1. 预约管理-检查项管理

学习目标：

1：基础环境搭建

\* 导入工具类

\* 规定json数据格式，请求、响应

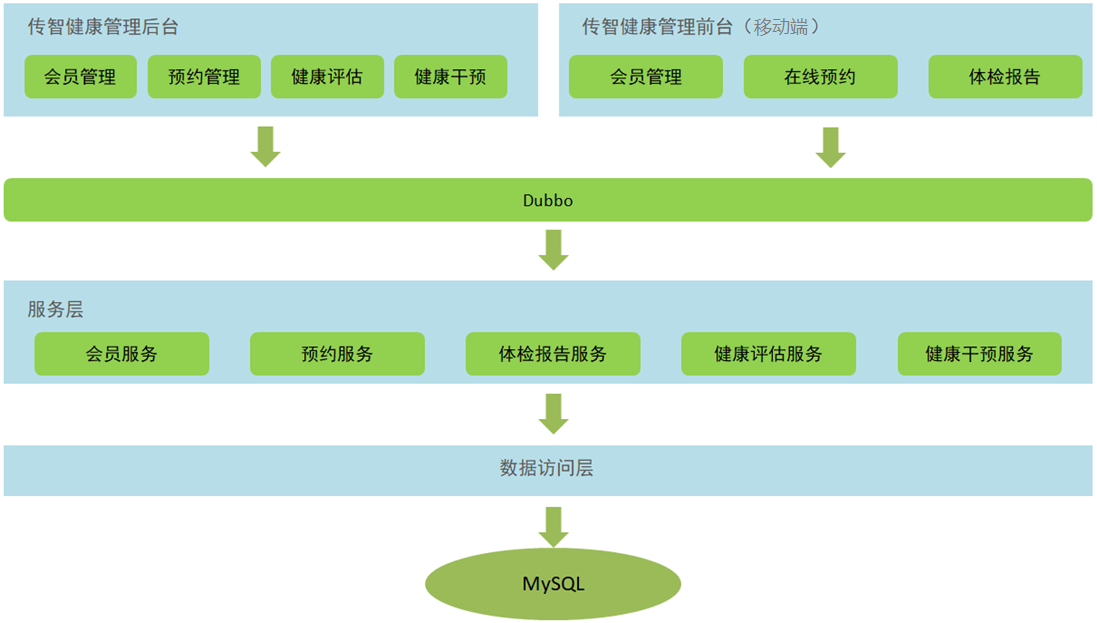
2：检查项管理（重点CRUD）

# 第一章. 需求分析

传智健康管理系统是一款应用于健康管理机构的业务系统，实现健康管理机构工作内容可视化、患者管理专业化、健康评估数字化、健康干预流程化、知识库集成化，从而提高健康管理师的工作效率，加强与患者间的互动，增强管理者对健康管理机构运营情况的了解。

系统分为传智健康后台管理系统和移动端应用两部分。其中后台系统提供给健康管理机构内部人员（包括系统管理员、健康管理师等）使用，微信端应用提供给健康管理机构的用户（体检用户）使用。

本项目功能架构图：



通过上面的功能架构图可以看到，传智健康后台管理系统有会员管理、预约管理、健康评估、健康干预等功能。移动端有会员管理、体检预约、体检报告等功能。后台系统和移动端应用都会通过Dubbo调用服务层发布的服务来完成具体的操作。本项目属于典型的SOA架构形式。

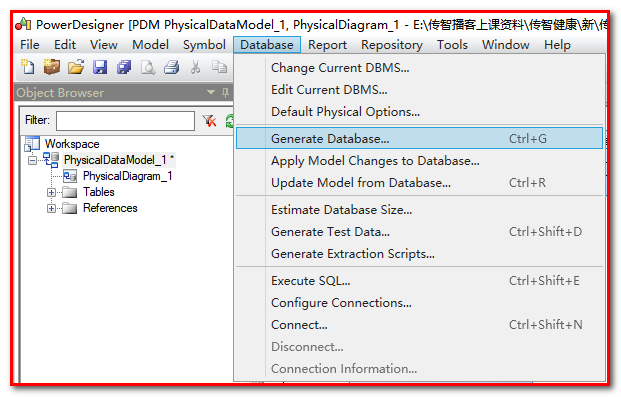
本章节完成的功能开发是预约管理功能，包括检查项管理（身高，体重）、检查组管理（外科）、体检套餐管理（公司健康体检）、预约设置等（参见产品原型）。预约管理属于系统的基础功能，主要就是管理一些体检的基础数据。

