Microsoft

LESSON 3.1

10754 Microsoft .NET Fundamentals

Understand the fundamentals of MSIL and CLI

LESSON 3.1

98-372 Microsoft .NET Fundamentals

Lesson Overview

Understand the fundamentals of Microsoft® Intermediate Language (MSIL) and Common Language Infrastructure (CLI).

In this lesson, you will:

- Learn about MSIL.
- Learn how code is Just-In-Time (JIT) compiled.
- See MSIL code that was created by the compiler.

LESSON 3.1

98-372 Microsoft .NET Fundamentals

Guiding Questions

1. How is source code converted into a format that the computer can execute?
2. How can a developer examine the MSIL code created by the .NET compiler?

Anticipatory Set

To the best of your ability, explain what happens when you Start Debugging an application created in Visual Studio®. In other words, what does Visual Studio have to do to run your application?

MSIL

- When Microsoft Visual Studio compiles (or "builds") source code written in one of the .NET languages, that code is first converted into MSIL.
- MSIL is a CPU-independent set of instructions that can be converted efficiently to native code.
- By default, this MSIL code is stored in a file format known as "portable executable" or PE.
  - A PE file also includes metadata, such as information about all the classes in your code.

MSIL (continued)

- Because MSIL is platform (CPU)-independent, the compiler produces the same MSIL code regardless of the type of machine or operating system that will execute the application.
- MSIL is also language-independent, meaning that no matter what .NET language is used to develop an application, the application will be compiled into MSIL.
- It's important to remember: Because MSIL is not "native" to a CPU, there is one more necessary step before the application can run. Namely, it must be converted to code that is specific to the target CPU.
- What does MSIL look like? Consider a basic application that simply outputs "Hello World" to the console…

Partial Example of MSIL Code

```
// ============== CLASS MEMBERS DECLARATION ==================

.class public auto ansi beforefieldinit Hello
       extends [mscorlib]System.Object
{
  .method public hidebysig static void  Main(string[] args) cil managed
  {
    .entrypoint
    // Code size       13 (0xd)
    .maxstack  8
    IL_0000:  nop
    IL_0001:  ldstr      "Hello World!"
    IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
    IL_000b:  nop
    IL_000c:  ret
  } // end of method Hello::Main
```

MSIL (continued)

- Microsoft Visual Studio includes an MSIL Disassembler that takes a PE file and outputs a text file with MSIL, as shown on the previous slide.
  - This tool is called Ildasm.exe and can be executed from the Visual Studio command prompt.
  - You'll use this disassembler in this lesson.
- Although MSIL may look confusing, it is essentially just another programming language.
- One important feature of MSIL is that it can be converted to native code very efficiently. That's important, because it still needs to be compiled by the JIT compiler…

JIT Compiler

- The MSIL code generated when a project is built cannot run by itself—it in turn must be compiled into native code for the target machine architecture.
- This last compilation step is handled by a JIT compiler.
  - It is called “just in time” because it compiles the code during program execution.
- The JIT compiler does not compile the entire application; rather, it compiles the code it needs as it runs and saves the resulting native code in memory so that it is available for additional calls.
  - In effect, when a method is called for the first time, it is compiled and saved for subsequent method calls.
  - This saves time and memory—code that is never executed is never compiled.

JIT Compiler (continued)

- JIT compilers are platform-specific and part of the .NET Framework.
- A PE file will not execute on a machine that does not have the appropriate .NET Framework installed—the MSIL can't be converted to native code without it!

Activity

Complete the MSIL Disassembly process in the Student Activity.

Lesson Review

1. List the steps that .NET code goes through as part of the execution process?

2. What is the function of MSIL?

3. What does the JIT compiler do?