BITA/5/21/022/TZ

a) To create springboot project

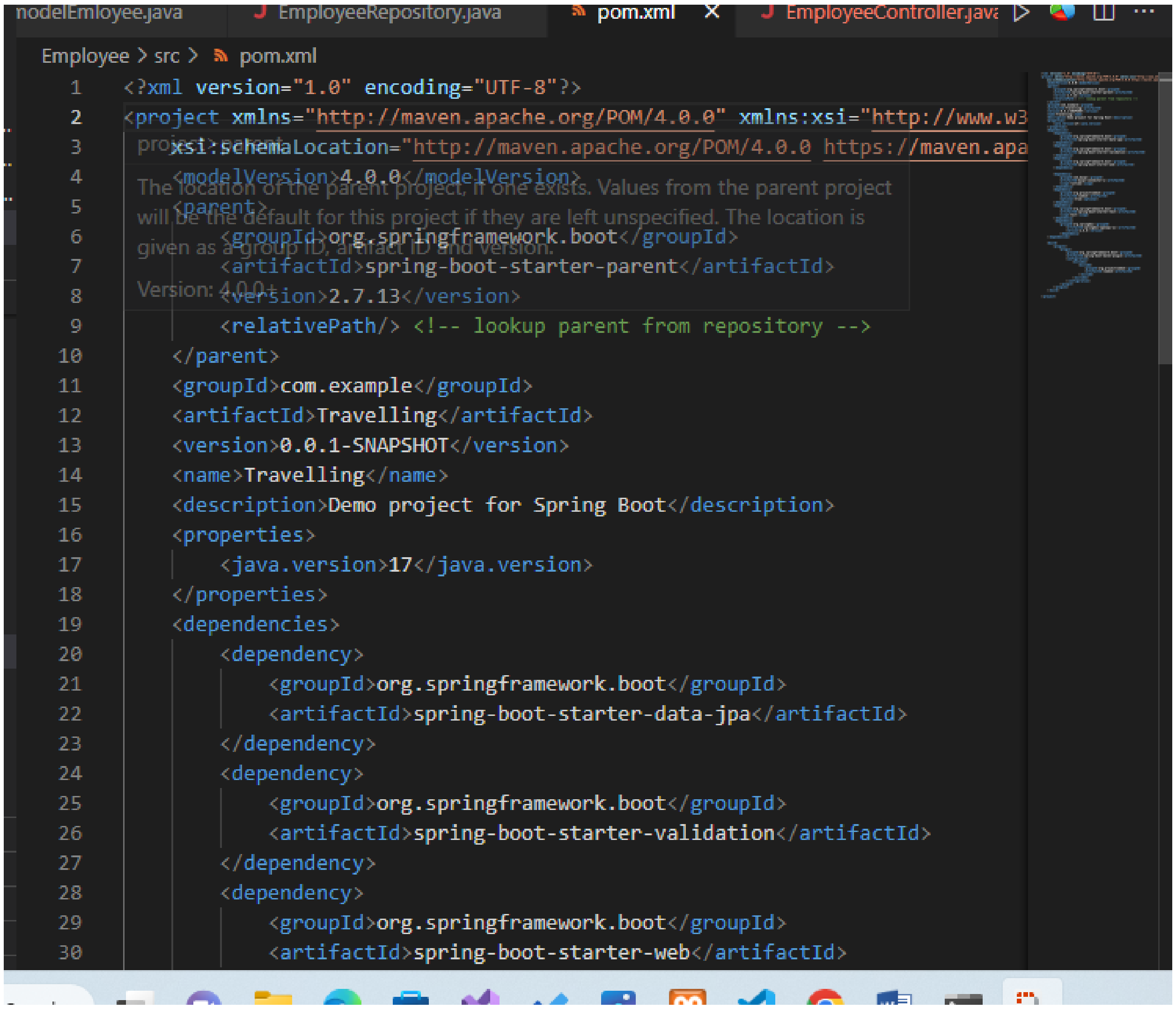
Steps for creating spring boot project

- Go to website to search a springboot initializer
- Give the group and artfact name
- Choose the generation and language use as java or cotllin
- Select Dependences on the right side, Most of dependences used are web JPA , Security and Lombok.
- Click Generate to generate your project
- Extract your project then open to the text editor by using command or another way. The text editor can be Vs code.

Create database on xamp.

b)

dependency com.xml

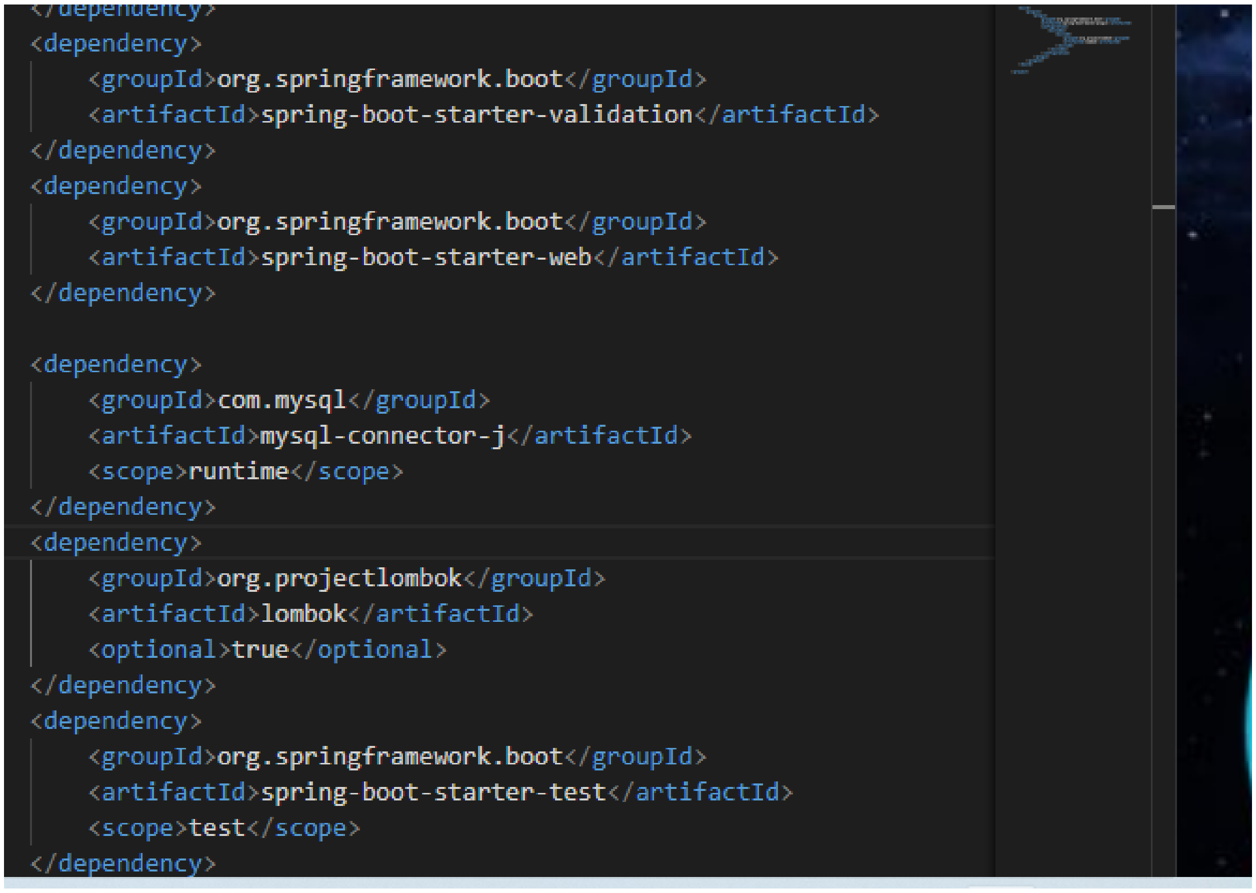


```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apa
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.13</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>Travelling</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Travelling</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
```

