**实验四 MybatisPlus实验**

**一、实验目的：**

(1)练习掌握SpringBoot程序的创建。

(2)练习掌握MybatisPlus的配置。

(3)练习掌握xml映射文件的编写及动态sql的编写。

(4)练习掌握对多表的增删改查功能，包括1对1和1对多的联查操作。

**二、实验内容：**

（1）创建SpringBoot程序，根据User表完成单表的增删改查操作。在Service层实现代码进行功能测试，要求独立实现，不允许代码自动生成。

代码：

Mapper层接口：

*package* top.qwwq.mapper;  
  
*import* org.apache.ibatis.annotations.*Mapper*;  
*import* top.qwwq.pojo.User;  
  
*@Mapper  
public interface* UserMapper {  
 */\*  
 \* 增加用户  
 \*/  
 void* insertUser(User user);  
 */\*  
 \* 删除用户  
 \*/  
 void* deleteUser(Long userID);  
 */\*  
 \* 修改用户  
 \*/  
 void* updateUser(User user);  
 */\*  
 \* 查询用户  
 \*/* User selectUser(Long userID);  
  
  
}

Mapper层xml文件：

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "https://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="top.qwwq.mapper.UserMapper">  
*<!-- 增加用户-->* <insert id="insertUser" parameterType="User">  
 insert into `mybatisplus-4`.user (user\_id, login\_name, password, sex, mobile, email)  
 value (#{userID}, #{loginName}, #{password}, #{sex}, #{mobile}, #{email})  
 </insert>  
*<!-- 删除用户-->* <delete id="deleteUser">  
 delete from `mybatisplus-4`.user  
 where user\_id=#{userID}  
 </delete>  
*<!-- 修改用户-->* <update id="updateUser" parameterType="User">  
 update `mybatisplus-4`.user   
 set login\_name = #{loginName},password = #{password},sex = #{sex},mobile = #{mobile},email = #{email}  
 where user\_id=#{userID}  
 </update>  
*<!-- 查找用户-->* <select id="selectUser" resultType="User">  
 select \*  
 from `mybatisplus-4`.user  
 where user\_id=#{userID}  
 </select>  
  
  
  
</mapper>

Service层接口：

*package* top.qwwq.service;  
  
*import* top.qwwq.mapper.UserMapper;  
*import* top.qwwq.pojo.User;  
*import* top.qwwq.utils.ResponseVo;  
  
*public interface* UserService {  
 *public* ResponseVo<String> insertUser(User user);  
  
 *public* ResponseVo<String> deleteUser(Long userID);  
  
 *public* ResponseVo<String> updateUser(User user);  
  
 *public* ResponseVo<User> selectUser(Long userID);  
}

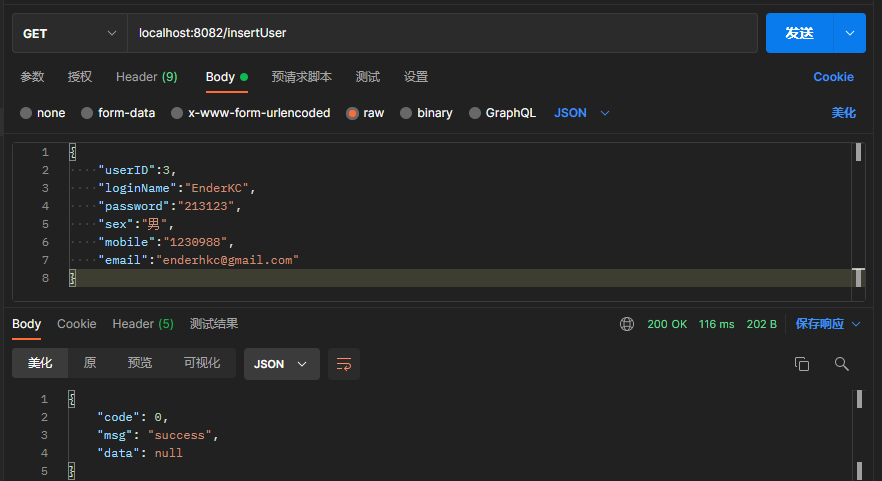
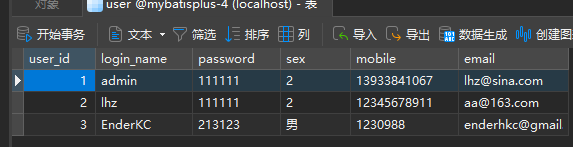
Service层实现：

