Acceso a datos

ORM – Mapeo objeto-relacional

JPA – Otras anotaciones

IES Clara del Rey – Madrid

JPA – Anotación @Enumerated

Se utiliza para almacenar valores de enumeración (tipo enum) en una columna de base de datos.

Puede tomar dos valores:

- EnumType.STRING: Guarda en la base de datos el nombre de la constante de enumeración.
- EnumType.ORDINAL: Guarda en la base de datos el ordinal de la constante de enumeración. Es la opción por defecto.

STRING suele ser más legible en la BD, mientras que ORDINAL es más eficiente en términos de espacio si la cantidad de columnas y registros con elementos enum es grande.

JPA – Anotación @Enumerated

```java
public enum TipoCombustible {
    GASOLINA, DIESEL, ELECTRICO
}

@Entity
public class Coche {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String modelo;

    @Enumerated
    private TipoCombustible tipoCombustible;

    // Otros campos, constructores, getters, setters, etc.
}
```

```java
@Enumerated(EnumType.STRING)
private TipoCombustible tipoCombustible;
```

```java
@Enumerated(EnumType.ORDINAL)
private TipoCombustible tipoCombustible;
```

JPA – Claves compuestas con @Embeddable y @EmbeddedId

Para definir claves primarias compuestas, también conocidas como claves primarias multicolumna, usando @Embeddable y @EmbeddedId:

- Se crea una clase que representa la clave primaria, con los atributos necesarios, y se anota con @Embeddable. Esta clase tiene que ser pública, ser serializable, tener constructor por defecto (sin parámetros) y definir equals y hashCode, para poder comparar claves.
- Se usa un objeto de esta clase como clave primaria de la entidad, de forma que la clave primaria ya no es un tipo primitivo o un Wrapper, sino un objeto con varios atributos.

Ejemplo: clase "LineaPedido" con una clave compuesta por el id del pedido y el id del producto. En la tabla de líneas de pedido cada registro se identifica por los dos campos (id del pedido e id del producto).

JPA – Claves compuestas con @Embeddable y @EmbeddedId

Clase para definir la clave primaria:

```java
@Embeddable
public class LineaPedidoId implements Serializable {
    private Long idProducto;
    private Long idPedido;

    // Constructores, equals, hashCode, etc.
}
```

Clase LineaPedido, que usa la clase anterior como clave primaria:

```java
@Entity
public class LineaPedido {
    @EmbeddedId
    private LineaPedidoId id;

    private int cantidad;

    // Otros campos, constructores, getters, setters, etc.
}
```

JPA – Claves compuestas con @Embeddable y @EmbeddedId

Creación de una línea de pedido con este mecanismo:

```
// Creación de un objeto LineaPedido con la versión @Embeddable
// y @EmbeddedId
LineaPedidoId id = new LineaPedidoId();
id.setIdProducto(1L);
id.setIdPedido(101L);

LineaPedido lineaPedido = new LineaPedido();
lineaPedido.setId(id);
lineaPedido.setCantidad(5);
```

JPA – Claves compuestas con @Id e @IdClass

Alternativa a @Embeddable y @EmbeddedId. Permite "desestructurar" el objeto para la clave primaria, y no tener un objeto como clave, sino los atributos por separado.

En este caso:

- También se crea una clase que representa la clave primaria, con los atributos necesarios, pero no es necesario anotarla con @Embeddable. El resto de los requisitos se mantienen.
- Se usa el atributo @IdClass para indicar la clase que representa la clave primaria en el objeto, pero no hay que usar el objeto como clave primaria. Se usan atributos independientes anotados con @Id.

JPA – Claves compuestas con @Id e @IdClass

Clase para definir la clave primaria:

```java
public class LineaPedidoId implements Serializable {
    private Long idProducto;
    private Long idPedido;
    // Constructores, equals, hashCode, etc.
}
```

Clase LineaPedido, que identifica la clase anterior como clave primaria:

```java
@Entity
@IdClass(LineaPedidoId.class)
public class LineaPedido {
    @Id
    private Long idProducto;
    @Id
    private Long idPedido;
    private int cantidad;
    // Otros campos, constructores, getters, setters, etc.
}
```

JPA – Claves compuestas con @Id e @IdClass

Creación de una línea de pedido con este mecanismo:

```
// Creación de un objeto LineaPedido con la versión @IdClass
LineaPedido lineaPedido = new LineaPedido();
lineaPedido.setIdProducto(1L);
lineaPedido.setIdPedido(101L);
lineaPedido.setCantidad(5);
```