目录

1 第六天：门店UD+订单查询

订单CRUD+ResultMap对象关联+下拉框

订单对象和门店对象的关系：一对一
订单对象和订单详情对象的关系：一对多

1.1 新增门店

1、 在列表页面点击【新增】访问toAdd方法，跳转到新增页面doorAdd

2、 在新增页面填写数据，点击【提交】，向服务器发送数据（发起insert语句）
3、 刷新列表

1.1.1 DoorController中添加toAdd

```
//1、在列表页面点击【新增】访问toAdd方法，
    //跳转到新增页面doorAdd
    @RequestMapping("toAdd")
    public String toAdd(){
       return "doorAdd";
    }
```

1.1.2 doorAdd.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>永和大王门店管理系统</title>
</head>
<body style="padding: 20px;">

    <div style="padding: 5px;">
       <h1>永和大王门店管理系统-门店添加</h1>
    </div>

    <form action="add" method="post">
       名称: <input type="text" name="name"/>
       电话: <input type="text" name="tel"/>
       <input type="submit" value="提交">
    </form>
</body>

</html>
```

1.1.3 DoorMapper.xml

```
    <!-- 添加门店 -->
    <insert id="add">
       insert into tb_door values(
       #{id},#{name},#{tel},#{updated},#{created}
       )
    </insert>
```

1.1.4 DoorMapper接口

```
//<!-- 添加门店 -->
    //<insert id="add">
    public void add(Door door);
```

1.1.5 DoorService接口

```
//<!-- 添加门店 -->
    //<insert id="add">
    public void add(Door door);
```

1.1.6 DoorServiceImpl

```
@Override
    public void add(Door door) {
       doorMapper.add(door);
    }
```

1.1.7 DoorController

添加页面点击【提交】，需要调回列表页面，需要在添加方法里修改返回值，变成转发到列表方法。

```
//在新增页面填写数据，点击【提交】，
    //就向服务器发送请求（发起insert语句）
    @RequestMapping("add")
    public String add(Door door){
       doorService.add(door);
       //添加页面点击【提交】，需要刷新列表页面，
       //需要在添加方法里修改返回值，
       //变成了重定向到列表方法，重新查数据做显示。
       return "redirect:list";
    }
```

1.2 修改门店

1、 点击列表页面的【修改】，向服务器发送请求toUpdate方法（**带着记录的id**）
2、 toUpdate方法中，拿着id查询数据，返回doorUpdate页面，在页面做回显
3、 点击【提交】，向服务器发送请求（**执行update语句**）
4、 刷新列表

1.2.1 DoorController添加toUpdate

```
//toUpdate方法中，拿着id查询数据，
    //返回doorUpdate页面，在页面做回显
    @RequestMapping("toUpdate")
    public String toUpdate(
              Integer id,
                  Model model){
       //拿着id查询数据
       Door door =
              doorService.SelectOne(id);

       //给页面准备数据Model
       model.addAttribute("door", door);

       //跳转修改页面
       return "doorUpdate";
    }
```

1.2.2 doorUpdate.jsp

在doorList.jsp中添加修改

```
<%@ page language="java" contentType="text/html; charset=utf-8"
```

```
        pageEncoding="utf-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>永和大王门店管理系统</title>
    </head>
    <body style="padding: 20px;">

        <div style="padding: 5px;">
           <h1>永和大王门店管理系统-门店更新</h1>
        </div>

        <form action="update" method="post">

     <input type="hidden" name="id" value="${door.id}"/>

           名称: <input type="text" name="name" value="${door.name}"/>
           电话: <input type="text" name="tel" value="${door.tel}"/>

           <input type="submit" value="提交">
        </form>
    </body>

    </html>
```

1.2.3 DoorMapper.xml

```
    <!-- 修改门店 now()获取当前时间-->
    <update id="update">
       update tb_door set
           name=#{name},
           tel=#{tel},
           created=now(),
           updated=now()
       where id=#{id}
    </update>
```

1.2.4 DoorMapper接口

