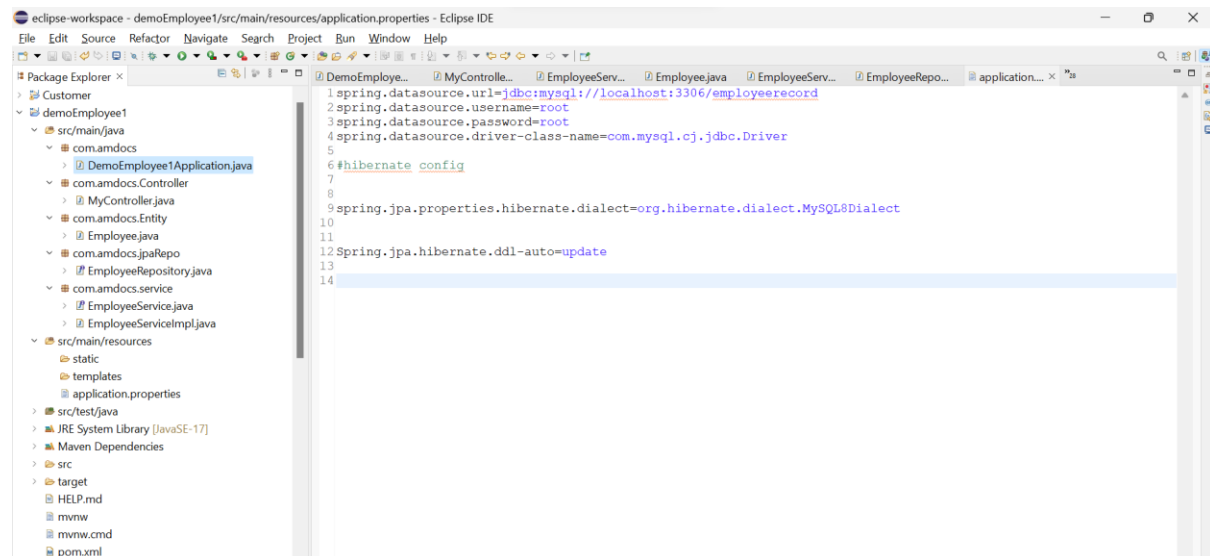


Project:EmployeeManagement

Code:

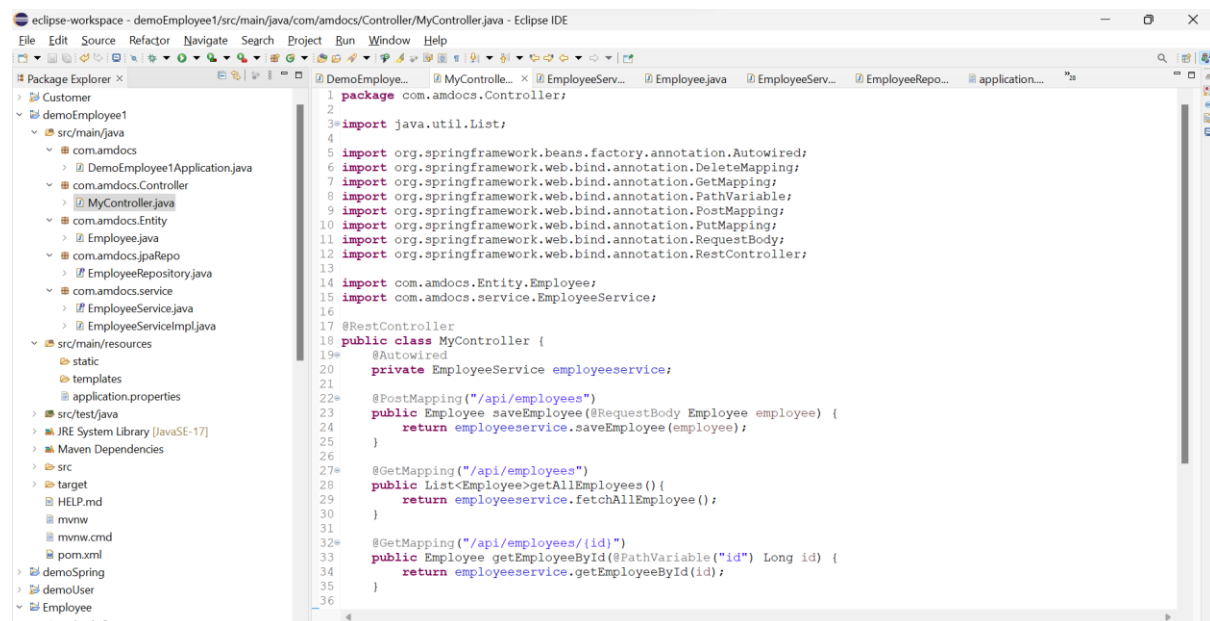
Main Class:



The screenshot shows the Eclipse IDE with the 'application.properties' file open in the editor. The Package Explorer on the left shows the project structure: 'demoEmployee1' with sub-packages 'com.amdocs' and 'src/main/resources'. The 'com.amdocs' package contains 'DemoEmployee1Application.java', 'Controller', 'Entity', 'jpaRepo', 'service', and 'EmployeeService'. The 'src/main/resources' package contains 'static', 'templates', and 'application.properties'. The 'application.properties' file contains the following configuration:

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/employeeeerecord
2 spring.datasource.username=root
3 spring.datasource.password=root
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6 #hibernate config
7
8
9 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
10
11
12 Spring.jpa.hibernate.ddl-auto=update
13
14
```

Controller:

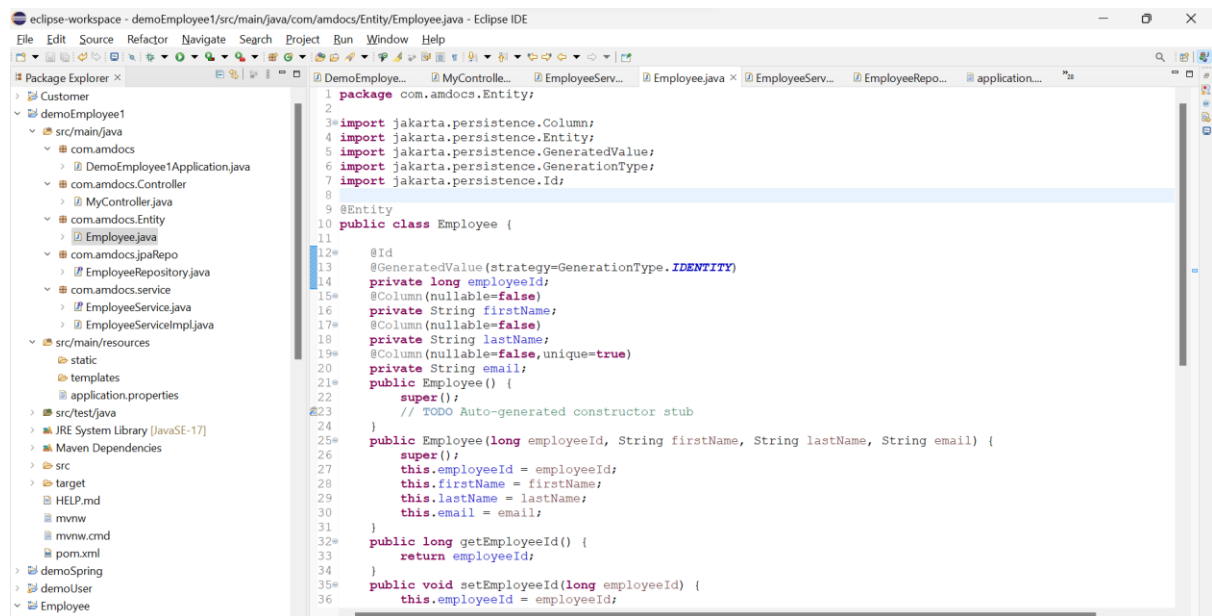


The screenshot shows the Eclipse IDE with the 'MyController.java' file open in the editor. The Package Explorer on the left shows the project structure: 'demoEmployee1' with sub-packages 'com.amdocs' and 'src/main/resources'. The 'com.amdocs' package contains 'DemoEmployee1Application.java', 'Controller', 'Entity', 'jpaRepo', 'service', and 'EmployeeService'. The 'src/main/resources' package contains 'static', 'templates', and 'application.properties'. The 'MyController.java' file contains the following code:

