一、 DAO层的实现的规律

- 实体类与数据表存在对应关系，并且是有规律的——只要知道了数据表的结构，就能够生成实体类；
- 所有实体的DAO接口中定义的方法也是有规律的，不同点就是实体类型不同
  - UserDAO

    ```
    public interface UserDAO extends GeneralDAO<User>{
        public int insert(User t);
    }
    ```

  - GoodsDAO

    ```
    public interface GoodsDAO extends GeneralDAO<Goods> {
       public int insert(Goods t);
    }
    ```

  - GeneralDAO

    ```
    public interface GeneralDAO<T>{
        //通用方法
        public int insert(T t);
        public T queryOneByPrimarykey(int i);
    }
    ```

- 对于GeneralDAO接口定义的数据库操作方法因为使用了泛型，`无需映射文件`；对于UserDAO和GoodsDAO需要映射文件，所有DAO的相同操作的映射文件也是有规律可循的
  - UserMapper

    ```
    <insert id="insert">
      insert into users(user_id,username) values(#{userId},#{username})
    </insert>
    ```

```java
@Table("users")
public class User{

    @Id
    @Column("user_id")
    private int userId;

    @Column("username")
    private String username;

}
```

- GoodsMapper

```xml
<insert id="insert">
  insert into goods(goods_id,goods_name) values(#{goodsId},#{goodsName})
</insert>
```

```java
@Table("product")
public class Goods{
    @Id
    @Column("goods_id")
    private int goodsId;

    @Column("goods_name")
    private String goodsName;
}
```

二、tkMapper简介

> 基于MyBatis提供了很多第三方插件，这些插件通常可以完成数据操作方法的封装（GeneralDAO）、数据库逆向工程工作(根据数据表生成实体类、生成映射文件)
>
> - MyBatis-plus
> - tkMapper

tkMapper就是一个MyBatis插件，是在MyBatis的基础上提供了很多工具，让开发变得简单，提高开发效率。

- 提供了针对单表通用的数据库操作方法

- 逆向工程（根据数据表生成实体类、dao接口、映射文件）

三、tkMapper整合

3.1 基于SpringBoot完成MyBatis的整合

3.2 整合tkMapper

3.2.1 添加tkMapper的依赖

```xml
<dependency>
    <groupId>tk.mybatis</groupId>
    <artifactId>mapper-spring-boot-starter</artifactId>
    <version>2.1.5</version>
</dependency>
```

3.2.2 修改启动类的`@MapperScan`注解的包

- 为`tk.mybatis.spring.annotation.MapperScan`

```java
import tk.mybatis.spring.annotation.MapperScan;

@SpringBootApplication
@MapperScan("com.qfedu.tkmapperdemo.dao")
public class TkmapperDemoApplication {

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

}
```

四、tkMapper使用

4.1 创建数据表

```sql
CREATE TABLE `users`  (
  `user_id` int(64) NOT NULL AUTO_INCREMENT COMMENT '主键id 用户id',
  `username` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '用户名 用户名',
  `password` varchar(64) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '密码 密码',
  `nickname` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '昵称 昵称',
  `realname` varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '真实姓名 真实姓名',
  `user_img` varchar(1024) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '头像 头像',
  `user_mobile` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '手机号 手机号',
  `user_email` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '邮箱地址 邮箱地址',
  `user_sex` char(1) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '性别 M(男) or F(女)',
  `user_birth` date NULL DEFAULT NULL COMMENT '生日 生日',
  `user_regtime` datetime(0) NOT NULL COMMENT '注册时间 创建时间',
  `user_modtime` datetime(0) NOT NULL COMMENT '更新时间 更新时间',
  PRIMARY KEY (`user_id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 2 CHARACTER SET = utf8 COLLATE = utf8_general_ci COMMENT = '用户 ' ROW_FORMAT = Compact;

```

4.2 创建实体类

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {

    private int userId;
    private String username;
    private String password;
    private String nickname;
    private String realname;
    private String userImg;
    private String userMobile;
    private String userEmail;
    private String userSex;
    private Date userBirth;
    private Date userRegtime;
    private Date userModtime;

}
```

4.3 创建DAO接口

> tkMapper已经完成了对单表的通用操作的封装，封装在Mapper接口和MySqlMapper接口；因此如果我们要完成对单表的操作，只需自定义DAO接口继承Mapper接口和MySqlMapper接口