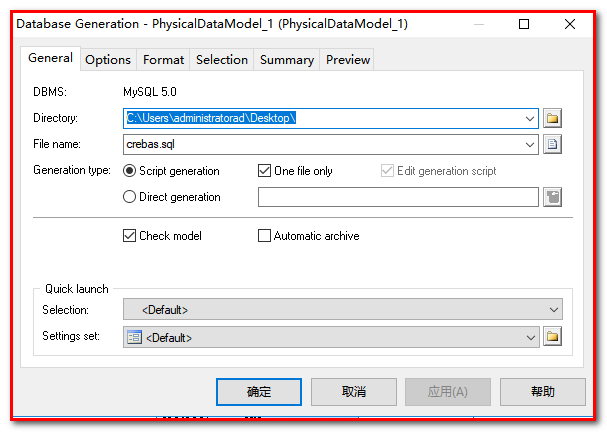
# 第二章. 基础环境搭建

## 导入预约管理模块数据表

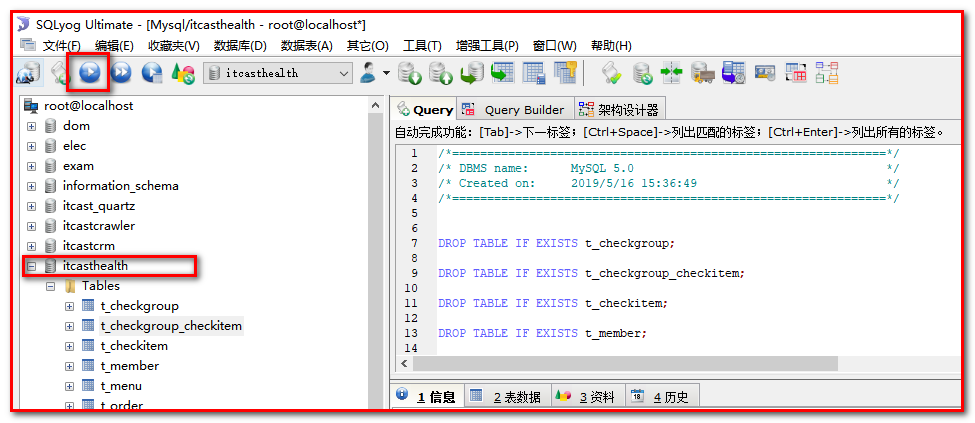
操作步骤：

（1）根据资料中提供的itcasthealth.pdm文件导出SQL脚本



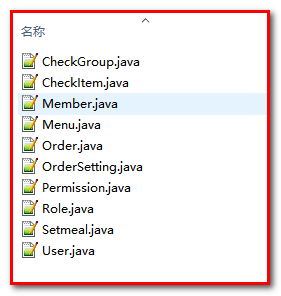


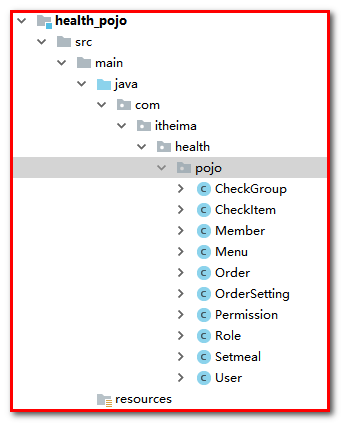
（2）在SQLYog中，执行crebas.sql



## 导入预约管理模块实体类

将资料中提供的POJO实体类复制到health\_pojo工程中。





## 导入项目所需公共资源

项目开发过程中一般会提供一些公共资源，供多个模块或者系统来使用。

本章节我们导入的公共资源有：

1. 返回消息常量类MessageConstant，放到health\_common工程中

**package** com.itheima.health.constant;  
  
*/\*\*  
 \* 消息常量  
 \*/***public class** MessageConstant {  
 **public static final** String ***DELETE\_CHECKITEM\_FAIL*** = **"删除检查项失败"**;  
 **public static final** String ***DELETE\_CHECKITEM\_SUCCESS*** = **"删除检查项成功"**;  
 **public static final** String ***ADD\_CHECKITEM\_SUCCESS*** = **"新增检查项成功"**;  
 **public static final** String ***ADD\_CHECKITEM\_FAIL*** = **"新增检查项失败"**;  
 **public static final** String ***EDIT\_CHECKITEM\_FAIL*** = **"编辑检查项失败"**;  
 **public static final** String ***EDIT\_CHECKITEM\_SUCCESS*** = **"编辑检查项成功"**;  
 **public static final** String ***QUERY\_CHECKITEM\_SUCCESS*** = **"查询检查项成功"**;  
 **public static final** String ***QUERY\_CHECKITEM\_FAIL*** = **"查询检查项失败"**;  
 **public static final** String ***UPLOAD\_SUCCESS*** = **"上传成功"**;  
 **public static final** String ***ADD\_CHECKGROUP\_FAIL*** = **"新增检查组失败"**;  
 **public static final** String ***ADD\_CHECKGROUP\_SUCCESS*** = **"新增检查组成功"**;  
 **public static final** String ***DELETE\_CHECKGROUP\_FAIL*** = **"删除检查组失败"**;  
 **public static final** String ***DELETE\_CHECKGROUP\_SUCCESS*** = **"删除检查组成功"**;  
 **public static final** String ***QUERY\_CHECKGROUP\_SUCCESS*** = **"查询检查组成功"**;  
 **public static final** String ***QUERY\_CHECKGROUP\_FAIL*** = **"查询检查组失败"**;  
 **public static final** String ***EDIT\_CHECKGROUP\_FAIL*** = **"编辑检查组失败"**;  
 **public static final** String ***EDIT\_CHECKGROUP\_SUCCESS*** = **"编辑检查组成功"**;  
 **public static final** String ***PIC\_UPLOAD\_SUCCESS*** = **"图片上传成功"**;  
 **public static final** String ***PIC\_UPLOAD\_FAIL*** = **"图片上传失败"**;  
 **public static final** String ***ADD\_SETMEAL\_FAIL*** = **"新增套餐失败"**;  
 **public static final** String ***ADD\_SETMEAL\_SUCCESS*** = **"新增套餐成功"**;  
 **public static final** String ***IMPORT\_ORDERSETTING\_FAIL*** = **"批量导入预约设置数据失败"**;  
 **public static final** String ***IMPORT\_ORDERSETTING\_SUCCESS*** = **"批量导入预约设置数据成功"**;  
 **public static final** String ***GET\_ORDERSETTING\_SUCCESS*** = **"获取预约设置数据成功"**;  
 **public static final** String ***GET\_ORDERSETTING\_FAIL*** = **"获取预约设置数据失败"**;  
 **public static final** String ***ORDERSETTING\_SUCCESS*** = **"预约设置成功"**;  
 **public static final** String ***ORDERSETTING\_FAIL*** = **"预约设置失败"**;  
 **public static final** String ***ADD\_MEMBER\_FAIL*** = **"新增会员失败"**;  
 **public static final** String ***ADD\_MEMBER\_SUCCESS*** = **"新增会员成功"**;  
 **public static final** String ***DELETE\_MEMBER\_FAIL*** = **"删除会员失败"**;  
 **public static final** String ***DELETE\_MEMBER\_SUCCESS*** = **"删除会员成功"**;  
 **public static final** String ***EDIT\_MEMBER\_FAIL*** = **"编辑会员失败"**;  
 **public static final** String ***EDIT\_MEMBER\_SUCCESS*** = **"编辑会员成功"**;  
 **public static final** String ***TELEPHONE\_VALIDATECODE\_NOTNULL*** = **"手机号和验证码都不能为空"**;  
 **public static final** String ***LOGIN\_SUCCESS*** = **"登录成功"**;  
 **public static final** String ***VALIDATECODE\_ERROR*** = **"验证码输入错误"**;  
 **public static final** String ***QUERY\_ORDER\_SUCCESS*** = **"查询预约信息成功"**;  
 **public static final** String ***QUERY\_ORDER\_FAIL*** = **"查询预约信息失败"**;  
 **public static final** String ***QUERY\_SETMEALLIST\_SUCCESS*** = **"查询套餐列表数据成功"**;  
 **public static final** String ***QUERY\_SETMEALLIST\_FAIL*** = **"查询套餐列表数据失败"**;  
 **public static final** String ***QUERY\_SETMEAL\_SUCCESS*** = **"查询套餐数据成功"**;  
 **public static final** String ***QUERY\_SETMEAL\_FAIL*** = **"查询套餐数据失败"**;  
 **public static final** String ***SEND\_VALIDATECODE\_FAIL*** = **"验证码发送失败"**;  
 **public static final** String ***SEND\_VALIDATECODE\_SUCCESS*** = **"验证码发送成功"**;  
 **public static final** String ***SELECTED\_DATE\_CANNOT\_ORDER*** = **"所选日期不能进行体检预约"**;  
 **public static final** String ***ORDER\_FULL*** = **"预约已满"**;  
 **public static final** String ***HAS\_ORDERED*** = **"已经完成预约，不能重复预约"**;  
 **public static final** String ***ORDER\_SUCCESS*** = **"预约成功"**;  
 **public static final** String ***GET\_USERNAME\_SUCCESS*** = **"获取当前登录用户名称成功"**;  
 **public static final** String ***GET\_USERNAME\_FAIL*** = **"获取当前登录用户名称失败"**;  
 **public static final** String ***GET\_MENU\_SUCCESS*** = **"获取当前登录用户菜单成功"**;  
 **public static final** String ***GET\_MENU\_FAIL*** = **"获取当前登录用户菜单失败"**;  
 **public static final** String ***GET\_MEMBER\_NUMBER\_REPORT\_SUCCESS*** = **"获取会员统计数据成功"**;  
 **public static final** String ***GET\_MEMBER\_NUMBER\_REPORT\_FAIL*** = **"获取会员统计数据失败"**;  
 **public static final** String ***GET\_SETMEAL\_COUNT\_REPORT\_SUCCESS*** = **"获取套餐统计数据成功"**;  
 **public static final** String ***GET\_SETMEAL\_COUNT\_REPORT\_FAIL*** = **"获取套餐统计数据失败"**;  
 **public static final** String ***GET\_BUSINESS\_REPORT\_SUCCESS*** = **"获取运营统计数据成功"**;  
 **public static final** String ***GET\_BUSINESS\_REPORT\_FAIL*** = **"获取运营统计数据失败"**;  
 **public static final** String ***GET\_SETMEAL\_LIST\_SUCCESS*** = **"查询套餐列表数据成功"**;  
 **public static final** String ***GET\_SETMEAL\_LIST\_FAIL*** = **"查询套餐列表数据失败"**;  
}

