本章学习目标

- CRM 开发环境搭建
- 客户列表展示
- 客户分页显示
- 客户添加
- 客户信息修改回显
- 客户信息更新保存
- 客户信息删除

1. 客户列表展示

1.1. Mapper 接口

```
package cn.sm1234.dao;


import java.util.List;


import cn.sm1234.domain.Customer;


public interface CustomerMapper {


    /**
     * 查询所有数据
     */
    public List<Customer> findAl();
}
```

1.2. Sql 映射配置

```
<?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">

<!-- 该文件编写 mybatis 中的 mapper 接口里面的方法提供对应的 sql 语句 -->

<mapper namespace="cn.sm1234.dao.CustomerMapper">


    <!-- 查询所有数据 -->

    <select id="findAl" resultType="cn.sm1234.domain.Customer">

        SELECT  id,

            NAME,

            gender,

            telephone,

            address

            FROM

        ssm.t_customer

    </select>


</mapper>
```

1.3. Service

接口：

```
package cn.sm1234.service;


import java.util.List;
```

```
import cn.sm1234.domain.Customer;



public interface CustomerService {

    /**
     * 查询所有数据
     */
    public List<Customer> findAll();
}
```

实现：

```
package cn.sm1234.service.impl;

import java.util.List;

import javax.annotation.Resource;

import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import cn.sm1234.dao.CustomerMapper;
import cn.sm1234.domain.Customer;
import cn.sm1234.service.CustomerService;

@Service("customerService")
@Transactional
public class CustomerServiceImpl implements CustomerService {
```

```
    //注入 Mapper 接口对象
    @Resource
    private CustomerMapper customerMapper;

    public List<Customer> findAll() {
        return customerMapper.findAl();
    }

}
```

1.4. Controller

```
package cn.sm1234.controller;

import java.util.List;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import cn.sm1234.domain.Customer;
import cn.sm1234.service.CustomerService;

@Controller
@RequestMapping("/customer")
public class CustomerController {
```

```
    //注入 service 对象
    @Resource
    private CustomerService customerService;

    /**
     * 查询所有数据，给页面返回 json 格式数据
     * easyui 的 datagrid 组件，需要展示数提供 json 数据：
[ {id:1,name:xxx},{id:1,name:xxx} ]
     */
    @RequestMapping("/list")
    @ResponseBody  // 用于转换对象为 json
    public List<Customer> list(){
        //查询数据
        List<Customer> list = customerService.findAll();
        return list;
    }
}
```

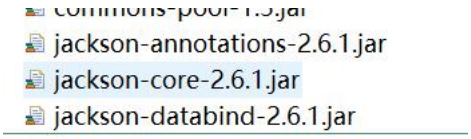
这时可以运行项目测试，发现错误：

```
r. NoSuchMethodError: com.fasterxml.jackson.databind.ObjectWriter.forType(Lcom/fasterxml/jackson/databind/JavaType;)Lcom/fasterxml/jackson/databind/ObjectWriter;
org.springframework.http.converter.json.AbstractJackson2HttpMessageConverter.writeInternal(AbstractJackson2HttpMessageConverter.java:265)
org.springframework.http.converter.AbstractGenericHttpMessageConverter.write(AbstractGenericHttpMessageConverter.java:100)
org.springframework.web.servlet.mvc.method.annotation.AbstractMessageConverterMethodProcessor.writeWithMessageConverters(AbstractMessageConverterMethodProcessor.jav
org.springframework.web.servlet.mvc.method.annotation.RequestResponseBodyMethodProcessor.handleReturnValue(RequestResponseBodyMethodProcessor.java:173)
org.springframework.web.method.support.HandlerMethodReturnValueHandlerComposite.handleReturnValue(HandlerMethodReturnValueHandlerComposite.java:81)
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:130)
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:827)
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:738)
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:85)
org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:963)
org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:897)
```

原因：SpringMVC 在转换 Java 对象为 json 数据的时候，jackson 插件的版本太低，导致转换失败。

解决办法：

升级 jackson 插件的版本，最好升级都 2.6 以上



1.5. 页面

在页面导入 easyui 的资源文件：

```
<!-- 导入 easyui 的资源文件 -->

    <script type="text/javascript" src="easyui/jquery.min.js"></script>

    <script type="text/javascript" src="easyui/jquery.easyui.min.js"></script>

    <script type="text/javascript" src="easyui/locale/easyui-lang-zh_CN.js"></script>

    <link rel="stylesheet" type="text/css" href="easyui/themes/icon.css">

    <link id="themeLink" rel="stylesheet" type="text/css" href="easyui/themes/default/easyui.css">
```

```
<table id="list"></table>


    <script type="text/javascript">
        $(function(){
            $("#list").datagrid({
                //url:后台数据查询的地址
                url:"customer/list.action",
                //columns：填充的列数据
                    //field:后台对象的属性
                    //tille:列标题
                columns:[[
                    {
                        field:"id",
                        title:"客户编号",
                        width:100,
```

```
                    checkbox:true
                },
                {
                    field:"name",
                    title:"客户姓名",
                    width:200
                },
                {
                    field:"gender",
                    title:"客户性别",
                    width:200
                },
                {
                    field:"telephone",
                    title:"客户手机",
                    width:200
                },
                {
                    field:"address",
                    title:"客户住址",
                    width:200
                }
            ]]
        });
    });

</script>
```

