Spring REST using Spring Boot 3

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:

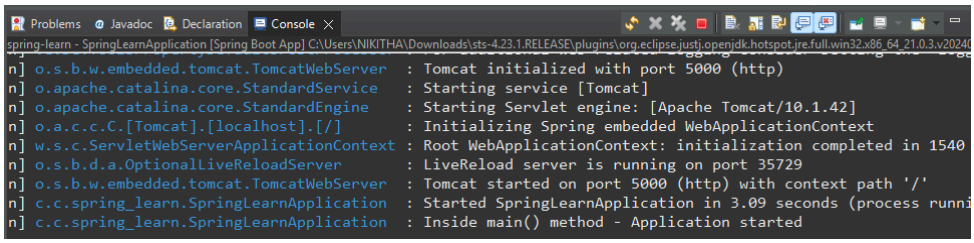
1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation
6. pom.xml
    a.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

SpringLearnApplication.java

```
package com.cognizant.spring_learn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringLearnApplication {

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

    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication.class, args);
        LOGGER.info("Inside main() method - Application started");
```

```
    }
}
```

Output:



2. 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:

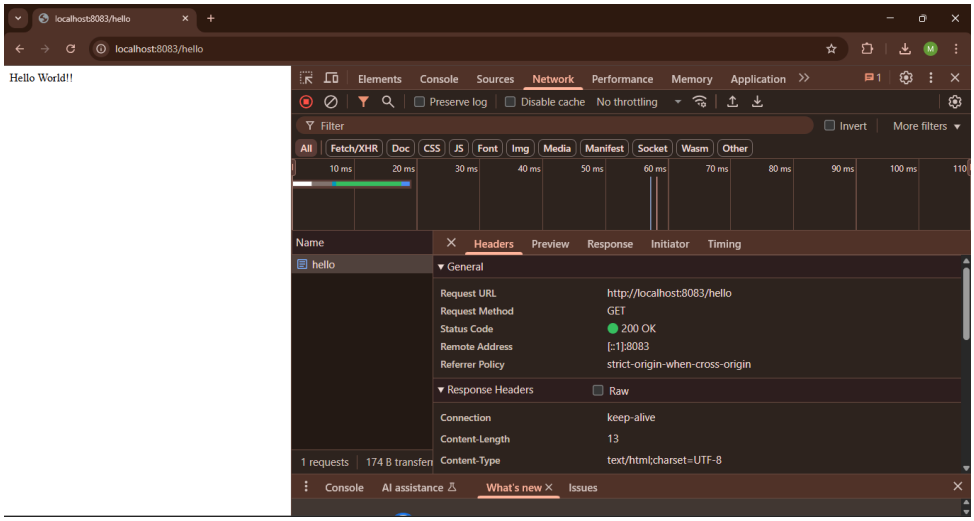
- In network tab of developer tools show the HTTP header details received
- In postman click on "Headers" tab to view the HTTP header details received

Output:

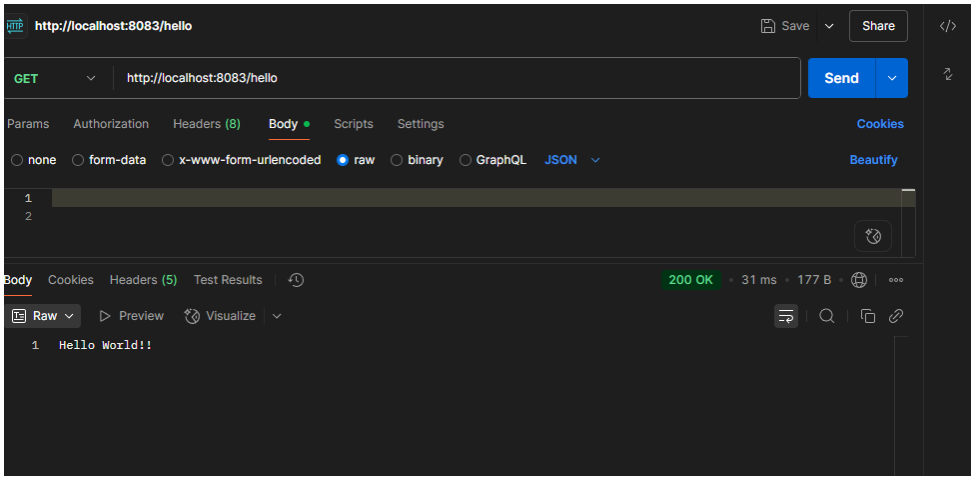
In Chrome:

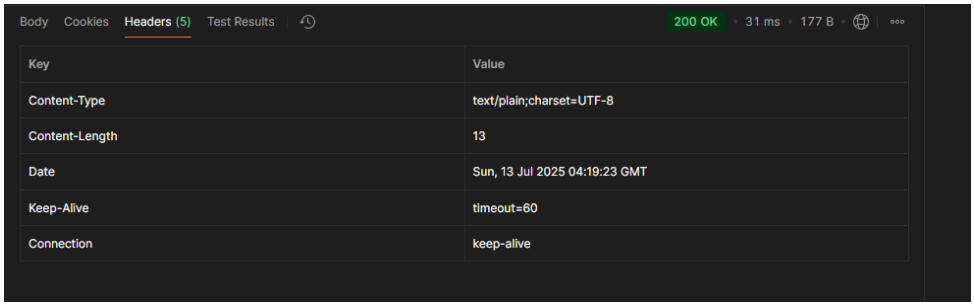

localhost:8083/hello
localhost:8083/hello
Hello World!!


