# Quality Issue #1 – Count vs. Count()

A measure of how good you are as a developer is how well you can write code. This is a start of a series of posts to help developers write better code. Today we will look at **Count** vs. **Count()** in .NET.

## Count

There are many collections in .NET that support a property called **Count**. This includes **List**, **List<T>**, **HashSet<T>**, and many more. The **Count** property is a value the represents the number of elements in the collection.

Here is the documentation for the **List<T>.Count** property.

[List<T>.Count Property (System.Collections.Generic) | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.count?view=net-5.0)

## Count()

.NET supports two interfaces, **IEnumerable** and **IEnumerable<T>**, which provide the ability to iterate over a collection. Along with these interfaces are a set of static extension methods defined in Enumerable which add functionality for querying a collection of objects based on LINQ. One of those methods is **Count()** which iterates over the collection to determine the number of items in the collection.

Here is the documentation for the **IEnumerable** interface and **Enumerable.Count** method.

[IEnumerable Interface (System.Collections) | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/api/system.collections.ienumerable?view=net-5.0)

[IEnumerable<T> Interface (System.Collections.Generic) | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerable-1?view=net-5.0)

[Enumerable.Count Method (System.Linq) | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/api/system.linq.enumerable.count?view=net-5.0)

## Comparison

So why do we care about the difference between **Count** and **Count()**? One simply reads a value in memory to determine the count of the elements in a collection and the other iterates over the entire collection in memory to determine the count of the number of items.

There is a big performance difference when it comes to these two approaches. What is worse is that the performance difference gets worse when the size of the collection grows. These might now seem like a big deal, but it does add up over time. If you have a large code base or high scale application, you will begin to see the impact over time.

Here is an example of code that gets a collection of people using **List<Person>**. We then use **Count** and **Count()** to get the number of people in the collection.

Text

Description automatically generated

We measure the performance difference between the two approaches. One can see that **Count()** extension method takes 7 times longer than using the **Count** property. This gets worse when the number of items in the collection increases.

Text

Description automatically generated

If you don’t believe that this is a quality issue, check out the code analysis tip in Visual Studio by hovering over the **Count()** method. There you will see code analysis rule **CA1829**.

Graphical user interface, text, application

Description automatically generated

The description of rule **CA1829** provides a clear reason as to why not to use the **Count()** method.

“The Count LINQ method was used on a type that supports an equivalent, more efficient Length or Count property.”

Here is the documentation for the **Code Analysis Performance Rule CA1829**.

[CA1829: Use Length/Count property instead of Enumerable.Count method (code analysis) - .NET | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/fundamentals/code-analysis/quality-rules/ca1829)

## Anecdote

I remember being one of a four architects on [Fidelity's Active Trader Pro](https://mill5-my.sharepoint.com/personal/rich_mill5_com/Documents/Fidelity's%20Active%20Trader%20Pro). This is amazing product put together by about 80 to 90 awesome developers. **Count** vs. **Count()** was one of the performance problems we would find in our code reviews. An even bigger challenge was the overuse of LINQ queries in a fluent style programming syntax. Finding bad LINQ queries was a large part of our performance optimization during the project. This leads me to one of my favorite things to tell developers, “LINQ is convenient not performant”. Interestingly, I am on a project at the moment where we are addressing quality issues such as **Count** vs. **Count()**.

## Conclusion

It is out hope that you have learned the proper the use of **Count** vs. **Count()** and that this is the beginning of your journey to improve the quality of the code that you write.

## Appreciation

Thanks to our friends at [MILL5](https://www.mill5.com/) for sponsoring this article.

## Author(s):

Richard Crane, Founder/CTO

## Disclaimer:

All source code is licensed under the Apache 2.0 license.

## References:

Count vs. Count() Code Example

<https://github.com/MILL5/quality/tree/main/fundamentals/Count>