

# Entity Framework

## Notes for Professionals

### Chapter 2: Code First Conventions

#### Section 2.1: Removing Conventions

You can remove any of the conventions defined in the `System.Data.Entity.ModelConfiguration` namespace, by overriding `OnModelCreating` method.

The following example removes `PluralizingTableNameConvention`.

```
public class BlogContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
    }
}
```

#### Section 2.2: Primary Key Convention

By default EF will create DB table with entity class name suffixed by 's'. In this example ignore `PluralizingTableNameConvention` on `Products` table also. `Products` table also. `Products` table also.

By default a property is a primary key if a property on a class is named "ID" (not case followed by "ID", if the type of the primary key property is numeric or GUID it will be column. Simple Example:

```
public class Blog
{
    // Primary key
    public int BlogId { get; set; }
}
```

#### Section 2.3: Type Discovery

By default Code First includes in model

- Types defined as a DbSet property in context class.
- Reference types included in entity types even if they are defined in derived classes even if only the base class is defined as DbSet property.
- Derived classes even if only the base class is defined as DbSet property.

Here is an example, that we are only adding company as DbSet property:

```
public class Company
{
    public int Id { get; set; }
    public string Name { get; set; }
    public virtual ICollection<Department> Departments { get; set; }
}

public class Department
{
    public int Id { get; set; }
}
```

### Chapter 3: Code First DataAnnotations

#### Section 3.1: [Column] attribute

```
public class Person
{
    public int PersonID { get; set; }

    [Column("nameOfPerson")]
    public string PersonName { get; set; }
}
```

Tells Entity Framework to use a specific column name instead using the name of the property. You can also specify the database data type and the order of the column in table:

```
[Column("nameOfPerson", TypeName = "varchar", Order = 1)]
public string PersonName { get; set; }
```

#### Section 3.2: [DatabaseGenerated] attribute

Specifies how the database generates values for the property. There are three possible values:

- None specifies that the values are not generated by the database.
- Identity specifies that the column is an `identity column`, which is typically used for integer primary keys.
- Computed specifies that the database generates the value for the column.

If the value is anything other than `None`, Entity Framework will not commit changes made to the property back to the database.

By default (based on the `StoreGeneratedIdentityColumnConvention`) an integer key property will be treated as an identity column. To override this convention and force it to be treated as a non-identity column you can use the `DatabaseGeneratedAttribute` with a value of `None`.

```
using System.ComponentModel.DataAnnotations.Schema;
```

```
public class Foo
{
    [Key]
    public int Id { get; set; } // identity (auto-increment) column
}

public class Bar
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
    public int Id { get; set; } // non-identity column
}
```

The following SQL creates a table with a computed column:

```
CREATE TABLE [Person] (
    Name varchar(150) PRIMARY KEY,
    DateOfBirth Date NOT NULL,
    Age AS DATEPART(year, DateOfBirth) - GETDATE()
)
```

Entity Framework Notes for Professionals

### Chapter 14: Transactions

#### Section 14.1: Database.BeginTransaction()

Multiple operations can be executed against a single transaction so that changes can be rolled back if any of the operations fail.

```
using (var context = new PlanetContext())
using (var transaction = context.Database.BeginTransaction())
{
    try
    {
        //lets assume this works
        var jupiter = new Planet { Name = "Jupiter" };
        context.Planets.Add(jupiter);
        context.SaveChanges();

        //and then this will throw an exception
        var neptune = new Planet { Name = "Neptune" };
        context.Planets.Add(neptune);
        context.SaveChanges();

        //Without this line, no changes will get applied to the database
        transaction.Commit();

        //there is no need to call transaction.Rollback() here as the transaction object
        //will go out of scope and disposing will roll back automatically
    }
}
```

Note that it may be a developers' convention to call `transaction.Rollback()` explicitly, because it makes the code more self-explanatory. Also, there may be (less well-known) query providers for Entity Framework out there that don't implement `Dispose` correctly, which would also require an explicit `transaction.Rollback()` call.

Entity Framework Notes for Professionals

**80+ pages**  
of professional hints and tricks

# Contents

<b>About</b>	1
<b>Chapter 1: Getting started with Entity Framework</b>	2
<a href="#">Section 1.1: Installing the Entity Framework NuGet Package</a>	2
<a href="#">Section 1.2: Using Entity Framework from C# (Code First)</a>	4
<a href="#">Section 1.3: What is Entity Framework?</a>	5
<b>Chapter 2: Code First Conventions</b>	6
<a href="#">Section 2.1: Removing Conventions</a>	6
<a href="#">Section 2.2: Primary Key Convention</a>	6
<a href="#">Section 2.3: Type Discovery</a>	6
<a href="#">Section 2.4: DecimalPropertyConvention</a>	7
<a href="#">Section 2.5: Relationship Convention</a>	9
<a href="#">Section 2.6: Foreign Key Convention</a>	10
<b>Chapter 3: Code First DataAnnotations</b>	11
<a href="#">Section 3.1: [Column] attribute</a>	11
<a href="#">Section 3.2: [DatabaseGenerated] attribute</a>	11
<a href="#">Section 3.3: [Required] attribute</a>	12
<a href="#">Section 3.4: [MaxLength] and [MinLength] attributes</a>	12
<a href="#">Section 3.5: [InverseProperty(string)] attribute</a>	13
<a href="#">Section 3.6: [ComplexType] attribute</a>	14
<a href="#">Section 3.7: [ForeignKey(string)] attribute</a>	15
<a href="#">Section 3.8: [Range(min,max)] attribute</a>	15
<a href="#">Section 3.9: [NotMapped] attribute</a>	16
<a href="#">Section 3.10: [Table] attribute</a>	17
<a href="#">Section 3.11: [Index] attribute</a>	17
<a href="#">Section 3.12: [Key] attribute</a>	18
<a href="#">Section 3.13: [StringLength(int)] attribute</a>	19
<a href="#">Section 3.14: [Timestamp] attribute</a>	19
<a href="#">Section 3.15: [ConcurrencyCheck] Attribute</a>	20
<b>Chapter 4: Entity Framework Code First</b>	21
<a href="#">Section 4.1: Connect to an existing database</a>	21
<b>Chapter 5: Entity framework Code First Migrations</b>	23
<a href="#">Section 5.1: Enable Migrations</a>	23
<a href="#">Section 5.2: Add your first migration</a>	23
<a href="#">Section 5.3: Doing "Update-Database" within your code</a>	25
<a href="#">Section 5.4: Seeding Data during migrations</a>	25
<a href="#">Section 5.5: Initial Entity Framework Code First Migration Step by Step</a>	26
<a href="#">Section 5.6: Using Sql() during migrations</a>	27
<b>Chapter 6: Inheritance with EntityFramework (Code First)</b>	29
<a href="#">Section 6.1: Table per hierarchy</a>	29
<a href="#">Section 6.2: Table per type</a>	29
<b>Chapter 7: Code First - Fluent API</b>	31
<a href="#">Section 7.1: Mapping models</a>	31
<a href="#">Section 7.2: Composite Primary Key</a>	32
<a href="#">Section 7.3: Maximum Length</a>	33
<a href="#">Section 7.4: Primary Key</a>	33
<a href="#">Section 7.5: Required properties (NOT NULL)</a>	34

Section 7.6: Explicit Foreign Key naming .....	34
<b>Chapter 8: Mapping relationship with Entity Framework Code First: One-to-one and variations</b> .....	36
Section 8.1: Mapping one-to-zero or one .....	36
Section 8.2: Mapping one-to-one .....	39
Section 8.3: Mapping one or zero-to-one or zero .....	40
<b>Chapter 9: Mapping relationship with Entity Framework Code First: One-to-many and Many-to-many</b> .....	41
Section 9.1: Mapping one-to-many .....	41
Section 9.2: Mapping one-to-many: against the convention .....	42
Section 9.3: Mapping zero or one-to-many .....	43
Section 9.4: Many-to-many .....	44
Section 9.5: Many-to-many: customizing the join table .....	45
Section 9.6: Many-to-many: custom join entity .....	46
<b>Chapter 10: Database first model generation</b> .....	49
Section 10.1: Generating model from database .....	49
Section 10.2: Adding data annotations to the generated model .....	50
<b>Chapter 11: Complex Types</b> .....	52
Section 11.1: Code First Complex Types .....	52
<b>Chapter 12: Database Initialisers</b> .....	53
Section 12.1: CreateDatabaseIfNotExists .....	53
Section 12.2: DropCreateDatabaseIfModelChanges .....	53
Section 12.3: DropCreateDatabaseAlways .....	53
Section 12.4: Custom database initializer .....	53
Section 12.5: MigrateDatabaseToLatestVersion .....	54
<b>Chapter 13: Tracking vs. No-Tracking</b> .....	55
Section 13.1: No-tracking queries .....	55
Section 13.2: Tracking queries .....	55
Section 13.3: Tracking and projections .....	55
<b>Chapter 14: Transactions</b> .....	57
Section 14.1: Database.BeginTransaction() .....	57
<b>Chapter 15: Managing entity state</b> .....	58
Section 15.1: Setting state Added of a single entity .....	58
Section 15.2: Setting state Added of an object graph .....	58
<b>Chapter 16: Loading related entities</b> .....	60
Section 16.1: Eager loading .....	60
Section 16.2: Explicit loading .....	60
Section 16.3: Lazy loading .....	61
Section 16.4: Projection Queries .....	61
<b>Chapter 17: Model Restraints</b> .....	63
Section 17.1: One-to-many relationships .....	63
<b>Chapter 18: Entity Framework with PostgreSQL</b> .....	65
Section 18.1: Pre-Steps needed in order to use Entity Framework 6.1.3 with PostgresSql using NpgsqlDdexpProvider .....	65
<b>Chapter 19: Entity Framework with SQLite</b> .....	66
Section 19.1: Setting up a project to use Entity Framework with an SQLite provider .....	66
<b>Chapter 20: .t4 templates in entity framework</b> .....	69
Section 20.1: Dynamically adding Interfaces to model .....	69

Section 20.2: Adding XML Documentation to Entity Classes .....	69
<b>Chapter 21: Advanced mapping scenarios: entity splitting, table splitting .....</b>	<b>71</b>
Section 21.1: Entity splitting .....	71
Section 21.2: Table splitting .....	72
<b>Chapter 22: Best Practices For Entity Framework (Simple &amp; Professional) .....</b>	<b>73</b>
Section 22.1: 1- Entity Framework @ Data layer (Basics) .....	73
Section 22.2: 2- Entity Framework @ Business layer .....	76
Section 22.3: 3- Using Business layer @ Presentation layer (MVC) .....	79
Section 22.4: 4- Entity Framework @ Unit Test Layer .....	81
<b>Chapter 23: Optimization Techniques in EF .....</b>	<b>85</b>
Section 23.1: Using AsNoTracking .....	85
Section 23.2: Execute queries in the database when possible, not in memory .....	85
Section 23.3: Loading Only Required Data .....	85
Section 23.4: Execute multiple queries async and in parallel .....	86
Section 23.5: Working with stub entities .....	87
Section 23.6: Disable change tracking and proxy generation .....	88
<b>Credits .....</b>	<b>89</b>
<b>You may also like .....</b>	<b>90</b>

# About

Please feel free to share this PDF with anyone for free,  
latest version of this book can be downloaded from:  
<https://goalkicker.com/EntityFrameworkBook>

This *Entity Framework Notes for Professionals* book is compiled from [Stack Overflow Documentation](#), the content is written by the beautiful people at Stack Overflow. Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official Entity Framework group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to [web@petercv.com](mailto:web@petercv.com)

[Click here to download full PDF material](#)