使用springboot3.X+spring security6+ JWT+MyBatisPlus搭建一个后端基础脚手架(3)
==========

代码生成

代码生成是一个争论不断的话题，有人对策很推崇，有人嗤之以鼻，我不参与争论，利用MyBatisPlus的工具，实验了一把，感觉还行。
功劳全是MyBatis Plus团队的，我只是一个搬运工，详细参照官网：
[baomidou.com/pages/98140...](http://cxyroad.com/”https://baomidou.com/pages/981406/”)

创建一个生成代码用的项目

新建一个项目，把相关的包导进来：

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.5.5</version>
```

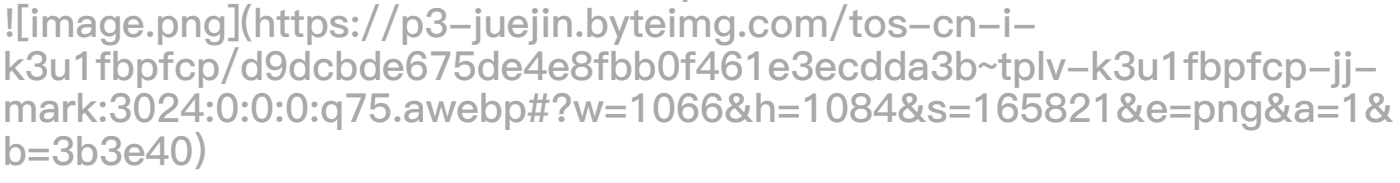
```
  </dependency>
  <dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-generator</artifactId>
    <version>3.5.5</version>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
  </dependency>
  <dependency>
    <groupId>org.freemarker</groupId>
    <artifactId>freemarker</artifactId>
    <version>2.3.31</version>
  </dependency>

</dependencies>
```

生成代码的起点不是main函数，是junit测试用例。
我根据上面工程的特点，做了一些定制。

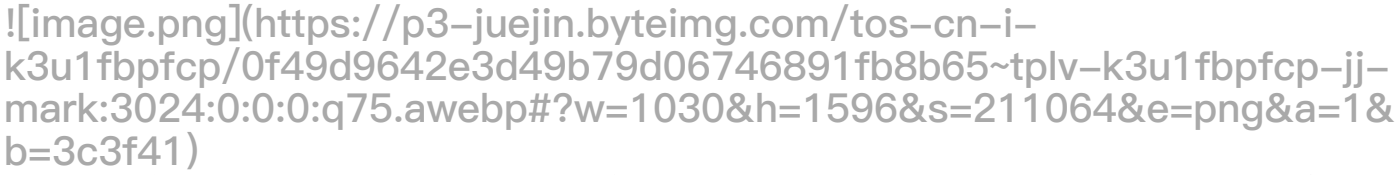
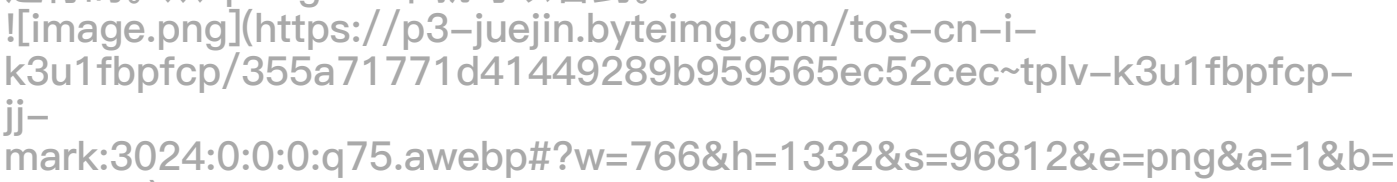

```
package com.sptan.gen;

import com.baomidou.mybatisplus.generator.FastAutoGenerator;
import com.baomidou.mybatisplus.generator.config.DataSourceConfig;
import com.baomidou.mybatisplus.generator.config.OutputFile;
import org.junit.jupiter.api.Test;

import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

/**
 * MySQL 代码生成
 *
 * @author lp
 * @since 3.5.3
 */
public class MySQLGeneratorTest extends BaseGeneratorTest {
```

