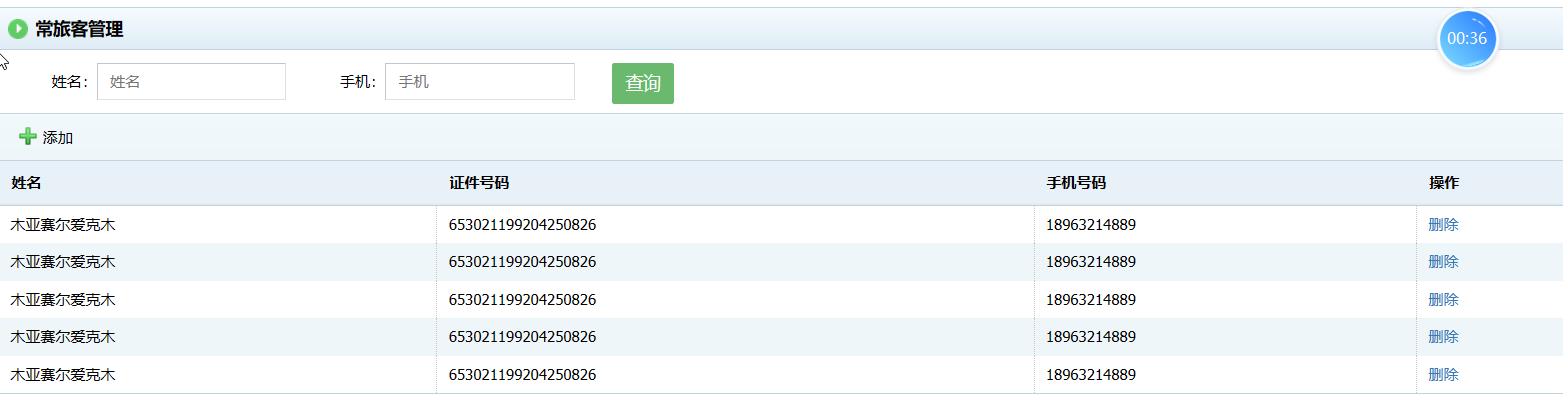
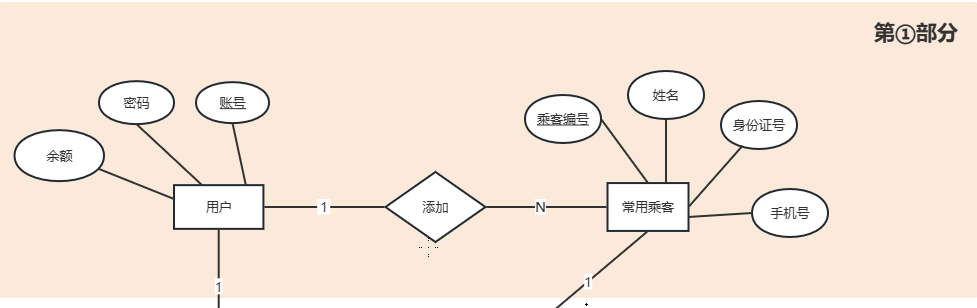
# 功能一：查询所有常旅客

## 静态原型changlvke.html

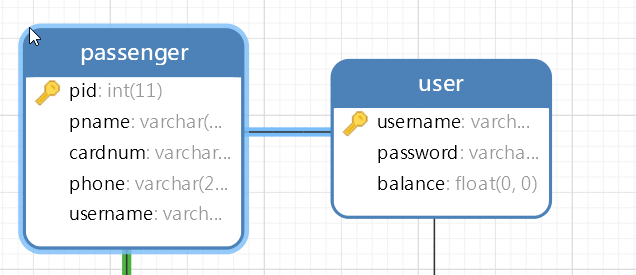


## 数据库的表设计



从ER图分析：用户表和常用旅客表

用户表和常用旅客表之间一对多的关系，需要在常用旅客表中添加一个外键关联用户表的主键



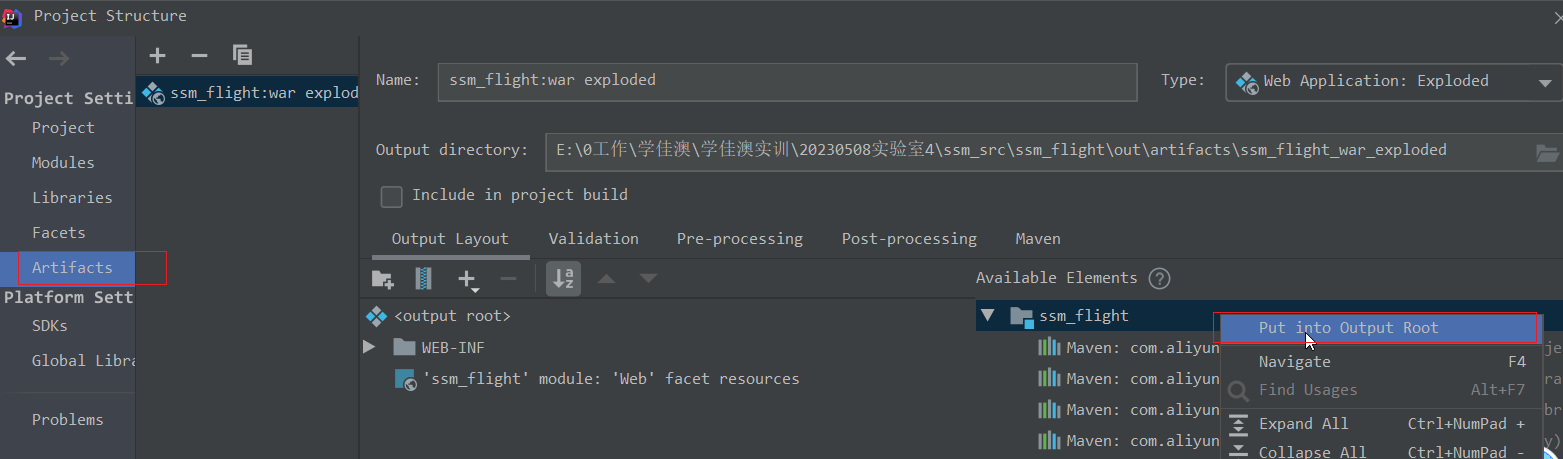
user表中主键：username,账号是唯一的

passenger:常用旅客表

## 搭建SSM项目

### pom.xml文件

jar包手动添加到项目中



### 1.3.2 web.xml文件

### 1.3.3 ssm配置文件

jdbc.properties：修改数据库名称

mybatis.xml：一般不需要修改

spring-context.xml：spring框架的配置文件

|  |
| --- |
| <context:component-scan base-package="com.sy4.flight.service"/> |

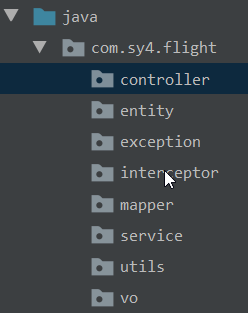
spring-mvc.xml: 修改报红的地方的包路径（要扫描的包路径）

spring-mybatis.xml： 修改报红的地方的包路径及下面的这个路径

|  |
| --- |
| <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  <property name="basePackage" value="com.sy4.flight.mapper"></property> </bean> |

### 1.3.4在web 下创建静态资源目录resources

### 1.3.5创建java源代码包结构



## 创建实体类

|  |
| --- |
| public class UserFt { // username varchar  private String username; // password varchar  private String password; // balance float  private Float balance; } |
| public class PassengerFt { // pid int  private Integer pid; // pname varchar  private String pname; // cardnum varchar  private String cardnum; // phone varchar  private String phone; // username varchar 业务主键  //多对一   private UserFt userFt; } |

## 创建一个条件值对象SearchVO

|  |
| --- |
| public class SearchVO {  //系统用户名  private String username;  //常用旅客名  private String pname;  //常用旅客手机号  private String phone;  } |
| 用来封装查询条件 |

## 1.6创建PassengerFtMapper接口

|  |
| --- |
| @Repository("PassengerFtMapper") public interface PassengerFtMapper {  /\*\*  \* 多条件查询常用旅客列表  \* @param searchVO 条件值对象  \* @return 旅客列表  \*/  public List<PassengerFt> select(SearchVO searchVO); } |

## 1.7创建PassengerFtMapper.xml映射文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"> <mapper namespace="com.sy4.flight.mapper.PassengerFtMapper">  <resultMap id="passengerFtMap" type="com.sy4.flight.entity.PassengerFt">  <id column="pid" property="pid" javaType="java.lang.Integer"></id>  <result column="pname" property="pname" javaType="java.lang.String"></result>  <result column="cardnum" property="cardnum" javaType="java.lang.String"></result>  <result column="phone" property="phone" javaType="java.lang.String"></result>  </resultMap>  <select id="select" resultMap="passengerFtMap">  select \* from passenger\_ft  <where>  username = #{username}  <if test="pname!=null and pname!=''">  and pname like concat('%',#{pname},'%')  </if>  <if test="phone!=null and phone!=''">  and phone = #{phone}  </if>  </where>  </select> </mapper> |

## 1.8创建PassengerFtService 接口及实现类

|  |
| --- |
| public interface PassengerFtService {  /\*\*  \* 多条件分页查询常用旅客列表  \* @param pageNum 当前页码  \* @param recordOfPage 每页显示的记录数  \* @param searchVO 条件值对象  \* @return 分页结果对象  \*/  public PageInfo<PassengerFt> findByPage(Integer pageNum, Integer recordOfPage, SearchVO searchVO); } |
| @Service("PassengerFtServiceImpl") public class PassengerFtServiceImpl implements PassengerFtService {  @Autowired@Qualifier("PassengerFtMapper")  private PassengerFtMapper passengerFtMapper;   @Override  @Transactional(propagation = Propagation.REQUIRED,readOnly = true)  public PageInfo<PassengerFt> findByPage(Integer pageNum, Integer recordOfPage, SearchVO searchVO) {  //分页助手  PageHelper.startPage(pageNum,recordOfPage);  List<PassengerFt> passengerFtList = this.passengerFtMapper.select(searchVO);  PageInfo<PassengerFt> pageInfo = new PageInfo<>(passengerFtList);  return pageInfo;  } } |

## 1.9创建控制器

|  |
| --- |
| @Controller @RequestMapping("/passengerFt") public class PassengerFtController {  @Autowired@Qualifier("PassengerFtServiceImpl")  private PassengerFtService passengerFtService;  //非前后端分离：jsp 或 html+ajax+json  //前后端分离开发：html+ajax+json  @RequestMapping("/list")  public String list(@RequestParam(defaultValue = "1") Integer pageNum,  @RequestParam(defaultValue = "4") Integer recordOfPage,  SearchVO searchVo,  HttpServletRequest request,  Model model){  HttpSession session = request.getSession();  UserFt userFt = (UserFt) session.getAttribute("userFt");  searchVo.setUsername(userFt.getUsername());  PageInfo<PassengerFt> pageInfo = this.passengerFtService.findByPage(pageNum, recordOfPage, searchVo);  model.addAttribute("pageInfo",pageInfo);  //转发到jsp页面  return "userft/changlvke";  } } |

