5-4 | 一起动手实现用户标签系统 - 底层标签读写组件的实现

建立用户标签单表的 SQL：

```
Java
CREATE TABLE `t_user_tag` (
  `user_id` bigint NOT NULL DEFAULT -1 COMMENT '用户 id',
  `tag_info_01` bigint NOT NULL DEFAULT '0' COMMENT '标签记录字段',
  `tag_info_02` bigint NOT NULL DEFAULT '0' COMMENT '标签记录字段',
  `tag_info_03` bigint NOT NULL DEFAULT '0' COMMENT '标签记录字段',
  `create_time` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '创建时
间',
  `update_time` datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_bin
COMMENT='用户标签记录';
```

建立 100 张用户标签分表的存储过程代码：

```
Java



DELIMITER ;;
CREATE DEFINER=`root`@`%` PROCEDURE `create_t_user_tag_100`()
BEGIN

         DECLARE i INT;
         DECLARE table_name VARCHAR(100);
         DECLARE sql_text VARCHAR(3000);
         DECLARE table_body VARCHAR(2000);
         SET i=0;
         SET sql_text='';
         SET table_body='(
   user_id bigint NOT NULL DEFAULT -1 COMMENT \'用户 id\',
   tag_info_01 bigint NOT NULL DEFAULT 0 COMMENT \'标签记录字段\',
   tag_info_02 bigint NOT NULL DEFAULT 0 COMMENT \'标签记录字段\',
   tag_info_03 bigint NOT NULL DEFAULT 0 COMMENT \'标签记录字段\',
   create_time datetime DEFAULT CURRENT_TIMESTAMP COMMENT \'创建时
间\',
   update_time datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT \'更新时间\',
  PRIMARY KEY (user_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_bin
COMMENT=\'用户标签记录\';';


            WHILE i<100 DO
                IF i<10 THEN
                    SET table_name = CONCAT('t_user_tag_0',i);
                ELSE
                    SET table_name = CONCAT('t_user_tag_',i);
                END IF;

                SET sql_text=CONCAT('CREATE TABLE ',table_name,
table_body);
            SELECT sql_text;
            SET @sql_text=sql_text;
            PREPARE stmt FROM @sql_text;
            EXECUTE stmt;
            DEALLOCATE PREPARE stmt;
            SET i=i+1;
        END WHILE;

    END;;
DELIMITER ;
```

实现用户标签的功能

```
Java
package org.qiyu.live.user.interfaces;


import org.qiyu.live.user.constants.UserTagsEnum;

/**
 * @Author idea
 * @Date: Created in 15:42 2023/4/16
 * @Description 用户标签 RPC 服务
 */
public interface IUserTagRpc {
    /**
     * 设置标签
     *
     * @param userId
     * @param userTagsEnum
     * @return
     */
    boolean setTag(Long userId, UserTagsEnum userTagsEnum);

    /**
     * 取消标签
     *
     * @param userId
     * @param userTagsEnum
     * @return
     */
    boolean cancelTag(Long userId,UserTagsEnum userTagsEnum);

    /**
     * 是否包含某个标签
     *
     * @param userId
     * @param userTagsEnum
     * @return
     */
    boolean containTag(Long userId,UserTagsEnum userTagsEnum);
}
```

Rpc 实现层：

```
Java
package org.qiyu.live.user.provider.rpc;

import jakarta.annotation.Resource;
import org.apache.dubbo.config.annotation.DubboService;
import org.qiyu.live.user.constants.UserTagsEnum;
import org.qiyu.live.user.interfaces.IUserTagRpc;
import org.qiyu.live.user.provider.service.IUserTagService;

/**
 * @Author idea
 * @Date: Created in 13:31 2023/5/26
 * @Description 用户标签 RPC 服务实现类
 */
@DubboService
public class UserTagRpcImpl implements IUserTagRpc {

    @Resource
    private IUserTagService userTagService;

    @Override
    public boolean setTag(Long userId, UserTagsEnum userTagsEnum)
{
        return userTagService.setTag(userId, userTagsEnum);
    }

    @Override
    public boolean cancelTag(Long userId, UserTagsEnum
userTagsEnum) {
        return userTagService.cancelTag(userId, userTagsEnum);
    }

    @Override
    public boolean containTag(Long userId, UserTagsEnum
userTagsEnum) {
        return userTagService.containTag(userId, userTagsEnum);
    }
}
```