```
1 package com.amdocs.Controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.DeleteMapping;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PathVariable;
9 import org.springframework.web.bind.annotation.PostMapping;
10 import org.springframework.web.bind.annotation.PutMapping;
11 import org.springframework.web.bind.annotation.RequestBody;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import com.amdocs.Entity.Employee;
15 import com.amdocs.service.EmployeeService;
16
17 @RestController
18 public class MyController {
19     @Autowired
20     private EmployeeService employeeservice;
21
22     @PostMapping("/api/employees")
23     public Employee saveEmployee(@RequestBody Employee employee) {
24         return employeeservice.saveEmployee(employee);
25     }
26
27     @GetMapping("/api/employees")
28     public List<Employee> getAllEmployees() {
29         return employeeservice.fetchAllEmployee();
30     }
31
32     @GetMapping("/api/employees/{id}")
33     public Employee getEmployeeById(@PathVariable("id") Long id) {
34         return employeeservice.getEmployeeById(id);
35     }
36
```

```
31
32     @GetMapping("/api/employees/{id}")
33     public Employee getEmployeeById(@PathVariable("id") Long id) {
34         return employeeservice.getEmployeeById(id);
35     }
36
37     @PutMapping("/api/employees/{id}")
38     public Employee updateEmployee(@PathVariable("id") Long id, @RequestBody Employee employee) {
39         return employeeservice.updateEmployeeById(id, employee);
40     }
41
42     @DeleteMapping("/api/employees/{id}")
43     public String deleteEmployee(@PathVariable("id") Long id) {
44         return employeeservice.deleteEmployeeById(id);
45     }
46 }
47
48
49
```

Employee Entity:

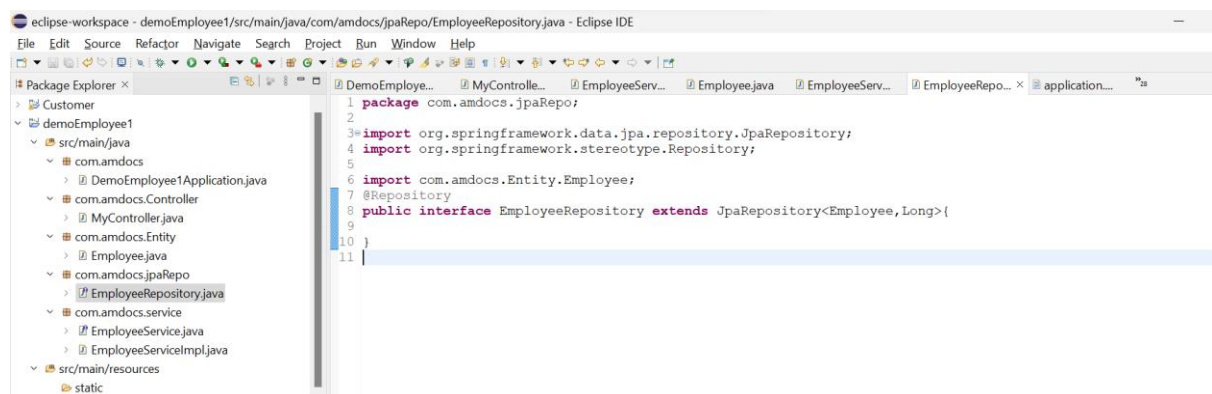


