**Learning Migrations**

* باید توجه داشته باشیم که به هیچ ‌عنوان، تا قبل از اولین مرحله Migration (Initial)، پروژه را به مشتری تحویل نداده! و یا پروژه (سایت/سامانه) را Launch نکنیم!
* باید توجه داشته باشیم که عملیات Migration، کاری بسیار مهم، دقیق و پر خطا می‌باشد! و باید در یک تیم، معمولا یک نفر، که معمولا با **تجربه‌ترین** و با **دقت‌ترین** فرد می‌باشد، این عمل را انجام دهد!

**ساختار پروژه، و نصب NuGet ها، قبل از Migration**

[[Domain]]

[[Persistence]] 🡪 Add Project Reference: [[Domain]]

Microsoft.EntityFrameworkCore.Proxies 🡪 **Just For Lazy Loading**

Microsoft.EntityFrameworkCore.SqlServer 🡪 **For Using Microsoft SQL Server**

[[MyApplication]] 🡪 Add Project Reference: [[Domain]] and [[Persistence]]

Microsoft.EntityFrameworkCore

**نکته مهم:** قبل از اولین نسخه محصول، اصلا و ابدا نیازی به Migration نیست! این بدان معنا است که بعد از هر تغییری که در ساختار Domain، و یا فایل‌های Configuration، به هر شکل، ایجاد می‌کنیم، صرفا کافی است، بانک اطلاعاتی را Delete کرده و مجددا برنامه را اجرا نماییم، تا برنامه، با داشتن دستور ذیل:

Database.EnsureCreated();

در سازنده کلاس ApplicationDbContext، به طور خودکار، بانک اطلاعاتی را ایجاد نماید.

**نکته مهم**

قبل از اولین نسخه محصول، باید حتما Migration صورت گیرد!

* قبل از اولین مرحله Migration
  + ابتدا پروژه راRun می‌کنیم
  + بررسی می‌کنیم که پروژه، هیچ‌گونه خطایی نداشته باشد
  + حال دستور Database.EnsureCreated(); را Comment و یا حذف می‌کنیم!

**شروع عملیات Migration**

**ساختار پروژه، و نصب NuGet ها، قبل از Migration**

[[Domain]]

[[Persistence]] 🡪 Using Domain

Microsoft.EntityFrameworkCore.Proxies 🡪 **Just For Lazy Loading**

Microsoft.EntityFrameworkCore.SqlServer 🡪 **For Using Microsoft SQL Server**

[[MyApplication]] 🡪 Using Domain and Persistence

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.Tools 🡪 **For Migration**

Menu 🡪 Tools 🡪 NuGet Package Manager 🡪 Package Manager Console:

**Note: Select [[Persistence]] Project:**

PM> **Add-Migration Initial**

PM> **Add-Migration "Initial"**

PM> **Add-Migration Initial Migration 🡪 Error!**

PM> **Add-Migration "Initial Migration"**

**Error!**

Add-Migration: The term 'Add-Migration' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

Resolve:

**Microsoft.EntityFrameworkCore.Tools**

Again…

PM> **Add-Migration Initial**

Build started...  
Build succeeded.  
To undo this action, use Remove-Migration.

**Check new Folder and Files:**

[[Persistence]]

[Migrations]

20241028094932\_Initial.cs

ApplicationDbContextModelSnapshot.cs

هنوز بانک اطلاعاتی ایجاد نشده است!

حال، پشیمان می‌شویم و می‌خواهیم تغییراتی اعمال کرده و سپس اولین Migration را ایجاد نماییم:

PM> **Remove-Migration**

با اجرای دستور فوق، پوشه Migrations و فایل‌های درون آن حذف می‌شوند.

Again…

PM> **Add-Migration Initial**

Build started...  
Build succeeded.  
To undo this action, use Remove-Migration.

در پوشه Migrations، محتوای فایل 20241028103529\_Initial را بررسی می‌کنیم.

در داخل این فایل، دستوراتی در جهت ایجاد، ویرایش و یا حذف جداول، فیلدها و غیره وجود دارد و در صورتی که Seed Data داشته باشیم نیز، در داخل این فایل اطلاعات آن‌ها قرار می‌گیرد!

می‌توانیم در این فایل، اقدام به ایجاد، ویرایش و یا حذف Stored Procedure ها نیز نماییم:

**# UP**

migrationBuilder.Sql(  
@"CREATE PROCEDURE MyCustomProcedure  
AS  
SELECT \* FROM Countries  
");

**# Down**

migrationBuilder.Sql(@"DROP PROCEDURE MyCustomProcedure");

**نکته مهم:** به فیلد Id جدول Country، که Auto Increment بوده، ولی ما، در داخل فایل Configuration، به آن مقدار داده‌ایم، توجه نمایید!