## 1.10 根据静态原型创建userft/changlvke.jsp页面

|  |
| --- |
| <%--条件查询--%>  <form action="passengerFt/list" method="post">  <div class="search-box clearfix">  <div class="kv-item clearfix">  <label>姓名：</label>  <div class="kv-item-content">  <input type="text" placeholder="姓名" name="pname" value="${param.pname}">  </div>  </div>  <div class="kv-item clearfix">  <label>手机：</label>  <div class="kv-item-content">  <input type="text" placeholder="手机" name="phone" value="${param.phone}">  </div>  </div> <%-- <a href="javascript:;" class="sapar-btn sapar-btn-recom query-btn">查询</a>--%>  <input type="submit" class="sapar-btn sapar-btn-recom query-btn" value="查询">  </div>  </form> |
| 数据展示  <tbody> <c:forEach items="${pageInfo.list}" var="pg">  <tr>  <td>${pg.pname}</td>  <td>${pg.cardnum}</td>  <td>${pg.phone}</td>  <td><a href="javascript:;">删除</a></td>  </tr> </c:forEach> </tbody> |

# 日志问题

https://blog.csdn.net/weixin\_44843569/article/details/121937471  
log4j2 ：日志的框架，打印Sql语句

还可以记录异常信息。操作信息等

<!--日志级别以及优先级排序: OFF > FATAL > ERROR > WARN > INFO > DEBUG > TRACE > ALL --> 从高到低，级别越低，越详细

## 导入依赖

|  |
| --- |
| <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-slf4j-impl</artifactId>  <version>2.14.0</version>  </dependency> |

## 2.2配置log4j2.xml配置文件 resources目录下

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <!--日志级别以及优先级排序: OFF > FATAL > ERROR > WARN > INFO > DEBUG > TRACE > ALL --> <!--Configuration后面的status，这个用于设置log4j2自身内部的信息输出，可以不设置，当设置成trace时，你会看到log4j2内部各种详细输出--> <!--monitorInterval：Log4j能够自动检测修改配置 文件和重新配置本身，设置间隔秒数--> <!--https://blog.csdn.net/qq\_16498553/article/details/107778358--> <Configuration status="debug" monitorInterval="30">  <!--全局属性-->  <Properties>  <Property name="APP\_NAME">ssm\_flight</Property>  <!--设置日志文件存储路径为F:/workspace/log4j/${APP\_NAME}-->  <Property name="LOG\_FILE\_PATH">D:/dev/logs/study/${APP\_NAME}</Property>  <!--<Property name="LOG\_FILE\_PATH">./logs/${APP\_NAME}</Property>-->  <!--设置日志输出格式-->  <Property name="PATTERN\_FORMAT">%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n</Property>  </Properties>   <!--配置输出源-->  <Appenders>  <!--输出到控制台-->  <Console name="Console" target="SYSTEM\_OUT">  <PatternLayout pattern="${PATTERN\_FORMAT}" />  </Console>  <!--输出info日志到文件，filePattern旧日志另存文件名-->  <RollingFile name="RollingInfoFile" fileName="${LOG\_FILE\_PATH}/info.log"  filePattern="${LOG\_FILE\_PATH}/$${date:yyyyMM}/info-%d{yyyyMMdd}-%i.log.gz">  <!--控制台只输出level及以上级别的信息（onMatch），其他的直接拒绝（onMismatch）-->  <Filters>  <ThresholdFilter level="debug" onMatch="DENY" onMismatch="NEUTRAL"/>  <ThresholdFilter level="debug" onMatch="ACCEPT" onMismatch="DENY"/>  </Filters>  <PatternLayout>  <pattern>${PATTERN\_FORMAT}</pattern>  </PatternLayout>  <Policies>  <!-- rollover on startup, daily and when the file reaches 10 MegaBytes -->  <!--当系统重启/日期更新/超过100M将会覆盖文件-->  <OnStartupTriggeringPolicy/>  <SizeBasedTriggeringPolicy size="100 MB"/>  <TimeBasedTriggeringPolicy/>  </Policies>  </RollingFile>  </Appenders>   <Loggers>  <Root level="debug">  <AppenderRef ref="applicationLog" />  <AppenderRef ref="Console" />  <AppenderRef ref="RollingInfoFile" />  </Root>  </Loggers> </Configuration> |

# 功能二：新增常用旅客

## 3.1静态原型



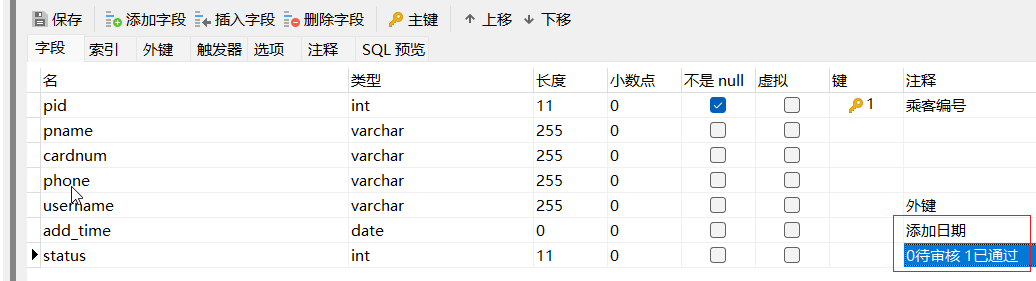
添加日期：记录什么时间添加的这个旅客（当前系统时间）

审核状态：默认添加的都是待审核

|  |
| --- |
| <button type="button"> 发送验证码 </button>  <div class="kv-item clearfix">  <label>验证码</label>  <div class="kv-item-content">  <input type="text" placeholder="验证码">  </div>  <div class="kv-item-tip">请正确填写手机收到验证码</div> </div> |

/

## 3.2 数据库原型



## 3.3修改PassengerFt实体类

|  |
| --- |
| //添加时间和审核状态  private Date addTime;  private Integer status; |

## 3.4编写PassengerFtMapper接口

|  |
| --- |
| /\*\*  \* 新增常用旅客  \* @param passengerFt 常用旅客  \*/ public void insert(PassengerFt passengerFt); |

## 3.5编写PassengerFtMapper.xml文件

|  |
| --- |
| ...  <result column="add\_time" property="addTime" javaType="java.util.Date"></result>  <result column="status" property="status" javaType="java.lang.Integer"></result>  </resultMap>  <insert id="insert">  insert into passenger\_ft(pname,cardnum,phone,username,add\_time,status)  values(#{pname},#{cardnum},#{phone},#{userFt.username},#{addTime},#{status}) </insert> |

## 3.6编写PassengerFtService接口

|  |
| --- |
| /\*\*  \* 新增常用旅客  \* @param passengerFt 常用旅客  \*/ public void save(PassengerFt passengerFt); |

## 3.7编写PassengerFtServiceImpl实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.REQUIRED,rollbackFor = Exception.class) public void save(PassengerFt passengerFt) {  this.passengerFtMapper.insert(passengerFt); } |

## 3.8编写PassengerFtController控制器

|  |
| --- |
| @RequestMapping("/doAdd") public String doAdd(PassengerFt passengerFt,HttpServletRequest request){  HttpSession session = request.getSession();  UserFt userFt = (UserFt) session.getAttribute("userFt");  userFt = new UserFt();  userFt.setUsername("Jerry");  passengerFt.setUserFt(userFt);  passengerFt.setStatus(0);  passengerFt.setAddTime(new Date());  this.passengerFtService.save(passengerFt);  return "redirect:passengerFt/list"; } |

## 3.9增加一些业务：旅客姓名不为空的校验，身份证号的校验和手机号的校验，发送验证码

身份证号校验，年龄大于等于18，身份证号不能重复。

手机号格式校验

### 3.9.1 修改addlvke.jsp页面

采用js,直接使用js进行校验

|  |
| --- |
| 旅客姓名不为空 |
| //检验旅客姓名不为空 函数要给返回值 一般给true 或者 false function checkPname(){  if($("#pname").val()==''){  alert("旅客姓名不能为空")  return false;  }else{  return true;  } } |
| //校验年龄是否 18岁 大于=18 返回true, 否则返回false function checkAge(){  let cardnum = $('#cardnum').val();  let age;  //当前系统时间  let now = new Date();  age = now.getFullYear()-cardnum.substring(6,10)-1  if((now.getMonth()+1)>cardnum.substring(10,12) ||  (now.getMonth()+1)==cardnum.substring(10,12) && now.getDate()>cardnum.substring(12,14)){  age++  }  if(age<18){  alert("未满18！")  return false;  }else{  return true;  } } |
| //校验身份证号格式 function checkCardNum(){  let cardnumObj = $('#cardnum');  //校验身份证格式  var reg=/^(\d{15}$|^\d{18}$|^\d{17}(\d|X|x))$/  var re = new RegExp(reg);  if(!re.test(cardnumObj.val())){  alert("身份证号格式不正确")  return false;  }  return true; } |
| //校验手机号格式 function checkPhone(){  //1.正则：检验手机号  let reg = /^1[3-9]\d{9}$/  let re = new RegExp(reg);  //用户填写的手机号  let phone = $('#phone').val();  if(!re.test(phone)){  alert("请输入正确的手机号！！！")  return false;  }else{  return true;  } } |
| //校验手机号在数据库中是否不存在，返回true 不存在，可以使用，返回false 存在，不可用 function phoneIsNotExist(){  let flag = true;  //2.发送ajax请求看手机号在数据库是否存在  $.ajax({  url:"passengerFt/checkPhone",  method:"post",  data:{"phone":$('#phone').val()},  dataType:"json",  //ajax 发送同步请求  async:false,  success:function (data) {  if(data.code==20001){//手机号在数据库不存在，手机号可用  flag = true;  }else{ //手机号不可用  flag = false;  alert(data.msg)  }  }  })  return flag; } |
| //校验身份证号在数据库中是否不存在 返回true 不存在， function cardnumIsNotExist(){  let flag = true;  //格式符合要求  if(checkCardNum()&&checkAge()){  // 校验身份证号在数据库中是否不存在  $.ajax({  url:"passengerFt/checkCardnum",  method:"post",  data:{"cardnum":$('#cardnum').val()},  dataType:"json",  async:false,  success:function (data) {  if(data.code==10001){ //身份证在数据库不存在，可用  flag = true;  }else{  flag = false;  alert(data.msg)  }  }  })  }  return flag; } |
| 事件： |
| //发送验证码事件 $('#sendCode').click(function () {  if(checkPhone()&&phoneIsNotExist()){  $.ajax({  url:"send/msg",  method:"post",  data:{"phone":$('#phone').val()},  dataType:"json",  success:function (data) {  if(data == "OK"){  alert("发送成功")  }else{  alert("发送失败")  }  }  })  } }) |
| //给身份证绑定离焦事件 $('#cardnum').blur(function(){  //1.检验格式  //2.检验年龄  //3.是否不存在  cardnumIsNotExist() }) |
| //表单提交前进行检验，返回true,通过检验，提交表单，返回false,未通过校验不会提交表单 //在form标签上添加 onsubmit="return validate()" 在表单提交之前 调用的验证函数（函数返回true 提交表单，返回false 不提交表单） function validate() {  if(checkPname()&&checkAge()&&checkCardNum()&&cardnumIsNotExist()&&checkPhone()&&phoneIsNotExist()){  return true;  }else{  return false;  } } |
| <form id="addForm" class="saper-form" action="#" method="post" onsubmit="return validate()"> |