```
    private static final String url =
"jdbc:mysql://localhost:3306/ssmp?characterEncoding=utf8&useSSL=fa
lse&serverTimezone=Asia/Shanghai";
    private static final String username = "root";

    private static final String password = "1234qweR";


    @Test
    public void testSimple() {
        FastAutoGenerator generator = FastAutoGenerator.create(url,
username, password);
        generator.strategyConfig(builder -> {
            builder
            .addTablePrefix("sys_")
            .addInclude("sys_menu")
            .entityBuilder()
            .enableLombok()
            // 这些是共通字段
            .addSuperEntityColumns("id", "delete_flag", "version", "ctime",
"utime", "cuid", "opuid")
            // entity的基类
.superClass("com.sptan.framework.mybatis.entity.AbstractFocusBaseEnti
ty")
            .serviceBuilder()
            .formatServiceFileName("%sService")
            .controllerBuilder()
            .enableRestStyle()
            .build();
        });
        generator.globalConfig(builder -> {
            builder
            .author("lp")
            .enableSpringdoc()
            .commentDate("yyyy-MM-dd")
            // 代码生成路径，是我们上面讨论的工程的代码
            .outputDir("/Users/liupeng/dev/springboot3-springsecurity6-
mybatisplus/src/main/java")
            .build();
        });

        generator.templateConfig(builder -> {
            // service和Controller使用了自定义的模板
            builder
            .service("templates/service.java")
            .serviceImpl("templates/serviceImpl.java")
            .controller("templates/controller.java")
```

```
            .build();
        });

        generator.packageConfig(builder -> {
            // entity的模板定制内容，改变了默认的xml文件放置位置
            builder
            .parent("com.sptan.ssmp")
            .entity("domain")
            .pathInfo(Collections.singletonMap(OutputFile.xml,
                                    "/Users/liupeng/dev/springboot3-
springsecurity6-mybatisplus/src/main/resources/mapper"))
            .build();
        });
        generator.injectionConfig(builder -> {
            Map<String, String> customFile = new HashMap<>();
            // 新增了DTO、VO的生成
            customFile.put("dto:DTO", "/templates/dto.java.ftl");
            customFile.put("dto:Criteria", "/templates/criteria.java.ftl");
            customFile.put("converter:Converter",
"/templates/converter.java.ftl");
            builder.customFile(customFile);
        });

        generator.templateEngine(new
EnhanceFreemarkerTemplateEngine());

        generator.execute();
    }
}


```

因为我们项目中使用到了DTO和查询条件，并且引入了MapStruct，这些内容是MyBatis Plus默认没有提供的，所以需要我们自定义模板。service和Controller层也根据我们项目需要更新了模板。
自定义的模板都放置在resources/templates目录中。


我们生成一下menu的相关代码，我们一次先限定生成一个表对应的代码，这里很灵活，可以生成全库的代码。


=2b2b2b)
执行这个测试用例

看到生成了很多类，由于我针对我们项目调试过模板了，生成的代码是直接可运行的。从springdoc中就可以看到。

看看生成的代码：

```
package com.sptan.ssmp.domain;

import com.baomidou.mybatisplus.annotation.TableName;
import com.sptan.framework.mybatis.entity.AbstractFocusBaseEntity;
import io.swagger.v3.oas.annotations.media.Schema;
import lombok.Getter;
import lombok.Setter;

/**
 * <p>
 * 菜单表
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
@Getter
@Setter
@TableName("sys_menu")
@Schema(name = "Menu", description = "菜单表")
public class Menu extends AbstractFocusBaseEntity {

    private static final long serialVersionUID = 1L;

    @Schema(description = "父级ID")
    private Long parentId;

    @Schema(description = "菜单或者按钮名称")
    private String name;
```

```
    @Schema(description = "节点类型，1文件夹，2页面，3按钮, 4:子页面或者页面元素")
    private Integer nodeType;

    @Schema(description = "图标地址")
    private String iconUrl;

