#28 JPA-Part7

Saturday, 18 January 2025 11:38 AM

One-to-Many: Unidirectional

- One entity associated with multiple records in another entity
  like: User can have many Orders.
- Reference exist only in 1 direction i.e. from Parent to Child.

- Since its 1:Many, means 1 parent have multiple child and we can not store multiple child ids in 1 parent row, so it creates a ***NEW TABLE*** and stores the mapping.
- By default its ***Lazy*** loading, means when query parents, child rows are not fetched.

```java
@Table(name = "user_details")
@Entity
public class UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String phone;

    @OneToMany(cascade = CascadeType.ALL)
    private List<OrderDetails> orderDetails = new ArrayList<>();


    // Constructors
    public UserDetails() {
    }

    //getters and setters
}
```

```java
@Table(name = "order_details")
@Entity
public class OrderDetails {

    @Id
    @GeneratedValue(strategy = GenerationType
    private Long id;

    private String productName;

    //getters and setters
}
```

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM USER_DETAILS
```

SELECT * FROM USER_DETAILS;

ID	NAME	PHONE

(no rows, 1 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ID	PRODUCT_NAME

(no rows, 4 ms)

Edit

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM USER_DETAILS_ORDER_DETAILS
```

SELECT * FROM USER_DETAILS_ORDER_DETAILS;

ORDER_DETAILS_ID	USER_DETAILS_ID

(no rows, 1 ms)

What if, we don't want to create new table:

- We can use ***@JoinColumn***, this also tells JPA that we want to store the FK in Child table inst of creating a new table.

```java
@Table(name = "user_details")
@Entity
public class UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long userId;
    private String name;
    private String phone;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "user_id_fk", referencedColumnName = "userId")
    private List<OrderDetails> orderDetails = new ArrayList<>();


    // Constructors
    public UserDetails() {
    }

    //getters and setters
}
```

```java
@Table(name = "order_details")
@Entity
public class OrderDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.ID
    private Long id;

    private String productName;

    //getters and setters
}
```

Run | Run Selected | Auto complete | Clear | SQL statement:

SELECT * FROM USER_DETAILS

SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE

(no rows, 1 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

SELECT * FROM ORDER_DETAILS

SELECT * FROM ORDER_DETAILS;

ID	USER_ID_FK	PRODUCT_NAME

(no rows, 1 ms)

1. INSERT CALL

localhost:8080/api/user

POST localhost:8080/api/user

Params • Authorization Headers (8) Body • Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "productName": "IceCream"
        },
        {
            "productName": "ColdDrinks"
        }
    ]
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "userId": 1,
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "id": 1,
            "productName": "IceCream"
        },
        {
            "id": 2,
            "productName": "ColdDrinks"
        }
    ]
}
```

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM USER_DETAILS
```

SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE
1	JohnXYZ	1234567890

(1 row, 1 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ID	USER_ID_FK	PRODUCT_NAME
1	1	IceCream
2	1	ColdDrinks

(2 rows, 1 ms)

2. GET CALL (LAZY)

```java
@RestController
@RequestMapping(value = "/api/")
public class UserController {

    @Autowired
    UserDetailsService userDetailsService;

    @PostMapping(path = "/user")
    public UserDetails insertUser(@RequestBody UserDetails userDetails) {
        return userDetailsService.saveUser(userDetails);
    }

    @GetMapping("/user/{id}")
    public UserDetailsDTO fetchUser(@PathVariable Long id) {
        UserDetails output =  userDetailsService.findByID(id);
        System.out.println("going to map UserDetails to UserDTO");
        UserDetailsDTO userDTO = output.mapUserDetailsToUserDTO();
        return userDTO;
    }
}
```

```java
@Table(name = "user_details")
@Entity
public class UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long userId;
    private String name;
    private String phone;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "user_id_fk", referencedColumnName = "userId")
    private List<OrderDetails> orderDetails = new ArrayList<>();

    public UserDetailsDTO mapUserDetailsToUserDTO() {

        return new UserDetailsDTO(this);
    }

    //getters and setters

}
```

