浅析StackTrace

我们在学习函数调用时，都知道每个函数都拥有自己的栈空间。一个函数被调用时，就创建一个新的栈空间。那么通过函数的嵌套调用最后就形成了一个函数调用堆栈。在c#中，使用StackTrace记录这个堆栈。你可以在程序运行过程中使用StackTrace得到当前堆栈的信息。

```
class Program
  {
     static void Main(string[] args)
     {
        Program a = new Program();
        a.FuncA();
        Console.ReadLine();
     }
     int FuncA()
     {
        FuncB();
        return 0;
     }

     private void FuncB()
     {
        MethodInfo method0 = (MethodInfo)(new StackTrace().GetFrame(0).GetMethod());
        MethodInfo method1 = (MethodInfo)(new StackTrace().GetFrame(1).GetMethod());
        MethodInfo method2 = (MethodInfo)(new StackTrace().GetFrame(2).GetMethod());

        Console.WriteLine("Current Method is : {0}",method0.Name);
        Console.WriteLine("Parent Method is : {0}", method1.Name);
        Console.WriteLine("GrandParent Method is : {0}", method2.Name);
     }
  }
```

程序的输出结果是：

```
Current Method is : FuncB
Parent Method is : FuncA
GrandParent Method is : Main
```

其中调用GetFrame得到栈空间，参数index 表示栈空间的级别，0表示当前栈空间，1表示上一级的栈空间，依次类推。

除了可以获取方法信息外，还可以调用StackFrame类的成员函数，在运行时得到代码的文件信息及行号和列号等。详情可以参考msdn上的一个example

```
// Display the stack frame properties.
StackFrame sf = st.GetFrame(i);
Console.WriteLine(" File: {0}", sf.GetFileName());
Console.WriteLine(" Line Number: {0}",
  sf.GetFileLineNumber());
// Note that the column number defaults to zero
// when not initialized.
Console.WriteLine(" Column Number: {0}",
  sf.GetFileColumnNumber());
if (sf.GetILOffset() != StackFrame.OFFSET_UNKNOWN)
{
  Console.WriteLine(" Intermediate Language Offset: {0}",
```

```
      sf.GetILOffset());
}
if (sf.GetNativeOffset() != StackFrame.OFFSET_UNKNOWN)
{
   Console.WriteLine(" Native Offset: {0}",
      sf.GetNativeOffset());
}
```