

## JPA - Java Persistence Api

- Persistence --> storing the data permanently
- In Java, we use JDBC API to store, retrieve, manipulate the data from java program to database.
- then, why we need of JPA or learning of it ?
- JPA is a framework of java. which is used for saving the data.
- with JPA we can save the data, retrieve the data.
- JDBC is typically tightly coupled to database. what ever the jdbc code was written, that has the tightly coupled
- JPA is an ORM Tool.
- ORM --- Object Relational Mapping.
- to achieve the loosely coupled nature, ORM framework is used.

In ORM there are 3 key principles are there, Developer no need to write JDBC Code.

- every java class is a table.
- every property present in a class are columns for the table.
- every instance of the class are the rows of the table.

**JPA -> Java Persistence API**

In java framework, if we want to work with database -> we need JDBC -> Java Database Connectivity .

JDBC code is tightly coupled to database. All DML,DQL and DDL -> will vary from database to database.

All the JDBC code which i wrote in java -> Needs to be rewritten

ORM -> Object Relational Mapping -> Java Class ----> RDBMS -> Mapping -> **Hibnerate**

- 1) If we change database no impact to my java code
- 2) Not having any database code in java side.

**Data JPA,Hibernate,ibatis,CrudRepository,JpaRepository ->**

**Spring is using EntityManager and spring boot created two advaced repo's -> CRUD and JPA**

**TicketBooking Application**

We will use In memory database which will store all the tickets information.  
 We will use CrudRepository for all database operations  
 We will use RestController as the front end engine for booking the tickets

**TicketController.java**      **TicketService.java**      **TicketDAO.java**

**Spring is using EntityManager and spring boot created two advaced repo's -> CRUD and JPA**

**TicketBooking Application**

We will use In memory database which will store all the tickets information.  
 We will use CrudRepository for all database operations  
 We will use RestController as the front end engine for booking the tickets

