**WEEK 03**

**1.Configuring a Basic Spring Application**

**Project Structure**

spring-core-demo/

├── src/

│ └── main/

│ ├── java/

│ │ └── com/example/springcore/

│ │ ├── App.java

│ │ └── HelloService.java

│ └── resources/

│ └── applicationContext.xml

├── pom.xml

**pom.xml – Maven Configuration**

<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-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version> <!-- or latest 5.x version -->

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**HelloService.java – A simple Spring Bean**

package com.example.springcore;

public class HelloService {

private String message;

public void setMessage(String message) {

this.message = message;

}

public void sayHello() {

System.out.println("Message from HelloService: " + message);

}

}

**applicationContext.xml – Spring Configuration (in src/main/resources)**

<?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.springcore.HelloService">

<property name="message" value="Welcome to Spring Core!" />

</bean>

</beans>

**App.java – The Main Class**

package com.example.springcore;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContextcontext=newClassPathXmlApplicationContext("applicationContext.xml");

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

service.sayHello();

}

}

**Running the Application**

mvn clean package

mvn exec:java -Dexec.mainClass="com.example.springcore.App"

**2. Implementing** **Dependency Injection**

**Project Structure**

spring-di-demo/

├── src/

│ └── main/

│ ├── java/

│ │ └── com/example/di/

│ │ ├── App.java

│ │ ├── TextEditor.java

│ │ └── SpellChecker.java

│ └── resources/

│ └── applicationContext.xml

├── pom.xml

**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-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

</project>

**SpellChecker.java**

package com.example.di;

public class SpellChecker {

public void checkSpelling() {

System.out.println("Spell checking performed.");

}

}

**TextEditor.java (Using Constructor Injection)**

package com.example.di;

public class TextEditor {

private SpellChecker spellChecker;

public TextEditor(SpellChecker spellChecker) {

this.spellChecker = spellChecker;

}

public void spellCheck() {

spellChecker.checkSpelling();

}

}

**Spring XML Configuration (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="spellChecker" class="com.example.di.SpellChecker"/>

<bean id="textEditor" class="com.example.di.TextEditor">

<constructor-arg ref="spellChecker"/>

</bean>

</beans>

**Main Class: App.java**

package com.example.di;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContextcontext=newClassPathXmlApplicationContext("applicationContext.xml");

TextEditor editor = (TextEditor) context.getBean("textEditor");

editor.spellCheck();

}

}

**Run the Application**

Build the project:

mvn clean compile

Run it using exec plugin:

mvn exec:java -Dexec.mainClass="com.example.di.App"

**3. Creating and Configuring a Maven project**

**Create Maven Project**

mvn archetype:generate -DgroupId=com.example.springcore \

-DartifactId=springcore-demo \

-DarchetypeArtifactId=maven-archetype-quickstart \

-DinteractiveMode=false

**Output Directory:**

springcore-demo/

├── pom.xml

└── src/

└── main/

├── java/

│ └── com/example/springcore/

│ ├── App.java

│ ├── MessageService.java

│ └── MessagePrinter.java

└── resources/

└── applicationContext.xml

**pom.xml Configuration**

<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>springcore-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

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

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

<version>3.1.0</version>

<configuration>

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

</configuration>

</plugin>

</plugins>

</build>

</project>

**Java Classes**

**MessageService.java**

package com.example.springcore;

public class MessageService {

public String getMessage() {

return "Hello from Spring Core with Maven!";

}

}

**MessagePrinter.java**

package com.example.springcore;

public class MessagePrinter {

private MessageService service;

public void setService(MessageService service) {

this.service = service;

}

public void printMessage() {

System.out.println(service.getMessage());

}

}

**App.java**

package com.example.springcore;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContextcontext=newClassPathXmlApplicationContext("applicationContext.xml");

MessagePrinter printer = (MessagePrinter) context.getBean("messagePrinter");

printer.printMessage();

}

}

**XML Configuration (src/main/resources/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.springcore.MessageService" />

<bean id="messagePrinter" class="com.example.springcore.MessagePrinter">

<property name="service" ref="messageService" />

</bean>

</beans>

**Build and Run**

**Compile the project:**

mvn clean compile

**Run using exec plugin:**

mvn exec:java

**4. Spring Data JPA – Quick Example**

**Project Structure**

springboot-jpa-demo/

├── src/

│ └── main/

│ ├── java/com/example/demo/

│ │ ├── DemoApplication.java

│ │ ├── entity/Employee.java

│ │ ├── repository/EmployeeRepository.java

│ │ └── runner/DataLoader.java

│ └── resources/

│ └── application.properties

├── pom.xml

**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>springboot-jpa-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.5</version>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Entity: Employee.java**

package com.example.demo.entity;

import jakarta.persistence.\*;

public class Employee {

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String role;

public Employee() {}

public Employee(String name, String role) {

this.name = name;

this.role = role;

}

public Long getId() { return id; }

public String getName() { return name; }

public String getRole() { return role; }

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

public void setRole(String role) { this.role = role; }

public String toString() {

return "Employee{id=" + id + ", name='" + name + "', role='" + role + "'}";

}

}

**Repository: EmployeeRepository.java**

package com.example.demo.repository;

import com.example.demo.entity.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**Data Loader: DataLoader.java**

package com.example.demo.runner;

import com.example.demo.entity.Employee;

import com.example.demo.repository.EmployeeRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

public class DataLoader implements CommandLineRunner {

private final EmployeeRepository repo;

public DataLoader(EmployeeRepository repo) {

this.repo = repo;

}

public void run(String... args) {

repo.save(new Employee("Alice", "Developer"));

repo.save(new Employee("Bob", "Manager"));

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

}

}

**Main App: DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**Config: 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.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

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

**JPA (Java Persistence API)**

- It is a specification (a set of interfaces).

- Defines how Java objects are mapped to relational database tables.

- Common annotations: @Entity, @Table, @Id, @GeneratedValue, etc.

- Does not contain any implementation code.

- Needs a provider (like Hibernate) to work.

**Hibernate**

- It is the most popular implementation of the JPA specification.

- Provides actual logic for persisting Java objects.

- Adds many extra features beyond JPA (e.g., caching, batch processing).

- Uses SessionFactory, but in JPA context, integrates with EntityManager.

**Spring Data JPA**

- It is a Spring project that sits on top of JPA.

- Provides repository interfaces like CrudRepository, JpaRepository, etc.

- Eliminates boilerplate code (like DAO implementations).

- Automatically implements basic CRUD operations using method names.

- Works with any JPA provider (Hibernate is default in Spring Boot).

**How They Work Together in Spring Boot:**

- You include: spring-boot-starter-data-jpa

- Spring Boot:

→ Enables Spring Data JPA

→ Uses Hibernate as the JPA provider by default

→ Auto-configures DataSource, EntityManagerFactory, TransactionManager

**Maven Dependency:**

<dependency>

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

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

</dependency>