

第12期-优雅的字符串内插法

2022年3月23日 9:56

字符串内插法格式

`$"{<interpolationExpression>[,<alignment>][:<formatString>]}"`

元素	描述
<code>interpolationExpression</code>	生成需要设置格式的结果的表达式。 null 的字符串表示形式为 String.Empty 。
<code>alignment</code>	常数表达式，它的值定义表达式结果的字符串表示形式中的最小字符数。如果值为正，则字符串表示形式为右对齐；如果值为负，则为左对齐。 请参阅 对齐组件 。
<code>formatString</code>	受表达式结果类型支持的格式字符串。 请参阅 格式字符串组件 。

字符串中使用转义序列

从 C# 8.0 开始，`$@"..."` 和 `@$"..."` 均为有效的内插逐字字符串。@ 特殊字符用作原义标识符。

内插字符串内支持三元运算符、方法调用、Switch 表达式等。

创建区域性特定的字符串

默认情况下，内插字符串将 `CultureInfo.CurrentCulture` 属性定义的当前区域性用于所有格式设置操作，使用内插字符串到 `System.FormattableString` 实例的隐式转换，并调用它的 `ToString(IFormatProvider)` 方法来创建区域性特定的结果字符串。

```
var cultureInfo = System.Globalization.CultureInfo.GetCultureInfo("en-US");
```

```
FormattableString formattableString = $"This date is {598,3:C}";
```

```
string str = formattableString.ToString(cultureInfo);
```

与语言文化无关的字符串格式化。

```
string str2 = FormattableString.Invariant($"This date is {598,3:C}");
```

使用原始字符串文本

可以使用 .NET 7 SDK 试用这些功能。或者，如果你有 .NET SDK 6.0.200 或更高版本，则可以将 csproj 文件中的 <LangVersion> 元素设置为 preview。

```
<LangVersion>preview</LangVersion>
```

```
int safetyScore = 88;
```

```
string message = $"The usage policy for {safetyScore} is {  
    safetyScore switch  
    {  
        > 90 => "Unlimited usage",  
        > 80 => "General usage, with daily safety check",  
        > 70 => "Issues must be addressed within 1 week",  
        > 50 => "Issues must be addressed within 1 day",  
        _ => "Issues must be addressed before continued use",  
    }  
}";
```

```
int X = 2;  
int Y = 3;
```

```
var pointMessage = $""The point "{X}, {Y}" is {Math.Sqrt(X * X + Y * Y)} from the  
origin"";
```

```
Console.WriteLine(pointMessage);  
// output: The point "2, 3" is 3.605551275463989 from the origin.
```

```
var pointMessage = $$""The point {{{X}}, {{{Y}}}} is {{{Math.Sqrt(X * X + Y * Y)}}  
from the origin"";  
Console.WriteLine(pointMessage);  
// output: The point {2, 3} is 3.605551275463989 from the origin.
```

优雅的字符串拼接方案

[String.Concat](#) 方法

[String.Join](#) 方法

[String.Format](#) 方法

[StringBuilder](#) 类

[DbConnectionStringBuilder](#) 类

[Path.Join](#) 方法

[Path.Combine](#) 方法

[QueryHelpers.AddQueryString](#) 方法

内插字符串编译

内置 `DefaultInterpolatedStringHandler`：提供语言编译器用来将内插字符串处理到 `String` 实例中的处理程序，可反编译查看。

如果内插字符串类型为 `string`，则通常将其转换为 `String.Format` 方法调用。如果分析的行为等同于串联，则编译器可将 `String.Format` 替换为 `String.Concat`。

如果内插字符串类型为 `IFormattable` 或 `FormattableString`，则编译器会生成对 `FormattableStringFactory.Create` 方法的调用。

从 C# 10 开始，使用内插字符串时，编译器将检查内插字符串是否被分配给满足内插字符串处理程序模式要求的类型。内插字符串处理程序是一种自定义类型，可将内插字符串转换为字符串。内插字符串处理程序是一种高级方案，通常出于性能原因使用。可以在内插字符串改进的语言规范中了解生成内插字符串处理程序的要求。可以按照“C# 新增功能”部分中的内插字符串处理程序教程生成一个内插字符串处理程序。在 .NET 6 中，当对 `string` 类型的参数使用内插字符串时，内插字符串由 `System.Runtime.CompilerServices.DefaultInterpolatedStringHandler` 处理。

参考资料

字符串内插参考

<https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/tokens/interpolated>

C#语言版本控制

<https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/configure-language-version>

C# 中的字符串内插

<https://docs.microsoft.com/zh-cn/dotnet/csharp/tutorials/string-interpolation>

编写自定义字符串内插处理程序

<https://docs.microsoft.com/zh-cn/dotnet/csharp/whats-new/tutorials/interpolated-string-handler>