**Any request from Angular/REACT/NODE/SOAPUI/POSTMAN/BROWSER** → **TicketController.java** (Controller)

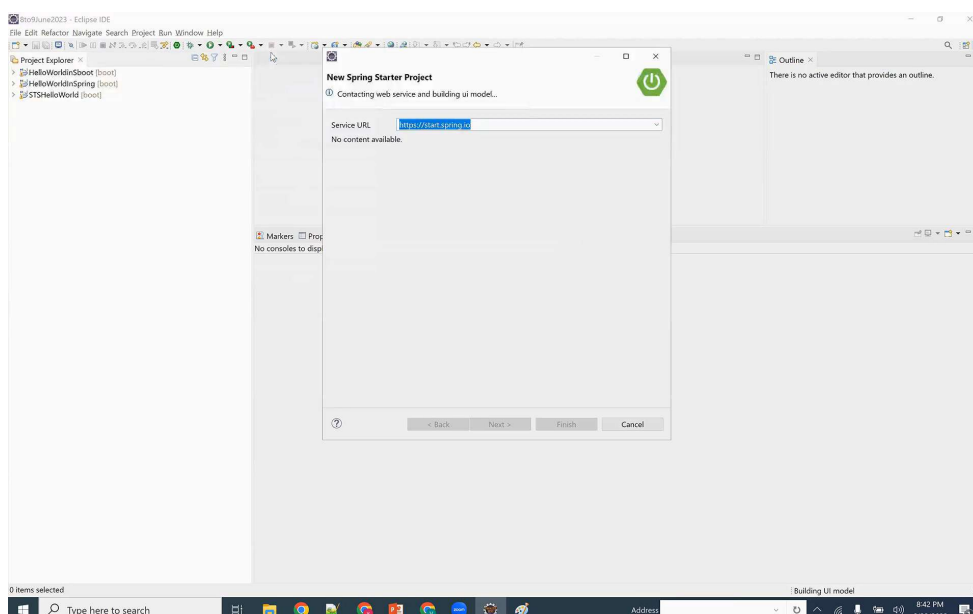
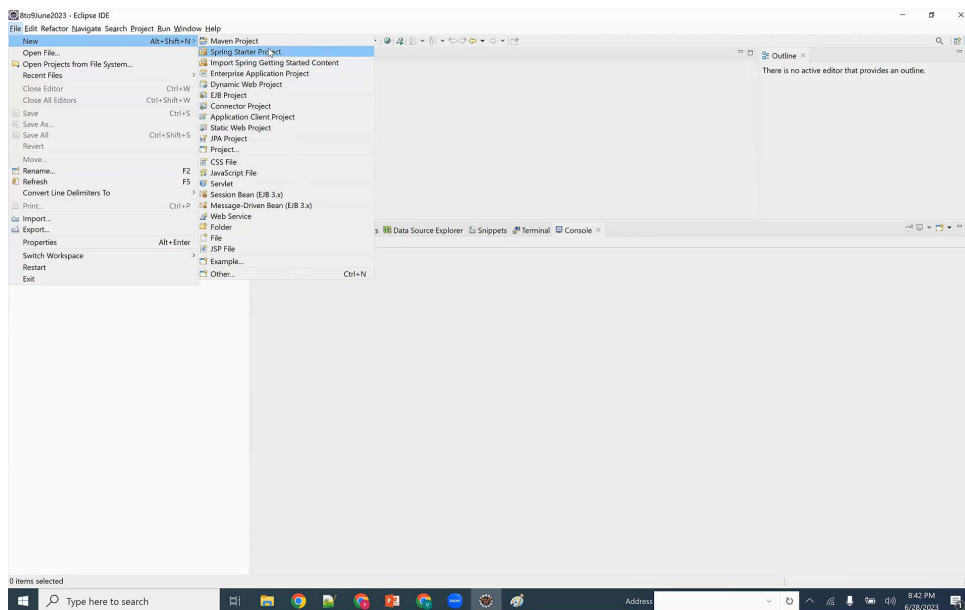
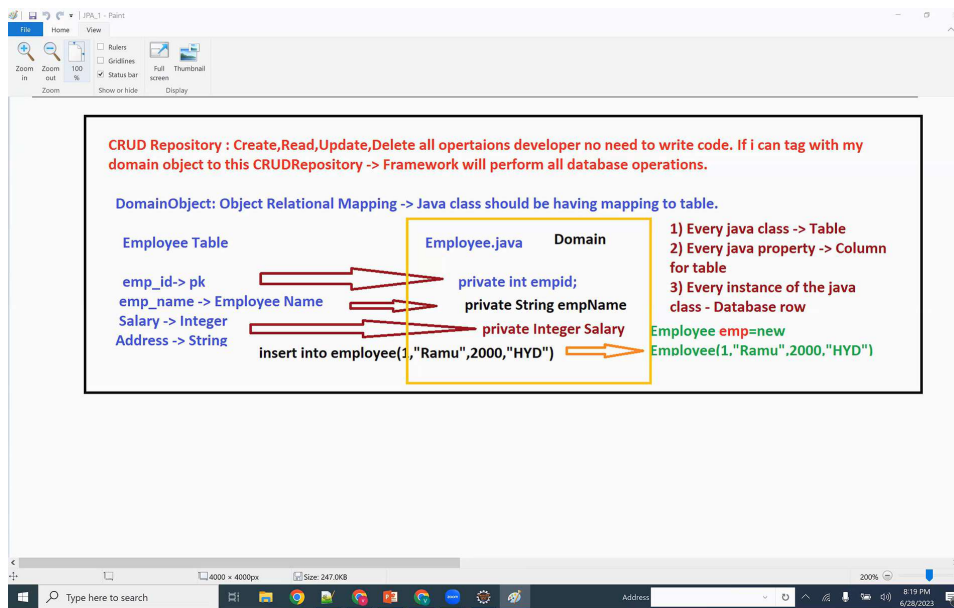
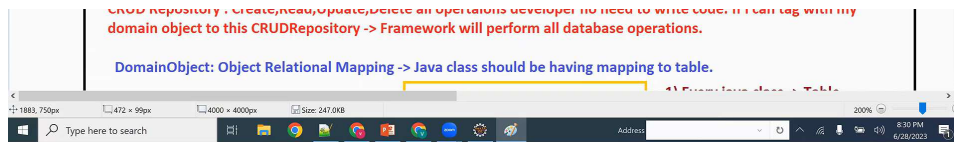
**TicketController.java** (Controller) → **TicketService.java** (Middle layer between controller and DAO)

