05

Alterando objetos com o Entity

Transcrição

Aprenderemos a atualizar um produto. Na classe `ProdutoDAO`, temos o método `Atualizar()`. O método recebe um produto, monta o comando SQL, criar e coloca os parâmetro no comando e por fim, executa. Agora, utilizaremos o Entity.

Dentro do método `Main()` da classe `Program`, comentaremos todas as chamadas de métodos, e em seguida, faremos uma nova chamada ao método `AtualizarProduto()` que ainda não existe.

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

Após isso, criaremos o método `AtualizarProduto()`. Porém, para verificarmos a atualização do produto, antes é necessário incluir um produto - já que removemos todos do banco de dados no vídeo anterior - e mostrá-lo, efetuar a atualização e mostrá-lo novamente.

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

private static void AtualizarProduto()
{
    // inclui um produto
    RecuperarProdutos();

    // atualiza o produto
    RecuperarProdutos();
}
```

O primeiro passo é simples, basta chamarmos o método `GravarUsandoEntity()` para que seja incluído no banco de dados o produto já preparado no método.

```
private static void AtualizarProduto()
{
    // inclui um produto
    GravarUsandoEntity();
```

```
        RecuperarProdutos();

        // atualiza o produto
        RecuperarProdutos();
    }
```

Com o produto incluído, vamos criar um objeto que é a instância da classe `LojaContext` .

```
    private static void AtualizarProduto()
    {
        // inclui um produto
        GravarUsandoEntity();
        RecuperarProdutos();

        // atualiza o produto
        using(var repo = LojaContext())
        {

        }
        RecuperarProdutos();
    }
```

Com o objeto, em vez de pedirmos uma lista de produtos, vamos pedir apenas o primeiro encontrado e armazená-lo em um variável chamada `primeiro` .

```
    private static void AtualizarProduto()
    {
        // inclui um produto
        GravarUsandoEntity();
        RecuperarProdutos();

        // atualiza o produto
        using(var repo = LojaContext())
        {
            Produto primeiro = repo.Produtos.First();
        }
        RecuperarProdutos();
    }
```

O `First()` irá retornar o primeiro produto encontrado no banco de dados. Basta agora modificarmos o nome do produto e passá-lo como argumento para o `repo.Produto.Update()` . Sem esquecer de no final salvar as alterações.

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

    private static void AtualizarProduto()
```

```
{
    // inclui um produto
    GravarUsandoEntity();
    RecuperarProdutos();

    // atualiza o produto
    using (var repo = new LojaContext())
    {
        Produto primeiro = repo.Produtos.First();
        primeiro.Nome = "Cassino Royale - Editado";
        repo.Produtos.Update(primeiro);
        repo.SaveChanges();
    }
    RecuperarProdutos();
}
```

Executaremos a classe e veremos que tudo ocorreu como esperado. Porém, a nossa classe com os métodos que usam o Entity foi montada apenas para testes. No próximo vídeo veremos como ficaria uma classe *DAO* usando o Entity.