

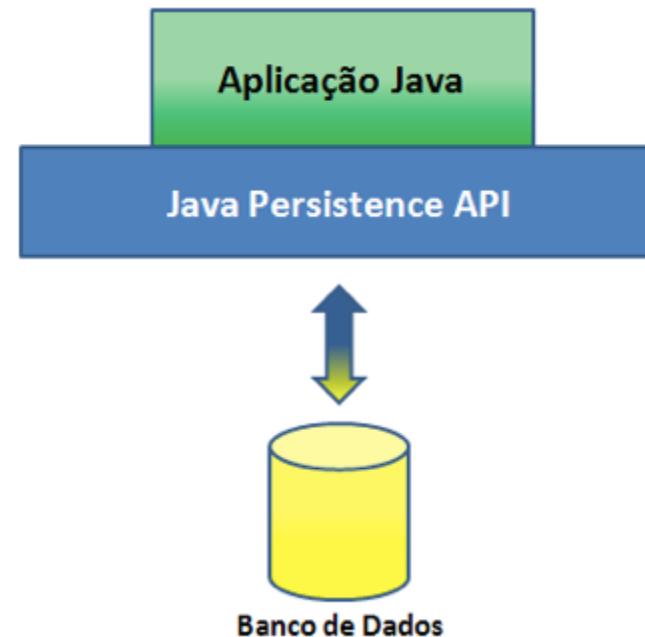
# Linguagem de Programação Orientada a Objetos 2

Java Persistence API – JPA – Parte 1

Prof. Tales Viegas

# Introdução

- ▶ Java Persistence API (JPA) é um framework para a camada de persistência dos dados, que fornece uma camada de comunicação entre a aplicação escrita em Java e uma base de dados

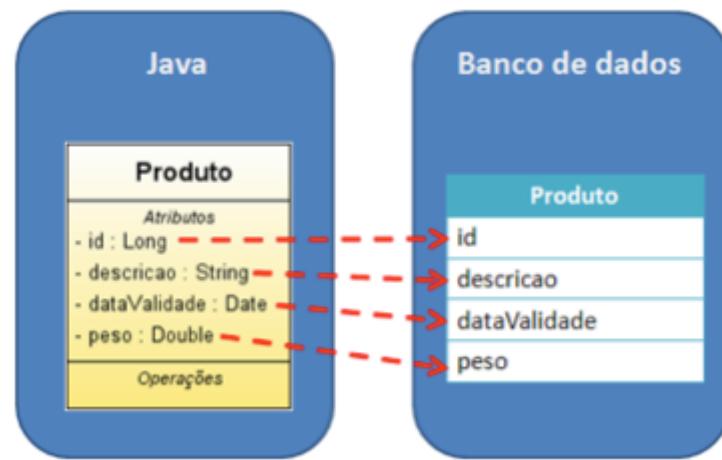


# Introdução

- ▶ Algumas funcionalidades são:
  - Conversão de Registros de Dados para Objetos Java
  - Não precisa criar códigos SQL para salvar, alterar ou remover registros do banco de dados
  - A aplicação não fica presa a um banco de dados

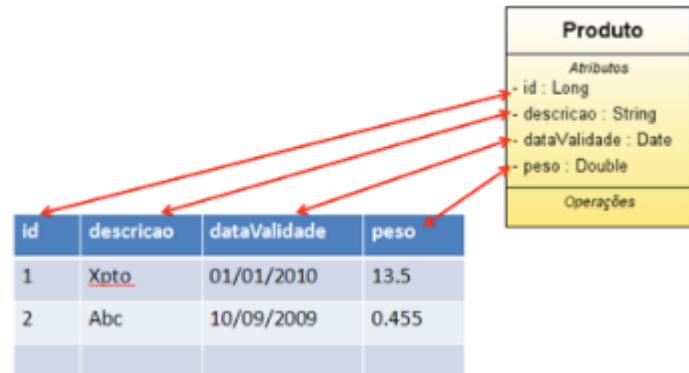
# Mapeamento Objeto–Relacional

- ▶ JPA utiliza o conceito de ORM (Object/Relational Mapping) para fazer a ponte entre uma base de dados relacional e os objetos em Java



# Mapeamento Objeto–Relacional

- ▶ O JPA cria uma instância da classe Produto para cada registro da tabela Produto, atribuindo os valores dos atributos de acordo com os valores da tabela.
  - ▶ Por padrão, o JPA realizará o mapeamento da classe e atributos com o mesmo nome



xpto : Produto
Atributos
id : 1
descricao : Xpto
dataValidade : 01/01/2010
peso : 13,5

<u>abc : Produto</u>
Atributos
id : 2
descricao : Abc
dataValidade : 10/09/2009
peso : 0.455

