C# String Generation with String.Format

by Richard Carr, published at http://www.blackwasp.co.uk/StringFormat.aspx

.NET 1.1+

The twentieth part of the C# Fundamentals tutorial adds to the previous examination of conversion between numeric and string data. This article describes the Format method of the String class that allows generation of strings containing text and numbers.

Previous: C# Number to String Conversion

Download Source Code

String.Format Method

The *Format* function is a static method of the String class. It allows the generation of strings that contain textual information and inserted data. This is ideal for outputting information for the user to read. The Format method requires several arguments. The first is the template string. The template contains text that is desired in the final string and one or more placeholders, which will be replaced with other values. The rest of the parameters are variables or literals to insert into the template.

NB: The variables or literals can be of any data type including numeric data, strings and objects. For this article, we will use numeric data.

Templates and Placeholders

As described above, the template string must include placeholders that determine where values will be inserted. Each placeholder consists of a number in braces {}. The number is an index that refers to one of the other parameters passed to the Format method. The index of the first non-template parameter is zero. An example can illustrate this.

```
string output;
int selection = 5;

// Generate the output string
output = string.Format("You selected item {0} from the list.", selection);

Console.WriteLine(output);      // Outputs "You selected item 5 from the list."
```

In the example, the template string includes one placeholder. The number between the braces is zero, indicating that the first parameter after the template should be used as a replacement for the placeholder. In this case the value of the "selection" variable. However, the Format method is not limited to including only one parameter after the template. The following example shows two placeholders.

```
string output;
int selection = 5;
int count = 10;

// Generate the output string
```

.NET Framework
Algorithms and Data Structures
Audio
C# Programming
Configuration
Debugging
Design Patterns
Documentation
Graphics
Input / Output
LINQ
Network and Internet
Parallel and Asynchronous
Performance
Programming Concepts
Refactoring
Reference Sheets
Reflection
Security
SQL Server
System Information
Testing
Tools
Visual Studio
Windows Programming
Windows Presentation Foundation
XML

```
output = string.Format("You selected item {0} from {1}.", selection, count);

Console.WriteLine(output);     // Outputs "You selected item 5 from 10."
```

Numeric Formatting

When inserting numbers into strings using String.Format, the numbers can be formatted by adding format specifiers within the placeholders. The available format specifiers are the same as those for the ToString method of numeric data types. A table of format specifiers is included in part 19 of the C# Fundamentals tutorial.

To include a format specifier, a colon (:) is inserted after the placeholder index number. The specifier is then added before the closing brace. The following example shows the same parameter being used by two placeholders. One uses formatting, the other does not.

```
string output;
int value = 100;

// Convert a number to hexadecimal
output = string.Format("The decimal value {0} = {0:X} in hex.", value);

Console.WriteLine(output);     // Outputs "The decimal value 100 = 64 in hex."
```

Including Braces in Templates

In some circumstance you may require brace characters in the template string to be included in the output string. If you do, add two brace characters wherever you want one in the result.

```
string output;
int value = 100;

// Convert a number to hexadecimal
output = string.Format("{{The decimal value {0} = {0:X} in hex.}}", value);

Console.WriteLine(output);     // Outputs "{The decimal value 100 = 64 in hex.}"
```

Console.WriteLine Method

Early in the tutorial the Console.WriteLine method was used to output text. Console.WriteLine uses the same functionality as String.Format when generating the string to be outputted. The above example can therefore be simplified as follows:

```
int value = 100;

// Convert a number to hexadecimal
Console.WriteLine("{{The decimal value {0} = {0:X} in hex.}}", value);
```

Next: C# Simple String Formatting Functions

18 November 2006