Microsoft

LESSON 7.2

10754 Microsoft .NET Fundamentals

Understand the Difference Between Managed and Unmanaged Applications

Lesson Overview

Understand the difference between managed and unmanaged applications.

In this lesson, you will:

- Learn the characteristics of native code.
- Explore the features and benefits of code managed by the Common Language Runtime (CLR).
- Compare native execution and managed execution.

Guiding Questions

1. What is native code?
2. What are the characteristics of managed code?
3. How is managed code execution different from native code execution?

Anticipatory Set

Summarize the steps that a program goes through from when you type the source code to when the program successfully finishes executing.

Native Code

- The term “native code” refers to code that is targeted for a specific hardware/operating system architecture.
  - It is sometimes called “machine code” or “machine language.”
  - Native code is said to “run on the processor”; that is, it does not require any further translation or conversion to be understood by the computer’s hardware.
  - Eventually, all code must be compiled into the target machine’s language.
- Programming languages that use compilers targeted to native code can be referred to as “unmanaged” languages. Examples include compilers for C/C++, Pascal, and many other high-level programming languages.

Managed Code

- Managed code is “managed” by an environment called the Common Language Runtime (CLR).
  - It’s called “Common Language” Runtime because it is the same regardless of the language used to develop the application or assembly.
- From the perspective of an application developer or programmer, one of the most important tasks of the CLR is resource (memory) management—it allocates the available memory and releases that memory when no longer needed.

Note: A high-level programming language is not inherently native or managed—that distinction depends upon how the code is compiled, as you will see.

Unmanaged Execution Process

In general, the steps for executing unmanaged code are as follows:

1. Develop the source code in a high-level programming language (C or Pascal, for example).
2. Compile the source code into machine language.
3. The processor executes the machine-language code.

Managed Execution Process

The steps for executing managed code are as follows:

1. Develop the source code in a high-level programming language (Microsoft® C# or Microsoft Visual Basic®, for example).
2. Compile the source code into Microsoft Intermediate Language (MSIL).
3. At run time, MSIL is “just-in-time (JIT)”-compiled into machine-language code.
4. The processor executes the machine-language code.

MSIL

- The high-level compiler converts source code into MSIL code.
- MSIL is a CPU-independent set of instructions that can be converted to native code efficiently.
  - Because it's CPU-independent, MSIL is the same regardless of the target platform.
  - Likewise, MSIL is language-independent; and an application will be compiled into MSIL regardless of the .NET language used during development.

JIT Compilation

- 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.
- When a managed application is executed, the CLR uses a JIT compiler to create native code.
  - It is called “just in time” because it compiles the code during program execution.
- The native code is executed by the processor; when the application has finished executing, the CLR disposes of the native code.

Benefits of Managed Code and the CLR

- Automatic memory management
  - Developers do not need to write code to allocate and release memory.
- Type safety
  - Managed code cannot access unauthorized memory locations.
- Security
  - The CLR controls code access and helps prevent the execution of dangerous code.
- Language interoperability
  - The CLR allows developers to use code libraries written in different programming languages.
- Platform independence
  - The developer does not need to target source code for a specific platform—the JIT compiler will convert the code as necessary.

Assignment

- Create the following table (in Microsoft Word or with a pencil on paper).
- Compare and contrast each type of code by listing their characteristics.

Native	Managed