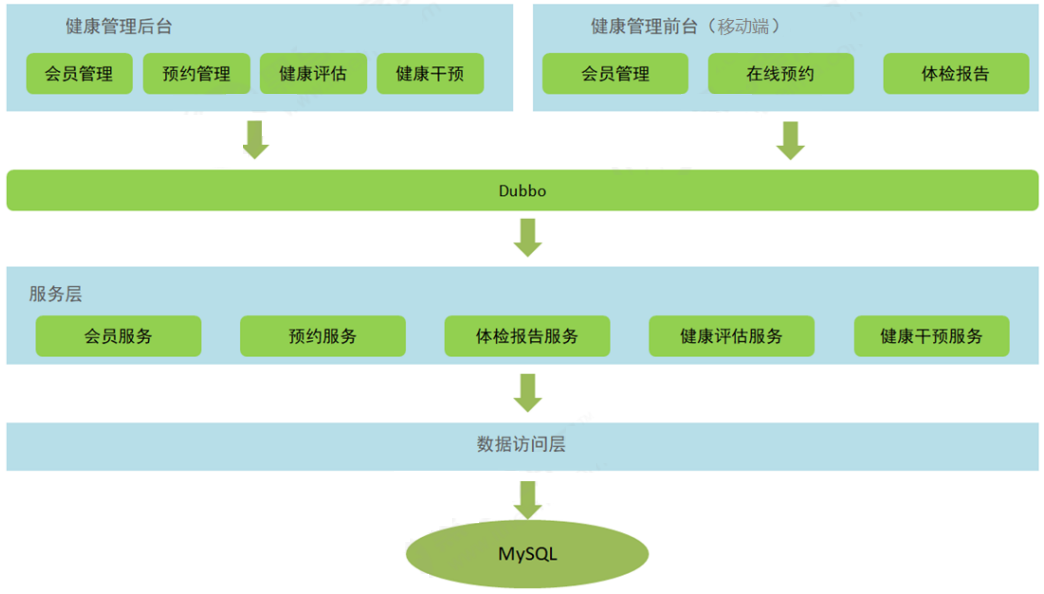
## 1.1需求分析

编易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。

系统分为编易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，编易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、检查组管理、体检套餐管 理、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。

