C#中CLR（公共语言运行时）与IL（中间代码） - 文章 - 伯乐在线

.net平台中的CLR

首先要说明的是，.NET平台与C#不是一回事 它是C#,VB.net等程序运行的平台。

CLR是公共语言运行时，是 .NET Framework的重要组成部分。它提供了内存管理、线程管理和异常处理等服务，而且还负责对代码实施严格的类型安全检查，保证了代码的正确性。

事实上，类型安全(Type Checker)、垃圾回收（Garbage Collector）、异常处理(Exception Manager)、向下兼容(COM Marshaler)等很多C#中的特性都是由CLR来提供的。

什么是IL

.NET Framework是架构在Windows平台上的一个虚拟的运行平台，你可以想象将最下层Windows换做其他的操作系统，例如说Linux,一样可以实现使用符合CLS(Common Language Specification,通用语言规范)的.NET语言，这其实就是Mono计划要实现的功能。因而，理论上，C#是一种可以跨平台的语言。

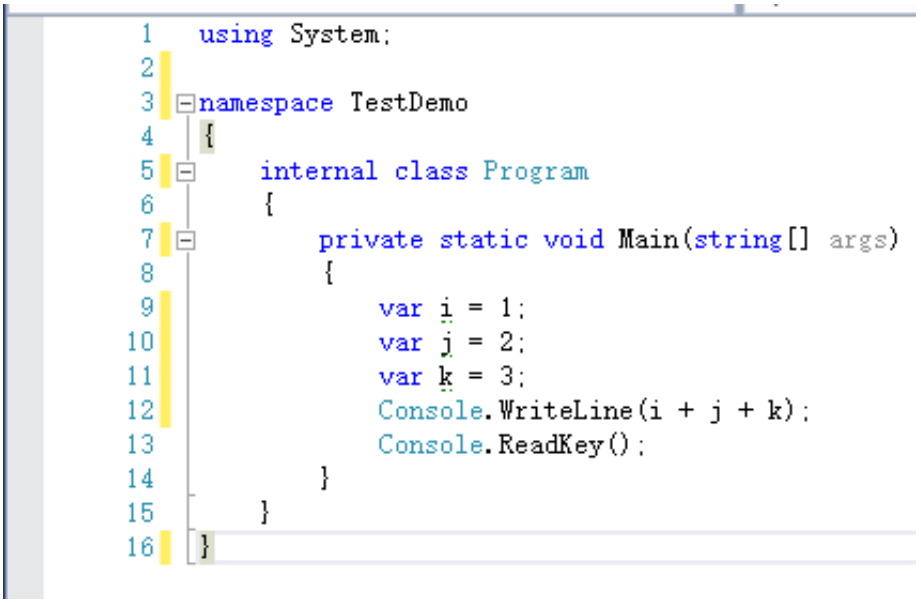
C#另一个比较象Java的地方是，它也是一种（特殊意义上的）语言，同Java一样，C#编写的程序代码也是先通过C#编译器编译为一种特殊的字节代码， (Microsoft Intermediate Language,MSIL，微软)中间语言，运行时再经由特定的编译器（JIT编译器，Just In tIME, JITer）编译为机器代码，以供操作系统执行。（关于JIT的介绍。可以看这篇博客，请点这里）

IL是一门中间语言 ，.NET平台上的各种高级语言（如C#，VB，F#）的编译器会将各自的文字表述方式转化为IL。各种不同的文字形式最终被统一到了IL的表述方式

CLR加载了IL之后，当每个方法第一次被执行时，就会使用JIT将IL代码进行编译为机器码，机器码和汇编其实也是一一对应的，可以这样理解：汇编是机器码的文字表现形式，提供了一些方便人们记忆的“助记符”。

对于同样的IL，JIT会把它为不同的CPU架构（如x86/IA64等等）生成不同的机器码。

C＃代码及其对应的IL中间代码



```
using System;

namespace TestDemo
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            var i = 1;
            var j = 2;
            var k = 3;
            Console.WriteLine(i + j + k);
            Console.ReadKey();
        }
    }
}
```

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

//hidebysig指令表示如果当前类为父类，用该指令标记的方法将不会被子类继承//cil managed表明方法体中的代码是IL代码，且是托管代码，即运行在CLR运行库上的代码

```
.method private hidebysig static void Main(string[] args)cil managed
{
```

.entrypoint //该指令代表该函数程序的入口函数。每一个托管应用程序都有且只有一个入口函数，CLR加载程序时，首先从.entrypoint函数开始执行。

.maxstack 2 //执行构造函数时，评估堆栈可容纳数据项的最大个数。评估堆栈是保存方法中所需要变量的值的一个内存区域，该区域在方法执行结束时会被清空，或者存储一个返回值。

```
.locals init (
```

```
        [0] int32 num,
        [1] int32 num2,
        [2] int32 num3) //表示定义int类型的变量，变量名分别为num,num2,num3。存储在调用栈。
    L_0000: nop    //No operation的意思，即没有任何操作。
    L_0001: ldc.i4.1    //将“1”压入评估栈，此时“1”处于评估栈的栈顶。
    L_0002: stloc.0    //此指令表示把值从评估栈中弹出，并赋值给调用栈的第0个变量num。
    L_0003: ldc.i4.2
    L_0004: stloc.1
    L_0005: ldc.i4.3
    L_0006: stloc.2 //从.locals init到L_0006，相当于C#代码的为i，j，k赋值。
    L_0007: ldloc.0    //取调用栈中位置为0的元素压入评估栈（取i的值）。
    L_0008: ldloc.1    //取调用栈中位置为1的元素压入评估栈（取j的值）。
    L_0009: add    //做加法操作
    L_000a: ldloc.2 //取调用栈中位置为2的元素压入评估栈（取k的值）。
    L_000b: add    //做加法操作
    L_000c: call void [mscorlib]System.Console::WriteLine(int32) //调用输出方法
    L_0011: nop    //No Operation
    L_0012: call valuetype [mscorlib]System.ConsoleKeyInfo
[mscorlib]System.Console::ReadKey() //调用ReadKey方法
    L_0017: pop //把评估栈的内容清空
    L_0018: ret    //return 标记返回值
} //Main方法结束
```

通过上面的代码，我们可以总结一下： .maxstack:代码中变量需要在调用栈（Call Stack）中占用几个位置； .locals int(……)：定义变量初始化并放入调用栈中（Call Stack）； nop：No Operation，没有任何操作； ldstr：Load String，把字符串压入评估栈（Evaluation Stack）中；

ldc.i4.1：把数值2以4字节长度整数的形式压入评估栈； stloc：把评估栈（Evaluation）中的值弹出赋值到调用栈中（Call Stack）； ldloc：把调用栈（Call Stack）中指定位置的值取出（Copy）压入评估栈（Evaluation Stack）中； call：调用指定的方法，这个指令一般用于调用静态方法；而callvir则一般用于调用实例方法； ret：return ，标记返回。

下面再看一个例子

C#