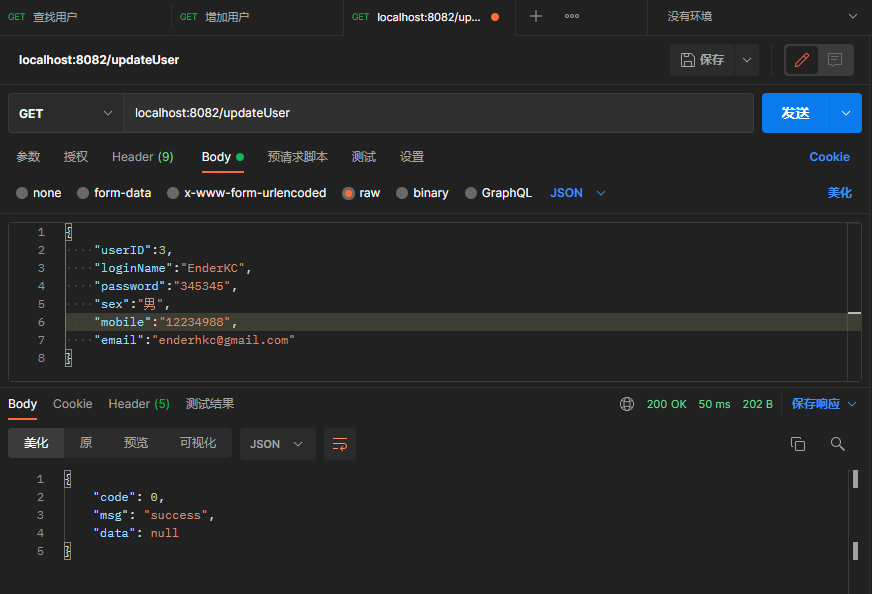
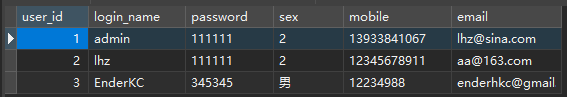
*package* top.qwwq.service;  
  
*import* org.springframework.stereotype.*Service*;  
*import* top.qwwq.mapper.UserMapper;  
*import* top.qwwq.pojo.User;  
*import* top.qwwq.utils.ResponseVo;  
  
*@Service  
public class* UserServiceImpl *implements* UserService {  
 UserMapper userMapper;  
  
 *public* UserServiceImpl(UserMapper userMapper) {  
 *this*.userMapper = userMapper;  
  
 }  
  
 *// 增加用户  
 public* ResponseVo<String> insertUser(User user) {  
 userMapper.insertUser(user);  
 *return new* ResponseVo<>(0, "success");  
 }  
  
 *// 删除用户  
 public* ResponseVo<String> deleteUser(Long userID) {  
 userMapper.deleteUser(userID);  
 *return new* ResponseVo<>(0, "success");  
 }  
  
 *// 修改用户  
 public* ResponseVo<String> updateUser(User user) {  
 userMapper.updateUser(user);  
 *return new* ResponseVo<>(0, "success");  
 }  
  
 *// 查询用户  
 public* ResponseVo<User> selectUser(Long userID) {  
 User user = userMapper.selectUser(userID);  
 *return new* ResponseVo<>(0, "success", user);  
 }  
  
}

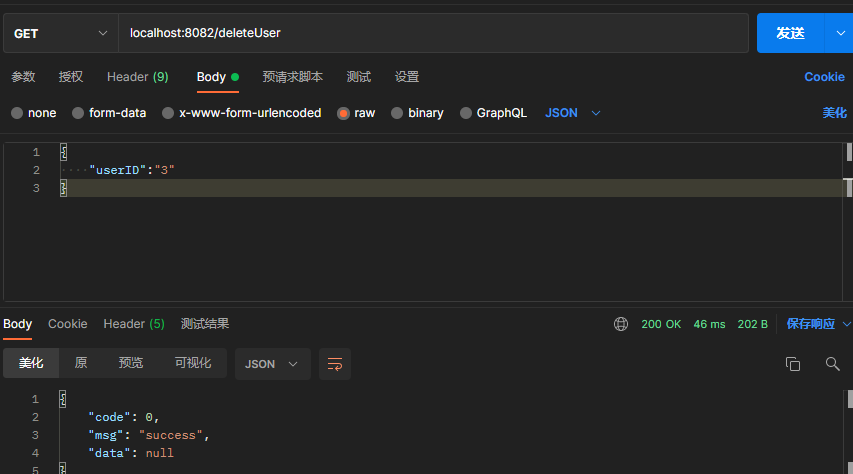
controller层实现：

*package* top.qwwq.controller;  
  
