Zery

让技术成为我们的一种能力，但不是所有能力

读懂IL代码就这么简单(二)

一 前言

IL系列 第一篇写完后 得到高人指点，及时更正了文章中的错误，也使得我写这篇文章时更加谨慎，自己在了解相关知识点时，也更为细致。个人觉得既然做为文章写出来，就一定要保证比较高的质量，和正确率 。感谢 @冰麟轻武 的指点

你没有看第一篇？ 点这里看第一篇 读懂IL代码就这么简单(一)

IL指令大全： IL指令详解

IL反编译工具： ILDasm

知识点回顾：

Managed Heap(托管堆):用于存放引用类型的值

Evaluation Statck(计算栈)：临时存放值类型数据，引用类型地址的堆栈(这个是栈，所以遵循栈的操作特点，先进后出)

Call Stack(调用栈)：其中的Record Frame 用于存放.locals init(int32 V_0)指令的参数值如：V_0 (Record Frame是一个局部变量表，所以不遵守FILO原则)

二 指令详解(基本介绍)

2.1 知识点介绍

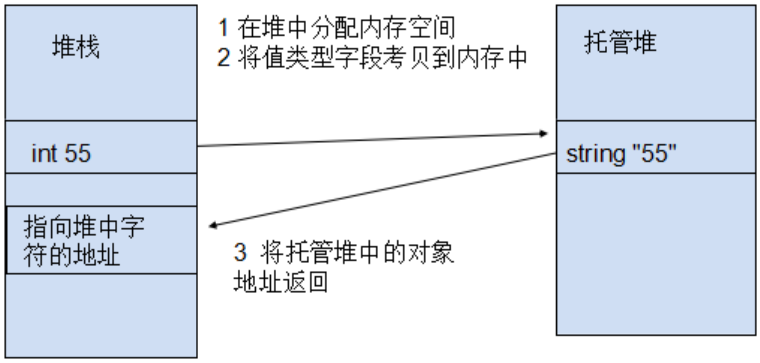
在第一篇时，我只详细的写了值类型的IL指令，这一篇会主要以引用类型为主，这一篇会有装箱操作，所以先写一下装箱操作在内存中是如何操作的

装箱操作：1 内存分配，在托管堆中分配内存空间，2 将值类型的字段拷贝到新分配的内存中，3 将托管堆中的对象地址返回给新的对象

操作过程如下图

博客地址：http://www.cnblogs.com/zery/

装箱过程



C#代码

```
        /*
         Author:zery-zhang
         BlogAddress:http://www.cnblogs.com/zery/
         */
        static void Main(string[] args)
        {
            string name = "Zery";
            int age = 22;
            Console.WriteLine(age.ToString() + name);//已ToString的操作
            Console.WriteLine(age+name);//未ToString操作

        }
```

IL代码

```
    /*
        Author:zery-zhang
        BlogAddress:http://www.cnblogs.com/zery/
    */
    .method private hidebysig static void  Main(string[] args) cil managed
{
  .entrypoint
  // Code size       48 (0x30)

 //以下代码 完成 C#代码中初始化变量的操作

  //计算栈(Evaluation Stack) 可容纳数据项的最大个数
  .maxstack  2
   //定义并初始化参数 并存入 局部变量表(Record Frame)中
  .locals init (string V_0,int32 V_1)
  IL_0000:  nop
  //把字符串压入计算栈(Evaluation Stack)中
  IL_0001:  ldstr      "Zery"
  // 从计算栈中弹出("Zery")字符，并赋值给局部变量表中第0个位置的元素V_0
  IL_0006:  stloc.0
```