2. 客户分页显示

2.1. 页面

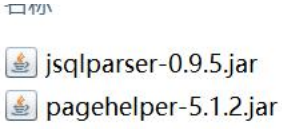
```
$(function(){
            $("#list").datagrid({
                //url:后台数据查询的地址
                url:"customer/listByPage.action",
                //columns：填充的列数据
                    //field:后台对象的属性
                    //tille:列标题
                columns:[[
                    {
                        field:"id",
                        title:"客户编号",
                        width:100,
                        checkbox:true
                    },
                    {
                        field:"name",
                        title:"客户姓名",
                        width:200
                    },
                    {
                        field:"gender",
                        title:"客户性别",
                        width:200
                    },
                    {
```

```
                    field:"telephone",
                    title:"客户手机",
                    width:200
                },
                {
                    field:"address",
                    title:"客户住址",
                    width:200
                }
            ]],
            //显示分页
            pagination:true
        });
    });
```

2.2. Controller

2.2.1. 使用 mybatis 分页插件

2.2.1.1. 导入 mybatis 分页插件的 jar 包



2.2.1.2. 配置 applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
```

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

    <!-- 读取 jdbc.properties -->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!-- 创建 DataSource -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="url" value="${jdbc.url}"/>
        <property name="driverClassName" value="${jdbc.driverClass}"/>
        <property name="username" value="${jdbc.user}"/>
        <property name="password" value="${jdbc.password}"/>
        <property name="maxActive" value="10"/>
        <property name="maxIdle" value="5"/>
    </bean>

    <!-- 创建 SqlSessionFactory 对象 -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 关联连接池 -->
```

```
        <property name="dataSource" ref="dataSource"/>
        <!-- 加载 sql 映射文件 -->
        <property name="mapperLocations" value="classpath:mapper/*.xml"/>
        <!-- 引入插件 -->
        <property name="plugins">
            <array>
                <!-- mybatis 分页插件 -->
                <bean class="com.github.pagehelper.PageInterceptor">
                    <property name="properties">
                        <!--
                        helperDialect:连接数据库的类型
                         -->
                        <value>
                            helperDialect=mysql
                        </value>
                    </property>
                </bean>
            </array>
        </property>
    </bean>

    <!-- Mapper 接口的扫描 -->
    <!--
        注意：如果使用 Mapper 接口包扫描，那么每个 Mapper 接口在 Spring 容器中的 id 名称为类名： 例如 CustomerMapper -> customerMapper
     -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 配置 Mapper 接口所在包路径  -->
        <property name="basePackage" value="cn.sm1234.dao"/>
```

```
    </bean>


    <!-- 开启 Spring 的 IOC 注解扫描 -->
    <context:component-scan base-package="cn.sm1234"/>


    <!-- 开启 Spring 的事务 -->
    <!-- -事务管理器 -->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"/>
    </bean>
    <!-- 启用 Spring 事务注解 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>


</beans>
```

```
package cn.sm1234.controller;


import java.util.HashMap;
import java.util.List;
import java.util.Map;


import javax.annotation.Resource;


import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;


import cn.sm1234.domain.Customer;
```

```
import cn.sm1234.service.CustomerService;


import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;


@Controller
@RequestMapping("/customer")
public class CustomerController {


    //注入 service 对象
    @Resource
    private CustomerService customerService;


    /**
     * 查询所有数据，给页面返回 json 格式数据
     * easyui 的 datagrid 组件，需要展示数提供 json 数据：
[ {id:1,name:xxx},{id:1,name:xxx} ]
     */
    @RequestMapping("/list")
    @ResponseBody  // 用于转换对象为 json
    public List<Customer> list(){
        //查询数据
        List<Customer> list = customerService.findAll();
        return list;
    }


    //设计 Map 聚合存储需要给页面的对象数据
    private Map<String,Object> result = new HashMap<String,Object>();
```

```
    /**
     * 分页查询
     */
    @RequestMapping("/listByPage")
    @ResponseBody
    public Map<String,Object> listByPage(Integer page,Integer rows){
        //设置分页参数
        PageHelper.startPage(page, rows);
        //查询所有数据
        List<Customer> list = customerService.findAll();

        //使用 PageInfo 封装查询结果
        PageInfo<Customer> pageInfo = new PageInfo<Customer>(list);

        //从 PageInfo 对象取出查询结果
        //总记录数
        long total = pageInfo.getTotal();
        //当前页数据列表
        List<Customer> custList = pageInfo.getList();

        result.put("total", total);
        result.put("rows", custList);

        return result;
    }
}
```

