1. LINQ DefaultIfEmpty Method in C#:

The DefaultIfEmpty method in LINQ is used in a sequence of values to return a default value if the sequence is empty. This method is commonly used with other operators, such as Select and SelectMany, to ensure that a query returns a result even if the source sequence contains no elements.

If the sequence or data source on which the DefaultIfEmpty method is called is not empty, then the original sequence or data source values will be returned. On the other hand, if the sequence or data source is empty, it returns a sequence with the default values based on the data type. There are two overloaded versions available for this DefaultIfEmpty method in LINQ. They are as follows.

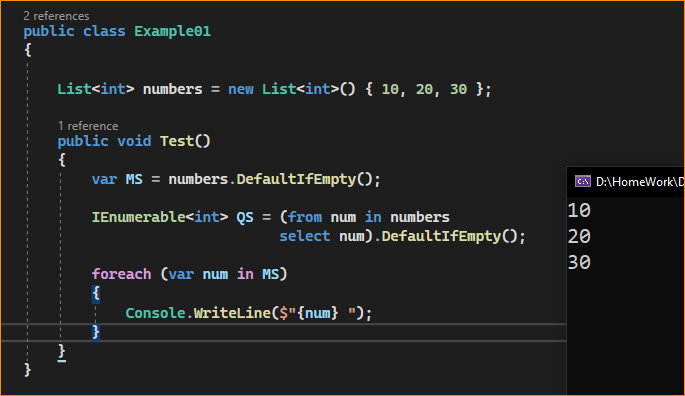


The first overloaded version does not take any parameter, and in this case, if the sequence is empty, it will return the default values based on the data type. That means this method returns the elements of the specified sequence or the type parameter’s default value if the sequence is empty.

You can pass the default value in the second overloaded version of the DefaultIfEmpty method. If the sequence is empty, then this default value (what you pass to the method) will be returned by the DefaultIfEmpty method. That means this method returns the elements of the specified sequence or the specified value if the sequence is empty.

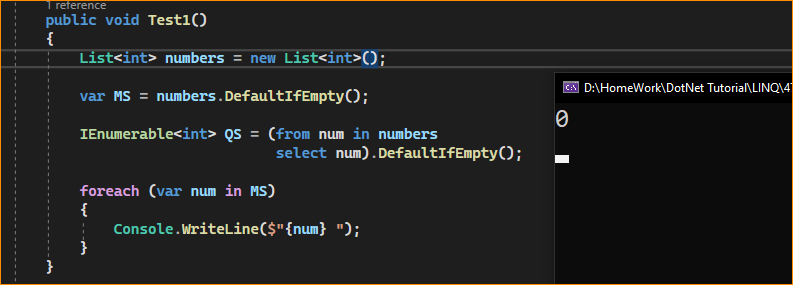
1. Example to Understand LINQ DefaultIfEmpty Method in C#:

Let us see an example of the LINQ DefaultIfEmpty Method in C# using both Method and Query Syntax. In the following example, the sequence is not empty. So, it is going to return a copy of the original values.



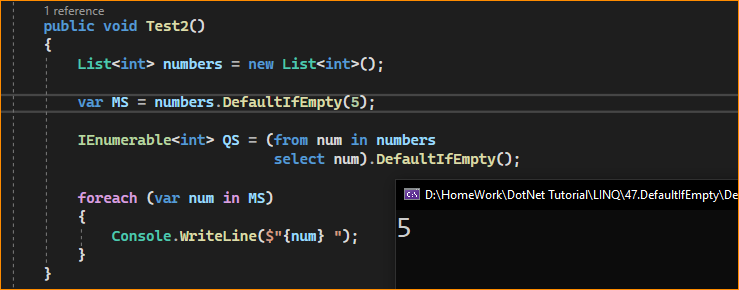
1. Example to Understand DefaultIfEmpty Method when Sequence is Empty:

Let us see an Example to Understand the DefaultIfEmpty Method when the Sequence is Empty. In the below example, the sequence is empty. So, in this case, it will return 0 as the default value. This is because 0 is the default value for the integer data type.



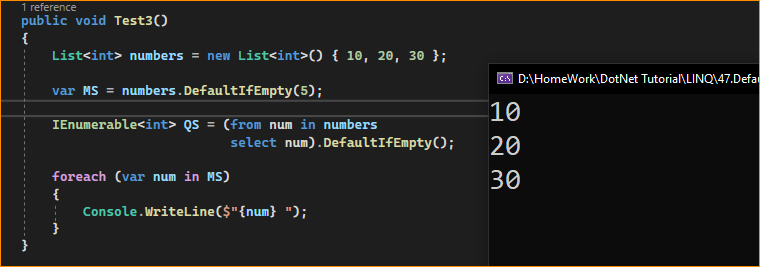
1. How to Supply User-Given Values when the Sequence is Empty?

In the following example, the sequence is empty, but we have supplied a default value (i.e., 5) to the DefaultIfEmpty method. So, in this case, the default value that we supplied (5) will be returned by the DefaultIfEmpty method.



