






# Improve Entity Framework Performance



 Bulk Insert
  Bulk Delete

 Bulk Update
  Bulk Merge

[LEARN MORE](#)

[< Previous](#)
[Next >](#)

## Fluent API in Entity Framework Core

Entity Framework Fluent API is used to configure domain classes to override conventions. EF Fluent API is based on a Fluent API design pattern (a.k.a [Fluent Interface](#)) where the result is formulated by [method chaining](#).

In Entity Framework Core, the [ModelBuilder](#) class acts as a Fluent API. By using it, we can configure many different things, as it provides more configuration options than data annotation attributes.

Entity Framework Core Fluent API configures the following aspects of a model:

1. **Model Configuration:** Configures an EF model to database mappings. Configures the default Schema, DB functions, additional data annotation attributes and entities to be excluded from mapping.
2. **Entity Configuration:** Configures entity to table and relationships mapping e.g. PrimaryKey, AlternateKey, Index, table name, one-to-one, one-to-many, many-to-many relationships etc.
3. **Property Configuration:** Configures property to column mapping e.g. column name, default value, nullability, Foreignkey, data type, concurrency column etc.

The following table lists important methods for each type of configuration.

Configurations	Fluent API Methods	Usage
Model Configurations	HasDbFunction()	Configures a database function when targeting a relational database.
	HasDefaultSchema()	Specifies the database schema.
	HasAnnotation()	Adds or updates data annotation attributes on the entity.

	HasSequence()	Configures a database sequence when targeting a relational database.
Entity Configuration	HasAlternateKey()	Configures an alternate key in the EF model for the entity.
	HasIndex()	Configures an index of the specified properties.
	HasKey()	Configures the property or list of properties as Primary Key.
	HasMany()	Configures the Many part of the relationship, where an entity contains the reference collection property of other type for one-to-Many or many-to-many relationships.
	HasOne()	Configures the One part of the relationship, where an entity contains the reference property of other type for one-to-one or one-to-many relationships.
	Ignore()	Configures that the class or property should not be mapped to a table or column.
	OwnsOne()	Configures a relationship where the target entity is owned by this entity. The target entity key value is propagated from the entity it belongs to.
	.ToTable()	Configures the database table that the entity maps to.
Property Configuration	HasColumnName()	Configures the corresponding column name in the database for the property.
	HasColumnType()	Configures the data type of the corresponding column in the database for the property.
	HasComputedColumnSql()	Configures the property to map to computed column in the database when targeting a relational database.

HasDefaultValue()	Configures the default value for the column that the property maps to when targeting a relational database.
HasDefaultValueSql()	Configures the default value expression for the column that the property maps to when targeting relational database.
HasField()	Specifies the backing field to be used with a property.
HasMaxLength()	Configures the maximum length of data that can be stored in a property.
IsConcurrencyToken()	Configures the property to be used as an optimistic concurrency token.
IsRequired()	Configures whether the valid value of the property is required or whether null is a valid value.
IsRowVersion()	Configures the property to be used in optimistic concurrency detection.
IsUnicode()	Configures the string property which can contain unicode characters or not.
ValueGeneratedNever()	Configures a property which cannot have a generated value when an entity is saved.
ValueGeneratedOnAdd()	Configures that the property has a generated value when saving a new entity.
ValueGeneratedOnAddOrUpdate()	Configures that the property has a generated value when saving new or existing entity.
ValueGeneratedOnUpdate()	Configures that a property has a generated value when saving an existing entity.

---

ADVERTISEMENT

---

## Fluent API Configurations

Override the `OnModelCreating` method and use a parameter `modelBuilder` of type `ModelBuilder` to configure domain classes, as shown below.

```
public class SchoolDbContext: DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        //Write Fluent API configurations here

        //Property Configurations
        modelBuilder.Entity<Student>()
            .Property(s => s.StudentId)
            .HasColumnName("Id")
            .HasDefaultValue(0)
            .IsRequired();
    }
}
```

In the above example, the `ModelBuilder` Fluent API instance is used to configure a property by calling multiple methods in a chain. It configures the `StudentId` property of the `Student` entity; it configures the name using `HasColumnName`, the default value using `HasDefaultValue` and nullability using `IsRequired` method in a single statement instead of multiple statements. This increases the readability and also takes less time to write compare to multiple statements, as shown below.

```
//Fluent API method chained calls
modelBuilder.Entity<Student>()
    .Property(s => s.StudentId)
    .HasColumnName("Id")
    .HasDefaultValue(0)
    .IsRequired();

//Separate method calls
modelBuilder.Entity<Student>().Property(s => s.StudentId).HasColumnName("Id");
modelBuilder.Entity<Student>().Property(s => s.StudentId).HasDefaultValue(0);
modelBuilder.Entity<Student>().Property(s => s.StudentId).IsRequired();
```

**Note:** Fluent API configurations have higher precedence than data annotation attributes.

Learn how to configure One-to-Many relationships using Fluent API next.

---

[< Previous](#)[Next >](#)

---

## ENTITYFRAMEWORKTUTORIAL

Learn Entity Framework using simple yet practical examples on EntityFrameworkTutorial.net for free. Learn Entity Framework DB-First, Code-First and EF Core step by step. While using this site, you agree to have read and accepted our terms of use and privacy policy.

---

✉ [feedback@entityframeworktutorial.net](mailto:feedback@entityframeworktutorial.net)

## TUTORIALS

- › EF Basics
- › EF Core
- › EF 6 DB-First
- › EF 6 Code-First

## E-MAIL LIST

Subscribe to EntityFrameworkTutorial email list and get EF 6 and EF Core Cheat Sheets, latest updates, tips & tricks about Entity Framework to your inbox.

Email address

GO

We respect your privacy.

---

[HOME](#) [PRIVACY POLICY](#) [ADVERTISE WITH US](#)

© 2020 EntityFrameworkTutorial.net. All Rights Reserved.