**Hands on 1**

**Create a Spring Web Project using Maven**

### **1. Create Project on Spring Initializr**

* Go to: [https://start.spring.io](https://start.spring.io/)
* Fill in the project metadata:  
  + **Group**: com.cognizant
  + **Artifact**: spring-learn
* **Dependencies**:  
  + Spring Boot DevTools
  + Spring Web
* Click **Generate** to download the .zip file.

### **2. Extract and Setup in Eclipse**

* **Extract** the zip to a preferred workspace folder (e.g., C:\eclipse-workspace\spring-learn).
* Open **Command Prompt** and **navigate to the extracted project folder**.

### **3. Run Maven Build with Proxy (if required):**

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

### **4. Import in Eclipse**

* In **Eclipse**:  
  + Go to File > Import > Maven > Existing Maven Projects
  + Browse and select the extracted project folder
  + Click **Finish**

### **5. Run the Project**

* Navigate to src/main/java/com/cognizant/springlearn/SpringLearnApplication.java
* Add a simple log in the main() method:

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

* Right-click the file → Run As → Java Application

**1. src/main/java**

* Contains **application source code**.
* Files here include:  
  + SpringLearnApplication.java: the entry point.
  + Additional controllers, services, etc. can be added here.

### **2. src/main/resources**

* Contains **configuration and static files**:  
  + application.properties: used for setting port, DB connections, etc.
  + Static files: templates, static assets (if using web views).

### **3. src/test/java**

* Contains **unit test classes**.
* Auto-generated class: SpringLearnApplicationTests.java
* Uses @SpringBootTest for testing the application context.

### **4. SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

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

SpringApplication.run(SpringLearnApplication.class, args);

}

}

### **5. @SpringBootApplication Annotation**

* It's a **meta-annotation** that combines:  
  + @Configuration: Defines Spring configuration.
  + @EnableAutoConfiguration: Enables Spring Boot's auto-config.
  + @ComponentScan: Scans current and sub-packages for beans.

### **6. pom.xml Walkthrough**

#### 

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Web Project</description>

<parent>

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

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

<version>3.x.x</version>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

### **7. Dependency Tree**

In Eclipse:

* Go to pom.xml → Tab **Dependency Hierarchy**

spring-boot-starter-web

└── spring-boot-starter

└── spring-boot

└── spring-boot-autoconfigure

└── slf4j-api

└── logback-classic

└── ...

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

## 

### **1. Create 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="country" class="com.cognizant.springlearn.Country">

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

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

</bean>

</beans>

### **2. Create Country.java class**

package com.cognizant.springlearn;

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 + "]";

}

}

### **3. Modify 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.debug("START");

displayCountry();

LOGGER.debug("END");

}

public static void displayCountry() {

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

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

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

}

}

## **Output**

### **Console Log**

DEBUG - START

DEBUG - Inside Country Constructor.

DEBUG - Inside setCode()

DEBUG - Inside setName()

DEBUG - Inside getCode()

DEBUG - Inside getName()

DEBUG - Country : Country [code=IN, name=India]

DEBUG - END

# **Hands-on Hello World RESTful Web Service**

## 

### **1. Project Setup**

# src/main/resources/application.properties

server.port=8083

**Path:** src/main/java/com/cognizant/springlearn/controller/HelloController.java

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.debug("START: sayHello()");

String response = "Hello World!!";

LOGGER.debug("END: sayHello()");

return response;

}

}

### **3. Run the Application**

SpringApplication.run(SpringLearnApplication.class, args);

## **Output**

### **Request:**

GET /hello HTTP/1.1

Host: localhost:8083

### **Response:**

Hello World!!

### **Console Logs:**

DEBUG - START: sayHello()

DEBUG - END: sayHello()

# **Hands-on: REST - Country Web Service**

### 

src/main/resources/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.springlearn.Country">

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

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

</bean>

</beans>

2.

src/main/java/com/cognizant/springlearn/controller/CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.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.debug("START: getCountryIndia()");

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

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

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

LOGGER.debug("END: getCountryIndia()");

return country;

}

}

### **Request and Response**

* **Request URL**: http://localhost:8083/country
* **Method**: GET

**JSON Response**:

{

"code": "IN",

"name": "India"

}

# **REST – Get Country Based on Country Code**

## **Project Structure**

spring-learn

├── controller

│ └── CountryController.java

├── model

│ └── Country.java

├── service

│ └── CountryService.java

├── resources

│ └── country.xml

## **1. country.xml – Define a list of countries**

src/main/resources/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="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

## **2. Country.java (POJO)**

com.cognizant.springlearn.Country

package com.cognizant.springlearn;

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; }

@Override

public String toString() {

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

}

}

## **3. CountryService.java**

com.cognizant.springlearn.service.CountryService

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.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> countries = context.getBean("countryList", List.class);

return countries.stream()

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

.findFirst()

.orElse(null); // Optional: Throw custom exception for not found

}

}

## **4. CountryController.java**

com.cognizant.springlearn.controller.CountryController

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.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("/countries/{code}")

public Country getCountry(@PathVariable String code) {

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

Country country = countryService.getCountry(code);

LOGGER.debug("END: getCountry() → {}", country);

return country;

}

}

## **Request**

GET http://localhost:8083/countries/in

## **JSON Response**

{

"code": "IN",

"name": "India"

}

## **Console Logs**

DEBUG - START: getCountry(in)

DEBUG - END: getCountry() → Country [code=IN, name=India]

# **Hands-on: JWT Authentication Service**

### 

## **Step 1: Create Authentication Controller**

### **Controller: AuthenticationController.java**

src/main/java/com/cognizant/springlearn/controller/AuthenticationController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.security.JwtTokenProvider;

import org.springframework.http.ResponseEntity;

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

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

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

import java.util.Base64;

@RestController

public class AuthenticationController {

private final JwtTokenProvider jwtTokenProvider;

public AuthenticationController(JwtTokenProvider jwtTokenProvider) {

this.jwtTokenProvider = jwtTokenProvider;

}

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authorizationHeader) {

// Extract the basic auth credentials from the Authorization header

String[] credentials = extractCredentials(authorizationHeader);

String username = credentials[0];

String password = credentials[1];

// Generate JWT token

String token = jwtTokenProvider.createToken(username);

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

}

private String[] extractCredentials(String authorizationHeader) {

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

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

String decodedString = new String(decoded);

return decodedString.split(":", 2);

}

}

## **Step 2: Create JWT Token Provider**

src/main/java/com/cognizant/springlearn/security/JwtTokenProvider.java

package com.cognizant.springlearn.security;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtTokenProvider {

private static final String SECRET\_KEY = "secretkey";

private static final long EXPIRATION\_TIME = 3600000L; // 1 hour

public String createToken(String username) {

return Jwts.builder()

.setSubject("user")

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.claim("username", username)

.signWith(SignatureAlgorithm.HS512, SECRET\_KEY)

.compact();

}

}

## **Step 3: Configure Spring Security**

### **SecurityConfig: Configure Spring Security to allow HTTP Basic Authentication for the /authenticate endpoint.**

src/main/java/com/cognizant/springlearn/security/SecurityConfig.java

package com.cognizant.springlearn.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.config.annotation.web.configuration.EnableWebSecurity;

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

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

.anyRequest().authenticated();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

## **Step 4: Testing with cURL**

### **Test Request:**

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

### **Response:**

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"

}