Mapeamento Lógico/Relacional com JPA

Elaine Quintino da Silva
Doutora em Ciência da Computação pelo ICMC-USP/São Carlos
Analista de Sistemas UOL – PagSeguro
Instrutora Globalcode

pagseguro UMA EMPRESA UOL

Globalcode

Agenda

- Persistência de dados
- JPA
- Como usar JPA
- Relacionamentos
- Conclusões

Agenda

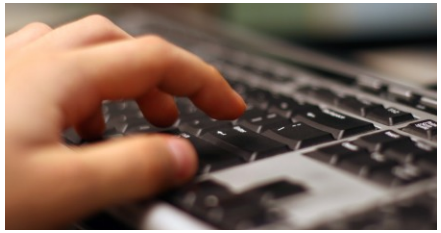
TDC2012
the developer's conference

- Persistência de dados
- JPA
- Como usar JPA
- Relacionamentos
- Conclusões

pagseguro
UMA EMPRESA UOL

Globalcode

Persistência de dados



- Tabelas
- Colunas
- Registros
- Chaves

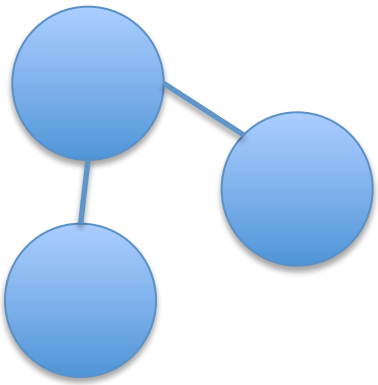
- **Arquivos**
- **Bancos de dados Hierárquicos**
- **Bancos de dados Relacionais**
- **Bancos de dados Orientados a Objetos**
- **Modelos mais recentes (NoSQL)...**

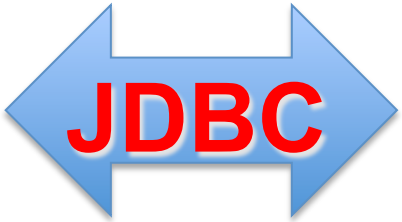
Ex.: SQL Server, Oracle, MySQL, Postgresql,...

Persistência de dados

TDC2012
the developer's conference

Programação Orientada a Objetos | **Banco de dados relacional**





Persistência de dados

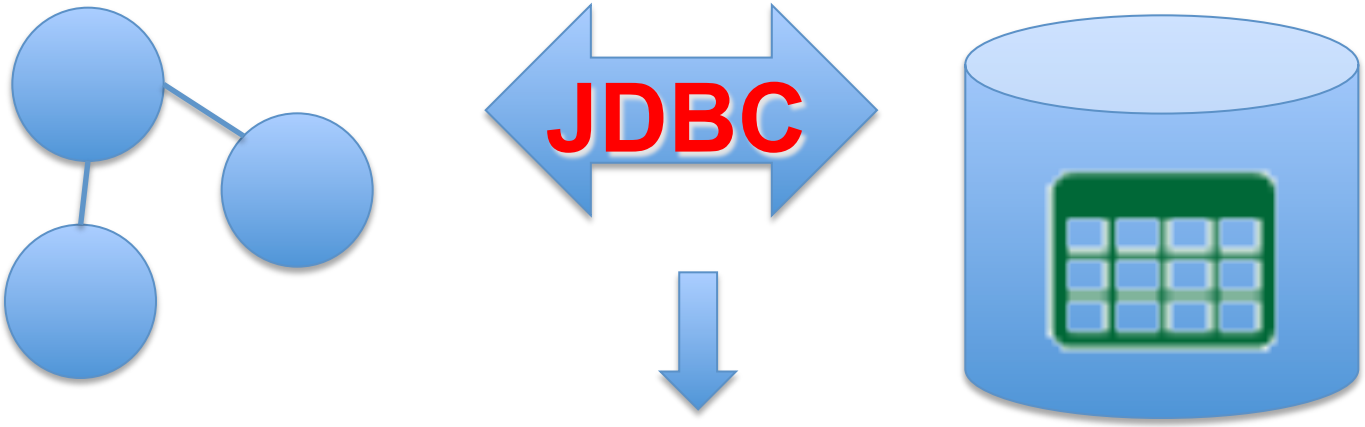
Programação Orientada a Objetos | **Banco de dados relacional**

```
public void save(Produto produto){
   String sql = "INSERT INTO produtos (nome ,preco) VALUES ('"
                + produto.getNome() + "', " + produto.getPreco() + ")";
   Connection conn = null;
   Statement stmt = null;
   try {
      Class.forName("org.gjt.mm.mysql.Driver");
      conn = DriverManager.getConnection("jdbc:mysql://host/bd",
             "us", "ps");
      stmt = conn.createStatement();
      stmt.executeUpdate(sql);
    } catch (ClassNotFoundException e) {
       //...
    } catch (SQLException e) {
       //...
    }finally {//...}
```

