

(https://reactiveprogramming.io/books/es?utm_source=oscarblancarteblog&utm_medium=header-banner)

Relaciones @ManyToMany

27 diciembre, 2018 (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/>)  
[oblancarte](https://www.oscarblancarteblog.com/author/oblancarte/) (<https://www.oscarblancarteblog.com/author/oblancarte/>)  0



Las relaciones Mucho a Muchos (@ManyToMany) se caracterizan por Entidades que están relacionadas con a muchos elementos de un tipo determinado, pero al mismo tiempo, estos últimos registros no son exclusivos de un registro en particular, si no que pueden ser parte de varios, por lo tanto, tenemos una Entidad A, la cual puede estar relacionada como muchos registros de la Entidad B, pero al mismo tiempo, la Entidad B puede pertenecer a varias instancias de la Entidad A.

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)

Algo muy importante a tomar en cuenta cuando trabajamos con relaciones @ManyToMany , es que en

realidad este tipo de relaciones no existen físicamente en la base de datos, y en su lugar, es necesario crear una tabla intermedia que relacione las dos Entidades, veremos más adelante como resolvemos eso.

Un ejemplo clásico de estas relaciones son los libros con sus autores, de esta forma, un libro puede tener varios autores, y a su vez, los autores pueden tener muchos libros. Pero para que quede más claro, veamos como quedarían las Entidades de Autor (Author), Libro (Book):

Entidad Book:

```
package com.oscarblancarteblog;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name = "books")
public class Book {
    @Id
    @Column(name="ID")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "NAME", nullable = false)
    private String name;

    @JoinTable(
        name = "rel_books_auths",
        joinColumns = @JoinColumn(name = "FK_BOOK", nullable = false),
        inverseJoinColumns = @JoinColumn(name="FK_AUTHOR", nullable = false)
    )
    @ManyToMany(cascade = CascadeType.ALL)
    private List<Author> authors;

    public void addAuthor(Author author){
        if(this.authors == null){
            this.authors = new ArrayList<>();
        }

        this.authors.add(author);
    }

    /** GET and SET */
}
```

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
Como podemos apreciar, hemos creado una lista de tipo Author , la cual es anotada con @ManyToMany , adicional, hemos definido la anotación @JoinTable , la cual nos sirve para definir la estructura de la tabla intermedia que contendrá la relación entre los libros y los autores.

La anotación `@JoinTable` no es obligatoria en sí, ya que en caso de no definirse JPA asumirá el nombre de la tabla, columnas, longitud, etc. Para no quedar a merced de la implementación de JPA, siempre es recomendable definirla, así, tenemos el control total sobre ella.

Hemos definidos las siguientes propiedades de la anotación `@JoinTable`:

- **name:** Nombre de la tabla que será creada físicamente en la base de datos.
- **joinColumns:** Corresponde al nombre para el ID de la Entidad Book.
- **inverseJoinColumns:** Corresponde al nombre para el ID de la Entidad Author

Entidad Author :

```
package com.oscarblancarteblog;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name="authors")
public class Author {

    @Id
    @Column(name="ID")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name="NAME", nullable = false)
    private String name;

    @ManyToMany(mappedBy = "authors")
    private List<Book> books;

    /** GET and SET */
}
```

El caso de la Entidad `Author` es más simple, pues solo marcamos la colección con `@ManyToMany`, pero en este caso ya no es necesario definir la anotación `@JoinTable`, en su lugar, definimos la propiedad `mappedBy` para indicar la relación bidireccional y al mismo tiempo, JPA puede tomar la configuración del `@JoinTable` de Books.

Como resultado de estas Entidades, tendremos las siguientes tablas auto generadas:

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic



Notemos en la tabla *authors* no tiene una columna que haga referencia a *books*, ni *books* a *authors*, si no que es necesario tener una tabla intermedia que haga el cruce entre las dos tablas.

La tabla intermedia (*rel_book_auths*) es generada por la anotación `@JoinTable` y sus dos columnas son llaves foraneas a las tablas *books* y *authors*.

Prueba de validación

Para comprobar que todo funciona como lo hemos dicho, vamos a realizar una prueba, la cual se ve de la siguiente manera:

X

```

public static void main(String[] args) {

    //Authors
    Author author1 = new Author();
    author1.setName("Juan Perez");

    Author author2 = new Author();
    author2.setName("Oscar Blancarte");

    Author author3 = new Author();
    author3.setName("Arturo Martinez");

    //Books
    Book book1 = new Book();
    book1.setName("El lago y el pato");
    book1.addAuthor(author1);
    book1.addAuthor(author2);
    book1.addAuthor(author3);

    Book book2 = new Book();
    book2.setName("Una mañana de verano");
    book2.addAuthor(author1);
    book2.addAuthor(author2);
    book2.addAuthor(author3);

    EntityManager em = EntityManagerUtil.getEntityManager();
    em.getTransaction().begin();
    em.persist(book1);
    em.persist(book2);
    em.getTransaction().commit();

    System.out.println("FIN");
}

```

Hemos creados dos libros y tres autores, y luego hemos asociado a los autores a los libros, con la intención de que los autores estén en dos libros y los libros tengan varios autores.

