Cognizant Java FSE – (Deep Skilling)

(WEEK-4)

**MODULE 1:** SPRING REST USING SPRING BOOT

**Submitted by**  
**Name:** SOLLETI VENKATA KUSUMA  
**Roll No:** 111522102145  
**Email:** [22102145@rmd.ac.in](mailto:22102145@rmd.ac.in)

**College:** RMD ENGINEERING COLLEGE  
**Batch:** Java FSE – 2026

**SPRING REST USING SPRING BOOT**

**Create a Spring Web Project using Maven**

**Follow steps below to create a project:**

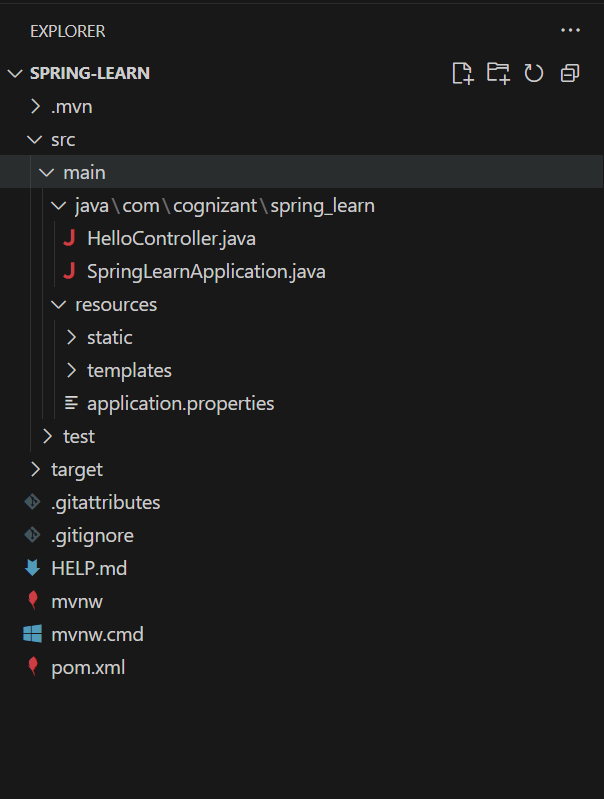
1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -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.

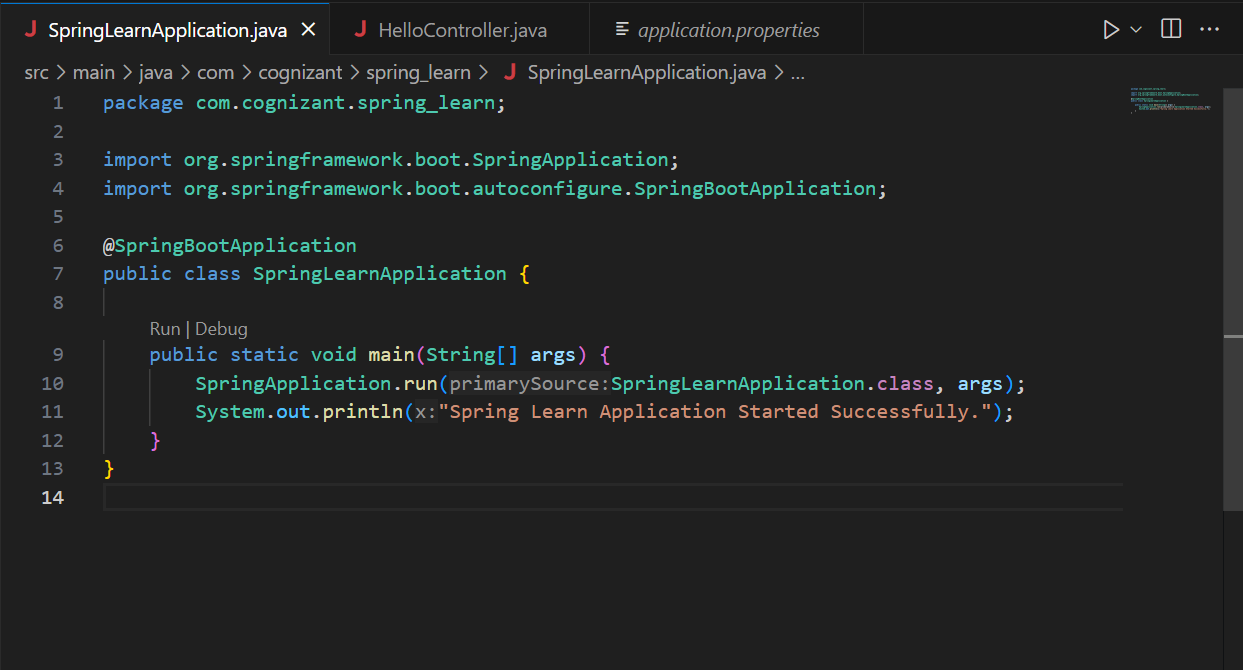
**SME to walk through the following aspects related to the project created:**

