Week-4

# Spring REST using Spring Boot 3

# Create a Spring Web Project using Maven

 **Open** <https://start.spring.io>.

 **Project** → Maven (or Gradle).

 **Language** → Java.

 **Spring Boot** version → e.g. *3.3.0* (latest GA).

 **Project Metadata**

* *Group* = com.example
* *Artifact* = demo

 **Dependencies**

* Choose *Spring Web* (adds spring‑boot‑starter‑web) to spin up a REST server.

 Click **Generate** → a .zip file (demo.zip) downloads.

 Unzip → open in your IDE.

## Project Structure

demo

├─ mvnw / mvnw.cmd → Maven wrapper (or gradlew for Gradle)

├─ pom.xml → Explicit dependencies + Spring Boot parent

└─ src

└─ main

├─ java

│ └─ com.example.demo

│ └─ DemoApplication.java ← main class

└─ resources

├─ application.properties ← default empty

└─ static / templates ← web assets (optional)

### @SpringBootApplication

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication // <— SINGLE annotation = 3 in one

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

@SpringBootApplication =

* @Configuration  (beans defined here)
* @EnableAutoConfiguration  (boot guesses sensible defaults)
* @ComponentScan  (finds controllers/services in same package or below)

## @RestController

public class GreetingController {

@GetMapping("/hello")

public String hello() {

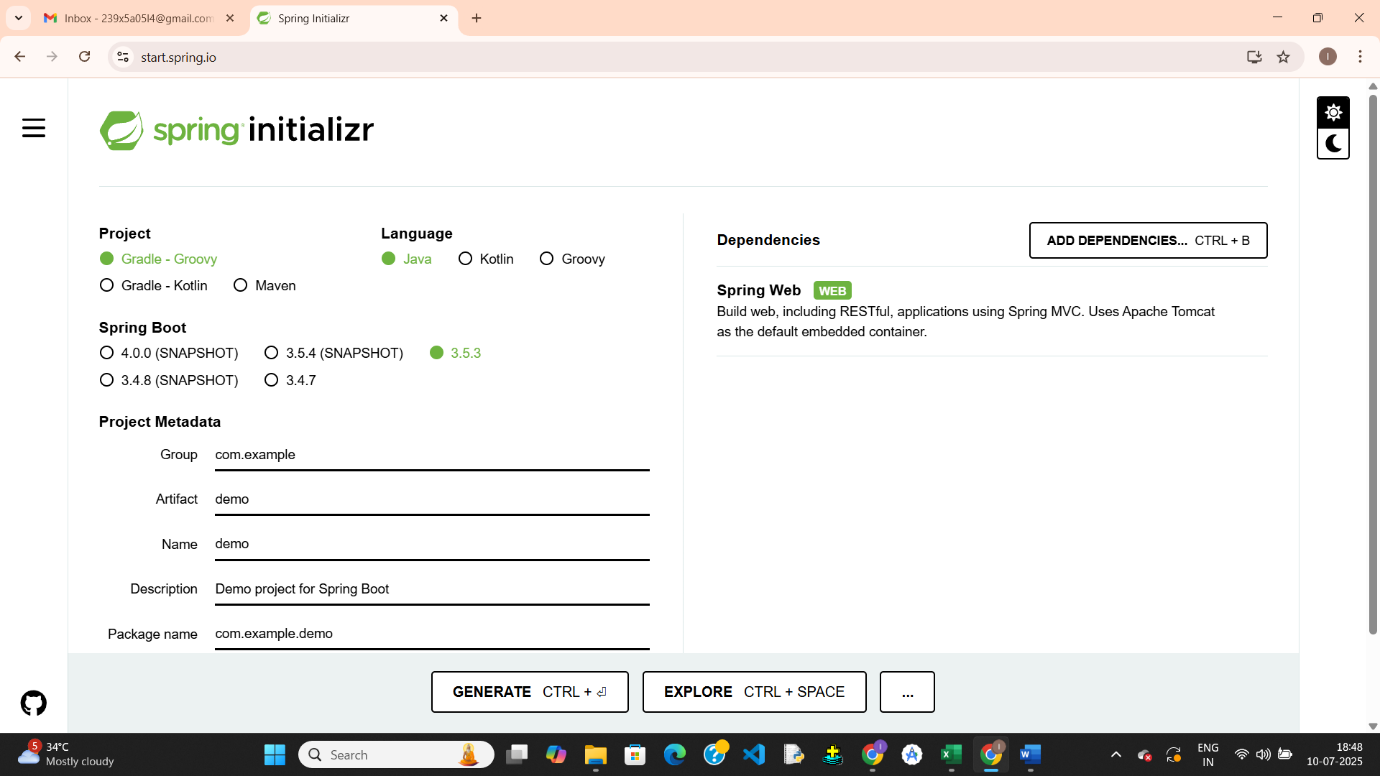
return "Hello, Spring Boot!";