*import* com.alibaba.fastjson2.JSONObject;  
*import* org.springframework.web.bind.annotation.*RequestBody*;  
*import* org.springframework.web.bind.annotation.*RequestMapping*;  
*import* org.springframework.web.bind.annotation.*RestController*;  
*import* top.qwwq.pojo.User;  
*import* top.qwwq.service.UserService;  
*import* top.qwwq.utils.ResponseVo;  
  
*@RestController  
public class* UserController {  
 UserService userService;  
 *public* UserController(UserService userService) {  
 *this*.userService = userService;  
 }  
  
 *@RequestMapping*("/insertUser")  
 *public* ResponseVo<String> insertUser(*@RequestBody* User user){  
 *return* userService.insertUser(user);  
 }  
  
 *@RequestMapping*("/deleteUser")  
 *public* ResponseVo<String> deleteUser(*@RequestBody* JSONObject request){  
 Long userID = request.getLong("userID");  
 *return* userService.deleteUser(userID);  
 }  
  
 *@RequestMapping*("/updateUser")  
 *public* ResponseVo<String> updateUser(*@RequestBody* User user){  
 *return* userService.updateUser(user);  
 }  
  
 *@RequestMapping*("/selectUser")  
 *public* ResponseVo<User> selectUser(*@RequestBody* JSONObject request){  
 Long userID = request.getLong("userID");  
 *return* userService.selectUser(userID);  
 }  
}

