public class SchoolContext : DbContext

{

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Server=.\SQLEXPRESS;Database=SchoolDB;Trusted\_Connection=True;");

}

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

}

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

namespace EFCoreTutorials.Models

{

public partial class SchoolDBContext : DbContext

{

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

public virtual DbSet<Standard> Standard { get; set; }

public virtual DbSet<Student> Student { get; set; }

public virtual DbSet<StudentAddress> StudentAddress { get; set; }

public virtual DbSet<StudentCourse> StudentCourse { get; set; }

public virtual DbSet<Teacher> Teacher { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. See http://go.microsoft.com/fwlink/?LinkId=723263 for guidance on storing connection strings.

optionsBuilder.UseSqlServer(@"Server=.\SQLExpress;Database=SchoolDB;Trusted\_Connection=True;");

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Course>(entity =>

{

entity.Property(e => e.CourseName)

.HasMaxLength(50)

.IsUnicode(false);

entity.HasOne(d => d.Teacher)

.WithMany(p => p.Course)

.HasForeignKey(d => d.TeacherId)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_Course\_Teacher");

});

modelBuilder.Entity<Standard>(entity =>

{

entity.Property(e => e.Description)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.StandardName)

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<Student>(entity =>

{

entity.Property(e => e.StudentId).HasColumnName("StudentID");

entity.Property(e => e.FirstName)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.LastName)

.HasMaxLength(50)

.IsUnicode(false);

entity.HasOne(d => d.Standard)

.WithMany(p => p.Student)

.HasForeignKey(d => d.StandardId)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_Student\_Standard");

});

modelBuilder.Entity<StudentAddress>(entity =>

{

entity.HasKey(e => e.StudentId);

entity.Property(e => e.StudentId)

.HasColumnName("StudentID")

.ValueGeneratedNever();

entity.Property(e => e.Address1)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.Address2)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.City)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.State)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.HasOne(d => d.Student)

.WithOne(p => p.StudentAddress)

.HasForeignKey<StudentAddress>(d => d.StudentId)

.HasConstraintName("FK\_StudentAddress\_Student");

});

modelBuilder.Entity<StudentCourse>(entity =>

{

entity.HasKey(e => new { e.StudentId, e.CourseId });

entity.HasOne(d => d.Course)

.WithMany(p => p.StudentCourse)

.HasForeignKey(d => d.CourseId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_StudentCourse\_Course");

entity.HasOne(d => d.Student)

.WithMany(p => p.StudentCourse)

.HasForeignKey(d => d.StudentId)

.HasConstraintName("FK\_StudentCourse\_Student");

});

modelBuilder.Entity<Teacher>(entity =>

{

entity.Property(e => e.StandardId).HasDefaultValueSql("((0))");

entity.Property(e => e.TeacherName)

.HasMaxLength(50)

.IsUnicode(false);

entity.HasOne(d => d.Standard)

.WithMany(p => p.Teacher)

.HasForeignKey(d => d.StandardId)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_Teacher\_Standard");

});

}

}

}

**Note:** EF Core creates entity classes only for tables and not for StoredProcedures or Views.

## DotNet CLI

If you use dotnet command line interface to execute EF Core commands then open command prompt and navigate to the root folder and execute the following dotnet ef dbcontext scaffold command:

> dotnet ef dbcontext scaffold "Server=.\SQLEXPRESS;Database=SchoolDB;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -o Models

Thus, you can create EF Core model for an existing database.

**Note:** Once you have created the model, you must use the Migration commands whenever you change the model to keep the database up to date with the model.

public class SchoolContext : DbContext

{

public SchoolContext()

{

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

}

//entities

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

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

}

DbContext Methods

|  |  |
| --- | --- |
| **Method** | **Usage** |
| Add | Adds a new entity to DbContext with Added state and starts tracking it. This new entity data will be inserted into the database when SaveChanges() is called. |
| AddAsync | Asynchronous method for adding a new entity to DbContext with Added state and starts tracking it. This new entity data will be inserted into the database when SaveChangesAsync() is called. |
| AddRange | Adds a collection of new entities to DbContext with Added state and starts tracking it. This new entity data will be inserted into the database when SaveChanges() is called. |
| AddRangeAsync | Asynchronous method for adding a collection of new entities which will be saved on SaveChangesAsync(). |
| Attach | Attaches a new or existing entity to DbContext with Unchanged state and starts tracking it. |
| AttachRange | Attaches a collection of new or existing entities to DbContext with Unchanged state and starts tracking it. |
| Entry | Gets an EntityEntry for the given entity. The entry provides access to change tracking information and operations for the entity. |
| Find | Finds an entity with the given primary key values. |
| FindAsync | Asynchronous method for finding an entity with the given primary key values. |
| Remove | Sets Deleted state to the specified entity which will delete the data when SaveChanges() is called. |
| RemoveRange | Sets Deleted state to a collection of entities which will delete the data in a single DB round trip when SaveChanges() is called. |
| SaveChanges | Execute INSERT, UPDATE or DELETE command to the database for the entities with Added, Modified or Deleted state. |
| SaveChangesAsync | Asynchronous method of SaveChanges() |
| Set | Creates a DbSet<TEntity> that can be used to query and save instances of TEntity. |
| Update | Attaches disconnected entity with Modified state and start tracking it. The data will be saved when SaveChagnes() is called. |
| UpdateRange | Attaches a collection of disconnected entities with Modified state and start tracking it. The data will be saved when SaveChagnes() is called. |
| OnConfiguring | Override this method to configure the database (and other options) to be used for this context. This method is called for each instance of the context that is created. |
| OnModelCreating | Override this method to further configure the model that was discovered by convention from the entity types exposed in DbSet<TEntity> properties on your derived context. |

DbContext Properties

|  |  |
| --- | --- |
| **Method** | **Usage** |
| ChangeTracker | Provides access to information and operations for entity instances this context is tracking. |
| Database | Provides access to database related information and operations for this context. |
| Model | Returns the metadata about the shape of entities, the relationships between them, and how they map to the database. |