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.

**PROJECT STRUCTURE:**



SpringLearnApplication.java

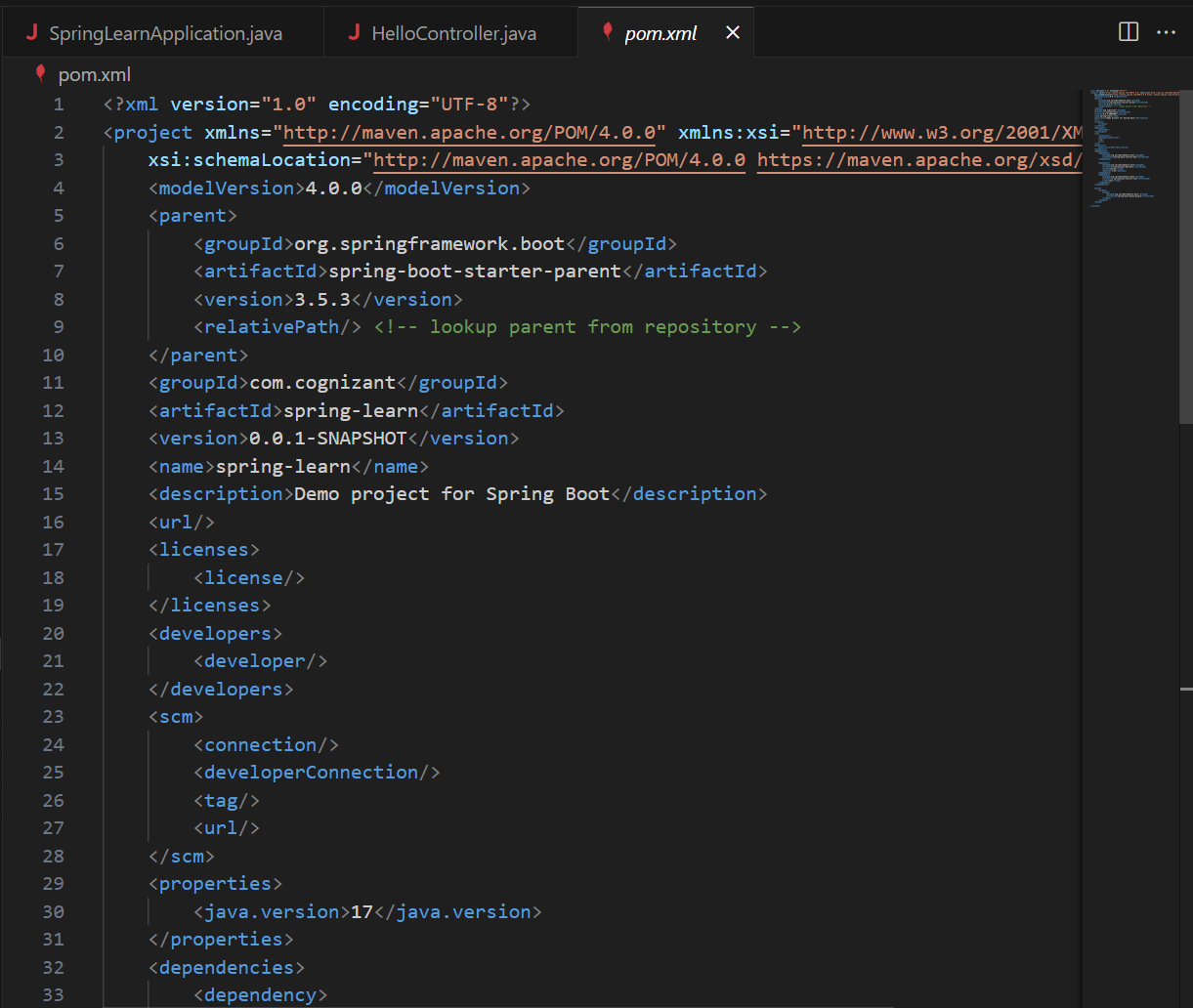


HelloController.java

A screen shot of a computer program

AI-generated content may be incorrect.

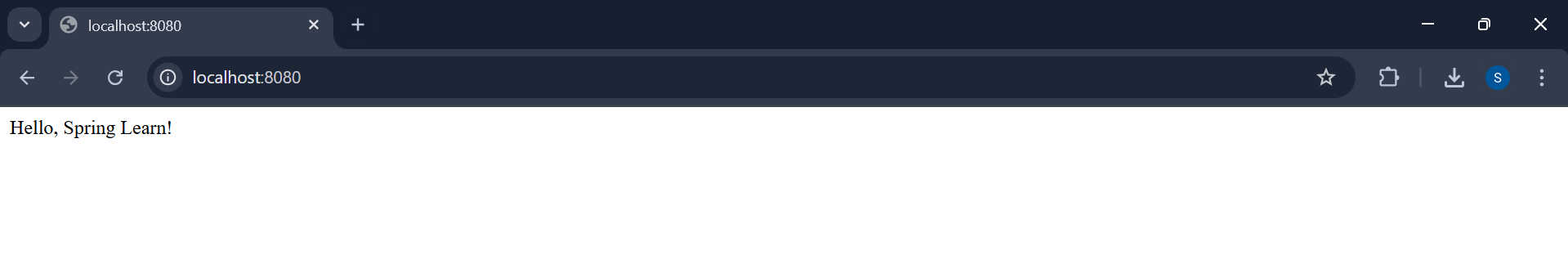
pom.xml



A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:



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

PROJECT STRUCTURE:

A screenshot of a computer

AI-generated content may be incorrect.

Country.java

A screen shot of a computer program

AI-generated content may be incorrect.

SpringLearnApplication.java

A screen shot of a computer program

AI-generated content may be incorrect.

Country.xml

A screenshot of a computer

AI-generated content may be incorrect.

Logback.xml

A screen shot of a computer

AI-generated content may be incorrect.

pom.xml

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

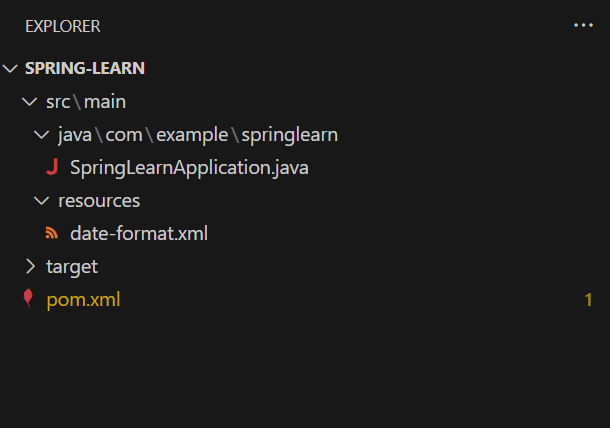
A screenshot of a computer

AI-generated content may be incorrect.

