**C# 6.0 Language Features – String Interpolation**

Welcome to C# 6.0 language features article series. As you might have heard about Visual Studio 2015 and .Net framework 4.6 announcements in various Microsoft events earlier, one of the most exciting news was the next version of C# language, which is 6.0 and new features added to it. C# has come a long way since 2002 when C# 1.0 was released along with Visual Studio. Since then with every major release of .Net framework, C# became better and better language, which has tremendously helped application developers to minimize the code complexity and focus on business functionality. With the new version of .Net framework and Visual Studio around the corner, you can start learning new language features today using Visual Studio 2015 preview release. There are many useful features added to the language and I don’t know at the moment, how many articles I will write in this series, but it will probably something like this.

* Static using statements
* Auto property initializer
* Null conditional operator
* **String interpolation – This Article**
* nameof operator
* Declaration expressions

**String interpolation – What is it?**

It is a process of evaluating a string literal containing one or more placeholders, yielding a result in which the placeholders are replaced with corresponding values. If you have used Console.WriteLine method earlier for string concatenation, you might think this is a not a new feature to the language. Well C# 6.0 simply adds syntactic sugar to string interpolation feature, which helps developer to write more readable code.

Let’s perform string concatenation using old way and then we will refactor it using C# 6.0. Below code snippet should be easy to understand. We just have a placeholders {0}, {1}, which gets replaced with actual values of product instance variable properties [Id, Name and Price] at runtime.

Console.WriteLine(("Id: {0}, Name: {1}, Price: {2}", product.Id, product.Name, product.Price);

Above code snippet can easily get complex if you have too many placeholders and you have to carefully provide exact number of values using correct sequence or else you will get an incorrect result. C# 6.0 solves this problem by allowing user to provide string expression directly into string literal. Let’s refactor above code using C# 6.0

Console.WriteLine("Id: \{product.Id}, Name: \{product.Name}, Price: \{product.Price}");

Again, it’s fairly easy to understand above code. Instead of using placeholder values like {0}, {1} and so forth, we can now directly embed the expression into string literal. In my opinion, this makes code more readable as you don’t have to match the values as in with the old syntax.

Additionally, you can also embed a conditional statement in the string literal as shown in below

Console.WriteLine("Product Price \{product.Price} \{product.Price = 1 ? "dollar" : "dollars"}");

I hope this will help you to write cleaner and less error prone code once C# 6 comes out with Visual Studio 2015.

Exciting time ahead, so stay tuned!