Week 4: Spring REST using Spring Boot 3

**Exercise 1: Create a Spring Web Project using Maven**

# Scenario:

Your company needs to develop a web application using the Spring Boot framework. You need to create a new Spring Web project using Maven and understand the project structure and configuration.

# Procedure:

# Step 1: Create Spring Boot Project using Spring Initializr

# Go to https://start.spring.io/

# Change Group as "com.cognizant".

# Change Artifact Id as "spring-learn".

* Select Spring Boot DevTools and Spring Web.
* Create and download the project as zip.

# Step 2: Extract and Build the Project

# Extract the zip in root folder to Eclipse Workspace.

# Build the project using 'mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456' command in command line.

# Step 3: Import Project in Eclipse

# Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish".

# Include logs to verify if main() method of SpringLearnApplication.

# Step 4: Run the Application

* Run the SpringLearnApplication class.

# Implementation:

# pom.xml:

<?xml version="1.0" encoding="UTF-8"?>

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>24</java.version>

</properties>

<dependencies>

<!-- Spring Web for building REST APIs -->

<dependency>

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

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

</dependency>

<!-- Developer tools: Auto-reload, etc. -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- For unit and integration testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Plugin to package and run Spring Boot application -->

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

# SpringLearnApplication Class:

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

public static void main(String[] args) {

System.***out***.println("Spring App is started");

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

}

}

# Output:

# A screenshot of a computer program AI-generated content may be incorrect.

**Exercise 4: Spring Core - Load Country from Spring Configuration XML**

# Scenario:

An airline’s website is going to support booking on four countries. There will be a dropdown on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country. The country data needs to be stored in a Spring configuration file and read programmatically to display the details.

**Country Data:**

* US - United States
* DE - Germany
* IN - India
* JP - Japan

# Procedure:

**Step 1: Create Spring XML Configuration**

* Create a Spring XML configuration file named **country.xml** in the src/main/resources.
* Configure a bean tag for country with property and values.
* Pick any one country from the given list to configure.

**Step 2: Create Country Class**

* Create a Country class with instance variables for code and name.
* Implement empty parameter constructor with debug log message "Inside Country Constructor."
* Generate getters and setters with debug logs in each method.
* Generate toString() method.

**Step 3: Create Display Method**

* Create a method **displayCountry()** in SpringLearnApplication.java.
* Use ClassPathXmlApplicationContext to read the country bean from spring configuration file.
* Display the country details using context.getBean("beanId", Country.class).

**Step 4: Execute and Test**

* Invoke displayCountry() method in main() method of SpringLearnApplication.java.
* Execute main() method and check the logs to verify constructor and method invocations.

# Implementation:

**country.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.cognizant.spring\_learn.Country">

<property name="code" value="IN" />

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

</bean>

</beans>

**Country Class:**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

private String code;

private String name;

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(Country.class);

public Country() {

***LOGGER***.debug("Inside Country Constructor.");

}

public String getCode() {

***LOGGER***.debug("Inside getCode.");

return code;

}

public void setCode(String code) {

***LOGGER***.debug("Inside setCode.");

this.code = code;

}

public String getName() {

***LOGGER***.debug("Inside getName.");

return name;

}

public void setName(String name) {

***LOGGER***.debug("Inside setName.");

this.name = name;

}

*@Override*

public String toString() {

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

}

}

**SpringLearnApplication Class:**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(SpringLearnApplication.class);

public static void main(String[] args) {

***LOGGER***.info("START");

*displayCountry*();

***LOGGER***.info("END");

}

public static void displayCountry() {

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

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

***LOGGER***.debug("Country : {}", country.toString());

}

}

# Output:

# A screenshot of a computer program AI-generated content may be incorrect.

**Exercise: Hello World RESTful Web Service**

# Scenario:

Your company needs to develop a RESTful web service using Spring Boot framework. You need to create a simple REST service that returns "Hello World!!" message when accessed via HTTP GET request. This will help us understand the basics of creating REST endpoints using Spring Web Framework.

# Procedure:

**Step 1: Configure Application Properties**

* Create **application.properties** file in src/main/resources.
* Set the application name and server port to 8083.

**Step 2: Create REST Controller**

* Create a controller package under **com.cognizant.spring\_learn.**
* Create **HelloController** class with @RestController annotation.
* Implement **sayHello**() method with @GetMapping("/hello") annotation.
* Include start and end log messages in the method.

**Step 3: Create the Main Application Class**

* Create the main application class with a simple **SpringApplication run().**

**Step 4: Test the Application**

* Run the SpringLearnApplication class.
* Test the URL <http://localhost:8083/hello> in a browser or in Postman.

# Implementation:

**application.properties:**

spring.application.name=spring-learn

server.port=8083

# HelloController.java:

package com.cognizant.spring\_learn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

*@RestController*

public class HelloController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(HelloController.class);

*@GetMapping*("/hello")

public String sayHello() {

***LOGGER***.info("START - sayHello()");

String message = "Hello, I'm Vivek. Nice to meet you!";

***LOGGER***.info("END - sayHello()");

return message;

}

}

# SpringLearnApplication.java:

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

# A white screen with black text AI-generated content may be incorrect.Output:

**Exercise: REST Country Web Service**

# Scenario:

Your company is building a RESTful service using the Spring Boot framework. This service will return country details for **India** when a user accesses a specific HTTP endpoint. The data is to be loaded from a Spring XML configuration file (country.xml).

**Procedure:**

**Step 1: Configure Application Properties**

* Create **application.properties** file in src/main/resources.
* Set the application name and server port to 8083.

**Step 2: Define Bean & Create a Model**

* Create **country.xml** in src/main/resources.
* Add India bean with id="country".
* Create Country class with code, name, getters/setters.

**Step 3: Create REST Controller**

* Create a package under **com.cognizant.spring\_learn.**
* Create **CountryController** with **@RestController**
* Add method **getCountry**(@PathVariable String code) and map it to @GetMapping("/countries/{code}")
* Call **countryService.getCountry(code)** and return result.

**Step 4: Create the Main Application Class and Test it**

* Run the SpringLearnApplication class and test the URL http://localhost:8083/country in a browser or in Postman.

# Implementation:

**country.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.cognizant.spring\_learn.Country">

<property name="code" value="IN"/>

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

</bean>

</beans>

**Country Class:**

package com.cognizant.spring\_learn;

public class Country {

private String code;

private String name;

public Country() {

}

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;

}

}

**CountryController Class:**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

*@*RestController

public class CountryController {

private static final Logger LOGGER = LoggerFactory.getLogger(CountryController.class);

*@*RequestMapping("/country")

public Country getCountryIndia() {

LOGGER.info("START - getCountryIndia()");

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

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

LOGGER.info("END - getCountryIndia()");

return country;

}

}

**SpringLearnApplication Class:**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise: REST (Get country based on country code)**

# Scenario:

Write a REST service that returns a specific country based on country code. The country code should be case insensitive.

**Procedure:**

**Step 1: Configure Application Properties**

* Create **application.properties** file in src/main/resources.
* Set the application name and server port to 8083.

**Step 2: Define Bean & Create a Model**

* Create **country.xml** in src/main/resources.
* Add India bean with id="country".
* Create Country class with code, name, getters/setters.

**Step 3: Create REST Controller**

* Create a controller package under **com.cognizant.spring\_learn.**
* Create **CountryController** with **@RestController**
* Add method **getCountry**(@PathVariable String code) and map it to @GetMapping("/countries/{code}")
* Call **countryService.getCountry(code)** and return result.

**Step 4: Create REST Service**

* Create a service package under **com.cognizant.spring\_learn.**
* Load countryList using **ClassPathXmlApplicationContext.**
* In **getCountry(String code)**, return the matching country (case-insensitive).