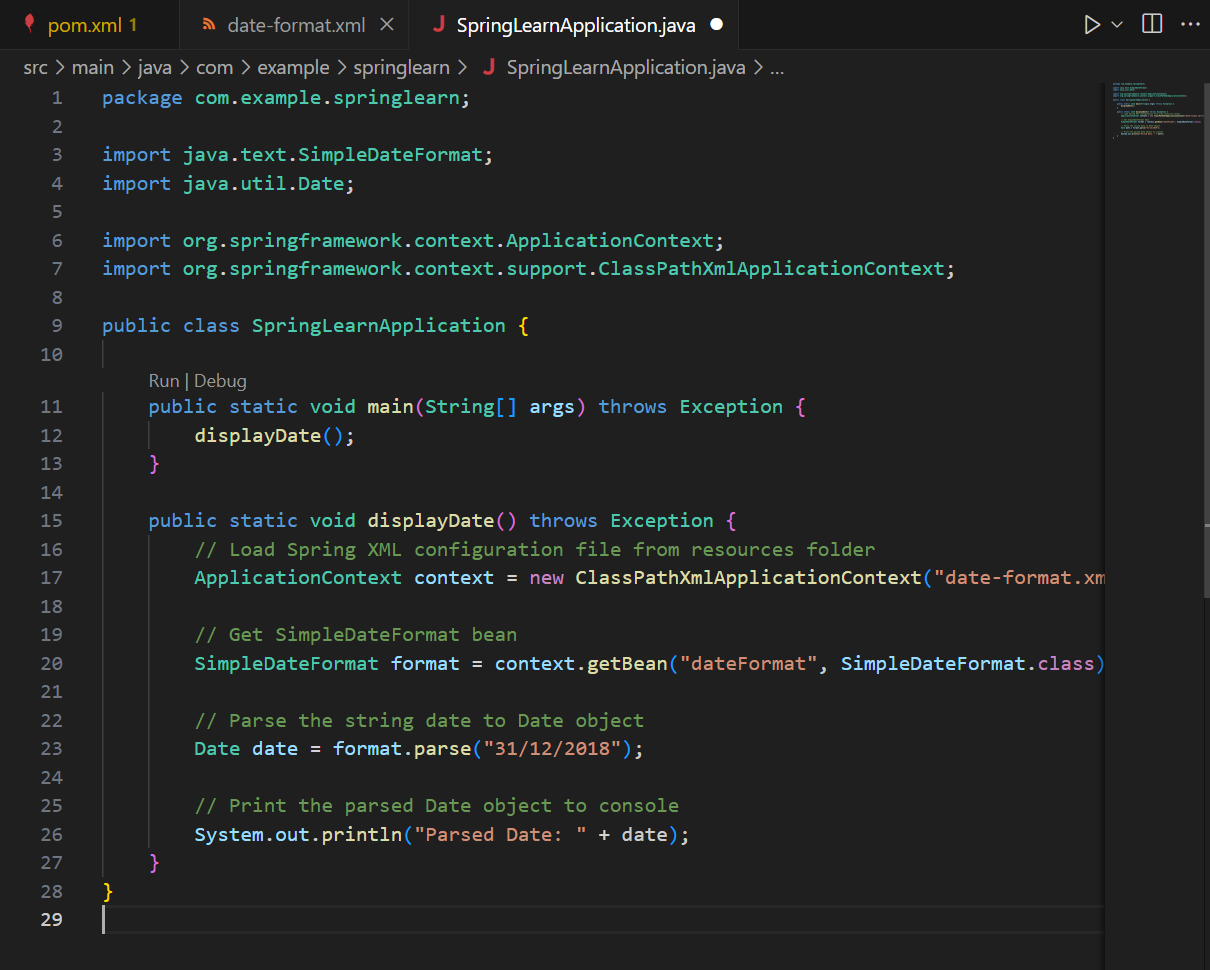
**Spring Core – Load SimpleDateFormat from Spring Configuration XML**

SimpleDateFormat with the pattern ‘dd/MM/yyyy’ is created in multiple places of an application. To avoid creation of SimpleDateFormat in multiple places, define a bean in Spring XML Configuration file and retrieve the date.  
  
