LINQ Fundamentals in C# 10

Where LINQ Fits into Your Toolbelt

Paul D. Sheriff
Business / IT Consultant

psheriff@pdsa.com www.pdsa.com

Version Check

This course was created by using:

- .NET 6
- C# 10
- Visual Studio Code 1
- Visual Studio 2022

Version Check

This course is 100% applicable to:

- .NET 6 / 7
- C# 10 / 11
- Visual Studio Code 1
- Visual Studio 2022

Course Goals

Advantages of using LINQ

Select and order data

Search for data

Extract subsets of data

What is in common within items in collections

What is in common between collections

Join and group data

Aggregate data using Min(), Max(), Sum(), etc.

Understand how deferred execution works

I assume you...

- Are a C# developer
- Are familiar with VS Code or Visual Studio
- Are familiar with SQL
- New to using LINQ

Prerequisites

- C# Generics
- C# Delegates, Lambda Expressions
- C# Extension Methods

About This Course

What's in This Course

Learn LINQ query/method syntax side-by-side

Over 140 demos!

How to Get the Most out of This Course

Watch this module for important LINQ basics

Download the starting exercises

Follow along with the demos

LINQ Community Resources

https://github.com/PaulDSheriff/LINQFundamentalsCSharp10

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

https://docs.microsoft.com/en-us/samples/dotnet/try-samples/101-linq-samples/

https://blogs.pdsa.com - Search for LINQ

What Is LINQ?

What Is LINQ?

