

# FIA/P GRADUAÇÃO

# ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#07 - RELACIONAMENTOS



thiagoyama



thiagoyama@gmail.com

## #07 - RELACIONAMENTOS

---

- Relacionamentos
- Um para um
- Um para muitos
- Muitos para muitos

- Existem 3 tipos de relacionamentos:
  - **Uma para um** (One to one): é um relacionamento onde uma entidade só possui uma ligação com outra entidade e esta só possui a ligação de volta;
  - **Um para muitos** (One to Many): uma entidade possui várias ligações com outra entidade e esta só possui a ligação de volta;
  - **Muitos para muitos** (Many to many): uma entidade possui várias ligações com outra entidade e esta possui também várias ligações de volta;

# UM PARA UM

# UM PARA UM

- Precisamos adicionar uma propriedade de navegação para a outra entidade:

```
public class Cliente
{
    public int ClienteId { get; set; }
    public string Nome { get; set; }
    public Endereco Endereco { get; set; }
}
```

Propriedade de navegação para a classe Endereco.

```
public class Endereco
{
    public int EnderecoId { get; set; }
    public string Logradouro { get; set; }
}
```

- Podemos adicionar também uma propriedade de foreign key na classe:

```
public class Cliente
{
    public int ClienteId { get; set; }
    public string Nome { get; set; }

    //Foreign Key
    public int EnderecoId { get; set; }
    //Navigation Property
    public Endereco Endereco { get; set; }
}
```

**EnderecoId** é o mesmo nome da chave primária da entidade **Endereço**.

# UM PARA MUITOS




# UM PARA MUITOS

- Precisamos adicionar uma propriedade de navegação para a outra entidade, mas dessa vez é uma **ICollection**:

```
public class Cliente
{
    public int ClienteId { get; set; }
    public string Nome { get; set; }

    public virtual ICollection<Qualificacao> Qualificacoes { get; set; }
}
```



Propriedade de coleção de  
Qualificações.

```
public class Qualificacao
{
    public int QualificacaoId { get; set; }
    public string Nome { get; set; }
    public int Score { get; set; }
}
```

# UM PARA MUITOS

- Podemos ter um relacionamento bi-direcional:

```
public class Cliente
{
    public int ClienteId { get; set; }
    public string Nome { get; set; }

    public virtual ICollection<Qualificacao> Qualificacoes { get; set; }
}
```

Relacionamento bi-direcional.

```
public class Qualificacao
{
    public int QualificacaoId { get; set; }
    public string Nome { get; set; }
    public int Score { get; set; } public
    public Cliente Cliente { get; set; }
}
```


# MUITOS PARA MUITOS

# MUITOS PARA MUITOS

- As duas entidades precisam possuir uma propriedade de navegação **ICollection**:

```
public class Cliente
{
    public int ClienteId { get; set; }
    public string Nome { get; set; }
    public virtual ICollection<Qualificacao> Qualificacoes { get; set; }

    public virtual ICollection<Veiculo> Veiculos { get; set; }
}
```



As duas entidades possuem as propriedades de coleções

```
public class Veiculo
{
    public int VeiculoId { get; set; }
    public string Modelo { get; set; }
    public ICollection<Cliente> Clientes { get; set; }
}
```

**Copyright © 2013 - 2017 - Prof. Me. Thiago T. I. Yamamoto**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*A vida vai ficando cada vez mais dura perto do topo.  
Friedrich Nietzsche*