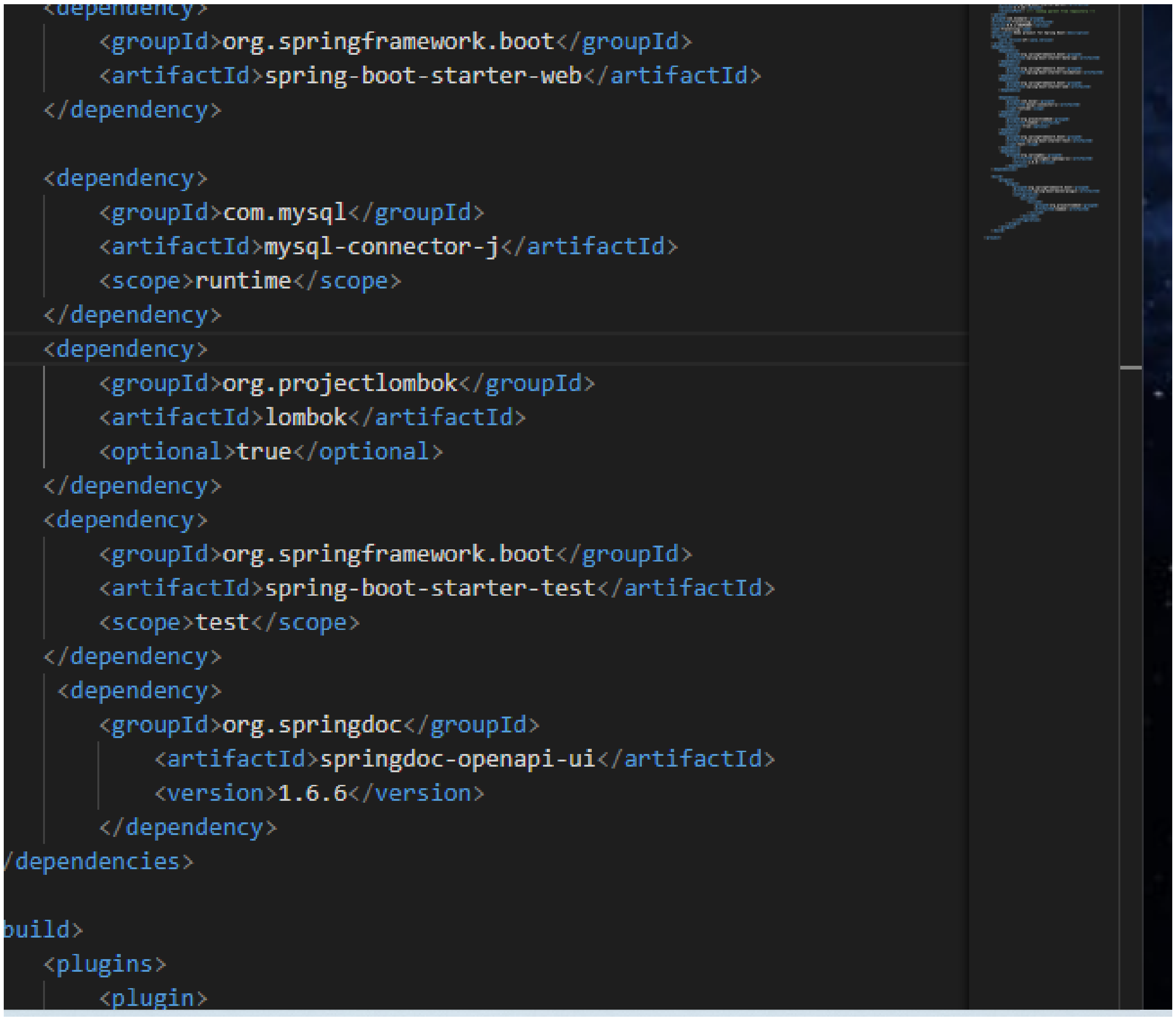
project > parent

The location of the parent project, if one exists. Values from the parent project will be the default for this project if they are left unspecified. The location is given as a group ID, artifact ID and version.

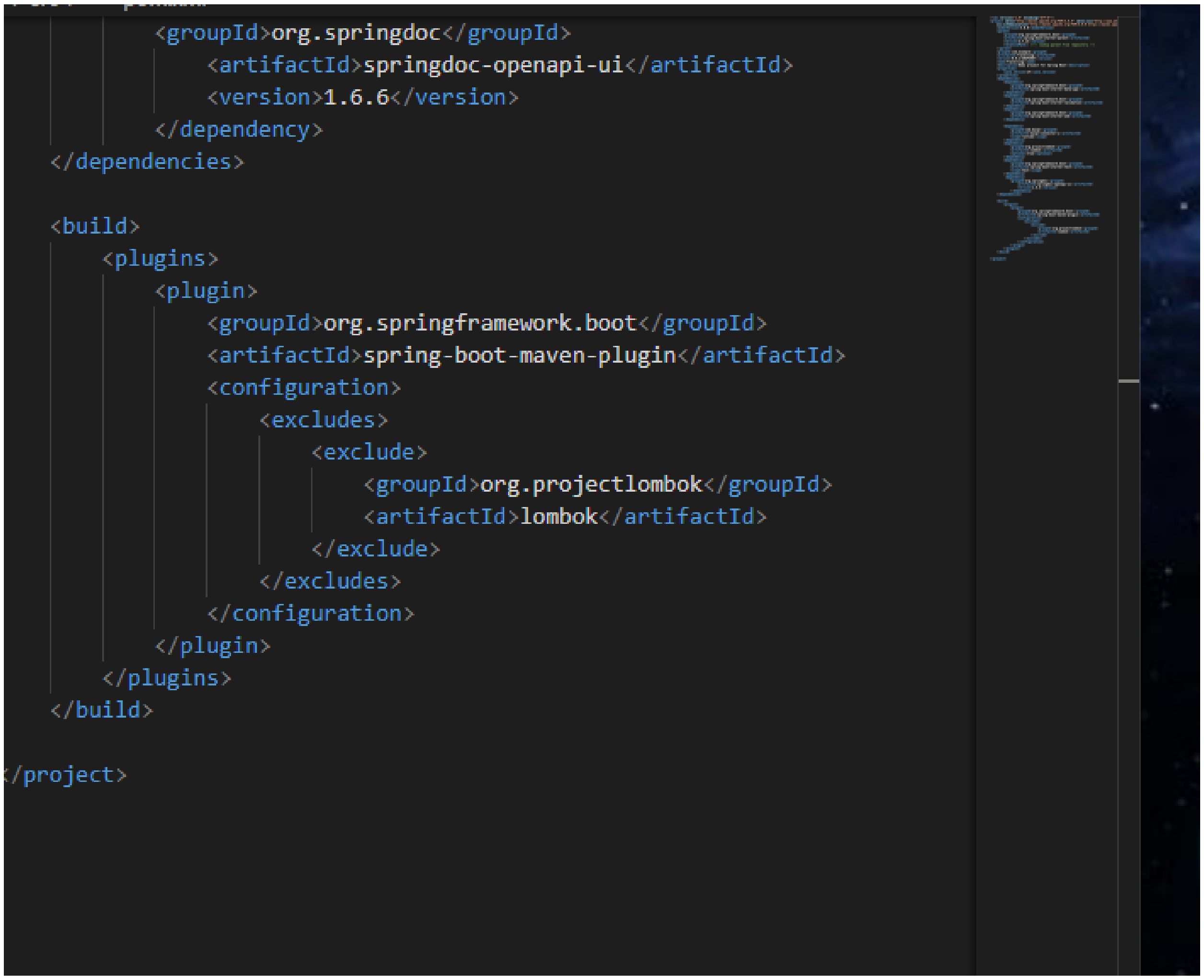
Version: 4.0.0+



```xml
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

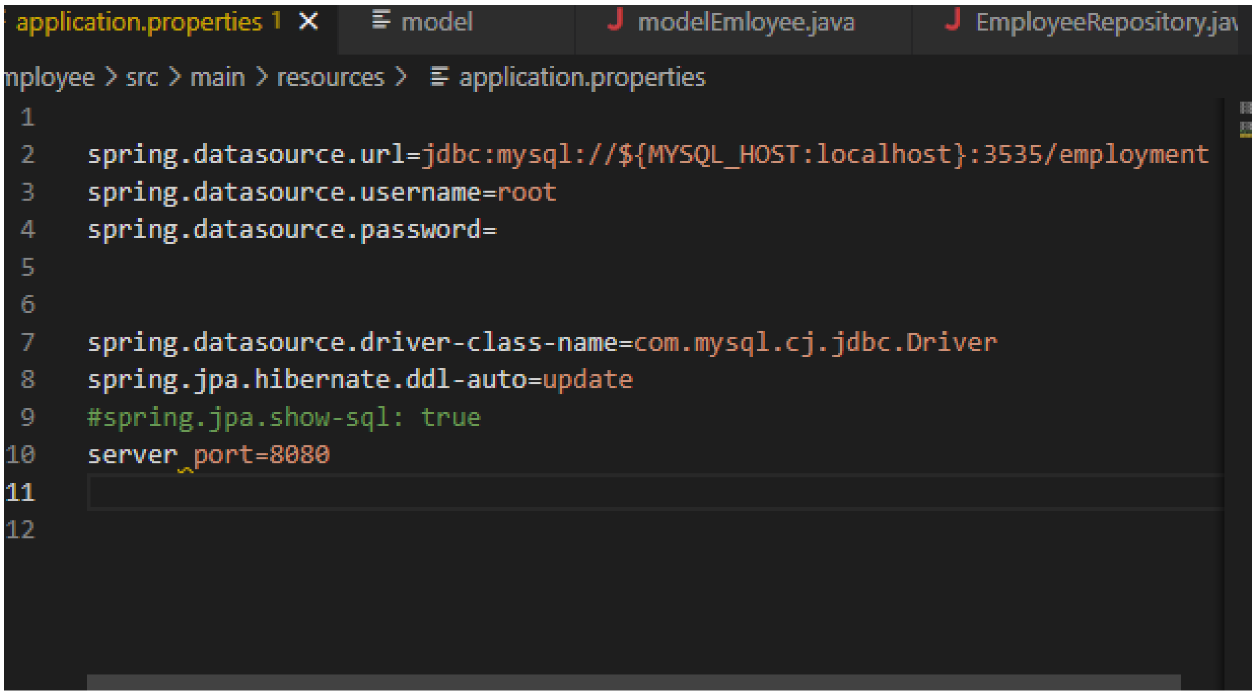


```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
     <dependency>
        <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-ui</artifactId>
            <version>1.6.6</version>
        </dependency>
/dependencies>

build>
    <plugins>
        <plugin>
```



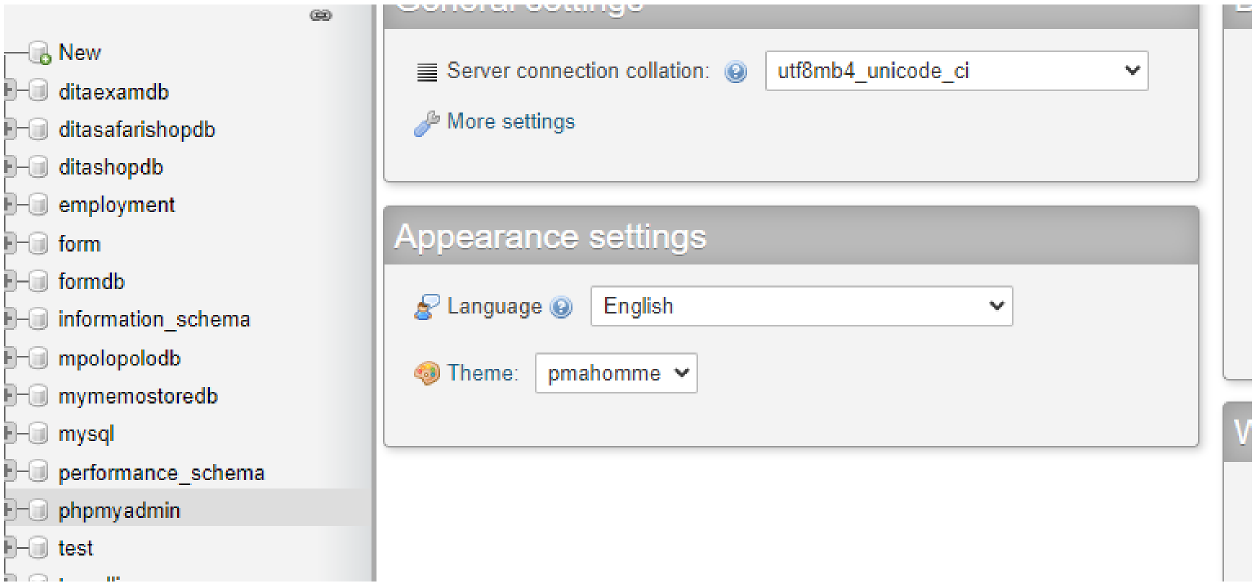
```xml
            <groupId>org.springdoc</groupId>
                <artifactId>springdoc-openapi-ui</artifactId>
                <version>1.6.6</version>
            </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

b) Applibation properties



```
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3535/employment
spring.datasource.username=root
spring.datasource.password=


spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
#spring.jpa.show-sql: true
server.port=8080
```

Database Created from xamp



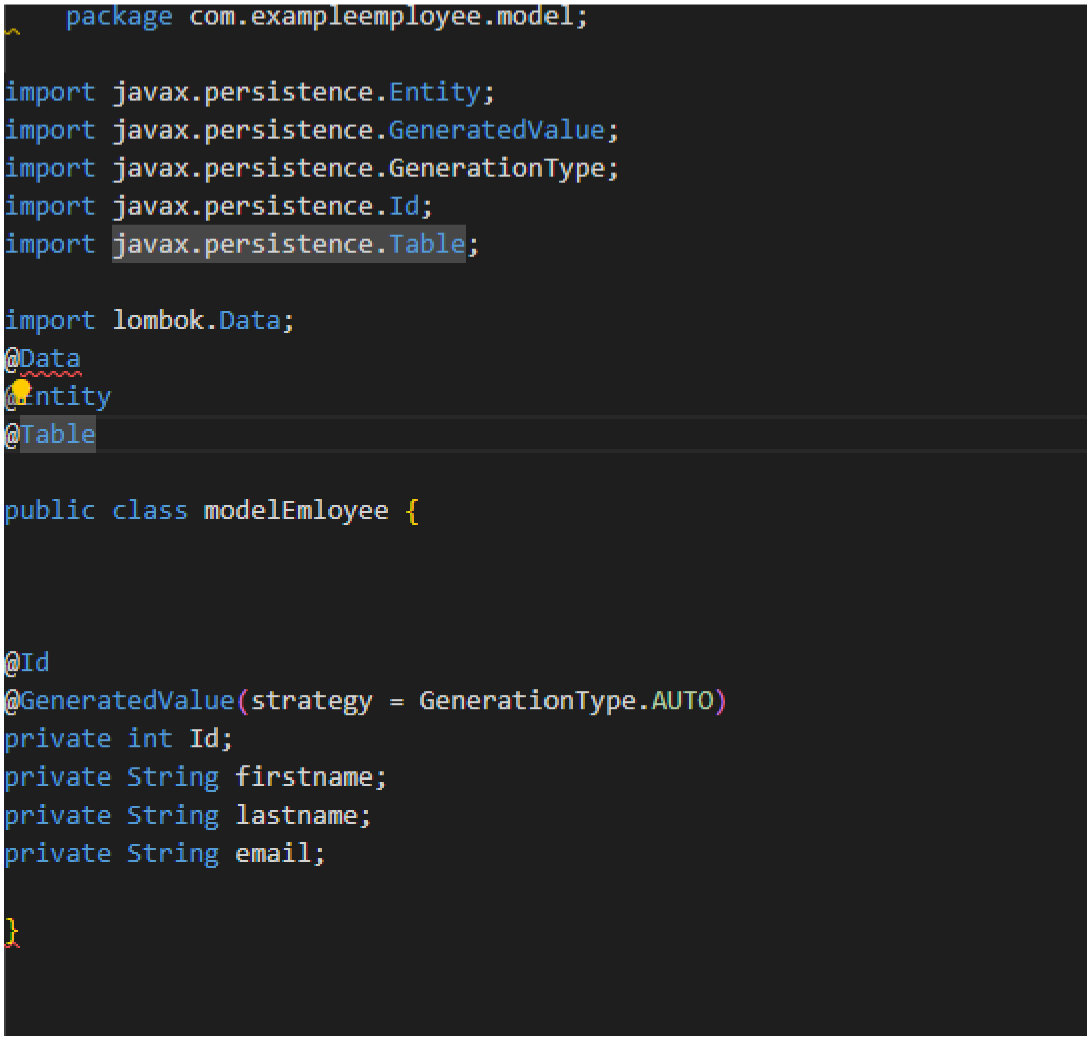
Server connection collation: utf8mb4_unicode_ci

