[**.NET平台开源项目速览(14)最快的对象映射组件Tiny Mapper**](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html)

**阅读目录**

* [1.Tiny Mapper基本介绍](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_label0)
* [2.Tiny Mapper 基本使用](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_label1)
* [3.Tiny Mapper 指定配置使用](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_label2)
* [4.Tiny Mapper复杂类型使用](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_label3)
* [5.资源](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_label4)

    好久没有写文章，工作甚忙，但每日还是关注.NET领域的开源项目。五一休息，放松了一下之后，今天就给大家介绍一个轻量级的对象映射工具Tiny Mapper：号称是.NET平台最快的对象映射组件。那就一起看看呢。

.NET开源目录：[【目录】本博客其他.NET开源项目文章目录](http://www.cnblogs.com/asxinyu/p/Bolg_Category_For_DotNet_OpenSource.html)

本文原文地址：<http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html>

临时更新：感谢@ [郭明锋](http://www.cnblogs.com/guomingfeng/)的意见，为了避免新手误解，这里说明一下，Tiny Mapper的映射关系在Bind的时候，其实是将映射关系保存在一个静态的私有字典中，大家在使用的时候可以不需要每次都配置或者Bind。方法比较灵活，例如：你可以在实体类中封装一个方法，在方法内部进行映射关系的配置；你还可以将系统所有的配置映射关系放在一起，在使用前初始化一次就可以了。

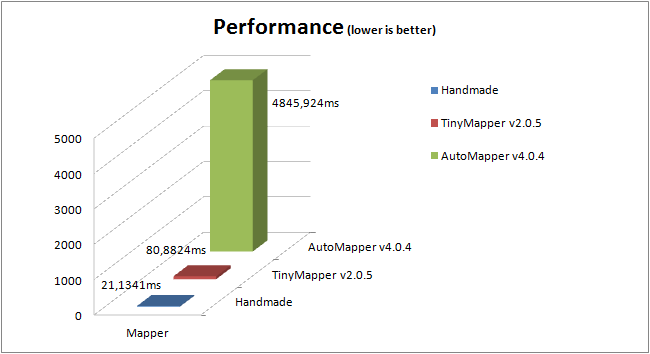
[回到目录](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_labelTop)

**1.Tiny Mapper基本介绍**

    Tiny Mapper是一个.net平台开源的对象映射组件。说到.NET平台的对象映射，大名鼎鼎的可能就是AutoMapper。那先看看Tiny Mapper的特点

**1.1 难以置信的快速**

    这是Tiny Mapper标榜最大的特点之一：快。那究竟有多快，我们看看官方的测试截图，由于时间紧张，我没有进行测试对比，有空试试。



**1.2 非常容易使用**

    使用TinyMapper非常简单，它的设计简单，使用NuGet包，可以快速安装使用，DLL也才50K大小不到，所以非常轻量级。

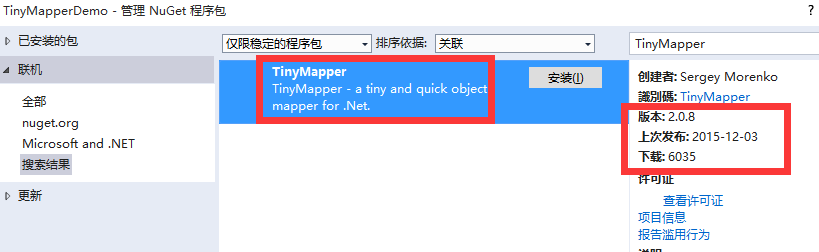
**1.3 开源**

    TinyMapper是一个开源项目，所以你可以参与更新和自己修改代码。

[回到目录](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_labelTop)

**2.Tiny Mapper 基本使用**

    首先，通过NuGet下载安装Tiny Mapper，如下图：



    然后添加引用：

|  |  |
| --- | --- |
| 1  2 | using Nelibur.ObjectMapper;  using Nelibur.ObjectMapper.Bindings; |

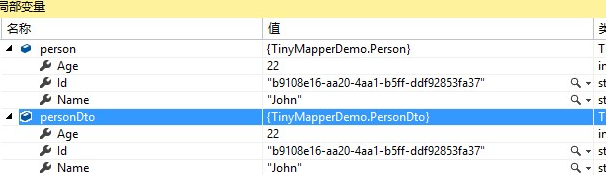
    接下来，我们准备2个要相互映射的类型，Person和PersonDto，结构分别如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public class Person  {      public String Id { get; set; }      public String Name { get; set; }      public Int32 Age { get; set; }  }  public class PersonDto  {      public String Id { get; set; }      public String Name { get; set; }      public Int32 Age { get; set; }  } |

    那我们在代码例如如何把其中一个类Person的对象映射为PersonDto呢？看看Tiny Mapper的使用，很简单就2行：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | /// <summary>最简单的2个类型之间的映射</summary>  static void Test1()  {      TinyMapper.Bind<Person,PersonDto>();      //实例化一个Person对象      var person = new Person      {          Id = Guid.NewGuid().ToString(),          Name = "John",          Age = 22      };      //映射      var personDto = TinyMapper.Map<PersonDto>(person);  } |

