**Hands on 1**

**Create a Spring Web Project using Maven**   
  
Follow steps below to create a project: 

1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. 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
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.
10. Run the SpringLearnApplication class.

SME to walk through the following aspects related to the project created:

1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation
6. pom.xml
   1. Walkthrough all the configuration defined in XML file
   2. Open 'Dependency Hierarchy' and show the dependency tree.

Home.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard - Spring Web App</title>

<!-- Bootstrap CDN -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

background-color: #f4f7fa;

}

.navbar {

background-color: #0d6efd;

}

.navbar-brand, .nav-link {

color: white !important;

}

.dashboard-cards {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));

gap: 1.5rem;

margin-top: 2rem;

}

.card {

border: none;

border-radius: 12px;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

transition: transform 0.2s;

}

.card:hover {

transform: translateY(-5px);

}

.footer {

text-align: center;

padding: 20px;

margin-top: 3rem;

color: #6c757d;

font-size: 0.9rem;

}

</style>

</head>

<body>

<nav class="navbar navbar-expand-lg">

<div class="container-fluid">

<a class="navbar-brand" href="#">Spring WebApp</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav ms-auto">

<li class="nav-item"><a class="nav-link" href="#">Dashboard</a></li>

<li class="nav-item"><a class="nav-link" href="#">Projects</a></li>

<li class="nav-item"><a class="nav-link" href="#">Reports</a></li>

<li class="nav-item"><a class="nav-link" href="#">Profile</a></li>

</ul>

</div>

</div>

</nav>

<div class="container">

<h2 class="mt-4">Welcome to Your Dashboard</h2>

<p class="text-muted">Here’s a quick overview of what's going on in your system today.</p>

<div class="dashboard-cards">

<div class="card p-4">

<h5>Total Users</h5>

<p class="text-primary fs-4">1,200</p>

</div>

<div class="card p-4">

<h5>Projects</h5>

<p class="text-success fs-4">28 Active</p>

</div>

<div class="card p-4">

<h5>New Messages</h5>

<p class="text-warning fs-4">5 Unread</p>

</div>

<div class="card p-4">

<h5>Reports</h5>

<p class="text-danger fs-4">3 Pending</p>

</div>

</div>

</div>

<div class="footer">

© 2025 Spring Web Dashboard. Built with ❤️ using Spring Boot + Bootstrap.

</div>

</body>

</html>

**Right click on com.example.springweb > New > Class**

Name it: HomeController

package com.example.springweb;

import org.springframework.stereotype.Controller;

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

@Controller

public class HomeController {

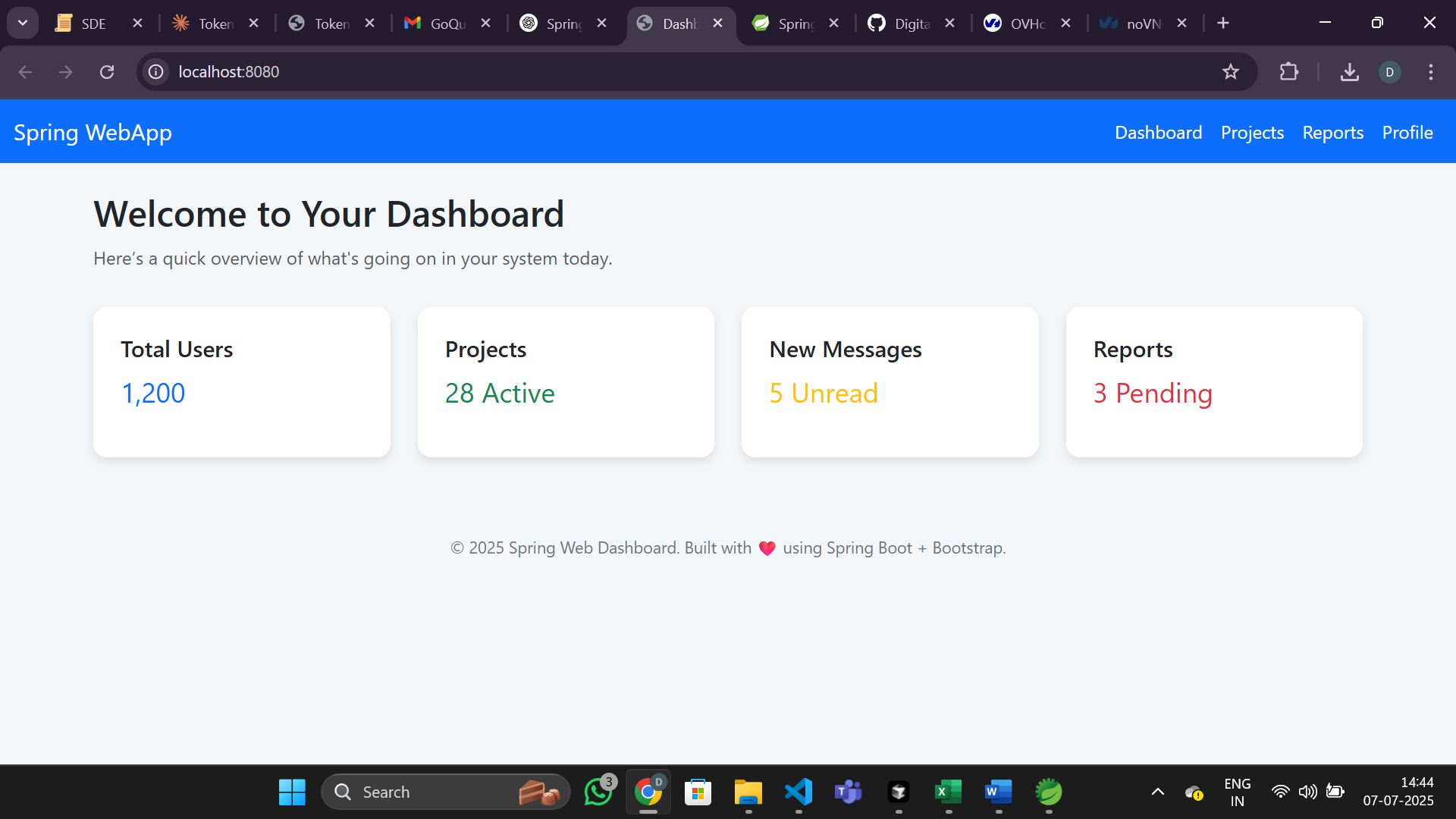
@GetMapping("/")

public String home() {

return "home"; // return home.html from templates

}

}



Spring Core – Load Country from Spring Configuration XML

Pom.xml

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>SpringCoreCountry</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

MainApp.java

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

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

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

}

}

src/main/resources/beans.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.example.Country">

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

<property name="capital" value="New Delhi"/>

</bean>

</beans>

package com.example;

public class Country {

private String name;

private String capital;

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCapital() {

return capital;

}

public void setCapital(String capital) {

this.capital = capital;

}

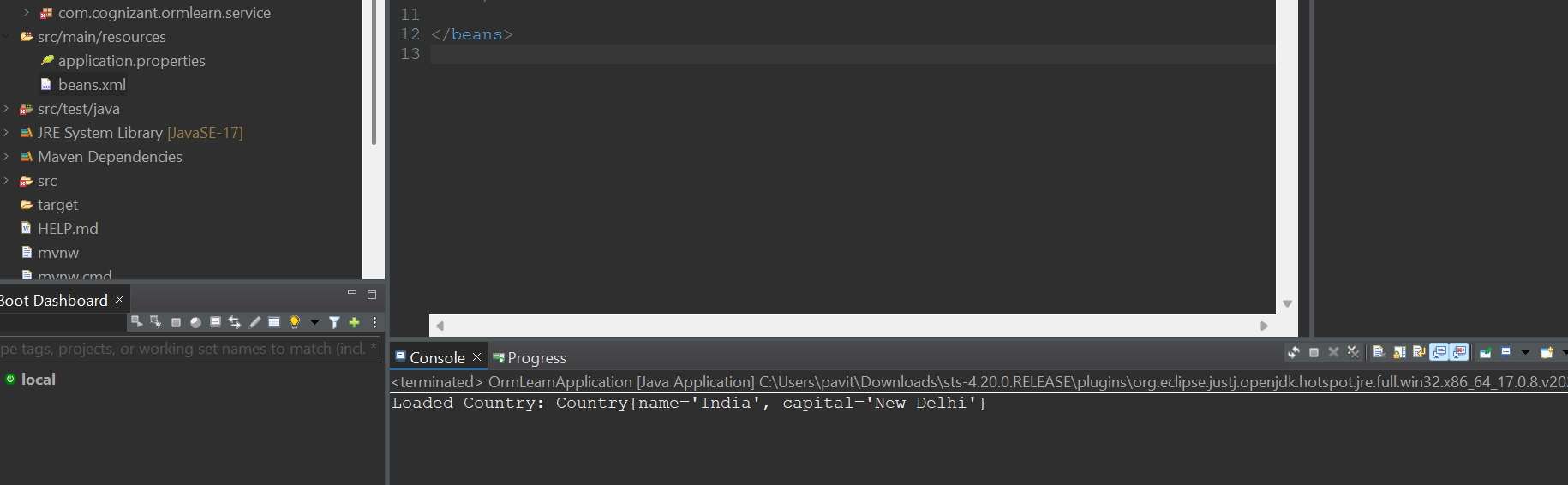
@Override

public String toString() {

return "Country{name='" + name + "', capital='" + capital + "'}";

}

}



**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. Refer details below:  
  
**Method:** GET  
**URL:** /hello  
**Controller:** com.cognizant.spring-learn.controller.HelloController  
**Method Signature:** public String sayHello()  
**Method Implementation:** return hard coded string "Hello World!!"  
**Sample Request**: http://localhost:8083/hello  
**Sample Response:** Hello World!!   
  
**IMPORTANT NOTE**: Don't forget to include start and end log in the sayHello() method.  
  
Try the URL http://localhost:8083/hello in both chrome browser and postman.  
  
SME to explain the following aspects:

* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

package com.example.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan(basePackages = "com.cognizant.spring\_learn")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

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 - sayHello()");

String message = "Hello World!!";

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

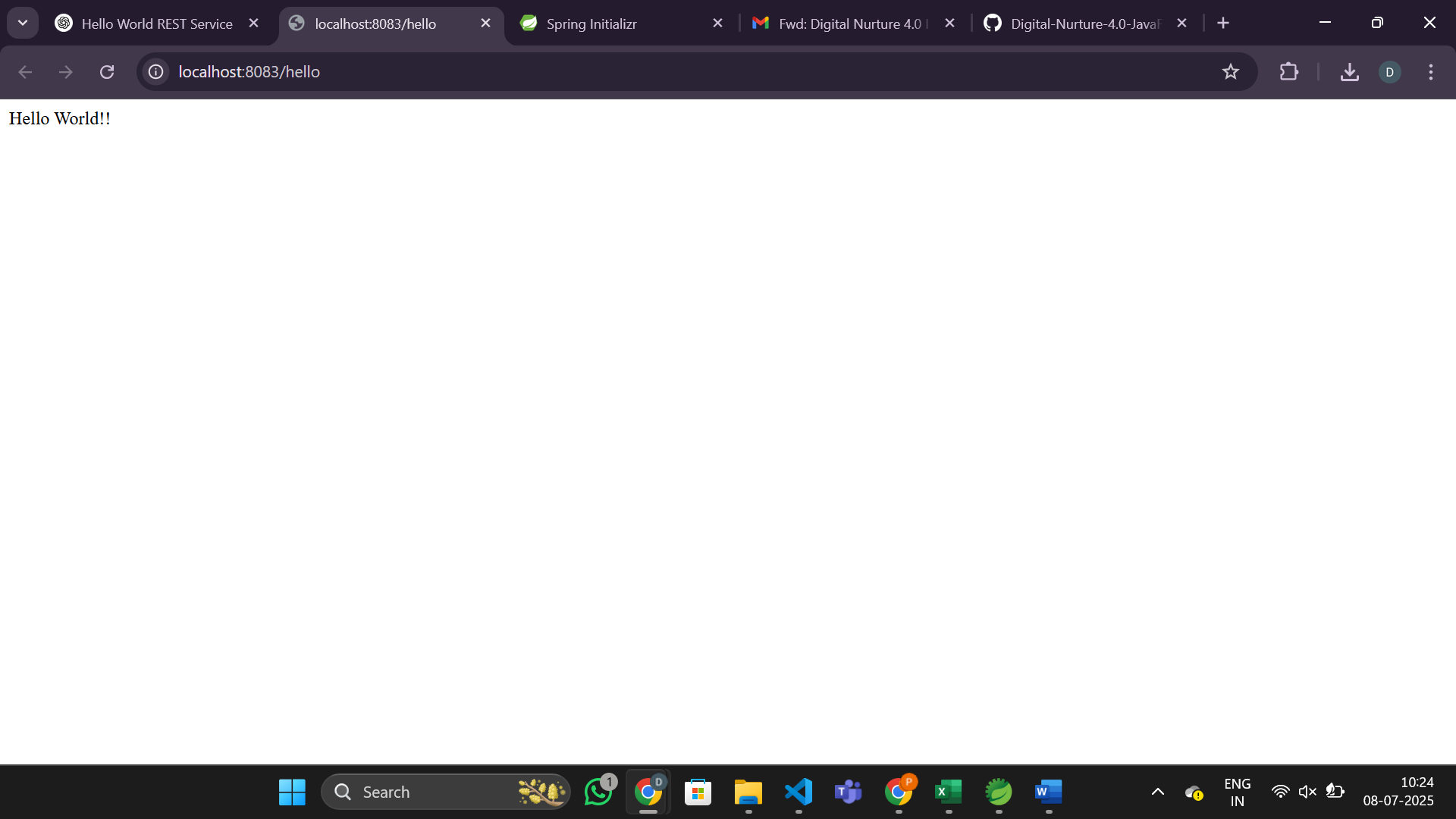
return message;

}

}

spring.application.name=spring-learn

server.port=8083



**REST - Country Web Service**   
  
Write a REST service that returns India country details in the earlier created spring learn application.  
  
**URL**: /country  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation**: @RequestMapping  
**Method Name**: getCountryIndia()  
**Method Implementation**: Load India bean from spring xml configuration and return  
**Sample Request**: http://localhost:8083/country  
**Sample Response**:

{

  "code": "IN",

  "name": "India"

}

SME to explain the following aspects:

* What happens in the controller method?
* How the bean is converted into JSON reponse?
* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

File: src/main/java/com/example/spring\_learn/model/Country.java

package com.example.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;

}

}

File: src/main/resources/beans.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="in" class="com.example.spring\_learn.model.Country">

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

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

</bean>

</beans>

File: src/main/java/com/example/spring\_learn/controller/CountryController.java

package com.example.spring\_learn.controller;

import com.example.spring\_learn.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("beans.xml");

Country country = (Country) context.getBean("in");

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

return country;

}

}

File: src/main/java/com/example/spring\_learn/SpringLearnApplication.java

package com.example.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

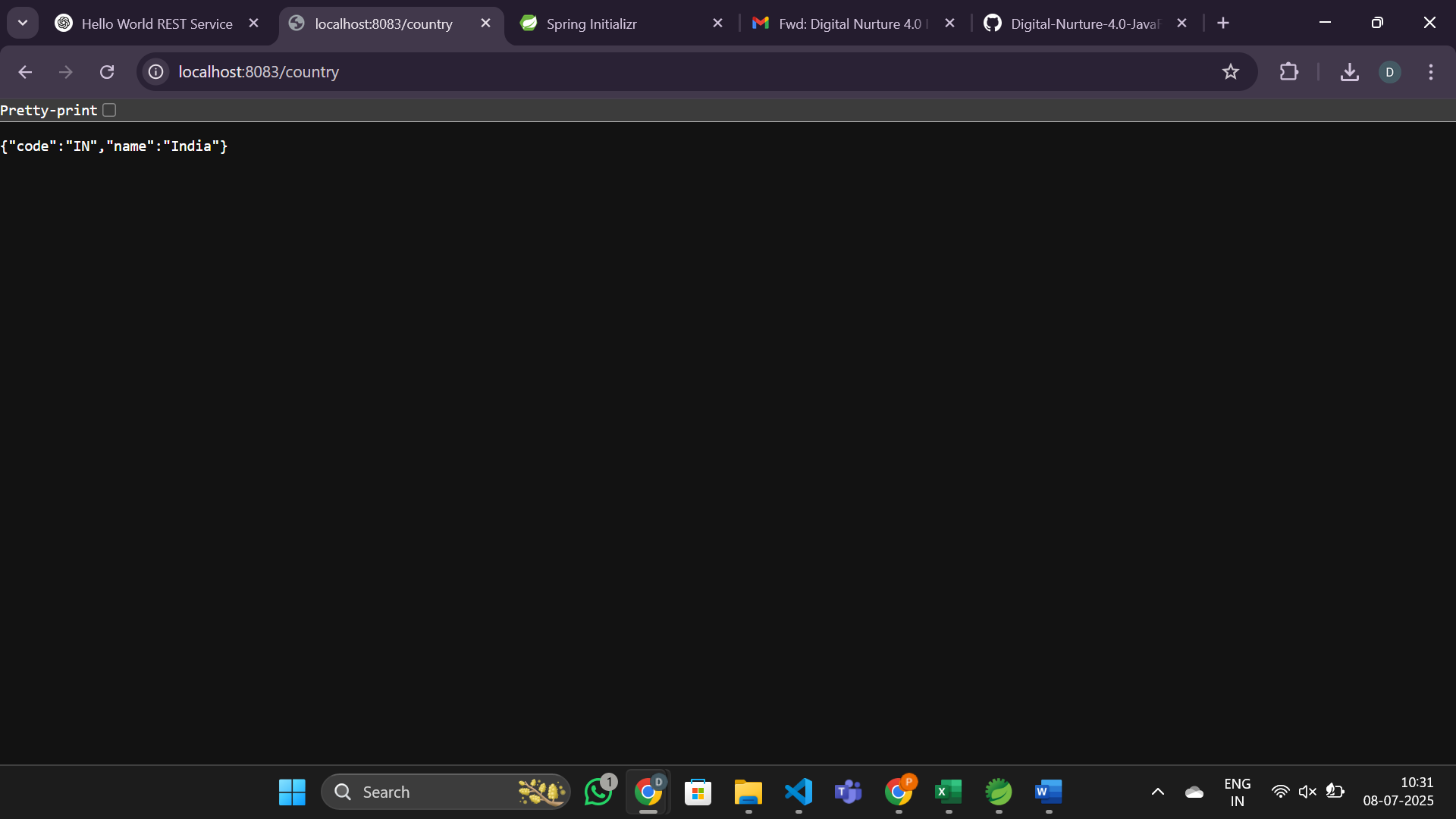
SpringApplication.run(SpringLearnApplication.class, args);

}

}

File: src/main/resources/application.properties

server.port=8083



**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.  
  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation:** @GetMapping("/countries/{code}")  
**Method Name**: getCountry(String code)  
**Method Implemetation**: Invoke countryService.getCountry(code)   
**Service Method:**com.cognizant.spring-learn.service.CountryService.getCountry(String code)  
  
**Service Method Implementation**:

* Get the country code using @PathVariable
* Get country list from country.xml
* Iterate through the country list
* Make a case insensitive matching of country code and return the country.
* Lambda expression can also be used instead of iterating the country list

**Sample Request**: http://localhost:8083/country/in  
  
**Sample Response**:

{

  "code": "IN",

  "name": "India"

}

File: src/main/java/com/example/spring\_learn/model/Country.java

package com.example.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;

}

}

File: src/main/java/com/example/spring\_learn/service/CountryService.java

package com.example.spring\_learn.service;

import com.example.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 {

public Country getCountry(String code) {

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

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

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

File: src/main/java/com/example/spring\_learn/controller/CountryController.java

package com.example.spring\_learn.controller;

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

import com.example.spring\_learn.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.info("START - getCountry()");

Country country = countryService.getCountry(code);

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

return country;

}

}

File: 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

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

<bean id="country1" class="com.example.spring\_learn.model.Country">

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

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

</bean>

<bean id="country2" class="com.example.spring\_learn.model.Country">

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

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

</bean>

<bean id="country3" class="com.example.spring\_learn.model.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="country1"/>

<ref bean="country2"/>

<ref bean="country3"/>

</list>

</constructor-arg>

</bean>

</beans>

File: src/main/java/com/example/spring\_learn/SpringLearnApplication.java

package com.example.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);

}

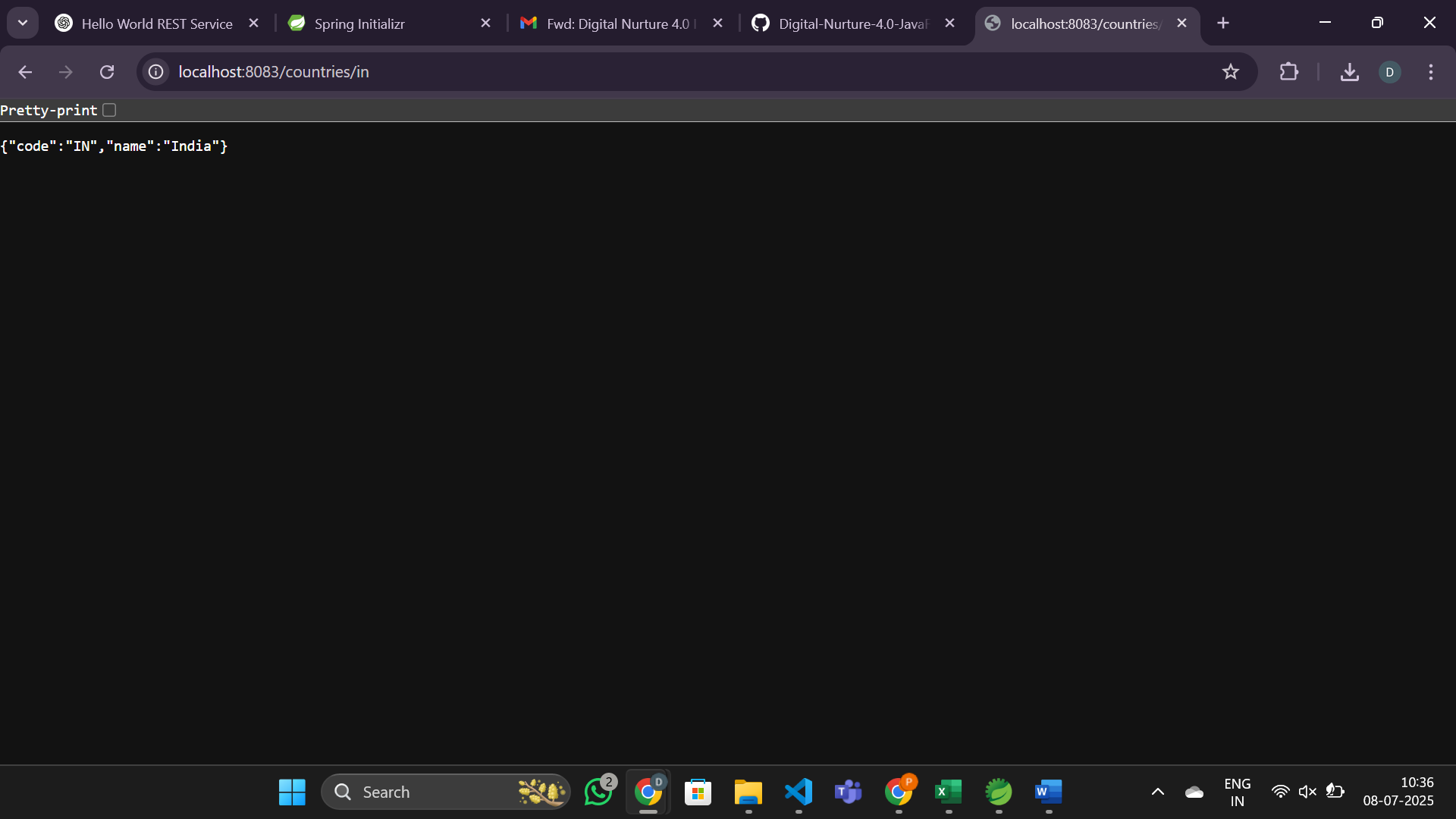
}

File: src/main/resources/application.properties

server.port=8083

Sample Request URL:

http://localhost:8083/countries/in



**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.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.  
  
**Request**

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

**Response**

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

This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.

**src/main/java/com/example/spring\_learn/SpringLearnApplication.java**

package com.example.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);

}

}

**src/main/java/com/example/spring\_learn/controller/AuthController.java**

package com.example.spring\_learn.controller;

import com.example.spring\_learn.util.JwtUtil;

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

import org.springframework.http.HttpHeaders;

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

@RestController

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader(HttpHeaders.AUTHORIZATION) String authHeader) {

System.out.println("Inside authenticate method");

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

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

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

String credentials = new String(decodedBytes);

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 + "\"}");

}

}

return ResponseEntity.status(401).body("{\"error\":\"Invalid Credentials\"}");

}

}

**src/main/java/com/example/spring\_learn/util/JwtUtil.java**

package com.example.spring\_learn.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 = "secret"; // use env variable in real apps

private long validityInMillis = 3600000; // 1 hour

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + validityInMillis))

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

**src/main/java/com/example/spring\_learn/config/SecurityConfig.java**

package com.example.spring\_learn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User

.withUsername("user")

.password("{noop}pwd") // {noop} for plain text password

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

}

**pom.xml (only relevant dependencies shown)**

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Security -->

<dependency>

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

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

</dependency>

<!-- JWT Library -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- JAXB (fix for javax.xml.bind on Java 11+) -->

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

</dependencies>

