

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus João Pessoa

Persistência de Objetos

Fausto Maranhão Ayres

6 Persistência com JPA: Básico

Revista SQL Magazine nº40



2007

INTRODUÇÃO

fausto.ayres@ifpb.edu.br

3

JPA – Java Persistence API

- É a especificação (documento) padrão do Java para:
 - **Mapeamento objeto-relacional:** classe \Leftrightarrow tabela
 - **Gerenciamento do CRUD** para banco relacional
- Grande novidade:
 - uso em **Java EE** e **Java SE**
- Versões
 - 1.0 (mai/**2006**),
 - 2.0 (dez/2009),
 - 2.1 (mai/2013),
 - 2.2 (jun/2018) – inclui novidades java8

fausto.ayres@ifpb.edu.br

4

Provedores JPA

- **Oracle TopLink Essentials**
 - 1º provedor oficial para JPA (2007)
- **EclipseLink**
 - Sucessor do Toplink Essentials
<https://www.eclipse.org/eclipselink/>
- **JBoss Hibernate**
 - Migrou para o padrão JPA
 - <http://hibernate.org/orm/>
- **DataNucleus JPOX**
 - <https://en.wikipedia.org/wiki/DataNucleus>

fausto.ayres@ifpb.edu.br

5

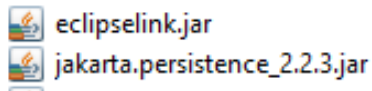
Provedores de outras Linguagens

- **PHP**
 - Doctrine
 - Laravel
 - Zend
- **C#**
 - NPersistence API
 - Entity Framework
- **Python**
 - Django
 - SQLAlchemy

6

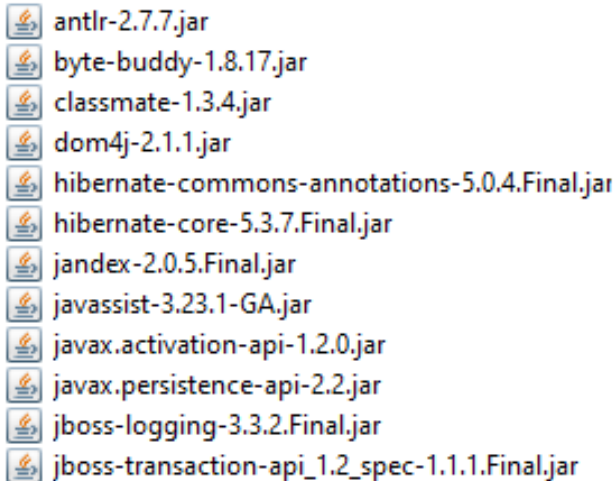
Download - Provedores JPA

Eclipselink



Faremos também o download através do arquivo de configuração do Maven (pom.xml)

Hibernate



fausto.ayres@ifpb.edu.br

7

Tópicos JPA

- Mapeamento de:
 - Atributos
 - Herança
 - Relacionamentos
 - Chaves
 - Etc
- DAO JPA
- Linguagem de consulta JPQL
- Ciclo de vida dos objetos
- Eventos (trigger)
- Controle de Concorrência (locking)

fausto.ayres@ifpb.edu.br

8

MAPEAMENTO OBJETO-RELACIONAL (BÁSICO)

fausto.ayres@ifpb.edu.br

9

Diferenças entre os modelos OO e Rel.

Java	BD Relacional
classe	tabela
Tipos de dados complexos (List, Map)	Tipos de dados simples (int, varchar,...)
Navegação de ponteiros	Navegação através de chaves estrangeiras
Referências bidirecionais (pai→filho e filho → pai)	Referências Unidirecionais (filho→pai)
Relacionamento N:N	Não tem
Herança	Não tem

Mapeamento Objeto-Relacional

- O **Mapeamento Objeto-Relacional** - MOR (ORM) é feito de forma explícita com uso de **anotações** (@) antes da classe ou dos atributos
- Os atributos não anotados receberão **mapeamento implícito** (default)
- As anotações estão no pacote:

```
import javax.persistence.*;
```


fausto.ayres@ifpb.edu.br

11

Anotações obrigatórias

@Entity sinaliza que a classe é persistente

@Id identifica a chave primária



```
@Entity  
public class Pessoa {  
    @Id  
    private int id;  
    private String nome;  
}
```

Resultado do mapeamento

```
create table pessoa(  
    id integer primary key,  
    nome varchar(255)  
)
```

