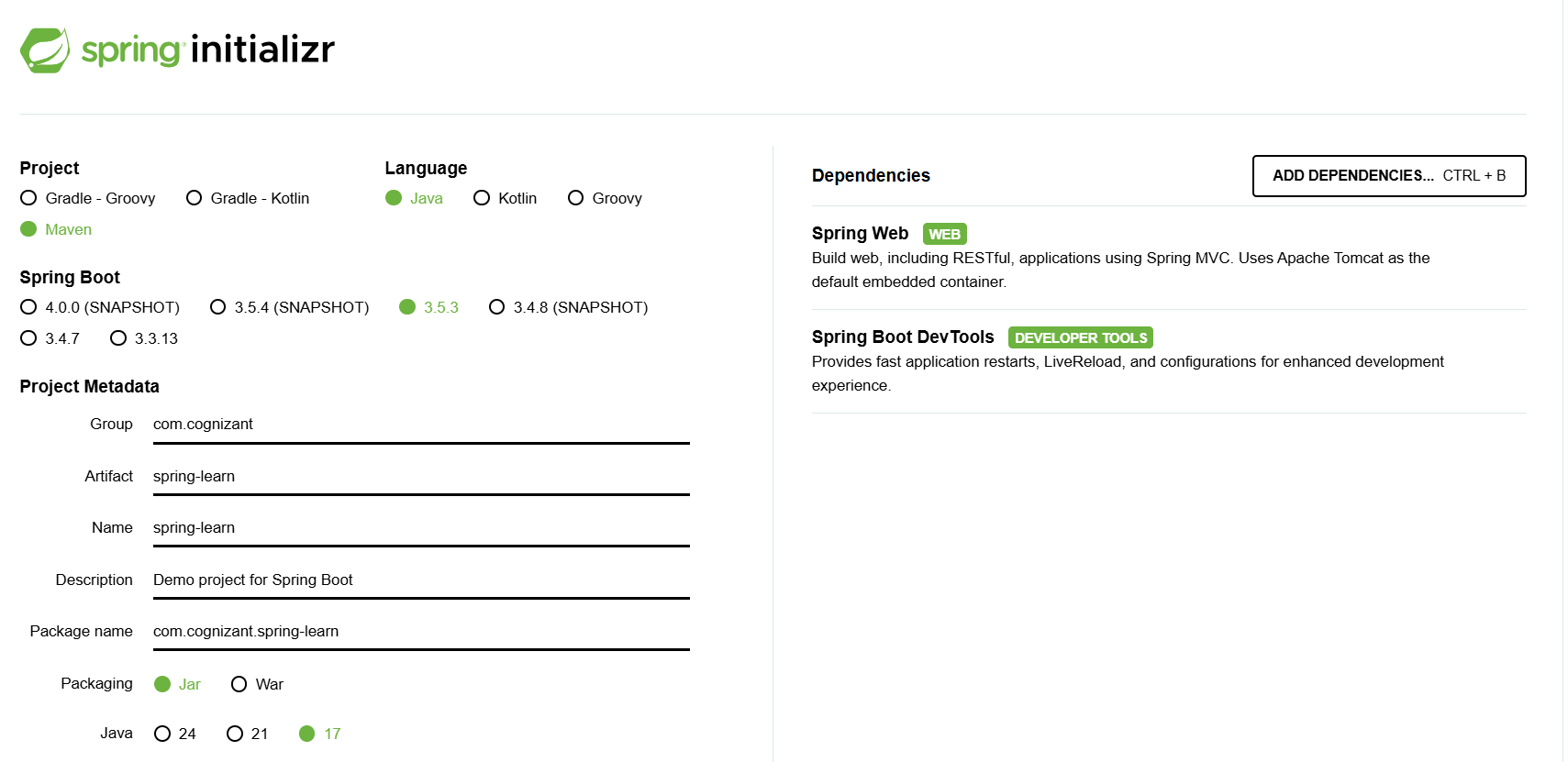
**1.Create a Spring Web Project using Maven**   


**Source Code:**

**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) {

System.***out***.println(">>> SpringLearnApplication Started <<<");

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

}

}

**Output:**>>> SpringLearnApplication Started <<<

>>> SpringLearnApplication Started <<<

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-07-08T21:59:44.414+05:30 INFO 29656 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : Starting SpringLearnApplication using Java 17.0.12 with PID 29656 (C:\Users\prane\Downloads\spring-learn\target\classes started by prane in C:\Users\prane\Downloads\spring-learn)

2025-07-08T21:59:44.420+05:30 INFO 29656 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-08T21:59:44.521+05:30 INFO 29656 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable

2025-07-08T21:59:44.521+05:30 INFO 29656 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'

2025-07-08T21:59:46.050+05:30 INFO 29656 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)

2025-07-08T21:59:46.071+05:30 INFO 29656 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2025-07-08T21:59:46.071+05:30 INFO 29656 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]

2025-07-08T21:59:46.134+05:30 INFO 29656 --- [spring-learn] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2025-07-08T21:59:46.136+05:30 INFO 29656 --- [spring-learn] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1612 ms

2025-07-08T21:59:46.765+05:30 INFO 29656 --- [spring-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2025-07-08T21:59:46.823+05:30 INFO 29656 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'

2025-07-08T21:59:46.841+05:30 INFO 29656 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : Started SpringLearnApplication in 3.242 seconds (process running for 4.374)

**Spring Core 2. Load Country from Spring Configuration XML**

**Source Code:**

**SpringLearnApplication.java:  
   
package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

@SpringBootApplication

**public** **class** SpringLearnApplication {

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

**public** **static** **void** main(String[] args) {

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

*displayCountry*();

}

**public** **static** **void** displayCountry() {

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

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

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

}

}

**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.spring\_learn.Country">

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

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

</bean>

</beans>

**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 + "]";

}

}

**3.Hello World RESTful Web Service**   
 **Source Code:  
Controller.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()");

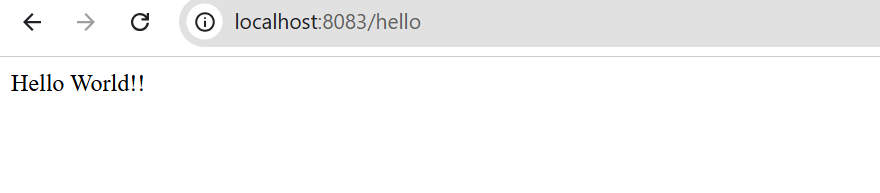
String message = "Hello World!!";

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

**return** message;

}

}

**Output:**

**4.REST - Country Web Service  
  
Source Code:**

**Controller.java:  
package** com.cognizant.spring\_learn.controller;

**import** com.cognizant.spring\_learn.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 = context.getBean("country", Country.**class**);

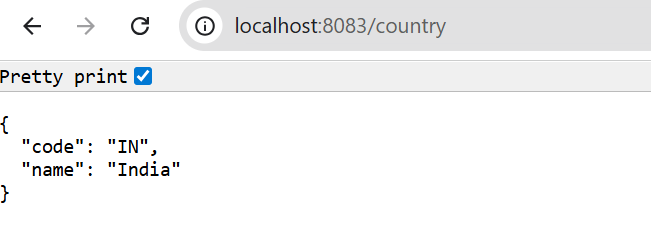
***LOGGER***.info("END - getCountryIndia()");

**return** country;

}

}

**Output:**



**5.REST - Get country based on country code** 

**Source Code:**

SpringLearnApplication.java:  
**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

@SpringBootApplication

**public** **class** SpringLearnApplication {

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

**public** **static** **void** main(String[] args) {

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

}

**public** **static** **void** displayCountry() {

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

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

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

}

}

CountryController.java:  
**package** com.cognizant.spring\_learn.controller;

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

**import** com.cognizant.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.\*;

**import** java.util.List;

@RestController

**public** **class** CountryController {

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

@Autowired

**private** CountryService countryService;

@GetMapping("/countries")

**public** List<Country> getAllCountries() {

***LOGGER***.info("START - getAllCountries()");

List<Country> countries = countryService.getAllCountries();

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

**return** countries;

}

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

**public** Country getCountry(@PathVariable String code) {

***LOGGER***.info("START - getCountry() - {}", code);

Country country = countryService.getCountry(code);

***LOGGER***.info("END - getCountry() - {}", country);

**return** country;

}

}

CountryService.java:  
**package** com.cognizant.spring\_learn.service;

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

**public** Country getCountry(String code) {

***LOGGER***.info("START - getCountry() with code: {}", code);

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

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

**return** countries.stream()

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

.findFirst()

.orElse(**null**); // Optionally: throw new CountryNotFoundException(code)

}

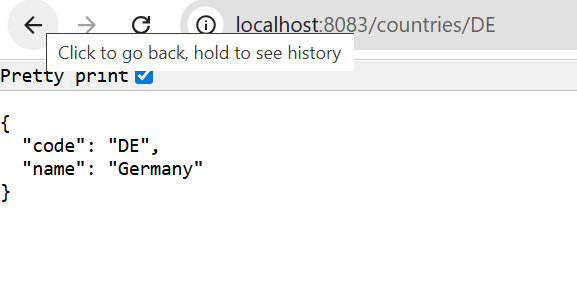
**public** List<Country> getAllCountries() {

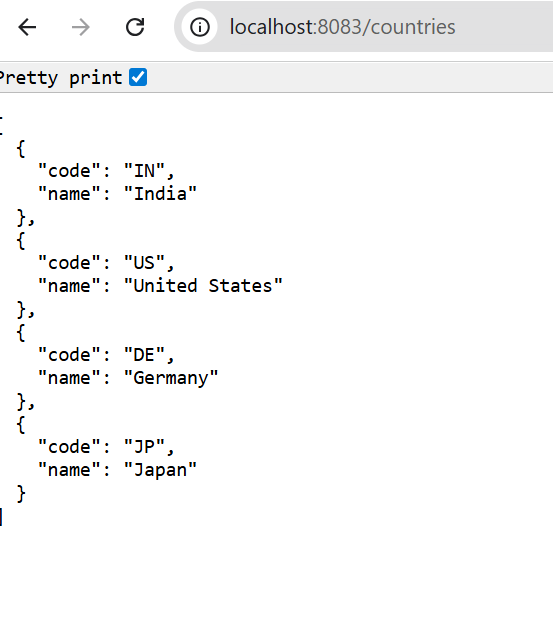
ApplicationContext context = **new** ClassPathXmlApplicationContext("country.xml");

**return** context.getBean("countryList", List.**class**);

}

}

**Output:**



**6.Create authentication service that returns JWT**

**Source Code:**JwtUtil.java:  
**package** com.cognizant.spring\_learn.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** io.jsonwebtoken.security.Keys;

**import** java.util.Date;

**public** **class** JwtUtil {

**private** **static** **final** String ***SECRET\_KEY*** = "8eM74tK9XyQaZzLmN1v3C6TpRqH8WbYk";

**public** **static** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

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

.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour

.signWith(Keys.*hmacShaKeyFor*(***SECRET\_KEY***.getBytes()), SignatureAlgorithm.***HS256***)

.compact();

}

}

SecurityConfig.java:  
**package** com.cognizant.spring\_learn.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

**import** org.springframework.security.web.SecurityFilterChain;

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

**import** org.springframework.security.config.Customizer;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(user);

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

@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();

}

}

AuthenticationController.java:

**package** com.cognizant.spring\_learn.controller;

**import** com.cognizant.spring\_learn.util.JwtUtil;

**import** org.springframework.http.ResponseEntity;

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

**import** java.util.Base64;

**import** jakarta.servlet.http.HttpServletRequest;

@RestController

**public** **class** AuthenticationController {

@GetMapping("/authenticate")

**public** ResponseEntity<?> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

**if** (authHeader != **null** && authHeader.startsWith("Basic ")) {

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

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

String credentials = **new** String(credDecoded);

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

String username = values[0];

String password = values[1];

// Sample validation (In production, validate against DB or service)

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

String token = JwtUtil.*generateToken*(username);

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

}

}

**return** ResponseEntity.*status*(401).body("{\"error\":\"Unauthorized\"}");

}

}Output:  
