**Spring Boot | JPA / Hibernate Composite Primary Key Example**

First let’s understand what is primary key and its uses. Whenever we want to define a primary key with combination of multiple column that concept is called as a composite primary key.

For example, let’s assume I have a user table and in this User table usually we took ID as a primary key. But sometimes we can’t uniquely identify a User object just by his ID, so in that scenario we may need to combine multiple column to define a primary key.



For example, here I can combine ID and Email as a Single primary key. Or else I can define ID and Phone number. Because Email and phone number is always unique. So email can’t be duplicate and phone number can’t be duplicate. So, I need to combine such 2 columns to make a primary key.

Now let’s understand exact Real time scenario where we can implement Composite Primary Key:-

Let’s consider another example that manage Employee of a Company with multiple Departments. And as you know each employee has a unique Id within his own department. But the same ID can be assigned to different employee in another department.



For example, if u observed this structure here, I have an employee id **EMP\_101** and the name is **John**

And the email id and the phone number. This guy can be from **ADMIN** team and there may be a chance same Employee Id **EMP\_101** can be assigned to another Employee in another department, let’s say department could be **IT/SUPPORT**.

So, if u can observe there are 2 employees having the same employee Id, so we can’t uniquely identify Employee just by his Employee Id even they belong from different departments.



In such scenario we need to go for Composite Primary Key concept.

In such scenario to uniquely identified an Employee we need to know his employee Id as well as Department Id so that we can combine these 2 columns to make a Primary Key. For example, if you go to below Schema



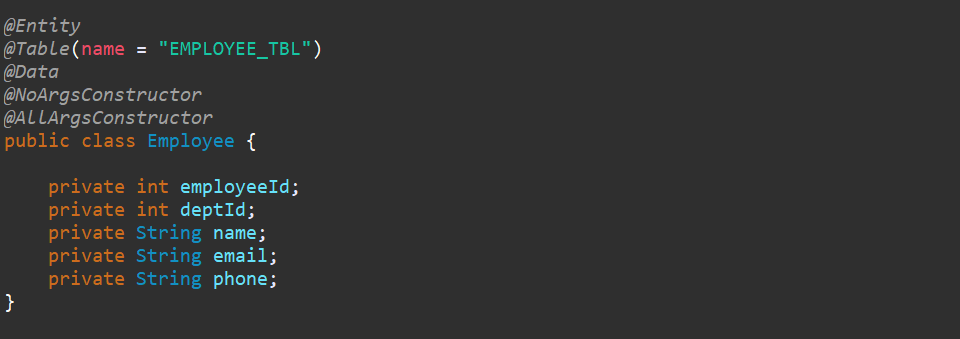
Here I have one employee and we have the Employee\_Id and Department\_Id, we need to combine these 2 (**Employee\_Id** and **Department\_Id**) columns to make a Primary Key. So, in an organization we can have a duplicate Employee Id but the department\_id is different. So, if you observed Employee EMP\_101 came from Dev team and EMP\_101 came form Devops. And the name is different, and the email and phone number is different. So, we may need to combine multiple columns to define a primary key. There can be some scenarios we need to go for composite primary key. And In spring data jpa the approach is not straight forward. There is multiple way to go with this composite primary key.

You can use the annotation **@IdClass** also there is another way we can use the annotation **@EmbededId.** so, I will walk you through both the ways. So, let’s get started with our implementation parts.

Application: **springboot-composite-key-main**

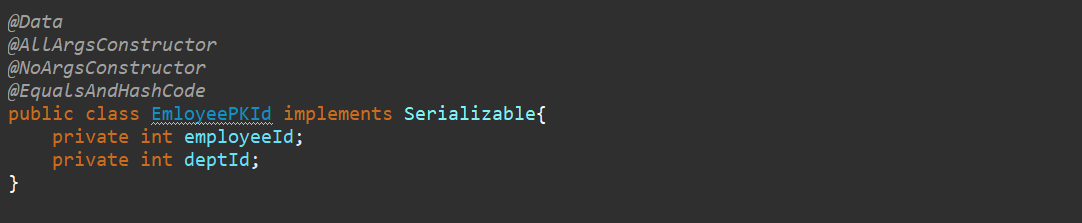
Dependencies: **Lombok, Spring Web, Spring Data JPA, MySQL Driver**

So, if u remember the architecture, we need to define these 2 **Employee\_Id** and **Department\_Id** as a Primary Key. So, we need to define these fields.