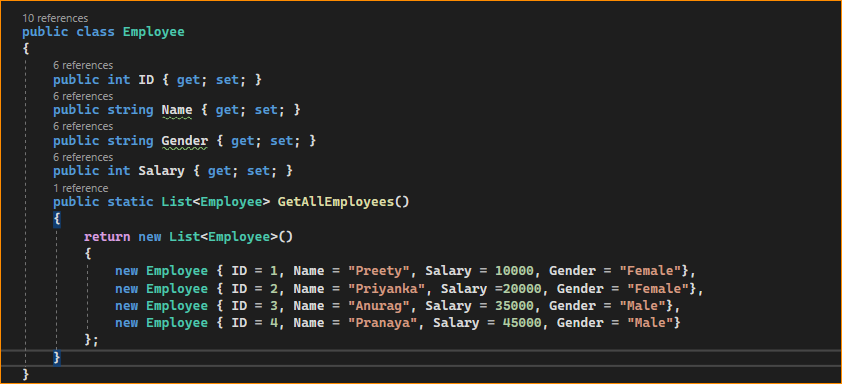
1. What Happens if the Sequence is not Empty and we have Supplied a Value to the DefaultIfEmpty method?

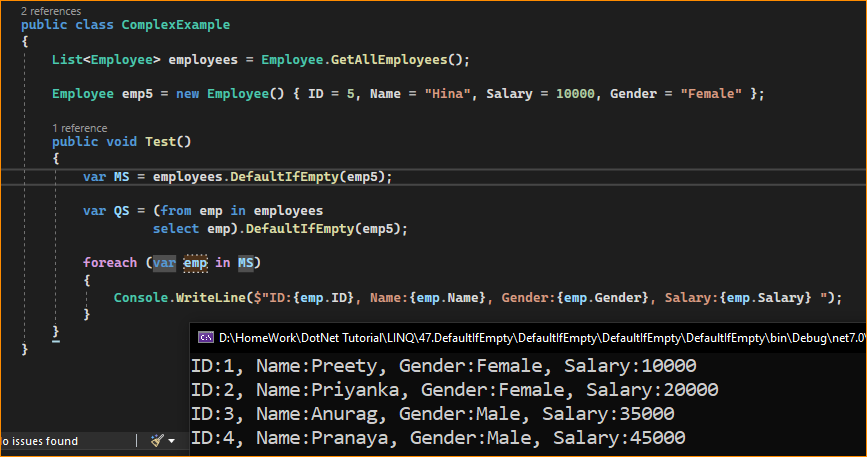
If we supplied a default value, but the sequence is not empty, then, in that case, the original values present in the sequence will be returned. In the below example, we have supplied a default value, i.e., 5, to the DefaultIfEmpty method, but the sequence is not empty. So, in this case, the elements present in the sequence will be returned.



1. LINQ DefaultIfEmpty Method with Complex Type in C#:

Let us see an example of how to use the LINQ DefaultIfEmpty Method with Complex Type in C#. For this, we are going to use the following Employee class. So, first, create a class file with the name Employee.cs and then copy and paste the following code into it. It is a simple class with 4 properties and one method to return a collection of employees.