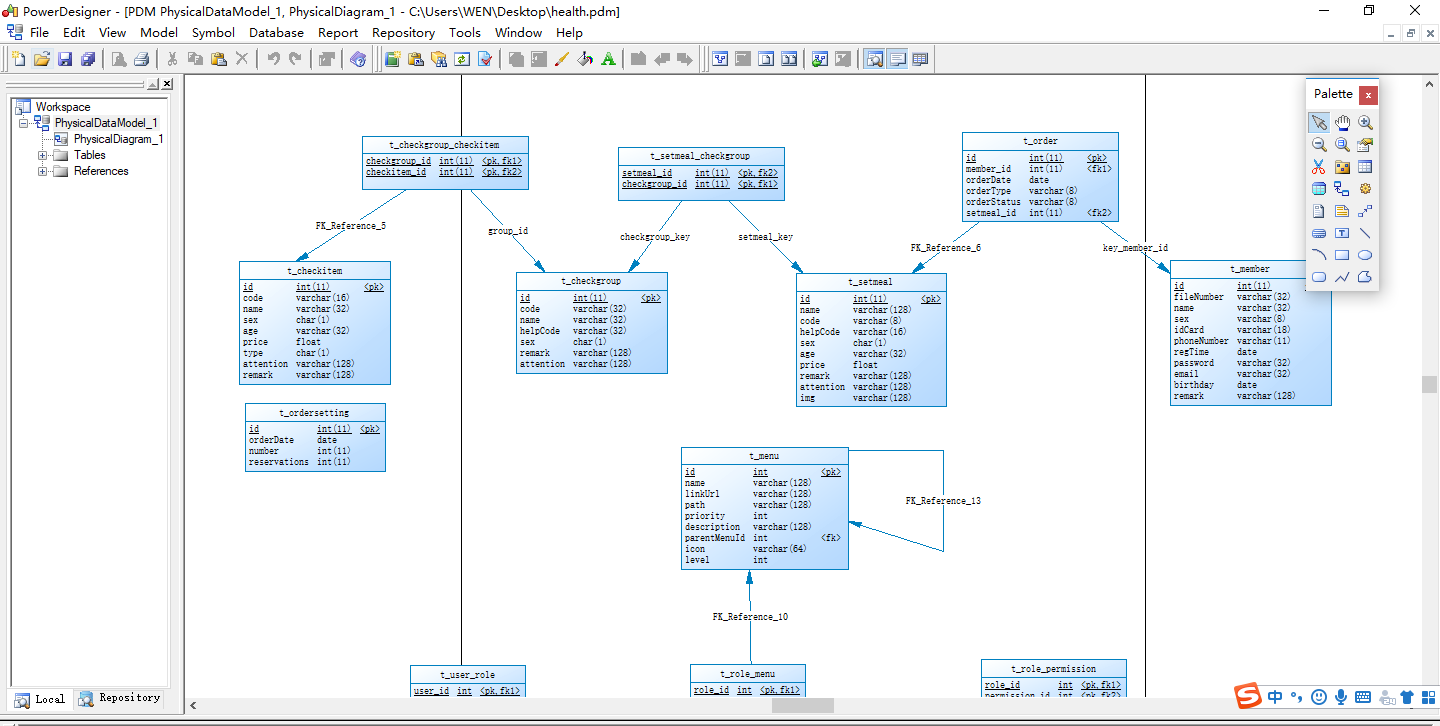
## 1.2基础环境搭建

### 1.2.1导入预约管理模块数据表

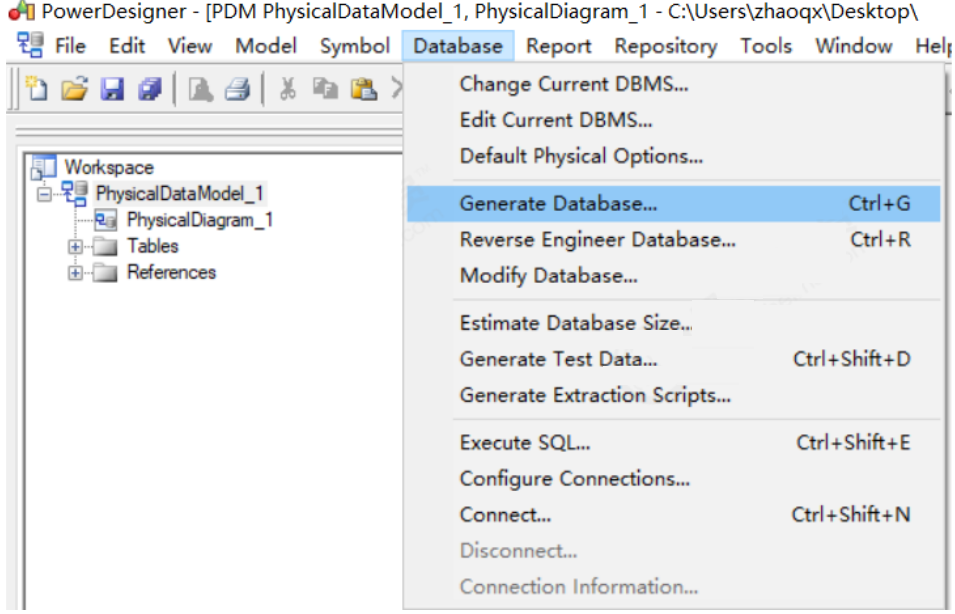
操作步骤：

1. 根据资料中提供的health.pdm文件导出SQL脚本

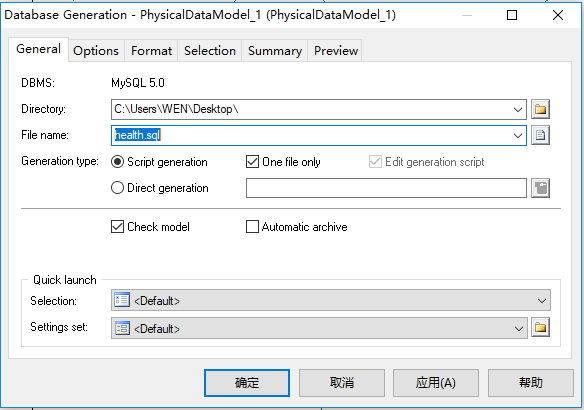
双击打开health.pdm文件



点击database下拉菜单选择generate database.

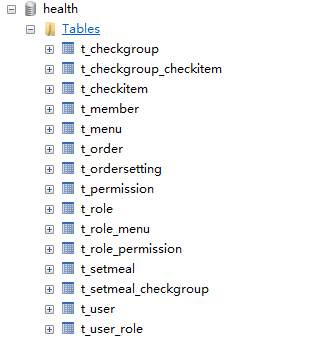


点击Generate Database 打开对话框



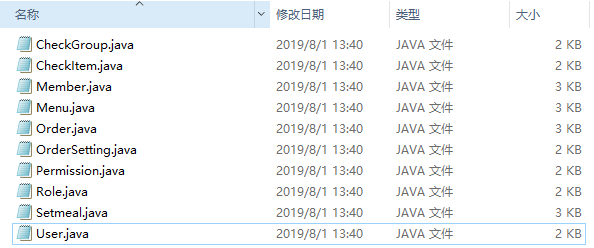
1. 创建health数据库。

将生成的sql脚本复制到sqlYog里面执行。创建成功之后的表如下：



### 1.2.2 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\_common工程中。



### 1.2.3导入项目所需公共资源

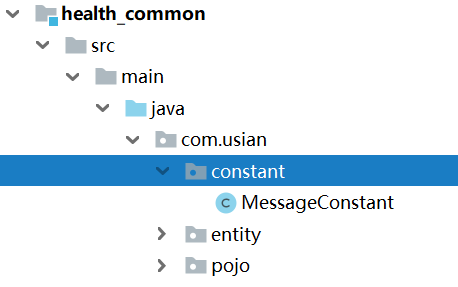
项目开发过程中一般会提供一些公共资源，供多个模块或者系统来使用。

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.qf.constant;  **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"**; } |

定义后的目录如下：



(2)返回结果Result和PageResult类，放到health\_common工程中

Result:

|  |
| --- |
| **package** com.qf.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:

|  |
| --- |
| **package** com.qf.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;  } } |

