**设计模式：**

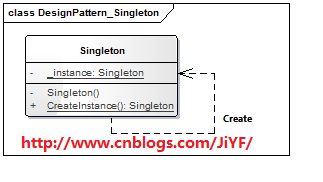
**单例模式（Singleton Pattern）**

单例模式简单介绍

单例模式的特点：

* 单例类只能有一个实例。
* 单例类必须自己创建自己的唯一实例。
* 单例类必须给所有其它对象提供这一实例。

**单例模式类图：**



**单例模式c#代码举例，分别用三种常见的单例模式举例**

**单例模式1 线程不安全**

在多线程的环境下有可能得到Singleton类的多个实例

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 单例模式1,线程不安全

3 /// </summary>

4 public class Singleton

5 {

6 private static Singleton \_instance = null;

7

8 /// <summary>

9 /// 构造方法私有，防止实例化

10 /// </summary>

11 private Singleton() { }

12 public static Singleton Instance

13 {

14 get

15 {

16 return \_instance ?? (\_instance = new Singleton()); //??表示，如果\_instance为null 则执行后面括号中的

17 }

18 }

19 }

[复制代码](javascript:void(0);)

**单例模式2 线程安全**

需要创建了一个静态只读的进程辅助对象\_instanceLock ，由于lock是确保当一个线程位于代码的临界区时，另一个线程不能进入临界区（同步操作）。如果其他线程试图进入锁定的代码，则它将一直等待，直到该对象被释放。从而确保在多线程下不会创建多个对象实例了。只是这种实现方式要进行同步操作，这将是影响系统性能的瓶颈和增加了额外的开销（只是针对系本身开销比较大的情况，一般情况下，这样是可以的，如果说对系统性能要求极高，最好采用第三种方式）

[复制代码](javascript:void(0);)

1 　　 /// <summary>

2 /// 单例模式2，线程安全

3 /// </summary>

4 public class Singleton

5 {

6 private static Singleton \_instance = null;

7 private static readonly object \_instanceLock = new object();

8 private Singleton() { }

9 public static Singleton Instance

10 {

11 get

12 {

13 lock (\_instanceLock)

14 {

15 return \_instance ?? (\_instance = new Singleton());

16 }

17 }

18 }

19 }

[复制代码](javascript:void(0);)

**单例模式3：线程安全(建议用这种方式)**

在同步操作之前，添加判断该实例\_instance是否为null就可以降低通过操作的次数了，提高系统的性能

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 单例模式3：线程安全(建议用这种方式)

3 /// </summary>

4 public class Singleton

5 {

6 private static Singleton \_instance = null;

7 private static readonly object \_instanceLock = new object();

8 private Singleton() { }

9 public static Singleton Instance

10 {

11 get

12 {

13 if (null == \_instance)

14 {

15 lock (\_instanceLock)

16 {

17 if (null == \_instance)

18 {

19 \_instance = new Singleton();

20 }

21 }

22 }

23 return \_instance;

24 }

25 }

26 }

[复制代码](javascript:void(0);)

**单例模式测试代码：**

[复制代码](javascript:void(0);)

1 class Program

2 {

3 static void Main(string[] args)

4 {

5 Singleton singleton1 = Singleton.Instance;

6 Singleton singleton2 = Singleton.Instance;

7

8 if (singleton1 == singleton2)

9 {

10 Console.WriteLine("ok");

11 }

12 else

13 {

14 Console.WriteLine("no");

15 }

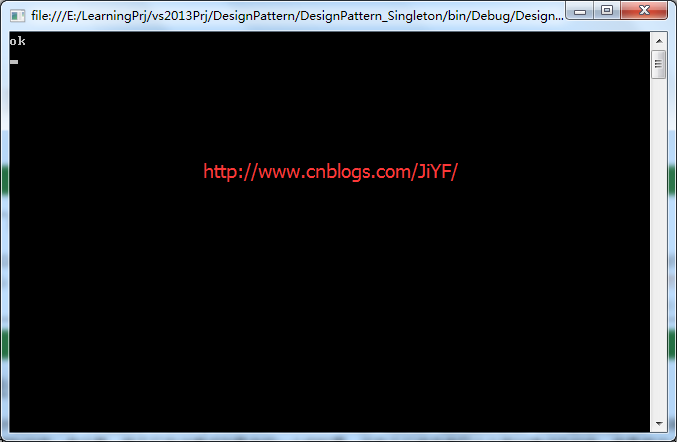
16 Console.Read();

17 }

18 }

[复制代码](javascript:void(0);)

**运行结果**



[**源代码工程文件下载**](http://files.cnblogs.com/files/JiYF/DesignPattern_Singleton.rar)