### 3.9.2 编写PassengerFtMapper接口

|  |
| --- |
| /\*\*  \* 根据身份证号查询常用旅客对象  \* @param cardnum 身份证号  \* @return 唯一旅客对象  \*/ public PassengerFt selectByCardnum(String cardnum); /\*\*  \* 根据手机号查询常用旅客对象  \* @param phone 手机号  \* @return 唯一旅客对象  \*/ public PassengerFt selectByPhone(String phone); |

### 3.9.3编写PassengerFtService接口

|  |
| --- |
| /\*\*  \* 根据身份证号查询常用旅客对象  \* @param cardnum 身份证号  \* @return 唯一旅客对象  \*/ public PassengerFt selectByCardnum(String cardnum); /\*\*  \* 根据手机号查询常用旅客对象  \* @param phone 手机号  \* @return 唯一旅客对象  \*/ public PassengerFt selectByPhone(String phone); |

### 3.9.4编写PassengerFtServiceImpl实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.REQUIRED,readOnly = true) public PassengerFt selectByCardnum(String cardnum) {  return this.passengerFtMapper.selectByCardnum(cardnum); }  @Override @Transactional(propagation = Propagation.REQUIRED,readOnly = true) public PassengerFt selectByPhone(String phone) {  return this.passengerFtMapper.selectByPhone(phone); } |

### 3.9.5编写PassengerFtController控制器

|  |
| --- |
| @RequestMapping("/checkCardnum") @ResponseBody public ResultVO checkCardnum(String cardnum){  ResultVO resultVO = new ResultVO();   PassengerFt passengerFt = this.passengerFtService.selectByCardnum(cardnum);  if(passengerFt==null){  resultVO.setCode(10001);  resultVO.setMsg("身份证未绑定，可用!");  }else{  resultVO.setCode(10000);  resultVO.setMsg("身份证已绑定，不可用!");  }  return resultVO; } |
| @RequestMapping("/checkPhone") @ResponseBody public ResultVO checkPhone(String phone){  ResultVO resultVO = new ResultVO();  PassengerFt passengerFt = this.passengerFtService.selectByPhone(phone);  if(passengerFt==null){  resultVO.setCode(20001);  resultVO.setMsg("手机号未绑定，可用!");  }else{  resultVO.setCode(20000);  resultVO.setMsg("手机号已绑定，不可用!");  }  return resultVO; } |
| @RequestMapping("/doAdd") public String doAdd(Integer code,PassengerFt passengerFt,HttpServletRequest request){  HttpSession session = request.getSession();  UserFt userFt = (UserFt) session.getAttribute("userFt");  //检验用户输入的code和session中保存的是否一致  Integer sessionCode = (Integer) session.getAttribute(passengerFt.getPhone());  if(sessionCode == null){  //手机号不对  return "userft/addlvke";  }else{  if(!sessionCode.equals(code)){  return "userft/addlvke";  }else{  userFt = new UserFt();  userFt.setUsername("Jerry");  passengerFt.setUserFt(userFt);  passengerFt.setStatus(0);  passengerFt.setAddTime(new Date());  this.passengerFtService.save(passengerFt);  return "redirect:/passengerFt/list";  }  } } |

### 3.9.6发送验证码的类

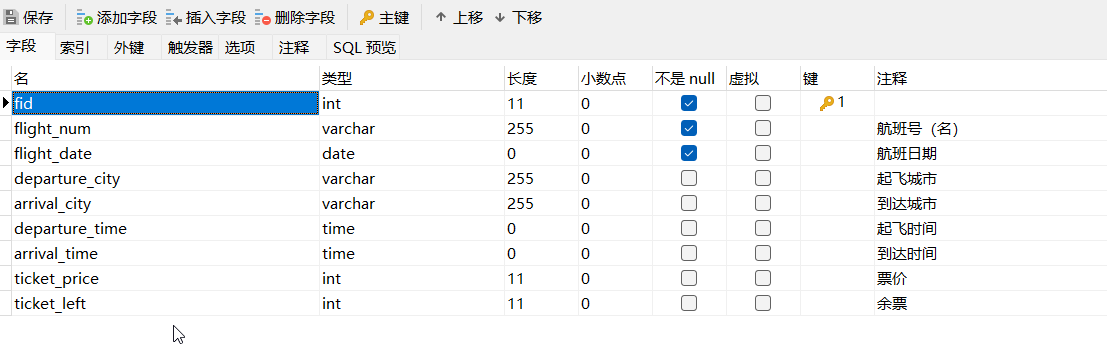
|  |  |  |
| --- | --- | --- |
| |  | | --- | | 发送验证码的类 | | /\*\*  \* @author: Mr.Shan  \* @date: 2023/07/04  \* @descriptions:  \* @version: 1.0  \*/ @Controller @RequestMapping("/send") public class SendMsgController {  public static final String ALIBABA\_CLOUD\_ACCESS\_KEY\_ID = "LTAI5tKdPsDePwXTYizRn12B1";  public static final String ALIBABA\_CLOUD\_ACCESS\_KEY\_SECRET = "ija3fVfve1TjDZSwoY60YaEMprrRim";    public com.aliyun.dysmsapi20170525.Client createClient(String accessKeyId, String accessKeySecret) throws Exception {  com.aliyun.teaopenapi.models.Config config = new com.aliyun.teaopenapi.models.Config()  // 必填，您的 AccessKey ID  .setAccessKeyId(accessKeyId)  // 必填，您的 AccessKey Secret  .setAccessKeySecret(accessKeySecret);  // 访问的域名  config.endpoint = "dysmsapi.aliyuncs.com";  return new com.aliyun.dysmsapi20170525.Client(config);  }   @RequestMapping("/msg")  @ResponseBody  public String sendMsg(String phone, HttpServletRequest request) throws Exception {  Random random = new Random();  //生成一个4位数的验证码  int code = random.nextInt(9000) + 1000;  // 请确保代码运行环境设置了环境变量 ALIBABA\_CLOUD\_ACCESS\_KEY\_ID 和 ALIBABA\_CLOUD\_ACCESS\_KEY\_SECRET。  // 工程代码泄露可能会导致 AccessKey 泄露，并威胁账号下所有资源的安全性。以下代码示例使用环境变量获取 AccessKey 的方式进行调用，仅供参考，建议使用更安全的 STS 方式，更多鉴权访问方式请参见：https://help.aliyun.com/document\_detail/378657.html  com.aliyun.dysmsapi20170525.Client client = createClient(ALIBABA\_CLOUD\_ACCESS\_KEY\_ID,ALIBABA\_CLOUD\_ACCESS\_KEY\_SECRET);  com.aliyun.dysmsapi20170525.models.SendSmsRequest sendSmsRequest = new com.aliyun.dysmsapi20170525.models.SendSmsRequest()  .setSignName("航空管理")  .setTemplateCode("SMS\_461855342")  .setPhoneNumbers(phone)  .setTemplateParam("{\"code\":\""+code+"\"}");   //将验证码放到Session中  HttpSession session = request.getSession();  //手机号当成key  session.setAttribute(phone,code);  System.out.println("发送的验证码是"+code);  com.aliyun.teautil.models.RuntimeOptions runtime = new com.aliyun.teautil.models.RuntimeOptions();  try {  // 复制代码运行请自行打印 API 的返回值 // SendSmsResponse response = client.sendSmsWithOptions(sendSmsRequest, runtime); // System.out.println(response.getBody().getCode()); // return response.getBody().getCode(); //"OK"  return "OK";  } catch (TeaException error) {  // 如有需要，请打印 error  com.aliyun.teautil.Common.assertAsString(error.message);  } catch (Exception \_error) {  TeaException error = new TeaException(\_error.getMessage(), \_error);  // 如有需要，请打印 error  com.aliyun.teautil.Common.assertAsString(error.message);  }  return "NO";  } } | |

# 功能三：查询航班

## 4.1静态原型



## 4.2表的设计



time：时间，只有时分秒

实体类：Date或者字符串

## 4.3编写实体类

|  |
| --- |
| public class Flight { // fid int  private Integer fid; // flight\_num varchar  private String flightNum; // flight\_date date  private Date flightDate; // departure\_city varchar  private String departureCity; // arrival\_city varchar  private String arrivalCity; // departure\_time time 时分秒  private Date departureTime; // arrival\_time time private Date arrivalTime; // ticket\_price int  private Integer ticketPrice; // ticket\_left int  private Integer ticketLeft;  } |

## 4.4编写SearchVOFlight条件

|  |
| --- |
| public class SearchVOFlight {  // flight\_date date  private Date flightDate;  // departure\_city varchar  private String departureCity;  // arrival\_city varchar  private String arrivalCity;  } |

## 4.5编写FlightMapper接口

|  |
| --- |
| @Repository("FlightMapper") public interface FlightMapper {  /\*\*  \* 多条件查询航班信息  \* @param searchVOFlight 条件  \* @return 航班列表  \*/  public List<Flight> selectBySearch(SearchVOFlight searchVOFlight); } |

## 4.6编写FlightMapper.xml映射文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"> <mapper namespace="com.sy4.flight.mapper.FlightMapper">  <resultMap id="flightMap" type="com.sy4.flight.entity.Flight">  <id column="fid" property="fid" javaType="java.lang.Integer"></id>  <result column="flight\_num" property="flightNum" javaType="java.lang.String"/>  <result column="flight\_date" property="flightDate" javaType="java.util.Date"></result>  <result column="departure\_city" property="departureCity" javaType="java.lang.String"></result>  <result column="arrival\_city" property="arrivalCity" javaType="java.lang.String"></result>  <result column="departure\_time" property="departureTime" javaType="java.util.Date"></result>  <result column="arrival\_time" property="arrivalTime" javaType="java.util.Date"></result>  <result column="ticket\_price" property="ticketPrice" javaType="java.lang.Integer"></result>  <result column="ticket\_left" property="ticketLeft" javaType="java.lang.Integer"></result>  </resultMap>  <select id="selectBySearch" resultMap="flightMap">  select \* from flight  <where>  <if test="departureCity!=null and departureCity!=''">  and departure\_city like concat('%',#{departureCity},'%')  </if>  <if test="arrivalCity!=null and arrivalCity!=''">  and arrival\_city like concat('%',#{arrivalCity},'%')  </if>  <if test="flightDate!=null">  and flight\_date = #{flightDate,jdbcType=DATE}  </if>  </where>  </select> </mapper> |
| jdbcType:mybaits会将传入的参数转为Date类型（年月日） |