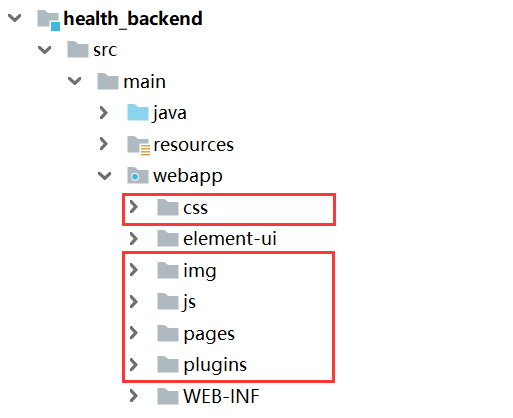
(3)封装查询条件的QueryPageBean类，放到health\_common工程中

QueryPageBean:

|  |
| --- |
| **package** com.qf.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;  } } |

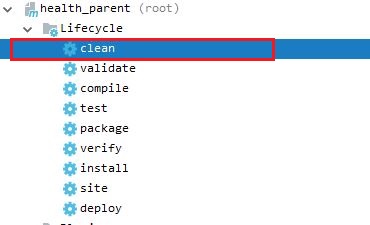
(4)html、js、css、图片等静态资源，放到health\_backend工程中

注意：后续随着项目开发还会陆续导入其他一些公共资源

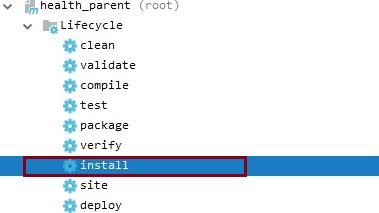


### 1.2.4 启动测试，查看页面效果

由于子模块中的代码进行了更新，所以在项目启动之前，先对父工程进行clean。

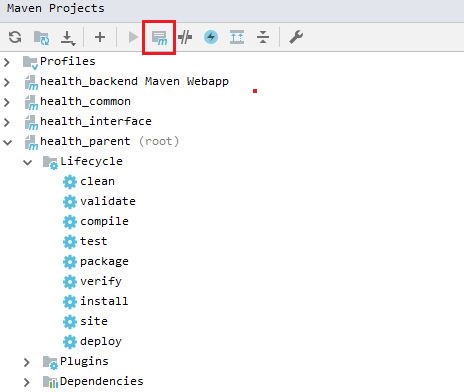


然后重新install。

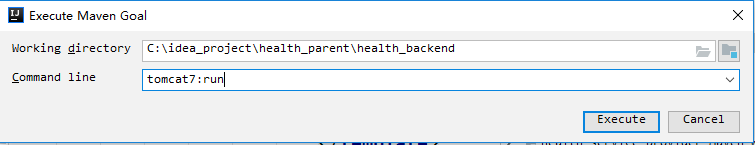


启动health\_backend模块

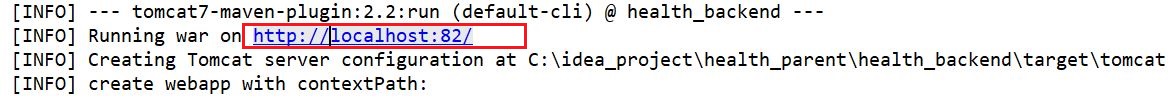
点击如下标识：



然后使用tomcat7插件(tomcat7:run)的指令启动



在控制台运行效果如下：

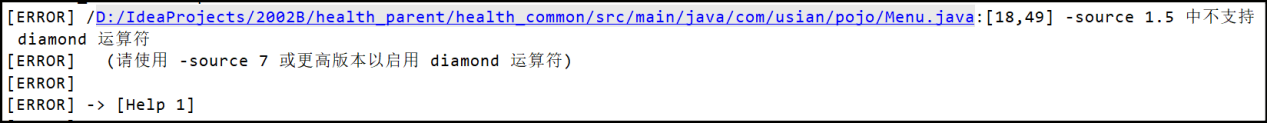


访问main.html主页面：

<http://localhost:82/pages/main.html>

### 1.2.5 问题

运行时出现如下错误：jdk版本问题



解决方案：在health\_parent控制maven编译时的jdk版本

|  |
| --- |
| <**build**>  <**plugins**>  <!-- java编译插件 -->  <**plugin**>  <**groupId**>org.apache.maven.plugins</**groupId**>  <**artifactId**>maven-compiler-plugin</**artifactId**>  <**version**>3.2</**version**>  <**configuration**>  <**source**>1.8</**source**>  <**target**>1.8</**target**>  <**encoding**>UTF-8</**encoding**>  </**configuration**>  </**plugin**>  </**plugins**>  <**pluginManagement**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  </**plugin**>  </**plugins**>  </**pluginManagement**> </**build**> |

## 1.3新增检查项

### 1.3.1 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。

#### 1.3.1.1 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。

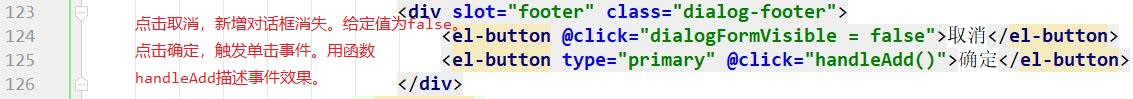






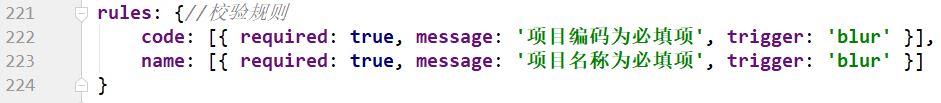


#### 1.3.1.2 数据校验及添加











### 1.3.2 后台代码

#### 1.3.2.1 创建controller

在health\_backend工程中创建CheckItemController。

|  |
| --- |
| @RestController @RequestMapping(**"/checkitem"**) **public class** CheckItemController {   *//去zookeeper服务注册中心查找服务* @Reference  **private** CheckItemService **checkItemService**;  *//新增检查项* @RequestMapping(**"/add"**)  **public** Result add(@RequestBody CheckItem checkItem){  **try**{  **checkItemService**.add(checkItem);  }**catch** (Exception e){  e.printStackTrace();  *//服务调用失败* **return new** Result(**false**, MessageConstant.***ADD\_CHECKITEM\_FAIL***);  }  **return new** Result(**true**, MessageConstant.***ADD\_CHECKITEM\_SUCCESS***);  } } |

