03

Idioma falado e original de um filme

Transcrição

Iremos criar na classe `Filme` as propriedades que representam `IdiomaFalado` e `IdiomaOriginal`.

```
namespace Alura.Filmes.App.Negocio
{
    public class Filme
    {
        public int Id { get; set; }
        public string Titulo { get; set; }
        public string Descricao { get; set; }
        public string AnoLancamento { get; set; }
        public string Duracao { get; set; }
        public IList<FilmeAtor> Atores { get; set;}
        public Idioma IdiomaFalado { get; set; }
        public Idioma IdiomaOriginal { get; set; }


        public Filme()
        {
            Atores = new List<FilmeAtor>()
        }

        public override string ToString()
        {
            return $"Filme ({Id}): {Titulo} - {AnoLancamento}";
        }
    }
}
```

As duas propriedades que criamos são de **navegação**, e como cada uma está apontando apenas para uma instância de idioma, são chamadas de **propriedades de navegação de referência**.

Normalmente, as propriedades de navegação criadas em uma ponta do relacionamento são o suficiente para o Entity estabelecer as conexões necessárias. No caso do nosso projeto há um problema, pois temos duas propriedades de navegação em uma classe (`Filme`), apontando para uma classe em comum (`Idioma`). Nestes casos o Entity não realiza as relações de forma automática. Portanto, teremos de assumir a configuração dos dois elementos.

Iremos em `FilmeConfiguration`. Na área "Pesquisador de Objetos" abriremos a tabela `film` para analisarmos quais são as colunas que representam o `IdiomaFalado` e `IdiomaOriginal`. Temos duas colunas: uma chamada `language_id` e `original_language_id`, ambas do tipo `tinyint` e chaves estrangeiras para a tabela `Idioma`.

Vale lembrar que as propriedades que criamos (`IdiomaFalado` e `IdiomaOriginal`) na classe `Filme` não representam uma chave estrangeira.

Poderíamos criar uma propriedade do tipo `byte` denominada `IdiomaId` na classe `filme`, mas isso de nada adiantaria, pois como temos duas propriedades de navegação apontando para uma mesma classe, o Entity não estabelecerá as relações. Outro motivo para não criarmos a propriedade é sujarmos a classe de negócio com informações que são específicas do banco de dados.

Em `FilmesConfiguration`, criaremos duas shadow properties do tipo `byte`, uma chamada `language_id` para representar a chave estrangeira para `IdiomaFalado`, e outra `original_language_id` para `IdiomaOriginal`.

```
namespace Alura.Filmes.App.Dados
{
    public class FilmeConfiguration : IEntityTypeConfiguration
    {
        public void Configure(EntityTypeBuilder<Filme>builder)
        {

            modelBuilder.Entity<Filme>()
                .ToTable("film")

            modelBuilder.Entity<Filme>()
                .Property(f => f.id)
                .HasColumnName("film_id");

            modelBuilder.Entity<Filme>()
                .Property(f => f.Titulo)
                .HasColumnName("title")
                .HasColumnType("varchar(255)")
                .IsRequired();

            modelBuilder.Entity<Filme>()
                .Property(f => f.Descricao)
                .HasColumnName("description")
                .HasColumnType("text");

            modelBuilder.Entity<Filme>()
                .Property(f => f.AnoLancamento)
                .HasColoumnName("release_year")
                .HasColumnTypoe("varchar(4)");

            modelBuilder.Entity<Filme>()
                .Property(f => f.Duracao)
                .HasColumnName("length")

            modelBuilder.Entity<Filme>()
                .Property<DateTime>("last_update")
                .HasColumnType("datetime")
                .IsRequired();

            builder.Property<byte>("language_id");
            builder.Property<byte>("original_language_id");

        }
    }
}
```

Feito isso, podemos configurar os relacionamentos. É importante salientar que a classe `Idioma` não apresenta nenhuma propriedade de coleção para `Filmes`. Para efetivar a nossa configuração precisaremos criar propriedades de navegação nas duas pontas do relacionamento.