```
1 package com.amdocs.Entity;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8
9 @Entity
10 public class Employee {
11
12     @Id
13     @GeneratedValue(strategy=GenerationType.IDENTITY)
14     private long employeeId;
15     @Column(nullable=false)
16     private String firstName;
17     @Column(nullable=false)
18     private String lastName;
19     @Column(nullable=false,unique=true)
20     private String email;
21     public Employee() {
22         super();
23         // TODO Auto-generated constructor stub
24     }
25     public Employee(long employeeId, String firstName, String lastName, String email) {
26         super();
27         this.employeeId = employeeId;
28         this.firstName = firstName;
29         this.lastName = lastName;
30         this.email = email;
31     }
32     public long getEmployeeId() {
33         return employeeId;
34     }
35     public void setEmployeeId(long employeeId) {
36         this.employeeId = employeeId;
```



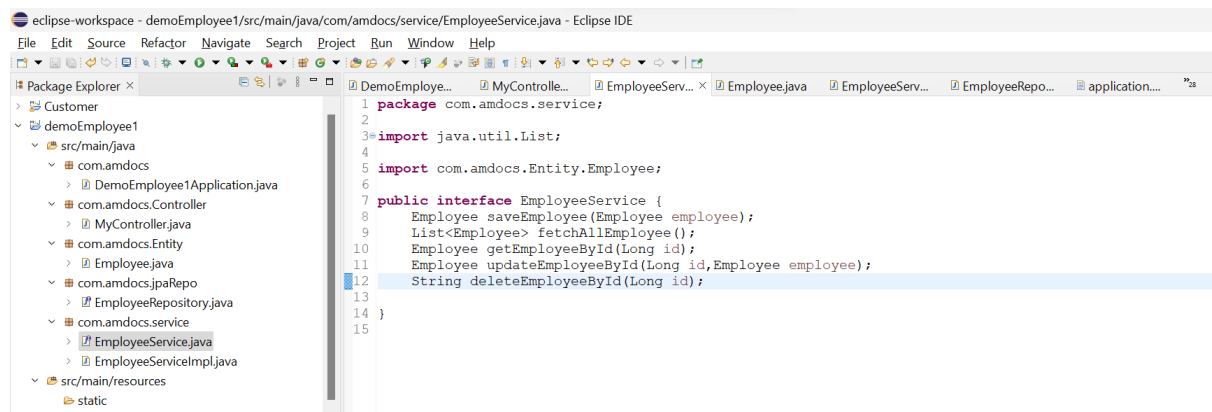
```
31 }
32 public long getEmployeeId() {
33     return employeeId;
34 }
35 public void setEmployeeId(long employeeId) {
36     this.employeeId = employeeId;
37 }
38 public String getFirstName() {
39     return firstName;
40 }
41 public void setFirstName(String firstName) {
42     this.firstName = firstName;
43 }
44 public String getLastName() {
45     return lastName;
46 }
47 public void setLastName(String lastName) {
48     this.lastName = lastName;
49 }
50 public String getEmail() {
51     return email;
52 }
53 public void setEmail(String email) {
54     this.email = email;
55 }
56 @Override
57 public String toString() {
58     return "Employee [employeeId=" + employeeId + ", firstName=" + firstName + ", lastName=" + lastName + ", email="
59         + email + " ]";
60 }
61 }
62 }
63
64
65
66
```

Employee Repository:



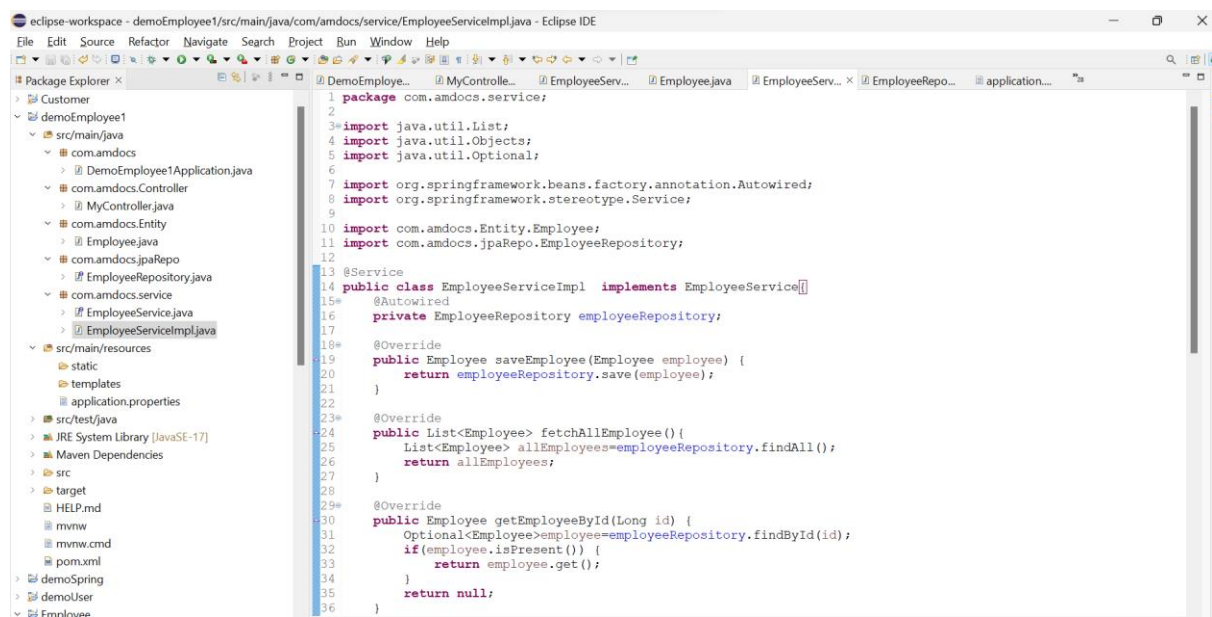
```
1 package com.amdocs.jpaRepo;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.amdocs.Entity.Employee;
7 @Repository
8 public interface EmployeeRepository extends JpaRepository<Employee, Long>{
9
10 }
11
```

Employee Service:

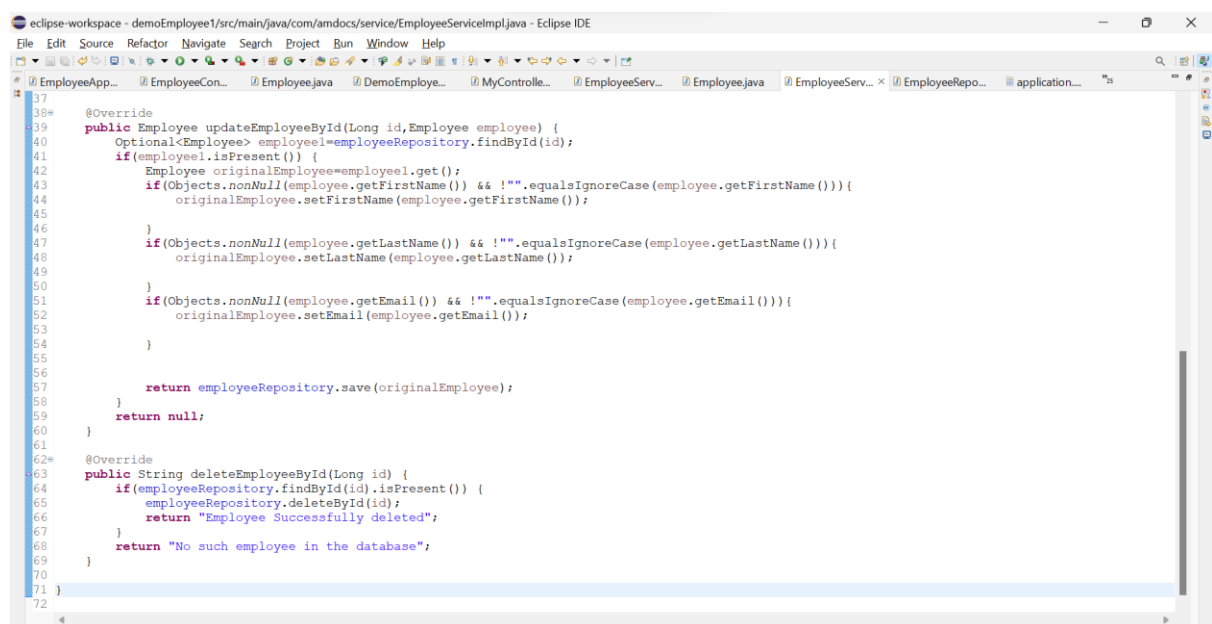


```
1 package com.amdocs.service;
2
3 import java.util.List;
4
5 import com.amdocs.Entity.Employee;
6
7 public interface EmployeeService {
8     Employee saveEmployee(Employee employee);
9     List<Employee> fetchAllEmployee();
10    Employee getEmployeeById(Long id);
11    Employee updateEmployeeById(Long id, Employee employee);
12    String deleteEmployeeById(Long id);
13
14 }
15
```

Employee service Implementaion Class:

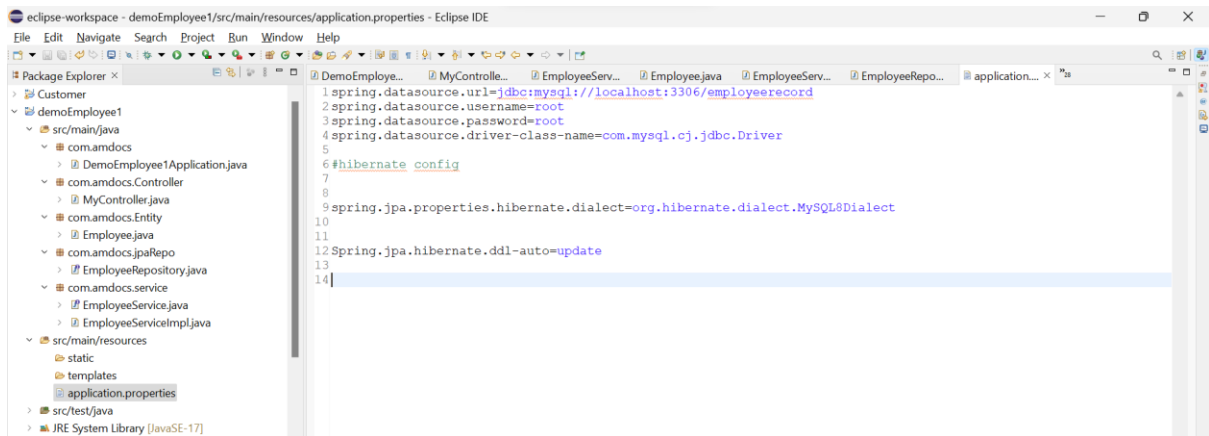


```
1 package com.amdocs.service;
2
3 import java.util.List;
4 import java.util.Objects;
5 import java.util.Optional;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9
10 import com.amdocs.Entity.Employee;
11 import com.amdocs.jpaRepo.EmployeeRepository;
12
13 @Service
14 public class EmployeeServiceImpl implements EmployeeService {
15     @Autowired
16     private EmployeeRepository employeeRepository;
17
18     @Override
19     public Employee saveEmployee(Employee employee) {
20         return employeeRepository.save(employee);
21     }
22
23     @Override
24     public List<Employee> fetchAllEmployee() {
25         List<Employee> allEmployees = employeeRepository.findAll();
26         return allEmployees;
27     }
28
29     @Override
30     public Employee getEmployeeById(Long id) {
31         Optional<Employee> employee = employeeRepository.findById(id);
32         if (employee.isPresent()) {
33             return employee.get();
34         }
35         return null;
36     }
37
38     @Override
39     public Employee updateEmployeeById(Long id, Employee employee) {
40         Optional<Employee> employee1 = employeeRepository.findById(id);
41         if (employee1.isPresent()) {
42             Employee originalEmployee = employee1.get();
43             if (Objects.nonNull(employee.getFirstName()) && !" ".equalsIgnoreCase(employee.getFirstName())) {
44                 originalEmployee.setFirstName(employee.getFirstName());
45             }
46             if (Objects.nonNull(employee.getLastName()) && !" ".equalsIgnoreCase(employee.getLastName())) {
47                 originalEmployee.setLastName(employee.getLastName());
48             }
49             if (Objects.nonNull(employee.getEmail()) && !" ".equalsIgnoreCase(employee.getEmail())) {
50                 originalEmployee.setEmail(employee.getEmail());
51             }
52             return employeeRepository.save(originalEmployee);
53         }
54         return null;
55     }
56
57     @Override
58     public String deleteEmployeeById(Long id) {
59         if (employeeRepository.findById(id).isPresent()) {
60             employeeRepository.delete(id);
61             return "Employee Successfully deleted";
62         }
63         return "No such employee in the database";
64     }
65 }
66
```



