04

Definindo valor padrão para colunas

Transcrição

Incluímos o nome do ator `Tom Hanks` ao nosso banco de dados e definimos o valor da propriedade `last_updtate` para que essa inclusão de informação operasse corretamente. O problema é que existem muitos a atores a serem adicionados ao banco de dados, e seguir esse procedimento todas as vezes seria exaustivo. A coluna `last_update` está configurada como `not null`, portanto, sempre teremos de determinar um valor para ela. O ideal é buscarmos uma maneira de definir um valor padrão para a coluna. Lembrem-se que a função da coluna `last_update` é guardar data e hora da última vez em que foi realizada alguma modificação no registro de atores. De forma geral, definimos o valor através de *triggers*, ou seja, gatilhos no banco de dados que são disparados a partir de algum evento. Na tabela `ator`, estamos interessados nos eventos de `insert` e `update`.

No que diz respeito à inclusão, com Entity conseguimos definir um valor padrão para determinada coluna. A documentação do Entity (https://docs.microsof t.com/en-us/ef/core/modeling/relational/default-values) para valores padrão informa que não existe nenhuma convenção para configurar valor padrão e que não podemos utilizar anotações para determina-lo. Só resta, portanto, o que a documentação denominaa **Fluent API**. Trata-se do código escrito no método `OnModelCreating()`.

Observemos o código da classe `Program`:

```
namespace Alura.Fulmes.App
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var contexto = new AluraFilmesContexto())
            {
                contexto.LogSQLToConsole();

                var ator = new Ator();
                ator.PrimeiroNome = "Tom";
                ator.UltimoNome = "Hanks";
                //contexto.Entry(ator).Property("last_update").CurrentValue = DateTime.Now;

                contexto.Atores.Add(ator);

                contexto.SaveChanges();

            }
        }
    }
}
```

Nosso objetivo é não precisar sempre definir o valor da propriedade `last_update` - parte comentada do código acima - quando formos inserir novas informações ao banco de dados. Se tentarmos rodar o programa sem este trecho, teremos uma mensagem de erro. Iremos configurar um valor padrão para a propriedade. Na área "Gerenciador de Soluções" localizada ao lado direito da tela, selecionamos "dados > AluraFilmesContexto.cs". Acessaremos, assim, a nossa classe de contexTo.

No método `OnModelCrating()`, na parte do código que diz respeito a propriedade a shadow property `last_update`, adicIonaremos mais um método denominado `HasDefaulValueSql()`, e apontaremos que é o SqL que determinará o valor. Para isso, devemos inserir um SqL válido para o banco de dados - no nosso caso, SqLServer - , e chamaremos uma função denominada `GetDate`. Essa função executa a mesma ação que o `DateTime` no **C#**, ou seja, registrará data e hora da última modificação no banco de dados.

```
namespace Alura.Filmes.App.Dados
{
    public class AluraFilmesContexto : DbContext
    {
        public DbSet<Ator> Atores { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=(localdb)mssqllocaldb;Database=AluraFilmes;Trusted_(
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Ator>()
                .ToTable("actor");

            modelBuilder.Entity<Ator>()
                .Property(a => a.Id)
                .HasColumnName("actor_id");

            modelBuilder.Entity<Ator>()
                .Property(a => a.PrimieroNome)
                .HasCoumnName("first_name")
                .HasColumnType("varchar(45)")
                .IsRequired();

            modelBuilder.Entity<Ator>()
                .Property(a => a.UltimoNome)
                .HasColumnType("varchar(45)")
                .IsRequired();

             modelBuilder.Entity<Ator>()
                .Property<DateTime>("last_update");
                .HasColumnType ("datetime")
                .HasDefaultValueSqL("getdate()")
                .IsRequired();
        }
    }
}
```

Antes de executarmos o programa devemos atualizar o modelo para que as configurações na classe contexto entrem em vigor. Abriremos o console e daremos o comando `Add-Migration` para adicionar uma nova migração - não conseguimos excluir a migração anterior, pois ela já está no banco e dados -. Para a coluna `DateTime`, a migração terá um valor `defaultValueSql`.

