第3章 预约管理-套餐管理

学习目标：

- 了解常见的图片存储方案
- 掌握新增套餐实现过程
- 掌握套餐分页查询实现过程
- 掌握编辑套餐实现过程
- 掌握删除套餐实现过程

# 1. 图片存储方案

## 【目标】

传智健康项目，图片存储方案

## 【路径】

1：介绍

（1）文件上传功能介绍

2：七牛云存储

（1）注册

（2）新建存储空间

（3）查看存储空间信息

（4）开发者中心

- 文件上传
- 文件删除

（5）鉴权

（6）Java SDK操作七牛云

（7）封装工具类

## 【讲解】

# 1.1. 介绍

在实际开发中，我们会有很多处理不同功能的服务器。例如：

应用服务器：负责部署我们的应用

数据库服务器：运行我们的数据库

文件服务器：负责存储用户上传文件的服务器

分服务器处理的目的是让服务器各司其职，从而提高我们项目的运行效率。

常见的图片存储方案：

方案一：使用nginx搭建图片服务器

方案二：使用开源的分布式文件存储系统，例如Fastdfs==、HDFS等

方案三：使用云存储，例如阿里云、七牛云等

# 1.2. 七牛云存储

七牛云（隶属于上海七牛信息技术有限公司）是国内领先的以视觉智能和数据智能为核心的企业级云计算服务商，同时也是国内知名智能视频云服务商，累计为 70 多万家企业提供服务，覆盖了国内80%网民。围绕富媒体场景推出了对象存储、融合 CDN 加速、容器云、大数据平台、深度学习平台等产品、并提供一站式智能视频云解决方案。为各行业及应用提供可持续发展的智能视频云生态，帮助企业快速上云，创造更广阔的商业价值。

官网：https://www.qiniu.com/

通过七牛云官网介绍我们可以知道其提供了多种服务，我们主要使用的是七牛云提供的对象存储服务来存储图片。

## 1.2.1. 注册、登录

要使用七牛云的服务，首先需要注册成为会员。地址：https://portal.qiniu.com/signup

注册完成后就可以使用刚刚注册的邮箱和密码登录到七牛云：

登录成功后点击页面右上角管理控制台：

注意：登录成功后还需要进行实名认证才能进行相关操作。

## 1.2.2. 新建存储空间

要进行图片存储，我们需要在七牛云管理控制台新建存储空间。点击管理控制台首页对象存储下的立即添加按钮，页面跳转到新建存储空间页面：

可以创建多个存储空间，各个存储空间是相互独立的。

## 1.2.3. 查看存储空间信息

存储空间创建后，会在左侧的存储空间列表菜单中展示创建的存储空间名称，点击存储空间名称可以查看当前存储空间的相关信息

课程中重点关注【内容管理】中的信息。

## 1.2.4. 开发者中心

可以通过七牛云提供的开发者中心学习如何操作七牛云服务，地址：https://developer.qiniu.com/

点击对象存储，跳转到对象存储开发页面，地址：https://developer.qiniu.com/kodo

操作步骤：

第一步：导入jar包：

第二步：鉴权

点击“管理控制台”，点击右上图标

可根据文档中提供的上传文件和删除文件进行测试：

在health_common中测试

### 1.2.4.1.文件上传

```
public class TestQiniu {

    // 上传本地文件
    @Test
    public void uploadFile(){
        //构造一个带指定Zone对象的配置类
        Configuration cfg = new Configuration(Zone.zone0());
        //...其他参数参考类注释
        UploadManager uploadManager = new UploadManager(cfg);
        //...生成上传凭证，然后准备上传
        String accessKey = "liyKTcQC5TP1LrhgZH6Xem8zqMXbEtVgfAINP53w";
        String secretKey = "f5zpuzKAPceEMG77-EK3XbwqgOBRDXDawG4UHRtb";
        String bucket = "itcast_health";
        //如果是Windows情况下，格式是 D:\\qiniu\\test.png，可支持中文
        String localFilePath = "D:/2.jpg";
        //默认不指定key的情况下，以文件内容的hash值作为文件名
        String key = null;
        Auth auth = Auth.create(accessKey, secretKey);
        String upToken = auth.uploadToken(bucket);
        try {
            Response response = uploadManager.put(localFilePath, key,
upToken);
            //解析上传成功的结果
            DefaultPutRet putRet = new
Gson().fromJson(response.bodyString(), DefaultPutRet.class);
            System.out.println(putRet.key);
            System.out.println(putRet.hash);
        } catch (QiniuException ex) {
            Response r = ex.response;
            System.err.println(r.toString());
            try {
                System.err.println(r.bodyString());
```

```
            } catch (QiniuException ex2) {
                //ignore
            }
        }
    }
}
```

### 1.2.4.2.文件删除

```
    // 删除空间中的文件
    @Test
    public void deleteFile(){
        //构造一个带指定Zone对象的配置类
        Configuration cfg = new Configuration(Zone.zone0());
        //...其他参数参考类注释
        String accessKey = "liyKTcQC5TP1LrhgZH6Xem8zqMXbEtVgfAINP53w";
        String secretKey = "f5zpuzKAPceEMG77-EK3XbwqgOBRDXDawG4UHRtb";
        String bucket = "itcast_health";
        String key = "Fu3Ic6TV6wIbJt793yaGeBmCkzTX";
        Auth auth = Auth.create(accessKey, secretKey);
        BucketManager bucketManager = new BucketManager(auth, cfg);
        try {
            bucketManager.delete(bucket, key);
        } catch (QiniuException ex) {
            //如果遇到异常，说明删除失败
            System.err.println(ex.code());
            System.err.println(ex.response.toString());
        }
    }
```

七牛云提供了多种方式操作对象存储服务，本项目采用Java SDK方式，地址：https://developer.qiniu.com/kodo/sdk/1239/java

使用Java SDK操作七牛云需要导入如下maven坐标：（项目已经引入）

```
<dependency>
    <groupId>com.qiniu</groupId>
    <artifactId>qiniu-java-sdk</artifactId>
    <version>7.2.0</version>
</dependency>
```

## 1.2.5. 鉴权

Java SDK的所有的功能，都需要合法的授权。授权凭证的签算需要七牛账号下的一对有效的Access Key和Secret Key，这对密钥可以在七牛云管理控制台的个人中心（https://portal.qiniu.com/user/key）获得，如下图：

## 1.2.6. Java SDK操作七牛云

本章节我们就需要使用七牛云提供的Java SDK完成图片上传和删除，我们可以参考官方提供的例子。

上传文件:

```
//构造一个带指定Zone对象的配置类，zone0表示华东地区（默认）
Configuration cfg = new Configuration(Zone.zone0());
//...其他参数参考类注释

UploadManager uploadManager = new UploadManager(cfg);
//...生成上传凭证，然后准备上传
String accessKey = "your access key";
String secretKey = "your secret key";
String bucket = "your bucket name";

//默认不指定key的情况下，以文件内容的hash值作为文件名
String key = null;

try {
    byte[] uploadBytes = "hello qiniu cloud".getBytes("utf-8");
    ByteArrayInputStream byteInputStream=new
ByteArrayInputStream(uploadBytes);
    Auth auth = Auth.create(accessKey, secretKey);
    String upToken = auth.uploadToken(bucket);

    try {
        Response response =
uploadManager.put(byteInputStream,key,upToken,null, null);
        //解析上传成功的结果
        DefaultPutRet putRet = new Gson().fromJson(response.bodyString(),
DefaultPutRet.class);
        System.out.println(putRet.key);
        System.out.println(putRet.hash);
    } catch (QiniuException ex) {
        Response r = ex.response;
        System.err.println(r.toString());
        try {
            System.err.println(r.bodyString());
        } catch (QiniuException ex2) {
            //ignore
        }
    }
} catch (UnsupportedEncodingException ex) {
    //ignore
}
```

删除文件:

```
//构造一个带指定Zone对象的配置类，zone0表示华东地区（默认）
Configuration cfg = new Configuration(Zone.zone0());
//...其他参数参考类注释

String accessKey = "your access key";
String secretKey = "your secret key";

String bucket = "your bucket name";
String key = "your file key";

Auth auth = Auth.create(accessKey, secretKey);
BucketManager bucketManager = new BucketManager(auth, cfg);
```

```
try {
    bucketManager.delete(bucket, key);
} catch (QiniuException ex) {
    //如果遇到异常，说明删除失败
    System.err.println(ex.code());
    System.err.println(ex.response.toString());
}
```

## 1.2.7. 封装工具类

为了方便操作七牛云存储服务，我们可以将官方提供的案例简单改造成一个工具类，在我们的项目中直接使用此工具类来操作就可以：

```
package com.itheima.health.utils;

import com.google.gson.Gson;
import com.itheima.health.constant.MessageConstant;
import com.qiniu.common.QiniuException;
import com.qiniu.common.Zone;
import com.qiniu.http.Response;
import com.qiniu.storage.BucketManager;
import com.qiniu.storage.Configuration;
import com.qiniu.storage.UploadManager;
import com.qiniu.storage.model.BatchStatus;
import com.qiniu.storage.model.DefaultPutRet;
import com.qiniu.storage.model.FileInfo;
import com.qiniu.util.Auth;

import java.util.ArrayList;
import java.util.List;

public class QiNiuUtils {

    private static final String ACCESSKEY = "SFEru-
42dYFEvmHot4dz1UhsepDmOPDql0AfFt8p";
    private static final String SECRETKEY =
"aCzett8JLbldF42PhQGzw6ymQyWreY4Fu8drPKG3";
    private static final String BUCKET = "sz98";
    public static final String DOMAIN= "http://qfhy5itom.hn-
bkt.clouddn.com/";

    public static void main(String[] args) {

 //uploadFile("C:\\Users\\Eric\\Desktop\\file\\timg.jpg","dd2.jpg");
        //removeFiles("20190529083159.jpg","20190529083241.jpg");
        listFile();
    }

    /**
     * 遍历7牛上的所有图片
     * @return
     */
    public static List<String> listFile(){
        BucketManager bucketManager = getBucketManager();
        //列举空间文件列表，第一个参数：图片的仓库（空间名）,第二个参数，文件名前缀过
滤。“”代理所有
```

```
        BucketManager.FileListIterator fileListIterator =
bucketManager.createFileListIterator(BUCKET,"");
        List<String> imageFiles = new ArrayList<String>();
        while (fileListIterator.hasNext()) {
            //处理获取的file list结果
            FileInfo[] items = fileListIterator.next();
            for (FileInfo item : items) {
                // item.key 文件名
                imageFiles.add(item.key);
                System.out.println(item.key);
            }
        }
        return imageFiles;
    }

    /**
     * 批量删除
     * @param filenames 需要删除的文件名列表
     * @return 删除成功的文件名列表
     */
    public static List<String> removeFiles(String... filenames){
        // 删除成功的文件名列表
        List<String> removeSuccessList = new ArrayList<String>();
        if(filenames.length > 0){
            // 创建仓库管理器
            BucketManager bucketManager = getBucketManager();
            // 创建批处理器
            BucketManager.Batch batch = new BucketManager.Batch();
            // 批量删除多个文件
            batch.delete(BUCKET,filenames);
            try {
                // 获取服务器的响应
                Response res = bucketManager.batch(batch);
                // 获得批处理的状态
                BatchStatus[] batchStatuses =
res.jsonToObject(BatchStatus[].class);
                for (int i = 0; i < filenames.length; i++) {
                    BatchStatus status = batchStatuses[i];
                    String key = filenames[i];
                    System.out.print(key + "\t");
                    if (status.code == 200) {
                        removeSuccessList.add(key);
                        System.out.println("delete success");
                    } else {
                        System.out.println("delete failure");
                    }
                }
            } catch (QiniuException e) {
                e.printStackTrace();
                throw new
RuntimeException(MessageConstant.PIC_UPLOAD_FAIL);
            }
        }
        return removeSuccessList;
    }

    public static void uploadFile(String localFilePath, String
savedFilename){
```

```
        UploadManager uploadManager = getUploadManager();
        String upToken = getToken();
        try {
            Response response = uploadManager.put(localFilePath, savedFilename, upToken);
            //解析上传成功的结果
            DefaultPutRet putRet = new Gson().fromJson(response.bodyString(), DefaultPutRet.class);
            System.out.println(String.format("key=%s, hash=%s",putRet.key, putRet.hash));
        } catch (QiniuException ex) {
            Response r = ex.response;
            System.err.println(r.toString());
            try {
                System.err.println(r.bodyString());
            } catch (QiniuException ex2) {
                //ignore
            }
            throw new RuntimeException(MessageConstant.PIC_UPLOAD_FAIL);
        }
    }

    public static void uploadViaByte(byte[] bytes, String savedFilename){
        UploadManager uploadManager = getUploadManager();
        String upToken = getToken();
        try {
            Response response = uploadManager.put(bytes, savedFilename, upToken);
            //解析上传成功的结果
            DefaultPutRet putRet = new Gson().fromJson(response.bodyString(), DefaultPutRet.class);
            System.out.println(putRet.key);
            System.out.println(putRet.hash);
        } catch (QiniuException ex) {
            Response r = ex.response;
            System.err.println(r.toString());
            try {
                System.err.println(r.bodyString());
            } catch (QiniuException ex2) {
                //ignore
            }
            throw new RuntimeException(MessageConstant.PIC_UPLOAD_FAIL);
        }
    }

    private static String getToken(){
        // 创建授权
        Auth auth = Auth.create(ACCESSKEY, SECRETKEY);
        // 获得认证后的令牌
        String upToken = auth.uploadToken(BUCKET);
        return upToken;
    }

    private static UploadManager getUploadManager(){
        //构造一个带指定Zone对象的配置类
        Configuration cfg = new Configuration(Zone.zone2());
        //构建上传管理器
        return new UploadManager(cfg);
```

```
    }

    private static BucketManager getBucketManager(){
        // 创建授权信息
        Auth auth = Auth.create(ACCESSKEY, SECRETKEY);
        // 创建操作某个仓库的管理器
        return new BucketManager(auth, new Configuration(Zone.zone2()));
    }

}

