# [CLR宿主程序的启动过程](http://www.cnblogs.com/ZorryZhuo/archive/2008/10/14/1310553.html)

最近突然对CLR产生了浓厚的兴趣，想想也是，.Net开发怎么说也搞了两年多了，如果对整个Framework的机制还没有一个大致的了解还真有点说不过去。话说回来，搞了五、六年甚至十多年开发的，对进程结构、对象内存分布还一无所知的也比比皆是啊，呵呵……

首先创建一个简单的Hello World命令行程式：

    public class HelloWorldClass

    {

        public static void Main()

        {

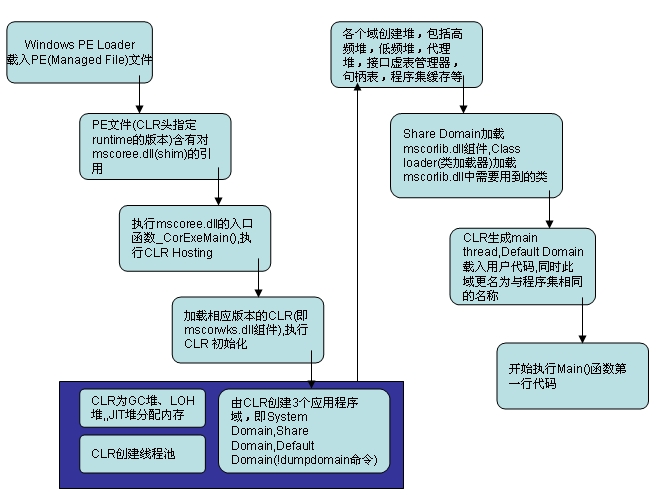
            Console.WriteLine("Hello World!");

                  Console.ReadLine();

        }

}

编译生成，得到一个“ConsoleApplication1.exe”可执行文件。双击执行，屏幕打出“Hello World!”。程序的输出非常简单，但背后的启动运行机制却相当复杂。总结起来，CLR宿主程序的启动过程大致可以用如下图表展示：



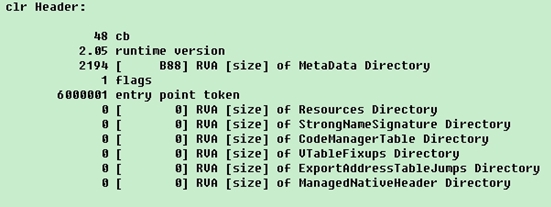
这里对每个步骤的描述都只是概括性的，实际的过程要复杂得多，比如启动CLR EE之后创建应用程序域和加载器堆以及句柄表、方法表等，都是一个个极复杂的过程，得详细阅读sscli才有可能对当中细节有个清晰的认识。

下面是对图表中一些涉及到的概念的解释：

**PE（Portable Executable）文件**：是Windows系列操作系统的可执行文件格式的统称。因为.Net的出现又与原先Win32的格式有所区别，比如增加了CLR Header。可以通过.Net Framework自带的dumpbin工具，在控制台输入如下命令来验证：

**>>dumpbin -all ConsoleApplication1.exe>c:"PE.txt**

这时在C盘根目录就可以找到命名为”PE.txt”的PE文件，通过这个文件我们可以查看到这个所谓的PE文件的具体格式。其中有一段就是CLR Header，如下所示：



**Mscoree.dll(shim,垫片)**:负责启动CLR的组件。用Depends工具查看每个.Net exe文件，都可以发现每个.Net程序都引用了此组件。同一机子上无论安装了多少个版本的.Net Framework，mscoree.dll都只会在系统路径“C:"WINDOWS"system32”(XP下)下存放一份，至于所加载的CLR的版本则取决于应用程序的配置文件、CorBindToRuntimeHost函数参数和环境变量以及注册表项。CLR的真正启动有赖于mscoree.dll的\_CorExeMain()函数，我们可以通过在程序运行过程中设置断点，查看call stack（调用堆栈）来验证，如下图：



**CLR**:即mscorwks.dll组件。此组件启动后还会载入相应的mscorjit.dll用于JIT（Just In Time）代码的生成。mscorwks.dl主要负责CLR运行环境的创建，比如应用程序域，GC堆，线程池等等，具体过程没有调试跟踪过，点到即止。

