01

O ChangeTracker e os estados Unchanged e Modified

Transcrição

Na aula anterior aprendemos como o Entity faz as operações de CRUD. Nessa aula veremos como esse framework gerencia os objetos que serão persistidos no banco de dados. Para começarmos, apagaremos todos os métodos da classe `Program`, mantendo apenas o método `Main()` vazio.

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

        }
    }
}
```

Usaremos o objeto de contexto do Entity, que é uma instância da classe `LojaContext`, nós armazenaremos em uma variável chamada `contexto`. Com o objeto `contexto`, usaremos a propriedade `contexto.Produtos` chamando o método `ToList()`.

```
namespace Alura.Loja.Testes.ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var contexto = new LojaContext())
            {
                var produtos = contexto.Produtos.ToList();
            }
        }
    }
}
```

Como queremos mostrar tudo o que temos no banco de dados, faremos um laço iterando pela lista `produtos`.

```
namespace Alura.Loja.Testes.ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            using(var contexto = LojaContext())
            {
                var produtos = contexto.Produtos.ToList();
                foreach (var p in produtos)
                {
                    Console.WriteLine(p);
```

```
                }
            }
        }
    }
}
```

Usaremos o "Ctrl + F5" para executar a aplicação. Recebemos como resultado o nome da classe em vez do produto, isso ocorre por que quando chamamos o `Console.WriteLine()` para a uma variável de referência, é chamado o método `ToString()` dessa variável. Olhando na classe `Produto`, não temos o método `ToString()` implementado, por isso vamos sobrescrevê-lo para que seja apresentado o nome do produto.

```
namespace Alura.Loja.Testes.ConsoleApp
{
    internal class Produto
    {
        public int Id { get; internal set; }
        public string Nome { get; internal set; }
        public string Categoria { get; internal set; }
        public double Preco { get; internal set; }

        public override string ToString()
        {
            return "Produto: " + this.nome;
        }
    }
}
```

Executando a aplicação com "Ctrl + F5" novamente, veremos os nomes dos produtos. Imaginando que o primeiro produto armazenado no banco de dados não esteja com o nome correto, poderemos atualizá-lo.

Começaremos pegando o primeiro produto da lista `produtos` com o método `First()` e armazenaremos em uma variável chamada `p1`. Em seguida, alteraremos o valor do nome e salvaremos as alterações. Para visualizarmos a mudança, vamos repetir o código onde pegamos todos os produtos do banco de dados e iteramos a lista com um laço de repetição.

```
namespace Alura.Loja.Testes.ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            using(var contexto = LojaContext())
            {
                var produtos = contexto.Produtos.ToList();
                foreach (var p in produtos)
                {
                    Console.WriteLine(p);
                }

                var p1 = produtos.First();
                p1.Nome = "Harry Potter";

                contexto.SaveChages();
```

```
                Console.WriteLine("=================");
                produtos = contexto.Produtos.ToList();
                foreach (var p in produtos)
                {
                    Console.WriteLine(p);
                }
            }
        }
    }
}
```

O resultado é o esperado. Mas como o Entity sabe que ao mudar uma propriedade de um dado na lista, ele precisa fazer o *Update* no banco de dados? O que acontece é que a classe que representa o contexto herda de `DbContext`. A instância chamada ***Change Tracker*** é responsável por rastrear todas as mudanças feitas no contexto. É possível recuperar todas as entidade usando o objeto `contexto` da seguinte forma:

```
contexto.ChangeTracker.Entries();
```

Vamos iterar a lista retornada pelo `Entries()` e imprimir os seus elementos. Comentaremos o trecho de código onde alteramos e mostramos o nome do produto.

```
static void Main(string[] args)
{
    using(var contexto = LojaContext())
    {
        var produtos = contexto.Produtos.ToList();
        foreach (var p in produtos)
        {
            Console.WriteLine(p);
        }

    Console.WriteLine("=================");
        foreach (var e in contexto.ChangeTracker.Entries())
        {
            Console.WriteLine(e);
        }

        //var p1 = produtos.First();
        //p1.Nome = "Harry Potter";

        //contexto.SaveChages();

        //Console.WriteLine("=================");
        //produtos = contexto.Produtos.ToList();
        //foreach (var p in produtos)
        //{
        //    Console.WriteLine(p);
        //}
    }
}
```

Executando a aplicação, vemos que foi apresentado o nome da classe `EntityEntry`. Essa classe possui uma propriedade chama `State` que registra o estado da entidade, e caso ela tenha sido alterada, o `SaveChanges()` terá que agir. Mostraremos

estado:

```
Console.WriteLine("=================");
foreach (var e in contexto.ChangeTracker.Entries())
{
    Console.WriteLine(e.State);
}
```

Como não modificamos as entidades, ao rodar a aplicação veremos como resultado que os dados estão como ***Unchanged***. Faremos uma alteração no último item da lista e verificaremos o estado novamente.

```
static void Main(string[] args)
{
    using(var contexto = LojaContext())
    {
        var produtos = contexto.Produtos.ToList();
        foreach (var p in produtos)
        {
            Console.WriteLine(p);
        }

        Console.WriteLine("=================");
        foreach (var e in contexto.ChangeTracker.Entries())
        {
            Console.WriteLine(e);
        }

        var p1 = produtos.Last();
        p1.Nome = "007 - O Espiao Que Me Amava";

        Console.WriteLine("=================");
        foreach (var e in contexto.ChangeTracker.Entries())
        {
            Console.WriteLine(e);
        }

        //contexto.SaveChages();

        //Console.WriteLine("=================");
        //produtos = contexto.Produtos.ToList();
        //foreach (var p in produtos)
        //{
        //    Console.WriteLine(p);
        //}
    }
}
```

Rodando a aplicação novamente vemos que após a mudança, o último elemento está como ***Modified***. É dessa forma que o Entity gerencia as entidades. Quando usamos o `SaveChanges`, o Entity emite um comando SQL diferente para cada estado.