```

将此工具类放在health_common工程中，后续会使用到。

## 【小结】

1：介绍

（1） 传统文件存储与云存储方案的区别， 使用云存储

2：七牛云存储

（1）注册

（2）新建存储空间

（3）查看存储空间信息

（4）开发者中心 认证

（5）创建密钥： Access Key, Sercret Key

（6）Java SDK操作七牛云 QiNiuUitls

（7）封装工具类

QiNiuUitls 存到小抄

# 2. 新增套餐

## 【目标】

新增套餐

## 【路径】

1. 图片上传
    - 创建SetmealController接收/setmeal/upload.do, 接收上传的文件MultipartFile 参数名必须与前端中的name(imgFile)要一致
    - 获取原文件名才可以获取它后缀名
    - 产生唯一标识，拼接后缀名，唯一的文件名(七牛的仓库 )
    - 调用7牛utils上传文件
    - 返回数据给页面
        - 完整路径 QiNiuUtils.DOMAIN+唯一文件名 http://qiqhd7v6v.hn-bkt.clouddn.com/dd2.jpg

        - 将来提交formData时，缺少img图片名。所以上传文件成功后，给formData补全img的值dd2.jpg
        - 考虑以上2点，返回的数据格式

```
{
    flag
    message
    data:{
        domain:  http://qiqhd7v6v.hn-bkt.clouddn.com/
        imgName: dd2.jpg
    }
}
```

    - 上传成功后，页面会回调handleAvatarSuccess
        - 回显图片 this.imageUrl=domain +imgName;
        - 给formData补全img this.formData.img=imgName

2. 提交数据
    - 新建 弹出添加的窗口，重置表单，加载检查组列表数据
        - CheckGroupController- service-dao，添加findAll方法
    - 确定，提交formData,checkgroupIds 到后，提示操作的结果，如果成功则要关闭新增窗口，刷新列表数据
    - SetmealController 用Setmeal接收formData, Integer[] 接收checkgroupIds ，调用服务添加，返回操作的结果给页面
    - SetmealService
        - 先添加套餐信息，获取id
        - 遍历循环checkgroupIds ，添加套餐与检查组的关系
        - 事务控制
    - SetmealDao
        - 提代添加套餐
        - 添加套餐与检查组的关系

## 【讲解】

# 2.1. 需求分析

套餐其实就是检查组的集合，例如有一个套餐为“入职体检套餐”，这个检查组可以包括多个检查组：一般检查、血常规、尿常规、肝功三项等。

所以在添加套餐时需要选择这个套餐包括的检查组。

套餐对应的实体类为Setmeal，

```java
public class Setmeal implements Serializable {
    private Integer id;
    private String name;
    private String code;
    private String helpCode;
    private String sex;//套餐适用性别：0不限 1男 2女
    private String age;//套餐适用年龄
    private Float price;//套餐价格
    private String remark;
    private String attention;
    private String img;//套餐对应图片名称（用于存放七牛云上的图片名称-唯一）
    private List<CheckGroup> checkGroups;//体检套餐对应的检查组，多对多关系
}
```

其中img字段表示套餐对应图片存储路径（用于存放七牛云上的图片名称）

对应的数据表为t_setmeal。套餐和检查组为多对多关系，所以需要中间表t_setmeal_checkgroup进行关联。

t_setmeal与t_setmeal_checkgroup表

 image-20200909155828388

# 2.2. 前台代码

套餐管理页面对应的是setmeal.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。

 img

## 2.2.1. 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接口显示出新增窗口。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。

由于新增套餐时还需要选择此套餐包含的检查组，所以新增套餐窗口分为两部分信息：基本信息和检查组信息，如下图：

 img

 img

（1）：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>
```

（2）：handleCreate()方法：

```
// 重置表单
resetForm() {
    // 清空表单内容
    this.formData = {};
    // 清空选中的检查组
    this.checkgroupIds = [];
```

```
    // 默认展示套餐基本信息标签页
    this.activeName = 'first';
    // 清除选中的图片
    this.imageUrl = '';
    this.formData.img = '';
},
// 弹出添加窗口
handleCreate() {
    this.resetForm();
    //弹出添加窗口
    this.dialogFormVisible = true;
},
```

## 2.2.2. 动态展示检查组列表

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

（1）定义模型数据