1. 返回结果Result和PageResult类，放到health\_pojo工程中

Result.java

**package** com.itheima.health.entity;  
  
**import** java.io.Serializable;  
  
*/\*\*  
 \* 封装返回结果  
 \*/***public class** Result **implements** Serializable{  
 **private boolean flag**;*//执行结果，true为执行成功 false为执行失败* **private** String **message**;*//返回结果信息* **private** Object **data**;*//返回数据* **public** Result(**boolean** flag, String message) {  
 **super**();  
 **this**.**flag** = flag;  
 **this**.**message** = message;  
 }  
  
 **public** Result(**boolean** flag, String message, Object data) {  
 **this**.**flag** = flag;  
 **this**.**message** = message;  
 **this**.**data** = data;  
 }  
  
 **public boolean** isFlag() {  
 **return flag**;  
 }  
 **public void** setFlag(**boolean** flag) {  
 **this**.**flag** = flag;  
 }  
 **public** String getMessage() {  
 **return message**;  
 }  
 **public void** setMessage(String message) {  
 **this**.**message** = message;  
 }  
  
 **public** Object getData() {  
 **return data**;  
 }  
  
 **public void** setData(Object data) {  
 **this**.**data** = data;  
 }  
}

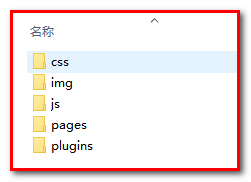
PageResult.java

**package** com.itheima.health.entity;  
  
**import** java.io.Serializable;  
**import** java.util.List;  
  
*/\*\*  
 \* 分页结果封装对象  
 \*/***public class** PageResult **implements** Serializable{  
 **private** Long **total**;*//总记录数* **private** List **rows**;*//当前页结果* **public** PageResult(Long total, List rows) {  
 **super**();  
 **this**.**total** = total;  
 **this**.**rows** = rows;  
 }  
 **public** Long getTotal() {  
 **return total**;  
 }  
 **public void** setTotal(Long total) {  
 **this**.**total** = total;  
 }  
 **public** List getRows() {  
 **return rows**;  
 }  
 **public void** setRows(List rows) {  
 **this**.**rows** = rows;  
 }  
}

1. 封装查询条件的QueryPageBean类，放到health\_pojo工程中

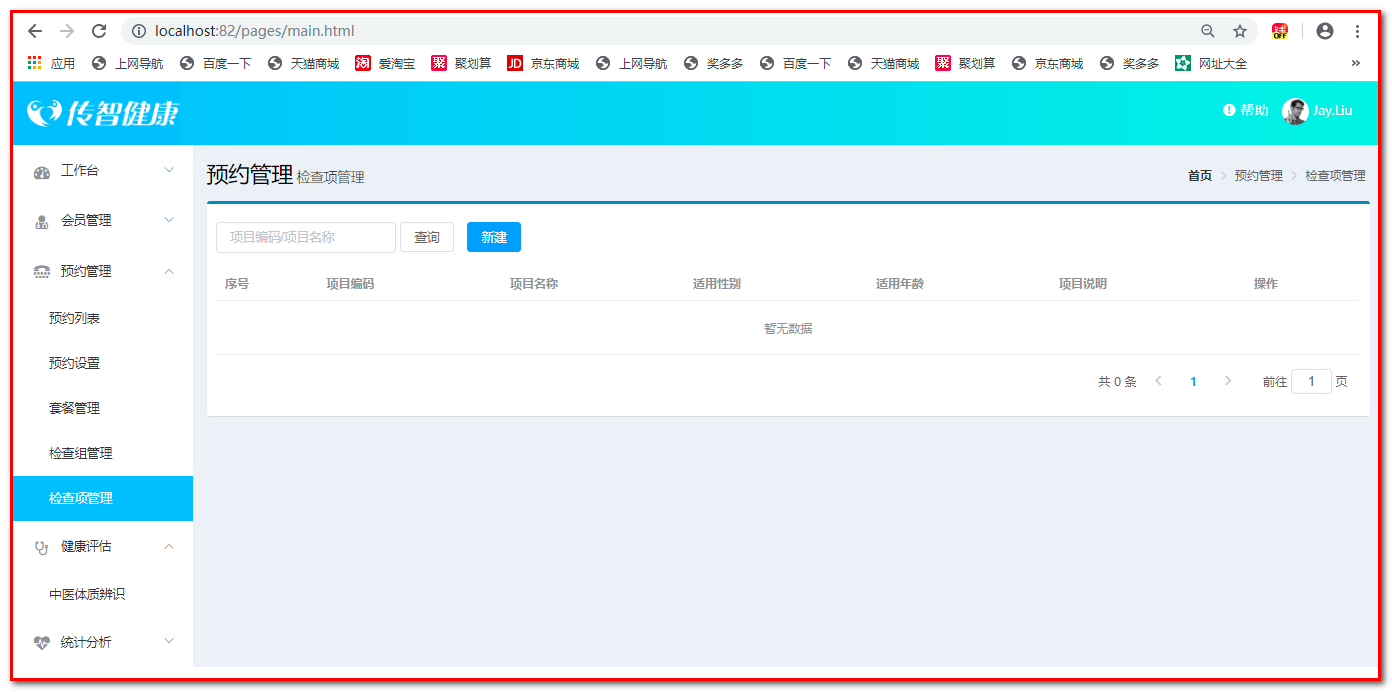
**package** com.itheima.health.entity;  
  
