03

O processo de migração

Transcrição

Temos o banco de dados com a tabela **Produtos**, que contém as colunas **Id** como chave primária, **Nome**, **Categoria** e **Preco**. A classe `Produto` tem as propriedades `Id` , `Nome` , `Categoria` , `PrecoUnitario` e `Unidade` . A classe não está sincronizada com a tabela, por isso usaremos o pacote de ***Migrations*** (Migrações) que instalamos.

As migrações serão feitas no Entity e para isso usaremos comandos. Com o **Console de Gerenciador de Pacotes** aberto, digitaremos o comando para abrir o manual do EntityFramework, assim poderemos ver todos os comandos disponíveis. No console digite:

```
Get-Help EntityFramework
```

Ao executar o comando, é apresentado no Console o símbolo, e descrição e a lista de comandos disponíveis.

- Add-Migration
- Remove-Migration
- Update-Database
- Script-Migration
- Drop-Database
- Scaffold-DbContext

A migração é feita em dois passos. O primeiro passo é executarmos o comando `Add-Migration` .

Já o segundo passo pode ser feitas de duas maneiras diferentes, sendo a primeira gerando um *script* de linguagem DDL com o comando `Script-Migration` . Esse cenário é mais utilizado quando existe uma equipe de banco de dados separada da equipe de desenvolvimento.

A outra maneira é usarmos o comando `Update-Database` , onde o Entity pega a nova versão que foi registrada e executa diretamente no banco de dados. Vamos utilizar essa segunda forma.

Começaremos executando o comando para adicionar a migração, passando o nome para ela. Como estamos adicionando uma nova coluna `Unidade` , daremos o nome da migração de **Unidade**. O comando ficará:

```
Add-Migration Unidade
```

Não obtivemos nenhum resultado relevante no Console, mas olhando o projeto podemos ver que uma pasta chamada ***Migrations*** foi criada, contendo duas classes.

A classe que usaremos possui o nome do **arquivo** com um ***Timestamp***, que é a data e hora que o arquivo foi gerado. O nome do arquivo ainda tem a separação por *underscore* e escrito **Unidade**.

Abrindo o arquivo que possui o *Timestamp*, vemos que é a classe `Unidade` . A classe `Unidade` herda de `Migration` , que nos fornece uma API que fará essa sincronização. Também possui dois métodos, `Up` que serve para atualizar para a versão mais nova das tabelas, e o `Down` que serve para voltar para uma versão anterior.

Executaremos o comando para atualizar a tabela, passando o parâmetro `-Verbose` para que o Console apresente todas as operações. O comando ficará da seguinte maneira:

```
Update-Database -Verbose
```

Após executar o comando, receberemos um erro informando que a tabela `Produtos` já existe no banco de dados:

There is already an object named 'Produtos' in the database

Porém o Entity criou uma tabela chamada `__EFMigrationHistory`, que é utilizada para manter o histórico de migrações e o Entity utilizará para controlar as versões executadas no banco de dados.

Como poderemos aplicar essa migração? Precisamos refletir essa alteração no banco, mas para isso precisaríamos da versão onde a tabela foi criada. Veremos como resolver esse problema no próximo vídeo.