}

}

Output:

Hello, Spring Boot!



2)Hello World RESTful Web Service

**Main class**

package com.example.restgetdemo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestGetDemoApplication {

public static void main(String[] args) {

SpringApplication.run(RestGetDemoApplication.class, args);

}

}

**Model class**

package com.example.restgetdemo.model;

public class Greeting {

private long id;

private String message;

public Greeting(long id, String message) {

this.id = id;

this.message = message;

}

public long getId() {

return id;

}

public String getMessage() {

return message;

}

}

**Controller class**

package com.example.restgetdemo.controller;

import com.example.restgetdemo.model.Greeting;

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

import java.util.Arrays;

import java.util.List;

@RestController

public class GreetingController {

// Basic GET - no parameters

@GetMapping("/hello")

public String sayHello() {

return "Hello from Spring Boot!";

}

// GET with @PathVariable

@GetMapping("/greet/{name}")

public Greeting greetUser(@PathVariable String name) {

return new Greeting(1, "Hello, " + name + "!");

}

// GET - return array of objects

@GetMapping("/greetings")

public List<Greeting> getGreetings() {

return Arrays.asList(

new Greeting(1, "Hello!"),

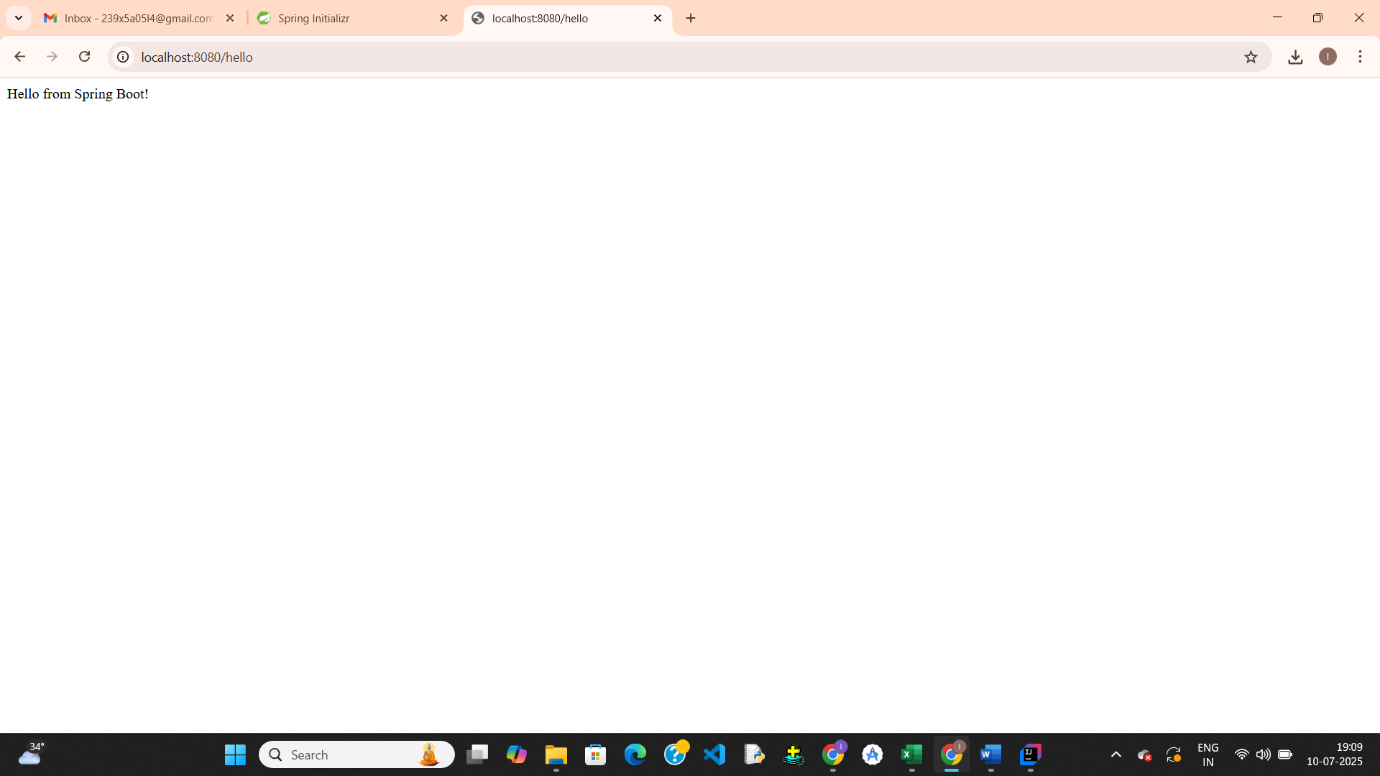
new Greeting(2, "Bonjour!"),

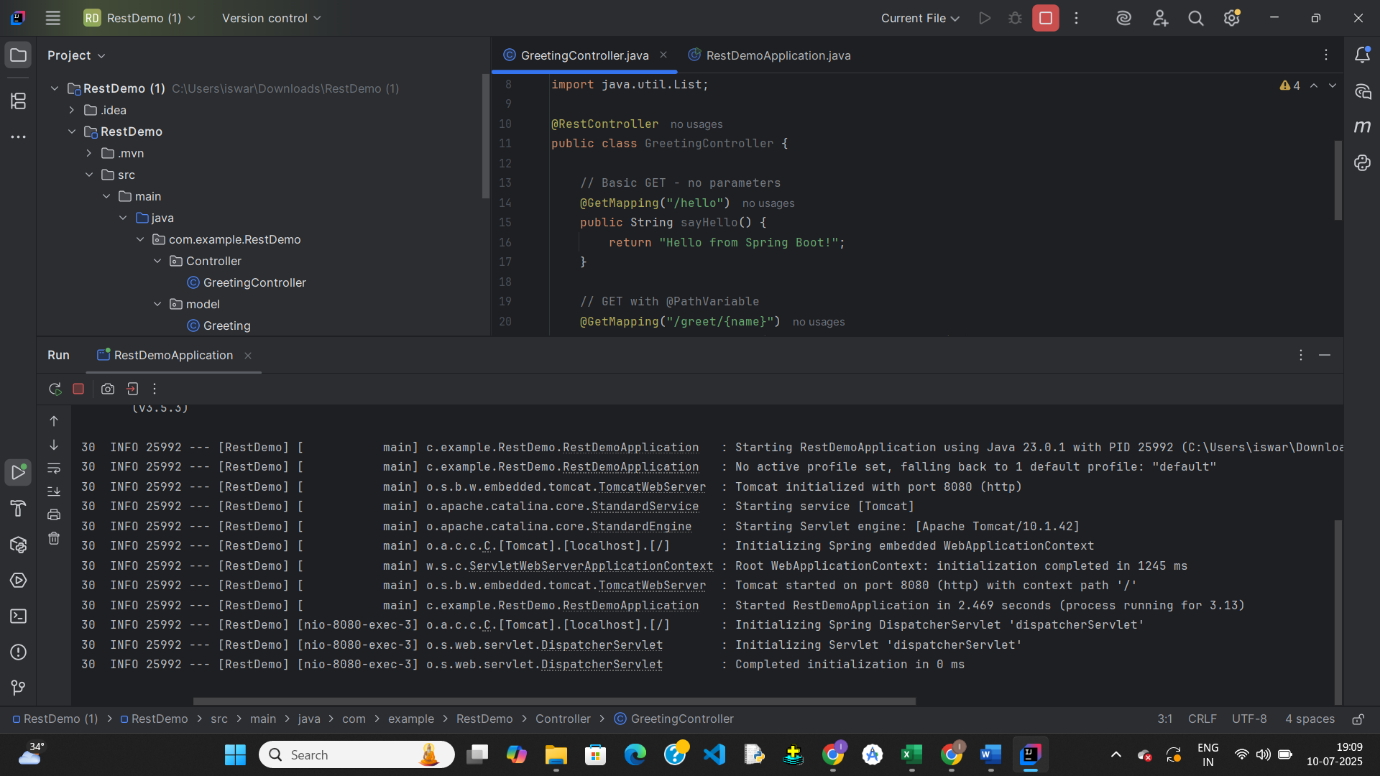
new Greeting(3, "Hola!")