También los quiero invitar a ver mi curso de JPA, donde explico todos estos temas aplicados con API REST, <https://codmind.com/courses/jpa> (<https://codmind.com/courses/jpa>)

X

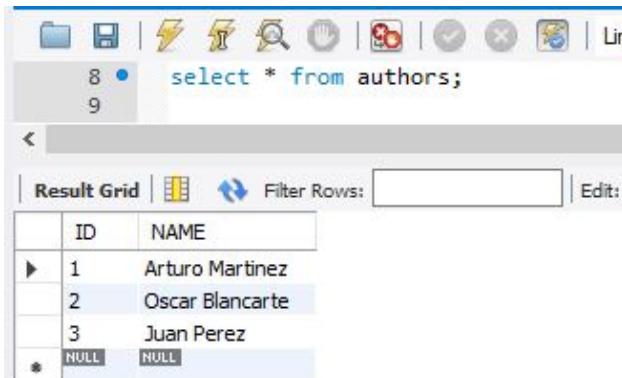
VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic



(<https://codmind.com/courses/jpa>)

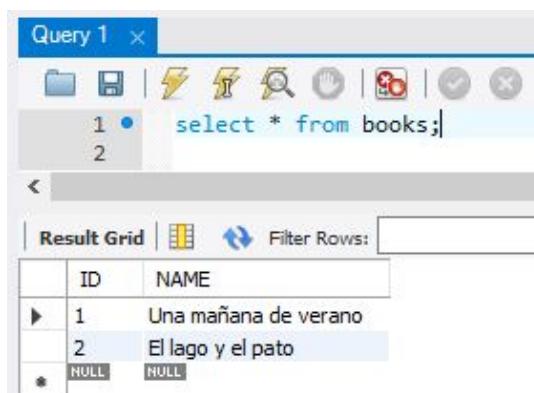
Los invito a mi Curso de Mastering JPA (<https://codmind.com/courses/jpa>), donde habla de todos estos temas y crearemos un API REST para probar todos los conceptos de persistencia.

Ahora veamos como se ven las tablas *authors*, *books* y *rel_books_auths*:



```
8 • select * from authors;
9
```

	ID	NAME
▶	1	Arturo Martinez
	2	Oscar Blancarte
	3	Juan Perez
*	NULL	NULL



```
Query 1 x
1 • select * from books;
2
```

	ID	NAME
▶	1	Una mañana de verano
	2	El lago y el pato
*	NULL	NULL

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic

	FK_AUTHOR	FK_BOOK
▶	1	1
	2	1
	3	1
	1	2
	2	2
*	3	2
	NULL	NULL

Para poder obtener la relación entre libros y autores solo faltaría hacer la unión entre las dos tablas.

Conclusiones

Para concluir solo faltaría resaltar que en las relaciones `@ManyToMany` los registros son independientes de los registros a los que son relacionados, por lo que en este caso, podrían existir los autores si no existieran los libros, y al revés.

[ANTERIOR \(HTTPS://WWW.OSCARBLANCARTEBLOG.COM/2018/12/20/RELACIONES-ONETOMANY/\)](https://www.oscarblancarteblog.com/2018/12/20/RELACIONES-ONETOMANY/)

[ÍNDICE \(HTTP://WWW.OSCARBLANCARTEBLOG.COM/TUTORIALES/JAVA-PERSISTENCE-API-JPA/\)](http://WWW.OSCARBLANCARTEBLOG.COM/TUTORIALES/JAVA-PERSISTENCE-API-JPA/)

[SIGUIENTE \(HTTPS://WWW.OSCARBLANCARTEBLOG.COM/2016/12/20/INTRODUCCION-JAVA-TRANSACTION-API-JTA/\)](https://www.oscarblancarteblog.com/2016/12/20/INTRODUCCION-JAVA-TRANSACTION-API-JTA/)

Comparte esto:

LinkedIn (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/?share=linkedin&nb=1>)

Facebook (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/?share=facebook&nb=1>)

Twitter (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/?share=twitter&nb=1>)

WhatsApp (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/?share=jetpack-whatsapp&nb=1>)

Relacionado



[VER CURSOS \(https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic\)](https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)



X

(<https://www.oscarblancarteblog.com/2018/12/14/relaciones-one-to-one/>)
Relaciones @OneToOne (<https://www.oscarblancarteblog.com/2018/12/14/relaciones-one-to-one/>)

(<https://www.oscarblancarteblog.com/2018/12/20/relaciones-one-to-many/>)
Relaciones @OneToMany (<https://www.oscarblancarteblog.com/2018/12/20/relaciones-one-to-many/>)

(<https://www.oscarblancarteblog.com/2018/12/06/relacion-muchos-a-uno-con-many-to-one/>)
Relación muchos a uno con @ManyToOne (<https://www.oscarblancarteblog.com/2018/12/06/relacion-muchos-a-uno-con-many-to-one/>)

hibernate (<https://www.oscarblancarteblog.com/tag/hibernate-2/>)

jpa (<https://www.oscarblancarteblog.com/tag/jpa-2/>)

49 thoughts to “Relaciones @ManyToOne”

ALDO

7 julio, 2019 a las 11:11 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-2550>)

Estimado Oscar:

Tengo una duda. Segun entiendo, con la configuracion de relacion que escribes arriba, si al intentar guardar un libro con el mismo autor, meteria en la tabla AUTHORS un nuevo registro con el mismso nombre del author pero con un id distinto obviamente. Entonces en esta tabla tendriamos el mismo author tantas veces hayamos agregado un libro con ese author, pero con ids distintos.

