第3章 模板与分类管理

学习目标

- 使用“黑马架构师”完成代码生成
- 完成规格参数模板管理
- 完成商品分类管理
- 完成图片库管理

1. 代码生成器-”黑马架构师“

1.1 ”黑马架构师“介绍

”黑马架构师“一款由传智播客教育集团研究院自主研发的“代码生成器”。即便是一个工程几百个表，也可以瞬间完成基础代码的构建！用户只需建立数据库表结构，简单的几步操作就可以快速生成可以运行的一整套代码，可以极大地缩短开发周期，降低人力成本。”黑马架构师“内置了当前java和前端主流的架构模板，如SSM+dubbo 、springBoot+springCloud+springData 、前后端分离的vue+elementUI 模板、swagger API模板、数据库文档模板等。 用户通过自己开发模板也可以实现生成php、python、C# 、c++、数据库存储过程等其它编程语言的代码。

https://gitee.com/chuanzhiliubei/codeutil

1.2 代码生成

1.2.1 安装

1. 安装jdk1.8 并配置环境变量
2. 双击codeutil-2.5.jar 或start.bat运行程序（注意不要放在有中文和空格的目录运行）

1.2.2 使用步骤

1. 建立数据库表，并设置字段的中文备注（中文备注用于作为生成表格的标题）
2. 双击codeutil-2.5.jar 或start.bat运行程序，输入用户名和密码，点击测试连接，选

择数据库表后点击“下一步”。
3. 选择模板，输入基本信息，注意：包名一定要3级，例如 com.itheima.goods 前两级为模块名称，第三级为模块名称。
4. 点击生成后，用IDE 打开代码后即可运行。

1.2.3 分库分表工程生成

我们《青橙》采用的是分库分表设计，有多个数据库，如何生成？

我们可以分别连接每一个数据库生成代码，生成代码到同一位置，这样公共模块不变。最后修改pom.xml的模块列表，将所有生成的模块都加入即可。

2. 规格参数模板管理

2.1 概念与需求

2.1.1 概念解析

了解需求前我们先弄懂三个概念

规格： 规格是用于区分同一商品的属性。 例如手机的网络制式、屏幕尺寸等。

规格选项：规格选项是规格的具体值。比如网络制式是规格，它包括的规格选项有移动4G、联通4G、电信4G等。

参数： 参数是用于描述商品的属性。例如手机的核数

参数选项：参数选项是参数的可选值。比如核数，包括参数选项有2核 、4核、8核

模板：类似于商品类型。比如手机、电视、笔记本等。每个模板都有多个规格和参数。例如手机包括网络制式、屏幕尺寸等规格，还包括核数、前置摄像头像素、后置摄像头像素等。

2.1.2 需求分析

详见静态原型。

（1）实现模板的增删改查

（2）从模板列表中点击规格列表，进入规格列表页面

（3）从模板列表中点击参数列表，进入参数列表页面

2.2 表结构分析

tb_template （模板表）

字段名称	字段含义	字段类型	字段长度	备注
id	ID	INT		
name	模板名称	VARCHAR		
spec_num	规格数量	INT		
para_num	参数数量	INT		

tb_spec （规格表）

字段名称	字段含义	字段类型	字段长度	备注
id	ID	INT		
name	名称	VARCHAR		
options	规格选项	VARCHAR		
seq	排序	INT		
template_id	模板ID	INT		

tb_para （参数表）

字段名称	字段含义	字段类型	字段长度	备注
id	id	INT		
name	名称	VARCHAR		
options	参数选项	VARCHAR		
seq	排序	INT		
template_id	模板ID	INT		

2.3 代码实现

2.3.1 规格参数模板列表查询

（1）修改template.html页面，添加方法

```
goSpec(id){ //规格管理
    location.href="spec.html?templateId="+id;
},
goPara(id){ //参数管理
    location.href="para.html?templateId="+id;
}
```

在表格的模板列中添加两个按钮

```
<el-button @click="goSpec(scope.row.id)" type="text" size="small">规格管理</el-button>
<el-button @click="goPara(scope.row.id)" type="text" size="small">参数管理</el-button>
```

适当调整列宽度，让其看起来舒服一点

（2）修改spec.html，引入util.js

```
<script src="/js/util.js"></script>
```

util.js中提供了可以获取地址栏参数的方法getQueryString

修改created钩子函数，添加代码

```
this.searchMap={templateId: getQueryString("templateId")};
```

添加返回按钮

```
<el-button type="primary" onclick="location.href='template.html'">返回模板</el-button>
```