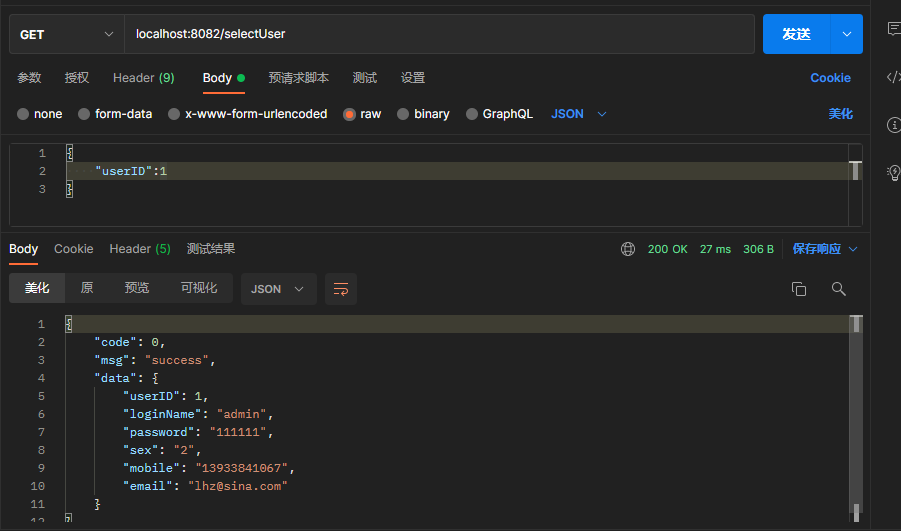
运行截图：

增加用户：

修改用户：

删除用户：



查找用户

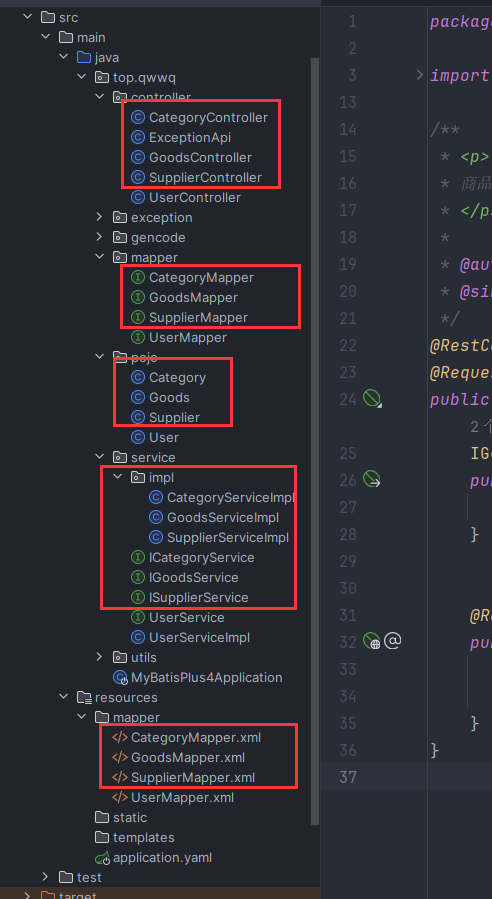
1. 根据Category表、Goods表、Suppiers表完成1对1连接查询操作，用代码自动生成实现对三个表的增删改查操作。在Service层实现goods表代码功能测试，要求独立实现。

代码：

代码生成的代码：

*package* top.qwwq.gencode;  
  
*import* com.baomidou.mybatisplus.generator.FastAutoGenerator;  
*import* com.baomidou.mybatisplus.generator.config.OutputFile;  
*import* com.baomidou.mybatisplus.generator.config.rules.DbColumnType;  
*import* com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;  
  
*import* java.sql.Types;  
*import* java.util.Collections;  
  
*public class* MySQLCodeGeneratop {  
 *public static void* main(String[] args) {  
 FastAutoGenerator.create("jdbc:mysql://localhost:3306/MyBatisPlus-4?characterEncoding=UTF-8&useUnicode=true&serverTimezone=GMT&connectTimeout=15000", "root", "20021001")  
 .globalConfig(builder -> {  
 builder.author("EnderKC") *// 设置作者* .enableSwagger() *// 开启 swagger 模式* .commentDate("yyyy-MM-dd") *// 设置日期格式* .fileOverride() *// 覆盖已生成文件* .outputDir(System.getProperty("user.dir")+"\\src\\main\\java"); *// 指定输出目录* })  
 .dataSourceConfig(builder -> builder.typeConvertHandler((globalConfig, typeRegistry, metaInfo) -> {  
 *int* typeCode = metaInfo.getJdbcType().TYPE\_CODE;  
 *if* (typeCode == Types.SMALLINT) {  
 *// 自定义类型转换  
 return* DbColumnType.INTEGER;  
 }  
 *return* typeRegistry.getColumnType(metaInfo);  
  
 }))  
 .packageConfig(builder -> {  
 builder.parent("top.qwwq") *// 设置父包名* .entity("pojo")  
 .service("service")  
 .serviceImpl("service.impl")  
 .mapper("mapper")  
 .xml("mapper.xml")  
 .controller("controller")  
 .pathInfo(Collections.singletonMap(OutputFile.xml, System.getProperty("user.dir")+"\\src\\main\\resources\\mapper")); *// 设置mapperXml生成路径* })  
 .strategyConfig(builder -> {  
 builder.addInclude("category") *// 设置需要生成的表名* .addInclude("goods")  
 .addInclude("supplier")  
 .entityBuilder()  
 .enableLombok(); *// 开启lombok模式* })  
 .templateEngine(*new* FreemarkerTemplateEngine()) *// 使用Freemarker引擎模板，默认的是Velocity引擎模板* .execute();  
 }  
}

自动生成的文件的截图：



多表连接查询：

Mapper层：

<?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="top.qwwq.mapper.GoodsMapper">  
 *<!-- 查-->* <select id="selectGoods" resultType="Map">  
 select category.cate\_name, supplier.supplier\_name, goods.\*  
 from `mybatisplus-4`.goods,  
 `mybatisplus-4`.category,  
 `mybatisplus-4`.supplier  
 where `mybatisplus-4`.goods.cate\_id = `mybatisplus-4`.category.cate\_id  
 and `mybatisplus-4`.goods.supplier\_id = `mybatisplus-4`.supplier.supplier\_id  
 and goods\_id = #{goodsId}  
 </select>  
  
