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);
   }
}
```

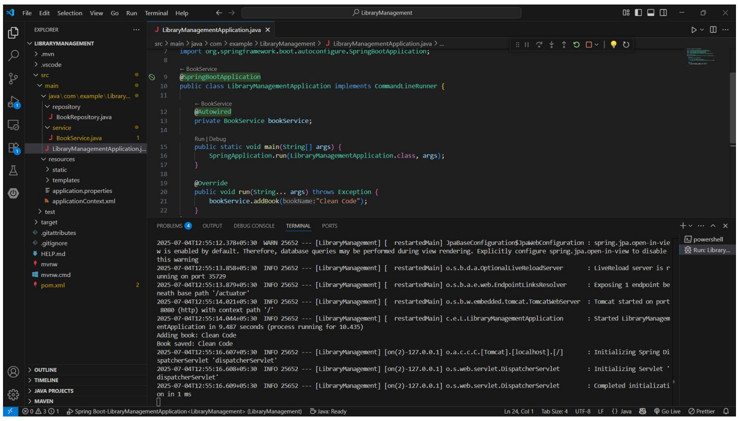
BookService.java

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

applicationContext.xml

```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">

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

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

Output



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

@SpringBootApplication
public class LibraryManagementApplication implements CommandLineRunner {

    @Autowired
    private BookService bookService;

    public static void main(String[] args) {
        SpringApplication.run(LibraryManagementApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        bookService.addBook(bookName:"Clean Code");
    }
```

```
2025-07-04T12:55:12.378+05:30  WARN 25652 --- [LibraryManagement] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-vie
w is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable
this warning
2025-07-04T12:55:13.858+05:30  INFO 25652 --- [LibraryManagement] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is r
unning on port 35729
2025-07-04T12:55:13.879+05:30  INFO 25652 --- [LibraryManagement] [  restartedMain] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1 endpoint be
neath base path '/actuator'
2025-07-04T12:55:14.021+05:30  INFO 25652 --- [LibraryManagement] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port
 8080 (http) with context path '/'
2025-07-04T12:55:14.044+05:30  INFO 25652 --- [LibraryManagement] [  restartedMain] c.e.L.LibraryManagementApplication       : Started LibraryManagem
entApplication in 9.487 seconds (process running for 10.435)
Adding book: Clean Code
Book saved: Clean Code
2025-07-04T12:55:16.607+05:30  INFO 25652 --- [LibraryManagement] [on(2)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Di
spatcherServlet 'dispatcherServlet'
2025-07-04T12:55:16.608+05:30  INFO 25652 --- [LibraryManagement] [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet '
dispatcherServlet'
2025-07-04T12:55:16.609+05:30  INFO 25652 --- [LibraryManagement] [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Completed initializati
on in 1 ms
```

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.");
   }
}
```

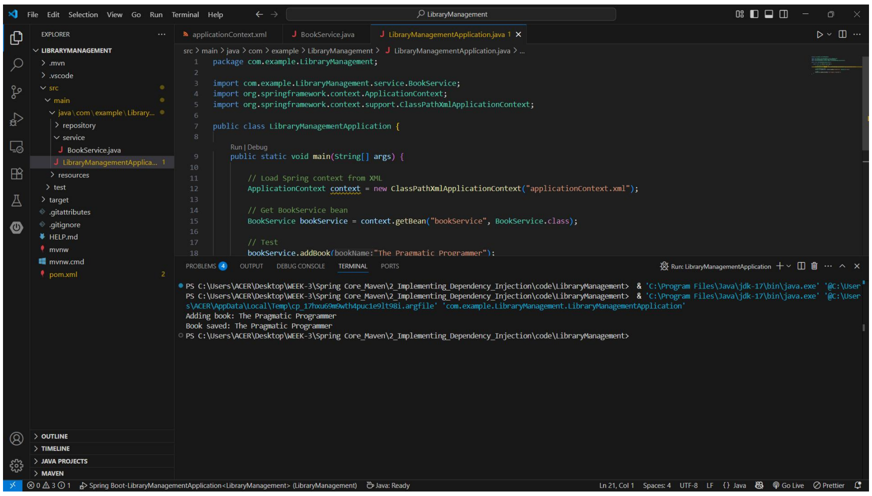
BookService.java

```
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 :



