还在从零开始搭建项目？手撸了款快速开发脚手架！

原创 梦想de星空 macrozheng 2020-09-09 09:02

收录于合集
#开源之路

20个

> 之前开源了一款项目骨架 `mall-tiny` ，完整继承了 `mall` 项目的整个技术栈。总感觉 `mall-tiny` 集成了太多中间件，过于复杂了。这次对其进行了简化和升级，使它成为了一款拥有完整权限管理功能的快速开发脚手架，希望对大家有所帮助！

简介

`mall-tiny` 是一款基于SpringBoot+MyBatis-Plus的快速开发脚手架，拥有完整的权限管理功能，可对接Vue前端，开箱即用。

项目演示

`mall-tiny` 项目可无缝对接 `mall-admin-web` 前端项目，秒变权限管理系统。前端项目地址：https://github.com/macrozheng/mall-admin-web

技术选型

技术	版本	说明
SpringBoot	2.3.0	容器+MVC框架
SpringSecurity	5.3.2	认证和授权框架
MyBatis	3.5.4	ORM框架
MyBatis-Plus	3.3.2	MyBatis增强工具
MyBatis-Plus Generator	3.3.2	数据层代码生成器
Swagger-UI	2.9.2	文档生产工具
Redis	5.0	分布式缓存
Docker	18.09.0	应用容器引擎
Druid	1.1.10	数据库连接池
JWT	0.9.0	JWT登录支持
Lombok	1.18.12	简化对象封装工具

数据库表结构

- 化繁为简，仅保留了权限管理功能相关的9张表，方便自由定制；
- 数据库源文件地址：https://github.com/macrozheng/mall-tiny/blob/master/sql/mall_tiny.sql

使用流程

环境搭建

简化依赖服务，只需安装最常用的MySql和Redis服务即可，服务安装具体参考**《mall在Windows环境下的部署》**，数据库中需要导入 `mall_tiny.sql` 脚本。

开发规约

项目包结构

```
src
├── common -- 用于存放通用代码
|   ├── api -- 通用结果集封装类
|   ├── config -- 通用配置类
|   ├── domain -- 通用封装对象
|   ├── exception -- 全局异常处理相关类
|   └── service -- 通用业务类
├── config -- SpringBoot中的Java配置
├── domain -- 共用封装对象
├── generator -- MyBatis-Plus代码生成器
├── modules -- 存放业务代码的基础包
|   └── ums -- 权限管理模块业务代码
|       ├── controller -- 该模块相关接口
|       ├── dto -- 该模块数据传输封装对象
|       ├── mapper -- 该模块相关Mapper接口
|       ├── model -- 该模块相关实体类
|       └── service -- 该模块相关业务处理类
└── security -- SpringSecurity认证授权相关代码
```

```
├── annotation -- 相关注解
├── aspect -- 相关切面
├── component -- 认证授权相关组件
├── config -- 相关配置
└── util -- 相关工具类
```

资源文件说明

```
resources
├── mapper -- MyBatis中mapper.xml存放位置
├── application.yml -- SpringBoot通用配置文件
├── application-dev.yml -- SpringBoot开发环境配置文件
├── application-prod.yml -- SpringBoot生产环境配置文件
└── generator.properties -- MyBatis-Plus代码生成器配置
```

接口定义规则

- 创建表记录：POST /{控制器路由名称}/create
- 修改表记录：POST /{控制器路由名称}/update/{id}
- 删除指定表记录：POST /{控制器路由名称}/delete/{id}
- 分页查询表记录：GET /{控制器路由名称}/list
- 获取指定记录详情：GET /{控制器路由名称}/{id}
- 具体参数及返回结果定义可以运行代码查看Swagger-UI的Api文档：http://localhost:8080/swagger-ui.html

项目运行

直接运行启动类 `MallTinyApplication` 的 `main` 函数即可。

业务代码开发流程

创建业务表

> 创建好 `pms` 模块的所有表，需要注意的是一定要写好表字段的 `注释` ，这样实体类和接口文档中就会自动生成字段说明了。

使用代码生成器

> 运行 `MyBatisPlusGenerator` 类的main方法来生成代码，可直接生成controller、service、mapper、model、mapper.xml的代码，无需手动创建。

