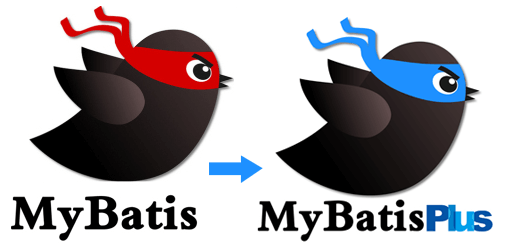
# 01【熟悉】简介

## 1，说明

Mybatis-Plus（简称MP）是一个 Mybatis 的增强工具，在 Mybatis 的基础上只做增强不做改变，为简化开发、提高效率而生。

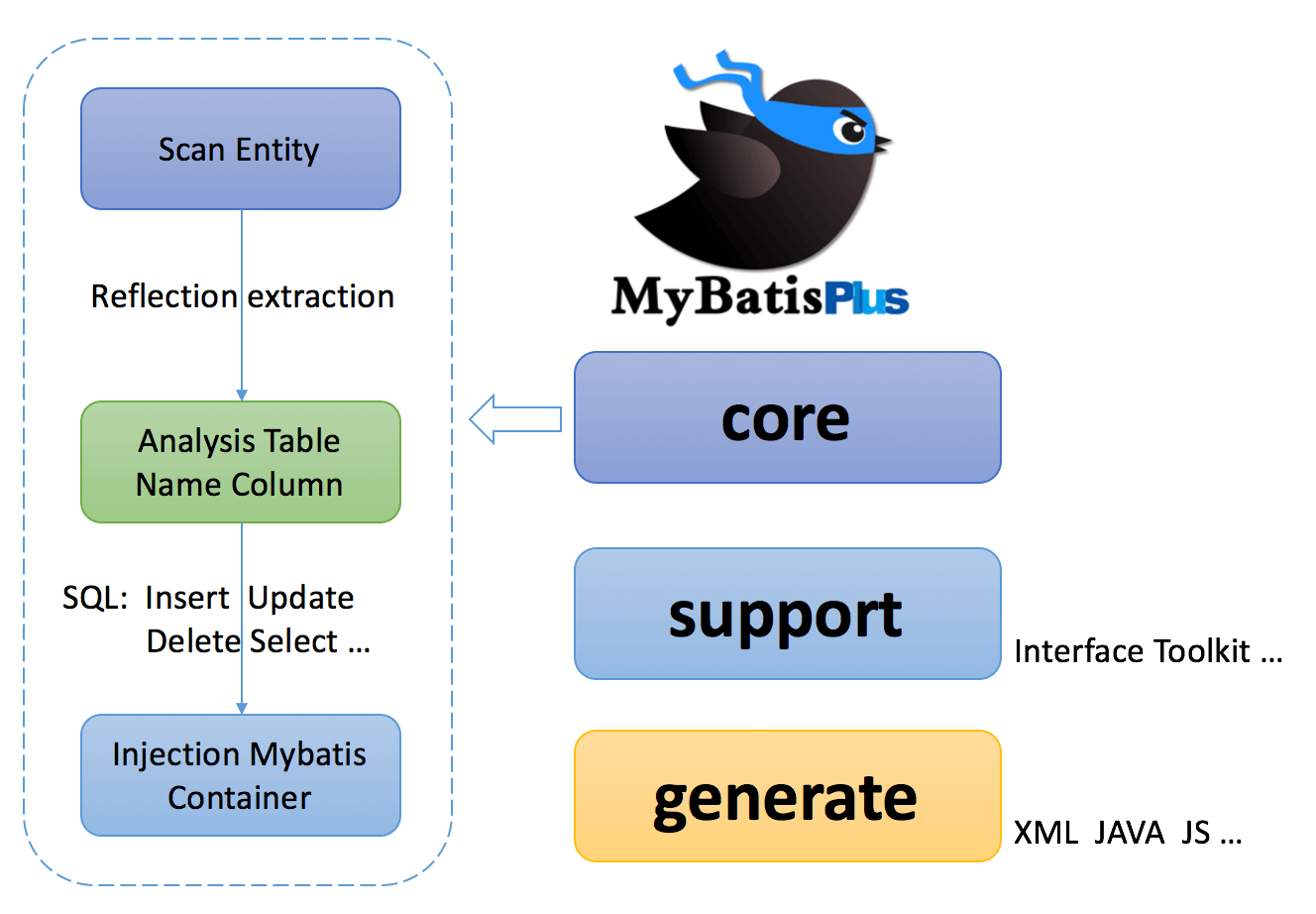
官方文档说的愿景是成为 Mybatis 最好的搭档，就像 魂斗罗 中的1P、2P，基友搭配，效率翻倍。