Restrição: Os atributos não podem ser **public**

fausto.ayres@ifpb.edu.br

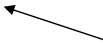
12

Construtor Vazio obrigatório

- Ele é **obrigatório**, pois será usado pelo manager para criar os objetos após serem lidos.
- Podem existir outros construtores além deste.

```
@Entity
public class Pessoa {
    @Id
    private int id;
    private String nome;

    public Pessoa () {}
    ...
}
```



fausto.ayres@ifpb.edu.br

13

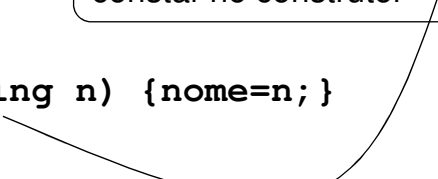
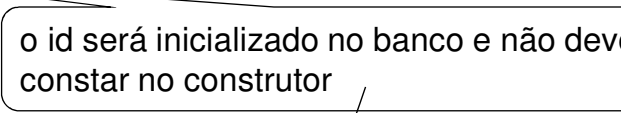
Anotação para Autonumeração

@GeneratedValue especifica uma coluna autoincrementada automaticamente pelo banco

```
@Entity
public class Pessoa {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int id;

    private String nome;

    public Pessoa(String n) {nome=n;}
    ...
}
```



fausto.ayres@ifpb.edu.br

14

Mapeamento sem anotações

- Antes de surgir as anotações, o mapeamento era feito dentro de um arquivo *xml*:

```
<entity class="Pessoa" name="Pessoa">
```

```
<table name="PESSOAS"/>
```

```
<attributes>
```

```
  <id name="id"> <generated-value strategy="TABLE"/> </id>
```

```
  <basic name="nome"> <column name="NOMEPESSOA" length="100"/> </basic>
```

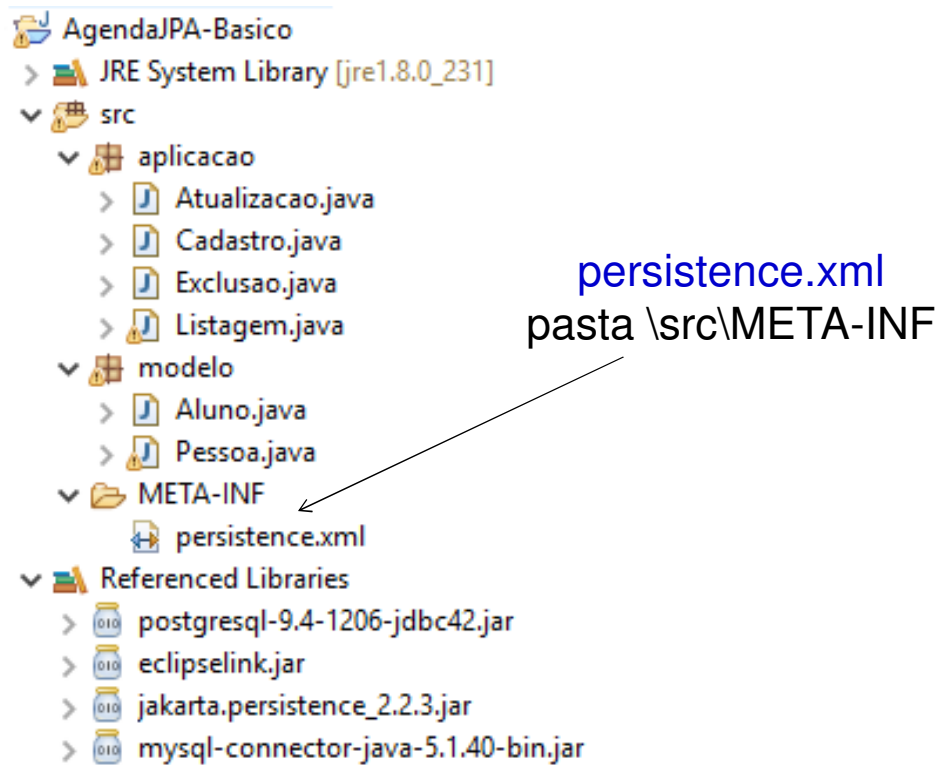
```
  <basic name="idade"> </basic>
```

```
</attributes>
```

```
</entity>
```