**Step 5: Create the Main Application Class and Test it**

* Run the SpringLearnApplication class and test the URL http://localhost:8083/country in a browser or in Postman.

# Implementation:

**country.xml:**

<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.cognizant.spring\_learn.Country">

<property name="code" value="IN" />

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

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="country"/>

<bean class="com.cognizant.spring\_learn.Country">

<property name="code" value="US" />

<property name="name" value="United States" />

</bean>

<bean class="com.cognizant.spring\_learn.Country">

<property name="code" value="DE" />

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

</bean>

<bean class="com.cognizant.spring\_learn.Country">

<property name="code" value="JP" />

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**Country Class:**

package com.cognizant.spring\_learn;

public class Country {

private String code;

private String name;

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; }

*@Override*

public String toString() {

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

}

}

**CountryService Class:**

package com.cognizant.spring\_learn.service;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.cognizant.spring\_learn.Country;

*@Service*

public class CountryService {

public Country getCountry(String code) {

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

List<Country> countryList = context.getBean("countryList", List.class);

return countryList.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(null);

}

}

**CountryController Class:**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

import com.cognizant.spring\_learn.service.CountryService;

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

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

*@RestController*

public class CountryController {

*@Autowired*

private CountryService countryService;

*@GetMapping*("/countries/{code}")

public Country getCountry(*@PathVariable* String code) {

return countryService.getCountry(code);

}

}

**SpringLearnApplication Class:**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

# Output:

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise: Create authentication service that returns JWT**

# Scenario:

You are building an authentication service in Spring Boot that returns a JWT (JSON Web Token) when valid user credentials are passed using HTTP Basic Authentication. This service will serve as the entry point for secure APIs where a valid token is required for subsequent requests.

**Procedure:**

**Step 1: Create Authentication Controller**

* Create a package **com.cognizant.JWT.controller** and annotate class with **@RestController.**
* Add method public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader).
* Decode credentials from the Authorization header and Generate JWT using a utility method and return as JSON.

**Step 2: Configure Security (SecurityConfig.java)**

* Allow unauthenticated access to /authenticate and Disable CSRF, authorize all other endpoints with authentication.
* Use in-memory users (e.g., user:pwd) for testing.

**Step 3: Create JWT Utility**

* Create a utility package under **com.cognizant.JWT.**
* Use io.jsonwebtoken.Jwts and JwtBuilder.
* Include method like generateToken(String username).

**Step 4: Create the Main Application Class and Test it**

* Run the JwtAuthApplication class and test the URL http://localhost:8090/authenticate in a browser or in Postman.

# Implementation:

# JwtAuthApplication Class:

package com.cognizant.JWT;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class JwtAuthApplication {

public static void main(String[] args) {

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

}

}

# AuthenticatonController Class:

package com.cognizant.JWT.controller;

import com.cognizant.JWT.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import jakarta.servlet.http.HttpServletRequest;

*@RestController*

public class AuthenticationController {

*@Autowired*

private JwtUtil jwtUtil;

*@GetMapping*("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

if (authHeader != null && authHeader.startsWith("Basic ")) {

String base64Credentials = authHeader.substring("Basic ".length());

byte[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String[] credentials = new String(decodedBytes).split(":", 2);

String username = credentials[0];

String password = credentials[1];

if ("user".equals(username) && "pwd".equals(password)) {

String token = jwtUtil.generateToken(username);

return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");

}

}

return ResponseEntity.*status*(401).body("Unauthorized");

}

}

# JwtUtil Class:

package com.cognizant.JWT.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.util.Date;

import java.security.Key;

import org.springframework.stereotype.Component;

*@Component*

public class JwtUtil {

private final Key key = Keys.*secretKeyFor*(*SignatureAlgorithm*.***HS256***);

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(key)

.compact();

}

}

# SecurityConfig Class:

package com.cognizant.JWT.security;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Bean;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf(csrf -> csrf.disable());

return http.build();

}

}

# Output:

# A screenshot of a computer program AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**