TkMapper

1. 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的方法的封装，数据库逆向生成等等。

- MyBatisPlus
- 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。

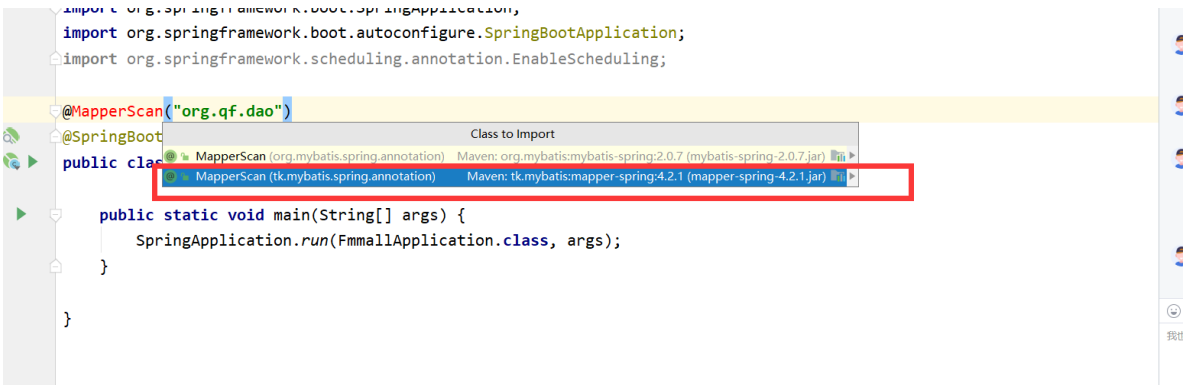
- 提供了针对单表通用的数据库操作方法
- 逆向工程

2. TkMapper整合Springboot

2.1 添加依赖

```
<dependency>
    <groupId>tk.mybatis</groupId>
    <artifactId>mapper-spring-boot-starter</artifactId>
    <version>4.2.1</version>
</dependency>
```

2.2 修改启动类的@MappperScan注解包



2.3 tkMapper的使用

1. 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，只需要自定义Dao接口继承Mapper和MySqlMapper接口

```
@Component
public interface UsersMapper extends Mapper<Users>, MySqlMapper<Users> {

    /**
     * 注册功能
     * @param users
```

对于单表来说，接口无需定义，那么mapper.xml无需使用

2. 业务层的实现

```
    //注册
    @Override
    public ResultVO userResgit(String name, String pwd) {
        //1.根据用户查询，这个用户是否已经注册
        Example example=new Example(Users.class);
        Example.Criteria criteria=example.createCriteria();
        criteria.andEqualTo( property: "username",name);
        List<Users> users = usersMapper.selectByExample(example);  //通过用户名进行条件查询的结果
        //2. 如果没有注册则进行保存操作
        if(users.size()==0){
            //密码加密  md5
            String md5Pwd=MD5Utils.md5(pwd);
            Users user=new Users();
            user.setUsername(name);
            user.setPassword(md5Pwd);
```

3. 自定义接口

> 在使用tkmapperdao继承mapper和MySqlMapper，可以 自定义接口方法

3.1定义工具类

> 继承mapper和mySqlmapper

```java
package org.qf.general;

import tk.mybatis.mapper.common.Mapper;
import tk.mybatis.mapper.common.MySqlMapper;

public interface GeneralDao<T> extends Mapper<T>, MySqlMapper<T> {

}
```

3.2 在dao接口中使用

```java
package org.qf.dao;

import org.qf.entity.Users;
import org.qf.general.GeneralDao;
import org.springframework.stereotype.Component;
import tk.mybatis.mapper.common.Mapper;
import tk.mybatis.mapper.common.MySqlMapper;

@Component
public interface UsersMapper extends GeneralDao<Users> {

    public boolean getUser();

}
```

4. 《锋迷商城》-商品搜索

4.1 SQL分析

```
# 商品搜索功能

select * from product where product_name like '%a%' limit 0,2

# limit 分页的关键字
# 查询第一页，一页显示2条数据， 共12条数据  6页  第一页的sql怎么写
product

select * from product limit 0,2

select * from product limit 2,2   limit （当前页-1）*页面容量，页面容量
# 页面容量：一页显示的条数

select * from product limit 4,2
```

4.2 后端实现

4.2.1 dao接口

```
    /**
     * 根据关键字模糊搜索商品信息
     * @param keyword  搜索的关键字
     * @param start    分页的第一个参数
     * @param limit    页面容量
     * @return
     */
    public List<ProductVO> selectProductByKw(@Param("kw")
String keyword,@Param("start") int start,@Param("limit") int
 limit);
```

4.2.2 mapper映射文件

```
 <select id="selectProductByKw" resultMap="productMapVo">
        select product_id, product_name, category_id,
sold_num, product_status, content, create_time, update_time
from product where product_name like #{kw} limit #{start},#
{limit}
    </select>
```

4.2.3 业务层接口

```
    /**
     *
     * @param kw  关键字
     * @param pageNum  当前页
     * @param limit  页面容量
     * @return
     */
    public ResultVO searchProduct(String kw,int pageNum,int
limit);
```

4.2.4 业务层接口实现类

```
 @Override
    public ResultVO searchProduct(String kw, int pageNum, int
limit) {
        //1.查询搜索结果
        kw='%'+kw+'%';
        int start=(pageNum-1)*limit;
        List<ProductVO> productVOS =
productMapper.selectProductByKw(kw, start, limit);

        //2. 查询总记录数
        Example example=new Example(Product.class);
        Example.Criteria criteria=example.createCriteria();
        criteria.andEqualTo("productName",kw);
        int count =
productMapper.selectCountByExample(example);

        //3.计算总页数
        int pageCount=count%limit==0?
count/limit:count/limit+1;

        //4.把数据封装到一个工具类中
        PageHelper<ProductVO> pageHelper=new PageHelper<>
(count,pageCount,productVOS);
        ResultVO resultVO=new
ResultVO(ResStatus.OK,"success",pageHelper);
        return resultVO;
    }
```

4.2.5 分页工具类

```
package org.qf.utils;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 分页的工具类
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class PageHelper<T> {

    //总记录数
    private Integer count;

    //总页数
    private Integer pageCount;

    //分页数据
    private List<T> list;
}
```

4.2.6 控制器

```
@RestController
@RequestMapping("/product")
@CrossOrigin   //允许跨域
@Api(value = "提供商品信息相关的接口",tags = "商品管理")
public class ProductController {

    @Resource
    private ProductService  productService;


    @ApiOperation("根据关键字查询商品接口")
```

```
    @GetMapping("/listbykeyword")
    @ApiImplicitParams({
            @ApiImplicitParam(dataType = "string",name =
"keyword",value = "搜索关键字",required = true),
            @ApiImplicitParam(dataType = "integer",name =
"pageNum",value = "当前页",required =true),
            @ApiImplicitParam(dataType = "integer",name =
"limit",value = "每页显示的条数",required =true)
    })
    public ResultVO searchProducts(String keyword,int
pageNum,int limit){
        return productService.searchProduct(keyword, pageNum,
limit);
    }
 }
```