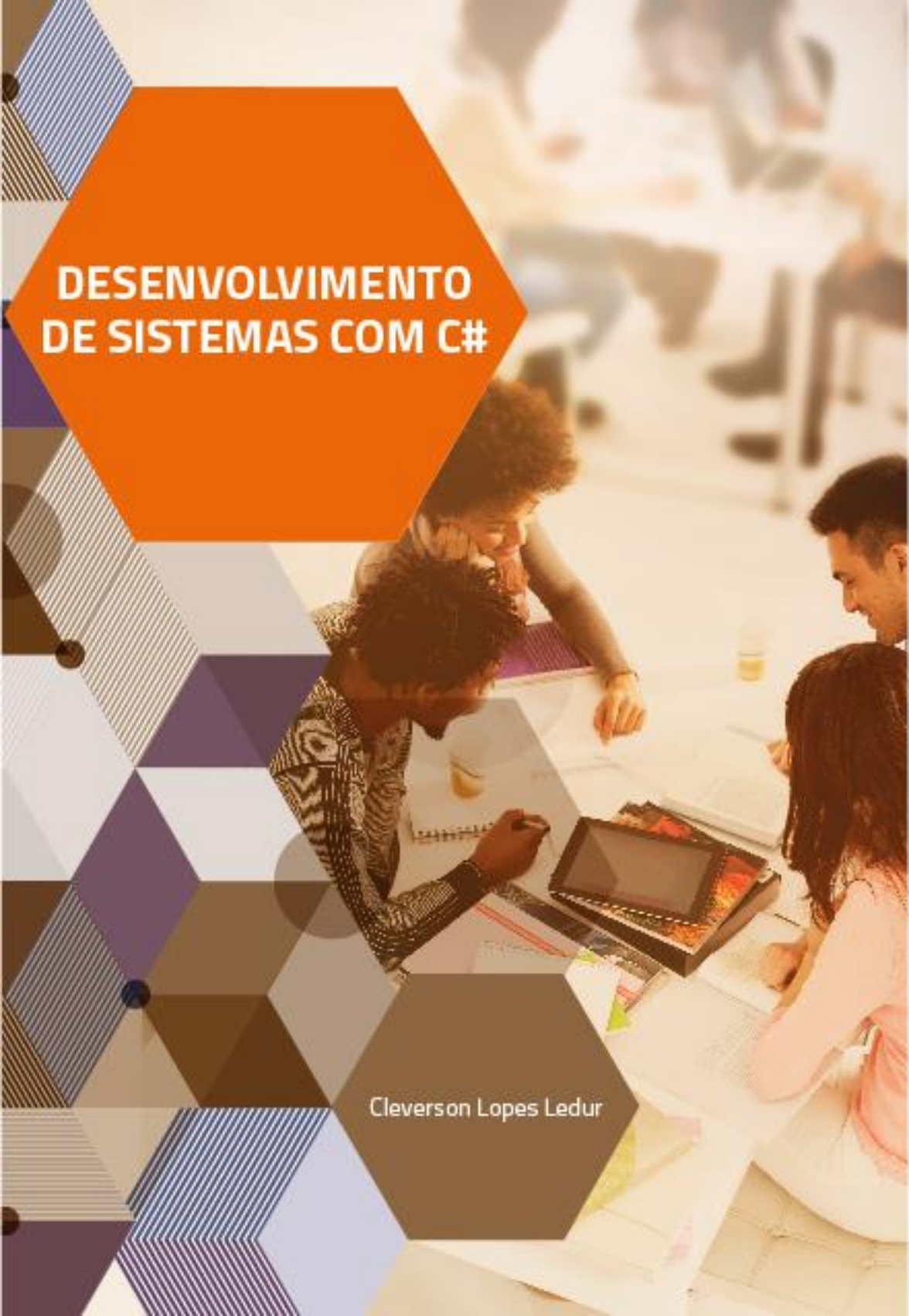
DESENVOLVIMENTO DE SISTEMAS COM C#



Cleverson Lopes Ledur

Revisão técnica:

Jeferson Faleiro Leon
Desenvolvimento de Sistemas
Especialista Formação Pedagógica de Professores
Professor do curso Técnico em informática

L475d Ledur, Cleverson Lopes.
Desenvolvimento de sistemas com #C [recurso eletrônico] / Cleverson Lopes Ledur; [revisão técnica: Jeferson Faleiro Leon]. – Porto Alegre : SAGAH, 2018.

ISBN 978-85-9502-314-7

1. Ciência da computação. 2. Linguagens de programação de computador. I. Título.

CDU 004.43

Catalogação na publicação: Karin Lorien Menoncin CRB-10/2147

Utilizar o Entity Framework para persistência em banco de dados – V

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Analisar como é realizada a alteração de um registro.
- Reconhecer a forma de remover um registro.
- Utilizar o Server Explorer para conectar e visualizar o banco de dados.

