C# 6.0 LANGUAGE FEATURES – Auto Property Initializers

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.

1. Static using statements
2. **Auto property initializers – This article**
3. Primary constructors
4. Null conditional operator
5. String interpolation
6. nameof operator
7. Declaration expressions

**Auto Property Initializers**

Auto property feature was added to the language during C# 3.0 release. It allows you to define a property without any backing field, however you still need to use constructor to initialize these auto properties to non-default value. C# 6.0 introduces a new feature called **auto property initializer** which allows you to initialize these properties without a constructor.

Code snippet shown below defines a class CustomerV1 with one auto property – City. A parameterized constructor has been defined to initialize this property to some value.

public class CustomerV1

{

public string City { get; private set; }

public Customer(string city)

{

this.City = city;

}

}

In C# 6.0, same code can be refactored as shown below. Note that, since we have initialized City property to a default value, we don’t need to explicitly specify the setter in the property.

public class CustomerV2

{

public string City { get; } = "London";

}

Although it seems a minor feature, I still believe it will help you to reduce the noise within your code.

Thanks for reading.