Persistência de dados

TDC2012 the developer's conference

Programação Orientada a Objetos | **Banco de dados relacional**



Soluções de ORM (Object Relational Mapping):
•Hibernate, KODO, JDO, TopLink, EJB EntityBean...

JPA

Agenda

- Persistência de dados
- JPA
- Como usar JPA
- Relacionamentos
- Conclusões

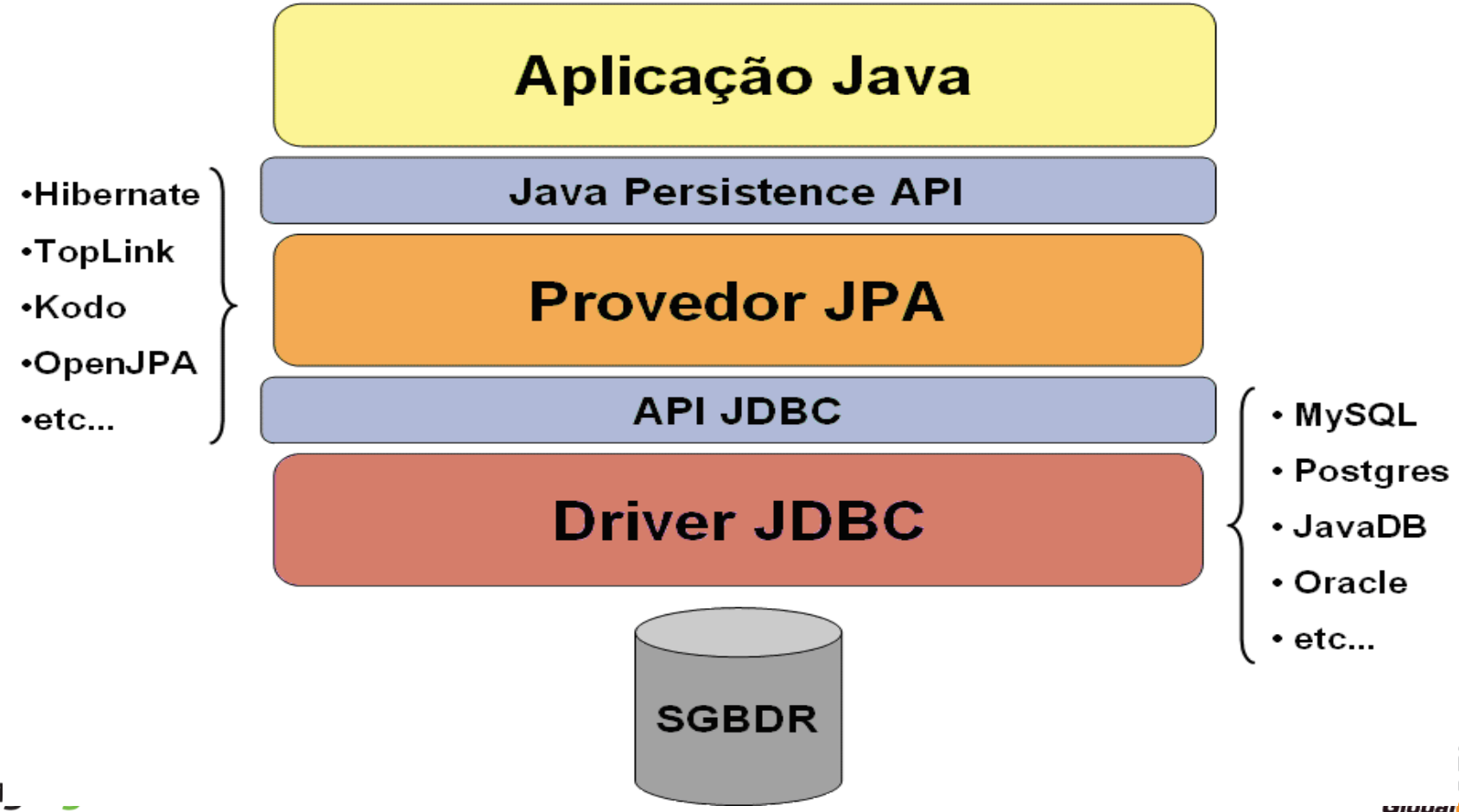
JPA – Java Persistence API

- API para abstração da camada de persistência das aplicações orientadas a objetos
  - v1.0 – 2006 / v2.0 – 2009
  - Integração com diversos frameworks (não pode ser usado isoladamente)
- Mapeamento objeto / relacional
  - Classe para tabela
  - Atributo para coluna
  - Relacionamento para chave estrangeira
- API para salvar, atualizar, excluir os objetos do banco

