**SPRING REST USING SPRING BOOT 3**

**Hands on 1**

**Create a Spring Web Project using Maven**

**CODE:**

**HelloController.java**

package com.cognizant.spring\_learn;

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

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

*@RestController*

public class HelloController {

*@GetMapping*("/")

public String hello() {

return "Welcome to Spring Boot!";

}

}

**application.properties**

spring.application.name=spring-learn

server.port=3031

**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) {

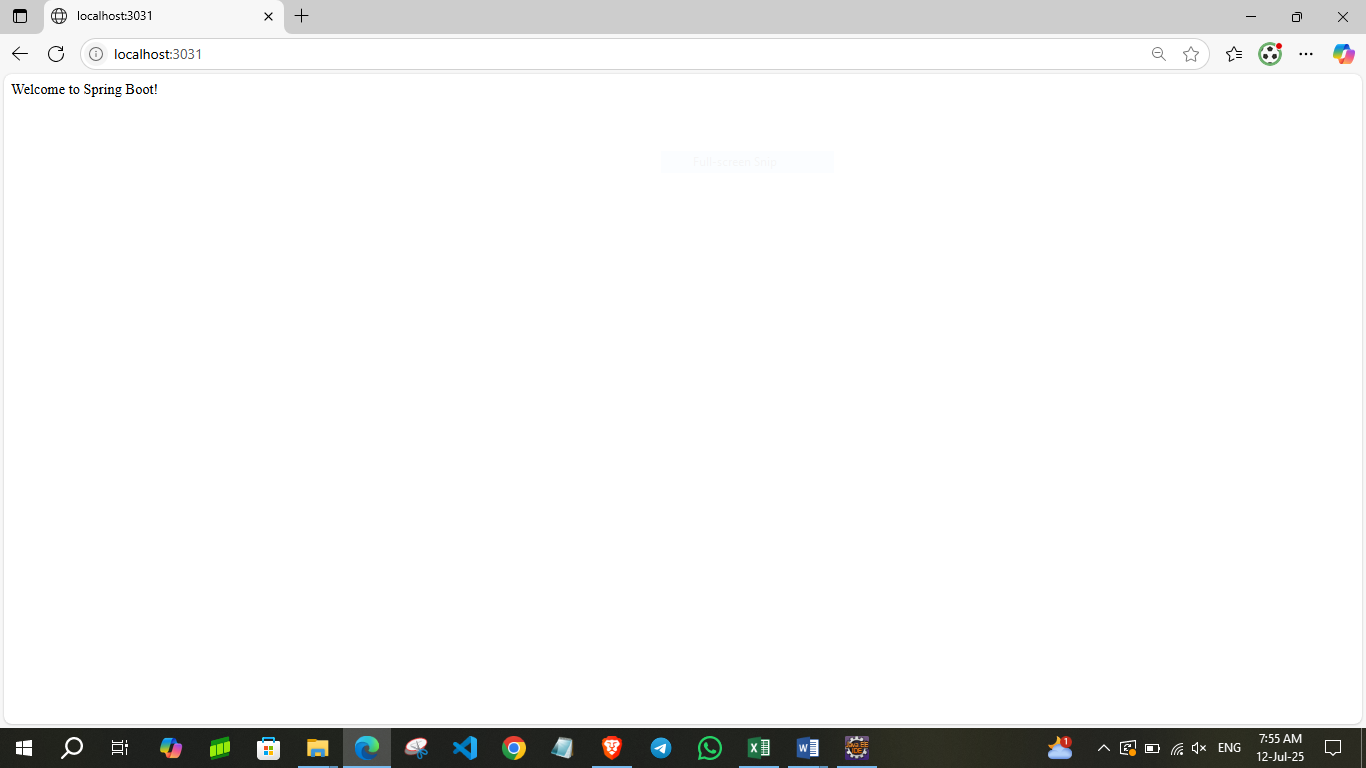
SpringApplication.*run*(SpringLearnApplication.class, args);

System.***out***.println("SpringLearnApplication started...");

}

}

**OUTPUT:**



**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML**

**CODE:**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

public static void main(String[] args) {

*displayDate*();

}

public static void displayDate() {

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

try {

Date currentDate = new Date();

String formattedDate = format.format(currentDate);

System.***out***.println("Current Date & Time: " + formattedDate);

} catch (Exception e) {

System.***out***.println("Error parsing date: " + e.getMessage());

}

((ClassPathXmlApplicationContext) context).close();

}

}

**date-format.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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="dateFormat" class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy HH:MM:SS" />

</bean>

</beans>

**Hands on 3**

**Hello World RESTful Web Service**   
  
Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework.

**CODE:**

**application.properties**

spring.application.name=spring-learn

server.port=8081

**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 World!!";

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

return message;

}

}

**SpringLearnApplcation.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);

}

}

**Hands on 4**

**REST - Country Web Service**   
  
Write a REST service that returns India country details in the earlier created spring learn application.

