnection(DB\_URL,USER,PASS);

Statement statement=conn.createStatement();

String sql="SELECT max\_num,visitor\_"+datediff+" FROM parks WHERE park\_id="+parkid+";";

ResultSet rs=statement.executeQuery(sql);

while(rs.next()){

max\_num = rs.getInt("max\_num");

v = rs.getInt("visitor\_"+datediff);

}

rs.close();

statement.close();

conn.close();

}

catch(SQLException e){e.printStackTrace();}

if(v+num > max\_num) {

return false;

}else {

try{

Connection conn=DriverManager.getConnection(DB\_URL,USER,PASS);

Statement statement=conn.createStatement();

String sql="UPDATE parks SET visitor\_"+datediff+" ="+(v+num)+" where park\_id="+parkid+";";

isSuc = statement.executeUpdate(sql);

statement.close();

conn.close();

}

catch(SQLException e){e.printStackTrace();}

}

if(isSuc == 0) return false;

else return true;

}

public boolean covidRiskDAO(int pkid) {

try{Class.forName(JDBC\_DRIVER);}catch(ClassNotFoundException e){e.printStackTrace();}

int isSuc = 0;

try{

Connection conn=DriverManager.getConnection(DB\_URL,USER,PASS);

Statement statement=conn.createStatement();

String sql="UPDATE parks SET covid\_risk=true where park\_id="+pkid+";";

isSuc = statement.executeUpdate(sql);

statement.close();

conn.close();

}

catch(SQLException e){e.printStackTrace();}

if(isSuc == 0) return false;

else return true;

}

}

#### Entity层

景区实体Park.java



package entity;

public class Park {

private int park\_id;

private String park\_name;

private float price;

private int open\_time;

private int close\_time;

private String park\_address;

private String contact;

private int max\_num;

private int visitor\_0;

private int visitor\_1;

private int visitor\_2;

private boolean covid\_risk;

private String description;

public int getPark\_id() {

return park\_id;

}

public void setPark\_id(int park\_id) {

this.park\_id = park\_id;

}

public String getPark\_name() {

return park\_name;

}

public void setPark\_name(String park\_name) {

this.park\_name = park\_name;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public int getOpen\_time() {

return open\_time;

}

public void setOpen\_time(int open\_time) {

this.open\_time = open\_time;

}

public int getClose\_time() {

return close\_time;

}

public void setClose\_time(int close\_time) {

this.close\_time = close\_time;

}

public String getPark\_address() {

return park\_address;

}

public void setPark\_address(String park\_address) {

this.park\_address = park\_address;

}

public String getContact() {

return contact;

}

public void setContact(String contact) {

this.contact = contact;

}

public int getMax\_num() {

return max\_num;

}

public void setMax\_num(int max\_num) {

this.max\_num = max\_num;

}

public int getVisitor\_0() {

return visitor\_0;

}

public void setVisitor\_0(int visitor\_0) {

this.visitor\_0 = visitor\_0;

}

public int getVisitor\_1() {

return visitor\_1;

}

public void setVisitor\_1(int visitor\_1) {

this.visitor\_1 = visitor\_1;

}

public int getVisitor\_2() {

return visitor\_2;

}

public void setVisitor\_2(int visitor\_2) {

this.visitor\_2 = visitor\_2;

}

public boolean isCovid\_risk() {

return covid\_risk;

}

public void setCovid\_risk(boolean covid\_risk) {

this.covid\_risk = covid\_risk;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

