Instead of designing database tables first, let's start creating classes for our school domain, as and when needed. First, create the Student and Grade classes where every Student is associated with one Grade as shown below. This is called a one-to-many relationship. Learn about how EF manages the relationship between entities (domain classes) [here](http://www.entityframeworktutorial.net/entity-relationships.aspx).

public class 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 Grade Grade { get; set; }

}

Create the Grade class as shown below.

public class Grade

{

public int GradeId { get; set; }

public string GradeName { get; set; }

public string Section { get; set; }

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

}

Now, we are done with the initial domain classes for our school application.

The Code-First approach also requires a [context class](http://www.entityframeworktutorial.net/basics/context-class-in-entity-framework.aspx) which should be derived from DbContext class. Create a context class as shown below. It derives from DBContext class and exposes DbSetproperties for the types that you want to be part of the model, e.g. Student and Grade classes in this case. The DbSet is a collection of entity classes (aka entity set), so we have given the property name as the plural of entity name like Students and Grades.

namespace EF6Console

{

public class SchoolContext: DbContext

{

public SchoolContext(): base()

{

}

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

public DbSet<Grade> Grades { get; set; }

}

}

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

namespace EF6Console

{

class Program

{

static void Main(string[] args)

{

using (var ctx = new SchoolContext())

{

var stud = new Student() { StudentName = "Bill" };

ctx.Students.Add(stud);

ctx.SaveChanges();

}

}

}

}

If you run the application, you will see that one student is successfully inserted into the database.

**But where is the database and which are the tables and their columns?**

This is the beauty of EF Code-First API. It creates the database based on the parameter passed in the base constructor of your context class. Since we have not passed any parameter in the constructor of our context class, it created **EF6Console.SchoolContext** database in the local SQLEXPRESS database, as shown below. It also created two tables in this database, Students and Standards based on Student and Grade domain classes defined above.

As you can see in the above figure, it has created Students and Grades tables and each table contains columns with appropriate datatype and length. The column names and datatype match the properties of the respective domain classes. It has also made StudentId and GradeId as PKs (primary keys) and created Grade\_GradeId column as FK (foreign key).

This way, without creating a database first, you can start writing an application that will eventually create a database from your domain classes.