06

Deleted e Detached

Transcrição

No último vídeo vimos o estado *Added* onde ele gera o comando SQL `insert`. Porém ficou faltando saber como enviar o comando SQL `delete`.

Você já deve estar imaginando que é por meio do método do Entity `contexto.Produtos.Remove()`. Faremos alguns exemplos para ver como o `ChangeTracker` irá se comportar ao chamar o método `contexto.Produtos.Remove()`.

Comentaremos o código onde adicionamos um produto ao contexto e onde salvamos as alterações. Em seguida, pegaremos um produto do banco de dados e armazenaremos em uma variável chamada `p1`, e passamos como argumento para o `contexto.Produtos.Remove()`.

```
static void Main(String [] args)
{
    using(var contexto = LojaContext())
    {
        var serviceProvider = contexto.GetInfrasctructure<IServiceProvider>();
        var loggerFactory = serviceProvider.GetService<ILoggerFactory>();
        loggerFactory.AddProvider(SqlLoggerProvider.Create());

        var produtos = contexto.Produtos.ToList();

        ExibeEntries(contexto.ChangeTracker.Entries());

        //var novoProduto = new Produto
        //{
        //    Nome = "Desinfetante",
        //    Categoria = "Limpeza",
        //    Preco = 2.99;
        //};
        //contexto.Produtos.Add(novoProduto);

        var p1 = produtos.First();
        contexto.Produtos.Remove(p1);

        ExibeEntries(contexto.ChangeTracker.Entries());

        //contexto.SaveChages();


        //ExibeEntries(contexto.ChangeTracker.Entries());

    }
}
```

Por enquanto não usaremos o `contexto.SaveChanges()` para salvar as alterações. Rodaremos a aplicação para ver qual estado o Entity colocou para o primeiro produto da lista.

No resultado, vemos que foi executado o comando SQL `SELECT` para trazer as informações do banco. Após isso, foram apresentados todos os dados com o estado ***Unchanged***. No código removemos o primeiro produto usando o `contexto.Produtos.Remove()`, dessa forma, o produto foi apresentado com o estado ***Deleted***.

Tiraremos os comentários do `contexto.SaveChanges()` e do `ExibeEntries()` que estão após a remoção.

```
static void Main(String [] args)
{
    using(var contexto = LojaContext())
    {
        var serviceProvider = contexto.GetInfrasctructure<IServiceProvider>();
        var loggerFactory = serviceProvider.GetService<ILoggerFactory>();
        loggerFactory.AddProvider(SqlLoggerProvider.Create());

        var produtos = contexto.Produtos.ToList();

        ExibeEntries(contexto.ChangeTracker.Entries());

        //var novoProduto = new Produto
        //{
        //    Nome = "Desinfetante",
        //    Categoria = "Limpeza",
        //    Preco = 2.99;
        //};
        //contexto.Produtos.Add(novoProduto);

        var p1 = produtos.First();
        contexto.Produtos.Remove(p1);

        ExibeEntries(contexto.ChangeTracker.Entries());

        contexto.SaveChages();

        ExibeEntries(contexto.ChangeTracker.Entries());

    }
}
```

Executando a aplicação, notamos que após apresentar os dados e colocar o estado do primeiro dado como *Deleted*, o Entity executou um comando SQL `DELETE`.

```
DELETE FROM [Produtos]
WHERE [Id] = @p0;
```

Ao mostrar novamente os dados, o primeiro produto da lista já não está sendo controlado pelo Entity e foi removido do banco de dados. Mas o que aconteceria se adicionarmos um produto no Entity e removê-lo antes do `contexto.SaveChanges()`? Ele enviaria um comando SQL mesmo sem ter o produto no banco de dados?

```
static void Main(String [] args)
{
    using(var contexto = LojaContext())
    {
        var serviceProvider = contexto.GetInfrasctructure<IServiceProvider>();
```

```
            var loggerFactory = serviceProvider.GetService<ILoggerFactory>();
            loggerFactory.AddProvider(SqlLoggerProvider.Create());

            var produtos = contexto.Produtos.ToList();

            ExibeEntries(contexto.ChangeTracker.Entries());

            var novoProduto = new Produto()
                    {
                        Nome = "Sabão em pó",
                        Categoria = "Limpeza",
                        Preco = 5.99

                    };

                    contexto.Produtos.Add(novoProduto);

                    contexto.Produtos.Remove(novoProduto);

                    ExibeEntries(contexto.ChangeTracker.Entries());

                    contexto.SaveChanges();

                    ExibeEntries(contexto.ChangeTracker.Entries());

        }
    }
```

Nesse caso, foi executado o comando SQL `SELECT` para trazer os produtos com o estado *Unchanged*, porém após adicionar e remover, o produto "Sabão em pó" não apareceu como um produto *Added*. Isso aconteceu porque após removermos, o Entity retira o produto da sua lista de rastreamento. Com isso, após salvarmos as alterações, não é enviada nenhum comando SQL para o banco.

Mas o que aconteceu com esse produto `novoProduto` ? Podemos ver o seu estado usando o método `contexto.Entry()` que nos retorna uma variável do tipo `EntityEntry` . Adicionaremos ao final do método `Main()` :

```
var entry = contexto.Entry(novoProduto);
Console.WriteLine("\n\n" + entry.Entity.ToString() + " - " + entry.State);
```

Rodando a aplicação podemos ver que o estado desse produto está como ***Detached***. Este estado representa que o objeto não está sendo monitorado.