

From Zero to LINQ (.NET Core)

Do you still find yourself writing a lot of loops, and you can't help but think there must be a better way? Well, you are probably right. LINQ can help you aggregate data, extract data from existing collections, compare data between collections, process XML, select data from a database, and much more. This seminar shows you common, and uncommon, examples where you might have used loops in the past, and how to translate those into LINQ queries. LINQ is very powerful and generally is much faster than using loops, so start using it today.

This course is for anyone who wants to learn to use LINQ syntax in their .NET applications to effectively query collections of data. In this two-day hands-on course, you learn the basics of LINQ such as selecting, ordering, and filtering data. You then move on to finding single elements and extracting data using the Take() and Skip() methods. The new Range operator in .NET Core is used as well as the new DistinctBy(), MinBy() and MaxBy().

After the basics are explored, you can then start looking at how to answer questions about what is in a collection, look for differences between collections and how to combine collections together. Just like the SQL language, you can also perform joins and subqueries using the LINQ syntax as well as grouping. Finally, you explore data aggregation, iterating over collections to perform an operation, and understanding how deferred execution works with LINQ. If time permits, you will be introduced to such technologies as LINQ to XML and using LINQ with the Entity Framework.

Learning Objectives

- Why LINQ is Important
- Selecting, filtering, and ordering data
- Finding single items in a collection
- Partition data using take and skip methods
- Aggregating data and determine if values exist
- How to iterate over collections
- Combining collections using unions and joins
- Comparing data within two collections
- Understanding deferred execution

Who Should Attend?

This is a technical VSLive! training seminar, and as such, is geared to developers, data engineers, and architects but can also be helpful to technology product management such as IT leaders, Chief Data Officers, and Chief Technology Officers. Coding experience is expected, and hands-on labs will be performed using the latest technologies.

Student Prerequisites

This course is designed for programmers who already have some experience with .NET and C#. You should be familiar with object-oriented programming and know how to create a console application using either Visual Studio or VS Code.

Hardware Requirements

A computer must be supplied by the student with the following technologies installed.

- Visual Studio 2022 or later
- or Visual Studio Code V1.5 or later
- .NET Core.x or later
- SQL Server (Developer Edition or higher) or SQL Server Express

Course Level

Introduction

Course Length

2 days

Module 1: Essentials of LINQ

Why use LINQ

LINQ to Objects and LINQ integrations

Differences between LINQ and writing loops

1.1: Selecting Data

Getting all data

Getting a subset of columns
Creating anonymous classes

1.2: Ordering Data

Ordering data ascending and descending
Using multiple fields for ordering

1.3: Filter Data Using Where

Filter data using the Where() method
Using multiple fields for filtering
Using an extension method
Adding multiple where clauses

1.4: Retrieving the First Item in a Collection

Using the FirstOrDefault() and First() methods

1.5: Retrieving the Last Item in a Collection

Using LastOrDefault() and Last() methods

1.6: Locating a Single Item in a Collection

Using SingleOrDefault() and Single() methods

1.7: Locate an Item Using an Indexer

Using the ElementAtOrDefault() and ElementAt() methods

1.8: Generate and Rearrange Data

Using the Range() and Repeat() methods to generate data
Use the Reverse() method to rearrange the collection in reverse order

Module 2: Partition Data

In this module you learn to partition collections of data into smaller data sets

2.1: Skipping Items in a Collection

Using Skip(), SkipLast(), and SkipWhile() methods

2.2: Grab a Range of Items in a Collection

Using Take(), TakeLast(), and TakeWhile() methods

Using the Take() method using the Range operator

2.3: Retrieve Unique Values in a Collection

Using Distinct() and DistinctBy() methods

2.4: Break a Collection into Smaller Arrays

Create an array of a collection using the Chunk() method

Module 3: Aggregating Data

In this module you learn to aggregate data to create sums, counts, averages, etc.

3.1: Counting, Summing and Averaging

Using the Count(), Sum() and Average() methods

3.2: Getting Minimum and Maximums using Min and Max

Using the Min() and Max() methods

3.3: Getting Minimum and Maximums using MinBy and MaxBy

Using the MinBy() and MaxBy() methods

3.4: Creating Custom Aggregators

Use the Aggregate() method to build your own accumulator

Learn how to make the Aggregate() method more efficient

Module 4: What is in a Collection

In this module you learn to determine if certain values exist within a collection

4.1: Do All Items Contain Something

Use All() to determine if all items meet a specific condition

4.2: Does Any One Item Contain Something

Use Any() to determine if any item meet a specific condition

4.3: Is There a Single Item in the Collection

Using Contains() to determine if an items exists in a collection

Using a class comparer

Module 5: Group Data

In this module you learn to create groups of data from within a collection

5.1: Using the GroupBy Query

Using the GroupBy() method

5.2: Using the 'Key' Property and 'into' Keyword

Using the into keyword

Ordering data by the 'Key' property

5.3: Using the Where Clause like a HAVING

Using the where clause to filter the grouped data

5.4: Using a Sub Query

Using a sub-query

Module 6: Iterating Over Collections

In this module you learn how to iterate over a collection to perform set operations within the collection

6.1: Iterating Over a Collection using ForEach

Using the ForEach() method

6.2: Iterating Over a Collection using a Sub Query

Using a sub-query

6.3: Iterating Over a Collection and Call a Method

Calling a method in the iterator

Module 7: Combining Collections

In this module you explore how to join and concatenate collections together

7.1: Joining Collections Together on a Common Key

Create an INNER JOIN between two collections

Join collections on more than one field

Simulate a LEFT OUTER JOIN

7.2: Combine Two Collections Together using Union

Combine two collections together using Union()

7.3: Combine Two Collections Together using UnionBy

Combine two collections together using UnionBy()

7.4: Concatenate Two Collections Together

Concatenate two collections together using Concat()

Module 8: Comparing Data in Collections

In this module you learn various methods to determine the similarities and differences between two collections

8.1: Determine if Collections are Equal using SequenceEqual

Using the Sequence() methods

Using a comparer class

8.2: Find Items in One Collection, But Not the Other using Except

Using the Except() methods

8.3: Find Items in One Collection, But Not the Other using ExceptBy

Using the ExceptBy() methods

Using a comparer class

8.4: Find Items in Common Between Two Collections using Intersect

Using the Intersect() methods

8.5: Find Items in Common Between Two Collections using IntersectBy

Using the IntersectBy() methods

Using a comparer class

Module 9: Understanding Deferred Execution

LINQ can defer the execution of a query until it is fully built. This module explores how that works.

9.1: Deferred Execution

What is deferred execution and why it is important

Streaming and Non-Streaming operations

Which operators are Streaming and which are Non-Streaming

9.1: Using Yield

What is Yield

Build your filters using Yield