CONFIGURAÇÃO DA PERSISTÊNCIA

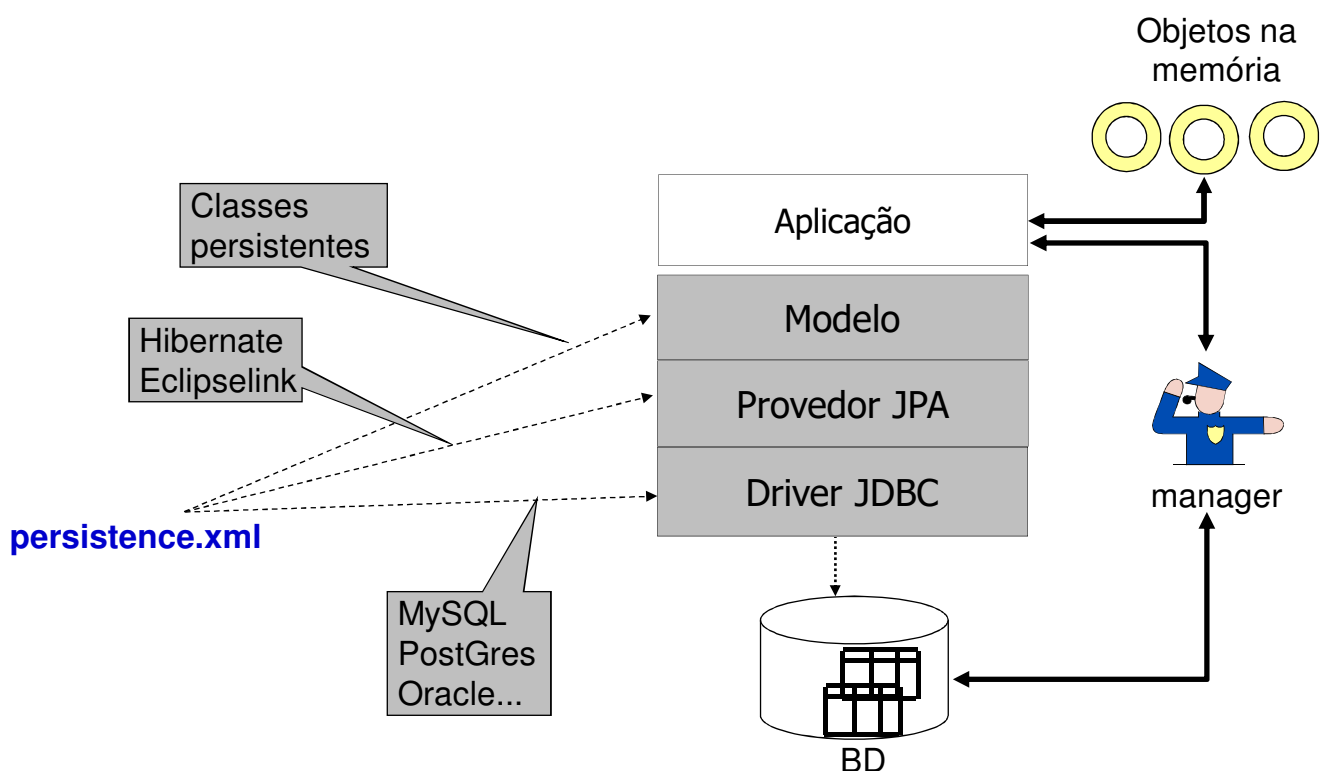
Projeto Agenda (sem DAO)