#### 1.3.2.2 服务接口

在health\_interface工程中创建CheckItemService接口

|  |
| --- |
| **package** service;  **import** com.qf.pojo.CheckItem;  **public interface** CheckItemService {   **public void** add(CheckItem checkItem); } |

#### 1.3.2.3服务实现类

在health\_service\_provider工程中创建CheckItemServiceImpl实现类

|  |
| --- |
| @Service(interfaceClass = CheckItemService.**class**)*//明确当前服务实现的是哪个接口* @Transactional **public class** CheckItemServiceImpl **implements** CheckItemService {   @Autowired  CheckItemDao **checkItemDao**;   @Override  **public void** add(CheckItem checkItem) {  **checkItemDao**.add(checkItem);  } } |

#### 1.3.2.4 Dao接口

接口：

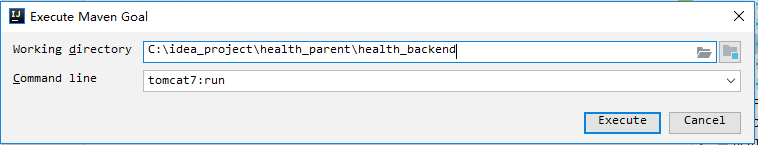
|  |
| --- |
| **public interface** CheckItemDao {  **public void** add(CheckItem checkItem); } |

Mapper映射文件：

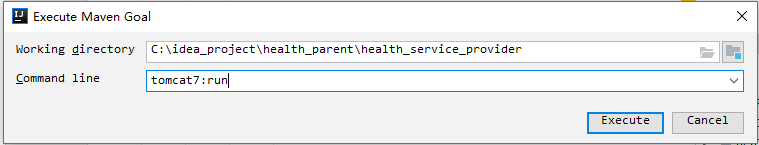
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"** *?>* **<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>*** <**mapper namespace="com.qf.dao.CheckItemDao"**>  *<!--插入检查项数据-->* <**insert id="add" parameterType="com.qf.pojo.CheckItem"**>  insert into t\_checkitem(code,name,sex,age,price,type,remark,attention)  values  (#{code},#{name},#{sex},#{age},#{price},#{type},#{remark},#{attention})  </**insert**> </**mapper**> |

#### 1.3.2.5 启动测试

启动health\_backend服务



启动health\_service\_provider服务



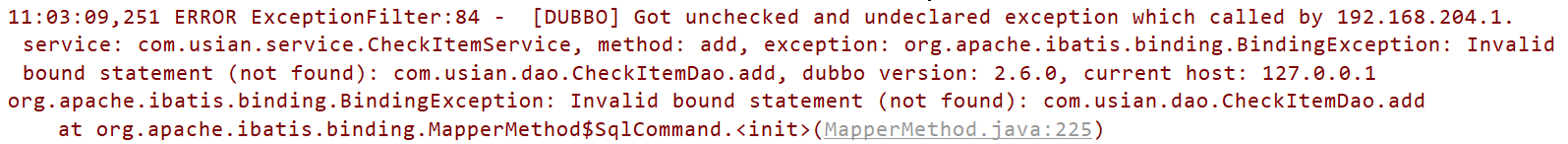
访问：<http://localhost:82/pages/main.html>点击新建进行数据添加。



填充数据之后，点击确定会显示添加成功。

#### 1.3.2.6问题

##### 1.3.2.6.1 maven没有编译java文件中的xml



解决方式：在父工程的bulid中添加配置

|  |
| --- |
| <!-- 如果不添加此节点src/main/java目录下的所有配置文件都会被漏掉。 --> <**resources**>  <**resource**>  <**directory**>src/main/java</**directory**>  <**includes**>  <**include**>\*\*/\*.xml</**include**>  </**includes**>  </**resource**>  <**resource**>  <**directory**>src/main/resources</**directory**>  <**includes**>  <**include**>\*\*/\*.xml</**include**>  <**include**>\*\*/\*.properties</**include**>  </**includes**>  </**resource**> </**resources**> |

##### 1.3.2.6.2Dubbo无法发布被事务代理的Service

原因：在CheckItemServiceImpl 类上加入事务注解后，Spring会为此类基于JDK动态代理技术创建代理对象，创建的代理对象完整类名为com.sun.proxy.$Proxy35，导致Dubbo在进行包匹配时没有成功（因为我们在发布服务时扫描的包为com.qf.service），所以后面真正发布服务的代码没有执行。

解决方案：

参考1.3.2.3

测试：

|  |
| --- |
| **public static void** main(String[] args) {  ApplicationContext ac = **new** ClassPathXmlApplicationContext(**"classpath:applicationContext-\*.xml"**);  System.***out***.println(ac.getBean(**"checkItemServiceImpl"**).getClass()); } |

