O Hibernate é uma das especificações mais utilizadas da JPA, e fornece diversas anotações para a utilização do Mapeamento Objeto-Relacional.

Vamos conhecer as principais delas?

**@Entity**

Essa anotação é usada para marcar uma classe como uma entidade que deve ser mapeada para uma tabela de banco de dados. Cada entidade corresponde a uma tabela no banco de dados.

**@Table**

Por padrão, o Hibernate usa o nome da classe como o nome da tabela no banco de dados, fazendo apenas a conversão de padrão de nomenclatura do PascalCase para o SnakeCase, que é o padrão utilizado no banco de dados, no entanto, caso seja necessário que o nome da classe seja diferente do nome da tabela no banco de dados, é possível utilizar esta anotação que permite personalizar o mapeamento entre a classe de entidade e a tabela de banco de dados. Com ela, você pode especificar o nome da tabela, o esquema e as restrições de chave primária.

@Entity

@Table(name = "minha\_tabela")

public class MinhaEntidade { ... }

**@Id**

Marca um campo como a chave primária da entidade. O Hibernate usa essa anotação para identificar exclusivamente os registros no banco de dados.

**@GeneratedValue**

Usada em conjunto com @Id, essa anotação especifica como a chave primária é gerada automaticamente. Pode ser usada com estratégias como AUTO, IDENTITY, SEQUENCE ou TABLE, dependendo do banco de dados.

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

COPIAR CÓDIGO

**@Column**

Similar ao que acontece na anotação @Table, o Hibernate utiliza o nome dos atributos e os converte como sendo idênticos aos nomes das colunas no banco de dados, e caso seja necessário utilizar nomes diferentes,você pode configurar o nome da coluna, bem como seu tipo, e se ela é obrigatória.

@Column(name = "nome\_completo", nullable = **false**)

**private** **String** nome;

COPIAR CÓDIGO

**@OneToMany e @ManyToOne**

Usadas para mapear relacionamentos de um-para-muitos e muitos-para-um entre entidades. Elas definem as associações entre as tabelas no banco de dados.

@Entity

public class Autor {

@OneToMany(mappedBy = "autor")

private List<Livro> livros;

}

@Entity

public class Livro {

@ManyToOne

@JoinColumn(name = "autor\_id")

private Autor autor;

}

COPIAR CÓDIGO

**@ManyToMany**

A anotação @ManyToMany é usada para mapear relacionamentos muitos-para-muitos entre entidades.

**@OneToOne**

A anotação @OneToOne é usada para mapear relacionamentos um-para-um entre entidades.

**@JoinColum**

A anotação @JoinColumn é usada para especificar a coluna que será usada para representar um relacionamento entre entidades. É frequentemente usada em conjunto com @ManyToOne e @OneToOne.

@ManyToOne

@JoinColumn(name = "autor\_id")

private Autor autor;

COPIAR CÓDIGO

**@JoinTable**

A anotação @JoinTable é usada para mapear tabelas de junção em relacionamentos muitos-para-muitos. Ela especifica a tabela intermediária que liga duas entidades.

@Entity

public class Estudante {

@ManyToMany

@JoinTable(name = "inscricao",

joinColumns = @JoinColumn(name = "estudante\_id"),

inverseJoinColumns = @JoinColumn(name = "curso\_id"))

private List<Curso> cursos;

}

COPIAR CÓDIGO

**@Transient**

A anotação @Transient é usada para marcar uma propriedade como não persistente. Isso significa que a propriedade não será mapeada para uma coluna no banco de dados.

@Transient

**private** transientProperty;

COPIAR CÓDIGO

**@Enumerated**

A anotação @Enumerated é usada para mapear campos enumerados (enum) para colunas do banco de dados.

@Enumerated(EnumType.STRING)

**private** Status status;

COPIAR CÓDIGO

**@NamedQuery**

Essa anotação é usada para definir consultas JPQL nomeadas que podem ser reutilizadas em várias partes do código.

@Entity

@NamedQuery(name = "Cliente.findAll", query = "SELECT c FROM Cliente c")

public class Cliente { ... }

COPIAR CÓDIGO

**@Cascade**

A anotação @Cascade é usada para especificar o comportamento de cascata das operações de persistência, como salvar e excluir, em relacionamentos. Por exemplo, você pode configurar para que as operações de salvar em cascata afetem entidades relacionadas.

@OneToMany(mappedBy = "departamento")

@Cascade(CascadeType.SAVE\_UPDATE)

private List<Funcionario> funcionarios;

COPIAR CÓDIGO

**@Embeddable e @Embedded**

Essas anotações são usadas para representar tipos incorporados (embeddable types) que podem ser usados como componentes em entidades.

@Embeddable

public class Endereco { ... }

@Entity

public class Cliente {

@Embedded

private Endereco endereco;

}

COPIAR CÓDIGO

Além dessas anotações, há muitas outras que podem ser consultadas na [documentação de anotações do Hibernate](https://docs.jboss.org/hibernate/stable/annotations/reference/en/html/), e que facilitam muito o dia a dia de pessoas desenvolvedoras que usam o ORM.