```
package com.example.LibraryManagement;

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

public class LibraryManagementApplication {

    public static void main(String[] args) {

        // Load Spring context from XML
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        // Get BookService bean
        BookService bookService = context.getBean("bookService", BookService.class);

        // Test
        bookService.addBook(bookName:"The Pragmatic Programmer");
```

```
PS C:\Users\ACER\Desktop\WEEK-3\Spring Core_Maven\2_Implementing_Dependency_Injection\code\LibraryManagement>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\User
PS C:\Users\ACER\Desktop\WEEK-3\Spring Core_Maven\2_Implementing_Dependency_Injection\code\LibraryManagement>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\User
s\ACER\AppData\Local\Temp\cp_17hxu69m9wth4puc1e9lt98i.argfile' 'com.example.LibraryManagement.LibraryManagementApplication'
Adding book: The Pragmatic Programmer
Book saved: The Pragmatic Programmer
PS C:\Users\ACER\Desktop\WEEK-3\Spring Core_Maven\2_Implementing_Dependency_Injection\code\LibraryManagement>
```

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.xl

```
<?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>
```

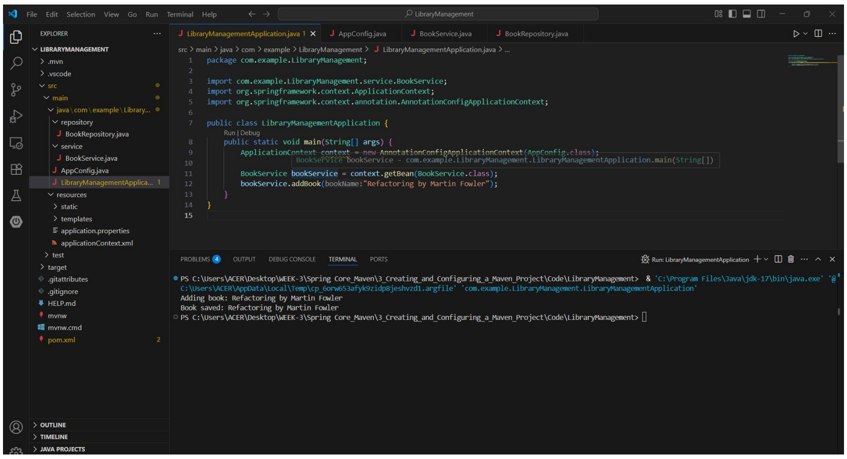
pom.xl

```
<?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>
```

```
                <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 :



```
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(bookName:"Refactoring by Martin Fowler");
    }
}
```

```
PS C:\Users\ACER\Desktop\WEEK-3\Spring Core_Maven\3_Creating_and_Configuring_a_Maven_Project\Code\LibraryManagement>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@
C:\Users\ACER\AppData\Local\Temp\cp_6orw653afyk9zidp8jeshvzd1.argfile' 'com.example.LibraryManagement.LibraryManagementApplication'
Adding book: Refactoring by Martin Fowler
Book saved: Refactoring by Martin Fowler
PS C:\Users\ACER\Desktop\WEEK-3\Spring Core_Maven\3_Creating_and_Configuring_a_Maven_Project\Code\LibraryManagement> 
```

1. spring-data-jpa-handson

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

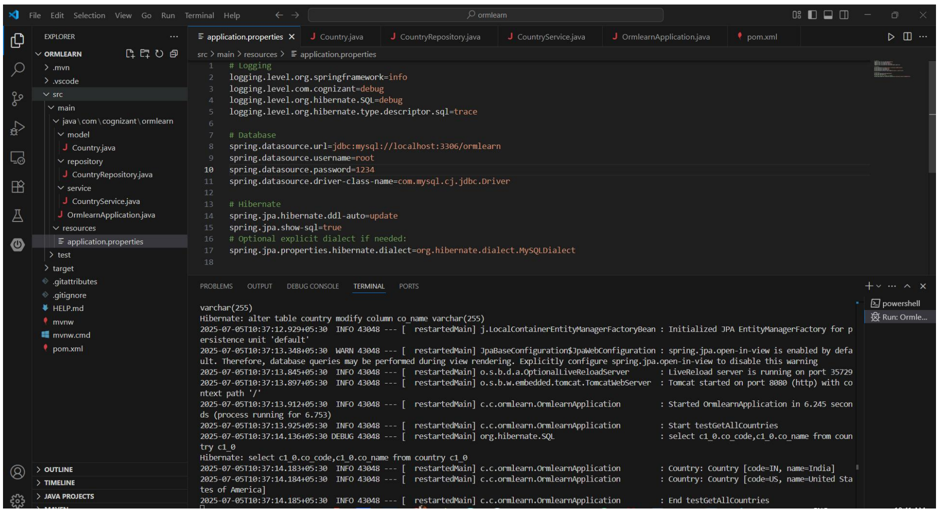
```
package com.cognizant.ormlearn;

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

@SpringBootTest
class OrmlearnApplicationTests {

        @Test
        void contextLoads() {
        }
}
```

Pom.xl

```
<?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 :



```
# Logging
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.type.descriptor.sql=trace

# Database
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
# Optional explicit dialect if needed:
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

```

```
varchar(255)
Hibernate: alter table country modify column co_name varchar(255)
2025-07-05T10:37:12.929+05:30  INFO 43048 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for p
ersistence unit 'default'
2025-07-05T10:37:13.348+05:30  WARN 43048 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by defa
ult. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-07-05T10:37:13.845+05:30  INFO 43048 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-05T10:37:13.897+05:30  INFO 43048 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with co
ntext path '/'
2025-07-05T10:37:13.912+05:30  INFO 43048 --- [  restartedMain] c.c.ormlearn.OrmlearnApplication         : Started OrmlearnApplication in 6.245 secon
ds (process running for 6.753)
2025-07-05T10:37:13.925+05:30  INFO 43048 --- [  restartedMain] c.c.ormlearn.OrmlearnApplication         : Start testGetAllCountries
2025-07-05T10:37:14.136+05:30 DEBUG 43048 --- [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from coun
try c1_0
Hibernate: select c1_0.co_code,c1_0.co_name from country c1_0
2025-07-05T10:37:14.183+05:30  INFO 43048 --- [  restartedMain] c.c.ormlearn.OrmlearnApplication         : Country: Country [code=IN, name=India]
2025-07-05T10:37:14.184+05:30  INFO 43048 --- [  restartedMain] c.c.ormlearn.OrmlearnApplication         : Country: Country [code=US, name=United Sta
tes of America]
2025-07-05T10:37:14.185+05:30  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

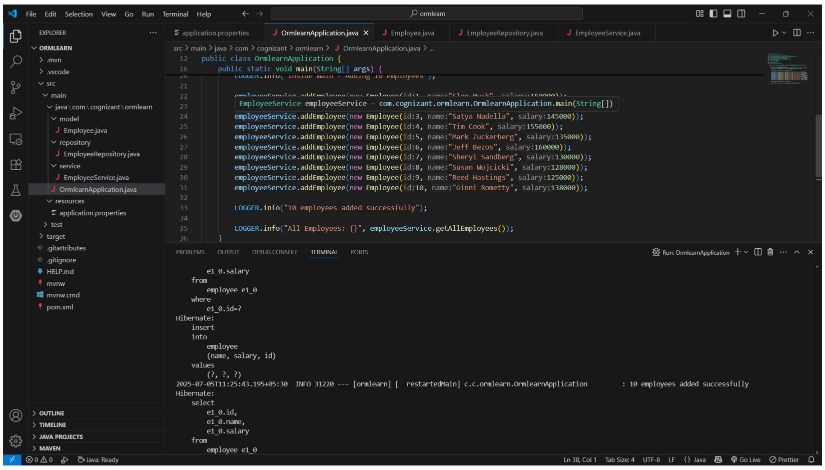
```
package com.cognizant.ormlearn;

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

@SpringBootTest
class OrmlearnApplicationTests {

	@Test
	void contextLoads() {
	}

}
```

pom.xl

```
<?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/>
```

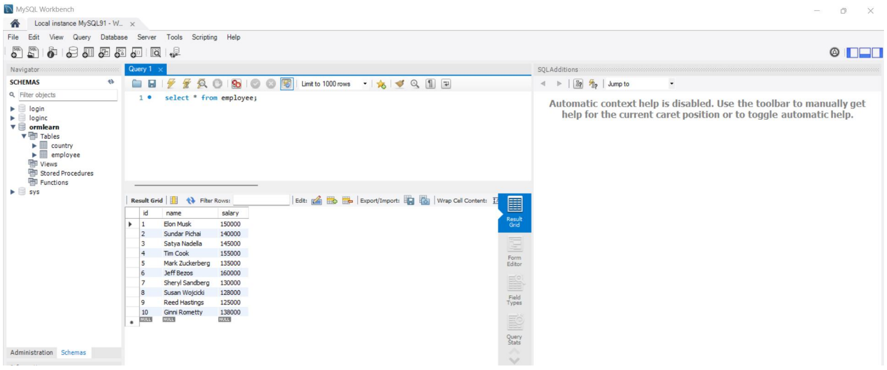
```xml
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <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 :



```java
public class OrmlearnApplication {
    public static void main(String[] args) {
        LOGGER.info("Inside main - Adding 10 employees");

        employeeService.addEmployee(new Employee(id:1, name:"Elon Musk", salary:150000));
        EmployeeService employeeService - com.cognizant.ormlearn.OrmlearnApplication.main(String[])
        employeeService.addEmployee(new Employee(id:3, name:"Satya Nadella", salary:145000));
        employeeService.addEmployee(new Employee(id:4, name:"Tim Cook", salary:155000));
        employeeService.addEmployee(new Employee(id:5, name:"Mark Zuckerberg", salary:135000));
        employeeService.addEmployee(new Employee(id:6, name:"Jeff Bezos", salary:160000));
        employeeService.addEmployee(new Employee(id:7, name:"Sheryl Sandberg", salary:130000));
        employeeService.addEmployee(new Employee(id:8, name:"Susan Wojcicki", salary:128000));
        employeeService.addEmployee(new Employee(id:9, name:"Reed Hastings", salary:125000));
        employeeService.addEmployee(new Employee(id:10, name:"Ginni Rometty", salary:138000));

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

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

```
        e1_0.salary
    from
        employee e1_0
    where
        e1_0.id=?
Hibernate:
    insert
    into
        employee
        (name, salary, id)
    values
        (?, ?, ?)
2025-07-05T11:25:43.195+05:30  INFO 31220 --- [ormlearn] [  restartedMain] c.c.ormlearn.OrmlearnApplication         : 10 employees added successfully
Hibernate:
    select
        e1_0.id,
        e1_0.name,
        e1_0.salary
    from
        employee e1_0
```



```sql
select * from employee;
```

id	name	salary
1	Elon Musk	150000
2	Sundar Pichai	140000
3	Satya Nadella	145000
4	Tim Cook	155000
5	Mark Zuckerberg	135000
6	Jeff Bezos	160000
7	Sheryl Sandberg	130000
8	Susan Wojcicki	128000
9	Reed Hastings	125000
10	Ginni Rometty	138000
NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.