**WEEK-3**

(Implemented in Eclipse )

**1) Hands on 1**

**Spring Data JPA - Quick Example**

ormlearnapplication.java

**package** com.cognizant.orm\_learn;

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

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

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

**import** java.util.List;

@SpringBootApplication

**public** **class** OrmLearnApplication **implements** CommandLineRunner{

@Autowired

**private** CountryRepository countryRepository;

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

SpringApplication.*run*(OrmLearnApplication.**class**, args);

}

@Override

**public** **void** run(String... args) **throws** Exception {

List<Country> countries = countryRepository.findAll();

countries.forEach(System.***out***::println);

}

}

Country.java

**package** com.cognizant.orm\_learn;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name = "country")

**public** **class** Country {

@Id

**private** String code;

**private** String name;

**private** **long** population;

// Getters and Setters

**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;

}

**public** **long** getPopulation() {

**return** population;

}

**public** **void** setPopulation(**long** population) {

**this**.population = population;

}

@Override

**public** String toString() {

**return** "Country [code=" + code + ", name=" + name + ", population=" + population + "]";

}

}

CountryRepository.java

**package** com.cognizant.orm\_learn;

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

**import** com.cognizant.orm\_learn.Country;

**public** **interface** CountryRepository **extends** JpaRepository<Country, String> {

}

applicationproperties

spring.application.name=orm-learn

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

# MySQL DB config

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=

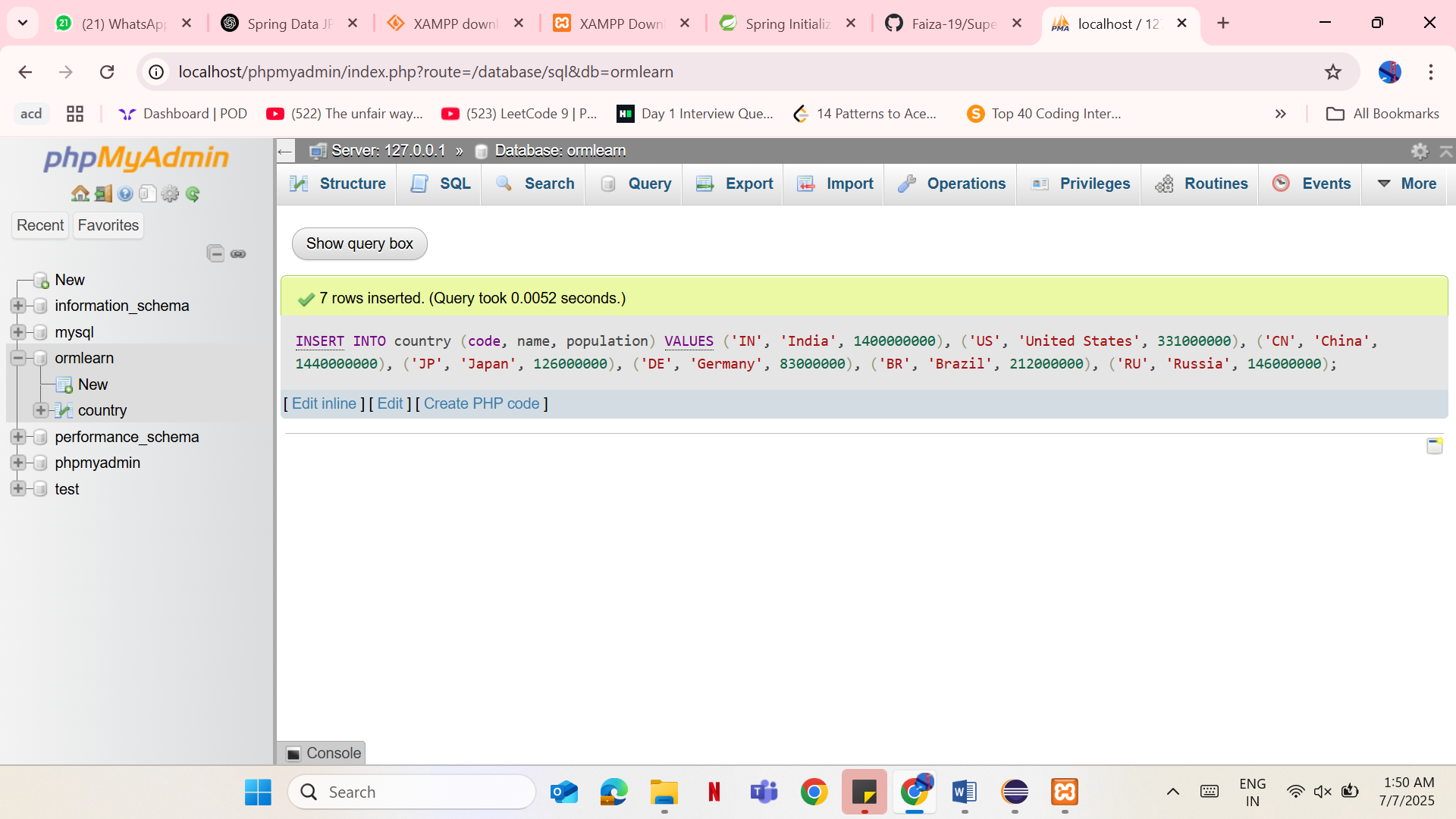
# Hibernate

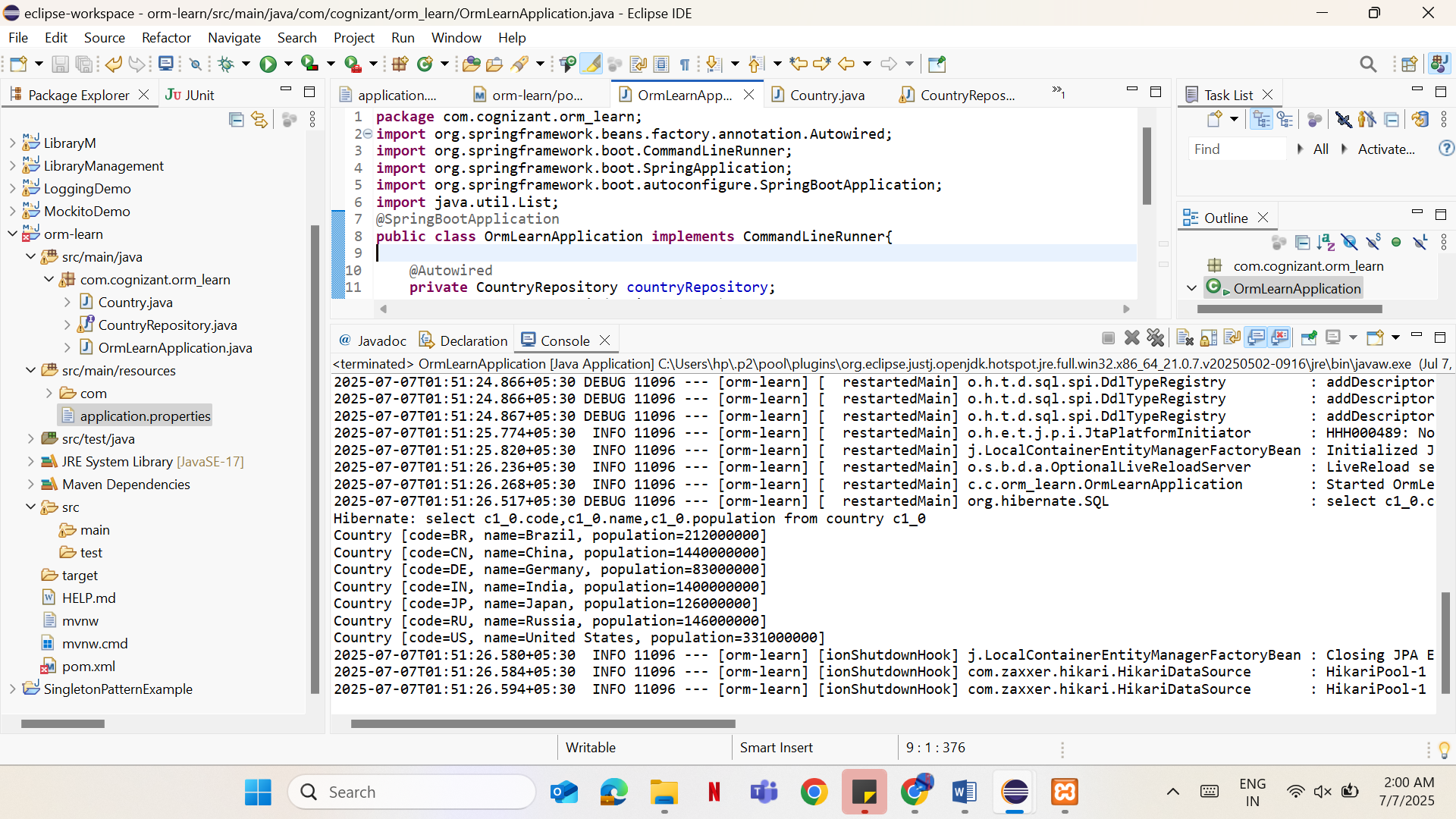
spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Output:**