</mapper>

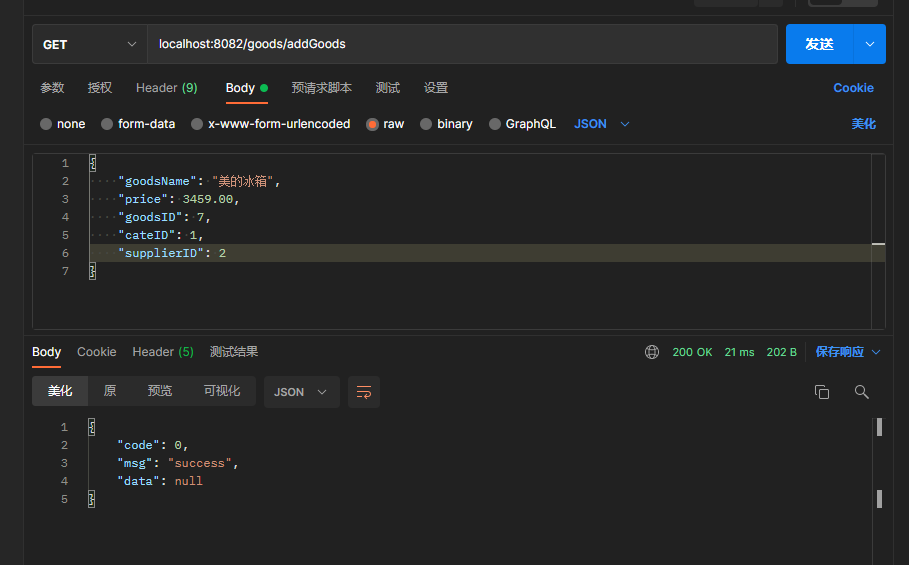
Service层：

*\*/  
@Service  
public class* GoodsServiceImpl *extends* ServiceImpl<GoodsMapper, Goods> *implements* IGoodsService {  
 GoodsMapper goodsMapper;  
 *public* GoodsServiceImpl(GoodsMapper goodsMapper) {  
 *this*.goodsMapper = goodsMapper;  
 }  
  
 *@Override  
 public* ResponseVo<Map<String, String>> selectGoods(*long* goodsID) {  
 Map<String,String> goods = goodsMapper.selectGoods(goodsID);  
 *if*(goods == *null*){  
 *return new* ResponseVo(-1,"商品不存在");  
 }  
 *return new* ResponseVo<>(0,"success",goods);  
 }  
}

Controller层：

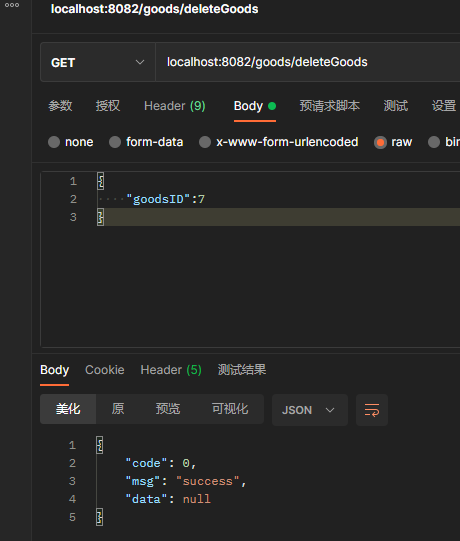
*package* top.qwwq.controller;  
  
*import* com.alibaba.fastjson2.JSONObject;  
*import* lombok.extern.slf4j.*Slf4j*;  
*import* org.springframework.web.bind.annotation.*RequestBody*;  
*import* org.springframework.web.bind.annotation.*RequestMapping*;  
*import* org.springframework.stereotype.*Controller*;  
*import* org.springframework.web.bind.annotation.*RestController*;  
*import* top.qwwq.mapper.GoodsMapper;  
*import* top.qwwq.pojo.Goods;  
*import* top.qwwq.pojo.User;  
*import* top.qwwq.service.IGoodsService;  
*import* top.qwwq.utils.ResponseVo;  
  
*import* java.util.Map;  
  
*/\*\*  
 \* <p>  
 \* 商品表 前端控制器  
 \* </p>  
 \*  
 \* @author EnderKC  
 \* @since 2023-12-03  
 \*/  
@RestController  
@RequestMapping*("/goods")  
*@Slf4j  
public class* GoodsController {  
 IGoodsService goodsService;  
 GoodsMapper goodsMapper;  
 *public* GoodsController(IGoodsService goodsService, GoodsMapper goodsMapper) {  
 *this*.goodsMapper = goodsMapper;  
 *this*.goodsService = goodsService;  
 }  
  