Language English

Theme: pmahomme

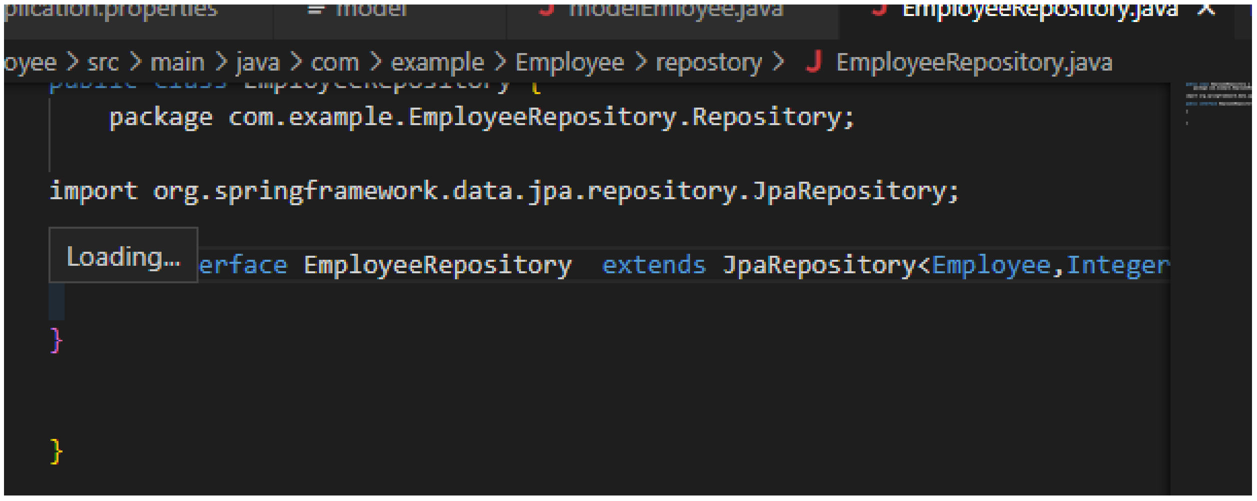
c) To create model

d) Model is create within a package inside the model folder there is a file which contain all objecties and all data required to be inserted in the database include id , first name ,lasr]tname to be stored in the database.



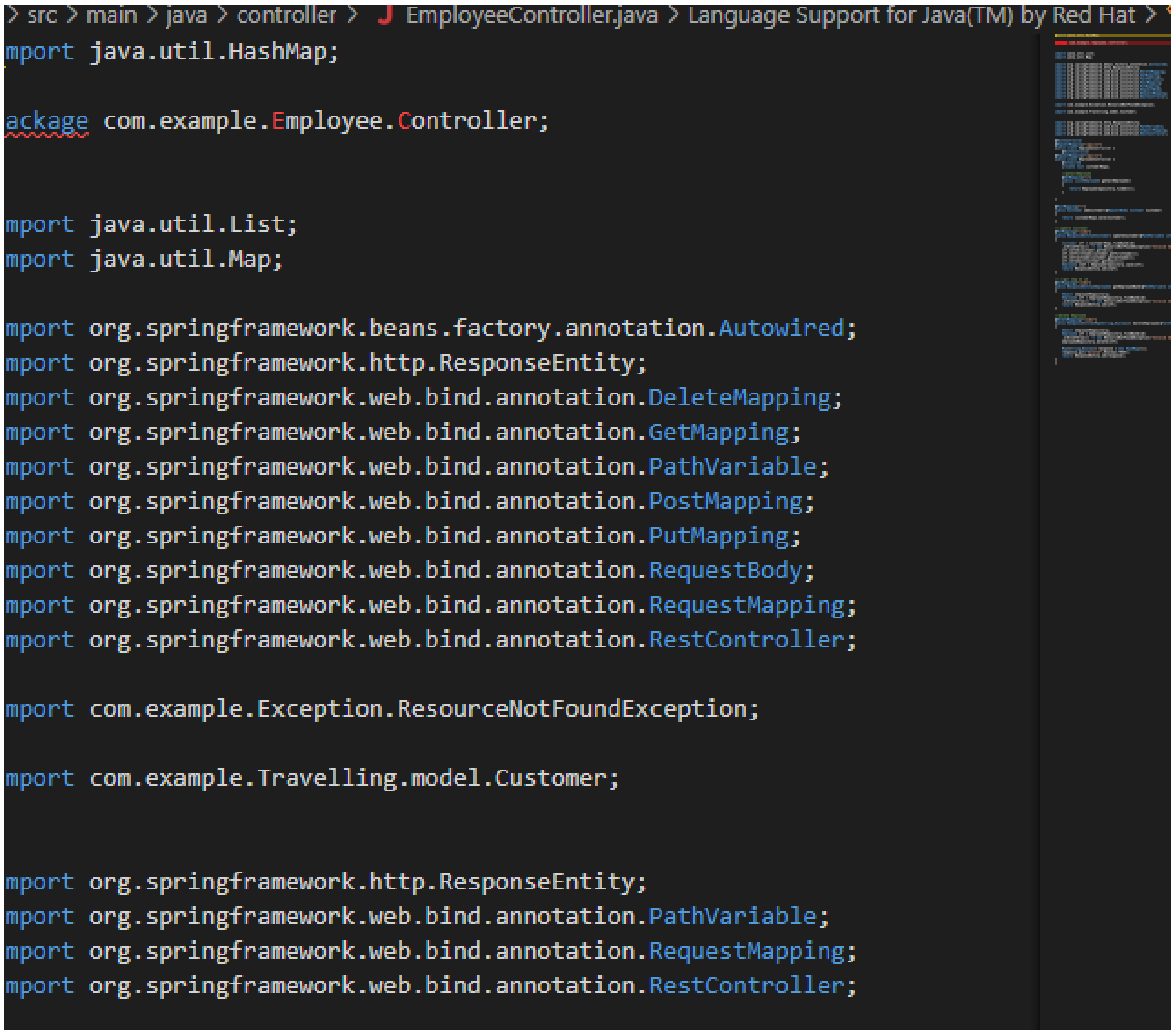
```
   package com.exampleemployee.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;
@Data
@Entity
@Table

public class modelEmloyee {



@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int Id;
private String firstname;
private String lastname;
private String email;

}
```

e)



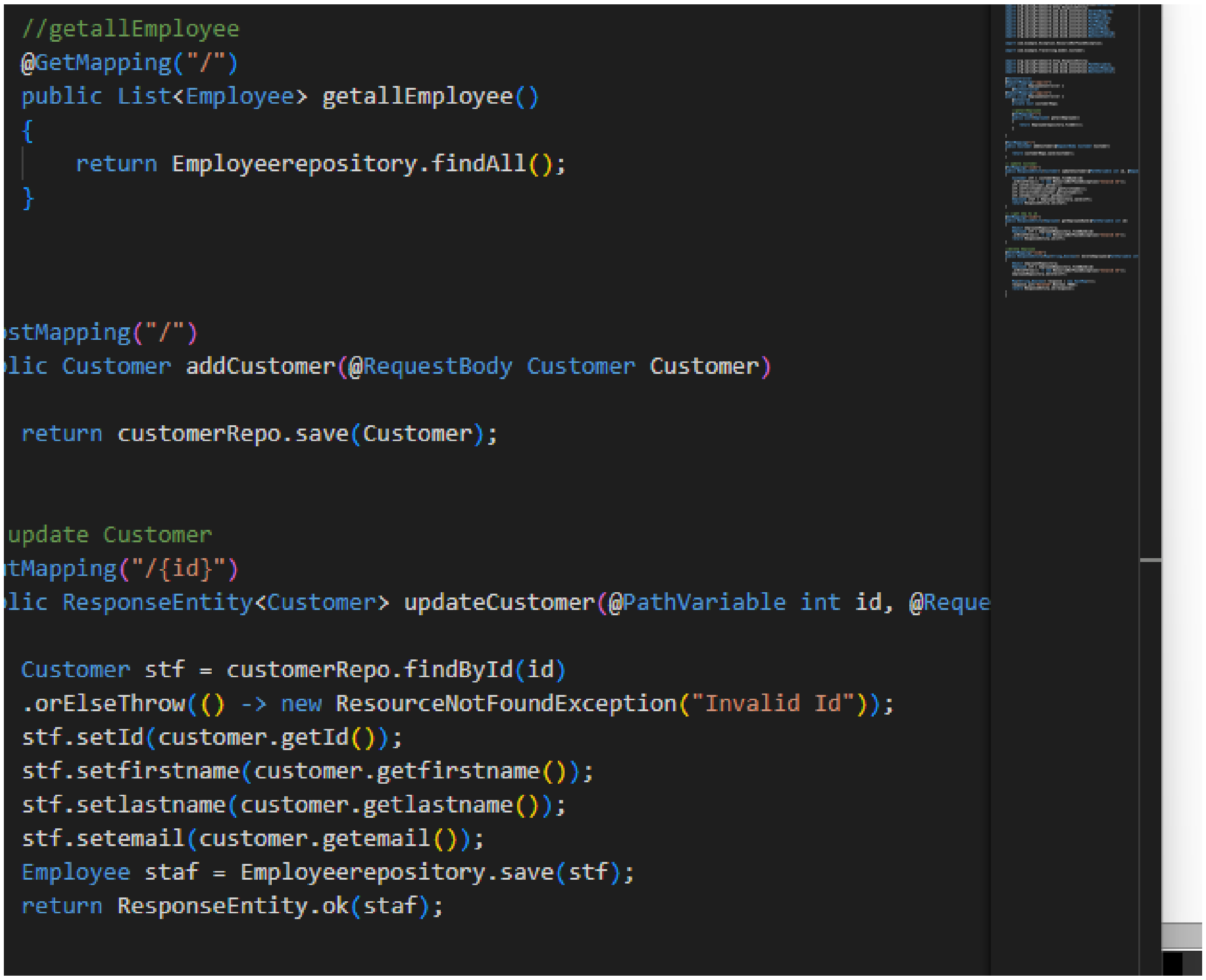
plication.properties
model
modelEmloyee.java
EmployeeRepository.java
oyee > src > main > java > com > example > Employee > repostory > EmployeeRepository.java
Loading...

```
    package com.example.EmployeeRepository.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

         erface EmployeeRepository  extends JpaRepository<Employee,Integer

}


}
```

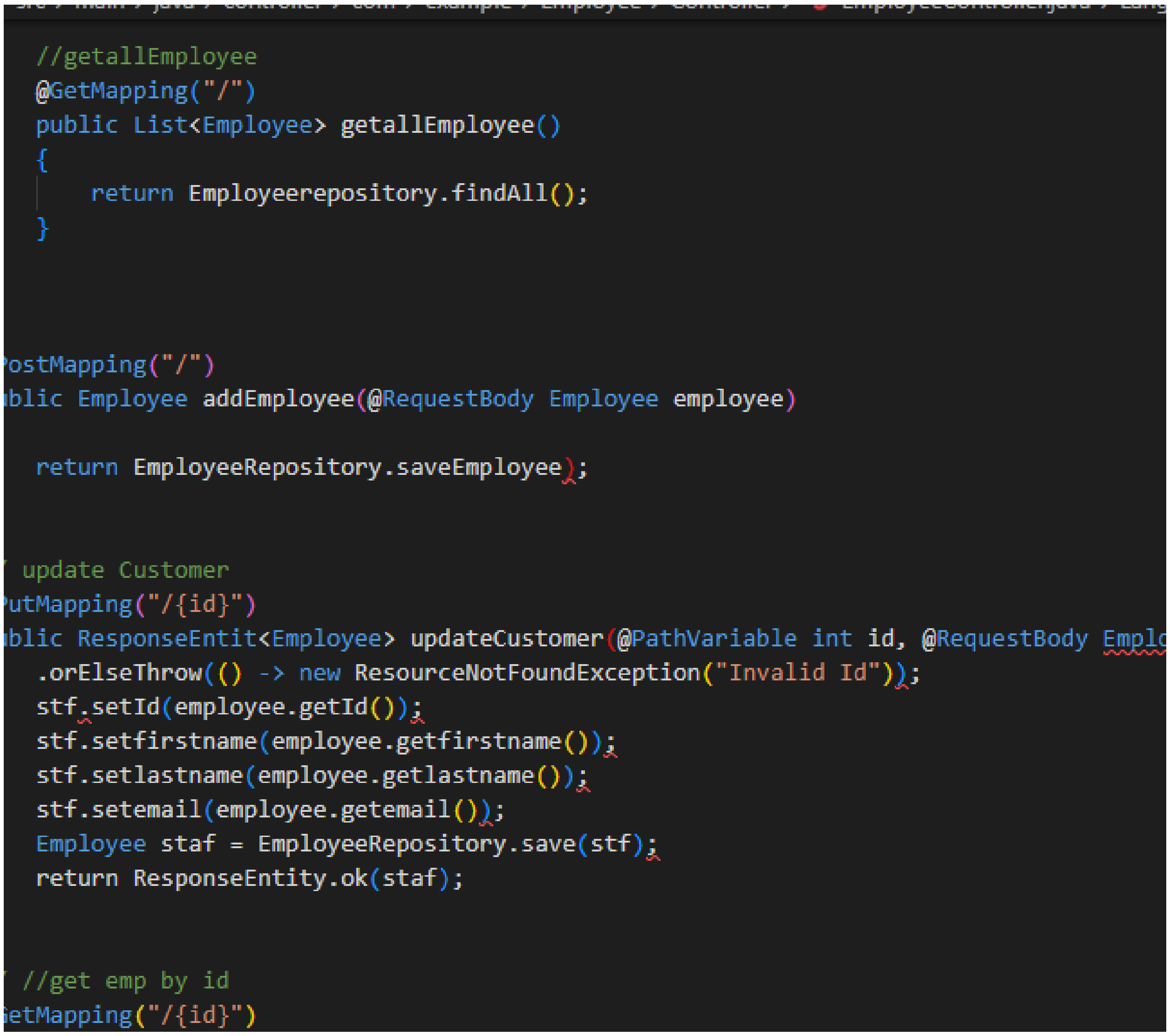


> src > main > java > controller > EmployeeController.java > Language Support for Java(TM) by Red Hat >

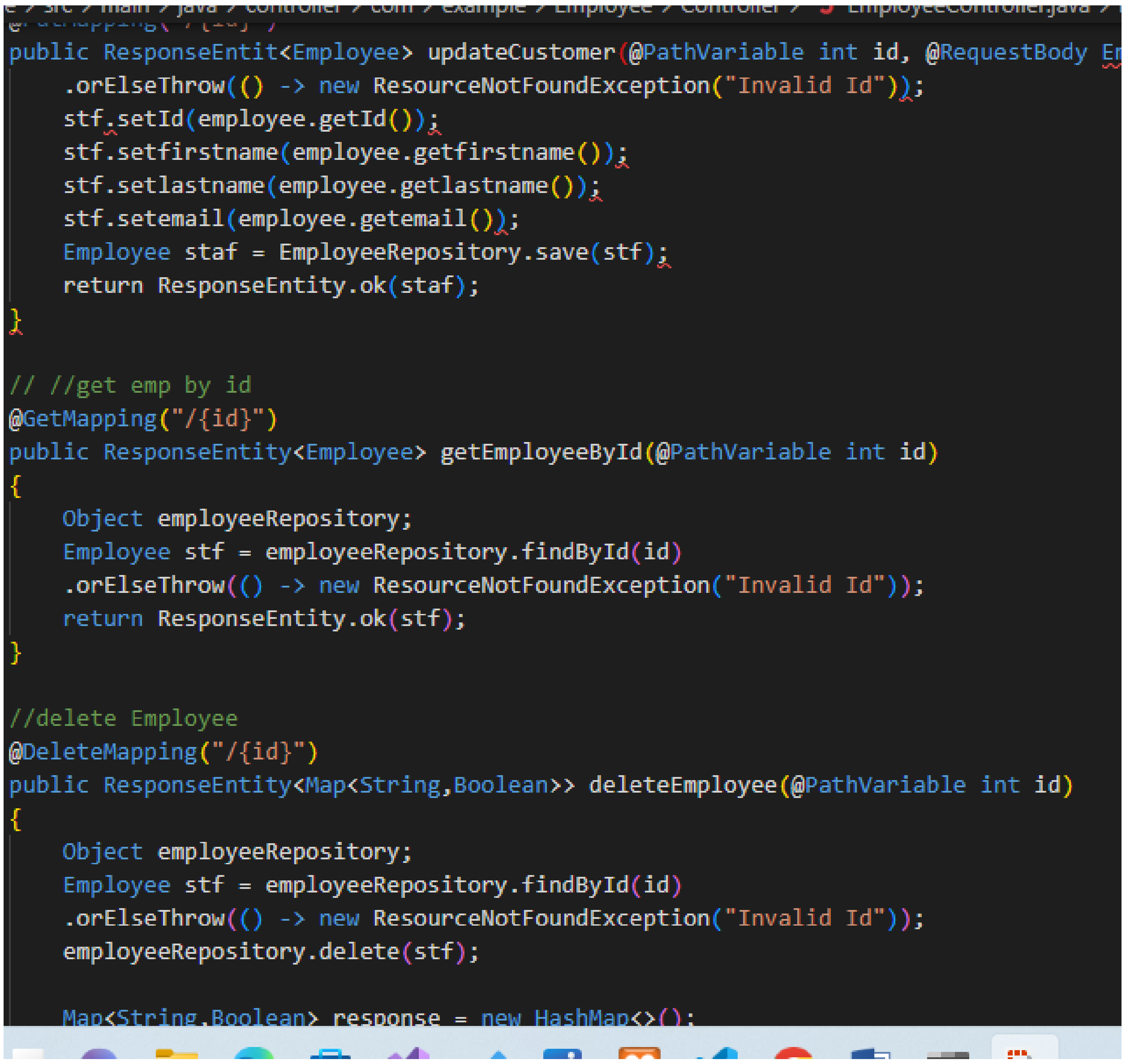
```
mport java.util.HashMap;

ackage com.example.Employee.Controller;


mport java.util.List;
mport java.util.Map;

mport org.springframework.beans.factory.annotation.Autowired;
mport org.springframework.http.ResponseEntity;
mport org.springframework.web.bind.annotation.DeleteMapping;
mport org.springframework.web.bind.annotation.GetMapping;
mport org.springframework.web.bind.annotation.PathVariable;
mport org.springframework.web.bind.annotation.PostMapping;
mport org.springframework.web.bind.annotation.PutMapping;
mport org.springframework.web.bind.annotation.RequestBody;
mport org.springframework.web.bind.annotation.RequestMapping;
mport org.springframework.web.bind.annotation.RestController;

mport com.example.Exception.ResourceNotFoundException;

mport com.example.Travelling.model.Customer;


mport org.springframework.http.ResponseEntity;
mport org.springframework.web.bind.annotation.PathVariable;
mport org.springframework.web.bind.annotation.RequestMapping;
mport org.springframework.web.bind.annotation.RestController;
```



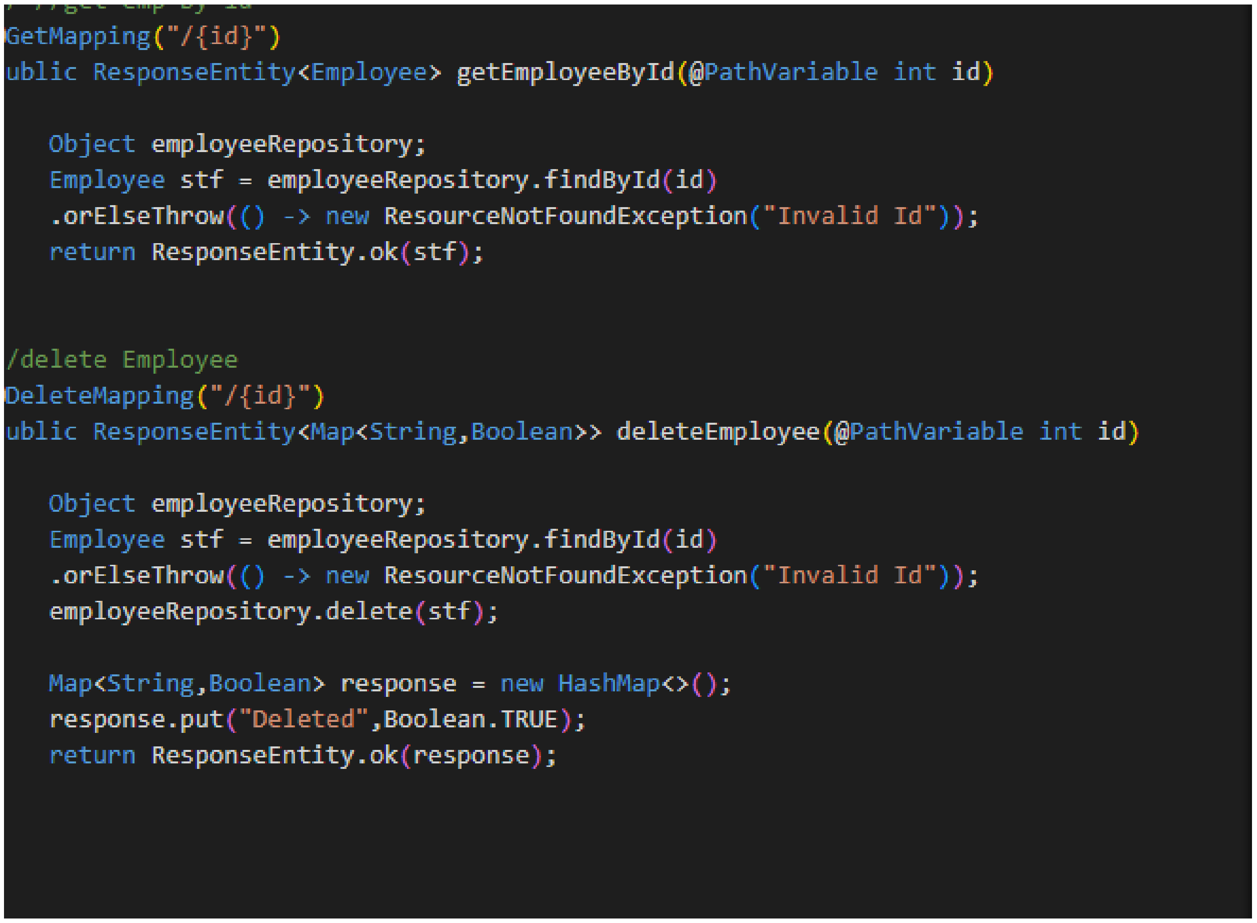
```
//getallEmployee
@GetMapping("/")
public List<Employee> getallEmployee()
{
    return Employeerepository.findAll();
}



stMapping("/")
lic Customer addCustomer(@RequestBody Customer Customer)

 return customerRepo.save(Customer);


update Customer
tMapping("/{id}")
lic ResponseEntity<Customer> updateCustomer(@PathVariable int id, @Reque

 Customer stf = customerRepo.findById(id)
 .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
 stf.setId(customer.getId());
 stf.setfirstname(customer.getfirstname());
 stf.setlastname(customer.getlastname());
 stf.setemail(customer.getemail());
 Employee staf = Employeerepository.save(stf);
 return ResponseEntity.ok(staf);
```



```
  //getallEmployee
  @GetMapping("/")
  public List<Employee> getallEmployee()
  {
      return Employeerepository.findAll();
  }



ostMapping("/")
blic Employee addEmployee(@RequestBody Employee employee)

  return EmployeeRepository.saveEmployee);


 update Customer
utMapping("/{id}")
blic ResponseEntit<Employee> updateCustomer(@PathVariable int id, @RequestBody Emplo
  .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
  stf.setId(employee.getId());
  stf.setfirstname(employee.getfirstname());
  stf.setlastname(employee.getlastname());
  stf.setemail(employee.getemail());
  Employee staf = EmployeeRepository.save(stf);
  return ResponseEntity.ok(staf);


 //get emp by id
etMapping("/{id}")
```

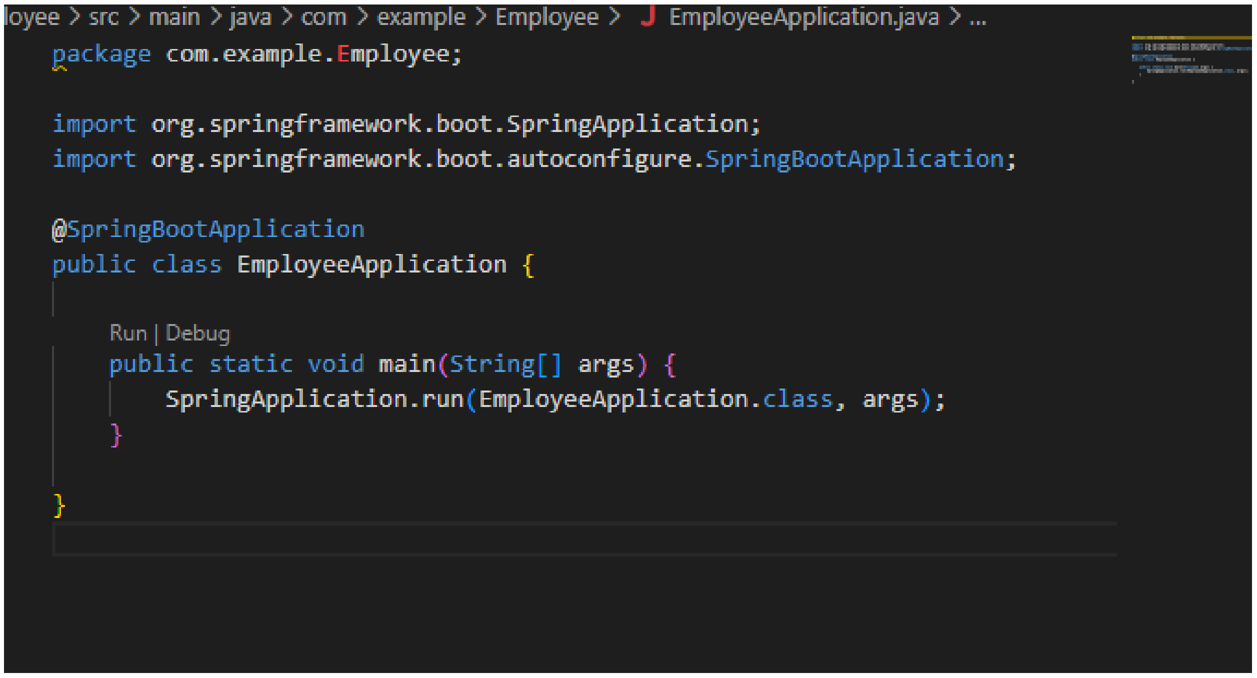


```
public ResponseEntit<Employee> updateCustomer(@PathVariable int id, @RequestBody Em
    .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
    stf.setId(employee.getId());
    stf.setfirstname(employee.getfirstname());
    stf.setlastname(employee.getlastname());
    stf.setemail(employee.getemail());
    Employee staf = EmployeeRepository.save(stf);
    return ResponseEntity.ok(staf);
}

// //get emp by id
@GetMapping("/{id}")
public ResponseEntity<Employee> getEmployeeById(@PathVariable int id)
{
    Object employeeRepository;
    Employee stf = employeeRepository.findById(id)
    .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
    return ResponseEntity.ok(stf);
}

//delete Employee
@DeleteMapping("/{id}")
public ResponseEntity<Map<String,Boolean>> deleteEmployee(@PathVariable int id)
{
    Object employeeRepository;
    Employee stf = employeeRepository.findById(id)
    .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
    employeeRepository.delete(stf);

    Map<String,Boolean> response = new HashMap<>();
```



```
GetMapping("/{id}")
ublic ResponseEntity<Employee> getEmployeeById(@PathVariable int id)

   Object employeeRepository;
   Employee stf = employeeRepository.findById(id)
   .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
   return ResponseEntity.ok(stf);


/delete Employee
DeleteMapping("/{id}")
ublic ResponseEntity<Map<String,Boolean>> deleteEmployee(@PathVariable int id)

   Object employeeRepository;
   Employee stf = employeeRepository.findById(id)
   .orElseThrow(() -> new ResourceNotFoundException("Invalid Id"));
   employeeRepository.delete(stf);

   Map<String,Boolean> response = new HashMap<>();
   response.put("Deleted",Boolean.TRUE);
   return ResponseEntity.ok(response);
```