随笔档案

```
21    //把整数22压入计算栈中
22    IL_0007:  ldc.i4.s   22
23    //把整数22弹出，并赋值给局部变量表中第1个位置的元素V_1
24    IL_0009:  stloc.1
25
26    //以下代码完成C#中的输出操作
27
28    //取出局部变量表中V_1元素的值 "22" (copy)并压入计算栈中
29    IL_000a:  ldloca.s   V_1
30    //弹出刚刚压入的值("22")调用ToString方法转成string类型并将引用存入计算栈中
31    IL_000c:  call       instance string
[mscorlib]System.Int32::ToString()
32     //取出局部变量表中第0个位置元素(V_0)的值("Zery")压入计算栈中(此时计算栈中有两个值，指向推管堆中"22"的引用地址和字符串"Zery")
33    IL_0011:  ldloc.0
34     //弹出计算栈中两个值调用String的Concat方法把字个字符拼接存入托管堆中(Managed Heap )并返回地址压入计算栈中
35    IL_0012:  call       string
[mscorlib]System.String::Concat(string,string)
36     //调用输出方法，调用输出方法后计算栈中的值(指向托管堆字符的地址)会被回收 。
37    IL_0017:  call       void
[mscorlib]System.Console::WriteLine(string)
38
39    //未ToString的操作
40    IL_001c:  nop
41     //取局部变量表中第1个位置的元素V_1的值("22") 压入计算栈中
42    IL_001d:  ldloc.1
43     //把刚刚压入的整数22 装箱并返回指向托管堆的地址存入计算栈中
44    IL_001e:  box        [mscorlib]System.Int32
45     //取局部变量表中第0个位置的f元素V_0的值("Zery")并压入计算栈中
46    IL_0023:  ldloc.0
47     //弹出计算栈中两个值调用String的Concat方法把字个字符拼接存入托管堆中(Managed Heap )并返回地址压入计算栈中
48    IL_0024:  call       string
[mscorlib]System.String::Concat(object,object)
49     //调用输出方法
50    IL_0029:  call       void
[mscorlib]System.Console::WriteLine(string)
51    IL_002e:  nop
52     //标记返回
53    IL_002f:  ret
54 } // end of method Program::Main
```

2.2 IL指令详解

.maxstack:计算栈(Evaluation Stack)可容纳数据项的最大个数

.locals init (int32 V_0,int32 V_1,int32 V_2)：定义变量并存入Call Stack中的Record Frame中

nop:即No Operation 没有任何操作，我们也不用管它，

ldstr.:即Load String 把字符串加压入Evaluation Stack中

stloc.：把Evaluation Stack中的值弹出赋值到Call Stack中的Record Frame中

ldloc.:把Call Stack里的Record Frame中指定位置的值取出(copy)存入Evaluation Stack中 以上两条指令为相互的操作stloc赋值，ldloc取值

call: 调用指定的方法

box:执行装箱操作

ret: 即return 标记返回

2017年9月(3)
2017年7月(2)
2017年6月(3)
2017年5月(1)
2017年4月(5)
2017年2月(1)
2017年1月(1)
2016年12月(1)
2016年11月(3)
更多

文章档案
2016年11月(1)

阅读排行榜
1. 读懂正则表达式就这么简单(286069)
2. HTTPS 原理解析(67065)
3. 读懂IL代码就这么简单 (一)(53793)
4. Centos7 ping 未知的名称或服务 DNS配置问题(49040)
5. C#操作XML方法集合(44638)
6. C#获取CPU占用率、内存占用、磁盘占用、进程信息(29194)
7. Linux jar包 后台运行(27601)
8. IL指令详细(25275)

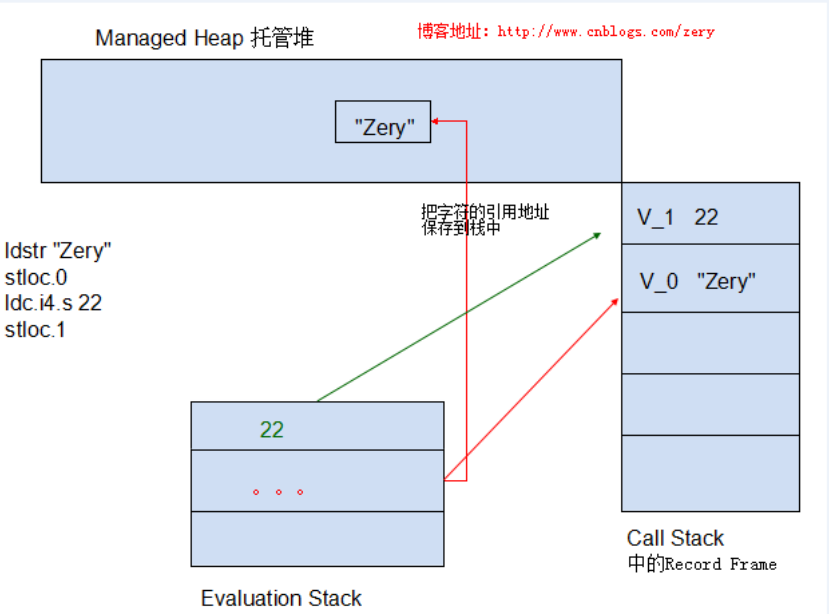
三 指令详解(深入介绍)

