**WEEK – 4**

1. **Hands-on 1: Create a Spring Web Project using Maven:**

**STEP 1: Go to Spring Initialize**

Open your browser and go to: <https://start.spring.io>

**STEP 2: Fill Project Metadata**

* **Group:** com.cognizant
* **Artifact:** spring-learn

**STEP 3: Add Dependencies**

Click "Add Dependencies" and include:

* Spring Web
* Spring Boot DevTools

**STEP 4: Generate and Extract Project**

* Click **Generate**, it downloads a ZIP file.
* Extract the ZIP to a folder inside your **Eclipse Workspace**.

**STEP 5: Import Project into Eclipse**

* Open Eclipse
* Go to: File → Import → Maven → Existing Maven Projects → Next
* Browse and select the extracted folder → Click **Finish**

**STEP 6: Build the Project via Command Line**

* Open Command Prompt → go to the project folder  
  **Run this command:**

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456

**STEP 7: Add Logging to Main Class**

Open SpringLearnApplication.java inside:

src/main/java/com/cognizant/springlearn/

**Edit it like this:**

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("END");

}

}

**STEP 8: Run the Application**

* Right-click on SpringLearnApplication.java  
  → Run As → Java Application

**STEP 9: Verify Console Output**

You should see logs:



1. **Hands-on 4: Spring Core – Load Country from Spring Configuration XML .**

**Steps:**

1. **Create a Project**
2. Go to <https://start.spring.io>
3. Group: com.cognizant
4. Artifact: spring-learn
5. Dependencies: None needed for this task
6. Extract the zip and import it in Eclipse via:
7. File > Import > Existing Maven Projects
8. **Create Country Java Bean**

**📁 Path: src/main/java/com/cognizant/springlearn/Country.java**

package com.cognizant.springlearn;

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("Getting code: {}", code);

return code;

}

public void setCode(String code) {

LOGGER.debug("Setting code: {}", code);

this.code = code;

}

public String getName() {

LOGGER.debug("Getting name: {}", name);

return name;

}

public void setName(String name) {

LOGGER.debug("Setting name: {}", name);

this.name = name;

}

@Override

public String toString() {

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

}

}

1. **Create country.xml**

📁 Path: src/main/resources/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="country" class="com.cognizant.springlearn.Country">

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

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

</bean>

</beans>

1. **Modify SpringLearnApplication.java**

📁 Path: src/main/java/com/cognizant/springlearn/SpringLearnApplication.java

package com.cognizant.springlearn;

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

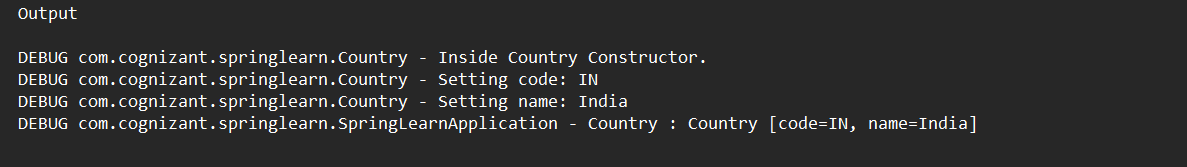
LOGGER.debug("Country : {}", country.toString());

}

}

1. **Run the Application**
2. Right-click SpringLearnApplication.java
3. Select Run As > Java Application

**Console Output:**

****

1. **Hands-on 5: Hello World RESTful Web Service**

**Steps:**

**STEP 1: Create Spring Boot Project**

1. Go to: <https://start.spring.io>
2. Fill the fields:
   1. Group: com.cognizant
   2. Artifact: spring-learn-rest
3. Add dependencies:
   1. Spring Web
4. Click on "Generate" to download the project zip.
5. Extract the ZIP to your Eclipse workspace folder.

**STEP 2: Import the Project into Eclipse**

1. Open Eclipse.
2. Go to File > Import.
3. Select Maven > Existing Maven Projects → Next.
4. Browse and select the extracted spring-learn-rest folder.
5. Click Finish.

**STEP 3: Create the HelloController**

📁 Location: src/main/java/com/cognizant/springlearn/controller/HelloController.java

Right-click src/main/java → New → Package → Name it com.cognizant.springlearn.controller  
Then inside it, create a new class named HelloController:

Paste the following code:

package com.cognizant.springlearn.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");

String message = "Hello World";

LOGGER.info("END");

return message;

}

}

**STEP 4: Update Application Properties**

To make sure it runs on port 8083:  
📁 src/main/resources → application.properties

Add this:

server.port=8083

**STEP 5: Run the Application**

1. Right-click SpringLearnApplication.java (in com.cognizant.springlearn).
2. Choose Run As > Java Application.
3. Wait for console to show:  
   Tomcat started on port 8083...

**STEP 6: Test the REST Endpoint**

1. Open Chrome or any browser.
2. Enter the URL: <http://localhost:8083/hello>
3. You should see:

**Hello World**

**Console Output:**

****

1. **Hands on 2 REST - Country Web Service**

**OBJECTIVE:**

Create a REST endpoint:

URL: http://localhost:8083/country

Returns a JSON response of **India's country details**

Controller: CountryController

Method: getCountryIndia()

Load India as a Spring bean from an XML configuration.

