



Step by step

// Many to Many

IT BOARDING

BOOTCAMP



Índice



01 Definición de clases

02 Ejecución

IT BOARDING

BOOTCAMP

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

IT BOARDING

BOOTCAMP

Definición de clases

IT BOARDING

BOOTCAMP





Definición de clases

Vamos a crear una clase Course y otra Student de la siguiente manera:

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

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

Ahora vamos a suponer que ambas clases tienen una relación Muchos a Muchos entre si.

Clase Student

@ManyToMany

indica la relación muchos a muchos con la clase Course.

@JoinTable: Se usa JoinTable del lado propietario de la relación para setear la tabla intermedia.

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

    @ManyToMany
    @JoinTable(
        name="course_like",
        joinColumns = @JoinColumn(name = "student_id"),
        inverseJoinColumns = @JoinColumn(name = "course_id")
    )
    private Set<Course> likedCourses;

    // All Getter And Setters
}
```

¡Recordatorio!



Recordemos que en los modelos de bases de datos tenemos una tabla intermedia cuando existe una relación muchos a muchos

{@JoinTable}

```
@JoinTable (  
    name="course_like",  
    joinColumns = @JoinColumn (name = "student_id"),  
    inverseJoinColumns = @JoinColumn (name =  
    "course_id")  
)
```

@JoinTable: Se usa para crear una nueva tabla externa (intermediaria).

name: Define el nombre de la nueva tabla.

joinColumns: Se usa para definir la columna en la tabla externa que apunta al id de la clase propietaria.

inverseJoinColumns:

Define la columna en la tabla externa que apunta al id de la tabla inversa de la asociación.



Clase Course

@ManyToMany es usada para la relación muchos a muchos con Student.

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

    @ManyToMany(mappedBy = "likedCourses")
    private Set<Student> likes;

    // All Getter And Setters
}
```

Ejecución

IT BOARDING

BOOTCAMP



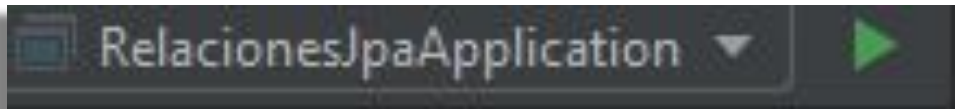


Corremos la aplicación

Configurar 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
```

Correr nuestra aplicación:





Resultados

Abrir el navegador en localhost:8080/h2-console

The screenshot shows the H2 Console Login dialog box. At the top, there is a language dropdown menu set to 'English' and three menu items: 'Preferences', 'Tools', and 'Help'. The main section is titled 'Login' in a blue header. Below this, there are several input fields and buttons. The 'Saved Settings' dropdown is set to 'Generic H2 (Embedded)'. Below it, the 'Setting Name' field also contains 'Generic H2 (Embedded)', with 'Save' and 'Remove' buttons to its right. A horizontal line separates this section from the connection details. The 'Driver Class' field contains 'org.h2.Driver'. The 'JDBC URL' field contains 'jdbc:h2:mem:testdb'. The 'User Name' field contains 'sa'. The 'Password' field is empty. At the bottom, there are two buttons: 'Connect' and 'Test Connection'.

English ▼ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▼

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb

User Name: sa

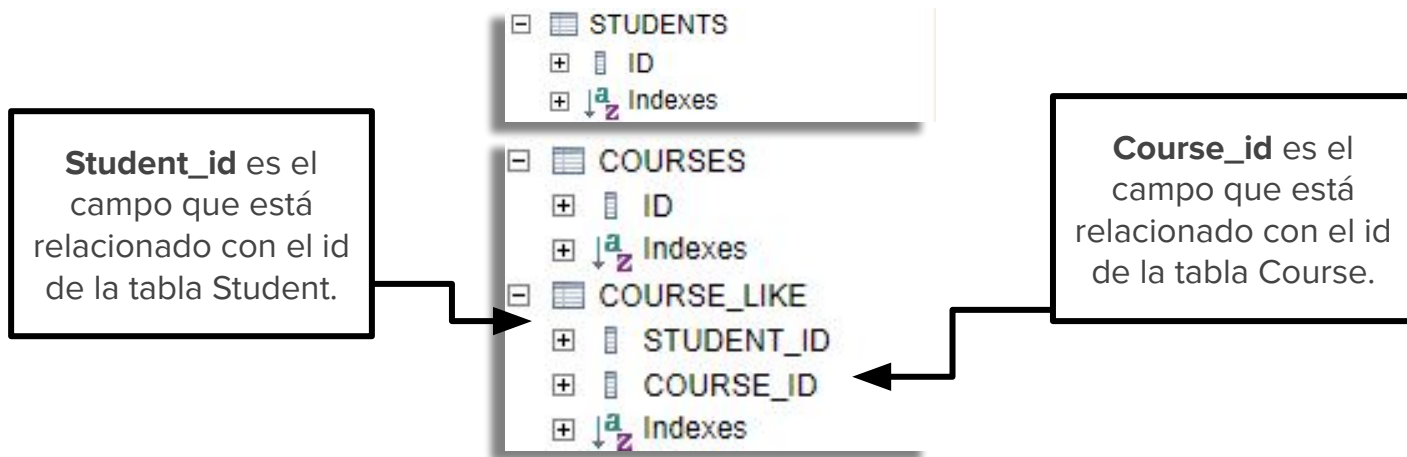
Password:

Connect Test Connection



Resultados

Se puede ver a la izquierda la creación de las tablas.



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

