免費電子書

學習

Entity Framework

Free unaffiliated eBook created from
Stack Overflow contributors.

#entity-framework

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: entity-framework

It is an unofficial and free Entity Framework ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Entity Framework.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1:

EFORM.NET。。

EF Designer。。

Microsoft.NET FrameworkMicrosoftORM。

1.0	2008-08-11
4	2010-04-12
4.1	2011-04-12
4.11	2011-07-25
4.3.1	2012-02-29
5	2012811
6	○
6.1	2014317
1.0	2016627

https //msdn.microsoft.com/en-ca/data/jj574253.aspx

Examples

C

GUI.edmx。*Code Entity*。。

```
public class Planet
{
    public string Name { get; set; }
    public decimal AverageDistanceFromSun { get; set; }
}
```

。 DbSet<>

```
using System.Data.Entity;

public class PlanetContext : DbContext
{
    public DbSet<Planet> Planets { get; set; }
```

```
}
```

```
using(var context = new PlanetContext())
{
    var jupiter = new Planet
    {
        Name = "Jupiter",
        AverageDistanceFromSun = 778.5
    };

    context.Planets.Add(jupiter);
    context.SaveChanges();
}
```

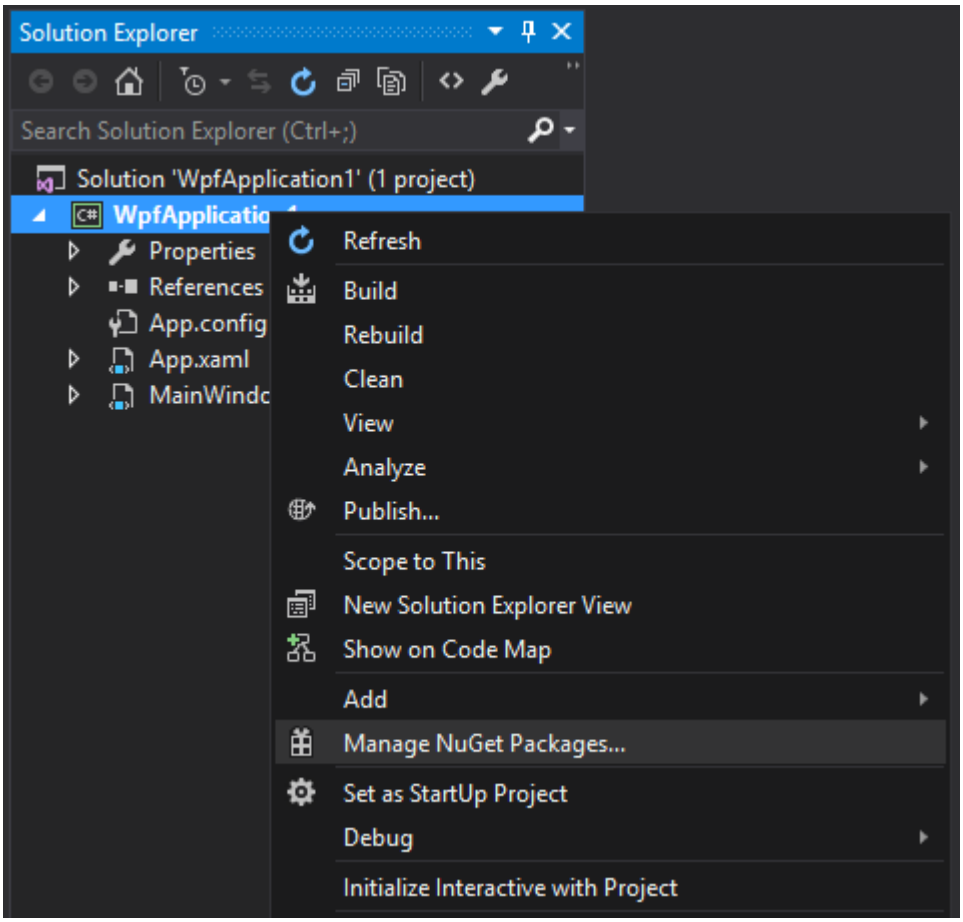
NamePlanet "Jupiter" AverageDistanceFromSun778.5

DbSetAdd()PlanetSaveChanges()◦

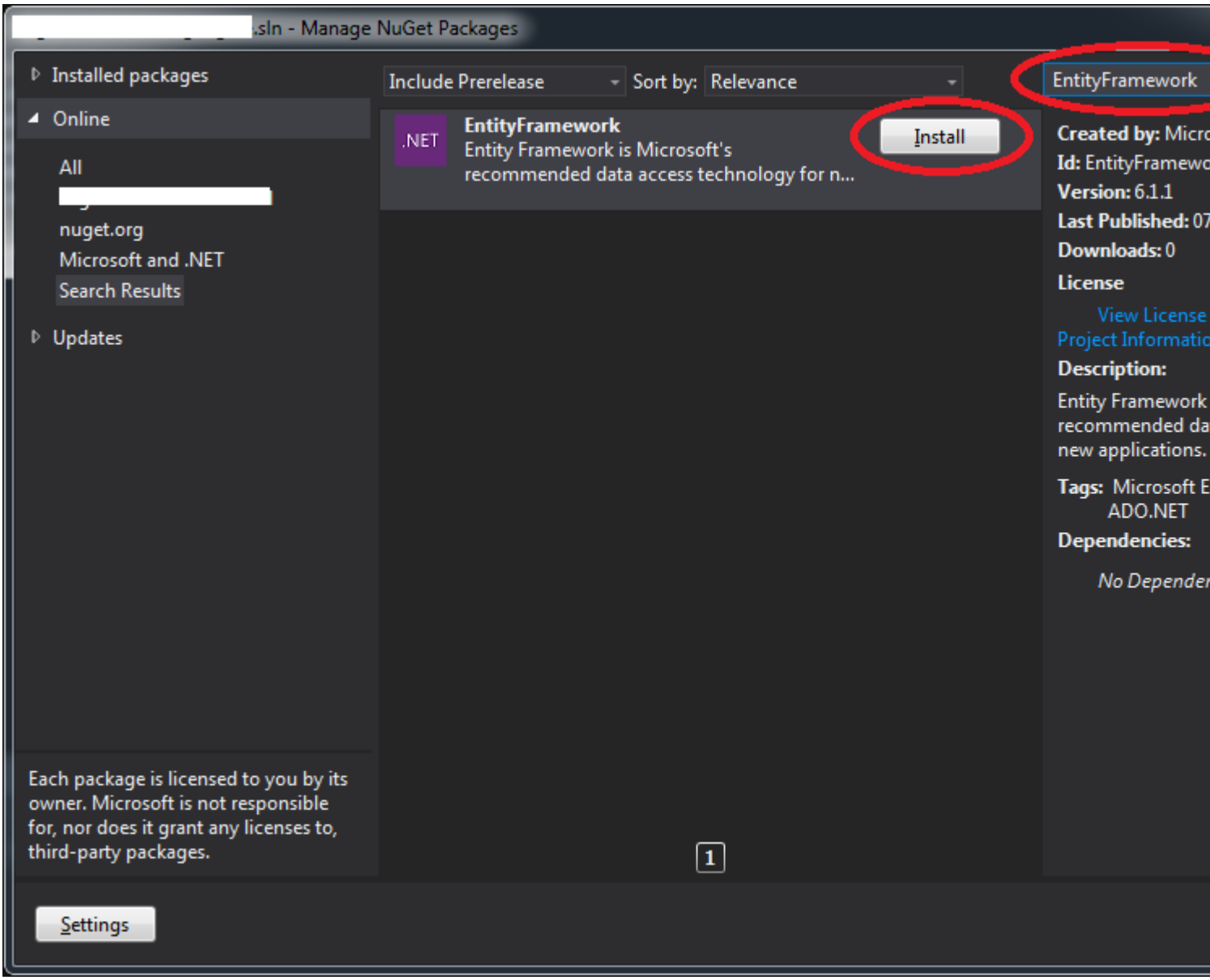
```
using(var context = new PlanetContext())
{
    var jupiter = context.Planets.Single(p => p.Name == "Jupiter");
    Console.WriteLine($"Jupiter is {jupiter.AverageDistanceFromSun} million km from the
sun.");
}
```

NuGet

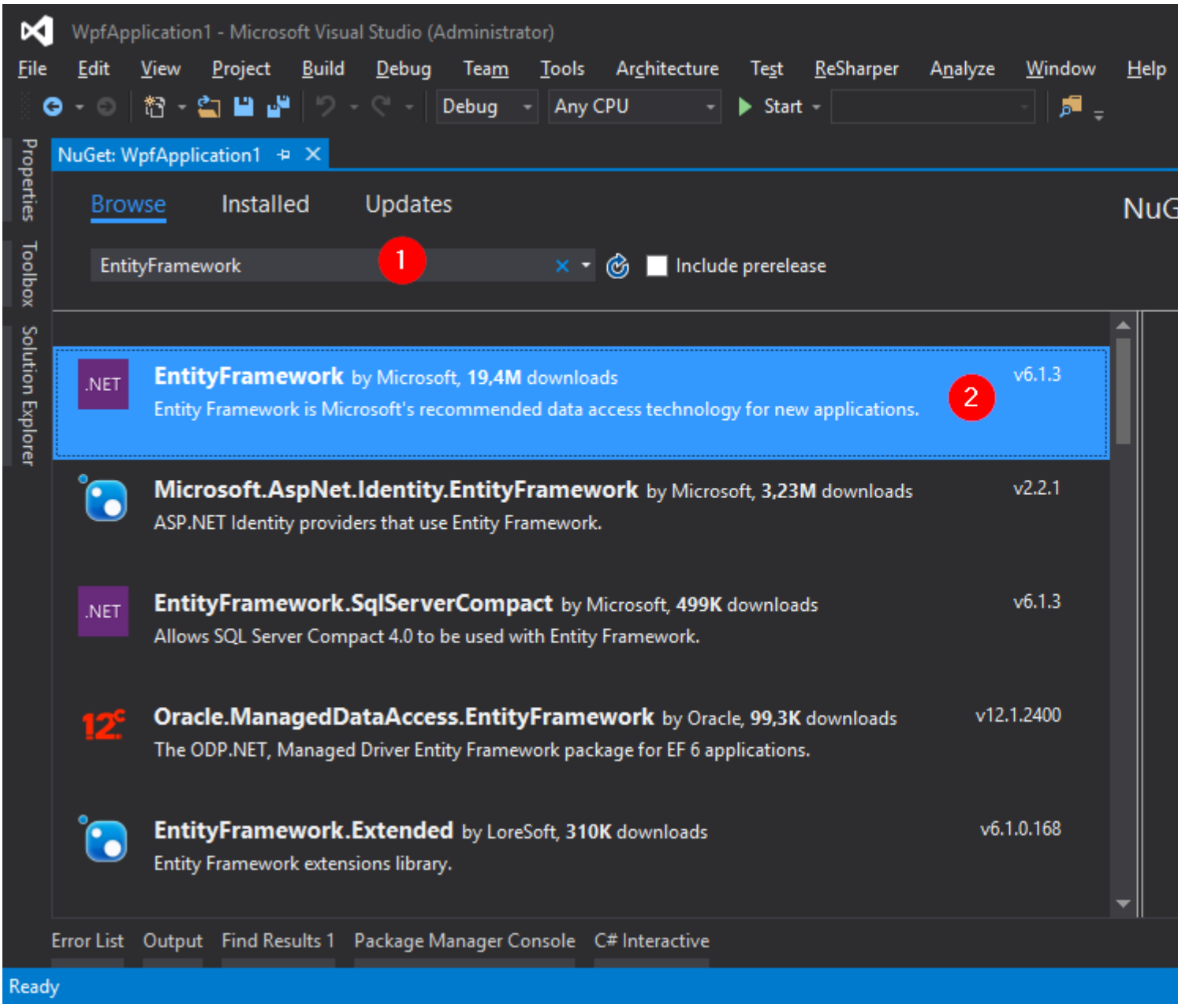
Visual Studio**Solution Explorer**□□□□*Manage NuGet Packages*



EntityFramework ◦



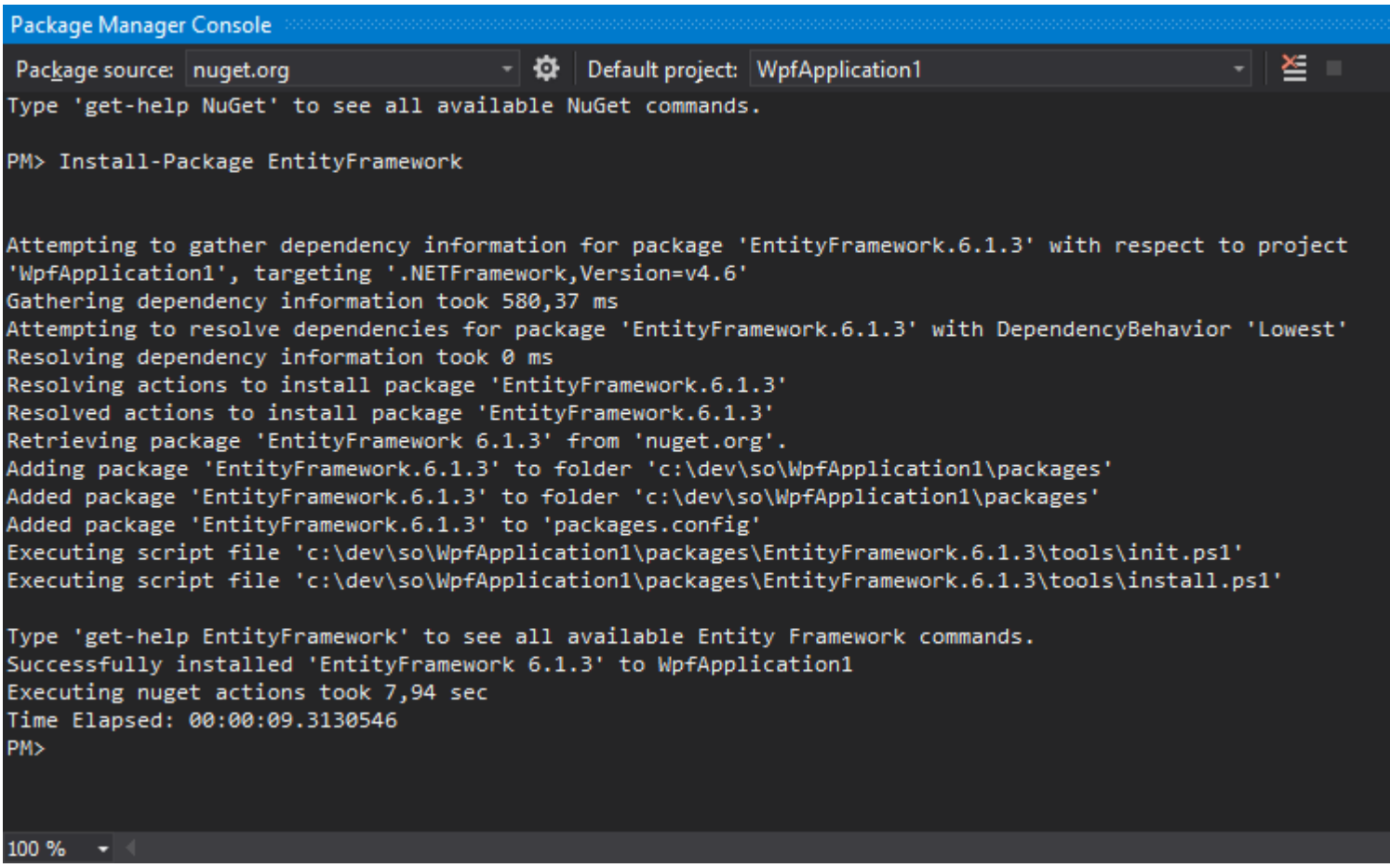
Visual Studio 2015



Install。

。 - > *NuGet* - >

```
Install-Package EntityFramework
```



Entity Framework。

ADO.Net。 Microsoft“”**O / RM**。

/O / RM。 ADO.NET。

O / RM

ORMMS SQL Server。 O / RM

1.
2.
3.

ORM。 。 CRUD。

https://riptutorial.com/zh-TW/entity-framework/topic/815/

2: Code First DataAnnotations

DataAnnotation。 DataAnnotationCode-First。

1. **System.ComponentModel.DataAnnotations**。
2. **System.ComponentModel.DataAnnotations.Schema**。

DataAnnotations。 Fluent APICode-First。

Examples

[Key]

Key/。

Code-First 。 。

```
using System.ComponentModel.DataAnnotations;

public class Person
{
    [Key]
    public int PersonKey { get; set; }        // <- will be used as primary key

    public string PersonName { get; set; }
}
```

[Key]。 [**KeyColumnOrder = x]**。

```
using System.ComponentModel.DataAnnotations;

public class Person
{
    [Key, Column(Order = 0)]
    public int PersonKey1 { get; set; }    // <- will be used as part of the primary key

    [Key, Column(Order = 1)]
    public int PersonKey2 { get; set; }    // <- will be used as part of the primary key

    public string PersonName { get; set; }
}
```

[Key] EntityFramework“Id”“{ClassName} Id”。

```
public class Person
{
    public int PersonID { get; set; }        // <- will be used as primary key

    public string PersonName { get; set; }
}
```

[]

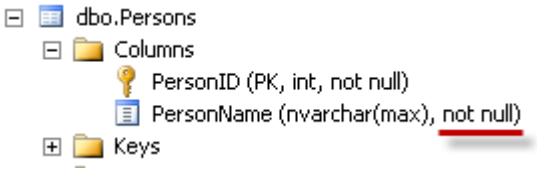
NOT NULL。

```
using System.ComponentModel.DataAnnotations;

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

    [Required]
    public string PersonName { get; set; }
}
```

NOT NULL



asp.net-mvc。

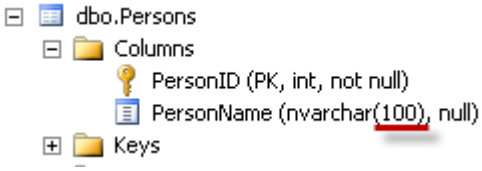
[MaxLength][MinLength]

[MaxLengthhint]。 。

```
using System.ComponentModel.DataAnnotations;

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

    [MinLength(3), MaxLength(100)]
    public string PersonName { get; set; }
}
```



[MinLengthhint]。 /PersonName3Person。 `DbUpdateConcurrencyException` 。

```
using (var db = new ApplicationDbContext())
{
     db.Staff.Add(new Person() { PersonName = "ng" });
     try
     {
          db.SaveChanges();
     }
     catch (DbEntityValidationException ex)
     {
```

```
          //ErrorMessage = "The field PersonName must be a string or array type with a minimum
length of '3'."
     }
}
```

[MaxLength][MinLength]asp.net-mvc。

[]

```
using System.ComponentModel.DataAnnotations;

public partial class Enrollment
{
    public int EnrollmentID { get; set; }

    [Range(0, 4)]
    public Nullable<decimal> Grade { get; set; }
}
```

/。 DbUpdateConcurrencyException 。

```
using (var db = new ApplicationDbContext())
{
    db.Enrollments.Add(new Enrollment() { Grade = 1000 });

    try
    {
        db.SaveChanges();
    }
    catch (DbEntityValidationException ex)
    {
        // Validation failed for one or more entities
    }
}
```

asp.net-mvc。

[DatabaseGenerated]

。

1. None。
2. Identity 。
3. Computed。

None Entity Framework。

StoreGeneratedIdentityKeyConvention 。 NoneDatabaseGenerated。

```
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
}
```

SQL

```
CREATE TABLE [Person] (
    Name varchar(100) PRIMARY KEY,
    DateOfBirth Date NOT NULL,
    Age AS DATEDIFF(year, DateOfBirth, GETDATE())
)
GO
```

ComputedDatabaseGenerated。

```
[Table("Person")]
public class Person
{
    [Key, StringLength(100)]
    public string Name { get; set; }
    public DateTime DateOfBirth { get; set; }
    [DatabaseGenerated(DatabaseGeneratedOption.Computed)]
    public int Age { get; set; }
}
```

[NotMapped]

Code-Firstgettersetter。 **[NotMapped]**。

。 。

```
public string FullName => string.Format("{0} {1}", FirstName, LastName);
```

“”gettersetter。

“AverageGrade”。AverageGrade;。

```
[NotMapped]
public float AverageGrade { set; get; }
```

“AverageGrade”**[NotMapped]**Entity Framework。

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

public class Student
{
    public int Id { set; get; }

    public string FirstName { set; get; }

    public string LastName { set; get; }

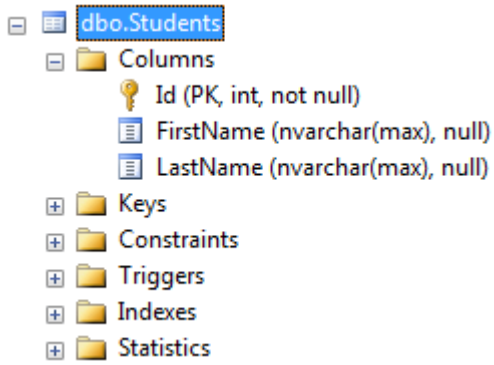
    public string FullName => string.Format("{0} {1}", FirstName, LastName);

    [NotMapped]
    public float AverageGrade { set; get; }
}
```

`DbMigration.cs`

```
CreateTable(
    "dbo.Students",
     c => new
         {
             Id = c.Int(nullable: false, identity: true),
             FirstName = c.String(),
             LastName = c.String(),
         })
     .PrimaryKey(t => t.Id);
```

SQL Server Management Studio



[]

```
[Table("People")]
public class Person
{
    public int PersonID { get; set; }
    public string PersonName { get; set; }
}
```

Entity Framework`PersonPersons`

[Table]

```
[Table("People", Schema = "domain")]
```

[Column]

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

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

Entity Framework。

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

[Index]

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

    [Index]
    public int Age { get; set; }
}
```

。

```
[Index("IX_Person_Age")]
public int Age { get; set; }
```

。

```
[Index(IsUnique = true)]
public int Age { get; set; }
```

。

```
[Index("IX_Person_NameAndAge", 1)]
public int Age { get; set; }

[Index("IX_Person_NameAndAge", 2)]
public string PersonName { get; set; }
```

2。。

IndexEntity Framework 6.1。。

[ForeignKeystring]

EF。

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

    [ForeignKey(nameof(IdAddress))]
    public virtual Address HomeAddress { get; set; }
}
```

。

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

public class Customer
{
    ...

    public int MailingAddressID { get; set; }
    public int BillingAddressID { get; set; }

    [ForeignKey("MailingAddressID")]
    public virtual Address MailingAddress { get; set; }
    [ForeignKey("BillingAddressID")]
    public virtual Address BillingAddress { get; set; }
}
```

`ForeignKey`EF`MailingAddressBillingAddressIDAddress_MailingAddress_Id`。

[StringLengthint]

```
using System.ComponentModel.DataAnnotations;

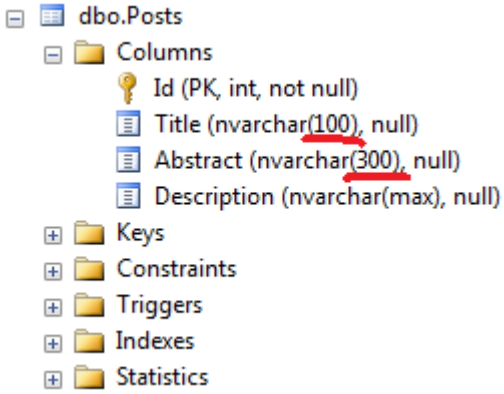
public class Post
{
    public int Id { get; set; }

    [StringLength(100)]
    public string Title { get; set;}

    [StringLength(300)]
    public string Abstract { get; set; }

    public string Description { get; set; }
}
```

。



asp.net-mvc。

[]

[TimeStamp]Entity。。TimeStamp。

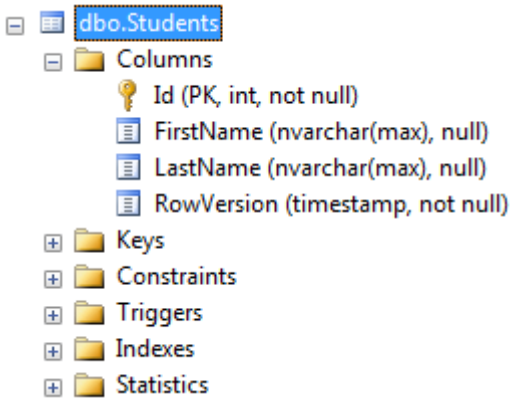
```
using System.ComponentModel.DataAnnotations.Schema;

public class Student
{
    public int Id { set; get; }

    public string FirstName { set; get; }

    public string LastName { set; get; }

    [Timestamp]
    public byte[] RowVersion { get; set; }
}
```



[ConcurrencyCheck]

。ConcurrencyCheck。

```
using System.ComponentModel.DataAnnotations;

public class Author
{
    public int AuthorId { get; set; }
```

```
    [ConcurrencyCheck]
    public string AuthorName { get; set; }
}
```

ConcurrencyCheckAuthorAuthorName。Code-FirstupdatewhereAuthorName。

[InversePropertystring]

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

public class Department
{
    ...

    public virtual ICollection<Employee> PrimaryEmployees { get; set; }
    public virtual ICollection<Employee> SecondaryEmployees { get; set; }
}

public class Employee
{
    ...

    [InverseProperty("PrimaryEmployees")]
    public virtual Department PrimaryDepartment { get; set; }

    [InverseProperty("SecondaryEmployees")]
    public virtual Department SecondaryDepartment { get; set; }
}
```

InverseProperty 。

。

。

。

。

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

public class TreeNode
{
    [Key]
    public int ID { get; set; }
    public int ParentID { get; set; }

    ...

    [ForeignKey("ParentID")]
    public TreeNode ParentNode { get; set; }
    [InverseProperty("ParentNode")]
    public virtual ICollection<TreeNode> ChildNodes { get; set; }
}
```

ForeignKey。 EmployeeForeignKey。

[ComplexType]

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

[ComplexType]
public class BlogDetails
{
    public DateTime? DateCreated { get; set; }

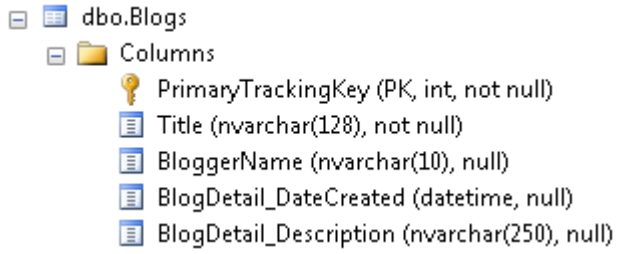
    [MaxLength(250)]
    public string Description { get; set; }
}

public class Blog
{
    ...

    public BlogDetails BlogDetail { get; set; }
}
```

Entity Framework。

- BlogDetails。



。

Code First DataAnnotations https://riptutorial.com/zh-TW/entity-framework/topic/4161/code-first-dataannotations

3: EF

Examples

AsNoTracking

```
var location =  dbContext.Location
                     .Where(l => l.Location.ID == location_ID)
                     .SingleOrDefault();

return location;
```

。

```
var location =  dbContext.Location.AsNoTracking()
                     .Where(l => l.Location.ID == location_ID)
                     .SingleOrDefault();

return location;
```

AsNoTracking()。 。 SaveChanges。

。 。

“location”10。 'LocationName'。

```
var location =  dbContext.Location.AsNoTracking()
                         .Where(l => l.Location.ID == location_ID)
                         .SingleOrDefault();

return location.Name;
```

```
var location =  dbContext.Location
                          .Where(l => l.Location.ID == location_ID)
                          .Select(l => l.LocationName);
                          .SingleOrDefault();

return location;
```

“”“LocationName”。

AsNoTracking() 。 。

```
var location = dbContext.Location
                     .Where(l => l.Location.ID == location_ID)
                     .Select(l => new { Name = l.LocationName, Area = l.LocationArea })
                     .SingleOrDefault();

return location.Name + " has an area of " + location.Area;
```

“LocationName”“LocationArea”“”。

。

```
var counties = dbContext.States.Single(s => s.Code == "tx").Counties.Count();
```

。 States.Single(…)。 Counties254。 Counties.Count() 。

```
var counties = dbContext.Counties.Count(c => c.State.Code == "tx");
```

SQL。 - 。

IQueryable<T>IEnumerable<T> 。

。 102010。

```
IEnumerable<TResult1> result1;
IEnumerable<TResult2> result2;

using(var context = new Context())
{
    result1 = await context.Set<TResult1>().ToListAsync().ConfigureAwait(false);
    result2 = await context.Set<TResult1>().ToListAsync().ConfigureAwait(false);
}
```

```
public async Task<IEnumerable<TResult>> GetResult<TResult>()
{
    using(var context = new Context())
    {
        return await context.Set<TResult1>().ToListAsync().ConfigureAwait(false);
    }
}



IEnumerable<TResult1> result1;
IEnumerable<TResult2> result2;

var result1Task = GetResult<TResult1>();
var result2Task = GetResult<TResult2>();

await Task.WhenAll(result1Task, result2Task).ConfigureAwait(false);

var result1 = result1Task.Result;
var result2 = result2Task.Result;
```

。 。

```
using(var context = new Context())
{
```

```
    return await context.Set<MyEntity>().ToListAsync().ConfigureAwait(false);
}
```

```
using(var context = new Context())
{
    context.Configuration.AutoDetectChangesEnabled = false;
    context.Configuration.ProxyCreationEnabled = false;

    return await context.Set<MyEntity>().ToListAsync().ConfigureAwait(false);
}
```

```
public class MyContext : DbContext
{
    public MyContext()
        : base("MyContext")
    {
        Configuration.AutoDetectChangesEnabled = false;
        Configuration.ProxyCreationEnabled = false;
    }

    //snip
}
```

`ProductCategory` S

```
public class Product
{
    public Product()
    {
        Categories = new HashSet<Category>();
    }
    public int ProductId { get; set; }
    public string ProductName { get; set; }
    public virtual ICollection<Category> Categories { get; private set; }
}

public class Category
{
    public Category()
    {
        Products = new HashSet<Product>();
    }
    public int CategoryId { get; set; }
    public string CategoryName { get; set; }
    public virtual ICollection<Product> Products { get; set; }
}
```

`ProductCategory` `Categories`

```
var product = db.Products.Find(1);
var category = db.Categories.Find(2);
product.Categories.Add(category);
db.SaveChanges();
```

`db` `DbContext`。

ProductCategory◦ Id◦ ◦

Id◦

```
// Create two stub entities
var product = new Product { ProductId = 1 };
var category = new Category { CategoryId = 2 };

// Attach the stub entities to the context
db.Entry(product).State = System.Data.Entity.EntityState.Unchanged;
db.Entry(category).State = System.Data.Entity.EntityState.Unchanged;

product.Categories.Add(category);
db.SaveChanges();
```

◦

◦

```
var exists = db.Categories.Any(c => c.Id == 1 && c.Products.Any(p => p.Id == 14));
```

◦ ◦

EF https://riptutorial.com/zh-TW/entity-framework/topic/2714/ef

4: Postgresql

Examples

NpgsqlddexproviderEntity Framework 6.1.3PostgresSql

1C\ Windows \ Microsoft.NET \ Framework \ v4.0.30319 \ ConfigC\ Windows \ Microsoft.NET \ Framework64 \ v4.0.30319 \ ConfigMachine.config

2

a`<system.data> <DbProviderFactories>`

```
            <add name="Npgsql Data Provider" invariant="Npgsql" support="FF"
            description=".Net Framework Data Provider for Postgresql Server"
            type="Npgsql.NpgsqlFactory, Npgsql, Version=2.2.5.0, Culture=neutral,
PublicKeyToken=5d8b90d52f46fda7" />
```

bverison。

3. 。
4. VS2013。
5. Npgsql“gacutil -u Npgsql”“gacutil -i [dll]”Npgsql 2.5.0
6. Mono.Security 4.0.0.0
7. NpgsqlDdexProvider-2.2.0-VS2013.zipNpgsqlDdexProvider.vsixVisual Studio
8. EFTools6.1.3-beta1ForVS2013.msi。
9. Manage Nuget PackagesEntityFramework6.1.3NpgSql2.5.0NpgSql.EntityFramework2.5.0。 .....MVc

Postgresql https://riptutorial.com/zh-TW/entity-framework/topic/7647/postgresql

5:

Examples

Database.BeginTransaction

。