حال می‌خواهیم که Database را ایجاد اولیه / ویرایش نماییم:

PM> **Update-Database**

Build started...  
Build succeeded.  
Applying migration '20241028103529\_Initial'.  
Done.

وارد محیط SSMS شده و بانک اطلاعاتی ایجاد / ویرایش شده را بررسی می‌کنیم.

**نکته مهم:** دقت نمایید که با این روش (Migration)، جدولی به نام:

\_\_EFMigrationsHistory

ایجاد می‌شود. در حالی که، اگر از دستور ذیل:

Database.EnsureCreated();

استفاده می‌کردیم، این جدول ایجاد نمی‌گشت!

در داخل این جدول و در فیلد MigrationId اولین رکورد، دقیقا نام فایل Migration ای که در داخل پوشه Migrations ایجاد شده است، قرار گرفته است.

حال تغییراتی را در Domain ایجاد می‌کنیم:

فرض کنید که مشتری، تغییراتی در اطلاعات بانک اطلاعاتی اعمال می‌کند:

* مثلا رکورد ایجاد می‌کند
* مثلا رکوردی را ویرایش می‌کند
* مثلا رکوردی را حذف می‌کند

**خطوط قرمز در تغییرات:**

* Rename: Table
* Rename: Field
* **Add Field: Is Not Nullable 🡨 Default Value**
* **Change Property Type: Field**
* **Change Relation Type: From "One to Many" to "Many to Many"**
* **Bad Practice!**

PM> **Add-Migration** "Version\_010"

PM> **Add-Migration** "Version\_011"

PM> **Add-Migration** "Version\_012"

PM> **Update-Database**

* **Best Practice!**

PM> **Add-Migration** "Version\_010"

PM> **Add-Migration** "Version\_011"

PM> **Add-Migration** "Version\_012"

PM> **Remove-Migration**

PM> **Remove-Migration**

PM> **Remove-Migration**

PM> **Add-Migration** "Version\_010"

PM> **Update-Database**

PM> **Add-Migration** "Create Log Model" 🡪 ‌**Bad Practice!**

PM> **Add-Migration** "Version\_001" 🡪 **Best Practice!**

PM> **Update-Database**

Version 1

Version 2

Version 3

Version 4

Version 5

1. Add a Property

In Some Model:

Add a new Property in some Model (**Better**: nullable)

$ Compile Application (Without any Error!)

PM> **Add-Migration** "Version\_01"

PM> **Update-Database**

وارد بانک‌اطلاعاتی مربوطه می‌شویم، و چک می‌کنیم که تغییراتی که در اطلاعات رکوردها قبل از آخرین Migration، اعمال کرده‌ایم، بعد از آخرین Migration، تغییری نکرده باشد.

1. Rename a Property

In Some Model:

Property: Name -> Rename -> NewName

PM> **Add-Migration** "Version\_02"

DropColumn "Googooli"

CreateColumn "Magooli"

غلط

RenameColumn "Googooli" "Magooli"

PM> **Update-Database**

حذف یک جدول به هر دلیلی و ایجاد یک جدول جدید با بخشی / همه داده‌های جدول قدیم

(1)

ایجاد جدول جدید

PM> **Add-Migration “…”**

در Control Panel سامانه، یک صفحه و یا یک API درست می‌کنیم، برای انتقال اطلاعات از نسخه قدیم به نسخه جدید

**هیچ‌وقت نباید جدول قدیم را حذف نماییم!**

برای این‌که عملیات Migration به صورت اتوماتیک صورت بگیرد و نیازی به اجرای دستور Update-Database نباشد، تنها کافی است که در Constructor کلاس ApplicationDbContext، دستور ذیل را بنویسیم:

public ApplicationDbContext() : base()  
{  
 **Database.Migrate();**  
 //Database.EnsureCreated();  
}

در صورتی که بخواهیم SQL مربوط به آخرین Migration را بدست آوریم، تنها کافی است که در پنجره Package Manger Console، از دستور ذیل استفاده نماییم:

PM> **Script-Migration**

Build started...  
Build succeeded.

**نکته آخر:**

به متغیر IsConfigured، در تابع OnConfiguring توجه نمایید:

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
{  
 **if (optionsBuilder.IsConfigured == false)**  
 {  
 var connectionString = "Server=…";  
 optionsBuilder  
 .UseLazyLoadingProxies()  
 .UseSqlServer(connectionString: connectionString)  
 ;  
 }  
}

با تشکر از آقای مهندس محمد مهدی محمدی بابت ویرایش متن این مقاله