# 第3章 预约管理-检查组管理

## 1. 需求分析

检查组其实就是多个检查项的集合，例如有一个检查组为“一般检查”，这个检查组可以包括多个检查项：身高、体重、收缩压、舒张压等。所以在添加检查组时需要选择这个检查组包括的检查项。

检查组对应的实体类为CheckGroup，对应的数据表为t_checkgroup。检查组和检查项为多对多关系，所以需要中间表t_checkgroup_checkitem进行关联。

## 2. 新增检查组

### 2.1 完善页面

检查组管理页面对应的是checkgroup.html页面，根据产品设计的原型已经完成了页面基本结构的编写，现在需要完善页面动态效果。

#### 2.1.1 弹出新增窗口

页面中已经提供了新增窗口，只是出于隐藏状态。只需要将控制展示状态的属性dialogFormVisible改为true即可显示出新增窗口。点击新建按钮时绑定的方法为handleCreate，所以在handleCreate方法中修改dialogFormVisible属性的值为true即可。同时为了增加用户体验度，需要每次点击新建按钮时清空表单输入项。

由于新增检查组时还需要选择此检查组包含的检查项，所以新增检查组窗口分为两部分信息：基本信息和检查项信息，如下图：


新建按钮绑定单击事件，对应的处理函数为handleCreate

```
<el-button type="primary" class="butT" @click="handleCreate()">新建</el-button>
```

```
// 重置表单
resetForm() {
    this.formData = {};
},
// 弹出添加窗口
handleCreate() {
    this.resetForm();
    this.dialogFormVisible = true;
}
```

## 2.1.2 动态展示检查项列表

现在虽然已经完成了新增窗口的弹出，但是在检查项信息标签页中需要动态展示所有的检查项信息列表数据，并且可以进行勾选。具体操作步骤如下：

（1）定义模型数据

```
tableData:[],//新增和编辑表单中对应的检查项列表数据
checkitemIds:[],//新增和编辑表单中检查项对应的复选框，基于双向绑定可以进行回显和数据提交
```

（2）动态展示检查项列表数据，数据来源于上面定义的tableData模型数据

```
<table class="datatable">
  <thead>
    <tr>
      <th>选择</th>
      <th>项目编码</th>
      <th>项目名称</th>
      <th>项目说明</th>
    </tr>
  </thead>
  <tbody>
    <tr v-for="c in tableData">
      <td>
        <input :id="c.id" v-model="checkitemIds" type="checkbox"
:value="c.id">
      </td>
      <td><label :for="c.id">{{c.code}}</label></td>
      <td><label :for="c.id">{{c.name}}</label></td>
      <td><label :for="c.id">{{c.remark}}</label></td>
    </tr>
  </tbody>
</table>
```

（3）完善handleCreate方法，发送ajax请求查询所有检查项数据并将结果赋值给tableData模型数据用于页面表格展示

```
// 弹出添加窗口
handleCreate() {
  this.dialogFormVisible = true;
  this.resetForm();
  //默认切换到第一个标签页（基本信息）
  this.activeName='first';
  //重置
  this.checkitemIds = [];
  //发送ajax请求查询所有检查项信息
  axios.get("/checkitem/findAll.do").then((res)=> {
    if(res.data.flag){
      //将检查项列表数据赋值给模型数据用于页面表格展示
      this.tableData = res.data.data;
    }else{
      this.$message.error(res.data.message);
    }
  });
}
```

（4）分别在CheckItemController、CheckItemService、CheckItemServiceImpl、CheckItemDao、CheckItemDao.xml中扩展方法查询所有检查项数据

CheckItemController：

```
//查询所有
@RequestMapping("/findAll")
public Result findAll(){
  List<CheckItem> checkItemList = checkItemService.findAll();
  if(checkItemList != null && checkItemList.size() > 0){
    Result result = new Result(true,
MessageConstant.QUERY_CHECKITEM_SUCCESS);
    result.setData(checkItemList);
    return result;
  }
  return new Result(false,MessageConstant.QUERY_CHECKITEM_FAIL);
}
```

CheckItemService：

```
public List<CheckItem> findAll();
```

CheckItemServiceImpl：

```
public List<CheckItem> findAll() {
  return checkItemDao.findAll();
}
```

CheckItemDao：

```
public List<CheckItem> findAll();
```

CheckItemDao.xml：

```
<select id="findAll" resultType="com.itheima.pojo.CheckItem">
  select * from t_checkitem
</select>
```

