


Jakarta EE Web Total


Relacionamentos na JPA




Tópicos Abordados




- Modelo OO x Relacional
- Definindo relacionamentos
- Navegabilidade
- Dono do relacionamento
- Relacionamentos *Eager* e *Lazy*

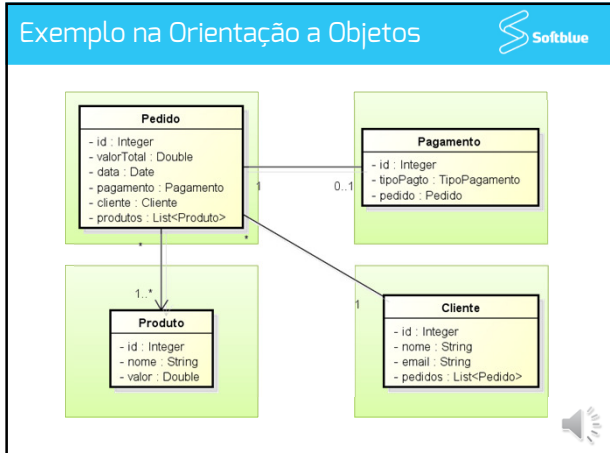


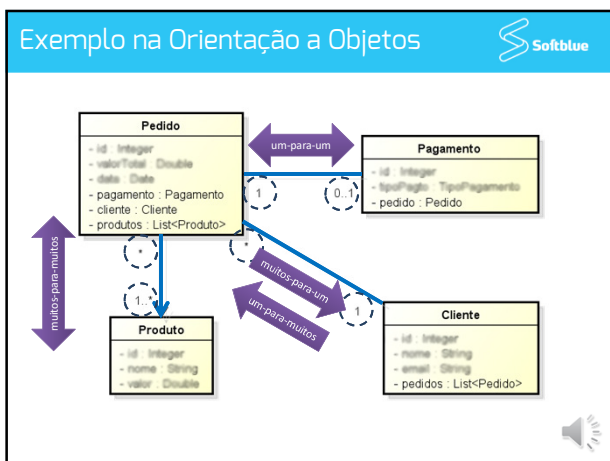
Relacionamentos em JPA

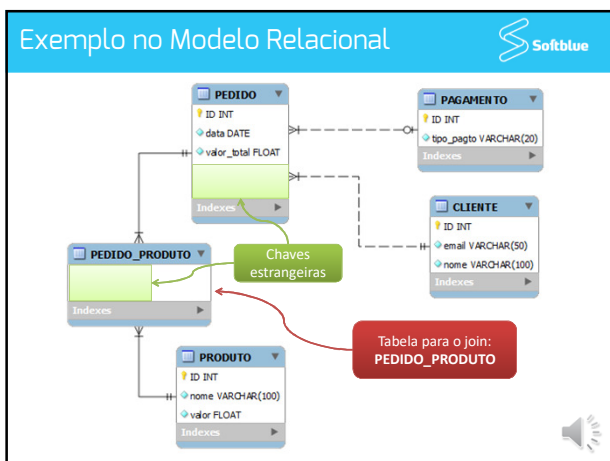



- O JPA é capaz de trabalhar com relacionamentos entre entidades
 - Em orientação a objetos, um relacionamento existe quando um objeto da classe A possui um atributo que referencia um objeto B, de outra classe
 - No modelo relacional, um relacionamento existe quando uma tabela A referencia uma tabela B através de uma chave estrangeira (*foreign key*)










Definindo Relacionamentos



- Os relacionamentos são definidos nas entidades através de anotações

Annotation	Tipo de Relacionamento
@OneToOne	Um-para-Um
@OneToMany	Um-para-Muitos
@ManyToOne	Muitos-para-Um
@ManyToMany	Muitos-para-Muitos

- Outras anotações utilizadas em relacionamentos

Annotation	Tipo de Relacionamento
@JoinColumn	Coluna para chave estrangeira
@JoinTable	Tabela auxiliar para o join



Definindo Relacionamentos



Pedido.java


```

@OneToOne
@JoinColumn(name = "pagamento_id")
private Pagamento pagamento;


@ManyToOne
@JoinColumn(name = "cliente_id", nullable = false)
private Cliente cliente;

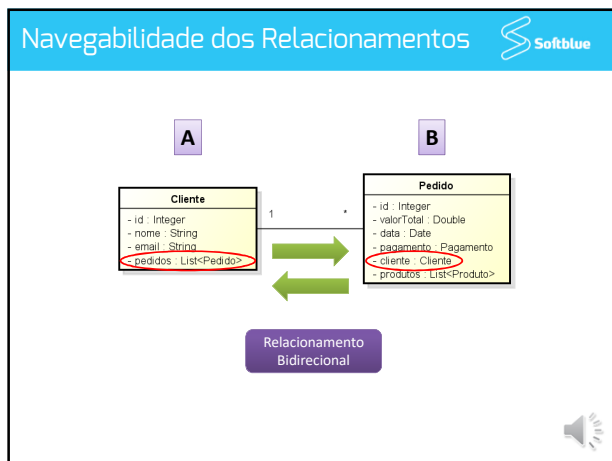
@ManyToMany
@JoinTable(name = "PEDIDO_PRODUTO",
    joinColumns = @JoinColumn(name = "pedido_id"),
    inverseJoinColumns = @JoinColumn(name = "produto_id"))
private List<Produto> produtos;
    
```

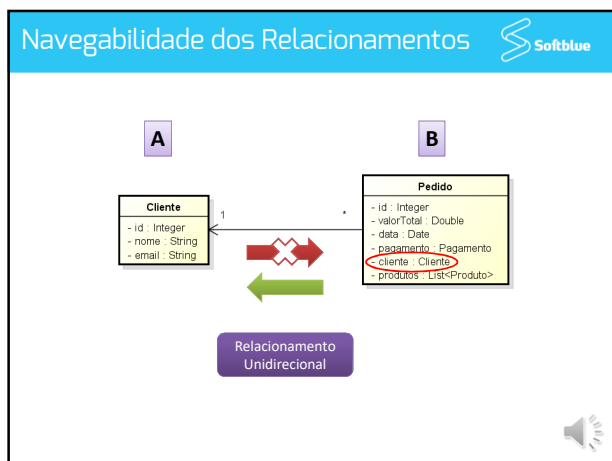


Navegabilidade dos Relacionamentos


- Relacionamentos unidirecionais
 - É possível navegar do lado *A* para o *B*
 - Não é possível navegar do lado *B* para o *A*
- Relacionamentos bidirecionais
 - É possível navegar do lado *A* para o *B*
 - É possível navegar do lado *B* para o *A*



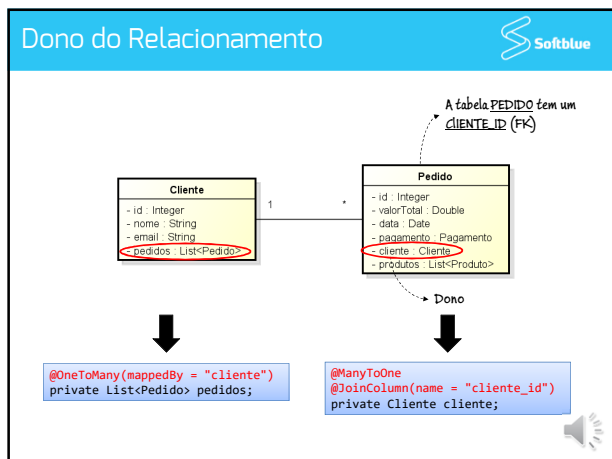


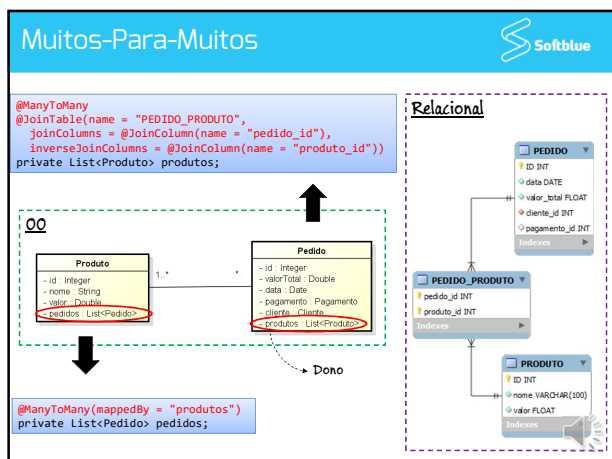


Dono do Relacionamento

- Em relacionamentos bidirecionais, um dos lados do relacionamento é o *dono do relacionamento* (*relationship owner*)

Tipo	Dono
Um-para-Um	Lado que possui a chave estrangeira (<i>foreign key</i>)
Um-para-Muitos	Lado "muitos"
Muitos-para-Um	Lado "muitos"
Muitos-para-Muitos	A critério da aplicação





Relacionamentos *Eager* e *Lazy*

- Quando uma entidade que possui relacionamentos é carregada, a JPA permite duas abordagens
 - Carregar automaticamente as entidades dos relacionamentos (*EAGER*)
 - Carregar os relacionamentos apenas quando eles forem necessários (*LAZY*)
 - A JPA assume um padrão
 - @OneToOne** e **@ManyToOne** = **EAGER**
 - @OneToMany** e **@ManyToMany** = **LAZY**

Exemplos

```

public class Pedido {
    @OneToOne(fetch = FetchType.EAGER)
    private Pagamento pagamento;

```

O pagamento será carregado automaticamente

```

public class Pedido {
    @OneToOne(fetch = FetchType.LAZY)
    private Pagamento pagamento;

```

O pagamento não será carregado automaticamente

Será carregado apenas quando for utilizado

Exemplos

```

public class Cliente {
    @OneToMany(fetch = FetchType.EAGER)
    private List<Pedido> pedidos;

```

Os pedidos serão carregados automaticamente

```

public class Cliente {
    @OneToMany(fetch = FetchType.LAZY)
    private List<Pedido> pedidos;

```

Os pedidos não serão carregados automaticamente

Serão carregados apenas quando forem utilizados

Qual Escolher: *EAGER* ou *LAZY*?

- Entre *EAGER* ou *LAZY*, não existe uma opção melhor ou pior
 - Vai depender de cada situação
- EAGER*
 - Reduz o acesso ao banco de dados para leitura de dados
 - Ocupa mais memória
- LAZY*
 - É preciso fazer vários acessos ao banco de dados para obter os dados conforme a necessidade
 - Ocupa menos memória