```java
public interface UserDAO extends Mapper<User>, MySqlMapper<User> {
}
```

4.4 测试

```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = TkmapperDemoApplication.class)
public class UserDAOTest {

    @Autowired
    private UserDAO userDAO;

    @Test
    public void test(){
        User user = new User();
        user.setUsername("aaaa");
        user.setPassword("1111");
        user.setUserImg("img/default.png");
        user.setUserRegtime(new Date());
        user.setUserModtime(new Date());
        int i = userDAO.insert(user);
        System.out.println(i);
    }

}
```

五、tkMapper提供的方法

```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = TkmapperDemoApplication.class)
public class CategoryDAOTest {
    @Autowired
    private CategoryDAO categoryDAO;

    @Test
    public void testInsert(){
        Category category = new Category(0,"测试类别
3",1,0,"03.png","xixi","aaa.jpg","black");
        //int i = categoryDAO.insert(category);
        int i = categoryDAO.insertUseGeneratedKeys(category);
        System.out.println(category.getCategoryId());
        assertEquals(1,i);
    }

    @Test
    public void testUpdate(){
        Category category = new Category(48,"测试类别
4",1,0,"04.png","heihei","aaa.jpg","black");
        int i = categoryDAO.updateByPrimaryKey(category);
        // 根据自定义条件修改，Example example就是封装条件的
        // int i1 = categoryDAO.updateByExample( Example example);
        assertEquals(1,i);
    }

    @Test
    public void testDelete(){
        int i = categoryDAO.deleteByPrimaryKey(48);
        // 根据条件删除
        //int i1 = categoryDAO.deleteByExample(Example example);
        assertEquals(1,i);
    }

    @Test
    public void testSelect1(){
        //查询所有
        List<Category> categories = categoryDAO.selectAll();
        for (Category category: categories) {
            System.out.println(category);
        }
    }

    @Test
    public void testSelect2(){
        //根据主键查询
        Category category = categoryDAO.selectByPrimaryKey(47);
        System.out.println(category);
    }

    @Test
    public void testSelect3(){
        //条件查询
        //1.创建一个Example封装 类别Category查询条件
        Example example = new Example(Category.class);
        Example.Criteria criteria = example.createCriteria();
        criteria.andEqualTo("categoryLevel",1);
        criteria.orLike("categoryName","%干%");

        List<Category> categories = 
categoryDAO.selectByExample(example);
        for (Category category: categories) {
            System.out.println(category);
        }
    }

    @Test
    public void testSelect4(){
        //分页查询
        int pageNum = 2;
        int pageSize = 10;
        int start = (pageNum-1)*pageSize;

        RowBounds rowBounds = new RowBounds(start,pageSize);
        List<Category> categories = categoryDAO.selectByRowBounds(new 
Category(), rowBounds);
        for (Category category: categories) {
            System.out.println(category);
        }

        //查询总记录数
        int i = categoryDAO.selectCount(new Category());
        System.out.println(i);
    }


    @Test
    public void testSelect5(){
        //带条件分页
        //条件
        Example example = new Example(Category.class);
        Example.Criteria criteria = example.createCriteria();
        criteria.andEqualTo("categoryLevel",1);
        //分页
        int pageNum = 2;
        int pageSize = 3;
        int start = (pageNum-1)*pageSize;
        RowBounds rowBounds = new RowBounds(start,pageSize);

        List<Category> categories =
categoryDAO.selectByExampleAndRowBounds(example,rowBounds);
        for (Category category: categories) {
            System.out.println(category);
        }

        //查询总记录数（满足条件）
        int i = categoryDAO.selectCountByExample(example);
        System.out.println(i);
    }

}
```

六、在使用tkMapper是如何进行关联查询

6.1 所有的关联查询都可以通过多个单表操作实现

```java
//查询用户同时查询订单
Example example = new Example(User.class);
Example.Criteria criteria = example.createCriteria();
criteria.andEqualTo("username","zhangsan");
//根据用户名查询用户
//1.先根据用户名查询用户信息
List<User> users = userDAO.selectByExample(example);
User user = users.get(0);
//2.再根据用户id到订单表查询订单
Example example1 = new Example(Orders.class);
Example.Criteria criteria1 = example1.createCriteria();
criteria1.andEqualTo("userId",user.getUserId());
List<Orders> ordersList = orderDAO.selectByExample(example1);
//3.将查询到订单集合设置到user
user.setOrdersList(ordersList);
```