（3）修改parameter.html，步骤同spec.html

2.3.2 添加规格

（1）修改spec.html页面，data添加属性templateId

```
templateId: 0
```

（2）created()钩子函数添加代码

```
this.templateId=getQueryString("templateId");
```

（3）修改新增按钮

```
<el-button type="primary" @click="formVisible=true;pojo={templateId:
templateId}">新增</el-button>
```

（4）删除表单中模板id

（5）将表单中规格选项文本框改为文本域

```
<el-form-item label="规格选项">
  <el-input v-model="pojo.options" type="textarea"  :autosize="{ minRows:
4, maxRows: 15}"> </el-input>
</el-form-item>
```

（6）修改save方法，在开始处添加代码，将回车符\n转换为逗号","

```
this.pojo.options=  this.pojo.options.replace(/\n/g,",");  //回车替换为逗号
```

（7）修改edit方法，在回调处添加代码，将逗号“，”替换为回车符\n

```
this.pojo.options=  this.pojo.options.replace(/,/g,"\n");  //逗号替换为回车
符
```

2.3.3 添加参数

思路同"添加规格"，代码略

2.3.4 规格与参数数量统计

需求：在每次添加和删除规格参数时，对规格和参数数量进行更新

实现思路：在每次添加规格和参数时，修改模板表对应的字段值，让其加1，在删除规格和参数时，修改模板表对应的字段值，让其减1

代码实现：

（1）修改TemplateServiceImpl的add方法

```
    /**
     * 新增
     * @param template
     */
    public void add(Template template) {
        template.setSpecNum(0);
        template.setParaNum(0);
        templateMapper.insert(template);
    }
```

（2）修改SpecServiceImpl的add方法

```
    /**
     * 新增
     * @param spec
     */
    @Transactional
    public void add(Spec spec) {
        specMapper.insert(spec);
        //将模板中的规格数量+1
        Template template =
templateMapper.selectByPrimaryKey(spec.getTemplateId());
        template.setSpecNum( template.getSpecNum()+1 );
        templateMapper.updateByPrimaryKey(template);
    }
```

（3）修改SpecServiceImpl的delete方法

```
    /**
     *  删除
     * @param id
     */
    @Transactional
    public void delete(Integer id) {
        //将模板中的规格数量减一
        Spec spec = specMapper.selectByPrimaryKey(id);
        Template template =
templateMapper.selectByPrimaryKey(spec.getTemplateId());
        template.setSpecNum( template.getSpecNum()-1 );
        templateMapper.updateByPrimaryKey(template);

        specMapper.deleteByPrimaryKey(id);
    }
```

（4）修改SpecServiceImpl类的注解@Service

```
@Service(interfaceClass = SpecService.class)
```

3. 商品分类

3.1 需求分析

详见静态原型。

3.2 表结构分析

tb_category （商品分类表）

字段名称	字段含义	字段类型	字段长度	备注
id	分类ID	INT		
name	分类名称	VARCHAR		
goods_num	商品数量	INT		
is_show	是否显示	CHAR		
is_menu	是否导航	CHAR		
seq	排序	INT		
parent_id	上级ID	INT		
template_id	模板ID	INT		

3.3 代码实现

3.3.1 三级分类列表展示

（1）查询一级分类列表。修改searchMap的属性，删除查询表单

```
searchMap: {parentId:0}
```

这样打开category.html就看到一级分类的列表了

（2）查询下级列表：新增方法

```
queryByParentId(parentId){

    this.searchMap.parentId=parentId;
    this.fetchData();//加载数据
},
```

表格的模板列新增"查询下级"按钮

```
<el-button @click="queryByParentId(scope.row.id)" type="text"
size="small">查询下级</el-button>
```

（3）返回上级列表

新增属性，用于记录点击的上级ID

```
parentIds:[]
```

修改方法queryByParentId，添加代码

```
this.parentIds.push(this.searchMap.parentId)
```

新增方法

```
returnQuery(){  //上级查询
    //获取上级ID
    if(this.parentIds.length>0){
        this.searchMap.parentId= this.parentIds[this.parentIds.length-1]
        this.parentIds.splice( this.parentIds.length-1,1 );//截掉最后一个
        this.fetchData();//加载数据
    }
}
```

新增按钮

```
<el-button type="primary" @click="returnQuery()">返回上级</el-button>
```

（4）三级列表不显示“查询下级”链接，在查询下级按钮上添加条件

```
<el-button v-if="parentIds.length<2"
@click="queryByParentId(scope.row.id)" type="text" size="small">查询下级
</el-button>
```