);

}

}





3) REST - Country Web Service

**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">

<!-- Single “India” bean -->

<bean id="country"

class="com.example.RestDemo.model.Country">

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

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

</bean>

<!-- List of all countries -->

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

<constructor-arg>

<list>

<bean class="com.example.RestDemo.model.Country">

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

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

</bean>

<bean class="com.example.RestDemo.model.Country">

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

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

</bean>

<bean class="com.example.RestDemo.model.Country">

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

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

</bean>

<bean class="com.example.RestDemo.model.Country">

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**Main class**

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) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Controller class**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

import java.util.List;

@RestController

public class CountryController {

@RequestMapping("/country")

public Country getCountryIndia() {

ApplicationContext context =

new ClassPathXmlApplicationContext("country.xml");

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

}

@GetMapping("/countries")

public List<Country> getAllCountries() {

ApplicationContext context =

new ClassPathXmlApplicationContext("country.xml");

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

}

}

**Model class**

package com.cognizant.springlearn.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;

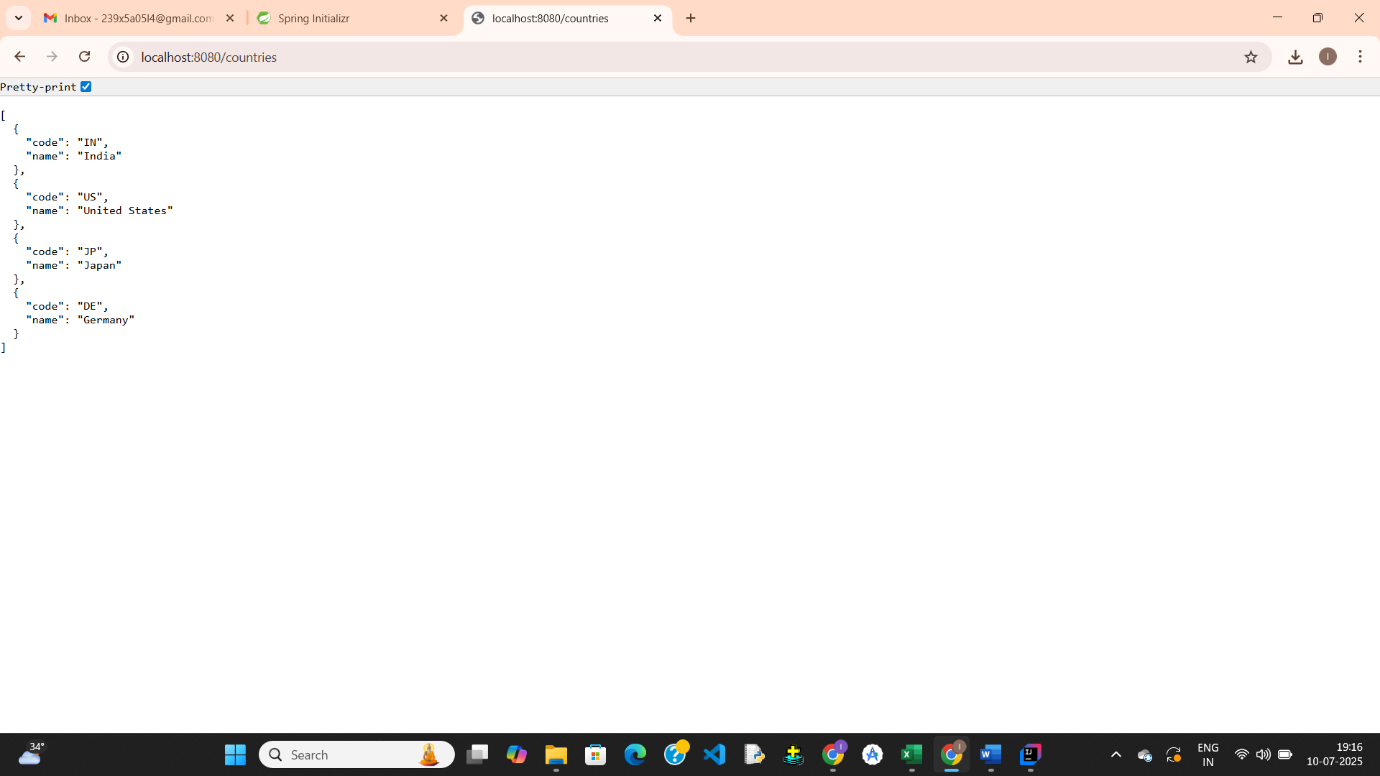
}

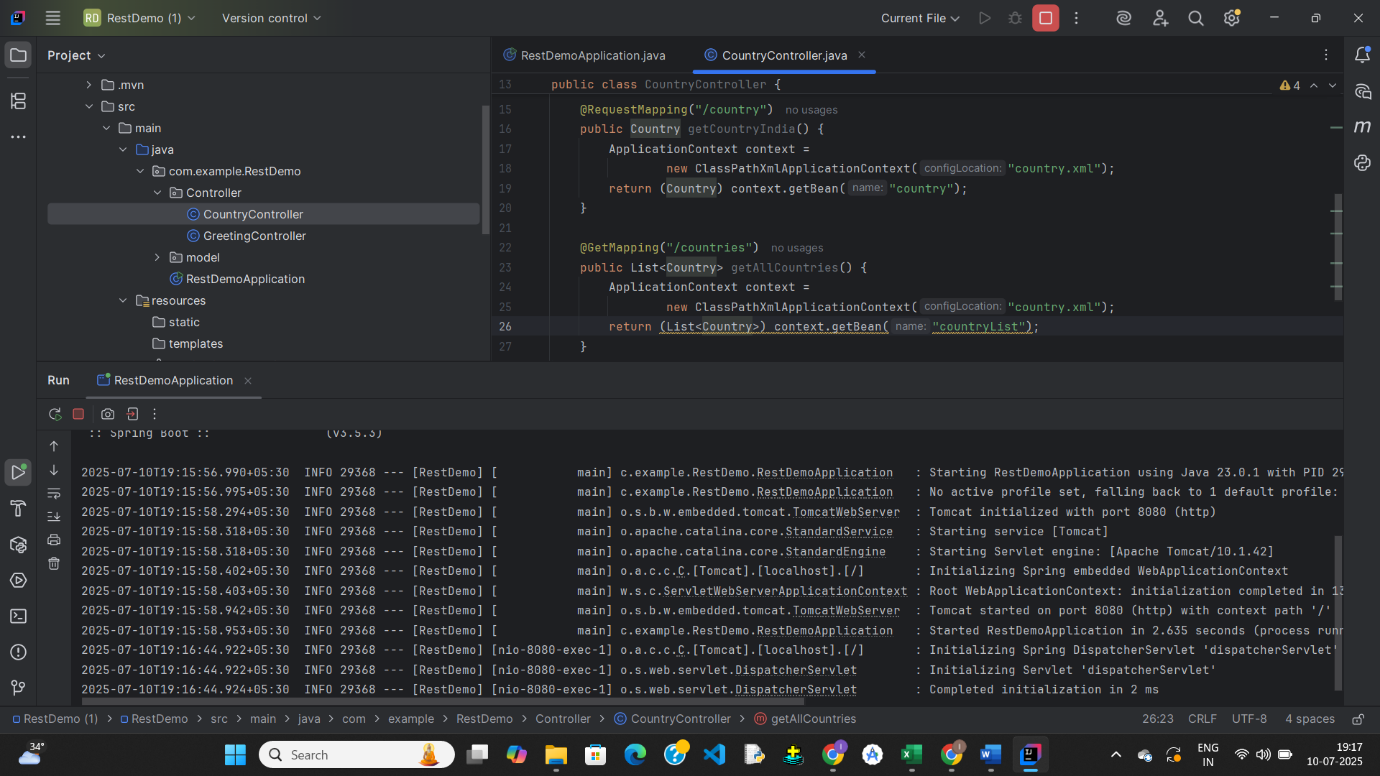
public void setName(String name) {

this.name = name;

}

}





## 4) REST - Get country based on country code

**Model Class**

package com.cognizant.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; }

}

**CountryService**

package com.cognizant.spring\_learn.service;

import com.cognizant.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) throws Exception {

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

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

return countryList.stream()

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

.findFirst()

.orElseThrow(() -> new Exception("Country Not Found: " + code));

}

}

**Country Controller**

package com.cognizant.spring\_learn.controller;

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

import com.cognizant.spring\_learn.service.CountryService;

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) throws Exception {

return countryService.getCountry(code);

}

}

**Main Class**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

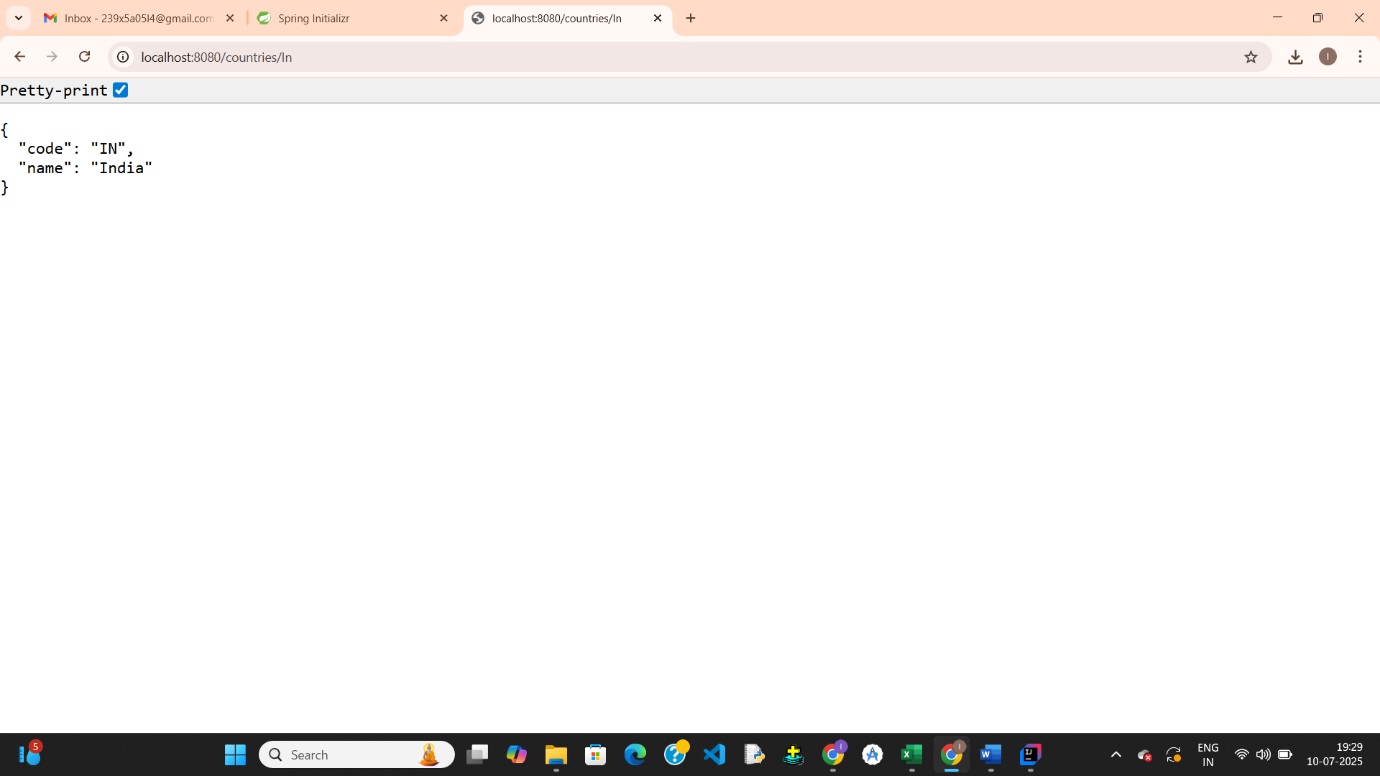