**CODE:**

**Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String name;

private String code;

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

}

**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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="in" class="com.cognizant.springlearn.model.Country">

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

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

</bean>

</beans>

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

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

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

*@RestController*

public class CountryController {

*@GetMapping*("/country")

public Country getCountry() {

Country country = new Country();

country.setName("India");

country.setCode("IN");

return country;

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

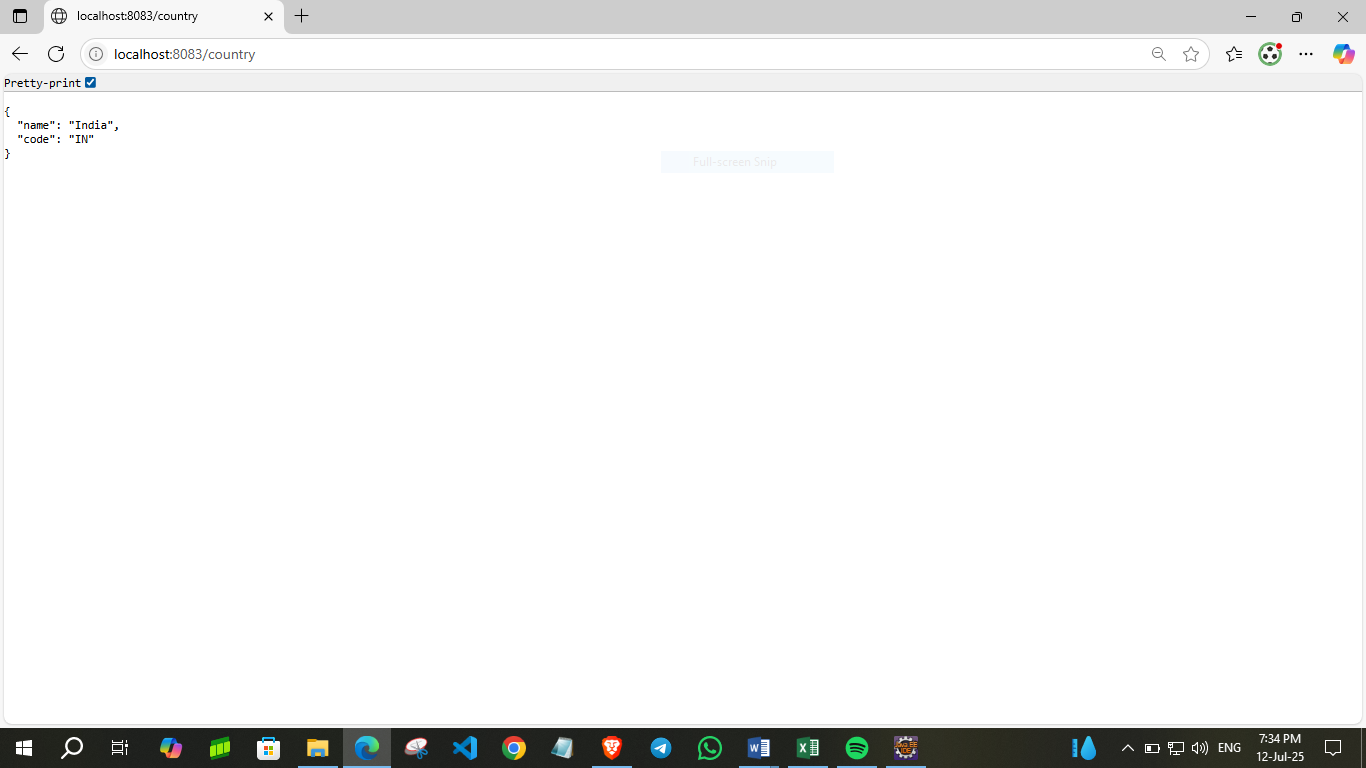
public static void main(String[] args) {

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

}

}

**OUTPUT:**



**Hands on 5**

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

**CODE:**

**Country.java**

package com.cognizant.spring\_learn.model;

import jakarta.xml.bind.annotation.XmlElement;

import jakarta.xml.bind.annotation.XmlRootElement;

// Root element name should match the XML tag (e.g., <country>)

*@XmlRootElement*(name = "country")

public class Country {

private String code;

private String name;

*@XmlElement*

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

*@XmlElement*

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**CountryService.java**

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.model.Country;

import com.cognizant.spring\_learn.model.CountryList;

import org.springframework.stereotype.Service;

import jakarta.xml.bind.JAXBContext;

import jakarta.xml.bind.Unmarshaller;

import java.io.InputStream;

*@Service*

public class CountryService {

public Country getCountry(String code) throws Exception {

// Load XML from resources

InputStream is = getClass().getResourceAsStream("/country.xml");

if (is == null) {

throw new Exception("country.xml not found in resources");

}

JAXBContext context = JAXBContext.*newInstance*(CountryList.class);

Unmarshaller unmarshaller = context.createUnmarshaller();

CountryList countryList = (CountryList) unmarshaller.unmarshal(is);

// Case-insensitive search for country code

return countryList.getCountries()

.stream()

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

.findFirst()

.orElseThrow(() -> new Exception("Country with code " + code + " not found"));