```
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();
        }
        catch (Exception ex)
        {
            //There is no need to call transaction.Rollback() here as the transaction object
            //will go out of scope and disposing will roll back automatically
        }
    }
}
```

`transaction.Rollback()`。 Dipsose `transaction.Rollback()`。

https://riptutorial.com/zh-TW/entity-framework/topic/4944/

6: - API

HOW Entity FrameworkPOCO **Data AnnotationsFluent API** 。

“”。 Fluent API。

Examples

EntityFramewok Fluent API。 。 *Fluent APIOnModelCreatingEntityTypeConfiguration OnModelCreatingmodelBuilder* 。 。

。

```
public class Employee
{
    public int Id { get; set; }
    public string Surname { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public short Age { get; set; }
    public decimal MonthlySalary { get; set; }

    public string FullName
    {
        get
        {
            return $"{Surname} {FirstName} {LastName}";
        }
    }
}
```

mapper

```
public class EmployeeMap
    : EntityTypeConfiguration<Employee>
{
    public EmployeeMap()
    {
        // Primary key
        this.HasKey(m => m.Id);

        this.Property(m => m.Id)
            .HasColumnType("int")
            .HasDatabaseGeneratedOption(DatabaseGeneratedOption.Identity);

        // Properties
        this.Property(m => m.Surname)
            .HasMaxLength(50);

        this.Property(m => m.FirstName)
            .IsRequired()
```

```
            .HasMaxLength(50);

        this.Property(m => m.LastName)
            .HasMaxLength(50);

        this.Property(m => m.Age)
            .HasColumnType("smallint");

        this.Property(m => m.MonthlySalary)
            .HasColumnType("number")
            .HasPrecision(14, 5);

        this.Ignore(m => m.FullName);

        // Table & column mappings
        this.ToTable("TABLE_NAME", "SCHEMA_NAME");
        this.Property(m => m.Id).HasColumnName("ID");
        this.Property(m => m.Surname).HasColumnName("SURNAME");
        this.Property(m => m.FirstName).HasColumnName("FIRST_NAME");
        this.Property(m => m.LastName).HasColumnName("LAST_NAME");
        this.Property(m => m.Age).HasColumnName("AGE");
        this.Property(m => m.MonthlySalary).HasColumnName("MONTHLY_SALARY");
    }
}
```

- **HasKey** - 。 。 *this.HasKeym => new {m.DepartmentIdm.PositionId}* 。
- - 。
- **HasColumnType** - 。 OracleMS SQL。
- **HasDatabaseGeneratedOption** - 。 PK*DatabaseGeneratedOption.Identity* *DatabaseGeneratedOption.None* 。
- **HasMaxLength** - 。
- **IsRequired** - 。
- **HasPrecision** - 。
- - 。 FullName。
- **ToTable** - 。
- **HasColumnName** - 。 。

。

EntityFrameworkmapper。 *OnModelCreatingmodelBuilder.Configurations*

```
public class DbContext()
    : base("Name=DbContext")
{
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Configurations.Add(new EmployeeMap());
    }
}
```

。 。

.HasKey。

```
using System.Data.Entity;
// ..

public class PersonContext : DbContext
{
    // ..

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // ..

        modelBuilder.Entity<Person>().HasKey(p => p.PersonKey);
    }
}
```

.HasKey。

```
using System.Data.Entity;
// ..

public class PersonContext : DbContext
{
    // ..

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // ..

        modelBuilder.Entity<Person>().HasKey(p => new { p.FirstName, p.LastName });
    }
}
```

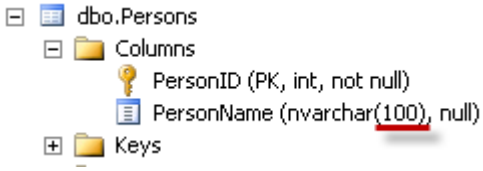
.HasMaxLength。

```
using System.Data.Entity;
// ..

public class PersonContext : DbContext
{
    // ..

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // ..

        modelBuilder.Entity<Person>()
                    .Property(t => t.Name)
                    .HasMaxLength(100);
    }
}
```



NOT NULL

.IsRequiredNOT NULL。

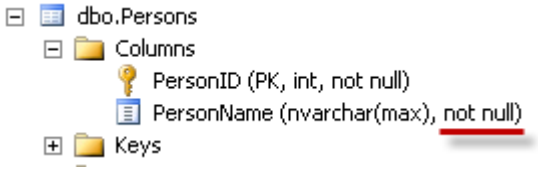
```
using System.Data.Entity;
// ..

public class PersonContext : DbContext
{
    // ..

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // ..

        modelBuilder.Entity<Person>()
                    .Property(t => t.Name)
                    .IsRequired();
    }
}
```

NOT NULL



Entity Framework。Fluent API。`Map`。

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

public class Employee
{
    property int Id { get; set; }
    property Company Employer { get; set; }
}

public class EmployeeContext : DbContext
{
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Employee>()
                    .HasRequired(x => x.Employer)
                    .WithRequiredDependent()
                    .Map(m => m.MapKey("EmployerId"));
    }
```

```
}
```

`MapMapKey`。 Employer_IdEmployerId。

- API https://riptutorial.com/zh-TW/entity-framework/topic/4530/---api

7:

Code-First。 Code-First*System.Data.Entity.ModelConfiguration.Conventions* EF 5EF 6 。

Examples

“ID”“ID”。 GUID。

```
public class Room
{
    // Primary key
    public int RoomId{ get; set; }
    ...
}
```

`OnModelCreating`System.Data.Entity.ModelConfiguration.Conventions。

PluralizingTableNameConvention。

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

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

EFDB's'。 Code First`dbo.Productsdbo.Productdbo.Products`。

Code First

1. DbSet。
2. 。
3. DbSet

`CompanyDbSet<Company>`

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

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

```
    public virtual ICollection<Person> Staff { set; get; }
}

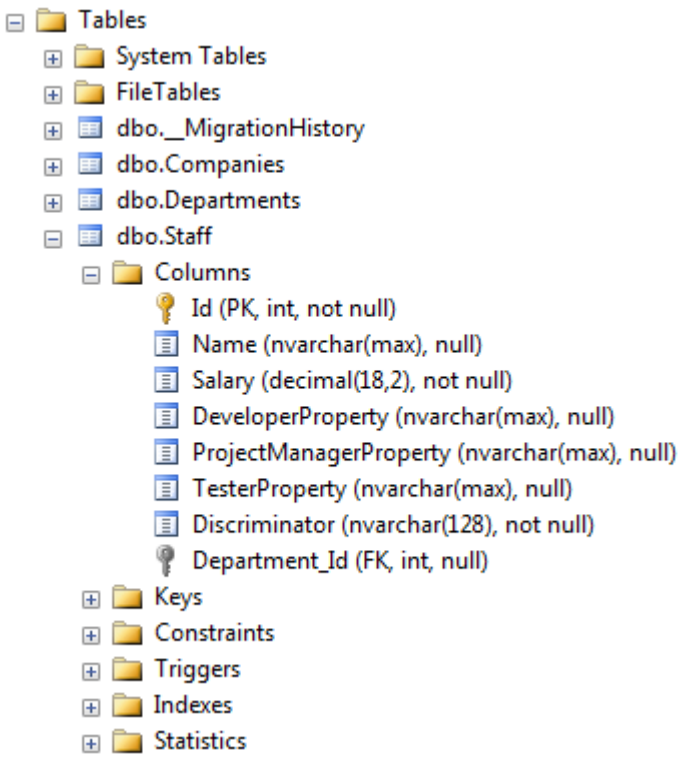
[Table("Staff")]
public class Person
{
    public int Id { set; get; }
    public string Name { set; get; }
    public decimal Salary { set; get; }
}

public class ProjectManager : Person
{
   public string ProjectManagerProperty { set; get; }
}

public class Developer : Person
{
    public string DeveloperProperty { set; get; }
}

public class Tester : Person
{
    public string TesterProperty { set; get; }
}

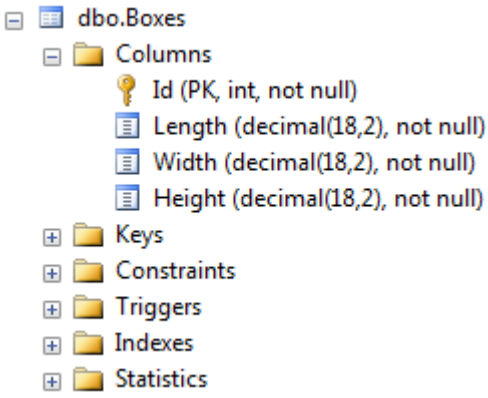
public class ApplicationDbContext : DbContext
{
    public DbSet<Company> Companies { set; get; }
}
```



DecimalPropertyConvention

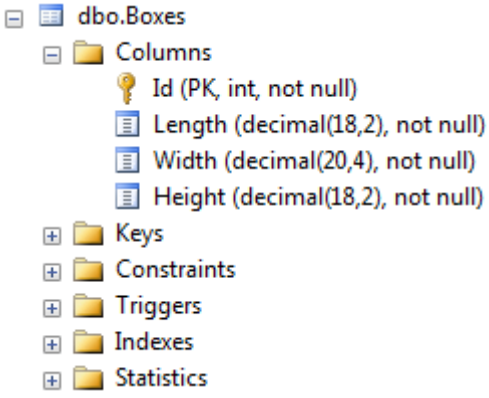
Entity Framework18,2。

```
public class Box
{
    public int Id { set; get; }
    public decimal Length { set; get; }
    public decimal Width { set; get; }
    public decimal Height { set; get; }
}
```



1.Fluent API

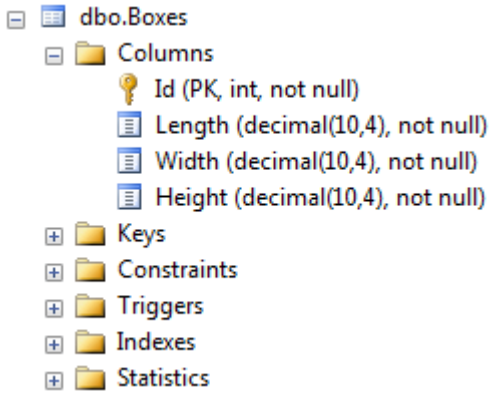
```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Entity<Box>().Property(b => b.Width).HasPrecision(20, 4);
}
```



“”20,4。

2.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Conventions.Remove<DecimalPropertyConvention>();
    modelBuilder.Conventions.Add(new DecimalPropertyConvention(10, 4));
}
```



10,4。

Code Firstnavigation。。StudentStandardICollection。Code FirstStudentsStandard_StandardId StandardsStudents DB。

```
public class Student
{

    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public DateTime DateOfBirth { get; set; }

    //Navigation property
    public Standard Standard { get; set; }
}

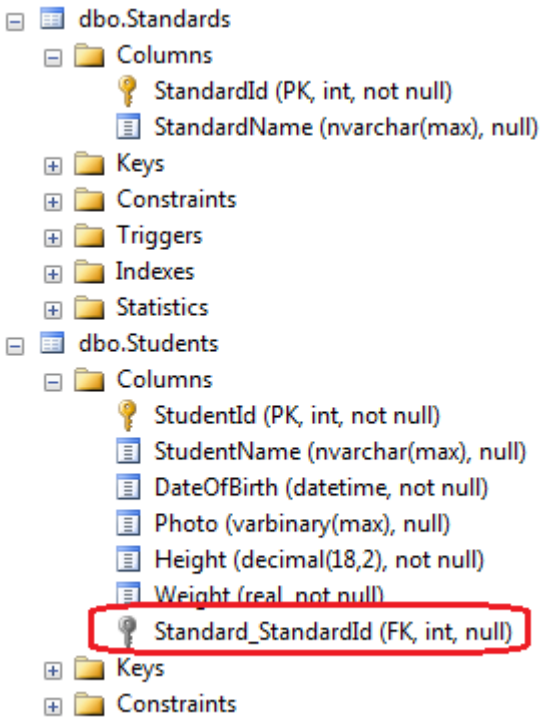
public class Standard
{

    public int StandardId { get; set; }
    public string StandardName { get; set; }

    //Collection navigation property
    public IList<Student> Students { get; set; }

}
```

Standard_StandardId。



ABBAEF。

```
public class Department
{
    public int DepartmentId { set; get; }
    public string Name { set; get; }
    public virtual ICollection<Person> Staff { set; get; }
}

public class Person
{
    public int Id { set; get; }
    public string Name { set; get; }
    public decimal Salary { set; get; }
    public int DepartmentId { set; get; }
    public virtual Department Department { set; get; }
}
```

DepartmentId。

https://riptutorial.com/zh-TW/entity-framework/topic/2447/

8: EntityFramework

Examples

。

```
public abstract class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime BirthDate { get; set; }
}

public class Employee : Person
{
    public DateTime AdmissionDate { get; set; }
    public string JobDescription { get; set; }
}

public class Customer : Person
{
    public DateTime LastPurchaseDate { get; set; }
    public int TotalVisits { get; set; }
}

// On DbContext
public DbSet<Person> People { get; set; }
public DbSet<Employee> Employees { get; set; }
public DbSet<Customer> Customers { get; set; }
```

IdBirthDate Discrimitator AdmissionDate JobDescription LastPurchaseDate TotalVisits

'Discriminator''AdmissionDate''JobDescription''LastPurchaseDate''TotalVisits'。

- 。
-
-

-
-

n + 1n。

```
public abstract class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime BirthDate { get; set; }
}

[Table("Employees")]
public class Employee : Person
{
    public DateTime AdmissionDate { get; set; }
```

```
    public string JobDescription { get; set; }
}

[Table("Customers")]
public class Customer : Person
{
    public DateTime LastPurchaseDate { get; set; }
    public int TotalVisits { get; set; }
}

// On DbContext
public DbSet<Person> People { get; set; }
public DbSet<Employee> Employees { get; set; }
public DbSet<Customer> Customers { get; set; }
```

IdBirthDate

PersonId AdmissionDate JobDescription

PersonId LastPurchaseDate TotalVisits

'PersonId'People.Id

-
-
-

-
-

EntityFramework https://riptutorial.com/zh-TW/entity-framework/topic/7715/entityframework--

9: SQLite

SQLiteSQL。.NET SQLiteEntity Framework SQLite.NET。Entity Framework SQLite。

Examples

Entity FrameworkSQLite

SQL Server。SQLite。NuGet。

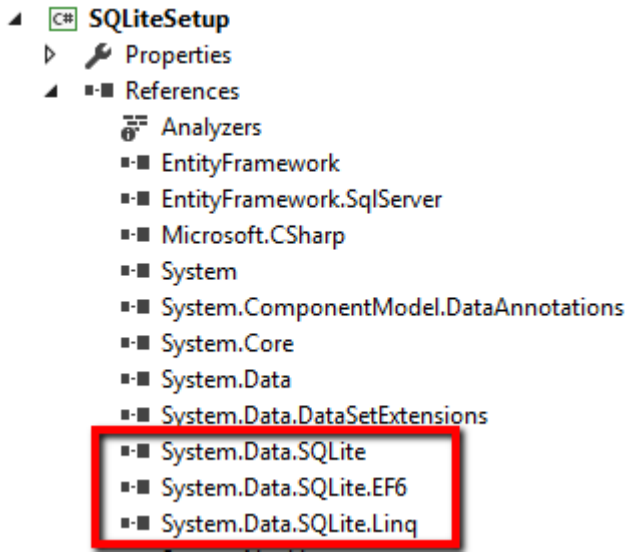
SQLite

NuGetmananged。`Install-Package System.Data.SQLite`。

```
PM> Install-Package System.Data.SQLite
Attempting to gather dependency information for package 'System.Data.SQLite.1.0.104' with respect to proje
Attempting to resolve dependencies for package 'System.Data.SQLite.1.0.104' with DependencyBehavior 'Lowes
Resolving actions to install package 'System.Data.SQLite.1.0.104'
Resolved actions to install package 'System.Data.SQLite.1.0.104'
Adding package 'EntityFramework.6.0.0' to folder 'c:\users\jason.tyler\documents\visual studio 2015\Projec
Added package 'EntityFramework.6.0.0' to folder 'c:\users\jason.tyler\documents\visual studio 2015\Project
Added package 'EntityFramework.6.0.0' to 'packages.config'
Executing script file 'c:\users\jason.tyler\documents\visual studio 2015\Projects\SQLiteSetup\packages\Ent
Executing script file 'c:\users\jason.tyler\documents\visual studio 2015\Projects\SQLiteSetup\packages\Ent