### 2.1.3 提交请求

当用户点击新增窗口中的确定按钮时发送ajax请求将数据提交到后台进行数据库操作。提交到后台的数据分为两部分：检查组基本信息（对应的模型数据为formData）和检查项id数组（对应的模型数据为checkitemIds）。

为确定按钮绑定单击事件，对应的处理函数为handleAdd

```
<el-button type="primary" @click="handleAdd()">确定</el-button>
```

完善handleAdd方法

```
//添加
handleAdd () {
  //发送ajax请求将模型数据提交到后台处理
  axios.post(
            "/checkgroup/add.do?checkitemIds=" + this.checkitemIds,
            this.formData
            )
    .then((response)=> {
      //关闭新增窗口
      this.dialogFormVisible = false;
      if(response.data.flag){
        //新增成功，弹出成功提示
        this.$message({
          message: response.data.message,
          type: 'success'
        });
      }else{
        //新增失败，弹出错误提示
        this.$message.error(response.data.message);
      }
  }).finally(()=> {
    this.findPage();
  });
}
```

## 2.2 后台代码

### 2.2.1 Controller

在health_backend工程中创建CheckGroupController

```
package com.itheima.controller;
import com.alibaba.dubbo.config.annotation.Reference;
import com.itheima.constant.MessageConstant;
import com.itheima.entity.PageResult;
import com.itheima.entity.QueryPageBean;
import com.itheima.entity.Result;
import com.itheima.pojo.CheckGroup;
import com.itheima.pojo.CheckItem;
import com.itheima.service.CheckGroupService;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;
/**
 * 检查组管理
 */
@RestController
@RequestMapping("/checkgroup")
public class CheckGroupController {
    @Reference
    private CheckGroupService checkGroupService;

    //新增
    @RequestMapping("/add")
    public Result add(@RequestBody CheckGroup checkGroup,Integer[]
checkitemIds){
        try {
            checkGroupService.add(checkGroup,checkitemIds);
        }catch (Exception e){
            //新增失败
            return new Result(false,
MessageConstant.ADD_CHECKGROUP_FAIL);
        }
        //新增成功
        return new Result(true,MessageConstant.ADD_CHECKGROUP_SUCCESS);
    }
}
```

### 2.2.2 服务接口

在health_interface工程中创建CheckGroupService接口

```
package com.itheima.service;
import com.itheima.entity.PageResult;
import com.itheima.pojo.CheckGroup;
import java.util.List;
/**
 * 检查组服务接口
 */
public interface CheckGroupService {
    void add(CheckGroup checkGroup,Integer[] checkitemIds);
}
```

### 2.2.3 服务实现类

在health_service_provider工程中创建CheckGroupServiceImpl实现类

```java
package com.itheima.service;
import com.alibaba.dubbo.config.annotation.Service;
import com.github.pagehelper.Page;
import com.github.pagehelper.PageHelper;
import com.itheima.dao.CheckGroupDao;
import com.itheima.entity.PageResult;
import com.itheima.pojo.CheckGroup;
import com.itheima.pojo.CheckItem;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
/**
 * 检查组服务
 */
@Service(interfaceClass = CheckGroupService.class)
@Transactional
public class CheckGroupServiceImpl implements CheckGroupService {
    @Autowired
    private CheckGroupDao checkGroupDao;

    //添加检查组合，同时需要设置检查组合和检查项的关联关系
    public void add(CheckGroup checkGroup, Integer[] checkitemIds) {
        checkGroupDao.add(checkGroup);
        setCheckGroupAndCheckItem(checkGroup.getId(),checkitemIds);
    }
    //设置检查组合和检查项的关联关系
    public void setCheckGroupAndCheckItem(Integer checkGroupId,Integer[]
checkitemIds){
          if(checkitemIds != null && checkitemIds.length > 0){
              for (Integer checkitemId : checkitemIds) {
                  Map<String,Integer> map = new HashMap<>();
                  map.put("checkgroup_id",checkGroupId);
                  map.put("checkitem_id",checkitemId);
                  checkGroupDao.setCheckGroupAndCheckItem(map);
              }
          }
      }

}
```

