DDD领域驱动设计初探（5）：AutoMapper使用 - 文章 - 伯乐在线

前言：前篇搭建了下WCF的代码，就提到了DTO的概念，对于为什么要有这么一个DTO的对象，上章可能对于这点不太详尽，在此不厌其烦再来提提它的作用：

- 从安全上面考虑，领域Model都带有领域业务，让Client端引用Domain Model就意味着Client端可以绕过应用层直接完成业务逻辑的调用，这样是一种不安全的机制。
- 从对象传递效率上面考虑，领域Model带有业务，而这些业务一般对于UI层是没有意义的，所以带有业务的model传递起来会加重网络负担。
- 网上还说了DTOmodel最大的意义在于跨平台，Domain Model都是与特定的语言的数据类型有关，而这些数据类型是不能跨平台的，比如Java的类型就不能被C#使用。但在分布式模式下，Client端与Server端的平台不同是很正常的，如果Service直接返回Domain Model，Client端根本无法解析，这就要求Service返回的结果必须是标准的格式字节流。让Domain Model只使用简单类型（字符和数值）？让数据类型约束Domain Model显然不是一个好想法，所以DTO似乎是必不可少的了。

既然我们要使用DTO，那么有一件事我们就非做不可了，我们从领域层得到的是领域Model，如何把领域Model转换成只带有数据属性的DTO传递到前台呢？又或者我们从前台提交一个DTO对象，如何将DTO转换成领域Model而提交到后台呢？这个时候就需要我们的对象映射工具，目前市面上对象映射工具较多，但博主最熟悉的还是Automapper，这章就来分享下Automapper的使用。

一、AutoMapper

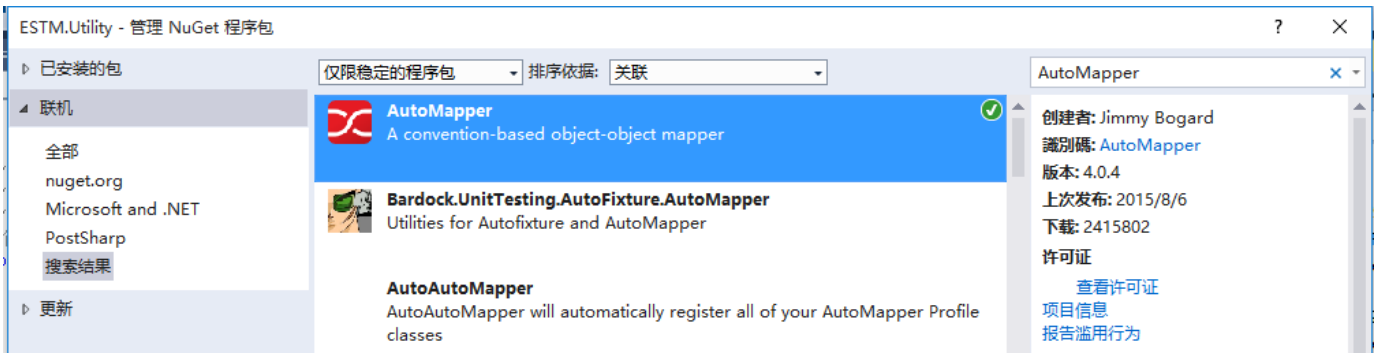
Automapper是一个object-object mapping（对象映射）工具，一般主要用于两个对象之间数据映射和交换。当然你也可以自己通过反射去写对象的映射，对于简单的两个属性间的数据转换，肯定没什么问题。但是如果遇到某些复杂的数据转换，比如指定某一个对象的某个属性映射到另一个对象的某一个属性，这种情况如果我们自己手动映射，恐怕就有点麻烦了吧。既然我们有现成的工具，为什么不用呢？

二、AutoMapper引用到项目中

向项目中添加AutoMapper的引用有两种方式：

1、Nuget方式

在需要使用AutoMapper的项目文件上面右键→管理Nuget程序包，打开Nuget界面，搜索Automapper，然后安装第一个即可。如下图：



2、程序包管理控制台方式

点击Visual Studio的工具菜单→程序包管理控制台，然后选择需要安装Automapper的项目（下图中的默认项目），最后在控制台里面输入命令“Install-Package AutoMapper”命令即可按照Automapper包：