3. 客户添加

3.1. 页面

- 设计一个工具条：

```
    <!-- 工具条 -->
    <div id="tb">
        <a id="addBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-add',plain:true">添加</a>
        <a id="editBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-edit',plain:true">修改</a>
        <a id="deleteBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-remove',plain:true">删除</a>
    </div>
```

在 datagrid 上面绑定：

```
                },
                {
                    field:"address",
                    title:"客户住址",
                    width:200
                }
            ]],
            //显示分页
            pagination:true,
            //工具条
            toolbar:"#tb"
        });
    });
</script>
```

- 设计一个录入窗口

```
<!-- 编辑窗口 -->
    <div id="win" class="easyui-window" title="客户数据编辑"
style="width:500px;height:300px"
        data-options="iconCls:'icon-save',modal:true,closed:true">
         <form method="post">
              客户姓名：<input type="text" name="name" class="easyui-validatebox"
data-options="required:true"/><br/>
              客户性别：
              <input type="radio" name="gender" value="男"/>男
              <input type="radio" name="gender" value="女"/>女
              <br/>
              客户手机：<input type="text" name="telephone" class="easyui-validatebox"
data-options="required:true"/><br/>
              客户住址：<input type="text" name="address" class="easyui-validatebox"
data-options="required:true"/><br/>
         <a id="saveBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-save'">保存</a>
     </form>
    </div>
```

- 点击按钮，打开窗口

```
            //打开编辑窗口
            $("#addBtn").click(function(){
                $("#win").window("open");
            });
```

- 提交表单数据

```
//保存数据
            $("#saveBtn").click(function(){
                $("#editForm").form("submit",{
                    //url: 提交到后台的地址
                    url:"customer/save.action",
                    //onSubmit: 表单提交前的回调函数，true：提交表单   false：不提交表
单
                    onSubmit:function(){
                        //判断表单的验证是否都通过了
                        return $("#editForm").form("validate");
                    },
                    //success:服务器执行完毕回调函数
                    success:function(data){ //data: 服务器返回的数据，类型字符串类
                        //要求 Controller 返回的数据格式：
                        //成功：{success:true} 失败:{success:false,msg:错误信息}

                        //把 data 字符串类型转换对象类型
                        data = eval("("+data+")");

                        if(data.success){
                            $.messager.alert("提示","保存成功","info");
                        }else{
                            $.messager.alert("提示","保存失败："+data.msg,"error");
                        }
                    }
                });
```

```
        });
```

3.2. Controller

```
    /**
     * 保存数据
     */
    @RequestMapping("/save")
    @ResponseBody
    public Map<String,Object> save(Customer customer){
        try {
            customerService.save(customer);
            result.put("success", true);
        } catch (Exception e) {
            e.printStackTrace();
            result.put("success", false);
            result.put("msg", e.getMessage());
        }
        return result;
    }
```

3.3. Service

接口：

```
public void save(Customer customer);
```

实现：

```
public void save(Customer customer) {
```

```
        customerMapper.save(customer);
    }
```

3.4. Mapper

```
package cn.sm1234.dao;


import java.util.List;


import cn.sm1234.domain.Customer;


public interface CustomerMapper {


    /**
     * 查询所有数据
     */
    public List<Customer> findAl();


    /**
     * 保存数据
     * @param customer
     */
    public void save(Customer customer);
}
```

3.5. sql 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">

<!-- 该文件编写 mybatis 中的 mapper 接口里面的方法提供对应的 sql 语句 -->

<mapper namespace="cn.sm1234.dao.CustomerMapper">


    <!-- 查询所有数据 -->

    <select id="findAl" resultType="cn.sm1234.domain.Customer">

        SELECT  id,

            NAME,

            gender,

            telephone,

            address

            FROM

        ssm.t_customer

    </select>


    <!-- 添加客户 -->

    <insert id="save" parameterType="cn.sm1234.domain.Customer">

        INSERT INTO ssm.t_customer

            (

            NAME,

            gender,

            telephone,

            address

            )
```

```
            VALUES
            (
            #{name},
            #{gender},
            #{telephone},
            #{address}
            )
    </insert>


