Data Transfer Objects（DTO）用来在应用层和展现层之间传输数据。

**领域层的抽象**

dto提供一个有效的方法从展现层抽象领域对象，因此，你的层正确分离开，即使你想完全地改变展现层，也可以继续使用已存在的应用层和领域层。相反，你可以重写你的领域层、完全改变数据库结构、实体和ORM框架，只要你的应用服务契约（方法签名和DTO）保持不变，展现层也不用做任何修改。

**DTO 约定和验证**

让我们看一个完整的示例，假设我们想开发一个通过name搜索people并返回一个people列表的应用服务，这样，我们应该有一个Person[实体](http://www.cnblogs.com/kid1412/p/5992443.html" \t "http://www.cnblogs.com/kid1412/p/_blank)，如：

public class Person : Entity

{

public virtual string Name { get; set; }

public virtual string EmailAddress { get; set; }

public virtual string Password { get; set; }

}

接着为我们的[应用服务](http://www.cnblogs.com/kid1412/p/6003490.html" \t "http://www.cnblogs.com/kid1412/p/_blank)定义一个接口：

public interface IPersonAppService : IApplicationService

{

SearchPeopleOutput SearchPeople(SearchPeopleInput input);

}

ABP建议命名输入/输出参数为：MethodName**Input**和MethodName**Output**，并为每个应用服务方法定义单独的输入和输入DTO。即使你的方法只接受/返回一个参数

让我们看一下这个例子的输入和输出DTO类：

public class SearchPeopleInput

{

[StringLength(40, MinimumLength = 1)]

public string SearchedName { get; set; }

}

public class SearchPeopleOutput

{

public List<PersonDto> People { get; set; }

}

public class PersonDto : EntityDto

{

public string Name { get; set; }

public string EmailAddress { get; set; }

}

**DTO和实体间自动映射**

幸运地是：有工具使这件事变得容易，[AutoMapper](http://automapper.org/" \t "http://www.cnblogs.com/kid1412/p/_blank)是其中之一，它发布在nuget上，你可以很容易地把它加入到你的项目里。让我们使用AutoMap再写一下SearchPeople方法：

public SearchPeopleOutput SearchPeople(SearchPeopleInput input)

{

var peopleEntityList = \_personRepository.GetAllList(person => person.Name.Contains(input.SearchedName));

// 自动映射

return new SearchPeopleOutput { People = Mapper.Map<List<PersonDto>>(peopleEntityList) };

}

这样就完事了。你可以添加更多的属性到实体和DTO里

Mapper.CreateMap<Person, PersonDto>();

**使用特性和扩展方法进行映射**

ABP提供了多个特性和扩展方法用来定义映射，为使用它，先在你的项目里添加[Abp.AutoMapper](http://www.nuget.org/packages/Abp.AutoMapper" \t "http://www.cnblogs.com/kid1412/p/_blank)的nuget包，然后使用AutoMap特性进行双向映射，AutoMapFrom和AutoMapTo进行单向映射。使用MapTo扩展方法映射一个对象到另一个。映射定义示例：

[AutoMap(typeof(MyClass2))] //定义双向映射

public class MyClass1

{

public string TestProp { get; set; }

}

public class MyClass2

{

public string TestProp { get; set; }

}

然后你可以使用MapTo扩展方法来映射它们：

var obj1 = new MyClass1 { TestProp = "Test value" };

var obj2 = obj1.MapTo<MyClass2>(); //创建一个新的MyClass2对象，从obj1拷贝TestProp