```
//<!-- 修改门店 now()获取当前时间-->
    //<update id="update">
    public void update(Door door);
```

1.2.5 DoorService接口

```
//<!-- 修改门店 now()获取当前时间-->
    //<update id="update">
    public void update(Door door);
```

1.2.6 DoorServiceImpl

```
//<!-- 修改门店 now()获取当前时间-->
    @Override
    public void update(Door door) {
       doorMapper.update(door);
```

```
    }
```

1.2.7 DoorController

```
//<!-- 修改门店 -->
    @RequestMapping("update")
    public String update(Door door){
        //更新数据
        doorService.update(door);

        //跳转到列表(访问列表方法)
        return "redirect:list";
    }
```

1.3 删除门店

1、 获取到当前记录的id，向服务器发送请求
2、 执行delete的SQL，返回的不是页面，而是刷新列表

1.3.1 DoorMapper.xml

```
<!-- 删除门店 -->
    <delete id="delete">
        delete from tb_door
        where id=#{id}
    </delete>
```

1.3.2 DoorMapper接口

```
//<!-- 删除门店 -->
    //<delete id="delete">
    public void delete(Integer id);
```

1.3.3 DoorService接口

```
//<!-- 删除门店 -->
    //<delete id="delete">
    public void delete(Integer id);
```

1.3.4 DoorServiceImpl

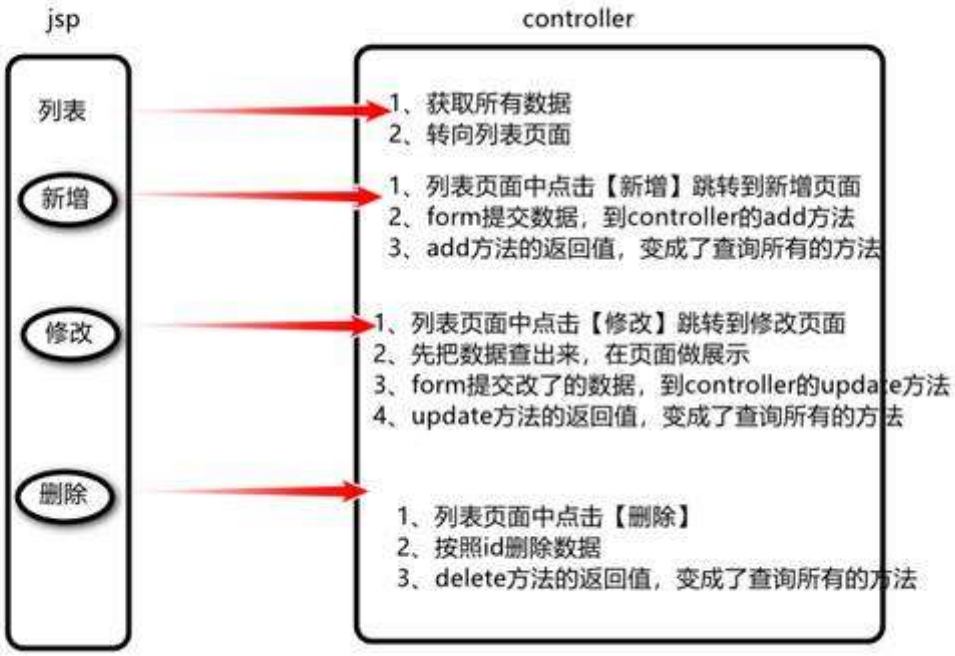
```
//删除门店
    @Override
    public void delete(Integer id) {
        doorMapper.delete(id);
    }
```

1.3.5 DoorController

```
    //删除门店
    @RequestMapping("delete")
    public String delete(Integer id){
        //删除门店
        doorService.delete(id);
```

```
        //跳转到列表方法（执行列表方法）
        return "redirect:list";
    }
```

总结



1.4 ====订单管理

1.5 查询所有订单

永和大王门店管理系统-订单管理

新增

序号	订单号	类型	人数	收银员	下单时间	结账时间	支付类型	价格	操作
1	P000011	堂食	1	张静	2018-04-26 14:49:07	2018-04-17 07:24:38	微支付	8.0	修改 删除 查看
2	P000001	堂食	4	王强	2018-04-24 16:05:00	2018-04-24 16:05:00	支付宝	18.0	修改 删除 查看

1.5.1 Order对象