### 2.2.4 Dao接口

创建CheckGroupDao接口

```
package com.itheima.dao;
import com.github.pagehelper.Page;
import com.itheima.pojo.CheckGroup;
import java.util.List;
import java.util.Map;
/**
 * 持久层Dao接口
 */
public interface CheckGroupDao {
    void add(CheckGroup checkGroup);
    void setCheckGroupAndCheckItem(Map map);
}
```

### 2.2.5 Mapper映射文件

创建CheckGroupDao.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.itheima.dao.CheckGroupDao">
    <!--新增-->
    <insert id="add" parameterType="com.itheima.pojo.CheckGroup">
        <selectKey resultType="java.lang.Integer" order="AFTER"
keyProperty="id">
            SELECT LAST_INSERT_ID()
        </selectKey>
        insert into t_checkgroup(code,name,sex,helpCode,remark,attention)
            values
        (#{code},#{name},#{sex},#{helpCode},#{remark},#{attention})
    </insert>
    <!--设置检查组和检查项的关联关系-->
    <insert id="setCheckGroupAndCheckItem" parameterType="hashmap">
        insert into t_checkgroup_checkitem(checkgroup_id,checkitem_id)
            values
        (#{checkgroup_id},#{checkitem_id})
    </insert>
</mapper>
```

# 3. 检查组分页

## 3.1 完善页面

### 3.1.1 定义分页相关模型数据

```
pagination: {//分页相关模型数据
  currentPage: 1,//当前页码
  pageSize:10,//每页显示的记录数
  total:0,//总记录数
  queryString:null//查询条件
},
dataList: [],//当前页要展示的分页列表数据
```

### 3.1.2 定义分页方法

在页面中提供了findPage方法用于分页查询，为了能够在checkgroup.html页面加载后直接可以展示分页数据，可以在VUE提供的钩子函数created中调用findPage方法

```
//钩子函数，VUE对象初始化完成后自动执行
created() {
  this.findPage();
}
```

```
//分页查询
findPage() {
  //分页参数
  var param = {
    currentPage:this.pagination.currentPage,//页码
    pageSize:this.pagination.pageSize,//每页显示的记录数
    queryString:this.pagination.queryString//查询条件
  };
  //请求后台
  axios.post("/checkgroup/findPage.do",param).then((response)=> {
    //为模型数据赋值，基于VUE的双向绑定展示到页面
    this.dataList = response.data.rows;
    this.pagination.total = response.data.total;
  });
}
```

### 3.1.3 完善分页方法执行时机

除了在created钩子函数中调用findPage方法查询分页数据之外，当用户点击查询按钮或者点击分页条中的页码时也需要调用findPage方法重新发起查询请求。

为查询按钮绑定单击事件，调用findPage方法

```
<el-button @click="findPage()" class="dalfBut">查询</el-button>
```

为分页条组件绑定current-change事件，此事件是分页条组件自己定义的事件，当页码改变时触发，对应的处理函数为handleCurrentChange

```
<el-pagination
                class="pagiantion"
                @current-change="handleCurrentChange"
                :current-page="pagination.currentPage"
                :page-size="pagination.pageSize"
                layout="total, prev, pager, next, jumper"
                :total="pagination.total">
</el-pagination>
```

定义handleCurrentChange方法

```
//切换页码
handleCurrentChange(currentPage) {
  //currentPage为切换后的页码
  this.pagination.currentPage = currentPage;
  this.findPage();
}
```

## 3.2 后台代码

### 3.2.1 Controller

在CheckGroupController中增加分页查询方法

```
//分页查询
@RequestMapping("/findPage")
public PageResult findPage(@RequestBody QueryPageBean queryPageBean){
  PageResult pageResult = checkGroupService.pageQuery(
    queryPageBean.getCurrentPage(),
    queryPageBean.getPageSize(),
    queryPageBean.getQueryString()
  );
  return pageResult;
}
```

### 3.2.2 服务接口

在CheckGroupService服务接口中扩展分页查询方法

```
public PageResult pageQuery(Integer currentPage, Integer pageSize, String
queryString);
```

### 3.2.3 服务实现类

在CheckGroupServiceImpl服务实现类中实现分页查询方法，基于Mybatis分页助手插件实现分页

```
public PageResult pageQuery(Integer currentPage, Integer pageSize, String
queryString) {
  PageHelper.startPage(currentPage,pageSize);
  Page<CheckItem> page = checkGroupDao.selectByCondition(queryString);
  return new PageResult(page.getTotal(),page.getResult());
}
```

### 3.2.4 Dao接口

在CheckGroupDao接口中扩展分页查询方法