Abriremos a classe `Idioma` e adicionaremos duas propriedades de navegação para `Filme`, uma para `FilmesFalados` e outra para `FilmesOriginais`. Criaremos também um construtor para inicializar as duas listas referentes a `FilmesFalados` e

`FilmesOriginais` .

```
namespace Alura.Filmes.App.Negocio
{
    public clas Idioma
    {
        public byte Id { get; set; }
        public string Nome { get; set }
        public IList<Filme> FilmesFalados { get; set; }
        public IList<Filme> FilmesOriginais { get; set; }

        public Idioma()
        {
            FilmesFalados = new List<Filme>();
            FilmesOriginais = new List<Filme>();
        }

        public override string ToString()
        {
            return $"Idioma ({Id}): {Nome}";
        }
    }
}
```

Iremos prosseguir com a nossa configuração de relaciomaneto. Um filme possui um idioma (`HasOne`), e o um mesmo idioma é falando em vários filmes (`WithMany`). Declareremos também a chave estrangeira representada pela shadow property `language_id` .

```
namespace Alura.Filmes.App.Dados
{
    public class FilmeConfiguration : IEntityTypeConfiguration
    {
        public void Configure(EntityTypeBuilder<Filme>builder)
        {

            modelBuilder.Entity<Filme>()
                .ToTable("film")

            modelBuilder.Entity<Filme>()
                .Property(f => f.id)
                .HasColumnName("film_id");

            modelBuilder.Entity<Filme>()
                .Property(f => f.Titulo)
                .HasColumnName("title")
                .HasColumnType("varchar(255)")
                .IsRequired();

            modelBuilder.Entity<Filme>()
                .Property(f => f.Descricao)
                .HasColumnName("description")
                .HasColumnType("text");

            modelBuilder.Entity<Filme>()
                .Property(f => f.AnoLancamento)
```

```
                .HasColoumnName("release_year")
                .HasColumnTypoe("varchar(4)");

            modelBuilder.Entity<Filme>()
                .Property(f => f.Duracao)
                .HasColumnName("length")

            modelBuilder.Entity<Filme>()
                .Property<DateTime>("last_update")
                .HasColumnType("datetime")
                .IsRequired();

            builder.Property<byte>("language_id");
            builder.Property<byte>("original_language_id");

            builder
                .HasOne(f => f.IdiomaFalado);
                .WithMany(i => i.FilmesFalados)
                .HasForeignKey("language_id");

        }
    }
}
```

O próximo relacionamento que iremos configurar é representado no `Filme` pela propriedade `IdiomaOriginal`, e na classe `Idioma` pela propriedade `FilmesOriginais`. A chave estrangeira é `original_language_id`.

```
namespace Alura.Filmes.App.Dados
{
    public class FilmeConfiguration : IEntityTypeConfiguration
    {
        public void Configure(EntityTypeBuilder<Filme>builder)
        {

            modelBuilder.Entity<Filme>()
                .ToTable("film")

            modelBuilder.Entity<Filme>()
                .Property(f => f.id)
                .HasColumnName("film_id");

            modelBuilder.Entity<Filme>()
                .Property(f => f.Titulo)
                .HasColumnName("title")
                .HasColumnType("varchar(255)")
                .IsRequired();

            modelBuilder.Entity<Filme>()
                .Property(f => f.Descricao)
                .HasColumnName("description")
                .HasColumnType("text");

            modelBuilder.Entity<Filme>()
                .Property(f => f.AnoLancamento)
                .HasColoumnName("release_year")
                .HasColumnTypoe("varchar(4)");
```

```
            modelBuilder.Entity<Filme>()
                .Property(f => f.Duracao)
                .HasColumnName("length")

            modelBuilder.Entity<Filme>()
                .Property<DateTime>("last_update")
                .HasColumnType("datetime")
                .IsRequired();

            builder.Property<byte>("language_id");
            builder.Property<byte>("original_language_id");

            builder
                .HasOne(f => f.IdiomaFalado);
                .WithMany(i => i.FilmesFalados)
                .HasForeignKey("language_id");

            builder
                .HasOne(f => f.IdiomaOriginal)
                .WithMany(i => i.FilmesOriginais)
                .HasForeignKey("original_language_id");

        }
    }
}
```

O mapeamento está finalizando.