JPA – Java Persistence API

TDC2012 the developer's conference

- JPA depende de JDBC e de um framework de persistência



Agenda

TDC2012
the developer's conference

- Persistência de dados
- JPA
- Como usar JPA
- Relacionamentos
- Conclusões

Como usar JPA

TDC2012 the developer's conference

- Configurar bibliotecas no projeto
  - JPA + Framework + Driver SQL
- Mapear as classes para as devidas tabelas/colunas e os relacionamentos para as chaves estrangeiras
  - Anotações Java
- Configurar uma unidade de persistência
  - persistence.xml
- Usar as classes da API JPA para gerenciar os objetos no banco de dados

Como usar JPA (1º)

TDC2012 the developer's conference

- Configurando as bibliotecas:

```
<dependency>
    <groupId>org.hibernate.javax.persistence</groupId>
    <artifactId>hibernate-jpa-2.0-api</artifactId>
    <version>1.0.1.Final</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>4.0.0.Final</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.21</version>
    <scope>compile</scope>
</dependency>
```

Como usar JPA (2º)

TDC2012 the developer's conference

- Mapeando a entidade de persistência

```
@Entity @Table(name = "produtos")
public class Produto implements Serializable {

        @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
        protected Integer id;
        @Column(name="nome_produto")
        protected String nome;
        protected double preco;
        protected String marca;

        public Produto() {}
```

Como usar JPA (2º)

TDC2012 the developer's conference

- Anotações para mapeamento de tabelas e colunas:

  @Entity

  @Table (name="")

  @Column(name="")

  @Id

  @GeneratedValue(strategy=GenerationType.X)

  - X → IDENTITY, SEQUENCE, TABLE, AUTO

  @Temporal(TemporalType.X)

  - X → TIMESTAMP, TIME, DATE

  @Transient

Como usar JPA (3º)

TDC2012 the developer's conference

- Configurando a unidade de persistência
  - *Persistence Unit*
    - Qual é o banco de dados ?
    - Quais são as classes persistentes ?
    - Como as transações são gerenciadas ?, etc
- Configurada no arquivo persistence.xml
  - Deve estar no diretório META-INF na raiz da estrutura de pacotes das classes persistentes.