Search
ENG
IN
09:47
13-07-2025


localhost:8083/hello
localhost:8083/hello
Hello World!!
Elements
Console
Sources
Network
Performance
Memory
Application
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
Name
hello
Headers
Preview
Response
Initiator
Timing
General
Request URL
http://localhost:8083/hello
Request Method
GET
Status Code
200 OK
Remote Address
[::1]:8083
Referrer Policy
strict-origin-when-cross-origin
Response Headers
Raw
Connection
keep-alive
Content-Length
13
Content-Type
text/html;charset=UTF-8
1 requests
174 B transferr
Console
AI assistance
What's new
Issues

In Postman:


http://localhost:8083/hello
Save
Share
GET
http://localhost:8083/hello
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Body
Cookies
Headers (5)
Test Results
200 OK
31 ms
177 B
Raw
Preview
Visualize
Hello World!!



3. 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

Country.java

```
package com.cognizant.spring_learn.model;

public class Country {
    private String code;
    private String name;

    // No-arg constructor
    public Country() {}

    // All-arg constructor
    public Country(String code, String name) {
        this.code = code;
        this.name = name;
    }

    // Getters & Setters
```

```
    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;
    }
}
```

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_learn.model.Country">
        <property name="code" value="IN" />
        <property name="name" value="India" />
    </bean>

</beans>
```

CountryController.java

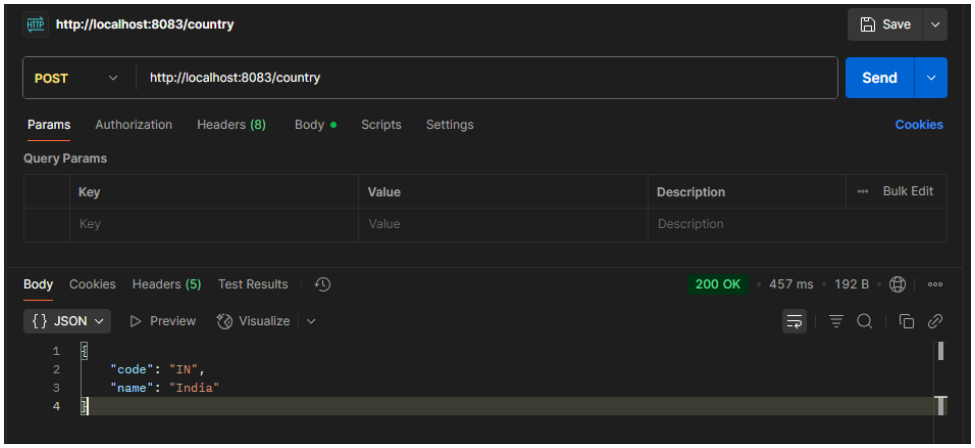
```
package com.cognizant.spring_learn.controller;

import com.cognizant.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("country.xml");
        Country country = (Country) context.getBean("in");
        LOGGER.info("END - getCountryIndia()");
```

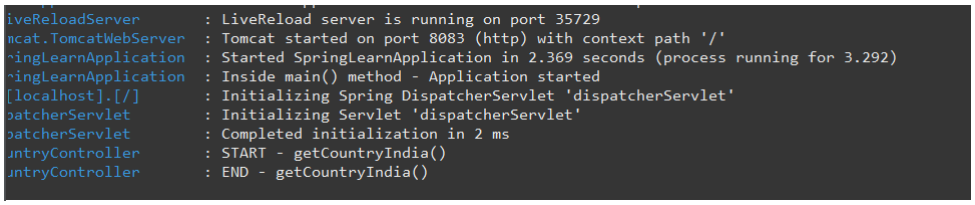
```
        return country;
    }
}
```

application.properties

```
spring.application.name=spring-learn
server.port=8083
```

Output:





4. 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"

}

country.xml

```
<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_learn.model.Country">
      <property name="code" value="IN"/>
      <property name="name" value="India"/>
   </bean>

   <bean id="us" class="com.cognizant.spring_learn.model.Country">
      <property name="code" value="US"/>
      <property name="name" value="United States"/>
   </bean>

