

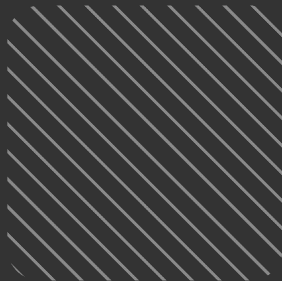


Step by step

// One to Many/Many to One

IT BOARDING

BOOTCAMP



Índice



01 Definición de clases

02 Ejecución

IT BOARDING

BOOTCAMP

En este paso a paso vamos a crear una relación
uno a muchos **bidireccional**.

IT BOARDING

BOOTCAMP

Definición de clases

IT BOARDING

BOOTCAMP





Definición de clases

Creemos una clase Cart y otra Items de la siguiente manera.

```
@Entity
@Table(name="cart")
public class Cart {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;
}
```

```
@Entity
@Table(name="items")
public class Item {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;
}
```

Ahora, vamos a suponer que tenemos una relación uno a muchos entre ellas (un carrito puede tener muchos artículos)



Clase Cart

@OneToMany
indica relación uno
a muchos con la
clase Item

Set: Usamos Set
porque no agrega el
mismo objeto dos
veces a la colección.

```
@Entity
@Table(name="cart")
public class Cart {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;

    @OneToMany(mappedBy = "cart")
    private Set<Item> items;

    // Add getters and setters
}
```



@One to Many a fondo

Anotación

La usamos para indicar que la relación entre la variable de instancia y la clase Item es de uno a muchos.

jpa


@OneToMany(mappedBy = "cart")

Parámetros

mappedBy es usado para definir el lado de referencia de la relación.

Clase Item

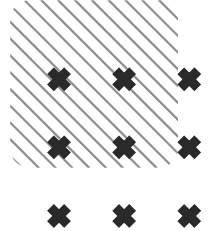
@ManyToOne asociada con la variable de clase `cart`. Muchos ítems asociados a 1 carrito.



```
@Entity
@Table(name="items")
public class Item {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;

    @ManyToOne
    @JoinColumn(name="cart_id", nullable=false)
    private Cart cart;

    // Add getters and setters
}
```

The owning side vs the referencing side

Cuando tenemos relaciones bidireccionales no existen diferencias con las relación unidireccional a nivel base de datos. Solo definimos las direcciones en nuestro modelo de dominio.

Para una relación bidireccional tenemos:

Owning Side

En una relación Uno a Muchos definimos el lado propietario del lado “muchos” de la relación y es donde dejamos la clave externa. Para eso usamos la anotación **@JoinColumn**.

Referencing Side

Para que la asociación sea bidireccional tenemos que definir el lado de la referencia. Podemos usar el atributo **mappedBy** de **@OneToMany** para realizar esto. En **mappedBy** definimos el nombre del atributo de mapeo de asociación en el lado propietario.



Ejecutar

IT BOARDING

BOOTCAMP



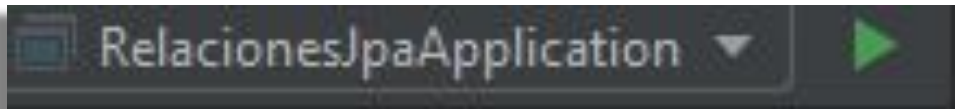


Corremos la aplicación

Configuramos nuestro application.properties:

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.h2.console.enabled=true
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driverClassName=org.h2.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
```

Corremos nuestra aplicación:



Resultados



Abrimos el navegador en localhost:8080/h2-console

The screenshot shows the H2 database console interface. At the top, there is a language dropdown menu set to 'English' and navigation links for 'Preferences', 'Tools', and 'Help'. Below this is a 'Login' section with a blue header. It contains a 'Saved Settings' dropdown menu currently showing 'Generic H2 (Embedded)'. Underneath, the 'Setting Name' is also 'Generic H2 (Embedded)', with 'Save' and 'Remove' buttons to its right. A horizontal line separates this from the main login fields: 'Driver Class' (org.h2.Driver), 'JDBC URL' (jdbc:h2:mem:testdb), 'User Name' (sa), and 'Password' (empty). At the bottom of the form are 'Connect' and 'Test Connection' buttons.

| | | | | |
|-----------------|-------------------------|-----------------|--------|------|
| English | ▼ | Preferences | Tools | Help |
| Login | | | | |
| Saved Settings: | Generic H2 (Embedded) ▼ | | | |
| Setting Name: | Generic H2 (Embedded) | Save | Remove | |
| <hr/> | | | | |
| Driver Class: | org.h2.Driver | | | |
| JDBC URL: | jdbc:h2:mem:testdb | | | |
| User Name: | sa | | | |
| Password: | | | | |
| | Connect | Test Connection | | |



Resultados

Podemos ver a la izquierda la creación de las tablas.

Cart_id es el campo que está relacionado con el id de la tabla Cart.

| | | |
|-----|---------------|---------|
| [-] | [Table Icon] | CART |
| [+] | [Column Icon] | ID |
| [+] | [Column Icon] | Indexes |
| [-] | [Table Icon] | ITEMS |
| [+] | [Column Icon] | ID |
| [+] | [Column Icon] | CART_ID |
| [+] | [Column Icon] | Indexes |



A nivel BD no existe diferencia entre un modelo unidireccional o bidireccional. La diferencia es a nivel dominio de la aplicación.



Gracias

IT BOARDING

BOOTCAMP