## 2，特性

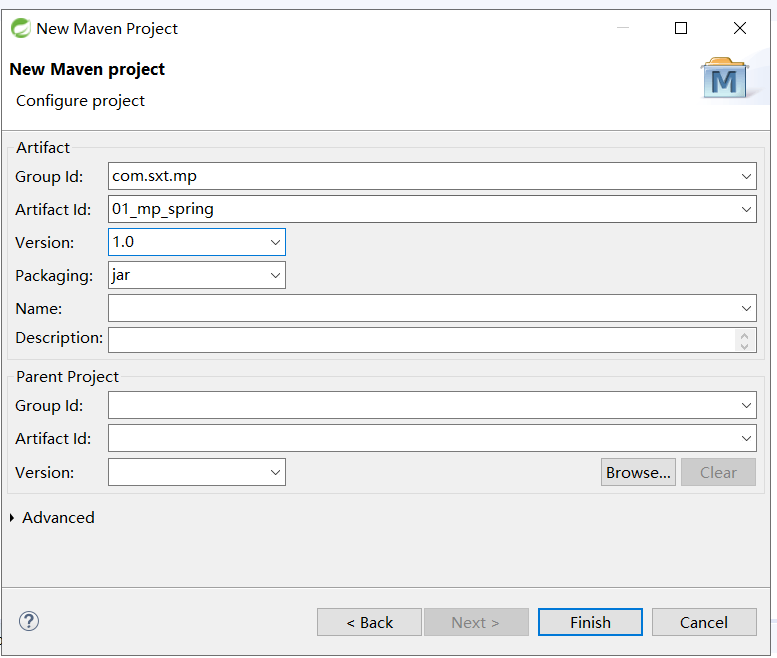
1. 无侵入：Mybatis-Plus 在 Mybatis 的基础上进行扩展，只做增强不做改变，引入 Mybatis-Plus 不会对您现有的 Mybatis 构架产生任何影响，而且 MP 支持所有
2. Mybatis 原生的特性
3. 依赖少：仅仅依赖 Mybatis 以及 Mybatis-Spring
4. 损耗小：启动即会自动注入基本 CURD，性能基本无损耗，直接面向对象操作
5. 预防Sql注入：内置 Sql 注入剥离器，有效预防Sql注入攻击
6. 通用CRUD操作：内置通用 Mapper、通用 Service，仅仅通过少量配置即可实现单表大部分 CRUD 操作，更有强大的条件构造器，满足各类使用需求
7. 多种主键策略：支持多达4种主键策略（内含分布式唯一ID生成器），可自由配置，完美解决主键问题
8. 支持热加载：Mapper 对应的 XML 支持热加载，对于简单的 CRUD 操作，甚至可以无 XML 启动
9. 支持ActiveRecord：支持 ActiveRecord 形式调用，实体类只需继承 Model 类即可实现基本 CRUD 操作
10. 支持代码生成：采用代码或者 Maven 插件可快速生成 Mapper 、 Model 、 Service 、 Controller 层代码，支持模板引擎，更有超多自定义配置等您来使用（P.S. 比 Mybatis 官方的 Generator 更加强大！）
11. 支持自定义全局通用操作：支持全局通用方法注入（ Write once, use anywhere ）
12. 支持关键词自动转义：支持数据库关键词（order、key......）自动转义，还可自定义关键词
13. 内置分页插件：基于 Mybatis 物理分页，开发者无需关心具体操作，配置好插件之后，写分页等同于普通List查询
14. 内置性能分析插件：可输出 Sql 语句以及其执行时间，建议开发测试时启用该功能，能有效解决慢查询
15. 内置全局拦截插件：提供全表 delete 、 update 操作智能分析阻断，预防误操作