fausto.ayres@ifpb.edu.br

17

Esquema de configuração

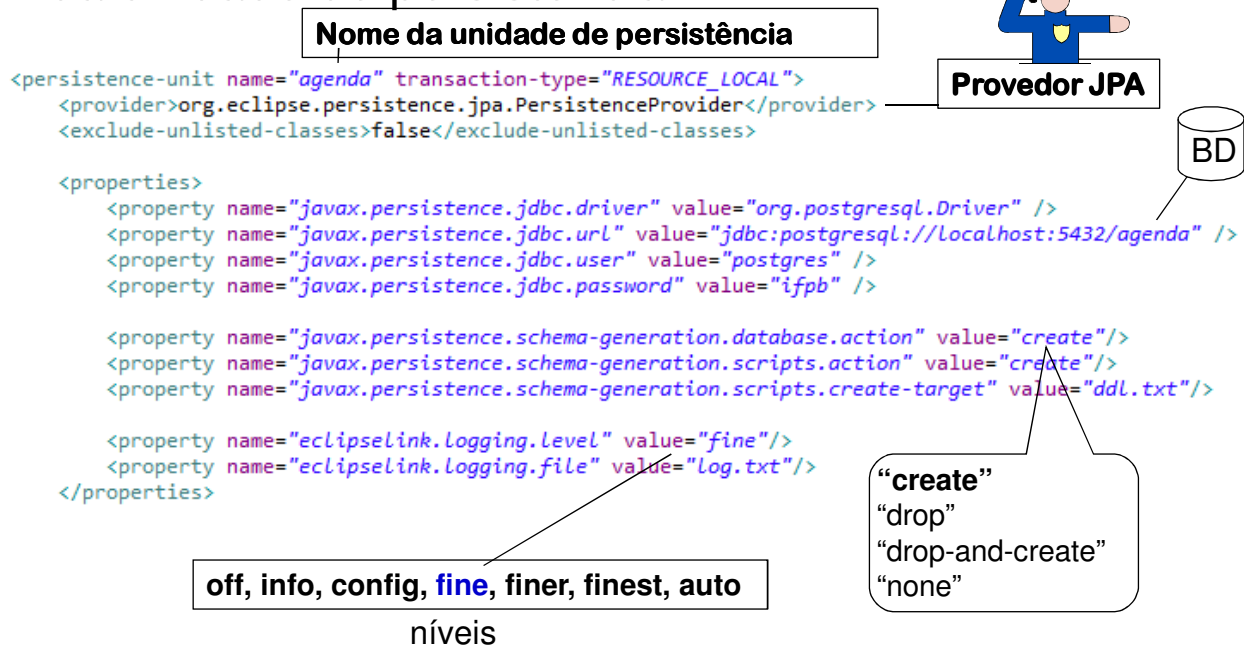


fausto.ayres@ifpb.edu.br

18

persistence.xml

- É processado na criação do manager pelo nome da unidade de persistência



fausto.ayres@ifpb.edu.br

19

Obs

- Algumas configurações do persistence.xml podem ser modificadas **em tempo de execução**, durante a criação do manager, como veremos em outro momento

fausto.ayres@ifpb.edu.br

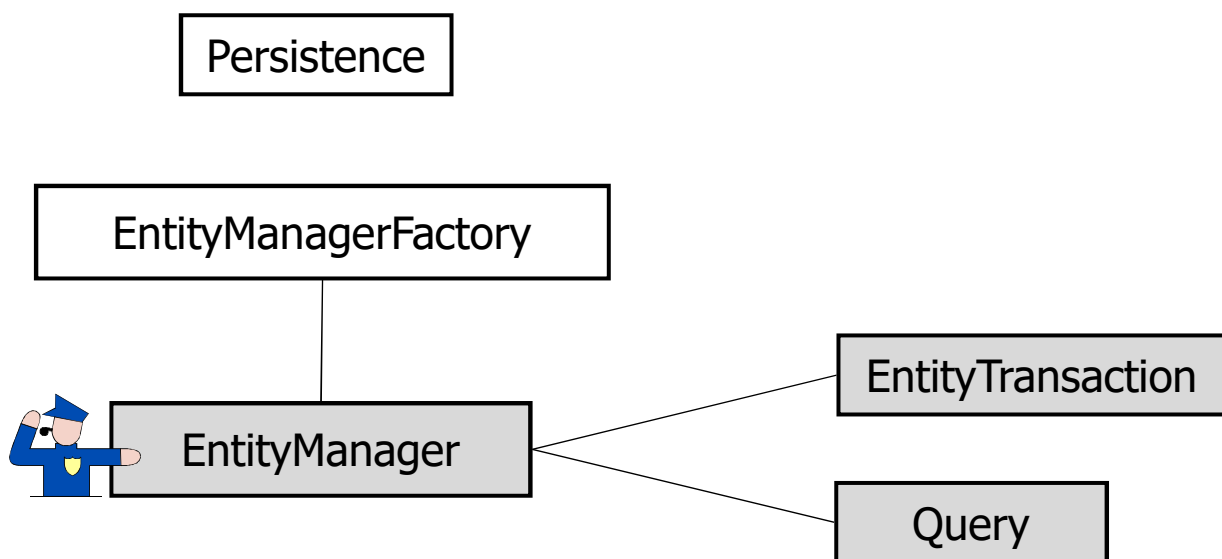
20

API BÁSICA

fausto.ayres@ifpb.edu.br

21

Classes / Interfaces

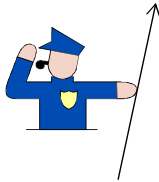


fausto.ayres@ifpb.edu.br

22

Criação do *manager*

```
EntityManagerFactory factory;  
EntityManager manager;  
  
factory= Persistence.createEntityManagerFactory("agenda");  
manager = factory.createEntityManager();  
  
...  
  
manager.close();  
factory.close();  
}}
```