## 1.4 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。

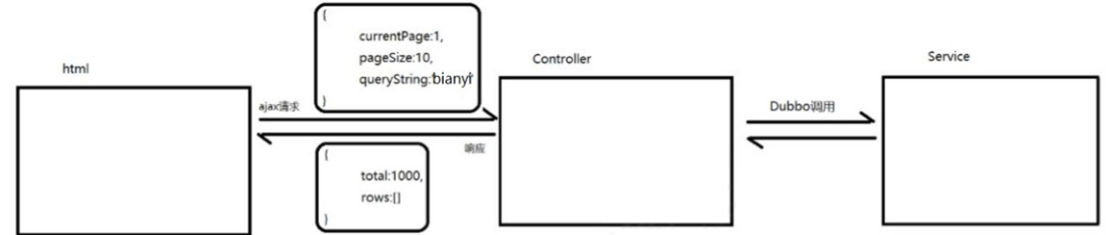
请求参数包括页码、每页显示记录数、查询条件。

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

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:[]}

如图：



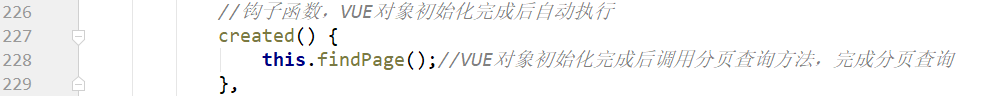
### 1.4.1 完善页面

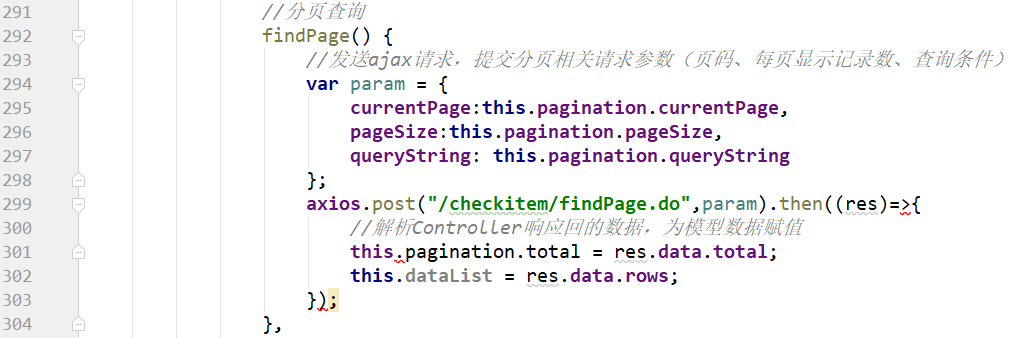
#### 1.4.1.1 定义分页相关模型数据



#### 1.4.1.2 定义分页的方法

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。





#### 1.4.1.3 完善分页方法执行时机

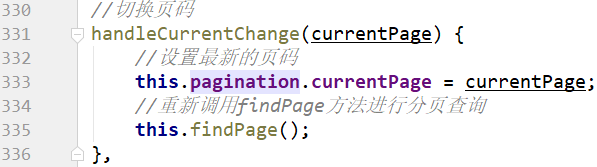
除了在created钩子函数中调用findPage方法查询分页数据之外，当用户点击查询按钮或 者点击分页条中的页码时也需要调用findPage方法重新发起查询请求。

为查询按钮绑定单击事件，调用findPage方法。



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





### 1.4.2 后台代码

#### 1.4.2.1 controller

在CheckItemController中增加分页查询方法

|  |
| --- |
| *//检查项分页查询* @RequestMapping(**"/findPage"**) **public** PageResult findPage(@RequestBody QueryPageBean queryPageBean){  PageResult pageResult = **checkItemService**.pageQuery(queryPageBean);  **return** pageResult; } |

#### 1.4.2.2 服务接口

在CheckItemService服务接口中扩展分页查询方法

|  |
| --- |
| **public interface** CheckItemService {   **public void** add(CheckItem checkItem);    **public PageResult pageQuery(QueryPageBean queryPageBean);** } |

#### 1.4.2.3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

|  |
| --- |
| @Override **public** PageResult pageQuery(QueryPageBean queryPageBean) {  PageHelper.*startPage*(queryPageBean.getCurrentPage(),queryPageBean.getPageSize());   List<CheckItem> checkItemList = **checkItemDao**.findPage(queryPageBean.getQueryString());  PageInfo<CheckItem> pageInfo = **new** PageInfo<>(checkItemList);   **return new** PageResult(pageInfo.getTotal(),pageInfo.getList()); } |

#### 1.4.2.4 dao接口及Mapper映射文件

在CheckItemDao接口中扩展分页查询方法

|  |
| --- |
| **public interface** CheckItemDao {  **public void** add(CheckItem checkItem);  **public Page<CheckItem> selectByCondition(String queryString);** } |

|  |
| --- |
| *<!--检查项分页查询-->* <**select id="selectByCondition" parameterType="String" resultType="com.qf.pojo.CheckItem"**>  select *\** from t\_checkitem  *<!--value就是传进来的值，如果传进来的值不为空，并且长度>0-->* <**if test="value != null and value.length > 0"**>  where code = #{value} or name = #{value}  </**if**> </**select**> |