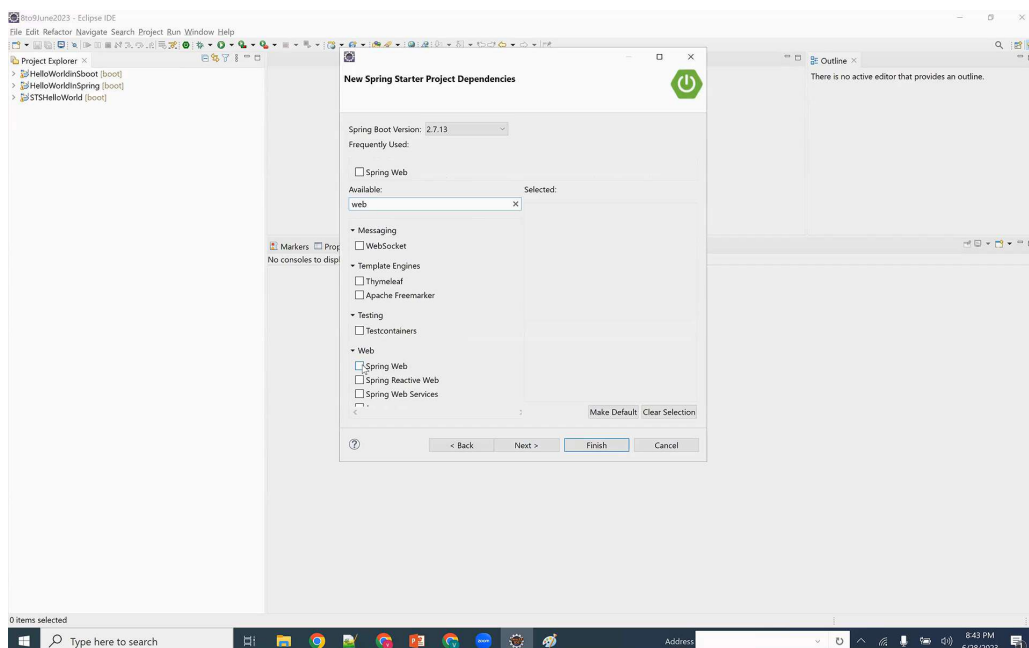
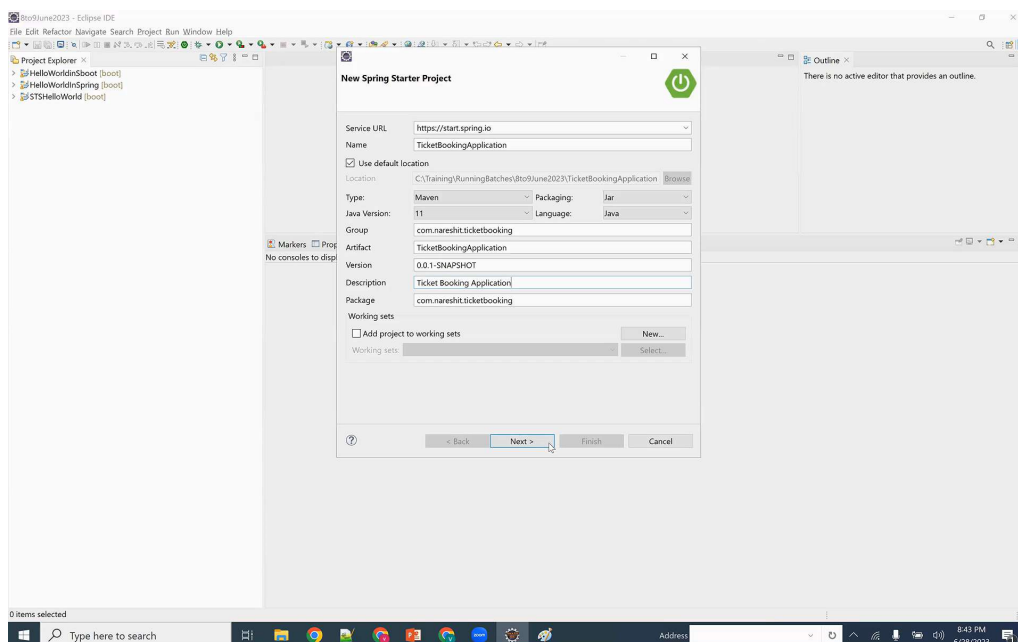
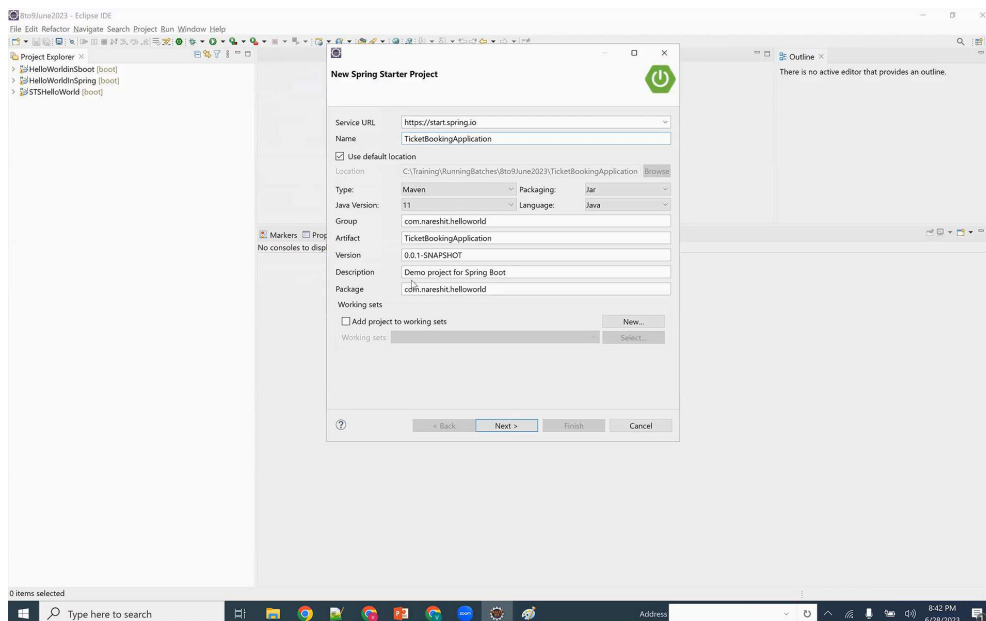
**TicketService.java** (Middle layer between controller and DAO) → **TicketDAO.java** (CRUD Repository, ORM)