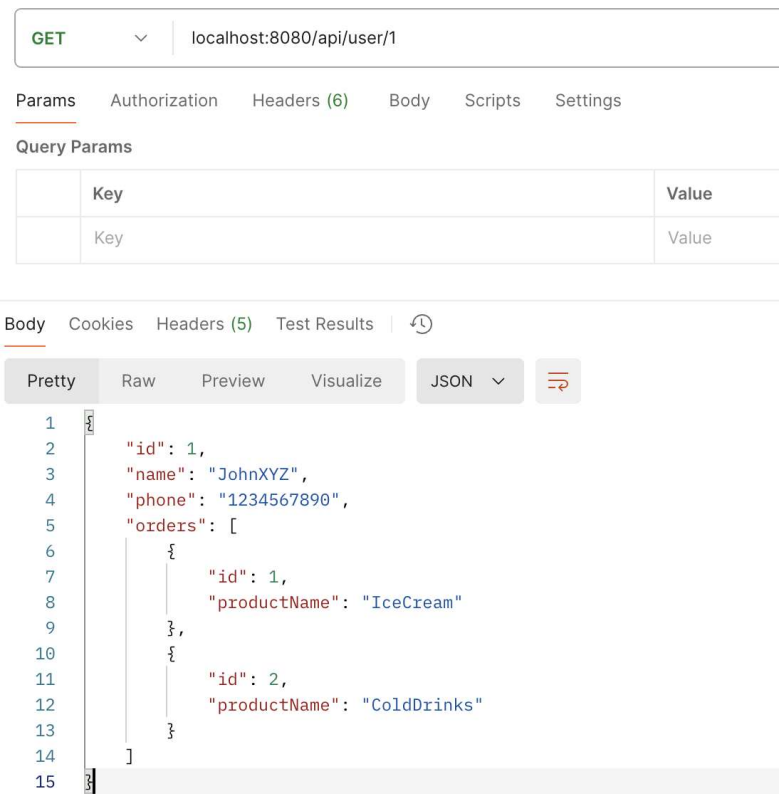
```java
public class UserDetailsDTO {

    private Long id;
    private String name;
    private String phone;
    private List<OrderDetails> orders;
```

```
    // Constructor to populate from UserDetails entity
    public UserDetailsDTO(UserDetails userDetails) {
        this.id = userDetails.getUserId();
        this.name = userDetails.getName();
        this.phone = userDetails.getPhone();
        System.out.println("going to query order table here now");
        this.orders = userDetails.getOrderDetails();
    }

    //getters and setters
}
```

Output:




```
Hibernate:
    select
        ud1_0.user_id,
        ud1_0.name,
        ud1_0.phone
    from
        user_details ud1_0
    where
        ud1_0.user_id=?
going to map UserDetails to UserDTO
going to query order table here now
Hibernate:
    select
        od1_0.user_id_fk,
        od1_0.id,
        od1_0.product_name
    from
        order_details od1_0
    where
        od1_0.user_id_fk=?
```

EAGER fetch:

```
@RestController
@RequestMapping(value = "/api/")
public class UserController {

    @Autowired
    UserDetailsService userDetailsService;

    @PostMapping(path = "/user")
    public UserDetails insertUser(@RequestBody UserDetails userDetails) {
        return userDetailsService.saveUser(userDetails);
    }

    @GetMapping("/user/{id}")
    public UserDetails fetchUser(@PathVariable Long id) {
        return userDetailsService.findByID(id);
    }
}
```

```
@Table(name = "user_details")
@Entity
public class UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY
    private Long userId;
    private String name;
    private String phone;

    @OneToMany(cascade = CascadeType.ALL, fetch = Fetc
    @JoinColumn(name = "user_id_fk", referencedColumnN
    private List<OrderDetails> orderDetails = new Arra


    // Constructors
    public UserDetails() {
    }

    //getters and setters

}
```

Output:

```
Hibernate:
    select
        ud1_0.user_id,
        ud1_0.name,
        ud1_0.phone,
        od1_0.user_id_fk,
        od1_0.id,
        od1_0.product_name
    from
        user_details ud1_0
    left join
        order_details od1_0
            on ud1_0.user_id=od1_0.user_id_fk
    where
        ud1_0.user_id=?
```

Different Cascade Types:

Cascade Type	Impact
CascadeType.PERSIST	Saving User(Parent) also saves related Orders (Child).
CascadeType.MERGE	Updating User also updates Orders.
CascadeType.REMOVE	Deleting User also deletes Orders.
CascadeType.ALL	Includes all cascade operations.