Type 'get-help EntityFramework' to see all available Entity Framework commands.
Successfully installed 'EntityFramework 6.0.0' to SQLiteSetup
Adding package 'System.Data.SQLite.Core.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 2
Added package 'System.Data.SQLite.Core.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 20
Added package 'System.Data.SQLite.Core.1.0.104' to 'packages.config'
Successfully installed 'System.Data.SQLite.Core 1.0.104' to SQLiteSetup
Adding package 'System.Data.SQLite.EF6.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 20
Added package 'System.Data.SQLite.EF6.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 201
Added package 'System.Data.SQLite.EF6.1.0.104' to 'packages.config'
Executing script file 'c:\users\jason.tyler\documents\visual studio 2015\Projects\SQLiteSetup\packages\Sys
Successfully installed 'System.Data.SQLite.EF6 1.0.104' to SQLiteSetup
Adding package 'System.Data.SQLite.Linq.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 2
Added package 'System.Data.SQLite.Linq.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 20
Added package 'System.Data.SQLite.Linq.1.0.104' to 'packages.config'
Successfully installed 'System.Data.SQLite.Linq 1.0.104' to SQLiteSetup
Adding package 'System.Data.SQLite.1.0.104', which only has dependencies, to project 'SQLiteSetup'.
Adding package 'System.Data.SQLite.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 2015\P
Added package 'System.Data.SQLite.1.0.104' to folder 'c:\users\jason.tyler\documents\visual studio 2015\Pr
Added package 'System.Data.SQLite.1.0.104' to 'packages.config'
Successfully installed 'System.Data.SQLite 1.0.104' to SQLiteSetup
```

`System.Data.SQLite`。`System.Data.SQLite.EF6` SQLiteEF。SQLite。



SQLite`SQLite.Interop.dll` SQLite SQLite

x86 / x64

App.config

SQLite `app.config`

`app.config`SQLiteSQLite EF

`DbProviderFactories` `system.data`

```
<DbProviderFactories>
  <remove invariant="System.Data.SQLite.EF6" />
  <add name="SQLite Data Provider (Entity Framework 6)" invariant="System.Data.SQLite.EF6"
description=".NET Framework Data Provider for SQLite (Entity Framework 6)"
type="System.Data.SQLite.EF6.SQLiteProviderFactory, System.Data.SQLite.EF6" />
  <remove invariant="System.Data.SQLite" /><add name="SQLite Data Provider"
invariant="System.Data.SQLite" description=".NET Framework Data Provider for SQLite"
type="System.Data.SQLite.SQLiteFactory, System.Data.SQLite" />
</DbProviderFactories>
```

```
<DbProviderFactories>
    <add name="SQLite Data Provider" invariant="System.Data.SQLite.EF6" description=".NET
Framework Data Provider for SQLite" type="System.Data.SQLite.SQLiteFactory,
System.Data.SQLite" />
</DbProviderFactories>
```

EF6 SQLiteSQLite

SQLite

SQLite

```
<connectionStrings>
    <add name="TestContext" connectionString="data source=testdb.sqlite;initial
catalog=Test;App=EntityFramework;" providerName="System.Data.SQLite.EF6"/>
</connectionStrings>
```

`provider` ◦ `System.Data.SQLite.EF6` ◦ EFSQLite◦ `data source`SQLite◦

SQLite DbContext

SQLite`DbContext` ◦

```
public class TestContext : DbContext
{
    public TestContext()
        : base("name=TestContext") { }
}
```

`name=TestContext` `app.config`TestContext◦ SQLiteSQLite◦

SQLite https://riptutorial.com/zh-TW/entity-framework/topic/9280/sqlite

10:

EntityFramework。 。

```
public class Company
{
    public int Id { get; set; }
    public string FullName { get; set; }
    public string ShortName { get; set; }

    // Navigation properties
    public virtual Person Founder { get; set; }
    public virtual ICollection<Address> Addresses { get; set; }
}

public class Address
{
    public int Id { get; set; }
    public int CompanyId { get; set; }
    public int CountryId { get; set; }
    public int CityId { get; set; }
    public string Street { get; set; }

    // Navigation properties
    public virtual Company Company { get; set; }
    public virtual Country Country { get; set; }
    public virtual City City { get; set; }
}
```

Examples

。 proeprties。 。

```
int companyId = ...;
Company company = context.Companies
    .First(m => m.Id == companyId);
Person founder = company.Founder; // Founder is loaded
foreach (Address address in company.Addresses)
{
    // Address details are loaded one by one.
}
```

Lazyvirtual

```
public Person Founder { get; set; } // "virtual" keyword has been removed
```

Context

```
public class MyContext : DbContext
{
    public MyContext(): base("Name=ConnectionString")
    {
        this.Configuration.LazyLoadingEnabled = false;
```

```
    }
}
```

。。。

。*Eager*。。

Include。

。

```
// Load one company with founder and address details
int companyId = ...;
Company company = context.Companies
    .Include(m => m.Founder)
    .Include(m => m.Addresses)
    .SingleOrDefault(m => m.Id == companyId);

// Load 5 companies with address details, also retrieve country and city
// information of addresses
List<Company> companies = context.Companies
    .Include(m => m.Addresses.Select(a => a.Country));
    .Include(m => m.Addresses.Select(a => a.City))
    .Take(5).ToList();
```

Entity Framework 4.1。`using System.Data.Entity;using System.Data.Entity;`。

。

```
// Load one company with founder and address details
int companyId = ...;
Company company = context.Companies
    .Include("Founder")
    .Include("Addresses")
    .SingleOrDefault(m => m.Id == companyId);

// Load 5 companies with address details, also retrieve country and city
// information of addresses
List<Company> companies = context.Companies
    .Include("Addresses.Country");
    .Include("Addresses.City"))
    .Take(5).ToList();
```

LazyLoad。 *Collection*。

```
Company company = context.Companies.FirstOrDefault();
// Load founder
context.Entry(company).Reference(m => m.Founder).Load();
// Load addresses
context.Entry(company).Collection(m => m.Addresses).Load();
```

*Eager*entiteis

```
Company company = context.Companies.FirstOrDefault();
// Load founder
context.Entry(company).Reference("Founder").Load();
// Load addresses
context.Entry(company).Collection("Addresses").Load();
```

-

Query

```
Company company = context.Companies.FirstOrDefault();
// Load addresses which are in Baku
context.Entry(company)
    .Collection(m => m.Addresses)
    .Query()
    .Where(a => a.City.Name == "Baku")
    .Load();
```

- -

```
var dbContext = new MyDbContext();
var denormalizedType = from company in dbContext.Company
                       where company.Name == "MyFavoriteCompany"
                       join founder in dbContext.Founder
                       on company.FounderId equals founder.Id
                       select new
                       {
                           CompanyName = company.Name,
                           CompanyId = company.Id,
                           FounderName = founder.Name,
                           FounderId = founder.Id
                       };
```

```
var dbContext = new MyDbContext();
var denormalizedType = dbContext.Company
                       .Join(dbContext.Founder,
                                   c => c.FounderId,
                                   f => f.Id ,
                                   (c, f) => new
                                   {
                                       CompanyName = c.Name,
                                       CompanyId = c.Id,
                                       FounderName = f.Name,
                                       FounderId = f.Id
                                   })
                       .Select(cf => cf);
```

https://riptutorial.com/zh-TW/entity-framework/topic/4678/

11: Code First Migrations

Examples

Code First Migrations

```
Enable-Migrations
```

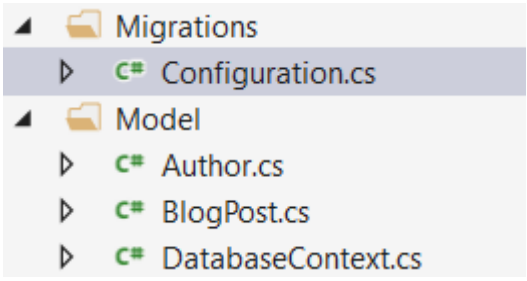
◦

DbContextEF◦ BlogPostAuthor

```
internal class DatabaseContext: DbContext
{
    public DbSet<Author> Authors { get; set; }

    public DbSet<BlogPost> BlogPosts { get; set; }
}
```

```
PM> Enable-Migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project <YourProjectName>.
PM>
```



MigrationsConfiguration.cs

◦

◦

```
Add-Migration <migration-name>
```

UpDown◦

Initial

```
PM> Add-Migration Initial
Scaffolding migration 'Initial'.
The Designer Code for this migration file includes a snapshot of your current Code
First model. This snapshot is used to calculate the changes to your model when you
scaffold the next migration. If you make additional changes to your model that you
want to include in this migration, then you can re-scaffold it by running
'Add-Migration Initial' again.
```

_Initial.cs

```
public override void Up()
{
   CreateTable(
       "dbo.Authors",
        c => new
           {
               AuthorId = c.Int(nullable: false, identity: true),
               Name = c.String(maxLength: 128),
           })
        .PrimaryKey(t => t.AuthorId);

    CreateTable(
       "dbo.BlogPosts",
       c => new
           {
                Id = c.Int(nullable: false, identity: true),
                Title = c.String(nullable: false, maxLength: 128),
                Message = c.String(),
                Author_AuthorId = c.Int(),
            })
        .PrimaryKey(t => t.Id)
        .ForeignKey("dbo.Authors", t => t.Author_AuthorId)
        .Index(t => t.Author_AuthorId);
}

public override void Down()
{
    DropForeignKey("dbo.BlogPosts", "Author_AuthorId", "dbo.Authors");
    DropIndex("dbo.BlogPosts", new[] { "Author_AuthorId" });
    DropTable("dbo.BlogPosts");
    DropTable("dbo.Authors");
}
```

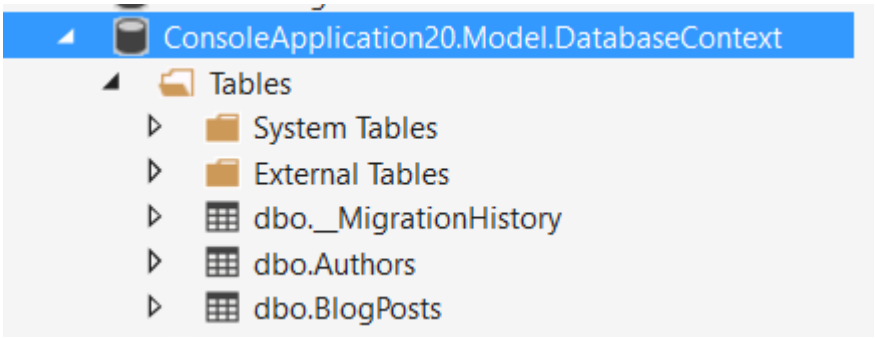
Up()AuthorsBlogPosts BlogPosts◦ Author_AuthorId◦ Down()◦

```
Update-Database
```

Initial -migration

```
PM> update-database
Specify the '-Verbose' flag to view the SQL statements being applied to the target
database.
Applying explicit migrations: [201609302203541_Initial].
Applying explicit migration: 201609302203541_Initial.
Running Seed method.
```

SQL



Add-MigrationUpdate-Database◦

```
get-help Add-Migration
```

```
get-help Update-Database
```

◦ enable-migrationsConfiguration'Seed'◦

Seed()EF

```
protected override void Seed(Model.DatabaseContext context);
```

Seed()◦ ◦

```
protected override void Seed(Model.DatabaseContext context)
{
    if (!context.Customers.Any()) {
        Customer c = new Customer{ Id = 1, Name = "Demo" };
        context.Customers.Add(c);
        context.SaveData();
    }
}
```

EFAddOrUpdate() ◦ Configuration.cs

```
protected override void Seed(Model.DatabaseContext context)
{
    context.People.AddOrUpdate(
      p => p.FullName,
      new Person { FullName = "Andrew Peters" },
      new Person { FullName = "Brice Lambson" },
      new Person { FullName = "Rowan Miller" }
    );
}
```

Seed() ◦ ◦

Sql

◦ NULL◦ “0”defaultdefaultSql◦ Up()Down()Sql()◦

◦ ◦ ◦ fullname@example.com ◦ [Required]Email

```
public partial class AuthorsEmailRequired : DbMigration
{
    public override void Up()
    {
        AlterColumn("dbo.Authors", "Email", c => c.String(nullable: false, maxLength: 512));
    }

    public override void Down()
    {
        AlterColumn("dbo.Authors", "Email", c => c.String(maxLength: 512));
    }
}
```

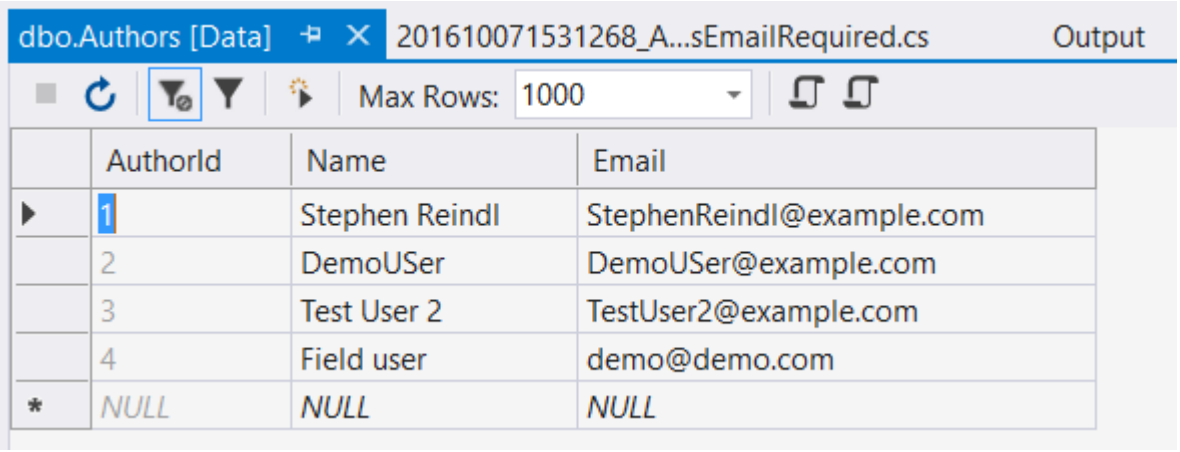
NULL

NULL“”'App.Model.DatabaseContext.dbo.Authors';◦ ◦

AlterColumn

```
Sql(@"Update dbo.Authors
    set Email = REPLACE(name, ' ', '') + N'@example.com'
    where Email is null");
```

update-database



Sql()DMLDDL◦ ;SQL◦

“ _ ”

◦ Add-Migration◦

- Visual Studio
- /◦

◦ ◦

```
void UpdateDatabase(MyDbConfiguration configuration) {
   DbMigrator dbMigrator = new DbMigrator( configuration);
   if ( dbMigrator.GetPendingMigrations().Any() )
   {
      // there are pending migrations run the migration job
      dbMigrator.Update();
   }
}
```

`MyDbConfiguration`

```
public class MyDbConfiguration : DbMigrationsConfiguration<ApplicationDbContext>
```

1. ◦
2. “”`Install-Package EntityFramework`EntityFramework nuget
3. app.config`providerName="System.Data.SqlClient"`◦
4. “ `Blog` ”
5. DbContextContextClass“ `BlogContext` ”
6. DbSet

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

    public string Name { get; set; }
}
public class BlogContext: DbContext
{
    public BlogContext(): base("name=Your_Connection_Name")
    {
    }

    public virtual DbSet<Blog> Blogs{ get; set; }
}
```

7. Your_Connection_Name
8. “ `Enable-Migration`
9. `Add-Migration Your_Arbitrary_Migraiton_Name`UpDown
10. `Update-Database`

Code First Migrations https://riptutorial.com/zh-TW/entity-framework/topic/7157/code-first-migrations

12: .t4

Examples

◦ ◦

.ttEntityClassOpeningPOLICY_NOIPolicyNumberIUniqueIdUNIQUE_ID

```
public string EntityClassOpening(EntityType entity)
{
    var stringsToMatch = new Dictionary<string,string> { { "POLICY_NO", "IPolicyNumber" }, {
"UNIQUE_ID", "IUniqueId" } };
    return string.Format(
        CultureInfo.InvariantCulture,
        "{0} {1}partial class {2}{3}{4}",
        Accessibility.ForType(entity),
        _code.SpaceAfter(_code.AbstractOption(entity)),
        _code.Escape(entity),
        _code.StringBefore(" : ", _typeMapper.GetTypeName(entity.BaseType)),
        stringsToMatch.Any(o => entity.Properties.Any(n => n.Name == o.Key)) ? " : " +
string.Join(", ", stringsToMatch.Join(entity.Properties, l => l.Key, r => r.Name, (l,r) =>
l.Value)) : string.Empty);
}
```

