

Configurando tipo e tamanho de colunas

Transcrição

Verificamos que o Entity possui uma regra implícita para configurar o tipo e o tamanho da tabela no banco de dados. Essa convenção, utiliza o tipo da propriedade da classe entidade. Ao abrirmos a classe `Ator`, veremos que para a propriedade `Id` do tipo `int`, o Entity irá inferir as características da coluna a partir dessa informação. O mesmo corre para a `string`.

```
using System.ComponentModel.DataAnnotations.Schema;

namespace Alura.Filmes.App.Negocio
{
    [Table("actor")]
    public class Ator
    {
        [Column("actor_id")]
        public int Id { get; set; }
        [Column("first_name")]
        public string PrimeiroNome { get; set; }
        [Column("last_name")]
        public string UltimoNome { get; set; }

        public override string ToString()
        {
            return $"Ator ({Id}): {PrimeiroNome} {UltimoNome}";
        }
    }
}
```

Precisaremos configurar a classe de negócio para quebrarmos a convenção do Entity com relação às tabelas. A propriedade `Id` está correta, então não iremos nos preocupar com ela. Trabalharemos com as propriedades `PrimeiroNome` e `UltimoNome`. Colocaremos um argumento no construtor da anotação `Column`, denominado `TypeName`. No argumento, especificaremos o `varchar(45)`

```
using System.ComponentModel.DataAnnotations.Schema;

namespace Alura.Filmes.App.Negocio
{
    [Table("actor")]
    public class Ator
    {
        [Column("actor_id")]
        public int Id { get; set; }

        [Column("first_name", TypeName = "varchar(45)")]
        public string PrimeiroNome { get; set; }

        [Column("last_name", TypeName = "varchar(45)")]
        public string UltimoNome { get; set; }

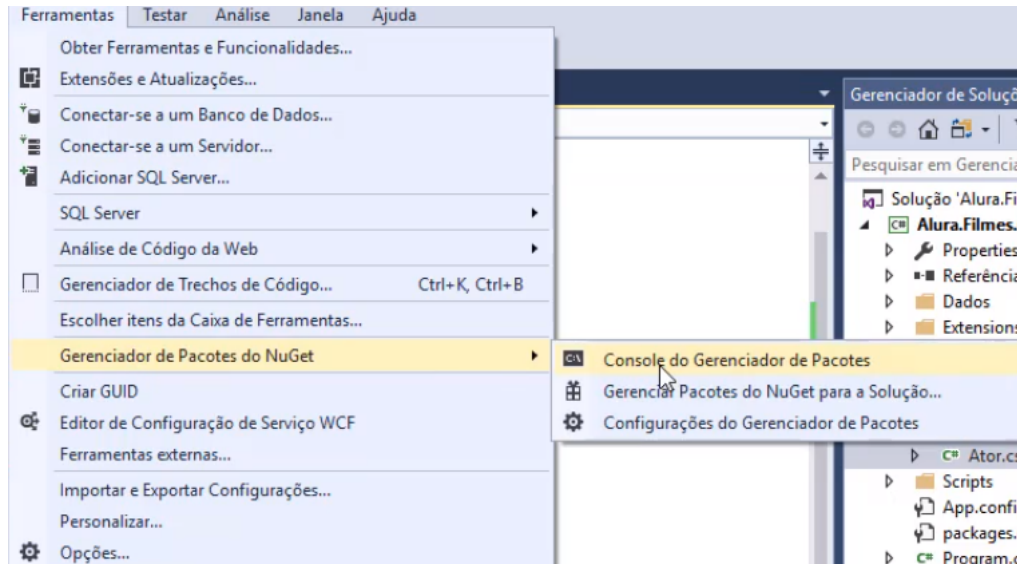
        public override string ToString()
        {
            return $"Ator ({Id}): {PrimeiroNome} {UltimoNome}";
        }
    }
}
```

```

    {
        return $"Ator ({Id}): {PrimeiroNome} {UltimoNome}";
    }
}
}

```

Iremos gerar novamente o script para testar se o Entity está lendo as configurações que fizemos. No cabeçalho, selecionamos "Ferramentas > Gerenciador de Pacotes do NuGet > Console do Gerenciador de Pacotes".



