Console.WriteLine Method (String, Object)

.NET Framework 4.5 | Este tópico ainda não foi avaliado como

Writes the text representation of the specified object, followed by the current line terminator, to the standard output stream using the specified format information.

Namespace: System
Assembly: mscorlib (in mscorlib.dll)

Syntax

C#

```
[HostProtectionAttribute(SecurityAction.LinkDemand, UI = true)]
public static void WriteLine(
        string format,
        Object arg0
)
```

Parameters

format
Type: System.String
A composite format string (see Remarks).

arg0
Type: System.Object
An object to write using *format*.

Exceptions

Exception	Condition
IOException	An I/O error occurred.
ArgumentNullException	*format* is **null**.
FormatException	The format specification in *format* is invalid.

Remarks

This method uses the composite formatting feature of the .NET Framework to convert the value of an object to its text representation and embed that representation in a string. The resulting string is written to the output stream.

The *format* parameter consists of zero or more runs of text intermixed with zero or more indexed placeholders, called format items, that correspond to an object in the parameter lis this method. The formatting process replaces each format item with the text representation of the value of the corresponding object.

The syntax of a format item is **{***index*[**,***alignment*][**:***formatString*]**}**, which specifies a mandatory index, the optional length and alignment of the formatted text, and an optional string format specifier characters that govern how the value of the corresponding object is formatted.

The .NET Framework provides extensive formatting support, which is described in greater detail in the following formatting topics.

- For more information about the composite formatting feature supported by methods such as Format, AppendFormat, and some overloads of WriteLine, see Composite Formatting.
- For more information about numeric format specifiers, see Standard Numeric Format Strings and Custom Numeric Format Strings.
- For more information about date and time format specifiers, see Standard Date and Time Format Strings and Custom Date and Time Format Strings.
- For more information about enumeration format specifiers, see Enumeration Format Strings.
- For more information about formatting, see Formatting Types.

For more information about the line terminator, see the Remarks section of the WriteLine method that takes no parameters.

Note

The HostProtectionAttribute attribute applied to this type or member has the following Resources property value: **UI**. The HostProtectionAttribute does not affect desktop applications (which are typically started by double-clicking an icon, typing a command, or entering a URL in a browser). For more information, see the HostProtectionAttribute clas SQL Server Programming and Host Protection Attributes.

Examples

The following example demonstrates the standard formatting specifiers for numbers, dates, and enumerations.