#### 1.4.2.5 启动测试

为了方便测试，可以导入一些数据进去，查看测试效果。

|  |
| --- |
| INSERT INTO `t\_checkitem` VALUES ('28', '0001', '身高', '0', '0-100', '5', '1', '无', '身高');  INSERT INTO `t\_checkitem` VALUES ('29', '0002', '体重', '0', '0-100', '5', '1', '无', '体重');  INSERT INTO `t\_checkitem` VALUES ('30', '0003', '体重指数', '0', '0-100', '5', '1', '无', '体重指数');  INSERT INTO `t\_checkitem` VALUES ('31', '0004', '收缩压', '0', '0-100', '5', '1', '无', '收缩压');  INSERT INTO `t\_checkitem` VALUES ('32', '0005', '舒张压', '0', '0-100', '5', '1', '无', '舒张压');  INSERT INTO `t\_checkitem` VALUES ('33', '0006', '裸视力（右）', '0', '0-100', '5', '1', '无', '裸视力（右）');  INSERT INTO `t\_checkitem` VALUES ('34', '0007', '裸视力（左）', '0', '0-100', '5', '1', '无', '裸视力（左）');  INSERT INTO `t\_checkitem` VALUES ('35', '0008', '矫正视力（右）', '0', '0-100', '5', '1', '无', '矫正视力（右）');  INSERT INTO `t\_checkitem` VALUES ('36', '0009', '矫正视力（左）', '0', '0-100', '5', '1', '无', '矫正视力（左）');  INSERT INTO `t\_checkitem` VALUES ('37', '0010', '色觉', '0', '0-100', '5', '1', '无', '色觉');  INSERT INTO `t\_checkitem` VALUES ('38', '0011', '白细胞计数', '0', '0-100', '10', '2', '无', '白细胞计数');  INSERT INTO `t\_checkitem` VALUES ('39', '0012', '红细胞计数', '0', '0-100', '10', '2', NULL, '红细胞计数');  INSERT INTO `t\_checkitem` VALUES ('40', '0013', '血红蛋白', '0', '0-100', '10', '2', NULL, '血红蛋白');  INSERT INTO `t\_checkitem` VALUES ('41', '0014', '红细胞压积', '0', '0-100', '10', '2', NULL, '红细胞压积');  INSERT INTO `t\_checkitem` VALUES ('42', '0015', '平均红细胞体积', '0', '0-100', '10', '2', NULL, '平均红细胞体积');  INSERT INTO `t\_checkitem` VALUES ('43', '0016', '平均红细胞血红蛋白含量', '0', '0-100', '10', '2', NULL, '平均红细胞血红蛋白含量');  INSERT INTO `t\_checkitem` VALUES ('44', '0017', '平均红细胞血红蛋白浓度', '0', '0-100', '10', '2', NULL, '平均红细胞血红蛋白浓度');  INSERT INTO `t\_checkitem` VALUES ('45', '0018', '红细胞分布宽度-变异系数', '0', '0-100', '10', '2', NULL, '红细胞分布宽度-变异系数');  INSERT INTO `t\_checkitem` VALUES ('46', '0019', '血小板计数', '0', '0-100', '10', '2', NULL, '血小板计数');  INSERT INTO `t\_checkitem` VALUES ('47', '0020', '平均血小板体积', '0', '0-100', '10', '2', NULL, '平均血小板体积');  INSERT INTO `t\_checkitem` VALUES ('48', '0021', '血小板分布宽度', '0', '0-100', '10', '2', NULL, '血小板分布宽度');  INSERT INTO `t\_checkitem` VALUES ('49', '0022', '淋巴细胞百分比', '0', '0-100', '10', '2', NULL, '淋巴细胞百分比');  INSERT INTO `t\_checkitem` VALUES ('50', '0023', '中间细胞百分比', '0', '0-100', '10', '2', NULL, '中间细胞百分比');  INSERT INTO `t\_checkitem` VALUES ('51', '0024', '中性粒细胞百分比', '0', '0-100', '10', '2', NULL, '中性粒细胞百分比');  INSERT INTO `t\_checkitem` VALUES ('52', '0025', '淋巴细胞绝对值', '0', '0-100', '10', '2', NULL, '淋巴细胞绝对值');  INSERT INTO `t\_checkitem` VALUES ('53', '0026', '中间细胞绝对值', '0', '0-100', '10', '2', NULL, '中间细胞绝对值');  INSERT INTO `t\_checkitem` VALUES ('54', '0027', '中性粒细胞绝对值', '0', '0-100', '10', '2', NULL, '中性粒细胞绝对值');  INSERT INTO `t\_checkitem` VALUES ('55', '0028', '红细胞分布宽度-标准差', '0', '0-100', '10', '2', NULL, '红细胞分布宽度-标准差');  INSERT INTO `t\_checkitem` VALUES ('56', '0029', '血小板压积', '0', '0-100', '10', '2', NULL, '血小板压积');  INSERT INTO `t\_checkitem` VALUES ('57', '0030', '尿比重', '0', '0-100', '10', '2', NULL, '尿比重');  INSERT INTO `t\_checkitem` VALUES ('58', '0031', '尿酸碱度', '0', '0-100', '10', '2', NULL, '尿酸碱度');  INSERT INTO `t\_checkitem` VALUES ('59', '0032', '尿白细胞', '0', '0-100', '10', '2', NULL, '尿白细胞');  