===================== One To Many ===============================

Unidirectional OneToMay conclusion : Coustomer ni use chesi Orders ni get cheyachhu but Orders ni vadi coustomer ni get cheuyalemu

Coustomer controller postman out put: user ni get chesinappudu user tho patu Orders details kuda vasthay

[

    {

        "name": "chandu",

        "orders": [

            {

                "product": "TV",

                "price": 2345.00,

                "orderid": 111

            },

            {

                "product": "Mobile",

                "price": 434.00,

                "orderid": 112

            }

        ],

        "customerid": 1

    }

]

Orders controller postman out put: Orders ni get chesinappudu only Items yee get avuthay

   {

        "product": "TV",

        "price": 2345.00,

        "orderid": 111

    },

Bidirectional OneToMany conclusion : Coustomer ni use chesi Ordersni alane Orders ni vadi coustomer ni Rendityni get cheyachhu

Commnet added //@JsonIgnore in parent (User) tabel

Bidirectional\_Coustomer\_Controller post man out put

 {

        "id": 2,

        "name": "kiran",

        "orders": [

            {

                "id": 222,

                "product": "Mobile",

                "price": 25434.00

            }

        ]

Bidirectional\_orders\_Controller post man out put

[

    {

        "id": 111,

        "product": "TV",

        "price": 4544.00

    },

    {

        "id": 222,

        "product": "Mobile",

        "price": 25434.00

    },

commented Added //@JsonIgnore in child (Add) tabel

Bidirectional\_Coustomer\_Controller post man out put

[

    {

        "id": 1,

        "name": "chandu"

    },

    {

        "id": 2,

        "name": "kiran"

    }

]

Bidirectional\_Orders\_Controller post man out put

[

    {

        "id": 111,

        "product": "TV",

        "price": 4544.00,

        "customer": {

            "id": 1,

            "name": "chandu"

        }

    },

    {

        "id": 222,

        "product": "Mobile",

        "price": 25434.00,

        "customer": {

            "id": 2,

            "name": "kiran"

        }

},

Comment Removed //@JsonIgnore both parent and child

Bidirectional\_Coustomer\_Controllerpost man out put

[

    {

        "id": 1,

        "name": "chandu"

    },

    {

        "id": 2,

        "name": "kiran"

    }

]

Bidirectional\_Orders\_Controllerpost man out put

{

        "id": 111,

        "product": "TV",

        "price": 4544.00

    },

    {

        "id": 222,

        "product": "Mobile",

        "price": 25434.00

    },

OneToMany Mapping main aim :

// Get Address details by User ID

// Get User details by Address ID

================ Unidirectional OneToMany coding ===================

**// Get Order details by Customer ID**

**Unidirectional\_Orders\_Controller**

@GetMapping("/ordersByCustomer/{customerId}")

public List<Unidirectional\_Orders> getOrdersByCustomerId(@PathVariable Long customerId) {

// Fetch the customer by ID

Optional<Unidirectional\_Customer> customerOptional = customerRepo.findById(customerId);

if (customerOptional.isPresent()) {

// Return the orders list from the customer entity

return customerOptional.get().getOrders();

} else {

// If customer is not found, return an empty list or handle the error accordingly

return new ArrayList<>();

}

}

**Unidirectional\_Customer\_Repo**

public interface Unidirectional\_Customer\_Repo extends JpaRepository<Unidirectional\_Customer, Long> {

Optional<Unidirectional\_Customer> findById(Long customerId);

}

Out put :

[

    {

        "product": "TV",

        "price": 2345.00,

        "orderid": 111

    },

    {

        "product": "Mobile",

        "price": 434.00,

        "orderid": 112

    }

]

Conculstion : Get Order details by Customer ID ni achive cheyali ante Repo and controller lo chnages cheyali

================ Bidirectional OneToMany coding ===================

**// Get Order details by Customer ID**

**Bidirectional\_Orders\_Controler**

// Get Order details by Customer ID

@GetMapping("/orders/customer/{customerId}")

public ResponseEntity<List<Bidirectional\_Orders>> getOrdersByCustomerId(@PathVariable Long customerId) {

// Fetch orders by customerId

List<Bidirectional\_Orders> orders = BidirectionalOrdersRepo.findByCustomer\_Id(customerId);

if (orders.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(null);

}

return ResponseEntity.ok(orders);

}

**Bidirectional\_Orders\_Repo**

public interface Bidirectional\_Orders\_Repo extends JpaRepository<Bidirectional\_Orders, Long> {

List<Bidirectional\_Orders> findByCustomer\_Id(Long customerId);

}

Out put : http://localhost:8080/BIOrders/orders/customer/1

[

    {

        "id": 111,

        "product": "TV",

        "price": 4544.00

    },

    {

        "id": 333,

        "product": "Notes",

        "price": 2112.00

    }

]

