EAPLI : Engenharia de Aplicações

ORM

. Embedded Objects

Embedded Objects

Embedded Objects

Objetos Embeddable:

- Úteis para representar classes não-entidade
- Os campos de dados são embebidos dentro da tabela da entidade a que pertencem
- Suportam o conceito de Value Object de Domain Driven Design
- Um objeto Embedded pode conter outros objetos Embedded

Exemplo: Uma Pessoa tem um Telefone

```
@Entity
public class Pessoa {
   @Id   @GeneratedValue
   private Long id;
   private String nome;
   private Telefone telefone;
}
```

Anotação @Embedded no atributo telefone não é necessária.

```
@Embeddable
public class Telefone {
   private String numero;
   private String indicativoPaís;
   . . .
}
```

```
public static void main(String[] args) {
    EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("JPAPU");
    EntityManager em = emf.createEntityManager();
    Pessoa p1 = new Pessoa("Manuel");
    Pessoa p2 = new Pessoa("José");
    Telefone t1 = new Telefone("123456789", "+351");
    Telefone t2 = new Telefone("234234234", null);
    p1.setTelefone(t1);  p2.setTelefone(t2);
    em.getTransaction().begin();
    em.persist(p1); em.persist(p2);
    em.getTransaction().co
    em.close();
    emf.close();
  }
```

Tabela criada:

PESSOA(ID, NOME, INDICATIVOPAÍS, NUMERO)

1, Manuel, +351, 123456789

2, José, <NULL>, 234234234

Embedded Objects

Exemplo: Uma Pessoa tem vários objetos Telefone

```
@Entity
public class Pessoa {
   @Id   @GeneratedValue
   private Long id;
   private String nome;
    @ElementCollection
   private Set<Telefone> telefones = new HashSet<>();
   . . .}
```

```
@Embeddable
public class Telefone {
   private String numero;
   private String indicativoPaís;
   . . .
}
```

```
public static void main(String[] args) {
...
Pessoa p1 = new Pessoa("Manuel");
      Telefone t1 = new Telefone("123456789", "+351");
      Telefone t2 = new Telefone("987654321", "+49");
      Telefone t3 = new Telefone("234234234", null);
      p1.addTelefone(t1); p1.addTelefone(t2);
      p1.addTelefone(t3);
      em.getTransaction().begin();
      em.persist(p1);
      em.getTransaction().commit();
      em.close();
      emf.close();
   }
```

A anotação **ElementCollection** no atributo Set<Telefone> telefones é necessária porque é uma coleção.

@ElementCollection é utilizado para mapear relações 1:N com tipos primitivos (String, ...) ou objetos @Embeddable.

@OneToMany mapeia Relações 1:N com outras Entity.

Tabelas criadas: **PESSOA(ID, NOME)**

1, Manuel

PESSOA_TELEFONES(INDICATIVOPAÍS, NUMERO, PESSOA_ID)

+351, 123456789, 1

<NULL>, 234234234, 1

+49, 987654321, 1

Embedded Objects

Exemplo: Uma Pessoa tem vários objetos Telefone

```
@Entity
public class Empregado {
  @Id
  @Column(name="EMP_ID")
  private long id;
  ...
  @ElementCollection
  @CollectionTable(
        name="TELEFONE",
        joinColumns=@JoinColumn(name="PROPRIETARIO_ID")
  )
  private List<Telefone> telefones;
  ...
}
```

Using the same Embedded Object more than once

```
@Embeddable
public class CafeteriaName implements ValueObject {

   @Embedded
   @AttributeOverrides({
      @AttributeOverride(name = "name", column = @Column(name = "unit_name")),
   })
   private Designation unit;

   @Embedded
   @AttributeOverrides({
      @AttributeOverride(name = "name", column = @Column(name = "cafe_name")),
   })
   private Designation cafe;

   …
}
```

```
@Embeddable
public class Designation implements ValueObject, Serializable {

        private final String name;
        …
}
```

EmbeddedID

- It is possible to use an embedded object as the primary key in the table

```
@Embeddeable
class Matricula {
      String part1;
      String part2;
      String part3;
}

@Entity
class Veiculo {
      @EmbeddedID
      Matricula matricula;
      …
}
```

- But, **avoid** compound keys in the database

Uniqueness of embedded objects

```
@Embeddeable
class Matricula {
      String part1;
      String part2;
      String part3;
}

@Entity
@Table(uniqueConstraints={
      @UniqueConstraint={columnNames={"part1", "part2",
"part3"}}})
class Veiculo {
      @Id
      Long pk;

      @Embedded
      Matricula matricula;
      …
}
```

Sumário

- Objetos Embeddable, atributos embebidos na mesma entidade&tabela apesar de serem de classes distintas (classes não-entidade)
- @Embeddable
- @Embedded
- @ElementCollection
- @ElementCollection vs @OneToMany
- @AttributeOverrides , desambiguar nomes das colunas quando temos várias instâncias de @Embeddable na mesma classe
- @EmbeddedId

- @UniqueConstraint