< Previous

Next >

Conventions in Entity Framework Core

Conventions are default rules using which Entity Framework builds a model based on your domain (entity) classes. In the First EF Core Application chapter, EF Core API creates a database schema based on domain and context classes, without any additional configurations because domain classes were following the conventions.

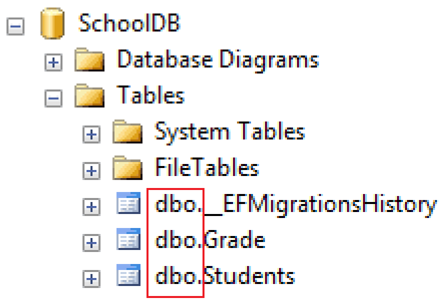
Consider the following sample entities and context class to understand the default conventions.

```
public class Student
{
    public int StudentId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public byte[] Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    public int GradeId { get; set; }
    public Grade Grade { get; set; }
}

public class Grade
{
    public int Id { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

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

public class SchoolContext : DbContext
{
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {

optionsBuilder.UseSqlServer(@"Server=.\SQLEXPRESS;Database=SchoolDB;Trusted_Connectio

    }

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

Let's understand the EF Core conventions and how EF Core API will create a database for the above entities.

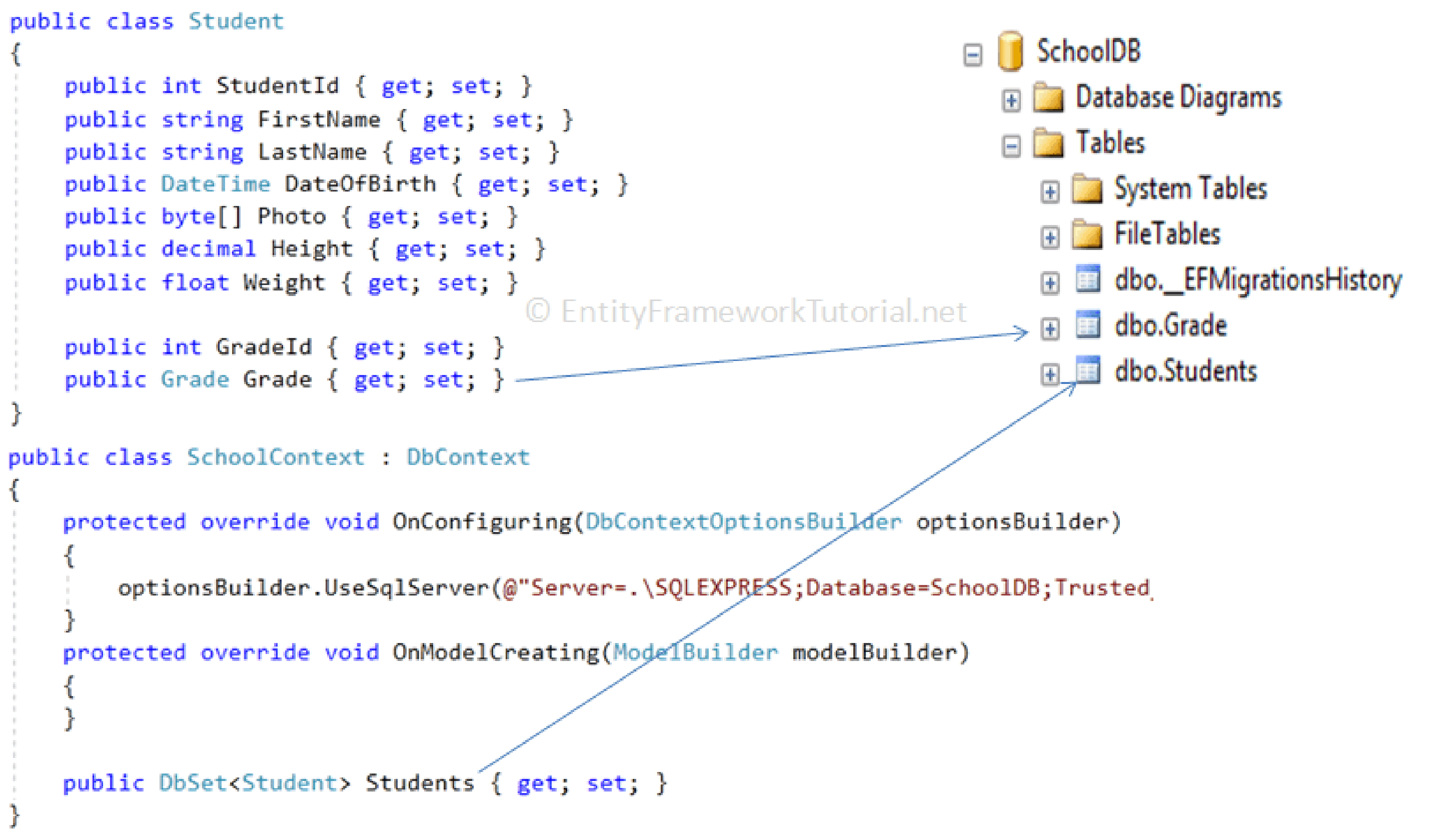
Schema

EF Core will create all the database objects in the **dbo** schema by default.



Table

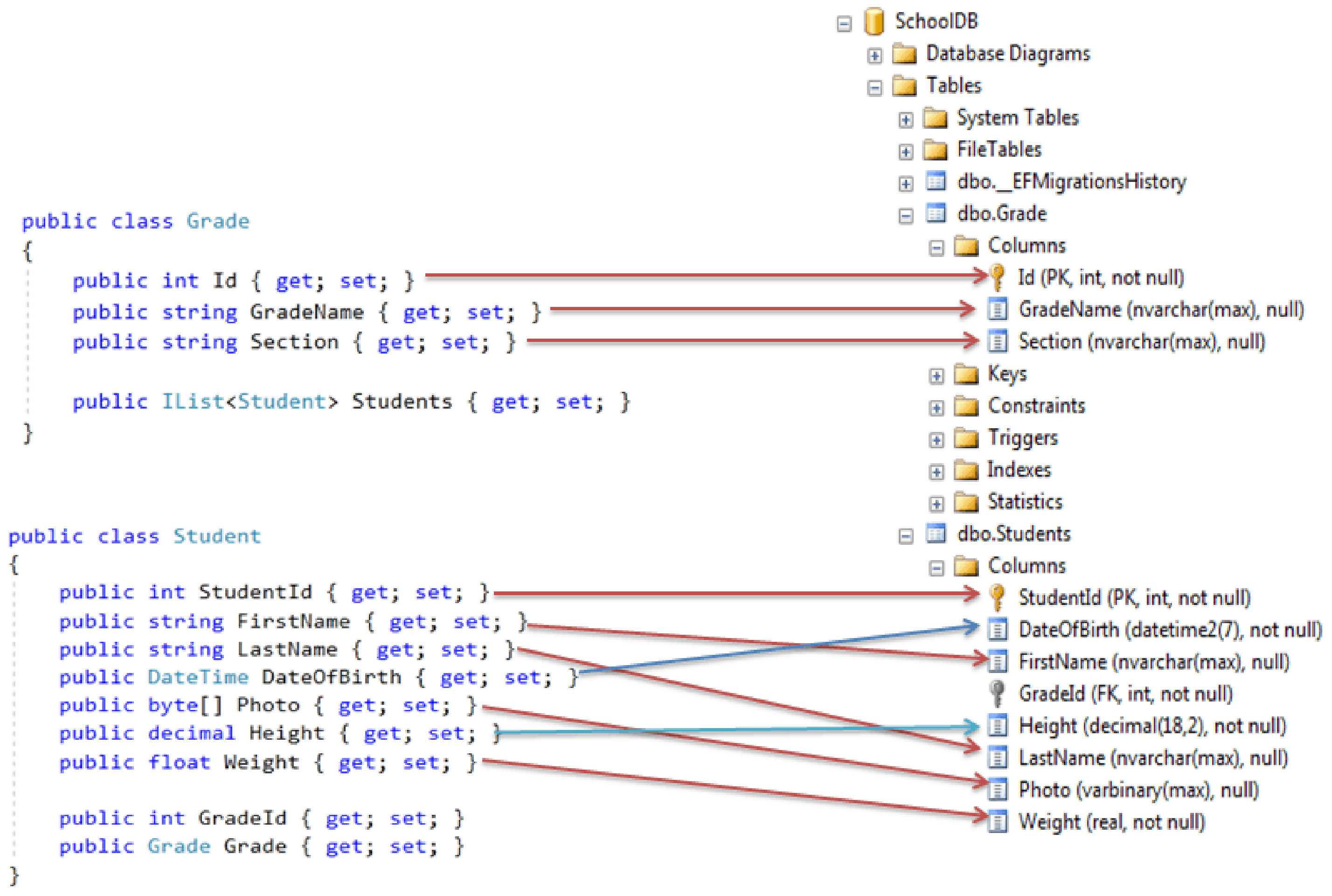
EF Core will create database tables for all `DbSet<TEntity>` properties in a context class with the same name as the property. It will also create tables for entities which are not included as `DbSet` properties but are reachable through reference properties in other `DbSet` entities. For the above example, EF Core will create the `Students` table for `DbSet<Student>` property in the `SchoolContext` class and the `Grade` table for a `Grade` property in the `Student` entity class, even though the `SchoolContext` class does not include the `DbSet<Grade>` property.



