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
    1. Walkthrough all the configuration defined in XML file
    2. Open 'Dependency Hierarchy' and show the dependency tree.

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);
	}
}
```

Application.properties

```
spring.application.name=spring-learn
```

SpringLearnApplicationsTests.java

```
package com.cognizant.spring_learn;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
```

```
class SpringLearnApplicationTests {

	@Test
	void contextLoads() {
	}

}
```

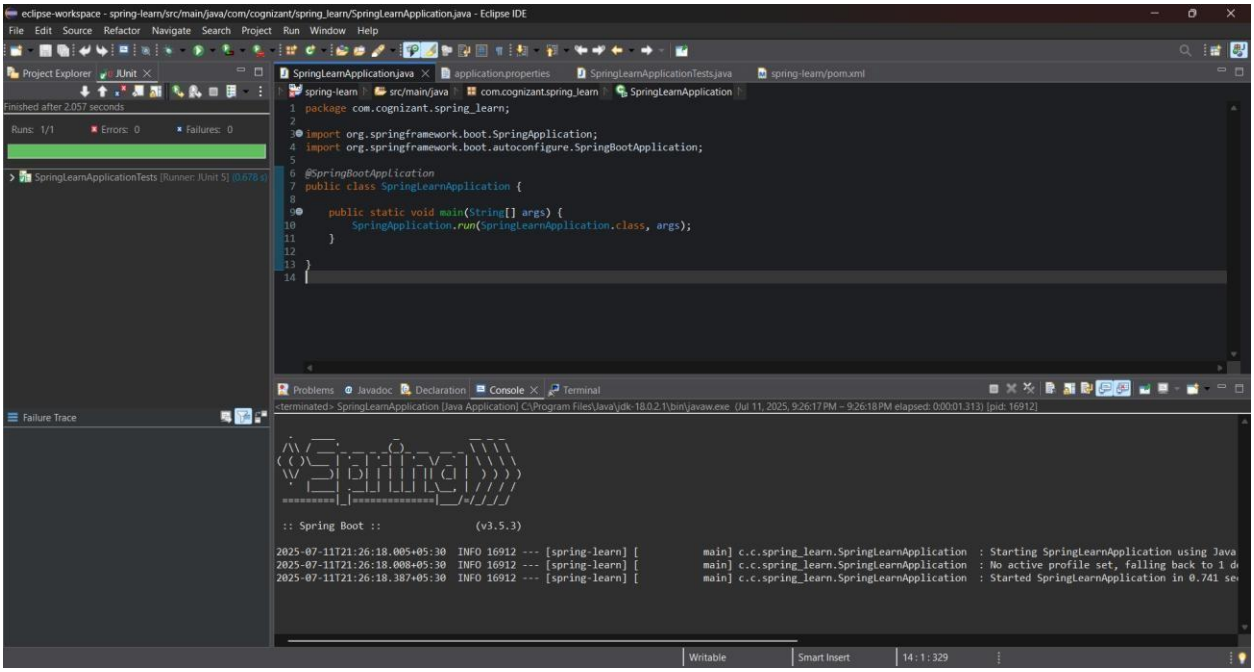
SpringLearn Pom.xml

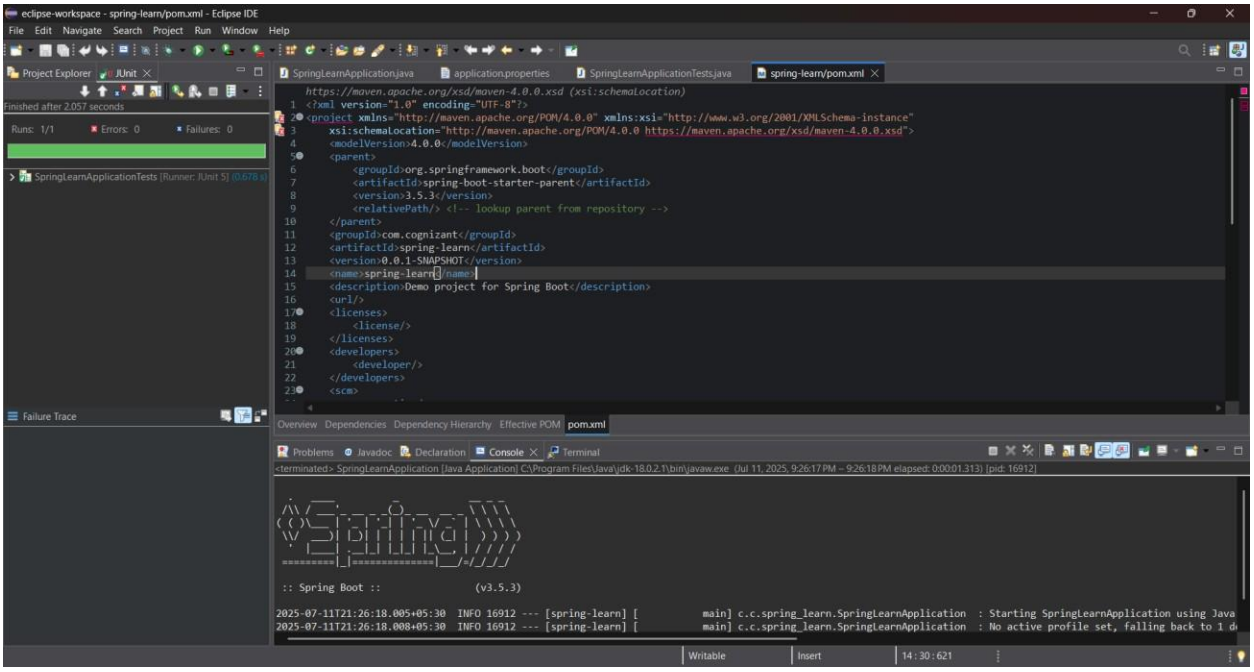
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.5.3</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.cognizant</groupId>
	<artifactId>spring-learn</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>spring-learn</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</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
```

```
            <plugins>
                <plugin>

        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-
plugin</artifactId>
                </plugin>
            </plugins>
        </build>

</project>
```

