Anotações JPA

Java Persistence API

Mapeamento básico dos campos

@Table(name = "tb\_music\_2421638")

– configuração do name do bd

@Entity

– Faz o mapeamento objeto relacional

@Id

– Faz o mapeamento da chave primária

@GeneratedValue(strategy = GenerationType.AUTO)

- Para gerar uma chave automática

@Basic(optional = false)

– Configuração do campo obrigatório

@Column(name = "NM\_MUSICA", length = 100, unique = true)

- Para trocar parâmetros do banco de dados

Eventos em cascata

CascadeType.PERSIST – disparado toda vez que uma nova entity for inserida no banco de dados pelo comando: entityManager.persist();

• CascadeType.DETACH – disparado toda vez que uma entity é retirada do Persistence Context. Comandos que podem disparar essa ação:

entityManager.detach(), entityManager.clear(). Ocorrerá um detach também quando o Persistence Context deixar de existir;

• CascadeType.MERGE – disparado toda vez que uma alteração é executada em uma entity. Essa alteração pode acontecer ao final de uma transação com a qual uma managed Entity foi alterada (4.2), ou pelo comando entityManager.merge();

• CascadeType.REFRESH – disparada quando uma entity for

atualizada com informações no banco de dados. Comando: entityManager.refresh();

• CascadeType.REMOVE – disparado quando uma entity é removida (apagada) do banco de dados, os relacionamentos marcados também serão eliminados. O comando utilizado é o entityManager.remove();

• CascadeType.ALL – todos os eventos anteriores serão sempre refletidos nas Entities relacionadas.

public class Endereco{

@OneToOne(mappedBy = "endereco")

private Pessoa pessoa;

}

public class Pessoa{

@OneToOne(cascade = CascadeType.PERSIST)

private Endereco endereco;

}

Mapeamento de relacionamentos

- Relacionaento one to one

@Entity

public class Pessoa {

@Id

private id;

private nome;

@OneToOne

private Endereco endereco;

// get and set

}

@Entity

public class Endereco {

@Id

private id;

private String nomeRua;

}

@OneToOne

@JoinColumn(name = "chave\_do\_endereco")

@Entity

public class Pessoa{

// outras informações

@OneToMany

private List<Cachorro> cachorros;

}

@Entity

public class Pessoa{

// outras informações

@OneToMany

@JoinColumn(name = "pessoa\_id")

private List<Cachorro> cachorros;

}

@Entity

public class Cachorro {

@Id

@GeneratedValue

private int id;

private String nome;

@ManyToOne

@JoinColumn(name = "pessoa\_id")

private Pessoa pessoa;

}

@Entity

public class Pessoa {

// outras coisas

@OneToMany(mappedBy = "pessoa")

private List<Cachorro> cachorros;

}

- Many to many

@Entity

public class Pessoa{

// outras informações

@ManyToMany

@JoinTable(name = "trabalhos\_pessoas")

private List<Trabalho> trabalhos;

}

@Entity

public class Trabalho {

@Id

@GeneratedValue

private int id;

private String nome;

@ManyToMany(mappedBy = "trabalhos")

private List<Pessoa> funcionarios;

// get and set

}

Exemplo de Atributos sobrepostos

@Embedded

@AttributeOverrides({

@AttributeOverride(name="rua",

column=@Column(name="end\_trabalho")),

@AttributeOverride(name="bairro",

column=@Column(name="end\_bairro\_trabalho")),

@AttributeOverride(name="numero",

column=@Column(name="end\_numero\_trabalho"))

})

private Endereco enderecoTrabalho;

Exemplo de enumeração

public enum PapelUsuario{

ADMIN, MANAGER

}

@Enumerated

@ElementCollection(targetClass = PerfilDoUsuario.class)

@Enumerated(EnumType.STRING)

private List<PerfilDoUsuario> perfis;

Exemplo de Herança

- Utiliza uma única entity para a relação

@Entity

@Inheritance(strategy = InheritanceType.SINGLE\_TABLE)

@DiscriminatorColumn(name = "pertence\_a\_classe")

public abstract class Pessoa {

@Id

@GeneratedValue

private int id;

private String nome;

}

@Entity

@DiscriminatorValue("PessoaJuridica")

public class PessoaJuridica extends Pessoa{

private String CNPJ;

// outras coisas

}

@Entity

@DiscriminatorValue("PessoaFisica")

public class PessoaFisica extends Pessoa{

private String CPF;

// outras coisas

}

- Utiliza duas tabelas.. uma para cada entity

@Entity

@Inheritance(strategy = InheritanceType.JOINED)

public abstract class Pessoa { ... }

@Entity

public class PessoaFisica extends Pessoa { ... }

@Entity

public class PessoaJuridica extends Pessoa { ... }

- Utiliza duas tabelas.. uma para cada entity

@Entity

@Inheritance(strategy = InheritanceType.TABLE\_PER\_CLASS)

public abstract class Pessoa { ... }

@Entity

public class PessoaFisica extends Pessoa{ ... }

@Entity

public class PessoaJuridica extends Pessoa{ ... }

Exemplo de classe com duas Tabelas

@Entity

@Table(name="MUSICA")

@SecondaryTable(

name="MUSICA\_2372",

pkJoinColumns={@PrimaryKeyJoinColumn(name="MUSICA\_ID")})

Exemplo de chave composta

@Embeddable

public class MusicaId implements Serializable {

private int duracaoSegundos;

private String nome;

public MusicaId(){ }

public MusicaId(int duracaoSegundos, String nome) {

super();

this.duracaoSegundos = duracaoSegundos;

this.nome = nome;

}

// hashCode e equals

}

Configuração Persistence JPA

<persistence-unit name="PERSISTENCE\_JTA" transaction-type="JTA">

<provider>org.hibernate.ejb.HibernatePersistence</provider>

<jta-data-source>java:comp/env/jdbc/sgiDS</jta-data-source>

<non-jta-data-source>

<persistence version="2.0">

<persistence-unit name="default">

<properties>

<property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>

<property

name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:3200/posjava"/>

<property

name="javax.persistence.jdbc.user"

value="postgres"/>

<property

name="javax.persistence.jdbc.password"

value="admin"/>

<property

name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect" />

<property

name="hibernate.hbm2ddl.auto" value="create" />

<property

name="hibernate.show\_sql"

value="true" />

<property

name="hibernate.format\_sql" value="true" />

</properties>

<exclude-unlisted-classes>true</exclude-unlisted-classes>

</persistence-unit>

</persistence>