**阅读目录**

* [1.SharpConfig简介](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html#_label0)
* [2.SharpConfig基本使用](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html#_label1)
* [3.特殊数据类型的配置项目](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html#_label2)
* [4.动态创建配置文件与使用](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html#_label3)
* [5.更简单的应用-对象映射](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html#_label4)
* [6.资源](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html#_label5)

　　在.NET平台日常开发中，读取配置文件是一个很常见的需求。以前都是使用System.Configuration.ConfigurationSettings来操作，这个说实话，搞起来比较费劲。不知道大家有没有同感。所以更多时候我还是喜欢使用开源的东西，更加方便简洁，也稳定。省去自己的麻烦。今天就介绍一个非常精致的.NET平台开源的操作配置文件(cfg/ini)的组件SharpConfig。走过路过，千万不要错过！上周我在[这些.NET开源项目你知道吗？让.NET开源来得更加猛烈些吧](http://www.cnblogs.com/asxinyu/p/dotnet_opensource_project.html)一文中也介绍过项目。我将会重点对部分开源项目进行深入的研究，形成**“.NET平台开源项目速览”**系列文章，该系列文章没有时间限制，会一直持续更新下去。

.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：[.NET平台开源项目速览(1)SharpConfig配置文件读写组件](http://www.cnblogs.com/asxinyu/p/dotnet_Opensource_project_SharpConfig.html)

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

**1.SharpConfig简介**

    SharpConfig是.NET平台一个使用非常简单的处理Cfg/ini配置文件的类库。

    项目主页：<https://github.com/cemdervis/SharpConfig>，目前最新版本是1.4.3；

    Nuget包主页：<https://www.nuget.org/packages/sharpconfig/>，其主要功能有：

    1.支持的配置文件类型有cfg,ini格式；

    2.支持注释功能，使用 # 符号开头；

    3.支持直接读取和写入指定节点的值；

    4.支持枚举，数组，以及和实体的直接映射；

    5.支持内存的创建和使用，相对一个配置类型，直接在代码中初始化和使用；

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

**2.SharpConfig基本使用**

    下面通过一个简单的Demo来大概知道如何使用。下载后，解压，新建项目，添加dll引用，需要添加命名空间:

|  |  |
| --- | --- |
| 1 | using SharpConfig; |

    我们添加一个如下格式的文件，文件名称为：example.ini，这个文件可以手动创建放在项目中，设置属性：“始终复制”即可，或者自己手动复制到bin目录也行。方便调用，放在其他路径也可以，最终都是fileName的路径来找文件的。example.ini的内容如下，这个例子也是项目主页的例子，我稍微修改了下：

[General] #这代表Section，也就是节点

SomeString = Hello SharpConfig!

SomeInteger = 10 #这里也是注释

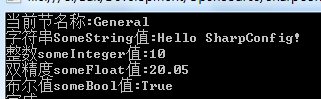
SomeFloat = 20.05

ABoolean = true

    下面使用代码直接读取这个配置文件的值：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | static void Test1()  {      //按文件名称加载配置文件      Configuration config = Configuration.LoadFromFile("example.ini");      //按照节的名称读取节      Section section = config["General"];      //依次根据每个配置项的名称来读取，如果配置文件类型搞错了，会报错      string someString = section["SomeString"].Value;      var someInteger = section["SomeInteger"].GetValue<Boolean>();      float someFloat = section["SomeFloat"].GetValue<float>();      Boolean someBool = section["ABoolean"].GetValue<Boolean>();      Console.WriteLine("当前节名称:{0}",section.Name );      Console.WriteLine("字符串SomeString值:{0}", someString);      Console.WriteLine("整数someInteger值:{0}", someInteger);      Console.WriteLine("双精度someFloat值:{0}", someFloat);      Console.WriteLine("布尔值someBool值:{0}", someBool);  } |

结果如下：

保存配置文件有以下一些方法：

|  |  |
| --- | --- |
| 1  2  3  4 | myConfig.Save( "myConfig.cfg" ); // 保存到cfg格式的文本文件  myConfig.Save( myStream ); //保存到文本流中  myConfig.SaveBinary( "myConfig.cfg" ); //保存到cfg格式的二进制文件  myConfig.SaveBinary( myStream ); // 保存到二进制文件 |

