Let's assume that we want to create a simple application for XYZ School. User of this School application should be able to add or update Students, Standard (Grade), Teacher and Course information.

So, instead of designing database tables first, let's start creating classes for our school domain as and when needed. First, we will create two simple Student and Standard classes where every Student is associated with one Standard as shown below.

public class Student

{

public Student()

{

}

public int StudentID { get; set; }

public string StudentName { get; set; }

public DateTime? DateOfBirth { get; set; }

public byte[] Photo { get; set; }

public decimal Height { get; set; }

public float Weight { get; set; }

public Standard Standard { get; set; }

}

The Standard (Grade) class should be able to accomodate multiple Students as shown below.

public class Standard

{

public Standard()

{

}

public int StandardId { get; set; }

public string StandardName { get; set; }

public ICollection<Student> Students { get; set; }

}

Now, we are done with the initial domain classes for our school application. Code-First approach also requires context class which should be derived from DbContext as we created in the basic tutorials section using database-frist approach. .

Create a context class as shown below, which derives DBContext class and exposes DbSet properties for the types that you want to be part of the model e.g. Student and Standard class in this case. DbSet is a collection of entity class aka entity set, so we have given property name as plural of entity name like Students and Standards.

namespace EF\_Code\_First

{

public class SchoolContext: DbContext

{

public SchoolContext(): base()

{

}

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

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

}

}

So now, we are done with the required classes for the code-first approach. We will now add student using context class as shown below.

class Program

{

static void Main(string[] args)

{

using (var ctx = new SchoolContext())

{

Student stud = new Student() { StudentName = "New Student" };

ctx.Students.Add(stud);

ctx.SaveChanges();

}

}

}

Now, if you run the application, then you will be surprised to see that application run successfully and one student is successfully inserted into the database.

**But.. where is the database and what are the tables and its columns?**

This is the beauty of Code-First APIs of Entity Framework. It creates the database based on parameter passed in the base constructor of your context class. We have not passed any parameter in the constructor of our context class, so it creates "EF\_Code\_First\_Tutorials.SchoolContext" database in the local SQLEXPRESS database as shown below. It also creates two tables in this database, Students and Standards table based on Student and Standard domain classes defined above.