**Cognizant Digital Nurture 4.0**

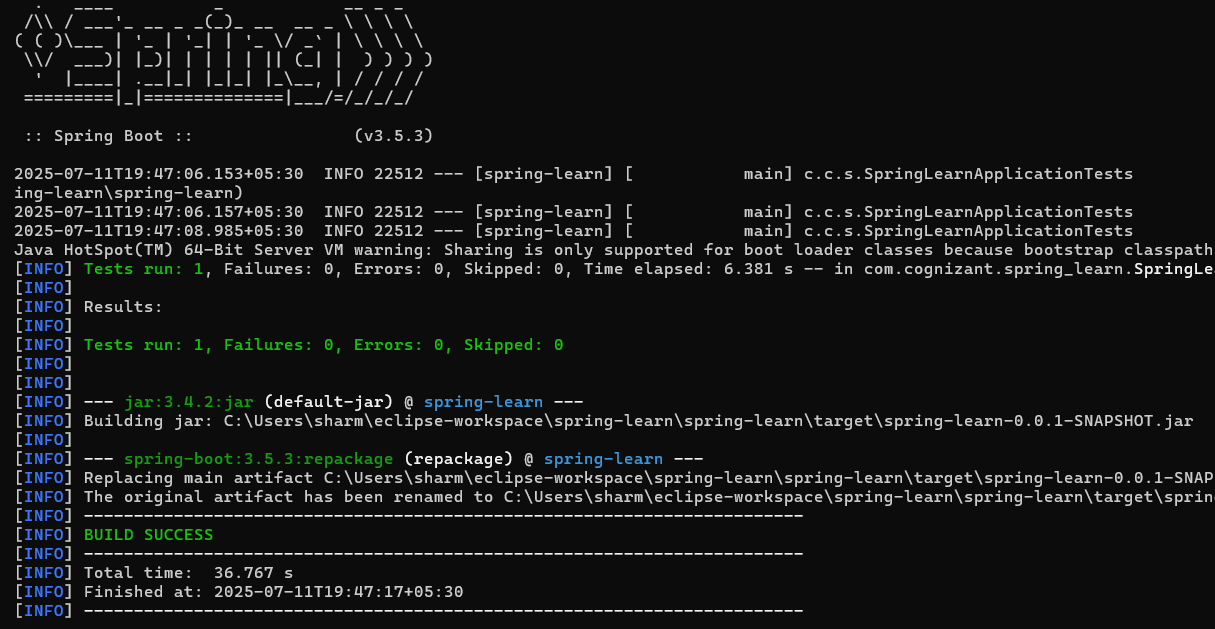
***WEEK-4 Module 7 – Spring REST using Spring Boot 3***

**1. SPRING-REST-HANDSON**

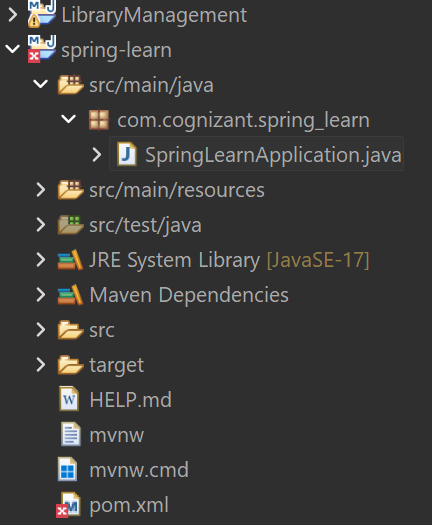
**Hands on 1**

**Create a Spring Web Project using Maven** **(MANDATORY)**  
  
Follow steps below to create a project: 

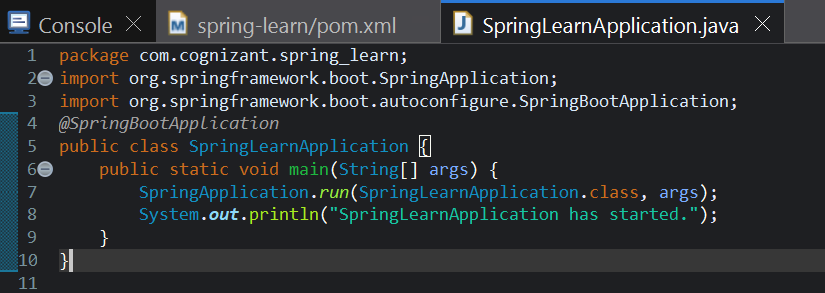
1. Go to <https://start.spring.io/>
   1. Change Group as “com.cognizant”
   2. Change Artifact Id as “spring-learn”
   3. Select Spring Boot DevTools and Spring Web
   4. Create and download the project as zip
   5. Extract the zip in root folder to Eclipse Workspace
2. Build the project using ‘mvn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



1. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"

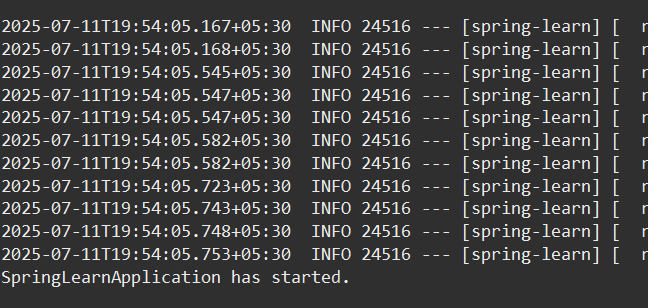


1. Include logs to verify if main() method of SpringLearnApplication.



1. Run the SpringLearnApplication class.

Output:

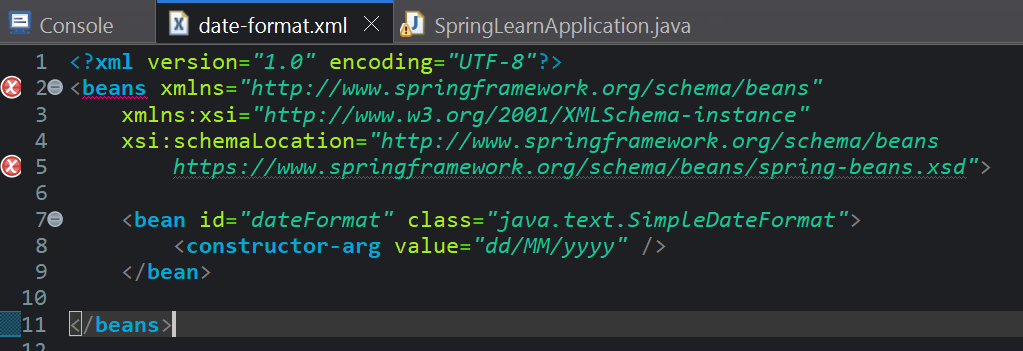




**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML** **(MANDATORY)**  
  
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

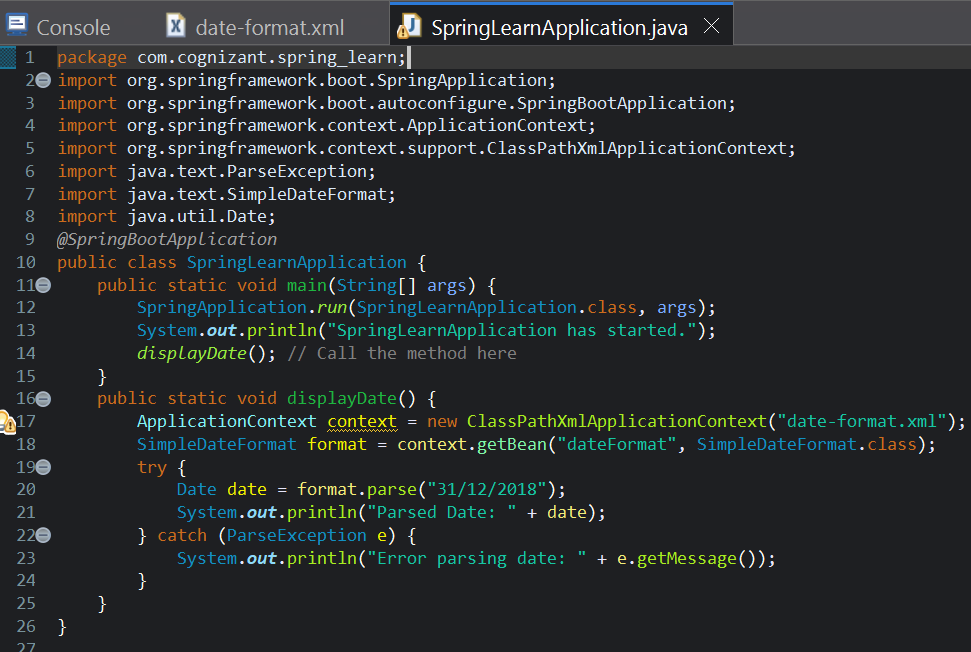


* Create new method displayDate() in SpringLearnApplication.java
* In displayDate() method create the ApplicationContext. Refer code below:

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

* Get the dateFormat using getBean() method. Refer code below.

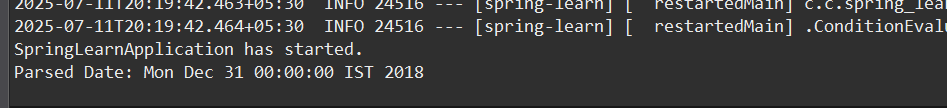
SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);



* 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.