```
public Page<CheckGroup> selectByCondition(String queryString);
```

### 3.2.5 Mapper映射文件

在CheckGroupDao.xml文件中增加SQL定义

```
<select id="selectByCondition" parameterType="string"
resultType="com.itheima.pojo.CheckGroup">
  select * from t_checkgroup
  <if test="value != null and value.length > 0">
    where code = #{value} or name = #{value} or helpCode = #{value}
  </if>
</select>
```

# 4. 编辑检查组

## 4.1 完善页面

用户点击编辑按钮时，需要弹出编辑窗口并且将当前记录的数据进行回显，用户修改完成后点击确定按钮将修改后的数据提交到后台进行数据库操作。此处进行数据回显的时候，除了需要检查组基本信息的回显之外，还需要回显当前检查组包含的检查项（以复选框勾选的形式回显）。

### 4.1.1 绑定单击事件

需要为编辑按钮绑定单击事件，并且将当前行数据作为参数传递给处理函数

```
<el-button type="primary" size="mini" @click="handleUpdate(scope.row)">编辑</el-button>
```

```
handleUpdate(row) {
  alert(row);
}
```

### 4.1.2 弹出编辑窗口回显数据

当前页面的编辑窗口已经提供好了，默认处于隐藏状态。在handleUpdate方法中需要将编辑窗口展示出来，并且需要发送多个ajax请求分别查询当前检查组数据、所有检查项数据、当前检查组包含的检查项id用于基本数据回显

```
handleUpdate(row) {
  //发送ajax请求根据id查询检查组信息，用于基本信息回显
  axios.get("/checkgroup/findById.do?id=" + row.id).then((res)=>{
    if(res.data.flag){
      //弹出编辑窗口
      this.dialogFormVisible4Edit = true;
      //默认选中第一个标签页
      this.activeName='first';
      //为模型数据赋值，通过VUE数据双向绑定进行信息的回显
      this.formData = res.data.data;
      //发送ajax请求查询所有的检查项信息，用于检查项表格展示
      axios.get("/checkitem/findAll.do").then((res)=> {
        if(res.data.flag){
          //为模型数据赋值，通过VUE数据双向绑定进行信息的回显
          this.tableData = res.data.data;
          //查询当前检查组包含的所有检查项id，用于页面回显
          axios.get("/checkgroup/findCheckItemIdsByCheckGroupId.do?id=" +
row.id).then((res)=> {
            //为模型数据赋值，通过VUE数据双向绑定进行信息的回显
            if(res.data.flag){
                this.checkitemIds = res.data.data;
            }else{
                this.$message.error(res.data.message);
            }
          });
        }else{
          this.$message.error(res.data.message);
        }
      });
    }else{
      this.$message.error("获取数据失败，请刷新当前页面");
    }
  });
}
```

### 4.1.3 发送请求

在编辑窗口中修改完成后，点击确定按钮需要提交请求，所以需要为确定按钮绑定事件并提供处理函数handleEdit

```
<el-button type="primary" @click="handleEdit()">确定</el-button>
```

```
//编辑
handleEdit() {
  //发送ajax请求，提交模型数据
  axios.post("/checkgroup/edit.do?
checkitemIds="+this.checkitemIds,this.formData).
  then((response)=> {
    //隐藏编辑窗口
    this.dialogFormVisible4Edit = false;
    if(response.data.flag){
      this.$message({
        message: response.data.message,
        type: 'success'
      });
    }else{
      this.$message.error(response.data.message);
    }
  }).finally(()=> {
    this.findPage();
  });
}
```

## 4.2 后台代码

### 4.2.1 Controller

在CheckGroupController中增加方法

```
//根据id查询
@RequestMapping("/findById")
public Result findById(Integer id){
  CheckGroup checkGroup = checkGroupService.findById(id);
  if(checkGroup != null){
    Result result = new Result(true,
MessageConstant.QUERY_CHECKGROUP_SUCCESS);
    result.setData(checkGroup);
    return result;
  }
  return new Result(false,MessageConstant.QUERY_CHECKGROUP_FAIL);
}