- 代码生成器支持两种模式，一种生成单表的代码，比如只生成 `pms_brand` 表代码可以先输入 `pms` ，后输入 `pms_brand` ；

- 生成代码结构一览；

- 另一种直接生成整个模块的代码，比如生成 `pms` 模块代码可以先输入 `pms` ，后输入 `pms_*` 。

编写业务代码

单表查询

> 由于MyBatis-Plus提供的增强功能相当强大，单表查询几乎不用手写SQL，直接使用ServiceImpl和BaseMapper中提供的方法即可。

比如我们的菜单管理业务实现类 `UmsMenuServiceImpl` 中的方法都直接使用了这些方法。

```
/**
 * 后台菜单管理Service实现类
 * Created by macro on 2020/2/2.
 */
@Service
public class UmsMenuServiceImpl extends ServiceImpl<UmsMenuMapper,UmsMenu>implements UmsMenuServi

    @Override
    public boolean create(UmsMenu umsMenu) {
        umsMenu.setCreateTime(new Date());
        updateLevel(umsMenu);
        return save(umsMenu);
    }

    @Override
    public boolean update(Long id, UmsMenu umsMenu) {
        umsMenu.setId(id);
        updateLevel(umsMenu);
        return updateById(umsMenu);
    }

    @Override
    public Page<UmsMenu> list(Long parentId, Integer pageSize, Integer pageNum) {
        Page<UmsMenu> page = new Page<>(pageNum,pageSize);
        QueryWrapper<UmsMenu> wrapper = new QueryWrapper<>();
        wrapper.lambda().eq(UmsMenu::getParentId,parentId)
                .orderByDesc(UmsMenu::getSort);
        return page(page,wrapper);
```

```
    }

    @Override
    public List<UmsMenuNode> treeList() {
        List<UmsMenu> menuList = list();
        List<UmsMenuNode> result = menuList.stream()
                .filter(menu -> menu.getParentId().equals(0L))
                .map(menu -> covertMenuNode(menu, menuList)).collect(Collectors.toList());
        return result;
    }

    @Override
    public boolean updateHidden(Long id, Integer hidden) {
        UmsMenu umsMenu = new UmsMenu();
        umsMenu.setId(id);
        umsMenu.setHidden(hidden);
        return updateById(umsMenu);
    }
}
```

分页查询

> 对于分页查询MyBatis-Plus原生支持，不需要再整合其他插件，直接构造Page对象，然后调用ServiceImpl中的page方法即可。

```
/**
 * 后台菜单管理Service实现类
 * Created by macro on 2020/2/2.
 */
@Service
public class UmsMenuServiceImpl extends ServiceImpl<UmsMenuMapper,UmsMenu>implements UmsMenuServi
    @Override
    public Page<UmsMenu> list(Long parentId, Integer pageSize, Integer pageNum) {
        Page<UmsMenu> page = new Page<>(pageNum,pageSize);
        QueryWrapper<UmsMenu> wrapper = new QueryWrapper<>();
        wrapper.lambda().eq(UmsMenu::getParentId,parentId)
                .orderByDesc(UmsMenu::getSort);
        return page(page,wrapper);
    }
}
```

多表查询

> 对于多表查询，我们需要手写mapper.xml中的SQL实现，由于之前我们已经生成了mapper.xml文件，所以我们直接在Mapper接口中定义好方法，然后在mapper.xml写好SQL实现即可。

- 比如说我们需要写一个根据用户ID获取其分配的菜单的方法，首先我们在 `UmsMenuMapper` 接口中添加好 `getMenuList` 方法；

```
/**
 * <p>
 * 后台菜单表 Mapper 接口
 * </p>
 *
 * @author macro
 * @since 2020-08-21
 */
public interface UmsMenuMapper extends BaseMapper<UmsMenu> {

    /**
     * 根据后台用户ID获取菜单
     */
    List<UmsMenu> getMenuList(@Param("adminId") Long adminId);

}
```

- 然后在 `UmsMenuMapper.xml` 添加该方法的对应SQL实现即可。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-ma
<mapper namespace="com.macro.mall.tiny.modules.ums.mapper.UmsMenuMapper">

