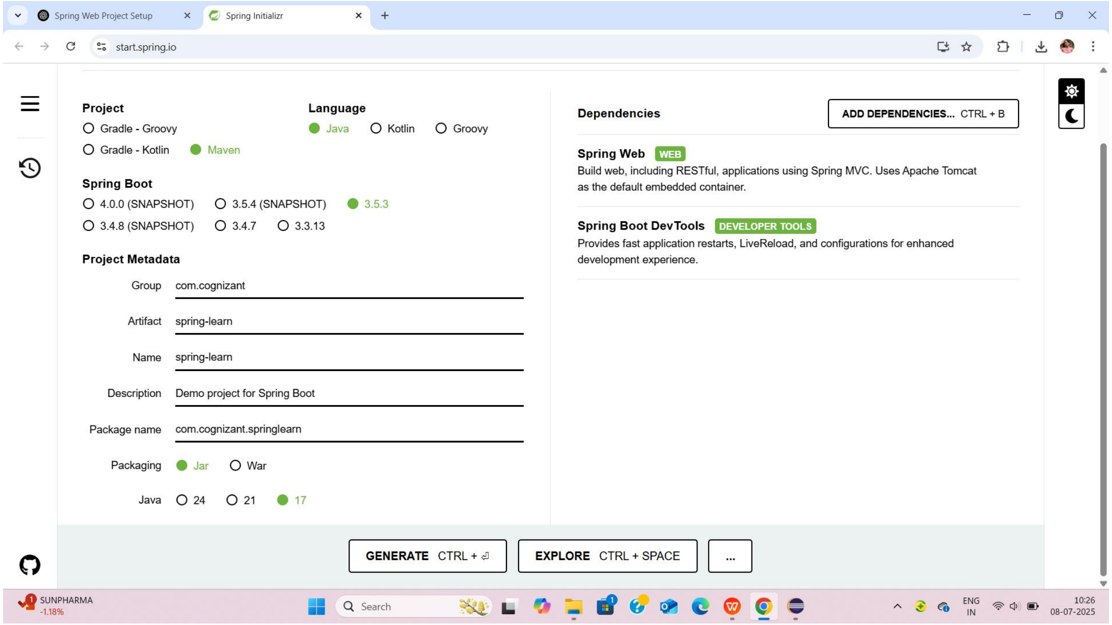
WEEK 4 HANDS ON

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.



Project
○ Gradle - Groovy
○ Gradle - Kotlin
● Maven

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 4.0.0 (SNAPSHOT) ○ 3.5.4 (SNAPSHOT) ● 3.5.3
○ 3.4.8 (SNAPSHOT) ○ 3.4.7 ○ 3.3.13

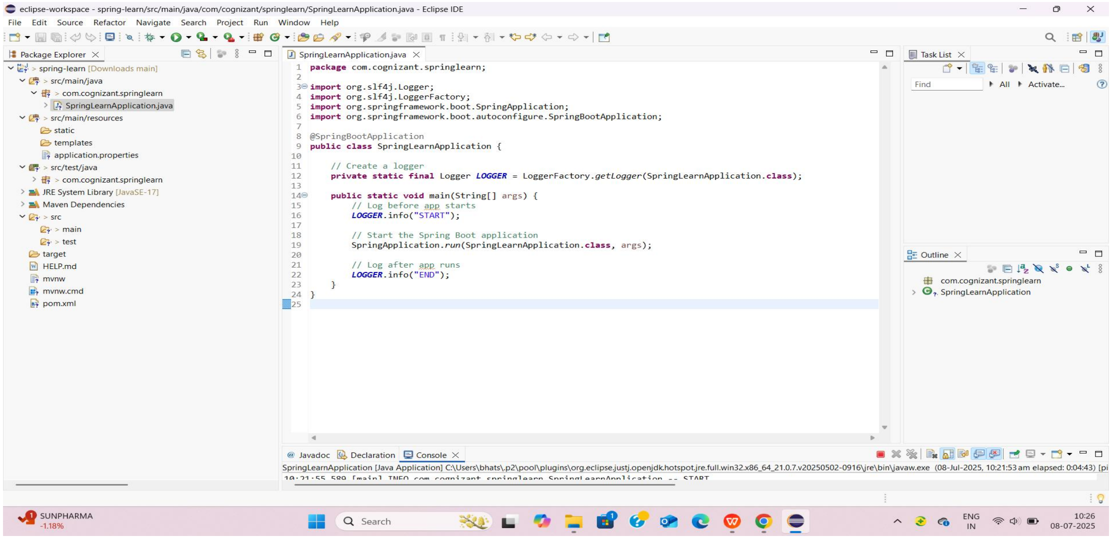
Project Metadata

Group: com.cognizant
Artifact: spring-learn
Name: spring-learn
Description: Demo project for Spring Boot
Package name: com.cognizant.springlearn
Packaging: ● Jar ○ War
Java: ○ 24 ○ 21 ● 17