</mapper>
```

4. 客户修改的数据回显

4.1. 页面

```
//修改数据
            $("#editBtn").click(function(){
                //判断只能选择一行
                var rows = $("#list").datagrid("getSelections");
                if(rows.length!=1){
                    $.messager.alert("提示","修改操作只能选择一行","warning");
                    return;
                }

                //表单回显

    $("#editForm").form("load","customer/findById.action?id="+rows[0].id);

                $("#win").window("open");
```

```
        });
```

4.2. Controller

```
    /**
     * 根据 id 查询对象
     */
    @RequestMapping("/findById")
    @ResponseBody
    public Customer findById(Integer id){
        Customer cust = customerService.findById(id);
        return cust;
    }
```

4.3. Service

接口：

```
public Customer findById(Integer id);
```

实现：

```
public Customer findById(Integer id) {
        return customerMapper.findById(id);
    }
```

4.4. Mapper

```
/**
     * 根据 id 查询对象
```

```
     * @param id
     * @return
     */
    public Customer findById(Integer id);
```

4.5. sql 映射文件

```
<!-- 根据 id 查询对象 -->
    <select id="findById" parameterType="int" resultType="cn.sm1234.domain.Customer">
        SELECT  id,
            NAME,
            gender,
            telephone,
            address
            FROM
        ssm.t_customer
            where id = #{value}
    </select>
```

5. 客户修改的更新保存

5.1. 页面

```
 <form id="editForm" method="post">
        <%--提供 id 隐藏域 --%>
        <input type="hidden" name="id">
            客户姓名：<input type="text" name="name" class="easyui-validatebox"
data-options="required:true"/><br/>
            客户性别：
```

```
            <input type="radio" name="gender" value="男"/>男
            <input type="radio" name="gender" value="女"/>女
            <br/>
            客户手机：<input type="text" name="telephone" class="easyui-validatebox"
data-options="required:true"/><br/>
            客户住址：<input type="text" name="address" class="easyui-validatebox"
data-options="required:true"/><br/>
        <a id="saveBtn" href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-save'">保存</a>
    </form>
```

5.2. Controller

无

5.3. Service

```
public void save(Customer customer) {
        //判断是添加还是修改
        if(customer.getId()!=null){
            //修改
            customerMapper.update(customer);
        }else{
            //增加
            customerMapper.save(customer);
        }
    }
```

5.4. Mapper

```
/**
     * 修改对象数据
     * @param customer
     */
    public void update(Customer customer);
```

5.5. sql 映射文件

```
    <!-- 根据 id 修改数据 -->
    <update id="update" parameterType="cn.sm1234.domain.Customer">
        UPDATE ssm.t_customer
            SET
            NAME = #{name} ,
            gender = #{gender} ,
            telephone = #{telephone} ,
            address = #{address}
        WHERE
            id = #{id}
    </update>
```

6. 客户删除

6.1. 页面

```
//删除
            $("#deleteBtn").click(function(){
                var rows =$("#list").datagrid("getSelections");
```

```
                if(rows.length==0){
                    $.messager.alert("提示","删除操作至少选择一行","warning");
                    return;
                }

                //格式:  id=1&id=2&id=3
                $.messager.confirm("提示","确认删除数据吗?",function(value){
                    if(value){
                        var idStr = "";
                        //遍历数据
                        $(rows).each(function(i){
                            idStr+=("id="+rows[i].id+"&");
                        });
                        idStr = idStr.substring(0,idStr.length-1);

                        //传递到后台
                        $.post("custmer/delete.action",idStr,function(data){
                            if(data.success){
                                //刷新 datagrid
                                $("#list").datagrid("reload");

                                $.messager.alert("提示","删除成功","info");
                            }else{
                                $.messager.alert("提示","删除失败:
"+data.msg,"error");
                            }
                        },"json");
                    }
                });
```

```
            });
```

6.2. Controller

```
/**
     * 删除数据
     */
    @RequestMapping("/delete")
    @ResponseBody
    public Map<String,Object> delete(Integer[] id){
        try {
            customerService.delete(id);
            result.put("success", true);
        } catch (Exception e) {
            e.printStackTrace();
            result.put("success", false);
            result.put("msg", e.getMessage());
        }
        return result;
    }
```

6.3. Service

接口：

```
public void delete(Integer[] id);
```

实现：

```
public void delete(Integer[] id) {
```

```
        customerMapper.delete(id);
    }
```

6.4. Mapper

```
/**
     * 删除数据
     * @param id
     */
    public void delete(Integer[] id);
```

6.5. sql 映射文件

```
<!-- 删除 -->
    <delete id="delete" parameterType="Integer[]">
    DELETE FROM ssm.t_customer
        <where>
            id
            <foreach collection="array" item="id" open="in (" close=")" separator=",">
                #{id}
            </foreach>
        </where>
    </delete>
```