## 配置【必须-可选】关系（1-0..1）

OfficeAssignment的键属性不符合命名约定，所以需要我们显式指定。下面的关系表明，OfficeAssignment的Instructor必须存在，但是Instructor的OfficeAssignment不是必须存在的。

modelBuilder.Entity<OfficeAssignment>()

  .HasKey(t => t.InstructorID);

// Map one-to-zero or one relationship

modelBuilder.Entity<OfficeAssignment>()

  .HasRequired(t => t.Instructor)

  .WithOptional(t => t.OfficeAssignment);

## 配置两端都是必须的关系（1-1）

大多数情况下，EF都能推断哪一个类型是依赖项或者是主体项。然而当关系的两端都是必须的或者都是可选的，那么EF就不能识别依赖项或者是主体项。如果关系两端都是必须的，那么在HasRequired方法后使用WithRequiredPrincipal或者WithRequiredDependent来确定主体。如果关系两端都是可选的，那么在HasRequired方法后使用WithOptionalPrincipal和WithOptionalDependent。

modelBuilder.Entity<OfficeAssignment>()

  .HasKey(t => t.InstructorID);

modelBuilder.Entity<Instructor>()

  .HasRequired(t => t.OfficeAssignment)

  .WithRequiredPrincipal(t => t.Instructor);

## 配置多对多关系

下面的代码配置了一个多对多关系，CodeFirst会使用命名约定来创建连接表，命名约定会使用Course\_CourseID 和 Instructor\_InstructorID作为连接表的列。

modelBuilder.Entity<Course>()

        .HasMany(t => t.Instructors)

        .WithMany(t => t.Courses);

如果想指定连接表的表名和列名，需要使用Map方法，如下：

modelBuilder.Entity<Course>()

        .HasMany(t => t.Instructors)

        .WithMany(t => t.Courses)

        .Map(m =>

        {

          m.ToTable("CourseInstructor");

          m.MapLeftKey("CourseID");

          m.MapRightKey("InstructorID");

        });

## 配置单向导航属

所谓单向导航属性指的是只在关系的一端定义了导航属性。按照约定，CodeFirst将单向导航理解为一对多关系，如果需要一对一的单向导航属性，需要使用如下方法：

modelBuilder.Entity<OfficeAssignment>()

  .HasKey(t => t.InstructorID);

modelBuilder.Entity<Instructor>()

  .HasRequired(t => t.OfficeAssignment)

  .WithRequiredPrincipal();

## 启用级联删除

使用WillCascadeOnDelete方法来配置关系是否允许级联删除。如果外键是不可空的，CodeFirst默认会设置级联删除；否则，不会设置级联删除，当主体被删除后，外键将会被置空。

可以使用如下代码移除此约定：

modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();

下面的代码片段配置为外键不能为空，而且禁用了级联删除。

modelBuilder.Entity<Course>()

  .HasRequired(t => t.Department)

  .WithMany(t => t.Courses)

  .HasForeignKey(d => d.DepartmentID)

  .WillCascadeOnDelete(false);

## 配置组合外键

下面的代码配置了组合外键

modelBuilder.Entity<Department>()

  .HasKey(d => new{ d.DepartmentID, d.Name });

// Composite foreign key

modelBuilder.Entity<Course>()

    .HasRequired(c => c.Department)

    .WithMany(d => d.Courses)

    .HasForeignKey(d => new { d.DepartmentID, d.DepartmentName });

## 

## 配置不符合命名约定的外键属性

SomeDepartmentID属性不符合外键命名约定，需要使用如下方法将其设置为外键属性：

modelBuilder.Entity<Course>()

         .HasRequired(c => c.Department)

         .WithMany(d => d.Courses)

         .HasForeignKey(c =>c.SomeDepartmentID);