En el caso de querer persistir un nuevo libro pero con author ya existente en la BD sin que se sobre-escriba o duplique con id diferente, como se tendria que hacer.

Mas aun, si tomaras un author existen y se lo metes al nuevo libro, lanzaria una excepcion tipo

“com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException:
Duplicate entry ‘id_de_la_tupla_author’ for key ‘PRIMARY’”.

Lo q estaria diciendo, logicamente, que no puede registrar una nueva tupla porque ya se encuentra grabada y la primary key se entiende univoca.

[RESPONDER](#)

X

OBLANCARTE

12 julio, 2019 a las 10:06 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-2551>)

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)

Hola Aldo, lo que tienes que haces es hacer un `find` del Autor y luego asignarlo a los libros, de esta forma el EntityManager sabrá que se trata de un Autor existente y no lo

persistirá, si no que solo tomará el ID para asignarlo al libro.

RESPONDER

ALDO

16 julio, 2019 a las 12:48 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-2659>)

Entiendo. Valida la aclaracion.

Muchas gracias por este blog y el trabajo que haces en el.

Saludos.

RESPONDER

OBLANCARTE

16 julio, 2019 a las 2:30 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-2661>)

Gracias Aldo

RESPONDER

VICTOR GUEVARA

31 diciembre, 2020 a las 1:41 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-9149>)

Hola profe, tengo la misma duda que Aldo, pero cuando dice hacer un find a que se refiere y cómo se hace? gracias

RESPONDER

OBLANCARTE

12 enero, 2021 a las 8:16 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-9340>)

Si, que tienes que buscar al cliente mediante el EntityManager.find y el resultado, lo asignas a la entidad, de esta forma, JPA en lugar de crear un nuevo registro, lo asociará al existente

X

RESPONDER

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic

YAS

14 agosto, 2019 a las 12:18 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3035>)

Hola Oscar, me puedes aclarar un poco esto, porque si estoy un poco perdida con el asunto, acá dices: "Para poder obtener la relación entre libros y autores solo faltaría hacer la unión entre las dos tablas.", para hacer esto debería tener una tabla entidad que se llame como la relación que creamos? sino es así de que forma pudiera hacerlo.

Gracias

[RESPONDER](#)

OBLANCARTE

17 agosto, 2019 a las 3:35 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3080>)

Hola Yas, la cuestión es muy simple, en las bases de datos no existen las relaciones ManyToMany, en su lugar, es necesario crear una tabla intermedia que relacione a los dos entendidas, en realidad no existe otra alternativa, tienes que crear una tabla intermedia si quieras implementar ManyToMany

[RESPONDER](#)

STIVEN

26 septiembre, 2019 a las 9:25 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3631>)

Hola, como se puede generar una tabla intermedia donde se relacionen a 3 tablas?

[RESPONDER](#)

OBLANCARTE

26 septiembre, 2019 a las 11:58 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3636>)

Te refieres que una sola tabla tiene relación con otras 3 o que las tres se une por medio de la segunda?

[RESPONDER](#)

X

[VER CURSOS](https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic) (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)

STIVEN

27 septiembre, 2019 a las 2:28 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany>)

/#comment-3646)

son 3 tablas que estan relacionadas . Las tablas son productos, pedido y sucursal. Existe una relación entre el pedido y producto. La cuestión es que cada producto del pedido puede tener diferente sucursal.

Por ende pienso en una relación en la cual la tabla pedido, producto, sucursal se relacionen con una tabla llamada pedido_producto la cual contenga foraneas de las tres tablas anteriores.

En el ejemplo que muestras, indicas como manejar una relacion de muchos a muchos. Quisiera saber si de esta manera puedo generar una referencia sobre otro entity para que me cree la relación que requiero

Gracias 😊

RESPONDER

OBLANCARTE

27 septiembre, 2019 a las 10:56 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3652>)

Yo lo que haría sería crear una relación de @OneToMany del pedido al producto y del producto a la sucursal sería @ManyToOne, aun que no conozco bien tu escenario de negocio, quizás este equivocado,

RESPONDER

STIVEN

27 septiembre, 2019 a las 3:19 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3647>)

Mira encontré algo interesante que me permitirá en teoría implementar mi requerimiento. Te lo comarto.

<https://hellokoding.com/jpa-many-to-many-extra-columns-relationship-mapping-example-with-spring-boot-maven-and-mysql/> (<https://hellokoding.com/jpa-many-to-many-extra-columns-relationship-mapping-example-with-spring-boot-maven-and-mysql/>)

Exitos!!!!

RESPONDER

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
8 octubre, 2019 a las 3:28 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3793>)

Hola, gracias por la explicacion. Queria preguntarte en el caso de querer eliminar un

book como seria el proceso, he estado intentando hacer algo parecido y me da un error de restriccion en la fk. Gracias !

