**C# 6.0 Language Features – Null conditional operator**

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 – This article**
* String interpolation
* nameof operator
* Declaration expressions

**Null conditional operator**

If you’re an experienced C# developer, it’s most likely that you’re familiar with NullReferenceException. This exception indicates a bug in C# program that you did not perform sufficient null checking condition before execution. Although it’s very trivial to check whether the instance is null or not, most of the time these checks simply adds noise to your code. C# 6.0 includes a new null-conditional operator that helps you write these checks without adding any noise and in a cleaner format.

Let’s assume that you want to display a list of customer details with address information, however not all the customer contains address information. In such case you need to check whether the address information is null or not. Before C# 6.0, it can be implemented as shown in below code snippet

if (cust.Address != null)

{

Console.WriteLine("Name : {0} \t Address : {1}", cust.Name, cust.Address.City);

}

else

{

Console.WriteLine("Name : {0} \t Address : {1}", cust.Name, string.Empty);

}

However, just notice the way you need to handle the null check and use if-else block to execute the code depending on the instance value. In C# 6.0, same code can be implemented using null conditional operator **?.**

Console.WriteLine("Name : {0} \t Address : {1}", cust.Name, cust.Address?.City);

**Null conditional operator usage in delegate invocation**

Null conditional operator really helps you to resolve an idiosyncrasy of C# that has existed since C# 1.0, which is to check for null before invoking delegates.

The OnWorkPerformed delegate handler is checking whether the event has at least one subscriber associated to it. Code snippet also assigns delegate instance to a local variable localHandler for thread safety.

public delegate void WorkPerformedHandler(int hours, WorkType workType);

public class Worker

{

public event WorkPerformedHandler WorkPerformed;

public void DoWork(int hours, WorkType type)

{

OnWorkPerformed(hours, type);

}

private void OnWorkPerformed(int hours, WorkType type)

{

WorkPerformedHandler localHandler = WorkPerformed;

if (localHandler != null)

{

localHandler(hours, type);

}

}

}

With the null conditional operator, same code can be implemented as shown below

private void OnWorkPerformed(int hours, WorkType type)

{

WorkPerformed?.Invoke(hours, type);

}

Sweet, isn’t it? ☺

Happy coding!