1. فرض کنيد که Solution شما از حداقل دو پروژه Models و يک پروژه اجرايی ديگری:
   1. ASP.NET MVC
   2. ASP.NET Web Form
   3. Windows Form Application
   4. Windows Presentation Foundation

تشکيل شده باشد.

1. و نيز فرض بر آن است، که می خواهيم **اولين نسخه عملياتی** را برای مشتری ايجاد و ارسال نماييم.
2. با استفاده از امکان NUGET، در داخل پروژه Models، و به ترتيب NUGET های ذيل را اجرا می کنيم:

PM> INSTALL-PACKAGE Microsoft.AspNet.Mvc

PM> INSTALL-PACKAGE EntityFramework

PM> INSTALL-PACKAGE SqlServerCompact

PM> INSTALL-PACKAGE EntityFramework.SqlServerCompact

1. با استفاده از امکان NUGET، در داخل پروژه اصلی، و به ترتيب NUGET های ذيل را اجرا می کنيم (دستور اول ذيل را در صورتی اجرا می کنيم که پروژه ASP.NET MVC باشد):

**PM> INSTALL-PACKAGE Microsoft.AspNet.Mvc**

PM> INSTALL-PACKAGE EntityFramework

PM> INSTALL-PACKAGE SqlServerCompact

PM> INSTALL-PACKAGE EntityFramework.SqlServerCompact

1. دقت کنيد که بايد Connection String در داخل فايل app.config پروژه Models، نوشته شده و نيز دقيقا مقدار آن با Connection String موجود در فايل app.config و/يا web.config پروژه اصلی، يکسان باشد.
2. پس از طراحی کلاس های مربوط به پروژه Models واعمال تنظيمات لازم در داخل آن، در داخل Static Constructor مربوط به کلاس DatabaseContext، دستور ذيل را می نويسيم:

namespace Models

{

public class DatabaseContext : System.Data.Entity.DbContext

{

**static DatabaseContext()**

{

**System.Data.Entity.Database.SetInitializer**

**(new System.Data.Entity.CreateDatabaseIfNotExists<DatabaseContext>());**

}

public DatabaseContext()

{

}

protected override void OnModelCreating(System.Data.Entity.DbModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

}

}

}

1. حال پروژه را اجرا می کنيم تا بانک اطلاعاتی ايجاد گردد. دقت کنيد که ممکن است در اين اجرا به خطاهايی برخورد نماييد! اصلا اهميتی ندارد! فقط مهم آن است که بانک اطلاعاتی شما ايجاد شده باشد. همين کافی است!
2. پس از ايجاد بانک اطلاعاتی، دستوری که قبلا در داخل Static Constructor، کلاس DatabaseContext نوشته ايد را Comment و يا حذف نموده و سپس از طريق برنامه NUGET، دستور ذيل را برای **فقط (دقيقا )پروژه Models** اجرا نماييد.

PM> Enable-Migrations

1. با اجرای اين دستور، پوشه ای به نام Migrations ايجاد شده و در داخل آن، دو فايل به نام های Configuration.cs و نيز xxxxxxxxxxx\_InitialCreate.cs ايجاد می گردد.
2. از آنجايی که تابع Seed مربوط به کلاس Configuration مربوط به Migration (بر خلاف سه سناريوی قبل) هميشه اجرا می شود! يک شرط قابل اطمينان نوشته که در صورت درست بودن، از داخل تابع Seed خارج شده و مجددا اقدام به ثبت اطلاعات پايه و تستی ننمايد.

using System.Linq;

using System.Data.Entity;

namespace Models.Migrations

{

internal sealed class Configuration :

System.Data.Entity.Migrations.DbMigrationsConfiguration<Models.DatabaseContext>

{

public Configuration()

{

ContextKey = "Models.DatabaseContext";

AutomaticMigrationsEnabled = false;

AutomaticMigrationDataLossAllowed = false;

}

protected override void Seed(DatabaseContext databaseContext)

{

if (databaseContext.Users.Count() != 0)

{

return;

}

try

{

// Initial Data For Base Tables!

// And/Or

// Initial Data For Testing!

}

catch (System.Exception ex)

{

Dtx.LogHandler.Report(GetType(), null, ex);

}

}

}

}

1. حال مجددا در داخل Static Constructor، مربوط به کلاس DatabaseContext دستور ذيل را نوشته و مجددا برنامه را اجرا نماييد:

static DatabaseContext()

{

//System.Data.Entity.Database.SetInitializer

// (new System.Data.Entity.CreateDatabaseIfNotExists<DatabaseContext>());

System.Data.Entity.Database.SetInitializer

(new System.Data.Entity.MigrateDatabaseToLatestVersion

<DatabaseContext, Migrations.Configuration>());

}

1. حال از اين پس، هرگاه تغييری در داخل پروژه Models اعمال نماييم، (مانند اضافه، اصلاح، حذف مدل، و/يا اضافه، اصلاح، حذف و يا تغيير نام يک Property از يک مدل و غيره...) مجددا از طريق ابزار NUGET، دستور ذيل را می نويسيم:

PM> Add-Migration -Name VersionXXX

تا در داخل پوشه Migrations کلاسی که در متن نام آن عبارت(VersionXXX) قرار دارد، ايجاد گرديده و تغييرات در درون آن نوشته شده تا در اجرای بعدی، به طور اتوماتيک، بانک اطلاعاتی را Migrate نمايد.