}

## 订单管理模块的详细设计

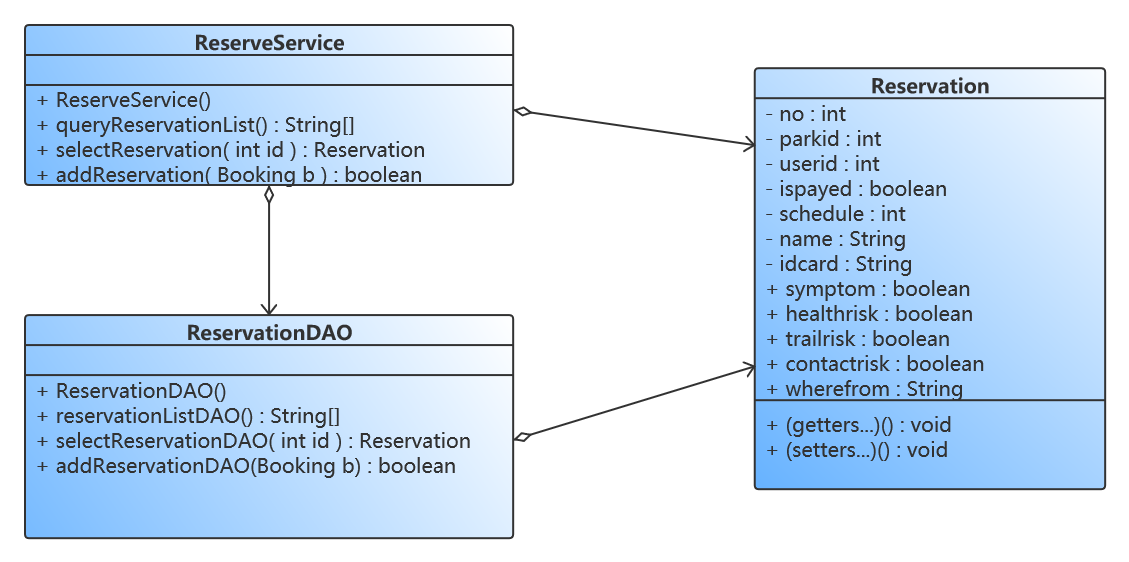
### 行程预订、防疫信息上报功能设计

游客可通过选择景点生成订单，为自己和同行人员预订行程，并上报防疫信息。

此功能所需资源如下表所示。

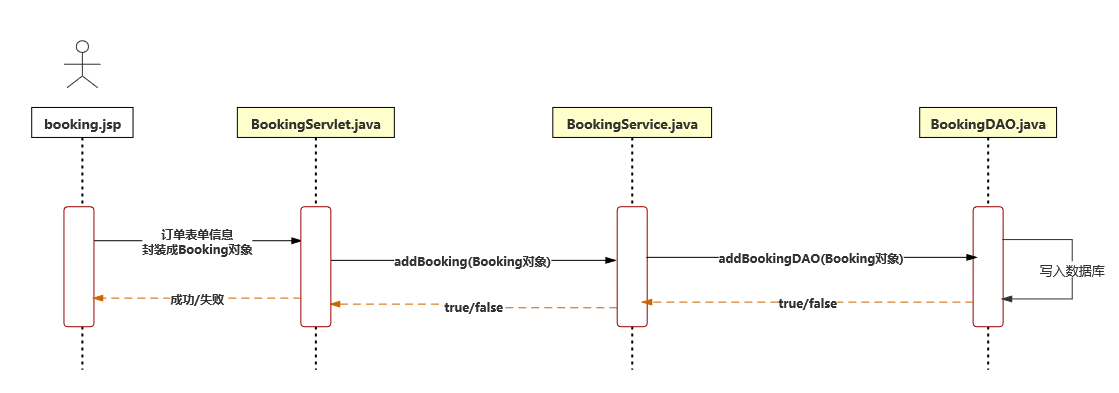
|  |  |  |
| --- | --- | --- |
| **MVC架构层** | **资源描述** | **资源名称** |
| View | 购票订单页面 | reserve.jsp |
| Controller | 订单业务控制器 | ReserveServlet.java |
| Service | 订单数据服务 | ReserveService.java |
| DAO | 订单数据DAO | ReservationDAO.java |
| Entity | 订单实体 | Reservation.java |