 *// 增加产品  
 @RequestMapping*("/addGoods")  
 *public* ResponseVo<String> addGoods(*@RequestBody* JSONObject request){  
 *int* goodsID = request.getIntValue("goodsID");  
 String goodsName = request.getString("goodsName");  
 *float* price = request.getFloat("price");  
 *int* cateID = request.getIntValue("cateID");  
 *int* supplierID = request.getIntValue("supplierID");  
 Goods goods = *new* Goods();  
 goods.setGoodsName(goodsName);  
 goods.setGoodsId(goodsID);  
 goods.setCateId(cateID);  
 goods.setSupplierId(supplierID);  
 goods.setPrice(price);  
 log.info(goods.toString());  
 *// 添加商品  
 try*{  
 goodsMapper.insert(goods);  
 *return new* ResponseVo<>(0,"success");  
 }*catch* (Exception e){  
 e.printStackTrace();  
 *return new* ResponseVo<>(-1,"添加失败");  
 }  
 }  
 *// 删除  
 @RequestMapping*("/deleteGoods")  
 *public* ResponseVo<String> deleteGoods(*@RequestBody* JSONObject request){  
 *int* goodsID = request.getIntValue("goodsID");  
 Goods goods = *new* Goods();  
 goods.setGoodsId(goodsID);  
 *try*{  
 goodsMapper.deleteById(goodsID);  
 *return new* ResponseVo<>(0,"success");  
 }*catch* (Exception e){  
 e.printStackTrace();  
 *return new* ResponseVo<>(-1,"删除失败");  
 }  
 }  
 *// 修改  
 @RequestMapping*("/updateGoods")  
 *public* ResponseVo<String> updateGoods(*@RequestBody* JSONObject request){  
 *int* goodsID = request.getIntValue("goodsID");  
 String goodsName = request.getString("goodsName");  
 *float* price = request.getFloat("price");  
 *int* cateID = request.getIntValue("cateID");  
 *int* supplierID = request.getIntValue("supplierID");  
 Goods goods = *new* Goods();  
 goods.setGoodsName(goodsName);  
 goods.setGoodsId(goodsID);  
 goods.setCateId(cateID);  
 goods.setSupplierId(supplierID);  
 goods.setPrice(price);  
 *try*{  
 goodsMapper.updateById(goods);  
 *return new* ResponseVo<>(0,"success");  
 }*catch* (Exception e){  
 e.printStackTrace();  
 *return new* ResponseVo<>(-1,"修改失败");  
 }  
 }  
  
  
 *@RequestMapping*("/selectGoods")  
 *public* ResponseVo<Map<String,String>> selectUser(*@RequestBody* JSONObject request){  
 Long goodsID = request.getLong("goodsID");  
 *return* goodsService.selectGoods(goodsID);  
 }  
}

运行截图：

增加

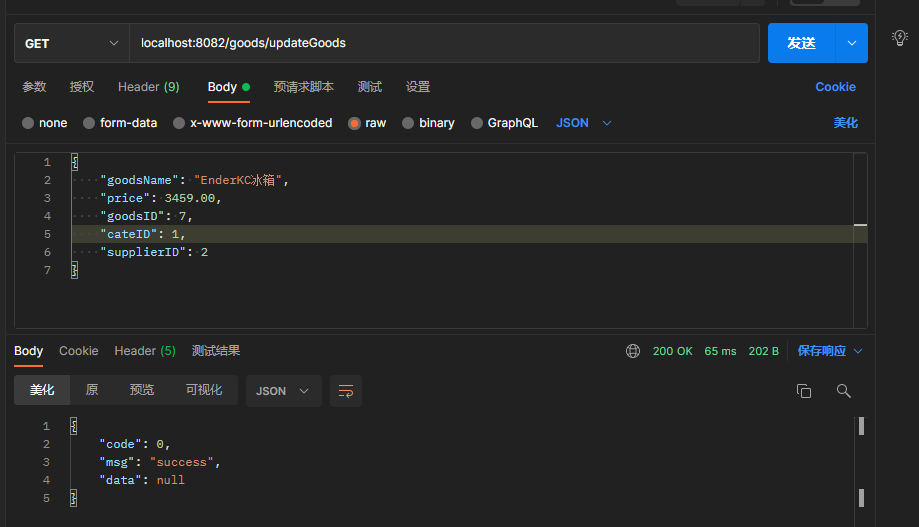
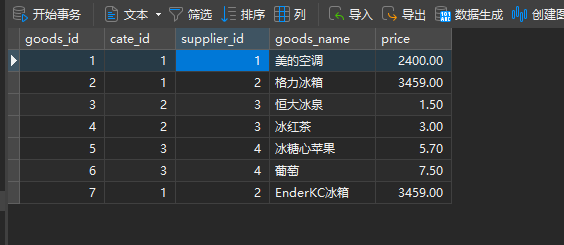