```
tableData:[],//添加表单窗口中检查组列表数据
checkgroupIds:[],//添加表单窗口中检查组复选框对应id
```

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

```
<el-tab-pane label="检查组信息" name="second">
<div class="checkScrol">
   <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>
      <!--循环遍历tableData-->
      <tr v-for="c in tableData">
         <td>
            <!--复选框绑定checkgroupIds，存放到值是id-->
            <input :id="c.id" v-model="checkgroupIds" 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>
</div>
</el-tab-pane>
```

其中：v-model="checkgroupIds"，用于回显复选框。

（3）完善handleCreate方法，发送ajax请求查询所有检查组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.resetForm();
    //弹出添加窗口
    this.dialogFormVisible = true;
    // 加载检查组列表数据
    axios.get('/checkgroup/findAll.do').then(res => {
        if(res.data.flag){
            this.tableData = res.data.data;
        }else{
            this.$message.error(res.data.message);
        }
    })
},
```

（4）分别在CheckGroupController、CheckGroupService、CheckGroupServiceImpl、CheckGroupDao、CheckGroupDao.xml中扩展方法查询所有检查组数据

1：CheckGroupController：

```
@GetMapping("/findAll")
public Result findAll(){
    List<CheckGroup> all = checkGroupService.findAll();
    return new Result(true, MessageConstant.QUERY_CHECKGROUP_SUCCESS,all);
}
```

2：CheckGroupService：

```
/**
 * 查询所有检查组
 * @return
 */
List<CheckGroup> findAll();
```

3：CheckGroupServiceImpl：

```
/**
 * 查询所有检查组
 * @return
 */
@Override
public List<CheckGroup> findAll() {
    return checkGroupDao.findAll();
}
```

4：CheckGroupDao：

```
/**
 * 查询所有检查组
 * @return
 */
List<CheckGroup> findAll();
```

5：CheckGroupDao.xml：

```
<select id="findAll" resultType="checkgroup">
    select * From t_checkgroup
</select>
```

## 2.2.3. 图片上传并预览

此处使用的是ElementUI提供的上传组件el-upload，提供了多种不同的上传效果，上传成功后可以进行预览。

实现步骤：

（1）定义模型数据，用于后面上传文件的图片预览：

```
imageUrl:null,//模型数据，用于上传图片完成后图片预览
```

（2）定义ElementUI上传组件：

```
<el-form-item label="上传图片">
    <!-- action: 提交图片的url, 接收图片的止传
        auto-upload: 是否自动上传文件，选择完文件后就马上上传到action的url里
        name: 上传文件的参数名, MultipartFile imgFile
        show-file-list: 显示上传的文件列表，这里只需要一张图片，不要显示
        on-success: 上传成功后回调的方法
        before-upload: 上传前调用的方法
    -->
    <el-upload
            class="avatar-uploader"
            action="/setmeal/upload.do"
            :auto-upload="autoUpload"
            name="imgFile"
            :show-file-list="false"
            :on-success="handleAvatarSuccess"
            :before-upload="beforeAvatarUpload">
        <!--  图片回显，如果imageUrl有值，则输出img标签
        上传图片成功后要用imageUrl赋值 完整的路径
        后台响应回来的数据格式  imageUrl=domain+imgName
        {
           flag:
           message:
           data: {
              imgName: 图片名称  保存到数据库
              domain: 七牛的域名
           }
        }
        -->
        <img v-if="imageUrl" :src="imageUrl" class="avatar">
```

```
        <i v-else class="el-icon-plus avatar-uploader-icon"></i>
    </el-upload>
</el-form-item>
```

(3) 修改上传成功后回调的方法:

```
handleAvatarSuccess(response, file) {
    // 提示成功或失败，要回显图片
    this.$message({
        message: response.message,
        type: response.flag?"success":"error"
    })
    if(response.flag){
        // 回显图片
        this.imageUrl = response.data.domain + response.data.imgName;
        // 表单中没有img参数，后台数据库用的就是img字段，补上它的值
        // 前端中的json数据 {key,value}=> map,
        // 对象.属性名(不存在) => map.put(不存在的key,value)
        this.formData.img = response.data.imgName;
    }
}
```

(4) 创建SetmealController，接收上传的文件

```
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.service.SetmealService;
import com.itheima.health.utils.QiNiuUtils;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

/**
 * Description: No Description
 * User: Eric
 */
@RestController
@RequestMapping("/setmeal")
public class SetmealController {

    @Reference
    private SetmealService setmealService;

    /**
     * 套餐上传图片
     * @param imgFile
     * @return
```

```
     */
    @PostMapping("/upload")
    public Result upload(MultipartFile imgFile){
        //- 获取原有图片名称，截取到后缀名
        String originalFilename = imgFile.getOriginalFilename();
        String extension =
originalFilename.substring(originalFilename.lastIndexOf("."));
        //- 生成唯一文件名，拼接后缀名
        String filename = UUID.randomUUID() + extension;
        //- 调用七牛上传文件
        try {
            QiNiuUtils.uploadViaByte(imgFile.getBytes(), filename);
            //- 返回数据给页面
            //{
            //    flag:
            //    message:
            //    data:{
            //        imgName: 图片名,
            //        domain: QiNiuUtils.DOMAIN
            //    }
            //}
            Map<String,String> map = new HashMap<String,String>();
            map.put("imgName",filename);
            map.put("domain", QiNiuUtils.DOMAIN);
            return new Result(true, MessageConstant.PIC_UPLOAD_SUCCESS,map);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return new Result(false, MessageConstant.PIC_UPLOAD_FAIL);
    }
}

```

注意：别忘了在spring配置文件中配置文件上传组件

已在springmvc.xml中配置

```
<!--文件上传组件-->
<bean id="multipartResolver"

 class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="104857600" /><!--最大上传文件大小-->
    <property name="maxInMemorySize" value="4096" />
    <property name="defaultEncoding" value="UTF-8"/>
</bean>
```

