1. Explain the steps to establish MySQL connectivity in a project.

MySQL Connectivity Steps

1. **Add Dependencies –** Include MySQL and Spring Boot Starter JPA in `pom.xml`.
2. **Configure Database –** Set up MySQL credentials in **`application.properties`**.
3. **Create Entity Class –** Define JPA entity with annotations.
4. **Create Repository –** Extend `JpaRepository` to handle database operations.
5. **Use Service Layer –** Implement business logic using the repository.
6. **Run Application –** Spring Boot auto-configures and connects to MySQL.

2. Explain the role of the application.properties file in MySQL connectivity.

Role of `application.properties` in MySQL Connectivity:

- Stores database configuration (URL, username, password, driver).
- Enables Hibernate auto-configuration.
- Defines properties like `spring.datasource.url`, `spring.datasource.username`, etc.

3. What is JPA Auditing?

JPA Auditing automatically tracks entity creation and modification timestamps. It helps in maintaining record logs.

4. Explain JPA Auditing using @CreatedDate and @LastModifiedDate.

`@CreatedDate` – Captures timestamp when an entity is first created.
`@LastModifiedDate` – Updates timestamp when an entity is modified.
Requires `@EnableJpaAuditing` in the main class and an entity with **`@EntityListeners(AuditingEntityListener.class)`**.

Q.2

Build a CRUD Rest API Project using the MySQL Database and JPA concept on Customer Entity created in the previous assignment.

1. **Create a CustomerController to handle CRUD operations for the Customer entity.**
2. **Create a CustomerService to manage business logic and interact with the CustomerRepository.**
3. **In the CustomerRepository interface, which should extend JpaRepository, implement the following query methods:**
    - **Find Customers by First Name**
    - **Find Customers by Last Name**
    - **Find Customers by Email**

- Find Customers by Salary Range
- Find Customers with Salary Greater Than a Specific Value
- Find Customers by Name Starting With
- Find Customers by Email Domain

4. Test the Query Methods using Postman

B]
With reference to the Customer entity class created in the previous assignments, apply @CreatedDate and @LastModifiedDate to your Customer entity class.

Hint:
Apply in the following way:

@CreateDate
private Instant CreatedAt;

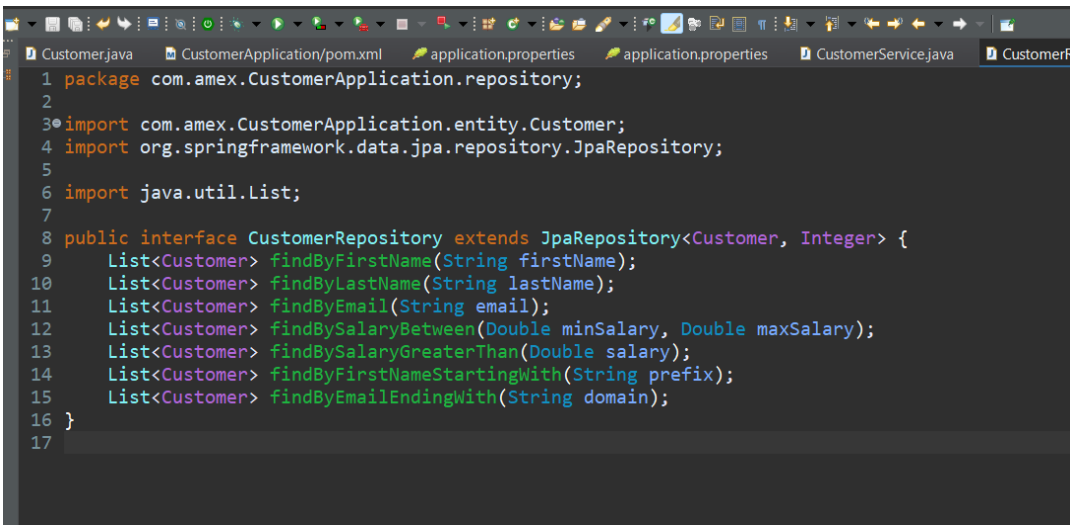
@LastModifiedDate
private Instant ModifiedAt;

