**1.Create a Spring Web Project using Maven**   
  
Follow steps below to create a project: 

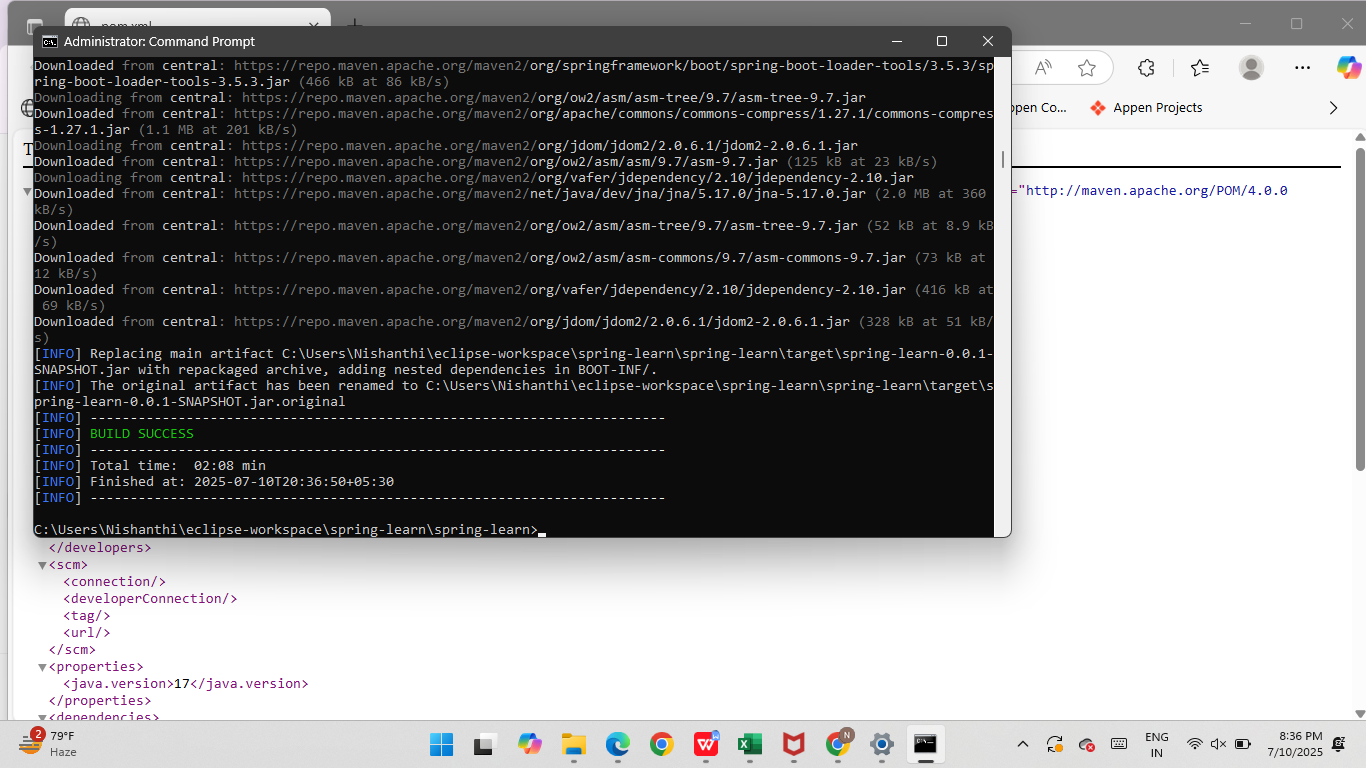
1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com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

