1. LINQ First and FirstOrDefault Methods in C#:

The LINQ First and FirstOrDefault methods in C# are used to retrieve the first element of a sequence. They are part of the System.Linq namespace can be applied to any type implementing IEnumerable<T>. These methods are similar but have a key difference in handling empty sequences.

1. LINQ First Method in C#:

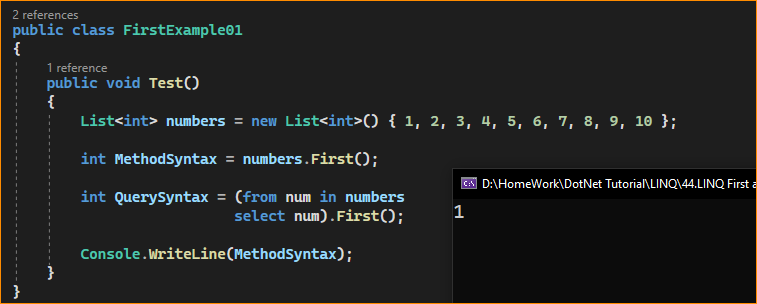
The LINQ First Method in C# returns the first element from a data source or a collection. If the data source or collection is empty, or if we specified a condition and with that condition, no element is found in the data source. The LINQ First method will throw an InvalidOperationException. If the Data Source is Null, then it will throw ArgumentNullException. Two overloaded versions are available for the LINQ First method, as shown in the image below.

As you can see, the first overloaded version does not take any parameter and returns the first element from the data source. The second overloaded version takes a predicate as a parameter, using which we can specify a condition, and then it will return the first element that satisfies the specified condition. If no element satisfies the given condition, it throws an InvalidOperationException.

**Note: First is used when you expect the sequence to have at least one element and you want to retrieve the first one. If the sequence is empty, it throws an InvalidOperationException.**

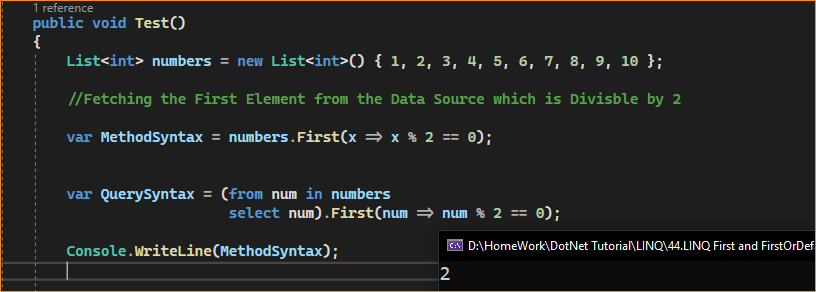
1. Example to Understand LINQ First Method in C#:

Let us see an example of understanding the LINQ First Method in C# with Method and Query Syntax. For a better understanding, please have a look at the following example. Here, we have created one data source, i.e., collection, that contains integer numbers from 1 to 10. Our requirement is to fetch the first element from the data source, i.e., in the output, we want to display the value 1 as it is the first element in the data source. Here, we fetch the first element from the collection using the LINQ First Method. No such operator is first available to write the Query Syntax, and If you want, you can combine both the method syntax and query syntax to write the code, as shown in the example below.



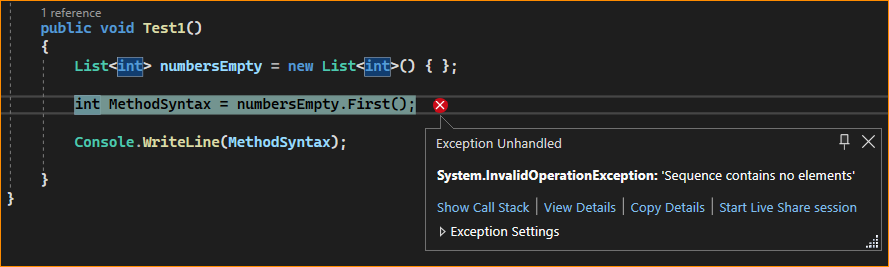
1. Example to Understand LINQ First Method, which takes Predicate as a Parameter in C#

Now, we need to fetch the first element from the divisible data source by 2. Here, we need to use the second overloaded version of the First method to specify our condition. For a better understanding, please have a look at the following example. The following program uses the second overloaded version to return the first element from the data source divisible by 2. See, many elements satisfy the given condition, but the first element that satisfies the given condition will be returned by the First method. In this case, the element will be 2. In the example below, I show how to use the second overloaded version of the LINQ First method in C# using both Method and Query Syntax.