service 层的接口定义：

```Java
package org.qiyu.live.user.provider.service;

import org.qiyu.live.user.constants.UserTagsEnum;

/**
 * @Author idea
 * @Date: Created in 17:12 2023/5/27
 * @Description
 */
public interface IUserTagService {

    /**
     * 设置标签 只能设置成功一次
     *
     * @param userId
     * @param userTagsEnum
     * @return
     */
    boolean setTag(Long userId, UserTagsEnum userTagsEnum);

    /**
     * 取消标签
     *
     * @param userId
     * @param userTagsEnum
     * @return
     */
    boolean cancelTag(Long userId,UserTagsEnum userTagsEnum);

    /**
     * 是否包含某个标签
     *
     * @param userId
     * @param userTagsEnum
     * @return
     */
    boolean containTag(Long userId,UserTagsEnum userTagsEnum);
}
```

service 层的实现代码：

```Java
package org.qiyu.live.user.provider.service.impl;

import jakarta.annotation.Resource;
import org.qiyu.live.user.constants.UserTagFieldNameConstants;
import org.qiyu.live.user.constants.UserTagsEnum;
import org.qiyu.live.user.provider.dao.mapper.IUserTagMapper;
import org.qiyu.live.user.provider.dao.po.UserTagPO;
import org.qiyu.live.user.provider.service.IUserTagService;
import org.qiyu.live.user.utils.TagInfoUtils;
import org.springframework.stereotype.Service;

/**
 * @Author idea
 * @Date: Created in 17:13 2023/5/27
 * @Description
 */
@Service
public class UserTagServiceImpl implements IUserTagService {

    @Resource
    private IUserTagMapper userTagMapper;

    @Override
    public boolean setTag(Long userId, UserTagsEnum userTagsEnum)
{
        return userTagMapper.setTag(userId,
userTagsEnum.getFieldName(), userTagsEnum.getTag()) > 0;
    }

    @Override
    public boolean cancelTag(Long userId, UserTagsEnum
userTagsEnum) {
        return userTagMapper.cancelTag(userId,
userTagsEnum.getFieldName(), userTagsEnum.getTag()) > 0;
    }

    @Override
    public boolean containTag(Long userId, UserTagsEnum
userTagsEnum) {
        UserTagPO userTagPO = userTagMapper.selectById(userId);
        if (userTagPO == null) {
            return false;
        }
        String queryFieldName = userTagsEnum.getFieldName();
        if
(UserTagFieldNameConstants.TAG_INFO_01.equals(queryFieldName)) {
            return
TagInfoUtils.isContain(userTagPO.getTagInfo01(),
userTagsEnum.getTag());
        } else if
(UserTagFieldNameConstants.TAG_INFO_02.equals(queryFieldName)) {
            return
TagInfoUtils.isContain(userTagPO.getTagInfo02(),
userTagsEnum.getTag());
        } else if
(UserTagFieldNameConstants.TAG_INFO_03.equals(queryFieldName)) {
            return
TagInfoUtils.isContain(userTagPO.getTagInfo03(),
userTagsEnum.getTag());
```

```
        }
        return false;
    }
}
```

最后是我们的 mapper：

```Java
package org.qiyu.live.user.provider.dao.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Update;
import org.qiyu.live.user.provider.dao.po.UserTagPO;

/**
 * @Author idea
 * @Date: Created in 17:13 2023/5/27
 * @Description
 */
@Mapper
public interface IUserTagMapper extends BaseMapper<UserTagPO> {

    @Update("update t_user_tag set ${fieldName}=${fieldName} |
#{tag} where user_id=#{userId}")
    int setTag(Long userId, String fieldName, long tag);

    @Update("update t_user_tag set ${fieldName}=${fieldName} &~
#{tag} where user_id=#{userId}")
    int cancelTag(Long userId, String fieldName, long tag);
}
```