**import** java.io.Serializable;  
  
*/\*\*  
 \* 封装查询条件  
 \*/***public class** QueryPageBean **implements** Serializable{  
 **private** Integer **currentPage**;*//页码* **private** Integer **pageSize**;*//每页记录数* **private** String **queryString**;*//查询条件* **public** Integer getCurrentPage() {  
 **return currentPage**;  
 }  
  
 **public void** setCurrentPage(Integer currentPage) {  
 **this**.**currentPage** = currentPage;  
 }  
  
 **public** Integer getPageSize() {  
 **return pageSize**;  
 }  
  
 **public void** setPageSize(Integer pageSize) {  
 **this**.**pageSize** = pageSize;  
 }  
  
 **public** String getQueryString() {  
 **return queryString**;  
 }  
  
 **public void** setQueryString(String queryString) {  
 **this**.**queryString** = queryString;  
 }  
}

1. html、js、css、图片等静态资源，放到health\_web工程中



1：安装health\_parent

2：启动health\_web（tomcat7:run）



注意：后续随着项目开发还会陆续导入其他一些公共资源。

# 第三章. 新增检查项

main.html页面介绍：

## 完善页面

检查项管理页面对应的是checkitem.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。

### 弹出新增窗口

页面中已经提供了新增窗口，只是处于隐藏状态。只需要将控制展示状态的属性dialogFormVisible改为true就可以显示出新增窗口。

新建按钮绑定的方法为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。

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

### 输入校验

**rules**: {*//校验规则* **code**: [{ **required**: **true**, **message**: **'项目编码为必填项'**, **trigger**: **'blur'** }],  
 **name**: [{ **required**: **true**, **message**: **'项目名称为必填项'**, **trigger**: **'blur'** }]  
}

### 提交表单数据

点击新增窗口中的确定按钮时，触发handleAdd方法，所以需要在handleAdd方法中进行完善。

*//添加*handleAdd () {  
 *//校验表单输入项是否合法* **this**.**$refs**[**'dataAddForm'**].validate((valid) => {  
 **if** (valid) {  
 *//表单数据校验通过，发送ajax请求将表单数据提交到后台* **axios**.post(**"/checkitem/add.do"**,**this**.**formData**).then((response)=> {  
 *//隐藏新增窗口* **this**.**dialogFormVisible** = **false**;  
 *//判断后台返回的flag值，true表示添加操作成功，false为添加操作失败* **if**(response.**data**.flag){  
 **this**.**$message**({  
 **message**: response.**data**.**message**,  
 **type**: **'success'** });  
 }**else**{  
 **this**.**$message**.error(response.**data**.**message**);  
 }  
 }).finally(()=> {  
 *//this.findPage();* });  
 } **else** {  
 **this**.**$message**.error(**"表单数据校验失败"**);  
 **return false**;  
 }  
 });  
},  
*//分页查询*findPage() {  
},

## 后台代码

### Controller

在health\_web工程中创建CheckItemController

**package** com.itheima.health.controller;  
  
**import** com.alibaba.dubbo.config.annotation.Reference;  
**import** com.itheima.health.constant.MessageConstant;  
**import** com.itheima.health.entity.Result;  
**import** com.itheima.health.pojo.CheckItem;  
**import** com.itheima.health.service.CheckItemService;  
**import** org.springframework.web.bind.annotation.RequestBody;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
*/\*\*  
 \* 体检检查项管理  
 \*/*@RestController  
@RequestMapping(**"/checkitem"**)  
**public class** CheckItemController {  
 @Reference  
 **private** CheckItemService **checkItemService**;  
  