```
37
38 @Override
39 public Employee updateEmployeeById(Long id, Employee employee) {
40     Optional<Employee> employee1 = employeeRepository.findById(id);
41     if (employee1.isPresent()) {
42         Employee originalEmployee = employee1.get();
43         if (Objects.nonNull(employee.getFirstName()) && !" ".equalsIgnoreCase(employee.getFirstName())) {
44             originalEmployee.setFirstName(employee.getFirstName());
45         }
46         if (Objects.nonNull(employee.getLastName()) && !" ".equalsIgnoreCase(employee.getLastName())) {
47             originalEmployee.setLastName(employee.getLastName());
48         }
49         if (Objects.nonNull(employee.getEmail()) && !" ".equalsIgnoreCase(employee.getEmail())) {
50             originalEmployee.setEmail(employee.getEmail());
51         }
52         return employeeRepository.save(originalEmployee);
53     }
54     return null;
55 }
56
57 @Override
58 public String deleteEmployeeById(Long id) {
59     if (employeeRepository.findById(id).isPresent()) {
60         employeeRepository.delete(id);
61         return "Employee Successfully deleted";
62     }
63     return "No such employee in the database";
64 }
65 }
66
```

Application.properties

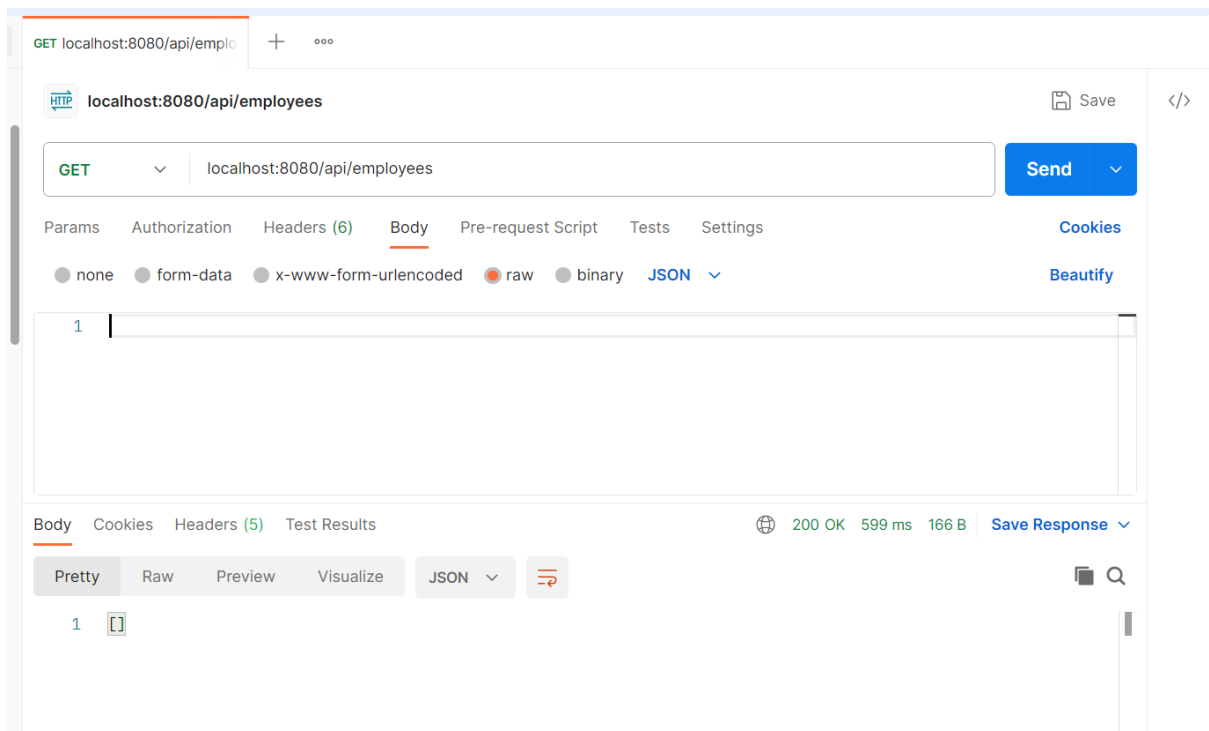


The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, with 'application.properties' selected under 'src/main/resources'. The main editor window shows the following configuration:

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/employeerecord
2 spring.datasource.username=root
3 spring.datasource.password=root
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6 #hibernate config
7
8
9 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
10
11
12 Spring.jpa.hibernate.ddl-auto=update
13
14
```