INSERT INTO `t\_checkitem` VALUES ('60', '0033', '尿亚硝酸盐', '0', '0-100', '10', '2', NULL, '尿亚硝酸盐');  INSERT INTO `t\_checkitem` VALUES ('61', '0034', '尿蛋白质', '0', '0-100', '10', '2', NULL, '尿蛋白质');  INSERT INTO `t\_checkitem` VALUES ('62', '0035', '尿糖', '0', '0-100', '10', '2', NULL, '尿糖');  INSERT INTO `t\_checkitem` VALUES ('63', '0036', '尿酮体', '0', '0-100', '10', '2', NULL, '尿酮体');  INSERT INTO `t\_checkitem` VALUES ('64', '0037', '尿胆原', '0', '0-100', '10', '2', NULL, '尿胆原');  INSERT INTO `t\_checkitem` VALUES ('65', '0038', '尿胆红素', '0', '0-100', '10', '2', NULL, '尿胆红素');  INSERT INTO `t\_checkitem` VALUES ('66', '0039', '尿隐血', '0', '0-100', '10', '2', NULL, '尿隐血');  INSERT INTO `t\_checkitem` VALUES ('67', '0040', '尿镜检红细胞', '0', '0-100', '10', '2', NULL, '尿镜检红细胞');  INSERT INTO `t\_checkitem` VALUES ('68', '0041', '尿镜检白细胞', '0', '0-100', '10', '2', NULL, '尿镜检白细胞');  INSERT INTO `t\_checkitem` VALUES ('69', '0042', '上皮细胞', '0', '0-100', '10', '2', NULL, '上皮细胞');  INSERT INTO `t\_checkitem` VALUES ('70', '0043', '无机盐类', '0', '0-100', '10', '2', NULL, '无机盐类');  INSERT INTO `t\_checkitem` VALUES ('71', '0044', '尿镜检蛋白定性', '0', '0-100', '10', '2', NULL, '尿镜检蛋白定性');  INSERT INTO `t\_checkitem` VALUES ('72', '0045', '丙氨酸氨基转移酶', '0', '0-100', '10', '2', NULL, '丙氨酸氨基转移酶');  INSERT INTO `t\_checkitem` VALUES ('73', '0046', '天门冬氨酸氨基转移酶', '0', '0-100', '10', '2', NULL, '天门冬氨酸氨基转移酶');  INSERT INTO `t\_checkitem` VALUES ('74', '0047', 'Y-谷氨酰转移酶', '0', '0-100', '10', '2', NULL, 'Y-谷氨酰转移酶');  INSERT INTO `t\_checkitem` VALUES ('75', '0048', '尿素', '0', '0-100', '10', '2', NULL, '尿素');  INSERT INTO `t\_checkitem` VALUES ('76', '0049', '肌酐', '0', '0-100', '10', '2', NULL, '肌酐');  INSERT INTO `t\_checkitem` VALUES ('77', '0050', '尿酸', '0', '0-100', '10', '2', NULL, '尿酸');  INSERT INTO `t\_checkitem` VALUES ('78', '0051', '总胆固醇', '0', '0-100', '10', '2', NULL, '总胆固醇');  INSERT INTO `t\_checkitem` VALUES ('79', '0052', '甘油三酯', '0', '0-100', '10', '2', NULL, '甘油三酯');  INSERT INTO `t\_checkitem` VALUES ('80', '0053', '高密度脂蛋白胆固醇', '0', '0-100', '10', '2', NULL, '高密度脂蛋白胆固醇');  INSERT INTO `t\_checkitem` VALUES ('81', '0054', '低密度脂蛋白胆固醇', '0', '0-100', '10', '2', NULL, '低密度脂蛋白胆固醇');  INSERT INTO `t\_checkitem` VALUES ('82', '0055', '磷酸肌酸激酶', '0', '0-100', '10', '2', NULL, '磷酸肌酸激酶');  INSERT INTO `t\_checkitem` VALUES ('83', '0056', '磷酸肌酸激酶同工酶', '0', '0-100', '10', '2', NULL, '磷酸肌酸激酶同工酶');  INSERT INTO `t\_checkitem` VALUES ('84', '0057', '乳酸脱氢酶', '0', '0-100', '10', '2', NULL, '乳酸脱氢酶');  INSERT INTO `t\_checkitem` VALUES ('85', '0058', '三碘甲状腺原氨酸', '0', '0-100', '10', '2', NULL, '三碘甲状腺原氨酸');  INSERT INTO `t\_checkitem` VALUES ('86', '0059', '甲状腺素', '0', '0-100', '10', '2', NULL, '甲状腺素');  INSERT INTO `t\_checkitem` VALUES ('87', '0060', '促甲状腺激素', '0', '0-100', '10', '2', NULL, '促甲状腺激素');  INSERT INTO `t\_checkitem` VALUES ('88', '0061', '子宫', '2', '0-100', '10', '2', NULL, '子宫');  INSERT INTO `t\_checkitem` VALUES ('89', '0062', '附件', '2', '0-100', '10', '2', NULL, '附件');  INSERT INTO `t\_checkitem` VALUES ('90', '0063', '总胆红素', '0', '0-100', '10', '2', NULL, '总胆红素');  INSERT INTO `t\_checkitem` VALUES ('91', '0064', '直接胆红素', '0', '0-100', '10', '2', NULL, '直接胆红素');  INSERT INTO `t\_checkitem` VALUES ('92', '0065', '间接胆红素', '0', '0-100', '10', '2', NULL, '间接胆红素'); |