    <select id="getMenuList" resultType="com.macro.mall.tiny.modules.ums.model.UmsMenu">
        SELECT
            m.id id,
            m.parent_id parentId,
            m.create_time createTime,
            m.title title,
            m.level level,
```

```
            m.sort sort,
            m.name name,
            m.icon icon,
            m.hidden hidden
        FROM
            ums_admin_role_relation arr
                LEFT JOIN ums_role r ON arr.role_id = r.id
                LEFT JOIN ums_role_menu_relation rmr ON r.id = rmr.role_id
                LEFT JOIN ums_menu m ON rmr.menu_id = m.id
        WHERE
            arr.admin_id = #{adminId}
          AND m.id IS NOT NULL
        GROUP BY
            m.id
    </select>

</mapper>
```

项目部署

mall-tiny已经集成了Docker插件，可以打包成Docker镜像来部署，具体参考：《使用**Maven**插件为**SpringBoot**应用构建**Docker**镜像》

其他说明

SpringSecurity相关

> 由于使用了SpringSecurity来实现认证和授权，部分接口需要token才可以访问，访问需要认证授权接口流程如下。

- 访问Swagger-UI接口文档：http://localhost:8080/swagger-ui.html
- 调用登录接口获取token；

- 点击右上角Authorize按钮输入token，然后访问相关接口即可。

请求参数校验

> 默认集成了 `Jakarta Bean Validation` 参数校验框架，只需在参数对象属性中添加 `javax.validation.constraints` 包中的注解注解即可实现校验功能，这里以登录参数校验为例。

- 首先在登录请求参数中添加 `@NotEmpty` 注解；

```
/**
 * 用户登录参数
 * Created by macro on 2018/4/26.
```

```
 */
@Data
@EqualsAndHashCode(callSuper = false)
public class UmsAdminLoginParam {
    @NotEmpty
    @ApiModelProperty(value = "用户名",required = true)
    private String username;
    @NotEmpty
    @ApiModelProperty(value = "密码",required = true)
    private String password;
}
```

- 然后在登录接口中添加 `@Validated` 注解开启参数校验功能即可。

```
/**
 * 后台用户管理
 * Created by macro on 2018/4/26.
 */
@Controller
@Api(tags = "UmsAdminController", description = "后台用户管理")
@RequestMapping("/admin")
public class UmsAdminController {

    @ApiOperation(value = "登录以后返回token")
    @RequestMapping(value = "/login", method = RequestMethod.POST)
    @ResponseBody
    public CommonResult login(@Validated @RequestBody UmsAdminLoginParam umsAdminLoginParam) {
        String token = adminService.login(umsAdminLoginParam.getUsername(), umsAdminLoginParam.get
        if (token == null) {
            return CommonResult.validateFailed("用户名或密码错误");
        }
        Map<String, String> tokenMap = new HashMap<>();
        tokenMap.put("token", token);
        tokenMap.put("tokenHead", tokenHead);
        return CommonResult.success(tokenMap);
    }
}
```

项目地址

> 开源不易，觉得项目有帮助的点个 `Star` 支持下吧！

https://github.com/macrozheng/mall-tiny

推荐阅读

- 还在手写**CRUD**代码？这款开源框架助你解放双手！
- 不要再重复造轮子了，这款开源工具类库贼好使！
- 还在手动部署**SpringBoot**应用？试试这个自动化插件！
- 为什么我要从 **Windows** 切换到 **Linux?**
- 一键生成数据库文档，堪称数据库界的**Swagger**，有点厉害**!**
- 面对成百上千台服务器产生的日志，试试这款轻量级日志搬运神器！
- 居然有人想白嫖我的日志，赶紧开启安全保护压压惊！
- **mall-swarm** 微服务电商项目发布重大更新，打造**Spring Cloud**最佳实践！
- **Mall** 电商实战项目发布重大更新，全面支持**SpringBoot 2.3.0** ！
- 我的**Github**开源项目，从**0**到**20000 Star**！

欢迎关注，点个在看

阅读原文

喜欢此内容的人还喜欢

项目中到底该不该用Lombok?

macrozheng