

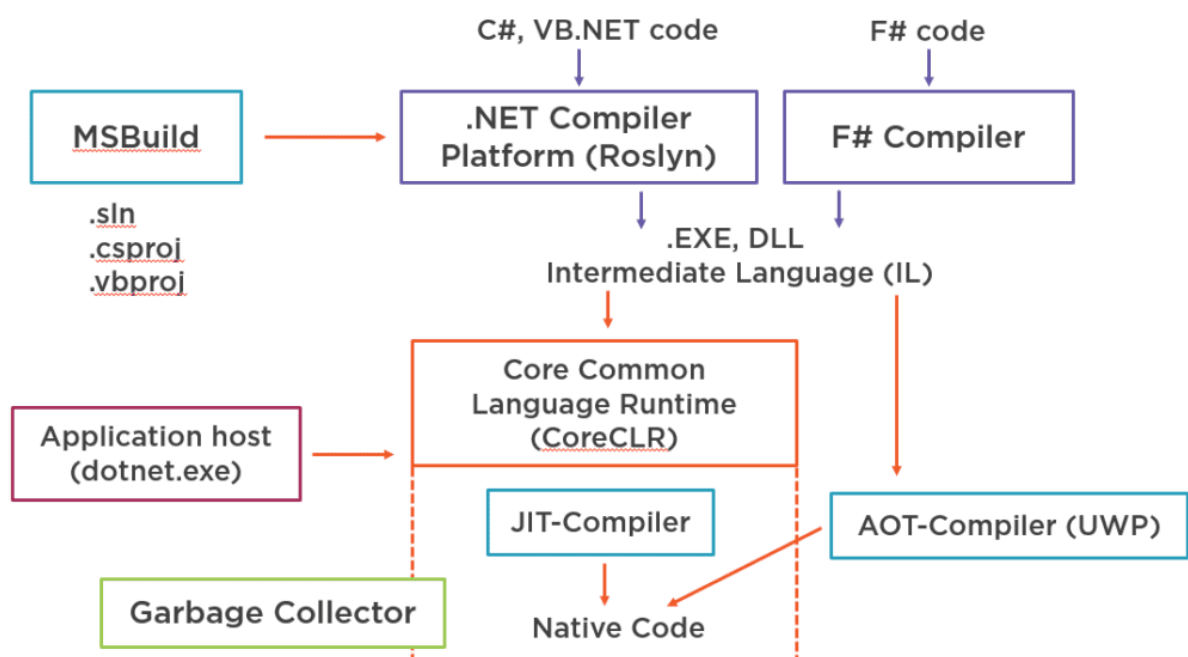
第02期-C#语言的反编译与代码保护

2021年12月16日 13:14

C# 语言介绍

C#（读作：See Sharp）是一种新式的，面向对象，类型安全的编程语言，开发人员利用 C# 能够生成在 .NET 中运行的多种安全可靠的应用程序，目前最新版本 C# 10 版本。