## 2.2.4. 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：套餐基本信息（对应的模型数据为formData）和检查组id数组（对应的模型数据为checkgroupIds）。

（1）为确定按钮绑定单击事件，对应的处理函数为handleAdd

```
<div slot="footer" class="dialog-footer">
    <el-button @click="dialogFormVisible = false">取消</el-button>
    <el-button type="primary" @click="handleAdd()">确定</el-button>
</div>
```

（2）完善handleAdd方法

```
//添加提交
handleAdd () {
    axios.post('/setmeal/add.do?checkgroupIds=' + this.checkgroupIds, this.formData).then(res=>{
        this.$message({
            message: res.data.message,
            type: res.data.flag?"success":"error"
        })
        if(res.data.flag){
            // 关闭添加窗口
            this.dialogFormVisible = false;
            // 刷新列表数据
            this.findPage();
        }
    })
},
```

# 2.3. 后台代码

## 2.3.1. Controller

在SetmealController中增加方法

```
@PostMapping("/add")
public Result add(@RequestBody Setmeal setmeal, Integer[] checkgroupIds){
    // 调用业务服务添加
    setmealService.add(setmeal, checkgroupIds);
    // 响应结果
    return new Result(true, MessageConstant.ADD_SETMEAL_SUCCESS);
}
```

## 2.3.2. 服务接口

创建SetmealService接口并提供新增方法

```
package com.itheima.health.service;

import com.itheima.health.pojo.Setmeal;

import java.util.List;

/**
 * Description: No Description
 * User: Eric
 */
```

```
public interface SetmealService {
    /**
     * 添加套餐
     * @param setmeal
     * @param checkgroupIds
     */
    void add(Setmeal setmeal, Integer[] checkgroupIds);
}
```

## 2.3.3. 服务实现类

创建SetmealServiceImpl服务实现类并实现新增方法

```
package com.itheima.health.service.impl;

import com.alibaba.dubbo.config.annotation.Service;
import com.itheima.health.dao.SetmealDao;
import com.itheima.health.pojo.Setmeal;
import com.itheima.health.service.SetmealService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;

/**
 * Description: No Description
 * User: Eric
 */
@Service(interfaceClass = SetmealService.class)
public class SetmealServiceImpl implements SetmealService {

    @Autowired
    private SetmealDao setmealDao;

    /**
     * 添加套餐
     * @param setmeal
     * @param checkgroupIds
     */
    @Override
    @Transactional
    public void add(Setmeal setmeal, Integer[] checkgroupIds) {
        // 添加套餐信息
        setmealDao.add(setmeal);
        // 获取套餐的id
        Integer setmealId = setmeal.getId();
        // 添加套餐与检查组的关系
        if(null != checkgroupIds){
            for (Integer checkgroupId : checkgroupIds) {
                setmealDao.addSetmealCheckGroup(setmealId, checkgroupId);
            }
        }
    }
}

```

## 2.3.4. Dao接口

创建SetmealDao接口并提供相关方法

```
package com.itheima.health.dao;

import com.itheima.health.pojo.Setmeal;
import org.apache.ibatis.annotations.Param;

/**
 * Description: No Description
 * User: Eric
 */
public interface SetmealDao {
    /**
     * 添加套餐
     * @param setmeal
     */
    void add(Setmeal setmeal);

    /**
     * 添加套餐与检查组的关系
     * @param setmealId
     * @param checkgroupId
     */
    void addSetmealCheckGroup(@Param("setmealId") Integer setmealId,
@Param("checkgroupId") Integer checkgroupId);

}

```

## 2.3.5. Mapper映射文件

创建SetmealDao.xml文件并定义相关SQL语句

```
<?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.SetmealDao">
    <insert id="add" parameterType="setmeal">
        <selectKey resultType="int" keyProperty="id" order="AFTER">
            select last_insert_id()
        </selectKey>
        insert into t_setmeal
(name,code,helpCode,sex,age,price,remark,attention,img)
        values(#{name},#{code},#{helpCode},#{sex},#{age},#{price},#
{remark},#{attention},#{img})
    </insert>

    <insert id="addSetmealCheckGroup" parameterType="int">
        insert into t_setmeal_checkgroup (setmeal_id, checkgroup_id)
        values (#{setmealId},#{checkgroupId})
    </insert>
</mapper>
```

## 【小结】

1. 上传到七牛上的图片名称要唯一

2. elementUI的upload组件成功时会回调handleAvatarSuccess(response) 经过elementUI处理过的，response=>Result.
3. 套餐中的图片：页面中要显示（全路径），数据库只存储它的名称，form表单对不会提交，没有img, 补全formData补全img。

# 3. 体检套餐分页

## 【目标】

体检套餐列表分页

## 【路径】

1：前台代码

（1）定义分页相关模型数据 pagination, QueryPageBean

（2）定义分页方法 findPage

（3）完善分页方法执行时机 加载，查询(1), 页码变更，添加成功后，修改成功，删除后

2：后台代码

业务：

- 体检套餐分页列表展示

findPage方法：

1. 提交查询条件分页信息 pagination, 提示结果，且把返回的结果集res.data.data.rows绑定dataList, 总计录数绑定pagination.total
2. SetmealController接收 QueryPageBean, 调用业务服务查询 pageResult, 包装到Result返回给前端
3. SetmealServiceImp 使用PageHelper分页，有查询条件实现模糊查询 拼接%，调用dao查询，返回PageResult对象
4. SetmealDao findByCondition

## 【讲解】

# 3.1. 前台代码

## 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

（1）在页面中提供了findPage方法用于分页查询，为了能够在setmeal.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();
},
```

（2）findPage()方法：

```
//分页查询
findPage() {
    axios.post('/setmeal/findPage.do', this.pagination).then(res => {
        if(res.data.flag){
            // 绑定分页的结果集
            this.dataList = res.data.data.rows;
            // 总计录数
            this.pagination.total = res.data.data.total;
        }else{
            this.$message.error(res.data.message);
        }
    })
},
```

## 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。

（1）为查询按钮绑定单击事件，调用handleCurrentChange方法，同时查询回到第一页

```
<el-button @click="handleCurrentChange(1)" class="dalfBut">查询</el-button>
```

（2）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();
}
```