 *//新增* @RequestMapping(**"/add"**)  
 **public** Result add(@RequestBody CheckItem checkItem){  
 **try** {  
 **checkItemService**.add(checkItem);  
 }**catch** (Exception e){  
 **return new** Result(**false**, MessageConstant.***ADD\_CHECKITEM\_FAIL***);  
 }  
 **return new** Result(**true**,MessageConstant.***ADD\_CHECKITEM\_SUCCESS***);  
 }  
}

### 服务接口

在health\_interface工程中创建CheckItemService接口

**package** com.itheima.health.service;  
  
**import** com.itheima.health.pojo.CheckItem;  
*/\*\*  
 \* 检查项服务接口  
 \*/***public interface** CheckItemService {  
 **void** add(CheckItem checkItem);  
}

### 服务实现类

在health\_service工程中创建CheckItemServiceImpl实现类

**package** com.itheima.health.service;  
  
**import** com.alibaba.dubbo.config.annotation.Service;  
**import** com.itheima.health.dao.CheckItemDao;  
**import** com.itheima.health.pojo.CheckItem;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.transaction.annotation.Transactional;  
  
*/\*\*  
 \* 检查项服务  
 \*/*@Service(interfaceClass = CheckItemService.**class**)  
@Transactional  
**public class** CheckItemServiceImpl **implements** CheckItemService {  
 @Autowired  
 **private** CheckItemDao **checkItemDao**;  
 *//新增* **public void** add(CheckItem checkItem) {  
 **checkItemDao**.add(checkItem);  
 }  
}

### Dao接口

在health\_dao工程中创建CheckItemDao接口，本项目是基于Mybatis的Mapper代理技术实现持久层操作，故只需要提供接口和Mapper映射文件，无须提供实现类

**package** com.itheima.health.dao;  
  
**import** com.itheima.health.pojo.CheckItem;  
*/\*\*  
 \* 持久层Dao接口  
 \*/*

@Repository**public interface** CheckItemDao {  
 **public void** add(CheckItem checkItem);  
}

### Mapper映射文件

在health\_dao工程中创建CheckItemDao.xml映射文件，需要和CheckItemDao接口在同一目录下

*<?***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.health.dao.CheckItemDao"**>  
 *<!--新增-->* <**insert id="add" parameterType="com.itheima.health.pojo.CheckItem"**>  
 insert into t\_checkitem(code,name,sex,age,price,type,remark,attention)  
 values  
 (#{code},#{name},#{sex},#{age},#{price},#{type},#{remark},#{attention})  
 </**insert**>  
</**mapper**>

# 第四章. 检查项分页

本项目所有分页功能都是基于ajax的异步请求来完成的，请求参数和后台响应数据格式都使用json数据格式。

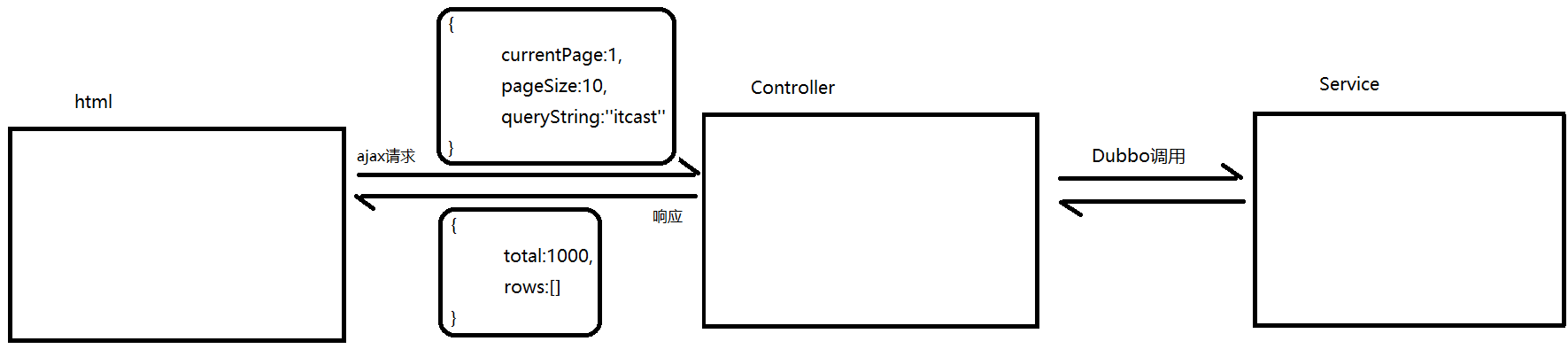
1：请求参数包括页码、每页显示记录数、查询条件。

请求参数的json格式为：{currentPage:1,pageSize:10,queryString:''itcast''}

2：后台响应数据包括总记录数、当前页需要展示的数据集合。

响应数据的json格式为：{total:1000,rows:[]}

如下图：



## 完善页面

### 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*

### 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checkitem.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(**"/checkitem/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**;  
 });  
},

### 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

<**div class="pagination-container"**>  
 <**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**>  
</**div**>