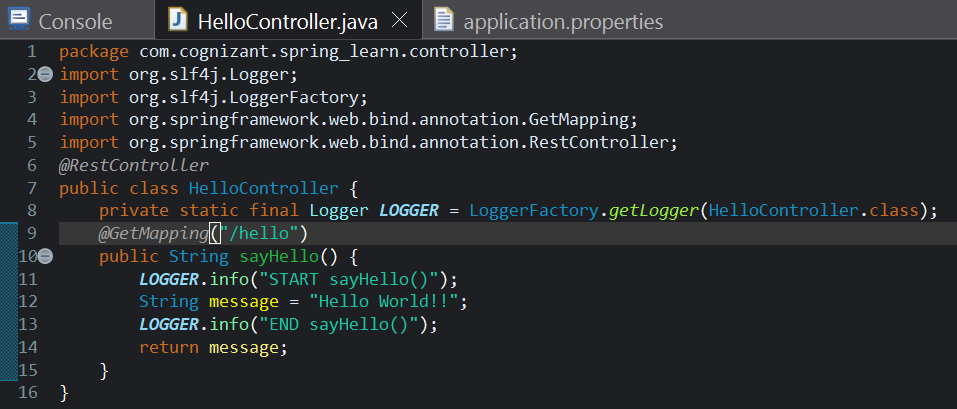
Output:



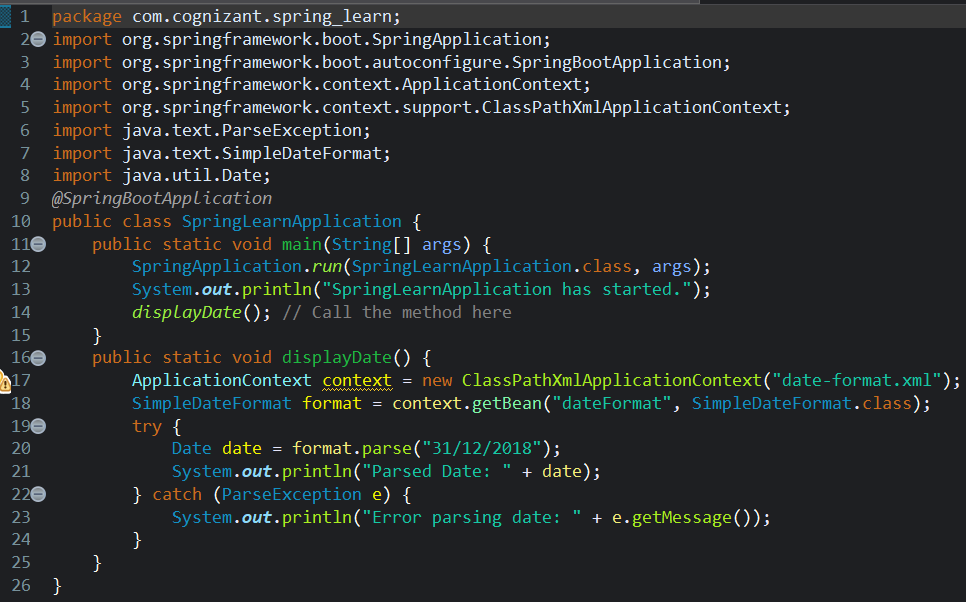
**2. SPRING-REST-HANDSON**

**Hand on exercise 1 – (MANDATORY)**

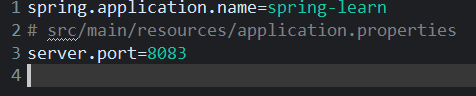
**Hello World RESTful Web Service**   
  
Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below:  
  
**Method:** GET  
**URL:** /hello  
**Controller:** com.cognizant.spring-learn.controller.HelloController



**Method Signature:** public String sayHello()  
**Method Implementation:** return hard coded string "Hello World!!"



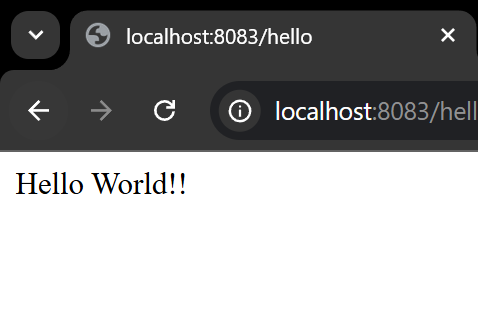
**Sample Request**: <http://localhost:8083/hello>