    @Schema(description = "页面对应的前端地址")
    private String linkUrl;

    @Schema(description = "层级")
    private Integer level;

    @Schema(description = "树id的路径 整个层次上的路径id，逗号分隔")
    private String fullPath;

    @Schema(description = "启用状态：0->禁用；1->启用")
    private Integer status;

    @Schema(description = "排序")
    private Integer sort;
}
```

mapper：

```
package com.sptan.ssmp.mapper;

import com.sptan.ssmp.domain.Menu;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;

/**
 * <p>
 * 菜单表 Mapper 接口
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
public interface MenuMapper extends BaseMapper<Menu> {

}
```

```

```
<?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.sptan.ssmp.mapper.MenuMapper”>

</mapper>
```

```

MapStruct:

```
package com.sptan.ssmp.converter;

import com.sptan.ssmp.domain.Menu;
import com.sptan.ssmp.dto.MenuDTO;
import org.mapstruct.Mapper;
import org.mapstruct.factory.Mappers;

import java.util.List;

/**
 * <p>
 * 菜单表 Mapstruct转换接口.
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
@Mapper(componentModel = ”spring”)
public interface MenuConverter {

    /**
     * The constant INSTANCE.
     */
    MenuConverter INSTANCE =
Mappers.getMapper(MenuConverter.class);
```

```
    /**
     * To dto base station dto.
     *
     * @param entity the entity
     * @return the base station dto
     */
    MenuDTO toDto(Menu entity);

    /**
     * dto to entity.
     *
     * @param dto the entity
     * @return the base station brief dto
     */
    Menu toEntity(MenuDTO dto);

    /**
     * To dto list.
     *
     * @param entities the entities
     * @return the list
     */
    List<MenuDTO> toDtoList(List<Menu> entities);

    /**
     * To entity list.
     *
     * @param dtos the dtos
     * @return the list
     */
    List<Menu> toEntities(List<MenuDTO> dtos);
}
```

Service层：

```
package com.sptan.ssmp.service;

import com.sptan.ssmp.domain.Menu;
import com.baomidou.mybatisplus.extension.service.IService;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.sptan.ssmp.dto.MenuDTO;
```

```
import com.sptan.ssmp.dto.MenuCriteria;
import com.sptan.framework.core.ResultEntity;

/**
 * <p>
 * 菜单表 服务类.
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
public interface MenuService extends IService<Menu> {

    /**
     * 保存.
     *
     * @param dto the dto
     * @return the result entity
     */
    ResultEntity<MenuDTO> save(MenuDTO dto);

    /**
     * 删除.
     *
     * @param id the id
     * @return the result entity
     */
    ResultEntity<Boolean> delete(Long id);

    /**
     * 查看详情.
     *
     * @param id the id
     * @return the result entity
     */
    ResultEntity<MenuDTO> detail(Long id);

    /**
     * 根据条件查询.
     *
     * @param criteria the criteria
     * @return the result entity
     */
    ResultEntity<IPage<MenuDTO>> search(MenuCriteria criteria);
}


```

```
package com.sptan.ssmp.service.impl;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.sptan.ssmp.converter.MenuConverter;
import com.sptan.ssmp.domain.Menu;
import com.sptan.ssmp.dto.MenuCriteria;
import com.sptan.framework.core.ResultEntity;
import com.sptan.framework.mybatis.page.PageCriteria;
import com.sptan.ssmp.dto.MenuDTO;
import com.sptan.ssmp.mapper.MenuMapper;
import com.sptan.ssmp.service.MenuService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;

import java.util.Objects;

/**
 * <p>
 * 菜单表 服务实现类.
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
@Service
@Slf4j
public class MenuServiceImpl extends ServiceImpl<MenuMapper, Menu>
    implements MenuService {

    /**
     * 保存.
     *
     * @param dto the dto
     * @return the result entity
     */
    @Override
    public ResultEntity<MenuDTO> save(MenuDTO dto) {
        Menu entity = MenuConverter.INSTANCE.toEntity(dto);
        this.saveOrUpdate(entity);
        return ResultEntity.ok(MenuConverter.INSTANCE.toDto(entity));
    }
```