# Entity (Entidade)

- ▶ Uma Entity é um objeto utilizado para representar uma tabela na base de dados, sendo que cada instância da entidade corresponde a um registro da tabela
- ▶ Uma Entity é baseada em uma simples classe Java do tipo POJO (Plain Old Java Object), ou seja, uma classe com atributos e os métodos get/set de cada um deles

# Notações Entity

- ▶ **@Id**
  - Representa a chave primária
- ▶ **@Column**
  - Representa uma coluna
  - Propriedades
    - nullable
    - name
    - precision
    - unique

# Notações Entity

- ▶ @GeneratedValue
  - @SequenceGenerator ou @TableGenerator
  - strategy: GenerationType AUTO, IDENTITY, SEQUENCE ou TABLE
- ▶ @Temporal
  - Representação de data/hora
  - value: DATE, TIME ou TIMESTAMP

# Exemplo

```
CREATE TABLE Usuario (
    id NUMBER(10) NOT NULL PRIMARY KEY,
    nome VARCHAR2(100) NOT NULL,
    dataNasc DATE NOT NULL,
    email VARCHAR2(150) NOT NULL,
    ativo NUMBER(1) NOT NULL,
    comentario VARCHAR2(200) );
```

# Exemplo

```
public class Usuario implements Serializable {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long id;  
  
    @Column(nullable = false)  
    private String nome;  
  
    @Temporal(TemporalType.DATE) @Column(name="dataNasc", nullable = false)  
  
    private Date dataNascimento;  
    @Column(nullable = false)  
    private String email;  
  
    @Column(nullable = false)  
    private Boolean ativo;  
    private String comentario;
```

# EntityManager

- ▶ Serviço responsável por gerenciar as entidades
- ▶ Através dele é possível gerenciar o ciclo de vida, operações de sincronização, consulta de entidades, entre outras

# Unidade de Persistência

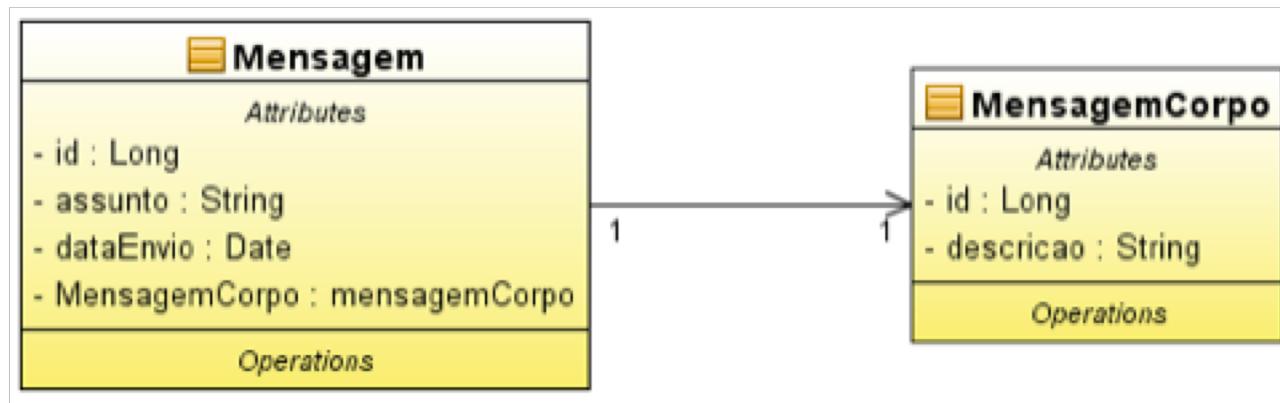
- ▶ A unidade de persistência é utilizada para configurar as informações referentes ao provedor do JPA (implementação da especificação JPA) e ao banco de dados.
- ▶ No Spring para definir a unidade de persistência é necessário configurar o datasource e as configurações do Hibernate no arquivo de configuração application.yml.

# Interfaces Repository

- ▶ Interfaces utilizadas para realizar as operações entre o banco de dados e as entidades

