



Módulo 31





BackEnd Java

Rodrigo Pires





ORM e JPA

A decorative graphic on the left side of the slide. It features a large cyan hexagon with the number '1' inside. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines.

1

ORM - Object Relational Mapper



O que é?

ORM (Object Relational Mapping) ou Mapeamento Objeto Relacional se trata de uma técnica na programação que consiste em fazer a ponte entre modelo relacional (banco de dados) e objetos (aplicação) através de frameworks capazes de converter dados entre sistemas utilizando linguagens de programação orientada a objetos





O que é?

Banco de Dados Relacional	Programação Orientada a Objetos
Tabela	Classe
Coluna	Atributo
Registro	Objeto



O que é?



OOP/POO = Programação Orientada a Objetos

ORM = Mapeamento Objeto Relacional

RDB = Banco de Dados Relacional

A decorative graphic on the left side of the slide. It features a large cyan hexagon with a white number '1' inside. Surrounding this central hexagon are several smaller hexagons in various shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, a gear, and a speech bubble. There is also a small network-like icon with a central node and several connecting lines.

1

JPA - Java Persistence API




O que é?

JPA (ou Java Persistence API) é uma especificação oficial que descreve como deve ser o comportamento dos frameworks de persistência Java que desejarem implementá-la.

Ser uma especificação significa que a JPA não possui código que possa ser executado.

Você pode pensar na especificação JPA como uma interface que possui algumas assinaturas, mas que precisa que uma classe a implemente.






O que é?

Implementação é algo que pode ser executado em nossa aplicação.

Qualquer pessoa ou equipe pode escrever sua própria implementação da especificação JPA.

Dentre as mais famosas temos o OpenJPA da Apache, o Hibernate da Red Hat e o EclipseLink da Eclipse Foundation.





O que é?

A grande ideia da especificação JPA é que a aplicação possa trocar de implementação sem que precise de mudanças no código. Apenas um pouco de configuração.



A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-teal gradient, containing the white number '2'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and teal. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and connecting lines.

2

JPA e Hibernate - Exemplos parte 1



Annotations JPA

- Entity
- Table
- Column
- Id
- GeneratedValue
- SequenceGenerator
- ManyToOne
- OneToMany
- OneToOne
- JoinColumn
- JoinTable
- ElementCollection
- ManyToMany
- OrderColumn
- MapKeyColumn





Entity, Table, Seq, ID e Column

```
@Entity
@Table(name = "TB_CLIENTE")
public class Cliente {

    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="cliente_seq")
    @SequenceGenerator(name="cliente_seq", sequenceName="sq_cliente", initialValue
    private Long id;

    @Column(name = "NOME", nullable = false)
    private String nome;


    @Column(name = "CPF", nullable = false)
    private Long cpf;

    @Column(name = "TEL", nullable = false)
    private Long tel;

    @Column(name = "ENDereco", nullable = false)
    private String end;

    @Column(name = "NUMERO", nullable = false)
    private Integer numero;

    @Column(name = "CIDADE", nullable = false)
    private String cidade;
```






ElementCollection

```
@Entity
@Table(name="TB_PESSOA")
public class Pessoa {

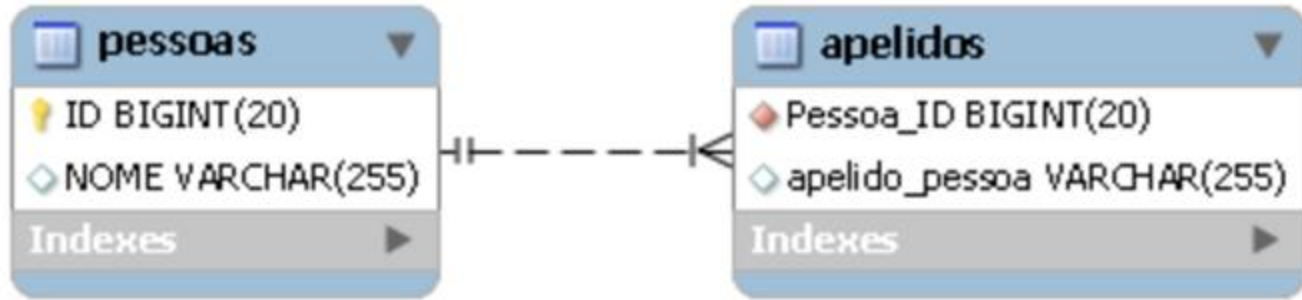
    @Id
    @Column(name="ID")
    private long id;

    @Column(name="nome")
    private String nome;

    @ElementCollection
    @CollectionTable(name="apelidos")
    @Column(name="apelido_pessoa")
    private Collection<String> apelidos;
```