****

****

**Hands on 4**

**Difference between JPA, Hibernate and Spring Data**

**1. What is JPA?**

• JPA (Java Persistence API) is a specification for persisting Java objects to relational databases.  
• It provides annotations and APIs to map Java classes to database tables.  
• JPA is not an implementation. It requires a provider like Hibernate to work.  
•Example Providers: Hibernate, EclipseLink, OpenJPA

**2. What is Hibernate?**

• Hibernate is an ORM (Object Relational Mapping) tool and the most popular JPA implementation.  
• It provides its own APIs and also fully implements the JPA specification.  
• Helps in managing database transactions, sessions, and mappings using Java classes.

**3. What is Spring Data JPA?**

• Spring Data JPA is not a JPA provider.  
• It is a Spring abstraction over JPA/Hibernate that removes boilerplate code.  
• Helps create JPA repositories without writing actual queries.  
• Comes with built-in methods like save(), findAll(), deleteById(), etc.  
• Works with Hibernate underneath (or other JPA providers).

**Comparison**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification (API only) | JPA Provider & ORM tool | Abstraction over JPA |
| Implementation | No | Yes (implements JPA) | No |
| Reduces Boilerplate? | Some | Moderate | Yes (very minimal code) |
| Configuration Needed | Yes | Yes | Minimal (uses Spring Boot magic) |
| Querying | JPQL, Criteria | JPQL, HQL | Derived querymethods, JPQL |
| Transaction Mgmt | Manual or via framework | Manual or via Spring | Handled by Spring (via annotations) |

**Hibernate vs Spring Data JPA — Code Snippet Comparison**

**Hibernate**

public Integer addEmployee(Employee employee){  
 Session session = factory.openSession();  
 Transaction tx = null;  
 Integer employeeID = null;  
  
 try {  
 tx = session.beginTransaction();  
 employeeID = (Integer) session.save(employee);  
 tx.commit();  
 } catch (HibernateException e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employeeID;  
}

**Spring Data JPA**

@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}  
  
@Service  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Transactional  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
 }  
}