## 4.7编写业务层FlightService

|  |
| --- |
| public interface FlightService {  /\*\*  \* 多条件查询航班信息  \* @param searchVOFlight 条件  \* @return 航班列表  \*/  public List<Flight> selectBySearch(SearchVOFlight searchVOFlight); } |

## 4.8编写FlightServiceImpl实现类

|  |
| --- |
| @Service("FlightServiceImpl") public class FlightServiceImpl implements FlightService {  @Autowired@Qualifier("FlightMapper")  private FlightMapper flightMapper;  @Override  @Transactional(propagation = Propagation.REQUIRED,readOnly = true)  public List<Flight> selectBySearch(SearchVOFlight searchVOFlight) {  return this.flightMapper.selectBySearch(searchVOFlight);  } } |

## 4.9编写FlightController控制器

|  |
| --- |
| @Controller @RequestMapping("/flight") public class FlightController {  @Autowired@Qualifier("FlightServiceImpl")  private FlightService flightService;  @RequestMapping("/list")  private String list(SearchVOFlight searchVOFlight, Model model){  List<Flight> flights = flightService.selectBySearch(searchVOFlight);  model.addAttribute("flights",flights);  //转发到jsp页面  return "ticket/order\_step1";  } } |

## 4.10修改order\_step1.jsp

|  |
| --- |
| <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  ***表单***  <form name="f1" id="f1" action="flight/list" method="get">  <input type="text" name="departureCity" value="${param.departureCity}" placeholder="出发城市">  <input type="text" name="arrivalCity" value="${param.arrivalCity}" placeholder="到达城市">  <input type="date" name="flightDate" value="${param.flightDate}" placeholder="出发日期">  <button type="submit" class="sapar-btn sapar-btn-recom query-btn" >查询</button> |
| 表格  <c:forEach items="${requestScope.flights}" var="flt">  <tr>  <td><fmt:formatDate value="${flt.flightDate}" pattern="yyyy-MM-dd" /></td>  <td>${flt.flightNum}</td>  <td>${flt.departureCity}</td>  <td>${flt.arrivalCity}</td>  <td><fmt:formatDate value="${flt.departureTime}" pattern="HH:mm" /></td>  <td><fmt:formatDate value="${flt.arrivalTime}" pattern="HH:mm" /></td>  <td>¥${flt.ticketPrice}</td>  <td>${flt.ticketLeft}</td>  <td>  <a href="order\_step2.html" class="">预订机票</a>  </td>  </tr> </c:forEach> |



@DateTimeFormat(pattern = "yyyy-MM-dd")

private Date flightDate;

# 功能四：预订机票

## 5.1静态原型



## 5.2 机票信息展示

### 5.2.1 编写FlightMapper接口

|  |
| --- |
| /\*\*  \* 根据航班id 查询航班信息  \* @param fid 航班id  \* @return 航班信息  \*/ public Flight selectByFid(Integer fid); |

### 5.2.2 编写FlightMapper.xml映射文件

|  |
| --- |
| <select id="selectByFid" resultMap="flightMap">  select *\** from flight where fid = #{fid} </select> |

### 5.2.3编写service 接口

|  |
| --- |
| ***/\*\*  \* 根据航班id 查询航班信息  \* @param fid 航班id  \* @return 航班信息  \*/* public Flight selectByFid(Integer fid);** |

### 5.2.4编写service接口的实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.*REQUIRED*,readOnly = true) public Flight selectByFid(Integer fid) {  return this.flightMapper.selectByFid(fid); } |

## 5.3乘机人列表展示 当前登录用户+审核状态通过的

### 5.3.1编写PassengerFtMapper接口

|  |
| --- |
| */\*\*  \* 根据登录用户名和常用旅客的状态来查询旅客列表  \* @param username 登录用户名  \* @param status 状态  \* @return 旅客列表  \*/* public List<PassengerFt> selectByUsernameAndStatus(@Param("username") String username, @Param("status") Integer status); |

### 5.3.2编写PassengerFtMapper.xml映射文件

|  |
| --- |
| <select id="selectByUsernameAndStatus" resultMap="passengerFtMap">  select *\** from passenger\_ft where username=#{username} and status=#{status} </select> |

### 5.3.3编写PassengerFtService接口

|  |
| --- |
| */\*\*  \* 根据登录用户名和常用旅客的状态来查询旅客列表  \* @param username 登录用户名  \* @param status 状态  \* @return 旅客列表  \*/* public List<PassengerFt> selectByUsernameAndStatus(String username,Integer status); |

### 5.3.4编写PassengerFtServiceImpl实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.*REQUIRED*,readOnly = true) public List<PassengerFt> selectByUsernameAndStatus(String username, Integer status) {  return this.passengerFtMapper.selectByUsernameAndStatus(username,status); } |

## 5.4编写FlightController控制器

|  |
| --- |
| @Autowired@Qualifier("FlightServiceImpl") private FlightService flightService; @Autowired@Qualifier("PassengerFtServiceImpl") private PassengerFtService passengerFtService; @RequestMapping("/selectById/{fid}") public String selectById(@PathVariable("fid") Integer fid, Model model, HttpServletRequest request){  HttpSession session = request.getSession();  UserFt userFt = (UserFt) session.getAttribute("userFt");  List<PassengerFt> passengerFts= passengerFtService.selectByUsernameAndStatus("Jerry", 1);  model.addAttribute("passengerFts",passengerFts);  Flight flight = this.flightService.selectByFid(fid);  model.addAttribute("flt",flight);  return "ticket/order\_step2"; } |

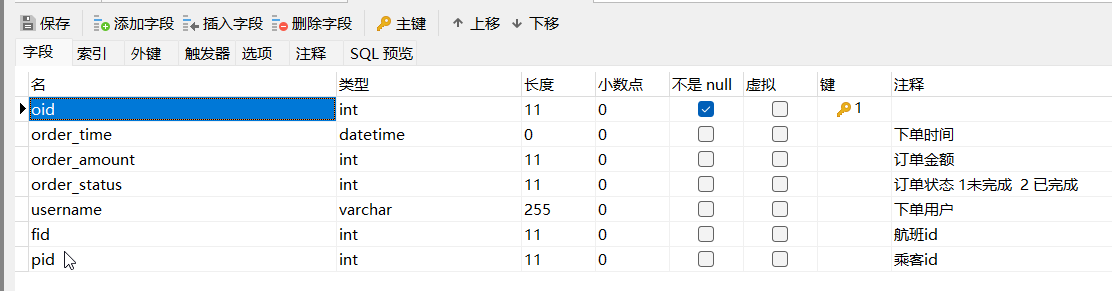
## 5.5编写order\_step2.jsp页面

|  |
| --- |
| <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %> |
| <base href="${pageContext.request.contextPath}/"> |
| <tr>  <td><fmt:formatDate value="${flt.flightDate}" pattern="yyyy-MM-dd" /></td>  <td>${flt.flightNum}</td>  <td>${flt.departureCity}</td>  <td>${flt.arrivalCity}</td>  <td><fmt:formatDate value="${flt.departureTime}" pattern="HH:mm" /></td>  <td><fmt:formatDate value="${flt.arrivalTime}" pattern="HH:mm" /></td>  <td>¥${flt.ticketPrice}</td> </tr> |
| <label>旅客选择：</label> <div class="kv-item-content">  <select name="pid">  <c:forEach items="${requestScope.passengerFts}" var="pft">  <option value="${pft.pid}">  ${pft.pname}-${fn:substring(pft.cardnum,fn:length(pft.cardnum)-6,fn:length(pft.cardnum))} </option>  </c:forEach>  </select> </div> |

# 功能五：生成订单

## 静态原型

## 数据库表原型



## 向order表中添加一条记录

### 编写Flightorder实体类

|  |
| --- |
| public class Flightorder { *// oid int* private Integer oid; *// order\_time datetime* private Date orderTime; *// order\_amount int* private Integer orderAmount; *// order\_status int* private Integer orderStatus; *// username varchar* private UserFt userFt; *// fid int* private Flight flight; *// pid int* private PassengerFt passengerFt;  } |

### 编写FlightorderMapper接口

|  |
| --- |
| @Repository("FlightorderMapper") public interface FlightorderMapper {  */\*\*  \* 保存一个订单  \* @param flightorder 订单信息  \*/* public void insert(Flightorder flightorder); } |

### 编写FlightorderMapper.xml映射文件

|  |
| --- |
| <insert id="insert" parameterType="com.sy4.flight.entity.Flightorder">  insert into flightorder(oid,order\_time,order\_amount,order\_status,username,fid,pid)  values (#{oid},#{orderTime},#{orderAmount},#{orderStatus},#{userFt.username},#{flight.fid},#{passengerFt.pid}) </insert> |

## 6.4更新航班余票

### 6.4.1编写FlightMapper接口

|  |
| --- |
| */\*\*  \* 根据航班id 更新余票  \* @param fid 航班id  \*/* public void updateTicketLeft(Integer fid); |

### 6.4.2编写FlightMapper.xml映射文件

|  |
| --- |
| <update id="updateTicketLeft">  update flight set ticket\_left=ticket\_left-1 where fid = #{fid} </update> |

## 6.5编写生成订单的业务FlightorderService接口

|  |
| --- |
| public interface FlightorderService {  */\*\*  \* 保存一个订单  \* @param flightorder 订单信息  \*/* public void save(Flightorder flightorder); } |

## 6.6编写FlightorderServiceImpl实现类

|  |
| --- |
| */\*\*  \* @author: Mr.Shan  \* @date: 2023/07/21  \* @descriptions: 生成订单  \* @version: 1.0  \*/* @Service("FlightorderServiceImpl") public class FlightorderServiceImpl implements FlightorderService {  @Autowired@Qualifier("FlightMapper")  private FlightMapper flightMapper;  @Autowired@Qualifier("FlightorderMapper")  private FlightorderMapper flightorderMapper;  @Override  public void save(Flightorder flightorder) {  *//如果有余票* if(this.flightMapper.selectByFid(flightorder.getFlight().getFid()).getTicketLeft()>=1){  *//更新余票* this.flightMapper.updateTicketLeft(flightorder.getFlight().getFid());  *//保存订单* flightorderMapper.insert(flightorder);  }else{  throw new TicketLeftNotException("余票不足");  }  } } |