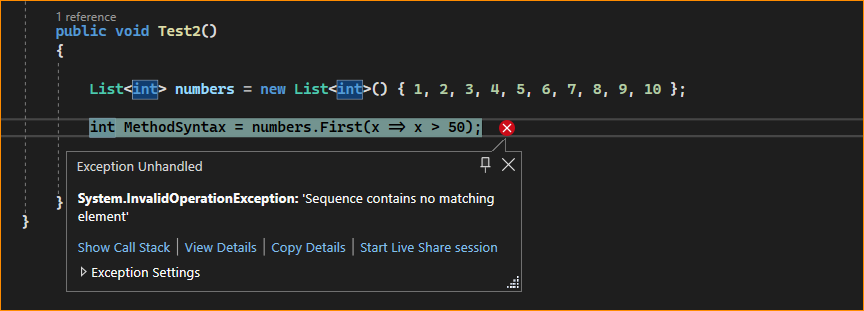
1. What Happens when we call the First Method on an Empty Data Source?

The LINQ First Method will throw an InvalidOperationException whenever the data source is empty. For a better understanding, please have a look at the following example. In the example below, we call the First method on an Empty Data Source; hence, when we run the following code, we will get one Runtime Exception.



1. What Happens If the Specified Condition in the First Method Does not Return any Data?

If the specified condition does not return any data, the LINQ First Method will throw an InvalidOperationException. For a better understanding, please have a look at the following example. In the example below, we call the First method and specify the condition to fetch the first element greater than 50. As you can see, the data source does not contain any element greater than 50; hence, running the below code will throw an InvalidOperationException.



Note: If you don’t want that Invalid Operation Exception. Instead, if you want a default value based on the data type, you need to use the LINQ FirstOrDefault method. For the reference types, the default value is NULL, and for the value types, the default value depends on the actual data type. For example, for numeric data types, it is 0. For string data types, it is null, and for boolean data types, it is False.

1. LINQ FirstOrDefault Method in C#:

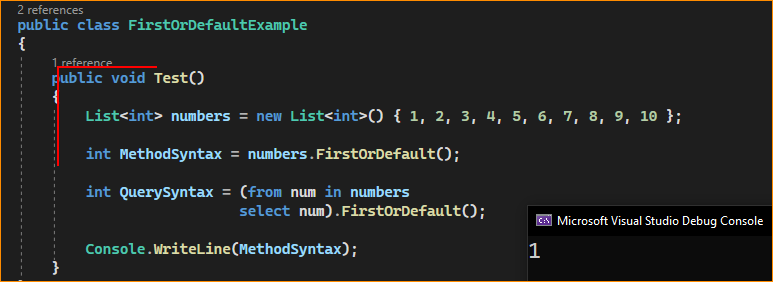
The LINQ FirstOrDefault Method in C# exactly does the same thing as the LINQ First method except that this method does not throw the InvalidOperationException exception when the data source is empty or when the specified condition does not match with any element in the data source. In such cases, it will return the default value based on the data type of the data source. If the Data Source is Null, then like the First Method, it will also throw ArgumentNullException. Like the First method, two overloaded versions are available for the FirstOrDefault method, as shown below.

First Overloaded Version: It returns the first element of a sequence or a default value if the sequence contains no elements.

First Overloaded Version: It returns the first element of the sequence that satisfies the given condition or a default value if no such element is found in the data source or if the data source is empty.

1. Example to Understand LINQ FirstOrDefault Method in C#.

Let us see an example to Understand the LINQ FirstOrDefault Method in C# with both Method and Query Syntax. Our requirement is to fetch the First Element from the Data Source using the FirstOrDefault Method. For a better understanding, please have a look at the following example. Here, we have created one data source that contains integer numbers from 1 to 10. Then, we need to fetch the first element from the collection using the FirstOrDefault method. There is no such operator called FirstOrDefault available to write the Query Syntax. You can combine the method syntax and query syntax to write the code, as shown in the example below.

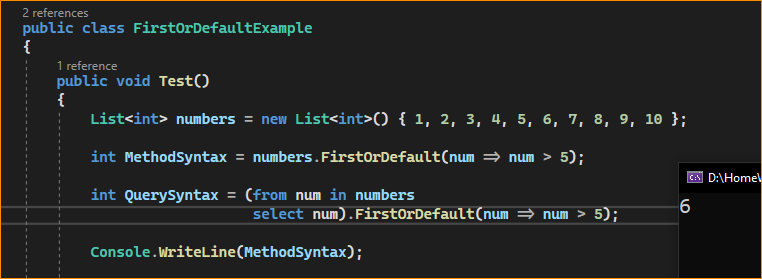


Note: FirstOrDefault is similar to First but returns a default value (like null for reference types, 0 for numeric types) instead of throwing an exception if the sequence is empty.

