Console.WriteLine Method

Reference

Definition

Namespace: System

Assembly: System.Console.dll

Writes the specified data, followed by the current line terminator, to the standard output stream.

Overloads

Expand table

WriteLine(String, Object, Object)	Writes the text representation of the specified objects, followed by the current line terminator, to the standard output stream using the specified format information.
WriteLine(String)	Writes the specified string value, followed by the current line terminator, to the standard output stream.
WriteLine(Char[], Int32, Int32)	Writes the specified subarray of Unicode characters, followed by the current line terminator, to the standard output stream.
WriteLine(String, Object[])	Writes the text representation of the specified array of objects, followed by the current line terminator, to the standard output stream using the specified format information.
WriteLine(String, Object)	Writes the text representation of the specified object, followed by the current line terminator, to the standard output stream using the specified format information.
WriteLine(UInt64)	Writes the text representation of the specified 64-bit unsigned integer value, followed by the current line terminator, to the standard output stream.
WriteLine(UInt32)	Writes the text representation of the specified 32-bit unsigned integer value, followed by the current line terminator, to the standard output stream.
WriteLine(Single)	Writes the text representation of the specified single-precision floating-point value, followed by the current line terminator, to the standard output stream.

WriteLine(Double)	Writes the text representation of the specified double-precision floating-point value, followed by the current line terminator, to the standard output stream.
WriteLine(Int64)	Writes the text representation of the specified 64-bit signed integer value, followed by the current line terminator, to the standard output stream.
WriteLine(Int32)	Writes the text representation of the specified 32-bit signed integer value, followed by the current line terminator, to the standard output stream.
WriteLine(Decimal)	Writes the text representation of the specified Decimal value, followed by the current line terminator, to the standard output stream.
WriteLine(Char[])	Writes the specified array of Unicode characters, followed by the current line terminator, to the standard output stream.
WriteLine(Char)	Writes the specified Unicode character, followed by the current line terminator, value to the standard output stream.
WriteLine(Boolean)	Writes the text representation of the specified Boolean value, followed by the current line terminator, to the standard output stream.
WriteLine()	Writes the current line terminator to the standard output stream.
WriteLine(String, Object, Object, Object)	Writes the text representation of the specified objects, followed by the current line terminator, to the standard output stream using the specified format information.
WriteLine(Object)	Writes the text representation of the specified object, followed by the current line terminator, to the standard output stream.

Remarks

The default line terminator is a string whose value is a carriage return followed by a line feed ("\r\n" in C#, or `vbCrLf` in Visual Basic). You can change the line terminator by setting the TextWriter.NewLine property of the Out property to another string.

WriteLine(String, Object, Object)

Source: Console.cs

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

C#

```csharp
public static void WriteLine (string format, object? arg0, object? arg1);
```

Parameters

`format` String

A composite format string.

`arg0` Object

The first object to write using `format`.

`arg1` Object

The second object to write using `format`.

Exceptions

IOException

An I/O error occurred.

ArgumentNullException

`format` is `null`.

FormatException

The format specification in `format` is invalid.

Examples

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

C#

```csharp
// 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 is a tip calculator that calculates an 18% tip and uses the WriteLine method to display the amount of the original charge, the amount of the tip, and the total amount. The example is a console application that requires the user to supply the amount of the original charge as a command-line parameter.

C#

```
using System;

public class TipCalculator
{
    private const double tipRate = 0.18;
    public static void Main(string[] args)
    {
        double billTotal;
        if (args.Length == 0 || ! Double.TryParse(args[0], out
```