public class SpringLearnApplication {

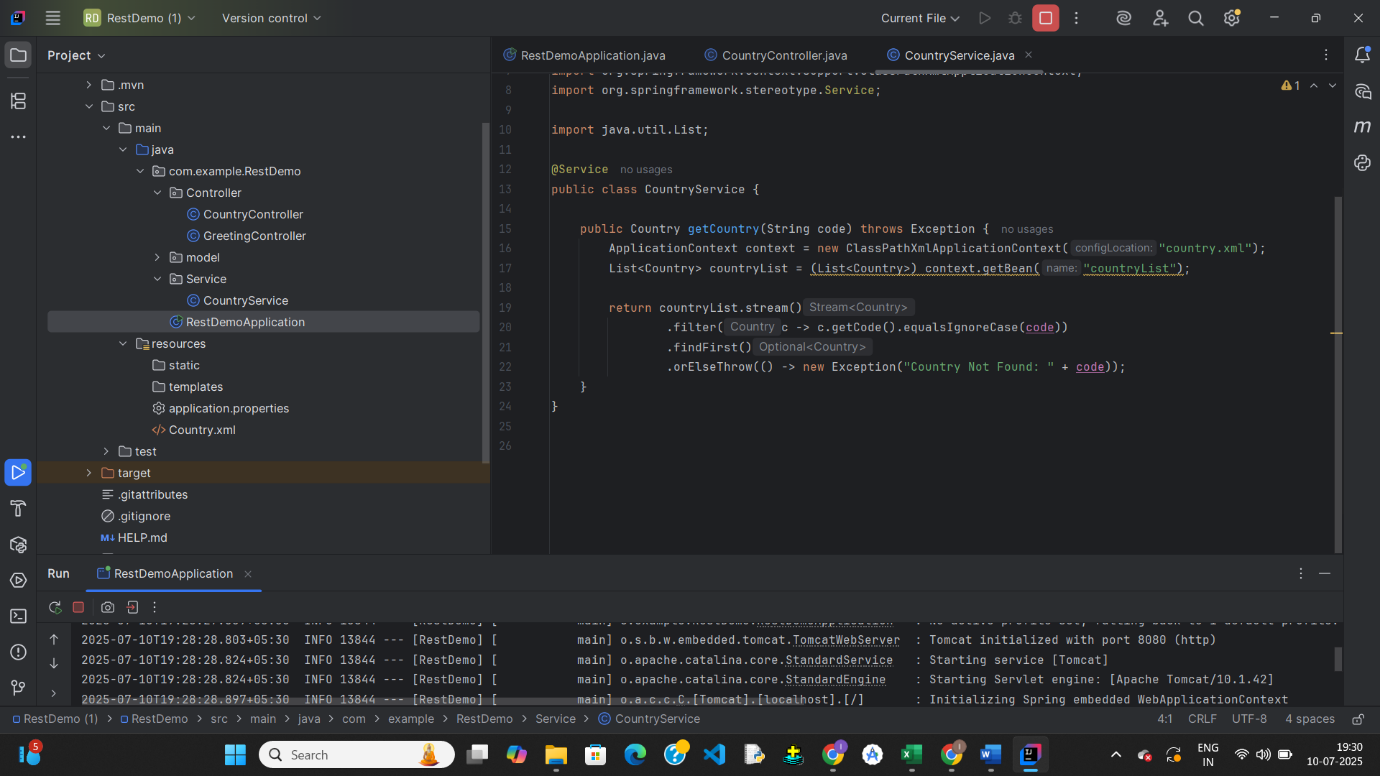
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}





# 5) Create authentication service that returns JWT

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.example</groupId>  
 <artifactId>RestDemo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>RestDemo</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-api</artifactId>  
 <version>0.11.5</version>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-impl</artifactId>  
 <version>0.11.5</version>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-jackson</artifactId>  
 <version>0.11.5</version>  
 <scope>runtime</scope>  
 </dependency>  
  
  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Application.properties**

spring.application.name=RestDemo  
  
spring.autoconfigure.exclude=\org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration,\  
org.springframework.boot.autoconfigure.security.servlet.ManagementWebSecurityAutoConfiguration

**JwtUtil**

package com.example.RestDemo.util;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
  
import java.security.Key;  
import java.util.Date;  
  
public class JwtUtil {  
  
 private static final Key *SECRET\_KEY* = Keys.*secretKeyFor*(SignatureAlgorithm.*HS256*);  
  
 public static String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.*currentTimeMillis*()))  
 .setExpiration(new Date(System.*currentTimeMillis*() + 60 \* 60 \* 1000)) // 1 hour  
 .signWith(*SECRET\_KEY*)  
 .compact();  
 }  
}

**AuthController**

package com.example.RestDemo.Controller;  
  
  
import com.example.RestDemo.model.JwtResponse;  
import com.example.RestDemo.util.JwtUtil;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
  
@RestController  
public class AuthController {  
  
 @GetMapping("/authenticate")  
 public ResponseEntity<?> authenticate(@RequestHeader(HttpHeaders.*AUTHORIZATION*) String authHeader) {  
 try {  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(credDecoded);  
  
 // defensive split  
 String[] values = credentials.split(":", 2);  
 if (values.length != 2) {  
 return ResponseEntity.*status*(400).body("Invalid basic auth format");  
 }  
  
 String username = values[0];  
 String password = values[1];  
  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = JwtUtil.*generateToken*(username);  
 return ResponseEntity.*ok*(new JwtResponse(token));  
 } else {  
 return ResponseEntity.*status*(403).body("Invalid Credentials");  
 }  
 } catch (Exception e) {  
 e.printStackTrace(); // log actual error  
 return ResponseEntity.*status*(500).body("Internal Server Error: " + e.getMessage());  
 }  
 }  
  
}

**JwtResponse**

package com.example.RestDemo.model;  
  
  
  
public class JwtResponse {  
 private String token;  
  
 public JwtResponse(String token) {  
 this.token = token;  
 }  
  
 public String getToken() {  
 return token;  
 }  
  
 public void setToken(String token) {  
 this.token = token;  
 }  
}

**main class**

package com.example.RestDemo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class RestDemoApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(RestDemoApplication.class, args);  
 }  
  
}