（5）修改fetchData方法

```
 axios.post(`/category/findList.do`,this.searchMap).then(response => {
      this.tableData = response.data
 });
```

3.3.2 新增分类

（1）添加新增按钮

```
<el-button type="primary" @click="formVisible=true;pojo={parentId:
searchMap.parentId}">新增</el-button>
```

（2）删除表单中的上级ID列

（3）使用开关控件控制是否显示和是否导航

```
        <el-form-item label="是否显示">
            <el-switch
                    v-model="pojo.isShow"
                    active-color="#13ce66"
                    inactive-color="#ff4949"
                    active-value="1"
                    inactive-value="0">
            </el-switch>
        </el-form-item>
        <el-form-item label="是否导航">
            <el-switch
                    v-model="pojo.isMenu"
                    active-color="#13ce66"
                    inactive-color="#ff4949"
                    active-value="1"
                    inactive-value="0">
            </el-switch>
        </el-form-item>
```

（4）下拉列表显示

```
<el-form-item label="模板ID">
    <el-select v-model="pojo.templateId" filterable placeholder="请选择">
        <el-option
                v-for="item in templateList"
                :key="item.id"
                :label="item.name"
                :value="item.id">
        </el-option>
    </el-select>
</el-form-item>
```

3.3.3 完善列表显示

（1）显示级别，添加模板列

```
<el-table-column label="级别" width="80">
  <template slot-scope="scope">
      {{parentIds.length+1}}
  </template>
</el-table-column>
```

（2）显示是否显示和是否菜单

```
 <el-table-column label="是否显示" width="80">
      <template slot-scope="scope">
          <el-switch
                  v-model="scope.row.isShow"
                  active-color="#13ce66"
                  inactive-color="#ff4949"
                  active-value="1"
                  inactive-value="0">
          </el-switch>
      </template>
  </el-table-column>
  <el-table-column  label="是否导航" width="80">
      <template slot-scope="scope">
          <el-switch
                  v-model="scope.row.isMenu"
                  active-color="#13ce66"
                  inactive-color="#ff4949"
                  active-value="1"
                  inactive-value="0">
          </el-switch>
      </template>
  </el-table-column>
```

（3）模板下拉列表

```
<el-table-column label="模板" width="300">
    <template slot-scope="scope">
        <el-select v-model="scope.row.templateId" disabled>
            <el-option
                    v-for="item in templateList"
                    :key="item.id"
                    :label="item.name"
                    :value="item.id">
            </el-option>
        </el-select>
    </template>
</el-table-column>
```

3.3.4 分类删除

需求：在删除某个分类前要判断这个分类下是否存在下级分类，如果有下级分类则不能删除。

修改CategoryServiceImpl的delete方法

```
/**
 *  删除
 * @param id
 */
public void delete(Integer id) {
    //判断是否存在下级分类
    Example example=new Example(Category.class);
    Example.Criteria criteria = example.createCriteria();
    criteria.andEqualTo("parentId",id);
    int count = categoryMapper.selectCountByExample(example);
    if(count>0){
        throw  new RuntimeException("存在下级分类不能删除");
    }
    categoryMapper.deleteByPrimaryKey(id);
}
```

4. 图片库管理

4.1 需求分析

图片库是做什么的？图片库也可以称为相册，是用于存储商品图片的空间，一个图片库（相册）下有多张图片。我们通常是将每一个商品建立一个图片库。

详见静态原型。

4.2 表结构分析

tb_album （相册表）

字段名称	字段含义	字段类型	字段长度	备注
id	编号	BIGINT		
title	相册名称	VARCHAR		
image	相册封面	VARCHAR		
image_items	图片列表	TEXT		图片用逗号分隔

4.3 代码实现（作业）

4.3.1 相册名称列表

实现步骤：

（1）修改album.html ，精简查询表单，只保留相册名称查询

（2）删除图片列表列

（3）修改封面为模板显示

（4）添加设置列，列中有“图片列表‘按钮，点击图片按钮执行list方法，传递id

（5）添加list方法 ，实现页面跳转 ，跳转到album_list.html并传递id

（6）取消注释的图片上传相关的代码，实现封面的上传

4.3.2 图片列表管理

实现步骤：

（1）新增album_list.html页面，接收参数id

（2）根据id查询相册的图片列表，字符串转换为数组

（3）遍历图片列表

（4）实现图片的上传功能，上传后追加到图片列表，并保存到相册。

（5）实现删除相册图片的功能。