Entity:

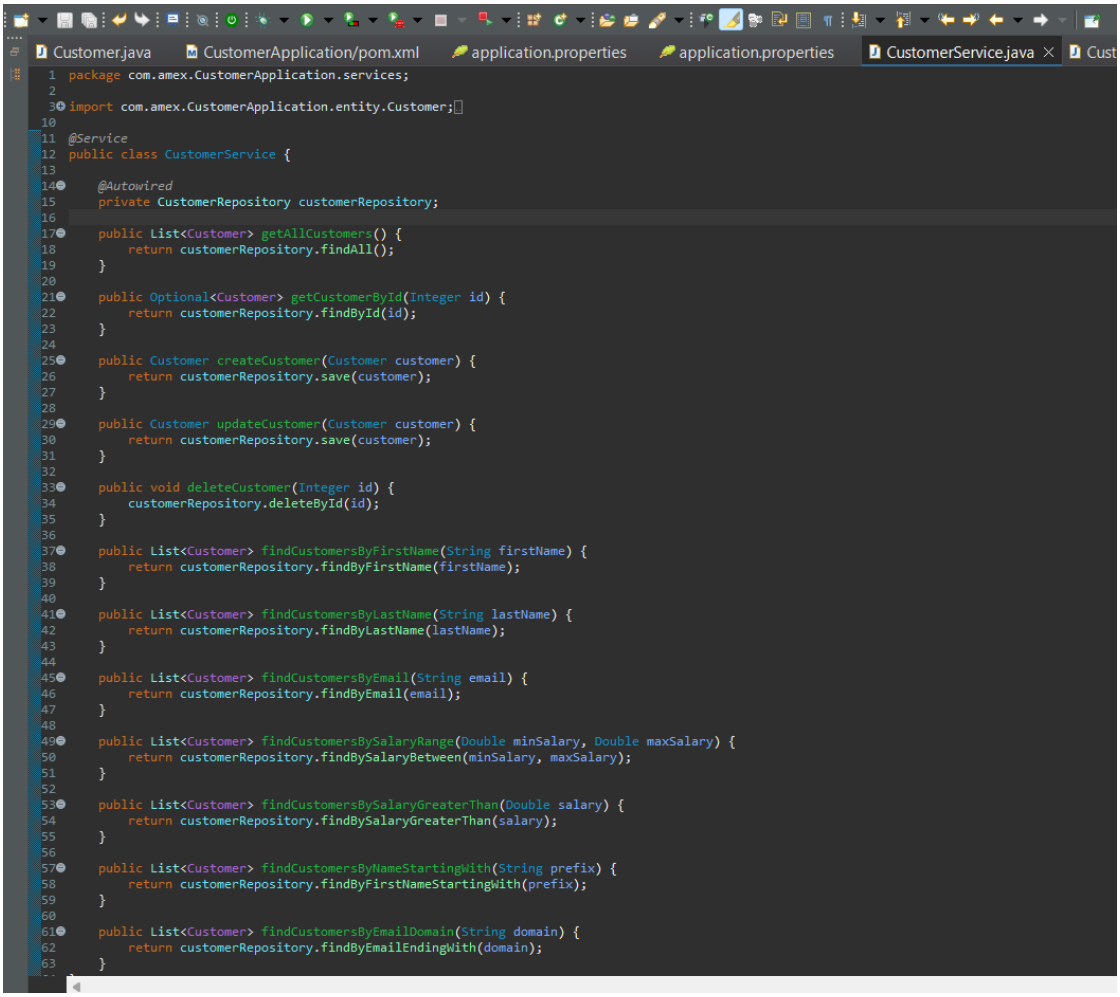
Customer.java | CustomerApplication/pom.xml | application.properties | application.properties | Cu

```java
package com.amex.CustomerApplication.entity;

import java.time.Instant;

import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedDate;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Integer customerId;
    private String firstName;
    private String lastName;
    private String email;
    private String address;

    @CreatedDate
    private Instant createdAt;

    @LastModifiedDate
    private Instant modifiedAt;
}
```

