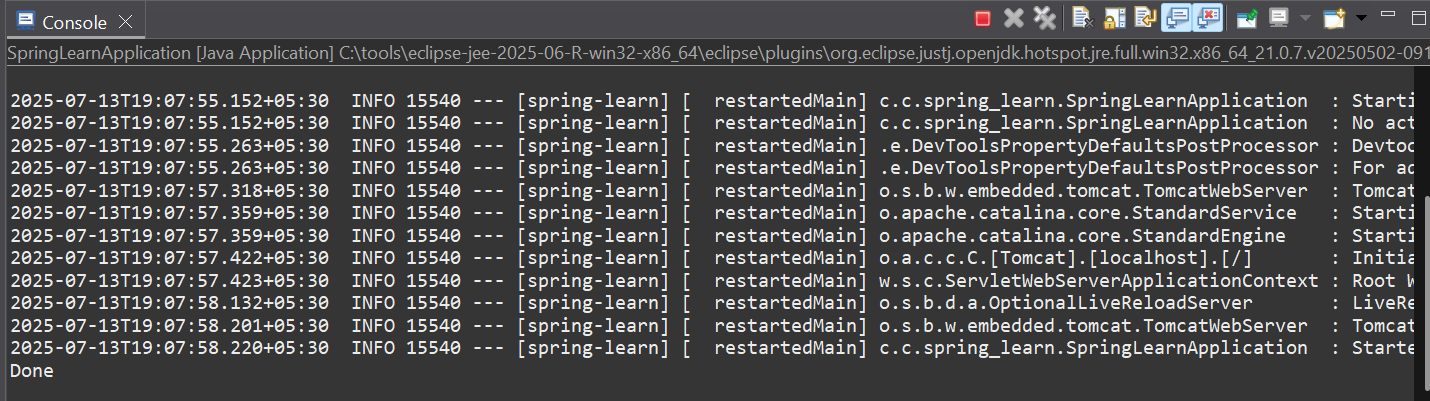
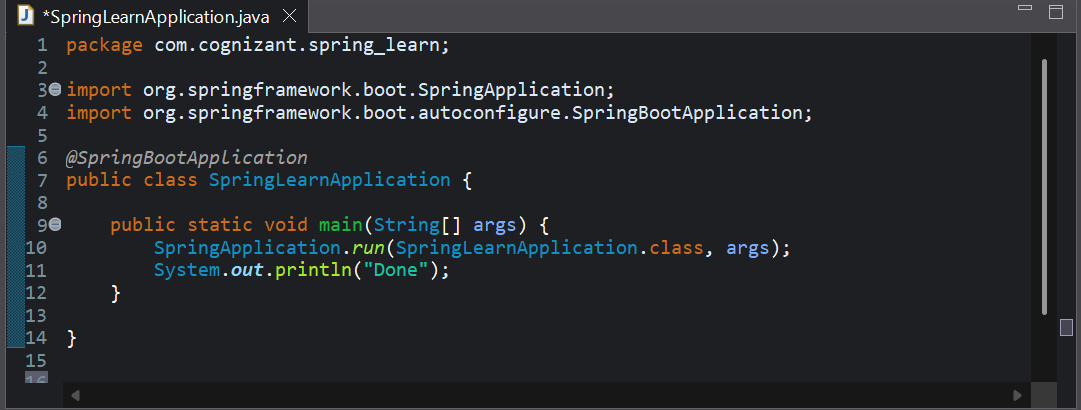
WEEK 4: Spring Core and Maven

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





**Exercise 2: Spring Core – Load Country from Spring Configuration XML**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

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

public static void displayCountry() {

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

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

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

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

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

System.***out***.println("Inside displayCountry() method");

System.***out***.println("Country details: " + country);

}

public static void main(String[] args) {

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

*displayCountry*();

}

}

**Country.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

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

private String code;

private String name;

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 + '\'' +

'}';

}

}

**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"*>

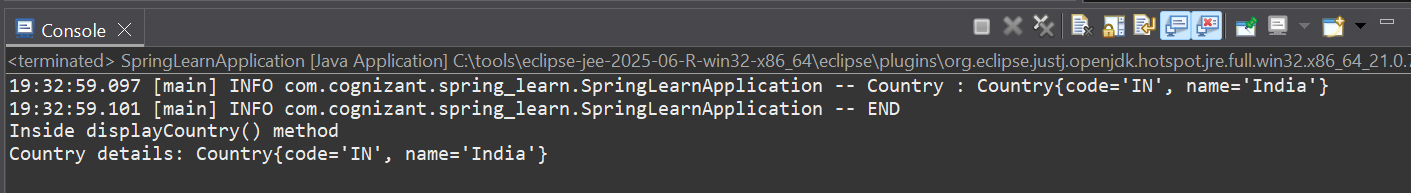
<property name=*"code"* value=*"IN"* />

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

</bean>

</beans>

**Output:**



**Exercise 3: Hello World RESTful Web Service**

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("sayHello() method START");

String response = "Hello World!!";

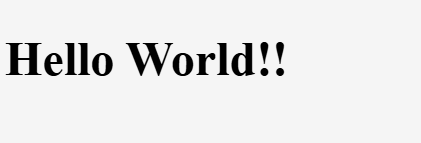
LOGGER.info("sayHello() method END");

return response;

}

}

**Output:**



**Exercise 4: REST - Get country based on country code**

**1. Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = 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;

}

// toString()

@Override

public String toString() {

return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';

}

}

**2. 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="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**3. CountryService.java**

package com.cognizant.spring\_learn.service;

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

private List<Country> countryList;

public CountryService() {

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

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

}

public Country getCountry(String code) {

if (code == null) return null;

// Using Java 8 Stream API for case-insensitive search

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

4. CountryController.java

java

CopyEdit

package com.cognizant.spring\_learn.controller;

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

@RestController

public class CountryController {

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

private final CountryService countryService;

public CountryController(CountryService countryService) {

this.countryService = countryService;

}

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

public Country getCountry(@PathVariable String code) {

LOGGER.info("getCountry() called with code: {}", code);

Country country = countryService.getCountry(code);

if (country == null) {

LOGGER.info("Country with code {} not found.", code);

// Optionally, throw exception or return null

} else {

LOGGER.info("Found country: {}", country);

}

return country;

}

}

**Output:**

****

**Exercise 5: Create authentication service that returns JWT**

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

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

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic(); // Enable Basic Auth for /authenticate

}

}

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.http.ResponseEntity;

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

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

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.Date;

@RestController

public class AuthenticationController {

private static final String SECRET\_KEY = "mysecretkey12345"; // Use a secure key in real apps

@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");

}

// Extract Base64 encoded username:password

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

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

String credentials = new String(credDecoded);

// credentials = "username:password"

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

if (values.length != 2) {

return ResponseEntity.status(400).body("Invalid basic authentication token");

}

String username = values[0];

String password = values[1];

// For demo: hardcoded user validation

if (!"user".equals(username) || !"pwd".equals(password)) {

return ResponseEntity.status(401).body("Invalid username or password");

}

// Generate JWT token

String jwt = Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

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

}

}

**Output:**

****