```
public class Student
{
    public int StudentId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public byte[] Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    public int GradeId { get; set; }
    public Grade Grade { get; set; }
}
public class SchoolContext : DbContext
{
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=.\SQLEXPRESS;Database=SchoolDB;Trusted
    }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
    }

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

Column

EF Core will create columns for all the scalar properties of an entity class with the same name as the property, by default. It uses the reference and collection properties in building relationships among corresponding tables in the database.



```
public class Grade
{
    public int Id { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

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

public class Student
{
    public int StudentId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public byte[] Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    public int GradeId { get; set; }
    public Grade Grade { get; set; }
}
```

```
SchoolDB
  Database Diagrams
  Tables
    System Tables
    FileTables
    dbo._EFMigrationsHistory
    dbo.Grade
      Columns
        Id (PK, int, not null)
        GradeName (nvarchar(max), null)
        Section (nvarchar(max), null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
    dbo.Students
      Columns
        StudentId (PK, int, not null)
        DateOfBirth (datetime2(7), not null)
        FirstName (nvarchar(max), null)
        GradeId (FK, int, not null)
        Height (decimal(18,2), not null)
        LastName (nvarchar(max), null)
        Photo (varbinary(max), null)
        Weight (real, not null)
```

Column Data Type

The data type for columns in the database table is depending on how the provider for the database has mapped C# data type to the data type of a selected database. The following table lists mapping between C# data type to SQL Server column data type.

C# Data Type	Mapping to SQL Server Data Type
int	int
string	nvarchar(Max)
decimal	decimal(18,2)
float	real
byte[]	varbinary(Max)

datetime	datetime
bool	bit
byte	tinyint
short	smallint
long	bigint
double	float
char	No mapping
sbyte	No mapping (throws exception)
object	No mapping