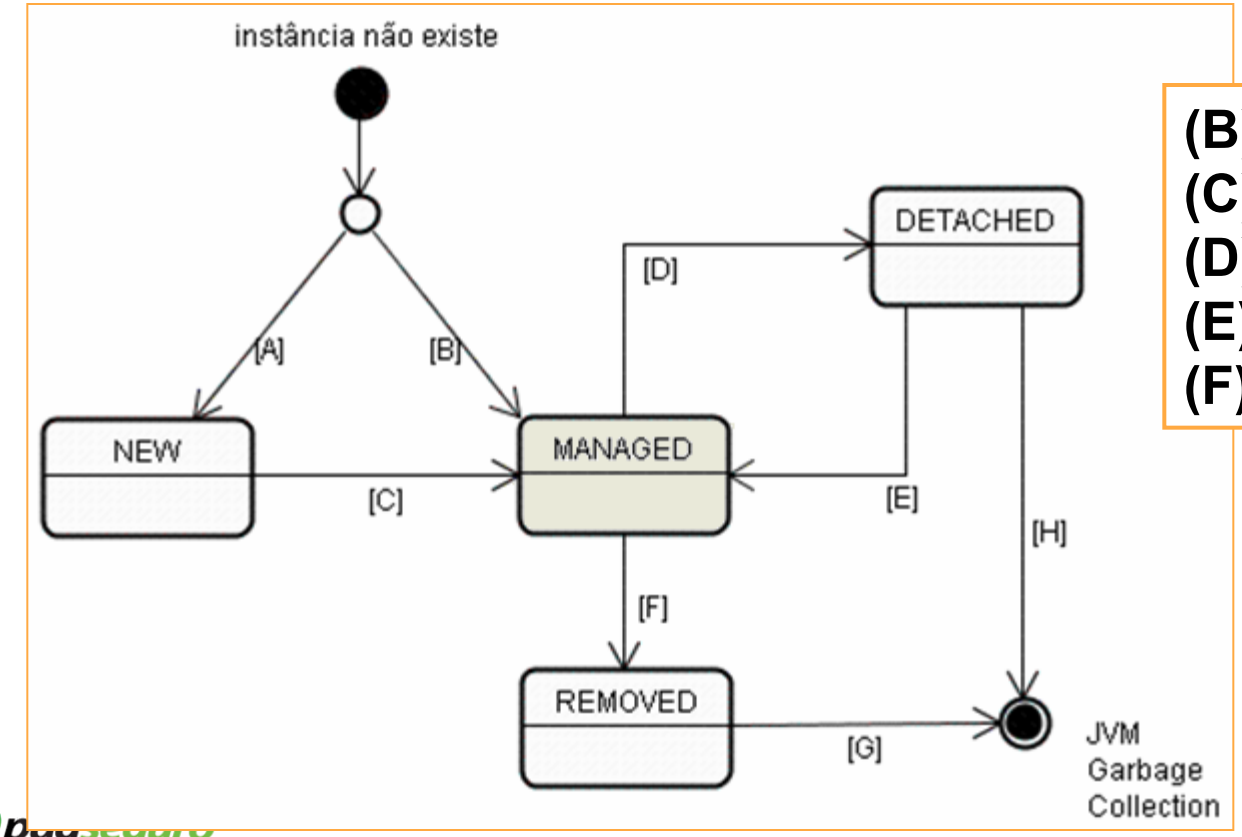
Como usar JPA

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" ...">
<persistence-unit name="JPA20" transaction-type="RESOURCE_LOCAL">
  <class>br.com.exemplo1.jpa.Produto</class>

  <properties>
   <property name="javax.persistence.jdbc.url"
                              value="jdbc:mysql://host/db"/>
    <property name="javax.persistence.jdbc.user" value="user"/>
    <property name="javax.persistence.jdbc.driver"
                                 value="com.mysql.jdbc.Driver"/>
    <property name="javax.persistence.jdbc.password" value="senha"/>

    <property name="hibernate.hbm2ddl.auto" value="create"/>
    <property name="hibernate.show_sql" value="true"/>
    <property name="hibernate.format-sql" value="true"/>
  </properties>

 </persistence-unit>
</persistence>
```

Como usar JPA (4º)

TDC2012 the developer's conference

- *EntityManager* *(javax.persistence)*
  - Métodos para gerenciar os objetos no banco de dados



(B) find / query
(C) persist
(D) clear
(E) merge
(F) remove

Como usar JPA (4º)

TDC2012 the developer's conference

- *EntityManagerFactory* *(javax.persistence)*
  - Fábrica de EntityManagers
  - Apenas uma instância por aplicação
    - alto custo de inicialização
      - Técnicas de inicialização serão vistas mais a diante
  - Definida em uma unidade de persistência

Como usar JPA (4º)

TDC2012 the developer's conference

```
Import javax.persistence.*;

public class App {
   public static void main(String[] args) {
      EntityManagerFactory emf =
             Persistence.createEntityManagerFactory("JPA20");
      EntityManager em = emf.createEntityManager();

