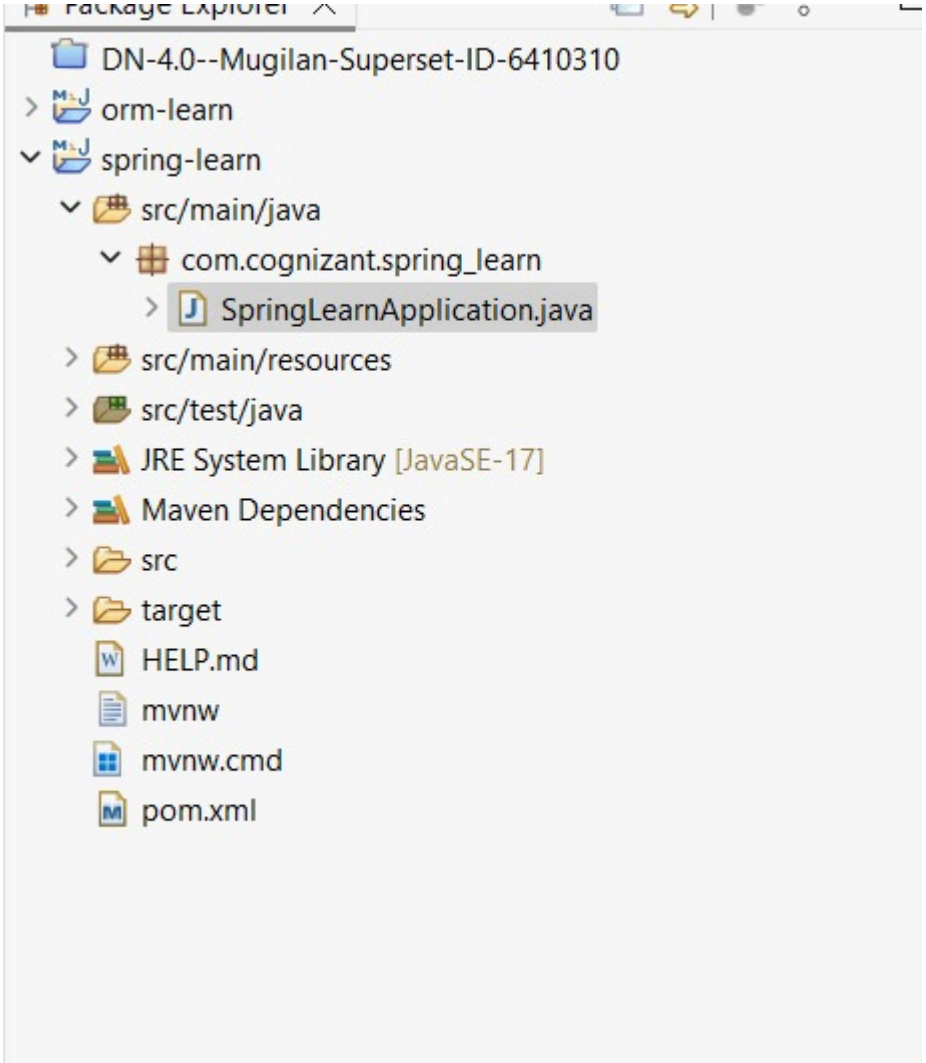
Create a Spring Web Project using Maven:



Proxy Build:

```
PS C:\Users\think\Downloads\spring-learn\spring-learn> mvn clean package "-Dhttp.proxyHost=proxy.cognizant.com" "-Dhttp.
proxyPort=6050" "-Dhttps.proxyHost=proxy.cognizant.com" "-Dhttps.proxyPort=6050" "-Dhttp.proxyUser=123456"
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.cognizant:spring-learn >----------------------
[INFO] Building spring-learn 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
```

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.289 s -- in com.cognizant.spring_learn.SpringLe
arnApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ spring-learn ---
[INFO] Building jar: C:\Users\think\Downloads\spring-learn\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.5.3:repackage (repackage) @ spring-learn ---
[INFO] Replacing main artifact C:\Users\think\Downloads\spring-learn\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar
 with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\think\Downloads\spring-learn\spring-learn\target\spring-learn-
0.0.1-SNAPSHOT.jar.original
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.044 s
[INFO] Finished at: 2025-07-11T11:10:04+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\think\Downloads\spring-learn\spring-learn>
```

SpringLearnApplication.java:

```
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);
		System.out.println(" Spring Boot App Started...");
	}

}
```

Output:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-11T11:20:27.046+05:30  INFO 7348 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Starting SpringLearnApplication using Java 17.0.11 with PID 7348 (C:\Users\think
2025-07-11T11:20:27.048+05:30  INFO 7348 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : No active profile set, falling back to 1 default profile: "default"
2025-07-11T11:20:27.225+05:30  INFO 7348 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-11T11:20:27.226+05:30  INFO 7348 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-11T11:20:27.226+05:30  INFO 7348 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-11T11:20:27.242+05:30  INFO 7348 --- [spring-learn] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-11T11:20:27.242+05:30  INFO 7348 --- [spring-learn] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 192 ms
2025-07-11T11:20:27.309+05:30  INFO 7348 --- [spring-learn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-11T11:20:27.317+05:30  INFO 7348 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-07-11T11:20:27.321+05:30  INFO 7348 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 0.296 seconds (process running for 425.481)
2025-07-11T11:20:27.323+05:30  INFO 7348 --- [spring-learn] [  restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
Spring Boot App Started...
```

Spring Core – Load Country from Spring Configuration XML:

SpringLearnApllication.java:

```
packagecom.cognizant.spring_learn;
importorg.slf4j.Logger;
importorg.slf4j.LoggerFactory;
importorg.springframework.context.support.ClassPathXmlApplicationContext;
publicclass SpringLearnApplication {
  privatestatic final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);
  publicstatic void main(String[] args) {
    LOGGER.info("START");
    displayCountry();
    LOGGER.info("END");
  }
  publicstatic void displayCountry() {
    try(ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("country.xml")) {
      Country country = context.getBean("country", Country.class);
      LOGGER.debug("Country : {}", country.toString());
    }
  }
}
```

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 + "]";
  }
}
```

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>
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> SpringLearnApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (11-Jul-2025, 4:43:11 pm – 4:43:14 pm elapsed: 0:00:02.281) [pid: 31724]
16:43:13.431 [main] INFO  c.c.s.SpringLearnApplication - START
16:43:13.541 [main] DEBUG o.s.c.s.ClassPathXmlApplicationContext - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@3b07a0d6
16:43:13.797 [main] DEBUG o.s.b.f.xml.XmlBeanDefinitionReader - Loaded 1 bean definitions from class path resource [country.xml]
16:43:13.857 [main] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'country'
16:43:13.857 [main] DEBUG com.cognizant.spring_learn.Country - Inside Country Constructor.
16:43:13.950 [main] DEBUG com.cognizant.spring_learn.Country - Inside setCode()
16:43:13.950 [main] DEBUG com.cognizant.spring_learn.Country - Inside setName()
16:43:13.977 [main] DEBUG c.c.s.SpringLearnApplication - Country : Country [code=IN, name=India]
16:43:14.071 [main] DEBUG o.s.c.s.ClassPathXmlApplicationContext - Closing org.springframework.context.support.ClassPathXmlApplicationContext@3b07a0d6, started on Fri
16:43:14.075 [main] INFO  c.c.s.SpringLearnApplication - END
```

Hello World RESTful Web Service:

HelloController.java:

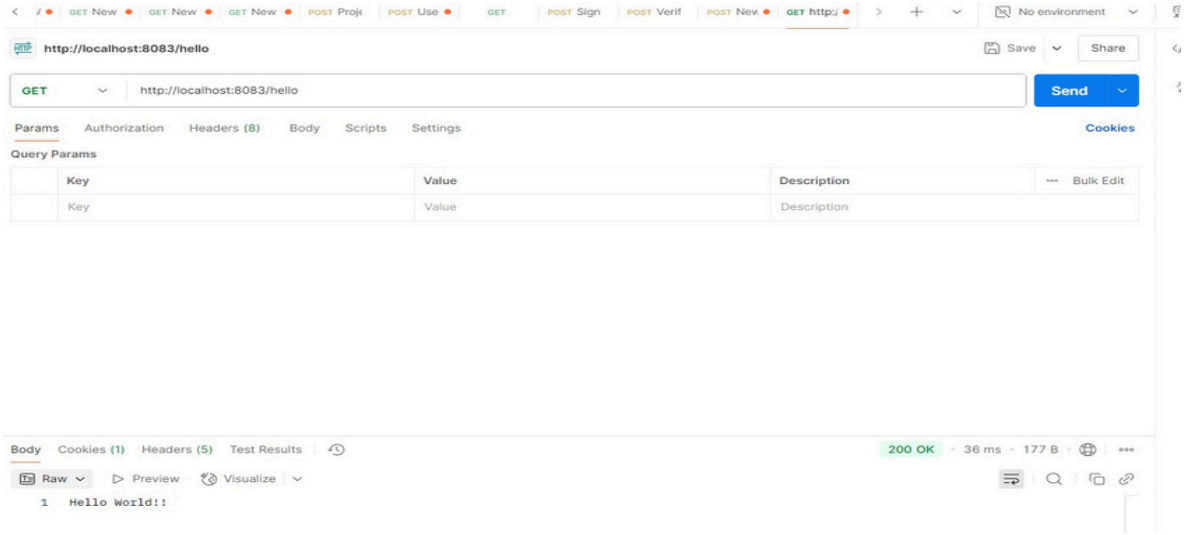
```
package com.cognizant.spring_web.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloController {
  private static final Logger LOGGER = LoggerFactory.getLogger(HelloController.class);
  @GetMapping("/hello")
  public String sayHello() {
     LOGGER.info("START");
     LOGGER.info("END");
     return "Hello World!!";
  }
}
```

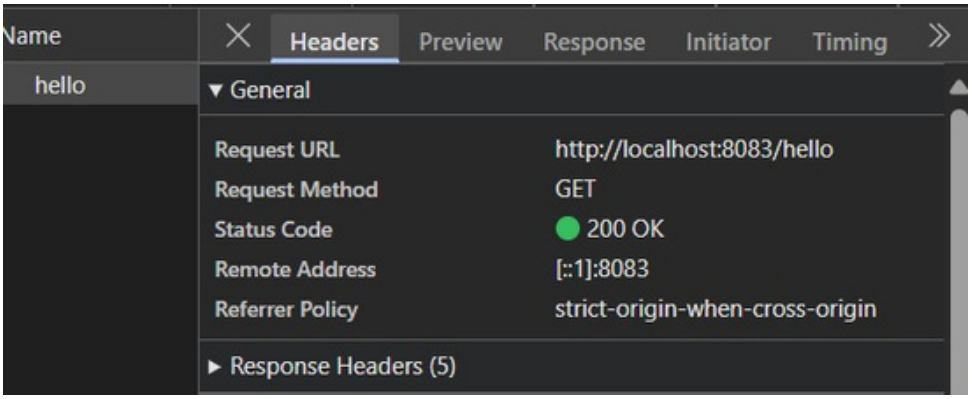
SpringWebApplication.java:

```
package com.cognizant.spring_web;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
@SpringBootApplication
public class SpringWebApplication {
  private static final Logger LOGGER = LoggerFactory.getLogger(SpringWebApplication.class);
 public static void main(String[] args) {
      LOGGER.info("START");
      SpringApplication.run(SpringWebApplication.class, args);
      LOGGER.info("END");
  }
}
```

Port:
8083

Output:





REST COUNTRY WEB SERVICE:

Country.java:

```
package com.cognizant.spring_web.model;
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;
  }
}
```

