Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

TRAINING SEMINAR

Paul D. Sheriff
President
Paul D. Sheriff Consulting

Level: Intermediate

LINQ in .NET 6 from the Ground Up

Paul.About

- Business/Technology Consultant
  - Over 35 years of IT experience
  - Software consulting business owner for 30+ years
  - Writer, speaker, consultant, mentor
- Pluralsight Author
  - 30+ courses on LINQ, JavaScript, Angular, C#, WPF, SQL Server, XML, etc.
  - pluralsight.com/profile/author/paul-sheriff
- Contact Info
  - psheriff@pdsa.com
  - www.pdsa.com

Paul D. Sheriff Consulting

- Business Consulting
- Software Architecture / Programming
- Training / Mentoring
- Pluralsight Courses

- www.pdsa.com

Logistics

- Class runs 9am - 5pm CST each day
- Labs will run from 1/2 to 3/4 of an hour
  - We will incorporate breaks into the labs
- We will take 45 minutes for lunch

- Hands-On Labs
  - PDFs on my github before we begin each lab
  - Finished labs on my github after the lab starts
  - https://github.com/PaulDSheriff/VSLive-Trainings

Assumptions

- You are familiar with…
  - .NET
  - C#
  - Generics and Lambda Expressions
  - Visual Studio or VS Code
- Want to learn new features in LINQ for .NET 6
- Want to learn to use LINQ more effectively

Agenda

- Advantages of using LINQ
- Selecting and order data
- Searching for data
- Find single items in collections
- Extract subsets of data
- Aggregate data
- What is contained within collections

Agenda

- Grouping data
- Iterating over collections
- Joining collections
- Concatenating collections
- Comparing two collections
- Understand how deferred execution works
- *If time permits…*
  - *Entity Framework*
  - *LINQ to XML*

What is LINQ?

What Is LINQ?

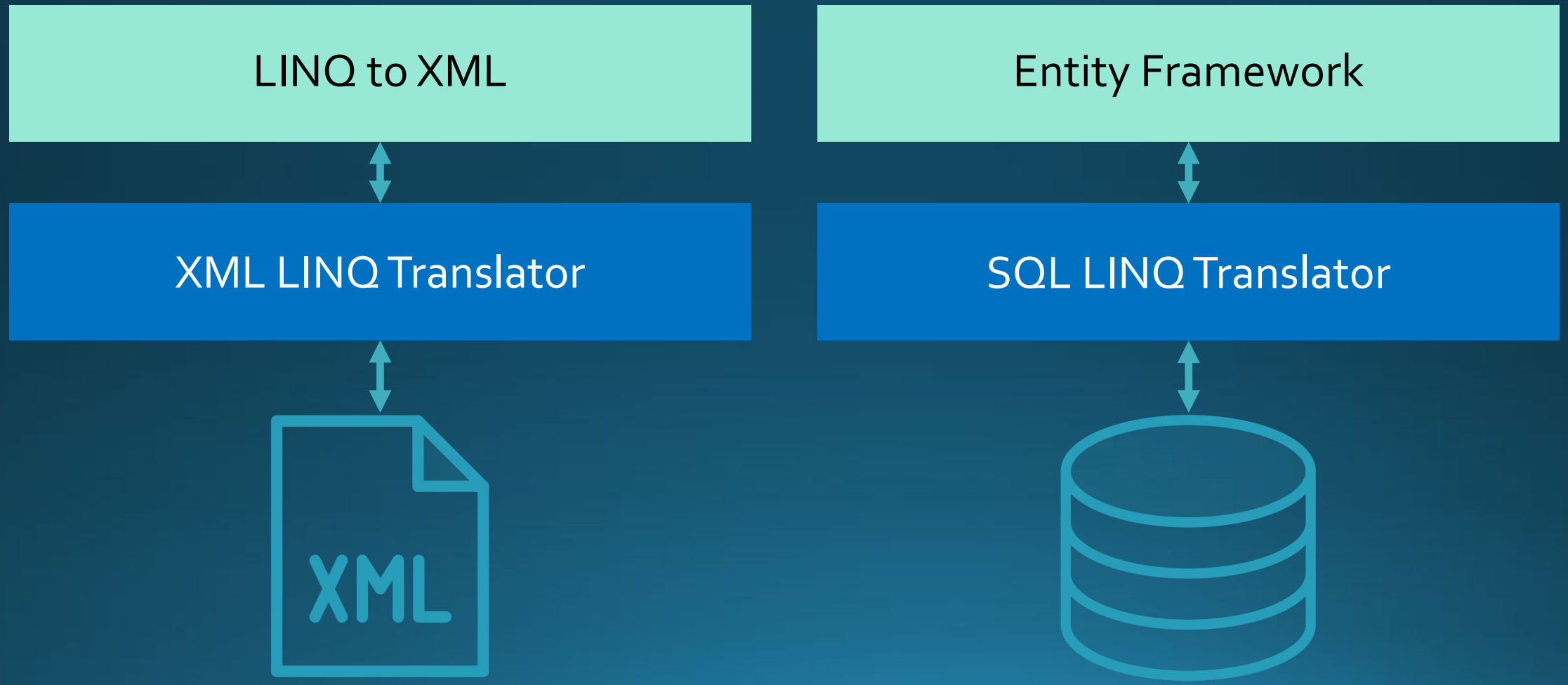
SQL-like syntax in C# and Visual Basic

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

Common IEnumerable Types

- Any array
- String (array of characters)
- List<T> (List<string>, List<Product>, List<Customer>)
- HashSet<T>, Dictionary<TKey, TValue>, LinkedList<T>, etc.

LINQ Integrations (IQueryable)



LINQ Integrations (IQueryable)

LINQ to XML	Entity Framework
Pluralsight Course: Working with XML in C#	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 WHERE clause looks like the following

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

Using a C# Loop

- Use a loop and if statement to filter rows

```
List<Product> products = GetProducts();
List<Product> list = new();
foreach (Product product in products) {
  if(product.ListPrice > 1000) {
    list.Add(product);
  }
}
```

Using a LINQ Where Clause

- LINQ is similar to SQL

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

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

Using a SQL DISTINCT Clause

- SQL DISTINCT clause looks like the following

```
SELECT DISTINCT Color
FROM Products
```

Using a C# Loop

- Use a loop and if statement to get distinct data

```
List<Product> products = GetProducts();
List<string> list = new();
foreach (Product product in products) {
  if (!list.Contains(product.Color)) {
    list.Add(product.Color);
  }
}
```

Using LINQ DISTINCT Method

- LINQ is similar to SQL

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

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

Using the SQL MIN() Aggregate

- SQL MIN() aggregate function looks like the following

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

Using a C# Loop

- Use a loop and if statement to get distinct data

```
List<Product> products = GetProducts();
decimal ret = decimal.MaxValue;
foreach (Product product in products) {
 if (product.ListPrice < ret) {
  ret = product.ListPrice;
 }
}
```

Using LINQ Min() Method

- LINQ is similar to SQL

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

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

SQL Query vs. LINQ Query Syntax

SQL

```
SELECT MAX(ListPrice)
FROM Products
```

```
SELECT AVG(ListPrice)
FROM Products
```

LINQ

```
(from prod in Products
 select prod.ListPrice)
 .Max()
```

```
(from prod in Products
 select prod.ListPrice)
 .Average()
```

SQL Query vs. LINQ Query Syntax

SQL

```
SELECT * FROM Products
    ORDER BY Name DESC
```

LINQ

```
from prod in Products
orderby prod.Name
descending
select prod
```

SQL

```
SELECT Name FROM
        Products
```

LINQ

```
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)