.NET 体系结构



从 Hello World 开始

```
Console.WriteLine("Hello, World!");
```

```
namespace MyApp
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

新的 C# 模板生成顶级语句

来自 <<https://docs.microsoft.com/zh-cn/dotnet/core/tutorials/top-level-templates>>

代码编译过程

dotnet build

内部使用 MSBuild 生成项目，因此它支持并行生成和增量生成。

来自 <<https://docs.microsoft.com/zh-cn/dotnet/core/tools/dotnet-build>>

`dotnet build --configuration Debug`

MSBuild

Microsoft 生成引擎是一个用于生成应用程序的平台。此引擎（也称为 MSBuild）为项目文件提供了一个 XML 架构，用于控制生成平台处理和生成软件的方式。Visual Studio 会使用 MSBuild，但 MSBuild 不依赖于 Visual Studio。通过在项目或解决方案文件中调用 `msbuild.exe`，可以在未安装 Visual Studio 的环境中安排和生成产品。

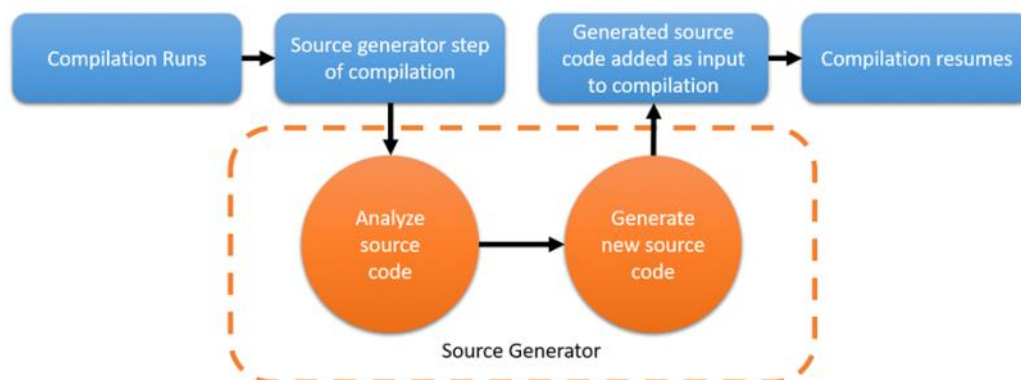
来自 <<https://docs.microsoft.com/zh-cn/visualstudio/msbuild/msbuild>>

"C:\Program Files\Microsoft Visual Studio\2022\Enterprise\MSBuild\Current\Bin\MSBuild.exe"

`MSBuild ConsoleApp1.csproj -property:Configuration=Debug`

`csc /target:exe TestApp.cs`

源代码生成器



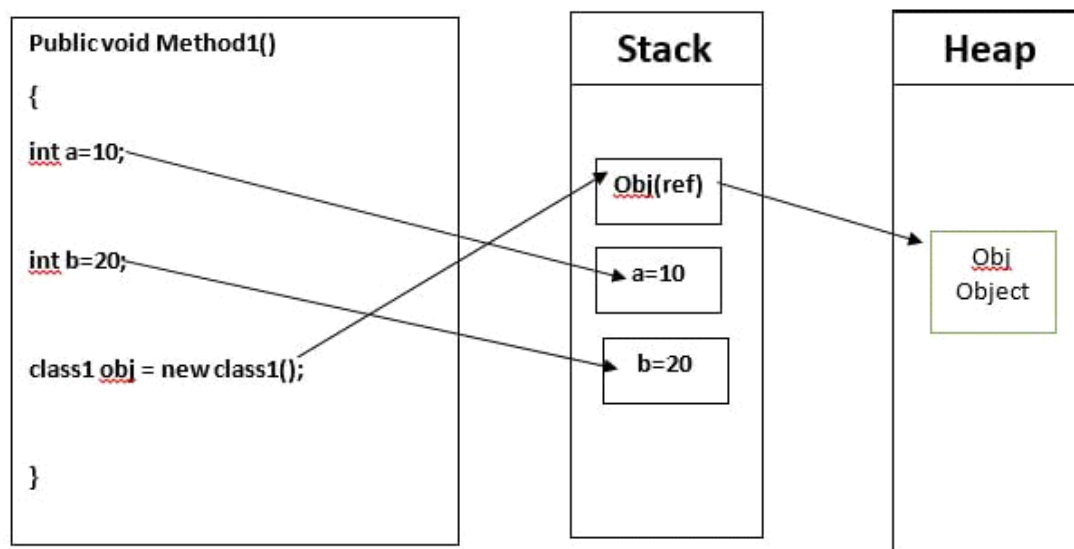
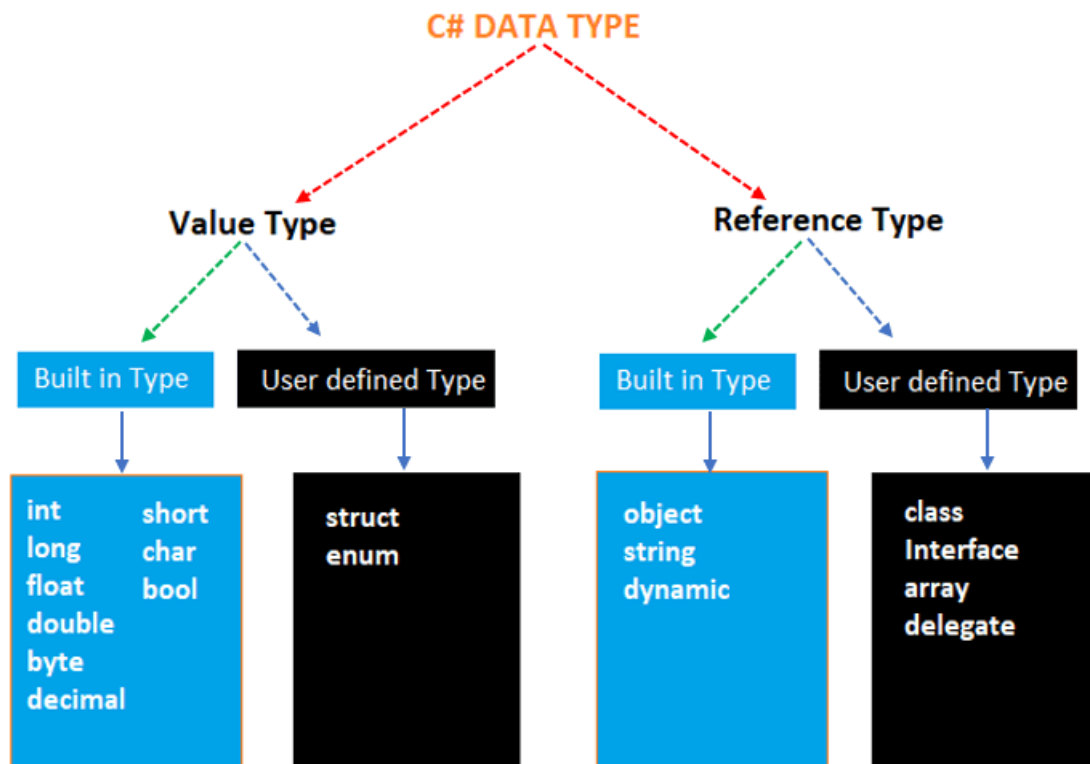
来自 <<https://docs.microsoft.com/zh-cn/dotnet/csharp/roslyn-sdk/source-generators-overview>>

类型和变量

类型：声明可以包含其成员、基类型、它实现的接口和该类型允许的操作。保留字：@

变量：用于引用特定类型的实例的标签。

两种类型：值类型 和 引用类型。值类型的变量直接包含它们的数据。引用类型的变量存储对数据（称为“对象”）的引用。对于引用类型，两个变量可以引用同一个对象。



程序的执行结构