Orphan Removal:

Automatically removes child entry when child removed from Parent collection

```
@RestController
@RequestMapping(value = "/api/")
public class UserController {

    @Autowired
    UserDetailsService userDetailsService;

    @PostMapping(path = "/user")
    public UserDetails insertUser(@RequestBody UserDetails userDetails) {
        return userDetailsService.saveUser(userDetails);
    }
```

```
@Table(name = "user_details")
@Entity
public class UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENT
    private Long userId;
    private String name;
    private String phone;

    @OneToMany(cascade = CascadeType.ALL, orphanRem
    @JoinColumn(name = "user_id_fk", referencedColu
    private List<OrderDetails> orderDetails = new A
```

```java
    @GetMapping("/user/{id}")
    public UserDetails testOrphan(@PathVariable Long id) {
        UserDetails output =  userDetailsService.findByID(id);
        output.getOrderDetails().remove( index: 0);
       userDetailsService.saveUser(output);
       return output;

    }
}
```

```java


    // Constructors
    public UserDetails() {
    }

    //getters and setters
}
```

1st: INSERT

localhost:8080/api/user

POST localhost:8080/api/user

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "productName": "IceCream"
        },
        {
            "productName": "ColdDrinks"
        }
    ]
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "userId": 2,
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "id": 3,
            "productName": "IceCream"
        },
        {
            "id": 4,
            "productName": "ColdDrinks"
        }
    ]
}
```

Run | Run Selected | Auto complete | Cle

```
SELECT * FROM USER_DETAILS
```

SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE
1	JohnXYZ	1234567890

(1 row, 1 ms)

Run | Run Selected | Auto com

```
SELECT * FROM ORDER_DE
```

SELECT * FROM ORDER_DE

ID	USER_ID_FK	PRODU
1	1	IceCrear
2	1	ColdDrin

(2 rows, 4 ms)

2nd: Testing Orphan Removal (With OrphanRemoval = false)

localhost:8080/api/user/1

GET localhost:8080/api/user/1

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "userId": 1,
```

Run | Run Selected | Auto complete | Cle

```
SELECT * FROM USER_DETAILS
```

SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE
1	JohnXYZ	1234567890

Run | Run Selected | Aut

```
SELECT * FROM ORDER
```

SELECT * FROM ORDE

ID	USER_ID_FK	PR

```
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "id": 2,
            "productName": "ColdDrinks"
        }
    ]
}
```

(1 row, 1 ms)

1	*null*	Ice(
2	1	Col

(2 rows, 0 ms)

Console Output: Persistence Context knows all the info, which is present in m

```
Hibernate:
    select
        ud1_0.user_id,
        ud1_0.name,
        ud1_0.phone
    from
        user_details ud1_0
    where
        ud1_0.user_id=?
Hibernate:
    select
        od1_0.user_id_fk,
        od1_0.id,
        od1_0.product_name
    from
        order_details od1_0
    where
        od1_0.user_id_fk=?
Hibernate:
    update
        order_details
    set
        user_id_fk=null
    where
        user_id_fk=?
        and id=?
```

```
@Table(name = "user_details")
@Entity
public class UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long userId;
    private String name;
    private String phone;

    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "user_id_fk", referencedColumnName = "userId")
    private List<OrderDetails> orderDetails = new ArrayList<>();


    // Constructors
    public UserDetails() {
    }

    //getters and setters
}
```

localhost:8080/api/user/1

GET localhost:8080/api/user/1

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "userId": 1,
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "id": 2,
            "productName": "ColdDrinks"
        }
    ]
}
```


Run Run Selected Auto complete Cle

```
SELECT * FROM USER_DETAILS
```

SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE
1	JohnXYZ	1234567890

(1 row, 1 ms)

Run Run Selected Auto complete Clear

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

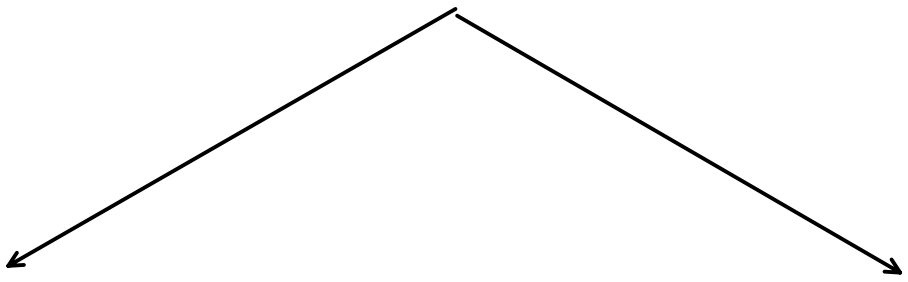
ID	USER_ID_FK	PRODUCT_NAME
2	1	ColdDrinks

(1 row, 1 ms)

