ENTITY FRAMEWORK

### Consola del Administrador de paquetes

Seleccione el comando de menú **Herramientas** > **Administrador de paquetes NuGet** > **Consola del Administrador de paquetes**.

Cuando se abra la consola, compruebe que la lista desplegable **Proyecto predeterminado** muestra el proyecto en el que quiere instalar el paquete. Si tiene un único proyecto en la solución, ya está seleccionado.

**Agregar nueva migración:**

Add-Migration NombreDeLaNuevaMigracion

**Actualizar la base de datos:**

Update-Database

**Volver a una versión anterior**

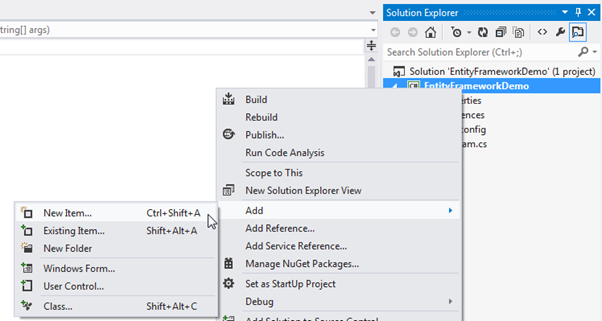
Update-Database –TargetMigration:NombreDeLaMigracionALaCualVolver

Generate Context and Entity Classes from an Existing Database in EF 6 Code-First Approach

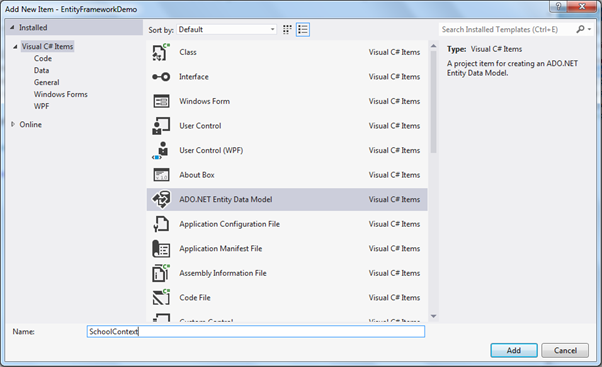
Here, you will learn how to generate context and entity classes for an existing database, using the code-first approach.

Entity Framework provides an easy way to use the code-first approach for an existing database. It will create entity classes for all the tables & views in your existing database and configure them with data annotations attributes and Fluent API.

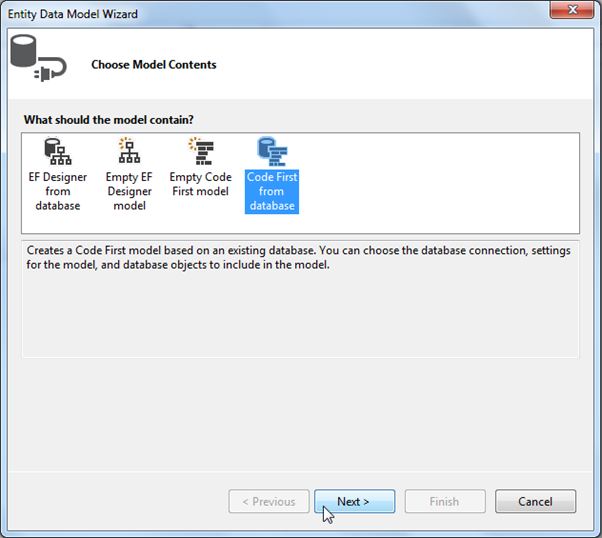
To use code-first for an existing database, right click on your project in Visual Studio -> Add -> New Item..

[](https://www.entityframeworktutorial.net/images/codefirst/codefirst-for-existing-db1.png)

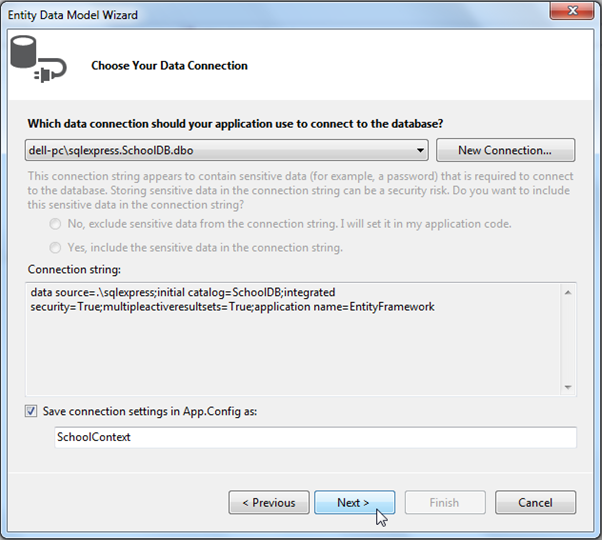
Select ADO.NET Entity Data Model in the **Add New Item** dialog box and specify the model name (this will be a context class name) and click on **Add**.