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();  
},

## 后台代码

### Controller

在CheckItemController中增加分页查询方法

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

### 服务接口

在CheckItemService服务接口中扩展分页查询方法

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

### 服务实现类

在CheckItemServiceImpl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 = **checkItemDao**.selectByCondition(queryString);  
 **return new** PageResult(page.getTotal(),page.getResult());  
}

### Dao接口

在CheckItemDao接口中扩展分页查询方法

Page<CheckItem> selectByCondition(String queryString);

### Mapper映射文件

在CheckItemDao.xml文件中增加SQL定义

*<!--条件查询-->*<**select id="selectByCondition" parameterType="string" resultType="com.itheima.health.pojo.CheckItem"**>  
 select *\** from t\_checkitem  
 <**if test="value != null and value.length > 0"**>  
 where code = #{value} or name = #{value}  
 </**if**>  
</**select**>

这里需要是value读取值，不能改成其他参数。

# 第五章. 删除检查项

## 完善页面

为了防止用户误操作，点击删除按钮时需要弹出确认删除的提示，用户点击取消则不做任何操作，用户点击确定按钮再提交删除请求。

### 绑定单击事件

需要为删除按钮绑定单击事件，并且将当前行数据作为参数传递给处理函数

<**el-button size="mini" type="danger" @click="handleDelete(scope.row)"**>删除</**el-button**>

调用的方法

*// 删除*handleDelete(row) {  
 alert(row.**id**);  
}

### 弹出确认操作提示

用户点击删除按钮会执行handleDelete方法，此处需要完善handleDelete方法，弹出确认提示信息。ElementUI提供了$confirm方法来实现确认提示信息弹框效果

*// 删除*handleDelete(row) {  
 *// alert(row.id);* **this**.**$confirm**(**"确认删除当前选中记录吗？"**,**"提示"**,{**type**:**'warning'**}).then(()=>{  
 *//点击确定按钮时只需此处代码* alert(**'用户点击的是确定按钮'**);  
 });  
}



### 发送请求

如果用户点击确定按钮就需要发送ajax请求，并且将当前检查项的id作为参数提交到后台进行删除操作

*// 删除*handleDelete(row) {  
 *// alert(row.id);* **this**.**$confirm**(**"确认删除当前选中记录吗？"**,**"提示"**,{**type**:**'warning'**}).then(()=>{  
 *//点击确定按钮时只需此处代码  
 // alert('用户点击的是确定按钮');* **axios**.get(**"/checkitem/delete.do?id="** + row.**id**).then((res)=> {  
 **if**(res.**data**.flag){  
 *//删除成功* **this**.**$message**({  
 **message**: res.**data**.**message**,  
 **type**: **'success'** });  
 *//调用分页，获取最新分页数据* **this**.findPage();  
 }**else**{  
 *//删除失败* **this**.**$message**.error(res.**data**.**message**);  
 }  
 });  
 });  
}

## 后台代码

### Controller

在CheckItemController中增加删除方法

*//删除*@RequestMapping(**"/delete"**)  
**public** Result delete(Integer id){  
 **try** {  
 **checkItemService**.delete(id);  
 }**catch** (RuntimeException e){  
 **return new** Result(**false**,e.getMessage());  
 }**catch** (Exception e){  
 **return new** Result(**false**, MessageConstant.***DELETE\_CHECKITEM\_FAIL***);  
 }  
 **return new** Result(**true**,MessageConstant.***DELETE\_CHECKITEM\_SUCCESS***);  
}

### 服务接口

在CheckItemService服务接口中扩展删除方法

**void** delete(Integer id);

### 服务实现类

注意：不能直接删除，需要判断当前检查项是否和检查组关联，如果已经和检查组进行了关联则不允许删除

*//删除***public void** delete(Integer id) **throws** RuntimeException{  
 *//查询当前检查项是否和检查组关联* **long** count = **checkItemDao**.findCountByCheckItemId(id);  
 **if**(count > 0){  
 *//当前检查项被引用，不能删除* **throw new** RuntimeException(**"当前检查项被检查组引用，不能删除"**);  
 }  
 **checkItemDao**.deleteById(id);  
}

### Dao接口

在CheckItemDao接口中扩展方法findCountByCheckItemId和deleteById

**long** findCountByCheckItemId(Integer id);  
**void** deleteById(Integer id);

### Mapper映射文件

在CheckItemDao.xml中扩展SQL语句

*<!--根据检查项id查询中间关系表-->*<**select id="findCountByCheckItemId" resultType="long" parameterType="int"**>  
 select count(\*) from t\_checkgroup\_checkitem where checkitem\_id = #{checkitem\_id}  
</**select**>

*<!--删除-->*<**delete id="deleteById" parameterType="int"**>  
 delete from t\_checkitem where id = #{id}  
</**delete**>

# 第六章. 编辑检查项

## 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。

### 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**>

处理事件：handleUpdate();

