

.NET Entity Framework Core 3 Using C# Models, Annotations, Fluent API, Querying, Saving, Advanced, EF Core+Domain Model, Testing, Project

Handling data is one of the most complex aspects of any enterprise application development project. (The fact that Microsoft's .NET has frequently significantly changed its data story proves this!). Previous approaches have not been entirely satisfactory and many devs thought "There's got to be a better way". Enter EF Core. EF Core 3 is a modern ORM that has rapidly evolved out of code-first Entity Framework 6 for .NET Framework. The latest EF Core 3 (part of .NET Core 3) provides C# 8 app developers with a rich data access capability, in tune with modern development ideas (domain models, testing, SQL&NoSQL, agile, ..).

EF Core uses LINQ ([improved in EF Core 3](#)) as its querying technology to great effect. LINQ is a radically different approach to querying that integrates object development and data access in a cohesive language environment.

This course covers the latest release of EF Core (v3, including C# 8 support). The goal is to take C# / .NET Core application developers along a journey which will end up where they can competently program against a database (SQL or NoSQL) or other kinds of data sources from within their .NET applications.

Contents of One-Day Training Course	
<p>Target Audience C# 8 / .NET Core 3 application developers wishing to create modern apps that need to access databases using the best ORM available for .NET, that is EF Core 3.</p> <p>Prerequisites Practical experience of C#, some previous database programming and SQL knowledge (any database).</p> <p>All demos and labs will be using .NET Core 3, C# 8 and Visual Studio 2019.</p>	<p>EF Core Overview Overview of modern data access A powerful ORM (Object-Relational Mapping) for .NET applications Efficient, configurable, improving rapidly How EF Core fits in with rest of .NET</p> <p>Tour Of Features Use an initial sample app to practically demonstrate main feature set of EF Core Role of DbContext, DbSet, DbQuery, .. Data flows, config Error handling</p> <p>Tooling Development / data environment Connection string Distinct SQL RBMS data providers Non-SQL data providers Specialist data providers Additional developer tooling</p> <p>Models POCO – Plain Old C# Objects From these, auto-gen database schema Conventions / Annotations / Fluent API OnModelCreating Ways to influence generated model</p> <p>Querying Applying usual querying syntax: filtering, relationship following, ordering, aggregate Query types</p> <p>Saving SaveChanges / tracking / concurrency Transactions, cascades, keys Disconnected entities</p> <p>LINQ And EF Core 3 Deeper look at query language Going beyond basics, how to really use LINQ with EF Core Complex queries (e.g. joins)</p> <p>EF Core and .. Migrations Security Database views Stored procedures GIS</p> <p>EF Core and Domain Model Place domain model and data model in separate assemblies Data model references domain model (not the other way around) Domain model does not reference EFCore Prefer use of Fluent API – why? DDD Bounded Context = DbContext Repositories as interface in domain model (no EF Core) and implementation in data model (can be switched out for alternative)</p> <p>Testing This arrangement greatly helps testing Importance of dependency injection Modern testing approach and EF Core Testing database queries</p> <p>Project Developing an end-to-end layered solution consisting of Angular UI, ASP.NET Core REST API, domain model and EF Core 3 data model Looking particularly at last of these</p>