}

}

**CountryList.java**

package com.cognizant.spring\_learn.model;

import java.util.List;

import jakarta.xml.bind.annotation.XmlElement;

import jakarta.xml.bind.annotation.XmlRootElement;

*@XmlRootElement*(name = "countries")

public class CountryList {

private List<Country> countries;

*@XmlElement*(name = "country")

public List<Country> getCountries() {

return countries;

}

public void setCountries(List<Country> countries) {

this.countries = countries;

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

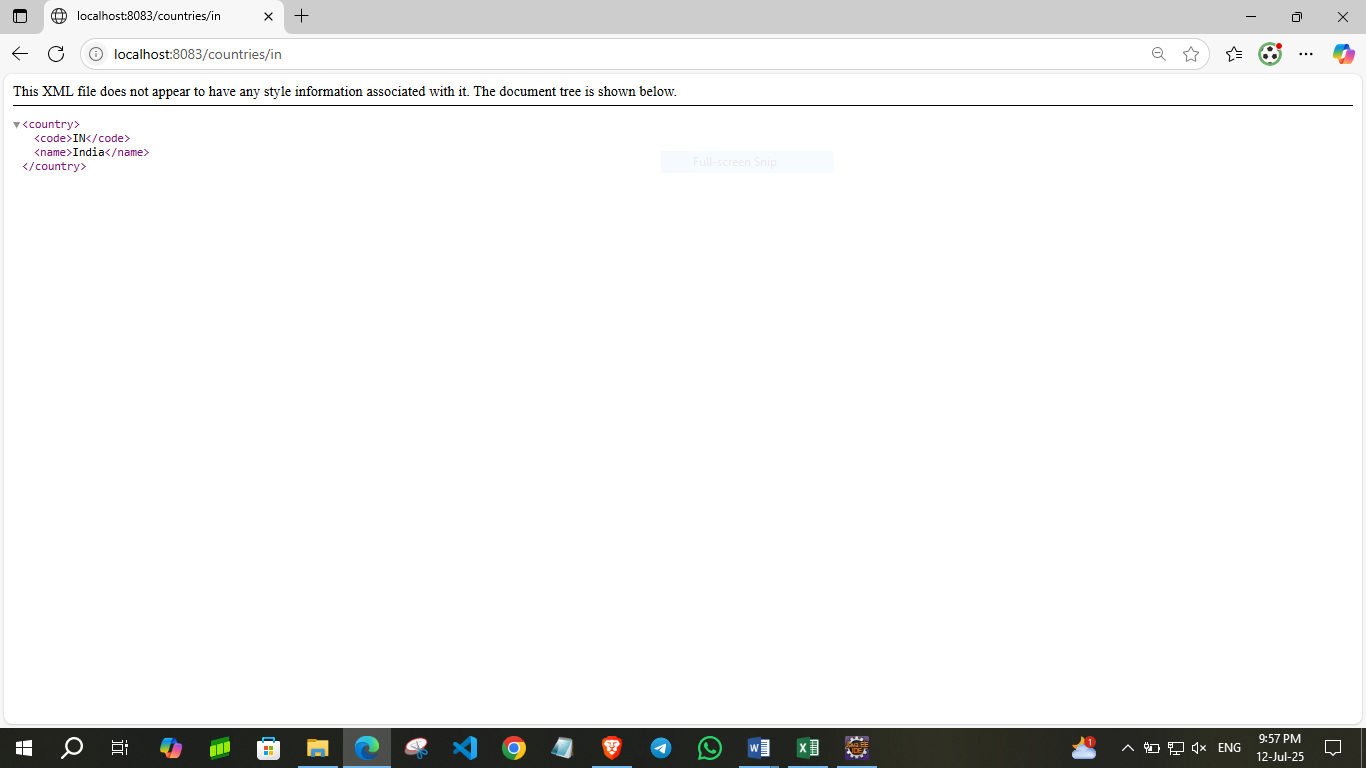
public static void main(String[] args) {

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

}

}

**OUTPUT:**



**Hands on 6**

**Create authentication service that returns JWT**   
  
As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

**CODE:**

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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().disable() // Disable CSRF for simplicity

.authorizeHttpRequests()

.anyRequest().permitAll()

.and()

.httpBasic(); // Enables basic auth

return http.build();

}

}

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.security.JwtUtil;

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

import org.springframework.http.ResponseEntity;

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

import jakarta.servlet.http.HttpServletRequest; // ✅ Updated import

import java.util.Base64;

@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 ")) {

return ResponseEntity.status(401).body("Missing or invalid Authorization header");

}

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

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

String credentials = new String(decoded);

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

String username = userDetails[0];

String password = userDetails[1];

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

String token = jwtUtil.generateToken(username);

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

} else {

return ResponseEntity.status(403).body("Invalid credentials");

}

}

}

**JwUtil.java**

package com.cognizant.spring\_learn.security;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256); // Generate once per instance

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 minutes

.signWith(key)

.compact();

}

}

**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);

}

}

**OUTPUT:**