```
    /**
     * 删除.
     *
     * @param id the id
     * @return the result entity
     */
    @Override
    public ResultEntity<Boolean> delete(Long id) {
        Menu entity = getById(id);
        if (entity == null || Objects.equals(true, entity.getDeleteFlag())) {
            return ResultEntity.error("数据不存在");
        }
        entity.setDeleteFlag(true);
        this.saveOrUpdate(entity);
        return ResultEntity.ok(true);
    }

    /**
     * 查看详情.
     *
     * @param id the id
     * @return the result entity
     */
    @Override
    public ResultEntity<MenuDTO> detail(Long id) {
        Menu entity = baseMapper.selectById(id);
        if (entity == null || Objects.equals(true, entity.getDeleteFlag())) {
            return ResultEntity.error("数据不存在");
        }
        MenuDTO dto = MenuConverter.INSTANCE.toDto(entity);
        return ResultEntity.ok(dto);
    }

    /**
     * 根据条件分页查询.
     *
     * @param criteria the criteria
     * @return the result entity
     */
    @Override
    public ResultEntity<IPage<MenuDTO>> search(MenuCriteria criteria) {
        LambdaQueryWrapper<Menu> wrapper =
getSearchWrapper(criteria);
        IPage<Menu> entityPage =
this.baseMapper.selectPage(PageCriteria.toPage(criteria), wrapper);
        return
ResultEntity.ok(entityPage.convert(MenuConverter.INSTANCE::toDto));
    }
```

```
    private LambdaQueryWrapper<Menu>
getSearchWrapper(MenuCriteria criteria) {
        LambdaQueryWrapper<Menu> wrapper = new
LambdaQueryWrapper<>();
        wrapper.eq(Menu::getDeleteFlag, false);
        if (criteria == null) {
            return wrapper;
        }
        return wrapper;
    }
}


```

DTO：

```
package com.sptan.ssmp.dto;

import io.swagger.v3.oas.annotations.media.Schema;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * <p>
 * 菜单表
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Schema(name = "Menu", description = "菜单表")
public class MenuDTO {

    @Schema(description = "ID")
    private Long id;

    @Schema(description = "父级ID")
```

```
    private Long parentId;

    @Schema(description = "菜单或者按钮名称")
    private String name;

    @Schema(description = "节点类型，1文件夹，2页面，3按钮, 4:子页面或者页面元素")
    private Integer nodeType;

    @Schema(description = "图标地址")
    private String iconUrl;

    @Schema(description = "页面对应的前端地址")
    private String linkUrl;

    @Schema(description = "层级")
    private Integer level;

    @Schema(description = "树id的路径 整个层次上的路径id，逗号分隔")
    private String fullPath;

    @Schema(description = "启用状态：0->禁用；1->启用")
    private Integer status;

    @Schema(description = "排序")
    private Integer sort;
}
```

```
package com.sptan.ssmp.dto;

import com.sptan.framework.mybatis.page.PageCriteria;
import io.swagger.v3.oas.annotations.media.Schema;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * <p>
 * 菜单表查询条件.
 * </p>
 *
 * @author lp
```

```
 * @since 2024-05-05
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Schema(name = "Menu", description = "菜单表")
public class MenuCriteria extends PageCriteria {

    @Schema(description = "ID")
    private Long id;

    @Schema(description = "父级ID")
    private Long parentId;

    @Schema(description = "菜单或者按钮名称")
    private String name;

    @Schema(description = "节点类型，1文件夹，2页面，3按钮, 4:子页面或者页面元素")
    private Integer nodeType;

    @Schema(description = "图标地址")
    private String iconUrl;

    @Schema(description = "页面对应的前端地址")
    private String linkUrl;

    @Schema(description = "层级")
    private Integer level;

    @Schema(description = "树id的路径 整个层次上的路径id，逗号分隔")
    private String fullPath;

    @Schema(description = "启用状态：0->禁用；1->启用")
    private Integer status;