相关类的设计如下图所示。



实现此功能的业务逻辑过程如下图所示。

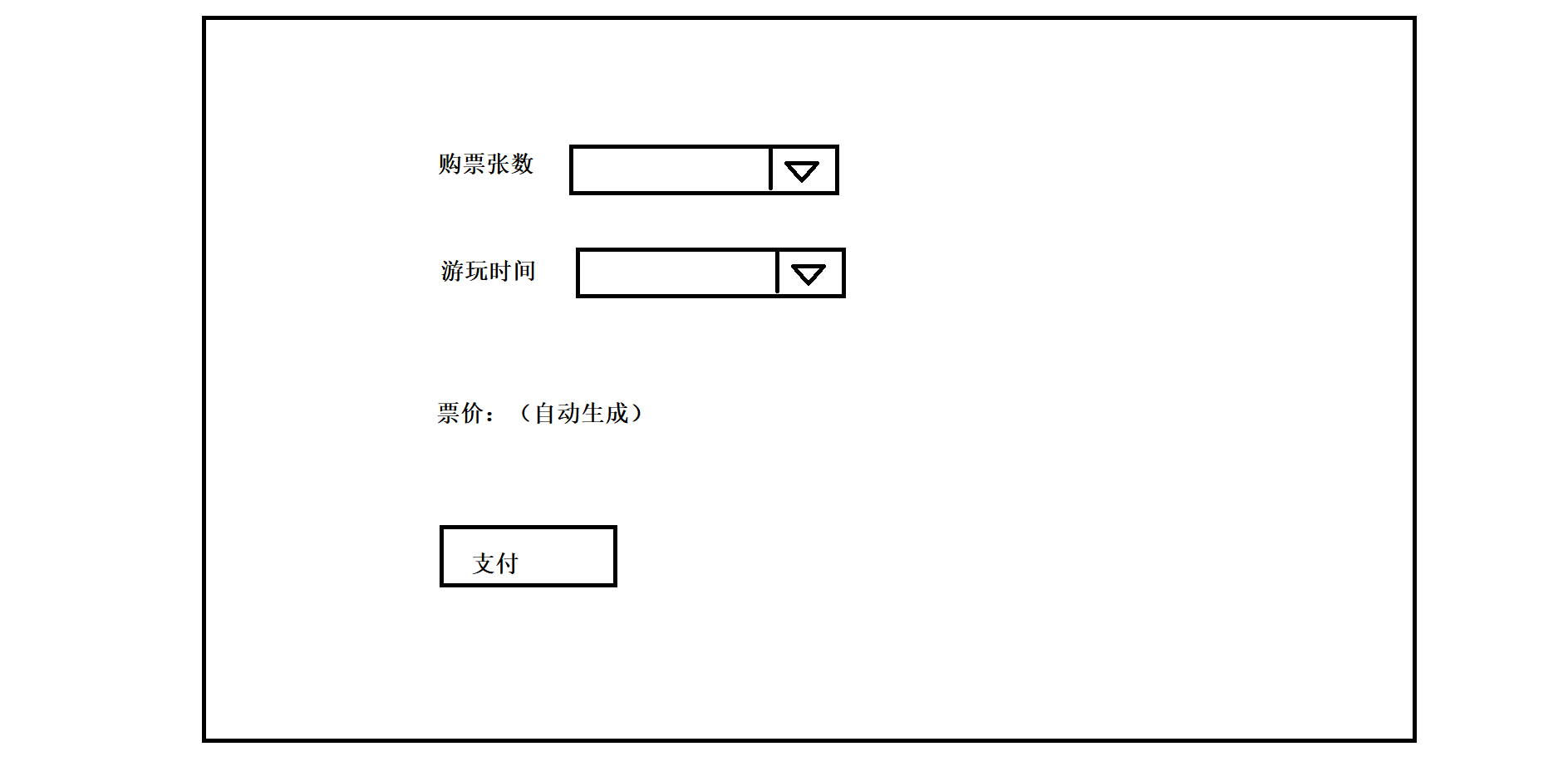
游客从景区详情页面进入订单页面对当前景区进行门票预订和防疫信息登记。当前景区的信息、预订用户的信息均可从session对象的attribute中提取，并与表单信息一并发送给Servlet。Servlet将其封装进Reservation类型的对象中，然后传给Service调用DAO层进行数据库写入操作。写入成功，则返回至Servlet并跳转至当前用户的订单记录页面。