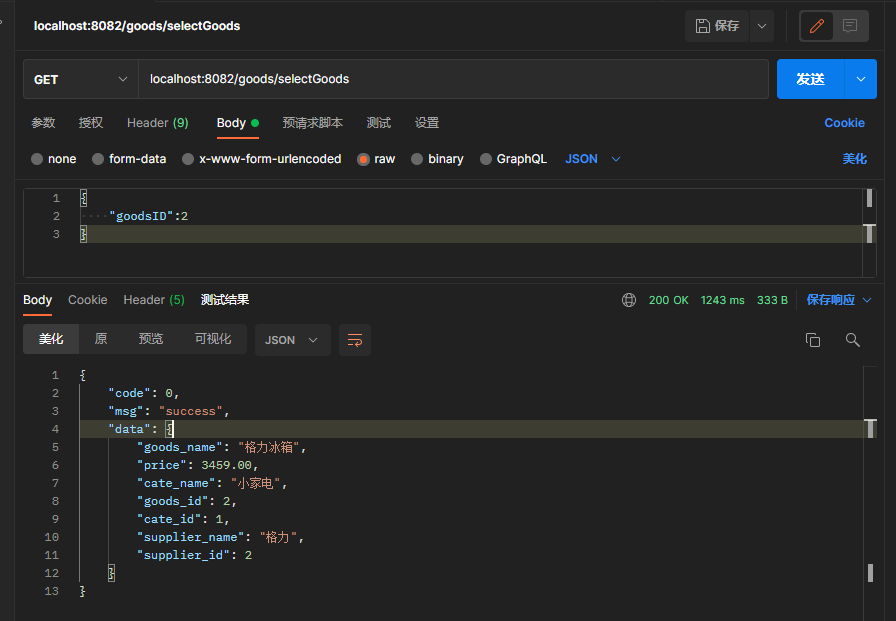
删除





修改：



查询：

1. 根据Category表、Goods表、Suppiers表完成1对多连接查询操作，要求分别按1+N方式和多表连接方式分别实现。在Service层实现代码功能测试。

代码：

接口：

*// 1+n 方式查询*Category selectCategoryAndGoods(*int* cateID);

Xml文件：

<resultMap id="CategoryResultMap" type="Category">  
 <id property="cateID" column="cate\_id" />  
 <id property="cateName" column="cate\_name" />  
 <collection property="goodsList" ofType="Goods" column="cate\_id" select="selectGoodsByCategoryId"/>  
</resultMap>  
  
<select id="selectCategoryAndGoods" resultMap="CategoryResultMap">  
 SELECT c.\*  
 FROM `mybatisplus-4`.category c  
 WHERE c.cate\_id = #{cateId}  
</select>

