01

Como sincronizar o modelo de classes com a estrutura de dados?

Transcrição

É muito difícil uma aplicação Web manter-se por bastante tempo sem sofrer evoluções. Um exemplo mesmo é o sistema de loja que usamos no curso, estamos registrando apenas os produtos, porém também precisamos registrar as vendas dos produtos. Discutiremos como o Entity ajuda a evoluir a aplicação.

Como já fizemos os testes e entendemos como o *Change Tracker* funciona, na classe `Program` podemos apagar tudo deixando apenas o método `Main()`.

```
namespace Alura.Loja.Testes.ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

A primeira evolução que iremos fazer na aplicação é na classe `Produto`. O produto tem a propriedade `Preco`, porém ele não representa o preço total da compra, e sim o **valor unitário**. Por esse motivo mudaremos a propriedade para `PrecoUnitario`.

```
internal class Produto
{
    public int Id { get; internal set; }
    public string Nome { get; internal set; }
    public string Categoria { get; internal set; }
    public double PrecoUnitario { get; internal set; }

    public override string ToString()
    {
        return $"Produto: {this.Id}, {this.Nome}, {this.Categoria}, {this.PrecoUnitario}";
    }
}
```

Se tentarmos executar a aplicação, teremos diversos erros de compilação. Se analisarmos os pontos de erro, estarão na classe `ProdutoDAO`, que usa o `ADO.Net`. Essa é uma das vantagens do Entity, mesmo após alterarmos a aplicação, ele está preparado para essas situações.

Como não estamos mais usando a classe `ProdutoDAO` iremos removê-la. No **Gerenciador de Soluções**, clicaremos com o botão direito em cima do nome da classe `ProdutoDAO`, em seguida selecionaremos a opção "Excluir".

Agora que a aplicação voltou a compilar, vamos criar um nova propriedade na classe `Produto` chamada `Unidade`. Essa propriedade vai armazenar a unidade do produto, por exemplo *quilos*, *litros* e assim por diante.

```
internal class Produto
```

```
{
    public int Id { get; internal set; }
    public string Nome { get; internal set; }
    public string Categoria { get; internal set; }
    public double PrecoUnitario { get; internal set; }
    public string Unidade { get; internal set; }

    public override string ToString()

    {
        return $"Produto: {this.Id}, {this.Nome}, {this.Categoria}, {this.PrecoUnitario}";
    }
}
```

Porém como essa mudanças vão refletir no banco de dados? Olhando a nossa tabela, vemos que ela possui apenas `Id`, `Nome`, `Categoria` e `Preco`. Nós criamos a tabela com base nas informações que estavam contidas no arquivo `ddl-produtos.txt`.

> A sigla ***DDL*** significa ***Data Definition Language***, ela é um subconjunto da linguagem SQL. Durante as aulas falamos bastante dos comandos `INSERT`, `DELETE`, `UPDATE` e `SELECT`, esses comandos são ***DML*** que significa ***Data Manipulation Language***.

Para ajustarmos a tabela, teríamos que fazer um `ALTER TABLE` para mudarmos o nome de uma coluna e adicionar uma nova. Porém não queremos mais ter que cuidar da tabela, queremos que o Entity faça essa função, de sincronizar o banco com a classe modelo.

Essa parte responsável pela de sincronização é chamada de ***Migrations***, que fica em outro pacote. Para instalá-lo, vamos em "Ferramentas > Gerenciador de Pacote do NuGet > Console do Gerenciador de Pacotes", e digitamos:

PM> Install-Package Microsoft.EntityFrameworkCore.Tools -Version 1.1.1

> Usamos a versão `1.1.1`, porém você pode utilizar uma versão mais atualizada!

No próximo vídeo veremos como usar o *Migration*.