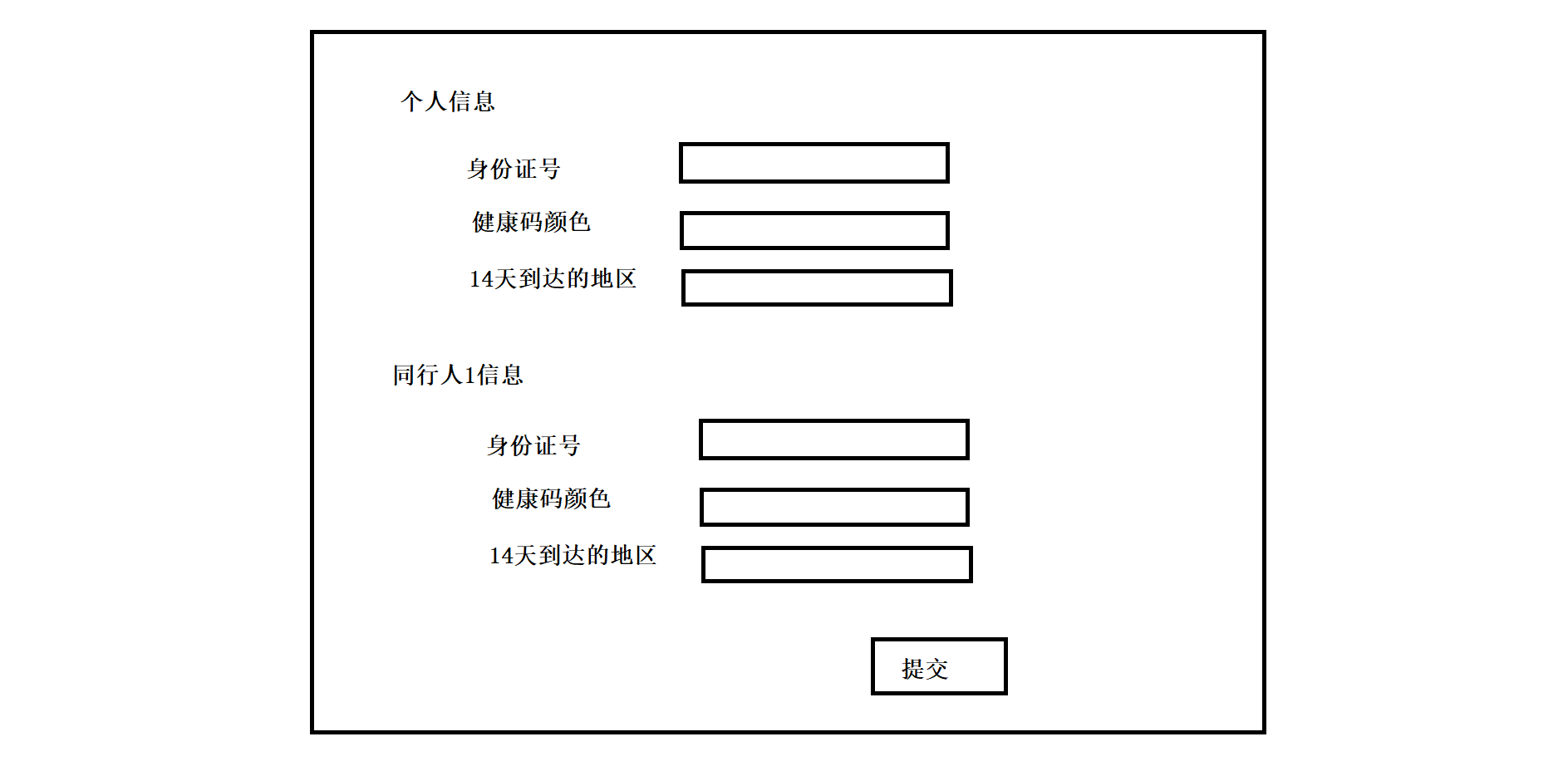


#### View层

包含一个jsp组件，购票订单页面reserve.jsp。

本功能所需用户界面的设计示意草图如下图所示。





#### Controller层

订单业务控制器ReserveServelet.java



package controller;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.GregorianCalendar;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.spec.SecretKeySpec;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.codec.DecoderException;

import org.apache.commons.codec.binary.Hex;

import org.apache.commons.codec.digest.DigestUtils;

import entity.Park;

import entity.Reservation;

import entity.User;

import service.ParkService;

import service.ReserveService;

public class ReserveServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public ReserveServlet() {

super();

}

public void destroy() {

super.destroy();

}

public String md5(String origin) {

String hash = null;

try {

hash = DigestUtils.md5Hex(origin.getBytes("UTF-8"));

} catch (Exception e) {

e.printStackTrace();

}

return hash;

}

public String aes128\_encrypt(String src, String key128) throws UnsupportedEncodingException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException {

SecretKeySpec keySpec = new SecretKeySpec(key128.getBytes("UTF-8"),"AES");

Cipher cphr = Cipher.getInstance("AES");

cphr.init(Cipher.ENCRYPT\_MODE, keySpec);

byte[] encrypted = cphr.doFinal(src.getBytes("UTF-8"));

return Hex.encodeHexString(encrypted);

}

public String aes128\_decrypt(String src, String key128) throws UnsupportedEncodingException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, DecoderException {

SecretKeySpec keySpec = new SecretKeySpec(key128.getBytes("UTF-8"),"AES");

Cipher cphr = Cipher.getInstance("AES");

cphr.init(Cipher.DECRYPT\_MODE, keySpec);

byte[] original = cphr.doFinal(Hex.decodeHex(src));

String oriStr = new String(original,"UTF-8");

return oriStr;

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

Park pk = new Park();

if(request.getSession().getAttribute("park\_selected")!=null){

pk = (Park)request.getSession().getAttribute("park\_selected");

}

User usr = new User();

if(request.getSession().getAttribute("user\_info")!=null){

usr = (User)request.getSession().getAttribute("user\_info");

}

String userKey = usr.getPassword().substring(0, 32);

//同行人数量

int member\_num = Integer.parseInt(request.getParameter("member\_num"));

//预订日期

int dateDiff = Integer.parseInt(request.getParameter("schedule\_date"));

ParkService pksvc = new ParkService();

boolean checkNum = pksvc.parkReserved(pk.getPark\_id(), member\_num+1, dateDiff);

if(checkNum == false) {

response.sendRedirect("reserve.jsp?rsv\_err=too\_many\_people");

return;