```
package cn.tedu.pojo;

import java.util.Date;

//描述订单表
public class Order extends BasePojo{
```

```
    //id字段
    private Integer id;

    //door_id字段
    private Integer doorId;

    //order_no字段
    private String orderNo;

    //order_type字段
    private String orderType;

    //person_num字段
    private Integer personNum;

    //cashier字段
    private String cashier;

    //create_time字段
    private Date createTime;

    //end_time字段
    private Date endTime;

    //payment_type字段
    private String paymentType;

    //price字段
    private Double price;

    //getters and setters

    public Integer getId() {
       return id;
    }

    public void setId(Integer id) {
       this.id = id;
    }

    public Integer getDoorId() {
       return doorId;
    }

    public void setDoorId(Integer doorId) {
       this.doorId = doorId;
    }

    public String getOrderNo() {
       return orderNo;
    }

    public void setOrderNo(String orderNo) {
       this.orderNo = orderNo;
    }

    public String getOrderType() {
       return orderType;
    }
```

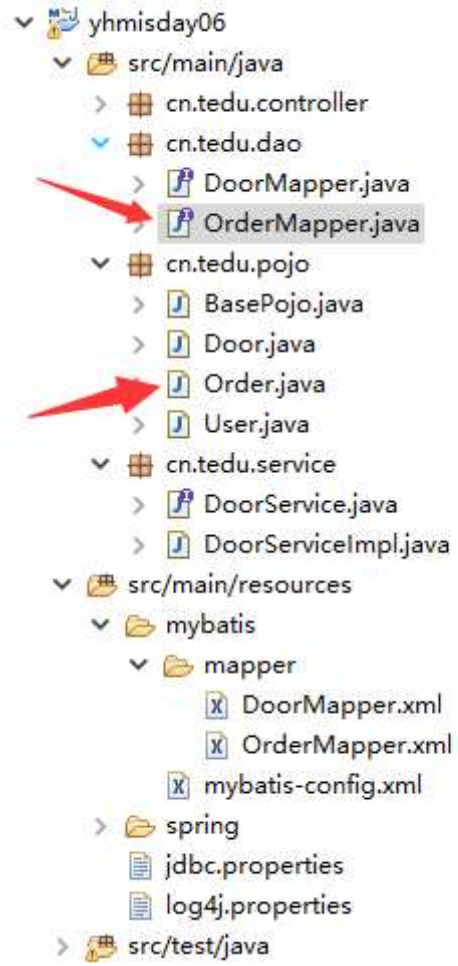
```
    public void setOrderType(String orderType) {
        this.orderType = orderType;
    }

    public Integer getPersonNum() {
        return personNum;
    }

    public void setPersonNum(Integer personNum) {
        this.personNum = personNum;
    }

    public String getCashier() {
        return cashier;
    }

    public void setCashier(String cashier) {
        this.cashier = cashier;
    }

    public Date getCreateTime() {
        return createTime;
    }

    public void setCreateTime(Date createTime) {
        this.createTime = createTime;
    }

    public Date getEndTime() {
        return endTime;
    }

    public void setEndTime(Date endTime) {
        this.endTime = endTime;
    }

    public String getPaymentType() {
        return paymentType;
    }

    public void setPaymentType(String paymentType) {
        this.paymentType = paymentType;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }


    //toString

    @Override
    public String toString() {
        return "Order [id=" + id + ", doorId=" + doorId + ", orderNo=" + orderNo + ",
orderType=" + orderType
                + ", personNum=" + personNum + ", cashier=" + cashier + ", createTime=" +
createTime + ", endTime="
```

```
                + endTime + ", paymentType=" + paymentType + ", price=" + price + "]";
    }