ElementCollection





ElementCollection com objeto

```
@Embeddable
public class Endereco {

    @Column(name="rua")
    private String rua;

    @Column(name="numero")
    private int numero;
```

```
@Entity
@Table(name="TB_PESSOA")
public class Pessoa {

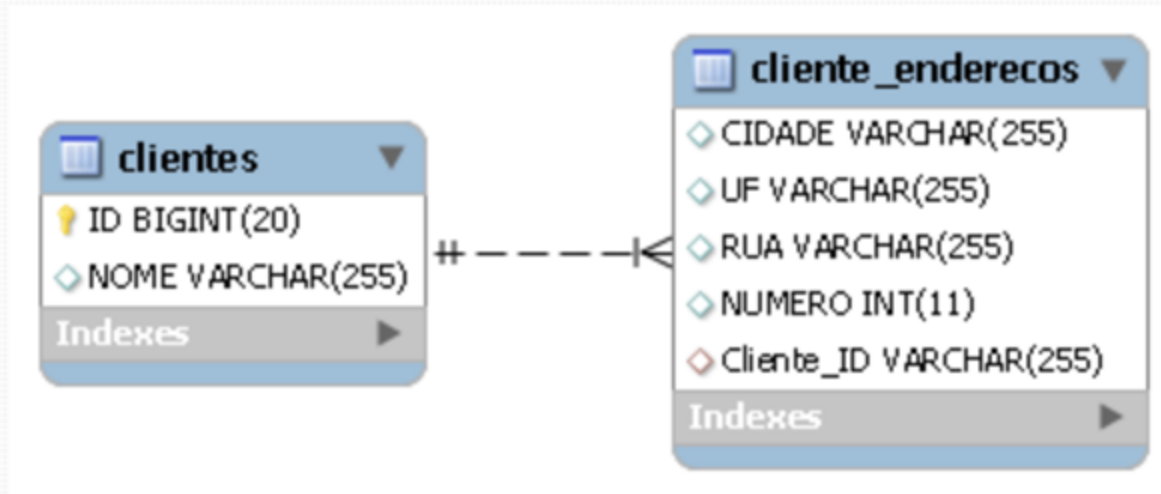
    @Id
    @Column(name="ID")
    private long id;

    @Column(name="nome")
    private String nome;

    @ElementCollection
    @CollectionTable(name="apelidos")
    @Column(name="apelido_pessoa")
    private Collection<String> apelidos;

    @ElementCollection
    @CollectionTable(name="cliente_enderecos")
    private Collection<Endereco> endereco;
```


ElementCollection com objeto



A decorative graphic on the left side of the slide. It features a large cyan hexagon with a white number '3' inside. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a small network-like icon with three nodes and a speech bubble icon.

3

JPA e Hibernate - Exemplos parte 2



OneToMany, ManyToOne e OrderColumn

```
@Entity
@Table(name = "TB_CURSO")
public class Curso {

    @Id
    @Column(name = "id")
    private long id;

    @Column(name = "nome")
    private String nome;

    @OneToMany(mappedBy = "curso")
    @OrderColumn(name="ordem_matricula")
    private List<Matricula> matriculas;
```


```
@Entity
@Table(name = "TB_MATRICULA")
public class Matricula {

    @Id
    @Column(name = "id")
    private long id;

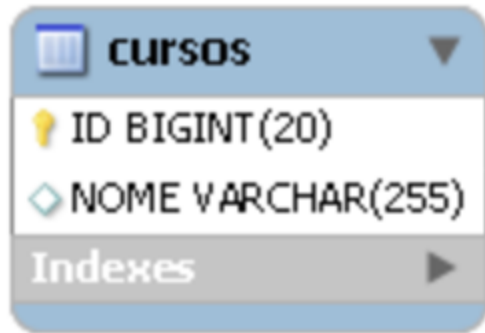
    @Column(name = "valor")
    private Double valor;

    @Column(name = "data")
    private Instant data;

    @ManyToOne
    private Curso curso;
```



OneToMany, ManyToOne e OrderColumn





ManyToMany

```
@Entity
@Table(name = "TB_EMPREGADO")
public class Empregado {

    @Id
    @Column(name = "id")
    private long id;

    @Column(name = "nome")
    private String nome;


    @ManyToMany(cascade = { CascadeType.ALL })
    @JoinTable(
        name = "TB_EMPREGADO_PROJETO",
        joinColumns = { @JoinColumn(name = "empregado_id") },
        inverseJoinColumns = { @JoinColumn(name = "projeto_id") }
    )
    Set<Projeto> projetos = new HashSet<>();
}
```

```
@Entity
@Table(name = "TB_PROJETO")
public class Projeto {

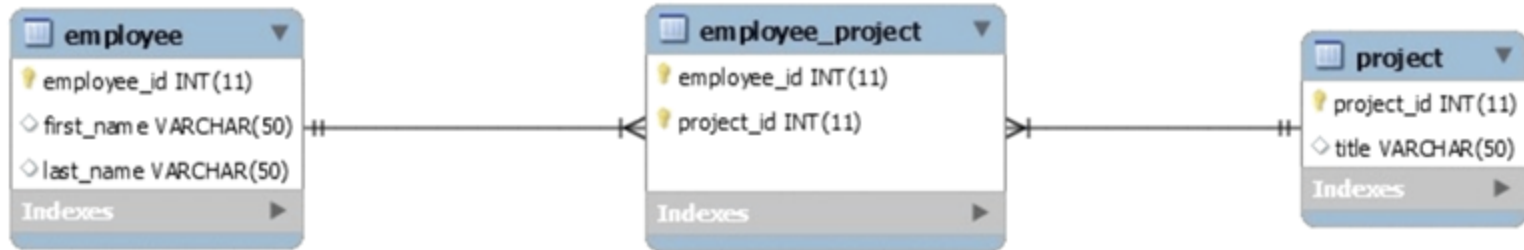
    @Id
    @Column(name = "id")
    private long id;

    @Column(name = "nome")
    private String nome;

    @ManyToMany(mappedBy = "projetos")
    private Set<Empregado> empregados = new HashSet<>();
}
```



ManyToMany






MapKeyColumn

```
@Entity
@Table(name = "TB_EMPREGADO")
public class Empregado {

    @Id
    @Column(name = "id")
    private long id;

    @Column(name = "nome")
    private String nome;

    @ElementCollection
    @CollectionTable(name="TB_EMPREGADOS_TELEFONES")
    @MapKeyColumn(name="tipo_telefone")
    @Column(name="numero_telefone")
    private Map<String, String> numeroTelefones;
```



MapKeyColumn





Referências

Exemplos disponíveis no meu github:

<https://github.com/digaomilleniun/backend-java-ebac>

Tutorial:

<https://www.devmedia.com.br/jpa-2-0-persistencia-a-toda-prova/17437>

