**Mybatisplus第二天**

上海千锋 文威

* ActiveRecord
* Mybatis-Plus的插件
* 代码生成器

## 1.1 ActiveRecord

ActiveRecord（简称AR）一直广受动态语言（ PHP 、 Ruby 等）的喜爱，而Java作为准静态语言，对于 ActiveRecord 往往只能感叹其优雅，所以我们也在AR道路上进行了一定的探索，喜欢大家能够喜欢。

什么是ActiveRecord?

ActiveRecord也属于ORM（对象关系映射）层，由Rails最早提出，遵循标准的ORM模型：表映射到记录，记 录映射到对象，字段映射到对象属性。配合遵循的命名和配置惯例，能够很大程度的快速实现模型的操作，而且简洁易懂。

ActiveRecord的主要思想是：

* 每一个数据库表对应创建一个类，类的每一个对象实例对应于数据库中表的一行记录；

通常表的每个字段在类中都有相应的Field；

* **ActiveRecord同时负责把自己持久化，在ActiveRecord中封装了对数据库的访问，即CURD;**
* ActiveRecord是一种领域模型(Domain Model)，封装了部分业务逻辑；

### 1.1.1 开启AR之旅

我们搭建mybatisplus整合spring的工程，和昨天的一样即可。

在MP中，开启AR非常简单，只需要将实体对象继承Model即可。

* 创建实体类继承Model

|  |
| --- |
| @Data @NoArgsConstructor @AllArgsConstructor**public class** User extends Model<User> {**private** Long **id**;  **private** String **userName**;**private** String **password**;  **private** String **name**;  **private** Integer **age**;**private** String **email**; } |

#### 1.1.1.1 查询

* 查询所有

|  |
| --- |
| @Test **public void** testFindAll(){  User user = **new** User();  *//查询所有* List<User> userList = user.selectAll();  **for**(User u : userList){  System.***out***.println(u);  } } |

* 根据主键查询

|  |
| --- |
| @Test **public void** testFindById(){  User user = **new** User();  user.setId(1l);  *//根据主键查询* User user1 = user.selectById();  System.***out***.println(user1); } |

#### 1.1.1.2 新增数据

|  |
| --- |
| @Test **public void** testInsertUser(){  User user = **new** User();  user.setName(**"邢道荣"**);  user.setAge(30);  user.setPassword(**"123456"**);  user.setUserName(**"xingdaorong"**);  user.setEmail(**"xingdaorong@qq.com"**);  **boolean** insert = user.insert();  System.***out***.println(insert); } |

#### 1.1.1.3 更新操作

|  |
| --- |
| @Test **public void** testUpdateUser(){  User user = **new** User();  user.setId(14L);  user.setAge(39);  **boolean** update = user.updateById();  System.***out***.println(update); } |

#### 1.1.1.4 删除操作

|  |
| --- |
| @Test **public void** testDeleteUser(){  User user = **new** User();  user.setId(7L);  **boolean** delete = user.deleteById();  System.***out***.println(delete); } |

#### 1.1.1.5 根据条件查询

|  |
| --- |
| @Test **public void** testAR() {  User user = **new** User();  QueryWrapper<User> userQueryWrapper = **new** QueryWrapper<>();  userQueryWrapper.le(**"age"**,**"20"**);  List<User> users = user.selectList(userQueryWrapper);  **for** (User user1 : users) {  System.***out***.println(user1);  } } |

## 1.2 Mybatisplus插件

### 1.2.1 mybatisplus性能插件

性能分析拦截器，用于输出每条 SQL 语句及其执行时间，可以设置最大执行时间，超过时间会抛出异常。

**该插件只用于开发环境，不建议生产环境使用。**

配置：

* 定义一个配置文件mybatis-config.xml

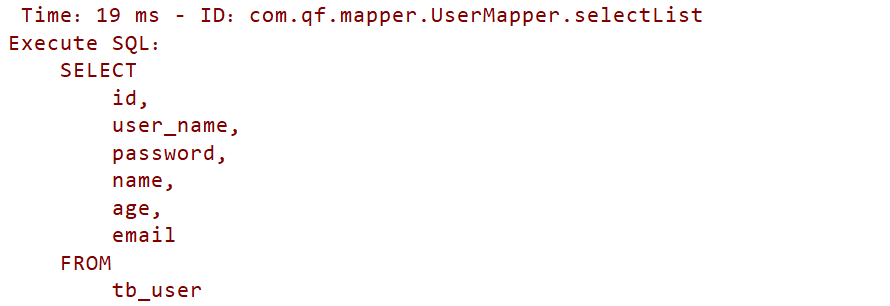
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"** *?>* **<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>*** <**configuration**>  <**plugins**>  *<!-- SQL 执行性能分析，开发环境使用，线上不推荐。 maxTime 指的是 sql 最大执行时长 -->* <**plugin interceptor="com.baomidou.mybatisplus.extension.plugins.PerformanceInterceptor"**>  <**property name="maxTime" value="100"** />  *<!--SQL是否格式化 默认false-->* <**property name="format" value="true"** />  </**plugin**>  </**plugins**> </**configuration**> |

* 在applicationContext.xml中引入配置文件

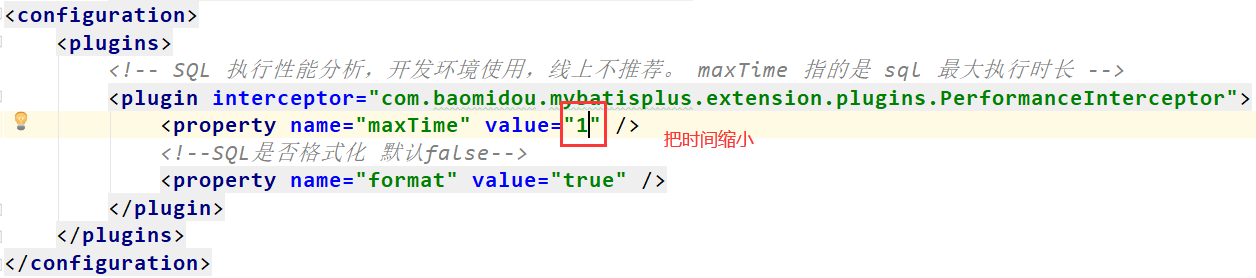
|  |
| --- |
| <**bean id="sqlSessionFactory"  class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean"**>  <**property name="dataSource" ref="dataSource"**/>  <**property name="configLocation" value="classpath:mybatis-config.xml"**></**property**>  <**property name="globalConfig"**>  <**bean class="com.baomidou.mybatisplus.core.config.GlobalConfig"**>  <**property name="dbConfig"**>  <**bean class="com.baomidou.mybatisplus.core.config.GlobalConfig$DbConfig"**>  <**property name="idType" value="AUTO"**></**property**>  <**property name="tablePrefix" value="tb\_"**/>  </**bean**>  </**property**>  </**bean**>  </**property**> </**bean**> |

* 执行一个查询的方法

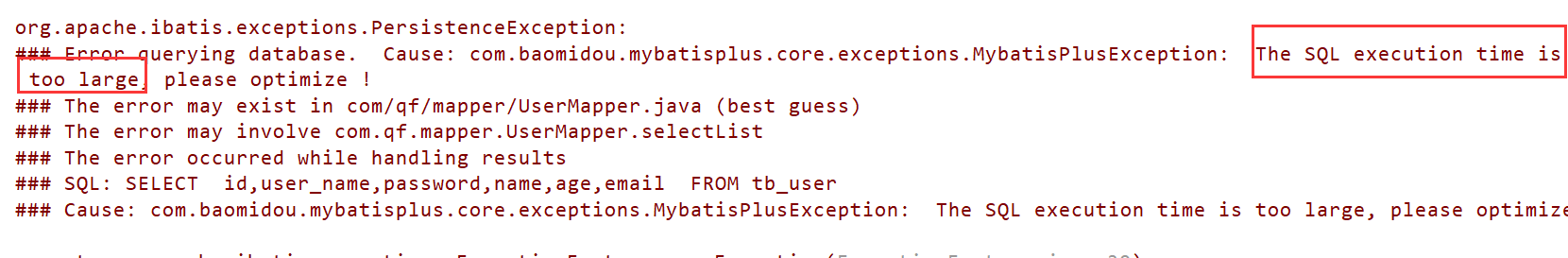
我们可以看到控制台sql语句执行情况



现在我们缩短sql执行时间



再次查看控制台



### 1.2.2 乐观锁插件

意图： 当要更新一条记录的时候，希望这条记录没有被别人更新

乐观锁实现方式：

* 取出记录时，获取当前version
* 更新时，带上这个version
* 执行更新时，set version = newVersion where version = oldVersion
* 如果version不对，就更新失败

第一步:插件配置:

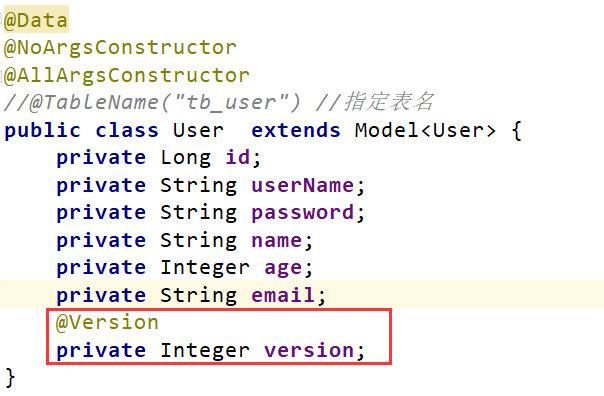
|  |
| --- |
| <**plugins**>  *<!-- SQL 执行性能分析，开发环境使用，线上不推荐。 maxTime 指的是 sql 最大执行时长 -->* <**plugin interceptor="com.baomidou.mybatisplus.extension.plugins.PerformanceInterceptor"**>  <**property name="maxTime" value="100"** />  *<!--SQL是否格式化 默认false-->* <**property name="format" value="true"** />  </**plugin**>  **<plugin interceptor="com.baomidou.mybatisplus.extension.plugins.OptimisticLockerInterceptor"></plugin>** </**plugins**> |

第二步:为表添加version字段，并且设置初始值为1

|  |
| --- |
| ALTER TABLE `tb\_user` ADD COLUMN `version` INT(10) NULL AFTER `email`;  UPDATE `tb\_user` SET `version`='1'; |



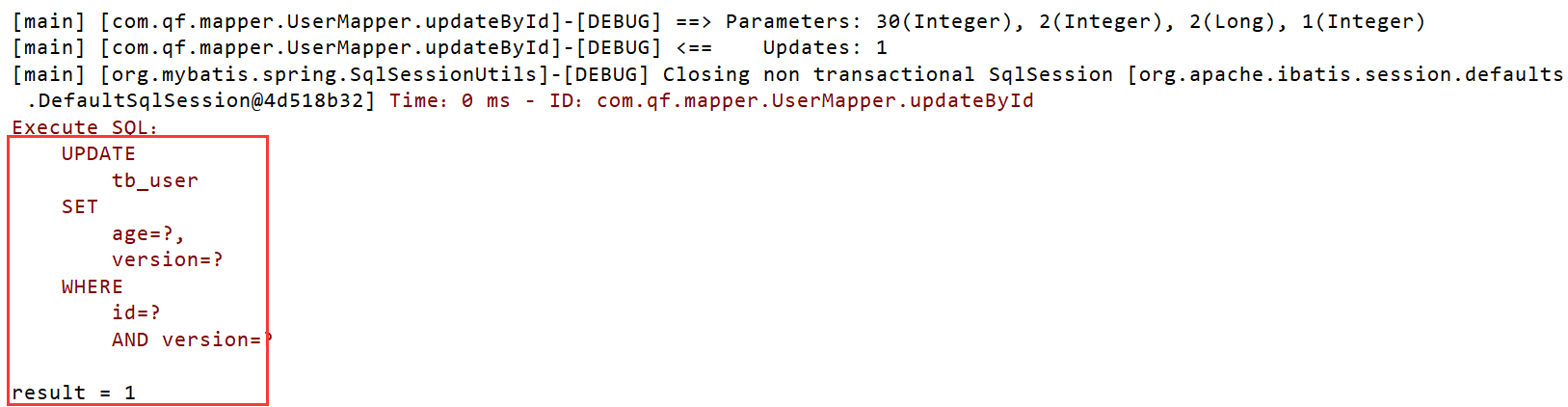
第三步: 为User实体对象添加version字段，并且添加@Version注解



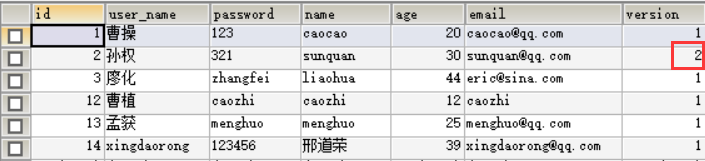
第四步:测试

|  |
| --- |
| @Autowired UserMapper **userMapper**;  @Test **public void** testUpdate(){  User user = **new** User();  user.setAge(30);  user.setId(2L);  user.setVersion(1); *//获取到version为1* **int** result = **this**.**userMapper**.updateById(user);  System.***out***.println(**"result = "** + result); } |

观察控制台输出信息



查看数据库:



我们发现版本号发生了改变。

如果再次执行，更新则不成功。这样就避免了多人同时更新时导致数据的不一致。

我们把第四步的测试用例重新执行一遍，我们会发现数据没有发生任何变化。

## 1.3 代码生成器

* 创建maven工程,导入依赖

|  |
| --- |
| *<?***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 http://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**>2.1.4.RELEASE</**version**>  </**parent**>   <**groupId**>com.qf</**groupId**>  <**artifactId**>mybatisplus-generator</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-test</**artifactId**>  <**scope**>test</**scope**>  </**dependency**>   *<!--mybatis-plus的springboot支持-->* <**dependency**>  <**groupId**>com.baomidou</**groupId**>  <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  <**version**>3.1.1</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.baomidou</**groupId**>  <**artifactId**>mybatis-plus-generator</**artifactId**>  <**version**>3.1.1</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-freemarker</**artifactId**>  </**dependency**>  *<!--mysql驱动-->* <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  <**version**>5.1.47</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  </**dependency**>   </**dependencies**>   <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-maven-plugin</**artifactId**>  </**plugin**>  </**plugins**>  </**build**>  </**project**> |

* 代码

|  |
| --- |
| **package** com.qf.generator;  **import** com.baomidou.mybatisplus.core.exceptions.MybatisPlusException; **import** com.baomidou.mybatisplus.core.toolkit.StringPool; **import** com.baomidou.mybatisplus.core.toolkit.StringUtils; **import** com.baomidou.mybatisplus.generator.AutoGenerator; **import** com.baomidou.mybatisplus.generator.InjectionConfig; **import** com.baomidou.mybatisplus.generator.config.\*; **import** com.baomidou.mybatisplus.generator.config.po.TableInfo; **import** com.baomidou.mybatisplus.generator.config.rules.NamingStrategy; **import** com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;  **import** java.util.ArrayList; **import** java.util.List; **import** java.util.Scanner;  **public class** MysqlGenerator {  */\*\*  \* <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 + **"！"**);  }  */\*\*  \* RUN THIS  \*/* **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.setAuthor(**"qf"**);  gc.setOpen(**false**);  mpg.setGlobalConfig(gc);  *// 数据源配置* DataSourceConfig dsc = **new** DataSourceConfig();  dsc.setUrl(**"jdbc:mysql://192.168.10.137:3306/mp?useUnicode=true&useSSL=false&characterEncoding=utf8"**);  *// dsc.setSchemaName("public");* dsc.setDriverName(**"com.mysql.jdbc.Driver"**);  dsc.setUsername(**"root"**);  dsc.setPassword(**"Admin123!"**);  mpg.setDataSource(dsc);  *// 包配置* PackageConfig pc = **new** PackageConfig();  pc.setModuleName(*scanner*(**"模块名"**));  pc.setParent(**"com.qf.mp.generator"**);  mpg.setPackageInfo(pc);  *// 自定义配置* InjectionConfig cfg = **new** InjectionConfig() {  @Override  **public void** initMap() {  *// to do nothing* }  };  List<FileOutConfig> focList = **new** ArrayList<>();  focList.add(**new** FileOutConfig(**"/templates/mapper.xml.ftl"**) {  @Override  **public** String outputFile(TableInfo tableInfo) {  *// 自定义输入文件名称* **return** projectPath + **"/qf/src/main/resources/mapper/"** + pc.getModuleName()  + **"/"** + tableInfo.getEntityName() + **"Mapper"** +  StringPool.***DOT\_XML***;  }  });  cfg.setFileOutConfigList(focList);  mpg.setCfg(cfg);  mpg.setTemplate(**new** TemplateConfig().setXml(**null**));  *// 策略配置* StrategyConfig strategy = **new** StrategyConfig();  strategy.setNaming(NamingStrategy.***underline\_to\_camel***);  strategy.setColumnNaming(NamingStrategy.***underline\_to\_camel***);  *//* strategy.setSuperEntityClass(**"com.baomidou.mybatisplus.samples.generator.common.BaseEntity"**);  strategy.setEntityLombokModel(**true**);  *//* strategy.setSuperControllerClass(**"com.baomidou.mybatisplus.samples.generator.common.BaseController"**);  strategy.setInclude(*scanner*(**"表名"**));  strategy.setSuperEntityColumns(**"id"**);  strategy.setControllerMappingHyphenStyle(**true**);  strategy.setTablePrefix(pc.getModuleName() + **"\_"**);  mpg.setStrategy(strategy);  *// 选择 freemarker 引擎需要指定如下加，注意 pom 依赖必须有！* mpg.setTemplateEngine(**new** FreemarkerTemplateEngine());  mpg.execute();}  } |

* 运行main方法，在控制台输入模块名称和表名称



* 运行之后查看工程目录

