Project Journal

# Northwind 2025

Use the Northwind database, create an entity framework version. It is still a good example of a database.

All it needs to do is contain some data to hang a web api, mvc site and react web-app from.

Make it portable - a MsSql version of entity framework and in-memory (so it can be used for testing without connecting to anything).

Maybe – a version which connects to another database. This covers the ‘we want to save money so let’s use postgresql/mysql ‘.

## 01-DEC-2022

Create repo on github.

Clone locally:

git clone https://github.com/dsaunders531/Northwind-2025.git

Add this journal.

Add database creation scripts.

Create the database using the creation scripts.

Create 2 projects these are going to be the base database libraries

1 – Northwind.Context – for the base classes

2- Northwind.Context.MsSql – for SQL server things. This will also contain migration info.

Map the database using the package manager console command.

Scaffold-DbContext "Server=[Instance];Database=Northwind-2025;Trusted\_Connection=true;MultipleActiveResultSets=true;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -DataAnnotations -Context NorthwindContext

Install the nuget packages it asks for.

This copies the database structure except triggers, stored procedures & functions.

Views are copied. However, these are database specific. If we migrate to another database type, it may have no concept of a view. Anything associated with views will be removed. The functionality they provide will be mapped later as part of stored procedures.

Re-arrange the output so the context is in the Context folder.

The next steps are for creating a context which is not attached to SQL server.

This means I can create an in-memory, postgresql or whatever else I may need and the context is not tied to MsSql server.

Drag the models into Northwind.Context and change their namespace.

Analyse the output and remove any dependancies on SqlServer.

Move the views code out into its own interface.

The end result – A context which is not dependant on a database and an interface for services which we will need to implement in each database type.

Back in the Northwind.Context.MsSql project, change the connection string for the context and create a migration using package manager console:

Add-Migration InitialCreate

Next – I want to add the stored procs and views. I would implement the same for any database if it supports these kinds of functions. Otherwise, I would replicate the functionality using entity framework context.

The steps for this are:

1. Create an empty migration: Add-Migration StoredProcsAndViews
2. Copy out all the stored procs as text.

Note: The migration steps are only needed if you are planning to move the database. If you just want to put EF onto an existing database or use an in-memory database then these steps are not required.

While writing up the service to run stored procedures and views, I could see there would be a lot of repetition of the same commands – open connection, create a command and execute a data reader.

By moving this code out into individual command classes, I can base class and keep the quantity of typing (and possible errors) low.

It also adds 2 examples of patterns – the command pattern and façade pattern. The service provides a means of running the all commands from one object – the caller does not know how the commands are being run.

Next –

Console app to perform the data migration (all data in order)

Add seed data (for in-memory and tests – a small subset of data)

Write tests

Create console app to perform migration (create database and add all seed data).

Add method to update the database timestamps.

Add api and swagger front end

Add tests for this (controller tests)

Change stored procs so they are not year dependant.

Add a routine to update the database timestamps so it is always relevant (for development and staging only)