//根据检查组合id查询对应的所有检查项id
@RequestMapping("/findCheckItemIdsByCheckGroupId")
public Result findCheckItemIdsByCheckGroupId(Integer id){
    try{
        List<Integer> checkitemIds =
            checkGroupService.findCheckItemIdsByCheckGroupId(id);
        return new
Result(true,MessageConstant.QUERY_CHECKITEM_SUCCESS,checkitemIds);
    }catch (Exception e){
        e.printStackTrace();
        return new Result(false, MessageConstant.QUERY_CHECKITEM_FAIL);
    }
}

//编辑
@RequestMapping("/edit")
public Result edit(@RequestBody CheckGroup checkGroup,Integer[]
checkitemIds){
  try {
    checkGroupService.edit(checkGroup,checkitemIds);
  }catch (Exception e){
    return new Result(false,MessageConstant.EDIT_CHECKGROUP_FAIL);
  }
  return new Result(true,MessageConstant.EDIT_CHECKGROUP_SUCCESS);
}
```

### 4.2.2 服务接口

在CheckGroupService服务接口中扩展方法

```
CheckGroup findById(Integer id);
List<Integer> findCheckItemIdsByCheckGroupId(Integer id);
public void edit(CheckGroup checkGroup,Integer[] checkitemIds);
```

### 4.2.3 服务实现类

在CheckGroupServiceImpl实现类中实现编辑方法

```
public CheckGroup findById(Integer id) {
  return checkGroupDao.findById(id);
}

public List<Integer> findCheckItemIdsByCheckGroupId(Integer id) {
  return checkGroupDao.findCheckItemIdsByCheckGroupId(id);
}

//编辑检查组，同时需要更新和检查项的关联关系
public void edit(CheckGroup checkGroup, Integer[] checkitemIds) {
  //根据检查组id删除中间表数据（清理原有关联关系）
  checkGroupDao.deleteAssociation(checkGroup.getId());
  //向中间表(t_checkgroup_checkitem)插入数据（建立检查组和检查项关联关系）
  setCheckGroupAndCheckItem(checkGroup.getId(),checkitemIds);
  //更新检查组基本信息
  checkGroupDao.edit(checkGroup);
}

//向中间表(t_checkgroup_checkitem)插入数据（建立检查组和检查项关联关系）
public void setCheckGroupAndCheckItem(Integer checkGroupId,Integer[]
checkitemIds){
  if(checkitemIds != null && checkitemIds.length > 0){
    for (Integer checkitemId : checkitemIds) {
      Map<String,Integer> map = new HashMap<>();
      map.put("checkgroup_id",checkGroupId);
      map.put("checkitem_id",checkitemId);
      checkGroupDao.setCheckGroupAndCheckItem(map);
    }
  }

}
```

### 4.2.4 Dao接口

在CheckGroupDao接口中扩展方法

```
CheckGroup findById(Integer id);
List<Integer> findCheckItemIdsByCheckGroupId(Integer id);
void setCheckGroupAndCheckItem(Map map);
void deleteAssociation(Integer id);
void edit(CheckGroup checkGroup);
```

### 4.2.5 Mapper映射文件

在CheckGroupDao.xml中扩展SQL语句

```
<select id="findById" parameterType="int"
resultType="com.itheima.pojo.CheckGroup">
  select * from t_checkgroup where id = #{id}
</select>
<select id="findCheckItemIdsByCheckGroupId" parameterType="int"
resultType="int">
  select checkitem_id from t_checkgroup_checkitem where checkgroup_id = #
{id}
</select>
<!--向中间表插入数据（建立检查组和检查项关联关系）-->
<insert id="setCheckGroupAndCheckItem" parameterType="hashmap">
  insert into t_checkgroup_checkitem(checkgroup_id,checkitem_id)
    values
  (#{checkgroup_id},#{checkitem_id})
</insert>
<!--根据检查组id删除中间表数据（清理原有关联关系）-->
<delete id="deleteAssociation" parameterType="int">
  delete from t_checkgroup_checkitem where checkgroup_id = #{id}
</delete>
<!--编辑-->
<update id="edit" parameterType="com.itheima.pojo.CheckGroup">
  update t_checkgroup
  <set>
    <if test="name != null">
      name = #{name},
    </if>
    <if test="sex != null">
      sex = #{sex},
    </if>
    <if test="code != null">
      code = #{code},
    </if>
    <if test="helpCode != null">
      helpCode = #{helpCode},
    </if>
    <if test="attention != null">
      attention = #{attention},
    </if>
    <if test="remark != null">

      remark = #{remark},
```

```
    </if>
  </set>
  where id = #{id}
</update>
```