Postman:

a)get the details :initially it will show empty .



b)Post data :

The screenshot shows the Postman interface for a POST request to `localhost:8080/api/employees`. The request is configured with the following body (JSON):

```
1 {
2   "firstName": "Shalini",
3   "lastName": "Singh",
4   "email": "singhshalini@gmail.com"
5 }
6
```

The response is displayed in the bottom pane, showing a 200 OK status with a response time of 191 ms and a body size of 254 B. The response body (JSON) is:

```
1 {
2   "employeeId": 1,
3   "firstName": "Shalini",
4   "lastName": "Singh",
5   "email": "singhshalini@gmail.com"
6 }
```

c)Fetch all Employees:

The screenshot shows the Postman interface for a GET request to `localhost:8080/api/employees`. The response is displayed in the bottom pane, showing a 200 OK status with a response time of 28 ms and a body size of 441 B. The response body (JSON) is:

```
1 [
2   {
3     "employeeId": 1,
4     "firstName": "Shalini",
5     "lastName": "Singh",
6     "email": "singhshalini@gmail.com"
7   },
8   {
9     "employeeId": 2,
10    "firstName": "Shikha",
11    "lastName": "Prajapati",
12    "email": "prajapatishikha30@gmail.com"
13  }
14 ]
```

d)fetch employee by Id:

The screenshot shows a REST client interface with a GET request to `localhost:8080/api/employees/3`. The response is a JSON object with the following details:

```
1 {
2   "employeeId": 3,
3   "firstName": "Malay",
4   "lastName": "Bhowmick",
5   "email": "malayB@gmail.com"
6 }
```

The status bar indicates a 200 OK response with a 44 ms latency and 249 B of data.

e)Update Employee Details:

The screenshot shows a REST client interface with a PUT request to `localhost:8080/api/employees/2`. The request body contains the following JSON:

```
1 {
2   "firstName": "Shikha",
3   "lastName": "Mishra",
4   "email": "mishrashikha@gmail.com"
5 }
```

The response is a JSON object with the updated details:

```
1 {
2   "employeeId": 2,
3   "firstName": "Shikha",
4   "lastName": "Mishra",
5   "email": "mishrashikha@gmail.com"
6 }
```

The status bar indicates a 200 OK response with a 32 ms latency and 254 B of data.

f) Delete Employee Details:

The screenshot shows a Postman interface for a DELETE request to `localhost:8080/api/employees/2`. The request is configured with the following details:

- Method:** DELETE
- URL:** localhost:8080/api/employees/2
- Body:** A JSON object with the following structure:

```
{  "firstName": "Shikha",  "lastName": "Mishra",  "email": "mishrashikha@gmail.com"}
```
- Headers:** 8 headers are listed.
- Response:** The response is a 200 OK status with a response time of 46 ms and a body size of 193 B. The response body is "Employee Successfully deleted".

g) If wrong id is selected for delete:

The screenshot shows a Postman interface for a DELETE request to `localhost:8080/api/employees/5`. The request is configured with the following details:

- Method:** DELETE
- URL:** localhost:8080/api/employees/5
- Body:** A JSON object with the following structure:

```
{  "firstName": "Shikha",  "lastName": "Mishra",  "email": "mishrashikha@gmail.com"}
```
- Headers:** 8 headers are listed.
- Response:** The response is a 404 Not Found status with a response time of 15 ms and a body size of 196 B. The response body is "No such employee in the database".

DataBase Image:

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- course
- employeeedb
- employeerecord
 - Tables
 - employee
 - Columns
 - employee_id
 - email
 - first_name
 - last_name
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - student

Administration Schemas

Query 1 x employee employeerecord - Schema

Limit to 1000 rows

```
1 select * from employeerecord.employee;
```

Result Grid

	employee_id	email	first_name	last_name
▶	1	singhshalini@gmail.com	Shalini	Singh
	2	mishrashikha@gmail.com	Shikha	Mishra
	3	malayB@gmail.com	Malay	Bhowmick
*	NULL	NULL	NULL	NULL

Result Grid