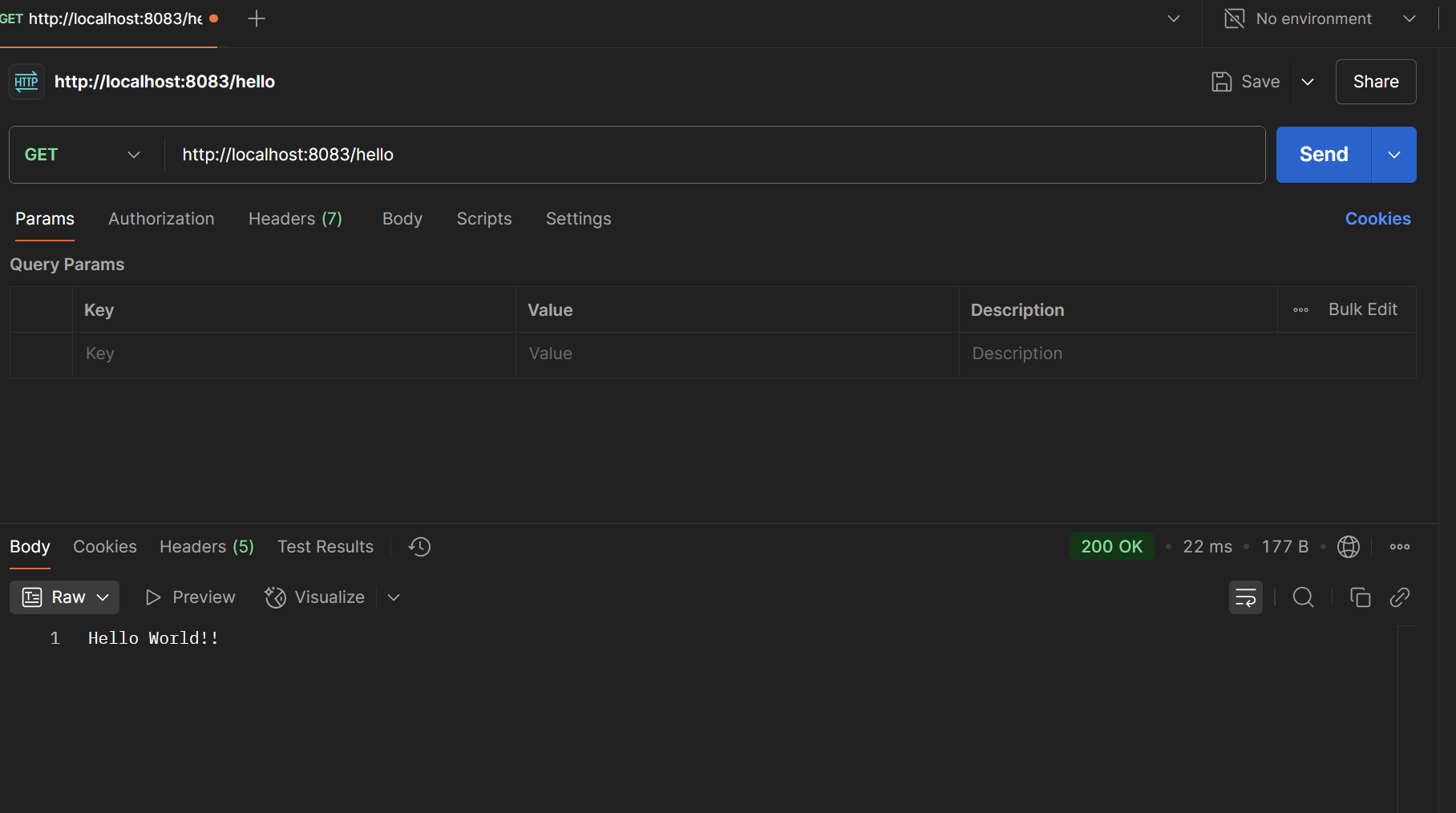
**Sample Response:** Hello World!!

Try the URL http://localhost:8083/hello in both chrome browser and postman.

