**5-7｜用户标签接口的优化以及初始化问题**

**Dao层的调整**

将原先的mapper接口做些小优化，只能允许第一次写操作成功。

|  |
| --- |
| 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> {   /\*\*  \* 使用或的思路来设置标签，只能允许第一次设置成功  \*  \* @param userId  \* @param fieldName  \* @param tag  \* @return  \*/  @Update("update t\_user\_tag set ${fieldName}=${fieldName} | #{tag} where user\_id=#{userId} and ${fieldName} & #{tag}=0")  int setTag(Long userId, String fieldName, long tag);   /\*\*  \* 使用先取反在与的思路来取消标签，只能允许第一次删除成功  \*  \* @param userId  \* @param fieldName  \* @param tag  \* @return  \*/  @Update("update t\_user\_tag set ${fieldName}=${fieldName} &~ #{tag} where user\_id=#{userId} and ${fieldName} & #{tag}=#{tag}")  int cancelTag(Long userId, String fieldName, long tag); } |

**设置标签接口调整**

|  |
| --- |
| Java @Resource private RedisTemplate<String, String> redisTemplate; @Resource private UserProviderCacheKeyBuilder cacheKeyBuilder;  @Override public boolean setTag(Long userId, UserTagsEnum userTagsEnum) {  boolean updateSuccess = userTagMapper.setTag(userId, userTagsEnum.getFieldName(), userTagsEnum.getTag()) > 0;  if (updateSuccess) {  return true;  }  UserTagPO userTagPO = userTagMapper.selectById(userId);  if (userTagPO != null) {  return false;  }  String result = redisTemplate.execute(new RedisCallback<String>() {  @Override  public String doInRedis(RedisConnection connection) throws DataAccessException {  String key = cacheKeyBuilder.buildTagLockKey(userId);  redisTemplate.getValueSerializer();  redisTemplate.getKeySerializer();  RedisSerializer valueSerializer = redisTemplate.getValueSerializer();  RedisSerializer keySerializer = redisTemplate.getKeySerializer();  String setResult = (String) connection.execute("set",  keySerializer.serialize(key),  valueSerializer.serialize(-1)  , "NX".getBytes(StandardCharsets.UTF\_8), "EX".getBytes(StandardCharsets.UTF\_8)  , "3".getBytes());  return setResult;  }  });  if ("OK".equals(result)) {  UserTagPO newUserTagPO = new UserTagPO();  newUserTagPO.setUserId(userId);  userTagMapper.insert(newUserTagPO);  return userTagMapper.setTag(userId, userTagsEnum.getFieldName(), userTagsEnum.getTag()) > 0;  }  return false; } |