**Visual Studio Toolbox**

**Entity Framework Core**

How to use entity framework core

https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Entity-Framework-Core-Part-1

Part 1

**Overview**

What is EF?

EF – is an ORM (Object-relational mapper)

EF is less about talking to the database, and more about how you get to work with the data in your app.

What is an ORM?

ORM (mapper) converts what is good for the database (relational) into what is good for the application (object).

What our application needs and what is optimized for the database

**Relational**

* Databases tend to store their data very relationally (relational databases)
* Normalized (specialized) tables based on the data:
  + Customer table, Products table

**Object**

* But applications don’t work that way, we tend to work with domain objects (models)
* Looking at a customer, also want to their orders, and order details, and the products related

EF vs EF Core

* Complete rewrite of classic version
* Similar in name, goal, use
* Both use ADO.NET/ Core under the covers
  + Still using data providers, connections, command objects. It’s just abstracted away.

Data access can be thought of as plumping, almost every application needs it. You want to streamline “data access”, to spend more time on other functionality, and less time on the plumping.

EF is the plumping that every application needs, but we don’t want to build.

The goal always is to spend less time writing plumping code and more time writing business logic that’s useful for the app.

**Use**

Two different paradigms of using EF.

1st You have this existing database and you want to start using EF Core to query the data and to work with data, but how does that work? Because we only have the thing called “code first”.

Code first - really means code centric.

EF classic

* 4 ways of talking to a database
  + Designer from an existing database
  + Designer with no database
  + Code first with an existing database
  + Code first with no database

EF is very modular only pull down what you need.

* (Relational, Abstractions, Analyzers, Design, SqlServer, Tools)

NuGet Package Manager:

* Search for (Microsoft.EntityFrameworkCore)
* Click on (Microsoft.EntityFrameworkCore.SqlServer)

Starting with an existing database (AdventureWorks2016)

Create all objects that we need with the scaffolding process

**Scaffolding**

Run a command line interface process that will take the existing database and create all of the EF objects that we need to work with it.

The command line interface (CLI) is “the queen of the root”, you can do everything you need to do with .NET Core and EF Core from the command line. And you can do most of what you need to from Visual Studio. The command line comes first, and Visual Studio follows.

.NET Core global tool

Similar to .NET Classic, putting something into global assembly cache (GAC) stores. So you can access it from anywhere.

To run any of the command line tools in EF Core, you need to install the global tool for EF Core.

(dotnet ef dbcontext scaffold "server=(localdb)\mssqllocaldb;Database=Adventureworks2016;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -d -c AwDbContext --context-dir EfStructures -o Entities)

**DbContext**

Like the master control program from Tron. It is the heart and soul of how EF works.

It derives from DbContext, it contains a mechanism to tie into the database, to open connections, works very closely with the change tracker, it holds all the DbSets, and relationship with a database provider.

Provides a mechanism for shaping our database (Mapping).

* Table names
* Schema
* Datatype of a column
* Column name

When installing (Microsoft.EntityFrameworkCore.SqlServer) package, we added the SqlServer data provider into our installation.

**DbSet**

Specialized collections that wrap our tables.

Table per class.

Every table in the database is represented by a DbSet.

*This is the “Object Relational Mapping” objects mapped to tables in the database.*

A key tenant throughout all of .NET Core is dependency injection.

We actually configure connection string and other options, by injecting them into the DbContext.

**OnConfiguring**

Don’t use OnConfiguring.

What OnConfiguring does, is provide a fallback mechanism for if you haven’t configured (optionsBuilder.IsConfigured) class, it will use its hard coded connection string to configure it for you.

C Sharp objects are called “entities” typically.

**OnModelCreating**

Provides a deeper mechanism for shaping the database.

A migration is the reverse process of scaffolding.

**Change tracker**

Provides the power to using EF Core. The core of EF Core.

Anytime you start working with the DeSets in your code the changes are tracked, and that is used to build very efficient queries.