    是不是很简单，速度也很快。下面看看其他功能的使用。上面只是一个最基本的使用例子。

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

**3.特殊数据类型的配置项目**

**3.1 枚举类型**

    枚举类型的支持其实和上面的值类型差不多，也是直接使用泛型的类型参数类获取枚举，而数组是相对特殊一点的，所以单独拿出来给大家介绍。

    假设有一个Section名称为DateInfo，项目的Day有一个Monday值，是枚举类型DayOfWeek的一个值。如下面所示：

    [DateInfo]

    Day = Monday

    调用的时候，使用一个枚举类型DayOfWeek 来读取：

|  |  |
| --- | --- |
| 1 | DayOfWeek day = config["DateInfo"]["Day"].GetValue<DayOfWeek>(); |

**3.2 数组类型**

    数组类型的支持，首先是统一转换为object数组，然后根据自己的需求要手动转换，其实可以写一个扩展方法，很简单，就不展开了。例如下面有一个节的项目是：

    [General]

    MyArray = {0,2,5,6}

    读取的方法为：

|  |  |
| --- | --- |
| 1 | object[] myArray = config["General"]["MyArray"].GetValue<object[]>(); |

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

**4.动态创建配置文件与使用**

    动态在内存中创建配置文件，其实就是把Configuration当作一个实体类一样，在代码中创建和使用。例如下面的代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | static void Test2()  {      var myConfig = new Configuration();      //节点Video      myConfig["Video"]["Width"].Value = "1920";      myConfig["Video"]["Height"].Value = "1080";      //设置数组      myConfig["Video"]["Formats"].SetValue(new string[] { "RGB32", "RGBA32" });      //可以使用循环获取节点以及节点的所有项目，进行操作      foreach ( var section in myConfig )      {          foreach ( var setting in section )          {              //TODO:          }      }      //也可以直接使用节点和项目的名称来访问：      Console.WriteLine("Width:{0}", myConfig["Video"]["Width"].GetValue<Int32>());      Console.WriteLine("Height:{0}", myConfig["Video"]["Height"].GetValue<Int32>());  } |

　　结果如下：

http://images0.cnblogs.com/blog2015/79603/201505/211117232449225.jpg

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

**5.更简单的应用-对象映射**

　　在文章发表之前，我在“[这些.NET开源项目你知道吗？让.NET开源来得更加猛烈些吧](http://www.cnblogs.com/asxinyu/p/4519513.html)”一文中已经提到该项目，网友回复如果字符串中包括了#字符，可能会被认为是注释的事情，这个的确是一个隐藏bug，不过应该很少会碰到，这个解决方案自己绕过弯吧，真的有，就在程序或者使用实体类映射的时候，单独处理一下。

    这种情况在以前使用XCode组件时用的比较多。不过SharpConfig也支持，挺好用的。相当于把实体类的字段和配置文件对应起来，可以更加方便使用。类型和配置文件映射后，可以直接操作对象了。不需要上面那些步骤。假设有一个Person对象，如下格式：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public class Person  {      public string Name { get; set; }      public int Age { get; set; }      public Gender Gender { get; set; }  } |

Gender是一个性别的枚举类型，如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5 | enum Gender  {      Male,      Female  } |

假设配置文件是这样保存的：

[Person]

Name = Peter

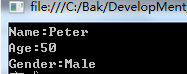
Age = 50

Gender = Male

    在实际使用的时候，使用下面的代码即可：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | static void Test3()  {      Configuration config = Configuration.LoadFromFile("example.ini");      Person person = config["Person"].CreateObject<Person>();      Console.WriteLine("Name:{0}",person.Name);      Console.WriteLine("Age:{0}", person.Age);      Console.WriteLine("Gender:{0}", person.Gender);  } |

结果如下：



   当然也可以将config["Person"]节直接映射对一个已经存在的对象，会直接覆盖原来的值。如下面代码：

|  |  |
| --- | --- |
| 1 | config["Person"].MapTo(person); |

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

**6.资源**

    大家可以从项目主页：<https://github.com/cemdervis/SharpConfig>下面源代码。亲，

    我在这里提供一个下载，以及我使用的案例Demo例子。官方提供了一个Winform的图形界面，非常直观，这里截图看看，如果大家要用，从我这里下载也可以。

    源码下载：<http://files.cnblogs.com/files/asxinyu/SharpConfig-master.rar>

　　编译4.0dll下载：[SharpConfig.rar](http://files.cnblogs.com/files/asxinyu/SharpConfig.rar)

    Winform显示配置文件的图形界面