```
Hibernate:
    select
        ud1_0.user_id,
        ud1_0.name,
        ud1_0.phone
    from
        user_details ud1_0
    where
        ud1_0.user_id=?
Hibernate:
    select
        od1_0.user_id_fk,
        od1_0.id,
        od1_0.product_name
    from
        order_details od1_0
    where
        od1_0.user_id_fk=?
Hibernate:
    update
        order_details
    set
        user_id_fk=null
    where
        user_id_fk=?
        and id=?
Hibernate:
    delete
    from
        order_details
    where
        id=?
```

One-to-Many: Bidirectional

- Parent reference to child.
- Each Child reference to Parent.



Inverse Side

Owning Side

- No Foreign key is created in table.
- Only holds ***Object*** reference of owing entity.

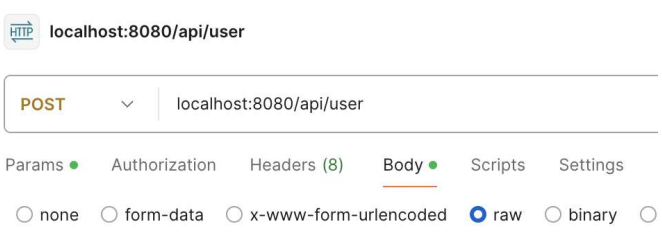
- Holds the Foreign Key relationship in a table.

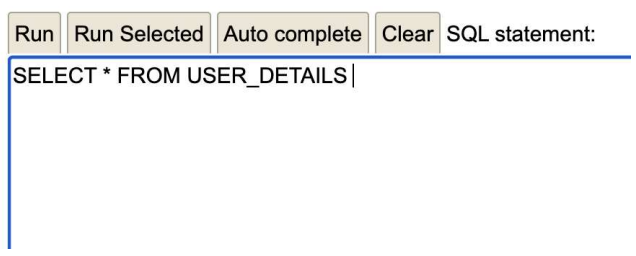
```java
@Table(name = "user_details")
@Entity
@JsonIdentityInfo(
        generator = ObjectIdGenerators.PropertyGenerator.class,
        property = "userId")
public class UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long userId;
    private String name;
    private String phone;
    @OneToMany(mappedBy = "userDetails", cascade = CascadeType.ALL)
    private List<OrderDetails> orderDetails = new ArrayList<>();

    public Long getUserId() {
        return userId;
    }
    public void setUserId(Long userId) {
        this.userId = userId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
    public List<OrderDetails> getOrderDetails() {
        return orderDetails;
    }
    public void setOrderDetails(List<OrderDetails> orderDetails) {
        this.orderDetails = orderDetails;
        for(OrderDetails order: orderDetails) {
            order.setUserDetails(this);
        }
    }
}
```

```java
@Table(name = "order_details")
@Entity
@JsonIdentityInfo(
        generator = ObjectIdGenerators.PropertyGenerator.
        property = "id")
public class OrderDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String productName;
    @ManyToOne
    @JoinColumn(name = "user_id_owing_fk", referencedColu
    private UserDetails userDetails;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getProductName() {
        return productName;
    }
    public void setProductName(String productName) {
        this.productName = productName;
    }
    public UserDetails getUserDetails() {
        return userDetails;
    }
    public void setUserDetails(UserDetails userDetails) {
        this.userDetails = userDetails;
    }
}
```





```
{
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "productName": "IceCream"
        },
        {
            "productName": "ColdDrinks"
        }
    ]
}
```


Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "userId": 1,
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "id": 1,
            "productName": "IceCream",
            "userDetails": 1
        },
        {
            "id": 2,
            "productName": "ColdDrinks",
            "userDetails": 1
        }
    ]
}
```


SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE
1	JohnXYZ	1234567890

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ID	USER_ID_OWING_FK	PRODUCT_NAME
1	1	IceCream
2	1	ColdDrinks

(2 rows, 1 ms)

localhost:8080/api/user/1

GET localhost:8080/api/user/1

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "userId": 1,
    "name": "JohnXYZ",
    "phone": "1234567890",
    "orderDetails": [
        {
            "id": 1,
            "productName": "IceCream",
            "userDetails": 1
        },
        {
            "id": 2,
            "productName": "ColdDrinks",
            "userDetails": 1
        }
    ]
}
```


Many-to-One: Unidirectional

- In this, we generally talk from the perceptive of Child like : Many Orders can be plac 1 User.
- So still, User is considered as Parent and Orders as Child.
- Parent don’t have reference to child.
- Each child has reference to parent.