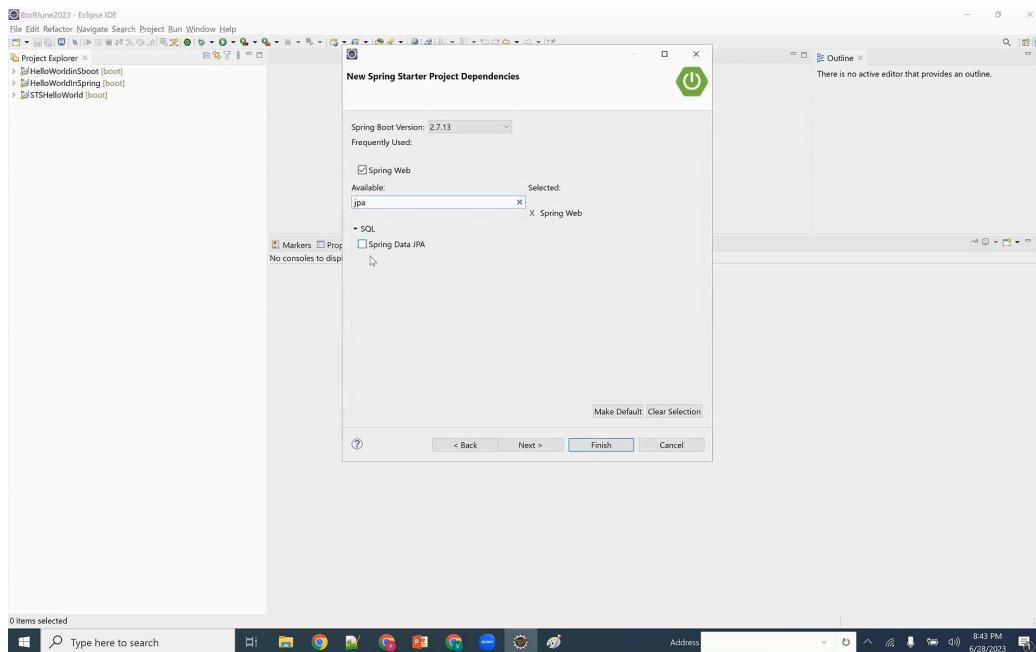
**TicketDAO.java** (CRUD Repository, ORM) → **DB**

**CRUD Repository :** Create,Read,Update,Delete all opertaions developer no need to write code. If i can tag with my domain object to this CRUDRepository -> Framework will perform all database operations.

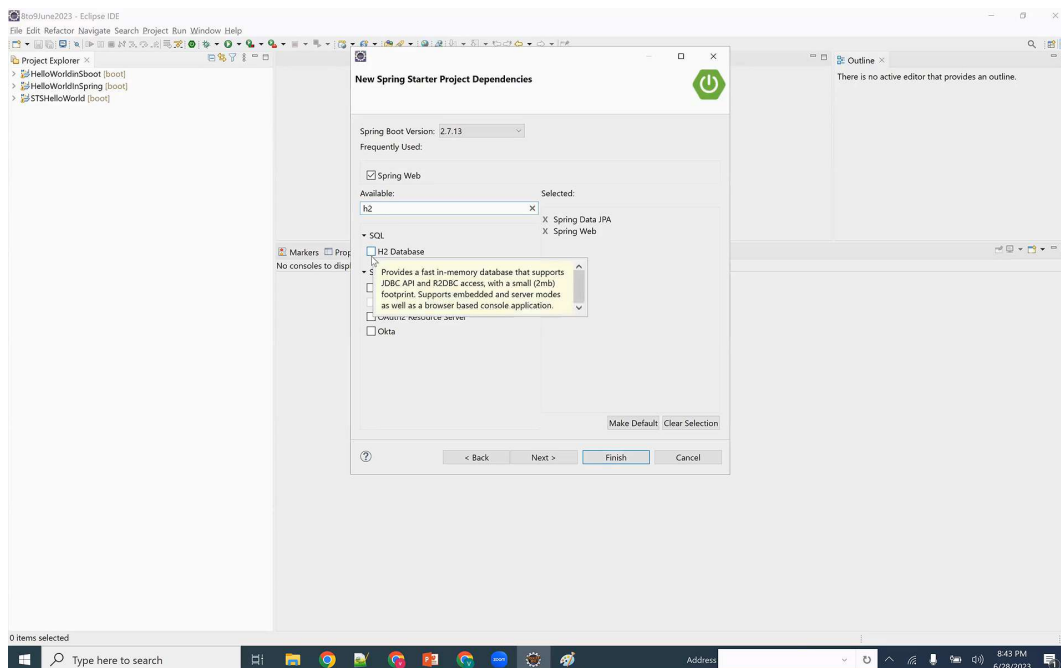
**DomainObject:** Object Relational Mapping -> Java class should be having mapping to table.