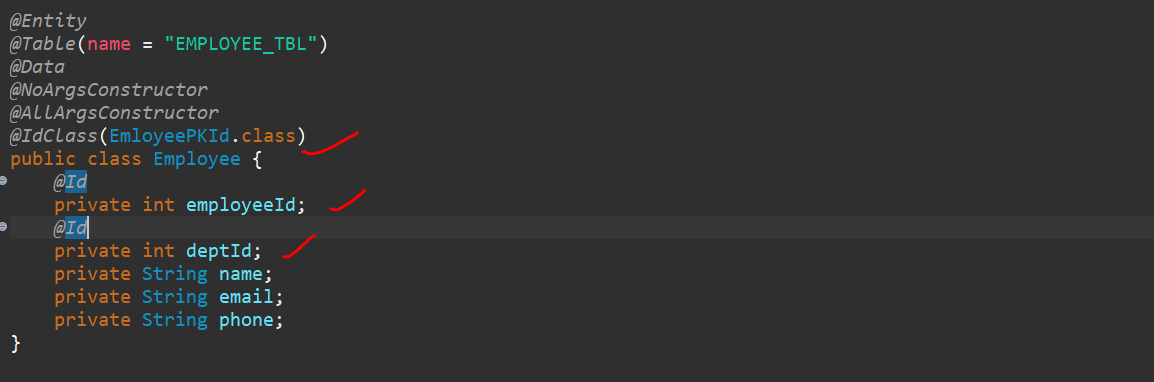


So, we need to define a Primary Key so we can’t annotate **@Id** here. Because employee\_id and deptId these 2 we need to consider as a primary key. So, what we can do let’s create another class give it name **EmloyeePKId.**

Now we need to implement this class **EmloyeePKId** from **Serializable.** Then we need to define these 2 fields here. Also, here we need to Implement **equals ()** and **hashcode ().** So, this is mandatory if u are using the composite primary key. If you are going for the **@IdClass** or **@EmbededId** must need to implement from Serializable and also you need to override these equals () and hashcode () methods. Now this **EmloyeePKId (**employeeId and deptId**)** will be consider as my Primary Key. So, now how I can specify in my Employee class.

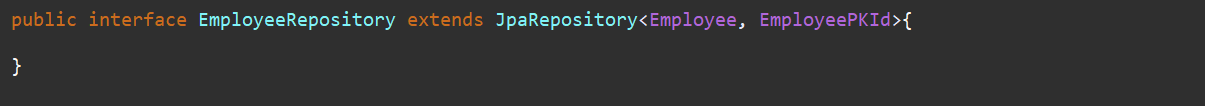


Now this **EmloyeePKId (**employeeId and deptId**)** will be consider as my Primary Key. So, now how I can specify in my Employee class. Simply I can annotate with **@IdClass (EmloyeePKId.class)** and I can give EmployeeId class. And also, both the field we need to annotate with **@Id.**



Which means we are telling the Hibernate or JPA we have 2 keys which we need to consider as a primary key.

Now let’s create a Repository…. Then extend it from JpaRepository. Specify the Model which is Employee then datatype of ur primary key. So, if u observed our datatype of primary key is **EmloyeePKId.**



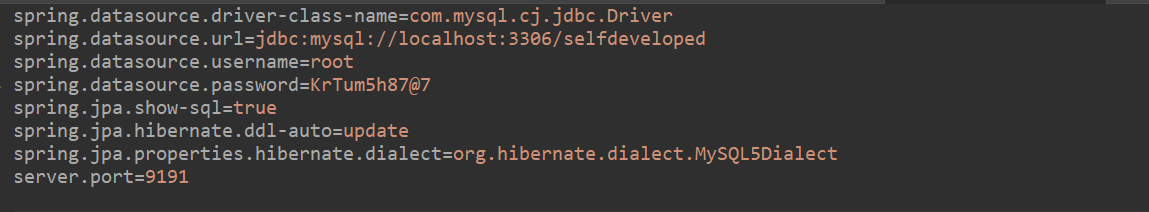
So now we have all Model, and we define our Composite Primary Key and we created our Repository.

Now let’s write the controller, I will write controller logic in my main class. For demo purpose I am injecting repository in my controller class, but u should do in ur service class.