## 3，架构原理

​

# 02【掌握】MP和spring的集成配置

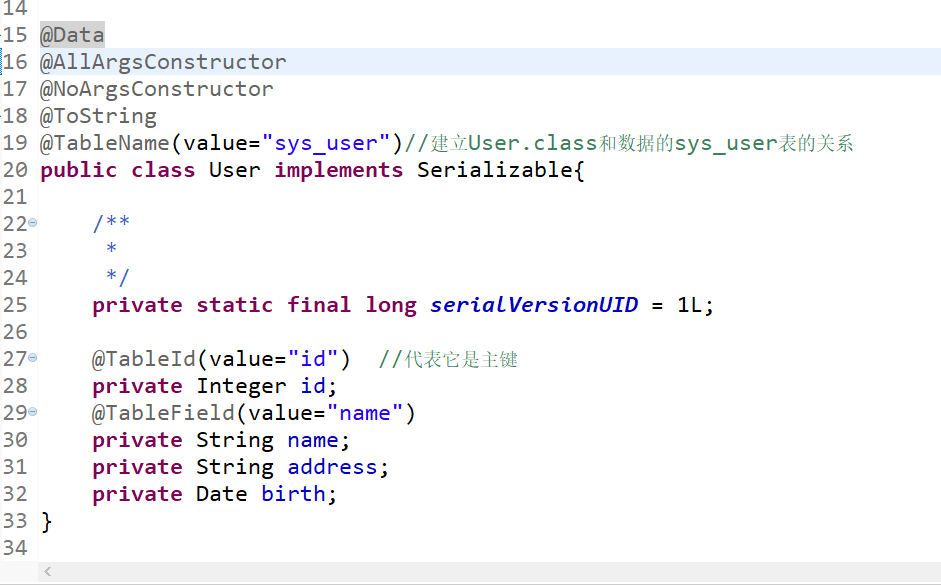
## 创建项目



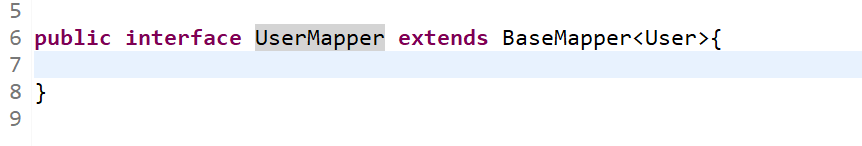
## 加入依赖

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.sxt.mp</groupId>  <artifactId>01\_mp\_spring</artifactId>  <version>1.0</version>  <properties>  <mybatisplus.version>3.2.0</mybatisplus.version>  <spring.version>4.3.24.RELEASE</spring.version>  <mysql.version>8.0.17</mysql.version>  <!-- 注意只能使用2.0以下的版本 -->  <log4j.version>1.2.17</log4j.version>  <lombok.version>1.18.8</lombok.version>  </properties>  <dependencies>  <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>${lombok.version}</version>  <scope>provided</scope>  </dependency>  <!-- 引入spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-expression</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-orm</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>${spring.version}</version>  </dependency>  <!-- mysql数据库驱动 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>${mysql.version}</version>  </dependency>  <!-- log4j -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>${log4j.version}</version>  </dependency>  <!-- 引入mp -->  <dependency>  <groupId>com.baomidou</groupId>  <artifactId>mybatis-plus</artifactId>  <version>${mybatisplus.version}</version>  </dependency>  </dependencies>  </project> |

## 创建User



## 创建UserMapper



## 创建UserMapper.xml[暂时可以不用]

|  |
| --- |
| <?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.sxt.mapper.UserMapper"* >    </mapper> |

## 创建db.properties

|  |
| --- |
| driver=com.mysql.cj.jdbc.Driver  url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf8&useSSL=true&serverTimezone=UTC  user=root  password=123456 |

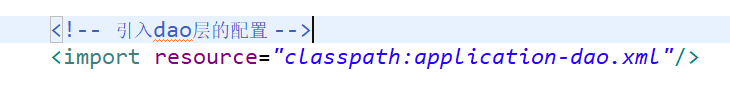
## 创建log4j.properties

|  |
| --- |
| # Global logging configuration  log4j.rootLogger=DEBUG, stdout  # MyBatis logging configuration...  log4j.logger.org.mybatis.example.BlogMapper=TRACE  # Console output...  log4j.appender.stdout=org.apache.log4j.ConsoleAppender  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n |

## 创建appliaction-dao.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd"*>  <!-- dao层的配置：核心是要产生 Mapper代理类对象 1.引入数据库配置信息 2.数据源配置 3.SqlSessionFactory  4.产生Mapper接口的代理类对象 -->  <!-- 1.引入数据库配置信息 -->  <context:property-placeholder  location=*"classpath:db.properties"* system-properties-mode=*"FALLBACK"* />  <!--2.数据源配置 -->  <bean id=*"dataSource"*  class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>  <property name=*"driverClassName"* value=*"${driver}"*></property>  <property name=*"url"* value=*"${url}"*></property>  <property name=*"username"* value=*"${user}"*></property>  <property name=*"password"* value=*"${password}"*></property>  </bean>  <!-- 3.SqlSessionFactory -->  <bean id=*"sqlSessionFactory"*  class=*"com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"*></property>    <property name=*"globalConfig"* ref=*"globalConfig"*></property>  <!-- 加载xxMapper.xml -->  <property name=*"mapperLocations"*>  <array>  <value>classpath:mapper/\*Mapper.xml</value>  </array>  </property>  <!-- 配置分页插件 -->  <property name=*"plugins"*>  <array>  <bean class=*"com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor"*>  </bean>  </array>  </property>  </bean>  <!-- 声明全局配置 -->  <bean id=*"globalConfig"* class=*"com.baomidou.mybatisplus.core.config.GlobalConfig"*>  <!-- 指定主键自动增长类型 -->  <property name=*"dbConfig"* ref=*"dbConfig"*></property>  </bean>  <bean id=*"dbConfig"* class=*"com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig"*>  <property name=*"idType"* value=*"AUTO"*></property>  </bean>    <!-- 4.产生Mapper接口的代理类对象 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <!-- 需要生成代理类对象的mapper接口包 -->  <property name=*"basePackage"*  value=*"com.sxt.mapper"*></property>  <!-- sqlSessionFactory 的name 用于为代理类中生成SqlSession -->  <property name=*"sqlSessionFactoryBeanName"*  value=*"sqlSessionFactory"*></property>  </bean>  </beans> |