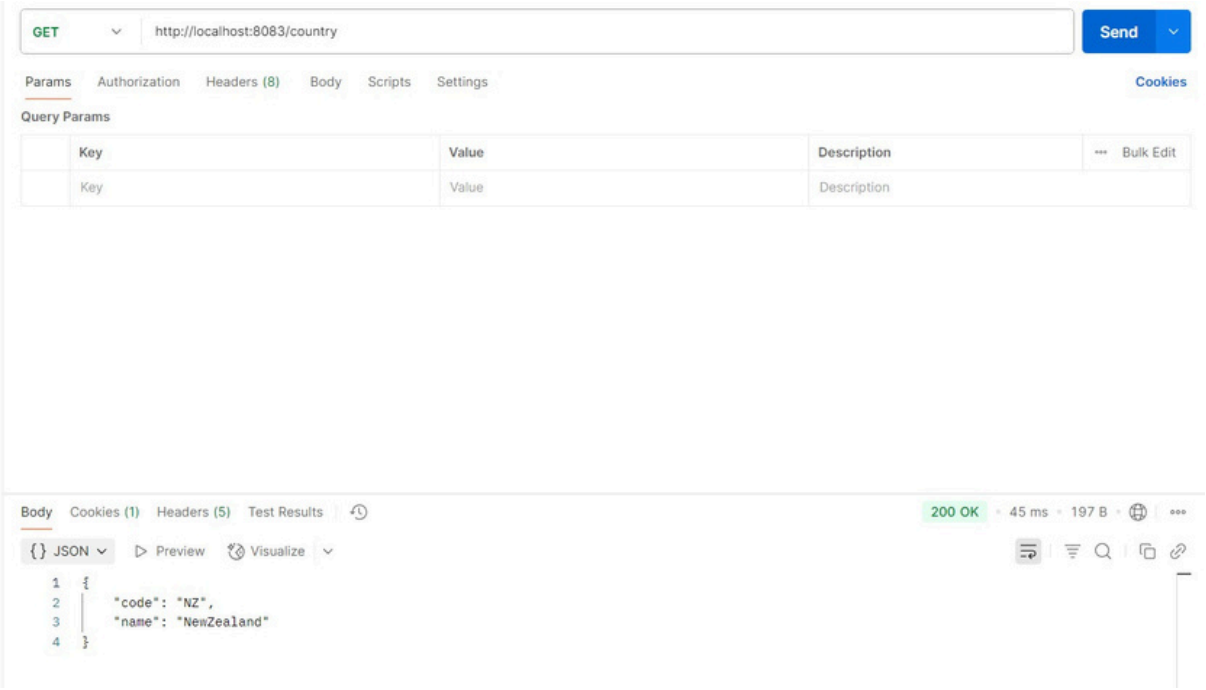
CountryController.java:

```
package com.cognizant.spring_web.controller;
import com.cognizant.spring_web.model.Country;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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");
     ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
     Country country = (Country) context.getBean("in");
     context.close();
     LOGGER.info("END");
     return country;
  }
}
```

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= "in"class="com.cognizant.spring_web.model.Country"
     <property name="code" value="NZ"/>
     <property name="name" value="NewZealand"/>
 </bean>
</beans>
```

Output:



Get all countries:

country.xml:

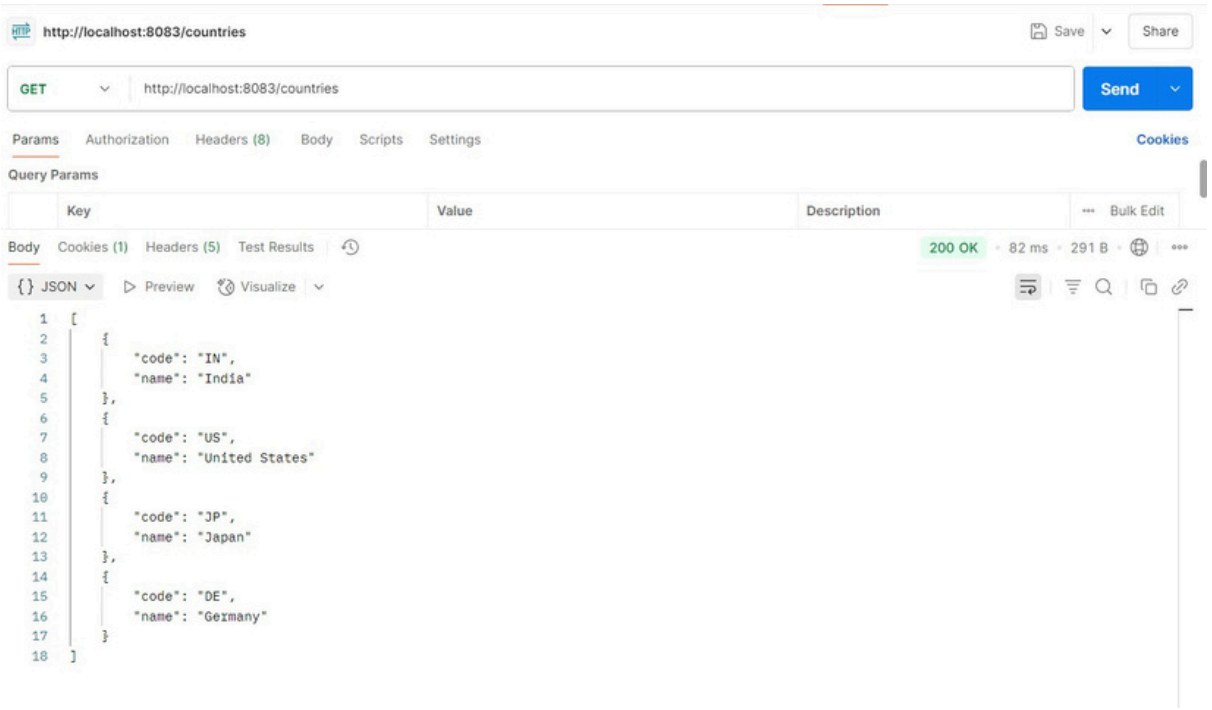
```
<bean id="countryList" class="java.util.ArrayList">
    <constructor-arg>
      <list>
        <bean class="com.cognizant.spring_web.model.Country">
          <property name="code" value="IN"/>
          <property name="name" value="India"/>
        </bean>
        <bean class="com.cognizant.spring_web.model.Country">
          <property name="code" value="US"/>
          <property name="name" value="United States"/>
        </bean>
```