C#

```
// This code example demonstrates the Console.WriteLine() method.
// Formatting for this example uses the "en-US" culture.

using System;
class Sample
{
    enum Color {Yellow = 1, Blue, Green};
    static DateTime thisDate = DateTime.Now;

    public static void Main()
    {
    Console.Clear();

// Format a negative integer or floating-point number in various ways.
    Console.WriteLine("Standard Numeric Format Specifiers");
    Console.WriteLine(
        "(C) Currency: . . . . . . . . {0:C}\n" +
        "(D) Decimal:. . . . . . . . . {0:D}\n" +
        "(E) Scientific: . . . . . . . {1:E}\n" +
        "(F) Fixed point:. . . . . . . {1:F}\n" +
        "(G) General:. . . . . . . . . {0:G}\n" +
        "    (default):. . . . . . . . {0} (default = 'G')\n" +
        "(N) Number: . . . . . . . . . {0:N}\n" +
        "(P) Percent:. . . . . . . . . {1:P}\n" +
        "(R) Round-trip: . . . . . . . {1:R}\n" +
        "(X) Hexadecimal:. . . . . . . {0:X}\n",
        -123, -123.45f);

// Format the current date in various ways.
    Console.WriteLine("Standard DateTime Format Specifiers");
    Console.WriteLine(
        "(d) Short date: . . . . . . . {0:d}\n" +
        "(D) Long date:. . . . . . . . {0:D}\n" +
        "(t) Short time: . . . . . . . {0:t}\n" +
        "(T) Long time:. . . . . . . . {0:T}\n" +
        "(f) Full date/short time: . . {0:f}\n" +
        "(F) Full date/long time:. . . {0:F}\n" +
        "(g) General date/short time:. {0:g}\n" +
        "(G) General date/long time: . {0:G}\n" +
        "    (default):. . . . . . . . {0} (default = 'G')\n" +
        "(M) Month:. . . . . . . . . . {0:M}\n" +
        "(R) RFC1123:. . . . . . . . . {0:R}\n" +
        "(s) Sortable: . . . . . . . . {0:s}\n" +
        "(u) Universal sortable: . . . {0:u} (invariant)\n" +
        "(U) Universal full date/time: {0:U}\n" +
        "(Y) Year: . . . . . . . . . . {0:Y}\n",
        thisDate);

// Format a Color enumeration value in various ways.
    Console.WriteLine("Standard Enumeration Format Specifiers");
    Console.WriteLine(
        "(G) General:. . . . . . . . . {0:G}\n" +
        "    (default):. . . . . . . . {0} (default = 'G')\n" +
        "(F) Flags:. . . . . . . . . . {0:F} (flags or integer)\n" +
        "(D) Decimal number: . . . . . {0:D}\n" +
        "(X) Hexadecimal:. . . . . . . {0:X}\n",
        Color.Green);
    }
}
/*
This code example produces the following results:

Standard Numeric Format Specifiers
(C) Currency: . . . . . . . . ($123.00)
(D) Decimal:. . . . . . . . . -123
(E) Scientific: . . . . . . . -1.234500E+002
(F) Fixed point:. . . . . . . -123.45
(G) General:. . . . . . . . . -123
    (default):. . . . . . . . -123 (default = 'G')
(N) Number: . . . . . . . . . -123.00
(P) Percent:. . . . . . . . . -12,345.00 %
(R) Round-trip: . . . . . . . -123.45
(X) Hexadecimal:. . . . . . . FFFFFF85

Standard DateTime Format Specifiers
(d) Short date: . . . . . . . 6/26/2004
(D) Long date:. . . . . . . . Saturday, June 26, 2004
(t) Short time: . . . . . . . 8:11 PM
(T) Long time:. . . . . . . . 8:11:04 PM
(f) Full date/short time: . . Saturday, June 26, 2004 8:11 PM
(F) Full date/long time:. . . Saturday, June 26, 2004 8:11:04 PM
(g) General date/short time:. 6/26/2004 8:11 PM
(G) General date/long time: . 6/26/2004 8:11:04 PM
    (default):. . . . . . . . 6/26/2004 8:11:04 PM (default = 'G')
(M) Month:. . . . . . . . . . June 26
(R) RFC1123:. . . . . . . . . Sat, 26 Jun 2004 20:11:04 GMT
```

```
(s) Sortable: . . . . . . . . 2004-06-26T20:11:04
(u) Universal sortable: . . . 2004-06-26 20:11:04Z (invariant)
(U) Universal full date/time: Sunday, June 27, 2004 3:11:04 AM
(Y) Year: . . . . . . . . . . June, 2004

Standard Enumeration Format Specifiers
(G) General:. . . . . . . . . Green
    (default):. . . . . . . . Green (default = 'G')
(F) Flags:. . . . . . . . . . Green (flags or integer)
(D) Decimal number: . . . . . 3
(X) Hexadecimal:. . . . . . . 00000003

*/
```

The following example illustrates the use of the **WriteLine** method. It assumes that the amount of a restaurant bill is passed to the example as a command-line argument.

C#

```
using System;

public class TipCalculator {
    private const double tipRate = 0.18;
    public static int Main(string[] args) {
        double billTotal;
        if (args.Length == 0) {
            Console.WriteLine("usage: TIPCALC total");
            return 1;
        }
        else {
            if (! Double.TryParse(args[0], out billTotal)) {
                Console.WriteLine("usage: TIPCALC total");
                return 1;
            }
            double tip = billTotal * tipRate;
            Console.WriteLine();
            Console.WriteLine("Bill total:\t{0,8:c}", billTotal);
            Console.WriteLine("Tip total/rate:\t{0,8:c} ({1:p1})", tip, tipRate);
            Console.WriteLine(("").PadRight(24, '-'));
            Console.WriteLine("Grand total:\t{0,8:c}", billTotal + tip);
            return 0;
        }
    }
}

/*
>tipcalc 52.23

Bill total:       $52.23
Tip total/rate:    $9.40 (18.0 %)
------------------------
Grand total:      $61.63
*/
```

Version Information

.NET Framework
Supported in: 4.5.2, 4.5.1, 4.5, 4, 3.5, 3.0, 2.0, 1.1, 1.0
.NET Framework Client Profile
Supported in: 4, 3.5 SP1
.NET for Windows Phone apps
Supported in: Windows Phone 8, Silverlight 8.1

.NET Framework Security

- UIPermission
  for modifying safe top-level windows and subwindows. Associated enumeration: UIPermissionWindow.SafeTopLevelWindows

Platforms

Windows Phone 8.1, Windows Phone 8, Windows 8.1, Windows Server 2012 R2, Windows 8, Windows Server 2012, Windows 7, Windows Vista SP2, Windows Server 2008 (Server Co
Role not supported), Windows Server 2008 R2 (Server Core Role supported with SP1 or later; Itanium not supported)

The .NET Framework does not support all versions of every platform. For a list of the supported versions, see .NET Framework System Requirements.

◢ See Also

Reference
Console Class
WriteLine Overload
System Namespace
Read
ReadLine
Write
Other Resources
Formatting Types
Composite Formatting

Contribuições da comunidade