Domain Modeling and Entity Framework Core Database Migration

# Background

This document describes the process of domain modeling with entities based on Domain Driven Design principle and using Code First development with Microsoft Entity Framework Core on database migration and update.

# System Requirements:

* ASP.NET Core 2.1
* Entity Framework 2.1
* .NET SDK 2.1
* VS2017 15.9.3 or the latest

# The Process

## Solution Setup

The solution contains three project

1. Web Application (Asp.net core web api) version 2.1
2. Class Library for Data Layer (persistent)
3. Class Library for Domain Layer (business core)

Nuget Packages Required:

Microsoft.EntityFrameworkCore (2.1.1)

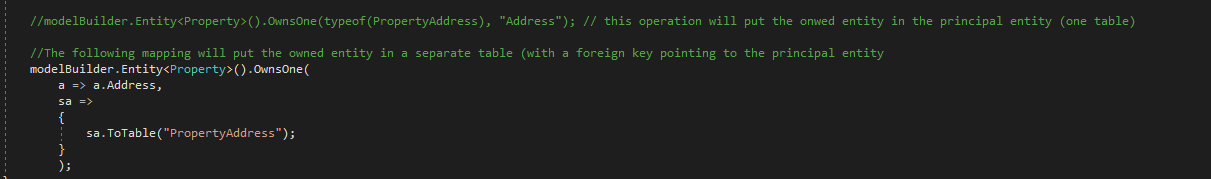
Microsoft.EntityFrameworkCore.SqlServer (2.1.1)

Microsoft.EntityFrameworkCore.Tools (2.1.1)

These can be installed by a few different ways.

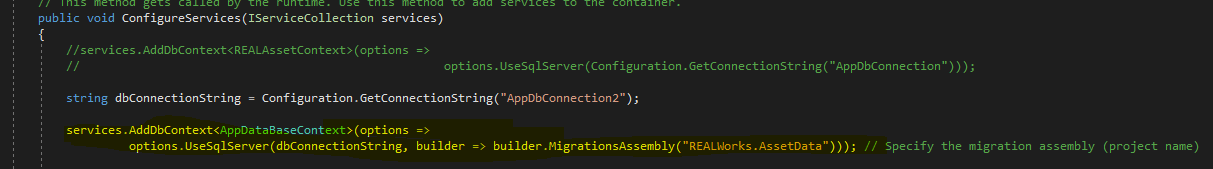
## Domain Modeling

1. Create entity class as required in Domain Layer project, e.g. Property
2. Create value object class as required in Domain Layer project, e.g. Address, **Note: there is NO id for this class, it can inherit from the base value object,**
3. Create db context, inherit from DbContext of the EF Core
   1. Make sure create appropriate constructors
   2. Create required DbSet, which will generate tables
   3. Create required mapping, referencing to the previously auto generated mapping from Database First operations
   4. Make sure configure Owned Entity properly. In this case, Address is owned by Property, ref: <https://docs.microsoft.com/en-us/ef/core/modeling/owned-entities>, make sure using mappings to create separate table. See below:



## Create Migration

First of all, specify the migration assembly in the Startup.cs file in the startup project, the Asp.net core web api project as follows:

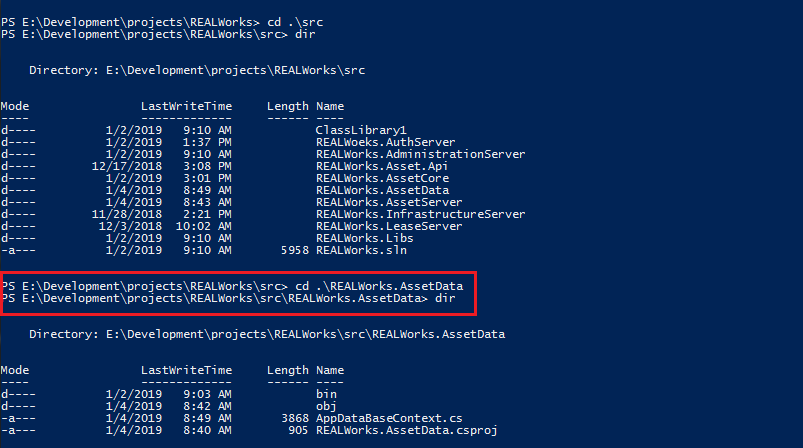


Ref: <https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/projects>

*Note: This is to make sure that the migration will be created in the Data Layer project, which is a Class Library project. By default, the migration will be created on the Startup project, in our case, that is the web api project.*

With .NET SDK installed, command of dotnet ef is available. Windows Powershell will be used for the operation of migration and database update.

* Open Powershell, change the directory to the Data project:

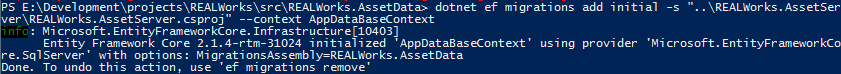


* Run the command:

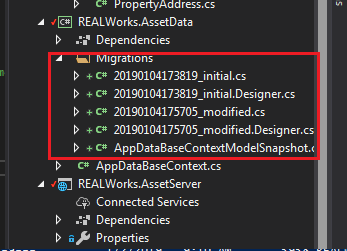
dotnet ef migrations add initial –s “..\REALWorks.AssetServer\ REALWorks.AssetServer.cproj” –context AppDataBaseContext

*where: -s to specify the startup project, which is the web api project, --context to specify the db context if multiple context exists in the solution, in our case, it is AppDataBaseContext*

* If successful, it will display:



Also, check the project folder to make sure the migration has been created:



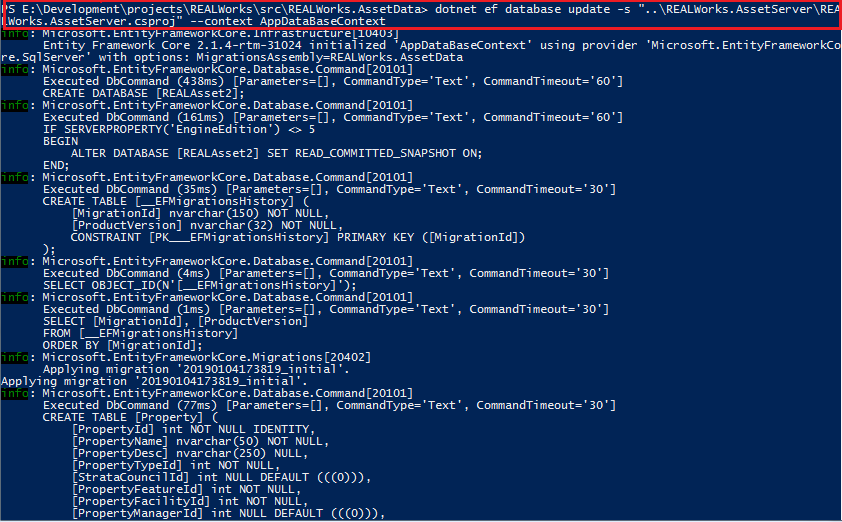
## Update Database

Run the command to update/create database (no need to create database manually in advance):

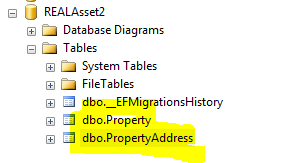
dotnet ef database update –s “..\REALWorks.AssetServer\ REALWorks.AssetServer.cproj” –context AppDataBaseContext

Similarly, startup project and db context need to be specified.

If successful, it will display something like:



Then check the database creation from SQL Server Management Studio (you may need refresh the view):



### References

1. <https://www.dustinhorne.com/post/2017/10/27/deeper-dive-into-ef-core-2-part-1>
2. <https://www.learnentityframeworkcore.com/configuration/fluent-api> (EF core fluent api mapping)