**Let’s write POST and GET API**



Now let’s add Database related properties in our **application. properties** file.



Let’s run our application. Yaa, it created the table.

Hibernate: create table employee\_tbl (dept\_id integer not null, employee\_id integer not null, email varchar (255), name varchar (255), phone varchar (255), primary key (dept\_id, employee\_id)) engine=MyISAM

Now let’s go to the postman and hit the **save** rest endpoint. So that we can verify how it will not allow the duplicate key and how we can use this composite key.

Request:

{

  "employeeId":101,

  "departmentId":10,

  "name":"Kaushal",

  "email":"kaushal@gmail.com",

  "phone":"8749064048"

}

Response:

{

    "employeeId": 101,

    "deptId": 0,

    "name": "Kaushal",

    "email": "kaushal@gmail.com",

    "phone": "8749064048"

}

Sorry we need to pass the deptId instead of ~~department\_id~~.

Request:

{

  "employeeId":100,

  "deptId":11,

  "name":"Kaushal",

  "email":"kaushal@gmail.com",

  "phone":"8749064048"

}

Response:

{

    "employeeId": 100,

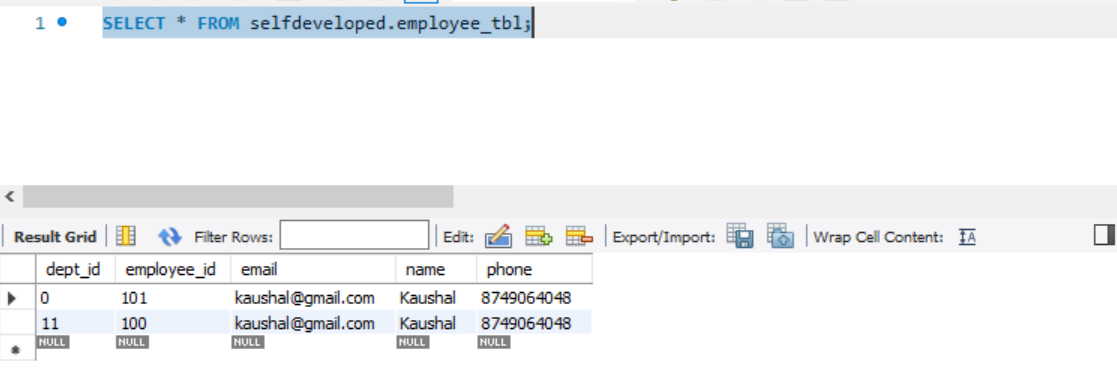
    "deptId": 11,

    "name": "Kaushal",

    "email": "kaushal@gmail.com",

    "phone": "8749064048"

}



1st row department\_id is 0 bcz we have passed the wrong name earlier.

So now 1 fresh correct record inserted. Now with the same department id and employee Id (11/100) I am trying to hit the same API. It won’t allow to save the particular record in our employee db. Because of the duplicate primary key. And if u observed here the primary key, we are considering both the employeeId and department\_id. And both are same, so it won’t allow.

In composite key at least one of primary key field should be different then only it will allow.

Even if I hit the rest endpoint multiple times it doesn’t allow to insert data into db, because of duplicate primary key.

Now if I will change anything here let’s say I am changing department\_id here. I will change it to 88. Now if I send even though employeeId is same 100 but the department\_id is different here.

