通常通过重写派生DbContext 上的OnModelCreating 方法来访问Code First Fluent API。

在网站运行时才执行

protected override void OnModelCreating(DbModelBuilder modelBuilder) {

}

## 配置主键

要显式将某个属性设置为主键，可使用 HasKey 方法。在以下示例中，使用了 HasKey 方法对 OfficeAssignment 类型配置 InstructorID 主键。

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

## 配置组合主键

以下示例配置要作为Department 类型的组合主键的DepartmentID 和 Name 属性。

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

## 关闭数值主键的标识

以下示例将DepartmentID 属性设置为System.ComponentModel.DataAnnotations.DatabaseGeneratedOption.None，以指示该值不由数据库自动生成。

modelBuilder.Entity<Department>().Property(t =>t.DepartmentID).HasDatabaseGeneratedOption(DatabaseGeneratedOption.None);

## 指定属性的最大长度

modelBuilder.Entity<Department>().Property(t =>t.Name).HasMaxLength(50);

## 将属性配置为必需

modelBuilder.Entity<Department>().Property(t =>t.Name).IsRequired();

## 指定不将属性映射到数据库中的列

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

## 将属性映射到数据库中的特定列

以下示例将Name 属性映射到DepartmentName 数据库列。

modelBuilder.Entity<Department>().Property(t =>t.Name).HasColumnName("DepartmentName");

## 重命名模型中未定义的外键

如果您选择不对CLR 类型定义外键，但希望指定它在数据库中应使用的名称，请编码如下：

modelBuilder.Entity<Course>()

        .HasRequired(c => c.Department)

        .WithMany(t => t.Courses)

        .Map(m => m.MapKey("ChangedDepartmentID"));

## 配置数据库列的数据类型

modelBuilder.Entity<Department>()

        .Property(p => p.Name)

        .HasColumnType("varchar");

## 配置复杂类型的属性

对复杂类型配置标量属性有两种方法。

可以对ComplexTypeConfiguration 调用Property。

modelBuilder.ComplexType<Details>()

        .Property(t => t.Location)

        .HasMaxLength(20);

## 类型映射

## 将类指定为复杂类型

按约定，没有指定主键的类型将被视为复杂类型。在一些情况下，Code First 不会检测复杂类型（例如，如果您有名为“ID”的属性，但不想将它用作主键）。在此类情况下，您将使用 Fluent API 显式指定某类型是复杂类型。

modelBuilder.ComplexType<Details>();

## 指定不将CLR 实体类型映射到数据库中的表

以下示例显示如何排除一个 CLR 类型，使之不映射到数据库中的表。

modelBuilder.Ignore<OnlineCourse>();

## 将CLR 实体类型映射到数据库中的特定表

Department 的所有属性都将映射到名为 t\_ Department 的表中的列。

modelBuilder.Entity<Department>().ToTable("t\_Department");

## 将实体类型的CLR 属性映射到数据库中的多个表（实体拆分）

实体拆分允许一个实体类型的属性分散在多个表中。在以下示例中，Department 实体拆分到两个表中：Department 和DepartmentDetails。实体拆分通过多次调用 Map 方法将一部分属性映射到特定表。

modelBuilder.Entity<Department>()

.Map(m=>

{

  m.Properties(t => new{ t.DepartmentID, t.Name });

  m.ToTable("Department");

})

.Map(m=>

{

  m.Properties(t=> new { t.DepartmentID, t.Administrator,t.StartDate, t.Budget });

  m.ToTable("DepartmentDetails");

 });

## 将多个实体类型映射到数据库中的一个表（表拆分）

以下示例将使用同一个主键的两个实体类型映射到同一个表。

modelBuilder.Entity<OfficeAssignment>()

  .HasKey(t => t.InstructorID);

modelBuilder.Entity<Instructor>()

    .HasRequired(t => t.OfficeAssignment)

    .WithRequiredPrincipal(t =>t.Instructor);

modelBuilder.Entity<Instructor>().ToTable("Instructor");

modelBuilder.Entity<OfficeAssignment>().ToTable("Instructor");