Nome da unidade de persistência
dentro do arquivo persistence.xml

fausto.ayres@ifpb.edu.br

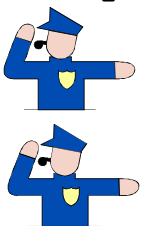
23

OBS:

- É possível instanciar vários managers usando diferentes *unidades de persistência*

```
EntityManagerFactory factory;  
EntityManager manager;  
  
factory1=Persistence.createEntityManagerFactory("compra");  
factory2=Persistence.createEntityManagerFactory("venda");  
manager1=factory.createEntityManager();  
manager2=factory.createEntityManager();  
  
...  
  
manager1.close();  
manager2.close();
```

Unidades de Persistência



fausto.ayres@ifpb.edu.br

24

Classe EntityManager

Principais métodos



```
void      persist(Object entidade)
T         find    (Class<T> classe, Object chave)
T         merge   (T entidade)
void      refresh(Object entidade)
void      remove  (Object entidade)
EntityTransaction getTransaction()
void      flush()
void      clear()
Query     createQuery(String consultaJPQL)
Query     createNamedQuery(String id)
Boolean   contains(Object entidade)
Boolean   isOpen()
```

...

fausto.ayres@ifpb.edu.br

25

persist()

- Gravar objetos persistentes em uma transação

```
manager.getTransaction().begin();
```

```
Pessoa p1 = new Pessoa("joao");
Pessoa p2 = new Pessoa("maria");
manager.persist(p1);
manager.persist(p2);
```

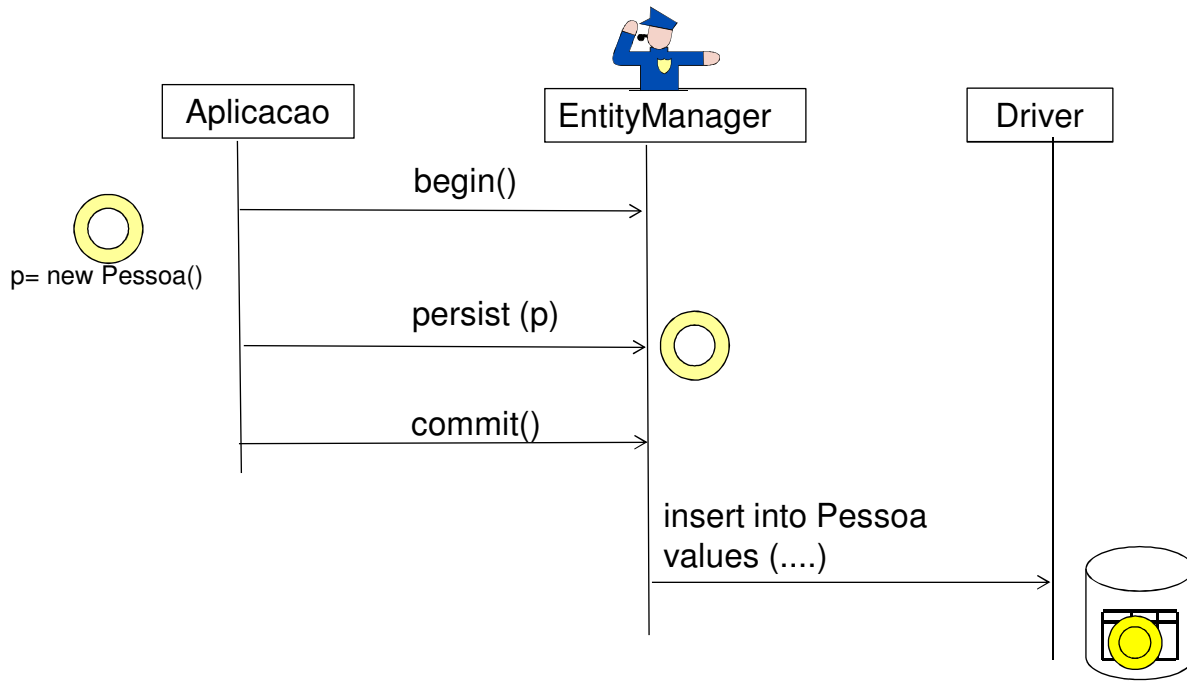
```
manager.getTransaction().commit();
```

pessoa

id	nome
1	joao
2	maria

Pode-se gravar um objeto por transação

Diagrama de Sequencia



fausto.ayres@ifpb.edu.br

27

find()

- Localizar um objeto no banco, usando a chave primaria (id)

```
Pessoa p = manager.find(Pessoa.class, 1);
```

Obs:

Pode-se localizar o objeto, usando uma query JPQL

```
select p from Pessoa p where p.id = 1
```

fausto.ayres@ifpb.edu.br

28

merge()

- Atualizar objeto no banco

```
manager.getTransaction().begin();
Pessoa p = manager.find(Pessoa.class, 1);
if (p!=null) {
    p.setNome("joana");
    p = manager.merge(p);
    manager.getTransaction().commit();
}
else {
    manager.getTransaction().rollback();
    throw new Exception("inexistente");
}
```

pessoa

id	nome
1	joana
2	maria

remove()

- Remover objeto do banco

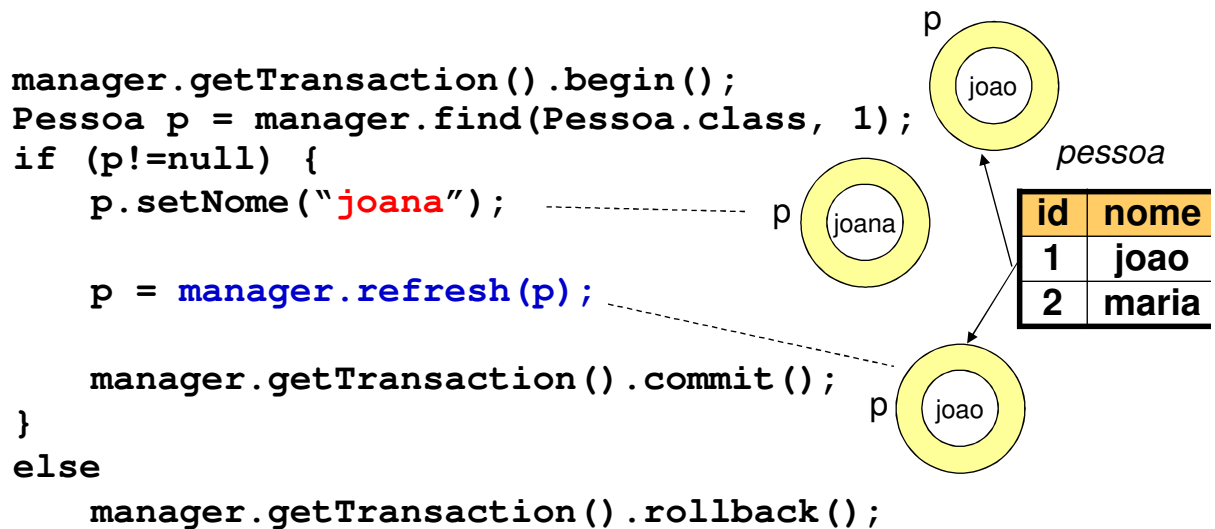
```
manager.getTransaction().begin();
Pessoa p = manager.find(Pessoa.class, 1);
if (p!=null) {
    manager.remove(p);
    manager.getTransaction().commit();
}
else {
    manager.getTransaction().rollback();
    throw new Exception("inexistente");
}
```

pessoa

id	nome
2	maria

refresh()

- Rer ler objeto do banco (descarta alteração feitas na memória)



Limpeza do Cache de objetos (primário)

- Os objetos são mantidos num *cache* após a primeira leitura do bd. As demais leituras vem do cache

```
Pessoa p = manager.find(Pessoa.class, 1); //bd
Pessoa p = manager.find(Pessoa.class, 1); //cache
```

- **clear()** limpa o *cache* de objetos, forçando releitura dos objetos do bd.

```
Pessoa p = manager.find(Pessoa.class, 1); //bd
manager.clear();
Pessoa p = manager.find(Pessoa.class, 1); //bd
```


JPA Query Language (JPQL)

- É uma linguagem declarativa orientada a objetos

```
SELECT objeto  
FROM Classe  
WHERE condição
```

- As consultas são convertidas pelo manager para SQL e enviadas ao SGBD via JDBC

Exemplo de consulta JPQL

```
Query query = manager.createQuery  
(  
    "select p from Pessoa p where p.nome='joao' "  
);
```

objeto

Classe
Java

```
Pessoa p = query.getSingleResult();  
System.out.println(p);
```