# Ver video [L62-AI FLUENTE, Tipos de realacion-clases.mp4](L62-AI%20FLUENTE,%20Tipos%20de%20realacion-clases.mp4)

# Definir tipo de relación

# No context

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.ModelConfiguration.Conventions;

using System.Linq;

using System.Web;

namespace EfCodeFirst.Models

{

public class Class1:DbContext

{

public DbSet<Persona> Persona { get; set; }

public DbSet<Direccion> Direccion { get; set; }

public DbSet<TarjetaDeCredito> TarjetaDeCredito { get; set; }

public DbSet<Curso> Cursos { get; set; }

public DbSet<Persona\_Cursos> Persona\_Curso { get; set; }//Para o ejemplo 3 esta podemola eliminar se queremos.

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();//Elimina que as tablas se creen co nome plural das clases

## //TIPOS DE RELACIOS:

//NOTA: Nas clases hay que definiros campos navegacionales (claves secundarias)

### //1) Toda Toda Direccion/Tarjeta se relaciona con unha persoa, nn sempre unha persoa ten unha direccion/tarjeta

modelBuilder.Entity<Direccion>().HasRequired(x => x.Persona).WithOptional(t => t.Direccion);

modelBuilder.Entity<TarjetaDeCredito>().HasRequired(x => x.Persona);

### //2:Toda direccion se relaciona con unha persoa,todapersoa ten unha direccion

modelBuilder.Entity<Direccion>().HasRequired(x => x.Persona).WithRequiredPrincipal(t => t.Direccion);

### /\*3:Relacion N:N. Todo curso se relaciona con varias personas, toda persona se relaciona con varios cursos.

Para unha relacionN:N se crea unha tabla intermedia

podemolo facer de duas formas:\*/

//A) EF pon o nome definido a tabla e as columnas

modelBuilder.Entity<Curso>().HasMany(x => x.Personas).WithMany(x => x.Cursos);

//B) Personalizando a tabla

modelBuilder.Entity<Curso>().HasMany(t => t.Personas).WithMany(x => x.Cursos)

.Map(m =>

{

m.ToTable("Persona\_Curso");

m.MapLeftKey("IdCurso");

m.MapRightKey("IdPersona");

}

);

modelBuilder.Entity<Persona\_Curso>().HasKey(x => new { x.PesonaId, x.CursoId });

//C) Creando unha clase Persona\_Cursos , asi podemos agregar mais campos

modelBuilder.Entity<Persona\_Curso>().HasKey(x => new { x.PesonaId, x.CursoId });

base.OnModelCreating(modelBuilder);

}

}

}