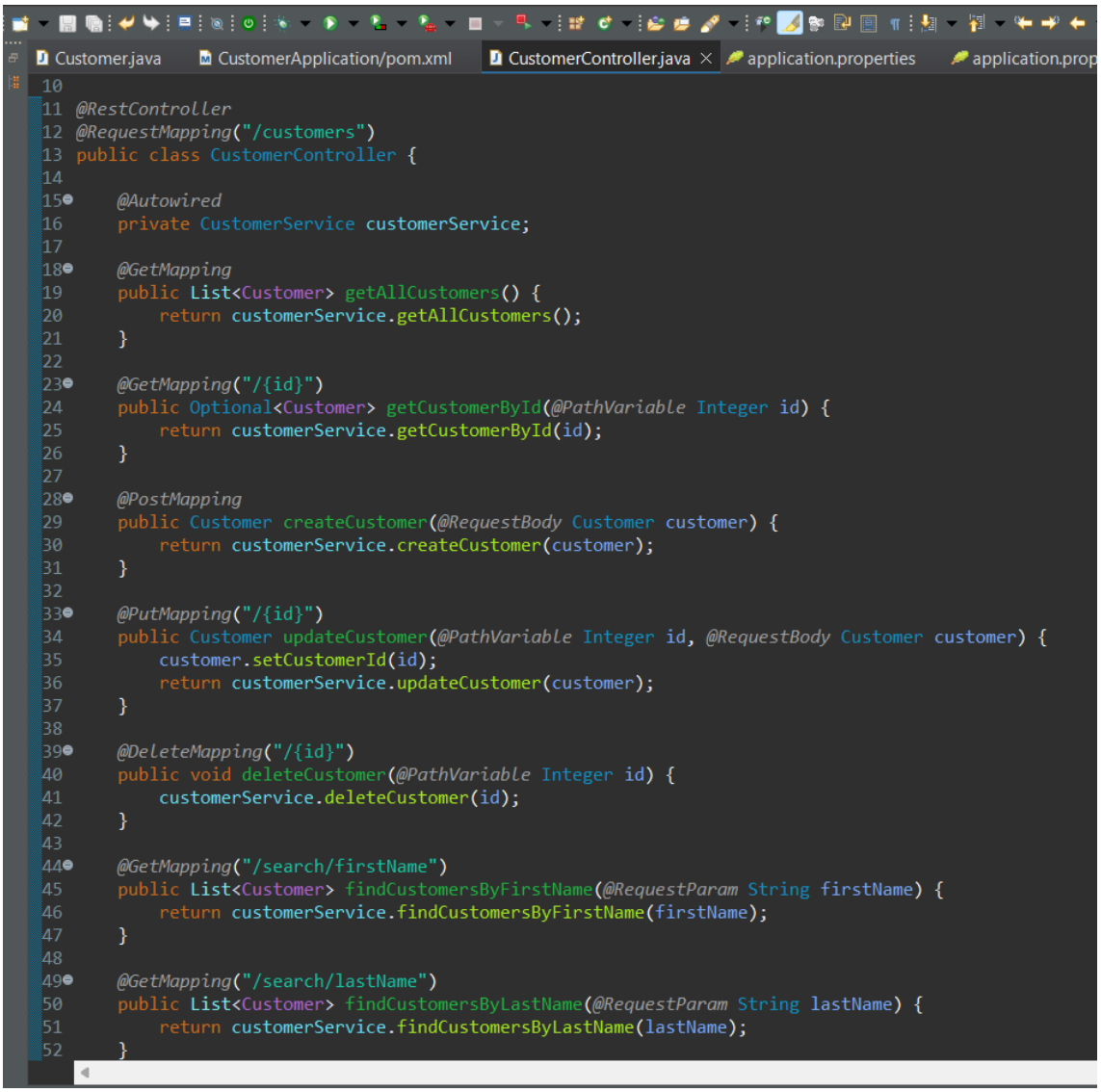
Repository:



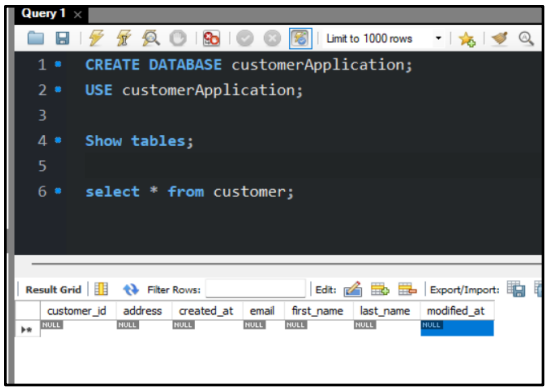
Service:

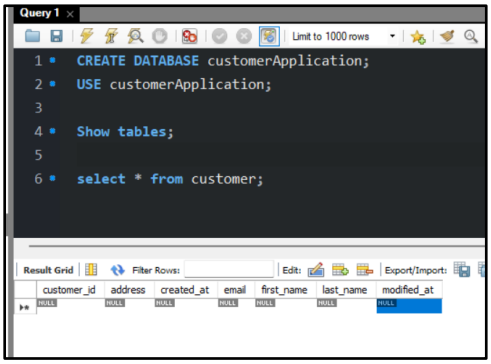


Controller:


Customer.java
CustomerApplication/pom.xml
CustomerController.java
application.properties
application.prop
10
11 @RestController
12 @RequestMapping("/customers")
13 public class CustomerController {
14
15 @Autowired
16 private CustomerService customerService;
17
18 @GetMapping
19 public List<Customer> getAllCustomers() {
20 return customerService.getAllCustomers();
21 }
22
23 @GetMapping("/{id}")
24 public Optional<Customer> getCustomerById(@PathVariable Integer id) {
25 return customerService.getCustomerById(id);
26 }
27
28 @PostMapping
29 public Customer createCustomer(@RequestBody Customer customer) {
30 return customerService.createCustomer(customer);
31 }
32
33 @PutMapping("/{id}")
34 public Customer updateCustomer(@PathVariable Integer id, @RequestBody Customer customer) {
35 customer.setCustomerId(id);
36 return customerService.updateCustomer(customer);
37 }
38
39 @DeleteMapping("/{id}")
40 public void deleteCustomer(@PathVariable Integer id) {
41 customerService.deleteCustomer(id);
42 }
43
44 @GetMapping("/search/firstName")
45 public List<Customer> findCustomersByFirstName(@RequestParam String firstName) {
46 return customerService.findCustomersByFirstName(firstName);
47 }
48
49 @GetMapping("/search/lastName")
50 public List<Customer> findCustomersByLastName(@RequestParam String lastName) {
51 return customerService.findCustomersByLastName(lastName);
52 }

Database:


Query 1
Limit to 1000 rows
1 CREATE DATABASE customerApplication;
2 USE customerApplication;
3
4 Show tables;
5
6 select * from customer;
Result Grid
Filter Rows:
Edit:
Export/Import:
customer_id
address
created_at
email
first_name
last_name
modified_at
NULL


Query 1
Limit to 1000 rows
1 CREATE DATABASE customerApplication;
2 USE customerApplication;
3
4 Show tables;
5
6 select * from customer;
Result Grid
Filter Rows:
Edit:
Export/Import:
customer_id
address
created_at
email
first_name
last_name
modified_at
NULL