**How to Migrations**

* قبل از ارائه اولین نسخه محصول، اصلا و ابدا نیازی به عملیات Migration نمی‌باشد!
* به هیچ عنوان، تا قبل از اولین مرحله Migration (InitialCreate)، نباید پروژه را به مشتری تحویل دهیم! و یا پروژه (سایت/سامانه) را Launch ننماییم!
* عملیات Migration، کاری بسیار مهم، دقیق و احتمالاً پر خطا می‌باشد! و باید در یک تیم، معمولا یک نفر، که معمولا با تجربه‌ترین و بادقت‌ترین فرد می‌باشد، این عمل را انجام دهد!

در صورتی که هنوز پروژه اجرایی نداریم، برای عملیات Migration، باید در کلاس DatabaseContext، یک Default Constructor نوشته و تابع OnConfiguring را Override نماییم!

ولی در صورتی که پروژه ASP.NET Core داشته باشیم، می‌توانیم به صورت Routine، از Constructor ای که ورودی Options دارد استفاده کرده و نیازی به Override کردن تابع OnConfiguring وجود ندارد.

Tools → NuGet Package Manager → NuGet Package Console:

**Note: Very Important!** Select [[Persistence]] Project:

Add-Migration InitialCreate

Remove-Migration

Add-Migration InitialCreate

Update-Database

Remove-Migration → Error!

Update-Database 0

Remove-Migration

**نکته مهم: مطالبی که در ادامه نوشته شده است!!! ممکن است نامرتبط بوده و یا غلط باشند! هنوز به روز رسانی نشده‌اند!**

**مهم مهم مهم**

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

پروژه را اجرا می‌کنیم

پروژه را تست می‌کنیم

حال مدل‌ها را تغییر می‌دادم!

مثلاً فیلدی اضافه می‌کنیم

یا مدلی اضافه می‌کنیم

قبلا مجبور بودیم که بانک اطلاعاتی را حذف نماییم!

و مجددا وقتی پروژه را اجرا می‌کردم، مشکلی به وجود نمی‌آمد!

- حال می‌خواهیم اولین نسخه پروژه را Launch نماییم

- باید Migration را شروع کنیم!

Tools → NuGet Package Manager → NuGet Package Console:

**Note: Very Important!** Select [[Persistence]] Project:

Add-Migration InitialCreate

(Persistence)

[[Persistence]]

[Migrations]

1. Create New Tables (CreateTable)

2. Seed Data (InsertData)

> Update-Database

اعمال تغییرات در مدل/مدل‌ها

> Add-Migration "Create Log Model" -> Not Happy

> Add-Migration "Version\_001"

> Update-Database

Version 1

Version 2

Version 3

Version 4

Version 5

(1)

Model:

Add a Field (Better: nullable)

(2)

Model:

Property: Googooli -> Rename -> Magooli

* Add-Migration “…”

DropColumn "Googooli"

CreateColumn "Magooli"

غلط

RenameColumn "Googooli" "Magooli"

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

(1)

ایجاد جدول جدید

* Add-Migration “…”

در کنترل پنل سامانه / یک صفحه یا یک ای‌پی‌آی درست می‌کنیم برای انتقال اطلاعات از قدیم به جدید

هیچ‌وقت جدول قدیم را حذف نکنیم!!!

در Migration چه کارهایی به شدت Bad Practice است!!!

* Rename: Table
* Rename: Field
* Change Type: Field!!!
* Add Field: Is Not Nullable!!!
* فرض می‌کنیم که می‌خواهیم پروژه‌ای ایجاد کنیم که بتواند با دو یا بیشتر Provider کار کند:
* SQLite
* SqlServer
* اطمینان حاصل می‌کنیم که در پروژه اصلی (Server)، Nuget ذیل نصب شده باشد:

<ItemGroup>

<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="7.0.4">

<PrivateAssets>all</PrivateAssets>

<IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>

</PackageReference>

</ItemGroup>

* اطمینان حاصل می‌کنیم که در پروژه Persistence، Nuget ذیل نصب شده باشد:

<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="7.0.4">

<PrivateAssets>all</PrivateAssets>

<IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>

</PackageReference>

* دستور ذیل (EnsureCreated) را از فایل DatabaseContext.cs حذف می‌کنیم!

public DatabaseContext(Microsoft.EntityFrameworkCore

.DbContextOptions<DatabaseContext> options) : base(options: options)

{

//Database.EnsureCreated();

}

و به جای آن از دستور ذیل استفاده می‌کنیم:

public DatabaseContext(Microsoft.EntityFrameworkCore

.DbContextOptions<DatabaseContext> options) : base(options: options)

{

// نوشتن دستورات ذیل کامل غلط است

// لااقل در اولین باری که بانک‌اطلاعاتی

// می‌خواهد ایجاد شود، کار نمی‌کند

//// using Microsoft.EntityFrameworkCore;

//if (Database.GetAppliedMigrations().Any())

//{

//// using Microsoft.EntityFrameworkCore;

//Database.Migrate();

//}

// using Microsoft.EntityFrameworkCore;

Database.Migrate();

}

* در پوشه مجازی (src \ Persistence) و دو پروژه به نام‌های ذیل ایجاد می‌کنیم:

Persistence.SQLite

Persistence.SqlServer

* Reference پروژه Persistence را به هر دو پروژه فوق اضافه می‌کنیم.
* Reference ای از هر دو پروژه فوق نیز، به پروژه اصلی (Server) اضافه می‌کنیم.
* تغییرات ذیل را در فایل Program.cs اعمال می‌کنیم:

switch (applicationSettings!. DatabaseSettings.DatabaseProviderType)

{

case Infrastructure.Settings.Enums.DatabaseProviderType.SQLite:

{

builder.Services

.AddDbContext<Persistence.DatabaseContext>(optionsAction: options =>

{

options.UseLazyLoadingProxies();

//options.UseSqlite(connectionString:

// applicationSettings.DatabaseSettings.SQLiteConnectionString);

options.UseSqlite(connectionString: applicationSettings

.DatabaseSettings.SQLiteConnectionString, sqliteOptionsAction: current =>

{

current.MigrationsAssembly(assemblyName: "Persistence.SQLite");

});

});

break;

}

case Infrastructure.Settings.Enums.DatabaseProviderType.SqlServer:

{

builder.Services

.AddDbContext<Persistence.DatabaseContext>(optionsAction: options =>

{

options.UseLazyLoadingProxies();

//options.UseSqlServer(connectionString:

// applicationSettings.DatabaseSettings.SqlServerConnectionString);

options.UseSqlServer(connectionString: applicationSettings

.DatabaseSettings.SqlServerConnectionString, sqlServerOptionsAction: current =>

{

current.MigrationsAssembly(assemblyName: "Persistence.SqlServer");

});

});

break;

}

}

* در پروژه اصلی (Server)، وارد فایل‌های ذیل شده:

appsettings.json

appsettings.Development.json

و مقدار Provider را تغییر می‌دهیم:

"DatabaseSettings": {

"Provider": "MSSqlServer",

"SQLiteConnectionString": "Data Source=Database\\MySQLite.db",

"SqlServerConnectionString": "Server=.;Database=DT\_CMS;TrustServerCertificate=True;Trusted\_Connection=True;MultipleActiveResultSets=true;"

}

* از طریق منوی Visual Studio، گزینه ذیل را انتخاب می‌کنیم:

Tools  NuGet Package Manager  Package Manager Console

* در قسمت Default Project گزینه ذیل را انتخاب می‌کنیم:

src\Persistence\Persistence.SqlServer

و دستور ذیل را می‌نویسیم:

Add-Migration InitialCreate

تا پوشه Migrations در داخل پروژه Persistence.SqlServer ایجاد شود.

* مجددا در پروژه اصلی (Server)، وارد فایل‌های ذیل شده:

appsettings.json

appsettings.Development.json

و مقدار Provider را تغییر می‌دهیم:

"DatabaseSettings": {

"Provider": "SQLite",

"SQLiteConnectionString": "Data Source=Database\\MySQLite.db",

"SqlServerConnectionString": "Server=.;Database=DT\_CMS;TrustServerCertificate=True;Trusted\_Connection=True;MultipleActiveResultSets=true;"

}

* از طریق منوی Visual Studio، گزینه ذیل را انتخاب می‌کنیم:

Tools  NuGet Package Manager  Package Manager Console

* در قسمت Default Project گزینه ذیل را انتخاب می‌کنیم:

src\Persistence\Persistence.SQLite

و دستور ذیل را می‌نویسیم:

Add-Migration InitialCreate

تا پوشه Migrations در داخل پروژه Persistence.SQLite ایجاد شود.

* حذف فایل‌های SQLite و بانک‌اطلاعاتی
  + فایل‌های موجود در پوشه Database که در پروژه Server وجود دارد را پاک می‌کنیم.
  + بانک اطلاعاتی DT\_CMS را SQL Server حذف می‌کنیم.
* در پروژه اصلی (Server)، وارد فایل‌های ذیل شده:

appsettings.json

appsettings.Development.json

و مقدار Provider را تغییر می‌دهیم:

"DatabaseSettings": {

"Provider": "MSSqlServer",

"SQLiteConnectionString": "Data Source=Database\\MySQLite.db",

"SqlServerConnectionString": "Server=.;Database=DT\_CMS;TrustServerCertificate=True;Trusted\_Connection=True;MultipleActiveResultSets=true;"

}

* پروژه را اجرا می‌کنیم تا بانک اطلاعاتی در SQL Server ایجاد شود و پروژه را تست می‌کنیم.
* در پروژه اصلی (Server)، وارد فایل‌های ذیل شده:

appsettings.json

appsettings.Development.json

و مقدار Provider را تغییر می‌دهیم:

"DatabaseSettings": {

"Provider": "SQLite",

"SQLiteConnectionString": "Data Source=Database\\MySQLite.db",

"SqlServerConnectionString": "Server=.;Database=DT\_CMS;TrustServerCertificate=True;Trusted\_Connection=True;MultipleActiveResultSets=true;"

}

* پروژه را اجرا می‌کنیم تا بانک اطلاعاتی SQLite در پوشه Database ایجاد شود و پروژه را تست می‌کنیم.