01_SpringBoot#DAO#TK-Mybatis#对比介绍

> 【TK-Mybatis#对比介绍】

时间：2023年04月24日 20:13:48

推荐阅读：

- https://gitee.com/free/Mapper/wikis/Home
- https://blog.csdn.net/zhong_jing/article/details/124962765

一、差异

💊1.1 名称不同、开发者不同

- tk-mybatis
  - *通用mapper*
  - *TK-Mybatis 的开发者是一位名叫常德鹏的中国开发者。他在 2013 年开源了 TK-Mybatis，当时它的名称是 Mybatis-Helper。后来，为了更好地表达其作为通用 Mapper 框架的定位，常德鹏将其改名为 TK-Mybatis（TK 是Toolkit的缩写，表示工具包）。这是一款基于 Mybatis 的通用 Mapper 框架，旨在简化 Mybatis 的开发流程，提高开发效率。*
- mybatis-plus
  - *mybatis+*
  - *MybatisPlus 的开发者是一位名叫杨晓峰的中国开发者。他在 2016 年开源了 MybatisPlus，这是一款基于 Mybatis 的快速开发框架，旨在简化 Mybatis 的开发流程，提高开发效率。*

💊1.2 MapperScan注解（定位DAO）

- `tk-mybatis` 是自己实现的 `@MapperScan`
- `mybatis-plus` 是使用mybatis原生的 `@MapperScan`

💊1.3 实体注解（表、主键、列）

- `tk-mybatis` 使用的是JPA（`Java Persistence API`）注解
  - `@Table`
  - `@Id`
  - `@Column`
- `mybatis-plus` 使用的是自己定义的一套实体注解
  - `@TableName`
  - `@TableId`
  - `@ColumnName`

💊1.4 三层模型架构中位置不同

- `tk-mybatis` 专注于DAL层
- `mybatis-plus` 提供了DAL层接口与实现，同时提供了封装完善的Service层接口与实现

二、`tk-mybatis` 和 `mybatis-plus` 可以同时使用吗?

> 可以，但需要注意以下几点：

2.1 注意版本兼容性⚡

- tk-mybatis和mybatis-plus不是由一个作者开发，不同的版本中会依赖不同的mybatis版本
  - mybatis大版本迭代会出现明显的兼容性问题（请确保mybatis版本一致或使用各自的最新版本）

2.2 企业中内部封装的BaseService⚡

- 部分企业使用 `tk-mybatis` ，没有封装BaseService层或封装的BaseService层不完全；
  - 他们会拓展 `tk-mybatis` 的Mapper以支持批量新增与批量更新等操作。
- 部分企业使用 `mybatis-plus` ，在其提供的 `serviceImpl` 基础上封装了内部的BaseService；
  - 这是推荐的做法。

2.3 配置文件

> *使用mybatis默认的配置文件属性即可，但是⬇️*

- ⚠️*mybatis-plus对于mapperXML文件的定位无法使用mybatis为前缀的配置：*
  - *所以推荐将 `mapperXML文件` 放到 `/resources/<DAO包名层级>/` 目录下，确保编译后 `mapperXML文件` 会和对应 `Mapper对象` 处于同一目录中。*
- *日志输出， `mybatis-plus` 和 `tk-mybatis` 对于下方配置均生效。*