**// Get Customer details by Order ID**

**Bidirectional\_Customer\_Controler**

@GetMapping("/customer/order/{orderId}")

public ResponseEntity<Bidirectional\_Customer> getCustomerByOrderId(@PathVariable Long orderId) {

// Find the order by its ID using the correct repository (Bidirectional\_Orders\_Repo)

Optional<Bidirectional\_Orders> orderOptional = BidirectionalOrdersRepo.findById(orderId);

if (orderOptional.isPresent()) {

// Get the associated customer from the order

Bidirectional\_Customer customer = orderOptional.get().getCustomer();

return ResponseEntity.ok(customer);

} else {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(null);

}

}

Out put : http://localhost:8080/BICustomer/customer/order/111

{

    "id": 1,

    "name": "chandu"

}

Conculstion : Get Order details by Customer ID ni achive cheyali ante Repo and controller lo chnages cheyali

----------------------- Doubts -----------------------------

why i created internal method in Bidirectional\_Orders\_Repo

public interface Bidirectional\_Orders\_Repo extends JpaRepository<Bidirectional\_Orders, Long> {

List<Bidirectional\_Orders> findByCustomer\_Id(Long customerId);

}

give clarity which entity have direct reference and which entity don’t have direct reference ?

Great question! Let me clarify why you created an internal method like findByCustomer\_Id(Long customerId) in the repository, and how it fits with your understanding of direct references in entities.

### Key Concept: Direct References vs. Indirect (Mapped) References

**Direct Reference**:

* 1. A **direct reference** means that an entity explicitly contains a foreign key field (i.e., it has a field that points to another entity).
  2. In the case of Bidirectional\_Orders, the foreign key column (customer\_id) **directly references** Bidirectional\_Customer because of the @ManyToOne and @JoinColumn annotations. So, Bidirectional\_Orders **has a direct reference to** Bidirectional\_Customer through the customer field.

**Indirect (Mapped) Reference**:

* 1. In **Bidirectional\_Customer**, the reference to Bidirectional\_Orders is **indirect** (via the mappedBy attribute). The mappedBy = "customer" indicates that the mapping for this relationship is managed by the customer field in Bidirectional\_Orders, but **there is no direct foreign key** in Bidirectional\_Customer. Instead, it is a collection of orders (List<Bidirectional\_Orders>) that references orders related to a customer.

### Why you created findByCustomer\_Id(Long customerId):

You are right to notice that you've created a custom method in the Bidirectional\_Orders\_Repo repository:

public interface Bidirectional\_Orders\_Repo extends JpaRepository<Bidirectional\_Orders, Long> {

List<Bidirectional\_Orders> findByCustomer\_Id(Long customerId);

}

The reason this method is needed is **because you want to query** Bidirectional\_Orders **based on the** customer\_id **foreign key** in that table. The findByCustomer\_Id(Long customerId) method essentially performs a **search by the foreign key** (customer\_id), which is stored in the Bidirectional\_Orders table.

* Even though Bidirectional\_Orders **has a direct reference** to Bidirectional\_Customer, you can't automatically navigate to Bidirectional\_Orders using the customer object in Bidirectional\_Customer without the custom query.
* JPA doesn't automatically provide a way to query for all Bidirectional\_Orders just by the customerId directly without defining a method like findByCustomer\_Id. This is because Bidirectional\_Customer **does not store any foreign key to** Bidirectional\_Orders, and thus, querying the Bidirectional\_Orders table based on a customer's ID requires this custom method.

### Entity Relationship Recap:

1. **Bidirectional\_Orders**:
   * **Direct reference** to Bidirectional\_Customer through the @ManyToOne and @JoinColumn.
   * This means Bidirectional\_Orders contains a foreign key (customer\_id) linking to Bidirectional\_Customer.
2. **Bidirectional\_Customer**:
   * **Indirect reference** to Bidirectional\_Orders through the @OneToMany(mappedBy = "customer").
   * This doesn't add a foreign key column in the Bidirectional\_Customer table. Instead, it manages the relationship through the customer field in Bidirectional\_Orders.

### Why is findByCustomer\_Id Needed?

* You need to use findByCustomer\_Id because the foreign key (customer\_id) is stored in Bidirectional\_Orders, not in Bidirectional\_Customer.
* The @ManyToOne relationship in Bidirectional\_Orders has the foreign key to Bidirectional\_Customer. When you want to find all orders related to a specific customer (identified by customerId), you query the Bidirectional\_Orders table by this foreign key, which is exactly what findByCustomer\_Id does.