RESPONDER

OBLANCARTE

8 octubre, 2019 a las 5:00 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-3796>)

El problema se puede deber que al borrar el Book, este tiene una relación con el Autor y una cascada CascadeType.ALL, lo que significa que cuando borras el Book, se intenta borrar en cascada el Autor, lo que puede provocar que otros Book este relacionado con el Autor que intentas eliminar, lo que podrás hacer es cambiar el CascadeType a solo PERSIST.

saludos,

RESPONDER

HABITANTE5079

26 octubre, 2019 a las 7:10 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-4107>)

Funciona si creas un libro y le añades autores y luego guardas el libro, entonces persisten ambas entidades y su relación en la tabla one-to-many/many-to-one llamada rel_books_auths.

Sin embargo la relación inversa es una situación también válida, crear un autor y añadirle libros, creando un método addBook en la clase Author, en ese caso si guardas el autor con los libros se crea el autor pero no los libros y por lo tanto se pierde tanto los libros como la relación.

¿Cómo se puede solucionar esto con @ManyToMany?, ¿o no se puede?

RESPONDER

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic

OBLANCARTE

30 octubre, 2019 a las 11:58 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-4172>)

Esto se debe a la cascada, observa que en la clase Book la relación con los autores tiene una cascada de tipo `CascadeType.ALL`, eso provoca que cuando guardar el libro se guarda la relación con los autores, para hacer lo que tu dices tendrías que definir la casca desde el Autor.

saludos.

RESPONDER

HABITANTE5079

31 octubre, 2019 a las 8:36 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-4177>)

Ok, gracias, no había tenido en cuenta ese hecho, lo probaré.

RESPONDER

WILBERT TORRES

21 noviembre, 2019 a las 10:53 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-4353>)

Hola Oscar, cómo se haría eso en un RestController con Spring Boot?

¿Cómo hago para enviar un Json del libro a crear, donde vayan anidados los autores y autor que no exista se cree, sin conocer todavía el id del libro al que pertenece?

Osea digamos que yo tengo un formulario con Angular para crear libros y en el formulario esté la opción para añadir autores y si el autor no existe haya un botón donde se pueda agregar uno nuevo, y al final enviar todo en un json al backend.

RESPONDER

OBLANCARTE

25 noviembre, 2019 a las 3:40 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-4372>)

Hola Wilbert, tu pregunta es compleja por que abarca varias cosas, pero te podría decir que deberías de crear un servicio que acepte un DTO

X

(<https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>), este DTO deberá aceptar todo lo que tienes en la pantalla, incluido el libro y

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic), los Autores, el primer paso de tu servicio deberá ser crear los autores que no existan, luego crear el libro, de esta forma, ya abras creado los autores antes de crear el libro. Te aconsejo que veas nuestro curso de Desarrollo de microservicios con Spring Boot

(<https://codmind.com/courses/api-rest-con-spring-boot>) donde explicamos todo esto

RESPONDER

JUAN JOSÉ URREGO

4 marzo, 2020 a las 9:53 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5611>)

Hola oscar

tengo una cuestion, tengo 3 tablas: grupos, investigadores y solicitudes
la cuestion es, entre grupos e investigadores va n*n, entonces creé la otra tabla esa que
son las dos llaves foraneas(llamemosla cruce)
la cuestión es que la tabla solicitudes se relaciona 1 a 1 con la tabla cruce(la que
relaciona las otras dos)
entonces le asigné una primary key a la tabla cruce, para despues en solicitud, realizar la
relacion 1 a 1 con ésta
pero entonces en java creo una entidad cruce
y veo en todos los ejemplos que encuentro, que eso no se codifica, no se crea esta
entidad, pero claro, porque no hay otra tercera tabla, está bien lo que pienso hacer? o
cómo lo haría?
o se tiene que hacer esa relación sin tener que crear una entidad cruce?

RESPONDER

OBLANCARTE

4 marzo, 2020 a las 11:17 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5615>)

Lo que tienes que haces es usar la anotación `@JoinTable` en tu relación `@ManyToMany`.
Esta anotación permite definir la estructura de esta tabla de "crece" definir el nombre
de las columnas y el nombre de la tabla, de esta forma, JPA se encarga de administrar
sin la necesidad de tener que crear una Entity para gestionar la tabla de "cruce".

RESPONDER

JUAN JOSÉ URREGO

4 marzo, 2020 a las 11:54 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5616>) X

pero eso con las dos tablas para que no salga cruce, que es como están los ejemplos
VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
pero, cómo haría para relacionar esas dos con solicitud?
supongamos que hago lo que me dices en la tabla de investigadores, y coloco los
jointable y manytomany

¿en solicitud hago relacion onebyone hacía investigadores?

RESPONDER

OBLANCARTE

5 marzo, 2020 a las 12:20 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5617>)

No comprendí la pregunta

RESPONDER

LEO

18 marzo, 2020 a las 11:59 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5763>)

Explicación muy clara y puntual, muchas gracias y saludos

RESPONDER

OBLANCARTE

22 marzo, 2020 a las 6:38 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5798>)

No hay de que, me alegra que te sirvió el material

RESPONDER

X
VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic

LISCO

22 marzo, 2020 a las 3:54 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5810>)

Hola Oscar.