Output:


eclipse-workspace - spring-learn/src/main/java/com/cognizant/spring_learn/SpringLearnApplication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JUnit
Finished after 2.057 seconds
Runs: 1/1
Errors: 0
Failures: 0
SpringLearnApplicationTests [Runner: JUnit 5] (0.676 s)
Failure Trace
SpringLearnApplication.java
application.properties
SpringLearnApplicationTests.java
spring-learn/pom.xml
spring-learn
src/main/java
com.cognizant.spring_learn
SpringLearnApplication
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);
}
}
Problems
Javadoc
Declaration
Console
Terminal
<terminated> SpringLearnApplication [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Jul 11, 2025, 9:26:17 PM – 9:26:18 PM elapsed: 0:00:01.313) [pid: 16912]
:: Spring Boot :: (v3.5.3)
2025-07-11T21:26:18.005+05:30 INFO 16912 --- [spring-learn] [main] c.c.spring_learn.SpringLearnApplication : Starting SpringLearnApplication using Java
2025-07-11T21:26:18.008+05:30 INFO 16912 --- [spring-learn] [main] c.c.spring_learn.SpringLearnApplication : No active profile set, falling back to 1 de
2025-07-11T21:26:18.387+05:30 INFO 16912 --- [spring-learn] [main] c.c.spring_learn.SpringLearnApplication : Started SpringLearnApplication in 0.741 se
Writable
Smart Insert
14 : 1 : 329


eclipse-workspace - spring-learn/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
JUnit
Finished after 2.057 seconds
Runs: 1/1
Errors: 0
Failures: 0
SpringLearnApplicationTests [Runner: JUnit 5] (0.676 s)
Failure Trace
SpringLearnApplication.java
application.properties
SpringLearnApplicationTests.java
spring-learn/pom.xml
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<?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.cognizant</groupId>
<artifactId>spring-learn</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>spring-learn</name>
<description>Demo project for Spring Boot</description>
<url/>
<licenses>
<license/>
</licenses>
<developers>
<developer/>
</developers>
<scm>
Overview
Dependencies
Dependency Hierarchy
Effective POM
pom.xml
Problems
Javadoc
Declaration
Console
Terminal
<terminated> SpringLearnApplication [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Jul 11, 2025, 9:26:17 PM – 9:26:18 PM elapsed: 0:00:01.313) [pid: 16912]
:: Spring Boot :: (v3.5.3)
2025-07-11T21:26:18.005+05:30 INFO 16912 --- [spring-learn] [main] c.c.spring_learn.SpringLearnApplication : Starting SpringLearnApplication using Java
2025-07-11T21:26:18.008+05:30 INFO 16912 --- [spring-learn] [main] c.c.spring_learn.SpringLearnApplication : No active profile set, falling back to 1 de
Writable
Insert
14 : 30 : 621

Spring Core – Load Country from Spring Configuration XML

An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country.

Code	Name
US	United States
DE	Germany
IN	India
JP	Japan

Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.

Steps to implement

- Pick any one of your choice country to configure in Spring XML configuration named country.xml.
- Create a bean tag in spring configuration for country and set the property and values

```
<bean id="country" class="com.cognizant.springlearn.Country">
    <property name="code" value="IN" />
    <property name="name" value="India" />
</bean>
```

- Create Country class with following aspects:
    - Instance variables for code and name
    - Implement empty parameter constructor with inclusion of debug log within the constructor with log message as “Inside Country Constructor.”
    - Generate getters and setters with inclusion of debug with relevant message within each setter and getter method.
    - Generate toString() method
- Create a method displayCountry() in SpringLearnApplication.java, which will read the country bean from spring configuration file and display the country details. ClassPathXmlApplicationContext, ApplicationContext and context.get

Bean(“beanId”, Country.class). Refer sample code for displayCountry() method below.

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

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

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

- Invoke displayCountry() method in main() method of SpringLearnApplication.java.
- Execute main() method and check the logs to find out which constructors and methods were invoked.

SME to provide more detailing about the following aspects:

- bean tag, id attribute, class attribute, property tag, name attribute, value attribute
- ApplicationContext, ClassPathXmlApplicationContext
- What exactly happens when context.getBean() is invoked

SpringLearnApplication.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplication
Context;

public class SpringLearnApplication {

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

    public static void main(String[] args) {
        LOGGER.debug("Inside main");
        displayCountry();
    }

    public static void displayCountry() {
        ApplicationContext context = new
ClassPathXmlApplicationContext("country.xml");
        Country country = context.getBean("country",
Country.class);
        LOGGER.debug("Country : {}", country.toString());
    }
}
```

Country.java

```
package com.cognizant.springlearn;

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()");
```

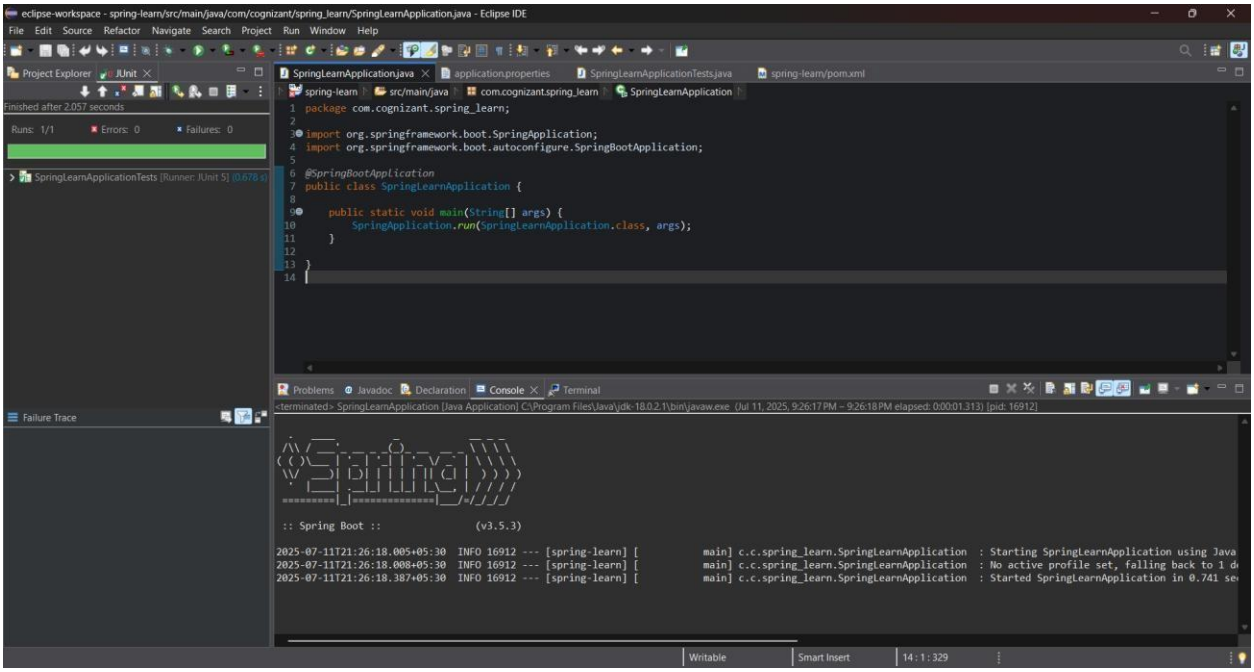
```
        return name;
    }
     public void setName(String name) {
        LOGGER.debug("Inside setName()");
        this.name = name;
    }
    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

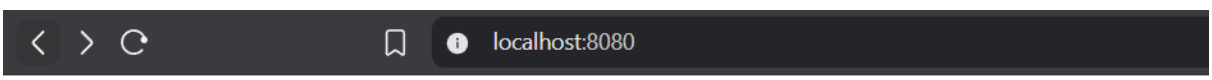
SpringLearn Pom.xml

```
<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
                             http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cognizant</groupId>
    <artifactId>spring-learn</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>
    <name>spring-learn</name>

    <properties>
        <java.version>17</java.version>
```

```
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>2.0.9</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>2.0.9</version>
        </dependency>
    </dependencies>
</project>
```

Output:





Select Your Country

-- Select Country -- Submit

-- Select Country --
United States
Germany
India
Japan

You selected: India (IN)

Back

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!!

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

SpringLearnApplication.java

```
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);
    }
}
```

Application.properties

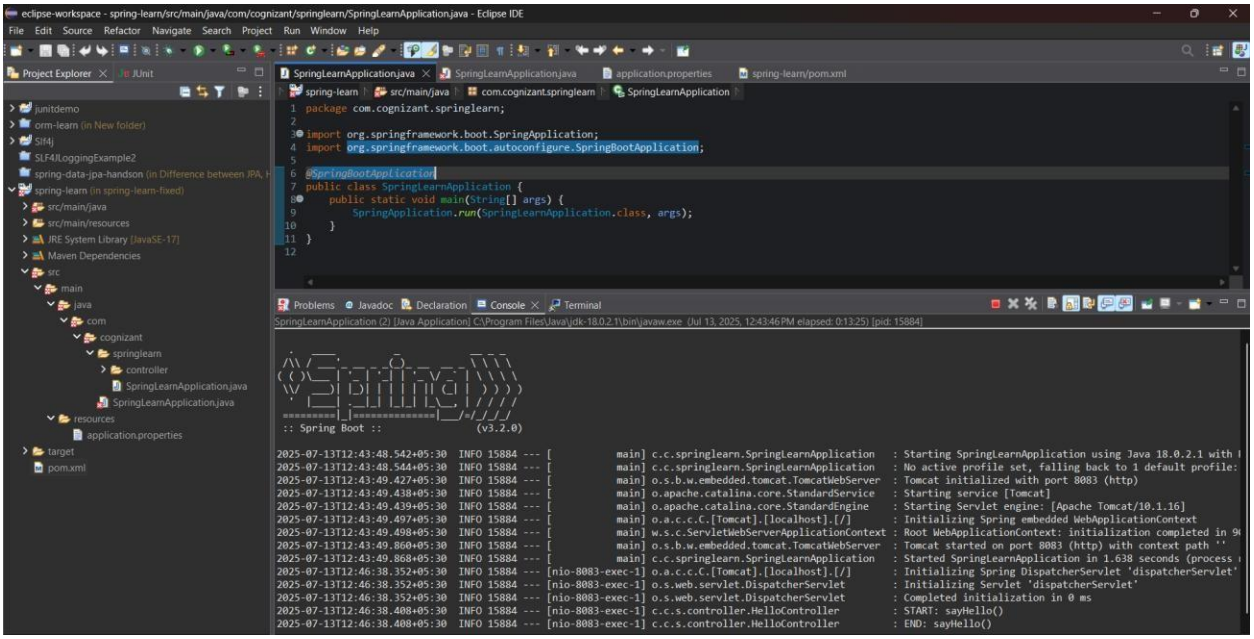
```
server.port=8083
```

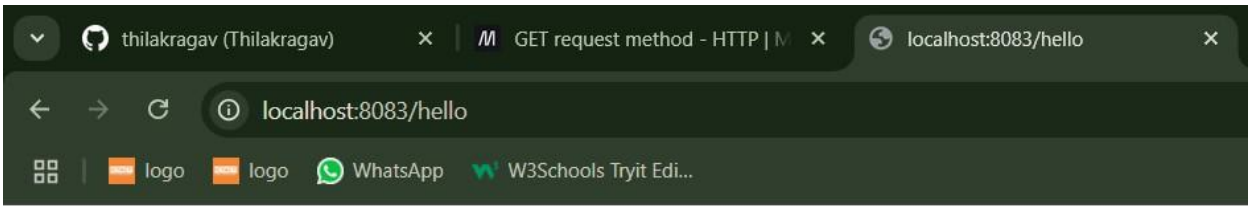
SpringLearn Pom.xml

```
<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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.cognizant</groupId>
    <artifactId>spring-learn</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
```

```
    <name>spring-learn</name>
    <description>Hello World Spring Boot App</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.0</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Output:





Hello World!!

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

SpringLearnApplication.java

```
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);
    }
}
```

Application.properties

```
server.port=8085
```

SpringLearn Pom.xml

```
<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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.cognizant</groupId>
    <artifactId>spring-learn</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
```

```
    <name>spring-learn</name>
    <description>Hello World Spring Boot App</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.0</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

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

Country.java:

```
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;
    }
}
```

CountryController.java:

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplication
```

```
Context;
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 india = (Country) context.getBean("in");
        LOGGER.info("END: getCountryIndia()");
        return india;
    }
}
```

HelloController.java:

```
package com.cognizant.springlearn.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 response = "Hello World!!";
        LOGGER.info("END: sayHello()");
        return response;
    }
}
```

Output:

```
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);
    }
}
```

```
SpringLearnApplication (2) [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Jul 13, 2025, 1:14:42 PM elapsed: 0:07:36) [pid: 10124]

 :: Spring Boot ::                (v3.2.0)

2025-07-13T13:14:45.117+05:30  INFO 10124 --- [           main] c.c.springlearn.SpringLearnApplication   : Starting SpringLearnApplication using Java 18.0.2.1 wit
2025-07-13T13:14:45.120+05:30  INFO 10124 --- [           main] c.c.springlearn.SpringLearnApplication   : No active profile set, falling back to 1 default profil
2025-07-13T13:14:46.073+05:30  INFO 10124 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8085 (http)
2025-07-13T13:14:46.081+05:30  INFO 10124 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-13T13:14:46.081+05:30  INFO 10124 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.16]
2025-07-13T13:14:46.131+05:30  INFO 10124 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-13T13:14:46.133+05:30  INFO 10124 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
2025-07-13T13:14:46.486+05:30  INFO 10124 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8085 (http) with context path ''
2025-07-13T13:14:46.493+05:30  INFO 10124 --- [           main] c.c.springlearn.SpringLearnApplication   : Started SpringLearnApplication in 1.694 seconds (proces
2025-07-13T13:15:29.633+05:30  INFO 10124 --- [nio-8085-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServle
```

```
localhost:8085/country

Pretty-print

{"code":"IN","name":"India"}
```

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"
}
```

SpringLearnApplication.java

```
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);
    }
}
```

Application.properties

```
server.port=8086
```

SpringLearn Pom.xml

```
<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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.cognizant</groupId>
    <artifactId>spring-learn</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
```

```
    <name>spring-learn</name>
    <description>Hello World Spring Boot App</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.0</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

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

   <bean id="us"
class="com.cognizant.springlearn.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>
```

Country.java:

```
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;
    }
}
```

CountryController.java:

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;
import com.cognizant.springlearn.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.ClassPathXmlApplication
Context;
import org.springframework.web.bind.annotation.*;

@RestController
public class CountryController {

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

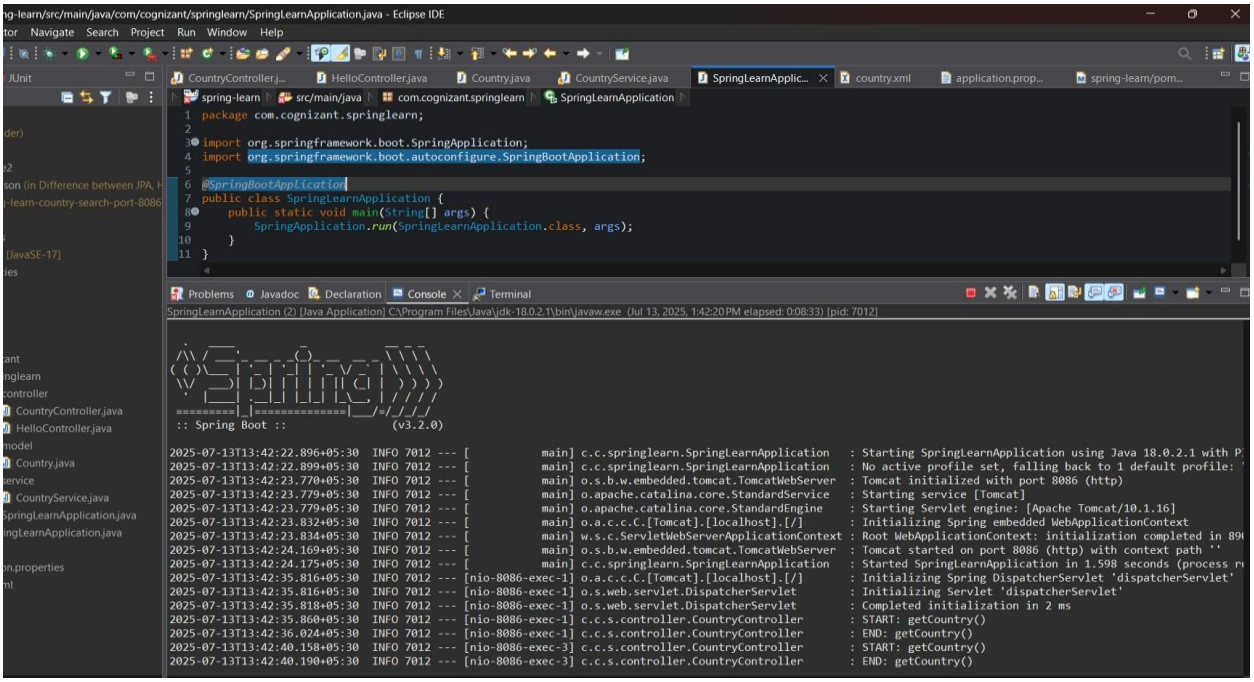
```
    @Autowired
    private CountryService countryService;

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

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

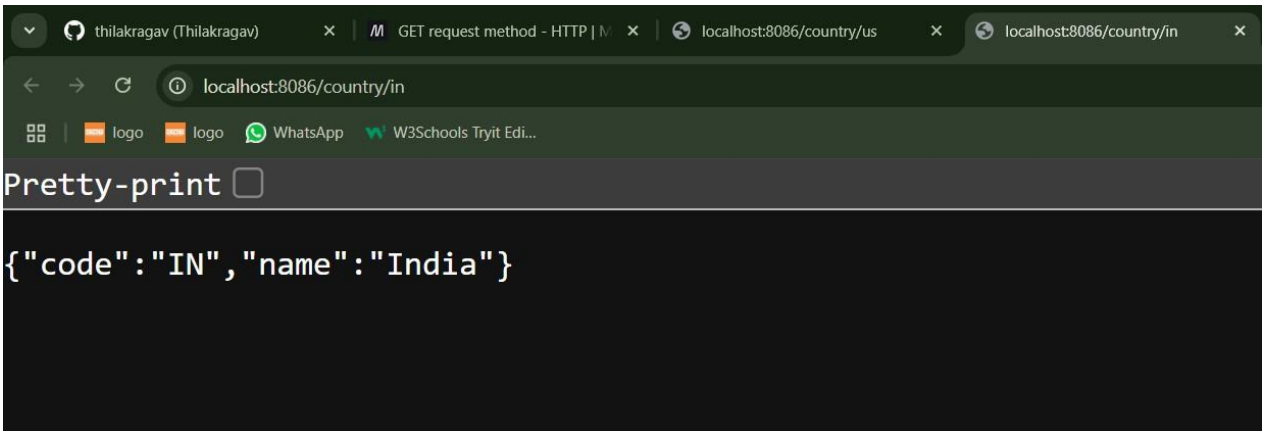
HelloController.java:

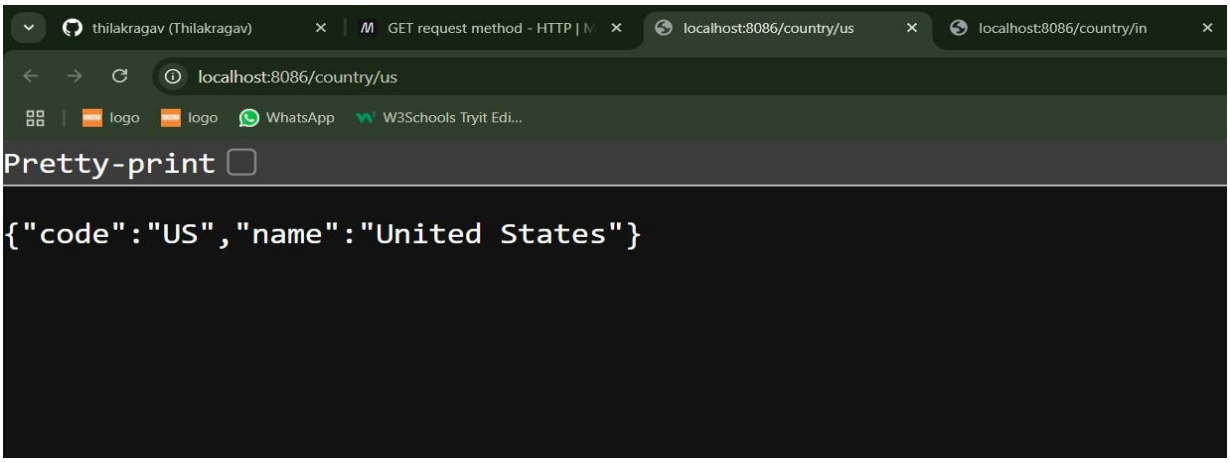
```
package com.cognizant.springlearn.controller;

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 response = "Hello World!!";
        LOGGER.info("END: sayHello()");
        return response;
    }
}
```

Output:


ng-learn/src/main/java/com/cognizant/springlearn/SpringLearnApplication.java - Eclipse IDE
Navigate Search Project Run Window Help
CountryController.j... HelloController.java Country.java CountryService.java SpringLearnApplic... country.xml application.prop... spring-learn/pom...
spring-learn src/main/java com.cognizant.springlearn SpringLearnApplication
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);
}
}
Problems Javadoc Declaration Console Terminal
SpringLearnApplication (2) [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Jul 13, 2025, 1:42:20 PM elapsed: 0:08:33) [pid: 7012]
:: Spring Boot :: (v3.2.0)
2025-07-13T13:42:22.896+05:30 INFO 7012 --- [main] c.c.springlearn.SpringLearnApplication : Starting SpringLearnApplication using Java 18.0.2.1 with P
2025-07-13T13:42:22.899+05:30 INFO 7012 --- [main] c.c.springlearn.SpringLearnApplication : No active profile set, falling back to 1 default profile:
2025-07-13T13:42:23.770+05:30 INFO 7012 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8086 (http)
2025-07-13T13:42:23.779+05:30 INFO 7012 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-13T13:42:23.779+05:30 INFO 7012 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.16]
2025-07-13T13:42:23.832+05:30 INFO 7012 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-13T13:42:23.834+05:30 INFO 7012 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 89
2025-07-13T13:42:24.169+05:30 INFO 7012 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8086 (http) with context path ''
2025-07-13T13:42:24.175+05:30 INFO 7012 --- [main] c.c.springlearn.SpringLearnApplication : Started SpringLearnApplication in 1.598 seconds (process r
2025-07-13T13:42:35.816+05:30 INFO 7012 --- [nio-8086-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-07-13T13:42:35.816+05:30 INFO 7012 --- [nio-8086-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-07-13T13:42:35.818+05:30 INFO 7012 --- [nio-8086-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
2025-07-13T13:42:35.860+05:30 INFO 7012 --- [nio-8086-exec-1] c.c.s.controller.CountryController : START: getCountry()
2025-07-13T13:42:36.024+05:30 INFO 7012 --- [nio-8086-exec-1] c.c.s.controller.CountryController : END: getCountry()
2025-07-13T13:42:40.158+05:30 INFO 7012 --- [nio-8086-exec-3] c.c.s.controller.CountryController : START: getCountry()
2025-07-13T13:42:40.190+05:30 INFO 7012 --- [nio-8086-exec-3] c.c.s.controller.CountryController : END: getCountry()


thilakragav (Thilakragav)
GET request method - HTTP | M
localhost:8086/country/us
localhost:8086/country/in
localhost:8086/country/in
logo logo WhatsApp W3Schools Tryit Edi...
Pretty-print
{"code":"IN","name":"India"}


thilakragav (Thilakragav)
GET request method - HTTP | M
localhost:8086/country/us
localhost:8086/country/in
localhost:8086/country/us
logo logo WhatsApp W3Schools Tryit Edi...
Pretty-print
{"code":"US","name":"United States"}

JWT_Handson

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.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}
```


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

Let incorporate the above as separate hands on exercises.

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.builders.Ht
tpSecurity;

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

import
org.springframework.security.core.userdetails.UserDetailsServic
e;

import
org.springframework.security.provisioning.InMemoryUserDetai
lsManager;

import org.springframework.security.web.SecurityFilterChain;


@Configuration

public class SecurityConfig {

    @Bean

    public UserDetailsService userDetailsService() {

        return new InMemoryUserDetailsManager(

            User.withUsername("user")
```

```
            .password("{noop}pwd")
            .roles("USER")
            .build()
        );
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/authenticate").authenticated()
                .anyRequest().permitAll()
            )
            .httpBasic();
        return http.build();
    }
}
```

AuthenticationController.java :

```
package com.cognizant.spring_learn.controller;
import com.cognizant.spring_learn.util.JwtUtil;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;
import java.util.HashMap;
import java.util.Map;
@RestController
public class AuthenticationController {
    private final JwtUtil jwtUtil;
    public AuthenticationController(JwtUtil jwtUtil) {
        this.jwtUtil = jwtUtil;
    }
    @GetMapping("/authenticate")
    public Map<String, String> createToken(Authentication
authentication) {
        String username = authentication.getName();
        String token = jwtUtil.generateToken(username);
        Map<String, String> response = new HashMap<>();
```

```
        response.put("token", token);
        return response;
    }
}
```

JwUtil.java:

```
package com.cognizant.spring_learn.util;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.stereotype.Component;
import java.util.Date;
@Component
public class JwtUtil {
    private final String secret = "secret-key";
    private final long expirationMs = 1000 * 60 * 10;
    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
```

```
            .setExpiration(new Date(System.currentTimeMillis()
+ expirationMs))
            .signWith(SignatureAlgorithm.HS256, secret)
            .compact();
    }
}
```

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);
    }
}
```

Output:

```
SpringLearnA...  application...  Authenticati...  SecurityConf...  spring-lear...  JwtUtil.java  SpringLearnA...  SecurityConf...  Authenticati...  JwtUtil.java

1  package com.cognizant.spring_learn;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class SpringLearnApplication {
8      public static void main(String[] args) {
9          SpringApplication.run(SpringLearnApplication.class, args);
10     }
11 }
12
```

```
Problems  Javadoc  Declaration  Search  Console  Properties  Eclipse IDE for Java Developers 2025-06 Release
SpringLearnApplication (2) [Java Application] C:\Users\Kaushic\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (13-Jul-2025, 3:27:58 pm elapsed: 0:09:48) [pid: 212

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.3)

2025-07-13T15:28:05.226+05:30  INFO 21200 --- [           main] c.c.spring_learn.SpringLearnApplication  : Starting SpringLearnApplication using Java 21.0.7 with PID 2
2025-07-13T15:28:05.232+05:30  INFO 21200 --- [           main] c.c.spring_learn.SpringLearnApplication  : No active profile set, falling back to 1 default profile: "d
2025-07-13T15:28:07.483+05:30  INFO 21200 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8090 (http)
2025-07-13T15:28:07.501+05:30  INFO 21200 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-13T15:28:07.501+05:30  INFO 21200 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.12]
2025-07-13T15:28:07.701+05:30  INFO 21200 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-13T15:28:07.702+05:30  INFO 21200 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2261
2025-07-13T15:28:08.416+05:30  INFO 21200 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.springframework.security.w
2025-07-13T15:28:08.691+05:30  INFO 21200 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8090 (http) with context path ''
2025-07-13T15:28:08.715+05:30  INFO 21200 --- [           main] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 4.135 seconds (process run
2025-07-13T15:28:19.662+05:30  INFO 21200 --- [nio-8090-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-07-13T15:28:19.665+05:30  INFO 21200 --- [nio-8090-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-13T15:28:19.672+05:30  INFO 21200 --- [nio-8090-exec-2] o.s.web.servlet.DispatcherServlet        : Completed initialization in 7 ms
```

Overview | GET http://localhost:8090/au | No environment

http://localhost:8090/authenticate Save Share

GET http://localhost:8090/authenticate Send

Params | Authorization | Headers (7) | Body | Scripts | Tests | Settings Cookies

Auth Type: Basic Auth

Username: user

Password: pwd

The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.

Body | Cookies | Headers (11) | Test Results 200 OK · 1.11 s · 475 B

{} JSON | Preview | Visualize


```
{
    "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUyNDAwNzAwLCJleHAiOjE3NTI0MDEzMDB9.
        QcM3Y_X6L7uW9TcGbGg1xbAq9m9NY4_nB5ASHYx68gE"
}
```