SQL-like syntax in C# and Visual Basic

Query any type of collections that implement IEnumerable<T> or IQueryable<T>

Common IEnumerable Types

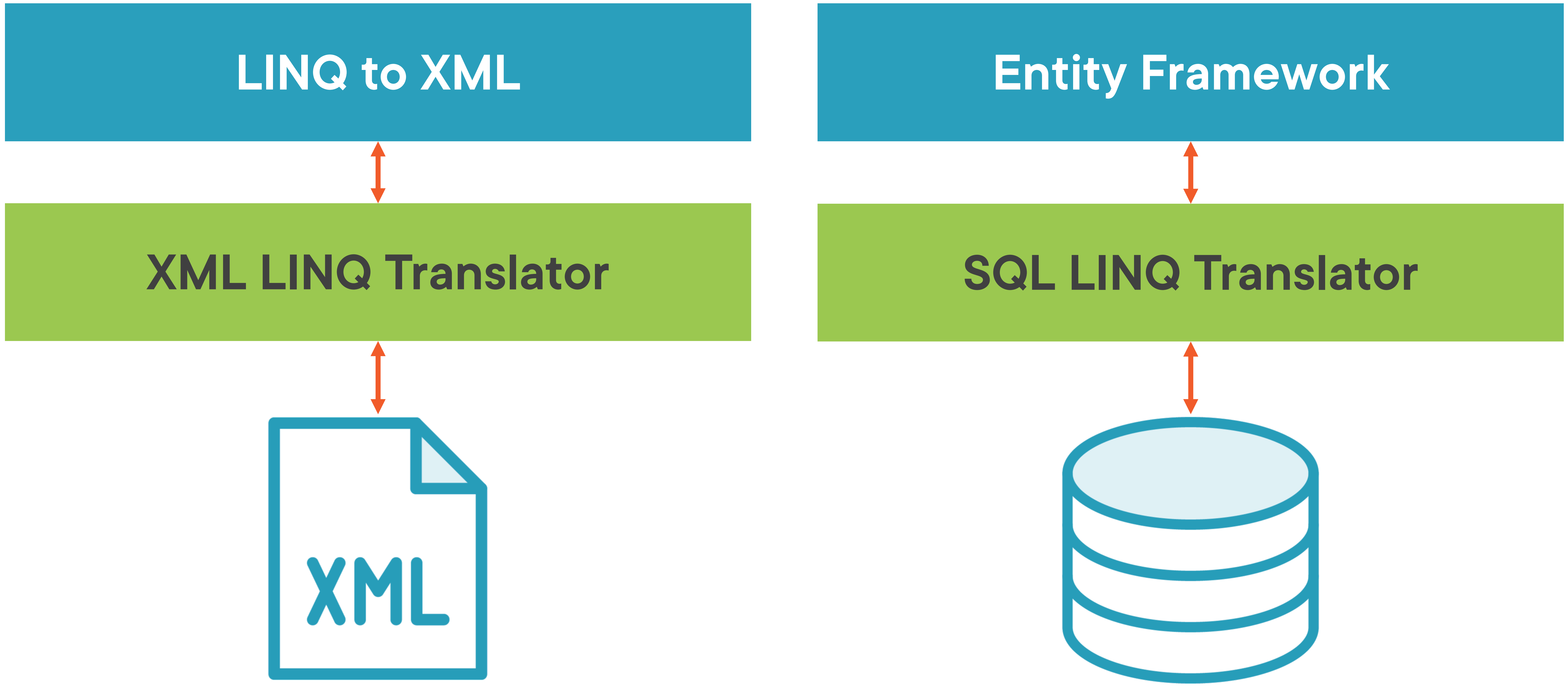
Any array

String (Array of characters)

List<T> (Examples: List<Product>, List<Customer>)

HashSet<T>, Dictionary<TKey, TValue>, LinkedList<T>, etc.

LINQ Integrations (IQueryable)



LINQ Integrations (IQueryable)

LINQ to XML

Pluralsight Course:

Working with XML in C#

Entity Framework

Pluralsight Course:

Getting Started with Entity Framework 6

LINQ to Objects

LINQ and Strings

LINQ and Reflection

LINQ and File Directories

LINQ to Entities

LINQ to DataSet

Using LINQ

Must add using statement

`using System.Linq;`

Adds extension methods of Enumerable and Queryable base classes

Examples of SQL, C# Loops, and LINQ

Comparison of SQL, Loops and LINQ

SQL is very similar to LINQ

Let's look at SQL, looping and LINQ

Using a SQL Where Clause

```sql
SELECT * FROM Products
WHERE ListPrice > 1000
```

Simulate a SQL Where Clause Using C#

```
List<Product> products = GetProducts();

List<Product> list = new ();

foreach (Product product in products) {

  if(product.ListPrice > 1000) {

    list.Add(product);

  }

}
```

C# LINQ Where Clause

```
List<Product> products = GetProducts();

var list = (from prod in products
            where prod.ListPrice > 1000
            select prod).ToList();
```

Using a SQL DISTINCT Clause

```sql
SELECT DISTINCT Color FROM Products
```

Simulate a SQL DISTINCT Clause Using C#

```
List<Product> products = GetProducts();

List<string> list = new();

foreach (Product product in products) {

  if (!list.Contains(product.Color)) {

    list.Add(product.Color);

  }

}
```

C# LINQ Distinct() Method

```
List<Product> products = GetProducts();

var colors = (from prod in products
              select prod.Color).Distinct().ToList()
```

Using a SQL MIN() Aggregate Function

```
SELECT MIN(ListPrice) FROM Products
```

Simulate SQL MIN() Using C#

```
List<Product> products = GetProducts();

decimal ret = decimal.MaxValue;

foreach (Product product in products) {

  if (product.ListPrice < ret) {

    ret = product.ListPrice;

  }

}
```

C# LINQ Min() Method

```
List<Product> products = GetProducts();

decimal value = (from prod in products
                 select prod.ListPrice).Min();
```

SQL Query vs. LINQ Query Syntax

SQL	LINQ
SELECT MAX(ListPrice) FROM Products	(from prod in Products select prod.ListPrice).Max()
SELECT AVG(ListPrice) FROM Products	(from prod in Products select prod.ListPrice).Average()

SQL Query vs. LINQ Query Syntax

SQL	LINQ
SELECT * FROM Products ORDER BY Name DESC	from prod in Products orderby prod.Name descending select prod
SELECT Name FROM Products	from prod in Products select prod.Name

Why Use LINQ?

Unified approach for querying any type of objects

Eliminate looping code

IntelliSense support

Type-checking of objects at compile time

What Can You Do With LINQ?

LINQ Operations

Select

Projection (change shape)

Order (ascending / descending)

Get an Element (find, first, last, single)

Filter (where)

LINQ Operations

Iteration / Partioning
(foreach, skip, take)

Quantify
(any, all, contains)

Set Comparison
(equal, except, intersection)

Set Operations
(union, concat)

LINQ Operations

Joining
(inner joins, outer joins)

Grouping
(groupby, subquery, groupjoin)

Distinct Sets
(distinct)

Aggregation
(count, sum, min, max, average)

Module Summary

LINQ is a sql-like syntax for C#/Visual Basic

Can be used with many types of collections

Can search, order, group, etc.

Can integrate with XML, databases

Up Next:
Use LINQ to Select Data within Collections