如果说，一个对象保证全局唯一，大家肯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，ok 看下 msdn 对 callcontent 的解释。

CallContext 是类似于方法调用的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共享。当 **CallContext** 沿执行代码路径往返传播并且由该路径中的各个对象检查时，可将对象添加到其中。

也就是说，当前线程对对象进行储存到线程本地储存区，对象随着线程的销毁而销毁。

使用代码：

```
static string key = "DbContext-Single";
      public static System.Data.Entity.DbContext Instance
      {
          get
          {
              DbContext temp = CallContext.GetData(key) as DbContext;

              if (temp == null)
              {
                  temp = new HelperModelContainer();
                  //放入数据槽中
                  CallContext.SetData(key, temp);
              }
              return temp;
          }
          private set { }
      }
```

使用场景：我个人认为，当对象需要线程内全局使用，而其他线程包扩子线程都不能访问的时候使用。比如 EF 的数据上下午，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。当然也许有人会问我可以创建一个变量来使用，同样可以达 到一样的目的，这当然也是可以的，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。