三、AutoMapper使用代码示例

1、最简单的对象映射

AutoMapper使用起来还是比较简单的，最简单的用法你只需要两句话：

C#

```
        ///
        /// AutoMapper帮助类
        ///
        public static class AutoMapperHelper
        {
                ///
                ///     单个对象映射
                ///
                public static T MapTo(this object obj)
                {
                        if (obj == null) return default(T);
                        Mapper.CreateMap(obj.GetType(), typeof(T));
                        return Mapper.Map(obj);
                }
```

```
        ///
        /// 集合列表类型映射
        ///
        public static List MapToList(this IEnumerable source)
        {
                Mapper.CreateMap();
                return Mapper.Map>(source);
        }
    }
```

当然，这是最简单的用法，稍微复杂点的用法我们在后面慢慢介绍。

2、指定字段的对象映射

前面说了，对于指定某一个对象的某个属性映射到另一个对象的某一个属性，这种场景，我们先来看看下面代码：

C#

这一句就帮我们搞定。

3、传递lamada的表达式映射

还记得我们在仓储里面封装了传递lamada表达式的查询方法么？试想，如果我们在Web层里面也希望传递lamada表达式去后台查询，那么这个时候就有点问题了，因为我们Web里面只能访问DTO的Model，所以只能传入DTO Model的lamada，而我们仓储里面需要传入的是领域Model的lamada，那么问题就来了，这两个lamada表达式之间必须存在一个转换关系，试想，这些东西如果让我们手动去处理，还是有难度的吧！还好，我们神奇的Automapper替我们想到了。它能够帮我们将DTO的lamada转换成领域Model的lamada，来看看代码吧：

C#

```
    [Import]
    public IUserRepository userRepository { get; set; }
    public virtual IList Find(Expressionbool>> selector)
        {
            //得到从Web传过来和DTOModel相关的lamaba表达式的委托
            Funcbool> match = selector.Compile();
            //创建映射Expression的委托
            Func mapper =
AutoMapper.QueryableExtensions.Extensions.CreateMapExpression(Mapper.Engine).Compile();
            //得到领域Model相关的lamada
            Expressionbool>> lamada = ef_t => match(mapper(ef_t));
            List list = userRepository.Find(lamada).ToList();
            return Mapper.Map, List>(list);
        }
```

上面方法完美实现了两种lamada之间的转换，但根据博主的使用经历，这种转换对属性的类型有很严格

的要求，必须保证领域model和DTO的Model同一个属性的类型完全相同，否则容易报异常。使用的时候需要注意。实际使用的方法：

```
C#
public class PowerManageWCFService :BaseService, IPowerManageWCFService
        {
                #region Fields
                [Import]
                private IUserRepository userRepository { get; set; }
                [Import]
                private IDepartmentRepository departmentRepository { get; set; }
                [Import]
                private IRoleRepository roleRepository { get; set; }
                [Import]
                private IMenuRepository menuRepository { get; set; }
                #endregion
                #region Constust
                public PowerManageWCFService()
                {
                        //注册MEF
                        Regisgter.regisgter().ComposeParts(this);
                }
                #endregion
                #region WCF服务接口实现
                public List GetUsers(Expressionbool>> selector)
                {
                        return base.GetDtoByLamada(userRepository, selector);
                }
                public List GetDepartments(Expressionbool>> selector)
                {
                        return base.GetDtoByLamada(departmentRepository, selector);
                }
                public List GetRoles(Expressionbool>> selector)
                {
                        return base.GetDtoByLamada(roleRepository, selector);
                }
                public List GetMenus(Expressionbool>> selector)
                {
                        return base.GetDtoByLamada(menuRepository, selector);
                }
                #endregion
}
```

4、Automapper的其他应用

除了上面介绍的Automapper的几个简单使用，其他还有其他的一些用法。

网上很多介绍DataReader对象和实体类之间的映射：

C#