## 6.7编写FlightorderController控制器

|  |
| --- |
| @Controller @RequestMapping("/flightorder") public class FlightorderController {  @Autowired@Qualifier("FlightorderServiceImpl")  private FlightorderService flightorderService;  @RequestMapping("/add")  public String add(Integer orderAmount,Integer fid,Integer pid){  Flightorder flightorder = new Flightorder();  flightorder.setOid(UUID.*randomUUID*().toString().replace("-",""));  flightorder.setOrderAmount(orderAmount);  Flight flight = new Flight();  flight.setFid(fid);  flightorder.setFlight(flight);  flightorder.setOrderStatus(1);  flightorder.setOrderTime(new Date());  PassengerFt passengerFt = new PassengerFt();  passengerFt.setPid(pid);  flightorder.setPassengerFt(passengerFt);  UserFt userFt = new UserFt();  userFt.setUsername("Jerry");  flightorder.setUserFt(userFt);  try {  this.flightorderService.save(flightorder);  *//当我们将oid放到model中，重定向时，会将oid 以参数的形式发送过去* model.addAttribute("oid",flightorder.getOid());  return "redirect:/flightorder/findByOid";  }catch (RuntimeException e){  return "";  }  } } |

# 功能六：加载选择支付的页面

## 7.1 编写FlightorderMapper接口

|  |
| --- |
| */\*\*  \* 根据订单id 查询订单信息  \* @param oid 订单id  \* @return 订单信息  \*/* public Flightorder selectByOid(String oid); |

## 7.2编写FlightorderMapper.xml映射文件

|  |
| --- |
| <select id="selectByOid" resultMap="orderMap">  select  fo.oid,fo.order\_amount,  f.departure\_city,f.arrival\_city,f.flight\_date,f.departure\_time, pf.pname,pf.cardnum  from flightorder fo  left join flight f on fo.fid = f.fid  left join passenger\_ft pf on fo.pid = pf.pid  where oid= #{oid}  </select> |
| <resultMap id="orderMap" type="com.sy4.flight.entity.Flightorder">  <id column="oid" property="oid" javaType="java.lang.String"/>  <result column="order\_amount" property="orderAmount" javaType="java.lang.Integer"/>  <association property="flight" resultMap="com.sy4.flight.mapper.FlightMapper.flightMap"/>  <association property="passengerFt" resultMap="com.sy4.flight.mapper.PassengerFtMapper.passengerFtMap"/> </resultMap> |

## 7.3编写FlightorderService接口

|  |
| --- |
| */\*\*  \* 根据订单id 查询订单信息  \* @param oid 订单id  \* @return 订单信息  \*/* public Flightorder findByOid(String oid); |

## 7.4编写FlightorderServiceImpl实现类

|  |
| --- |
| @Service("FlightorderServiceImpl") public class FlightorderServiceImpl implements FlightorderService {  @Autowired@Qualifier("FlightMapper")  private FlightMapper flightMapper;  @Autowired@Qualifier("FlightorderMapper")  private FlightorderMapper flightorderMapper;   @Override  @Transactional(propagation = Propagation.*REQUIRED*,readOnly = true)  public Flightorder findByOid(String oid) {  return this.flightorderMapper.selectByOid(oid);  }  @Override  @Transactional(propagation = Propagation.*REQUIRED*,rollbackFor = {Exception.class})  public void save(Flightorder flightorder) {  *//如果有余票* if(this.flightMapper.selectByFid(flightorder.getFlight().getFid()).getTicketLeft()>=1){  *//更新余票* this.flightMapper.updateTicketLeft(flightorder.getFlight().getFid());  *//保存订单* flightorderMapper.insert(flightorder);  }else{  throw new TicketLeftNotException("余票不足");  }  } } |

## 7.5编写FlightorderController控制器

|  |
| --- |
| @RequestMapping("/findByOid") public String findByOId(String oid, Model model){  Flightorder flightorder = this.flightorderService.findByOid(oid);  model.addAttribute("flightorder",flightorder);  *//转到选择支付方法的页面* return "ticket/order\_step3"; } |

## 7.6编写order\_step3.jsp页面

|  |
| --- |
| <div class="OrderInfo w1190 boxS">  <div class="one">  订单金额:&nbsp;<span style="color: #6AB96E;font-size: 18px;">￥${flightorder.orderAmount}</span>  </div>  <div class="two">  <p>  单程机票&nbsp;&nbsp;&nbsp;<span>${flightorder.flight.departureCity}-${flightorder.flight.arrivalCity}</span> 出发时间：<span>${flightorder.flight.flightDate} ${flightorder.flight.departureTime}</span>  </p>  <p>  乘机人：<span>${flightorder.passengerFt.pname}</span>&nbsp; 乘机证件:&nbsp;身份证${flightorder.passengerFt.cardnum}  </p>  </div> </div> |
| <div class="way w1190 boxS">  <h3>  支付金额 ：<b>￥${flightorder.orderAmount}</b>  </h3>  <div class="wayArea">  <ul class="clearfix">  <li class="clearfix">  <input type="radio" checked="" name="funct">  <p class="checkbox checked">  <img src="images/zfb.png" alt="">  <em></em>  </p>  </li>  </ul>  <form style="display: none" name="f1" action="flightorder/pay" method="post">  <input name="orderId" value="${flightorder.oid}">  <input name="orderPrice" value="${flightorder.orderAmount}"/>  <input name="orderName" value="${flightorder.passengerFt.pname}"/>  <input name="orderAddr" value="哈哈"/>  <input type="submit" >  </form>  <a href="javascript:document.f1.submit()">用支付宝支付</a>  </div> </div> |

# 功能七：支付宝沙箱支付

https://opendocs.alipay.com/common/02kipk?pathHash=0d20b438（官方文档）

## 8.1什么是沙箱支付

沙箱环境是协助开发者进行接口开发及主要功能联调的模拟环境

## 8.2沙箱账号

商家和买家账号信息

商家 收钱

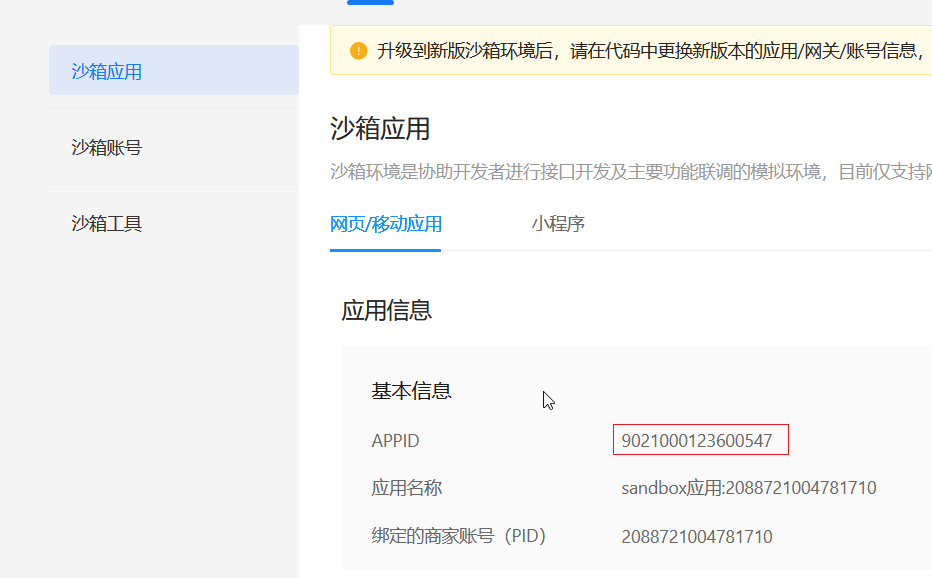
买家 付钱

## 8.3沙箱应用

登录支付宝开发平台

沙箱应用：https://open.alipay.com/develop/sandbox/app

APPID:标识



## 8.4 沙箱应用做的事儿

给我们提供支付界面（支付二维码）

调用沙箱支付的接口，接口返回二维码，付款界面

买家扫码之后，跳转到支付成功界面，修改订单状态（我们做）

## 8.5 编写配置类AlipayConfig

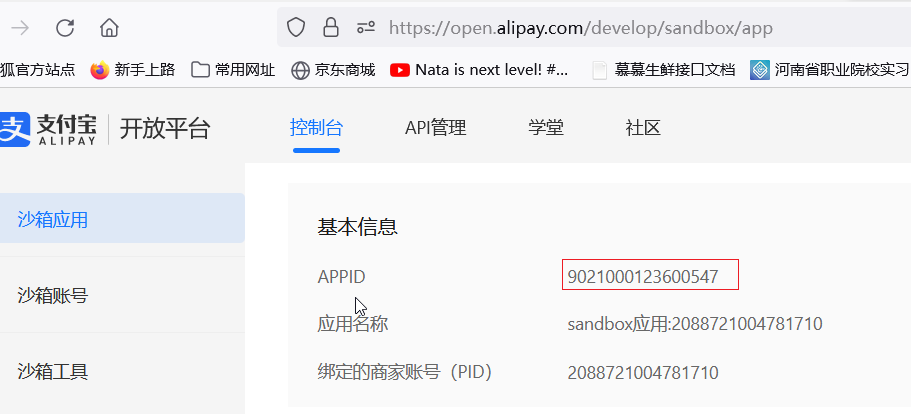
### 8.5.1导入依赖

|  |
| --- |
| <dependency>  <groupId>com.alipay.sdk</groupId>  <artifactId>alipay-sdk-java</artifactId>  <version>4.34.0.ALL</version> </dependency> |