**CLR Hosting(CLR 宿主):**初始启动.Net Application时，Windows进程的执行和初始化跟传统的Win32程序是一样的，执行的还是非托管代码，只不过由于PE文件中引入了CLR Header,OS进程加载了mscoree.dll，从而启动了CLR，CLR本身不是一个可执行程序，它需要一个进程来装载并启动它，从而接管进程并创建自身的程序运行上下文，这个过程可称之为CLR Hosting。从本质上来讲，CLR是一个COM服务器，它自身封装了一系列称之为CLR Hosting APIs的接口，以便于CLR寄宿于非托管程序，从而在非托管环境上下文中执行托管程序。比如SQL Server2005和ASP.NET都运用了此种技术使CLR寄宿于其运行环境。

**AppDomain(应用程序域):**区别于进程，且位于进程内的一种程序上下文边界。CLR初始化时会创建三个域，分别为System Domain,Share Domain和Default Domain，其中System Domain负责创建和初始化Share Domain和Default Domain。它将系统库mscorlib.dll载入共享域，并且维护进程范围内部使用的隐含或者显式字符串符号。所有不属于任何特定域的代码被加载到系统库SharedDomain.Mscorlib，对于所有应用程序域的用户代码都是必需的。它会被自动加载到Share Domain中。Default Domain则负责载入并执行用户代码。

我们可以利用VS 2005调试器和SOS工具来验证这三个域的存在。在程序调试过程中，设置断点，并在“即时窗口”输入如下命令：

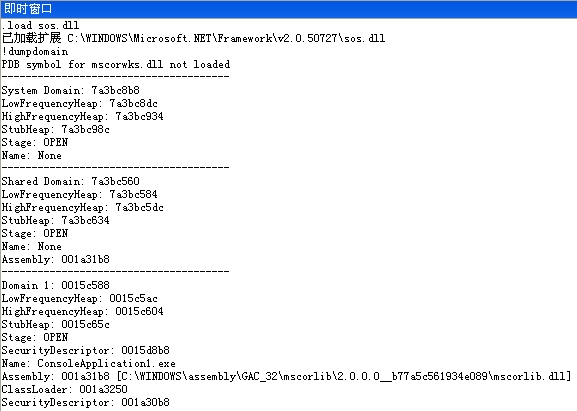
**.load sos.dll**

这时会提示“已加载扩展C:"WINDOWS"Microsoft.NET"Framework"v2.0.50727"sos.dll”

载入SOS后即可以利用它提供的各种命令来查看CLR进程。输入：

**!dumpdomain**

这时窗口会显示出如下提示：



    可以清楚地看到System Domain,Shared Domain和Domain1。Domain1即为Defautl Domain,名字为“ConsoleApplication1.exe”。

**Loader heaps(加载器堆):**在上面的截图中，在每个域的明细中都有这样三个堆，分别为LowFrequencyHeap（低频堆）、HighFrequencyHeap（高频堆）和StubHeap（代理堆），这些就是所谓的加载器堆。

在参考文章“深入探索.NET框架内部了解CLR如何创建运行时对象”中对于加载器堆是这样描述的“加载器堆不同于垃圾回收堆（或者对称多处理器上的多个堆），垃圾回收堆保存对象实例，而加载器堆同时保存类型系统。经常访问的部件如方法表，方法描述，域描述和接口图，分配在高频堆上，而较少访问的数据结构如EEClass和类加载器及其查找表，分配在低频堆。代理堆保存用于代码访问安全性（code access security， CAS）的代理部件，如COM封装调用和平台调用（P/Invoke）”。

但我通过命令 **!dumpheap**分别查看各个堆都有哪些类型时却发现三个堆所装载的类型是一样的，而不是文章中所说的依访问频率和被调用方的不同而分别存放于各个堆中。这个问题有待于以后深入研究。

总结起来，CLR宿主是一个比较复杂的过程。当中涉及到非托管与托管程序之间的交互、CLR维护其运行环境的一系列机制。以后有时间我会尝试学习用Windbg调试跟踪整个CLR的启动执行过程和探索CLR中的类型和对象内存分布。

**参考文章：**

关于.Net 程序执行过程：

<http://www.microsoft.com/taiwan/msdn/columns/DoNet/loader.htm>

关于CLR Hosting:

<http://www.microsoft.com/taiwan/msdn/columns/DoNet/NETCLR.htm>

深入探索.NET框架内部了解CLR如何创建运行时对象：

<http://msdn.microsoft.com/zh-cn/magazine/cc163791(en-us).aspx>