.tt◦

XML

◦ XML*[modelname]* .tt *modelname*EDMX

```
foreach (var entity in typeMapper.GetItemsToGenerate<EntityType>(itemCollection))
{
    fileManager.StartNewFile(entity.Name + ".cs");
    BeginNamespace(code); // used to write model namespace
#>
<#=codeStringGenerator.UsingDirectives(inHeader: false)#>
```

UsingDirectivesXML

```
foreach (var entity in typeMapper.GetItemsToGenerate<EntityType>(itemCollection))
{
    fileManager.StartNewFile(entity.Name + ".cs");
    BeginNamespace(code);
#>
/// <summary>
/// <#=entity.Name#> model entity class.
/// </summary>
<#=codeStringGenerator.UsingDirectives(inHeader: false)#>
```

◦

```
/// <summary>
```

```
/// Example model entity class.
/// </summary>
public partial class Example
{
    // model contents
}
```

.t4 https://riptutorial.com/zh-TW/entity-framework/topic/3964/-t4

13:

Examples

Entity Framework - MSSQL`ExampleDatabase`。

`dbo.People`。

```
    class Person
    {
        public int PersonId { get; set; }
        public string FirstName { get; set; }
    }
```

Entity Framework`dbo.People` `PersonId`varcharmax`FirstName`。

`System.Data.Entity.DbContext`。

```
    class Context : DbContext
    {
        public Context(string connectionString) : base(connectionString)
        {
            Database.SetInitializer<Context>(null);
        }

        public DbSet<Person> People { get; set; }
    }
```

null - Entity Framework。

`FirstName`

```
    class Program
    {
        static void Main(string[] args)
        {
            using (var ctx = new Context("DbConnectionString"))
            {
                var firstPerson = ctx.People.FirstOrDefault();
                if (firstPerson != null) {
                    firstPerson.FirstName = "John";
                    ctx.SaveChanges();
                }
            }
        }
    }
```

“DbConnectionString”Context。 `app.config`

```
  <connectionStrings>
    <add  name="DbConnectionString"
```

```
    connectionString="Data Source=.;Initial Catalog=ExampleDatabase;Integrated Security=True"
    providerName="System.Data.SqlClient"/>
  </connectionStrings>
```

https://riptutorial.com/zh-TW/entity-framework/topic/5337/

14:

Entity Framework。

SOLID

.....

Examples

1-@

“Company”

[dbo]。[][CategoryID][CategoryName]

[dbo]。[][ProductID][CategoryID][ProductName]

1-1

```
public partial class CompanyContext : DbContext
public partial class Product
public partial class Category
```

1-2

```
public interface IDbRepository : IDisposable
{
    #region Tables and Views functions

    IQueryable<TResult> GetAll<TResult>(bool noTracking = true) where TResult : class;
    TEntity Add<TEntity>(TEntity entity) where TEntity : class;
    TEntity Delete<TEntity>(TEntity entity) where TEntity : class;
    TEntity Attach<TEntity>(TEntity entity) where TEntity : class;
    TEntity AttachIfNot<TEntity>(TEntity entity) where TEntity : class;

    #endregion Tables and Views functions

    #region Transactions Functions

    int Commit();
    Task<int> CommitAsync(CancellationToken cancellationToken = default(CancellationToken));

    #endregion Transactions Functions

    #region Database Procedures and Functions

    TResult Execute<TResult>(string functionName, params object[] parameters);

    #endregion Database Procedures and Functions
}
```

1-3

```
/// <summary>
/// Implementing basic tables, views, procedures, functions, and transaction functions
/// Select (GetAll), Insert (Add), Delete, and Attach
/// No Edit (Modify) function (can modify attached entity without function call)
/// Executes database procedures or functions (Execute)
/// Transaction functions (Commit)
/// More functions can be added if needed
/// </summary>
/// <typeparam name="TEntity">Entity Framework table or view</typeparam>
public class DbRepository : IDbRepository
{
    #region Protected Members

    protected DbContext _dbContext;

    #endregion Protected Members

    #region Constractors

    /// <summary>
    /// Repository constructor
    /// </summary>
    /// <param name="dbContext">Entity framework databse context</param>
    public DbRepository(DbContext dbContext)
    {
        _dbContext = dbContext;

        ConfigureContext();
    }

    #endregion Constractors

    #region IRepository Implementation

    #region Tables and Views functions

    /// <summary>
    /// Query all
    /// Set noTracking to true for selecting only (read-only queries)
    /// Set noTracking to false for insert, update, or delete after select
    /// </summary>
    public virtual IQueryable<TResult> GetAll<TResult>(bool noTracking = true) where TResult :
class
    {
        var entityDbSet = GetDbSet<TResult>();

        if (noTracking)
            return entityDbSet.AsNoTracking();

        return entityDbSet;
    }

    public virtual TEntity Add<TEntity>(TEntity entity) where TEntity : class
    {
        return GetDbSet<TEntity>().Add(entity);
    }

    /// <summary>
    /// Delete loaded (attached) or unloaded (Detached) entitiy
```

```
    /// No need to load object to delete it
    /// Create new object of TEntity and set the id then call Delete function
    /// </summary>
    /// <param name="entity">TEntity</param>
    /// <returns></returns>
    public virtual TEntity Delete<TEntity>(TEntity entity) where TEntity : class
    {
        if (_dbContext.Entry(entity).State == EntityState.Detached)
        {
            _dbContext.Entry(entity).State = EntityState.Deleted;
            return entity;
        }
        else
            return GetDbSet<TEntity>().Remove(entity);
    }

    public virtual TEntity Attach<TEntity>(TEntity entity) where TEntity : class
    {
        return GetDbSet<TEntity>().Attach(entity);
    }

    public virtual TEntity AttachIfNot<TEntity>(TEntity entity) where TEntity : class
    {
        if (_dbContext.Entry(entity).State == EntityState.Detached)
            return Attach(entity);

        return entity;
    }

    #endregion Tables and Views functions

    #region Transactions Functions

    /// <summary>
    /// Saves all changes made in this context to the underlying database.
    /// </summary>
    /// <returns>The number of objects written to the underlying database.</returns>
    public virtual int Commit()
    {
        return _dbContext.SaveChanges();
    }

    /// <summary>
    /// Asynchronously saves all changes made in this context to the underlying database.
    /// </summary>
    /// <param name="cancellationToken">A System.Threading.CancellationToken to observe while
waiting for the task to complete.</param>
    /// <returns>A task that represents the asynchronous save operation.  The task result
contains the number of objects written to the underlying database.</returns>
    public virtual Task<int> CommitAsync(CancellationToken cancellationToken =
default(CancellationToken))
    {
        return _dbContext.SaveChangesAsync(cancellationToken);
    }

    #endregion Transactions Functions

    #region Database Procedures and Functions

    /// <summary>
    /// Executes any function in the context
```

```
    /// use to call database procesdures and functions
    /// </summary>>
    /// <typeparam name="TResult">return function type</typeparam>
    /// <param name="functionName">context function name</param>
    /// <param name="parameters">context function parameters in same order</param>
    public virtual TResult Execute<TResult>(string functionName, params object[] parameters)
    {
        MethodInfo method = _dbContext.GetType().GetMethod(functionName);

        return (TResult)method.Invoke(_dbContext, parameters);
    }

    #endregion Database Procedures and Functions

    #endregion IRepository Implementation

    #region IDisposable Implementation

    public void Dispose()
    {
        _dbContext.Dispose();
    }

    #endregion IDisposable Implementation

    #region Protected Functions

    /// <summary>
    /// Set Context Configuration
    /// </summary>
    protected virtual void ConfigureContext()
    {
        // set your recommended Context Configuration
        _dbContext.Configuration.LazyLoadingEnabled = false;
    }

    #endregion Protected Functions

    #region Private Functions

    private DbSet<TEntity> GetDbSet<TEntity>() where TEntity : class
    {
        return _dbContext.Set<TEntity>();
    }

    #endregion Private Functions

}
```

2-@

。

。

```
/// <summary>
/// Contains Product Business functions
/// </summary>
public interface IProductBusiness
```

```
{
    Product SelectById(int productId, bool noTracking = true);
    Task<IEnumerable<dynamic>> SelectByCategoryAsync(int CategoryId);
    Task<Product> InsertAsync(string productName, int categoryId);
    Product InsertForNewCategory(string productName, string categoryName);
    Product Update(int productId, string productName, int categoryId);
    Product Update2(int productId, string productName, int categoryId);
    int DeleteWithoutLoad(int productId);
    int DeleteLoadedProduct(Product product);
    IEnumerable<GetProductsCategory_Result> GetProductsCategory(int categoryId);
}



/// <summary>
/// Implementing Product Business functions
/// </summary>
public class ProductBusiness : IProductBusiness
{
    #region Private Members

    private IDbRepository _dbRepository;

    #endregion Private Members

    #region Constructors

    /// <summary>
    /// Product Business Constructor
    /// </summary>
    /// <param name="dbRepository"></param>
    public ProductBusiness(IDbRepository dbRepository)
    {
        _dbRepository = dbRepository;
    }

    #endregion Constructors

    #region IProductBusiness Function

    /// <summary>
    /// Selects Product By Id
    /// </summary>
    public Product SelectById(int productId, bool noTracking = true)
    {
        var products = _dbRepository.GetAll<Product>(noTracking);

        return products.FirstOrDefault(pro => pro.ProductID == productId);
    }

    /// <summary>
    /// Selects Products By Category Id Async
    /// To have async method, add reference to EntityFramework 6 dll or higher
    /// also you need to have the namespace "System.Data.Entity"
    /// </summary>
    /// <param name="CategoryId">CategoryId</param>
    /// <returns>Return what ever the object that you want to return</returns>
    public async Task<IEnumerable<dynamic>> SelectByCategoryAsync(int CategoryId)
    {
        var products = _dbRepository.GetAll<Product>();
        var categories = _dbRepository.GetAll<Category>();
```

```
        var result = (from pro in products
                      join cat in categories
                      on pro.CategoryID equals cat.CategoryID
                      where pro.CategoryID == CategoryId
                      select new
                      {
                          ProductId = pro.ProductID,
                          ProductName = pro.ProductName,
                          CategoryName = cat.CategoryName
                      }
                              );

        return await result.ToListAsync();
    }

    /// <summary>
    /// Insert Async new product for given category
    /// </summary>
    public async Task<Product> InsertAsync(string productName, int categoryId)
    {
        var newProduct = _dbRepository.Add(new Product() { ProductName = productName,
CategoryID = categoryId });

        await _dbRepository.CommitAsync();

        return newProduct;
    }

    /// <summary>
    /// Insert new product and new category
    /// Do many database actions in one transaction
    /// each _dbRepository.Commit(); will commit one transaction
    /// </summary>
    public Product InsertForNewCategory(string productName, string categoryName)
    {
        var newCategory = _dbRepository.Add(new Category() { CategoryName = categoryName });
        var newProduct = _dbRepository.Add(new Product() { ProductName = productName, Category
= newCategory });

        _dbRepository.Commit();

        return newProduct;
    }

    /// <summary>
    /// Update given product with tracking
    /// </summary>
    public Product Update(int productId, string productName, int categoryId)
    {
        var product = SelectById(productId,false);
        product.CategoryID = categoryId;
        product.ProductName = productName;

        _dbRepository.Commit();

        return product;
    }

    /// <summary>
    /// Update given product with no tracking and attach function
```

```
    /// </summary>
    public Product Update2(int productId, string productName, int categoryId)
    {
        var product = SelectById(productId);
        _dbRepository.Attach(product);

        product.CategoryID = categoryId;
        product.ProductName = productName;

        _dbRepository.Commit();

        return product;
    }

    /// <summary>
    /// Deletes product without loading it
    /// </summary>
    public int DeleteWithoutLoad(int productId)
    {
        _dbRepository.Delete(new Product() { ProductID = productId });

        return _dbRepository.Commit();
    }

    /// <summary>
    /// Deletes product after loading it
    /// </summary>
    public int DeleteLoadedProduct(Product product)
    {
        _dbRepository.Delete(product);

        return _dbRepository.Commit();
    }

    /// <summary>
    /// Assuming we have the following procedure in database
    /// PROCEDURE [dbo].[GetProductsCategory] @CategoryID INT, @OrderBy VARCHAR(50)
    /// </summary>
    public IEnumerable<GetProductsCategory_Result> GetProductsCategory(int categoryId)
    {
        return
_dbRepository.Execute<IEnumerable<GetProductsCategory_Result>>("GetProductsCategory",
categoryId, "ProductName DESC");
    }


    #endregion IProductBusiness Function
}
```

3-@MVC

PresentationBusiness。MVC。

IoCUnityIoC

3-1MVCUnity

3-1-1“ASP.NET MVCUnity bootstrapper”NuGet

3-1-2UnityWebActivator.Start;Global.asax.csApplication_Start

3-1-3UnityConfig.RegisterTypes

```
    public static void RegisterTypes(IUnityContainer container)
    {
        // Data Access Layer
        container.RegisterType<DbContext, CompanyContext>(new PerThreadLifetimeManager());
        container.RegisterType(typeof(IDbRepository), typeof(DbRepository), new
PerThreadLifetimeManager());

        // Business Layer
        container.RegisterType<IProductBusiness, ProductBusiness>(new
PerThreadLifetimeManager());

    }
```

3-2@MVC

```
public class ProductController : Controller
{
    #region Private Members

    IProductBusiness _productBusiness;

    #endregion Private Members

    #region Constractors

    public ProductController(IProductBusiness productBusiness)
    {
        _productBusiness = productBusiness;
    }

    #endregion Constractors

    #region Action Functions

    [HttpPost]
    public ActionResult InsertForNewCategory(string productName, string categoryName)
    {
        try
        {
            // you can use any of IProductBusiness functions
            var newProduct = _productBusiness.InsertForNewCategory(productName, categoryName);

            return Json(new { success = true, data = newProduct });
        }
        catch (Exception ex)
        {   /* log ex*/
            return Json(new { success = false, errorMessage = ex.Message});
        }
    }

    [HttpDelete]
    public ActionResult SmartDeleteWithoutLoad(int productId)
    {
        try
        {
```

```
            // deletes product without load
            var deletedProduct = _productBusiness.DeleteWithoutLoad(productId);

            return Json(new { success = true, data = deletedProduct });
        }
        catch (Exception ex)
        {   /* log ex*/
            return Json(new { success = false, errorMessage = ex.Message });
        }
    }

    public async Task<ActionResult> SelectByCategoryAsync(int CategoryId)
    {
        try
        {
            var results = await _productBusiness.SelectByCategoryAsync(CategoryId);

            return Json(new { success = true, data = results },JsonRequestBehavior.AllowGet);
        }
        catch (Exception ex)
        {   /* log ex*/
            return Json(new { success = false, errorMessage = ex.Message
},JsonRequestBehavior.AllowGet);
        }
    }
    #endregion Action Functions
}
```

4-@

。。

IDbRepository

4-1

```
class FakeDbRepository : IDbRepository
{
    #region Protected Members

    protected Hashtable _dbContext;
    protected int _numberOfRowsAffected;
    protected Hashtable _contextFunctionsResults;

    #endregion Protected Members

    #region Constractors

    public FakeDbRepository(Hashtable contextFunctionsResults = null)
    {
        _dbContext = new Hashtable();
        _numberOfRowsAffected = 0;
        _contextFunctionsResults = contextFunctionsResults;
    }

    #endregion Constractors

    #region IRepository Implementation
```

```
    #region Tables and Views functions

    public IQueryable<TResult> GetAll<TResult>(bool noTracking = true) where TResult : class
    {
        return GetDbSet<TResult>().AsQueryable();
    }

    public TEntity Add<TEntity>(TEntity entity) where TEntity : class
    {
        GetDbSet<TEntity>().Add(entity);
        ++_numberOfRowsAffected;
        return entity;
    }

    public TEntity Delete<TEntity>(TEntity entity) where TEntity : class
    {
        GetDbSet<TEntity>().Remove(entity);
        ++_numberOfRowsAffected;
        return entity;
    }

    public TEntity Attach<TEntity>(TEntity entity) where TEntity : class
    {
        return Add(entity);
    }

    public TEntity AttachIfNot<TEntity>(TEntity entity) where TEntity : class
    {
        if (!GetDbSet<TEntity>().Contains(entity))
            return Attach(entity);

        return entity;
    }

    #endregion Tables and Views functions

    #region Transactions Functions


    public virtual int Commit()
    {
        var numberOfRowsAffected = _numberOfRowsAffected;
        _numberOfRowsAffected = 0;
        return numberOfRowsAffected;
    }

    public virtual Task<int> CommitAsync(CancellationToken cancellationToken =
default(CancellationToken))
    {
        var numberOfRowsAffected = _numberOfRowsAffected;
        _numberOfRowsAffected = 0;
        return new Task<int>(() => numberOfRowsAffected);
    }

    #endregion Transactions Functions

    #region Database Procedures and Functions

    public virtual TResult Execute<TResult>(string functionName, params object[] parameters)
    {
```

```
        if (_contextFunctionsResults != null &&
_contextFunctionsResults.Contains(functionName))
            return (TResult)_contextFunctionsResults[functionName];

        throw new NotImplementedException();
    }

    #endregion Database Procedures and Functions

    #endregion IRepository Implementation

    #region IDisposable Implementation

    public void Dispose()
    {

    }

    #endregion IDisposable Implementation

    #region Private Functions

    private List<TEntity> GetDbSet<TEntity>() where TEntity : class
    {
        if (!_dbContext.Contains(typeof(TEntity)))
            _dbContext.Add(typeof(TEntity), new List<TEntity>());

        return (List<TEntity>)_dbContext[typeof(TEntity)];
    }

    #endregion Private Functions
}
```

4-2

```
[TestClass]
public class ProductUnitTest
{
    [TestMethod]
    public void TestInsertForNewCategory()
    {
        // Initialize repositories
        FakeDbRepository _dbRepository = new FakeDbRepository();

        // Initialize Business object
        IProductBusiness productBusiness = new ProductBusiness(_dbRepository);

        // Process test method
        productBusiness.InsertForNewCategory("Test Product", "Test Category");

        int _productCount = _dbRepository.GetAll<Product>().Count();
        int _categoryCount = _dbRepository.GetAll<Category>().Count();

        Assert.AreEqual<int>(1, _productCount);
        Assert.AreEqual<int>(1, _categoryCount);
    }

    [TestMethod]
    public void TestProceduresFunctionsCall()
    {
```

```
        // Initialize Procedures / Functions result
        Hashtable _contextFunctionsResults = new Hashtable();
        _contextFunctionsResults.Add("GetProductsCategory", new
List<GetProductsCategory_Result> {
            new GetProductsCategory_Result() { ProductName = "Product 1", ProductID = 1,
CategoryName = "Category 1" },
            new GetProductsCategory_Result() { ProductName = "Product 2", ProductID = 2,
CategoryName = "Category 1" },
            new GetProductsCategory_Result() { ProductName = "Product 3", ProductID = 3,
CategoryName = "Category 1" }});

        // Initialize repositories
        FakeDbRepository _dbRepository = new FakeDbRepository(_contextFunctionsResults);

        // Initialize Business object
        IProductBusiness productBusiness = new ProductBusiness(_dbRepository);

        // Process test method
        var results = productBusiness.GetProductsCategory(1);

        Assert.AreEqual<int>(3, results.Count());
    }
}
```

https://riptutorial.com/zh-TW/entity-framework/topic/8879/--

15:

Examples

CreateDatabaseIfNotExists

IDatabaseInitializerEntityFramework◦ ◦ ◦

```
public class MyContext : DbContext {
    public MyContext() {
        Database.SetInitializer(new CreateDatabaseIfNotExists<MyContext>());
    }
}
```

DropCreateDatabaseIfModelChanges

IDatabaseInitializer◦

```
public class MyContext : DbContext {
    public MyContext() {
        Database.SetInitializer(new DropCreateDatabaseIfModelChanges<MyContext>());
    }
}
```

DropCreateDatabaseAlways

IDatabaseInitializer◦ ◦

```
public class MyContext : DbContext {
    public MyContext() {
        Database.SetInitializer(new DropCreateDatabaseAlways<MyContext>());
    }
}
```

IDatabaseInitializer◦

0◦ DbMigrationsConfiguration◦

```
public class RecreateFromScratch<TContext, TMigrationsConfiguration> :
IDatabaseInitializer<TContext>
where TContext : DbContext
where TMigrationsConfiguration : DbMigrationsConfiguration<TContext>, new()
{
    private readonly DbMigrationsConfiguration<TContext> _configuration;

    public RecreateFromScratch()
    {
        _configuration = new TMigrationsConfiguration();
    }
```

```
    public void InitializeDatabase(TContext context)
    {
        var migrator = new DbMigrator(_configuration);
        migrator.Update("0");
        migrator.Update();
    }
}
```

MigrateDatabaseToLatestVersion

`IDatabaseInitializer`Code First Migrations。 `DbMigrationsConfiguration`。

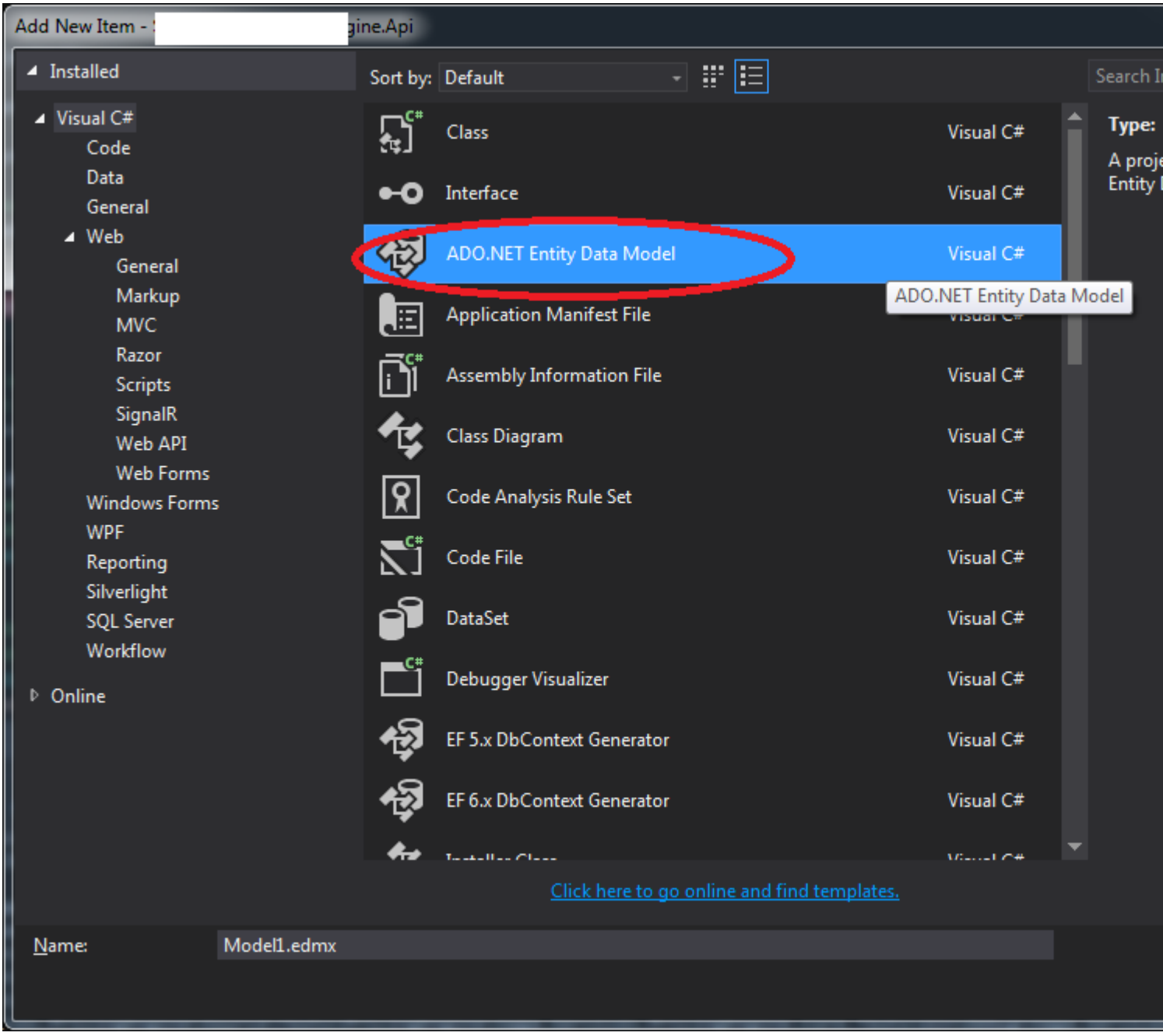
```
public class MyContext : DbContext {
    public MyContext() {
        Database.SetInitializer(
            new MigrateDatabaseToLatestVersion<MyContext, Configuration>());
    }
}
```

https://riptutorial.com/zh-TW/entity-framework/topic/5526/

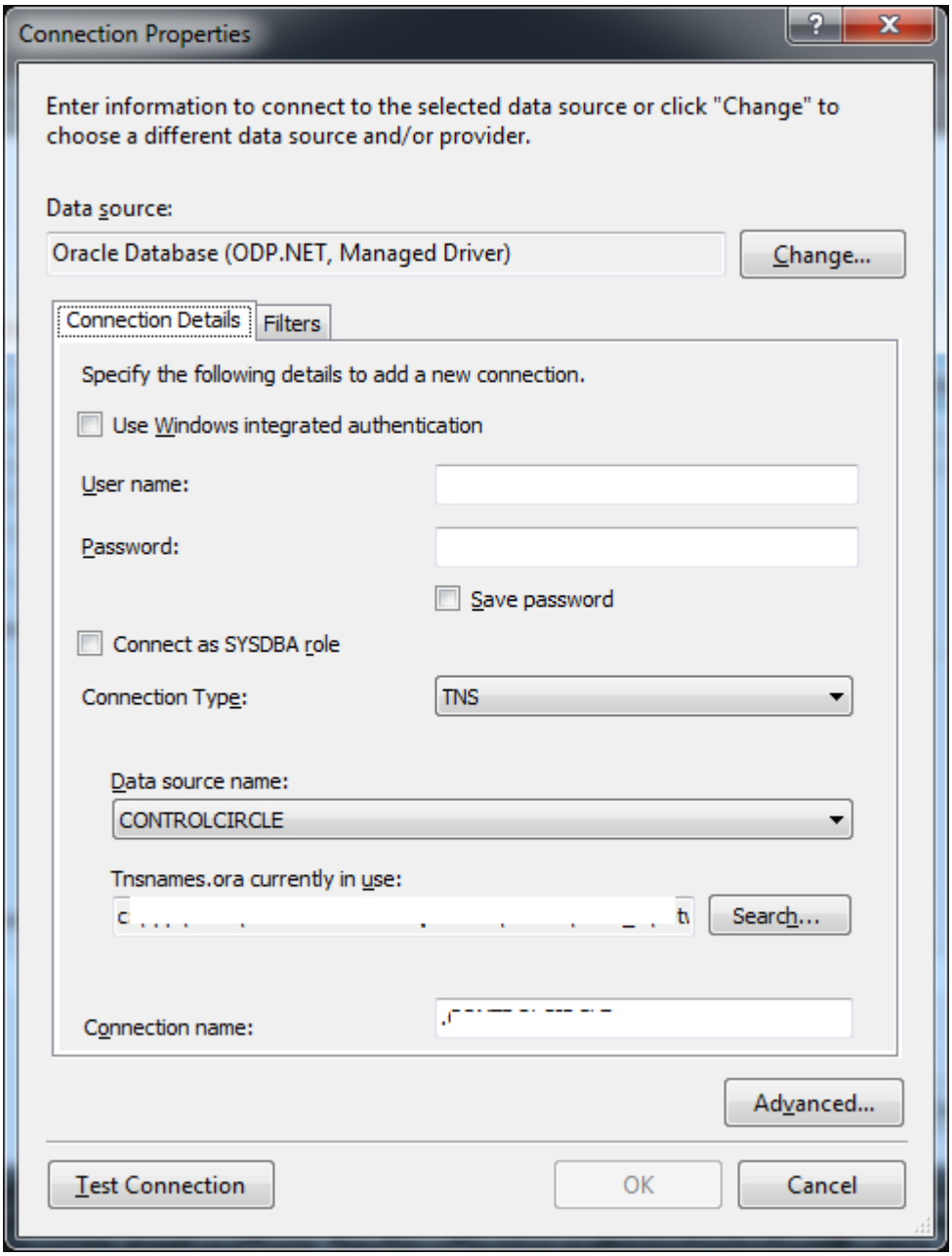
16:

Examples

Visual StudioSolution Explorer“ Project□□□□ ◦ ADO.NET Entity Data Model



Generate from databaseNext New Connection...MSSQL MySQLOracle



“ Test Connection◦

“ Next◦

“ Next ◦

Entity Framework 5T4◦ EDMX.tt UsingDirectivesusing

```
foreach (var entity in typeMapper.GetItemsToGenerate<EntityType>
(itemCollection))
{
    fileManager.StartNewFile(entity.Name + ".cs");
    BeginNamespace(code);
#>
<#=codeStringGenerator.UsingDirectives(inHeader: false)#>
using System.ComponentModel.DataAnnotations;  // --> add this line
```

KeyAttribute ◦ KeyAttribute codeStringGenerator.Property

```
var simpleProperties = typeMapper.GetSimpleProperties(entity);
    if (simpleProperties.Any())
    {
        foreach (var edmProperty in simpleProperties)
        {
#>
    <#=codeStringGenerator.Property(edmProperty)#>
<#
        }
    }
```

if

```
var simpleProperties = typeMapper.GetSimpleProperties(entity);
    if (simpleProperties.Any())
    {
        foreach (var edmProperty in simpleProperties)
        {
             if (ef.IsKey(edmProperty)) {
#>    [Key]
<#       }
#>
    <#=codeStringGenerator.Property(edmProperty)#>
<#
        }
    }
```

KeyAttribute ◦

```
using System;

public class Example
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

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

public class Example
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
}
```

https://riptutorial.com/zh-TW/entity-framework/topic/4414/

17:

Examples

UserType< - >UserType

```
public class UserType
{
    public int UserTypeId {get; set;}
}
public class User
{
    public int UserId {get; set;}
    public int UserTypeId {get; set;}
    public virtual UserType UserType {get; set;}
}

Entity<User>().HasRequired(u => u.UserType).WithMany().HasForeignKey(u => u.UserTypeId);
```

Nullable

```
public class UserType
{
    public int UserTypeId {get; set;}
}
public class User
{
    public int UserId {get; set;}
    public int? UserTypeId {get; set;}
    public virtual UserType UserType {get; set;}
}

Entity<User>().HasOptional(u => u.UserType).WithMany().HasForeignKey(u => u.UserTypeId);
```

/

```
public class UserType
{
    public int UserTypeId {get; set;}
    public virtual ICollection<User> Users {get; set;}
}
public class User
{
    public int UserId {get; set;}
    public int UserTypeId {get; set;}
    public virtual UserType UserType {get; set;}
}
```

```
Entity<User>().HasRequired(u => u.UserType).WithMany(ut => ut.Users).HasForeignKey(u =>
u.UserTypeId);
```

```
Entity<User>().HasOptional(u => u.UserType).WithMany(ut => ut.Users).HasForeignKey(u =>
u.UserTypeId);
```

https://riptutorial.com/zh-TW/entity-framework/topic/4528/

18:

System.Data.Entity.EntityState。

```
Added
Deleted
Detached
Modified
Unchanged
```

POCO。 。 ObjectStateManager。

。

Examples

EntityState.Added

1. ```
   context.Entry(entity).State = EntityState.Added;
   ```

2. DbSet

   ```
   context.Entities.Add(entity);
   ```

SaveChanges。 SaveChanges 。

AddedAdded 。

```
public class Planet
{
    public Planet()
    {
        Moons = new HashSet<Moon>();
    }
    public int ID { get; set; }
    public string Name { get; set; }
    public ICollection<Moon> Moons { get; set; }
}

public class Moon
{
    public int ID { get; set; }
    public int PlanetID { get; set; }
    public string Name { get; set; }
}
```