### 8.5.2编写配置类AlipayConfig

|  |
| --- |
| */\*\*  \* @author: Mr.Shan  \* @date: 2023/07/21  \* @descriptions: 支付宝支付配置类  \* @version: 1.0  \*/* public class AlipayConfig { *// ↓↓↓↓↓↓↓↓↓↓请在这里配置您的基本信息↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓  // 应用ID,您的APPID，收款账号既是您的APPID对应支付宝账号* public static String *app\_id* = "9021000123600547";  *// 商户私钥，您的PKCS8格式RSA2私钥* public static String *merchant\_private\_key* = "商户私钥";  *// 支付宝公钥,查看地址：https://openhome.alipay.com/platform/keyManage.htm  // 对应APPID下的支付宝公钥。* public static String *alipay\_public\_key* = "支付宝公钥";  *// 服务器异步通知页面路径 需http://格式的完整路径，不能加?id=123这类自定义参数，必须外网可以正常访问* public static String *notify\_url* = "http://localhost:8082/ssm\_flight\_war\_exploded/flightorder/payresult";  *// 页面跳转同步通知页面路径 需http://格式的完整路径，不能加?id=123这类自定义参数，必须外网可以正常访问  //支付成功会向这个url映射的控制器发请求==>我们可以在这个控制器中修改订单状态等* public static String *return\_url* = "http://localhost:8082/ssm\_flight\_war\_exploded/flightorder/payresult";  *// 签名方式* public static String *sign\_type* = "RSA2";   *// 字符编码格式* public static String *charset* = "utf-8";  *// 支付宝网关 //  // public static String gatewayUrl ="https://openapi.alipay.com/gateway.do";// 正式* public static String *gatewayUrl* = "https://openapi-sandbox.dl.alipaydev.com/gateway.do";*// 测试   // 支付宝网关 日志* public static String *log\_path* = "D:\\alipay\\";  *// ↑↑↑↑↑↑↑↑↑↑请在这里配置您的基本信息↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑  /\*\*  \* 写日志，方便测试（看网站需求，也可以改成把记录存入数据库）  \*  \* @param sWord  \* 要写入日志里的文本内容  \*/* public static void logResult(String sWord) {  FileWriter writer = null;  try {  writer = new FileWriter(*log\_path* + "alipay\_log\_" + System.*currentTimeMillis*() + ".txt");  writer.write(sWord);  } catch (Exception e) {  e.printStackTrace();  } finally {  if (writer != null) {  try {  writer.close();  } catch (IOException e) {  e.printStackTrace();  }  }  }  } } |

### 8.5.3 APPID



### 8.5.4 商户私钥使用密钥工具生成



### 8.5.5支付宝公钥

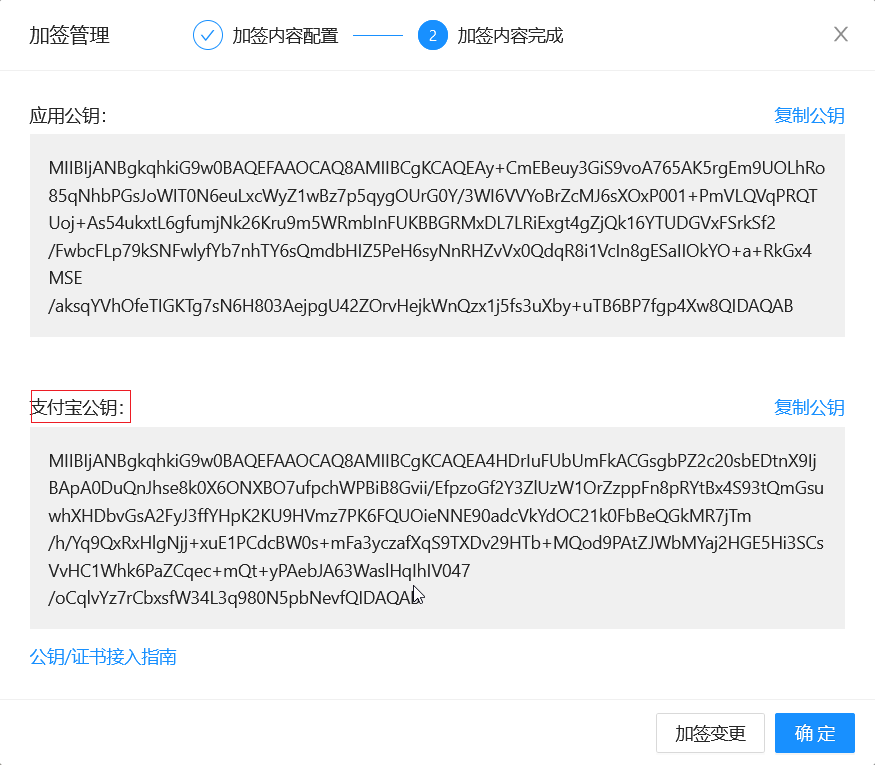


第一次进来时是设置与查看，点击弹出如下窗口

将8.5.3使用密钥工具生成的应用公钥复制进去，保存

会生成一个支付宝公钥





### 8.5.6支付宝网关

https://openapi-sandbox.dl.alipaydev.com/gateway.do

## 8.6 编写FlightorderController控制器

### 8.6.1向支付宝沙箱发请求请求支付二维码界面

|  |
| --- |
| *//pay 向支付宝沙箱发请求请求支付二维码界面* @RequestMapping("/pay") public String pay(String orderId,String orderPrice,String orderName,String orderAddr,Model model){  *// 获得初始化的AlipayClient* AlipayClient alipayClient = new DefaultAlipayClient(AlipayConfig.*gatewayUrl*, AlipayConfig.*app\_id*,  AlipayConfig.*merchant\_private\_key*, "json", AlipayConfig.*charset*, AlipayConfig.*alipay\_public\_key*,  AlipayConfig.*sign\_type*);  *// 设置请求参数* AlipayTradePagePayRequest alipayRequest = new AlipayTradePagePayRequest();  alipayRequest.setReturnUrl(AlipayConfig.*return\_url*);  alipayRequest.setNotifyUrl(AlipayConfig.*notify\_url*);  *// 商户订单号，商户网站订单系统中唯一订单号，必填* String out\_trade\_no=orderId;  *// 付款金额，必填* String total\_amount=orderPrice;  *// 订单名称，必填* String subject=orderName;  *// 商品描述，可空* String body=orderAddr;  try {  alipayRequest.setBizContent("{\"out\_trade\_no\":\"" + out\_trade\_no + "\"," + "\"total\_amount\":\""  + total\_amount + "\"," + "\"subject\":\"" + subject + "\"," + "\"body\":\"" + body + "\","  + "\"product\_code\":\"FAST\_INSTANT\_TRADE\_PAY\"}");  *// 请求* String result = alipayClient.pageExecute(alipayRequest).getBody();  *// 输出 pay页面取出* model.addAttribute("result", result);  } catch (Exception e) {  e.printStackTrace();  }  *// 这个是渲染到视图中,在pay视图中使用el方式将 result对象数据展示,由result对象进行向阿里支付系统发送构建的请求* return "ticket/alipay"; } |

### 8.6.2支付成功之后的回调（加更新）

|  |
| --- |
| @RequestMapping("/payresult") public String payResult(HttpServletRequest request){  try {  */\*  \* \* 功能：支付宝服务器同步通知页面 日期：2017-03-30 说明：  \* 以下代码只是为了方便商户测试而提供的样例代码，商户可以根据自己网站的需要，按照技术文档编写,并非一定要使用该代码。  \* 该代码仅供学习和研究支付宝接口使用，只是提供一个参考。 页面功能说明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 该页面仅做页面展示，业务逻辑处理请勿在该页面执行  \*/  // 获取支付宝GET过来反馈信息* Map<String, String> params = new HashMap<String, String>();  Map<String, String[]> requestParams = request.getParameterMap();  for (Iterator<String> iter = requestParams.keySet().iterator(); iter.hasNext();) {  String name = (String) iter.next();  String[] values = (String[]) requestParams.get(name);  String valueStr = "";  for (int i = 0; i < values.length; i++) {  valueStr = (i == values.length - 1) ? valueStr + values[i] : valueStr + values[i] + ",";  }  *// 乱码解决，这段代码在出现乱码时使用* valueStr = new String(valueStr.getBytes("ISO-8859-1"), "utf-8");  params.put(name, valueStr);  }  *// 验签* boolean signVerified = AlipaySignature.*rsaCheckV1*(params, AlipayConfig.*alipay\_public\_key*,  AlipayConfig.*charset*, AlipayConfig.*sign\_type*); *// 调用SDK验证签名  // ——请在这里编写您的程序（以下代码仅作参考）——* if (signVerified) {  *// 商户订单号* String out\_trade\_no = new String(request.getParameter("out\_trade\_no").getBytes("ISO-8859-1"), "UTF-8");  *// 支付宝交易号* String trade\_no = new String(request.getParameter("trade\_no").getBytes("ISO-8859-1"), "UTF-8");  *// 付款金额* String total\_amount = new String(request.getParameter("total\_amount").getBytes("ISO-8859-1"), "UTF-8");  *// String body=new  // String(request.getParameter("body").getBytes("ISO-8859-1"),"UTF-8");  //根据订单id 调用业务修改订单号的状态*  *//代码。。。。。  out.println("trade\_no:"+trade\_no+"<br/>out\_trade\_no:"+out\_trade\_no+"<br/>total\_amount:"+total\_amount);* request.setAttribute("reuslt", "trade\_no:" + trade\_no + "<br/>out\_trade\_no:" + out\_trade\_no  + "<br/>total\_amount:" + total\_amount);  } else {  *// out.println("验签失败");* request.setAttribute("reuslt", "支付失败");  }  } catch (Exception e) {  e.printStackTrace();  }  *//转到order\_step4页面* return "ticket/order\_step4"; } |

## 8.7更新订单状态

### 8.7.1编写FlightorderService

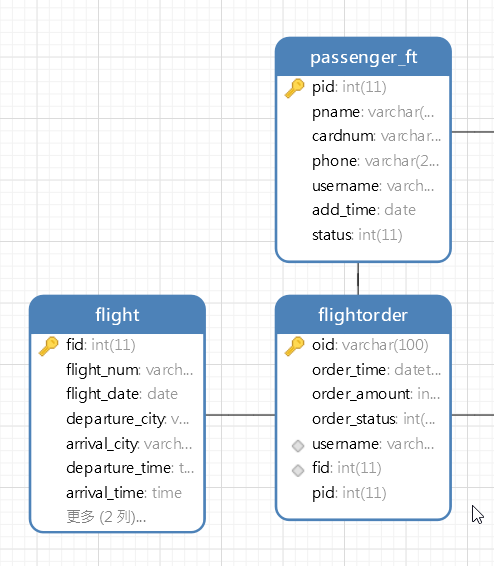
### 8.7.2编写FlightorderMapper

# 功能八：我的订单列表

## 9.1表态原型



## 9.2 数据库原型



## 9.3 编写搜索条件值对象

|  |
| --- |
| public class SearchVoFlightOrder {  private String oid;  private String pname;  private Date flightDate;  } |

## 9.4编写FlightorderMapper接口

|  |
| --- |
| */\*\*  \*多条件查询我的订单列表  \* @param sfo 订单的条件值对象  \* @param username 操作人  \* @return 订单列表  \*/* public List<Flightorder> selectBySearch(@Param("sfo") SearchVoFlightOrder sfo,@Param("username") String username); |