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

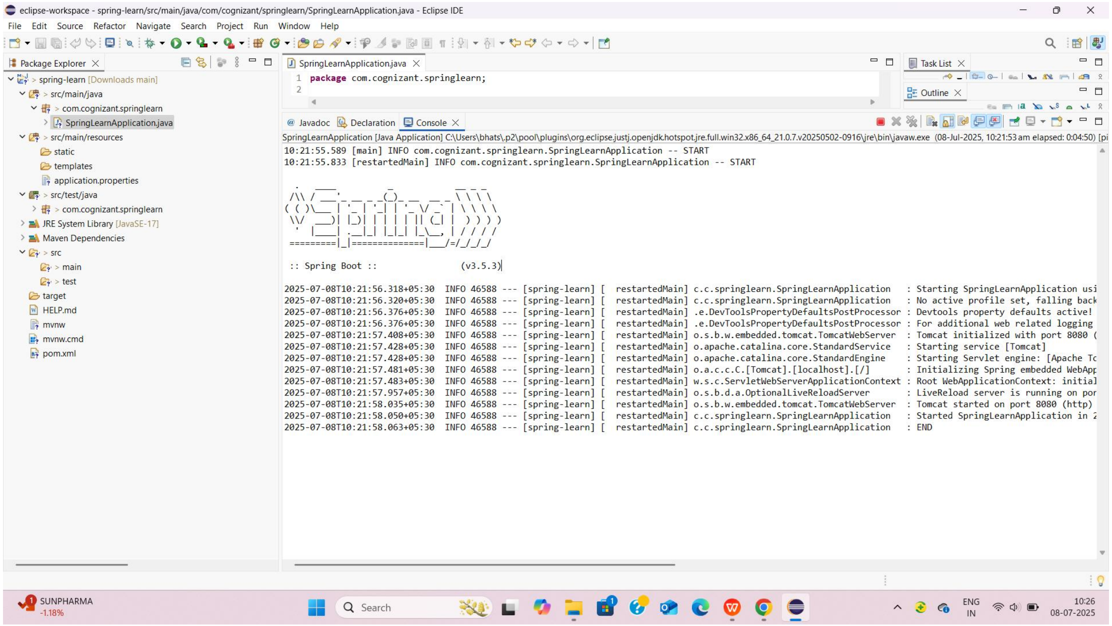
Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.



eclipse-workspace - spring-learn/src/main/java/com/cognizant/springlearn/SpringLearnApplication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
spring-learn [Downloads main]
src/main/java
com.cognizant.springlearn
SpringLearnApplication.java
src/main/resources
static
templates
application.properties
src/test/java
com.cognizant.springlearn
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringLearnApplication.java
Task List
Find
All
Activate...
Outline
com.cognizant.springlearn
SpringLearnApplication
Javadoc
Declaration
Console
SUNPHARMA
-1.18%
Search
ENG
IN
10:26
08-07-2025

```java
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication {

    // Create a logger
    private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

    public static void main(String[] args) {
        // Log before app starts
        LOGGER.info("START");

        // Start the Spring Boot application
        SpringApplication.run(SpringLearnApplication.class, args);

        // Log after app runs
        LOGGER.info("END");
    }
}

```

```
SpringLearnApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (08-Jul-2025, 10:21:53 am elapsed: 0:04:43) [pi
```



eclipse-workspace - spring-learn/src/main/java/com/cognizant/springlearn/SpringLearnApplication.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
spring-learn [Downloads main]
src/main/java
com.cognizant.springlearn
SpringLearnApplication.java
src/main/resources
static
templates
application.properties
src/test/java
com.cognizant.springlearn
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringLearnApplication.java
Task List
Outline
Javadoc
Declaration
Console
SUNPHARMA
-1.18%
Search
ENG
IN
10:26
08-07-2025

```java
package com.cognizant.springlearn;