}

//用户填写的手机号

String tourist\_phoneno = request.getParameter("tourist\_phoneno");

//入园时间

String enterTime = " " + request.getParameter("schedule\_time") + "时";

//订单时间戳

Date nowdate=new Date();

SimpleDateFormat datef=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String time\_stamp = datef.format(nowdate);

//入园日期

Calendar calendar=new GregorianCalendar();

calendar.setTime(nowdate);

calendar.add(Calendar.DATE, dateDiff);

SimpleDateFormat dateG=new SimpleDateFormat("yyyy-MM-dd");

String enterDate = dateG.format(calendar.getTime());

//入园日期和时间

String schedule = enterDate + enterTime;

for(int i=0;i<=member\_num;i++) {

Reservation r = new Reservation();

ReserveService rssv = new ReserveService();

int thisReserveID = rssv.getNextIdno();

String id\_cipher = null;

try {

id\_cipher = aes128\_encrypt(String.valueOf(thisReserveID), md5("HippoY315"));

} catch (Exception e) {

e.printStackTrace();

}

r.setId\_cipher(id\_cipher);

r.setUser\_phoneno(usr.getPhoneno());

r.setPark\_selected(pk.getPark\_name());

r.setReserve\_id(thisReserveID);

r.setMember\_num(member\_num);

r.setTourist\_phoneno(tourist\_phoneno);

r.setSchedule(schedule);

boolean isHealthy = false;

if(request.getParameter("is\_healthy\_"+i).equals("true")) isHealthy = true;

r.setHealthy(isHealthy);

if(isHealthy == false) {

boolean declareCovidRiskResult = pksvc.declareCovidRisk(pk.getPark\_id());

if(! declareCovidRiskResult) System.out.println("▲ 出现疫情风险，写入数据库失败！");

else System.out.println("▲ 出现疫情风险！已记录");

}

try {

String name\_cipher = aes128\_encrypt(request.getParameter("tourist\_name\_"+i).toString(), userKey);

r.setTourist\_name(name\_cipher);

String idcard\_cipher = aes128\_encrypt(request.getParameter("tourist\_idcard\_"+i).toString(), userKey);

r.setTourist\_idcard(idcard\_cipher);

} catch (Exception e) {

e.printStackTrace();

}

r.setWhere\_from(request.getParameter("where\_from\_"+i).toString());

r.setTime\_stamp(time\_stamp);

r.setPayed(true); //TODO Payment

rssv.newReservation(r);

}

response.sendRedirect("tickets.jsp");

}

public void init() throws ServletException {

}

}

#### Service层

订单数据服务ReserveService.java



package service;

import java.util.ArrayList;

import dao.ReservationDAO;

import entity.Reservation;

public class ReserveService {

public boolean newReservation(Reservation r) {

ReservationDAO rdao=new ReservationDAO();

return rdao.reserveDAO(r);

}

public int getNextIdno() {

ReservationDAO rdao=new ReservationDAO();

return rdao.nextIdnoDAO();

}

public ArrayList<Reservation> getReservationSet(String userphoneno){

ReservationDAO rdao=new ReservationDAO();

return rdao.resvSetDAO(userphoneno);

}

}

#### DAO层

订单数据DAO ReservationDAO.java



package entity;

public class Reservation {

private int reserve\_id;

private String user\_phoneno;

private String park\_selected;

private String schedule;

private String tourist\_phoneno;

private int member\_num;

private String tourist\_name;

private String tourist\_idcard;

private String where\_from;

private boolean isHealthy;

private boolean isPayed;

private String time\_stamp;

private String id\_cipher;

public int getReserve\_id() {

return reserve\_id;

}

public void setReserve\_id(int reserve\_id) {

this.reserve\_id = reserve\_id;

}

public String getUser\_phoneno() {

return user\_phoneno;

}

public void setUser\_phoneno(String user\_phoneno) {

this.user\_phoneno = user\_phoneno;

}

public String getPark\_selected() {

return park\_selected;

}

public void setPark\_selected(String park\_selected) {

this.park\_selected = park\_selected;

}

public String getSchedule() {

return schedule;

}

public void setSchedule(String schedule) {

this.schedule = schedule;

}

public String getTourist\_phoneno() {

return tourist\_phoneno;

}

public void setTourist\_phoneno(String tourist\_phoneno) {

this.tourist\_phoneno = tourist\_phoneno;

}

public int getMember\_num() {

return member\_num;

}

public void setMember\_num(int member\_num) {

this.member\_num = member\_num;

}

public String getTourist\_name() {

return tourist\_name;

}

public void setTourist\_name(String tourist\_name) {

this.tourist\_name = tourist\_name;

}

public String getTourist\_idcard() {

return tourist\_idcard;