*// 弹出编辑窗口*handleUpdate(row) {  
 alert(row.**id**);  
},

### 弹出编辑窗口回显数据

当前页面中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ajax请求查询当前检查项数据用于回显

*// 弹出编辑窗口*handleUpdate(row) {  
 *// alert(row.id);  
 //发送请求获取检查项信息* **axios**.get(**"/checkitem/findById.do?id="** + row.**id**).then((res)=>{  
 **if**(res.**data**.flag){  
 *//设置编辑窗口属性，dialogFormVisible4Edit为true表示显示* **this**.**dialogFormVisible4Edit** = **true**;  
 *//为模型数据设置值，基于VUE双向数据绑定回显到页面* **this**.**formData** = res.**data**.**data**;  
 }**else**{  
 **this**.**$message**.error(**"获取数据失败，请刷新当前页面"**);  
 }  
 });  
},

### 发送请求更改数据

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

<**div slot="footer" class="dialog-footer"**>  
 <**el-button @click="dialogFormVisible4Edit = false"**>取消</**el-button**>  
 <**el-button type="primary" @click="handleEdit()"**>确定</**el-button**>  
</**div**>

handleEdit()方法

*//编辑*handleEdit() {  
 *//表单校验* **this**.**$refs**[**'dataEditForm'**].validate((valid)=>{  
 **if**(valid){  
 *//表单校验通过，发送请求* **axios**.post(**"/checkitem/edit.do"**,**this**.**formData**).then((response)=> {  
 *//隐藏编辑窗口* **this**.**dialogFormVisible4Edit** = **false**;  
 **if**(response.**data**.flag){  
 *//编辑成功，弹出成功提示信息* **this**.**$message**({  
 **message**: response.**data**.**message**,  
 **type**: **'success'** });  
 }**else**{  
 *//编辑失败，弹出错误提示信息* **this**.**$message**.error(response.**data**.**message**);  
 }  
 }).finally(()=> {  
 *//重新发送请求查询分页数据* **this**.findPage();  
 });  
 }**else**{  
 *//表单校验失败* **this**.**$message**.error(**"表单数据校验失败"**);  
 **return false**;  
 }  
 });  
},

## 后台代码

### Controller

在CheckItemController中增加编辑方法

*// 跳转到检查项编辑页面*@RequestMapping(**"/findById"**)  
**public** Result findById(Integer id){  
 **try** {  
 CheckItem checkItem = **checkItemService**.findById(id);  
 **return new** Result(**true**,MessageConstant.***QUERY\_CHECKITEM\_SUCCESS***,checkItem);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 **return new** Result(**false**,MessageConstant.***QUERY\_CHECKITEM\_FAIL***);  
 }  
}

*//编辑保存*@RequestMapping(**"/edit"**)  
**public** Result edit(@RequestBody CheckItem checkItem){  
 **try** {  
 **checkItemService**.edit(checkItem);  
 }**catch** (Exception e){  
 **return new** Result(**false**,MessageConstant.***EDIT\_CHECKITEM\_FAIL***);  
 }  
 **return new** Result(**true**,MessageConstant.***EDIT\_CHECKITEM\_SUCCESS***);  
}

### 服务接口

在CheckItemService服务接口中扩展编辑方法

CheckItem findById(Integer id);  
**void** edit(CheckItem checkItem);

### 服务实现类

在CheckItemServiceImpl实现类中实现编辑方法

*// 主键查询*@Override  
**public** CheckItem findById(Integer id) {  
 CheckItem checkItem = **checkItemDao**.findCheckItemById(id);  
 **return** checkItem;  
}  
  
*// 编辑保存*@Override  
**public void** edit(CheckItem checkItem) {  
 **checkItemDao**.edit(checkItem);  
}

### Dao接口

在CheckItemDao接口中扩展edit方法

CheckItem findCheckItemById(Integer id);  
  
**void** edit(CheckItem checkItem);

### Mapper映射文件

在CheckItemDao.xml中扩展SQL语句

*<!--根据检查项id查询检查项信息-->*<**select id="findCheckItemById" resultType="com.itheima.health.pojo.CheckItem" parameterType="int"**>  
 select *\** from t\_checkitem where id = #{id}  
</**select**>  
  
*<!--编辑-->*<**update id="edit" parameterType="com.itheima.health.pojo.CheckItem"**>  
 update t\_checkitem  
 <**set**>  
 <**if test="name != null"**>  
 name = #{name},  
 </**if**>  
 <**if test="sex != null"**>  
 sex = #{sex},  
 </**if**>  
 <**if test="code != null"**>  
 code = #{code},  
 </**if**>  
 <**if test="age != null"**>  
 age = #{age},  
 </**if**>  
 <**if test="price != null"**>  
 price = #{price},  
 </**if**>  
 <**if test="type != null"**>  
 type = #{type},  
 </**if**>  
 <**if test="attention != null"**>  
 attention = #{attention},  
 </**if**>  
 <**if test="remark != null"**>  
 remark = #{remark},  
 </**if**>  
 </**set**>  
 where id = #{id}  
</**update**>