如果看代码中的注释你还不是很理解，那就看看下面的图解过程吧，如果每一步都画图，那工程太大了，所以我会把简单的的步组合成一张图并做上注释

先画初始化的代码详解图 注：为了减少图片所以栈的弹出与压入操作就省去了，都只画出了结果

IL_0001: ldstr "Zery"
IL_0006: stloc.0
IL_0007: ldc.i4.s 22
IL_0009: stloc.1

因为字符串是引用类型，所以是保存在托管堆中，而栈中只保存对字符引用的地址，可以看到图中的字符串是在托管中的，而计算栈中只保存了引用

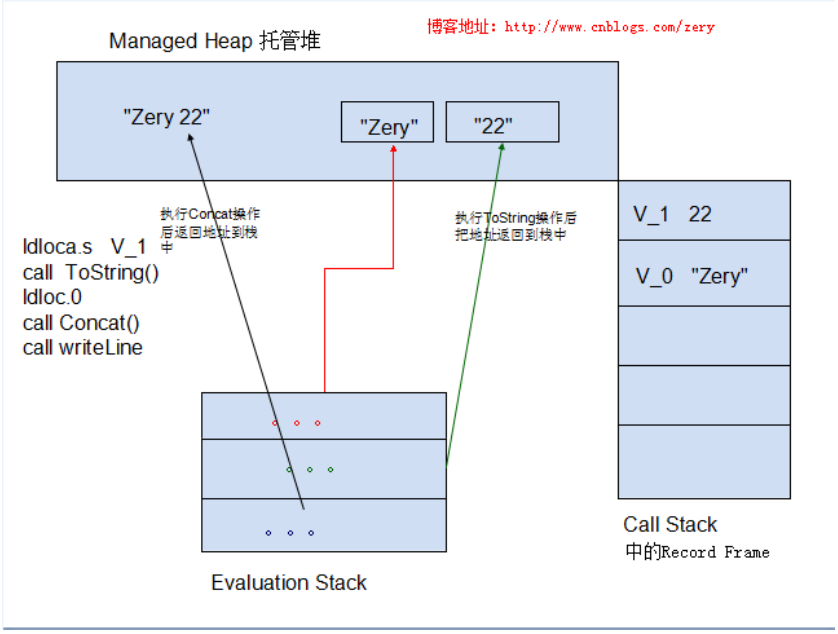


IL_000a: ldloca.s V_1
IL_000c: call instance string [mscorlib]System.Int32::ToString()
IL_0011: ldloc.0
IL_0012: call string [mscorlib]System.String::Concat(string,string)
IL_0017: call void [mscorlib]System.Console::WriteLine(string)

9. Mysql 查看死锁，解除死锁 方式(17713)
10. 做为技术人员为什么要写博客(15433)

评论排行榜
1. 读懂IL代码就这么简单 (一)(104)
2. 2014年读书计划(102)
3. 做为技术人员为什么要写博客(77)
4. 读懂正则表达式就这么简单(53)
5. 采集博客园文章，用瀑布流+无限滚动展示(附源码)(49)
6. 百度广告 高亮 Chrome插件(附源码)(48)
7. 文件夹管理工具(MVC+zTree+layer)(附源码)(44)
8. 委托 你怎么看？(37)
9. 读懂IL代码就这么简单(二)(34)
10. 常用加解密方法汇总 工具 (附源码)(33)

推荐排行榜
1. 读懂正则表达式就这么简单(178)
2. 读懂IL代码就这么简单 (一)(167)
3. 做为技术人员为什么要写博客(138)
4. 常用加解密方法汇总 工具 (附源码)(100)
5. 让数据决策你的行为--拉勾网数据分析(87)