```
            <bean class="com.cognizant.spring_web.model.Country"
                <property name="code" value="JP"/>
                <property name="name" value="Japan"/>
            </bean>
            <bean class="com.cognizant.spring_web.model.Country"
                <property name="code" value="DE"/>
                <property name="name" value="Germany"/>
            </bean>
        </list>
    </constructor-arg>
  </bean>
```

Endpoint:

```
@GetMapping("/countries")
  public List<Country> getAllCountries() {
      LOGGER.info("START");
      ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
      List<Country> countryList = (List<Country>) context.getBean("countryList");
      context.close();
      LOGGER.info("END");
      return countryList;
  }
```

Output:



Create authentication service that returns JWT:

AuthController.java:

```
package com.cognizant.jwt_auth.controller;
import com.cognizant.jwt_auth.util.JwtUtil;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.Base64;
import java.util.Collections;
@RestController
public class AuthController {
  @GetMapping("/authenticate")
  public ResponseEntity<?> authenticate(
        @RequestHeader(value = "Authorization", required = false) String authHeader) {
    if (authHeader==  null) {
      return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                    .body("Missing Authorization header");
    }
    try {
      String[]credentials = extractAndDecodeHeader(authHeader);
      Stringusername = credentials[0];
      Stringpassword = credentials[1];
      if (isValidUser(username, password)) {
        Stringtoken = JwtUtil.generateToken(username);
        return ResponseEntity.ok(Collections.singletonMap("token", token));
      }
      return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                    .body("Invalid credentials");
    } catch (Exception e) {
      return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                    .body("Error" + e.getMessage());
    }
  }
  private boolean isValidUser(String username, String password) {
    return "Mugilan".equals(username) && "Hero".equals(password);
  }
  private String[]extractAndDecodeHeader(String header) {
    if (!header.startsWith("Basic ")) {
      throw new RuntimeException("Invalid Authorization");
    }
    byte[]base64Token = Base64.getDecoder().decode(header.substring(6));
    Stringtoken=   new String(base64Token);
    String[]parts=token.split(":", 2);
    if (parts.length!=2) {
      throw new RuntimeException("Invalid format");
    }
    return parts;
  }
}
```

Jwtutil.java:

```
package com.cognizant.jwt_auth.util;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import java.security.Key;
import java.util.Date;
publicclass  JwtUtil {
  privatestatic final  StringSECRET = "my-key";
  privatestatic final  KeyKEY= Keys.hmacShaKeyFor( SECRET.getBytes());
  privatestatic final long  EXPIRY_MS = 60 * 60 * 1000;
  private JwtUtil() {}
  publicstatic  StringgenerateToken(String username) {
    return Jwts.builder()
            .setSubject(username)
            .setIssuedAt(new Date())
            .setExpiration(new Date(System.currentTimeMillis() + EXPIRY_MS))
            .signWith(KEY,SignatureAlgorithm.HS256)
            .compact();
  }
}
```