YAML | 复制代码

```yaml
mybatis:
  mapper-locations: ["classpath*:/mapper/**/*.xml"]
  configuration:
    ### 关闭日志打印org.apache.ibatis.logging.nologging.NoLoggingImpl
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
pagehelper:
  helper-dialect: mysql
  reasonable: true
  ### 改参数决定分页时能否拿到total
  default-count: true

spring:
  datasource:
    # type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://192.168.204.101:23306/dev_test?autoReconnect=true&useServerPreparedStmts=true&cachePrepStmts=true&rewriteBatchedStatements=true&allowMultiQueries=true&characterEncoding=utf8&serverTimezone=Asia/Shanghai&allowPublicKeyRetrieval=true
    username: root
    password: Mysql12345
    type: com.zaxxer.hikari.HikariDataSource
    hikari:
      ### 最小空闲连接，默认值10，小于0或大于maximum-pool-size，都会重置为maximum-pool-size
      minimum-idle: 10
      ### 最大连接数，小于等于0会被重置为默认值10；大于零小于1会被重置为minimum-idle的值
      maximum-pool-size: 20
      ### 连接池名称，默认HikariPool-1
      pool-name: HikariPool-Abc-1
      ### 连接超时时间:毫秒，小于250毫秒，会被重置为默认值30秒
      connection-timeout: 10000
      ### 连接测试查询
      connection-test-query: SELECT 1
      ### 空闲连接超时时间，默认值600000（10分钟），大于等于max-lifetime且max-lifetime>0，会被重置为0；不等于0且小于10秒，会被重置为10秒。
      idle-timeout: 600000
      ### 连接最大存活时间.不等于0且小于30秒，会被重置为默认值30分钟.设置应该比mysql设置的超时时间短；单位ms
      max-lifetime: 1800000
      ### 自动提交从池中返回的连接，默认值为true
      auto-commit: true
```

三、如何同时使用 `tk-mybatis` 和 `mybatis-plus`？

> 首先明确：前者在Mapper层进行抽象，后者在Mapper层和Service层进行抽象。
>
> - 我们可以在Mapper层同时使用二者提供的接口，在Service层上基于 `mybatis-plus` 进行二次封装；
> - 同时可以在 `tk-mybatis` 的Mapper层进行拓展。

3.1 实际案例

> `SpringBoot` + `tk-mybatis` + `mybatis-plus`

3.1.1 引入依赖

> 注意事项（lombok和mapstruct）：
>
> -

```xml
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.5.3.Final</version>
</dependency>
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct-processor</artifactId>
    <version>1.5.3.Final</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok-mapstruct-binding</artifactId>
    <version>0.2.0</version>
</dependency>
```

XML | 复制代码

```xml
<?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>2.7.11</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>spring-boot-mpAndTk</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-boot-mpAndTk</name>
    <description>spring-boot-mpAndTk</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>

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

        <!-- mybatis-plus -->
        <!-- tk-mybatis -->
        <!-- pagehelper -->
        <!-- mysql-connector-java -->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>3.5.3.1</version>
        </dependency>
        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper-spring-boot-starter</artifactId>
            <version>4.2.2</version>
        </dependency>
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper-spring-boot-starter</artifactId>
```

```
            <version>1.4.6</version>

        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.31</version>
        </dependency>

        <!-- commons-lang3 -->
        <!-- fastjson2 -->
        <!-- lombok -->
        <!-- mapstruct -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.12.0</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba.fastjson2</groupId>
            <artifactId>fastjson2</artifactId>
            <version>2.0.27</version>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct</artifactId>
            <version>1.5.3.Final</version>
        </dependency>
        <dependency>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct-processor</artifactId>
            <version>1.5.3.Final</version>
        </dependency>

        <!-- test -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
```

```
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

3.1.2 `MapperScan` 注解

> 添加在启动类上，同时配置二者的 `@MapperScan`

Java | 复制代码

```java
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@MapperScan({"com.example.springbootmpandtk.dal.persistence"})
@tk.mybatis.spring.annotation.MapperScan({"com.example.springbootmpandtk.dal.persistence"})
public class SpringBootMpAndTkApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootMpAndTkApplication.class, args);
    }