## 3.2. 后台代码

### 3.2.1. Controller

在SetmealController中增加分页查询方法

```
/**
 * 分页查询
 */
@PostMapping("/findPage")
public Result findPage(@RequestBody QueryPageBean queryPageBean){
    // 调用服务分页查询
    PageResult<Setmeal> pageResult = setmealService.findPage(queryPageBean);
    return new Result(true,
MessageConstant.QUERY_SETMEAL_SUCCESS,pageResult);
}
```

### 3.2.2. 服务接口

在SetmealService服务接口中扩展分页查询方法

```
/**
 * 分页查询
 * @param queryPageBean
 * @return
 */
PageResult<Setmeal> findPage(QueryPageBean queryPageBean);
```

### 3.2.3. 服务实现类

在SetmealServiceImpl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

```
/**
 * 分页查询
 * @param queryPageBean
 * @return
 */
@Override
public PageResult<Setmeal> findPage(QueryPageBean queryPageBean) {
    PageHelper.startPage(queryPageBean.getCurrentPage(),
queryPageBean.getPageSize());
    // 查询条件
    if(!StringUtils.isEmpty(queryPageBean.getQueryString())){
        // 模糊查询 %
        queryPageBean.setQueryString("%" + queryPageBean.getQueryString() +
"%");
    }
    // 条件查询，这个查询语句会被分页
    Page<Setmeal> page =
setmealDao.findByCondition(queryPageBean.getQueryString());
    return new PageResult<Setmeal>(page.getTotal(), page.getResult());
}
```

### 3.2.4. Dao接口

在SetmealDao接口中扩展分页查询方法

```
/**
 * 条件查询
 * @param queryString
 * @return
 */
Page<Setmeal> findByCondition(String queryString);
```

### 3.2.5. Mapper映射文件

在SetmealDao.xml文件中增加SQL定义

```
<select id="findByCondition" parameterType="String" resultType="setmeal">
    select * From t_setmeal
    <if test="value != null and value.length > 0">
        where code like #{queryString} or name like #{queryString} or
helpCode like #{queryString}
    </if>
</select>
```

### 【小结】

1：前台代码

（1）定义分页相关模型数据

（2）定义分页方法

（3）完善分页方法执行时机

# 4. 编辑套餐

### 【目标】

编辑套餐

### 【路径】

(0) 补全编辑窗口、编辑按钮事件，删除事件

（1）绑定“编辑”单击事件

（2）弹出编辑窗口回显数据, 表单初始化

- 回显套餐数据, 图片预览(setmeal.img=图片名称) 设置图片的完整路径 {setmeal: setmeal, imageUrl: domain+setmeal.getImg()}

```
{
    flag:
    message:
    data:{
         setmeal: setmeal,
         domain: QiNiuUtils.DOMAIN
    }
}
this.formData = res.data.data.setmeal
this.imageUrl =  res.data.data.domain +  this.formData.img
```

- 查询检查组列表
- 当前套餐具有的检查组的ID, 供复选框需要选中

(3) 发送请求，编辑保存套餐

- 编辑保存套餐 提交formData, checkgroupIds
- 删除套餐检查组中间表数据，重新添加套餐检查组中间表数据
- 事务控制

**【讲解】**

# 4.1. 前台页面

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，除了需要将套餐基本信息的回显之外，还需要回显当前套餐包含的检查组（以复选框勾选的形式回显）。

## 4.1.1. 绑定单击事件

(1) 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-table-column label="操作" align="center">
    <template slot-scope="scope">
        <el-button type="primary" size="mini"
@click="handleUpdate(scope.row)">编辑</el-button>
        <el-button size="mini" type="danger"
@click="handleDelete(scope.row)">删除</el-button>
    </template>
</el-table-column>
```

(2) 在methods中添加handleUpdate与handleDelete方法

```
// 弹出编辑窗口
handleUpdate(row) {
    alert(row.id);
},
// 删除
handleDelete(){
    alert("点了删除");
},
```

## 4.1.2. 弹出编辑窗口回显数据

编写当前页面的编辑窗口，可以复制新增套餐窗口进行修改，默认需要处于隐藏状态。

（1）在vue的data中定义模型

```
dialogFormVisible4Edit:false//控制编辑窗口显示/隐藏
```

（2）定义编辑窗口页面

在新增窗口的下方 添加编辑弹出窗口，【注意】控制窗口展示的变量dialogFormVisible4Edit

```
<div class="edit-form">
    <el-dialog title="编辑套餐" :visible.sync="dialogFormVisible4Edit">
        <template>
            <el-tabs v-model="activeName" type="card">
                <el-tab-pane label="基本信息" name="first">
                    <el-form label-position="right" label-width="100px">
                        <el-row>
                            <el-col :span="12">
                                <el-form-item label="编码">
                                    <el-input v-model="formData.code"/>
                                </el-form-item>
                            </el-col>
                            <el-col :span="12">
                                <el-form-item label="名称">
                                    <el-input v-model="formData.name"/>
                                </el-form-item>
                            </el-col>
                        </el-row>
                        <el-row>
                            <el-col :span="12">
                                <el-form-item label="适用性别">
                                    <el-select v-model="formData.sex">
                                        <el-option label="不限" value="0">
</el-option>
                                        <el-option label="男" value="1">
</el-option>
                                        <el-option label="女" value="2">
</el-option>
                                    </el-select>
                                </el-form-item>
                            </el-col>
                            <el-col :span="12">
                                <el-form-item label="助记码">
                                    <el-input v-
model="formData.helpCode"/>
                                </el-form-item>
                            </el-col>
                        </el-row>
                        <el-row>
                            <el-col :span="12">
                                <el-form-item label="套餐价格">
                                    <el-input v-model="formData.price"/>
                                </el-form-item>
```