ADVERTISEMENT

Nullable Column

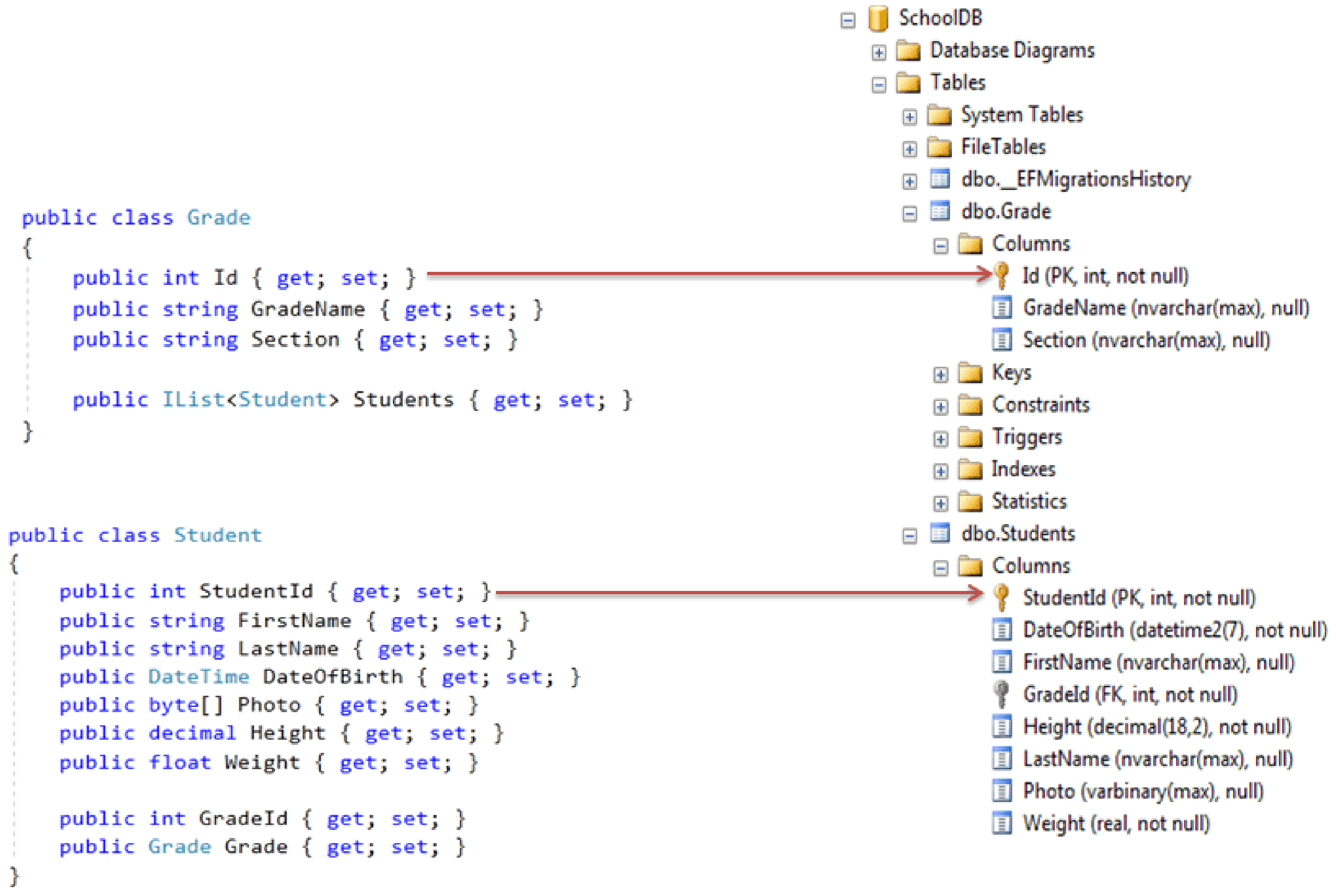
EF Core creates null columns for all reference data type and nullable primitive type properties e.g. string, Nullable<int>, decimal?.

NotNull Column

EF Core creates NotNull columns in the database for all primary key properties, and primitive type properties e.g. int, float, decimal, DateTime etc..

Primary Key

EF Core will create the primary key column for the property named `Id` or `<Entity Class Name>Id` (case insensitive). For example, EF Core will create a column as PrimaryKey in the `Students` table if the `Student` class includes a property named id, ID, iD, Id, studentid, StudentId, STUDENTID, or sTUdentID.



```
public class Grade
{
    public int Id { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

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


public class Student
{
    public int StudentId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public byte[] Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    public int GradeId { get; set; }
    public Grade Grade { get; set; }
}
```