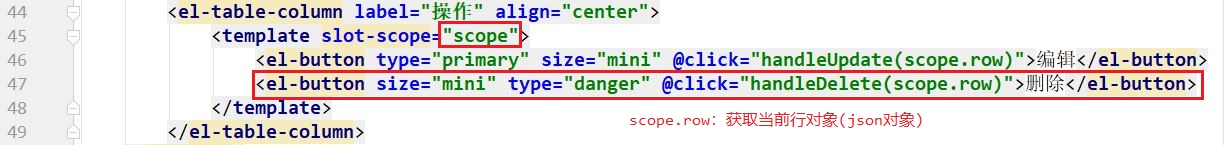
## 1.5 删除检查项

### 1.5.1 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。

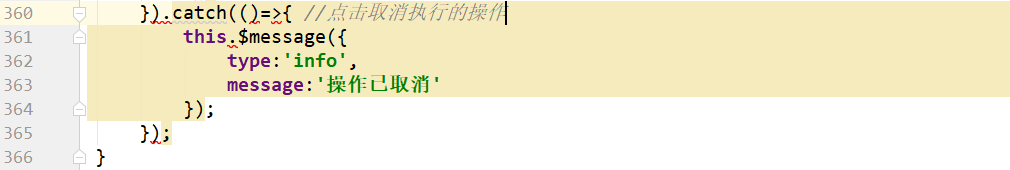
#### 1.5.1.1绑定单击事件

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



#### 1.5.1.2弹出确认消息框，并发送ajax请求





### 1.5.2 后台代码

#### 1.5.2.1 controller

在CheckItemController中增加删除方法

|  |
| --- |
| *//删除检查项* @RequestMapping(**"/delete"**) **public** Result delete(Integer id){  **try**{  **checkItemService**.deleteById(id);  }**catch** (Exception e){  e.printStackTrace();  *//服务调用失败* **return new** Result(**false**, MessageConstant.***DELETE\_CHECKITEM\_FAIL***);  }  **return new** Result(**true**, MessageConstant.***DELETE\_CHECKITEM\_SUCCESS***); } |

#### 1.5.2.2 服务接口

在CheckItemService服务接口中扩展删除方法

|  |
| --- |
| **public void** deleteById(Integer id); |

#### 1.5.2.3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**

|  |
| --- |
| *//根据ID删除检查项* @Override **public void** deleteById(Integer id) {  *//判断当前检查项是否已经关联到检查组* **long** count = **checkItemDao**.findCountByCheckItemId(id);  **if**(count > 0){  *//当前检查项已经被关联到检查组，不允许删除* **new** RuntimeException();  }  **checkItemDao**.deleteById(id); } |

#### 1.5.2.4 dao及Mapper映射文件

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:

|  |
| --- |
| *<!--根据检查项ID统计数据量-->* <**select id="findCountByCheckItemId" parameterType="int" resultType="long"**>  select *count*(*\**) from t\_checkgroup\_checkitem where checkitem\_id = #{checkitem\_id} </**select**> *<!--根据ID删除-->* <**delete id="deleteById" parameterType="int"**>  delete from t\_checkitem where id = #{id} </**delete**> |

#### 1.5.2.5 测试

略

## 1.6 编辑检查项

### 1.6.1 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

#### 1.6.1.1 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



#### 1.6.1.2 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。

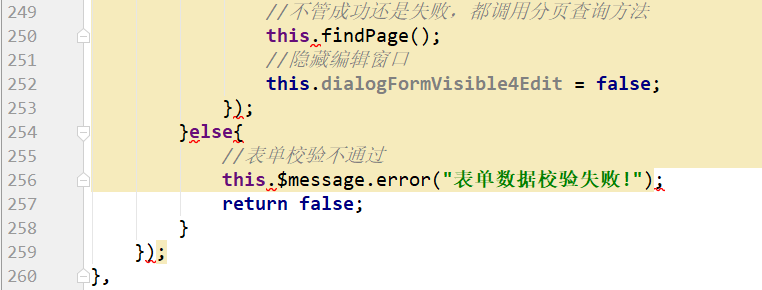


#### 1.6.1.3发送请求

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







### 1.6.2 后台代码

#### 1.6.2.1 controller

在CheckItemController中增加编辑方法

|  |
| --- |
| *//编辑检查项* @RequestMapping(**"/edit"**) **public** Result edit(@RequestBody CheckItem checkItem){  **try**{  **checkItemService**.edit(checkItem);  }**catch** (Exception e){  e.printStackTrace();  *//服务调用失败* **return new** Result(**false**, MessageConstant.***EDIT\_CHECKITEM\_FAIL***);  }  **return new** Result(**true**, MessageConstant.***EDIT\_CHECKITEM\_SUCCESS***); }  @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();  *//服务调用失败* **return new** Result(**false**, MessageConstant.***QUERY\_CHECKITEM\_FAIL***);  } } |

#### 1.6.2.2 服务接口

在CheckItemService服务接口中扩展编辑方法

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

#### 1.6.2.3 服务实现类

在CheckItemServiceImpl实现类中实现编辑方法

|  |
| --- |
| @Override **public void** edit(CheckItem checkItem) {  **checkItemDao**.edit(checkItem); }  @Override **public** CheckItem findById(Integer id) {  **return checkItemDao**.findById(id); } |

#### 1.6.2.4 dao及Mapper映射文件

Dao接口

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

Mapper映射文件

|  |
| --- |
| *<!--编辑检查项-->* <**update id="edit" parameterType="com.qf.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**>  <**select id="findById" parameterType="int" resultType="com.qf.pojo.CheckItem"**>  select *\** from t\_checkitem where id = #{id} </**select**> |

#### 1.6.2.5 测试

测试略