## 创建appliactionContext.xml

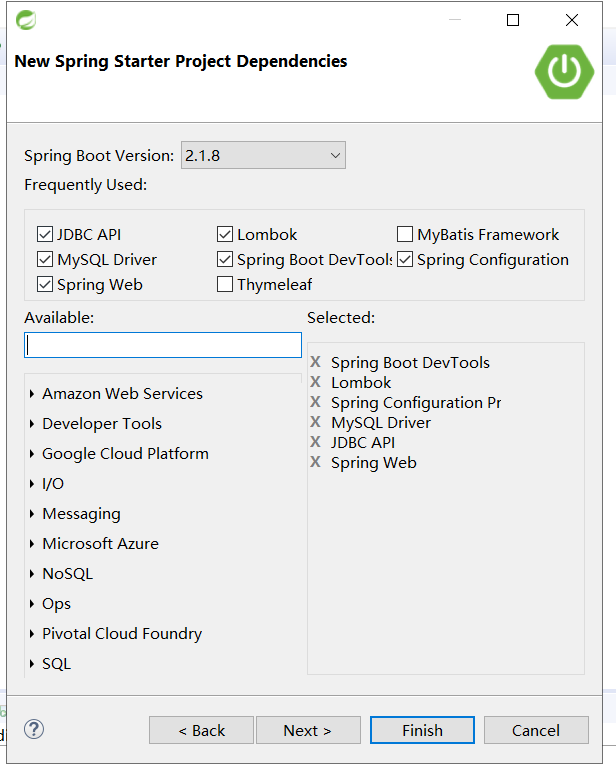


## 测试

|  |
| --- |
| **public** **class** TestApp {      **public** **static** **void** main(String[] args) {  ApplicationContext context=**new** ClassPathXmlApplicationContext("classpath:applicationContext.xml");  UserMapper userMapper=context.getBean(UserMapper.**class**);  // userMapper.insert(new User(111, "小明明", "浦东", new Date()));  // updateUser(userMapper);  // deleteUser(userMapper);  // query1(userMapper);  // query2(userMapper);  // query3(userMapper);  // query4(userMapper,"小明");  *query5*(userMapper);  System.***out***.println("操作成功");    }    **private** **static** **void** query5(UserMapper userMapper) {  IPage<User> page=**new** Page<>(1, 5);  userMapper.selectPage(page, **null**);  **long** total = page.getTotal();  System.***out***.println("总条数:"+total);  List<User> list = page.getRecords();  *print*(list);    }  **private** **static** **void** query4(UserMapper userMapper,String name) {  // Integer count = userMapper.selectCount(null);  QueryWrapper<User> queryWrapper=**new** QueryWrapper<>();  queryWrapper.like(name!=**null**, "name", name);  Integer selectCount = userMapper.selectCount(queryWrapper);  System.***out***.println(selectCount);  }  /\*\*  \* 查询  \* **@param** userMapper  \*/  **private** **static** **void** query3(UserMapper userMapper) {  Map<String, Object> columnMap=**new** HashMap<>();  columnMap.put("name", "大明明");  columnMap.put("address", "武汉");  List<User> list = userMapper.selectByMap(columnMap);  *print*(list);  }    /\*\*  \* 查询  \* **@param** userMapper  \*/  **private** **static** **void** query2(UserMapper userMapper) {  Collection<Serializable> idList=**new** ArrayList<Serializable>();  idList.add(7);  idList.add(8);  idList.add(9);  List<User> list = userMapper.selectBatchIds(idList);  *print*(list);  }  /\*\*  \* 查询  \* **@param** userMapper  \*/  **private** **static** **void** query1(UserMapper userMapper) {  User user = userMapper.selectById(7);  System.***out***.println(user);  }  /\*\*  \* 删除用户  \* **@param** userMapper  \*/  **private** **static** **void** deleteUser(UserMapper userMapper) {  // userMapper.deleteById(111);//根据主键删除  // Collection<Serializable> idList=new ArrayList<Serializable>();  // idList.add(22);  // idList.add(112);  // userMapper.deleteBatchIds(idList);  // Map<String, Object> columnMap=new HashMap<String, Object>();  // columnMap.put("id", 6);  // columnMap.put("name", "小明");  // userMapper.deleteByMap(columnMap);  QueryWrapper<User> wrapper=**new** QueryWrapper<>();  wrapper.between("id", 20, 21);  userMapper.delete(wrapper);    }  /\*\*  \* 修改  \* **@param** userMapper  \*/  **private** **static** **void** updateUser(UserMapper userMapper) {  // userMapper.updateById(new User(112, "大明明", "武汉", new Date()));//根据主键修改  UpdateWrapper<User> updateWrapper=**new** UpdateWrapper<>();  // updateWrapper.eq("name", "小明"); //sql ===where name="小明"  updateWrapper.between("id", 10, 20);  userMapper.update(**new** User(112, "小明", "武汉", **new** Date()), updateWrapper);  }    **private** **static** **void** print(List<User> list) {  **for** (User user : list) {  System.***out***.println(user);  }  }  } |