1. Example to Understand LINQ FirstOrDefault Method, which takes Predicate as a Parameter in C#

We need to fetch the first element from the data source greater than 5. Here, we need to use the second overloaded version of the FirstOrDefault method to specify our condition.

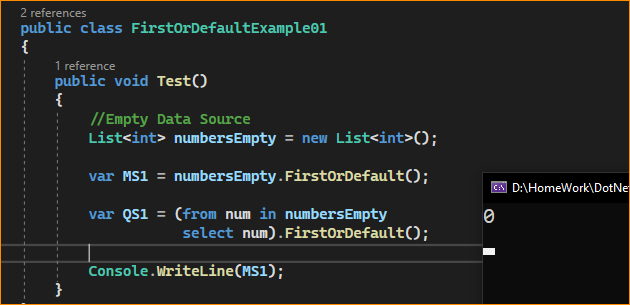
For a better understanding, please have a look at the following example. The following program uses the second overloaded version of the LINQ FirstOrDefault to return the first element from the data source, which is greater than 5. Many elements satisfy the given condition, but the first element satisfies the given condition is 6, which will be returned by the FirstOrDefault method. In the example below, I show how to use the second overloaded version of the LINQ FirstOrDefault method in C# using both Method and Query Syntax

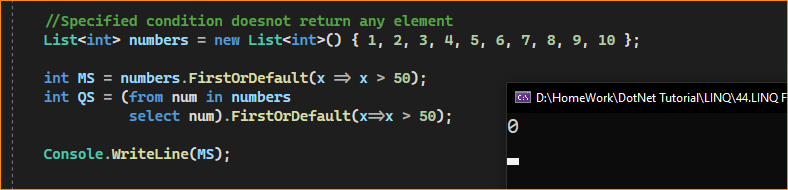


1. What Happens when we call the FirstOrDefault Method on an Empty Data Source or when the condition does not satisfy any element?

The LINQ **FirstOrDefault** Method will not throw an **InvalidOperationException** whenever the data source is empty. Instead, it will return a default value based on the data type. Similarly, if no elements are found with the given condition, it will not throw any exception. Instead, it will return the default value based on the data type.

For a better understanding, please have a look at the following example. In the below example, we have created two data sources. The first data source contains no item; we are trying to fetch the first element here. The second data source contains 10 elements from the numbers 1 to 10. We are trying to retrieve the first element greater than 50 from this data source.





Here, it will print the values as 0 and 0. This is because the data source contains integers. The default for integers is 0.

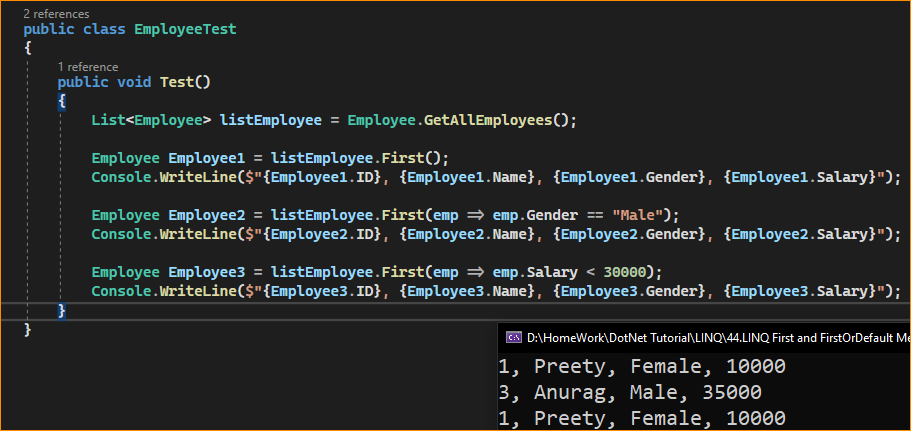
Key Points to Remember:

* Use First when you are sure the sequence contains at least one element and you want an exception to be thrown if it’s empty.
* Use FirstOrDefault when you want to avoid exceptions on empty sequences and are okay with getting a default value instead.
* Both methods can be used with or without a predicate to specify a condition for the element to meet.
* These methods are often used when you need to access the first element of a collection quickly or the first element that meets a certain condition and is particularly useful for querying data from in-memory collections or databases via LINQ to SQL or Entity Framework.

