/// <summary>  
/// 导航属性加载器  
/// </summary>  
public interface IIncludeLoader<TEntity> : ITransientDependency where TEntity : class, IEntity  
{  
 IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, TResult>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, IEnumerable<TResult>>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, ICollection<TResult>>> func) where TResult : class, IEntity;

IEnumerable<TEntity> ToEnumerable();  
}

/// <summary>  
/// 导航属性加载器  
/// </summary>  
public interface IIncludeLoader<TEntity, TProperty> : ITransientDependency where TEntity : class, IEntity where TProperty : class  
{  
 IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, TResult>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, IEnumerable<TResult>>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, ICollection<TResult>>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> ThenIncludeLoad<TResult>(Expression<Func<TProperty, TResult>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> ThenIncludeLoad<TResult>(Expression<Func<TProperty, IEnumerable<TResult>>> func) where TResult : class, IEntity;

IIncludeLoader<TEntity, TResult> ThenIncludeLoad<TResult>(Expression<Func<TProperty, ICollection<TResult>>> func) where TResult : class, IEntity;

IEnumerable<TEntity> ToEnumerable();

}

public class IncludeLoader<TEntity> : EfCoreRepositoryBase<MemberApiDbContext, TEntity, int>, IIncludeLoader<TEntity> where TEntity : class, IEntity  
{  
 private DbSet<TEntity> \_dbSet { get; set; }

public IncludeLoader(  
 IDbContextProvider<MemberApiDbContext> dbContextProvider  
 ):base(dbContextProvider)  
 {  
 }

public IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, TResult>> func) where TResult : class, IEntity  
 {  
 return new IncludeLoader<TEntity, TResult>(Context.Set<TEntity>().Include(func));  
 }

public IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, IEnumerable<TResult>>> func) where TResult : class, IEntity  
 {  
 return new IncludeLoader<TEntity, TResult>(Context.Set<TEntity>().Include(func));  
 }

public IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, ICollection<TResult>>> func) where TResult : class, IEntity  
 {  
 return new IncludeLoader<TEntity, TResult>(Context.Set<TEntity>().Include(func));  
 }

public IEnumerable<TEntity> ToEnumerable()  
 {  
 return Context.Set<TEntity>();  
 }  
}

public class IncludeLoader<TEntity, TProperty> : IIncludeLoader<TEntity, TProperty> where TEntity : class, IEntity where TProperty : class  
{  
 private IIncludableQueryable<TEntity, TProperty> \_includableQueryable { get; set; }

private IIncludableQueryable<TEntity, IEnumerable<TProperty>> \_includableQueryables { get; set; }

public IncludeLoader(IIncludableQueryable<TEntity, TProperty> includableQueryable)  
 {  
 \_includableQueryable = includableQueryable;  
 }

public IncludeLoader(IIncludableQueryable<TEntity, IEnumerable<TProperty>> includableQueryables)  
 {  
 \_includableQueryables = includableQueryables;  
 }

public IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, TResult>> func) where TResult : class, IEntity  
 {  
 if (\_includableQueryable != null)  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryable.Include(func));  
 }  
 else {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryables.Include(func));  
 }  
 }

public IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, IEnumerable<TResult>>> func) where TResult : class, IEntity  
 {  
 if (\_includableQueryable != null)  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryable.Include(func));  
 }  
 else  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryables.Include(func));  
 }  
 }

public IIncludeLoader<TEntity, TResult> IncludeLoad<TResult>(Expression<Func<TEntity, ICollection<TResult>>> func) where TResult : class, IEntity  
 {  
 if (\_includableQueryable != null)  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryable.Include(func));  
 }  
 else  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryables.Include(func));  
 }  
 }

public IIncludeLoader<TEntity, TResult> ThenIncludeLoad<TResult>(Expression<Func<TProperty, TResult>> func) where TResult : class, IEntity  
 {  
 if (\_includableQueryable != null)  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryable.ThenInclude(func));  
 }  
 else  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryables.ThenInclude(func));  
 }  
 }

public IIncludeLoader<TEntity, TResult> ThenIncludeLoad<TResult>(Expression<Func<TProperty, IEnumerable<TResult>>> func) where TResult : class, IEntity  
 {  
 if (\_includableQueryable != null)  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryable.ThenInclude(func));  
 }  
 else  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryables.ThenInclude(func));  
 }  
 }

public IIncludeLoader<TEntity, TResult> ThenIncludeLoad<TResult>(Expression<Func<TProperty, ICollection<TResult>>> func) where TResult : class, IEntity  
 {  
 if (\_includableQueryable != null)  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryable.ThenInclude(func));  
 }  
 else  
 {  
 return new IncludeLoader<TEntity, TResult>(\_includableQueryables.ThenInclude(func));  
 }  
 }

public IEnumerable<TEntity> ToEnumerable()  
 {  
 if (\_includableQueryable != null)  
 {  
 return \_includableQueryable;  
 }  
 else  
 {  
 return \_includableQueryables;  
 }  
 }  
}