```java

System.out.println(user);
```

6.2 自定义连接查询

- 在使用tkMapper,DAO继承Mapper和MySqlMapper之后，还可以自定义查询

6.2.1 在DAO接口自定义方法

```java
public interface UserDAO extends GeneralDAO<User> {

    public User selectByUsername(String username);

}
```

6.2.2 创建Mapper文件

```xml
<?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.qfedu.fmmall.dao.UserDAO">

    <insert id="insertUser">
        insert into users(username,password,user_img,user_regtime,user_modtime)
        values(#{username},#{password},#{userImg},#{userRegtime},#{userModtime})
    </insert>

    <resultMap id="userMap" type="User">
        <id column="user_id" property="userId"/>
        <result column="username" property="username"/>
        <result column="password" property="password"/>
        <result column="nickname" property="nickname"/>
        <result column="realname" property="realname"/>
        <result column="user_img" property="userImg"/>
        <result column="user_mobile" property="userMobile"/>
        <result column="user_email" property="userEmail"/>
        <result column="user_sex" property="userSex"/>
        <result column="user_birth" property="userBirth"/>
        <result column="user_regtime" property="userRegtime"/>
```

```xml
            <result column="user_modtime" property="userModtime"/>
        </resultMap>

        <select id="queryUserByName" resultMap="userMap">
            select
                user_id,
                username,
                password,
                nickname,
                realname,
                user_img,
                user_mobile,
                user_email,
                user_sex,
                user_birth,
                user_regtime,
                user_modtime
            from users
            where username=#{name}
        </select>

</mapper>
```

七、逆向工程

> 逆向工程，根据创建好的数据表，生成实体类、DAO、映射文件

7.1 添加逆向工程依赖

> 是依赖是一个mybatis的maven插件