}
```

1.5.2 OrderMapper.xml

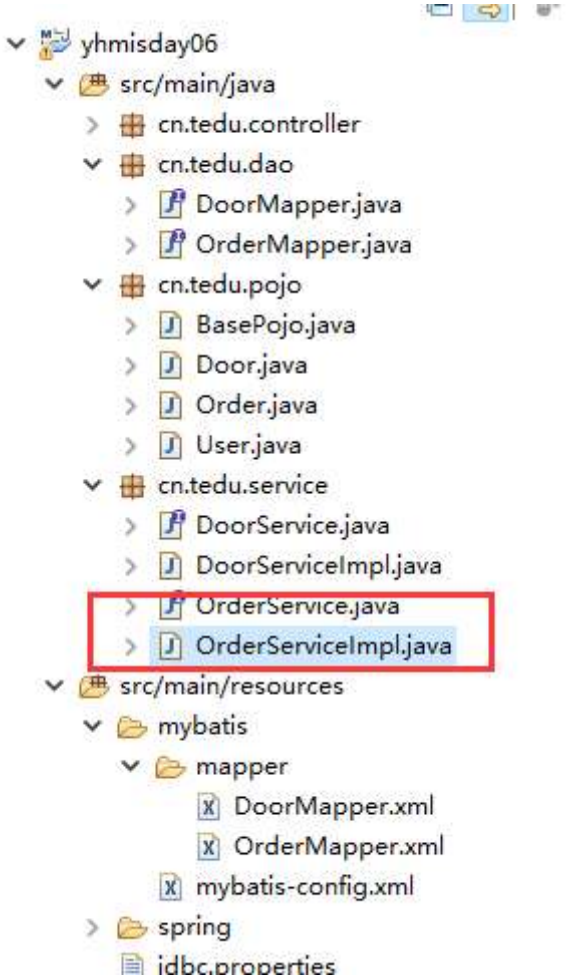
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<!-- 订单的映射文件
namespace：接口的全路径
-->
<mapper namespace="cn.tedu.dao.OrderMapper">

    <!-- 查询所有订单
       resultType代表结果集要封装给谁
    -->
    <select id="SelectAll"
           resultType="cn.tedu.pojo.Order">
       select * from tb_order
    </select>



</mapper>
```

1.5.3 OrderMapper接口




```java
package cn.tedu.dao;

import java.util.List;

import cn.tedu.pojo.Order;

public interface OrderMapper {

    //查询所有订单
    //<select id="SelectAll"
    //resultType="cn.tedu.pojo.Order">
    public List<Order> SelectAll();



}
```

1.5.4 OrderService接口




```java
package cn.tedu.service;

import java.util.List;

import cn.tedu.pojo.Order;

public interface OrderService {

    //查询所有订单
    //<select id="SelectAll"
    //resultType="cn.tedu.pojo.Order">
    public List<Order> SelectAll();

}
```

1.5.5 OrderServiceImpl.java

```java
package cn.tedu.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import cn.tedu.dao.OrderMapper;
import cn.tedu.pojo.Order;

//作用1：交给spring管理对象
//作用2：代表是业务层代码
@Service
public class OrderServiceImpl implements OrderService {
```

```
    @Autowired//自动注入dao层
    private OrderMapper orderMapper;


    //查询所有订单
    @Override
    public List<Order> SelectAll() {
       //调用dao层干活
       return orderMapper.SelectAll();
    }

}
```

1.5.6 OrderController.java

```
package cn.tedu.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import cn.tedu.pojo.Order;
import cn.tedu.service.OrderService;

@Controller
@RequestMapping("order")
public class OrderController {

    @Autowired //自动注入service层
    private OrderService orderService;

    //查询所有订单
    @RequestMapping("list")
    public String list(Model model){
       //查询所有数据
       List<Order> list =
              orderService.SelectAll();

       //给页面准备数据
       model.addAttribute("orderList",list);

       //跳转页面
       return "order";

    }



}
```

