**单例模式VS静态方法**

一个典型例子：

public class SomeClass

{

private string myfield = null;

public static instance = new SomeClass();

public instranceMethod() {};

public static staticMethod() {};

}

public class AnotherClass

{

public static Main()

{

//第一种方式, 声明实例，调用实例方法

SomeClass someClass = new SomeClass();

someClass.instanceMethod();

//第二种方式，通过一个静态的实例，去调用实例方法

SomeClass.instance.instanceMethod();

//第三种方式，直接调用静态方法

SomeClass.staticMethod();

}

}

1.什么时候使用静态类代替singleton :

这里有几个很好的静态类比singleton更好的应用场景. 最基本的例子就是在Java中的java.lang.Math类和.NET中的Math类的实现方式, Math类就是用过静态方法来实现的,而不是单例来实现的。

如果你的singleton不维持任何状态, 仅仅是提供全局的访问 , 这个时候就适合用静态类 , 这样速度也更快。因此，静态类适用于一些工具类 , 其他的如单个访问资源就可以用singleton。

2.静态类和singleton之间的区别 :

① static类有更好的访问效率；

③ singleton比static class更容易[**测试**](http://lib.csdn.net/base/softwaretest)；

④如果你的需求是维护(maintain)状态, 那么singleton比static class更好；

⑤ singleton支持延迟加载 , 而static class 则不支持这样的特性 , 在重量级的对象, 延迟加载就显得非常重要；

⑥在一些依赖注入(Dependency injection framework)的框架 , 它能够很好的管理singleton对象。

3.什么时候使用singleton :

singleton 对于static class 主要的优点是更加面向对象。对于singleton你可以使用继承(Inheritance)和多态(polymorphism)来继承一个基类, 实现一个接口, 提供不同功能的实现。例如 , Java中java.lang.Runtime ,该类就是一个singleton的, 调用getRuntime(),基于不同的JVM ,返回不同的实现对象, 针对一个一个JVM,确保只有一个Runtime对象 , 如果使用static class就不能很好的来实现这样的功能了。