**Assessment**

1. **Create a Spring Web Project using Maven**

**Step 1-Create Maven Project Structure**

mvn archetype:generate -DgroupId=com.example.demo -DartifactId=spring-web-demo -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

**Step 2- Add Spring Dependencies in (pom.xml)**

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Optional: Spring Boot Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<parent>

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

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

<version>3.2.2</version>

</parent>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

**Step 3-Create Main Application Class**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringWebDemoApplication {

public static void main(String[] args) {

SpringApplication.run(SpringWebDemoApplication.class, args);

}

}

**Step 4-Create a Simple Controller**

package com.example.demo.controller;

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

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

@RestController

public class HelloController {

@GetMapping("/hello")

public String hello() {

return "Hello, Spring Web with Maven!";

}

}

**Step 5- Build and Run the program**

mvn clean install

mvn spring-boot:run

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

**Step 1- Create Maven Project Dependencies**

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version> <!-- Or latest stable 5.x -->

</dependency>

</dependencies>

**Step 2- Define Country Class**

package com.example.demo;

public class Country {

private String countryName;

private String capital;

// Getters and Setters

public String getCountryName() {

return countryName;

}

public void setCountryName(String countryName) {

this.countryName = countryName;

}

public String getCapital() {

return capital;

}

public void setCapital(String capital) {

this.capital = capital;

}

@Override

public String toString() {

return "Country [countryName=" + countryName + ", capital=" + capital + "]";

}

}

**Step 3-Create application Context.xml Configuration File**

<?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.demo.Country">

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

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

</bean>

</beans>

**Step 4- Load and test using ApplicationContext**

package com.example.demo;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class CountryApp {

public static void main(String[] args) {

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

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

System.out.println(country);

}

}

1. Hello World RESTful Web Service

Step 1 – Project setup with Maven Dependencies

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Test (Optional for unit tests) -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<parent>

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

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

<version>3.2.2</version> <!-- Use latest stable -->

</parent>

Step 2 – Main Application 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);

}

}

Step 3 – Create a REST controller

package com.example.helloworld.controller;

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

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

@RestController

public class HelloWorldController {

@GetMapping("/hello")

public String helloWorld() {

return "Hello, World!";

}

}

Step 4 – How to run a Project

mvn spring-boot:run

Step 5 – Test your RESTful Web service

1. REST - Country Web Service

Step 1 – Add these dependencies in pom.xml

<dependencies>

<!-- Spring Boot Web Starter -->

<dependency>

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

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

</dependency>

<!-- Optional: Spring Boot Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<parent>

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

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

<version>3.2.2</version> <!-- Use latest stable -->

</parent>

Step 2 – Create Main Boot Spring Application

package com.example.countryapi;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryApiApplication {

public static void main(String[] args) {

SpringApplication.run(CountryApiApplication.class, args);

}

}

Step 3 –Create Country Model

package com.example.countryapi.model;

public class Country {

private String name;

private String capital;

private long population;

// Constructors

public Country() {}

public Country(String name, String capital, long population) {

this.name = name;

this.capital = capital;

this.population = population;

}

// Getters and Setters

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 long getPopulation() { return population; }

public void setPopulation(long population) { this.population = population; }

}

Step 4 – Create Country Controller

package com.example.countryapi.controller;

import com.example.countryapi.model.Country;

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

import java.util.\*;

@RestController

@RequestMapping("/api/countries")

public class CountryController {

private static final List<Country> countries = new ArrayList<>();

static {

countries.add(new Country("India", "New Delhi", 1390000000));

countries.add(new Country("USA", "Washington, D.C.", 331000000));

countries.add(new Country("Japan", "Tokyo", 125000000));

}

@GetMapping

public List<Country> getAllCountries() {

return countries;

}

@GetMapping("/{name}")

public Country getCountryByName(@PathVariable String name) {

return countries.stream()

.filter(c -> c.getName().equalsIgnoreCase(name))

.findFirst()

.orElse(null);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

countries.add(country);

return country;

}

}

Step 5 – Run Your App

mvn spring-boot:run

Step 6 –Test your End RESTpoints.

1. REST - Get country based on country code