STEP 1: **Create the Country Model Class**

**Country.java**

**package com.cognizant.springlearn.model;**

public class Country {

private String code;

private String name;

// Getters and Setters

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;

}

}

**STEP 2: Create Spring XML Configuration**

**country.xml in src/main/resources**

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

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

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

</bean>

</beans>

**STEP 3: Create the Controller**

**CountryController.java**

**java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.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 = (Country) context.getBean("in");

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

return country;

}

}

**STEP 4: application.properties**

**Ensure this exists:**

**Properties:**

server.port=8083

**STEP 5: Run the Application In Eclipse:**

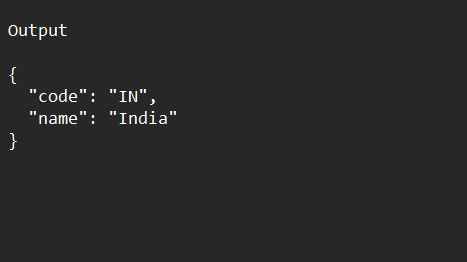
Run your main class (typically SpringLearnApplication.java)

**STEP 6: Test the Endpoint**

**In Browser**

http://localhost:8083/country

**Console Output:**

****

**STEP 7: Understanding the Execution & Internals**

* ApplicationContext loads country.xml
* Bean with ID "in" is fetched and cast to Country
* That Country object is returned
* Spring Boot automatically serializes the object to JSON

**5)Hands on 2 REST - Get country based on country code**

**OBJECTIVE**

Create endpoint:

* URL: /country/{code}
* Method: GET
* Returns: JSON object of country that matches the code (case insensitive)
* Service class handles the logic of searching from XML.

**STEP 1: country.xml — Add a List of Countries**

**xml**

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

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

</list>

</constructor-arg>

</bean>

**STEP 2: CountryService.java**

**java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.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 {

public Country getCountry(String code) {

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

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

// Case-insensitive search using stream

return countryList.stream()

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

.findFirst()

.orElse(null); // You can throw a custom exception instead

}

}

**STEP 3: CountryController.java**

**java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

LOGGER.info("START - getCountry() with code: " + code);

Country country = countryService.getCountry(code);

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

return country;

}

}

**STEP 4: Run and Test**

**Run the Application**

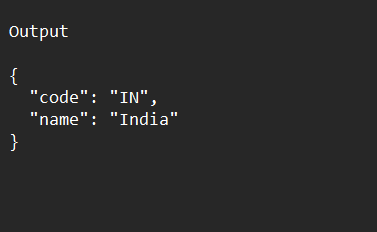
In Eclipse, run your main Spring Boot application class (with @SpringBootApplication)

**Test the Endpoint**

**In Browser or Postman:**

**URL:**

<http://localhost:8083/country/in>



**//country/us, /country/CN — all will work case-insensitively.**

**6)Hands on 5 Create authentication service that returns JWT**

**STEP 1: Project Setup**

**xml**

<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>

**Package Structure**

arduino

com.cognizant.springsecurityjwt

controller

└──**>** AuthenticationController.java

config

└──**>** SecurityConfig.java

util

└──**>** JwtUtil.java

**1. AuthenticationController.java**

**java**

package com.cognizant.springsecurityjwt.controller;

import com.cognizant.springsecurityjwt.util.JwtUtil;

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

import org.springframework.security.core.Authentication;

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public Map<String, String> generateToken(Authentication authentication) {

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

Map<String, String> response = new HashMap<>();

response.put("token", token);

return response;

}

}

**2. JwtUtil.java**

**java**

package com.cognizant.springsecurityjwt.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secretKey = "mysecretkey"; // Ideally, store this in properties or env

public String generateToken(String username) {

long currentTime = System.currentTimeMillis();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(currentTime))

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

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

**3. SecurityConfig.java**

**java**

package com.cognizant.springsecurityjwt.config;

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;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").authenticated()

.anyRequest().permitAll()

.and()

.httpBasic(); // Enables -u user:pwd basic auth

return http.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance(); // Only for demo purposes

}

}

**4. Add User Credentials**

**application.properties (optional)**

Add in-memory user in SecurityConfig (alternative to application.properties)

**java**

import org.springframework.security.core.userdetails.User;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Bean

public InMemoryUserDetailsManager userDetailsService() {

var user = User.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

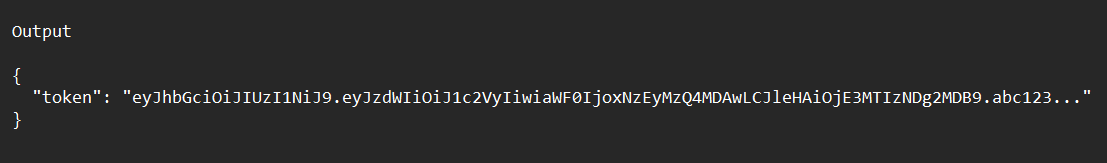
**5. Run & Test**

**Start your app on port 8090**

**Curl command**

curl -s -u user:pwd <http://localhost:8090/authenticate>

**Console Output:**

****