

.NET LINQ, Expression Trees And Rx

Lambda/Query Expressions, LINQ-To-Objects, Trees, Providers, Rx Observ[able|er], Subjects, Operators

.NET's LINQ (Language INtegrated Query) is an innovative querying capability built into the .NET Framework and languages such as C#. Querying plays a major role in most applications. For .NET and C# to pay particular attention to how code can query a variety of data sources (e.g. SQL databases with EF Core, XML documents, observable streams with Rx, collection classes, management objects, ..) can be most beneficial.

Once .NET application developers learn the basics of LINQ they can then explore how to extend it and how to expose custom data sources as LINQ targets.

Think of .NET [expression trees](#) as an AST of a query. They are used in LINQ and many custom solutions. When LINQ providers talk to remote data stores, expression trees are at the heart. Expression trees can automatically be created by a .NET compiler and also directly interacted with by application code.

Rx (Reactive extensions) is "an API for asynchronous programming with observable streams" [\[link\]](#). It is based on the observable pattern ([one of the GoF patterns](#)). Rx is available for many languages (e.g. Angular uses it extensively), this course covers Rx+C#.

Contents of One-Day Training Course	
<p>Target Audience Software engineers wishing to learn about the foundation of LINQ, Expression Trees and Rx.</p> <p>Prerequisites Attendees are expected to be experienced C#/.NET developers with some awareness of data querying, collections and abstract syntax tree usage.</p>	<p>Overview: LINQ, Expression Trees and Rx Introduction to each, how they fit together and how they might be used in applications Functional programming ideas</p> <p>Review of C# Features Modern features of C# that are of interest Lambda expressions, anonymous types, extension methods, var, flexible object initialization, partial classes/methods Func<> and Action<></p> <p>LINQ Concepts Immediate & deferred execution Query Syntax vs. Method Syntax What are operators</p> <p>Use Query Expressions Query Expression Syntax SQL-like format Typing</p> <p>Standard Query Operators Tour System.Linq namespace Exploring provided operators Adding custom query operators Inputs and outputs Moving to saying what you want Lambda, delegates and expression trees IEnumerable<T> vs. IQueryable<T></p> <p>Designing LINQ Providers How is the network involved? How does data flow? Where, and when, are expressions run? Creating a custom LINQ Data Provider Programmatically querying custom data</p> <p>Expression Trees Introduction to expression trees Expression trees and IQueryable<> Dynamically compiling expression trees Review of important expression types System.Linq.Expressions The LambdaExpression type Factory methods</p> <p>Debugging How to debug LINQ code Writing your own debug visualizer for expression trees</p> <p>ReactiveX for C# Observable and Observer Enumerable vs. observable Rx operators Subject Disposables How to use Rx within application code</p> <p>Advanced Rx Rx and .. Handling time / Exceptions Threading / Subscriptions Notification / Materialization Cold vs. hot / Scheduling</p> <p>Async Streams in C# 8 Important improvements to <code>async/await</code> in C# 8 allows precise control of async streams with rich new features</p> <p>Custom Rx Building your own observers, observables and operators Managing subscription lifetimes</p>