



Uma **consulta simples por id** que retorna um único **objeto** no JPA é uma chamada de um ***find()*** no ***EntityManager***, onde preciso passar apenas o **tipo da entidade** e o **ID**:

```
public class ProdutoDao {  
    private EntityManager entityManager;  
  
    public ProdutoDao(EntityManager entityManager) { this.entityManager = entityManager; }  
  
    public void cadastrar(Produto produto) { entityManager.persist(produto); }  
  
    public Produto buscarPorId(Long id) {  
        return entityManager.find(Produto.class, id);  
    }  
}  
  
public static void main(String[] args) {  
    criarObjetos();  
  
    EntityManager entityManager = JPAUtil.criarEntityManager();  
    ProdutoDao produtoDao = new ProdutoDao(entityManager);  
  
    System.out.println(produtoDao.buscarPorId(Long.valueOf(2)).getNome());  
}  
  
Hibernate: select produto0_.id as id1_1_0_, produto0_.categoria_id as categ  
Geladeira gamer
```

Uma consulta por múltiplos resultados (***sem filtros***) é feita através do método ***createQuery()*** que cria uma **query** baseada em um JPQL (***uma linguagem baseada em SQL, mas orientada a Objetos***) além de ser uma boa **prática especificar a classe da entidade**, depois de montar a **query** é preciso executá-la no banco de dados com ***getResultList()***:

```
public Produto buscarPorId(Long id) {  
    return entityManager.find(Produto.class, id);  
}  
  
public List<Produto> buscarTodos() {  
    String jpql = "from Produto";  
    return entityManager.createQuery(jpql, Produto.class).getResultList();  
}  
  
public static void main(String[] args) {  
    criarObjetos();  
  
    EntityManager entityManager = JPAUtil.criarEntityManager();  
    ProdutoDao produtoDao = new ProdutoDao(entityManager);  
  
    produtoDao.buscarTodos().stream().forEach(p -> System.out.println(p.getNome() + " preço: " + p.getPreco()));  
}  
  
Hibernate: select produto0_.id as id1_1_, produto0_.categoria_id as categoria0_1_, produto0_.dataCadastro as dataCa  
Hibernate: select categoria0_.id as id1_0_0_, categoria0_.nome as nome2_0_0_ from categorias categoria0_ where ca  
Hibernate: select categoria0_.id as id1_0_0_, categoria0_.nome as nome2_0_0_ from categorias categoria0_ where ca  
Nokia preço: 10000.00  
Geladeira gamer preço: 20000.00
```

Obviamente os **SELECT's** só tem poder quando são feitos através de **filtros**, colocar filtros em consultas **JPQL** é bem parecido com **SQL**, a maior diferença é que os nomes de **tabela** são o nome das **classes** e as **colunas** são os **atributos**.

O **JPQL** recebe um placeholder com **“:xxxx”**, que precisa ser **substituído** antes de disparar a query:

```
public List<Produto> buscarPorNome(String nome) {  
    String jpql = "FROM Produto p WHERE p.nome = :nome";  
  
    return entityManager.createQuery(jpql, Produto.class)  
        .setParameter("nome", nome)  
        .getResultList();  
}  
  
public static void main(String[] args) {  
    criarObjetos();  
  
    EntityManager entityManager = JPAUtil.criarEntityManager();  
    ProdutoDao produtoDao = new ProdutoDao(entityManager);  
  
    produtoDao.buscarPorNome("Nokia").stream().forEach(p -> System.out.println(p.getNome() +  
        ", " + p.getPreco() + ", " + p.getDescricao()));  
}  
  
Nokia, 10000.00, Velho
```

Joins são bem simples também, é como chamar um **objeto dentro de outro**, basta **referencia-lo** que o **Hibernate** se vira para montar a query:

```

public List<Produto> buscarPorNomeCategoria(String nome){
    String jpql = "FROM Produto p WHERE p.categoria.nome = :nome";

    return entityManager.createQuery(jpql, Produto.class)
        .setParameter(s: "nome", nome)
        .getResultList();
}

public static void main(String[] args) {
    criarObjetos();

    EntityManager entityManager = JPAUtil.criarEntityManager();
    ProdutoDao produtoDao = new ProdutoDao(entityManager);

    produtos.buscarPorNomeCategoria("Eletrodoméstico").stream().forEach(p -> System.out.println(p.getNome() +
        ", " + p.getPreco() + ", " + p.getDescricao()));
}

```

Hibernate:

```

select
    produto0_.id as id1_1_,
    produto0_.categoria_id as categoria6_1_,
    produto0_.dataCadastro as datacada2_1_,
    produto0_.descricao as descricao3_1_,
    produto0_.nome as nome4_1_,
    produto0_.preco as preco5_1_
from
    produtos produto0_ cross
join
    categorias categoria1_
where
    produto0_.categoria_id=categoria1_.id
    and categoria1_.nome=?

```

Obviamente **também é possível limitar quais colunas do registro eu quero buscar**, muitas vezes **não faz sentido carregar toda a entidade**. Também é muito simples fazer isso, igual **SQL** só que usando a classe como referência:

```

public BigDecimal buscarPrecoPorNomeProduto(String nome){
    String jpql = "SELECT p.preco FROM Produto p WHERE p.nome = :nome";

    return entityManager.createQuery(jpql, BigDecimal.class)
        .setParameter(s: "nome", nome)
        .getSingleResult();
}

public static void main(String[] args) {
    criarObjetos();

    EntityManager entityManager = JPAUtil.criarEntityManager();
    ProdutoDao produtoDao = new ProdutoDao(entityManager);

    System.out.println(produtoDao.buscarPrecoPorNomeProduto("Geladeira gamer"));
}

```

Hibernate:

```

select
    produto0_.preco as col_0_0_
from
    produtos produto0_
where
    produto0_.nome=?
20000.00

```