C#如何如何保证对象线程内唯一：数据槽（CallContext）

如果说，一个对象保证全局唯一，大家一定会想到一个经典的设计模式：单例模式，如果要使用的对象必须是线程内唯一的呢？

数据槽：CallContext 看下msdn对callContext的解释。

CallContext是类似于方法调用线程本地存储区的专用集合对象，并提供对每个逻辑执行线程都唯一的数据槽。数据槽不在其他逻辑线程上的 调用上下文之间的共享。当CallContext沿执行代码路径往返传播并且由该路径中的各种对象检查时，可将对象添加到其中。

也就是说，当前线程对对象进行存储到线程本地存储区，对象随着线程的销毁而销毁。

在程序中，方法一级一级调下去，比如A->B->C->D…->Z。在调用过程中，希望在调用函数之间传递一些数据，常见的是将特定的数据从高处往低处传递。首先想的是通过一个参数传递，这样每个函数都要声明一个参数，就像事件的调用一样，都有一个eventArg结构一样。这不是个好方法，尤其是需要调用一些以前的程序方法时，以前的方法可能没有这个参数。再一个办法就是使用全局变量，这个在dos时代比较常用，那时候多线程还不多见，全局变量比较有效。但这种方法只适合单线程程序。提到多线程，写过ASP.NET程序的人可能会想到另外一个解决办法，session！session确实能部分实现这个问题，但是也存在多线程的问题。在.NET 框架中，有个CallContext，顾名思义，就是调用上下文，大家可以查一下这个东西，这个东西可以在调用过程中，共享一些数据，并能在盗用边界进行传递，且支持多线程。

综上可以说CallContext是一个实现多线程之间进行数据传递的数据集合对象。

沿着程序执行路径进行传递。

基本使用方法：

1. 存储：使用CallContext.SetData(string, object)方法指定缓存指定的对象。
2. 取值：使用CallContext.GetData(string)方法来获取缓存中指定的对象。
3. 清除：使用CallContext.FreeNameDataSlot(string )方法来移除缓存中指定的对象。

使用场景：当对象需要线程内全局使用，而其他线程包括子线程都不能访问的时候使用。比如EF的数据上下文，每次请求都会生成一个线程处理请求，这时候创建一个数据上下文对象给不同的函数使用，最后一起提交就完全可以避免事务的问题。当然也可以创建一个变量来实现这个目的。只是这个对象你也是可以和其他线程数据进行交互的，这就违背了线程内唯一的概念了。