Muchas gracias por el Post, muy claro. Tengo una duda, aunque no sé si esta entrada es la más oportuna. Cuando intento mostrar esto, obtengo un bucle infinito, y no se me muestra. ¿Cómo puedo conseguir esto?

- cuando pido los libros, me saque todos los atributos de los libros pero sólo me muestre el id y el nombre de los autores
- cuando pido los autores, me saque todos los atributos de los autores, pero sólo me muestre el id y el nombre de los libros

Muchas gracias de antemano

[RESPONDER](#)

OBLANCARTE

23 marzo, 2020 a las 1:31 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-5814>)

Hola Lisco, de casualidad estás retornando las Entidades por medio de un EJB, Webservice, Rest? por que esto suele pasar cuando se serializan las entidades para viajar por la red. en tal caso, debes evitar las relaciones cíclicas mediante el patrón DTO (<https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>)

[RESPONDER](#)

LORENA

14 mayo, 2020 a las 2:22 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-6684>)

hola!! quisera comentarte lo que me sucedio con la relacion oneToMany, tengo dos entidades

habitacion y tipoHabitacion, cuando utilizo el metodo findById(id) en @GetMapping me generaba un bucle infinito, entonces probe haciendo la relacion unidireccional (solo mapeada desde el lado de "habitacion y funcion"), pero queria que la relacion fuera bidireccional o al menos saber como mapear bien la relacion bidireccional, para hacer esto lei en stackoverflow que puede generar este tipo de problemas el hecho de que los id de cada tabla se llamen igual (ya los cambie para probar) y evidentemente esa era el problema! como funciona perfecto ahora me pregunto si tiene que ver con todo el link <https://codinasion.com/funcionamiento-perfecto-borneo-mapeo-unidireccional-sobre-vistas-por-todo-el-lado/> que utilizan el mismo nombre "id" en las dos clases y no tiene problemas!! quisiera saber el porque a mi no me funciona con el mismo nombre? te agradeceria muchisimo

X

si me lo explicas.

aclaro por las dudas, no se si tendra relacion pero uso la libreria de Lombok (@Data @AllArgsConstructor @NoArgsConstructor) porque tambien lei que puede ser el metodo `toString()` el que puede ocasionar este tipo de errores

estoy recien empezando con spring boot, y con este problema tambien me planteo la duda de cuando es conveniente usar relacion bidireccional o unidireccional

```
public class Habitacion implements Serializable{  
  
    @Id  
    @GeneratedValue(strategy=GenerationType.AUTO, generator="native")  
    @GenericGenerator(name="native", strategy="native")  
    @Column  
    private long id_habitacion;  
  
    @Column  
    @NotBlank  
    private String numerohabitacion;  
  
    /*union con Tipos de habitacion*/  
    @ManyToOne  
    @JoinColumn(name="id_tipohabitacion")  
    private TipoHabitacion tipoHabitacion;  
}  
  
public class TipoHabitacion implements Serializable{  
    @Id  
    @GeneratedValue(strategy=GenerationType.AUTO, generator="native")  
    @GenericGenerator(name="native", strategy="native")  
    @Column  
    private long id_tipo;  
  
    @Column  
    @NotBlank  
    private String clase;  
  
    @OneToMany(cascade = CascadeType.ALL)  
    private Set habitacion;  
}
```

RESPONDER

X

OBLANCARTE

15 mayo, 2020 a las 5:46 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany>)
VER CURSOS (https://codemind.com?utm_source=oscarblancarteblog&utm_medium=stic)

Hola Lorena,

Creo que el problema es por que no estas mapenado adecuadamente la Entidad,

faltaría que agregaras la propiedad mappedBy a la anotación @OneToMany de la clase TipoHabitacion para que quede de la siguiente forma
 @OneToMany(mappedBy="tipoHabitacion") .
 Intenta eso y me dices como te fue.

RESPONDER

LORENA

15 mayo, 2020 a las 6:10 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-6708>)

I lo pongo me genera el bucle infinito

RESPONDER

OBLANCARTE

15 mayo, 2020 a las 7:56 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-6710>)

Huy... entonces no sé, tendría que ver más a detalle el código, desde aquí es difícil diagnosticar un problema

RESPONDER

KARINA ULLOA

7 septiembre, 2020 a las 11:26 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-7655>)

Hola Julio, antes de todo quiero darte las gracias por todo el material que me ha ayudado a crecer. Mi duda es la siguiente, si ya tengo una DB con relaciones, es obligatorio en las Entity realizar las relaciones o se puede manejar desde Spring Boot como tablas separadas?.

Desde ya Muchísimas gracias.

RESPONDER

OBLANCARTE

7 septiembre, 2020 a las 2:45 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-7657>)

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
Hola Karina, lo más recomendable es que las Entidades cumplan con las relaciones que ya tiene tu base de datos, de lo contrario, podrías tener muchos problemas al persistir, actualizar o borrar, ya que JPA utiliza las relaciones para optimizar los Query,

X

así como las operaciones CRUD.
saludos.

