**R Tejashree**  
**Superset ID: 6419704**

**WEEK 4 – Spring REST using Spring Boot 3**

* Part A: Java Database Connectivity (JDBC)
* Part B: Hibernate ORM Basics

**Objectives** :

* Demonstrate creation of Spring Boot Application
  + Spring initializr, https://start.spring.io, @SpringBootApplication, SpringApplication.run()
    - Ref - https://start.spring.io
* Explain the need and benefits of Spring Boot
  + Makes Java development easy, avoids tedious development steps, reduces development time, avoids writing boilerplate code, provides embedded tomcat server, avoid XML configuration
    - Ref - https://www.journaldev.com/7969/spring-boot-tutorial
* Demonstrate loading bean from spring configuration file
  + Spring configuration xml, spring xml schema spring-beans.xsd, <bean>, id, class, <constructor-arg>, <property>, name, value, ClassPathXmlApplicationContext, ApplicationContext, context.getBean(), singleton scope, prototype scope
    - Ref - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html
    - IoC Container - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans
    - Scopes - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-factory-scopes
    - Constructor Injection - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-constructor-injection
    - Setter method injection - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-setter-injection
* Demonstrate inclusion of logging in Spring Boot Application
  + application.properties, logging.level, logging.pattern, server.port, LoggerFactory, Logger, log levels (trace, debug, info, warn, error)
    - Ref - <https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-logging.html>

**Mandatory Hands-on:**

**PART A: JDBC Module**

**Step 1: Open Eclipse**

* Launch Eclipse IDE.
* Ensure JDK is properly set up.

**Step 2: Create Java Project**

* Go to **File > New > Java Project**
* Name it: jdbc-demo
* Click **Finish**

**Step 3: Add JDBC Driver**

* Download the MySQL Connector/J .jar file (or use existing one).
* Right-click on the project > **Build Path > Configure Build Path**
* Under **Libraries** tab, click **Add External JARs**
* Select mysql-connector-java-x.x.xx.jar and click **Apply and Close**

**Step 4: Create Package and Class**

* Right-click src > **New > Package**
* Name: com.cognizant.jdbc
* Right-click the package > **New > Class**
* Name: JDBCExample
* Check **public static void main**

**Step 5: Write JDBC Code**

package com.cognizant.jdbc;

import java.sql.\*;

public class JDBCExample {

public static void main(String[] args) {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/demo", "root", "root"

);

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM employee");

while (rs.next())

System.out.println(rs.getInt(1) + " " + rs.getString(2));

con.close();

} catch (Exception e) {

System.out.println(e);

}

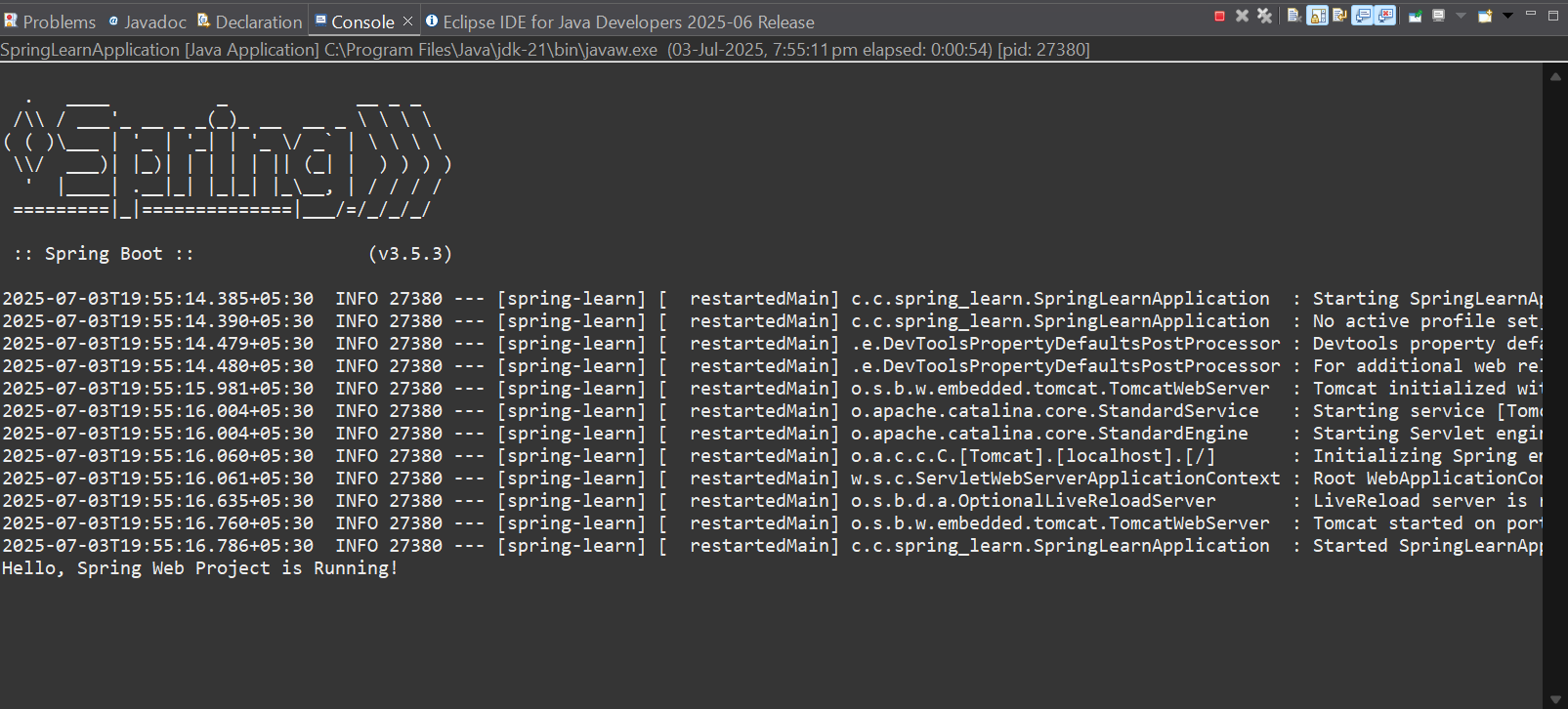
}

}

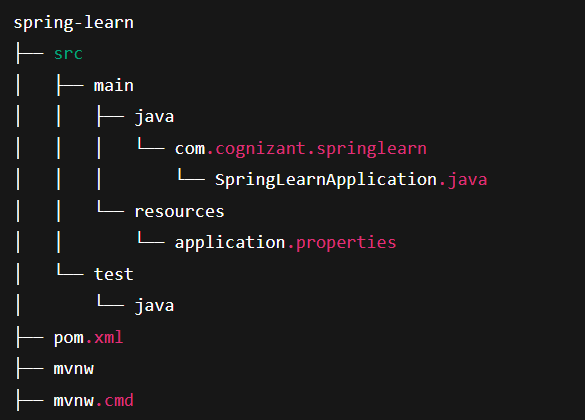
**Step 6: Run the Application**

* Right-click the file > **Run As > Java Application**
* Output will show employee data from the MySQL database

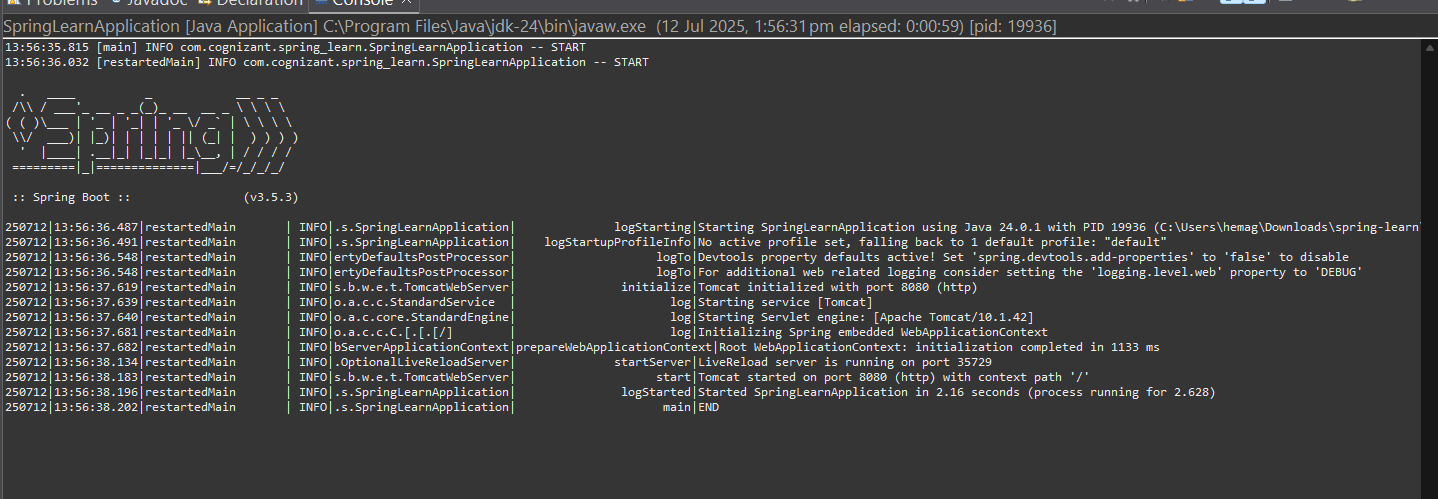
**Sample Output in Console:**



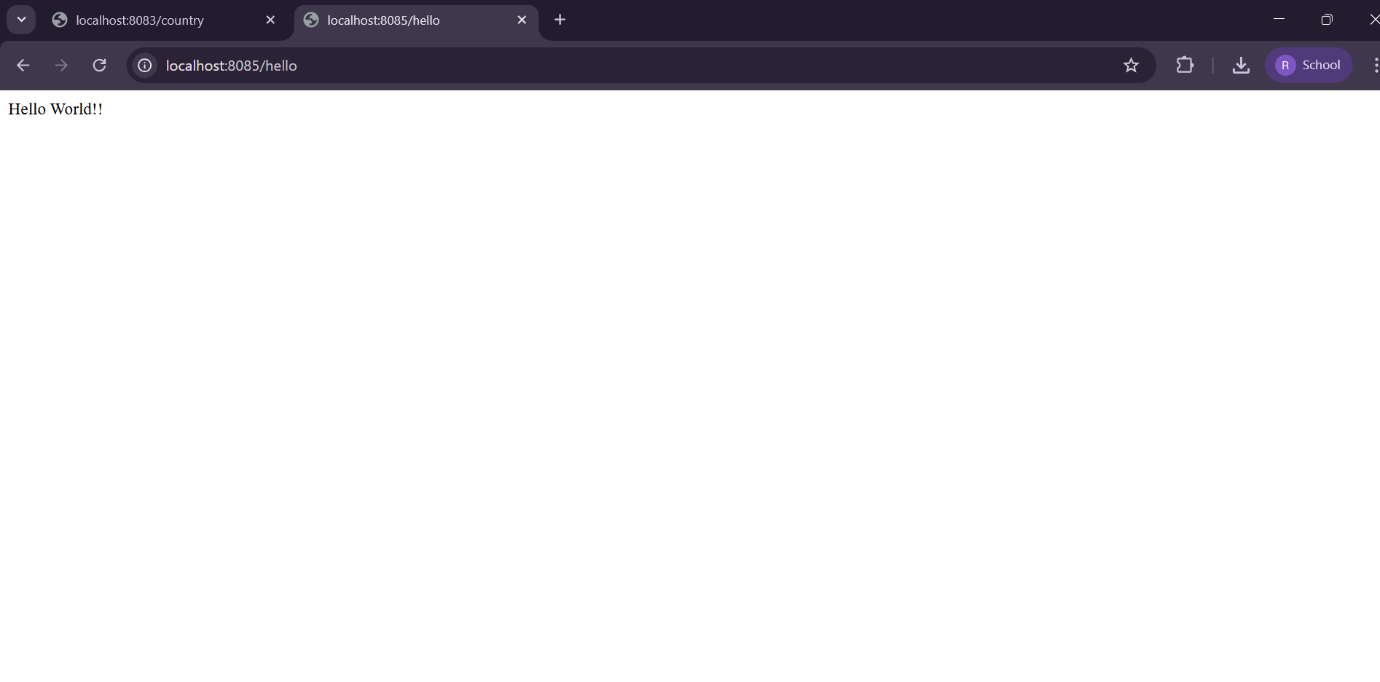
**📁 Project Structure Overview:**



**EX1 Create a Spring Web Project using Maven**



**EX1 Hello World**



**PART B: Hibernate Module**

**Step 1: Create Maven Project**

* Go to **File > New > Maven Project**
* Choose **Simple Project**, click **Next**
* Enter:
  + Group Id: com.cognizant.hibernate
  + Artifact Id: hibernate-demo
* Click **Finish**

**Step 2: Add Dependencies in pom.xml**

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.6.15.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.33</version>

</dependency>

</dependencies>

**Step 3: Create Model Class**

java

CopyEdit

package com.cognizant.hibernate.model;

import javax.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

private int id;

private String name;

// getters and setters

}

**Step 4: Create hibernate.cfg.xml**

Place this file in src/main/resources:

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/demo</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<mapping class="com.cognizant.hibernate.model.Employee"/>

</session-factory>

</hibernate-configuration>

**Step 5: Create Main Class**

package com.cognizant.hibernate;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import com.cognizant.hibernate.model.Employee;

public class App {

public static void main(String[] args) {

Configuration cfg = new Configuration().configure();

SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession();

Employee emp = new Employee();

emp.setId(1);

emp.setName("John");

session.beginTransaction();

session.save(emp);

session.getTransaction().commit();

session.close();

factory.close();