```java
@RestController
@RequestMapping(value = "/api/")
public class OrderController {

    @Autowired
    OrderService orderService;

    @GetMapping("/order/{id}")
    public OrderDetails fetchUser(@PathVariable Long id) {
        return orderService.findByID(id);
    }

    @PostMapping(path = "/order")
    public OrderDetails insertOrder(@RequestBody OrderDetails orderDetails) {
        return orderService.saveOrder(orderDetails);
    }
}
```

```java
@Service
public class OrderService {

    @Autowired
    OrderDetailsRepository orderDetailsRepositor

    public OrderDetails findByID(Long primaryKey
        return  orderDetailsRepository.findById(
    }

    public OrderDetails saveOrder(OrderDetails o
        return orderDetailsRepository.save(order
    }
}
```

```java
@Table(name = "user_details")
@Entity
public class UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long userId;
    private String name;
    private String phone;

  //getters and setters
}
```

```java
@Table(name = "order_details")
@Entity
public class OrderDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.ID
    private Long id;
    private String productName;
    @ManyToOne (cascade = CascadeType.ALL)
    @JoinColumn(name = "user_id_owing_fk", refer
    private UserDetails userDetails;

    //getter and setter
}
```

POST localhost:8080/api/order

Params • Authorization Headers (8) Body • Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```json
{
    "productName": "IceCream",
    "userDetails":
        {
            "name": "SJ",
            "phone": "111212121221"
        }
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```json
{
    "id": 1,
    "productName": "IceCream",
    "userDetails": {
        "userId": 1,
        "name": "SJ",
        "phone": "111212121221"
    }
}
```

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM USER_DETAILS
```

SELECT * FROM USER_DETAILS;

USER_ID	NAME	PHONE
1	SJ	111212121221

(1 row, 1 ms)

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ID	USER_ID_OWING_FK	PRODUCT_NAME
1	1	IceCream

(1 row, 1 ms)

Try it out:

Many-to-One: Bidirectional would be same as OneToMany Bidirectional

Many-to-Many: Unidirectional

- Reference from One way only

```java
@Table(name = "order_details")
@Entity
public class OrderDetails {

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


    @ManyToMany (cascade = CascadeType.ALL)
    @JoinTable(
            name = "order_product", // new Join table name
            joinColumns = @JoinColumn(name = "order_id"), // Foreign key for Order
            inverseJoinColumns = @JoinColumn(name = "product_id") // Foreign key for Product
    )
    private List<ProductDetails> productDetails = new ArrayList<>();

    public Long getOrderNo() {
        return orderNo;
    }

    public void setOrderNo(Long orderNo) {
        this.orderNo = orderNo;
    }

    public List<ProductDetails> getProductDetails() {
        return productDetails;
    }

    public void setProductDetails(List<ProductDetails> productDetails) {
        this.productDetails = productDetails;
    }
}
```

```java
@Table(name = "product_details")
@Entity
public class ProductDetails {

    @Id
    @GeneratedValue(strategy = GenerationTy
    private Long productId;

    private String name;
    private double price;

   //getters and setters
}
```

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ORDER_NO

(no rows, 1 ms)

Run | Run Selected | Auto complete | Clear | SQL statem

```
SELECT * FROM PRODUCT_DETAILS
```

SELECT * FROM PRODUCT_DETAILS;

PRICE	PRODUCT_ID	NAME

(no rows, 0 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM ORDER_PRODUCT
```

SELECT * FROM ORDER_PRODUCT;

ORDER_ID	PRODUCT_ID

(no rows, 1 ms)

```java
@RestController
```

```java
@RestController
@RequestMapping(value = "/api/")
public class OrderController {

    @Autowired
    OrderService orderService;
```

```java
@RequestMapping(value = "/api/")
public class ProductController {

    @Autowired
    ProductService productService;

    @PostMapping(path = "/product")
    public ProductDetails insertUser(@RequestBody ProductDetails product) {
        return productService.saveProduct(product);
    }
}
```

```java
    @Autowired
    ProductService productService;

    @GetMapping("/order/{id}")
    public OrderDetails fetchUser(@PathVa
        return orderService.findByID(id);
    }

    @PostMapping(path = "/order")
    public OrderDetails insertOrder(@Requ

        List<ProductDetails> managedProdu
                .map(product -> productSe
                .collect(Collectors.toLis

        orderDetail.setProductDetails(man
        return orderService.saveOrder(ord
    }

}
```

1st : CREATE PRODUCTS

POST localhost:8080/api/product

Params • Authorization Headers (8) Body • Scripts Settings

none form-data x-www-form-urlencoded raw binary

```
{
    "name" : "ice-cream",
    "price" : "100"
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "productId": 1,
    "name": "ice-cream",
    "price": 100.0
}
```

POST localhost:8080/api/product

Params • Authorization Headers (8) Body • Scripts Settings

none form-data x-www-form-urlencoded raw binary Graph(

```
{
    "name" : "cold-drink",
    "price" : "150"
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "productId": 2,
    "name": "cold-drink",
    "price": 150.0
}
```

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM PRODUCT_DETAILS
```

SELECT * FROM PRODUCT_DETAILS;

PRICE	PRODUCT_ID	NAME
100.0	1	ice-cream
150.0	2	cold-drink

(2 rows, 2 ms)

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ORDER_NO

(no rows, 1 ms)

2nd : CREATE Multiple Orders

POST localhost:8080/api/order

Params • Authorization Headers (8) Body • Scripts Se

none form-data x-www-form-urlencoded raw b

```
{
    "productDetails": [
        {
            "productId": 1

        },
        {
            "productId": 2

        }
    ]
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "orderNo": 1,
    "productDetails": [
        {
            "productId": 1,
            "name": "ice-cream",
            "price": 100.0
        },
        {
            "productId": 2,
            "name": "cold-drink",
            "price": 150.0
        }
    ]
}
```

POST localhost:8080/api/order

Params • Authorization Headers (8) Body • Scripts S

none form-data x-www-form-urlencoded raw

```
{
    "productDetails": [

        {
            "productId": 2

        }
    ]
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "orderNo": 2,
    "productDetails": [
        {
            "productId": 2,
            "name": "cold-drink",
            "price": 150.0
        }
    ]
}
```

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM ORDER_DETAILS
```

SELECT * FROM ORDER_DETAILS;

ORDER_NO
1
2

(2 rows, 1 ms)

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM PRODUCT_DETAILS
```

SELECT * FROM PRODUCT_DETAILS;

PRICE	PRODUCT_ID	NAME
100.0	1	ice-cream
150.0	2	cold-drink

(2 rows, 1 ms)

Run Run Selected Auto complete Clear SQL st

```
SELECT * FROM ORDER_PRODUCT
```

SELECT * FROM ORDER_PRODUCT;

ORDER_ID	PRODUCT_ID
1	1
1	2
2	2

(3 rows, 1 ms)

Many-to-Many: Bidirectional

- Since its Many to Many, anyone can be Owing and inverse side.

```
@Table(name = "order_details")
@Entity
public class OrderDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
@Table(name = "product_details")
@Entity
```

```
    private Long orderNo;

    @ManyToMany (cascade = CascadeType.ALL)
    @JoinTable(
            name = "order_product", // new Join table name
            joinColumns = @JoinColumn(name = "order_id"), // Foreign key for Order
            inverseJoinColumns = @JoinColumn(name = "product_id") // Foreign key for Product
    )
    private List<ProductDetails> productDetails = new ArrayList<>();

    public Long getOrderNo() {
        return orderNo;
    }

    public void setOrderNo(Long orderNo) {
        this.orderNo = orderNo;
    }

    public List<ProductDetails> getProductDetails() {
        return productDetails;
    }

    public void setProductDetails(List<ProductDetails> productDetails) {
        this.productDetails = productDetails;
    }
}
```

```
public class ProductDetails {

    @Id
    @GeneratedValue(strategy = GenerationTy
    private Long productId;

    private String name;
    private double price;

    @ManyToMany(mappedBy = "productDetails"
    @JsonIgnore
    List<OrderDetails> orders = new ArrayLi

    //GETTERS AND SETTERS
}
```