```

```
SpringLearnApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (08-Jul-2025, 10:21:53 am elapsed: 0:04:50) [pi
10:21:55.589 [main] INFO com.cognizant.springlearn.SpringLearnApplication -- START
10:21:55.833 [restartedMain] INFO com.cognizant.springlearn.SpringLearnApplication -- START

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-08T10:21:56.318+05:30  INFO 46588 --- [spring-learn] [  restartedMain] c.c.springlearn.SpringLearnApplication   : Starting SpringLearnApplication usi
2025-07-08T10:21:56.320+05:30  INFO 46588 --- [spring-learn] [  restartedMain] c.c.springlearn.SpringLearnApplication   : No active profile set, falling back
2025-07-08T10:21:56.376+05:30  INFO 46588 --- [spring-learn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active!
2025-07-08T10:21:56.376+05:30  INFO 46588 --- [spring-learn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging
2025-07-08T10:21:57.408+05:30  INFO 46588 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (
2025-07-08T10:21:57.428+05:30  INFO 46588 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-08T10:21:57.428+05:30  INFO 46588 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tc
2025-07-08T10:21:57.481+05:30  INFO 46588 --- [spring-learn] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApp
2025-07-08T10:21:57.483+05:30  INFO 46588 --- [spring-learn] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initial
2025-07-08T10:21:57.957+05:30  INFO 46588 --- [spring-learn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on por
2025-07-08T10:21:58.035+05:30  INFO 46588 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http)
2025-07-08T10:21:58.050+05:30  INFO 46588 --- [spring-learn] [  restartedMain] c.c.springlearn.SpringLearnApplication   : Started SpringLearnApplication in 2
2025-07-08T10:21:58.063+05:30  INFO 46588 --- [spring-learn] [  restartedMain] c.c.springlearn.SpringLearnApplication   : END
```

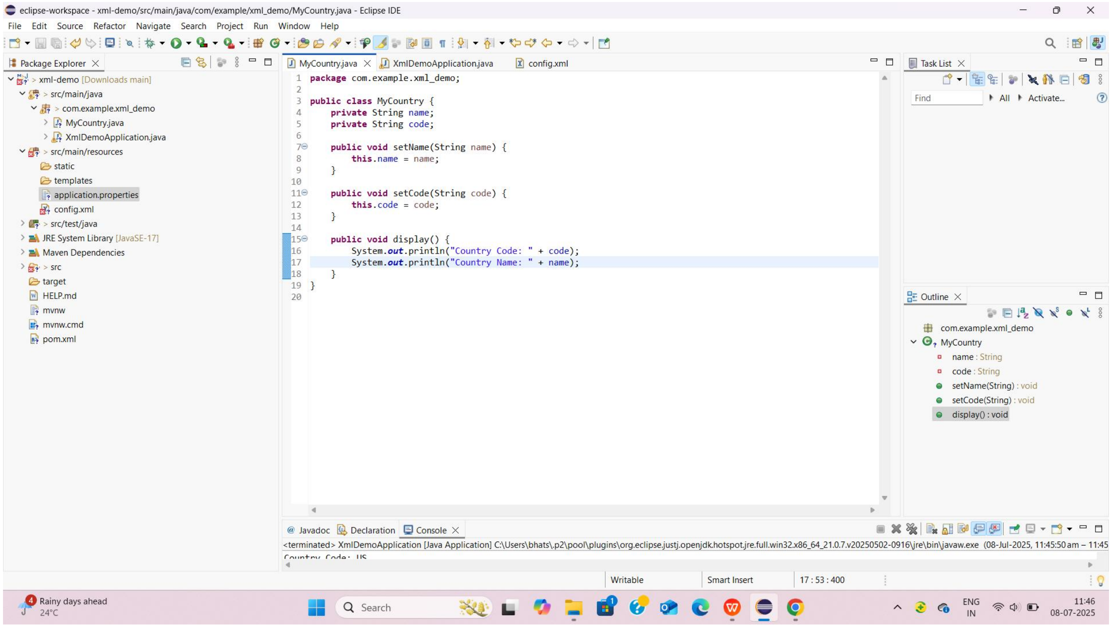
2.Spring Core – Load Country from Spring Configuration XML

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

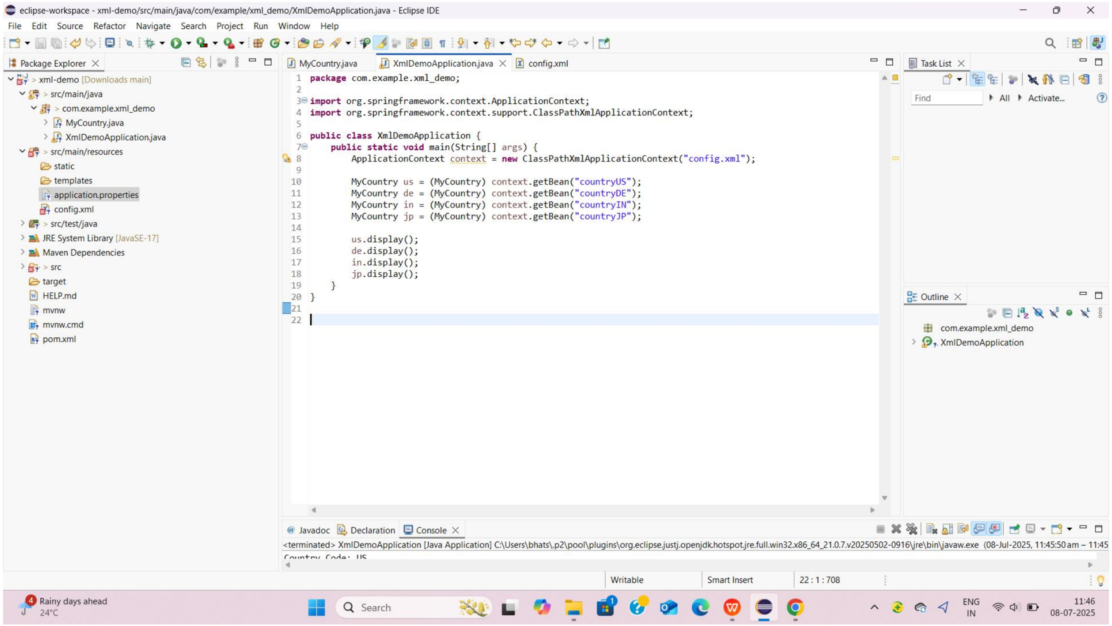
Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.



eclipse-workspace - xml-demo/src/main/java/com/example/xml_demo/MyCountry.java - Eclipse IDE

```java
package com.example.xml_demo;

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

    public void setName(String name) {
        this.name = name;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public void display() {
        System.out.println("Country Code: " + code);
        System.out.println("Country Name: " + name);
    }
}
```

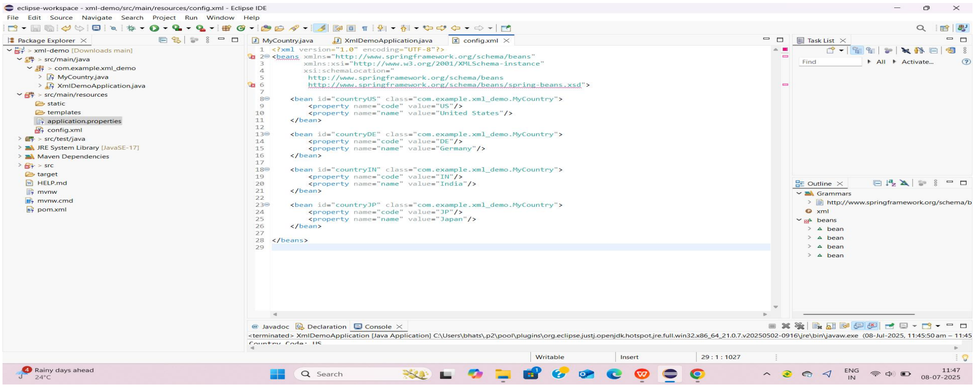
```
<terminated> XmlDemoApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 11:45:50 am – 11:45
Country Code: US
```



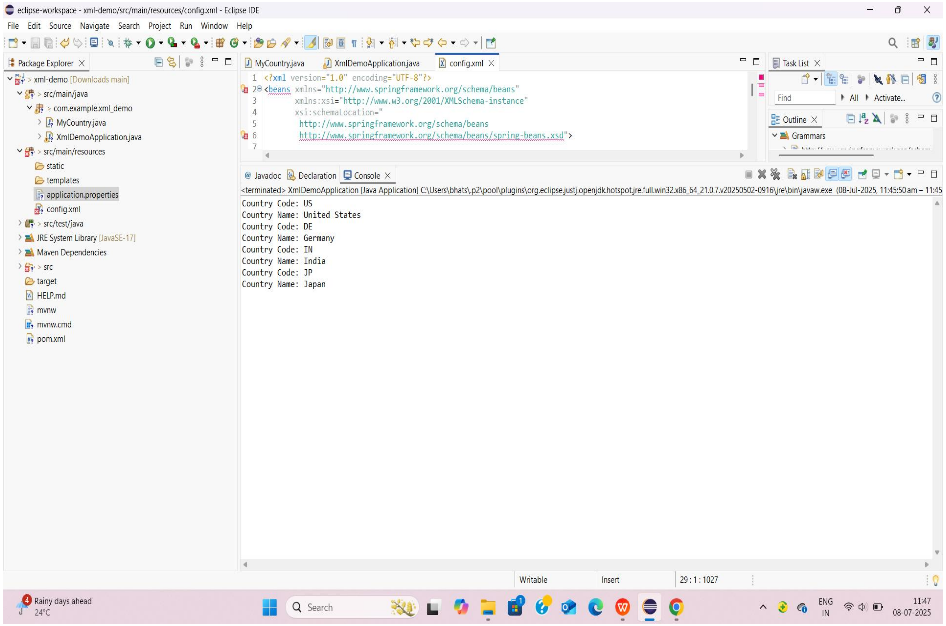
eclipse-workspace - xml-demo/src/main/java/com/example/xml_demo/XmlDemoApplication.java - Eclipse IDE

```java
package com.example.xml_demo;

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

public class XmlDemoApplication {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");

        MyCountry us = (MyCountry) context.getBean("countryUS");
        MyCountry de = (MyCountry) context.getBean("countryDE");
        MyCountry in = (MyCountry) context.getBean("countryIN");
        MyCountry jp = (MyCountry) context.getBean("countryJP");

        us.display();
        de.display();
        in.display();
        jp.display();
    }
}

```

```
<terminated> XmlDemoApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 11:45:50 am – 11:45
Country Code: US
```



```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="countryUS" class="com.example.xml_demo.MyCountry">
        <property name="code" value="US"/>
        <property name="name" value="United States"/>
    </bean>

    <bean id="countryDE" class="com.example.xml_demo.MyCountry">
        <property name="code" value="DE"/>
        <property name="name" value="Germany"/>
    </bean>

    <bean id="countryIN" class="com.example.xml_demo.MyCountry">
        <property name="code" value="IN"/>
        <property name="name" value="India"/>
    </bean>

    <bean id="countryJP" class="com.example.xml_demo.MyCountry">
        <property name="code" value="JP"/>
        <property name="name" value="Japan"/>
    </bean>

</beans>
```



```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">
```

```
<terminated> XmlDemoApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 11:45:50 am – 11:45
Country Code: US
Country Name: United States
Country Code: DE
Country Name: Germany
Country Code: IN
Country Name: India
Country Code: JP
Country Name: Japan
```

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!!

