Microsoft

LESSON 4.2

10754 Microsoft .NET Fundamentals

Understand Console I/O

LESSON 4.2

10754 Microsoft .NET Fundamentals

Lesson Overview

Understand console I/O.

In this lesson, you will:

- Learn about outputting to the console.
- Learn about getting user input via the console.
- Create a simple console application.

Guiding Questions

1. What is a console application?
2. How is the `Console` class used for basic I/O?
3. How is a simple console application created in Microsoft® Visual Studio®?

Anticipatory Set

With a partner, generate an idea for a simple computer game that could be implemented without any graphical user interface (GUI) elements—that means no images, sprites, buttons, text boxes, or other GUI elements.

Console Applications

- The simplest type of application in the .NET Framework is a console application.
  - Console applications rely on text input and output at the command line as a user interface.
- A console application can be defined as a program that accesses three basic data streams: standard input, standard output, and standard error data streams.
- In practice, console applications often are executed from the command line and are characterized by a black window with white text—the Command Prompt window.
- Visual Studio provides templates for creating console applications in each of the .NET languages.

Console Output

- Many programmers learn to use `System.Console.WriteLine` very early, both as a way to make simple applications and as a debugging tool—using console output to display variable values.
- `Console.Write` and `Console.WriteLine` (with a variety of overloads) write data to the standard output stream (the Console window); `WriteLine` adds a line terminator.
- Example:

```
System.Console.Write("Hello ");
System.Console.WriteLine("World");
System.Console.WriteLine("The End");
```

Outputs the following:

```
Hello World
The End
```

Console Input

- Can be implemented using a different set of `Console` methods: `System.Console.Read`, `System.Console.ReadKey`, and `System.Console.ReadLine`.
  - All three methods block their return and wait while the user types.
- `Console.Read` retrieves and returns the next character entered.
- `Console.ReadKey` retrieves and returns data indicating the character or function key entered, including modifier keys used (such as SHIFT, ALT, CTRL).
- `Console.ReadLine` retrieves and returns (as a string) one line (a sequence of characters terminated by a line feed or line feed). Typically, the line is terminated when the user presses the ENTER key.
- Often used as part of an assignment statement, such as `string line = System.Console.ReadLine();`

Console Applications

- The methods in `System.Console` can be used throughout .NET applications.
- They are used most commonly in console applications, which rely on console I/O for most, if not all, user interaction.
- The Console Application templates in Visual Studio create simple projects with a main method or procedure.

Console Applications in Visual Basic

Here is a “Hello World” program in Visual Basic®:

```
Module Module1
    Sub Main()
        System.Console.WriteLine("Hello World")
        System.Console.Read()
    End Sub
End Module
```

Console Applications in Visual C#

Here is a “Hello World” program in Visual C#®:

```
class Program
   {
       static void Main(string[] args)
       {
         //System.Console.WriteLine("Hello World‟);
         System.Console.WriteLine("Hello World");
         System.Console.Read();
       }
   }
```

Assignment

Create a console application that prompts the user to enter a name, then greets the user by name.

Example:

"What is your name?"

<user input: Joe>

"Hello, Joe"

Sample Solution in Visual Basic

```
Sub Main()
        System.Console.Write("Enter your name -> ")
        Dim name As String
        name = System.Console.ReadLine()
        System.Console.Write("Hello, " + name)
        System.Console.Write(" - Press ENTER to end")
        System.Console.ReadLine()
End Sub
```

Sample Solution in Visual C#

```
class Program
{
    static void Main(string[] args)
    {
        System.Console.Write("Enter your name -> ");
        string name = System.Console.ReadLine();
        System.Console.Write("Hello, " + name);
        System.Console.Write(" - Press ENTER to end");
        System.Console.ReadLine();
    }
}
```

Ticket Out the Door

- Explain how the `Console` class can be used to create a simple, text-based user interface.
- Do you have any questions about console I/O?