链滴

用户点赞收藏浏览等记录的统一解决方案

作者：loogn

原文链接：https://ld246.com/article/1567048400651

来源网站：链滴

项目中，我们经常会处理用户对某个对象（如帖子、文章、动态、另一个用户等等）有种种的操作，如点赞、收藏、浏览、关注等等其他标记，我称之为用户对对象的轨迹，其实这个模型基本是一样的所以我们可以共用一个数据表，一套处理逻辑来解决这些问题。

数据模型

```
/// <summary>
/// 轨迹记录
/// </summary>
public class TraceRecord
{
    [OrmLiteField(IsPrimaryKey = true,InsertIgnore = true)]
    public long Id { get; set; }

    /// <summary>
    /// 用户编号
    /// </summary>
    public long UserId { get; set; }

    /// <summary>
    /// 对象编号
    /// </summary>
    public long ObjectId { get; set; }

    /// <summary>
    /// 对象类型
    /// </summary>
    public int ObjectType { get; set; }

    /// <summary>
    /// 轨迹类型 1-查看，2-收藏，3-点赞，.....
    /// </summary>
    public int Type { get; set; }

    /// <summary>
    /// 添加时间
    /// </summary>
    public DateTime AddTime { get; set; }
}
```

其他的字段比较好理解，ObjectType可以说一下，比如有这种情况，有一个供求信息表，本身就有个类型（供应信息，需求信息）这个使用就可能使用到ObjectType了。

数据访问层方法

```
public class TraceRecordDao:BaseDao<TraceRecord>
{

    /// <summary>
    ///  获取是否有轨迹
    /// </summary>
    /// <param name="userId"></param>
    /// <param name="objectId"></param>
```

```
        /// <param name="type"></param>
        /// <returns></returns>
        public bool IsTrace(long userId, long objectId, int type)
        {
            var count = CountWhere(DictBuilder
                .Assign("UserId", userId)
                .Assign("ObjectId", objectId)
                .Assign("Type", type));
            return count > 0;
        }

        /// <summary>
        /// 统计object的轨迹
        /// </summary>
        /// <param name="objectId"></param>
        /// <param name="type"></param>
        /// <returns></returns>
        public long CountByObject(long objectId, int type)
        {
            var count = CountWhere(DictBuilder
                .Assign("ObjectId", objectId)
                .Assign("Type", type));
            return count;
        }

        /// <summary>
        /// 统计user某个类型的轨迹
        /// </summary>
        /// <param name="userId"></param>
        /// <param name="type"></param>
        /// <returns></returns>
        public long CountByUser(long userId, int type)
        {
            var count = CountWhere(DictBuilder
                .Assign("UserId", userId)
                .Assign("Type", type));
            return count;
        }
    }
```

这里都是经常使用的方法，比如文章详细中，是否点赞了，就使用IsTrace判断，文章浏览量可以使用ountByObject计算，我关注的人数可以使用CountByUser获取。

服务层

```
public ResultObject TraceToggle(TraceToggleRequest request)
        {
            if (traceRecordDao.IsTrace(request.LoginedUser.Id, request.ObjectId, request.Type))
            {
                var condict = DictBuilder
                    .Assign("UserId", request.LoginedUser.Id)
                    .Assign("ObjectId", request.ObjectId)
                    .Assign("Type", request.Type);
                traceRecordDao.DeleteWhere(condict);
```

```
        }
        else
        {
            traceRecordDao.Insert(new TraceRecord()
            {
                AddTime = DateTime.Now,
                ObjectId = request.ObjectId,
                ObjectType = request.ObjectType,
                Type = request.Type,
                UserId = request.LoginedUser.Id
            });
        }

        return new ResultObject(true);
    }
```

服务层可以提供一个公共方法，就是开关操作，比如点赞、取消点赞，关注、取消关注，收藏、取消藏等等。当然有些操作是不可逆的，比如查看记录。

服务层还有其他的列表查看，就是和具体业务相关了，比如我的收藏、我的点赞等等，需要关联具体表查询了。

结束

上述代码可以当做是伪代码，因为使用的是自己的数据访问和一些模型类，主要是处理逻辑已经表明希望能在实际项目中对你有所帮助！