}

public void setTourist\_idcard(String tourist\_idcard) {

this.tourist\_idcard = tourist\_idcard;

}

public String getWhere\_from() {

return where\_from;

}

public void setWhere\_from(String where\_from) {

this.where\_from = where\_from;

}

public boolean isHealthy() {

return isHealthy;

}

public void setHealthy(boolean isHealthy) {

this.isHealthy = isHealthy;

}

public boolean isPayed() {

return isPayed;

}

public void setPayed(boolean isPayed) {

this.isPayed = isPayed;

}

public String getTime\_stamp() {

return time\_stamp;

}

public void setTime\_stamp(String time\_stamp) {

this.time\_stamp = time\_stamp;

}

public String getId\_cipher() {

return id\_cipher;

}

public void setId\_cipher(String id\_cipher) {

this.id\_cipher = id\_cipher;

}

}

#### Entity层

订单实体Reservation.java



package entity;

public class Reservation {

private int reserve\_id;

private String user\_phoneno;

private String park\_selected;

private String schedule;

private String tourist\_phoneno;

private int member\_num;

private String tourist\_name;

private String tourist\_idcard;

private String where\_from;

private boolean isHealthy;

private boolean isPayed;

private String time\_stamp;

private String id\_cipher;

public int getReserve\_id() {

return reserve\_id;

}

public void setReserve\_id(int reserve\_id) {

this.reserve\_id = reserve\_id;

}

public String getUser\_phoneno() {

return user\_phoneno;

}

public void setUser\_phoneno(String user\_phoneno) {

this.user\_phoneno = user\_phoneno;

}

public String getPark\_selected() {

return park\_selected;

}

public void setPark\_selected(String park\_selected) {

this.park\_selected = park\_selected;

}

public String getSchedule() {

return schedule;

}

public void setSchedule(String schedule) {

this.schedule = schedule;

}

public String getTourist\_phoneno() {

return tourist\_phoneno;

}

public void setTourist\_phoneno(String tourist\_phoneno) {

this.tourist\_phoneno = tourist\_phoneno;

}

public int getMember\_num() {

return member\_num;

}

public void setMember\_num(int member\_num) {

this.member\_num = member\_num;

}

public String getTourist\_name() {

return tourist\_name;

}

public void setTourist\_name(String tourist\_name) {

this.tourist\_name = tourist\_name;

}

public String getTourist\_idcard() {

return tourist\_idcard;

}

public void setTourist\_idcard(String tourist\_idcard) {

this.tourist\_idcard = tourist\_idcard;

}

public String getWhere\_from() {

return where\_from;

}

public void setWhere\_from(String where\_from) {

this.where\_from = where\_from;

}

public boolean isHealthy() {

return isHealthy;

}

public void setHealthy(boolean isHealthy) {

this.isHealthy = isHealthy;

}

public boolean isPayed() {

return isPayed;

}

public void setPayed(boolean isPayed) {

this.isPayed = isPayed;

}

public String getTime\_stamp() {

return time\_stamp;

}

public void setTime\_stamp(String time\_stamp) {

this.time\_stamp = time\_stamp;

}

public String getId\_cipher() {

return id\_cipher;

}

public void setId\_cipher(String id\_cipher) {

this.id\_cipher = id\_cipher;