```
                            </el-col>
                            <el-col :span="12">
                                <el-form-item label="适用年龄">
                                    <el-input v-model="formData.age"/>
                                </el-form-item>
                            </el-col>
                        </el-row>
                        <el-row>
                            <el-col :span="24">
                                <el-form-item label="上传图片">
                                    <el-upload
                                            class="avatar-uploader"
                                            action="/setmeal/upload.do"
                                            :auto-upload="autoUpload"
                                            name="imgFile"
                                            :show-file-list="false"
                                            :on-
success="handleAvatarSuccess"
                                            :before-
upload="beforeAvatarUpload">
                                        <img v-if="imageUrl"
:src="imageUrl" class="avatar">
                                        <i v-else class="el-icon-plus
avatar-uploader-icon"></i>
                                    </el-upload>
                                </el-form-item>
                            </el-col>
                        </el-row>
                        <el-row>
                            <el-col :span="24">
                                <el-form-item label="说明">
                                    <el-input v-model="formData.remark"
type="textarea"></el-input>
                                </el-form-item>
                            </el-col>
                        </el-row>
                        <el-row>
                            <el-col :span="24">
                                <el-form-item label="注意事项">
                                    <el-input v-model="formData.attention"
type="textarea"></el-input>
                                </el-form-item>
                            </el-col>
                        </el-row>
                    </el-form>
                </el-tab-pane>
                <el-tab-pane label="检查组信息" name="second">
                    <div class="checkScrol">
                        <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="checkgroupIds" 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>
                        </div>
                    </el-tab-pane>
                </el-tabs>
            </template>
            <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>
        </el-dialog>
</div>
```

在vue中的methods添加确定事件的方法

```
// 编辑提交
handleEdit(){

}
```

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请求分别查询当前套餐数据、所有检查组列表数据、当前套餐包含的检查组id用于基本数据回显

```
// 弹出编辑窗口
handleUpdate(row){
    this.resetForm();
    this.dialogFormVisible4Edit = true;
    // 套餐的id
    var id = row.id;
    axios.get("/setmeal/findById.do?id=" + id).then(res => {
        if(res.data.flag){
            // 回显套餐信息
            // res.data.data => resultMap {setmeal, domain}
            this.formData = res.data.data.setmeal;
            // 回显图片
            this.imageUrl = res.data.data.domain+this.formData.img;
            axios.get('/checkgroup/findAll.do').then(resp => {
                if(resp.data.flag){
                    this.tableData = resp.data.data;
                    // 获取选中的检查组id
                    axios.get('/setmeal/findCheckgroupIdsBySetmealId.do?id='
+ id).then(response => {
```

```
                        if(response.data.flag){
                            this.checkgroupIds = response.data.data;
                        }else{
                            this.$message.error(response.data.message);
                        }
                    })
                }else{
                    this.$message.error(resp.data.message);
                }
            })
        }else{
            this.$message.error(res.data.message);
        }
    })
},
```

### 4.1.3. 发送请求，编辑保存套餐

（1）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>
```

（2）handleEdit()方法

```
// 编辑提交
handleEdit(){
    //提交检查组信息this.formData, 选中的检查项id this.checkgroupIds
    axios.post('/setmeal/update.do?checkgroupIds=' + this.checkgroupIds,
this.formData).then(res => {
        this.$message({
            message: res.data.message,
            type: res.data.flag?"success":"error"
        })
        if(res.data.flag){
            // 关闭编辑窗口
            this.dialogFormVisible4Edit = false;
            // 刷新列表数据
            this.findPage();
        }
    })
}
```

## 4.2. 后台代码

### 4.2.1. Controller

在SetmealController中增加方法

```
/**
 * 通过id查询套餐信息
 */
@GetMapping("/findById")
```

```
public Result findById(int id){
    // 调用服务查询
    Setmeal setmeal = setmealService.findById(id);
    // 前端要显示图片需要全路径
    // 封装到map中，解决图片路径问题
    Map<String,Object> resultMap = new HashMap<String,Object>();
    resultMap.put("setmeal", setmeal); // formData
    resultMap.put("domain", QiNiuUtils.DOMAIN); // domain
    return new Result(true,
MessageConstant.QUERY_SETMEAL_SUCCESS,resultMap);
}

/**
 * 通过id查询选中的检查组ids
 */
@GetMapping("/findCheckgroupIdsBySetmealId")
public Result findCheckgroupIdsBySetmealId(int id){
    List<Integer> checkgroupIds =
setmealService.findCheckgroupIdsBySetmealId(id);
    return new Result(true,
MessageConstant.QUERY_CHECKGROUP_SUCCESS,checkgroupIds);
}

/**
 * 修改
 */
@PostMapping("/update")
public Result update(@RequestBody Setmeal setmeal, List<Integer>[]
checkgroupIds){
    // 调用业务服务修改
    setmealService.update(setmeal, checkgroupIds);
    // 响应结果
    return new Result(true, MessageConstant.EDIT_SETMEAL_SUCCESS);
}
```

## 4.2.2. 服务接口

在SetmealService服务接口中扩展方法

```
/**
 * 通过id查询
 * @param id
 * @return
 */
Setmeal findById(int id);

/**
 * 通过id查询选中的检查组ids
 * @param id
 * @return
 */
List<Integer> findCheckgroupIdsBySetmealId(int id);

/**
 * 修改套餐
```

```
 * @param setmeal
 * @param checkgroupIds
 */
void update(Setmeal setmeal, Integer[] checkgroupIds);
```

### 4.2.3. 服务实现类

在SetmealServiceImpl实现类中实现编辑方法

```
@Override
public Setmeal findById(int id) {
    return setmealDao.findById(id);
}

/**
 * 通过id查询选中的检查组ids
 * @param id
 * @return
 */
@Override
public List<Integer> findCheckgroupIdsBySetmealId(int id) {
    return setmealDao.findCheckgroupIdsBySetmealId(id);
}

/**
 * 修改套餐
 * @param setmeal
 * @param checkgroupIds
 */
@Override
@Transactional
public void update(Setmeal setmeal, Integer[] checkgroupIds) {
    // 先更新套餐信息
    setmealDao.update(setmeal);
    // 删除旧关系
    setmealDao.deleteSetmealCheckGroup(setmeal.getId());
    // 添加新关系
    if(null != checkgroupIds){
        for (Integer checkgroupId : checkgroupIds) {
            setmealDao.addSetmealCheckGroup(setmeal.getId(), checkgroupId);
        }
    }
}
```