## 9.5编写FlightorderMapper.xml映射文件

|  |
| --- |
| <select id="selectBySearch" resultMap="orderMap">  <include refid="flightOrderField"></include>  <where>  fo.username = #{username}  <if test="sfo.oid!=null and sfo.oid!=''">  and oid = #{sfo.oid}  </if>  <if test="sfo.pname!=null and sfo.pname!=''">  and pname = #{sfo.pname}  </if>  <if test="sfo.flightDate!=null and sfo.flightDate!=''">  and flight\_date = #{sfo.flightDate}  </if>  </where> </select> |
| *<!-- 将一些重复的sql 抽取出来-->* <sql id="flightOrderField">  select  fo.oid, fo.order\_amount, fo.username, fo.order\_time, fo.order\_status, f.fid, f.departure\_city, f.arrival\_city, f.flight\_date, f.departure\_time, f.flight\_num, pf.pname, -- 旅客姓名 pf.cardnum -- 身份证号  from flightorder fo  left join flight f on fo.fid = f.fid  left join passenger\_ft pf on fo.pid = pf.pid  </sql> |
| <resultMap id="orderMap" type="com.sy4.flight.entity.Flightorder">  <id column="oid" property="oid" javaType="java.lang.String"/>  <result column="order\_amount" property="orderAmount" javaType="java.lang.Integer"/>  <result column="username" property="userFt.username" javaType="java.lang.String"/>  <result column="order\_time" property="orderTime" javaType="java.util.Date"/>  <result column="order\_status" property="orderStatus" javaType="java.lang.Integer"/>  <association property="flight" resultMap="com.sy4.flight.mapper.FlightMapper.flightMap"/>  <association property="passengerFt" resultMap="com.sy4.flight.mapper.PassengerFtMapper.passengerFtMap"/> </resultMap> |

## 9.6编写FlightorderService接口

|  |
| --- |
| */\*\*  \*多条件查询我的订单列表  \* @param sfo 订单的条件值对象  \* @param username 操作人  \* @return 订单列表  \*/* public List<Flightorder> selectBySearch(SearchVoFlightOrder sfo, String username); |

## 9.7编写FlightorderServiceImpl实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.*REQUIRED*,readOnly = true) public List<Flightorder> selectBySearch(SearchVoFlightOrder sfo, String username) {  return this.flightorderMapper.selectBySearch(sfo,username); } |

## 9.8编写FlightorderController控制器

|  |
| --- |
| @RequestMapping("/list") public String list(SearchVoFlightOrder sfo,Model model){  List<Flightorder> flightorders = this.flightorderService.selectBySearch(sfo, "Jerry");  model.addAttribute("flightorders",flightorders);  *//转发到订单列表页面视图渲染* return "ticket/my\_order"; } |

## 9.9编写my\_order.jsp页面

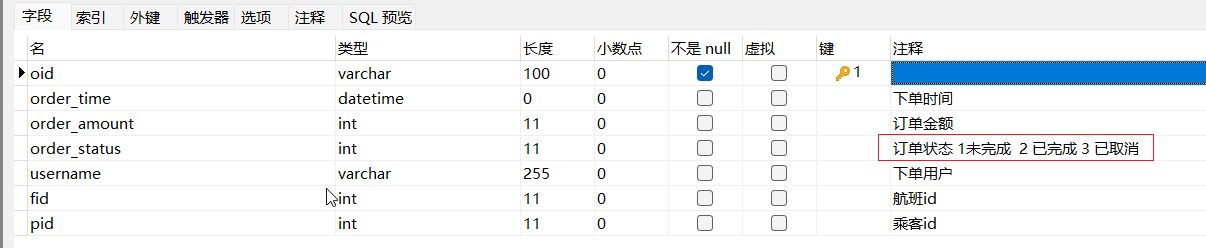
|  |
| --- |
| <form name="f1" id="f1" action="flightorder/list" method="post">  <input type="text" name="oid" placeholder="订单号" value="${param.oid}">  <input type="text" name="pname" value="${param.pname}" placeholder="乘机人">  <input type="date" name="flightDate" value="${param.flightDate}" placeholder="">  <a href="javascript:document.f1.submit()" class="sapar-btn sapar-btn-recom query-btn">查询</a>  </form> |

# 10功能九：退票

## 10.1 静态原型



## 10.2数据库原型



## 10.3修改订单状态

### 10.3.1 编写FlightorderMapper接口

|  |
| --- |
| */\*\*  \* 根据订单id 修改订单状态  \* @param orderId 订单id  \* @param status 订单状态  \*/* public void updateStatus(@Param("orderId") String orderId,@Param("status") Integer status); |

### 10.3.2编写FlightorderMapper.xml文件

|  |
| --- |
| <update id="updateStatus">  update flightorder set order\_status = #{status} where oid = #{orderId} </update> |

## 10.4修改航班余票

### 10.4.1 编写FlightMapper接口

|  |
| --- |
| */\*\*  \* 根据航班id 更新余票  \* @param fid 航班id  \*/* public void updateTicketLeft(Integer fid,Integer subTicket); |

### 10.4.2 编写FlightMapper.xml文件

|  |
| --- |
| */\*\*  \* 根据航班id 更新余票  \* @param fid 航班id  \*/* public void updateTicketLeft(@Param("fid") Integer fid, @Param("subTicket") Integer subTicket); |

## 10.5 编写退票的业务方法

|  |
| --- |
| */\*\*  \* 退票的业务  \* @param orderId 订单id  \* @param fid 航班id  \*/* public void ticketBack(String orderId,Integer fid); |

## 10.6 编写退票的业务实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.*REQUIRED*,rollbackFor = {Exception.class}) public void ticketBack(String orderId, Integer fid) {  *//改余票* this.flightMapper.updateTicketLeft(fid,1);  *//改订单状态* this.flightorderMapper.updateStatus(orderId,3); } |

## 10.7编写FlightorderController控制器

|  |
| --- |
| @RequestMapping("/back") public String backTicket(String orderId,Integer fid){  this.flightorderService.ticketBack(orderId,fid);  return "redirect:/flightorder/list"; } |

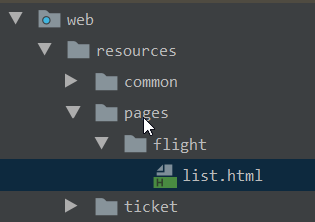
## 10.8修改my\_order.jsp

|  |
| --- |
| <a href="flightorder/back?orderId=${fo.oid}&&fid=${fo.flight.fid}">[退票]</a> |

# 11后台 html+ajax+json

# 功能十:加载航班列表

## 静态页面位置

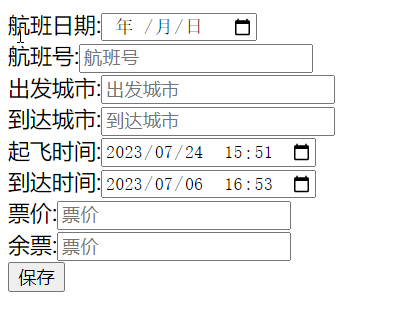


## 编写list.html

|  |
| --- |
| <script src="../../common/js/jquery-3.6.0.min.js"></script> <script>  *//入口函数：* $(function () {  $.ajax({  url:"/admin/flight/list",  type:"get",  data:{},  dataType:"json",  success:function (data) { *//list  //清空tbody中的数据* $('#tbody').children().remove();  *//数据渲染，dom操作，遍历列表* $.each(data,function (index,value) {  *//创建tr td  //添加到tbody中* $('<tr><td>'+value.flightDate+'</td>' +  '<td>'+value.flightNum+'</td>' +  '<td>'+value.departureCity+'</td>' +  '<td>'+value.arrivalCity+'</td>' +  '<td>'+value.departureTime+'</td>' +  '<td>'+value.arrivalTime+'</td>' +  '<td>'+value.ticketPrice+'</td>' +  '<td>'+value.ticketLeft+'</td>' +  '</tr>').appendTo($('#tbody'))  })  }  })  }) </script> |

# 功能十一：新增航班

## 13.1画静态原型



|  |
| --- |
| 航班日期:<input type="date"><br> 航班号:<input type="text" placeholder="航班号"><br>  出发城市:<input type="text" placeholder="出发城市"><br>  到达城市:<input type="text" placeholder="到达城市"><br>  起飞时间:<input type="datetime-local"><br>  到达时间:<input type="datetime-local"><br>  票价:<input type="text" placeholder="票价"><br>  余票:<input type="text" placeholder="票价"><br> <button>保存</button> |

## 13.2 编写js 发送ajax请求

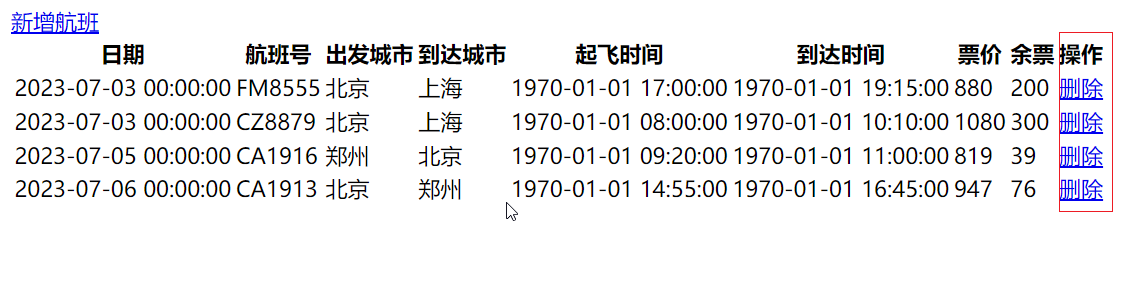
|  |
| --- |
| <script>  $(function () {  *//给保存按钮绑定一个单击事件* $('#btn-save').click(function () {  *//获取输入框对象* let flightDate = $('input[name="flightDate"]');  let flightNum = $('input[name="flightNum"]');  let departureCity = $('input[name="departureCity"]');  let arrivalCity = $('input[name="arrivalCity"]');  let departureTime = $('input[name="departureTime"]');  let arrivalTime = $('input[name="arrivalTime"]');  let ticketPrice = $('input[name="ticketPrice"]');  let ticketLeft = $('input[name="ticketLeft"]');  *//json 格式的参数* let params = {"flightDate":flightDate.val(),  "flightNum":flightNum.val(),  "departureCity":departureCity.val(),  "arrivalCity":arrivalCity.val(),  "departureTime":departureTime.val(),  "arrivalTime":arrivalTime.val(),  "ticketPrice":ticketPrice.val(),  "ticketLeft":ticketLeft.val()  };  $.ajax({  url:"/admin/flight/add",  type:"post",  data:params,  dataType:"json",  success:function(data){  }  })  })  }) </script> |

## 13.3编写控制器

|  |
| --- |
| @RequestMapping("/add") @ResponseBody public ResultVO add(Flight flight){  System.*out*.println(flight);  ResultVO resultVO = new ResultVO();  *//调用业务完成新增* resultVO.setCode(200);  resultVO.setData("新增成功");  return resultVO; } |