    运行看看结果：



[回到目录](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_labelTop)

**3.Tiny Mapper 指定配置使用**

有时候对象的字段名称并不一样，而且可能要忽略某些字段，这个时候就要使用更加灵活的配置了。

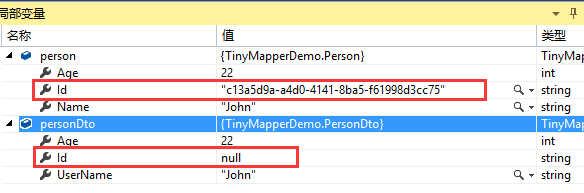
看例子，为了演示，我们特意修改2个类型的字段名称不一样.如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class Person  {      public String Id { get; set; }      public String Name { get; set; }      public Int32 Age { get; set; }    }  public class PersonDto  {      public String Id { get; set; }      //注意这里的字段名称:UserName      public String UserName { get; set; }      public Int32 Age { get; set; }  } |

    映射代码和配置如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | /// <summary>使用配置项指定和忽略字段</summary>  static void Test2()  {      TinyMapper.Bind<Person, PersonDto>(config =>      {          config.Ignore(x => x.Id);//忽略ID字段          config.Bind(x => x.Name, y => y.UserName);//将源类型和目标类型的字段对应绑定起来          config.Bind(x => x.Age, y => y.Age);//将源类型和目标类型的字段对应绑定起来      });      var person = new Person      {          Id = Guid.NewGuid().ToString(),          Name = "John",          Age = 22      };      var personDto = TinyMapper.Map<PersonDto>(person);  } |

    看看效果如何？我们指定的Id字段忽略映射，以及将源对象的字段映射到目标对象的指定字段了:



[回到目录](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_labelTop)

**4.Tiny Mapper复杂类型使用**

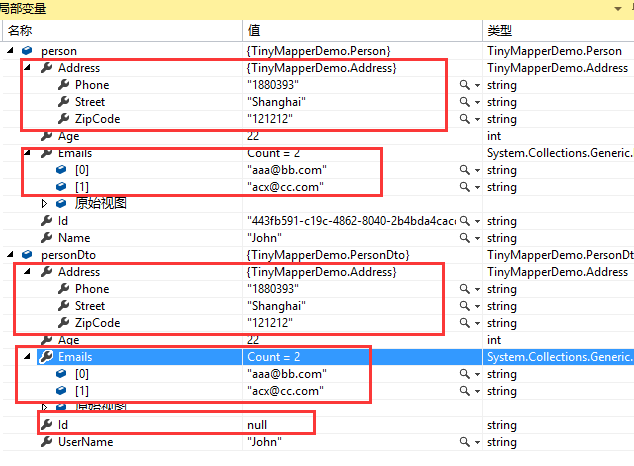
    有时候对象的类型里面嵌套着对象，或者数组等等，其实这个时候的使用和上述完全一样，这里只是举个例子，大家看看而已：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public class Person  {      public String Id { get; set; }      public String Name { get; set; }      public Int32 Age { get; set; }      public Address Address { get; set; }      public List<String> Emails { get; set; }  }  public class PersonDto  {      public String Id { get; set; }      public String UserName { get; set; }      public Int32 Age { get; set; }      public Address Address { get; set; }      public List<String> Emails { get; set; }  }  public sealed class Address  {      public string Phone { get; set; }      public string Street { get; set; }      public string ZipCode { get; set; }  } |

    映射的代码和第3节类似：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | /// <summary>更复杂对象的映射例子</summary>  static void Test3()  {      TinyMapper.Bind<Person, PersonDto>(config =>      {          config.Ignore(x => x.Id);//忽略ID字段            //将源类型和目标类型的字段对应绑定起来          config.Bind(x => x.Name, y => y.UserName);          config.Bind(x => x.Age, y => y.Age);          config.Bind(x => x.Address, y => y.Address);          config.Bind(x => x.Emails, y => y.Emails);      });      var person = new Person      {          Id = Guid.NewGuid().ToString(),          Name = "John",          Age = 22,          Address = new Address() { Phone = "1880393", Street = "Shanghai", ZipCode = "121212" },          Emails = new List<string>() { "aaa@bb.com", "acx@cc.com" }      };      var personDto = TinyMapper.Map<PersonDto>(person);  } |

效果如下：



[回到目录](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_TinyMapper.html#_labelTop)

**5.资源**

    Tiny Mapper官方网站：<http://tinymapper.net/>

    github项目地址：<https://github.com/TinyMapper/TinyMapper>

    本文使用的代码下载：[TinyMapperDemo.rar](http://files.cnblogs.com/files/asxinyu/TinyMapperDemo.rar)