08

Primeiro uso da ferramenta

Transcrição

Já instalamos o Entity, vamos começar a usá-lo. Comentaremos a linha `GravarUsandoAdoNet();` que está dentro do método `Main()` da classe `Program`. Em seguida, colocaremos uma nova chamada para o método `GravarUsandoEntity()`, que usamos para persistir os dados com o Entity.

```
static void Main(string [] args)
{
    //GravarUsandoAdoNet();
    GravarUsandoEntity();
}
```

Criaremos o método `GravarUsandoEntity()`, e dentro dele colocaremos todo o conteúdo contido no método `GravarUsandoAdoNet()`.

```
static void Main(string [] args)
{
    //GravarUsandoAdoNet();
    GravarUsandoEntity();
}

private static void GravarUsandoEntity()
{
    Produto p = new Produto();
    p.Nome = "Harry Potter e a Ordem da Fênix";
    p.Categoria = "Livros";
    p.Preco = 19.89;

    using (var repo = new ProdutoDAO())
    {
        repo.Adicionar(p);
    }
}
```

Vamos fazer algumas modificações no método `GravarUsandoEntity()`. A estratégia do ADO.Net, era criar uma classe que representasse o *Data Access Object* de um modelo que seria persistido.

O Entity trabalha um pouco diferente, criaremos uma classe que vai representar e permitir persistirmos todas as classes necessárias, e não apenas uma específica como a de `Produto`.

Essa classe possui o conceito no Entity de contexto da aplicação. Como o modelo de negócios do projeto é uma loja, então mudaremos a chamada ao `DAO` para a classe `LojaContext()`. É uma convenção chamarmos as classe pelo modelo de negócios seguido do sufixo ***Context***. Também trocaremos o nome da variável `repo` para `contexto`.

```
private static void GravarUsandoEntity()
{
    Produto p = new Produto();
```

```
        p.Nome = "Harry Potter e a Ordem da Fênix";
        p.Categoria = "Livros";
        p.Preco = 19.89;

        using (var contexto = new LojaContext())
        {
            contexto.Adicionar(p);
        }
    }
```

O Visual Studio mostra um erro após a alteração, isso porque não criamos a classe ainda. Vamos criá-la. A classe criada ficará da seguinte maneira:

```
using System;

namespace Alura.Loja.Testes.ConsoleApp
{
    public class LojaContext : IDisposable
    {

    }
}
```

Primeiramente faremos com que a classe permita *usar a API do Entity* dentro dela. Para que isso aconteça, faremos a classe *herdar* de `DbContext`.

```
using Microsoft.EntityFrameworkCore;
using System;

namespace Alura.Loja.Testes.ConsoleApp
{
    public class LojaContext : DbContext
    {

    }
}
```

Informaremos *quais classe serão persistidas pelo Entity. Mas como será feito? Faremos isso por meio de uma propriedade que vai representar o conjunto de objetos da classe* `Produto`. *A propriedade ficará definida com o tipo* `DbSet<Produto>`, *e o nome da propriedade colocaremos o *mesmo nome* da tabela do banco de dados.

```
using Microsoft.EntityFrameworkCore;
using System;

namespace Alura.Loja.Testes.ConsoleApp
{
    public class LojaContext : DbContext
    {
        public DbSet<Produto> Produtos { get; set; }
    }
}
```

Ainda na classe, definiremos qual é o *banco de dados* e qual é o *endereço*. Para isso, iremos sobrescrever o método `OnConfiguring()` da classe `DbContext`.

```
using Microsoft.EntityFrameworkCore;
using System;

namespace Alura.Loja.Testes.ConsoleApp
{
    public class LojaContext : DbContext
    {
        public DbSet<Produto> Produtos { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {

        }
    }

}
```

Esse método tem como argumento de entrada, um *construtor de opções*. Usando o `optionsBuilder` chamaremos o método `UseSqlServer()`, passando como argumento o endereço do banco que usamos no `ProdutoDAO`.

```
using Microsoft.EntityFrameworkCore;
using System;

namespace Alura.Loja.Testes.ConsoleApp
{
    public class LojaContext : DbContext
    {
        public DbSet<Produto> Produtos { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=LojaDB;Trusted_Conn
        }
    }
}
```

Na classe `Program` dentro do `using`, substituiremos a chamada `repo.Adicionar(p)` por um código que usará o Entity. Tudo o que quisermos fazer no banco de dados, vai passar em uma instância do `LojaContext`.

Usando a variável `contexto`, chamaremos a propriedade `Produtos` que por sua vez chamará o método `Add()`, que recebe o objeto que iremos persistir.

```
private static void GravarUsandoEntity()
{
    Produto p = new Produto();
    p.Nome = "Harry Potter e a Ordem da Fênix";
    p.Categoria = "Livros";
    p.Preco = 19.89;
```

```
        using (var contexto = new LojaContext())
        {
            contexto.Produtos.Add(p);
        }
    }
```

Em seguida, basta salvar as mudanças usando o método `contexto.SaveChanges()` .

```
using (var contexto = new LojaContext())
{
    contexto.Produtos.Add(p);
    contexto.SaveChanges();
}
```

Isso é tudo o que precisamos para salvar um produto no banco de dados usando o Entity. Rodaremos a aplicação com o "Ctrl + F5". Se olharmos no banco de dados, veremos que o produto foi salvo.