**WEEK – 4(Hands-on Exercises)**

1. **Create a Spring Web (Spring MVC) Project Using Maven**

**Step 1: Create a New Maven Project**

1. Open Eclipse IDE.
2. Go to File → New → Maven Project.
3. Check Create a simple project (skip archetype selection) → Click Next.
4. Fill in the details:
   * Group ID: com.example
   * Artifact Id: spring-web-demo
   * Packaging: war
5. Click Finish.

**Step 2: Update pom.xml with Spring Dependencies**

1. In Project Explorer, expand spring-web-demo.
2. Double-click pom.xml.
3. Go to the pom.xml tab (bottom of the editor).
4. Replace <dependencies> with:

<dependencies>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Servlet API -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<!-- JSTL -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

Save the file.

Right-click the project → Maven → Update Project → Check Force Update of Snapshots/Releases → Click OK.

**Step 3: Convert Project to Dynamic Web Project**

1. Right-click the project → **Properties**.
2. Go to **Project Facets**.
3. If it says "Convert to faceted form," click it.
4. Enable:
   * Java
   * Maven
   * Dynamic Web Module
5. Apply and OK.

**Step 4: Create web.xml Configuration**

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

**Step 5: Create Spring Configuration File**

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:mvc="http://www.springframework.org/schema/mvc"

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

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd">

<mvc:annotation-driven />

<context:component-scan base-package="com.example.controller" />

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/views/" />

<property name="suffix" value=".jsp" />

</bean>

</beans>

**Step 6: Create Controller Class**

package com.example.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class HomeController {

@RequestMapping("/")

public ModelAndView home() {

ModelAndView mv = new ModelAndView("home");

mv.addObject("message", "Hello from Spring MVC!");

return mv;

}

}

**Step 7: Create JSP View File**

<html>

<body>

<h2>${“Hiii”}</h2>

</body>

</html>

**Step 8: Deploy and Run on Tomcat**

**1.** Make sure Tomcat is configured in Eclipse.

2. Right-click the project → Run As → Run on Server.

3. Open a browser and visit:  
http://localhost:8080/spring-web-demo

**Output:**

Hiii

**2. Spring Core – Load Country from Spring Configuration XML**

**1. Create a Simple Java Class (Country.java)**

package com.example;

public class Country {

private String name;

private String capital;

// Getter and Setter

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCapital() {

return capital;

}

public void setCapital(String capital) {

this.capital = capital;

}

public void displayCountryInfo() {

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

System.out.println("Capital: " + capital);

}

}

**2. Create Spring Configuration File (spring-config.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="country" class="com.example.Country">

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

<property name="capital" value="New Delhi" />

</bean>

</beans>

**3. Create Main Application Class**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("spring-config.xml");

Country country = (Country) context.getBean("country");

country.displayCountryInfo();

}

}

**Output:**

Country: India

Capital: New Delhi

**3. Hello World RESTful Web Service**

**1. Create a New Spring Boot Project**

* Tool: Spring Initializer or Eclipse STS
* Dependencies:
  + Spring Web

**2. Controller Class**

package com.example.helloworld;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

@GetMapping("/hello")

public String sayHello() {

return "Hello, World!";

}

}

**3. Main Class**

package com.example.helloworld;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(HelloWorldApplication.class, args);

}

}

**4. Run and Test**

* Run HelloWorldApplication.
* Open browser → visit:  
  http://localhost:8080/hello

**Output:**

Hello, World!

**4. REST – Country Web Service**

**1. Create Country Class**

package com.example.countryservice;

public class Country {

private String name;

private String capital;

public Country() {}

public Country(String name, String capital) {

this.name = name;

this.capital = capital;

}

public String getName() {

return name;

}

public String getCapital() {

return capital;

}

public void setName(String name) {

this.name = name;

}

public void setCapital(String capital) {

this.capital = capital;

}

}

**2. Create CountryController Class**

package com.example.countryservice;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class CountryController {

@GetMapping("/country")

public Country getCountry() {

return new Country("India", "New Delhi");

}

}

**3. Main Application Class**

package com.example.countryservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CountryServiceApplication.class, args);

}

}

**4. Running and Testing**

* Run CountryServiceApplication as a Java application.
* Open Browser/Postman:

Click on http://localhost:8080/country

1. **Output:**

{

"name": "India",

"capital": "New Delhi"

}

**5. REST – Get Country Based on Country Code**

**Country Model Class**

package com.example.countrycodeservice;

public class Country {

private String code;

private String name;

private String capital;

public Country() {}

public Country(String code, String name, String capital) {

this.code = code;

this.name = name;

this.capital = capital;

}

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 String getCapital() { return capital; }

public void setCapital(String capital) { this.capital = capital; }

}

**CountryController Class with Country Code Logic**

package com.example.countrycodeservice;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@RestController

public class CountryController {

private static Map<String, Country> countryMap = new HashMap<>();

static {

countryMap.put("IN", new Country("IN", "India", "New Delhi"));

countryMap.put("US", new Country("US", "United States", "Washington D.C."));

countryMap.put("JP", new Country("JP", "Japan", "Tokyo"));

}

@GetMapping("/country/{code}")

public Country getCountryByCode(@PathVariable String code) {

return countryMap.getOrDefault(code.toUpperCase(), null);

}

}

**Main Application Class**

package com.example.countrycodeservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CountryServiceApplication.class, args);

}

}

**Testing the API**

* Run the application.
* Test URL examples:

http://localhost:8080/country/IN

http://localhost:8080/country/US

http://localhost:8080/country/JP

For /country/IN:

{

"code": "IN",

"name": "India",

"capital": "New Delhi"

}

**6. Create an Authentication Service That Returns JWT**

**Dependencies in pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

**JWT Utility Class**

package com.example.jwtapp.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "mysecretkey123";

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60)) // 1 hour

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**Authentication Controller**

package com.example.jwtapp.controller;

import com.example.jwtapp.util.JwtUtil;

import org.springframework.web.bind.annotation.\*;

@RestController

public class AuthController {

@PostMapping("/authenticate")

public String authenticate(@RequestParam String username, @RequestParam String password) {

if ("admin".equals(username) && "password".equals(password)) {

return JwtUtil.generateToken(username);

} else {

return "Invalid credentials";

}

}

}

When we POST to:

http://localhost:8080/authenticate?username=admin&password=password

* If the username/password is correct, we’ll receive a JWT token as the response.