```
namespace Acme.Collections;
```

```

public class Stack<T>
{
    Entry _top;

    public void Push(T data)
    {
        _top = new Entry(_top, data);
    }

    public T Pop()
    {
        if (_top == null)
        {
            throw new InvalidOperationException();
        }
        T result = _top.Data;
        _top = _top.Next;

        return result;
    }

    class Entry
    {
        public Entry Next { get; set; }
        public T Data { get; set; }

        public Entry(Entry next, T data)
        {
            Next = next;
            Data = data;
        }
    }
}

```

```

class Example
{
    public static void Main()
    {
        var s = new Acme.Collections.Stack<int>();
        s.Push(1); // stack contains 1
        s.Push(10); // stack contains 1, 10
        s.Push(100); // stack contains 1, 10, 100
        Console.WriteLine(s.Pop()); // stack contains 1, 10
        Console.WriteLine(s.Pop()); // stack contains 1
        Console.WriteLine(s.Pop()); // stack is empty
    }
}

```

使用 ILDASM 查看 IL 字节码

Ilasm.exe (IL 汇编程序)

来自 <<https://docs.microsoft.com/zh-cn/dotnet/framework/tools/ilasm-exe-il-assembler>>

Ildasm.exe (IL 反汇编程序)

来自 <<https://docs.microsoft.com/zh-cn/dotnet/framework/tools/ildasm-exe-il-disassembler>>

```
ilasm myTestFile /dll /output:myNewTestFile.dll
```

推荐《.NET探秘MSIL权威指南》这本书

反编译工具

<https://github.com/icsharpcode/ILSpy>

应用程序保护

PreEmptive Protection - Dotfuscator 提供全方位的 .NET 应用程序保护。

来自 <<https://docs.microsoft.com/zh-cn/visualstudio/ide/dotfuscator>>

命令行解析器库 CommandLineParser

来自 <<https://github.com/commandlineparser/commandline/wiki>>