[](https://www.entityframeworktutorial.net/images/codefirst/codefirst-for-existing-db2.png)

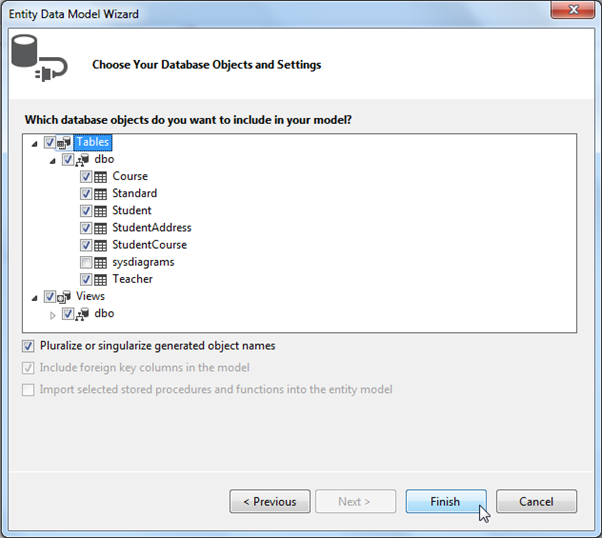
This will open the Entity Data Model wizard as shown below. Select **Code First from database** option and click **Next**.

[](https://www.entityframeworktutorial.net/images/codefirst/codefirst-for-existing-db3.png)

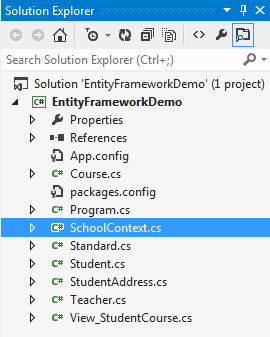
Now, select the data connection for the existing database. Create a new connection for your database if the dropdown does not include the connection to your existing database. Click **Next** to continue.

[](https://www.entityframeworktutorial.net/images/codefirst/codefirst-for-existing-db4.png)

Now, choose the tables and views for which you want to generate classes and click on **Finish**.

[](https://www.entityframeworktutorial.net/images/codefirst/codefirst-for-existing-db5.png)

This will generate all the entity classes for your DB tables and views as shown below.

[](https://www.entityframeworktutorial.net/images/codefirst/codefirst-for-existing-db6.png)

For example, it will create the following context class which uses Fluent API to configure entity classes as per your database.

namespace EFDemo

{

using System;

using System.Data.Entity;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

public partial class SchoolContext : DbContext

{

public SchoolContext()

: base("name=SchoolContext2")

{

}

public virtual DbSet<Course> Courses { get; set; }

public virtual DbSet<Standard> Standards { get; set; }

public virtual DbSet<Student> Students { get; set; }

public virtual DbSet<StudentAddress> StudentAddresses { get; set; }

public virtual DbSet<Teacher> Teachers { get; set; }

public virtual DbSet<View\_StudentCourse> View\_StudentCourse { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Entity<Course>()

.Property(e => e.CourseName)

.IsUnicode(false);

modelBuilder.Entity<Course>()

.HasMany(e => e.Students)

.WithMany(e => e.Courses)

.Map(m => m.ToTable("StudentCourse").MapLeftKey("CourseId").MapRightKey("StudentId"));

modelBuilder.Entity<Standard>()

.Property(e => e.StandardName)

.IsUnicode(false);

modelBuilder.Entity<Standard>()

.Property(e => e.Description)

.IsUnicode(false);

modelBuilder.Entity<Standard>()

.HasMany(e => e.Students)

.WithOptional(e => e.Standard)

.WillCascadeOnDelete();

modelBuilder.Entity<Standard>()

.HasMany(e => e.Teachers)

.WithOptional(e => e.Standard)

.WillCascadeOnDelete();

modelBuilder.Entity<Student>()

.Property(e => e.StudentName)

.IsUnicode(false);

modelBuilder.Entity<Student>()

.Property(e => e.RowVersion)

.IsFixedLength();

modelBuilder.Entity<Student>()

.HasOptional(e => e.StudentAddress)

.WithRequired(e => e.Student)

.WillCascadeOnDelete();

modelBuilder.Entity<StudentAddress>()

.Property(e => e.Address1)

.IsUnicode(false);

modelBuilder.Entity<StudentAddress>()

.Property(e => e.Address2)

.IsUnicode(false);

modelBuilder.Entity<StudentAddress>()

.Property(e => e.City)

.IsUnicode(false);

modelBuilder.Entity<StudentAddress>()

.Property(e => e.State)

.IsUnicode(false);

modelBuilder.Entity<Teacher>()

.Property(e => e.TeacherName)

.IsUnicode(false);

modelBuilder.Entity<Teacher>()

.HasMany(e => e.Courses)

.WithOptional(e => e.Teacher)

.WillCascadeOnDelete();

modelBuilder.Entity<View\_StudentCourse>()

.Property(e => e.StudentName)

.IsUnicode(false);

modelBuilder.Entity<View\_StudentCourse>()

.Property(e => e.CourseName)

.IsUnicode(false);

}

}

}