```xml
<plugin>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-maven-plugin</artifactId>
    <version>1.3.5</version>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.47</version>
        </dependency>
        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper</artifactId>
            <version>3.4.4</version>
        </dependency>
    </dependencies>
</plugin>
```

7.2 逆向工程配置

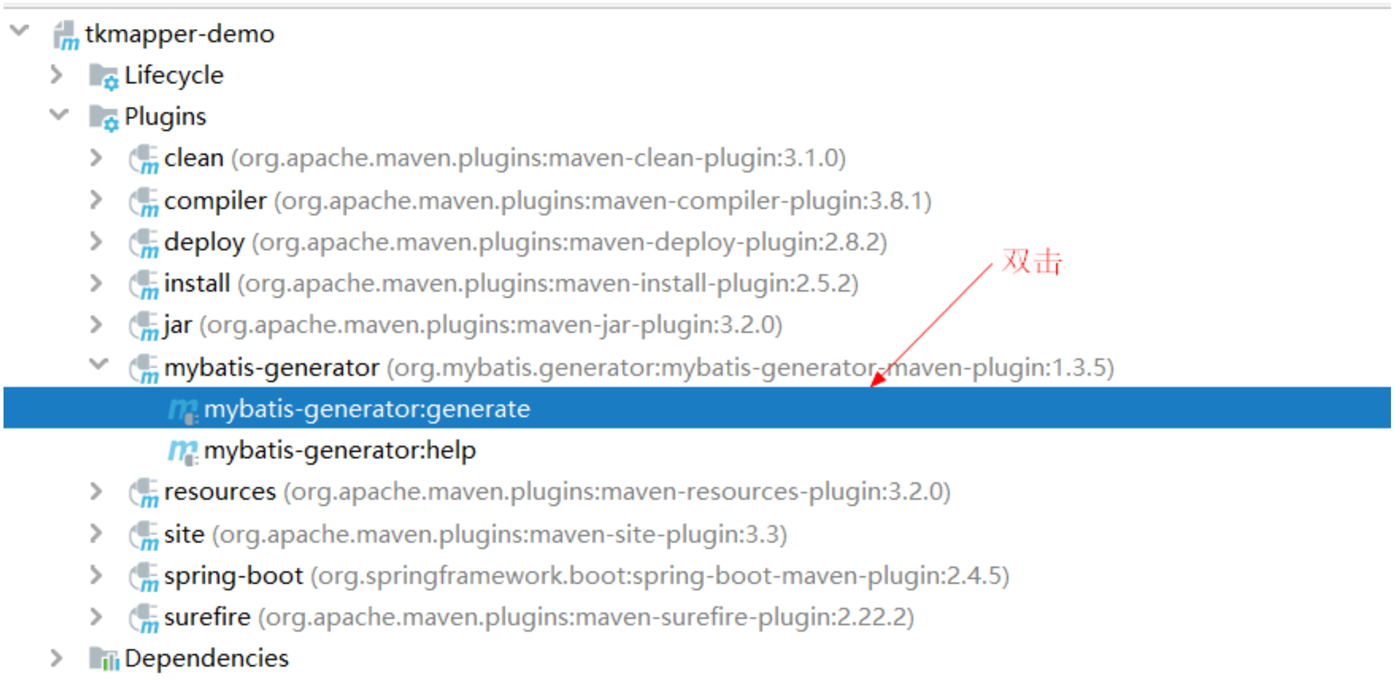
- 在resources/generator目录下创建generatorConfig.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
        PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
        "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>
    <!-- 引入数据库连接配置 -->
<!--    <properties resource="jdbc.properties"/>-->

    <context id="Mysql" targetRuntime="MyBatis3Simple" defaultModelType="flat">
        <property name="beginningDelimiter" value="`"/>
        <property name="endingDelimiter" value="`"/>

        <!-- 配置 GeneralDAO -->
        <plugin type="tk.mybatis.mapper.generator.MapperPlugin">
            <property name="mappers" value="com.qfedu.tkmapperdemo.general.GeneralDAO"/>
        </plugin>

        <!-- 配置数据库连接 -->
        <jdbcConnection driverClass="com.mysql.jdbc.Driver"
                connectionURL="jdbc:mysql://localhost:3306/fmmall2"
                userId="root" password="admin123">
        </jdbcConnection>

        <!-- 配置实体类存放路径 -->
        <javaModelGenerator targetPackage="com.qfedu.tkmapperdemo.beans" targetProject="src/main/java"/>

        <!-- 配置 XML 存放路径 -->
        <sqlMapGenerator targetPackage="/" targetProject="src/main/resources/mappers"/>

        <!-- 配置 DAO 存放路径 -->
        <javaClientGenerator targetPackage="com.qfedu.tkmapperdemo.dao" targetProject="src/main/java" type="XMLMAPPER"/>

        <!-- 配置需要指定生成的数据库和表，% 代表所有表 -->
        <table tableName="%">
            <!-- mysql 配置 -->
<!--            <generatedKey column="id" sqlStatement="Mysql" identity="true"/>-->
        </table>
<!--        <table tableName="tb_roles">-->
<!--            &lt;!&ndash; mysql 配置 &ndash;&gt;-->
<!--            <generatedKey column="roleid" sqlStatement="Mysql" identity="true"/>-->
<!--        </table>-->
<!--        <table tableName="tb_permissions">-->
<!--            &lt;!&ndash; mysql 配置 &ndash;&gt;-->
<!--            <generatedKey column="perid" sqlStatement="Mysql" identity="true"/>-->
<!--        </table>-->
    </context>
</generatorConfiguration>
```

7.3 将配置文件设置到逆向工程的maven插件

```xml
<plugin>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-maven-plugin</artifactId>
    <version>1.3.5</version>
    <configuration>
        <configurationFile>${basedir}/src/main/resources/generator/generatorConfig.xml</configurationFile>
    </configuration>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.47</version>
        </dependency>
        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper</artifactId>
            <version>3.4.4</version>
        </dependency>
    </dependencies>
</plugin>
```

7.4 执行逆向生成



tkmapper-demo
Lifecycle
Plugins
clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)
compiler (org.apache.maven.plugins:maven-compiler-plugin:3.8.1)
deploy (org.apache.maven.plugins:maven-deploy-plugin:2.8.2)
install (org.apache.maven.plugins:maven-install-plugin:2.5.2)
jar (org.apache.maven.plugins:maven-jar-plugin:3.2.0)
mybatis-generator (org.mybatis.generator:mybatis-generator-maven-plugin:1.3.5)
mybatis-generator:generate
mybatis-generator:help
resources (org.apache.maven.plugins:maven-resources-plugin:3.2.0)
site (org.apache.maven.plugins:maven-site-plugin:3.3)
spring-boot (org.springframework.boot:spring-boot-maven-plugin:2.4.5)
surefire (org.apache.maven.plugins:maven-surefire-plugin:2.22.2)
Dependencies
双击

千锋教育Java教研院 关注公众号【Java架构栈】下载所有课程代码课件及工具 让技术回归本该有的纯静！