RESPONDER

BEATRIZ

8 octubre, 2020 a las 9:30 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-7979>)

Hola Oscar! Muy útil la información. Una pregunta, en la tabla de relación se puede añadir un campo más? Una fecha por ejemplo. Es posible o se debe agregar otra tabla?
Gracias de antemano

RESPONDER

OBLANCARTE

9 octubre, 2020 a las 7:19 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-7986>)

Siempre que sean parte de llave es posible, solo basta agregar más anotaciones @JoinColum o @InverserJoinColumns dentro de las anotaciones @JoinColumns (observa que termina con S)

RESPONDER

BEATRIZ

10 octubre, 2020 a las 1:06 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-7987>)

Muchas gracias!! Creo que me toca una nueva entidad, es un campo nuevo para llevar el orden. Excelente contenido

RESPONDER

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic

JUAN MARCOS ANTONACCIO CAORSI

4 noviembre, 2020 a las 9:43 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-8324>)

Hola Oscar muy util tu informacion y aclaracion. Te comento que tengo un problema parecida por el tema de la duplicacion de los autores. Use un find, y le asigne el autor con la id, y al tratar de insertar un nuevo libro con un atuor ya existente me lanza una excepcion con problemas en la base de datos. Estoy usando la base de datos sql.
Gracias.

RESPONDER

OBLANCARTE

5 noviembre, 2020 a las 4:47 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-8332>)

Cual es el error que te lanza?

RESPONDER

FABIAN AUCACHI

5 diciembre, 2020 a las 6:13 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-8742>)

Sos un genio, eh leido fragmentos de tu libro "Introduccion a los patrones de diseño un enfoque practico", el mejor libro que vi hasta ahora, gracias por tu aporte, me gustaria comprar tu libro en su version completa jeje

RESPONDER

OBLANCARTE

9 diciembre, 2020 a las 5:57 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-8795>)

Hola Fabian, muchas gracias por el comentario... y que te detiene a conseguir el libro?

RESPONDER

X

GERARDO TORREBLANCA

VER CURSOS ([#comment-10698](https://edumind.com/?utm_source=source.oscarblancarteblog&utm_medium=stic))

Tento mi relacion Many to Many muy parecido a tu ejemplo, solo que yo tengo alumnos

y grupos, un alumno puede pertenecer a muchos grupos y un grupo puede tener muchos alumnos. Al obtener un alumno, me trae bastantes objetos de las relaciones, y hace que mi instancia GAE se caiga, pues trae demasiados objetos, miles!, como puedo evitar ese costo?

RESPONDER

OBLANCARTE

9 abril, 2021 a las 3:18 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-10717>)

Ese es un problema bastante frecuente y se debe a que no realizamos las optimizaciones correctas a nivel de anotaciones de JPA, pero tambien, la consulta que lanzamos quizás no es la más adecuada, en este caso, tendría que entender dos cosas, primero, que objetos relacionados tiene el alumno, y segundo que estrategia de fetch estás utilizando para consultar los grupos, ya que si cargas un alumno, este puede cargar todos los grupos, y luego, los grupos pueden cargar los alumnos, y esos alumnos a la vez, pueden cargar los grupos y luego, esos grupos pueden cargar a los alumnos, me explico? se vuelve una consulta recursiva.

saludos.

RESPONDER

JOSÉ ANTONIO GUTIÉRREZ

15 mayo, 2021 a las 6:46 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-11194>)

Hola Óscar.

Antes que nada decirte que te sigo habitualmente, ya que haces fácil lo difícil.
Mi consulta es la siguiente: ¿cómo se construye en Hibernate una relación M:M:M entre 3 tablas a través de una cuarta tabla que, además de las 3 FK (una con cada tabla), tiene sus propios campos? ¿Es posible?
Si al ejemplo que pones, le añades por ejemplo una tabla de Editoriales (y suponiendo que tuviera sentido una M:M:M), ¿qué cambios habría que hacer?
Gracias de antemano.

RESPONDER

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic

OBLANCARTE

22 mayo, 2021 a las 4:50 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-11264>)

Tendrías que usar la anotación @ManyToMany y @JoinTable en las tres Entidades

[RESPONDER](#)

JOSE ANTONIO

22 mayo, 2021 a las 10:46 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-11270>)

Y qué hago en la cuarta entidad, la que tiene las 3 FK, su propia PK (un auto incrementado) y los otros campos adicionales?

[RESPONDER](#)

OBLANCARTE

26 mayo, 2021 a las 10:29 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-11310>)

Hola José, no logro comprender del todo la estructura de tus entidades, quizás si cargaras una foto o algo donde pudiera ver gráficamente la estructura,

[RESPONDER](#)

JUAN MALLEA TARQUI

20 noviembre, 2021 a las 2:52 am (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-14444>)

Hola Oscar.

Estoy trabajando en un proyecto con SpringMVC + Hibernate, y me gustaría saber como insertar una relación entre 2 tablas que ya cuentan con registros cargados pasando solo el ID de cada Entidad a la tabla intermedia

[RESPONDER](#)

OBLANCARTE

7 enero, 2022 a las 10:24 pm (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/#comment-15601>)

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
Hola Juan, lo que necesitas es aprender a usar JPA + Hibernate, por lo que es complicado darte ese tipo de respuestas por este medio. Te invito a que veas mi curso

X

de JPA donde aprenderás eso y muchas cosas mas: <https://codmind.com/courses/jpa>
[\(https://codmind.com/courses/jpa\)](https://codmind.com/courses/jpa)

RESPONDER

DEJA UN COMENTARIO

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

Recibir un email con los siguientes comentarios a esta entrada.