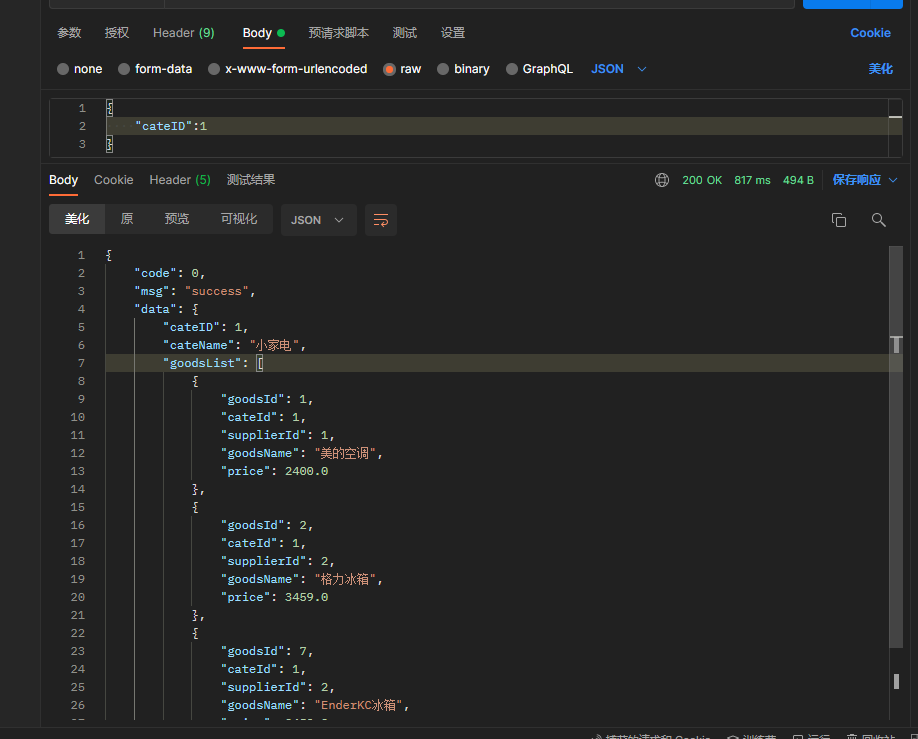
接口：

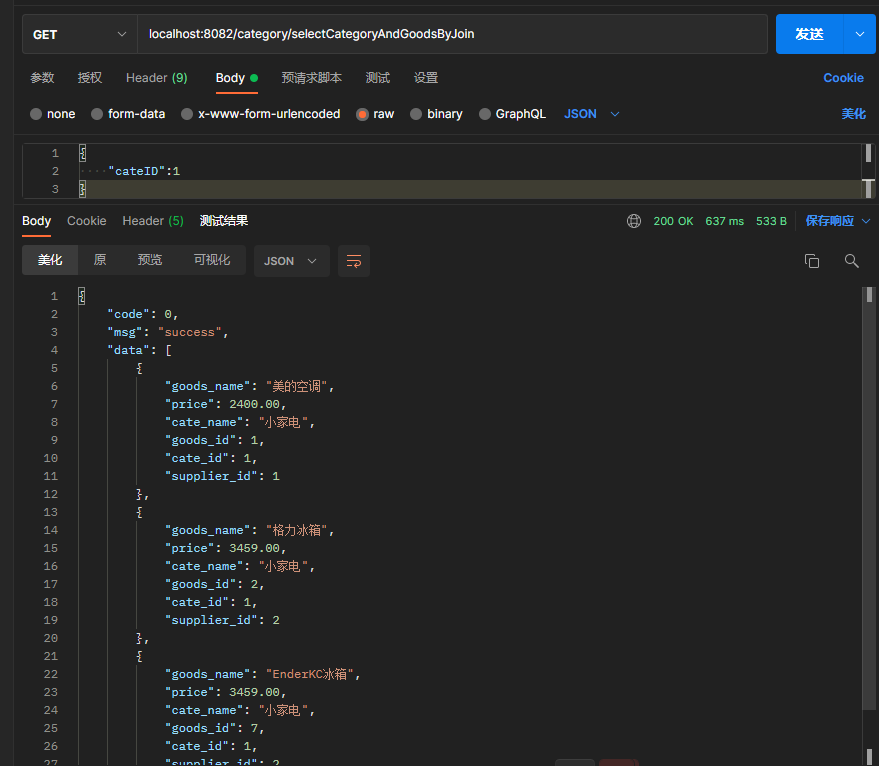
*// 多表连接方式查询*Category selectCategoryAndGoodsByJoin(*int* cateID);

Xml文件：

*<!-- 左连接方式查询-->* <select id="selectCategoryAndGoodsByJoin" resultType="Map">  
 select \*  
 from `mybatisplus-4`.category c  
 join `mybatisplus-4`.goods g on c.cate\_id = g.cate\_id  
 where c.cate\_id = #{cateId}  
 </select>

运行截图：





**三、心得体会：**

1. MybatisPlus的版本和SpringBoot的最新版本可能有时候不兼容
2. 如果SQL查询的字段在pojo中没有相关对象，可以直接返回一个List<Map<>>的类型的东西，MAP的key里面包含查询到的所有字段