{

  "employeeId":100,

  "deptId":88,

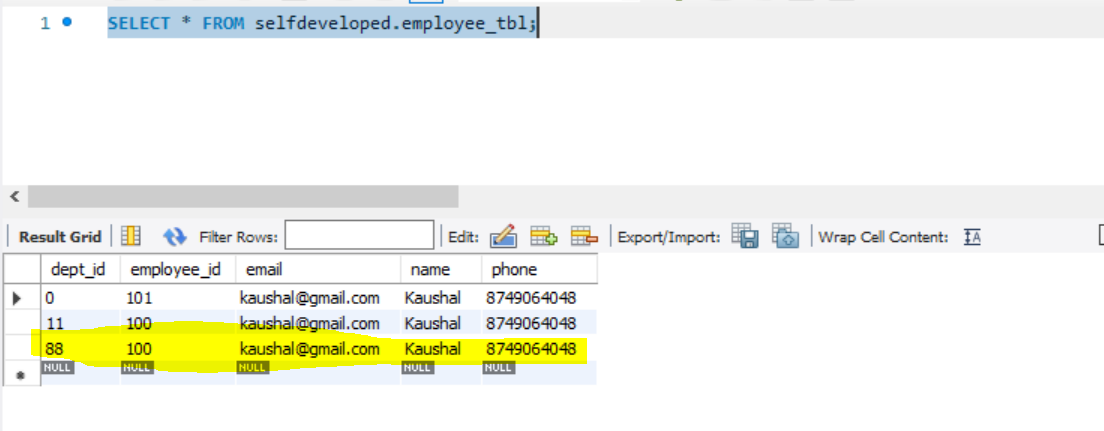
  "name":"Kaushal",

  "email":"kaushal@gmail.com",

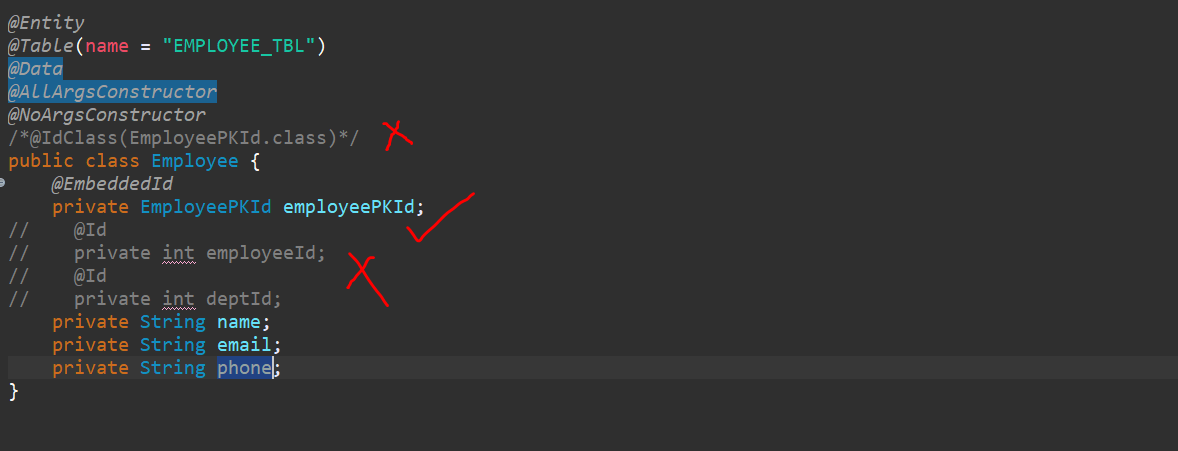
  "phone":"8749064048"

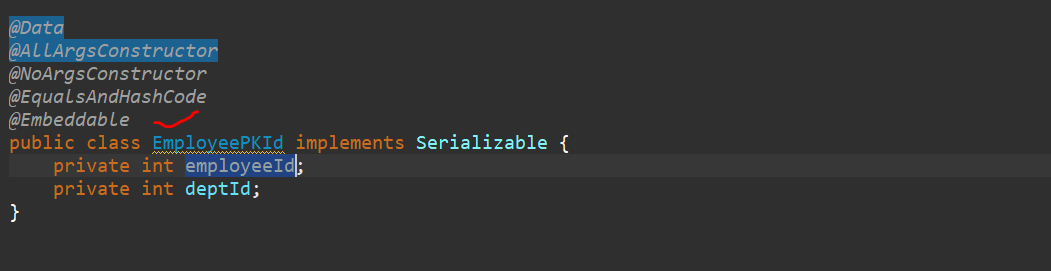
}

Now if I will go and verify in our DB. U can see the record bcz u changed the department\_id.



So, this is how we can go for the **@IdClass.** And already we mentioned we have another approach we can go for **@EmbededId.**





So, I don’t want to use @IdClass and @Id annotation…now remove the primary key from Student table we don’t want to keep repeatedly, I ll keep primary key related stuff in a single class…so just remove it.

So now in our primary id class we need to annotate **@Embeddable** and then go to your Entity and define this class and need to annotate **@EmbeddedId.** so, these all are the changes required for if u are using EmbededId.

And if u observed in EmbeddedId no need to repeatedly define the primary key. In case of IdClass we defined these 2 primary key here also in our EmloyeePKId class. But in case of Embeddable or EmbeddedId approach we just need to define it once that’s what the advantages of @Embeddable.

