**Hands on 1**

**Create a Spring Web Project using Maven**

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

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

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

**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) {

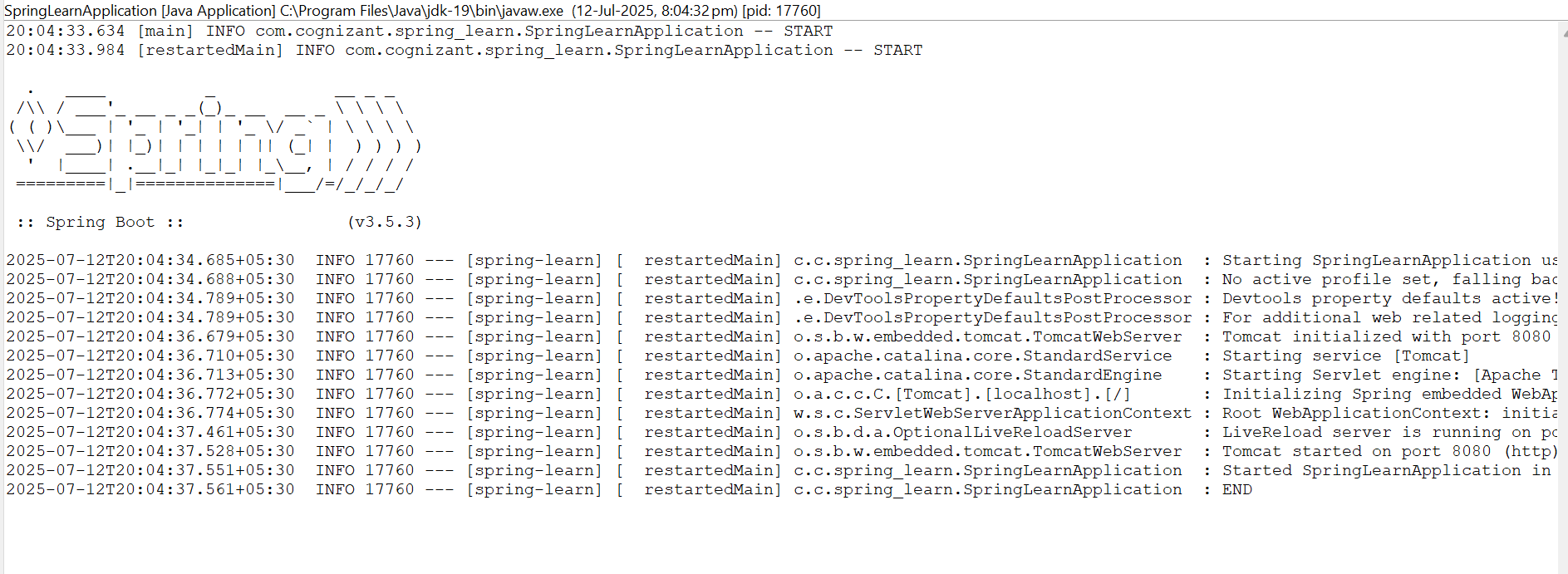
***LOGGER***.info("START");

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

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

}

}

Output:

**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML**

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

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**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");

SpringLearnApplication app = **new** SpringLearnApplication();

app.displayDate();

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

}

**public** **void** displayDate() {

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

ApplicationContext context = **new** ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.**class**);

**try** {

Date date = format.parse("31/12/2018");

***LOGGER***.debug("Parsed Date: {}", date);

} **catch** (ParseException e) {

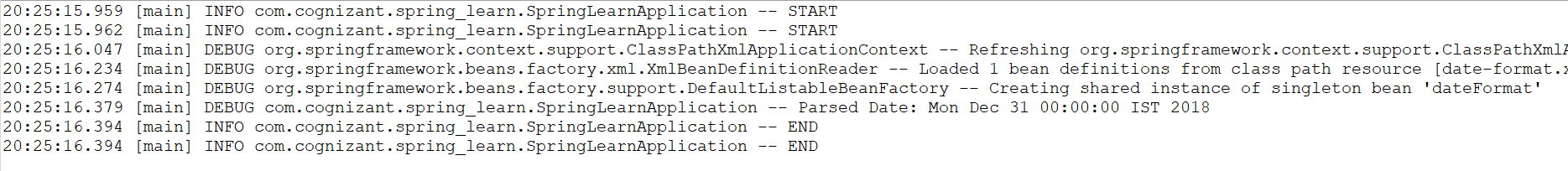
***LOGGER***.error("Error parsing date", e);

}

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

}

}

Output:

**Hello World RESTful Web Service**

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() {

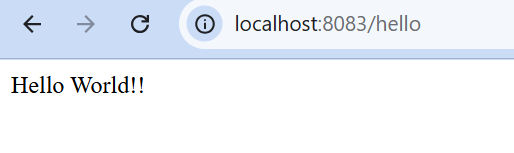
LOGGER.info("START");

LOGGER.info("END");

return "Hello World!!";

}

}

**Output:**

**REST - Country Web Service :**

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");

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

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

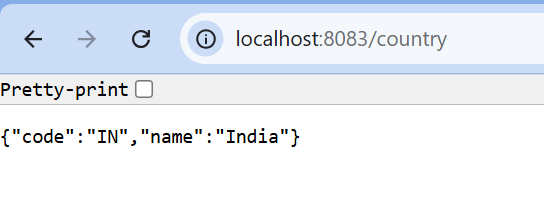
LOGGER.debug("Country: {}", country);

LOGGER.info("END");

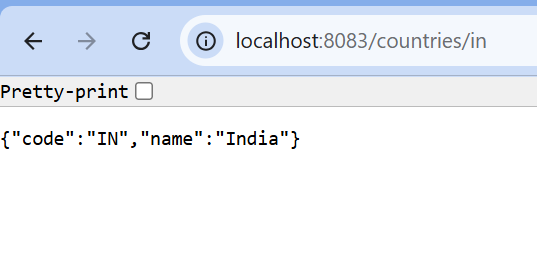
return country;

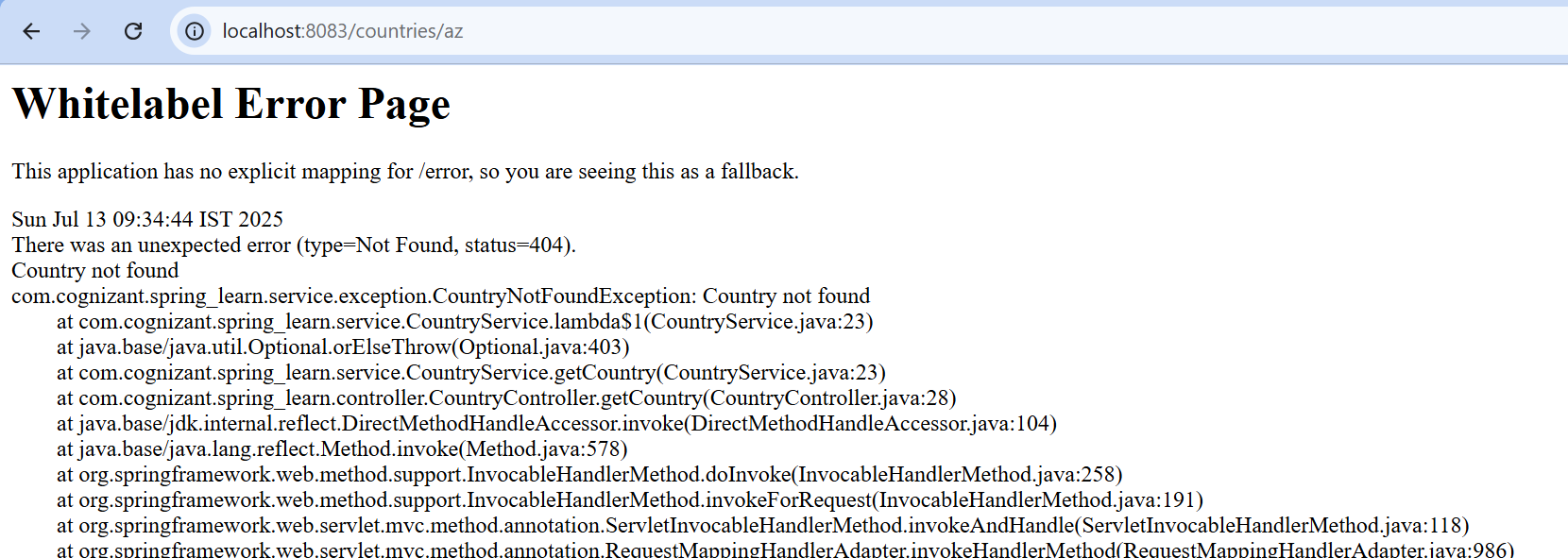
}

}

**Output:**

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

****

****

**Create authentication service that returns JWT**:

JwtUtil.java

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

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.io.Encoders;

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

**import** org.springframework.stereotype.Component;

**import** javax.crypto.SecretKey;

**import** java.util.Date;

@Component

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

**private** **static** **final** **long** ***EXPIRATION\_TIME*** = 10 \* 60 \* 1000; // 10 minutes

// ✅ New non-deprecated way to create a key

**private** **static** **final** SecretKey ***key*** = Jwts.SIG.***HS256***.key().build();

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

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

.subject(username)

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

.expiration(**new** Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))

.signWith(***key***)

.compact();

}

// Optional: for debugging

**public** String getSecretKey() {

**return** Encoders.***BASE64***.encode(***key***.getEncoded());

}

}

AuthController.java

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

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

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

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

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

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

**import** java.nio.charset.StandardCharsets;

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthController {

@Autowired

**private** JwtUtil jwtUtil;

@GetMapping("/authenticate")

**public** Map<String, String> authenticate(HttpServletRequest request) {

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

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

**throw** **new** RuntimeException("Missing or invalid Authorization header");

}

// decode credentials

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

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

String credentials = **new** String(decodedCreds, StandardCharsets.***UTF\_8***);

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

String username = values[0];

String password = values[1];

// validate credentials (hardcoded for now)

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

**throw** **new** RuntimeException("Invalid credentials");

}

// generate JWT token

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

**return** response;

}

}