   <bean id="countryList" class="java.util.ArrayList">
      <constructor-arg>
         <list>
            <ref bean="in"/>
            <ref bean="us"/>
         </list>
      </constructor-arg>
   </bean>
</beans>
```

CountryService.java

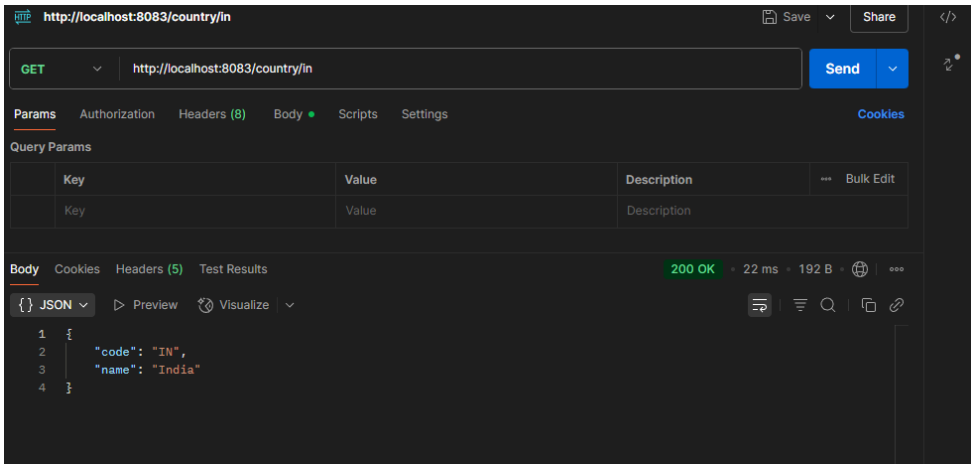
```
package com.cognizant.spring_learn.service;

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

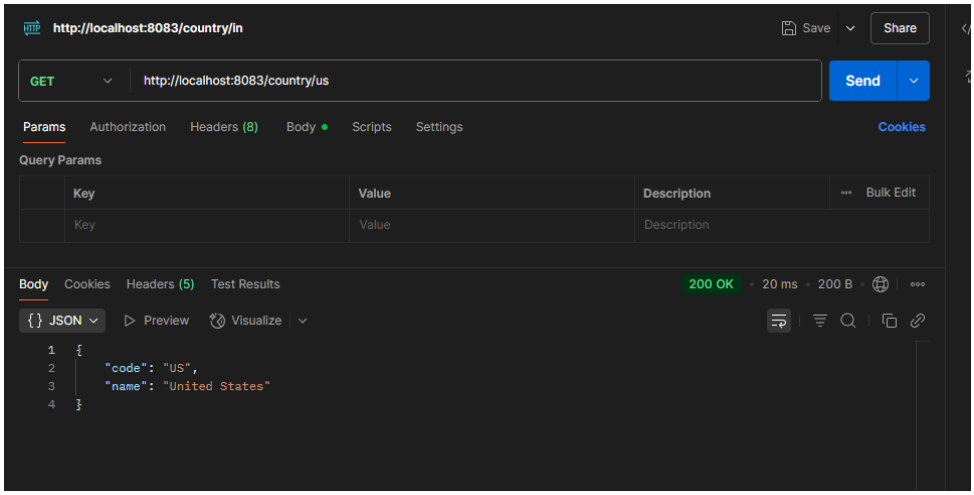
```
        List<Country> countryList = (List<Country>) context.getBean("countryList");

        return countryList.stream()
                .filter(c → c.getCode().equalsIgnoreCase(code))
                .findFirst()
                .orElse(null);  // You can throw a custom exception instead
    }
}
```

CountryController.java

```
package com.cognizant.spring_learn.controller;

import com.cognizant.spring_learn.model.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.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
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);
    @Autowired
    private CountryService countryService;

    @GetMapping("/country/{code}")
    public Country getCountry(@PathVariable String code) {
        LOGGER.info("START - getCountry(): code = {}", code);
        Country country = countryService.getCountry(code);
        LOGGER.info("END - getCountry()");
        return country;
    }


    @RequestMapping("/country")
    public Country getCountryIndia() {
        LOGGER.info("START - getCountryIndia()");
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        Country country = (Country) context.getBean("in");
        LOGGER.info("END - getCountryIndia()");
        return country;
    }
}
```

Output:





5. 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.t3LRvICV-hwKfoqZYIaVQqEUiBIoWcWn0ft3tgv0dL0"}

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

JwtUtil.java

```
package com.cognizant.spring_learn.security;


import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import java.util.Date;

@Component
public class JwtUtil {

    // Generate a secure 256-bit key for HS256
    private final SecretKey secretKey = Keys.secretKeyFor(SignatureAlgorithm.HS256);
    private final 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(secretKey)
                .compact();
    }
}
```

AuthController.java

```
package com.cognizant.spring_learn.controller;

import com.cognizant.spring_learn.security.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);

            if (userDetails.length == 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\"}");
    }
}
```

SecurityConfig.java

```
package com.cognizant.spring_learn.security;
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 encoder for demo
            .roles("USER")
            .build();
        return new InMemoryUserDetailsManager(user);
    }
}
```

Output: