

# Digital Nuture program 4.0

## Spring Core and Maven

### Week 3 Mandatory hands-on

#### 1) Configuring a Basic Spring Application

##### **pom.xml**

```
<dependencies>
  <!-- Spring Core -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.30</version>
  </dependency>
</dependencies>
```

##### **MainApp.java**

```
package com.library;

import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
        ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = context.getBean("bookService",
        BookService.class);
        bookService.addBook("Spring in Action");
    }
}
```

##### **BookRepository.java**

##### **package com.library.repository;**

```
public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Saving book: " + bookName);
    }
}
```



## 2) Implementing Dependency Injection

### applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Repository Bean -->
    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>

    <!-- Service Bean with DI -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>
</beans>
```

### BookRepository.java

```
package com.library.repository;
```

```
public class BookRepository {
    public void save() {
        System.out.println("Book saved to the repository.");
    }
}
```

### BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void performService() {
        System.out.println("Service Layer: Performing service...");
        bookRepository.save();
    }
}
```

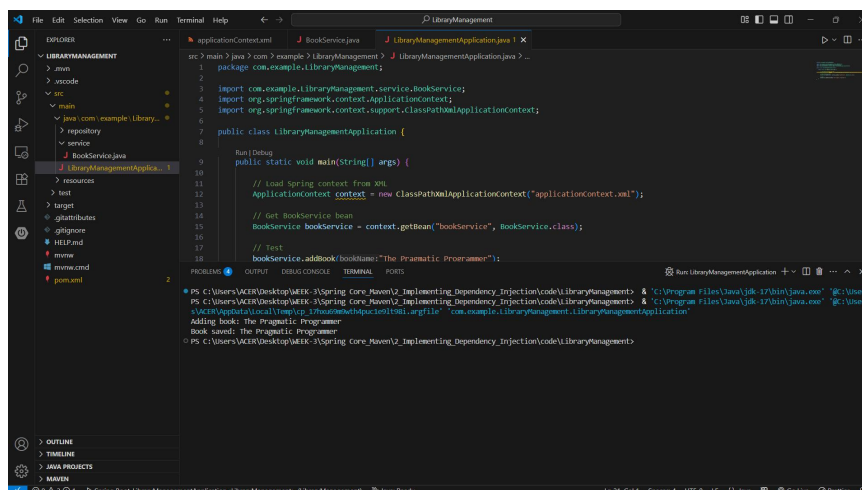
## MainApp.java

```
package com.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
        ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService)
        context.getBean("bookService");
        bookService.performService();
    }
}
```

## Output :



## 3) Creating and Configuring a Maven Project

### BookRepository.java

```
package com.example.LibraryManagement.repository;

public class BookRepository {

    public void saveBook(String bookName) {
        System.out.println("Book saved: " + bookName);
    }
}
```

### **BookService.java**

```
package com.example.LibraryManagement.service;
import com.example.LibraryManagement.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

### **AppConfig.java**

```
package com.example.LibraryManagement;

import com.example.LibraryManagement.repository.BookRepository;
import com.example.LibraryManagement.service.BookService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public BookRepository bookRepository() {
        return new BookRepository();
    }

    @Bean
    public BookService bookService() {
        BookService bookService = new BookService();
        bookService.setBookRepository(bookRepository());
        return bookService;
    }
}
```

### **LibraryManagementApplication**

```
package com.example.LibraryManagement;

import com.example.LibraryManagement.service.BookService;
import org.springframework.context.ApplicationContext;
```

```

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        BookService bookService = context.getBean(BookService.class);
        bookService.addBook("Refactoring by Martin Fowler");
    }
}

```

### **applicationContext.xml**

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.library.repository.BookRepository"
/>

    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>

</beans>

```

### **pom.xml**

```

<?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>3.3.1</version>
        <relativePath/>
    </parent>

    <groupId>com.example</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>LibraryManagement</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-actuator</artifactId>
  </dependency>

  <!-- Spring Boot Starter Data JPA -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </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>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.11.0</version>
    </plugin>
  </plugins>
</build>

```

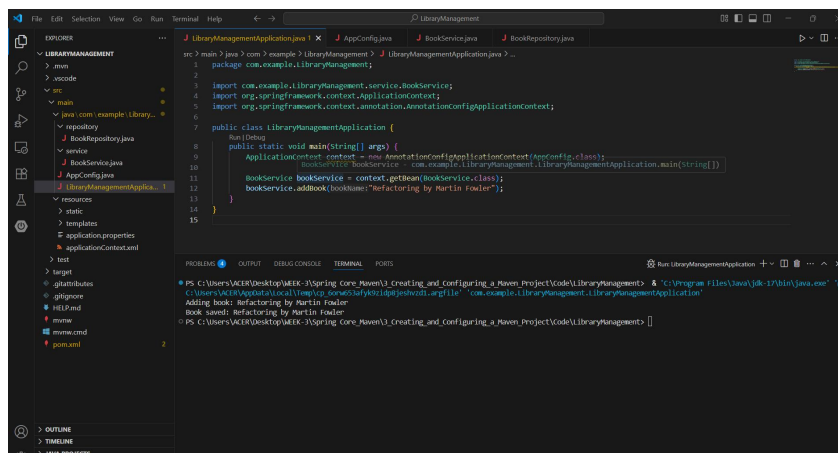
```

<configuration>
  <source>17</source>
  <target>17</target>
  <annotationProcessorPaths>
    <path>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.32</version>
    </path>
  </annotationProcessorPaths>
</configuration>
</plugin>
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

</project>

```

**Output :**



## 4) Spring Data JPA - Quick Example

### Country.java

```

package com.cognizant.ormlearn.model;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.persistence.Column;
@Entity

```



```

@Table(name = "country")
public class Country {
    @Id
    @Column(name = "co_code")
    private String code;
    @Column(name = "co_name")
    private String name;
    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}

```

### **CountryRepository.java**

```

package com.cognizant.ormlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.cognizant.ormlearn.model.Country;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {
}

```

### **CountryService.java**

```

package com.cognizant.ormlearn.service;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}

```

```
}  
}
```

### **OrmlearnApplication.java**

```
package com.cognizant.ormlearn;  
  
import java.util.List;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.service.CountryService;  
  
@SpringBootApplication  
public class OrmlearnApplication {  
  
    private static final Logger LOGGER =  
        LoggerFactory.getLogger(OrmlearnApplication.class);  
    private static CountryService countryService;  
  
    public static void main(String[] args) {  
        ApplicationContext context =  
            SpringApplication.run(OrmlearnApplication.class, args);  
        countryService = context.getBean(CountryService.class);  
  
        testGetAllCountries();  
    }  
  
    private static void testGetAllCountries() {  
        LOGGER.info("Start testGetAllCountries");  
        List<Country> countries = countryService.getAllCountries();  
        countries.forEach(country -> LOGGER.info("Country: {}", country));  
        LOGGER.info("End testGetAllCountries");  
    }  
}
```

### **OrmlearnApplicationTests.java**

```
package com.cognizant.ormlearn;  
  
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
  
@SpringBootTest  
class OrmlearnApplicationTests {  
  
    @Test
```

```

        void contextLoads() {
        }
    }
}

```

## Pom.xml

```

<?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>3.5.3</version>
        <relativePath/>
    </parent>
    <groupId>com.cognizant</groupId>
    <artifactId>ormlearn</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>ormlearn</name>
    <description>Demo project for Spring Data JPA and
Hibernate</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-web</artifactId>
        </dependency>
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>

```

```

<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
</project>

```

## Output :

The screenshot shows an IDE with the following components:

- EXPLORER:** Shows the project structure with folders like .mvn, .vscode, src, and main. The main folder contains a java package com.cognizant.ormlearn with sub-packages model, repository, service, and resources.
- application.properties:** Contains the following configuration:
 

```

1 # logging
2 logging.level.org.springframework=info
3 logging.level.com.cognizant=debug
4 logging.level.org.hibernate.SQL=debug
5 logging.level.org.hibernate.type.descriptor.sql=trace
6
7 # Database
8 spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
9 spring.datasource.username=root
10 spring.datasource.password=1234
11 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
12
13 # Hibernate
14 spring.jpa.hibernate.ddl-auto=update
15 spring.jpa.show-sql=true
16 # Optional explicit dialect if needed:
17 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
18

```
- Terminal:** Shows the output of the application run. It includes messages from JPA, Hibernate, and the application itself. The application starts successfully and prints the results of a test:
 

```

2025-07-05T10:37:13.348495:30 INFO 43048 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-05T10:37:13.348495:30 WARN 43048 --- [ restartedMain] JpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default; therefore, database queries may be performed during view rendering; explicitly configure spring.jpa.open-in-view to disable this warning
2025-07-05T10:37:13.845495:30 INFO 43048 --- [ restartedMain] o.s.b.a.a.optional.LivenessServer : Liveness server is running on port 35729
2025-07-05T10:37:13.897495:30 INFO 43048 --- [ restartedMain] o.s.b.a.embedded.TomcatEmbeddedServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-05T10:37:13.912495:30 INFO 43048 --- [ restartedMain] c.c.ormlearn.OrmlearnApplication : Started OrmlearnApplication in 6.245 seconds (process running for 6.753)
2025-07-05T10:37:14.183495:30 INFO 43048 --- [ restartedMain] c.c.ormlearn.OrmlearnApplication : Start testGetAllCountries
2025-07-05T10:37:14.136495:30 DEBUG 43048 --- [ restartedMain] org.hibernate.SQL : select c1_0.co_code,c1_0.co_name from country c1_0
2025-07-05T10:37:14.183495:30 INFO 43048 --- [ restartedMain] c.c.ormlearn.OrmlearnApplication : Country: Country [code-IN, name-India]
2025-07-05T10:37:14.184495:30 INFO 43048 --- [ restartedMain] c.c.ormlearn.OrmlearnApplication : Country: Country [code-US, name-United States of America]
2025-07-05T10:37:14.185495:30 INFO 43048 --- [ restartedMain] c.c.ormlearn.OrmlearnApplication : End testGetAllCountries

```

## 5) Difference between JPA, Hibernate and Spring Data JPA

### Employee.java

package com.cognizant.ormlearn.model;

```

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

```

```
import jakarta.persistence.Table;
import jakarta.persistence.Column;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @Column(name = "id")
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "salary")
    private double salary;

    public Employee() {
    }

    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

```

@Override
public String toString() {
    return "Employee{id=" + id + ", name=" + name + ", salary=" + salary +
}";
}
}

```

### **EmployeeRepository.java**

```

package com.cognizant.ormlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.cognizant.ormlearn.model.Employee;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee,
Integer> {
}

```

### **EmployeeService.java**

```

package com.cognizant.ormlearn.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cognizant.ormlearn.model.Employee;
import com.cognizant.ormlearn.repository.EmployeeRepository;
import java.util.List;

@Service
public class EmployeeService {

    @Autowired
    private EmployeeRepository employeeRepository;

    @Transactional
    public void addEmployee(Employee employee) {
        employeeRepository.save(employee);
    }

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

```

### **OrmlearnApplication.java**

```

package com.cognizant.ormlearn;

```

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import com.cognizant.ormlearn.model.Employee;
import com.cognizant.ormlearn.service.EmployeeService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@SpringBootApplication
public class OrmlearnApplication {

    private static final Logger LOGGER =
LoggerFactory.getLogger(OrmlearnApplication.class);

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(OrmlearnApplication.class, args);
        EmployeeService employeeService =
context.getBean(EmployeeService.class);

        LOGGER.info("Inside main - Adding 10 employees");

        employeeService.addEmployee(new Employee(1, "Elon Musk", 150000));
        employeeService.addEmployee(new Employee(2, "Sundar Pichai",
140000));
        employeeService.addEmployee(new Employee(3, "Satya Nadella",
145000));
        employeeService.addEmployee(new Employee(4, "Tim Cook", 155000));
        employeeService.addEmployee(new Employee(5, "Mark Zuckerberg",
135000));
        employeeService.addEmployee(new Employee(6, "Jeff Bezos", 160000));
        employeeService.addEmployee(new Employee(7, "Sheryl Sandberg",
130000));
        employeeService.addEmployee(new Employee(8, "Susan Wojcicki",
128000));
        employeeService.addEmployee(new Employee(9, "Reed Hastings",
125000));
        employeeService.addEmployee(new Employee(10, "Ginni Rometty",
138000));

        LOGGER.info("10 employees added successfully");

        LOGGER.info("All Employees: {}", employeeService.getAllEmployees());
    }
}

```

**OrmlearnApplicationTests.java**

```

package com.cognizant.ormlearn;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class OrmlearnApplicationTests {

    @Test
    void contextLoads() {
    }

}

```

## **pom.xml**

```

<?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>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.cognizant</groupId>
  <artifactId>ormlearn</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ormlearn</name>
  <description>Demo project for Spring Data JPA and
Hibernate</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>

```

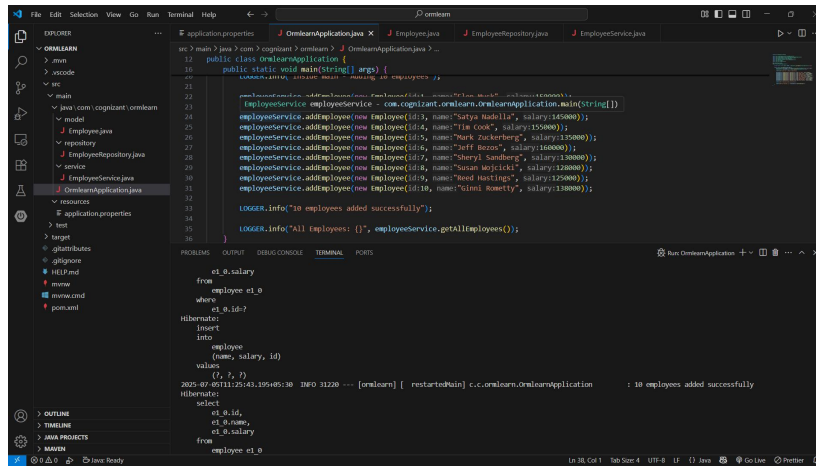


```
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

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

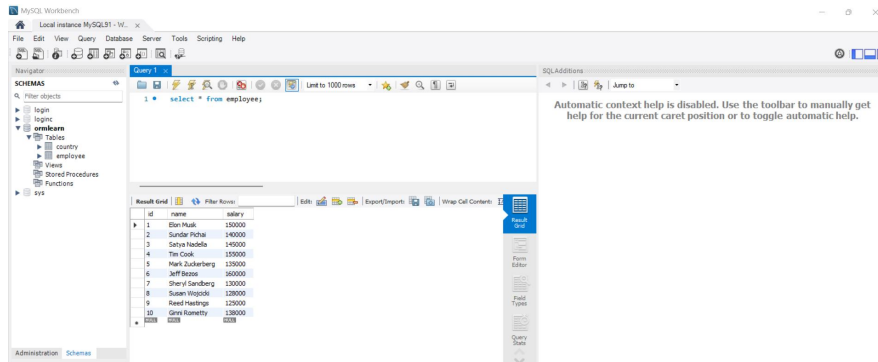
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

## Output :



The screenshot shows an IDE with a Java project named 'ormlearn'. The main class, 'OrmlearnApplication', is being executed. The code in the main method adds 10 employees to the database. The output console shows the following log messages:

```
2025-07-07 11:25:43.105495:30 INFO 31220 --- [ormlearn] [ restartedMain] c.c.ormlearn.OrmlearnApplication : 10 employees added successfully
Hibernate:
select
  employee o1_0
from
  employee o1_0
where
  o1_0.id=?
Hibernate:
insert
into
  employee
  (name, salary, id)
values
  (?, ?, ?)
2025-07-07 11:25:43.105495:30 INFO 31220 --- [ormlearn] [ restartedMain] c.c.ormlearn.OrmlearnApplication : 10 employees added successfully
Hibernate:
select
  employee o1_0,
  o1_0.name,
  o1_0.salary
from
  employee o1_0
```



The screenshot shows MySQL Workbench with a query executed: `select * from employees;`. The result is displayed in a table with 10 rows and 3 columns: id, name, and salary.

	id	name	salary
1	1	Earl Hask	120000
2	2	Sunder Pitha	140000
3	3	Satya Nadella	140000
4	4	Tim Cook	120000
5	5	Mark Zuckerberg	120000
6	6	Jeff Bezos	140000
7	7	Sheryl Sandberg	130000
8	8	Susan Wojcicki	120000
9	9	Reed Hastings	120000
10	10	Gina Rometty	130000
11	1038	GDSD	GDSD