Entity Framework's Database Seed Method

The Entity Framework can automatically create/update/drop databases when the application runs. We can specify that this should be done every time the application runs or only when the model is out of sync with the existing database or in other words it runs whenever the model changes. You spent the time to insert records in the database and when you made any changes in the model, the Entity Framework deletes your database as well as records.

The Entity Framework recommends the use of "Database Migrations" to stop loss of a Database or of Records. Usually, we don't use "Database Migrations" in our normal application development and for this, the Entity Framework has the Database "Seed" Method that allows you to seed some dummy data in the database for testing purposes and this could be used in MVC, ASP.NET Web Forms, Windows Form apps, etc.

Creating a database initializer can be done by inheriting from either of the generic classes:

* DropCreateDatabaseIfModelChanges<DBContext>
* DropCreateDatabaseAlways<DBContext>

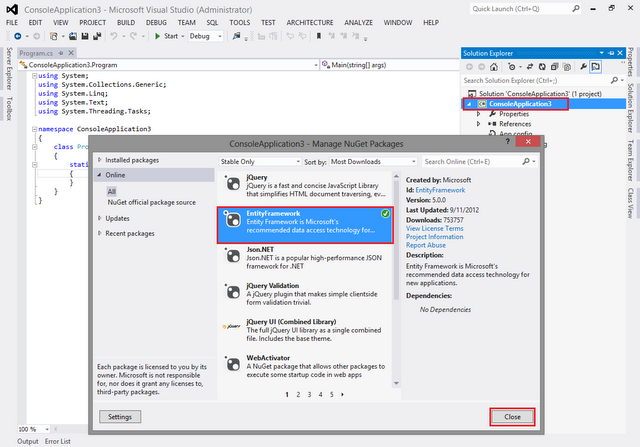
Now, follow the steps to set up a demo Console Application, and then we will explore the "Seed Method".

**Step 1: Create a New Project**

Create a new console application "File" > "New" > "Project..." then "Visual C#" > "Console Application".

**Step 2: Install EF5 from NuGet**

At the very first, we need to enable the Entity Framework 5 for this console project from the NuGet Package Manager. For this, in "Solution Explorer" right-click on the project and click on "Manage NuGet Packages" and install Entity Framework 5. Alternatively, you can install this using the power console using the command "Install-Package EntityFramework".

[[](https://1.bp.blogspot.com/-ZdrJ2Rlb6ic/UHKK-jIWdlI/AAAAAAAACds/47Rr9f_DDZM/s1600/1.png)](https://1.bp.blogspot.com/-ZdrJ2Rlb6ic/UHKK-jIWdlI/AAAAAAAACds/47Rr9f_DDZM/s1600/1.png)

**Step 3: Create a Model**

Now to define a new model using the Code First Approach.

**public** **class** Student

{

**public** **int** StudentId { **get**; **set**; }

**public** **string** Name { **get**; **set**; }

**public** **string** Address { **get**; **set**; }

}

**Step 4: Create DbContext**

Now, go ahead and define the DbContext.

**public** **class** StudentContext : DbContext

{

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

}

Note: To use the namespace put "using System.Data.Entity;" at the top.

**Step 5: Database Initializer (Dummy Data)**

Now we are done with the Model and DbContext. Let's decide what data we want to initialize when the database is first created. This can be done using any class that inherits DropCreateDatabaseIfModelChanges<StudentContext> or DropCreateDatabaseAlways<StudentContext>. The first one will be called when the model is changed and the second one will be called each time we run the application. I'm using the first one. We can place this code in a separate class file also.

**public** **class** StudentDbInitializer : DropCreateDatabaseIfModelChanges<StudentContext>

{

**protected** **override** **void** Seed(StudentContext context)

    {

        var students = **new** List<Student>

        {

**new** Student { Name = "Abhimanyu K Vatsa", Address = "Bokaro" },

**new** Student { Name = "Deepak Kumar Gupta", Address = "Bokaro" },

**new** Student { Name = "Manish Kumar", Address = "Muzaffarpur" },

**new** Student { Name = "Rohit Ranjan", Address = "Patna" },

**new** Student { Name = "Shivam", Address = "Motihari" }

        };

        students.ForEach(s => context.Students.Add(s));

        context.SaveChanges();

    }

}

**Step 6: Seed Dummy Data to Database**

So, we have dummy data that I will seed to the database and the database will be created at run time. Now, we need to make a call, here you go.

Database.SetInitializer<StudentContext>(**new** StudentDbInitializer()); 

**Step 6: Complete Code**

Find the complete code of my console application.

**using** System;

**using** System.Collections.Generic;

**using** System.Data.Entity;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** ConsoleApplication5\_Seed {

**public** **class** Student {

**public** **int** StudentId { **get**;

**set**; }

**public** **string** Name { **get**;

**set**; }

**public** **string** Address { **get**;

**set**; }

    }

**public** **class** StudentContext: DbContext {

**public** DbSet < Student > Students { **get**;

**set**; }

    }

**public** **class** StudentDbInitializer: DropCreateDatabaseIfModelChanges < StudentContext > {

**protected** **override** **void** Seed(StudentContext context) {

            var students = **new** List < Student > {

**new** Student { Name = "Abhimanyu K Vatsa", Address = "Bokaro" },

**new** Student { Name = "Deepak Kumar Gupta", Address = "Bokaro" },

**new** Student { Name = "Manish Kumar", Address = "Muzaffarpur" },

**new** Student { Name = "Rohit Ranjan", Address = "Patna" },

**new** Student { Name = "Shivam", Address = "Motihari" }

            };

            students.ForEach(s => context.Students.Add(s));

            context.SaveChanges();

        }

    }

**class** Program {

**static** **void** Main(**string**[] args) {

**using**(var context = **new** StudentContext()) {

                Database.SetInitializer < StudentContext > (**new** StudentDbInitializer());

                var stdQuery = (from d **in** context.Students select **new** { Id = d.StudentId, Name = d.Name });

**foreach**(var q **in** stdQuery) {

                    Console.WriteLine("ID : " + q.Id + ", Name : " + q.Name);

                }

                Console.ReadKey();

            }

        }

    }

}

Now, if you run the application you will see all the dummy data that we have created and initialized above.

**Step 7: MVC Apps**

In MVC we call the SetInitializer method from Application\_Start() method that can be found in Global.asax.cs file. Find here a sample:

**protected** **void** Application\_Start()

{

    System.Data.Entity.Database.SetInitializer(**new** StudentDbInitializer());

    AreaRegistration.RegisterAllAreas();

    WebApiConfig.Register(GlobalConfiguration.Configuration);

    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

    RouteConfig.RegisterRoutes(RouteTable.Routes);

}