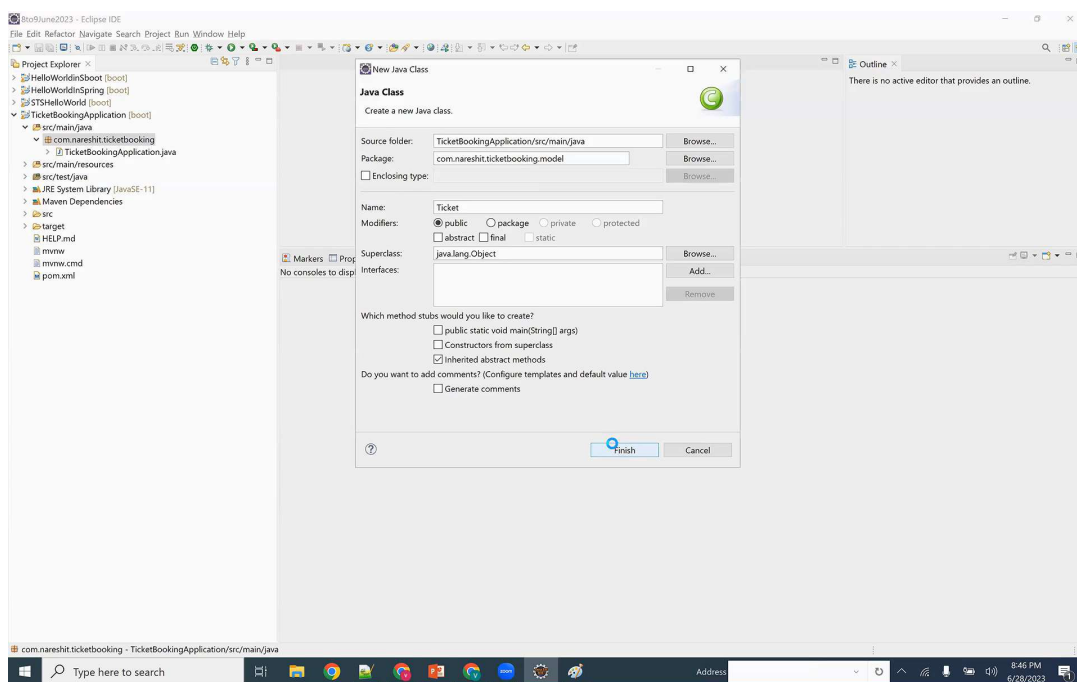
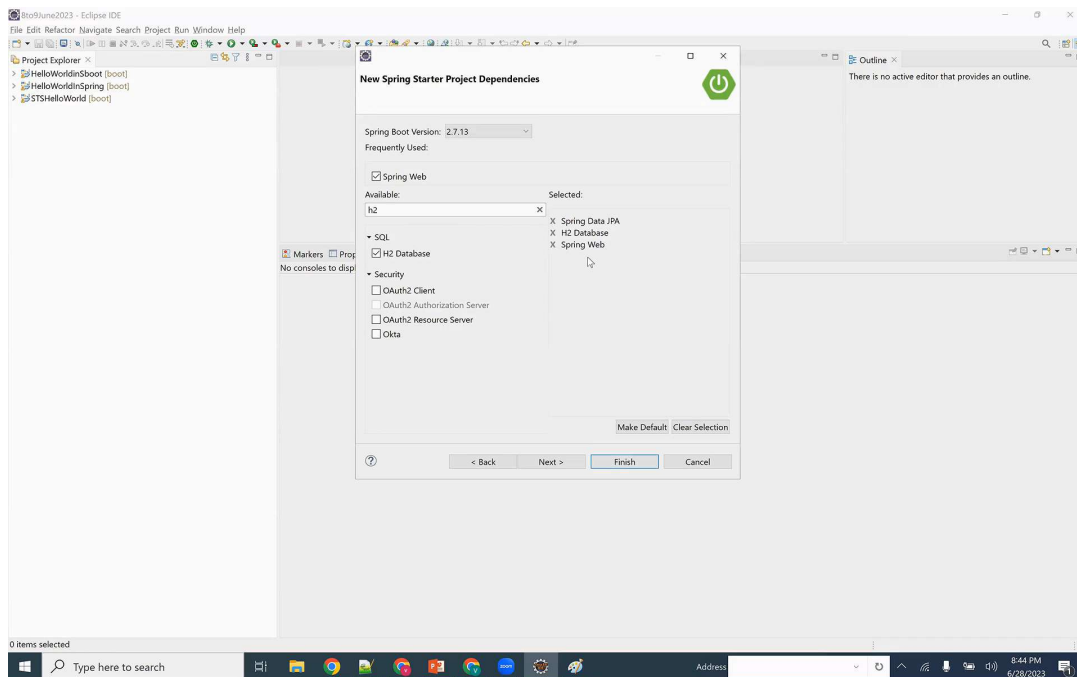






in memory database -- h2 Database.





we need to write Annotation "@Entity"

- "@Entity"
- This annotation is mandatory
- when we use the above annotation, then we making the class to consider it as a Entity Class /Model Class /Domain Class / Database Class
- when we write the above annotation, we are making java class to consider it is a table
- This will create a sql statement with "create table..."

@Table(name="table\_name")

- this annotation used to customize the table name
- this annotation is optional

Every Java Property will be a table column name.