1.5.7 order.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
       <title>永和大王门店管理系统</title>
    </head>
    <body style="padding: 20px;">
       <div style="padding: 5px;">
           <h1>永和大王门店管理系统-订单管理</h1>
       </div>
       <a href="toAdd">新增</a>
       <table border="1" cellspacing="0">
           <tr align="center">
              <th>序号</th>
              <th>订单号</th>
              <th>类型</th>
              <th>人数</th>
              <th>收银员</th>
              <th>下单时间</th>
              <th>结账时间</th>
              <th>支付类型</th>
              <th>价格</th>
              <th>操作</th>
           </tr>
           <c:forEach items="${orderList}" var="o" varStatus="i">
              <tr>
                  <td align="center">${i.index+1}</td>
                  <td>${o.orderNo}</td>
                  <td align="center">${o.orderType}</td>
                  <td align="center">${o.personNum}</td>
                  <td align="center">${o.cashier}</td>
                  <td align="center"><fmt:formatDate value="${o.createTime}"
                         pattern="yyyy-MM-dd HH:mm:ss" /></td>
                  <td align="center"><fmt:formatDate value="${o.endTime}"
                         pattern="yyyy-MM-dd HH:mm:ss" /></td>
                  <td align="center">${o.paymentType}</td>
                  <td align="right">${o.price}</td>
                  <td>
                     <a href="toUpdate?id=${o.id}">修改</a>
                     <a href="delete?id=${o.id}">删除</a>
                  </td>
              </tr>
           </c:forEach>
       </table>
    </body>
</html>
```

1.5.8 ！！！问题：数据没封装成功，需要使用resultMap

由于实体中的属性名和表中的字段名不一致，导致无法使用resultType来封装，这时需要使用resultMap来解决。

```
<?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">

    <!-- 订单的映射文件
    namespace：接口的全路径
    -->
    <mapper namespace="cn.tedu.dao.OrderMapper">

        <!-- 映射不规则字段
           type：要把数据封装给哪个对象
           id：该resultMap的唯一标志
        -->
        <resultMap type="cn.tedu.pojo.Order"
                   id="OrderRM">

    <!-- id属性 column是字段名  property是属性名-->
           <id column="id" property="id"/>

    <!-- 其他属性 column是字段名  property是属性名-->
           <result column="door_id" property="doorId"/>
           <result column="order_no" property="orderNo"/>
           <result column="order_type" property="orderType"/>
           <result column="person_num" property="personNum"/>
           <result column="cashier" property="cashier"/>
           <result column="create_time" property="createTime"/>
           <result column="end_time" property="endTime"/>
           <result column="payment_type" property="paymentType"/>
           <result column="price" property="price"/>
           <result column="created" property="created"/>
           <result column="updated" property="updated"/>
        </resultMap>

        <!-- 查询所有订单
           resultMap：引用一个id值
        -->
        <select id="SelectAll"
                resultMap="OrderRM">
           select * from tb_order
        </select>



    </mapper>
```

1.6 拓展

1.6.1 配置别名

在applicationContext-mybatis.xml中配置别名包，就会把类名作为别名：

```
    <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
```

```
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-
tx-4.0.xsd
        http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-4.0.xsd">

        <!-- 配置sqlsessionfactory -->
        <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
           <property name="configLocation" value="classpath:mybatis/mybatis-config.xml">
</property>
           <property name="dataSource" ref="dataSource"></property>

           <!-- 扫描mapper -->
           <property name="mapperLocations" value="classpath:mybatis/mapper/*.xml">
</property>

           <!-- 配置别名包 -->
    <property name="typeAliasesPackage" value="cn.tedu.pojo"></property>

        </bean>

        <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
           <property name="basePackage" value="cn.tedu.dao"></property>
        </bean>

    </beans>
```

1.6.1 中文乱码

在jsp中的form中加属性accept-charset=”utf-8”
在web.xml中添加字符集过滤器:

```
<filter>
<filter-name>characterEncodingFilter</filter-name>
<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
<init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
</init-param>
<init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
</init-param>
</filter>

<filter-mapping>
    <filter-name>characterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

1.6.2 查询关联的门店对象

1.6.2.1 修改Order实体类

1.6.2.2 修改OrderMapper.xml

```
<?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">
```