Entity Framework

Table of Contents

[Steps 3](#_Toc27424005)

[Create new project 3](#_Toc27424006)

[Model 3](#_Toc27424007)

[Connection String 3](#_Toc27424008)

[DBContext 3](#_Toc27424009)

[Enable Migrations 4](#_Toc27424010)

[Add Migration 4](#_Toc27424011)

[Update Migration 4](#_Toc27424012)

[Package Manager Console 4](#_Toc27424013)

[Enable Migrations 4](#_Toc27424014)

[Add Migration 4](#_Toc27424015)

[Update Migration 4](#_Toc27424016)

[Data Annotation 4](#_Toc27424017)

[Key 4](#_Toc27424018)

[Composite keys 4](#_Toc27424019)

[Required 4](#_Toc27424020)

[MaxLength and MinLength 5](#_Toc27424021)

[NotMapped 5](#_Toc27424022)

[ComplexType 6](#_Toc27424023)

[ConcurrencyCheck 6](#_Toc27424024)

[TimeStamp 6](#_Toc27424025)

[Table 6](#_Toc27424026)

[Column 7](#_Toc27424027)

[Index 7](#_Toc27424028)

[Unique Index 7](#_Toc27424029)

[Multiple-Column Indexes 7](#_Toc27424030)

[Relationships 7](#_Toc27424031)

[Fluent API 7](#_Toc27424032)

[Default Schema 8](#_Toc27424033)

[Table Name 8](#_Toc27424034)

[Primary Key 8](#_Toc27424035)

[Primary Key without auto number 8](#_Toc27424036)

[Composite Primary Key 8](#_Toc27424037)

[Column 8](#_Toc27424038)

[Index 8](#_Toc27424039)

[Multiple Column Index 8](#_Toc27424040)

[Not Map 8](#_Toc27424041)

[Complex Type 8](#_Toc27424042)

[One-to-many Relationship 9](#_Toc27424043)

[Many-to-many Relationship 9](#_Toc27424044)

[One-to-zero/one Relationship 9](#_Toc27424045)

[One-to-one Relationship 9](#_Toc27424046)

[Cascade Delete 9](#_Toc27424047)

[Loading 9](#_Toc27424048)

[Lazy Loading 9](#_Toc27424049)

[Eager Loading 9](#_Toc27424050)

[Explicit Loading 9](#_Toc27424051)

# Steps

## Create new project

Install NuGet EntityFramework

## Model

Create Model (in Program.cs)

public class Course

{

public int Id { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public CourseLevel Level { get; set; }

public float FullPrice { get; set; }

public Author Author { get; set; }

public IList<Tag> Tags { get; set; }

}

public class Author

{

public int Id { get; set; }

public string Name { get; set; }

public IList<Course> Courses { get; set; }

}

public class Tag

{

public int Id { get; set; }

public string Name { get; set; }

public IList<Course> Courses { get; set; }

}

## Connection String

Add ConnectionString to App.Config

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=RonnieDev;Initial Catalog=Pluto2;Persist Security Info=True;User ID=sa;Password=qweQWE123!@#" providerName="System.Data.SqlClient"/>

</connectionStrings>

## DBContext

Create DBContext (in Program.cs) add the name of the ConnectionString in the App.Config

public class PlutoContext : DbContext

{

public DbSet<Course> Courses { get; set; }

public DbSet<Author> Authors { get; set; }

public DbSet<Tag> Tags { get; set; }

public DbSet<Category> Categories { get; set; }

public PlutoContext()

: base("name=DefaultConnection")

{

}

}

## Enable Migrations

Execute command in Package Manager Console (only once per project)

enable-migrations

## Add Migration

add-migration MigrationName -Force

## Update Migration

update-database

# Package Manager Console

## Enable Migrations

Execute command in Package Manager Console (only once per project)

enable-migrations

## Add Migration

add-migration MigrationName -Force

## Update Migration

update-database

# Data Annotation

<https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/data-annotations#notmapped>

## Key

Primary key, defaults to id or ClassName+ID

    [Key]

    public int PrimaryTrackingKey { get; set; }

## Composite keys

[Key]

[Column(Order=1)]

public int PassportNumber { get; set; }

[Key]

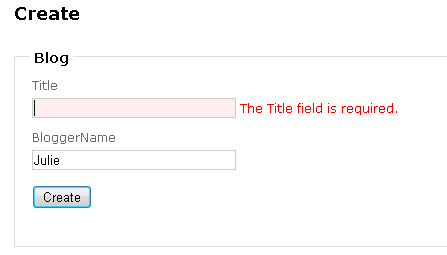
[Column(Order = 2)]

public string IssuingCountry { get; set; }

## Required

[Required, ErrorMessage = ”Error message”]

public string Title { get; set; }



## MaxLength and MinLength

[MaxLength(10),MinLength(5)]

public string BloggerName { get; set; }

MaxLength will be used for the database nVarChar(10)

[MaxLength(10, ErrorMessage="BloggerName must be 10 characters or less"),MinLength(5)]

public string BloggerName { get; set; }



## NotMapped

[NotMapped]

public string BlogCode

{

get

{

return Title.Substring(0, 1) + ":" + BloggerName.Substring(0, 1);