**SOLUTION:**

**COMMAND LINE OUTPUT**



**package** - com.cognizant.springlearn

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) {

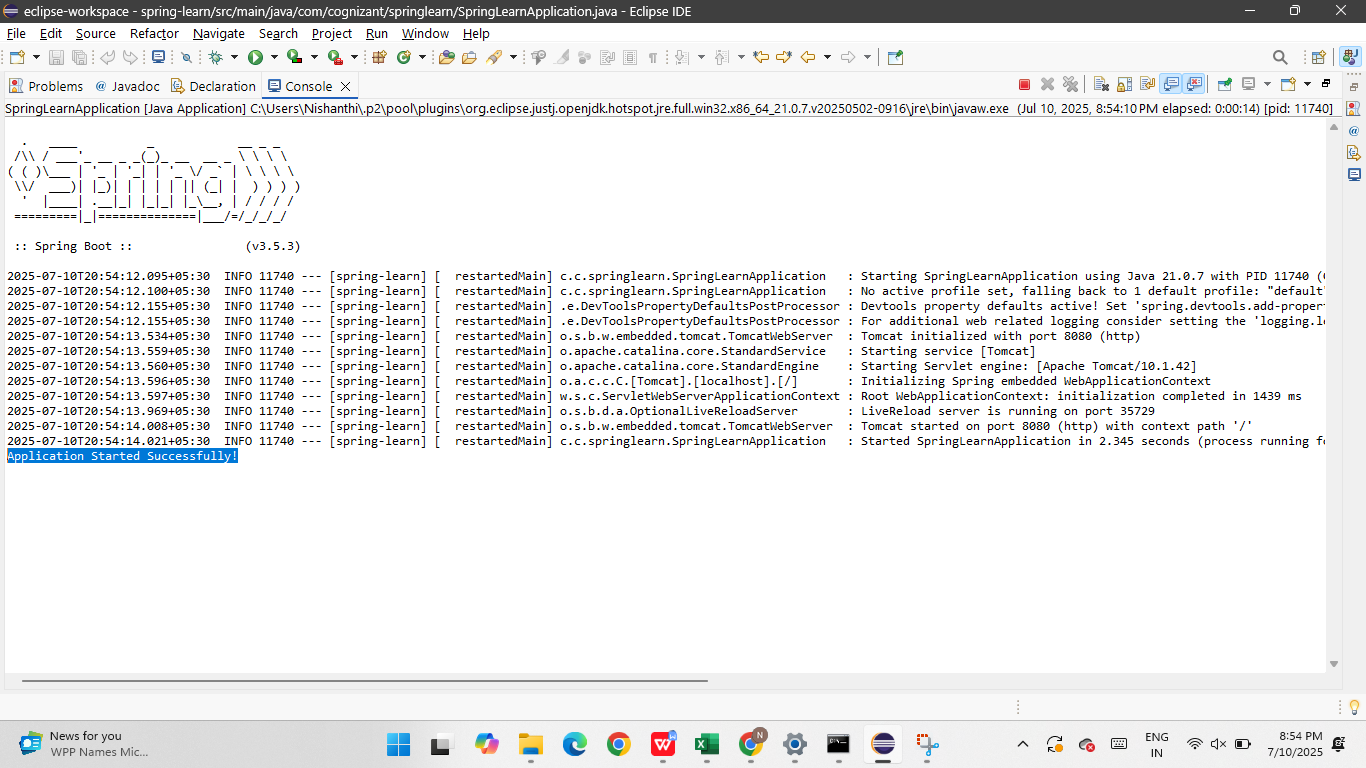
SpringApplication.*run*(SpringLearnApplication.**class**, args);

System.***out***.println("Application Started Successfully!");

}

}

**OUTPUT:**



**2.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.

**SOLUTION:**

PACKAGE - com.cognizant.springlearn

SpringLearnApplication.java

**package** com.cognizant.springlearn;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

*displayDate*(); // Call the method from main

}

**public** **static** **void** displayDate() {

// Load Spring XML configuration

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

// Get the dateFormat bean

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

**try** {

// Parse and print the date

Date date = format.parse("31/12/2018");

System.***out***.println("Parsed Date: " + date);

} **catch** (Exception e) {

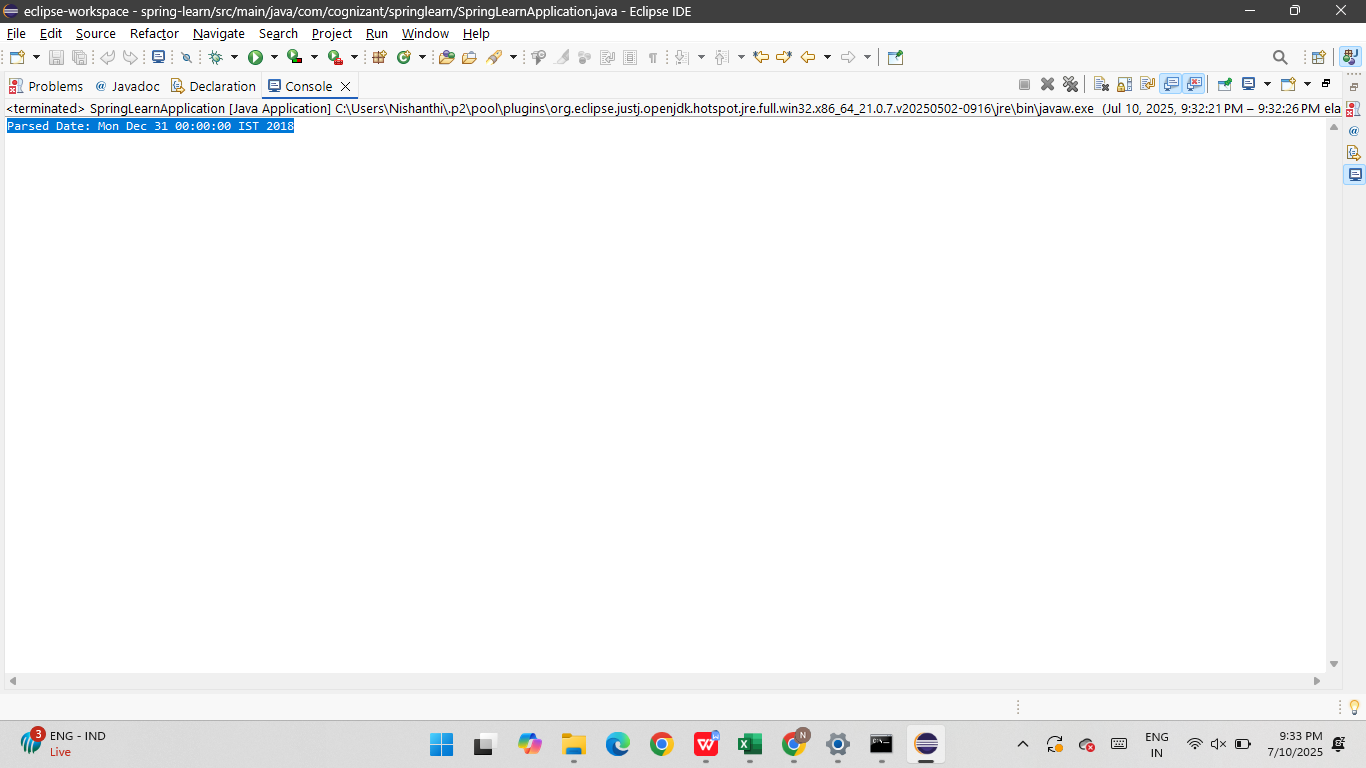
System.***out***.println("Error parsing date: " + e.getMessage());

}

}

}

**OUTPUT:**



3..**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

**SOLUTION:**

**Controller**

**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()");

**return** "Hello World!!";

}

}

**SpringBootMain.java**

**package** com.cognizant.springlearn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringBootMain {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootMain.**class**, args);

}

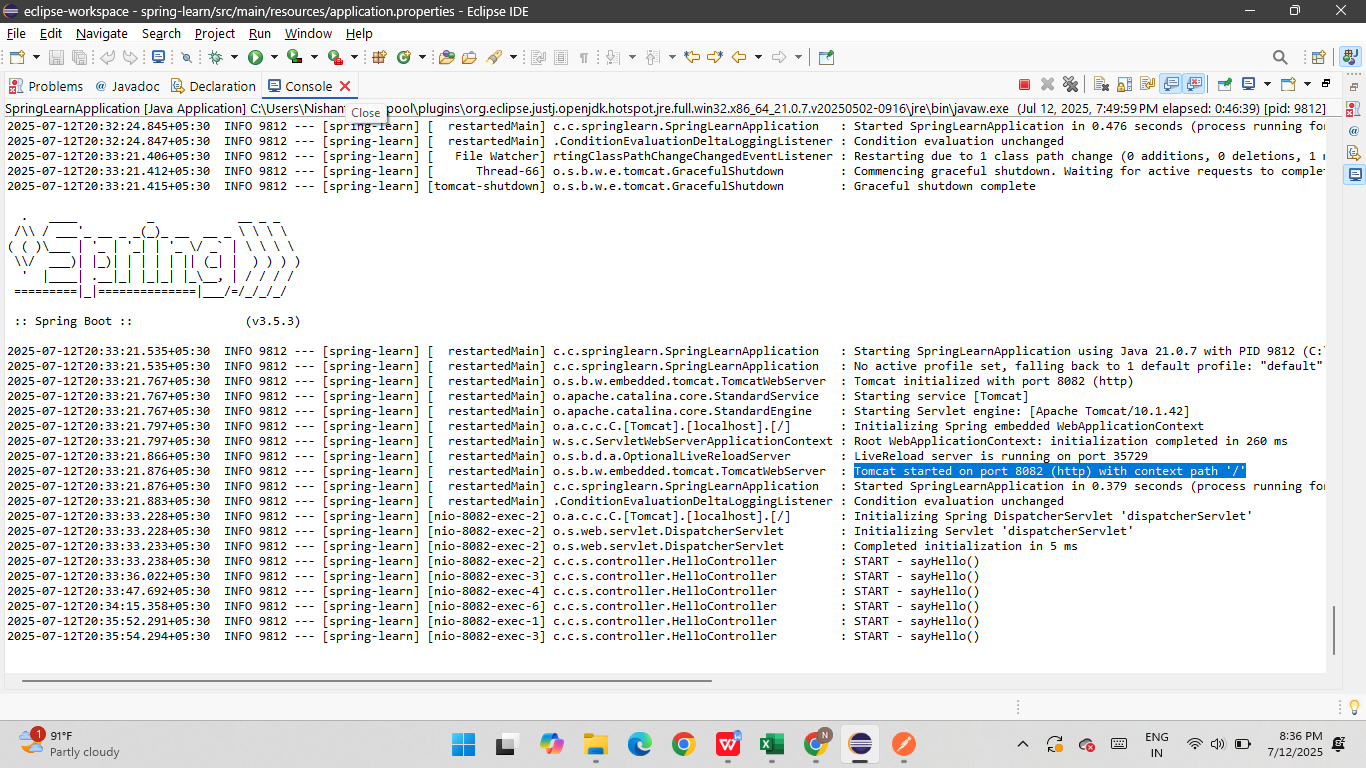
}

**Application.properties**

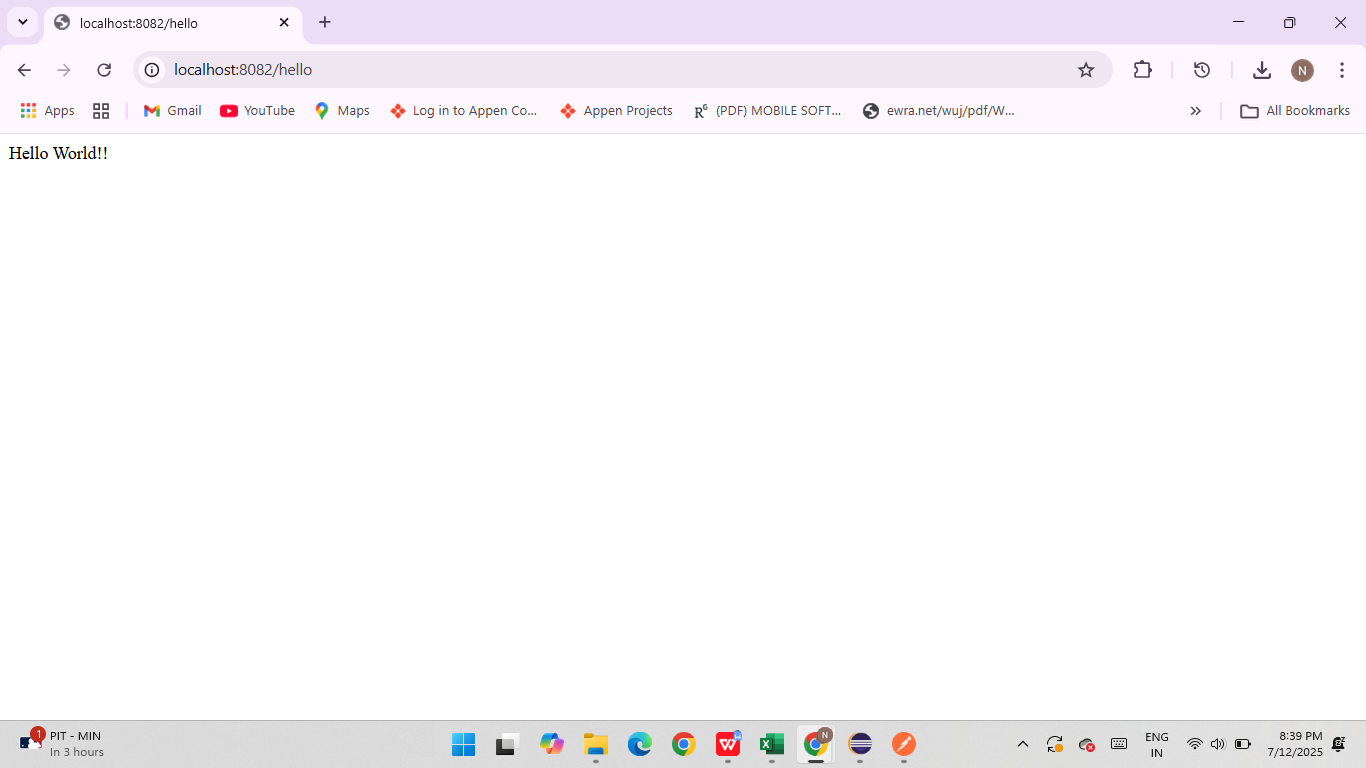
spring.application.name=spring-learn

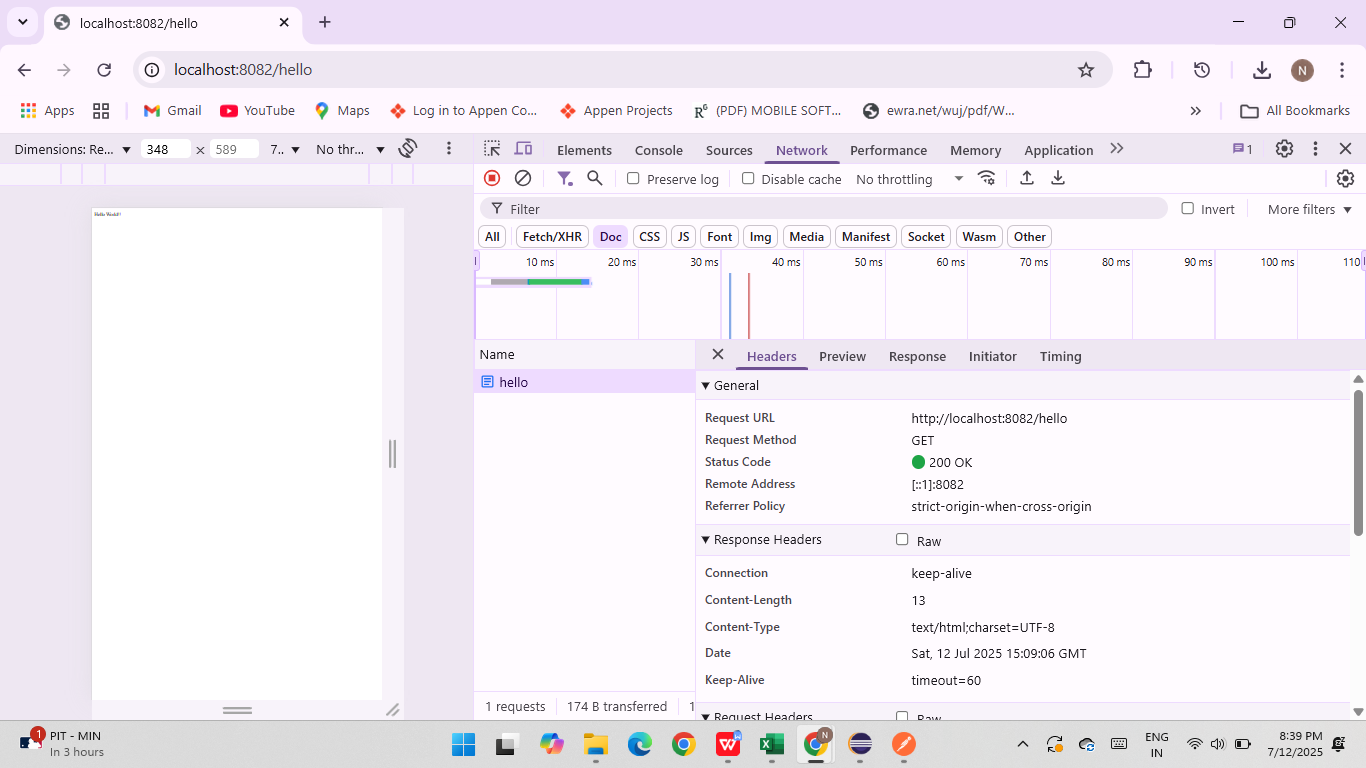
server.port=8082

**OUTPUT:**

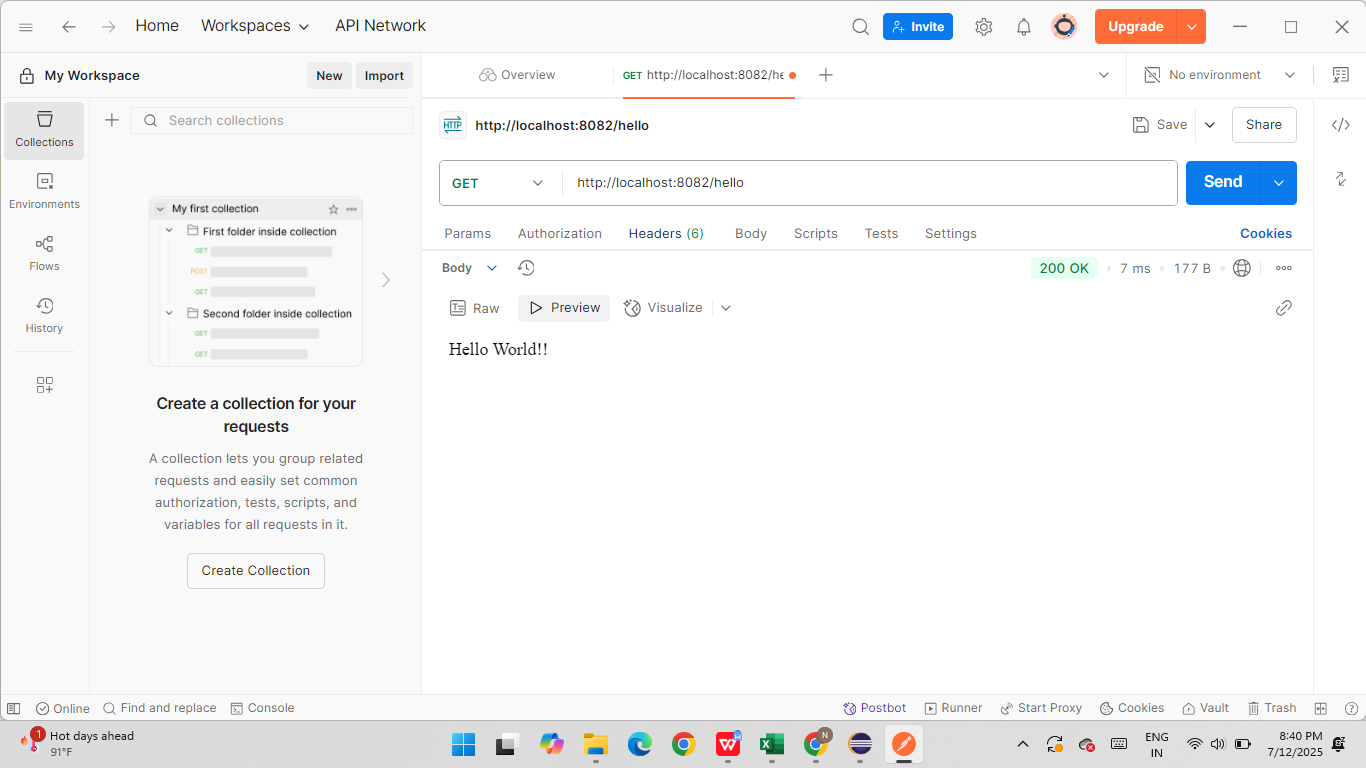


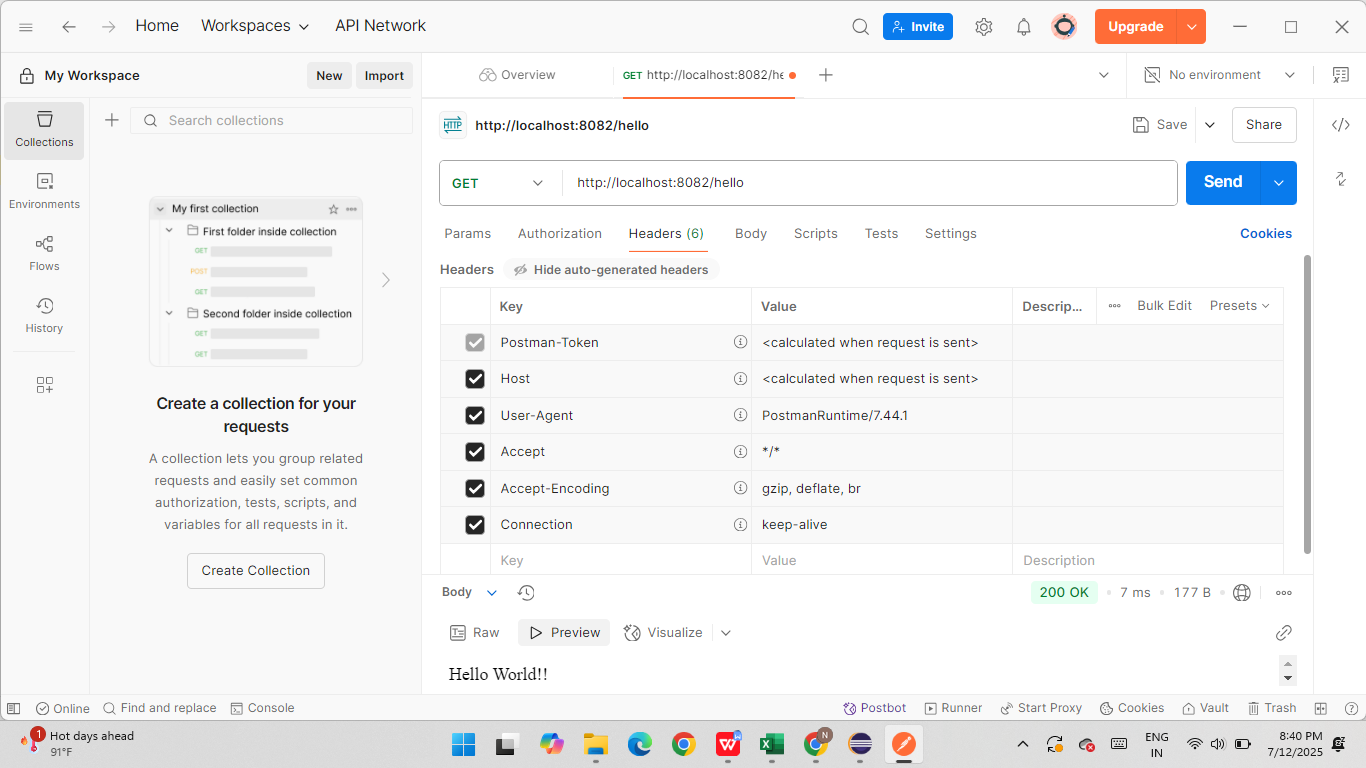
**CHROME OUTPUT**





POSTMAN OUTPUT:





**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

**SOLUTION:**

Com.cognizant.springlearn.model

Country.java

**package** com.cognizant.springlearn.model;

**public** **class** Country {

**private** String code;

**private** String name;

// Getters and 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;

}

}

com.cognizant.springlearn.controller

CountryController.java

**package** com.cognizant.springlearn.model;

**public** **class** Country {

**private** String code;

**private** String name;

// Getters and 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;

}

}

application.properties

spring.application.name=spring-learn

server.port=8091

SpringBootMain.java

**package** com.cognizant.springlearn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringBootMain {

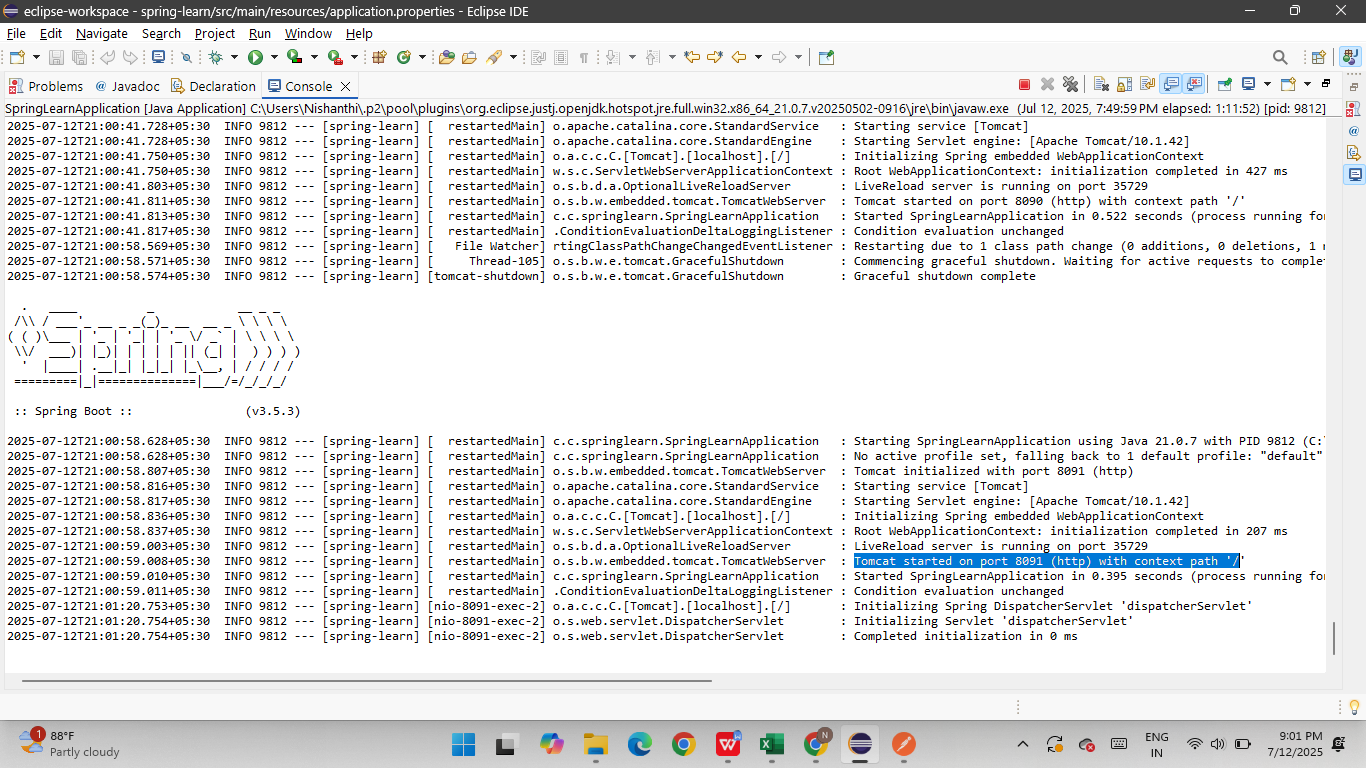
**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootMain.**class**, args);

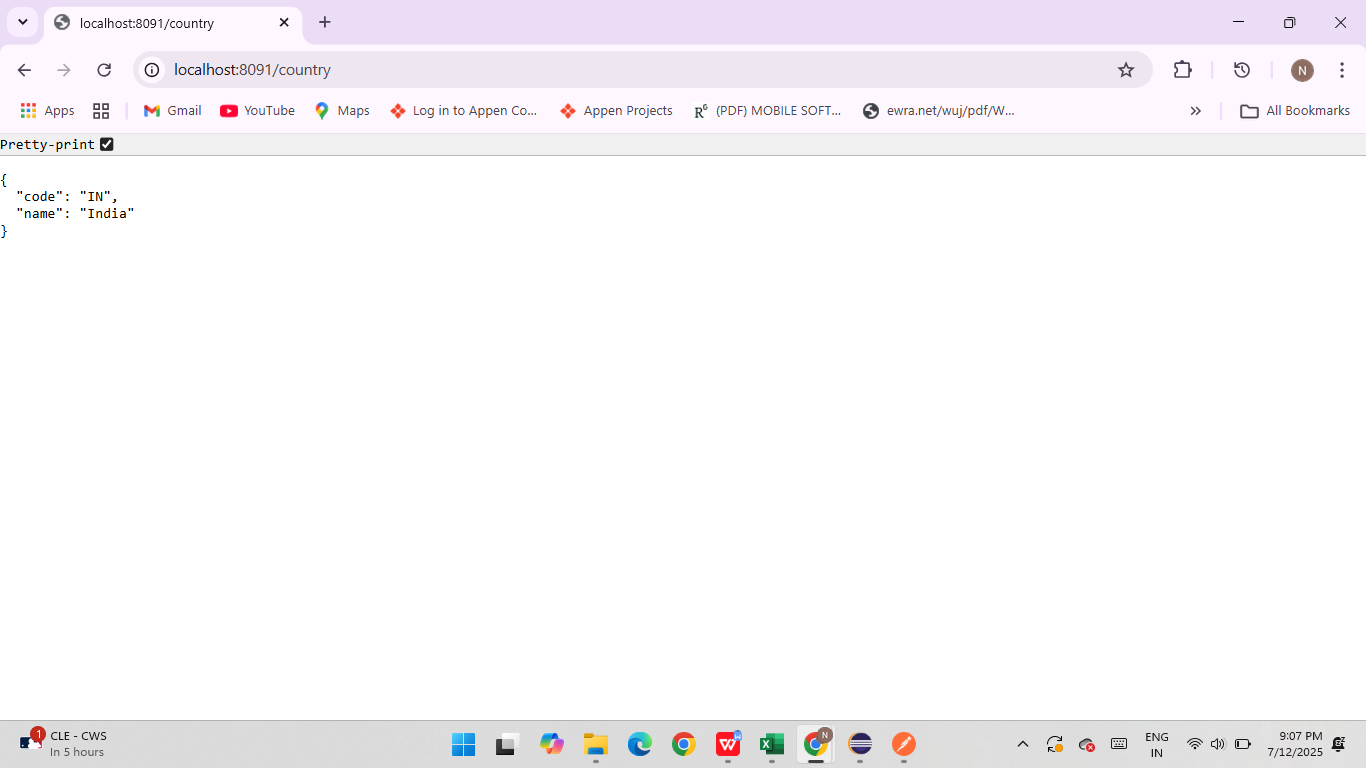
}

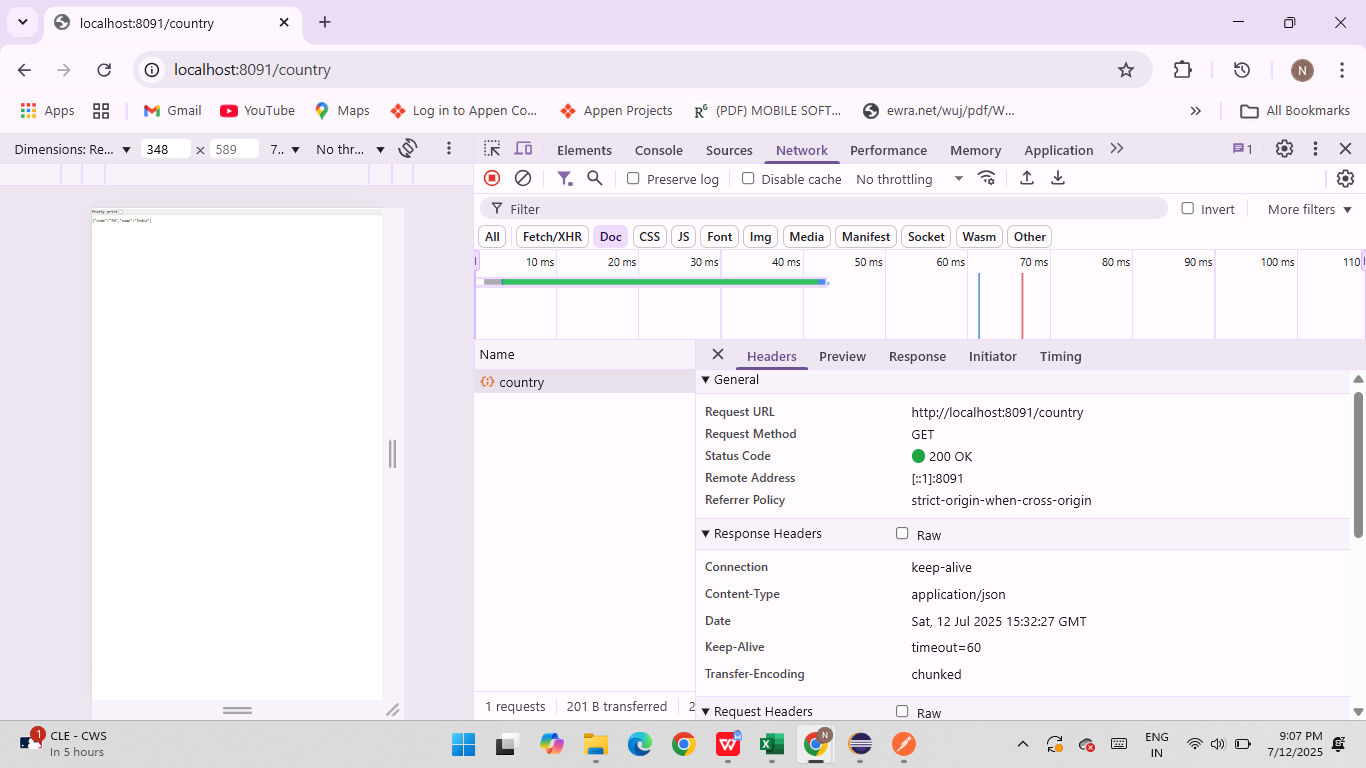
}

**OUTPUT:**



**CHROME OUTPUT:**





**POSTMAN OUTPUT**