# 14.功能十二：删除航班

## 14.1 静态原型



## 14.2编写js

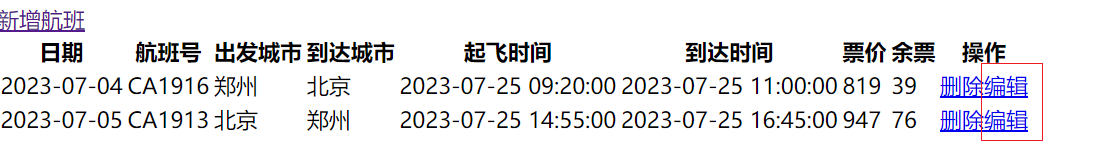
|  |
| --- |
| 引入Jquery |
| <script src="../../common/js/jquery-3.6.0.min.js"></script> |
| 入口函数 |
| *//入口函数：* $(function () {  *//调用list  list*(); }) |
| 初始化数据列表 |
| *//初始化航班列表* function *list*(){  $.ajax({  url:"/admin/flight/list",  type:"get",  data:{},  dataType:"json",  success:function (data) { *//list  //清空tbody中的数据* $('#tbody').children().remove();  *showData*(data);  }  }) } |
| 渲染数据 |
| *//渲染数据* function *showData*(data){  *//数据渲染，dom操作，遍历列表* $.each(data,function (index,value) {  *//创建tr td  //添加到tbody中* $('<tr><td>'+value.flightDate+'</td>' +  '<td>'+value.flightNum+'</td>' +  '<td>'+value.departureCity+'</td>' +  '<td>'+value.arrivalCity+'</td>' +  '<td>'+value.departureTime+'</td>' +  '<td>'+value.arrivalTime+'</td>' +  '<td>'+value.ticketPrice+'</td>' +  '<td>'+value.ticketLeft+'</td>' +  '<td><a href="javascript:del('+value.fid+')">删除</a></td>' +  '</tr>').appendTo($('#tbody'))  }) } |
| *删除的函数* |
| *//这个函数不要放在入口函数中 //1.程序从上往下执行 //2.入口函数是页面加载完成才会执行* function *del*(fid) {  *//确认删除吗？* if(*confirm*("确认删除吗？")){  *// alert(fid)  //发送ajax请求* $.ajax({  url:"/admin/flight/del",  type:"post",  data:{"fid":fid},  dataType: "json",  success:function (data) {  if(data.code==200){  *alert*("删除成功！")  *//1.标准格式  //重新加载当前页面  // location.href="";  // location.reload();  //2.只刷新数据列表  list*();  }  }  })  } } |

## 14.3编写控制器

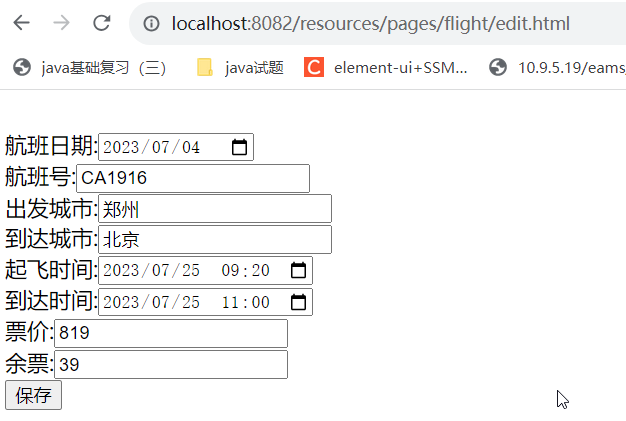
|  |
| --- |
| @RequestMapping("/del") @ResponseBody public ResultVO del(Integer fid){  System.*out*.println(fid);  ResultVO resultVO = new ResultVO();  *//todo:调用业务完成新增* resultVO.setCode(200);  resultVO.setData("删除成功");  return resultVO; } |

# 15功能十三：修改航班

## 15.1 静态原型



## 15.2 数据回显



### 15.2.1 编写edit.html

|  |
| --- |
| list.html |
| '<a href="javascript:edit('+value.fid+')">编辑</a>' + |
| *//编辑* function *edit*(fid) {  *//页面与页面之间数据共享* sessionStorage.setItem("fid",fid)  *//跳转到edit.html* location.href="edit.html" } |
| edit.html |
| $(function () {  let fid = sessionStorage.getItem("fid");  *// alert(fid)* $.ajax({  url:"/admin/flight/getByFid",  type:"get",  data:{"fid":fid},  dataType:"json",  success:function (data) { *//resultVO = {code:200,msg:"",data:{fid:1,flightNum:""..}}* if(data.code == 200){  let fli = data.data;  $('input[name="fid"]').val(fli.fid)  $('input[name="flightDate"]').val(fli.flightDate)  $('input[name="flightNum"]').val(fli.flightNum)  $('input[name="departureCity"]').val(fli.departureCity)  $('input[name="arrivalCity"]').val(fli.arrivalCity)  $('input[name="departureTime"]').val(fli.departureTime)  $('input[name="arrivalTime"]').val(fli.arrivalTime)  $('input[name="ticketPrice"]').val(fli.ticketPrice)  $('input[name="ticketLeft"]').val(fli.ticketLeft)  }  } }) |

### 15.2.2编写AdminFlightController控制器

|  |
| --- |
| @RequestMapping("/getByFid") @ResponseBody public ResultVO getByFid(Integer fid){  System.*out*.println(fid);  ResultVO resultVO = new ResultVO();  *//todo:调用业务完成查询* Flight flight = this.flightService.selectByFid(fid);  resultVO.setCode(200);  resultVO.setMsg("查询成功");  resultVO.setData(flight);  return resultVO; } |

### 15.2.3编写FlightService接口

|  |
| --- |
| */\*\*  \* 根据航班id 查询航班信息  \* @param fid 航班id  \* @return 航班信息  \*/* public Flight selectByFid(Integer fid); |

### 15.2.4编写FlightServiceImpl实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.*REQUIRED*,readOnly = true) public Flight selectByFid(Integer fid) {  return this.flightMapper.selectByFid(fid); } |

### 15.2.5编写FlightMapper接口

|  |
| --- |
| */\*\*  \* 根据航班id 查询航班信息  \* @param fid 航班id  \* @return 航班信息  \*/* public Flight selectByFid(Integer fid); |

### 15.2.6编写FlightMapper.xml映射文件

|  |
| --- |
| <select id="selectByFid" resultMap="flightMap">  select *\** from flight where fid = #{fid} </select> |

## 15.3 修改数据

### 15.3.1编写edit.html

|  |
| --- |
| *edit.html* |
| <form id="f1" action="#" method="post" onsubmit="return false;">  <input type="hidden" name="fid"><br>  航班日期:<input type="date" name="flightDate"><br>  航班号:<input type="text" name="flightNum" value="" placeholder="航班号"><br>  出发城市:<input type="text" name="departureCity" placeholder="出发城市"><br>  到达城市:<input type="text" name="arrivalCity" placeholder="到达城市"><br>  起飞时间:<input type="datetime-local" name="departureTime"><br>  到达时间:<input type="datetime-local" name="arrivalTime"><br>  票价:<input type="text" placeholder="票价" name="ticketPrice"><br>  余票:<input type="text" placeholder="票价" name="ticketLeft"><br> <button type="submit" id="btn-save">保存</button> </form> |
| type="datetime-local":支持时分秒的客户端时间，传递时格式为yyyy-MM-dd'T'HH:mm |
| $('#btn-save').click(function () {  *// 表单项的序列化：flightDate=value&flightNum=value  //表单：阻止表单提交  //表单项得有name 属性，得是post请求  // $('#f1').serialize()* $.ajax({  url:"/admin/flight/edit",  type:"post",  data:$('#f1').serialize(),  dataType:"json",  success:function(data){  if(data.code==200){  location.href="list.html";  }  }  }) }) |
| $('#f1').serialize():表单的序列化 |

### 15.3.2编写FlightMapper接口

|  |
| --- |
| */\*\*  \* 修改航班信息  \* @param flight 新的航班信息  \*/* public void updateByFid(Flight flight); |

### 15.3.3编写FlightMapper.xml文件

|  |
| --- |
| <update id="updateByFid">  update flight set flight\_num = #{flightNum},flight\_date=#{flightDate},departure\_city=#{departureCity},arrival\_city=#{arrivalCity}, departure\_time=#{departureTime},arrival\_time=#{arrivalTime},ticket\_price=#{ticketPrice},ticket\_left=#{ticketLeft} where fid=#{fid}  </update> |

### 15.3.4 编写FlightService接口

|  |
| --- |
| */\*\*  \* 修改航班信息  \* @param flight 新的航班信息  \*/* public void updateByFid(Flight flight); |

### 15.3.5 编写FlightServiceImpl实现类

|  |
| --- |
| @Override @Transactional(propagation = Propagation.*REQUIRED*,rollbackFor =Exception.class) public void updateByFid(Flight flight) {  this.flightMapper.updateByFid(flight); } |

### 15.3.6 编写AdminFlightController控制器

|  |
| --- |
| @RequestMapping("/edit") @ResponseBody public ResultVO edit(Flight flight){  System.*out*.println(flight);  ResultVO resultVO = new ResultVO();  *//todo:调用修改业务完成新增* this.flightService.updateByFid(flight);  resultVO.setCode(200);  resultVO.setData("修改成功");  return resultVO; } |

### 15.3.7 修改实体类

|  |
| --- |
| public class Flight { *// fid int* private Integer fid; *// flight\_num varchar* private String flightNum; *// flight\_date date  //请求参数中有日期（String ==>java.util.Date ）  //响应结果中有日期类型* @DateTimeFormat(pattern = "yyyy-MM-dd")  @JsonFormat(pattern = "yyyy-MM-dd",timezone = "GMT+8")  private Date flightDate; *// departure\_city varchar* private String departureCity; *// arrival\_city varchar* private String arrivalCity; *// departure\_time time 时分秒  //请求参数* @DateTimeFormat(pattern = "yyyy-MM-dd'T'HH:mm")  private Date departureTime; *// arrival\_time time* @DateTimeFormat(pattern = "yyyy-MM-dd'T'HH:mm")  private Date arrivalTime; *// private Timestamp arrivalTime; // ticket\_price int* private Integer ticketPrice; *// ticket\_left int* private Integer ticketLeft;  @DateTimeFormat(pattern = "yyyy-MM-dd'T'HH:mm")  将参数中日期格式的字符串转换成Date类型  "yyyy-MM-dd'T'HH:mm":此格式是前端传入的日期字符串格式 |
| @JsonFormat(pattern = "yyyy-MM-dd",timezone = "GMT+8")  将json结果中日期类型以指定的格式返回,JsonFormat默认时区为GMT,比北京时间早八个小时，需要设置一个时区 |
|  |

## 15.4 日期类型

mysql :年月日时分秒

datetime

timestamp 支持时区 1970-1-1 ：time

实体类中：

Date:

Timestamp

# 16功能十四:审核的功能