**预先加载**

预先加载：在查询实体前，将其关联的实体一并查询回来

使用Include和ThenInclude

using (var context = new BloggingContext())

{

    var blogs = context.Blogs

        .Include(blog => blog.Posts)            // 加载 blog->Posts

            .ThenInclude(post => post.Author)       // 加载 blog->Posts->Author

                .ThenInclude(author => author.Photo)    // 加载 blog->Posts->Author->Photo

        .Include(blog => blog.Owner)

            .ThenInclude(owner => owner.Photo)

        .ToList();

}

**显示加载**

要访问管理实体时，才去加载

using (var context = new BloggingContext())

{

    var blog = context.Blogs

        .Single(b => b.BlogId == 1);

    context.Entry(blog)

        .Collection(b => b.Posts)   // 加载集合

        .Load();

    context.Entry(blog)

        .Reference(b => b.Owner)    // 加载实体

        .Load();

    context.Entry(blog)

        .Collection(b => b.Posts)

        .Query()                    // 使用linq进行查询

        .Where(p => p.Rating > 3)   // 筛选

        .ToList();

}

**延迟加载**

延迟加载需要安装 Microsoft.EntityFrameworkCore.Proxies 包

延迟加载：我们访问该属性时，自动加载

startup中配置：

.AddDbContext<BloggingContext>(

    b => b.UseLazyLoadingProxies()

          .UseSqlServer(myConnectionString));

需要延迟加载的属性必须是virtual且可被继承

public class Blog

{

    public int Id { get; set; }

    public string Name { get; set; }

    // Posts 必须是 virtual 且可被继承

    public virtual ICollection<Post> Posts { get; set; }

}

public class Post

{

    public int Id { get; set; }

    public string Title { get; set; }

public string Content { get; set; }

    // Blog 必须是 virtual 且可被继承

    public virtual Blog Blog { get; set; }

}