In request payload we need to pass StudentPKId as a separate object in our json.

Let me create the json structure…

**Url: -** <http://localhost:9090/employee>

Request Payload:

{

  "employeePKId":{

      "employeeId":200,

      "deptId":44

  },

  "name":"Kaushal",

  "email":"kaushal@gmail.com",

  "phone":"1234567890"

}

Response:

{

    "employeePKId": {

        "employeeId": 200,

        "deptId": 44

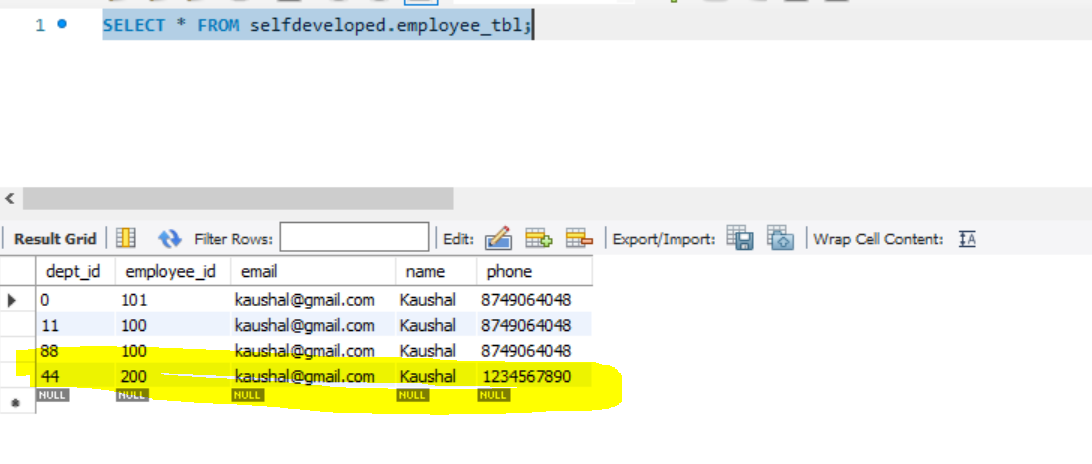
    },

    "name": "Kaushal",

    "email": "kaushal@gmail.com",

    "phone": "1234567890"

}



So with the same employeeId and department\_id if u tried multiple time also it will not record multiple record now if I will change anything either department id or employee id the changes will reflect here…

So this is how we can Implement composite primary key concept in Spring Data JPA using @IdClass or @EmbeddedId.

**Complete Code Using @IdClass**

POST <http://localhost:9090/employee>

Request:

**{**

**"employeeId":300,**

**"deptId":388,**

**"name":"Bharat",**

**"email":"bharat@gmail.com",**

**"phone":"8749064048"**

**}**

Response:

**{**

**"employeeId": 300,**

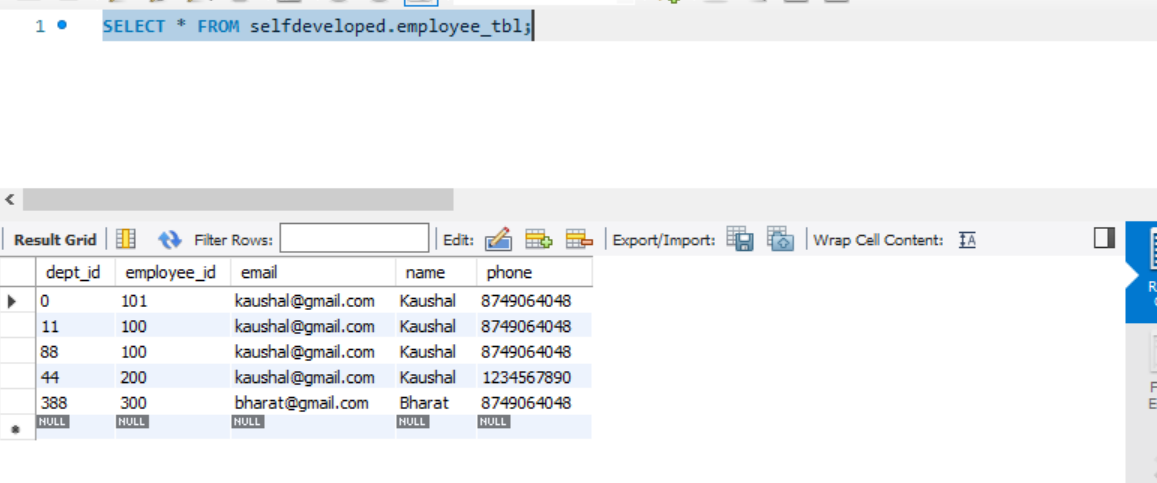
**"deptId": 388,**

**"name": "Bharat",**

**"email": "bharat@gmail.com",**

**"phone": "8749064048"**

**}**





***@Entity***

***@Table*(name = "EMPLOYEE\_TBL")**

***@Data***

***@AllArgsConstructor***

***@NoArgsConstructor***

***@IdClass*(EmployeePKId.class)**

**public class Employee {**

**// @EmbeddedId**

**// private EmployeePKId employeePKId;**

***@Id***

**private int employeeId;**

***@Id***

**private int deptId;**

**private String name;**

**private String email;**

**private String phone;**

**}**

**----------------------------------------**

***@Data***

***@AllArgsConstructor***

***@NoArgsConstructor***

***@EqualsAndHashCode***

**//@Embeddable**

**public class EmployeePKId implements Serializable {**

**private int employeeId;**

**private int deptId;**

**}**

**Complete Code Using @Embedded**

POST <http://localhost:9090/employee>

Request:

**{**

**"employeePKId":{**

**"employeeId":700,**

**"deptId":77**

**},**

**"name":"Puppy",**

**"email":"puppy@gmail.com",**

**"phone":"9262456789"**

**}**

Response:

**{**

**"employeePKId": {**

**"employeeId": 700,**

**"deptId": 77**

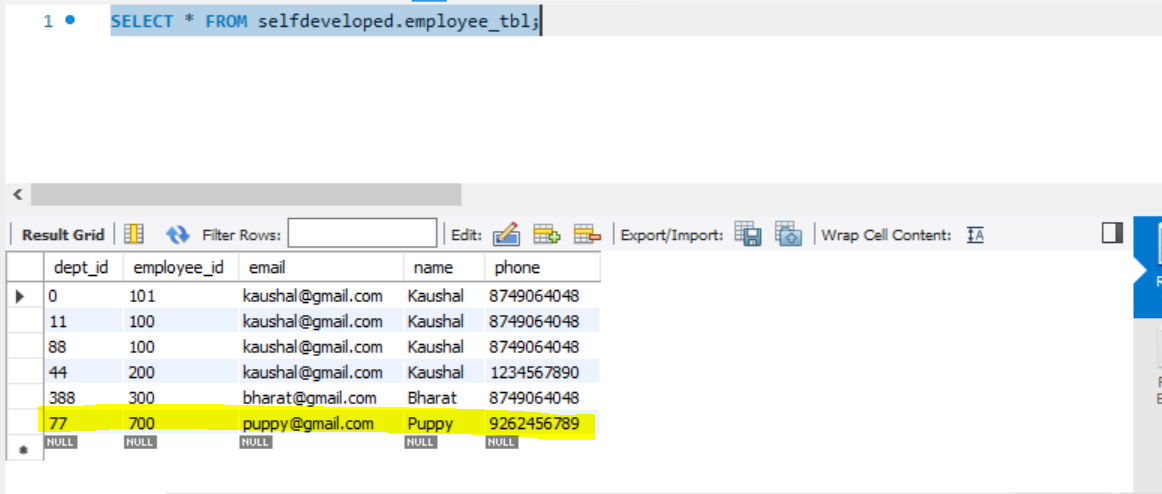
**},**

**"name": "Puppy",**

**"email": "puppy@gmail.com",**

**"phone": "9262456789"**

**}**



***@*Entity**

***@*Table(name = "EMPLOYEE\_TBL")**

***@*Data**

***@*AllArgsConstructor**

***@*NoArgsConstructor**

**//@IdClass(EmployeePKId.class)**

**public class Employee {**

***@*EmbeddedId**

**private EmployeePKId employeePKId;**

**// @Id**

**// private int employeeId;**

**// @Id**

**// private int deptId;**

**private String name;**

**private String email;**

**private String phone;**

**}**

**--------------------**

***@Data***

***@AllArgsConstructor***

***@NoArgsConstructor***

***@EqualsAndHashCode***

***@Embeddable***

**public class EmployeePKId implements Serializable {**

**private int employeeId;**

**private int deptId;**

**}**