**Chrome Browser Version:**



Postman Version:



**Hand on 2:**

**REST - Get country based on country code** **(MANDATORY)**  
  
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**:

{

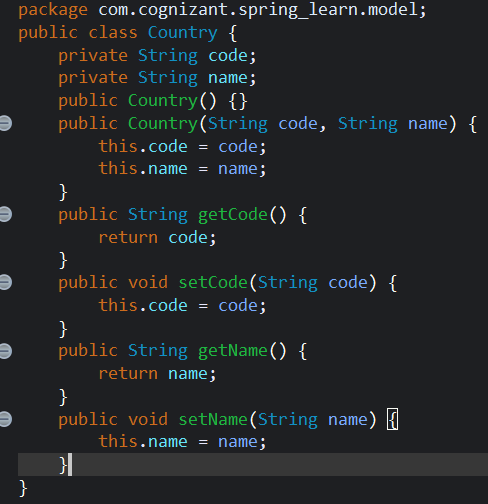
  "code": "IN",

  "name": "India"

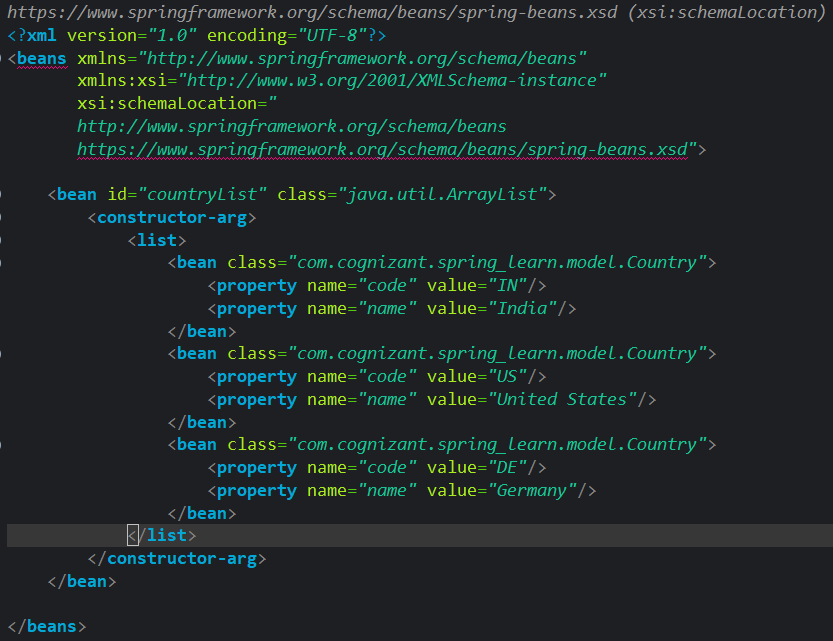
}

**Steps :**

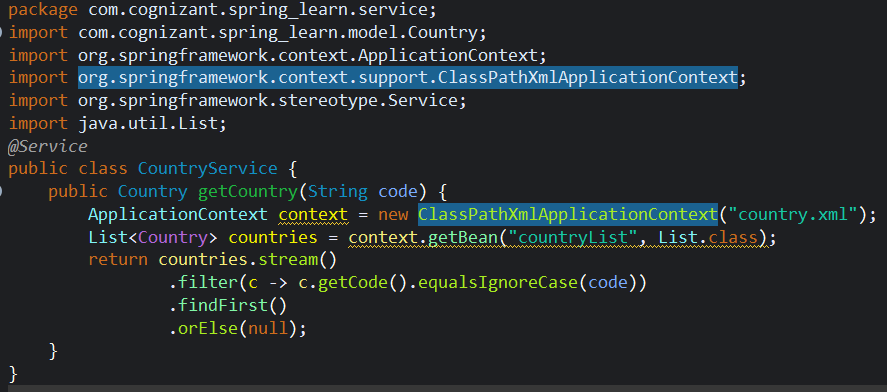
* 1. Created class "Country" in package "com.cognizant.spring\_learn.model"



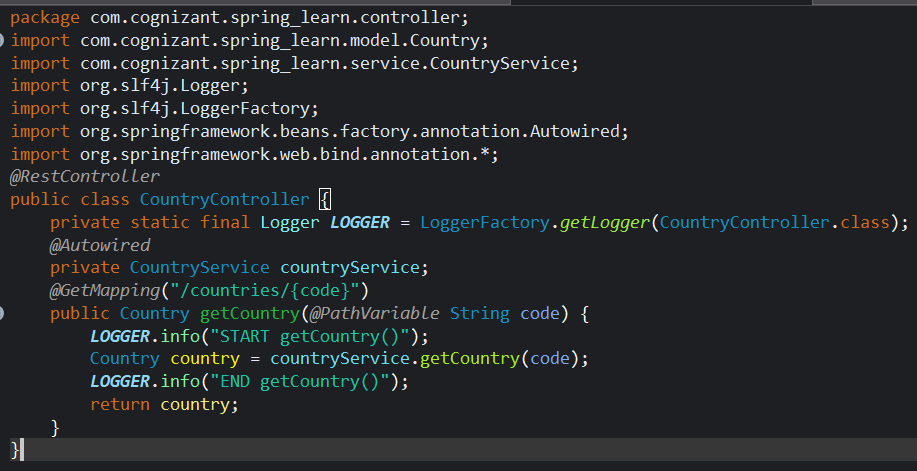
* 1. **Create country.xml (Spring Bean Config File)**

****

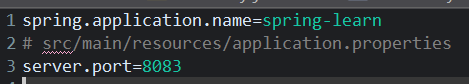
* 1. **Create the CountryService Class**

****

* 1. **Create the CountryController Class**

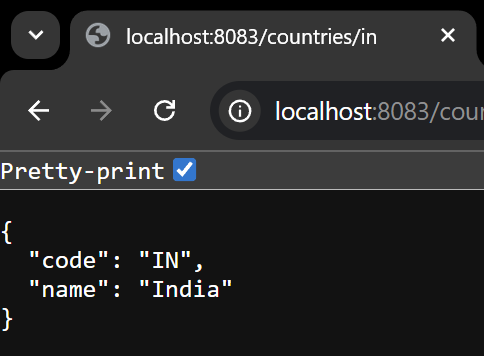
****

* 1. **Set the Server Port**

****

1. **Run and Test the API**

[**http://localhost:8083/countries/in**](http://localhost:8083/countries/in)

****

**5. JWT-handson**

**Create authentication service that returns JWT (MANDATORY)**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"}

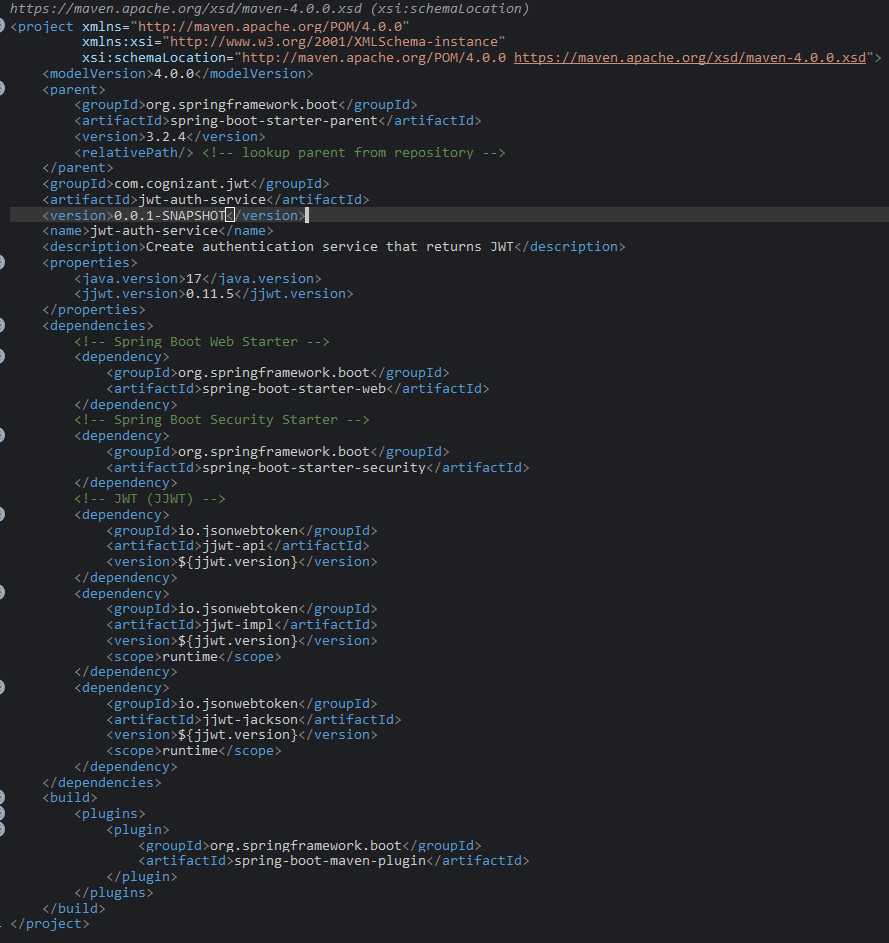
This can be incorporated as three major steps:

* 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

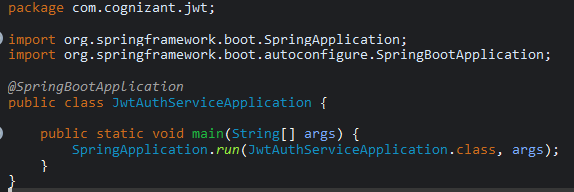
Let incorporate the above as separate hands on exercises.

**STEPS:**

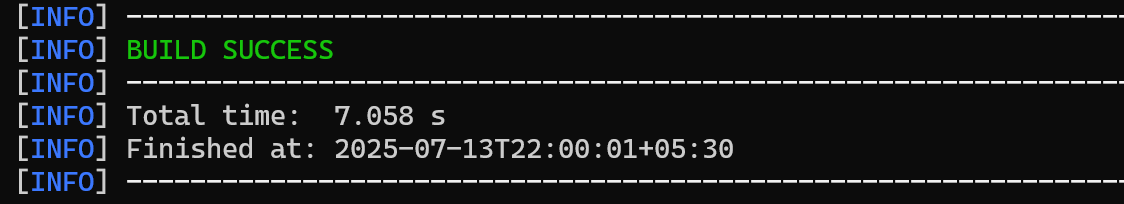
* 1. Created new maven project in eclipse
  2. updated pom.xml with every dependency we need



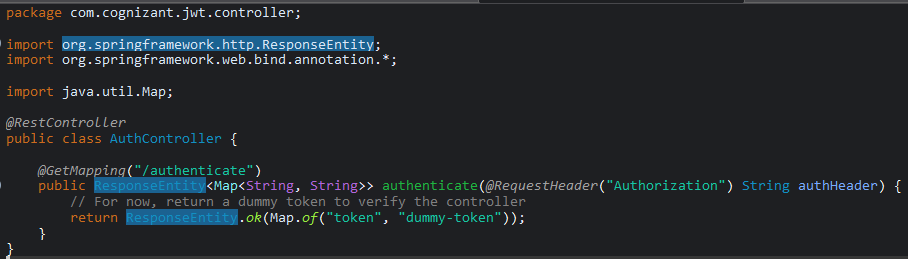
* 1. Create the main application class

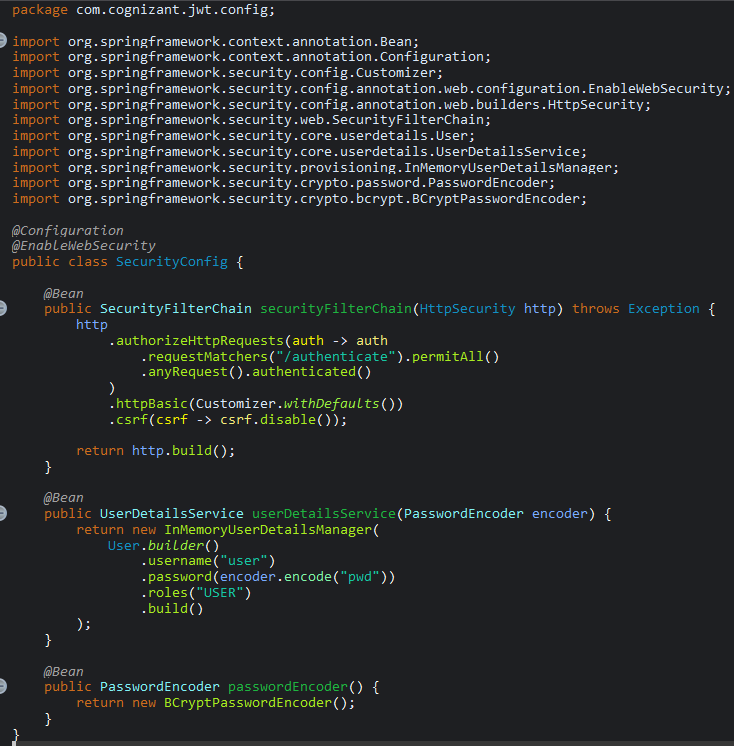


* 1. Run the build



* 1. Create Authentication Controller and Configure Security

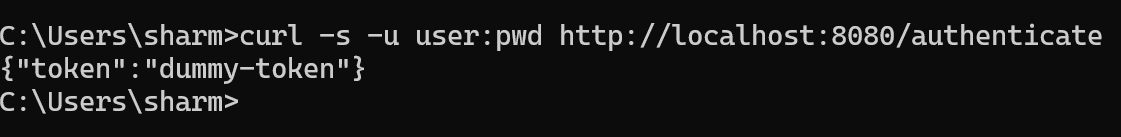




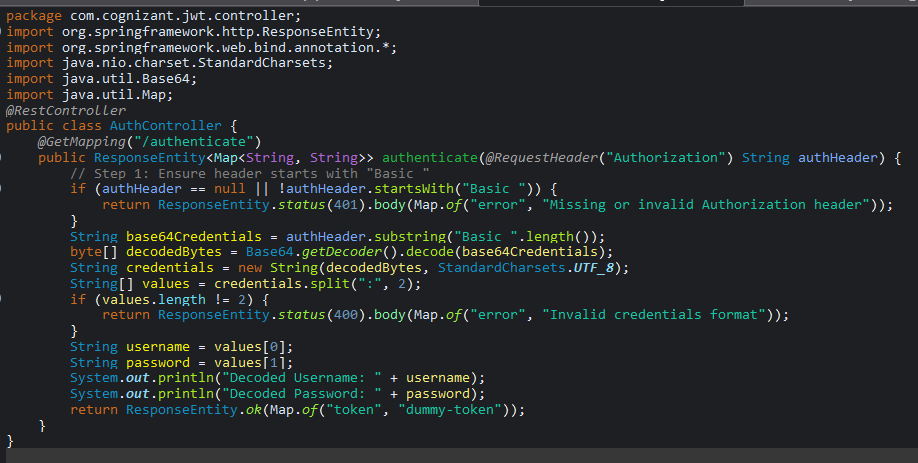
* 1. Start your Spring Boot app and run:

curl -s -u user:pwd <http://localhost:8080/authenticate>

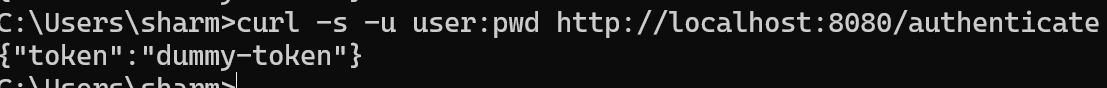
output:

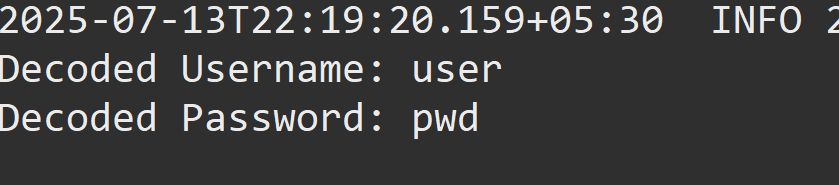


1. Updating /authenticate method to decode the Basic Auth header:



1. Run the program again:

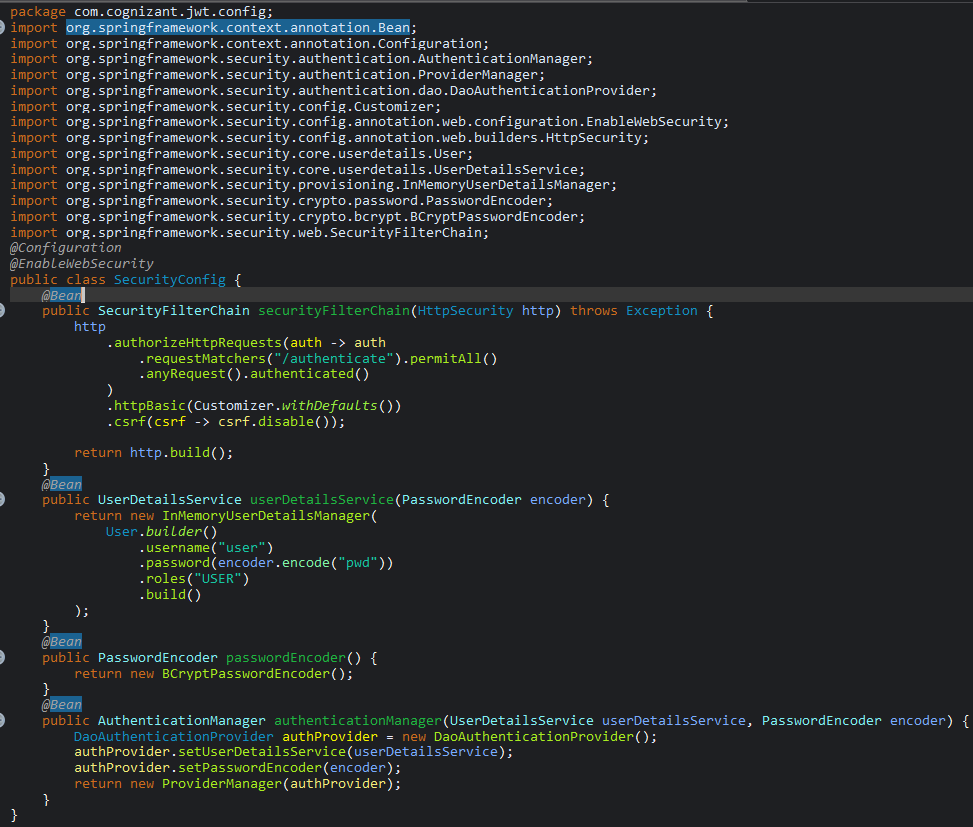




1. Create a helper to generate the JWT - JwtUtil.java



1. Inject AuthenticationManager (custom bean) for manual auth in SecurityConfig.java.



1. Run it

**Output:**