}

}

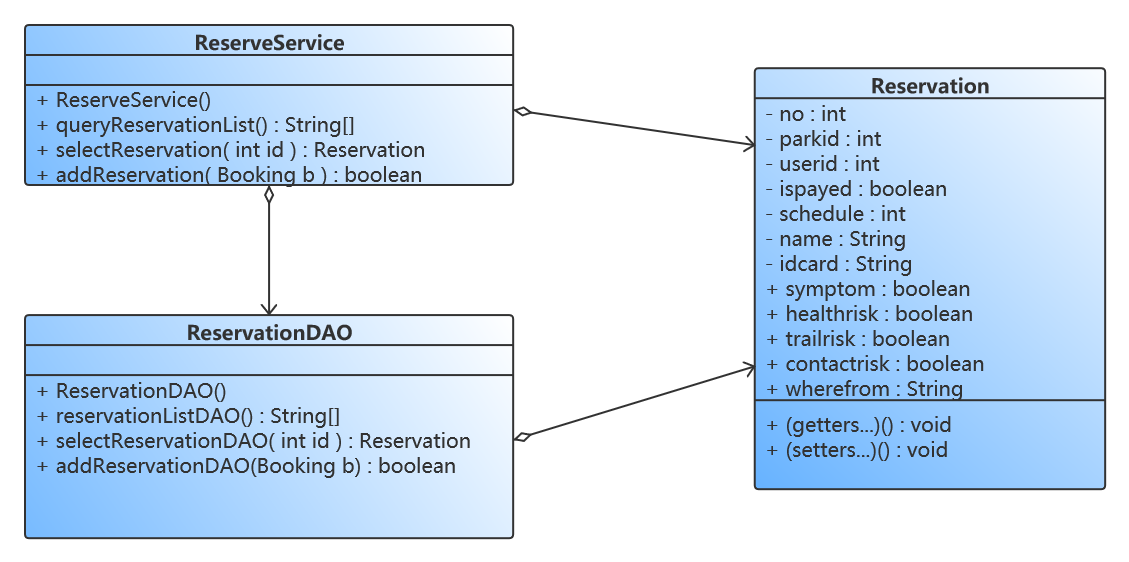
### 历史订单功能设计

游客可通过选择景点查看自己的历史订单信息。

此功能所需资源如下表所示。

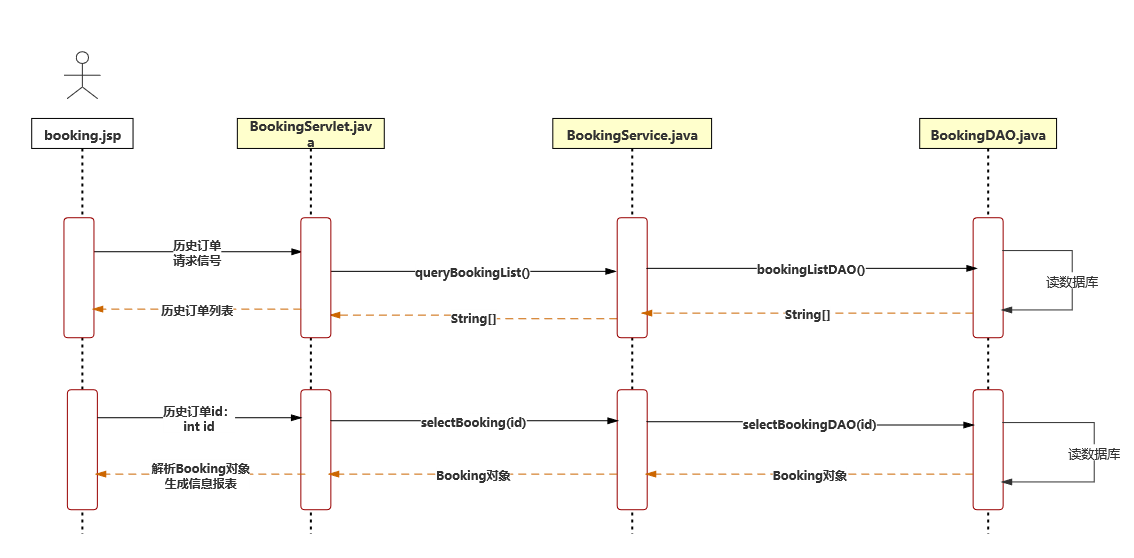
|  |  |  |
| --- | --- | --- |
| **MVC架构层** | **资源描述** | **资源名称** |
| View | 订单记录页面 | tickets.jsp |
| Service | 订单数据服务 | ReserveService.java |
| DAO | 订单数据DAO | ReservationDAO.java |
| Entity | 订单实体 | Reservation.java |

相关类的设计如下图所示。



实现此功能的业务逻辑过程如下图所示。

用户打开订单记录页面后，JSP页面抽取session对象中存储的当前用户的手机号码请求Service层的ReserveService.java调用DAO，返回ArrayList<Reservation>类型的对象，记录了当前用户的所有订单，通过JSP表达式的形式显示在当前页面上。