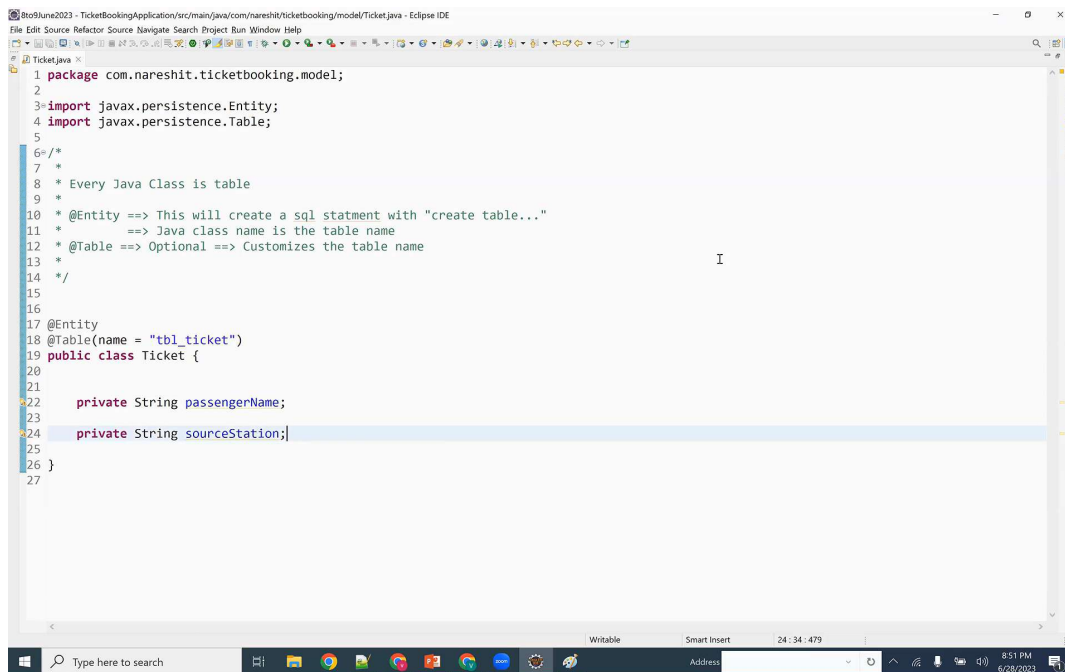
by default java property name will be the column name

@Column(name="column\_name")

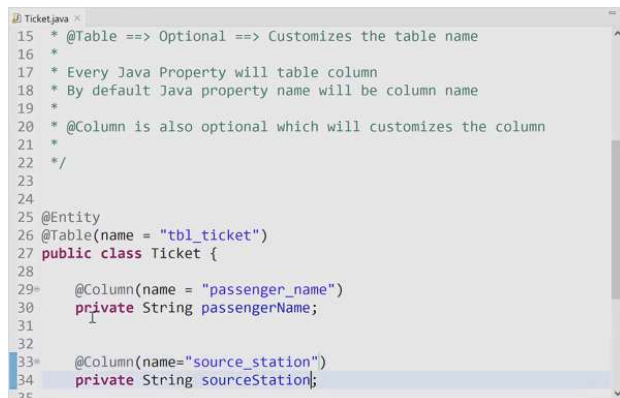
- this annotation is used to customize the column name
- this annotation is Optional

@Id

- this annotation is used to create primary key



```
1 package com.nareshit.ticketbooking.model;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Table;
5
6 /*
7  *
8  * Every Java Class is table
9  *
10  * @Entity ==> This will create a sql statment with "create table..."
11  * ==> Java class name is the table name
12  * @Table ==> Optional ==> Customizes the table name
13  *
14  */
15
16
17 @Entity
18 @Table(name = "tbl_ticket")
19 public class Ticket {
20
21
22     private String passengerName;
23
24     private String sourceStation;
25 }
26
27
```



```
15 * @Table ==> Optional ==> Customizes the table name
16 *
17 * Every Java Property will table column
18 * By default Java property name will be column name
19 *
20 * @Column is also optional which will customizes the column
21 *
22 */
23
24
25 @Entity
26 @Table(name = "tbl_ticket")
27 public class Ticket {
28
29     @Column(name = "passenger_name")
30     private String passengerName;
31
32
33     @Column(name="source_station")
34     private String sourceStation;
35
36 }
```

```
8to9June2023 - TicketBookingApplication/src/main/java/com/nareshit/ticketbooking/model/Ticket.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

Ticket.java
14 * @Entity ==> This will create a sql statment with "create table..."
15 * ==> Java class name is the table name
16 * @Table ==> Optional ==> Customizes the table name
17 *
18 * Every Java Property will table column
19 * By default Java property name will be column name
20 *
21 * @Column is also optional which will customizes the column I
22 *
23 * @Id ==> It will create primary Key
24 *
25 */
26
27
28 @Entity
29 @Table(name = "tbl_ticket")
30 public class Ticket {
31
32     @Id
33     @Column(name="ticket_id")
34     private Integer ticketId;
35
36     @Column(name = "passenger_name")
37     private String passengerName;
38
39
40     @Column(name="source_station")
41     private String sourceStation;
42
43     @Column(name="destination_station")
44     private String destinationStation;
45
46     @Column(name="travel_date")
47     private Date travelDate;
48
49     private String email;
```