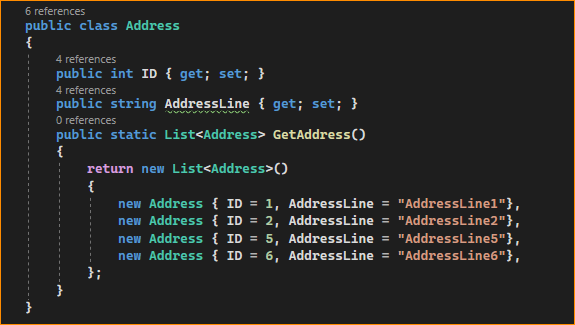


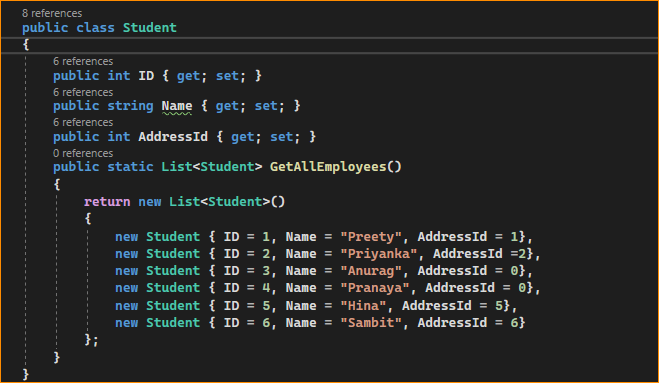


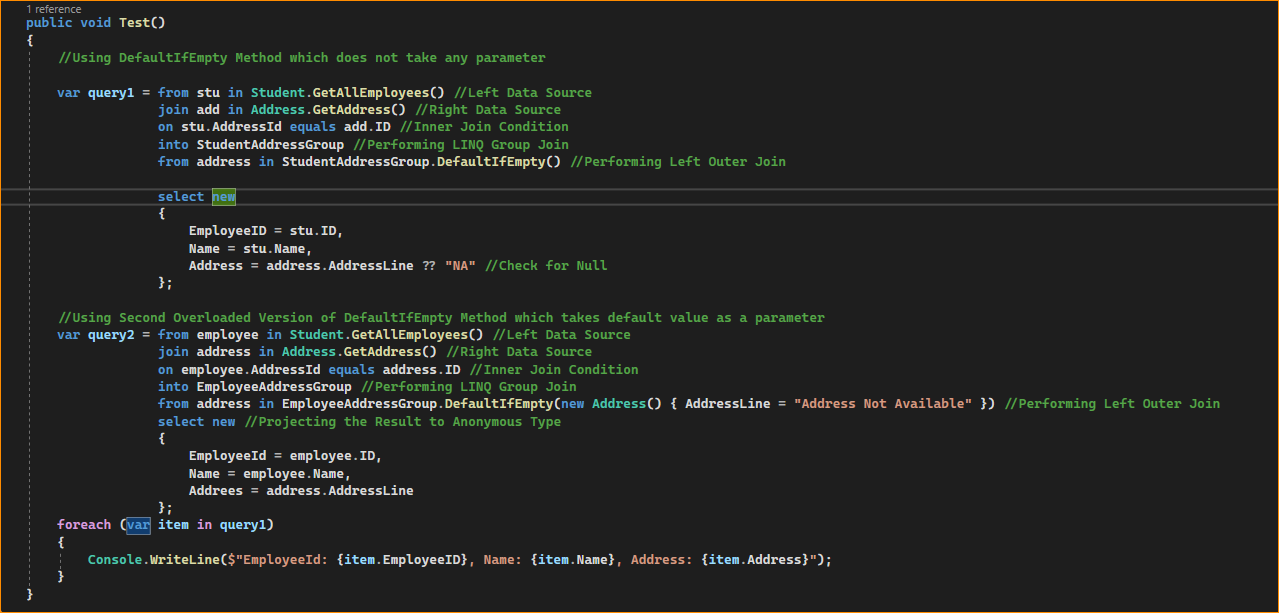


1. LINQ DefaultIfEmpty in Left Outer Join:

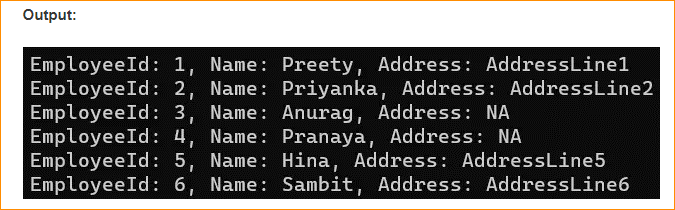
DefaultIfEmpty is particularly useful in the context of left outer joins in LINQ. It ensures that, if there are no matching elements in the join, the result will still include the elements from the left sequence, paired with a default value for the right sequence. Here’s an example of how it might be used in a left outer join:







Result –



In this join, if an Employee has no matching Address in the Addresses collection, DefaultIfEmpty will insert a default value (null in this case), and the ?? operator is used to provide a fallback value (“NA”) when rendering the result. In the second overloaded version. we have specified a default when the Address is not available.

1. When to Use the LINQ DefaultIfEmpty Method in C#?

You should use the LINQ DefaultIfEmpty method in C# to ensure that you always have at least one element in the result of a LINQ query, even if the source collection might be empty. This method is particularly useful in the following scenarios:

* Left Outer Joins: LINQ has no explicit left outer join operation, but you can simulate it using a group join (GroupJoin) followed by SelectMany and DefaultIfEmpty. This ensures that for every element in the outer sequence if there are no matching elements in the inner sequence, you still get a result with the outer element and a default value for the inner element.
* Providing Default Values: When you have an empty sequence, you need to ensure that the subsequent query operators have a value to work with. DefaultIfEmpty allows you to specify a default value that can be more meaningful in context than a simple null or zero.
* Coalescing Empty Results: In data processing, when an empty sequence would cause issues down the line, using DefaultIfEmpty can coalesce the empty result to a single default item, allowing for smoother processing and avoiding the need for additional null-checking.
* Fallback Scenarios: In cases where you expect a sequence always to have data, but due to some exceptional condition, it doesn’t, DefaultIfEmpty can be used to provide a fallback value, which can indicate the absence of expected data.
* Ensuring Collection Materialization: When using certain LINQ providers (like Entity Framework), an empty collection might not materialize as an instance but remain null. Using DefaultIfEmpty can ensure that you always get an IEnumerable instance, even if it contains only a default value, which can prevent NullReferenceException.
* Single Element Processing: If you want to ensure that a single-element operation (like Single or First) does not throw an exception on an empty sequence, preface it with DefaultIfEmpty. It guarantees at least one element to process, which will be the default value if the sequence is otherwise empty.

It’s important to note that DefaultIfEmpty will produce a sequence with a single element, a default value of the type of elements in the sequence if the original sequence is empty. If the sequence is not empty, DefaultIfEmpty has no effect, and the original sequence is returned unchanged.