跟着如下示例：如下根据门店id过滤数据

1. 新建ICurrentShop

public interface ICurrentShop : IScopedDependency

{

    Guid? Id { get; }

}

并注册对应的实现

public class CurrentShop : ICurrentShop

{

    public Guid? Id { get; set; }

    public CurrentShop(ICurrentUser currentUser) {

        string shopId = currentUser?.FindClaim("sp")?.Value;

        if (shopId != null)

        {

            Id = new Guid(shopId);

        }

    }

}

1. 修改DBContext

public class StoreDbContext : AbpDbContext<StoreDbContext>, IStoreDbContext

{

    // 是否禁用数据过滤

    // 注：这是个属性，但他是动态获取的

    protected bool IsShopFilterEnabled => DataFilter?.IsEnabled<IMultiShop>() ?? false;

    // 获取门店Id

    // 注：这是个属性，但他是动态获取的

protected Guid? CurrentShopId => LazyServiceProvider.LazyGetRequiredService<ICurrentShop>()?.Id;

    // 判断是否对实体启用数据过滤，该方法在进程中只会调用一次

    protected override bool ShouldFilterEntity<TEntity>(IMutableEntityType entityType)

    {

        if (typeof(IMultiShop).IsAssignableFrom(typeof(TEntity)))

        {

            return true;

        }

        return base.ShouldFilterEntity<TEntity>(entityType);

}

    // 生成数据过滤，该方法在进程中只会调用一次

    protected override Expression<Func<TEntity, bool>> CreateFilterExpression<TEntity>()

    {

        var expression = base.CreateFilterExpression<TEntity>();

        // 如果实体继承 IMultiShop，则添加过滤表达式

        if (typeof(IMultiShop).IsAssignableFrom(typeof(TEntity)))

        {

            // 生成实体的过滤表达式

            // 注：这里的 IsShopFilterEnabled 和 CurrentShopId 都是 函数型的属性

            Expression<Func<TEntity, bool>> shopFilter = e => !IsShopFilterEnabled || EF.Property<Guid>(e, "ShopId") == CurrentShopId;

            expression = expression == null ? shopFilter : CombineExpressions(expression, shopFilter);

        }

        return expression;

    }

}