### Conclusion:

* Bidirectional\_Orders **has a direct reference** to Bidirectional\_Customer because it contains the foreign key (customer\_id).
* Bidirectional\_Customer **has an indirect reference** to Bidirectional\_Orders because it holds a collection (List<Bidirectional\_Orders>) that is mapped by the customer field in Bidirectional\_Orders, but it doesn't have a foreign key column in the Bidirectional\_Customer table.
* **You created findByCustomer\_Id because JPA does not automatically generate queries that use the foreign key (customer\_id) in Bidirectional\_Orders to retrieve all orders for a given customer. This is why you needed a custom repository method to perform the query.**

so that means i have FK in Bidirectional\_Orders in this table i have customer\_id column also if the column is there also the JPA does not automatically generate queries that use the foreign key (customer\_id) right , suppose if customer\_id is not FK means JPA will write query right so JPA will not write query directly due to that we created custom method in repo right ?

You're almost there! Let's refine the understanding:

### Key Points to Clarify:

**Foreign Key in Bidirectional\_Orders (customer\_id)**:

* 1. Yes, in the Bidirectional\_Orders table, you **do** have a foreign key column (customer\_id) that references the Bidirectional\_Customer table.
  2. This foreign key is created because of the @ManyToOne annotation in Bidirectional\_Orders and the @JoinColumn(name = "customer\_id") that specifies the actual foreign key column.

**JPA and Query Generation**:

* 1. **JPA does generate queries** for you, but the automatic query generation depends on **how you define the repository methods**.
  2. JPA **does not automatically generate a query** when you simply want to query based on the foreign key column (customer\_id) unless you specifically define a method in your repository, like findByCustomer\_Id(Long customerId).
     1. **Why?** Because, in JPA, querying based on a foreign key column (like customer\_id) in the associated entity requires a specific query method in the repository. It doesn't automatically generate such a method unless the relationship is explicitly navigated using the entity references.

**What JPA Automatically Generates**:

* 1. If you were querying for a **customer's orders** via the relationship, JPA could generate a query using the associated **Java objects** (like the customer field in Bidirectional\_Orders).
     1. For example, if you used something like Bidirectional\_Customer customer in a method, JPA could automatically generate a query to get all orders for a particular customer.
  2. But, **since you're directly querying the** customer\_id **column in** Bidirectional\_Orders, JPA doesn't automatically know to generate that query based on the foreign key unless you define it yourself. This is why you created the custom method findByCustomer\_Id(Long customerId) in the repository.

**If** customer\_id **is Not a Foreign Key (FK)**:

* 1. If customer\_id in Bidirectional\_Orders was **not a foreign key** and you had no relationship between the entities, JPA would **not** know that customer\_id should link to a Bidirectional\_Customer.
  2. In that case, JPA wouldn't automatically create a query to fetch orders by customer, because there wouldn't be an entity relationship defined. So, in that case, you'd need to manually query the database using @Query or custom methods like you did.

### Summary:

* **JPA does not automatically generate a query** to fetch orders using the foreign key (customer\_id) unless you explicitly define a repository method, like findByCustomer\_Id(Long customerId). The reason is that JPA needs to know how to map that relationship, and in this case, you want to query by the foreign key, so a custom method is needed.
* **If** customer\_id **is not a foreign key**, there would be no direct relationship between Bidirectional\_Orders and Bidirectional\_Customer, and JPA wouldn't know to generate any query to fetch orders by customer\_id because there would be no entity mapping.

So yes, you created a custom method in the repository because you wanted to query using the foreign key (customer\_id), and JPA doesn't automatically generate such queries unless you explicitly define a method to do so.

Final Conclustion : FK meda direct ga JAP query rayaledu

===================== \*\*\*\* \*\*\*\*\* ===============================

**Unidirectional OneToMany conclusion :**

Parent table Entity (Customer) ni vadi Child table Entity (Orders) details get cheyachhu kani / Child table Entity (Orders) ni vadi Parent table Entity details (Customer) ni get cheyalamu

**How table script in DB**

child table (Orders) lo FK script anedi create avuthundi Parent table lo kadu ( table Script in DB)

**How data insertion will work in DB**

UNIQUE KEY anedi create cheyamu kanuka oka coustomer ki vunna order\_id anedi yenthamandi users ki avina vunadachhu

**key note** : Unidirectional OneToMany lo Parent table Entity class lo mapping relation rastamu. Child table Entity class lo rayamu (java code lo )

same like OneToOne lagane with out UNIQUE key

**Bidirectional OneToMany conclusion :**

Bidirectional lo Parent table Entity (Customer) nuodi Child table Entity (Orders) details alane Child table Entity (Orders) nuodi Parent table Entity (Customer) deatils rendu get cheyachhu

**key note** : Bidirectional OneToMany lo Parent and Child Rendu Entity class lo mapping relation rastamu ( java code lo ) mapping relation direct and indirect both

Enika remaining all same table scrpi / data insertion in data all same, only Java coding lo ne change by codig dwara details get chesukovadame diffrent

**Note :** Bidirectional OneToMany vadinappudu post man responces anedi infinite data loop anedi avuthundi danni over come cheyataniki

"@JsonIgnore" Entity class lo rastham