[**.NET中的设计模式---单件模式**](http://www.cnblogs.com/niyw/archive/2010/07/14/1776944.html)

如众所知,单件模式做为<Gof 23中设计模式>之一,其意图仅允许单件类的一个实例存在(扩展单件模式不在此文范围内),并提供全局的访问方法.UML类图如下.

<http://csharpindepth.com/Articles/General/Singleton.aspx>一文中列举了单件模式实现的5种方法,但诸多文章中都或多或少的遗漏一些问题:如何保证线程安全,如何保证仅有一个实例存在,如何根据业务变化扩展.

下面以C#实现单件为例,对这些问题一一说明.

1. **1.         如何保证线程安全**

在C#中可以使用**static,readonly**两个关键字保证单件的线程安全.**static**关键字修饰的静态成员在程序运行时由 .NET Framework 公共语言运行库 (CLR) 自动加载管理,这些成员是密封的,全局的,且不能被实例化。诸如此类的特点,让C#中的单件实现更加方便简洁.**readonly**关键字可用于保证对单件实例的引用在任何情况下不被修改.也因此与**static**一起保证单件的线程安全.

1. **2.         如何保证仅有一个实例存在**

在C#中,**static**关键字保证了对单件实例的唯一引用,隐藏构造函数保证无法通过**new**构造对象.但这并不能保证单件实例的唯一性.如我们可以反序列化出一个单件类型的实例,也可以通过实现克隆接口克隆一个单件的实例.

因此我们要保证单件类型不能被序列化,不能被克隆,即不能为单件类型及扩展类型添加**Serializable**标签和实现克隆接口或方法.

1. **3.         如何扩展**

除了在<Gof 23种设计模式>一书中提及对单件类型的扩展之外,很少有文章提到对单件类的继承扩展,甚至有的C#实现方式中直接将单件类前加上了**sealed**关键字来密封单件类.但是在实际项目中,业务需求的多变必然要求单件类能够被继承.Gof一书中提到了使用注册单件的方法实现单件类的继承.在C#中,我们可以使用继承类扩展单件类,使用反射来根据需求装载不同的单件子类型.

下面是具体的代码示例,示例中演示了C#中单件的一种实现方式和对单件类的继承.

[复制代码](javascript:void(0);)

1 using System;  
 2  using System.Collections.Generic;  
 3  using System.Linq;  
 4  using System.Text;  
 5  using System.Reflection;  
 6   
 7  namespace SingletonDemo  
 8 {  
 9 class Program{  
10 static void Main(string[] args){  
11 try{  
12 Singleton s1 = Singleton.Instance;  
13 Console.WriteLine("This is {0}.", s1.Name);  
14 Singleton sc = s1.Clone();  
15 Console.WriteLine("This is {0}.", sc.Name);  
16 s1.Name = "小小";  
17 Console.WriteLine("This is {0}.", s1.Name);  
18 Console.WriteLine("This is {0}.", sc.Name);  
19 MySingleton s2 = MySingleton.Instance as MySingleton;  
20 Console.WriteLine("{0} is {1} year old.", s2.Name, s2.Age);  
21 }  
22 catch (Exception ex){  
23 Console.WriteLine("Error:{0}",ex.Message);  
24 }  
25 }  
26 }  
27 public class Singleton{  
28 public string Name = "倪大虾";  
29 //readonly可以保证线程安全  
30   static readonly Singleton instance = Create();   
31 static Singleton(){  
32 }  
33 protected Singleton(){  
34 }   
35 public static Singleton Instance{  
36 get{  
37 return instance;  
38 }  
39 }  
40 private static Singleton Create(){  
41 try{  
42 //从外部(注册表,配置文件...)导入  
43   string typeName = "SingletonDemo.MySingleton";  
44 Type t = Type.GetType(typeName);  
45 ConstructorInfo ci = t.GetConstructor(new Type[]{});  
46 return (Singleton)ci.Invoke(null);  
47 }  
48 catch (Exception ex){  
49 throw ex;  
50 }  
51 }  
52 /\*//不能实现克隆方法  
53 public Singleton Clone(){  
54 return new Singleton();  
55 }\*/  
56 }  
57 [Serializable]  
58 public class MySingleton:Singleton{  
59 public int Age = 28;  
60 }  
61 }

[复制代码](javascript:void(0);)

特别说明:示例中单件的实现方法是个人比较推荐的众多实现方法中的一种.

分类: [.NET与设计模式](http://www.cnblogs.com/niyw/category/255528.html)

标签: [设计模式](http://www.cnblogs.com/niyw/tag/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F/), [单件模式](http://www.cnblogs.com/niyw/tag/%E5%8D%95%E4%BB%B6%E6%A8%A1%E5%BC%8F/), [Singleton](http://www.cnblogs.com/niyw/tag/Singleton/), [线程安全](http://www.cnblogs.com/niyw/tag/%E7%BA%BF%E7%A8%8B%E5%AE%89%E5%85%A8/)