```
public class PlanetDb : DbContext
{
    public property DbSet<Planet> Planets { get; set; }
}
```
```

```
var mars = new Planet { Name = "Mars" };
mars.Moons.Add(new Moon { Name = "Phobos" });
mars.Moons.Add(new Moon { Name = "Deimos" });

context.Planets.Add(mars);

Console.WriteLine(context.Entry(mars).State);
Console.WriteLine(context.Entry(mars.Moons.First()).State);
```

```
Added
Added
```

PlanetAdded ◦

“Added “”Planet.Moons Added ◦

https://riptutorial.com/zh-TW/entity-framework/topic/5256/

19:

Entity Framework。

Examples

```
public class Person
{
  public int PersonId { get; set; }
  public string Name { get; set; }
}

public class Car
{
  public int CarId { get; set; }
  public string LicensePlate { get; set; }
}

public class MyDemoContext : DbContext
{
  public DbSet<Person> People { get; set; }
  public DbSet<Car> Cars { get; set; }
}
```

CarAPersonAPersonA"CarA。

```
public class Person
{
  public int PersonId { get; set; }
  public string Name { get; set; }
  public int CarId { get; set; }
  public virtual Car Car { get; set; }
}

public class Car
{
  public int CarId { get; set; }
  public string LicensePlate { get; set; }
  public int PersonId { get; set; }
  public virtual Person Person { get; set; }
}
```

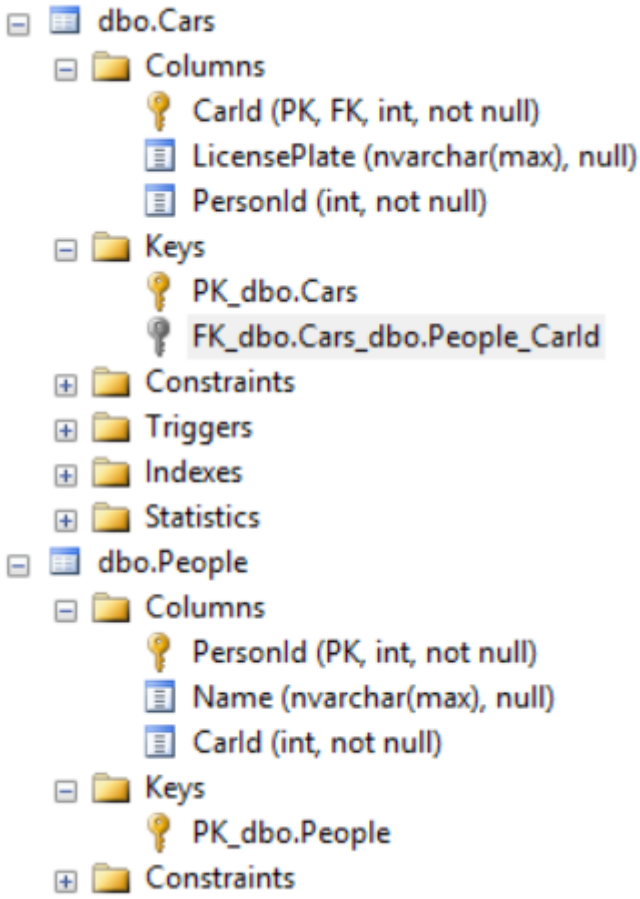
```
public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
{
  public CarEntityTypeConfiguration()
  {
     this.HasRequired(c => c.Person).WithOptional(p => p.Car);
  }
}
```

。 *HasRequired* *WithOptional* 。 Has / WithRequired / Optional。 `Person`

```
public class PersonEntityTypeConfiguration : EntityTypeConfiguration<Person>
```

```
{
  public PersonEntityTypeConfiguration()
  {
     this.HasOptional(p => p.Car).WithOptional(c => c.Person);
  }
}
```

db



FK`PeopleCar`。 `Car`FK`PersonId` `CarId`。 FK

```
ALTER TABLE [dbo].[Cars]  WITH CHECK ADD  CONSTRAINT [FK_dbo.Cars_dbo.People_CarId] FOREIGN
KEY([CarId])
REFERENCES [dbo].[People] ([PersonId])
```

`CarId``PersonId` `CarId`。 。 FKEF。 。

。 。

`PersonId``Car``CarId``People`。 `PersonId``Car`。 `PersonId``People``PersonId``NULL` SQL Server 2008;RDBMS。

.....

`People``Car`“”。 `CarId` in `Car`PKPKFK。 。 `Car` `PersonId`PK / FK

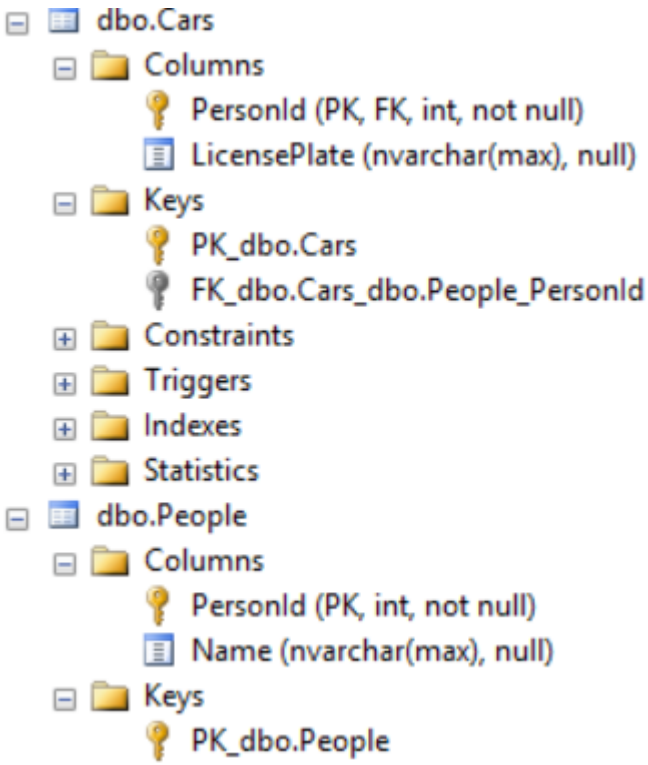
```
public class Person
{
  public int PersonId { get; set; }
```

```
  public string Name { get; set; }
  public virtual Car Car { get; set; }
}

public class Car
{
  public string LicensePlate { get; set; }
  public int PersonId { get; set; }
  public virtual Person Person { get; set; }
}

public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
{
  public CarEntityTypeConfiguration()
  {
     this.HasRequired(c => c.Person).WithOptional(p => p.Car);
     this.HasKey(c => c.PersonId);
  }
}
```

。 。



。 。

。

。 `PeopleCarIdCarId` in `Car` `PersonId` in Car`PersonId` in `People` 。

`PersonId` 。 `PersonId` `People`。 。 `CarId` - 。 - :)。

“”。 。 。 “”“”。 PKFKFK。

```
public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
```

```
{
  public CarEntityTypeConfiguration()
  {
    this.HasRequired(c => c.Person).WithRequiredDependent(p => p.Car);
    this.HasKey(c => c.PersonId);
  }
}
```

:)WithRequiredDependent / PrincipalCarPK。

```
public class PersonEntityTypeConfiguration : EntityTypeConfiguration<Person>
{
  public PersonEntityTypeConfiguration()
  {
    this.HasRequired(p => p.Car).WithRequiredPrincipal(c => c.Person);
  }
}
```

。EF。:)

。

:)FK-s。EF。

```
public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
{
  public CarEntityTypeConfiguration()
  {
    this.HasOptional(c => c.Person).WithOptionalPrincipal(p => p.Car);
    this.HasKey(c => c.PersonId);
  }
}
```

:)

https://riptutorial.com/zh-TW/entity-framework/topic/9412/-

20:

Entity Framework Code First。

Examples

```
public class Person
{
   public int PersonId { get; set; }
   public string Name { get; set; }
}

public class Car
{
   public int CarId { get; set; }
   public string LicensePlate { get; set; }
}

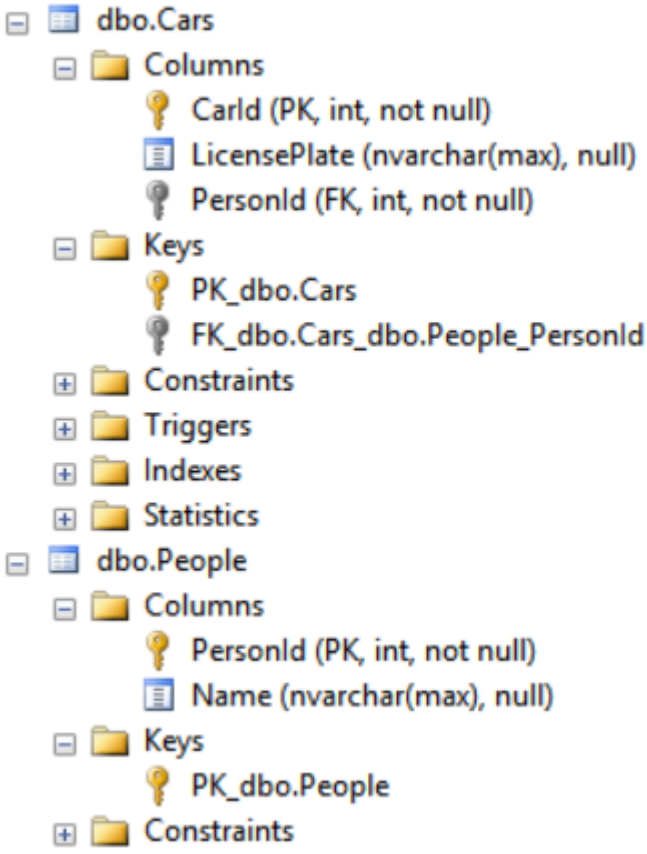
public class MyDemoContext : DbContext
{
   public DbSet<Person> People { get; set; }
   public DbSet<Car> Cars { get; set; }
}
```

。 。

```
public class Person
{
   public int PersonId { get; set; }
   public string Name { get; set; }
   public virtual ICollection<Car> Cars { get; set; } // don't forget to initialize (use
HashSet)
}

public class Car
{
   public int CarId { get; set; }
   public string LicensePlate { get; set; }
   public int PersonId { get; set; }
   public virtual Person Person { get; set; }
}
```

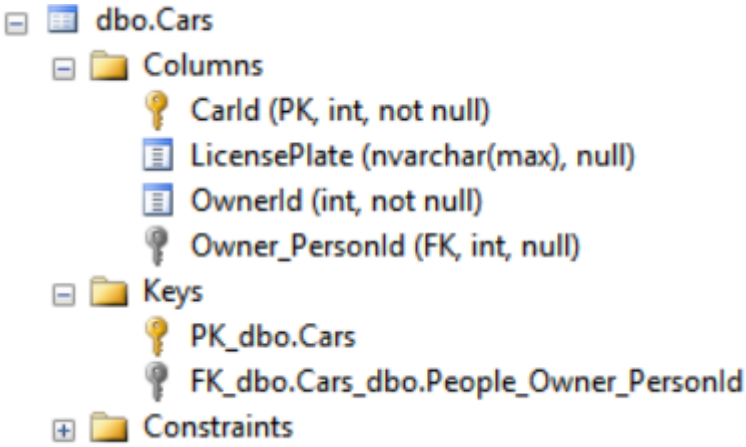
:)。 。



EF◦ ◦ `PersonPersonPersonId`EF`PersonIdPerson` ◦

PERSONIDOWNERID

```
public class Car
{
   public int CarId { get; set; }
   public string LicensePlate { get; set; }
   public int OwnerId { get; set; }
   public virtual Person Owner { get; set; }
}
```

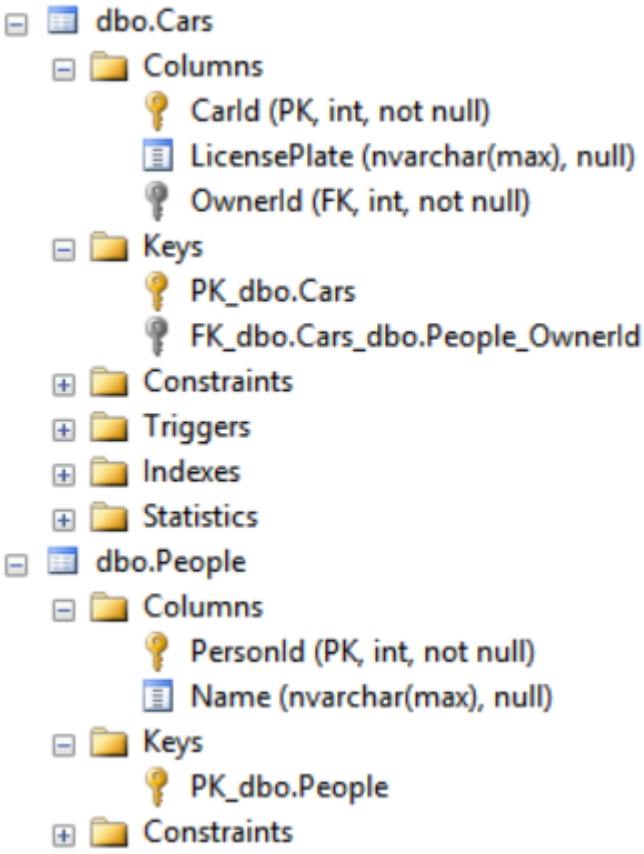


;EF◦ `CarOwnerId`FK◦ `OnModelCreating()`

```
public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
{
```

```
  public CarEntityTypeConfiguration()
  {
     this.HasRequired(c => c.Owner).WithMany(p => p.Cars).HasForeignKey(c => c.OwnerId);
  }
}
```

Car Owner *HasRequired* Owner Cars *WithMany*。 *HasForeignKey*。



Person

```
public class PersonEntityTypeConfiguration : EntityTypeConfiguration<Person>
{
  public PersonEntityTypeConfiguration()
  {
    this.HasMany(p => p.Cars).WithRequired(c => c.Owner).HasForeignKey(c => c.OwnerId);
  }
}
```

"。 PersonCar。

。 。 CarPersonId

```
public class Car
{
    public int CarId { get; set; }
    public string LicensePlate { get; set; }
    public int? PersonId { get; set; }
    public virtual Person Person { get; set; }
}
```

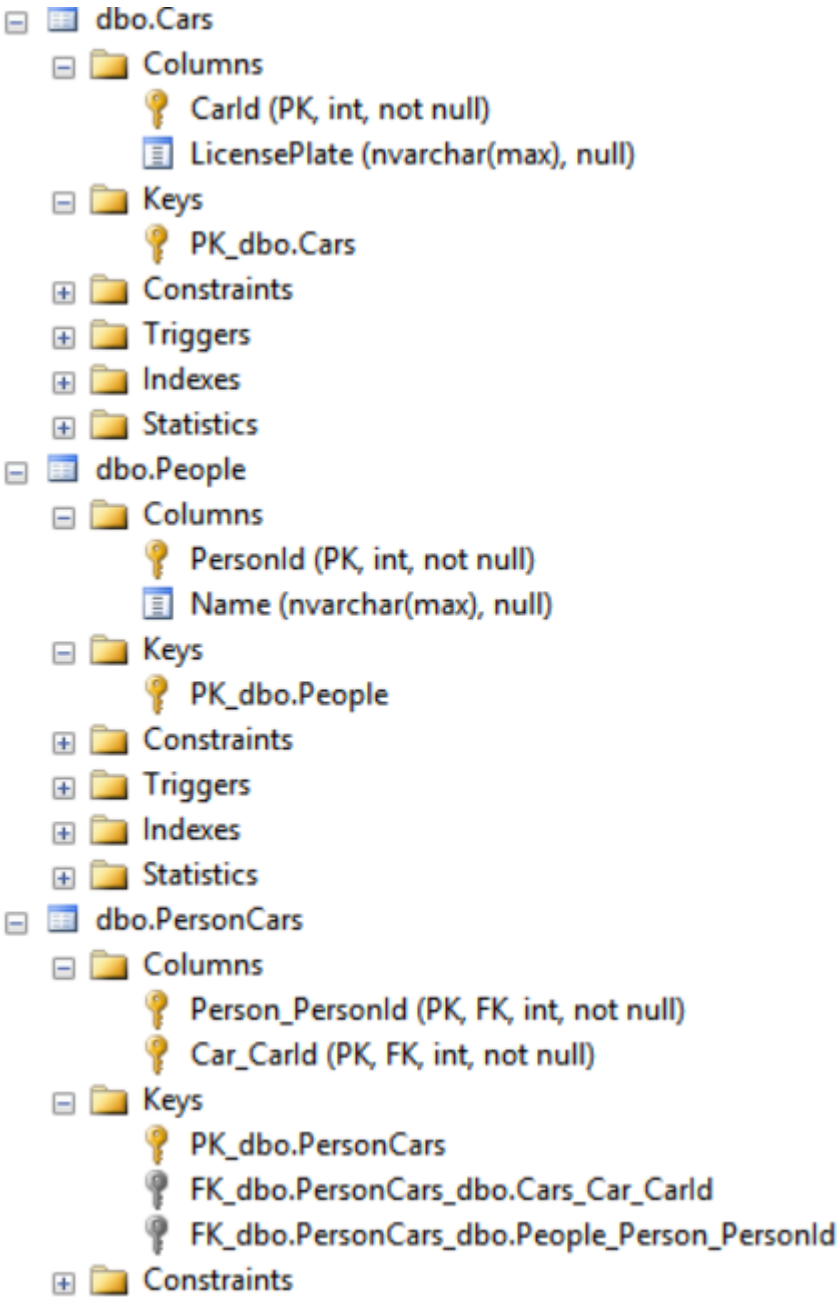
HasOptional WithOptional

```
public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
{
  public CarEntityTypeConfiguration()
  {
     this.HasOptional(c => c.Owner).WithMany(p => p.Cars).HasForeignKey(c => c.OwnerId);
  }
}
```