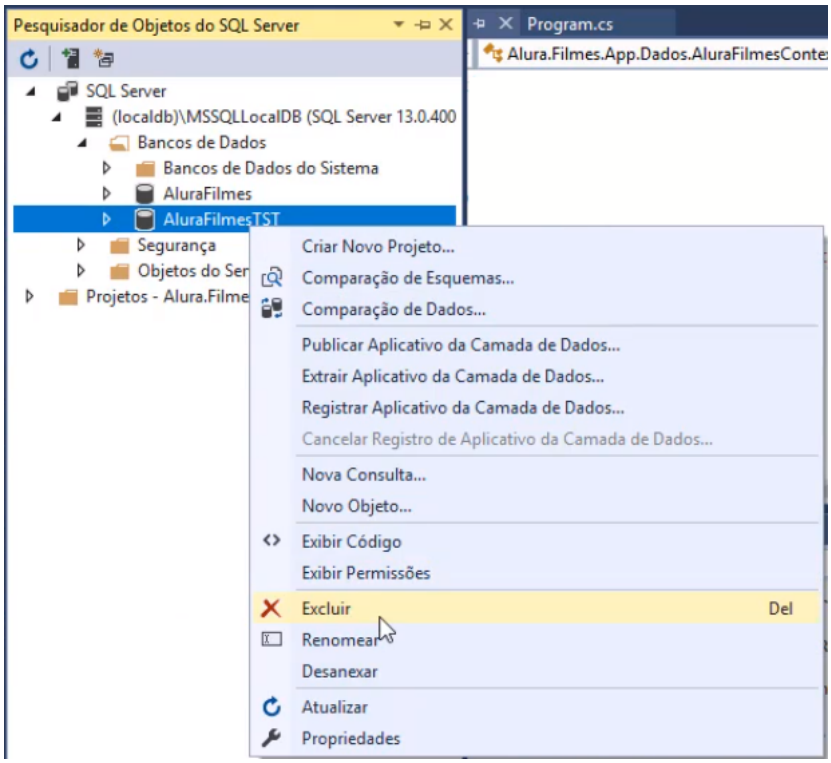
```
public partial class Inicial : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
```

```
        {
            migrationBuilder.CrateTable(
                name: "actor"
                columns: table => new
                {
                    actor_id.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationsStrate
                    first_name = table.Column<string>(type: "varchar(45)", nullable: false),
                    last_name = table.Column<string>(type:"varchar(45)"), nullable: false),
                    last_update = table.Column<DateTime>(type: "datetime", nullable: false, defaultValue
                },
                constratints: table =>


    <!-- ... -->
```

Acionaremos o comando `Update-Database` no console para atualizar o banco de dados. Ao realizarmos essa ação, perceberemos uma mensagem de erro: "There is already an object named 'actor' in the database". Esse erro ocorreu porque estamos apontando para o banco de dados `AluraFilmes`, enquanto deveríamos apontar para `AluraFilmesTST`. Devemos ficar atentos para o banco de dados que está sendo referenciado.

Como não conseguimos excluir a antiga migração, teremos de apagar o banco de dados `AluraFilmesTST` e criarmos outro. Na área `Pesquisador de Objetos`, localizada ao lado esquerdo da tela, selecionaremos o banco de dados `AluraFilmesTST`, clicamos com o botão direito e escolhemos a opção "Excluir".



Para criarmos um novo banco de dados, ainda na área "Pesquisador de Objetos", clicamos com o botão direito sob a pasta "Bancos de Dados" e selecionamos a opção "Adicionar Novo Banco de Dados". O nome do banco será `AluraFilmesTST`. No console acionaremos o comando `Update-Database`, e teremos novamente as tabelas vazias no banco de dados. Rodaremos o programa pressionando o atalho "Ctrl + F5", e observaremos que tudo ocorreu sem erros. Aprendemos, portanto, que conseguimos configurar valores padrão para nossas colunas, sendo elas shadow properties ou não.

Portanto, conseguimos pelo Entity definir um valor padrão para quando for incluído algum registro. Ao ser necessário mudar o valor do `last_update`, só podemos realizar através do *trigger* de update no banco de dados.

Há a documentação (https://docs.microsoft.com/en-us/ef/core/modeling/generated-properties) própria do Entity Framework Core, em que existe um tópico explicativo - **Value generated on add or update** - sobre valores gerados em inclusão ou atualização. Neste tópico, explica-se que para a propriedade do tipo `DateTime`, é necessário configurar uma maneira para os valores serem gerados. Uma das maneiras de realizar isso é configurar o *default value* do `GETDATE`, como fizemos. Mas é necessário gerar um *trigger* para definir valores durante os updates.