01

Aplicando condições nas consultas

Transcrição

Nas aulas anteriores conversamos sobre relacionamentos. Nessa aula usaremos esses relacionamentos para montarmos *querys* um pouco mais complexas.

Como já fizemos em aulas anteriores, vamos mover todo o código contido no método `Main()` para um outro método chamado `UmParaUm()`. Em seguida criaremos o objeto de `LojaContext` e uma promoção para janeiro.

```
class Program
{
    static void Main(string[] args)
    {
        using(var contexto = new LojaContext())
        {
            var promocao = new Promocao();
            promocao.Descricao = "Queima Total 2017";
            promocao.DataInicio = new DateTime(2017, 1, 1);
            promocao.DataTermino = new DateTime(2017, 1, 31);
        }
    }
    // ...
}
```

Adicionaremos produtos nessa promoção, mas ao invés de criarmos novos produtos, pegaremos do banco de dados todos os produtos da categoria `"Bebidas"`. Estamos acostumados a pegar todos os produtos usando apenas o `ToList()`, mas o *LINQ* do C# possui métodos que nos auxiliam. Um desses métodos é o `Where()` que recebe uma expressão lambda booleana. Depois de pegar os produtos, adicionaremos dentro da promoção e a promoção no contexto. A classe ficará da seguinte maneira:

```
class Program
{
    static void Main(string[] args)
    {
        using(var contexto = new LojaContext())
        {
            var promocao = new Promocao();
            promocao.Descricao = "Queima Total Janeiro 2017";
            promocao.DataInicio = new DateTime(2017, 1, 1);
            promocao.DataTermino = new DateTime(2017, 1, 31);

            var produtos = contexto
                .Produtos
                .Where(p => p.Categoria == "Bebidas")
                .ToList();

            foreach(var item in produtos)
            {
                promocao.IncluiProduto(item);
            }
```

```
            contexto.Promocoes.Add(promocao);
        }
    }

    // ...
}
```

Após o `contexto.Promocoes.Add(promocao)` chamaremos o método `ExibeEntries()` para verificarmos como o Entity está tratando as informações.

```
class Program
{
    static void Main(string[] args)
    {
        using(var contexto = new LojaContext())
        {
            var promocao = new Promocao();
            promocao.Descricao = "Queima Total Janeiro 2017";
            promocao.DataInicio = new DateTime(2017, 1, 1);
            promocao.DataTermino= new DateTime(2017, 1, 31);

            var produtos = contexto
                .Produtos
                .Where(p => p.Categoria == "Bebidas")
                .ToList();

            foreach(var item in produtos)
            {
                promocao.IncluiProduto(item);
            }

            contexto.Promocoes.Add(promocao);
        ExibeEntries(contexto.ChangeTracker.Entries());
        }
    }

    // ...
}
```

Veremos que ele buscou apenas produtos com a categoria `"Bebidas"` (caso não tenha nada no banco cadastre alguns produtos para testar). Em seguida ele adicionou em `promocao` e deixou com o estado *Added*.

Como tudo parece estar funcionando, adicionaremos o `contexto.SaveChanges()`. Fora do escopo do `contexto`, criaremos outro objeto de `LojaContext()` para mostrarmos os produtos cadastrados na `promocao`. Quando fechamos um contexto do Entity, os objetos deixam de ser rastreados por aquele contexto.

```
class Program
{
    static void Main(string[] args)
    {
        using(var contexto = new LojaContext())
        {
            var promocao = new Promocao();
```

```
            promocao.Descricao = "Queima Total Janeiro 2017";
            promocao.DataInicio = new DateTime(2017, 1, 1);
            promocao.DataTermino = new DateTime(2017, 1, 31);

            var produtos = contexto
                .Produtos
                .Where(p => p.Categoria == "Bebidas")
                .ToList();

            foreach(var item in produtos)
            {
                promocao.IncluiProduto(item);
            }


            contexto.Promocoes.Add(promocao);
        ExibeEntries(contexto.ChangeTracker.Entries());
            contexto.SaveChanges();
        }

        using(var contexto2 = new LojaContext())
        {
            var promocao = contexto2.Promocoes.FirstOrDefault();
            Console.WriteLine("\nMotrando os produtos da promoção...");
            foreach(var item in promocao.Produtos)
            {
                Console.WriteLine(item.Produto);
            }
        }
    }

    // ...
}
```

Após executarmos a aplicação veremos que o primeiro contexto fez o `SELECT` em `Produtos` com a categoria `"Bebida"`, os produtos foram adicionados no *Change Tracker*, e então foi feito o `INSERT` em `Promocoes` e em `PromocaoProduto`.

Já o segundo contexto fez o `SELECT` na primeira promoção e não mostrou nada. Olhando no banco de dados vemos que os produtos foram adicionados no `ProdutoPromocao`, lembrando que para isso é necessário que os produtos existam no banco.

O próximo passo é descobrir por que ele não está mostrando os produtos.