。。EF。

```
 public class Person
{
   public int PersonId { get; set; }
   public string Name { get; set; }
   public virtual ICollection<Car> Cars { get; set; }
}

public class Car
{
   public int CarId { get; set; }
   public string LicensePlate { get; set; }
   public virtual ICollection<Person> Owners { get; set; }
}
```

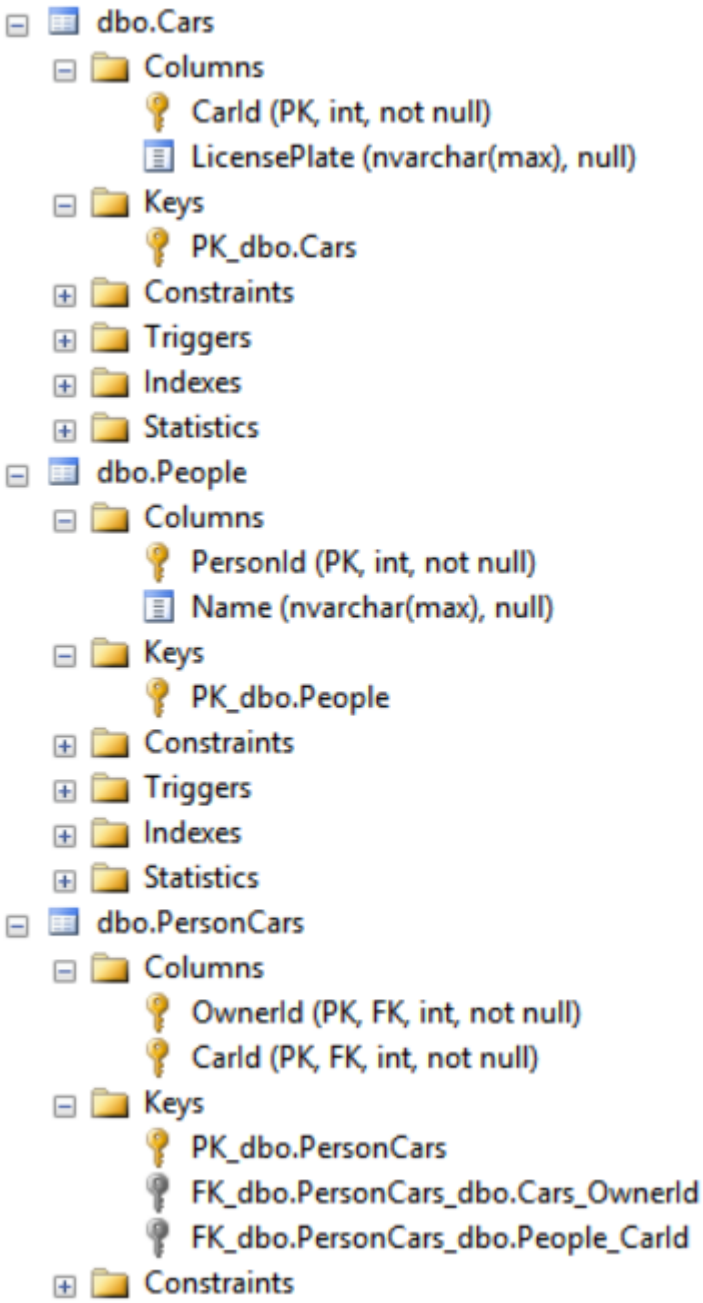


。EF。

。

```
public class CarEntityTypeConfiguration : EntityTypeConfiguration<Car>
{
   public CarEntityTypeConfiguration()
   {
      this.HasMany(c => c.Owners).WithMany(p => p.Cars)
          .Map(m =>
              {
                 m.MapLeftKey("OwnerId");
                 m.MapRightKey("CarId");
                 m.ToTable("PersonCars");
              }
        );
   }
}
```

HasMany *WithMany*。 OwnerId *MapLeftKey* CarId *MapRightKey* PersonCars *ToTable*。



EF。 。

`CarId`。 EF/ - 。

。

```
public class PersonToCar
{
   public int PersonToCarId { get; set; }
   public int CarId { get; set; }
   public virtual Car Car { get; set; }
   public int PersonId { get; set; }
   public virtual Person Person { get; set; }
   public DateTime ValidFrom { get; set; }
}

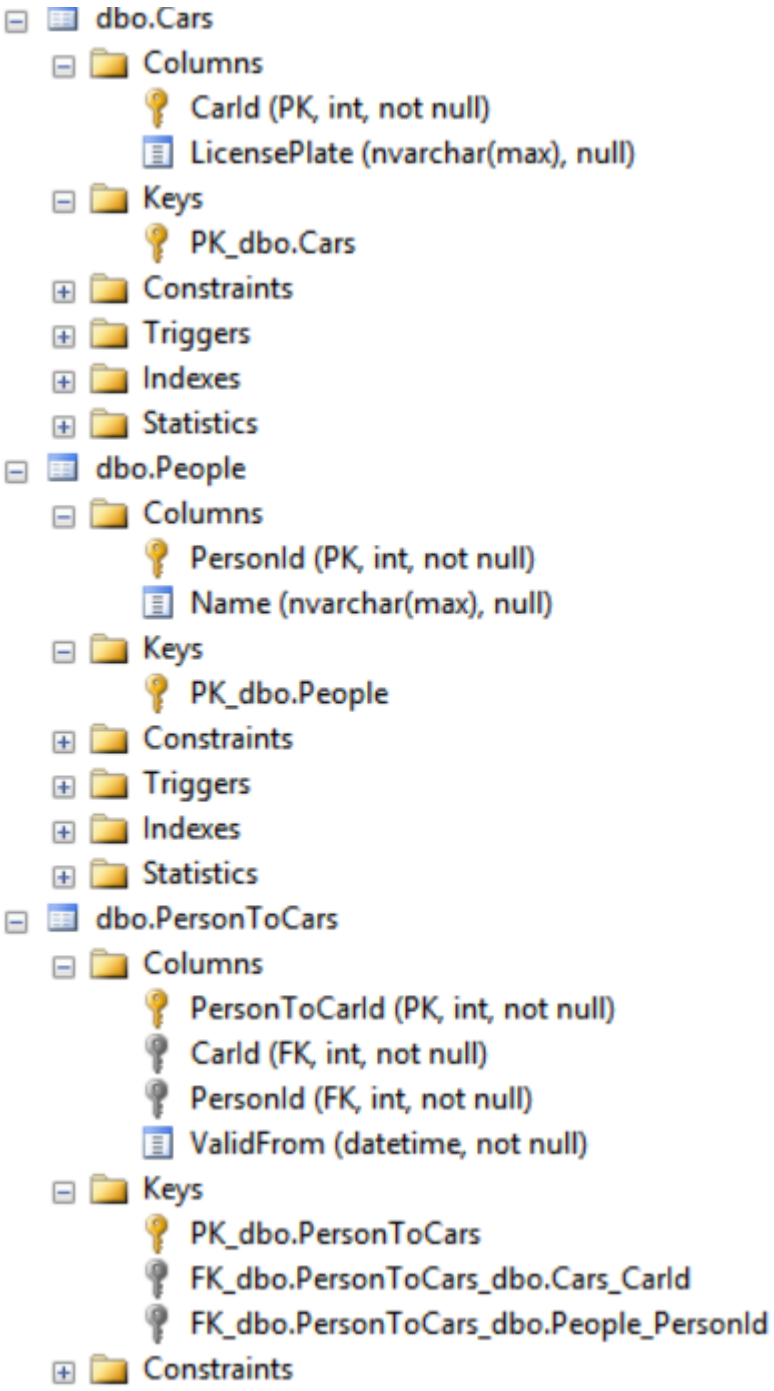
public class Person
{
```

```
  public int PersonId { get; set; }
  public string Name { get; set; }
  public virtual ICollection<PersonToCar> CarOwnerShips { get; set; }
}

public class Car
{
  public int CarId { get; set; }
  public string LicensePlate { get; set; }
  public virtual ICollection<PersonToCar> Ownerships { get; set; }
}

public class MyDemoContext : DbContext
{
  public DbSet<Person> People { get; set; }
  public DbSet<Car> Cars { get; set; }
  public DbSet<PersonToCar> PersonToCars { get; set; }
}
```

◦ ◦



。

https://riptutorial.com/zh-TW/entity-framework/topic/9413/-

21:

Examples

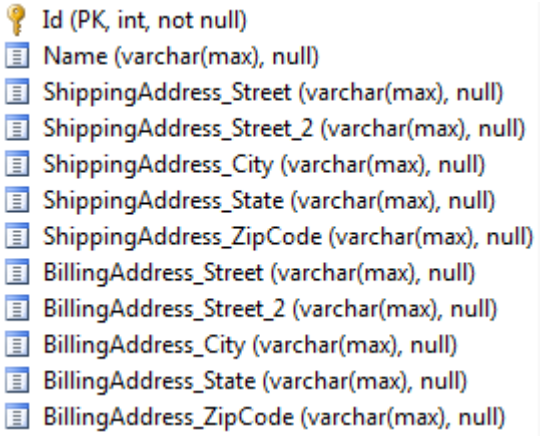
。

```
[ComplexType]
public class Address
{
    public string Street { get; set; }
    public string Street_2 { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string ZipCode { get; set; }
}
```

。 。

```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    ...
    public Address ShippingAddress { get; set; }
    public Address BillingAddress { get; set; }
}
```

。



1n。

https://riptutorial.com/zh-TW/entity-framework/topic/5527/

22:

◦ SaveChanges()◦

Examples

-
- SaveChanges()

- SaveChanges()bookSaveChanges()◦

```
  using (var context = new BookContext())
  {
    var book = context.Books.FirstOrDefault(b => b.BookId == 1);
    book.Rating = 5;
    context.SaveChanges();
  }
```

- read-only
- quicker to execute

```
using (var context = new BookContext())
{
    var books = context.Books.AsNoTracking().ToList();
}
```

EF Core 1.0context instance◦

```
using (var context = new BookContext())
{
    context.ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;

    var books = context.Books.ToList();
}
```

- contains entitytracked by defaulttracked by default

- anonymous type will be trackedBook

  ```
   using (var context = new BookContext())
   {
      var book = context.Books.Select(b => new { Book = b, Authors = b.Authors.Count() });
   }
  ```

- does notentity no trackingno tracking

- anonymous type no instancesentityno instances ◦

  ```
  using (var context = new BookContext())
  {
  ```

```
    var book = context.Books.Select(b => new { Id = b.BookId, PublishedDate = b.Date });
}
```

https://riptutorial.com/zh-TW/entity-framework/topic/6836/

23:

EF。

Examples

```
public class Person
{
   public int PersonId { get; set; }
   public string Name { get; set; }
   public string ZipCode { get; set; }
   public string City { get; set; }
   public string AddressLine { get; set; }
}
```

Person - PersonIdName。PersonId。。。

```
public class MyDemoContext : DbContext
{
  public DbSet<Person> Products { get; set; }

  protected override void OnModelCreating(DbModelBuilder modelBuilder)
  {
      modelBuilder.Entity<Person>().Map(m =>
      {
         m.Properties(t => new { t.PersonId, t.Name });
         m.ToTable("People");
      }).Map(m =>
      {
        m.Properties(t => new { t.PersonId, t.AddressLine, t.City, t.ZipCode });
        m.ToTable("PersonDetails");
      });
  }
}
```

PeoplePersonDetails。PersonPersonIdNamePersonDetailsPersonIdAddressLineCityZipCode。PeoplePersonId。PersonDetailsPersonIdPersonPersonId。

People DbSetEFPersonIds。

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
  modelBuilder.Entity<Person>().Map(m =>
  {
    m.Properties(t => new { t.PersonId });
    m.Property(t => t.Name).HasColumnName("PersonName");
    m.ToTable("People");
  }).Map(m =>
  {
    m.Property(t => t.PersonId).HasColumnName("ProprietorId");
    m.Properties(t => new { t.AddressLine, t.City, t.ZipCode });
    m.ToTable("PersonDetails");
  });
```

```
}
```

PeoplePersonNameNamePersonDetailsProprietorIdPersonId。

。。PersonIdNameAddressLineCityZipCodePersonId。EF

```
public class Person
{
  public int PersonId { get; set; }
  public string Name { get; set; }
  public Address Address { get; set; }
}

public class Address
{
  public string ZipCode { get; set; }
  public string City { get; set; }
  public string AddressLine { get; set; }
  public int PersonId { get; set; }
  public Person Person { get; set; }
}
```

AddressId。。。Address。

```
public class MyDemoContext : DbContext
{
  public DbSet<Person> Products { get; set; }
  public DbSet<Address> Addresses { get; set; }

  protected override void OnModelCreating(DbModelBuilder modelBuilder)
  {
     modelBuilder.Entity<Address>().HasKey(t => t.PersonId);
     modelBuilder.Entity<Person>().HasRequired(t => t.Address)
                                  .WithRequiredPrincipal(t => t.Person);

     modelBuilder.Entity<Person>().Map(m => m.ToTable("People"));
     modelBuilder.Entity<Address>().Map(m => m.ToTable("People"));
  }
}
```

https://riptutorial.com/zh-TW/entity-framework/topic/9362/--

S. No		Contributors
1		Adil Mammadov, Community, DavidG, Eldho, Jacob Linney, Martin4ndersen, Matas Vaitkevicius, Nasreddine, NovaDev, Parth Patel, Stephen Reindl, tmg
2	Code First DataAnnotations	bubi, CptRobby, Daniel A. White, Daniel Lemke, DavidG, Diego, Gert Arnold, Jozef Lačný, Mark Shevchenko, Matas Vaitkevicius , Parth Patel, Piotrek, tmg, Tushar patel
3	EF	Amit Shahani, Anshul Nigam, DavidG, Gert Arnold, Jacob Linney, Kobi, lucavgobbi, Stephen Reindl, tmg, wertzui
4	Postgresql	skj123
5		CptRobby, DavidG, Gert Arnold
6	- API	Adil Mammadov, Daniel Lemke, Jason Tyler, tmg
7		MacakM, Parth Patel, Sivanantham Padikkasu, Stephen Reindl, tmg
8	EntityFramework	lucavgobbi
9	SQLite	Jason Tyler
10		Adil Mammadov, Florian Haider, Gert Arnold, hasan, Joshit, Matas Vaitkevicius, tmg
11	Code First Migrations	CGritton, hasan, Joshit, Mostafa, RamenChef, Stephen Reindl
12	.t4	Matas Vaitkevicius, Tetsuya Yamamoto
13		Balázs Nagy, Jozef Lačný
14		Braiam, Mina Matta
15		Jozef Lačný
16		Matas Vaitkevicius, Tetsuya Yamamoto
17		SOfanatic, Tushar patel
18		Gert Arnold
19		Akos Nagy

20		Akos Nagy
21		CptRobby, Gert Arnold
22		hasan, Sampath, Stephen Reindl, tmg
23		Akos Nagy