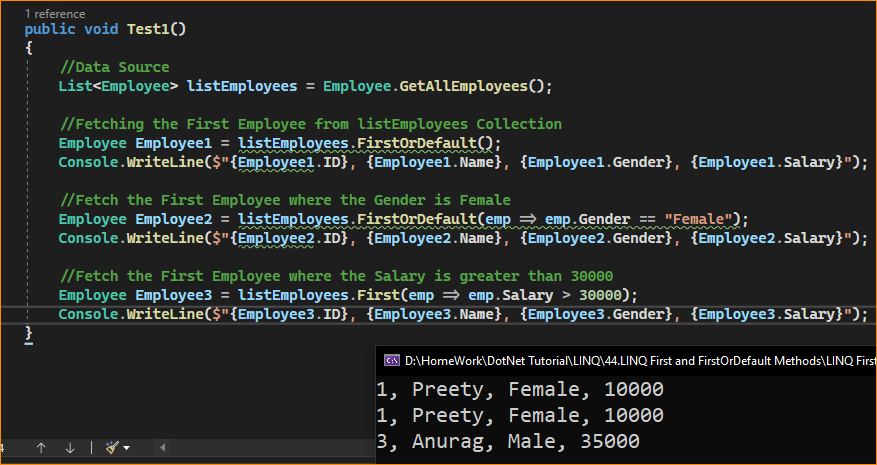
1. LINQ First and FirstOrDefault Method with Complex Type in C#:

Let us see an example of using LINQ First and FirstOrDefault 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.

First-



FirstOrDefault-



1. When to Use LINQ First and FirstOrDefault Methods in C#?

The LINQ First and FirstOrDefault methods in C# are used in different scenarios based on the nature of the data being queried and the operation’s specific requirements. Here’s when to use each:

1. When to Use LINQ First Method in C#:

Use First when:

* You Expect at Least One Element: When you’re confident that the collection contains at least one element that meets the condition, you want to retrieve it.
* You Want an Exception on Empty Sequences: If the sequence is empty or no element matches the condition, First throws an InvalidOperationException. Use it to handle such cases as exceptional conditions in your logic explicitly.
* Querying Non-empty Data Sets: In scenarios like database queries where the dataset is known to be non-empty and you need the first row of the result set.
* Ensuring Data Presence: In validation scenarios, where the absence of data should trigger an error or an exceptional condition.

1. When to Use LINQ FirstOrDefault Method in C#:

Use FirstOrDefault when:

* Empty Sequences are Common: When working with potentially empty collections or when an element might not meet the condition, and you want to handle such cases without exceptions.
* Default Value Handling: If you need a default value (like null for reference types, 0 for integers) in case of no matching element or an empty sequence.
* Optional Data Retrieval: In scenarios where the data is optional, the absence of data should not be considered an error.
* Safe Queries: When you prefer a safe approach that avoids exceptions if the sequence is empty or no element matches the condition.

Both methods can be used with or without a predicate. The choice between First and FirstOrDefault often depends on the specific requirements of your application logic, especially how you want to handle empty collections or the absence of elements that match a condition.

1. What is the Difference Between the First and FirstOrDefault Methods in LINQ?

Both First and FirstOrDefault methods in LINQ are used to return the first element from a data source. But if the data source is empty or the specified condition does not return any data, then the First method will throw an exception while the FirstOrDefault method will not throw an exception; instead, it returns a default value based on the element’s data type.

The difference between the LINQ methods First and FirstOrDefault in C# lies in how they handle empty sequences or sequences where no elements satisfy the given condition:

First Method:

* Behavior: Returns the first element of a sequence or the first element that satisfies a specified condition.
* Exception Handling: If the sequence is empty or no element satisfies the condition, First throws an InvalidOperationException.
* Use Case: Use First when you are certain that the sequence contains at least one element or at least one element meets the condition. It’s appropriate when an empty sequence or the absence of a matching element is considered an exceptional condition in your application logic.

FirstOrDefault Method:

* Behavior: Returns the first element of a sequence or the first element that satisfies a specified condition. If the sequence is empty or no element satisfies the condition, it returns the default value for the element type (null for reference types, 0 for numeric types, etc.).
* Exception Handling: Does not throw an exception for an empty sequence or if no element satisfies the condition.
* Use Case: Use FirstOrDefault when an empty sequence or the absence of a matching element is a normal or expected scenario. It is safer to avoid exceptions when you are unsure if the sequence contains any elements or if any element satisfies the condition.