IL_001d: ldloc.1

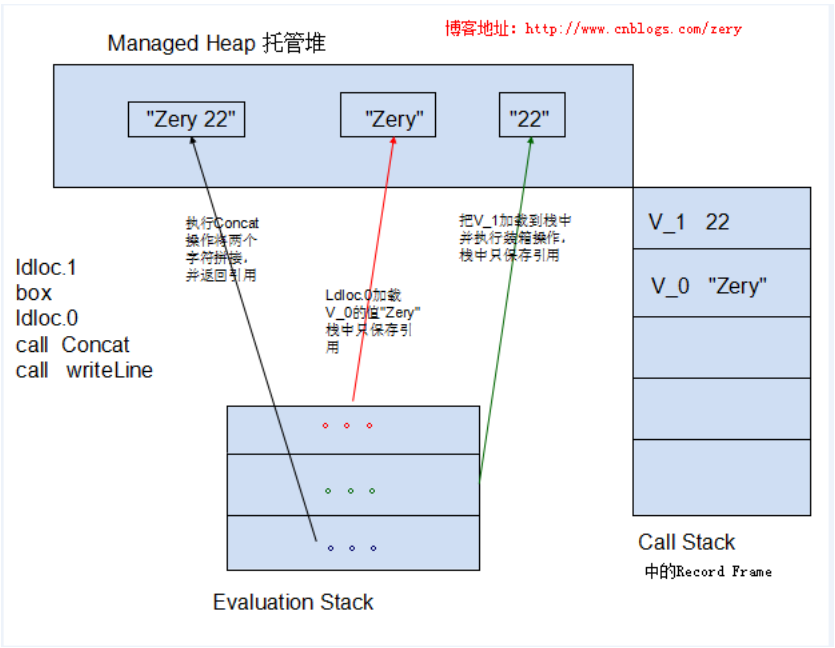
IL_001e: box [mscorlib]System.Int32

IL_0023: ldloc.0

IL_0024: call string [mscorlib]System.String::Concat(object, object)

IL_0029: call void [mscorlib]System.Console::WriteLine(string)

装箱的过程图在上面已给出此处只把结果画出



四 总结

1 用两篇把 值类型与引用类型在内存中不同的位置与不同的操作详细的写了，我觉得还是很有必要的，因为所有的数据类型由这两种类型组成，把这两种类型的操作了解了

看其它IL指令就是透过本质看现象了。

2 关于画图，我觉得画图是程序员必学的知识，牛X的程序员画出来的，系统架构图，系统设计图等都是很有结构很清晰的。我的画图技能还有太多不足，只是画这种简单的图都觉得，无法完美的表达自己头脑所想的那样，

如果觉得文章与给您带来一点 收获 那就 帮忙点个**推荐**吧，也让更多的人能关注并了解IL 您的推荐是我源源不断的写作力

如果您希望您的技术路上能有更多的朋友，那就关注一下吧

注：本人不才，水平有限，如有不对之处，希望能及时提出，我会马上更正，以免误导他人 谢谢！

成长在于积累

分类： C#基础知识

标签： IL 系列

Zery
关注 - 91
粉丝 - 738
+加关注

50 1

« 上一篇： 读懂IL代码就这么简单 (一)
» 下一篇： 委托 你怎么看?

posted @ 2013-10-21 08:54 Zery 阅读(10130) 评论(34) 编辑 收藏 举报

编辑推荐:
· 流量录制与回放技术实践
· 熟悉而陌生的新朋友——IAsyncDisposable
· 对象池在 .NET (Core)中的应用[3]: 扩展篇
· 奇思妙想 CSS 3D 动画 | 仅使用 CSS 能制作出多惊艳的动画?
· 一个测试工程师的成长复盘

最新新闻:
· 官方发布！神舟十二号航天员在轨拍摄作品震撼来袭！（2021-08-31 14:50）
· 营收大增却巨额亏损，王兴打的是什么算盘？（2021-08-31 14:37）
· AI独角兽“第四范式”冲刺港股：2021半年营收7.8亿元、研发费用占七成（2021-08-31 14:16）
· 拼多多、京东，正在成为对方的样子（2021-08-31 14:00）
· 阿里兵合一处战美团，高德将步前人后尘？（2021-08-31 13:45）
» 更多新闻...