***Spring boot JDBC****is a powerful framework that simplifies database interactions in Spring applications. It provides a high-level abstraction over JDBC, making database operations more concise and maintainable.*

**Essential Steps for Spring JDBC**

**1. Include Necessary Dependencies:**

To use Spring JDBC, you’ll need to include the following dependencies in your project’s build configuration (e.g., Maven or Gradle).

**2. Develop the DAO (Data Access Object) Class:**

Create a DAO class to encapsulate database interactions. Annotate it with @Repositoryto mark it as a data access component.

**3. Define JdbcTemplate as a Property:**

Inject the JdbcTemplate bean into your DAO class using the @Autowiredannotation. This provides access to JDBC methods for executing SQL queries.

**4. Execute Database Operations:**

Write methods in your DAO class to perform various database operations, such as querying, inserting, updating, and deleting data. Use the JdbcTemplate to execute SQL statements and map the results to Java objects.

**5. Set Up Configurations:**

Configure your application’s datasource properties in your application properties file.

**6. Access DAO Methods:**

From your service or controller classes, call the DAO methods to perform database operations.

**Spring Boot Specific Steps**

**1. Generate a Spring Boot Project:**

Use Spring Initializr to create a new Spring Boot project or use any Spring boot IDE, selecting the necessary dependencies for Spring Data JDBC, MySQL connector, and web.

**2. Configure Datasource:**

In your application properties file, specify the datasource parameters as mentioned above.

**3. Create DAO Interface and Class:**

Create a DAO interface with CRUD methods and a concrete DAO class implementing the interface, annotating it with @Repository.

**4. Service and Controller Layers:**

Create a Service layer to handle business logic and call DAO methods. Annotate the service class with @Service. Create a Controller layer to expose RESTful endpoints for interacting with the application. Annotate the controller class with @Controller.

**5. Runner Class :**

For testing or initial setup, you can create a CommandLIneRunner bean to call controller methods from the main application context.

By following these steps, you can effectively create Spring JDBC applications to interact with databases in your Spring projects.

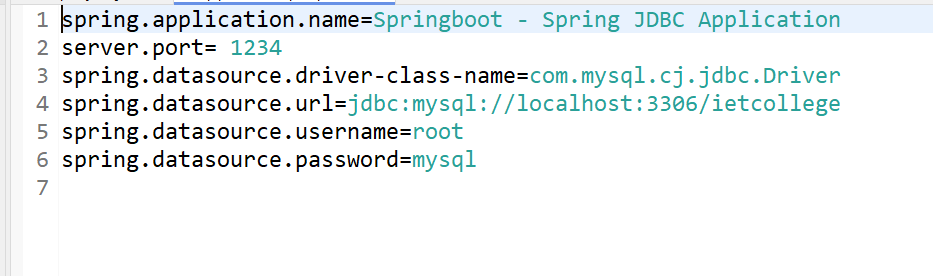
**Project [ SpringBootCRUD]**

When starting the project, add the following dependencies:

* MYSQL Driver
* Spring Boot Dev Tools
* Spring Web

**[application.properties]**

Zoom image will be displayed

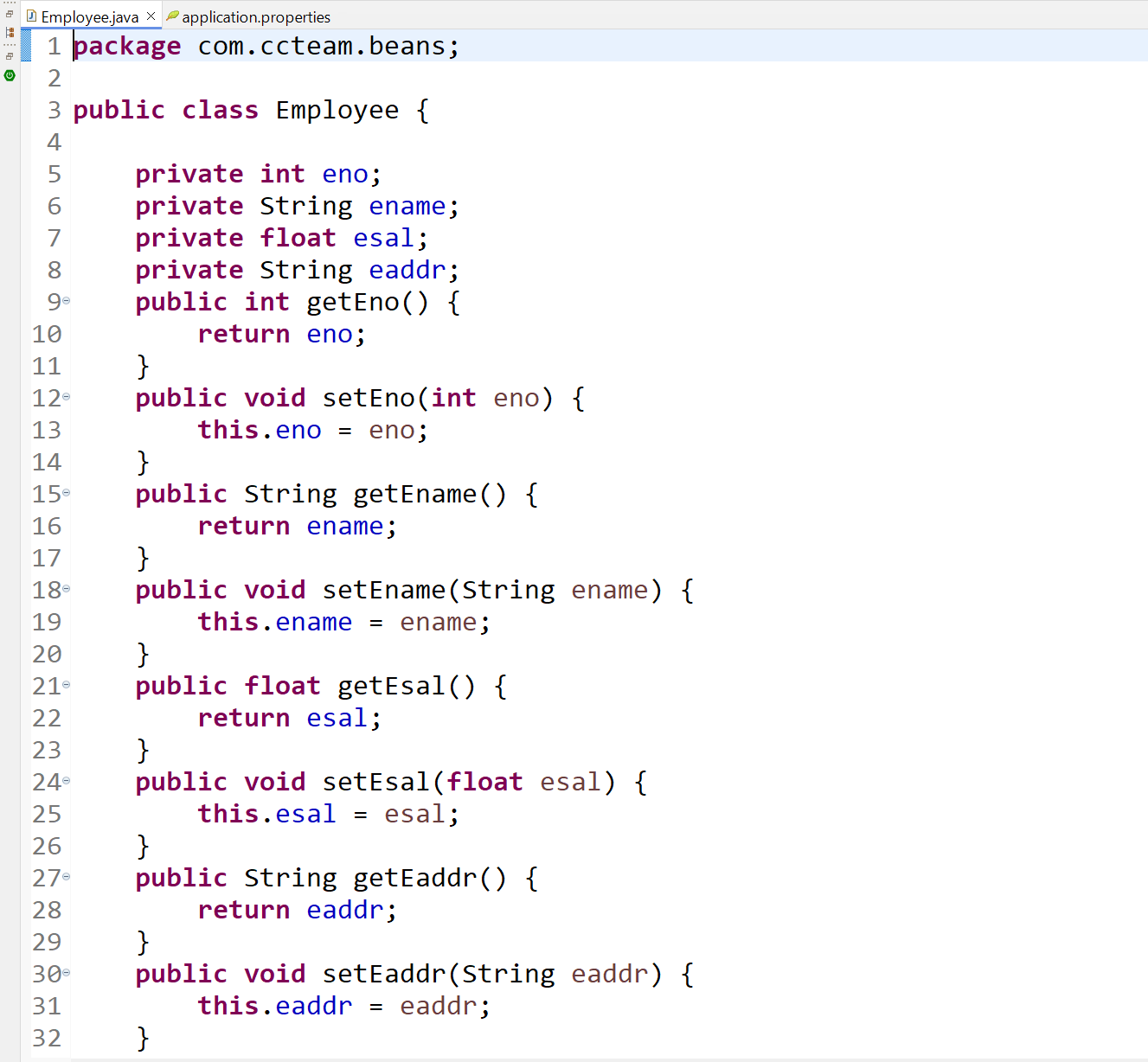


application.properties

Note : Create several packages [beans/dao/service/controller].

**[Employee.java]**

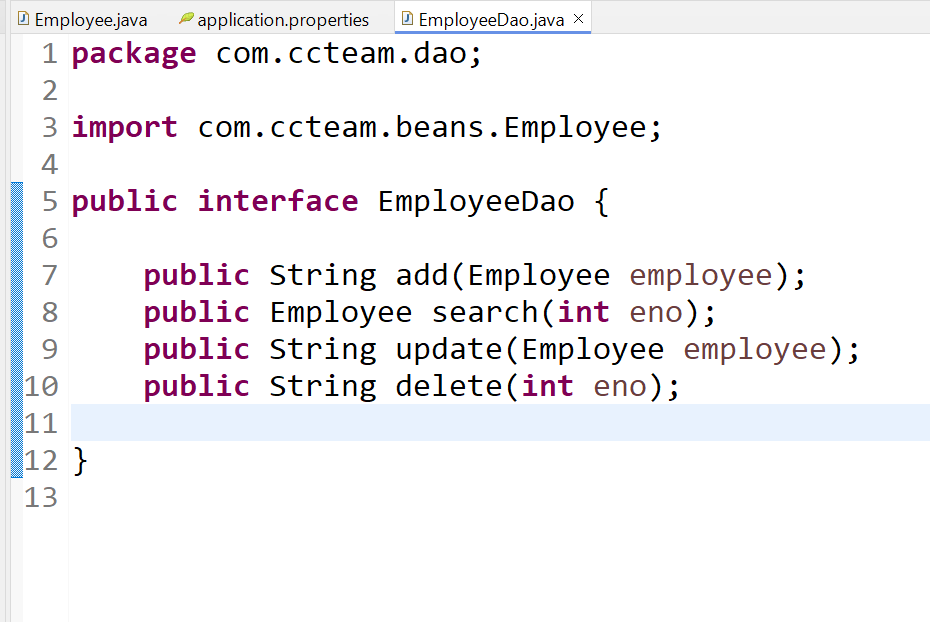
Zoom image will be displayed



Employee.java inside the beans package

**[EmployeeDao.java]**

Zoom image will be displayed



EmployeeDao.java Interface inside dao package

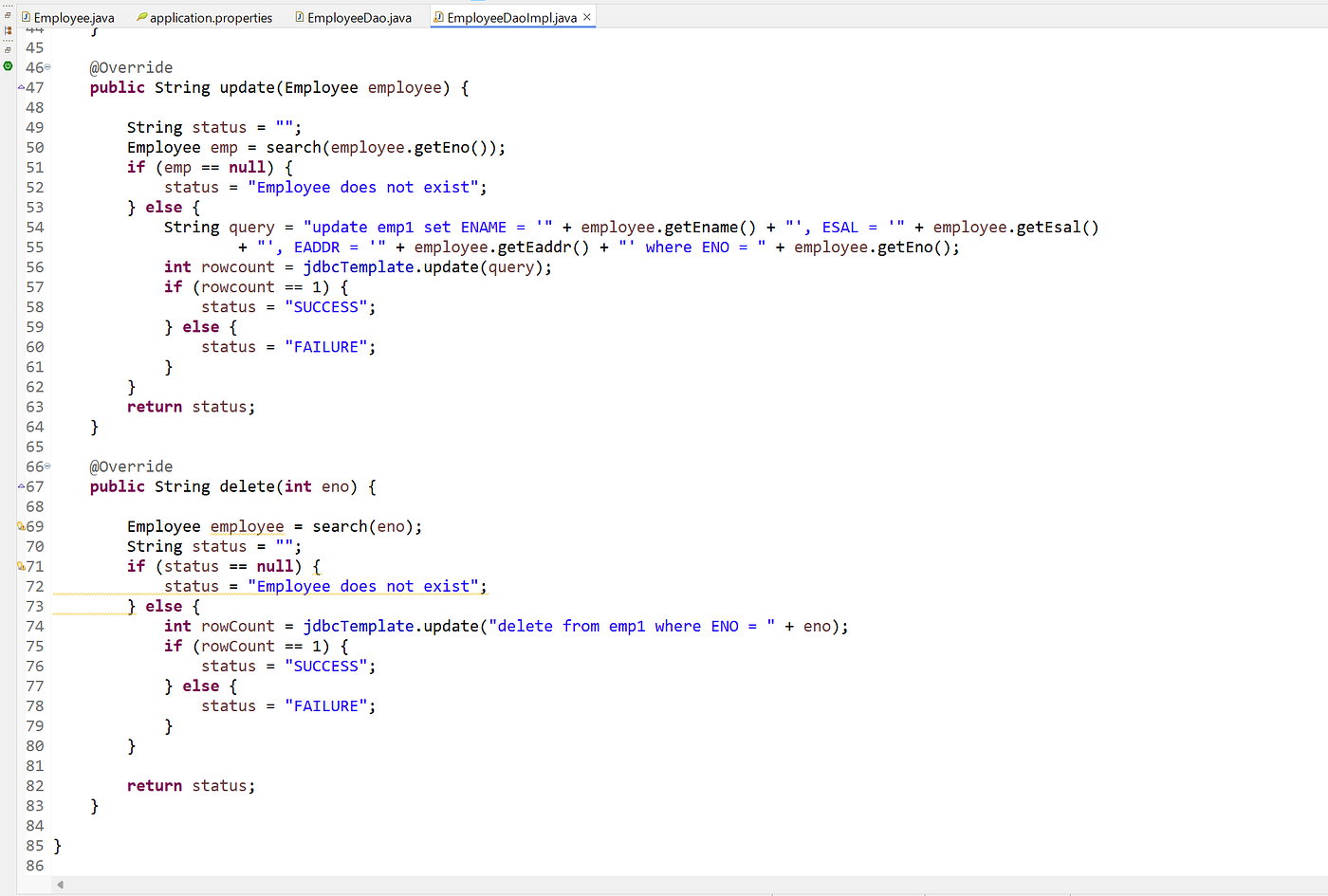
**[EmployeeDaoImpl.java]**

Zoom image will be displayed



EmployeeDaoImpl.java inside dao package — Part 1

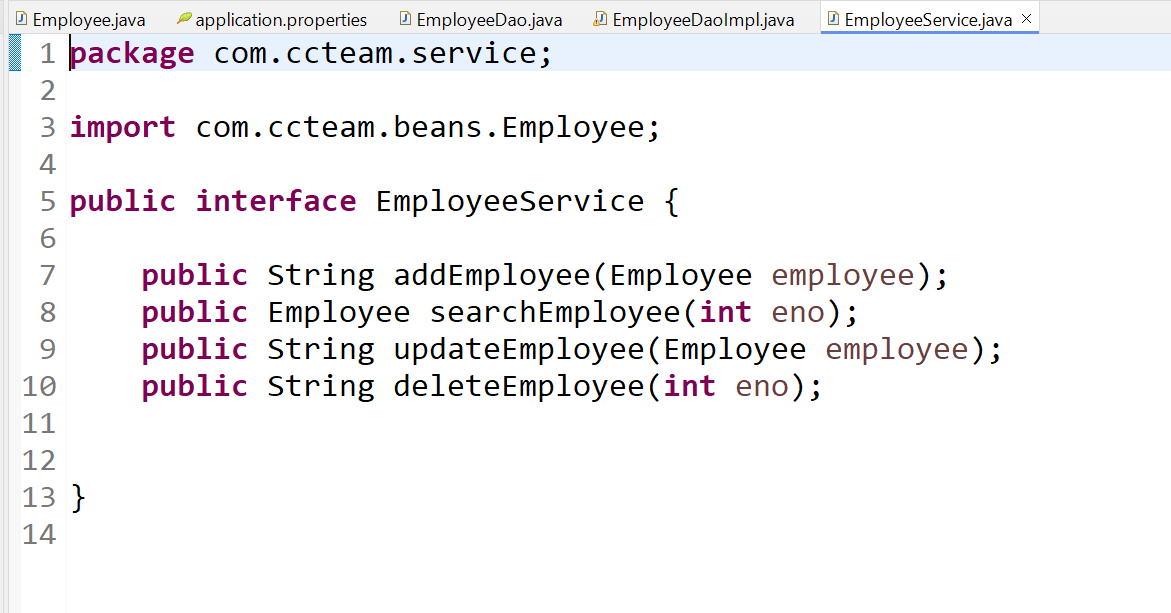
Zoom image will be displayed



EmployeeDaoImpl.java inside dao package — Part 2

**[EmployeeService.java]**

Zoom image will be displayed



EmployeeService.java Interface inside service package

**[EmployeeServiceImpl.java]**

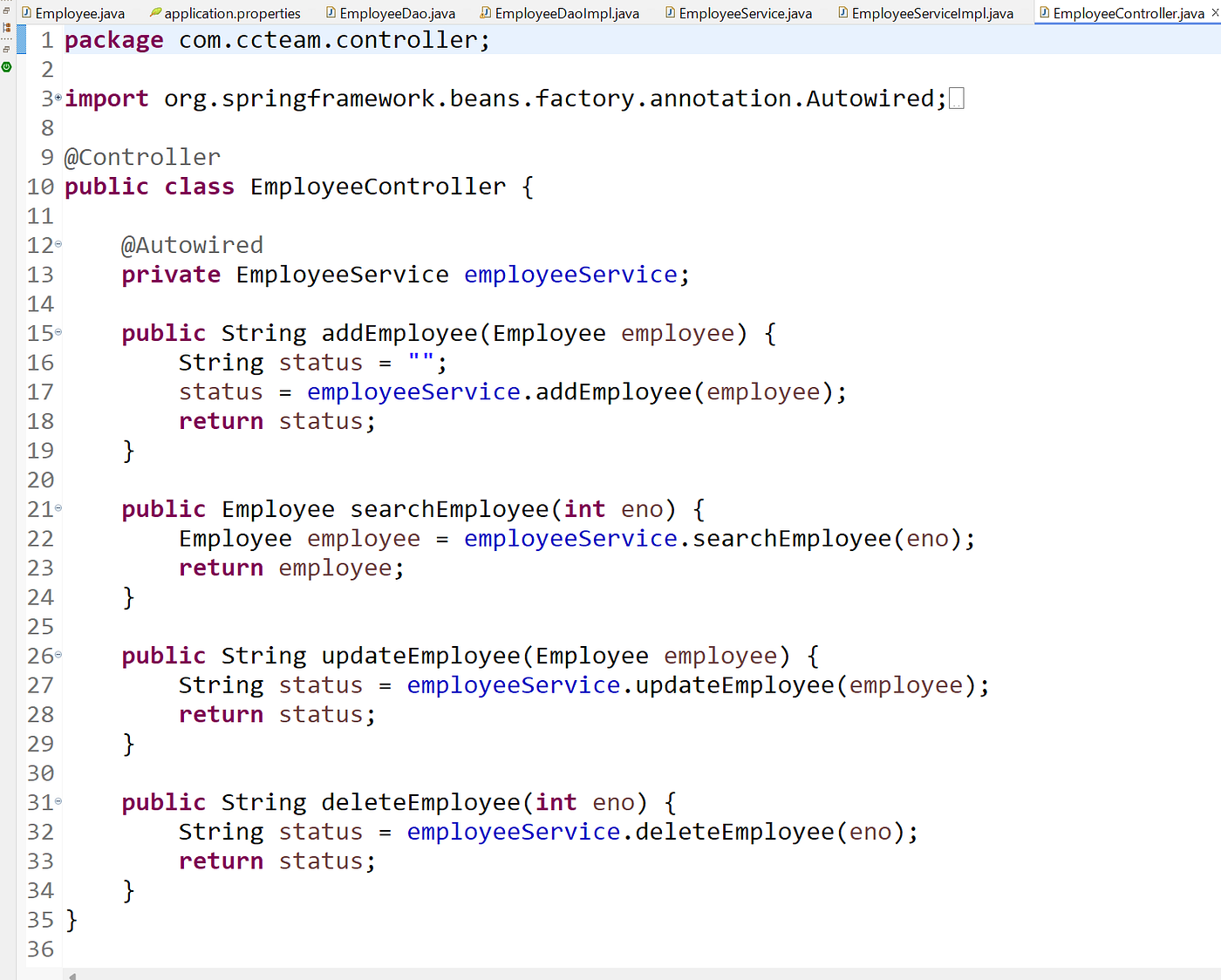
Zoom image will be displayed



EmployeeServiceImpl.java inside service package

**[EmployeeController.java]**

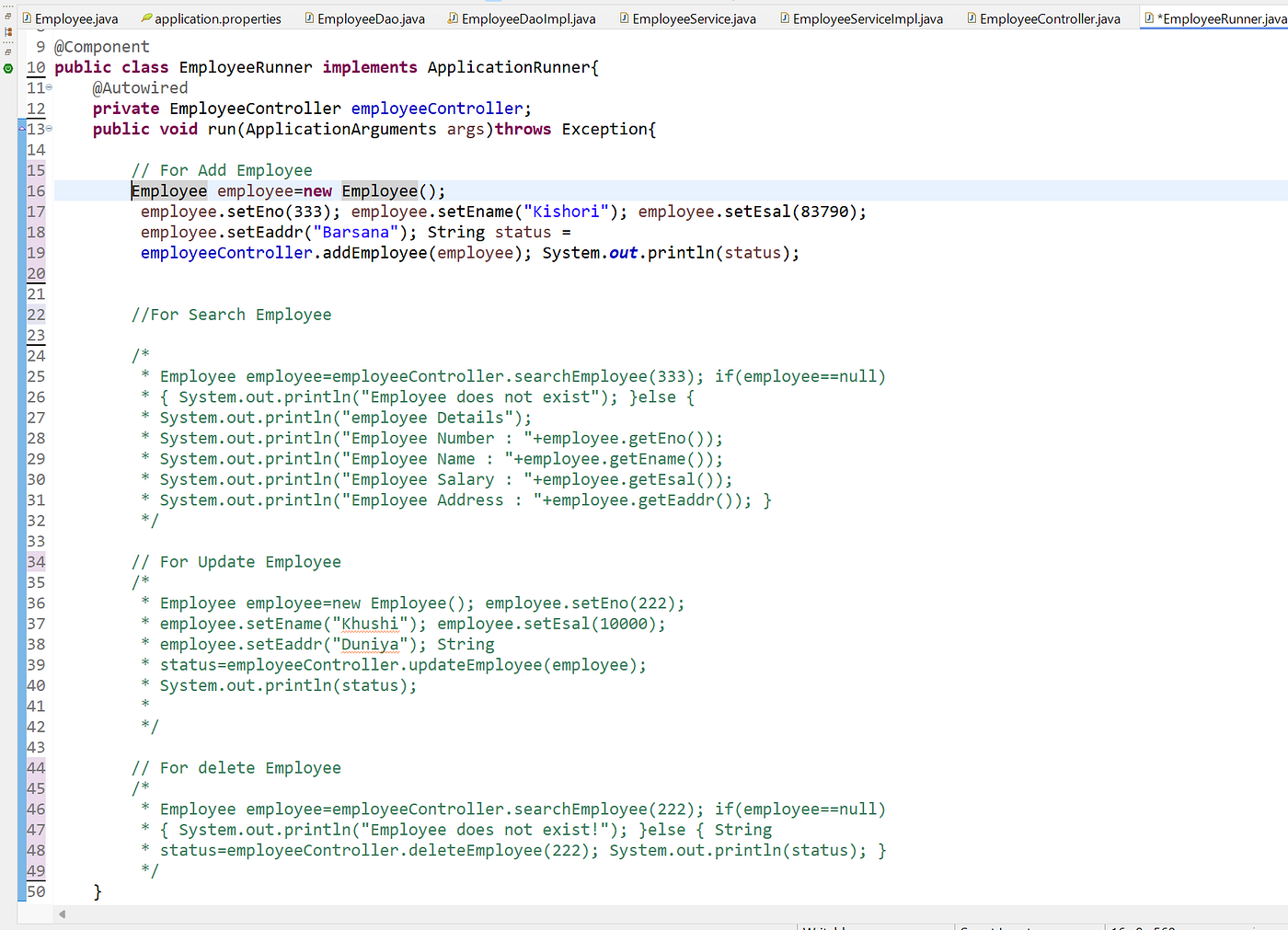
Zoom image will be displayed



EmployeeController.java inside controller package

**[EmployeeRunner.java]**

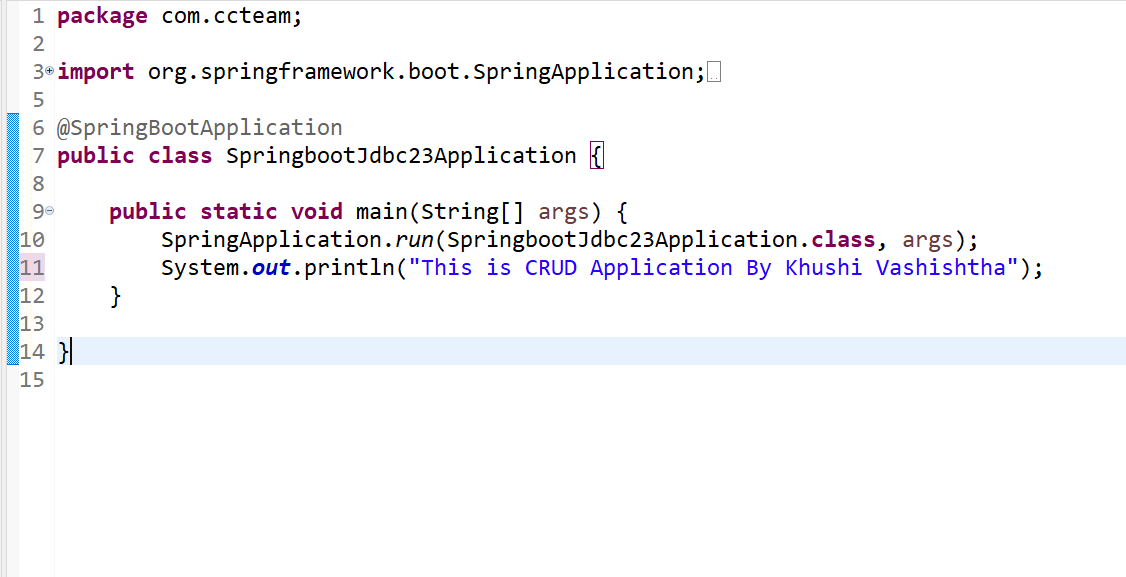
Zoom image will be displayed



EmployeeRunner.java inside runner package

**SpringBootCRUD.java**

Zoom image will be displayed



SpringBootCRUD.java with main method

**Must create Emp1 Table in mysql Database**

create table emp1   
(  
ENO int (4) PRIMARY KEY auto\_increment,  
ENAME VARCHAR(30),  
ESAL float(10),  
EADDR VARCHAR(40)  
);