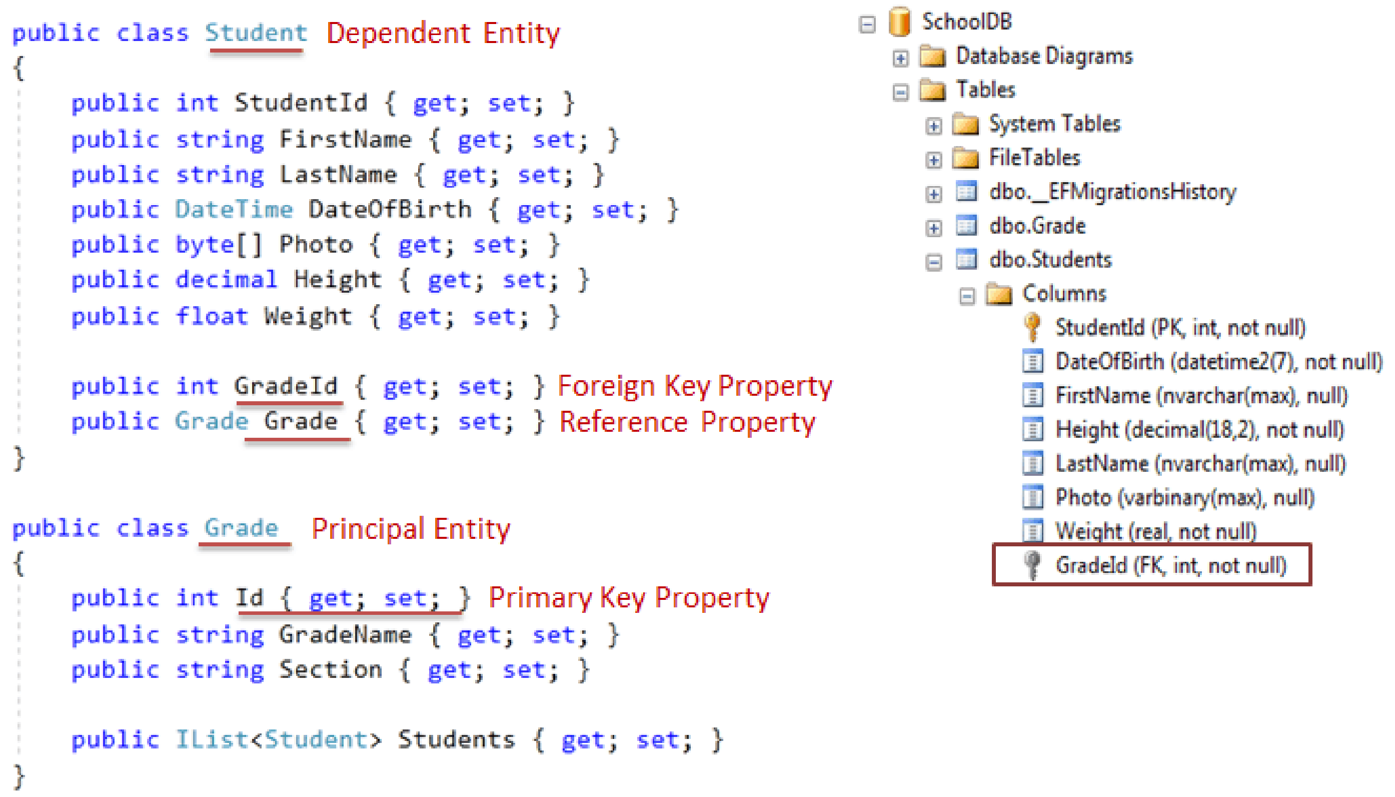
```
SchoolDB
  Database Diagrams
  Tables
    System Tables
    FileTables
    dbo.__EFMigrationsHistory
    dbo.Grade
      Columns
        Id (PK, int, not null)
        GradeName (nvarchar(max), null)
        Section (nvarchar(max), null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
    dbo.Students
      Columns
        StudentId (PK, int, not null)
        DateOfBirth (datetime2(7), not null)
        FirstName (nvarchar(max), null)
        GradeId (FK, int, not null)
        Height (decimal(18,2), not null)
        LastName (nvarchar(max), null)
        Photo (varbinary(max), null)
        Weight (real, not null)
```

Foreign Key

As per the foreign key convention, EF Core API will create a foreign key column for each reference navigation property in an entity with one of the following naming patterns.

- `<Reference Navigation Property Name>Id`
- `<Reference Navigation Property Name><Principal Primary Key Property Name>`

In our example (`Student` and `Grade` entities), EF Core will create a foreign key column `GradeId` in the `Students` table, as depicted in the following figure.



```
public class Student
{
    public int StudentId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public byte[] Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    public int GradeId { get; set; }
    public Grade Grade { get; set; }
}

public class Grade
{
    public int Id { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

    public IList<Student> Students { get; set; }
}
```

```
SchoolDB
  Database Diagrams
  Tables
    System Tables
    FileTables
    dbo.__EFMigrationsHistory
    dbo.Grade
    dbo.Students
      Columns
        StudentId (PK, int, not null)
        DateOfBirth (datetime2(7), not null)
        FirstName (nvarchar(max), null)
        Height (decimal(18,2), not null)
        LastName (nvarchar(max), null)
        Photo (varbinary(max), null)
        Weight (real, not null)
        GradeId (FK, int, not null)
```

The following table lists foreign key column names for different reference property names and primary key property names.

Reference Property Name in Dependent Entity	Foreign Key Property Name in Dependent Entity	Principal Primary Key Property Name	Foreign Key Column Name in DB
Grade	GradeId	GradeId	GradeId
Grade	-	GradeId	GradeId
Grade	-	Id	GradeId
CurrentGrade	CurrentGradeId	GradeId	CurrentGradeId
CurrentGrade	-	GradeId	CurrentGradeGradeId
CurrentGrade	-	Id	CurrentGradeId
CurrentGrade	GradeId	Id	GradeId

Index

EF Core creates a clustered index on Primarykey columns and a non-clustered index on ForeignKey columns, by default.

Learn about relationship conventions in the next chapter.

‹ Previous Next ›