**IsoftDelete**

软删除接口

public class Book : AggregateRoot<Guid>, ISoftDelete

{

    public string Name { get; set; }

    public bool IsDeleted { get; set; } //Defined by ISoftDelete

}

当实体继承该接口时，调用仓储的删除方法只会将实体的IsDeleted 设为true，查询时会自动过滤掉软删除的实体

**IMultiTenant**

多租户接口

public class Book : AggregateRoot<Guid>, ISoftDelete, IMultiTenant

{

    public string Name { get; set; }

    public bool IsDeleted { get; set; } //Defined by ISoftDelete

    public Guid? TenantId { get; set; } //Defined by IMultiTenant

}

当实体继承该接口时，仓储会自动过滤掉其他租户的数据

**启用禁用数据过滤**

如果你软删除了实体，但你又想查询它怎么办？如下

我通过IDataFilter来禁用数据过滤

public class MyBookService : ITransientDependency

{

    private readonly IDataFilter \_dataFilter;

    private readonly IRepository<Book, Guid> \_bookRepository;

    public MyBookService(

        IDataFilter dataFilter,

        IRepository<Book, Guid> bookRepository)

    {

        \_dataFilter = dataFilter;

        \_bookRepository = bookRepository;

    }

    public async Task<List<Book>> GetAllBooksIncludingDeletedAsync()

    {

        // 禁用软删数据过滤

        using (\_dataFilter.Disable<ISoftDelete>())

        {

            // 查询数据

            return await \_bookRepository.GetListAsync();

        }

    }

}

**全局禁用数据过滤**

如果你想全局禁用数据过滤，你可以在模块的ConfigureServices方法中进行配置

Configure<AbpDataFilterOptions>(options =>

{

options.DefaultStates[typeof(ISoftDelete)] =

new DataFilterState(isEnabled: false);

});

详细文档：https://docs.abp.io/zh-Hans/abp/latest/Data-Filtering