#### View层

包含一个jsp组件，订单记录页面ticket.jsp。

#### Service层

订单数据服务ReserveService.java（同上）

#### DAO层

订单数据DAO ReservationDAO.java（同上）

#### Entity层

订单实体Reservation.java（同上）

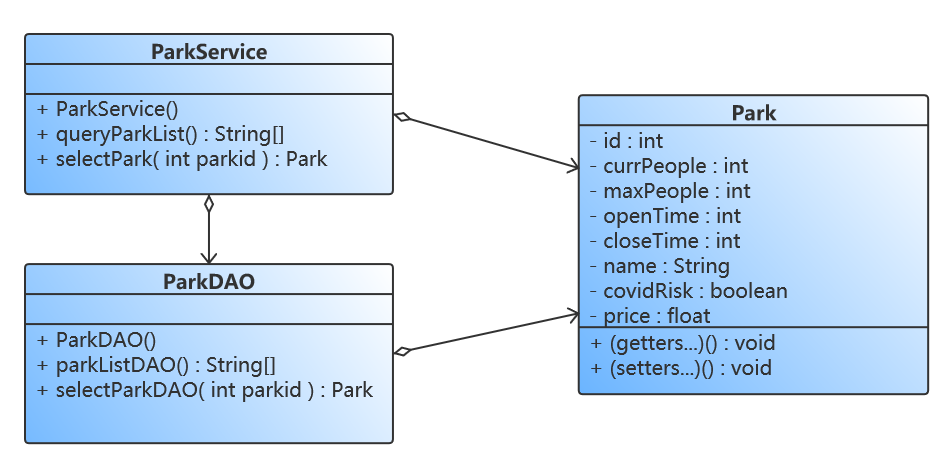
## 疫情防控模块的详细设计

本功能基于3.2.2节（二）中用户身份的识别，对有关部门管理员角色细化了景点信息呈现，突出了本系统疫情防控功能的鲜明特色。

此功能所需资源如下表所示。

|  |  |  |
| --- | --- | --- |
| **MVC架构层** | **资源描述** | **资源名称** |
| View | 景区数据大屏 | admin.jsp |
| Service | 景区数据服务 | ParkService.java |
| DAO | 景区数据DAO | ParkDAO.java |
| Entity | 景区实体 | Park.java |

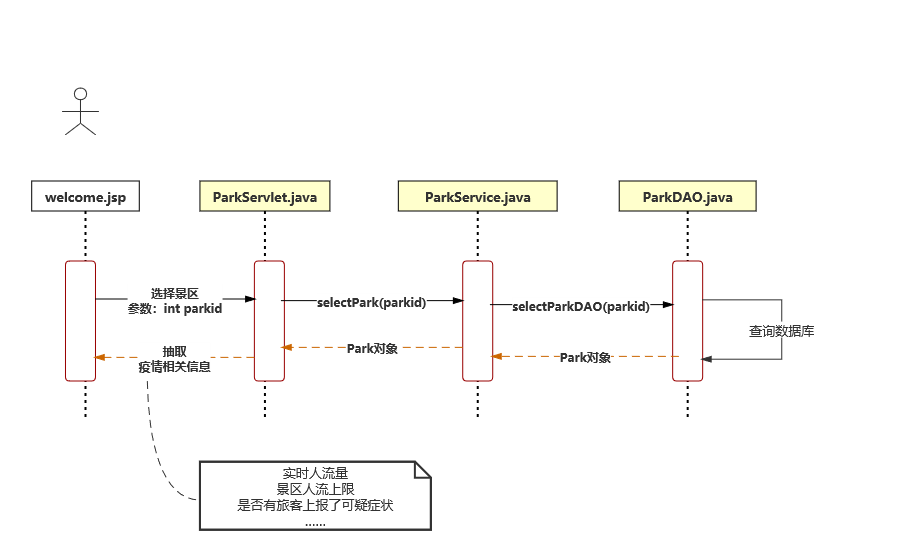
相关类的设计如下图所示。



实现此功能的业务逻辑过程如下图所示。

管理员在景区数据大屏页面可以看到实时人流量。通过jsp表达式计算实时人流量占限流量的比值，以rgba()（黄色）函数显示成表格底色，黄色越深表示越拥挤，人流量相对小的景区底色近乎白色。

同时，若有“苏康码”不是绿码的游客进入了景区，则该景区的covid\_risk属性变为true，数据大屏页面上相应景区的底色将变为红色。此功能可以有效帮助疫情防控。



#### View层

包含一个jsp组件，景区数据大屏admin.jsp。

#### Service层

景区数据服务ParkService.java（同上）

#### DAO层

景区数据DAO ParkDAO.java（同上）

#### Entity层

景区实体Park.java（同上）

## 配置文件

### 项目配置文件pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>SoftwareCourseDesign</groupId>

<artifactId>SoftwareCourseDesign</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>SoftwareCourseDesign</name>

<description/>

<properties>

<webVersion>4.0</webVersion>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>8.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.15</version>

<scope>system</scope>

<systemPath>

${project.basedir}/src/main/webapp/WEB-INF/lib/commons-codec-1.15.jar

</systemPath>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.6</version>

<configuration>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

</project>

### 数据库配置文件SqlMapConfig.xml

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<environments default="development">

<environment id="development">

<transactionManager type="JDBC" />

<!-- 数据库连接池 -->

<dataSource type="POOLED">

<property name="driver" value="org.postgresql.Driver" />

<property name="url"

value="jdbc:postgresql://127.0.0.1:5432/software\_course\_design" />

<property name="username" value="chenrz" />

<property name="password" value="jsnj130601\*" />

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource="SQLmap/register.xml"/>

</mappers>

</configuration>