**指南**

**网站https://mp.baomidou.com/guide/**

**配置**

## [#](https://mp.baomidou.com/guide/quick-start.html" \l "配置)**配置**

在 application.yml 配置文件中添加 H2 数据库的相关配置：

# DataSource Configspring:

datasource:

driver-class-name: org.h2.Driver

schema: classpath:db/schema-h2.sql

data: classpath:db/data-h2.sql

url: jdbc:h2:mem:test

username: root

password: test

在 Spring Boot 启动类中添加 @MapperScan 注解，扫描 Mapper 文件夹：

@SpringBootApplication

@MapperScan("com.baomidou.mybatisplus.mapper")public class Application {

public static void main(String[] args) {

SpringApplication.run(QuickStartApplication.class, args);

}

}

编写实体类 User.java（此处使用了 [Lombok (opens new window)](https://www.projectlombok.org/" \t "https://mp.baomidou.com/guide/_blank)简化代码）

@Data

public class User {

private Long id;

private String name;

private Integer age;

private String email;}

编写Mapper类 UserMapper.java

public interface UserMapper extends BaseMapper<User> {

}

## [#](https://mp.baomidou.com/guide/quick-start.html" \l "开始使用)**开始使用**

添加测试类，进行功能测试：

@RunWith(SpringRunner.class)@SpringBootTestpublic class SampleTest {

@Autowired

private UserMapper userMapper;

@Test

public void testSelect() {

System.out.println(("----- selectAll method test ------"));

List<User> userList = userMapper.selectList(null);

Assert.assertEquals(5, userList.size());

userList.forEach(System.out::println);

}

}

**TIP**

UserMapper 中的 selectList() 方法的参数为 MP 内置的条件封装器 Wrapper，所以不填写就是无任何条件

**代码生成器**

**在com.TianzeJ9.项目名目录下建立一个CodeGenerator**

public static String scanner(String tip) {  
 Scanner scanner = new Scanner(System.*in*);  
 StringBuilder help = new StringBuilder();  
 help.append("请输入" + tip + "：");  
 System.*out*.println(help.toString());  
 if (scanner.hasNext()) {  
 String ipt = scanner.next();  
 if (StringUtils.*isNotBlank*(ipt)) {  
 return ipt;  
 }  
 }  
 throw new MybatisPlusException("请输入正确的" + tip + "！");  
}  
  
public static void main(String[] args) {  
 // 代码生成器  
 AutoGenerator mpg = new AutoGenerator();  
  
 // 全局配置  
 GlobalConfig gc = new GlobalConfig();  
 //用户的目录 固定得到  
 String projectPath = System.*getProperty*("user.dir");  
 gc.setOutputDir(projectPath + "/src/main/java");  
 gc.setAuthor("jobob");  
 gc.setOpen(false);  
 // gc.setSwagger2(true); 实体属性 Swagger2 注解  
 mpg.setGlobalConfig(gc);  
  
 // 数据源配置  
 DataSourceConfig dsc = new DataSourceConfig();  
 dsc.setUrl("jdbc:mysql://localhost:3306/xlsy?useUnicode=true&useSSL=false&characterEncoding=utf8");  
 // dsc.setSchemaName("public");  
 dsc.setDriverName("com.mysql.jdbc.Driver");  
 dsc.setUsername("root");  
 dsc.setPassword("1234");  
 mpg.setDataSource(dsc);  
  
 // 包配置  
 PackageConfig pc = new PackageConfig();  
 pc.setModuleName(*scanner*("模块名"));  
 //生成代码的父路径  
 pc.setParent("com.tianzej9.thymemybatis01");  
 mpg.setPackageInfo(pc);  
  
 // 自定义配置  
 InjectionConfig cfg = new InjectionConfig() {  
 @Override  
 public void initMap() {  
 // to do nothing  
 }  
 };  
  
 // 如果模板引擎是 freemarker  
 // String templatePath = "/templates/mapper.xml.ftl";  
 // 如果模板引擎是 velocity  
 String templatePath = "/templates/mapper.xml.vm";  
  
 // 自定义输出配置  
 List<FileOutConfig> focList = new ArrayList<>();  
 // 自定义配置会被优先输出  
 focList.add(new FileOutConfig(templatePath) {  
 @Override  
 public String outputFile(TableInfo tableInfo) {  
 // 自定义输出文件名 ， 如果你 Entity 设置了前后缀、此处注意 xml 的名称会跟着发生变化！！  
 return projectPath + "/src/main/resources/mapper/" + pc.getModuleName()  
 + "/" + tableInfo.getEntityName() + "Mapper" + StringPool.*DOT\_XML*;  
 }  
 });  
 /\*  
 cfg.setFileCreate(new IFileCreate() {  
 @Override  
 public boolean isCreate(ConfigBuilder configBuilder, FileType fileType, String filePath) {  
 // 判断自定义文件夹是否需要创建  
 checkDir("调用默认方法创建的目录，自定义目录用");  
 if (fileType == FileType.MAPPER) {  
 // 已经生成 mapper 文件判断存在，不想重新生成返回 false  
 return !new File(filePath).exists();  
 }  
 // 允许生成模板文件  
 return true;  
 }  
 });  
 \*/  
 cfg.setFileOutConfigList(focList);  
 mpg.setCfg(cfg);  
  
 // 配置模板  
 TemplateConfig templateConfig = new TemplateConfig();  
  
 // 配置自定义输出模板  
 //指定自定义模板路径，注意不要带上.ftl/.vm, 会根据使用的模板引擎自动识别  
 // templateConfig.setEntity("templates/entity2.java");  
 // templateConfig.setService();  
 // templateConfig.setController();  
  
 templateConfig.setXml(null);  
 mpg.setTemplate(templateConfig); // 配置模板  
  
  
 // 配置自定义输出模板  
 //指定自定义模板路径，注意不要带上.ftl/.vm, 会根据使用的模板引擎自动识别  
 // templateConfig.setEntity("templates/entity2.java");  
 // templateConfig.setService();  
 // templateConfig.setController();  
  
 templateConfig.setXml(null);  
 mpg.setTemplate(templateConfig);  
  
 // 策略配置  
 StrategyConfig strategy = new StrategyConfig();  
 strategy.setNaming(NamingStrategy.*underline\_to\_camel*);  
 strategy.setColumnNaming(NamingStrategy.*underline\_to\_camel*);  
 //strategy.setSuperEntityClass("你自己的父类实体,没有就不用设置!");  
 strategy.setEntityLombokModel(true);  
 strategy.setRestControllerStyle(true);  
 // 公共父类  
 // strategy.setSuperControllerClass("你自己的父类控制器,没有就不用设置!");  
 // 写于父类中的公共字段  
 strategy.setSuperEntityColumns("id");  
 strategy.setInclude(*scanner*("表名，多个英文逗号分割").split(","));  
 strategy.setControllerMappingHyphenStyle(true);  
 strategy.setTablePrefix(pc.getModuleName() + "\_");  
 mpg.setStrategy(strategy);  
 mpg.setTemplateEngine(new VelocityTemplateEngine());  
 mpg.execute();  
}