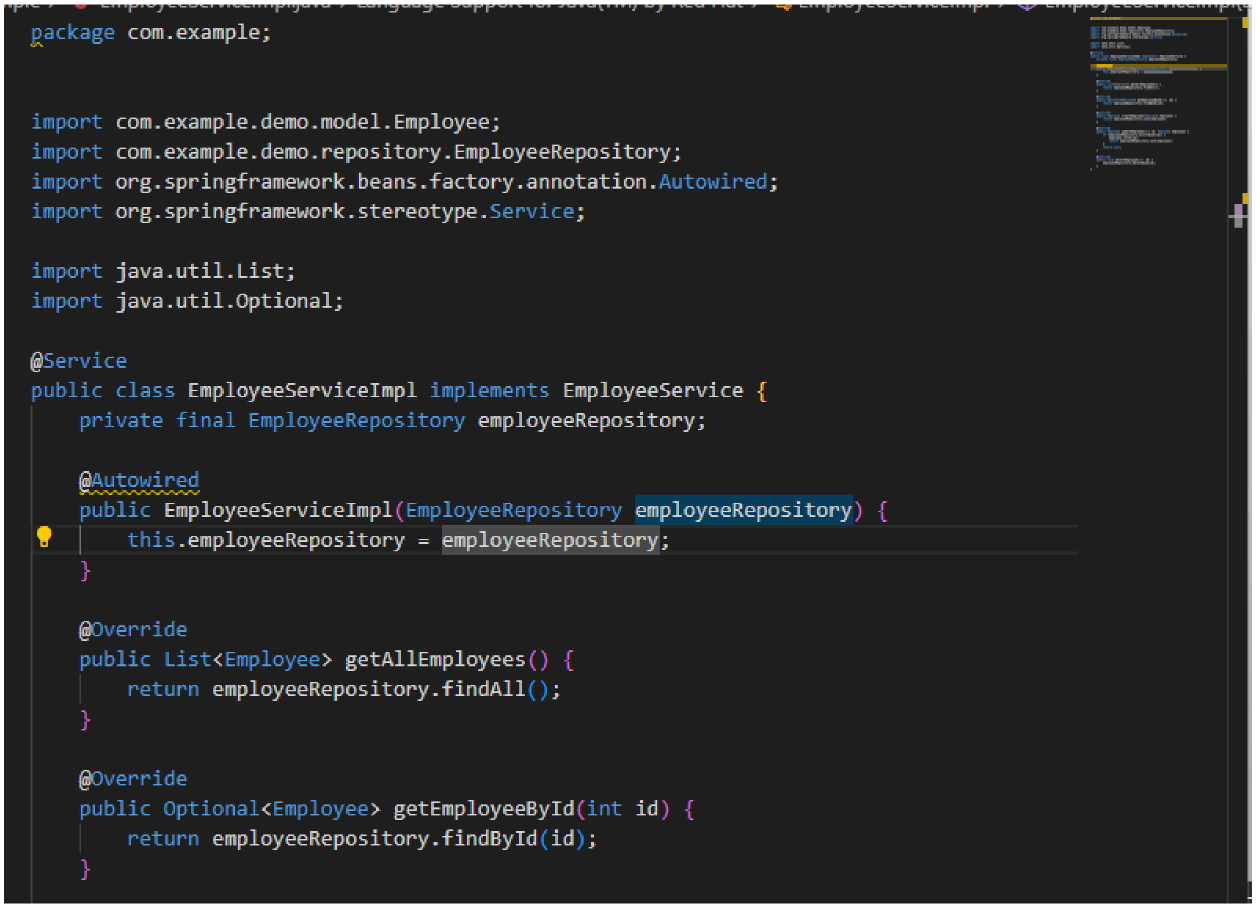
Main class application.java



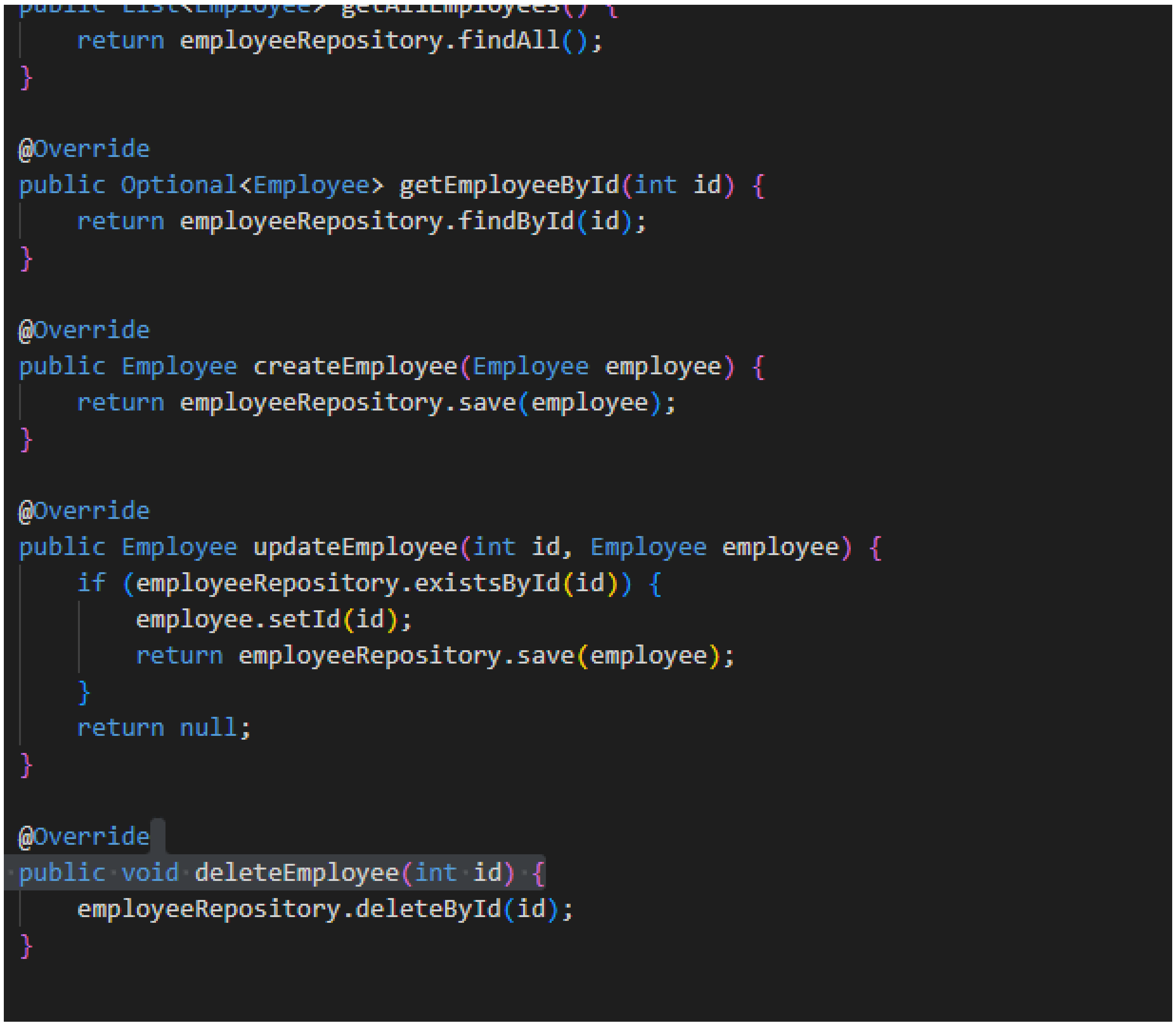
loyee > src > main > java > com > example > Employee > EmployeeApplication.java > ...

```
package com.example.Employee;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EmployeeApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(EmployeeApplication.class, args);
    }

}
```

Controller



```
package com.example;


import com.example.demo.model.Employee;
import com.example.demo.repository.EmployeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class EmployeeServiceImpl implements EmployeeService {
    private final EmployeeRepository employeeRepository;

    @Autowired
    public EmployeeServiceImpl(EmployeeRepository employeeRepository) {
        this.employeeRepository = employeeRepository;
    }

    @Override
    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }

    @Override
    public Optional<Employee> getEmployeeById(int id) {
        return employeeRepository.findById(id);
    }
```



```java
public List<Employee> getAllEmployees() {
    return employeeRepository.findAll();
}

@Override
public Optional<Employee> getEmployeeById(int id) {
    return employeeRepository.findById(id);
}

@Override
public Employee createEmployee(Employee employee) {
    return employeeRepository.save(employee);
}

@Override
public Employee updateEmployee(int id, Employee employee) {
    if (employeeRepository.existsById(id)) {
        employee.setId(id);
        return employeeRepository.save(employee);
    }
    return null;
}

@Override
public void deleteEmployee(int id) {
    employeeRepository.deleteById(id);
}
```