       Produto p1 = new Produto("LG", "TV 5\'", 2000.0);
       em.getTransaction().begin();
       em.persist(p1);
       em.getTransaction().commit();
       em.close();
       emf.close();
  }
}
```

Agenda

- Persistência de dados
- JPA
- Como usar JPA
- Relacionamentos
- Conclusões

Relacionamentos

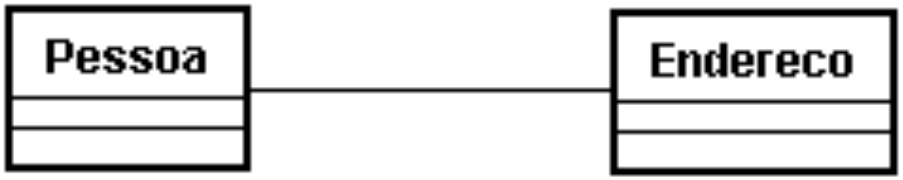
TDC2012 the developer's conference

- JPA permite mapear os relacionamentos entre as classes para tabelas
  - Associações
    - @OneToOne, @OneToMany, @ManyToOne
  - Herança
    - @Inheritance
    - Estratégias:
      - Única tabela
      - Tabela por subclasse
      - Tabela por classe concreta

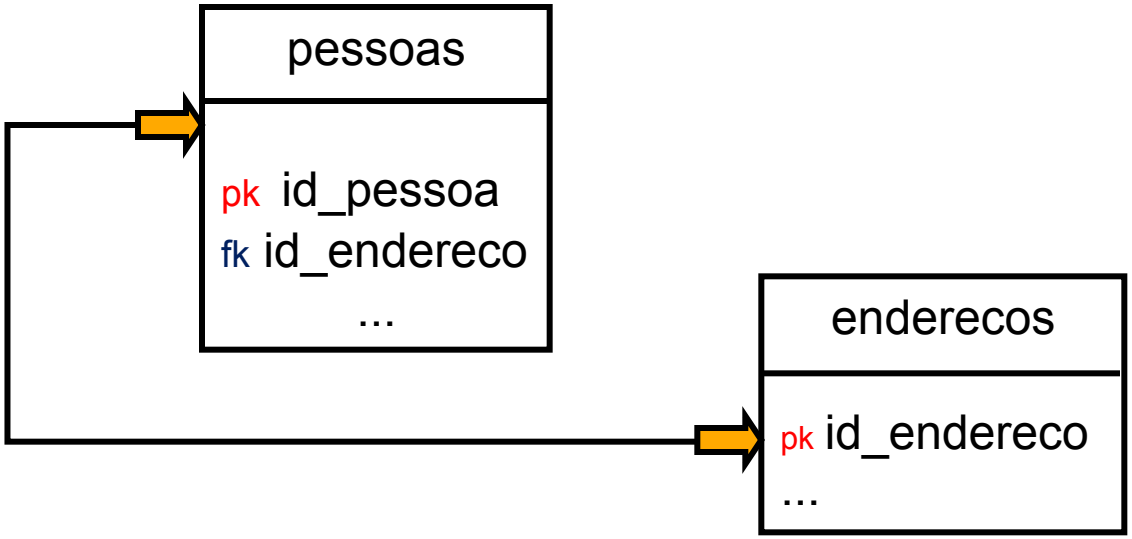
Relacionamentos

TDC2012 the developer's conference

- OneToOne
  - Diagrama de classes:



  - Tabelas:



Relacionamentos

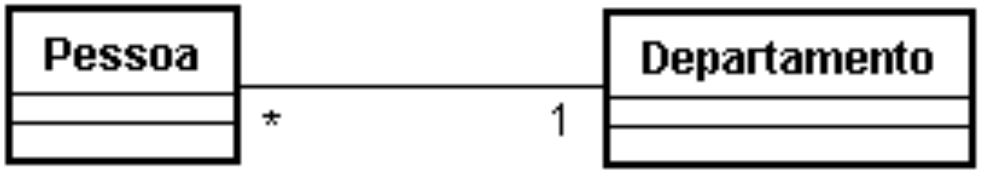
TDC2012 the developer's conference

```
@Entity @Table(name="pessoas")
class Pessoa implements Serializable{
    @OneToOne (cascade=CascadeType.ALL)
    @JoinColumn(name="id_endereco")
    private Endereco end;
    …
}
@Entity @Table(name="enderecos")
class Endereco implements Serializable {
    @OneToOne (mappedBy="end")
    private Pessoa pessoa;
    …
}
```

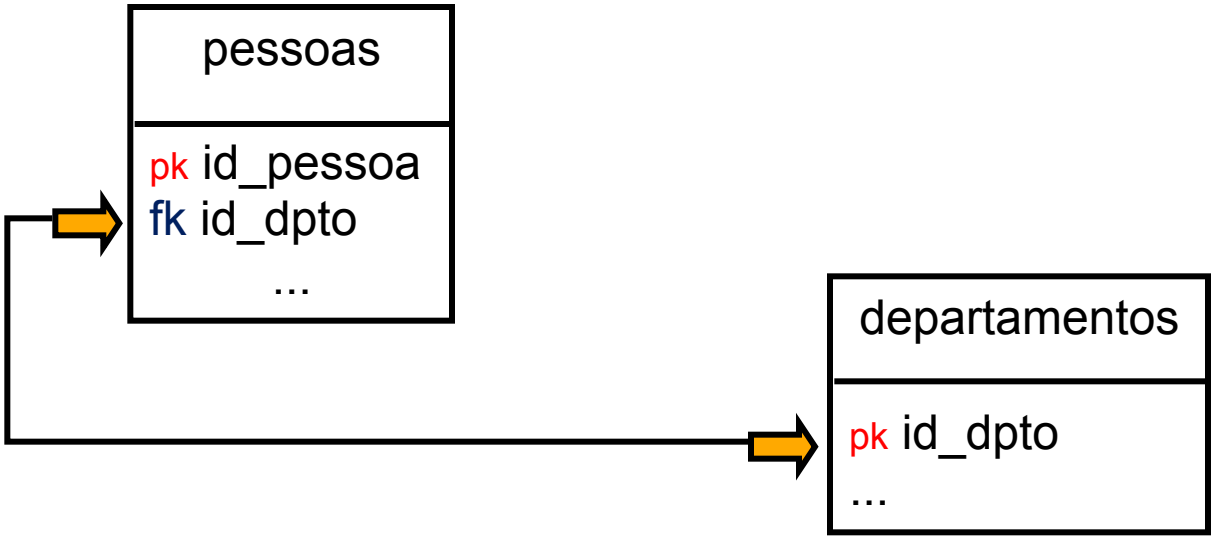
Relacionamentos

TDC2012 the developer's conference

- ManyToOne e OneToMany
  - Diagrama de classes:



  - Tabelas:



Relacionamentos

TDC2012 the developer's conference

```
@Entity @Table(name="pessoas")
class Pessoa implements Serializable{
    @ManyToOne @JoinColumn(name="id_departamento")
    private Departamento depto;
    …
}
@Entity @Table(name="departamentos")
class Departamento implements Serializable {
    @OneToMany (mappedBy="depto",
                targetEntity=Pessoa.class,
                cascade=CascadeType.ALL)
    private Collection pessoas;
    …
}
```

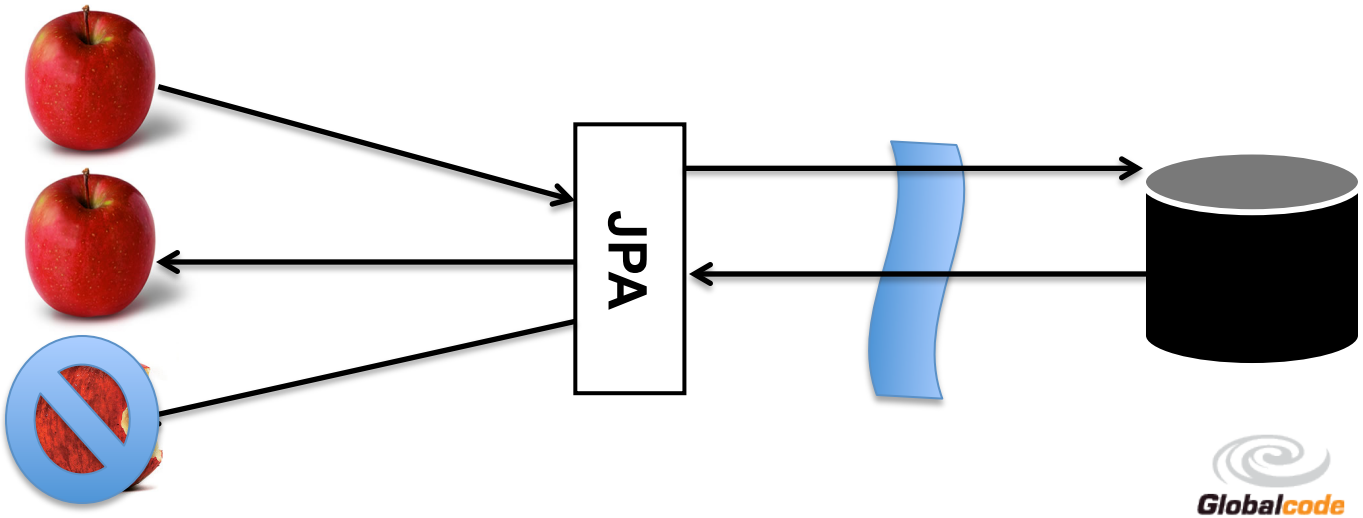
Agenda

- Persistência de dados
- JPA
- Como usar JPA
- Relacionamentos
- Conclusões

Conclusões

TDC2012 the developer's conference

- Solução padrão amplamente utilizada
- Versão 2.0 bastante robusta com grande parte dos recursos oferecidos pelos frameworks mais comuns
- Redução do GAP entre OO e BD relacional
- Transparência do mecanismo de persistência que está sendo utilizado



Dúvidas?

elaineqsilva@gmail.com