# Relacionamentos

► 1:1



# Relacionamentos

## ► 1:1

```
@Entity
public class Mensagem implements Serializable {

    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

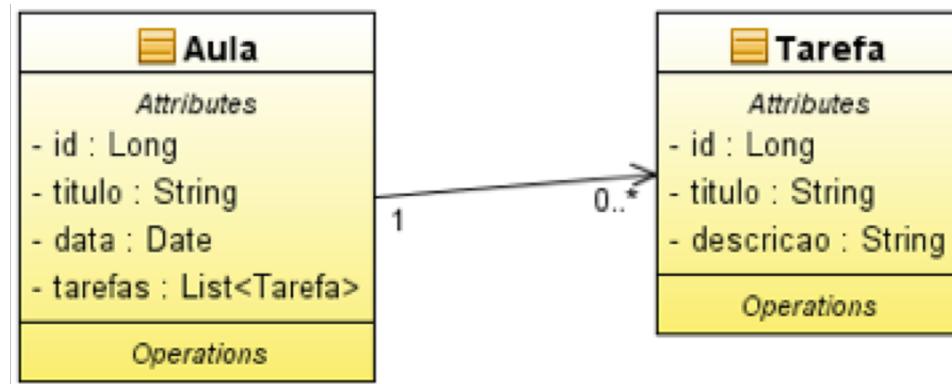
    private String assunto;    @Temporal(TemporalType.DATE)

    private Date dataEnvio;

    @OneToOne(cascade=CascadeType.ALL)
    private MensagemCorpo mensagemCorpo;
```

# Relacionamentos

- ▶ 1:n



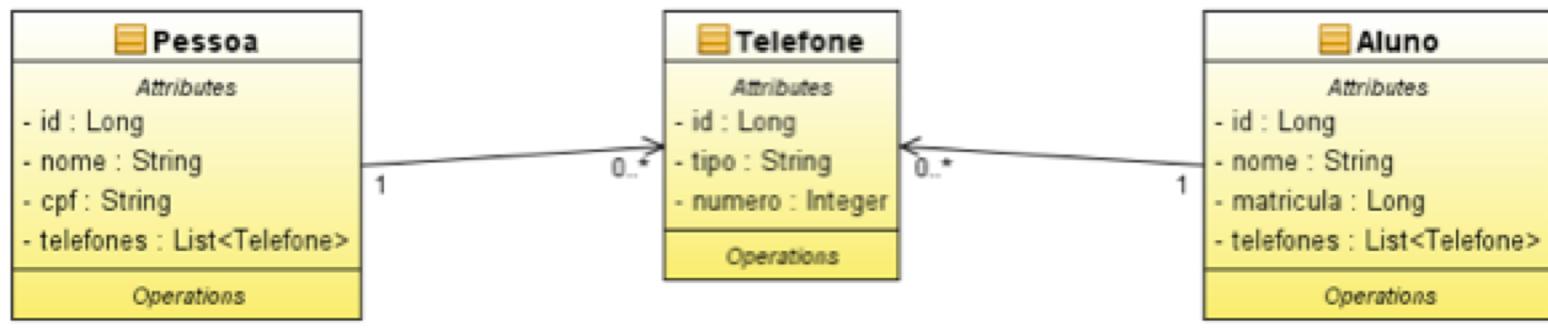
# Relacionamentos

## ► 1:n

```
@Entity  
public class Aula implements Serializable {  
  
    @Id @GeneratedValue(strategy =  
GenerationType.AUTO)  
private Long id;  
  
private String titulo;  
  
@Temporal(TemporalType.DATE)  
private Date data;  
  
@OneToMany(cascade = CascadeType.ALL)  
@JoinColumn(name="aula_id")  
private List<Tarefa> tarefas;
```

# Relacionamentos

► N:N



# Relacionamentos

```
@Entity public class Pessoa implements Serializable {  
  
    @Id @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long id;  
    private String nome;  
    private String cpf;  
  
    @OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)  
    @JoinTable(name="PESSOA_TELEFONE",  
              joinColumns={@JoinColumn(name = "PESSOA_ID")},  
              inverseJoinColumns={@JoinColumn(name = "TELEFONE_ID")})  
    private List<Telefone> telefones;
```

# CascadeType

- ▶ **PERSIST** – Quando salvar a Entidade A, também será salvo todas as Entidades B associadas.
- ▶ **MERGE** – Quando atualiza as informações da Entidade A, também será atualizado no banco de dados todas as informações das Entidades B associadas.
- ▶ **REMOVE** – Quando remover a Entidade A, também será removida todas as entidades B associadas.
- ▶ **REFRESH** – Quando houver atualização no banco de dados na Entidade A, todas as entidades B associadas serão atualizadas.
- ▶ **ALL** – Corresponde a todas as operações acima (MERGE, PERSIST, REFRESH e REMOVE)

# FetchType

- ▶ **EAGER** – Traz todas as entidades que estão relacionadas, ou seja, se a Entidade A possui um relacionamento com a Entidade B, então quando consultar a Entidade A, também será consultado suas referencias na Entidade B.
- ▶ **LAZY** – Não traz as entidades que estão relacionadas, ou seja, se a Entidade A possui um relacionamento com a Entidade B, então quando consultar a Entidade A só serão retornadas as informações referentes a esta Entidade.