Recibir un email con cada nueva entrada.

PUBLICAR COMENTARIO

◀ Relaciones @OneToMany (<https://www.oscarblancarteblog.com/2018/12/20/relaciones-onetomany/>)

Comandos útiles para NPM ➤ (<https://www.oscarblancarteblog.com/2018/12/29/comandos-utiles-para-npm/>)

...



X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
MI PROGRAMA DE ARQUITECTURA



(https://reactiveprogramming.io/books/software-architecture/es?utm_source=oscarblancarteblog&utm_medium=asside-retro)

SUSCRÍBETE AL BLOG POR CORREO ELECTRÓNICO

Introduce tu correo electrónico para suscribirte a este blog y recibir notificaciones de nuevas entradas.

Únete a otros 1,240 suscriptores

Dirección de correo electrónico

SUSCRIBIR

HOLA, SOY OSCAR BLANCARTE



X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)
Autor, Software Architect & Full Stack Developer con más de 14 años de experiencia en el desarrollo de software y en la industria de las tecnologías de la información. Apasionado por las tecnologías y el software en general.

MIS LIBROS

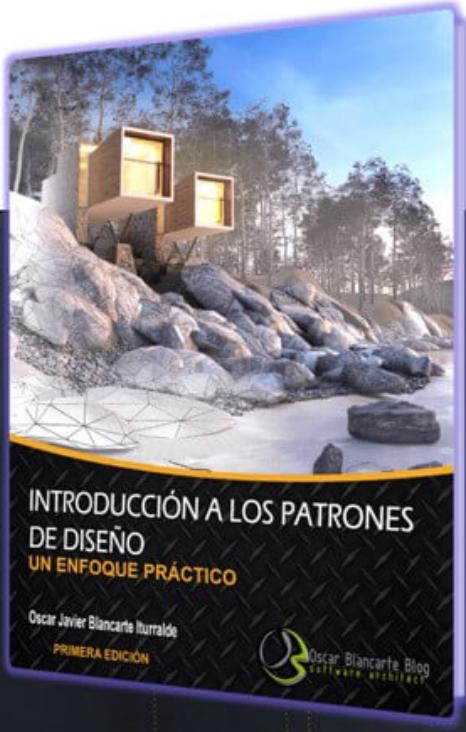


(https://reactiveprogramming.io/books/aplicaciones-reactivas-con-react-nodejs-mongodb/es?utm_source=oscarblancarteblog&utm_medium=aside-banner)



(https://reactiveprogramming.io/books/software-architecture/es?utm_source=oscarblancarteblog&utm_medium=aside-banner)

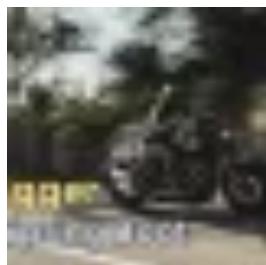
VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)



Introducción a los patrones de diseño

(https://reactiveprogramming.io/books/design-patterns/es?utm_source=oscarblancarteblog&utm_medium=aside-banner)

ARTICULOS POPULARES



(<https://www.oscarblancarteblog.com/2020/08/28/documentar-un-api-rest-con-swagger-y-spring-boot/>)

Documentar un API REST con Swagger y Spring Boot

(<https://www.oscarblancarteblog.com/2020/08/28/documentar-un-api-rest-con-swagger-y-spring-boot/>)



(<https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>)

Data Transfer Object (DTO) – Patrón de diseño

(<https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>)



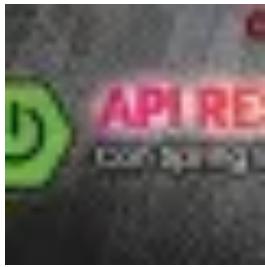
(<https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/>)

Estructura de datos - Árboles (<https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/>)

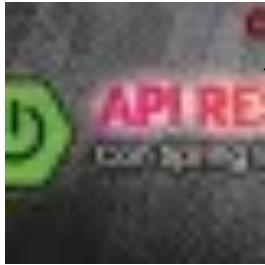
X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)

(<https://www.oscarblancarteblog.com/2017/03/06/soap-vs-rest-2/>)



SOAP vs REST ¿cuál es mejor? (<https://www.oscarblancarteblog.com/2017/03/06/soap-vs-rest-2/>)

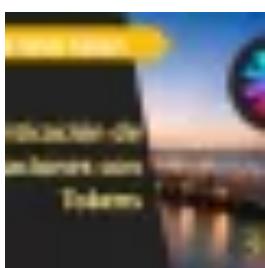


(<https://www.oscarblancarteblog.com/2018/06/25/creando-un-api-rest-en-java-parte-1/>)

Creando un API REST en Java (parte 1)
(<https://www.oscarblancarteblog.com/2018/06/25/creando-un-api-rest-en-java-parte-1/>)