# 03【掌握】MP和springboot集成

## 创建项目



## 添加依赖

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.8.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <groupId>com.sxt</groupId>  <artifactId>02\_mp\_springboot</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>02\_mp\_springboot</name>  <description>springboot集成mp</description>  <properties>  <java.version>1.8</java.version>  <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>  <mybatisplus.version>3.2.0</mybatisplus.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-configuration-processor</artifactId>  <optional>true</optional>  </dependency>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <optional>true</optional>  </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>${mybatisplus.version}</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

## 创建分页配置类

|  |
| --- |
| /\*\*  \* mybatisplus的配置类  \* **@author** LJH  \*  \*/  @Configuration  @ConditionalOnClass(value= {PaginationInterceptor.**class**})  **public** **class** MybatisPlusConfig {    @Bean  **public** PaginationInterceptor paginationInterceptor() {  **return** **new** PaginationInterceptor();  }  } |

## 配置yml

|  |
| --- |
| #数据源  spring:  datasource:  driver-class-name: com.mysql.cj.jdbc.Driver  url: jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf8&useSSL=true&serverTimezone=UTC  username: root  password: 123456  type: org.springframework.jdbc.datasource.DriverManagerDataSource  #配置mybatisplus  mybatis-plus:  mapper-locations:  - classpath:mapper/\*Mapper.xml  global-config:  db-config:  id-type: auto  banner: **true** |

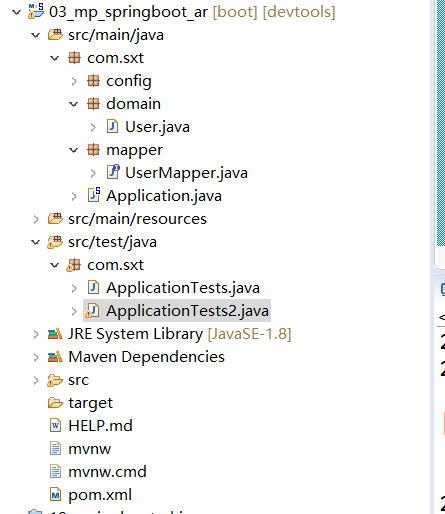
## 复制上个项目User UserMapper UserMapper.xml

## 测试

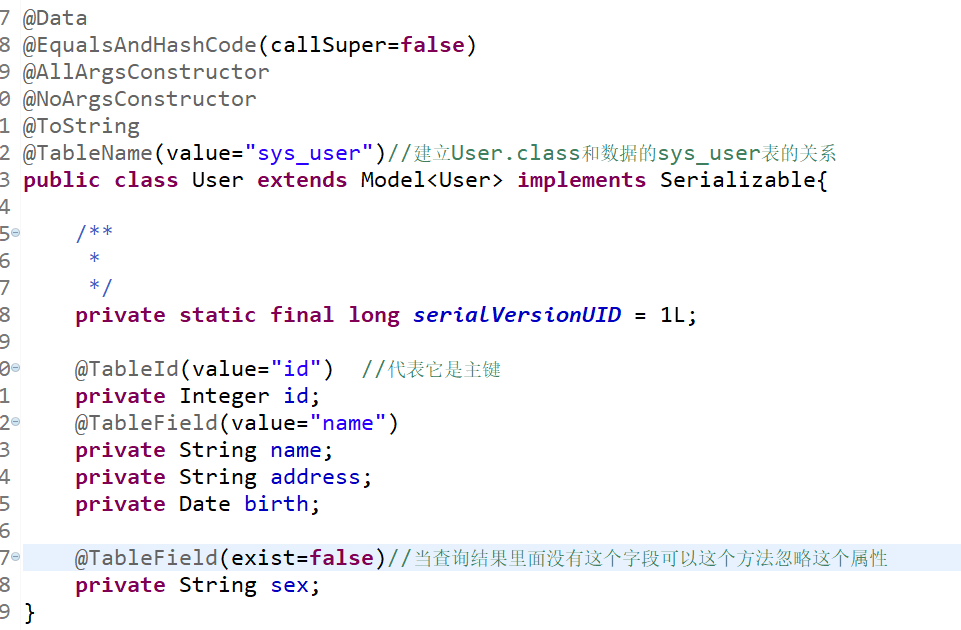


# 04【熟悉】MP-AR模式开发

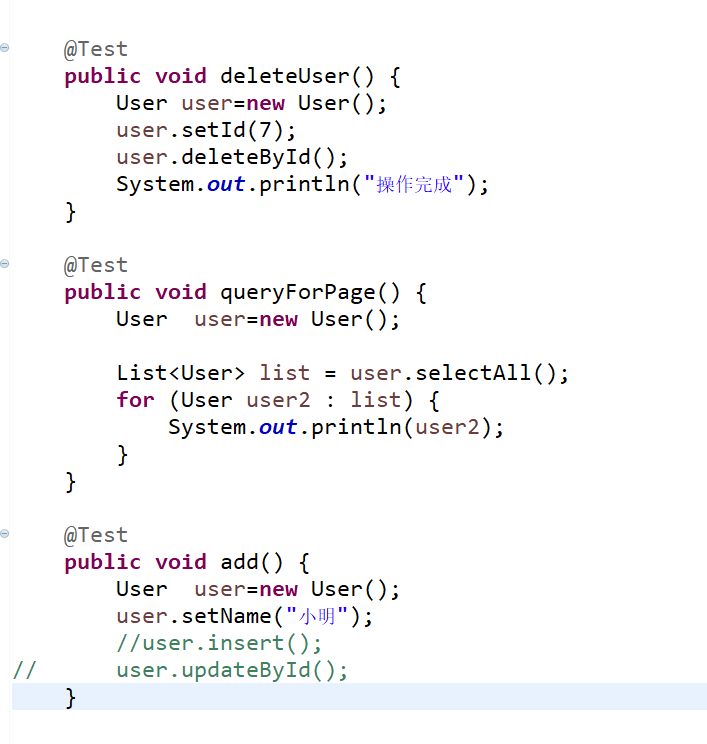
## 创建项目



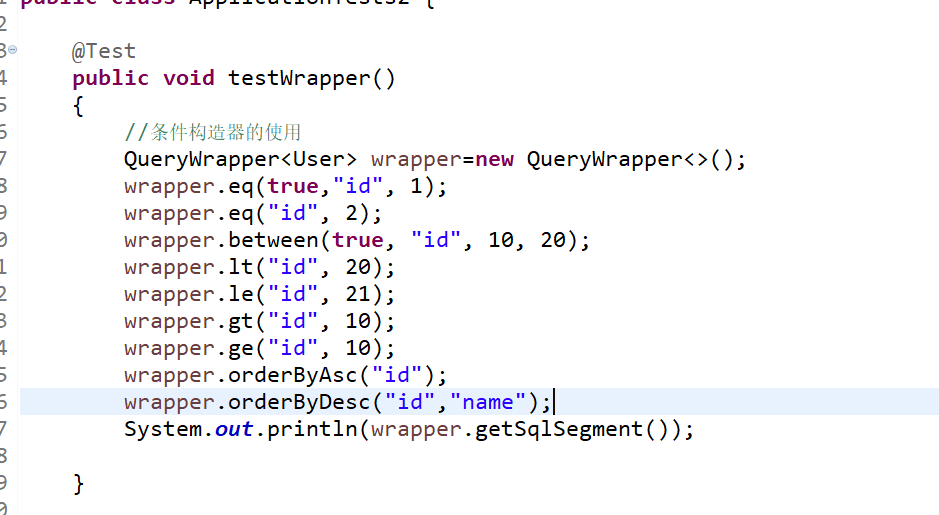
## 修改User



## 测试

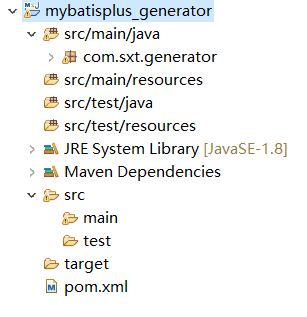


# 05【掌握】条件构造器



# 06【掌握】代码生成器

## 创建项目



## 加入依赖

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.sxt</groupId>  <artifactId>mybatisplus\_generator</artifactId>  <version>1.0</version>    <properties>  <mysql.version>8.0.17</mysql.version>  </properties>  <dependencies>  <dependency>  <groupId>com.baomidou</groupId>  <artifactId>mybatis-plus-generator</artifactId>  <version>3.2.0</version>  </dependency>  <dependency>  <groupId>org.apache.velocity</groupId>  <artifactId>velocity-engine-core</artifactId>  <version>2.1</version>  </dependency>  <dependency>  <groupId>org.freemarker</groupId>  <artifactId>freemarker</artifactId>  <version>2.3.29</version>  </dependency>  <dependency>  <groupId>com.ibeetl</groupId>  <artifactId>beetl</artifactId>  <version>3.0.11.RELEASE</version>  </dependency>  <!-- mysql数据库驱动 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>${mysql.version}</version>  </dependency>    </dependencies>  </project> |

## 创建生成类

|  |
| --- |
| /\*\*  \* mp生成器  \* **@author** LJH  \*  \*/  **public** **class** CodeGenerator {    /\*\*  \* <p>  \* 读取控制台内容  \* </p>  \*/  **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.*isNotEmpty*(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();  String projectPath = System.*getProperty*("user.dir");  gc.setOutputDir(projectPath + "/src/main/java");  // gc.setOutputDir("D:\\workspace-sts\\0520adv\\02\_mp\_springboot/src/main/java");  gc.setAuthor("老雷");  gc.setOpen(**false**);//当代码生成完成之后是否打开代码所在的文件夹  // gc.setSwagger2(true); 实体属性 Swagger2 注解  // gc.setServiceName("%sService");  mpg.setGlobalConfig(gc);  // 数据源配置  DataSourceConfig dsc = **new** DataSourceConfig();  dsc.setUrl("jdbc:mysql://localhost:3306/test?useUnicode=true&useSSL=false&characterEncoding=utf8&serverTimezone=UTC");  // dsc.setSchemaName("public");  dsc.setDriverName("com.mysql.jdbc.Driver");  dsc.setUsername("root");  dsc.setPassword("123456");  mpg.setDataSource(dsc);  // 包配置  PackageConfig pc = **new** PackageConfig();  // pc.setModuleName(scanner("模块名"));  pc.setModuleName("sys");  pc.setParent("com.sxt");//controller entity service service.impl  pc.setController("controller");  pc.setEntity("domain");  pc.setMapper("mapper");  pc.setService("service");  pc.setServiceImpl("service.impl");  pc.setXml("mapper.xml");  mpg.setPackageInfo(pc);  // 策略配置  StrategyConfig strategy = **new** StrategyConfig();  //设置字段和表名的是否把下划线完成驼峰命名规则  strategy.setNaming(NamingStrategy.***underline\_to\_camel***);  strategy.setColumnNaming(NamingStrategy.***underline\_to\_camel***);  //设置生成的实体类继承的父类  // strategy.setSuperEntityClass("com.sxt.BaseEntity");  strategy.setEntityLombokModel(**true**);//是否启动lombok  strategy.setRestControllerStyle(**true**);//是否生成resetController  // 公共父类  // strategy.setSuperControllerClass("com.sxt.BaseController");  // 写于父类中的公共字段  // strategy.setSuperEntityColumns("person\_id","person\_name");  //要设置生成哪些表 如果不设置就是生成所有的表  // strategy.setInclude(scanner("表名，多个英文逗号分割").split(","));  strategy.setControllerMappingHyphenStyle(**true**);  // strategy.setTablePrefix(pc.getModuleName() + "\_");  strategy.setTablePrefix("sys\_");  mpg.setStrategy(strategy);  mpg.execute();  }  } |