WEEK 3 :

Skill : Spring core and maven

Exercise 1: configuring a basic spring application :

Pom.xml  
<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-core-app</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

</project>

HelloService.java :

package com.example;

public class HelloService {

public void sayHello() {

System.out.println("Hello from Spring Core!");

}

}

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">

<bean id="helloService" class="com.example.HelloService"/>

</beans>

App.java :

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

HelloService helloService = (HelloService) context.getBean("helloService");

helloService.sayHello();

}

}

OUTPUT :

Hello from Spring Core!

SKILL : Spring core and maven

Exercise 2: Implementing dependency injection :

Pom.xml :

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-di-app</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

</project>

Student.java :

package com.example;

public class Student {

private String name;

public void setName(String name) {

this.name = name;

}

public void display() {

System.out.println("Student name: " + name);

}

}

StudentService.java :

package com.example;

public class StudentService {

private Student student;

// Setter for dependency injection

public void setStudent(Student student) {

this.student = student;

}

public void showStudent() {

student.display();

}

}

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">

<!-- Define Student bean -->

<bean id="student" class="com.example.Student">

<property name="name" value="Harshitha"/>

</bean>

<!-- Define StudentService bean and inject student -->

<bean id="studentService" class="com.example.StudentService">

<property name="student" ref="student"/>

</bean>

</beans>

App.java :

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

StudentService service = (StudentService) context.getBean("studentService");

service.showStudent();

}

}

OUTPUT :

Student name: Harshitha

Skill : Spring core and Maven

Exercise 4 : Creating and configuring a maven project

Pom.xml :

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-maven-app</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven plugin to run Java apps -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.example.App</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

MessageService.java :

package com.example;

public class MessageService {

public void printMessage() {

System.out.println("Spring Core Project with Maven is working!");

}

}

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">

<bean id="messageService" class="com.example.MessageService"/>

</beans>

App.java :

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

MessageService service = (MessageService) context.getBean("messageService");

service.printMessage();

}

}

OUTPUT :

Spring Core Project with Maven is working!

Skill : Spring data JPA with Spring boot, hibernate

Exercise : Spring data JPA - Quick example :

Pom.xml :

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-data-jpa-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<parent>

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

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

<version>2.7.15</version>

</parent>

<dependencies>

<!-- Spring Boot + JPA -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 in-memory database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

</project>

User.java :

package com.example.demo.model;

import javax.persistence.\*;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

// Constructors

public User() {}

public User(String name, String email) {

this.name = name;

this.email = email;

}

// toString()

@Override

public String toString() {

return "User{id=" + id + ", name='" + name + "', email='" + email + "'}";

}

// Getters & Setters

public Long getId() { return id; }

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

public String getName() { return name; }

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

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

}

UserRepository.java :

package com.example.demo.repository;

import com.example.demo.model.User;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

DataLoader.java :

package com.example.demo.runner;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class DataLoader implements CommandLineRunner {

private final UserRepository userRepository;

public DataLoader(UserRepository userRepository) {

this.userRepository = userRepository;

}

@Override

public void run(String... args) {

User user = new User("John", "john@example.com");

userRepository.save(user);

System.out.println("User saved: " + user.getName());

System.out.println("All users in DB:");

userRepository.findAll().forEach(System.out::println);

}

}

DemoApplication.java :

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

application.properties :

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=create

OUTPUT :

User saved: John

All users in DB:

User{id=1, name='John', email='john@example.com'}

Skill : Spring data JPA with Spring boot, hibernate

Exercise : Differentiate between JPA, Hibernate and Spring Data JPA :

// Main.java

import javax.persistence.\*;

public class Main {

public static void main(String[] args) {

EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-pu");

EntityManager em = emf.createEntityManager();

User user = new User("JPA\_User", "jpa@example.com");

em.getTransaction().begin();

em.persist(user);

em.getTransaction().commit();

System.out.println("User saved using plain JPA!");

}

}

persistence.xml :

<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">

<persistence-unit name="my-pu">

<class>com.example.User</class>

<properties>

<property name="javax.persistence.jdbc.url" value="jdbc:h2:mem:test"/>

<property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>

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

</properties>

</persistence-unit>

</persistence>

HibernateMain.java :

// HibernateMain.java

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateMain {

public static void main(String[] args) {

Configuration cfg = new Configuration().configure().addAnnotatedClass(User.class);

SessionFactory sessionFactory = cfg.buildSessionFactory();

Session session = sessionFactory.openSession();

User user = new User("Hibernate\_User", "hib@example.com");

session.beginTransaction();

session.save(user);

session.getTransaction().commit();

System.out.println("User saved using Hibernate!");

}

}

hibernate.cfg.xml :

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.url">jdbc:h2:mem:test</property>

<property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>

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

</session-factory>

</hibernate-configuration>

// UserRepository.java

public interface UserRepository extends JpaRepository<User, Long> {}

// DataLoader.java

@Component

public class DataLoader implements CommandLineRunner {

private final UserRepository repo;

public DataLoader(UserRepository repo) {

this.repo = repo;

}

public void run(String... args) {

repo.save(new User("SpringData\_User", "spring@example.com"));

System.out.println("User saved using Spring Data JPA!");

}

}

OUTPUT :

User saved using plain JPA!

User saved using Hibernate!

User saved using Spring Data JPA!

...

Hibernate:

insert

into user (email, name, id)

values (?, ?, ?)

User saved using Spring Data JPA!