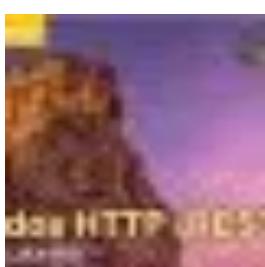
(<https://www.oscarblancarteblog.com/2018/12/20/relaciones-onetomany/>)
Relaciones @OneToMany (<https://www.oscarblancarteblog.com/2018/12/20/relaciones-onetomany/>)



(<https://www.oscarblancarteblog.com/2017/06/08/autenticacion-con-json-web-tokens/>)

Autenticación con JSON Web Tokens

(<https://www.oscarblancarteblog.com/2017/06/08/autenticacion-con-json-web-tokens/>)



(<https://www.oscarblancarteblog.com/2018/12/03/metodos-http-rest/>)

Métodos HTTP (REST) (<https://www.oscarblancarteblog.com/2018/12/03/metodos-http-rest/>)



(<https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/>)

Data Access Object (DAO) Pattern (<https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/>)

(<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/>)

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic



Relaciones @ManyToMany (<https://www.oscarblancarteblog.com/2018/12/27/relaciones-manytomany/>)

TEMAS

2phase commite (<https://www.oscarblancarteblog.com/tag/2phase-commite/>)

API REST (<https://www.oscarblancarteblog.com/tag/api-rest/>)

Arboles (<https://www.oscarblancarteblog.com/tag/arboles/>)

arquitectura (<https://www.oscarblancarteblog.com/tag/arquitectura/>)

Arquitectura de software (<https://www.oscarblancarteblog.com/tag/arquitectura-de-software/>)

BaaS (<https://www.oscarblancarteblog.com/tag/baas/>)

Base de datos (<https://www.oscarblancarteblog.com/tag/base-de-datos/>)

bpel (<https://www.oscarblancarteblog.com/tag/bpel-2/>)

centripio (<https://www.oscarblancarteblog.com/tag/centripio/>)

certificaciones (<https://www.oscarblancarteblog.com/tag/certificaciones/>)

composite (<https://www.oscarblancarteblog.com/tag/composite/>)

desarrollo web (<https://www.oscarblancarteblog.com/tag/desarrollo-web/>)

diseño (<https://www.oscarblancarteblog.com/tag/diseno/>)

Estructura de datos (<https://www.oscarblancarteblog.com/tag/estructura-de-datos/>)

express (<https://www.oscarblancarteblog.com/tag/express/>)

hibernate (<https://www.oscarblancarteblog.com/tag/hibernate-2/>)

HTML (<https://www.oscarblancarteblog.com/tag/html/>) java (<https://www.oscarblancarteblog.com/tag/java-2/>)

java8 (<https://www.oscarblancarteblog.com/tag/java8/>)

javascript (<https://www.oscarblancarteblog.com/tag/javascript/>)

JAX-RS (<https://www.oscarblancarteblog.com/tag/jax-rs/>) JMS (<https://www.oscarblancarteblog.com/tag/jms/>)

jpa (<https://www.oscarblancarteblog.com/tag/jpa-2/>)

Marca personal (<https://www.oscarblancarteblog.com/tag/marca-personal/>)

microservicios (<https://www.oscarblancarteblog.com/tag/microservicios/>)

mongodb (<https://www.oscarblancarteblog.com/tag/mongodb/>)

NodeJS (<https://www.oscarblancarteblog.com/tag/nodejs/>)

oracle (<https://www.oscarblancarteblog.com/tag/oracle/>)

VER CURSOS (https://codmind.com/?utm_source=oscarblancarteblog&utm_medium=stic)

Patrones de diseño (<https://www.oscarblancarteblog.com/tag/patrones-de-diseno/>)

POO (<https://www.oscarblancarteblog.com/tag/poo/>)

X

Programación orientada a objetos (<https://www.oscarblancarteblog.com/tag/programacion-orientada-a-objetos/>)

proxy (<https://www.oscarblancarteblog.com/tag/proxy/>)

react (<https://www.oscarblancarteblog.com/tag/react/>)

Schema (<https://www.oscarblancarteblog.com/tag/schema/>)

Seguridad (<https://www.oscarblancarteblog.com/tag/seguridad/>)

soa (<https://www.oscarblancarteblog.com/tag/soa/>)

soa suite (<https://www.oscarblancarteblog.com/tag/soa-suite-2/>)

software (<https://www.oscarblancarteblog.com/tag/software/>)

spring boot (<https://www.oscarblancarteblog.com/tag/spring-boot/>)

Weblogic (<https://www.oscarblancarteblog.com/tag/weblogic/>)

webservice (<https://www.oscarblancarteblog.com/tag/webservice/>)

WebSocket (<https://www.oscarblancarteblog.com/tag/websocket/>)

XML (<https://www.oscarblancarteblog.com/tag/xml/>) XSD (<https://www.oscarblancarteblog.com/tag/xsd/>)

sparkling lla diseñada por Colorlib (<http://colorlib.com/>) Powered by WordPress (<http://wordpress.org/>)

X

VER CURSOS (https://codmind.com?utm_source=oscarblancarteblog&utm_medium=stic)