Hibernate es una de las implementaciones más utilizadas de JPA y proporciona varias anotaciones para la utilización del Mapeo Objeto-Relacional.

¿Conocemos las principales?

@Entity

Esta anotación se utiliza para marcar una clase como una entidad que debe ser mapeada a una tabla de la base de datos. Cada entidad corresponde a una tabla en la base de datos.

@Table

Por defecto, Hibernate utiliza el nombre de la clase como el nombre de la tabla en la base de datos, realizando solo la conversión del formato de nomenclatura de PascalCase a SnakeCase, que es el formato utilizado en la base de datos. Sin embargo, si es necesario que el nombre de la clase sea diferente al nombre de la tabla en la base de datos, es posible utilizar esta anotación que permite personalizar el mapeo entre la clase de entidad y la tabla de la base de datos. Con ella, puedes especificar el nombre de la tabla, el esquema y las restricciones de la clave primaria.

```
@Entity
@Table(name = "mi_tabla")
public class MiEntidad { ... }
```

@Id

Marca un campo como la clave primaria de la entidad. Hibernate utiliza esta anotación para identificar de manera única los registros en la base de datos.

@GeneratedValue

Utilizada en conjunto con @Id, esta anotación específica cómo se genera automáticamente la clave primaria. Puede utilizarse con estrategias como AUTO, IDENTITY, SEQUENCE o TABLE, según el tipo de base de datos.

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

@Column

Similar a lo que sucede con la anotación @Table, Hibernate utiliza los nombres de los atributos y los considera idénticos a los nombres de las columnas en la base de datos. En caso de que sea necesario utilizar nombres diferentes, puedes configurar el nombre de la columna, así como su tipo y si es obligatoria.

```
@Column(name = "nombre_completo", nullable = false)
private String nombre;
```

@OneToMany y @ManyToOne

Se utilizan para mapear relaciones uno a muchos y muchos a uno entre entidades. Definen las asociaciones entre las tablas en la base de datos.

```
@Entity
public class Autor {
    @OneToMany(mappedBy = "autor")
    private List<Libro> libros;
}

@Entity
public class Libro {
    @ManyToOne
    @JoinColumn(name = "autor_id")
    private Autor autor;
}
```

@ManyToMany

La anotación @ManyToMany se utiliza para mapear relaciones muchos a muchos entre entidades.

@OneToOne

La anotación @OneToOne se utiliza para mapear relaciones uno a uno entre entidades.

@JoinColumn

La anotación @JoinColumn se utiliza para especificar la columna que se utilizará para representar una relación entre entidades. Se utiliza con frecuencia en conjunto con @ManyToOne y @OneToOne.

```
@ManyToOne
@JoinColumn(name = "autor_id")
private Autor autor;
```

@JoinTable

La anotación @JoinTable se utiliza para mapear tablas de unión en relaciones muchos a muchos. Especifica la tabla intermedia que conecta dos entidades.

```
@Entity
public class Estudiante {
    @ManyToMany
    @JoinTable(name = "inscripcion",
            joinColumns = @JoinColumn(name = "estudiante_id"),
            inverseJoinColumns = @JoinColumn(name = "curso_id"))
    private List<Curso> cursos;
}
```

@Transient

La anotación @Transient se utiliza para marcar una propiedad como no persistente. Esto significa que la propiedad no se mapeará a una columna en la base de datos.

```
@Transient
private transientProperty;
```

@Enumerated

La anotación @Enumerated se utiliza para mapear campos enumerados (enum) a columnas de la base de datos.

```
@Enumerated(EnumType.STRING)
private Status status;
```

@NamedQuery

Esta anotación se utiliza para definir consultas JPQL (Java Persistence Query Language) nombradas que pueden reutilizarse en varias partes del código.

```
@Entity
@NamedQuery(name = "Cliente.findAll", query = "SELECT c FROM Cliente c")
public class Cliente { ... }
```

@Cascade

La anotación @Cascade se utiliza para especificar el comportamiento de cascada de operaciones de persistencia, como guardar y eliminar, en relaciones. Por ejemplo, puedes configurar para que las operaciones de guardado en cascada afecten a entidades relacionadas.

```
@OneToMany(mappedBy = "departamento")
@Cascade(CascadeType.SAVE_UPDATE)
private List<Empleado> empleados;
```

@Embeddable y @Embedded

Estas anotaciones se utilizan para representar tipos integrados (embeddable types) que pueden ser utilizados como componentes en entidades.

```
@Embeddable
public class Direccion{ ... }

@Entity
public class Cliente {
    @Embedded
    private Direccion direccion;
}
```

Además de estas anotaciones, hay muchas otras que se pueden consultar en la documentación de anotaciones de Hibernate, y que facilitan mucho la vida diaria de los desarrolladores que utilizan el ORM.