Introdução

Atualmente, poucos sistemas trabalham localmente. A maior parte dos sistemas atuais utiliza servidores para realizar tarefas de computação e armazenamento de dados. Muitos desses servidores não são físicos e ficam localizados nas chamadas nuvens computacionais. Para utilizar esses servidores e também bancos de dados, é preciso criar conexões por meio do Server Explorer ou do Gerenciador de Servidores do Visual Studio. É pela conexão com bancos de dados que o Entity Framework realiza as modificações solicitadas por meio do código. Duas dessas alterações consistem em atualizar e excluir registros. Para elas, o Entity Framework oferece métodos de alto nível em que é possível, com pequenas alterações em objetos e poucas chamadas, realizar a persistência de modificações de registros de uma tabela.

Neste texto, você irá aprender a realizar atualizações e exclusões em registros por meio do ORM Entity Framework. Além disso, vai ver como adicionar um banco de dados por meio do Server Explorer.

Atualizando dados

A atualização de dados utilizando o Entity Framework é bastante simples e fácil. Com ela, você pode alterar as informações utilizando objetos que representam os registros de uma tabela (FREEMAN, 2017). Logo, você altera apenas os objetos e então, por meio do método `SaveChanges()`, você informa ao Entity Framework que deseja aplicar as alterações na tabela.

No código a seguir, você pode ver um exemplo.

```
public Usuario Bloquear(Usuario)
        {
            usuario.bloqueado = true;
            context.SaveChanges();
            return usuario;
        }
```

Como foi visto no exemplo anterior, estão sendo realizadas as seguintes etapas:

- obtenção do objeto que representa um registro no banco de dados;
- alteração de um dos atributos desse objeto;
- chamada do método SaveChanges() para atualizar as informações no banco de dados.

Abaixo você pode ver outro exemplo. Nele, é feito um update de um registro por meio do Entity Framework. Nesse exemplo, há uma classe de modelo chamada Estudante. Você pode verificar no código a seguir que há alguns atributos nessa classe modelo.

```
using System;
using System.Collections.Generic;

public partial class Estudante
{
    public Estudante()
    {
        this.Cursos = new HashSet<Curso>();
    }

    public int EstudanteID { get; set; }
    public string EstudanteName { get; set; }
    public Nullable<int> StandardId { get; set; }
    public byte[] RowVersion { get; set; }
```

```
        public virtual Standard Standard { get; set; }
        public virtual EstudanteEndereco EstudanteEndereco { get; set; }
        public virtual ICollection<Curso> Cursos { get; set; }
    }
```

Junto a essa classe modelo, você também precisa da classe de contexto, o `SchoolDBEntities`. A seguir, no exemplo, você pode ver a especificação da classe `Estudante` e outras que se relacionam com ela. Com o `DBContext`, você pode fazer então a configuração do Entity Framework para ele saber quais classes são mapeadas para quais tabelas no banco de dados.

```
using System;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Core.Objects;
using System.Linq;

public partial class SchoolDBEntities : DbContext
{
    public SchoolDBEntities()
        : base("name=SchoolDBEntities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {

    }

    public virtual DbSet<Curso> Cursos { get; set; }
    public virtual DbSet<Standard> Standards { get; set; }
    public virtual DbSet<Estudante> Estudantes { get; set; }
    public virtual DbSet<EstudanteEndereco> EstudanteEnderecos { get; set;
}
    public virtual DbSet<Professor> Professores { get; set; }
}
```

Agora que já viu onde está trabalhando, você pode observar o código em que realiza a alteração. Veja, no próximo código, um trecho de um método que realiza inicialmente uma consulta na tabela Estudantes utilizando a cláusula

`Where`. Nesse exemplo também há uma expressão lambda para selecionar o estudante que possui como nome `Novo Estudante1`.

```
Estudante stud;

using (var ctx = new SchoolDBEntities())
{
    stud = ctx.Estudantes.Where(s => s.EstudanteName == "Novo
Estudante1").FirstOrDefault<Estudante>();
}

if (stud != null)
{
    stud.EstudanteName = "Estudante1 Atualizado";
}

using (var dbCtx = new SchoolDBEntities())
{
   dbCtx.SaveChanges();
}
```

Logo após a consulta, você deve verificar se o retorno é nulo. Caso contrário, precisa realizar a alteração do atributo `EstudanteName`. Ao fim, deve realizar a gravação do registro no banco de dados por meio da chamada `SaveChanges()` do objeto context.

Removendo dados

Em alguns momentos do desenvolvimento de sistemas, você também vai precisar remover registros do banco de dados. Por exemplo, quando um item de uma loja não é mais vendido, é necessário retirá-lo do sistema.

A utilização do Entity Framework torna a remoção de registros de banco de dados bastante fácil. Basicamente, o que você precisa fazer é, por meio do objeto context e do atributo da entidade da qual está realizando a alteração, chamar o método `Remove()`, passando o objeto que será excluído como parâmetro.

Veja no exemplo.

```
public void Remover(Usuario usuario)
        {
            context.Usuarios.Remove(usuario);
        }
```

Outra forma de remover uma entidade é por meio do State (estado) do objeto. Observe o código a seguir. Ela mostra um caso em que, utilizando um exemplo similar ao usado na seção de alteração, você poderia ter uma exclusão apenas modificando o State do objeto.

```
Estudante EstudanteToDelete;

using (var ctx = new SchoolDBEntities())
{
    EstudanteToDelete = ctx.Estudantes.Where(s => s.EstudanteName ==
"Estudante1").FirstOrDefault<Estudante>();
}


using (var newContext = new SchoolDBEntities())
{
    newContext.Entry(EstudanteToDelete).State =
System.Data.Entity.EntityState.Deleted;

    newContext.SaveChanges();
}
```

Utilizando o Server Explorer

Atualmente, poucos sistemas trabalham localmente. A maior parte dos sistemas atuais utiliza servidores para realizar tarefas de computação e armazenamento de dados. Muitos desses servidores não são físicos e ficam localizados nas chamadas nuvens computacionais (DEITEL; DEITEL, 2016). A Microsoft possui uma plataforma própria de nuvem computacional chamada Azure. Nela, você pode criar instâncias para computação e armazenamento de dados. Essa plataforma também disponibiliza uma grande variedade de opções de APIs que você pode utilizar para incrementar sua aplicação. Tanto o uso de nuvens computacionais quanto a adição de servidores e bancos de dados são realizados por meio do Gerenciador de Servidores do Visual Studio (Figura 1).

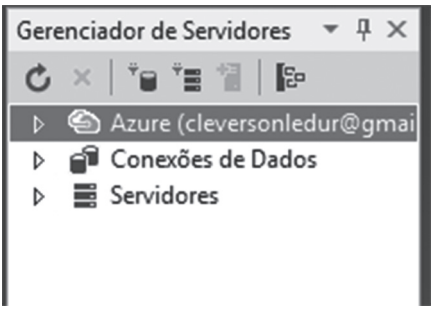


Figura 1. Gerenciador de Servidores.

No Gerenciador de Servidores, você adicionar um servidor, remover um servidor, criar um novo banco de dados ou se conectar a um banco de dados existente. A seguir, você pode verificar os passos para realizar cada ação.

Para adicionar um link para um servidor no Gerenciador de Servidores:

1. Selecione **Conectar ao Servidor** no menu **Ferramentas** ou clique no ícone **Conectar ao Servidor** no Gerenciador de Servidores.
2. A caixa de diálogo **Adicionar Servidor** é aberta.
3. Na caixa de texto **Computador**, insira o nome ou o endereço IP do servidor.

Para remover um link para um servidor no Gerenciador de Servidores:

1. No Gerenciador de Servidores, expanda o nó **Servidores**.
2. Clique com o botão direito do mouse no **servidor** que não é mais necessário.
3. Selecione **Excluir** no menu de atalho.
4. Não há efeito sobre o servidor real. Você está removendo o link da sua exibição.

Para conectar-se novamente a um servidor remoto como um usuário diferente:

1. Exclua todas as conexões existentes para o computador remoto como o usuário desejado.

2. Você não pode criar várias conexões para um servidor remoto como o mesmo usuário desse computador. Isso inclui conexões NET USE.
3. Selecione **Conectar ao Servidor** no menu **Ferramentas** ou clique no ícone **Conectar ao Servidor** na barra de ferramentas.
4. A caixa de diálogo **Adicionar Servidor** é aberta.
5. Na caixa **Computador**, insira o nome ou o endereço IP do servidor.
6. Selecione **Conectar-se** usando outro nome de usuário.
7. A caixa de diálogo **Conectar como** é aberta.
8. Insira um novo nome de usuário e senha.
9. Na caixa **Computador**, insira o nome ou o endereço IP do servidor.

Para conectar-se novamente a um servidor em seu computador local como um usuário diferente:

1. Saia do Visual Studio e finalize todos os outros programas que estão em execução no seu computador.
2. Você deve se conectar a um servidor localizado em seu próprio computador como o usuário atual do computador.
3. No menu **Iniciar** do Windows, selecione **Desligar**.
4. A caixa de diálogo **Desligar o Windows** aparecerá.
5. Clique em **Fazer logoff**.
6. A caixa de diálogo **Fazer logon** aparecerá.
7. Insira um novo nome de usuário e senha.
8. Abra o Visual Studio e selecione **Conectar ao Servidor** no menu **Ferramentas** ou clique no ícone **Conectar ao Servidor** na barra de ferramentas.
9. A caixa de diálogo **Adicionar Servidor** é aberta.
10. Na caixa **Computador**, insira o nome do servidor desejado no seu computador.

Referências

DEITEL, P.; DEITEL, H. *Visual C#:* how to program. 6th ed. London: Pearson, 2016.

FREEMAN, A. *Essential C# features*: Pro ASP.NET Core MVC 2. 7th ed. Berkeley: Apress, 2017.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo: