



**Project**  
☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ **Java** ☐ Kotlin ☐ Groovy

**Spring Boot**  
☐ 3.2.0 (SNAPSHOT) ☐ 3.2.0 (M1) ☐ 3.1.3 (SNAPSHOT) ☒ **3.1.2**  
☐ 3.0.10 (SNAPSHOT) ☐ 3.0.9 ☐ 2.7.15 (SNAPSHOT) ☐ 2.7.14

#### Project Metadata

Group   
Artifact   
Name   
Description   
Package name   
Packaging ☒ Jar ☐ War  
Java ☐ 20 ☒ **17** ☐ 11 ☐ 8

#### Dependencies

ADD DEPENDENCIES... CTRL + B

##### JDBC API SQL

Database Connectivity API that defines how a client may connect and query a database.

##### MySQL Driver SQL

MySQL JDBC driver.

##### Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

##### Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

##### Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
```

```
<parent>
```

```
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-starter-parent</artifactId>
```

```
<version>3.1.1</version>
```

```
<relativePath/> <!-- lookup parent from repository -->
```

```
</parent>
```

```
<groupId>com.myappecommerce</groupId>
```

```
<artifactId>myappecommerce</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<name>myappecommerce</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-jdbc</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-security</artifactId>-->
<!-- </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>
```

li

Configure MySQL/PostgreSQL Database: Using application.properties or application.yml file for database

connectivity, create a properties file.

```
spring.datasource.url=jdbc:mysql://localhost:3306/product?useSSL=false&serverTimezone=UTC
```

```
spring.datasource.username=root
```

```
spring.datasource.password=
```

```
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.show-sql=true
```

Create Models

Write a code for employee model

Employee Repository: Write a code

```
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
```

@Entity

```
public class Employee {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private String employeeId;
```

```
    private String name;
```

```
    private int age;
```

```
    private String levelEducation;
```

```
    private String znzId;
```

```
    private String university;
```

```
}
```

Employee Repository: Write a code for employee repository

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

```
public interface EmployeeRepository extends JpaRepository<Employee, Long> {  
  
}
```

Services Layer: Write a code for employee service layer

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Service;
```

```
import java.util.List;
```

```
@Service
```

```
public class EmployeeService {
```

```
    private final EmployeeRepository employeeRepository;
```

```
@Autowired
```

```
public EmployeeService(EmployeeRepository employeeRepository) {
```

```
    this.employeeRepository = employeeRepository;
```

```
}
```

```
public List<Employee> getAllEmployees() {
```

```
    return employeeRepository.findAll();
```

```
}
```

```
public Employee createEmployee(Employee employee) {
```

```
    return employeeRepository.save(employee);
```

```
}
```

```
}
```

Controller Layer: Write a code for GET,POST,DELETE,PUT and GET by ID

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/api/employees")
```

```
public class EmployeeController {
```

```
    private final EmployeeService employeeService;
```

```
    @Autowired
```

```
    public EmployeeController(EmployeeService employeeService) {
```

```
        this.employeeService = employeeService;
```

```
    }
```

```
    @GetMapping
```

```
    public List<Employee> getAllEmployees() {
```

```
        return employeeService.getAllEmployees();
```

```
    }
```

```
    @PostMapping
```

```
    public Employee createEmployee(@RequestBody Employee employee) {
```

```
        return employeeService.createEmployee(employee);
```

```
    }
```

```
}
```

Create anEmployee Application class which will be used as Main Class.

Show how you can test your API using API client.

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

```
@SpringBootApplication
```

```
public class EmployeeApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(EmployeeApplication.class, args);
```

```
    }
```

```
}
```

```
{
```

```
    "employeeId": "m122",
```

```
    "name": "suleiman ali mpwani",
```

```
    "age": 23
```

```
    "levelEducation": "cheti",
```

```
    "znzId": "Z00091",
```

```
    "university": "Suza"
```

```
}
```