}

}

## ComplexType

1 to1 relationship between two tables / classes

[ComplexType]

public class BlogDetails

{

public DateTime? DateCreated { get; set; }

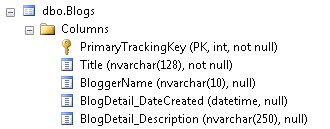
    [MaxLength(250)]

    public string Description { get; set; }

}

If the Blog Class add the following property

    public BlogDetails BlogDetail { get; set; }



## ConcurrencyCheck

[ConcurrencyCheck, MaxLength(10, ErrorMessage="BloggerName must be 10 characters or less"),MinLength(5)]

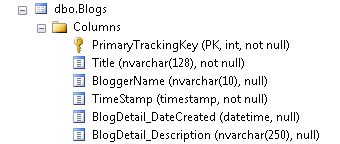
public string BloggerName { get; set; }

## TimeStamp

non-nullable timestamp

[Timestamp]

public Byte[] TimeStamp { get; set; }



## Table

[Table("InternalBlogs")]

public class Blog

## Column

[Column("BlogDescription", TypeName="ntext")]

public String Description {get;set;}

## Index

[Index]

public int Rating { get; set; }

## Unique Index

[Index(IsUnique = true)]

[StringLength(200)]

public string Username { get; set; }

## Multiple-Column Indexes

[Index("IX\_BlogIdAndRating", 2)]

public int Rating { get; set; }

[Index("IX\_BlogIdAndRating", 1)]

public int BlogId { get; set; }

## Relationships

..

..

..

# Fluent API

In DBContext derived class

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Configurations.Add(new VideoConfiguration());

base.OnModelCreating(modelBuilder);

}

public class VideoConfiguration : EntityTypeConfiguration<Video>

{

public VideoConfiguration()

{

Property(v => v.Name)

.IsRequired()

.HasMaxLength(255);

HasRequired(v => v.Genre)

.WithMany(g => g.Videos)

.HasForeignKey(v => v.GenreId);

HasMany(v => v.Tags)

.WithMany(t => t.Videos)

.Map(m =>

{

m.ToTable("VideoTags");

m.MapLeftKey("VideoId");

m.MapRightKey("TagId");

});

}

}

## Default Schema

modelBuilder.HasDefaultSchema("sales");

## Table Name

Entity<Course>.ToTable(“tbl\_Course”, “catalog”);

## Primary Key

modelBuilder.Entity<OfficeAssignment>().HasKey(t => t.InstructorID);

## Primary Key without auto number

modelBuilder.Entity<Department>().Property(t => t.DepartmentID)

.HasDatabaseGeneratedOption(DatabaseGeneratedOption.None);

## Composite Primary Key

modelBuilder.Entity<Department>().HasKey(t => new { t.DepartmentID, t.Name });

## Column

Entity<Course>.Property(t => t.Name)

.HasColumnName(“sName”)

.HasColumnType(“varchar”)

.HasColumnOrder(2)

.IsRequired()

.HasMaxLength(255)

.IsUnicode(false); // varChar vs nVarChar

## Index

modelBuilder

.Entity<Department>()

.Property(t => t.Name)

.HasColumnAnnotation("Index", new IndexAnnotation(new IndexAttribute()));

## Multiple Column Index

modelBuilder

.Entity<Department>()

.Property(t => t.Name)

.HasColumnAnnotation(

"Index",

new IndexAnnotation(new[]

{

new IndexAttribute("Index1"),

new IndexAttribute("Index2") { IsUnique = true }

})));

## Not Map

modelBuilder.Entity<Department>().Ignore(t => t.Budget);

## Complex Type

modelBuilder.ComplexType<Details>()

.Property(t => t.Location)

.HasMaxLength(20);

## One-to-many Relationship

Entity<Author>.HasMany(a => a.Courses)

.WithRequired(c => c.Author)

.HasForeignKey(c => c.AuthorId);

## Many-to-many Relationship

Entity<Course>.HasMany(c => c.Tags)

.WithMany(t => t.Courses)

.Map(m => {

m.ToTable(“CourseTag”);

m.MapLeftKey(“CourseId”);

m.MapRightKey(“TagId”);

});

## One-to-zero/one Relationship

Entity<Course>.HasOptional(c => c.Caption)

.WithRequired(c => c.Course);

## One-to-one Relationship

Entity<Course>.HasRequired(c => c.Cover)

.WithRequiredPrincipal(c => c.Course);

Entity<Cover>.HasRequired(c => c.Course)

.WithRequiredDependent(c => c.Cover);

## Cascade Delete

modelBuilder.Entity<Course>()

.HasRequired(t => t.Department)

.WithMany(t => t.Courses)

.HasForeignKey(d => d.DepartmentID)

.WillCascadeOnDelete(false);

# Loading

## Lazy Loading

Declare the property as virtual

public virtual Author Author { get; set; }

Or add in DBBContext ctor

this.Configuration.LazyLoadingEnabled = false;

## Eager Loading

add Include cause the SQL to have a join with the relatioship

var users = dlContext.Users.Include(x => x.DepartmentID).ToList();

## Explicit Loading

Explicit loading of joined data

First load the main object

var user = dlContext.Users.FirstOrDefault(x => x.ID == 1);

Then load the related data

dlContext.Departments.Where(x => x.ID == user.DepartmentID).Load();