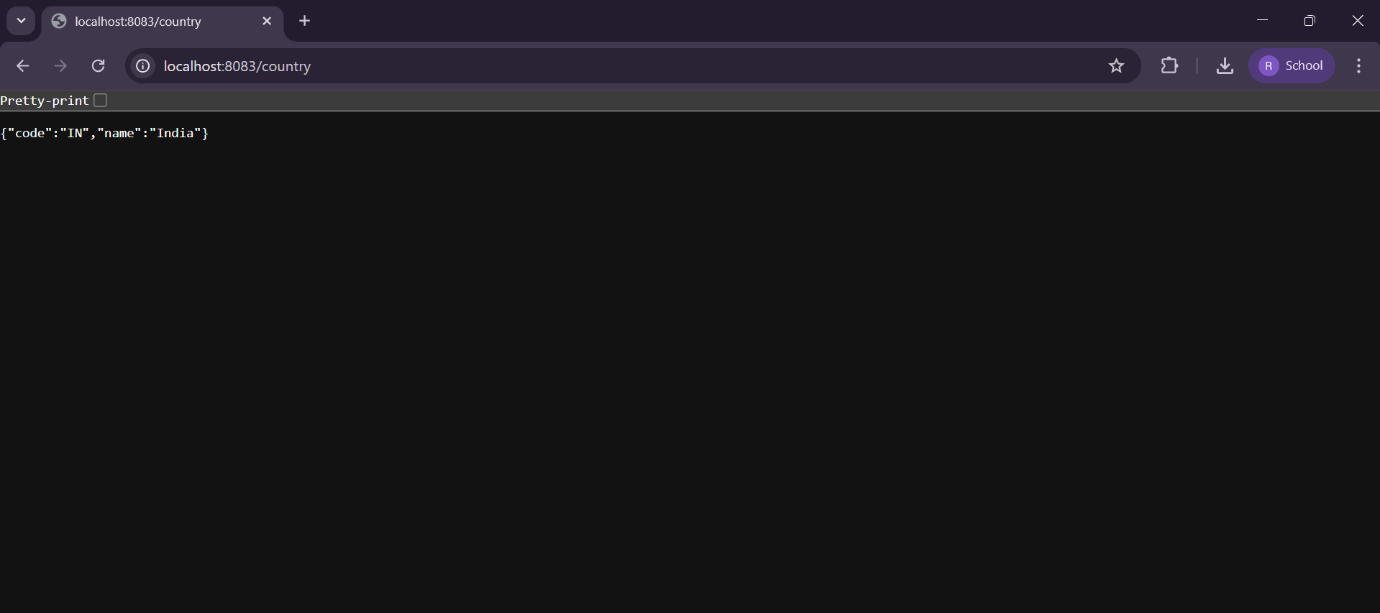
}

}

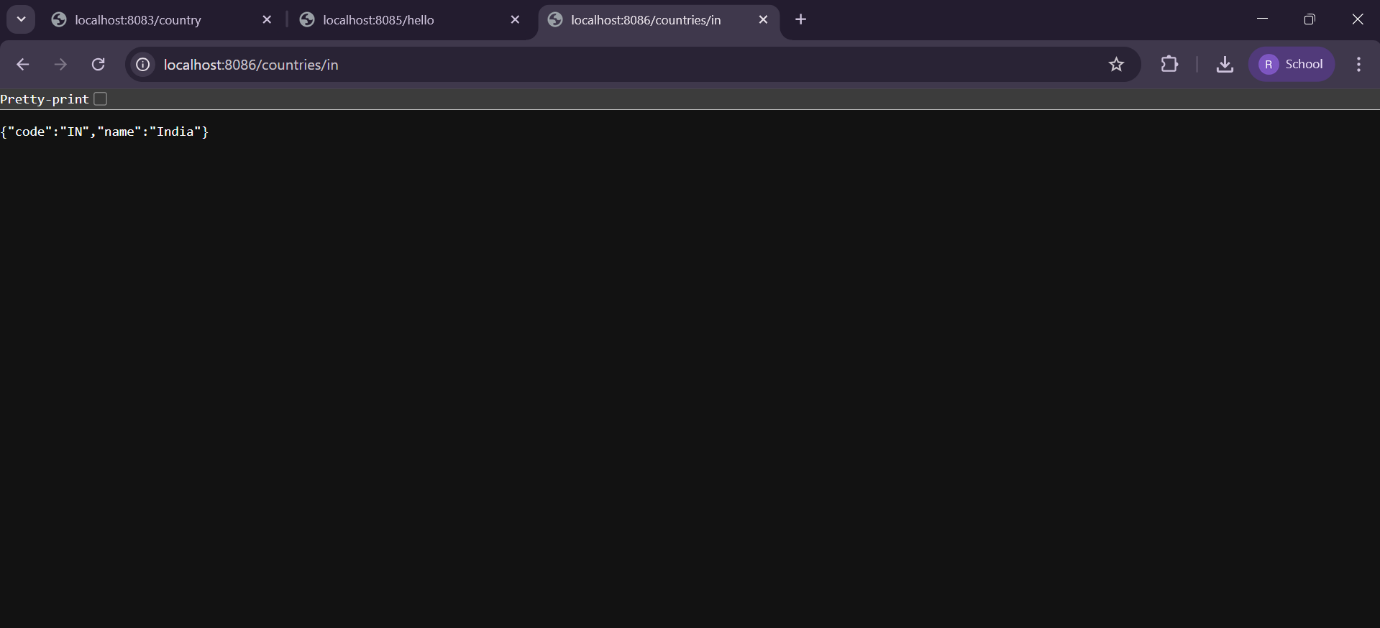
**Step 6: Run the Program**

* Right-click App.java > **Run As > Java Application**
* A new employee record should be inserted into the database

**EX2 REST - Country Web Service**



**EX3 REST - Get country based on country code**



**JWT Handson**