Follow steps below to implement:

* Create spring configuration file date-format.xml in src/main/resources folder of 'spring-learn' project
* Open https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-factory-metadata
* Copy the XML defined in the section of previous step URL and paste it into date-format.xml
* Define bean tag in the XML with for date format. Refer code below.
* Create new method displayDate() in SpringLearnApplication.java
* In displayDate() method create the ApplicationContext. Refer code below:
* Get the dateFormat using getBean() method. Refer code below.
* Using the format variable try to parse string '31/12/2018' to Date class and display the result using System.out.println.
* Run the application as 'Java Application' and check the result in console log output.

**PROJECT STRUCTURE:**  
  


SpringLearnApplication.java



Date-format.xml



pom.xml

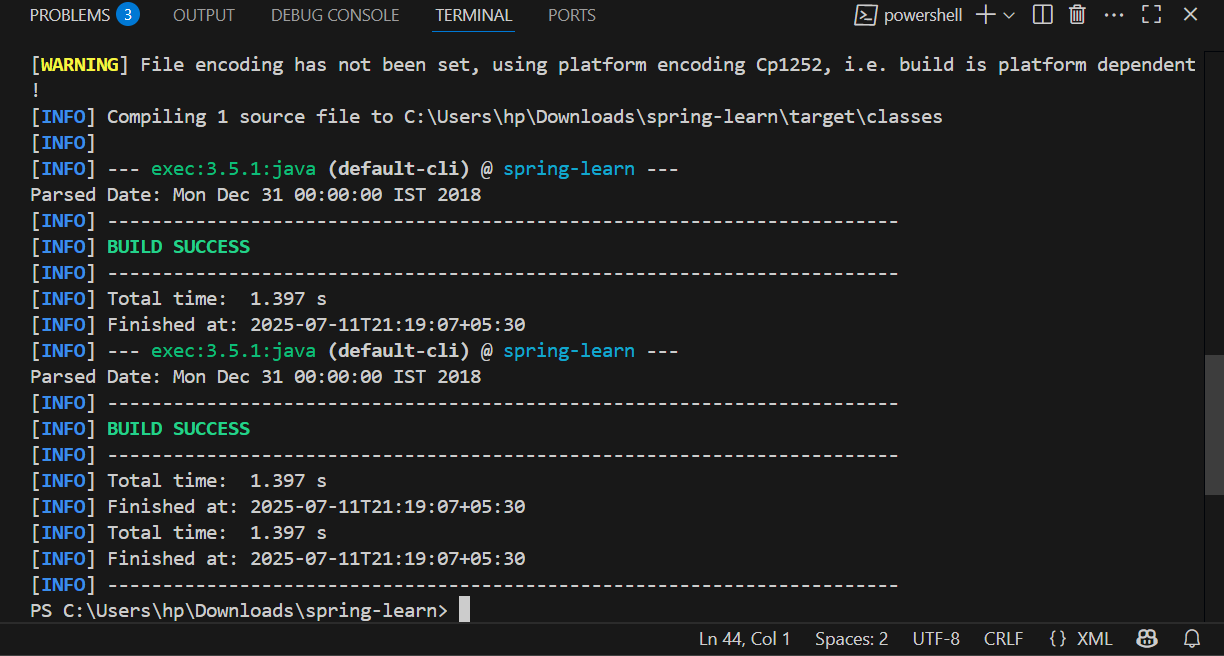
A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:



**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:

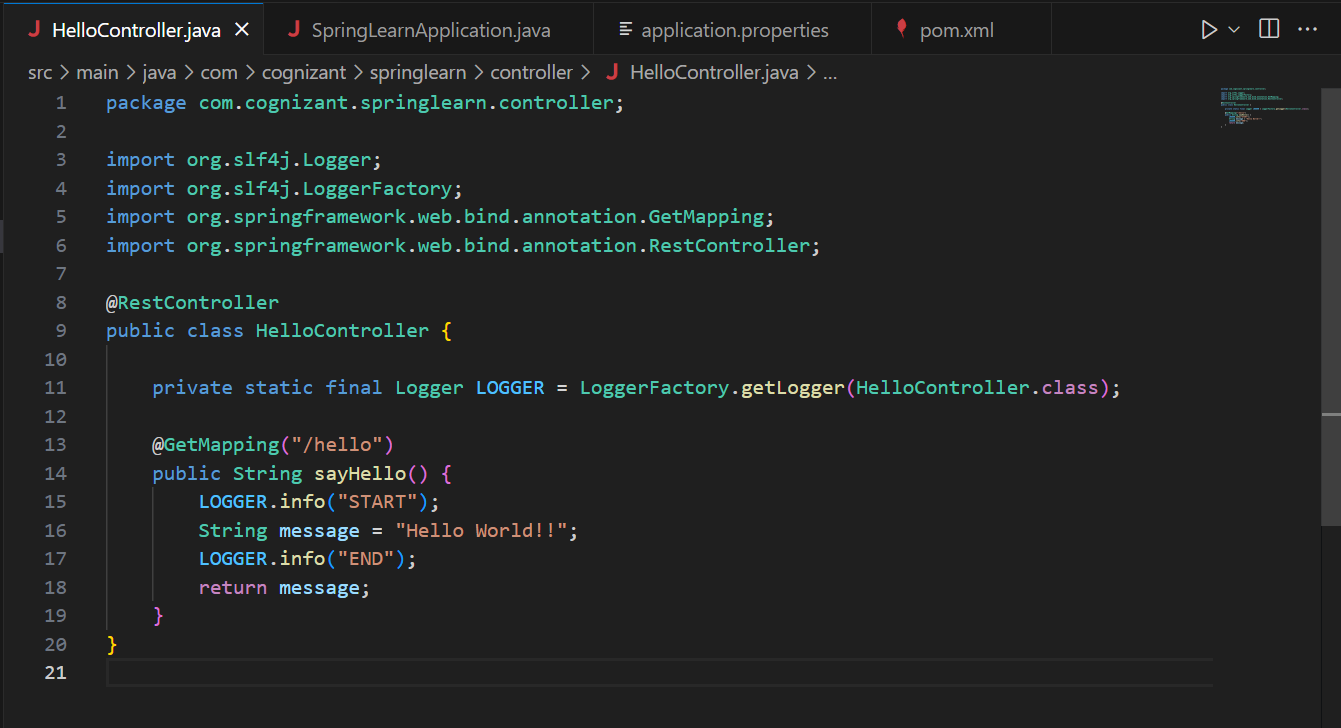
* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

PROJECT STRUCTURE:

A screenshot of a computer

AI-generated content may be incorrect.

HelloController.java



SpringLearnApplication.java

A screen shot of a computer program

AI-generated content may be incorrect.

application.properties

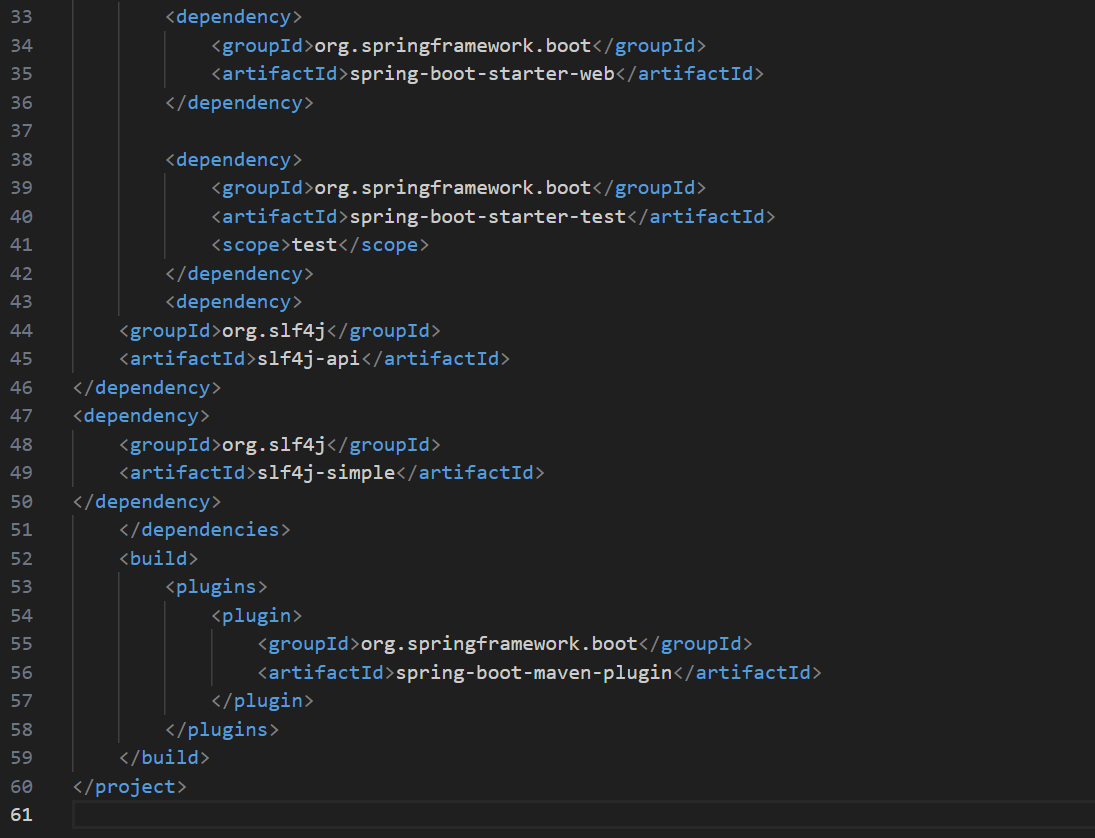
A screenshot of a computer

AI-generated content may be incorrect.

pom.xml

A screen shot of a computer program

AI-generated content may be incorrect.



OUTPUT:

A screen shot of a computer

AI-generated content may be incorrect.

**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

PROJECT STRUCTURE:

A screenshot of a computer program

AI-generated content may be incorrect.

country.xml

A screenshot of a computer program

AI-generated content may be incorrect.

Country.java

A screen shot of a computer

AI-generated content may be incorrect.

CountryController.java

A screen shot of a computer program

AI-generated content may be incorrect.

SpringLearnApplication.java

A screen shot of a computer program

AI-generated content may be incorrect.

pom.xml

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

application.properties

A black screen with text

AI-generated content may be incorrect.

OUTPUT:

A black and grey rectangular object

AI-generated content may be incorrect.

**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"

}

PROJECT STRUCTURE:

A screenshot of a computer program

AI-generated content may be incorrect.

Country.java

A screen shot of a computer program

AI-generated content may be incorrect.

CountryController.java

A screenshot of a computer program

AI-generated content may be incorrect.

CountryService.java

A screen shot of a computer program

AI-generated content may be incorrect.

Country.xml

A screenshot of a computer program

AI-generated content may be incorrect.

pom.xml

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

application.properties

A screenshot of a computer

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

A black and grey rectangular object

AI-generated content may be incorrect.

**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**

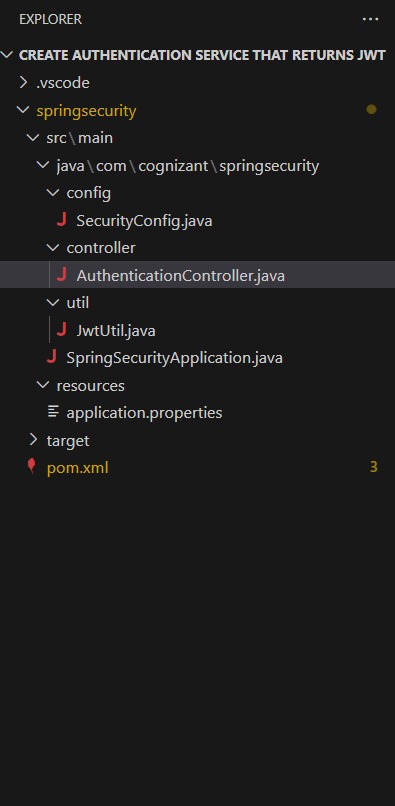
curl -s -u user:pwd http://localhost:8090/authenticate

**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

PROJECT STRUCTURE:



SpringSecurityApplication.java

A computer screen with text on it

AI-generated content may be incorrect.

AuthenticationController.java

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen with colorful text

AI-generated content may be incorrect.

SecurityConfig.java

A computer screen with text on it

AI-generated content may be incorrect.

A computer screen with text on it

AI-generated content may be incorrect.

Application.properties

A screenshot of a computer

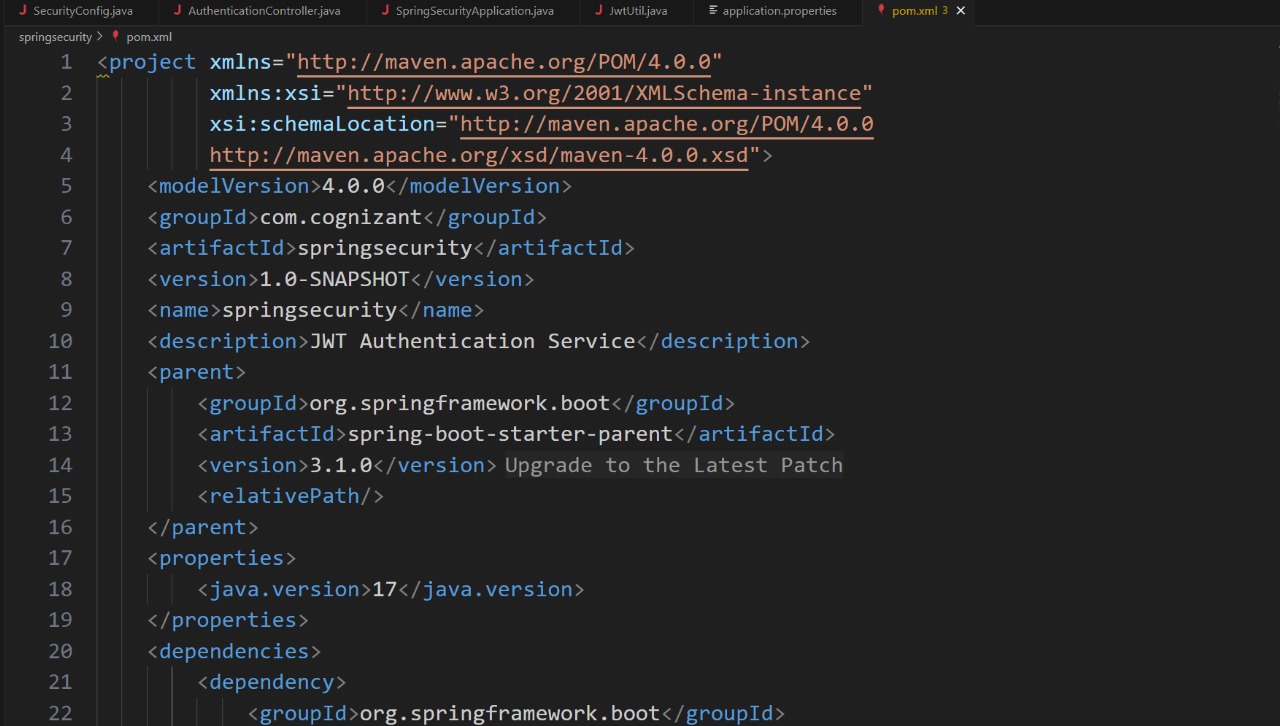
AI-generated content may be incorrect.

Jwtutil.java

A computer screen with many colorful text

AI-generated content may be incorrect.

pom.xml



A screen shot of a computer

AI-generated content may be incorrect.

A computer screen with white and blue text

AI-generated content may be incorrect.

OUTPUT:

