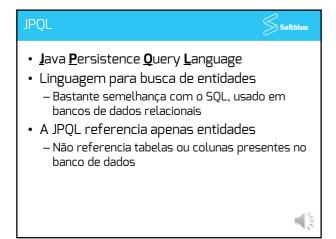


# Tópicos Abordados • JPQL - Exemplos de queries - Relacionamentos - JPQL e o Fetch Mode EAGER - Parâmetros em queries - Named queries - Named queries - Update & Delete • Criteria API - Passos para usar a API - Exemplos - Relacionamentos - Metamodel API



### Exemplos de Queries em JPQl



- Buscar todas as contas correntes

  SELECT c FROM ContaCorrente c
- Buscar contas correntes com limite superior a 1000
   SELECT c FROM ContaCorrente c WHERE c.limite > 1000
- Buscar os nomes dos titulares de contas da agência 3456
   SELECT c.titularConta FROM ContaCorrente c MHERE c.numAgencia = 3456
- Buscar a quantidade de contas do titular José Silva

  SELECT COUNT(c) FROM ContaCorrente c WHERE c.titularConta = 'José Silva'



### Exemplos de Queries em JPQL



• Buscar a soma dos saldos das contas correntes, agrupada por agência

SELECT c.numAgencia, SUM(c.saldo) FROM ContaCorrente c GROUP BY c.numAgencia ORDER BY c.numAgencia

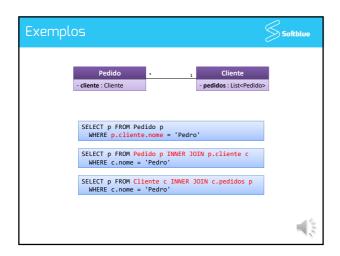


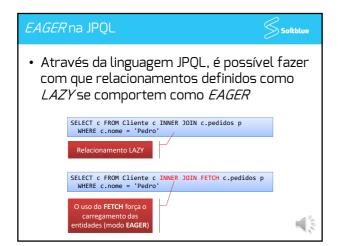
### Relacionamentos e a JPQL

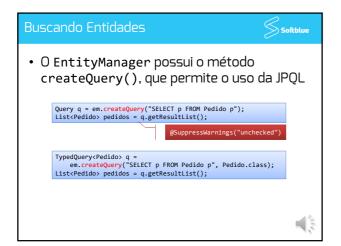


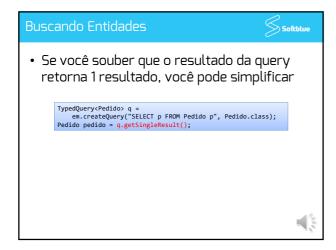
- A linguagem JPQL também suporta relacionamentos
- Os relacionamentos podem ser expressos de duas formas
  - Usando o ".", como acontece com qualquer propriedade de uma entidade
    - Possível em relacionamentos um-para-um e muitos-paraum
  - Usando o conceito de *join*, similar ao aplicado na linguagem SQL
    - (INNER) JOIN
    - LEFT (OUTER) JOIN
    - RIGHT (OUTER) JOIN

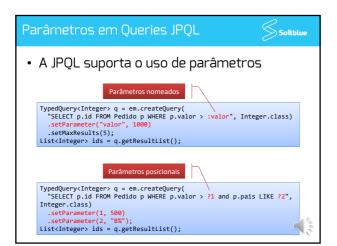


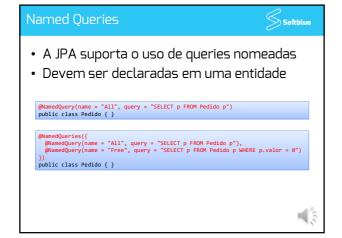






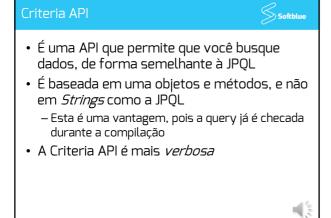






## Named Queries • A execução de uma named query é parecida com a de qualquer query JPQL TypedQuery<Pedido> q = em.createNamedQuery("All", Pedido.class); List<Pedido> pedidos = q.getResultList();

## Update & Delete com JPQL A JPQL suporta a atualização e exclusão de registros em lote, de acordo com os critérios desejados Query q = em.createQuery( "UPDATE Pedido p SET p.valor = 100 WHERE p.valor = 50"); q.executeUpdate(); Query q = em.createQuery( "DELETE FROM Pedido p WHERE p.valor = 20"); q.executeUpdate();



### • Exemplo de query – Listar os números de contas bancárias cujo saldo seja inferior a R\$ 5000, e ordená-las por ordem decrescente de saldo • Em JPOL: SELECT c.numero FROM ContaBancaria c WHERE c.saldo < 5000 ORDER BY c.saldo DESC • Usando a Criteria API... Passos Para Usar a Criteria API 1. Obter um CriteriaBuilder CriteriaBuilder cb = em.getCriteriaBuilder(); 2. Criar um CriteriaQuery CriteriaQuery<String> cq = cb.createQuery(String.class); 3. Definir o FROM Root<ContaBancaria> contaBancaria = cq.from(ContaBancaria.class); 4. Definir o SELECT cq.select(contaBancaria.get(ContaBancaria\_.numero)); Passos Para Usar a Criteria API 5. Definir o WHERE cq.where(cb.lessThan(contaBancaria.get(ContaBancaria\_.saldo), 5000.0)); 6. Definir o ORDER BY cq.orderBy(cb.desc(contaBancaria.get(ContaBancaria\_.saldo)));

7. Executar a query

TypedQuery<String> q = em.createQuery(cq); List<String> result = q.getResultList();