}
```

3.1.3 Mapper层接口继承

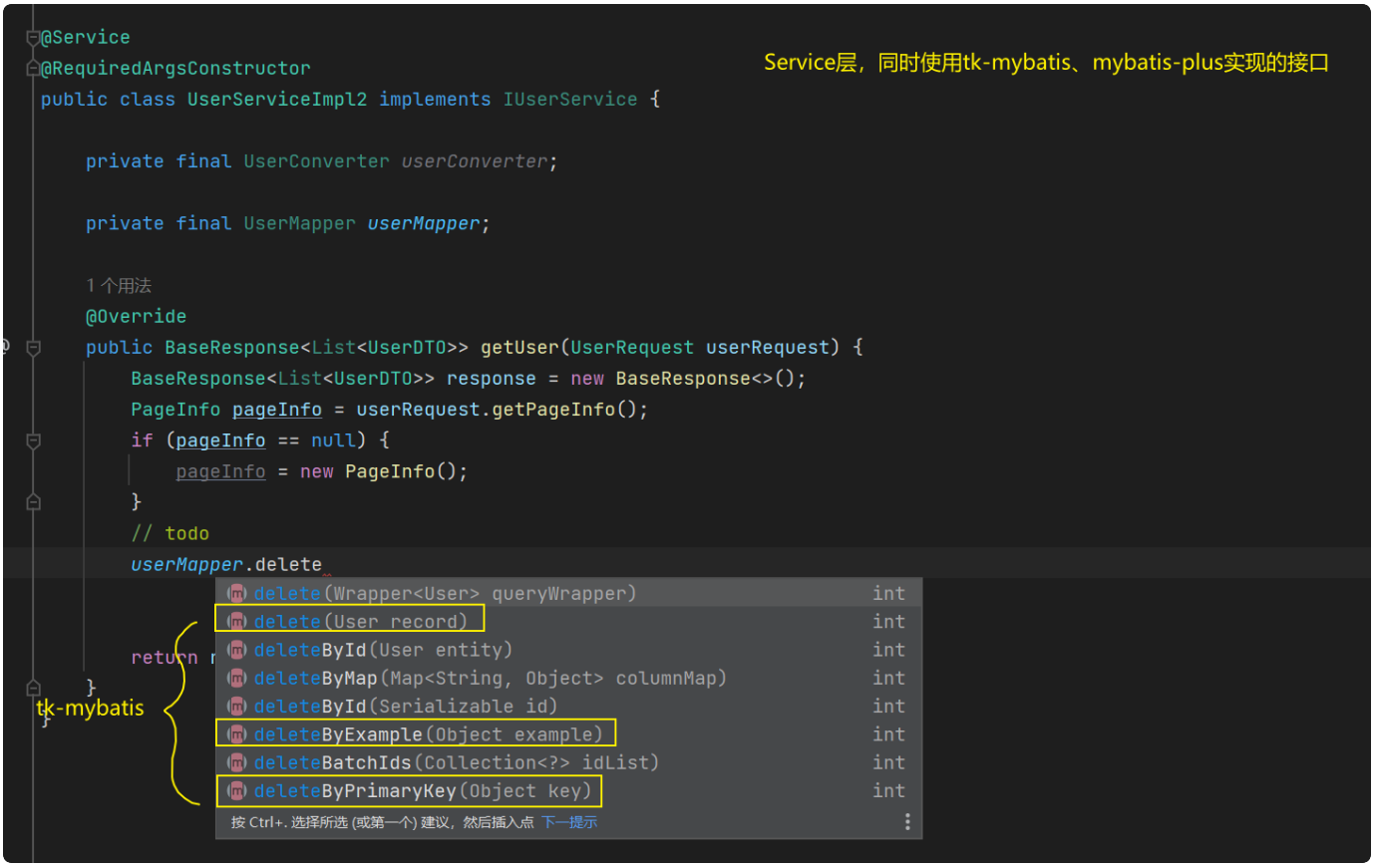
> 同时继承 `mybatis-plus` 的BaseMapper<T>和 `tk-mybatis` 的Mapper<T>

```java
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.example.springbootmpandtk.dal.entity.User;
import tk.mybatis.mapper.common.Mapper;

public interface UserMapper extends BaseMapper<User>, Mapper<User> {
}
```

3.1.4 Service层注入Mapper

> 此时，你可以使用 `tk-mybatis` 和 `mybatis-plus` 同时提供的多种API实现



3.1.5 在Service层针对 `mybatis-plus` 提供的 `ServiceImpl` 进行二次封装

Java | 复制代码

```java
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.baomidou.mybatisplus.extension.service.IService;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

public class BaseService<M extends BaseMapper<T>, T> extends ServiceImpl<M, T> implements IService<T> {

}
```

END