    @Schema(description = "排序")
    private Integer sort;
}


```

Controller：

```
package com.sptan.ssmp.controller;

import com.baomidou.mybatisplus.core.metadata.IPage;
import com.sptan.framework.core.ResultEntity;
import com.sptan.ssmp.dto.MenuDTO;
import com.sptan.ssmp.dto.MenuCriteria;
import com.sptan.ssmp.service.MenuService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.Parameters;
import io.swagger.v3.oas.annotations.enums.ParameterIn;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

/**
 * <p>
 * 菜单表 前端控制器
 * </p>
 *
 * @author lp
 * @since 2024-05-05
 */
@RestController
@RequestMapping("/biz/menu")

@RequiredArgsConstructor
@Slf4j
public class MenuController {

    /**
     * The Menu service.
     */
    private final MenuService menuService;

    /**
     * 根据条件查询.
     *
     * @param criteria the criteria
     * @return the contract info
     */
    @PostMapping("/search")
    @Operation(summary = "条件查询", description = "根据条件分页查询")
    @Parameters({
        @Parameter(in = ParameterIn.HEADER, name = "Authorization",
```

```
description = ”标识用户信息的请求头”, required = true)
    })
    public ResponseEntity<ResultEntity<IPage<MenuDTO>>>
search(@Validated @RequestBody MenuCriteria criteria) {
        ResultEntity<IPage<MenuDTO>> resultEntity =
menuService.search(criteria);
        return ResponseEntity.ok(resultEntity);
    }

    /**
     * 查看详情.
     *
     * @param id the id
     * @return the response entity
     */
    @PostMapping(”/detail/{id}”)
    @Operation(summary = ”详情”, description = ”根据ID查看详情”)
    @Parameters({
        @Parameter(in = ParameterIn.HEADER, name = ”Authorization”,
description = ”标识用户信息的请求头”, required = true),
        @Parameter(in = ParameterIn.PATH, name = ”id”, description =
”ID”, required = true)
    })
    public ResponseEntity<ResultEntity<MenuDTO>>
detail(@PathVariable(”id”) Long id) {
        ResultEntity<MenuDTO> resultEntity = menuService.detail(id);
        return ResponseEntity.ok(resultEntity);
    }

    /**
     * 删除.
     *
     * @param id the id
     * @return the response entity
     */
     @PostMapping(”/delete/{id}”)
     @Operation(summary = ”删除”, description = ”根据ID逻辑删除对象”)
     @Parameters({
         @Parameter(in = ParameterIn.HEADER, name = ”Authorization”,
description = ”标识用户信息的请求头”, required = true),
         @Parameter(in = ParameterIn.PATH, name = ”id”, description =
”ID”, required = true)
     })
     public ResponseEntity<ResultEntity<Boolean>>
delete(@PathVariable(”id”) Long id) {
         ResultEntity<Boolean> resultEntity = menuService.delete(id);
         return ResponseEntity.ok(resultEntity);
     }
```

```
    /**
     * 保存.
     *
     * @param dto the dto
     * @return the response entity
     */
    @PostMapping("/save")
    @Operation(summary = "保存", description = "保存对象,包括新增和修改")
    @Parameters({
        @Parameter(in = ParameterIn.HEADER, name = "Authorization", description = "标识用户信息的请求头", required = true)
    })
    public ResponseEntity<ResultEntity<MenuDTO>> save(@Validated @RequestBody MenuDTO dto) {
        ResultEntity<MenuDTO> resultEntity = menuService.save(dto);
        return ResponseEntity.ok(resultEntity);
    }

}


```

虽然远远不能说是完美，但是生成的代码能直接运行，有springdoc注释，基本符合规范。最重要的是你可以随意定制，让它符合你自己的期望，并且一旦做好了定制，可以很迅速的生成普通的CRUD代码，能节省宝贵的时间。
这个小的工程我也传到码云上，感兴趣的可以看看。
[gitee.com/peng.liu.s/...](http://cxyroad.com/"https://gitee.com/peng.liu.s/gen-ssmp")