### 4.2.4. Dao接口

在SetmealDao接口中扩展方法

```
/**
 * 通过id查询
 * @param id
 * @return
 */
Setmeal findById(int id);

/**
```

```
 * 通过id查询选中的检查组ids
 * @param id
 * @return
 */
List<Integer> findCheckgroupIdsBySetmealId(int id);

/**
 * 更新套餐信息
 * @param setmeal
 */
void update(Setmeal setmeal);

/**
 * 删除旧关系
 * @param id
 */
void deleteSetmealCheckGroup(Integer id);
```

## 4.2.5. Mapper映射文件

在SetmealDao.xml中扩展SQL语句

```
<select id="findById" parameterType="int" resultType="setmeal">
    select * From t_setmeal where id=#{id}
</select>

<select id="findCheckgroupIdsBySetmealId" parameterType="int"
resultType="int">
    select checkgroup_id from t_setmeal_checkgroup where setmeal_id=#{id}
</select>

<update id="update" parameterType="setmeal">
    update t_setmeal
    set
        name=#{name},
        code=#{code},
        helpCode=#{helpCode},
        sex=#{sex},
        age=#{age},
        price=#{price},
        remark=#{remark},
        attention=#{attention},
        img=#{img}
    where id=#{id}
</update>

<delete id="deleteSetmealCheckGroup" parameterType="int">
    delete from t_setmeal_checkgroup where setmeal_id=#{id}
</delete>
```

## 【小结】

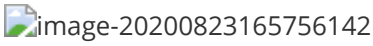
- 添加 绑定“编辑”单击事件

  image-20200624163134301

- 复制新增窗口，改为编辑窗口，添加dialogFormVisable4Edit定义为编辑窗口的展示

（2）弹出编辑窗口回显数据， 复制添加的窗口: 确定按钮绑定的事件，编辑窗口的显示控制的变量dialogFormVisiable4Edit(窗口定义，data定义)，在methods中添加handleUpdate方法(methods), 添加handleEdit方法(methods)

- 回显套餐数据

  
- 查询检查组列表
- 当前套餐具有的检查组的复选框需要选中

（3）发送请求，编辑保存套餐

- 编辑套餐更新
- 删除套餐检查组中间表数据，重新添加套餐检查组中间表数据

# 5.删除套餐

## 【目标】

删除套餐

## 【路径】

1：前台代码

- 给删除按钮(实现编辑套餐功能时添加的)绑定删除handleDelete事件
- 获取删除的套餐id
- 弹出询问，确定后发送请求把id传给controller, 提示结果，成功则刷新列表

2：后台代码

- SetmealController接收id, 调用服务删除，返回结果给前端
- SetmealServiceImpl
  - 判断是否被订单使用了
  - 使用了，则报错，接口要做异常抛出声明
  - 没使用
    - 先删除套餐与检查组的关系
    - 再删除套餐数据
  - 事务控制
- SetmealDao
  - 查询订单表中是否存在这个套餐的id select count(1) from t_order where setmeal_id=?id
  - delete from t_setmeal where id=#{id}

## 【讲解】

# 5.1. 前台代码

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。

## 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

```
<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);
}
```

## 5.1.2. 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.$confirm('此操作将【永久删除】该套餐，是否继续?', '提示', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning',
        center: true
    }).then(() => {
        // 执行删除
    }).catch(() => {
        this.$message({
            type: 'info',
            message: '已取消删除'
        });
    });
}
```

## 5.1.3. 发送请求

如果用户点击确定按钮就需要发送ajax请求，并且将当前套餐的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('此操作将【永久删除】该套餐，是否继续?', '提示', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning',
        center: true
```

```
    }).then(() => {
        // 使用id作为查询条件，删除数据
        axios.get("/setmeal/delete.do?id="+row.id).then((response)=>{
            // 返回的结果Result(flag,message,data)
            if(response.data.flag){
                this.$message({
                type: 'success',
                message: response.data.message
            });
            }else{
                this.$message({
                    type: 'error',
                    message: response.data.message
                });
            }
            // 刷新页面
            this.findPage();
        })
    }).catch(() => {
        this.$message({
            type: 'info',
            message: '已取消删除'
        });
    });
}
```

# 5.2. 后台代码

health_common添加常量定义

 image-20200626095524675

## 5.2.1. Controller

在SetmealController中增加删除方法

```
// 删除套餐
@RequestMapping(value = "/delete")
public Result delete(Integer id){
    setmealService.deleteById(id);
    return new Result(true, MessageConstant.DELETE_SETMEAL_SUCCESS);
}
```

## 5.2.2. 服务接口

在SetmealService服务接口中扩展删除方法

```
void deleteById(Integer id) throws HealthException;
```

## 5.2.3. 服务实现类

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

SetmealServiceImpl.java

```java
// 删除套餐
@Override
@Transactional
public void deleteById(Integer id) throws HealthException {
    // 是否存在订单，如果存在则不能删除
    int count = setmealDao.findOrderCountBySetmealId(id);
    if(count > 0){
        // 已经有订单使用了这个套餐，不能删除
        throw new HealthException("已经有订单使用了这个套餐，不能删除！");
    }
    // 先删除套餐与检查组的关系
    setmealDao.deleteSetmealCheckGroup(id);
    // 再删除套餐
    setmealDao.deleteById(id);
}
```

## 5.2.4. Dao接口

在SetmealDao接口中扩展方法findSetmealAndCheckGroupCountBySetmealId和deleteById

```java
/**
 * 通过套餐的id查询使用了这个套餐的订单个数
 * @param id
 * @return
 */
int findOrderCountBySetmealId(int id);

/**
 * 通过id删除套餐信息
 * @param id
 */
void deleteById(int id);

```

## 5.2.5. Mapper映射文件

在SetmealDao.xml中扩展SQL语句

```xml
<select id="findOrderCountBySetmealId" parameterType="int" resultType="int">
    select count(1) from t_order where setmeal_id=#{id}
</select>

<delete id="deleteById" parameterType="int">
    delete from t_setmeal where id=#{id}
</delete>
```

## 【小结】

是否能删除 判断的点在于是否有订单使用了这个套餐，而不是套餐与检查组的关系

注意：删除的业务里加上事务控制，删除的接口要加异常声明