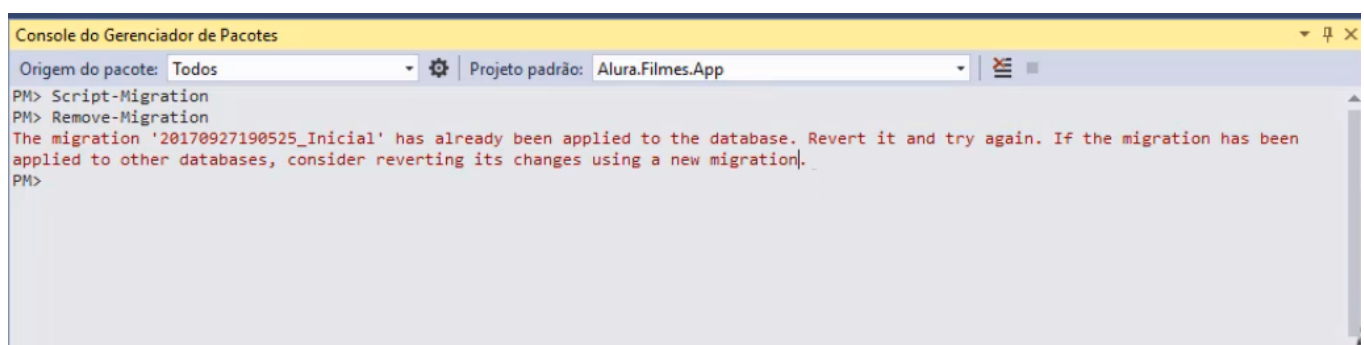
No console, acionamos o `Script-Migration` para analisarmos qual será o formato do script que o Entity irá gerar.

```

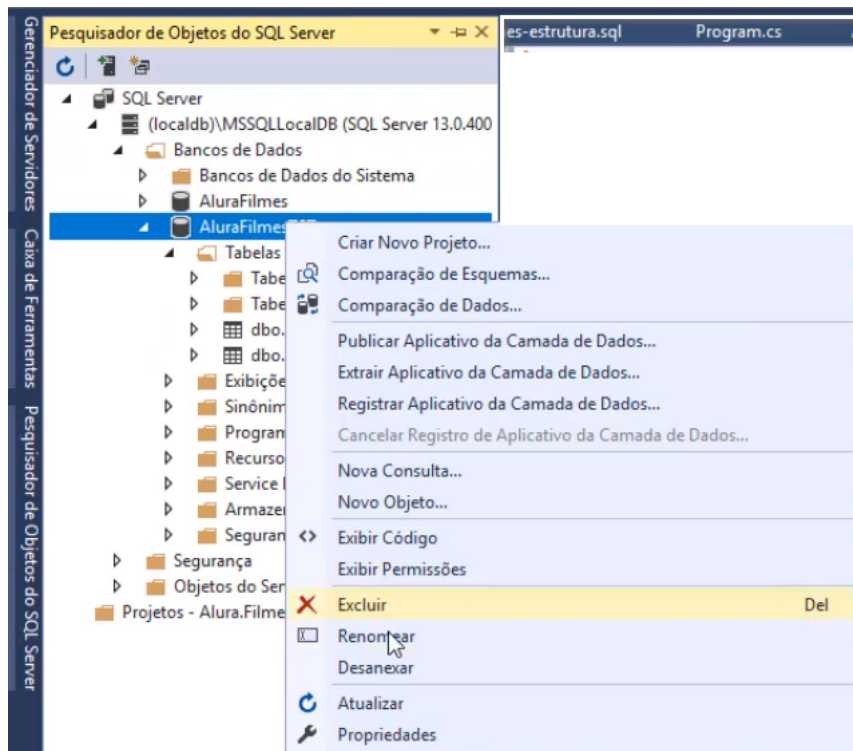
CREATE TABLE [actor] (
    [actor_id] int NOT NULL IDENTITY
    [first_name] nvarchar(max) NULL,
    [last_name] nvarchar(max) NULL,
    CONSTRAINT [PK_actor] PRIMARY KEY ([actor_id])
);

```

O *software* não realizou nenhuma modificação no script. Isso ocorreu, porque pulamos uma etapa do processo. Primeiro, devermos acionar o `Add-Migration`, que por sua vez, cria uma classe que representa qual será a versão a ser utilizada. A classe `migration` já determinou qual será o tipo de a ser utilizado (`nvarchar(max)`), e o `Script_Migration` olhará para essa classe e irá elaborar o script. Para gerarmos um novo script - dessa vez correto - teremos de nos desfazer da migração antiga e gerar uma nova. O problema é que a migração atual já foi aplicada ao banco de dados, portanto, ao tentarmos acionar o comando `Remove-Migration` do console, será exibida uma mensagem de erro. Esse dado foi adquirido através da consulta à tabela `Migration History`.



Iremos apagar o banco e iniciar uma nova migração. Na área "Pesquisador de Objetos", clicamos com o botão direito no banco de dados `AluraFilmesTST` e selecionamos "Excluir".



Após excluirmos o banco de dados, faremos um novo com o mesmo nome. Na área "Pesquisador de Objetos", selecionaremos o "SQL Server > (localdb) > Banco de dados", clicamos com o botão direito em "Banco de dados" e acionamos "Adicionar Novo Banco de Dados". Teremos, então, um novo banco vazio chamado `AluraFilmesTST`.

No console, escreveremos `Add-Migration Inicial` e depois `Script-Migration`. Depois, iremos conferir o resultado no script.

```
CREATE TABLE [actor] (
    [actor_id] int NOT NULL IDENTITY
    [first_name] nvarchar(45) NULL,
    [last_name] nvarchar(45) NULL,
    CONSTRAINT [PK_actor] PRIMARY KEY ([actor_id])
);
```

Iremos compara-lo com o nosso script inicial.

```
CREATE TABLE actor (
    actor_id int NOT NULL IDENTITY ,
    first_name VARCHAR(45) NOT NULL,
    last_name VARCHAR(45) NOT NULL,
    last_updante DATETIME NOT NULL,
    CONSTRAINT pk_actor PRIMARY KEY NONCLUSTERED (actor_id)
)
```

Houve modificações no script gerado, mas ele ainda não está correto. Reparem que a `first_name` está assinalado como `NULL` no scrip gerado, e no inicial estava marcado como `NOT NULL`. Teremos de corrigir a nulidade da coluna.