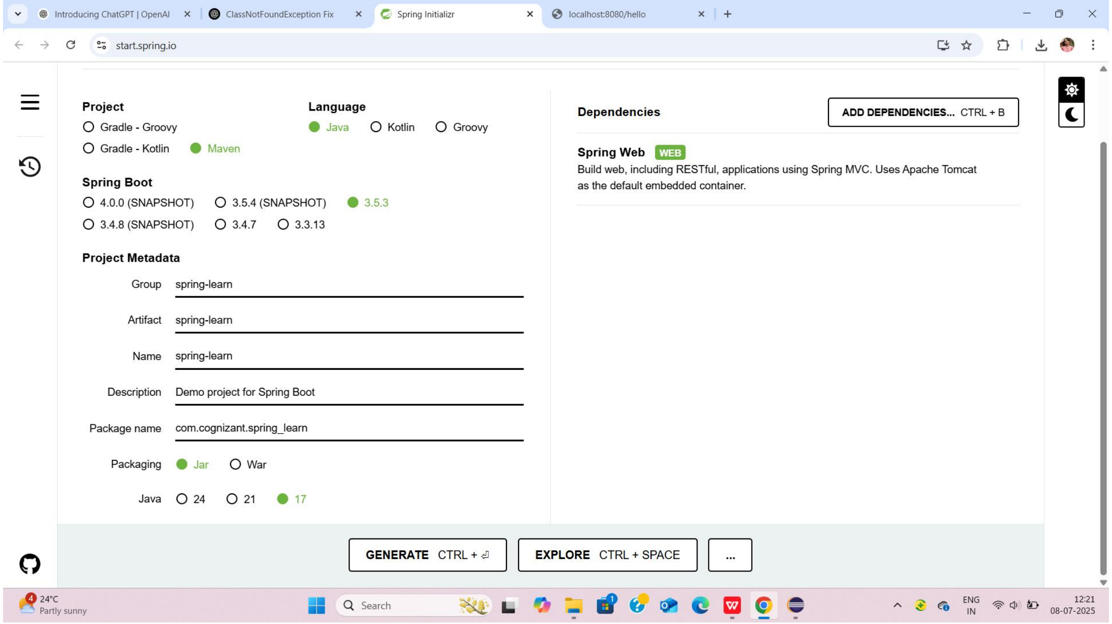
IMPORTANT NOTE: Don't forget to include start and end log in the sayHello() method.

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.



Project
○ Gradle - Groovy
○ Gradle - Kotlin
● Maven

Language
● Java
○ Kotlin
○ Groovy

Spring Boot
○ 4.0.0 (SNAPSHOT)
○ 3.5.4 (SNAPSHOT)
● 3.5.3
○ 3.4.8 (SNAPSHOT)
○ 3.4.7
○ 3.3.13

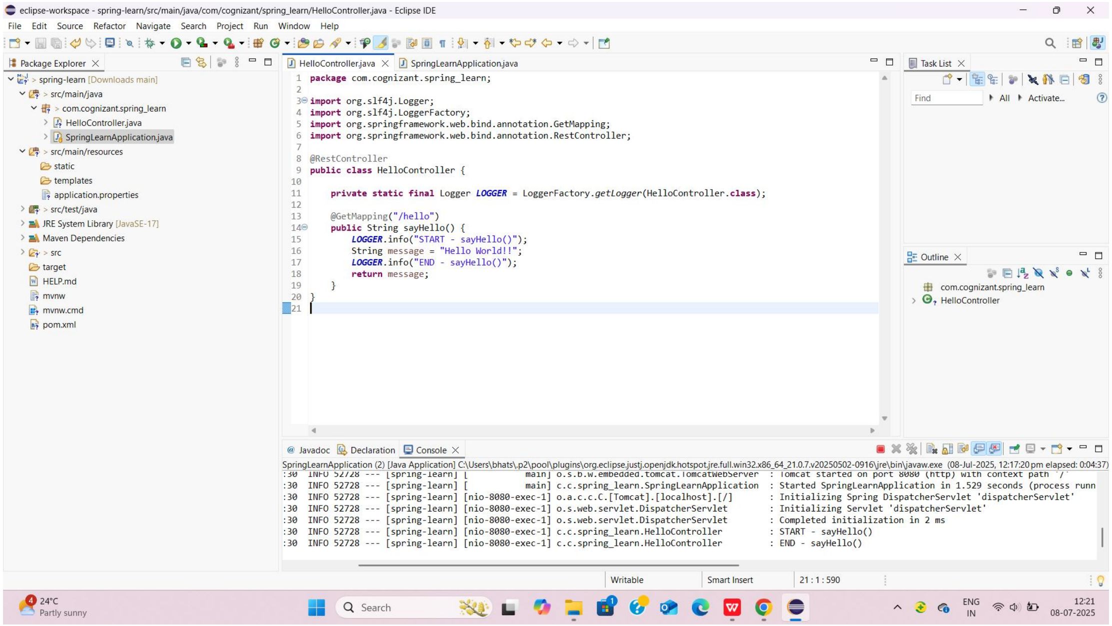
Project Metadata

Group	spring-learn
Artifact	spring-learn
Name	spring-learn
Description	Demo project for Spring Boot
Package name	com.cognizant.spring_learn
Packaging	● Jar ○ War
Java	○ 24 ○ 21 ● 17

Dependencies

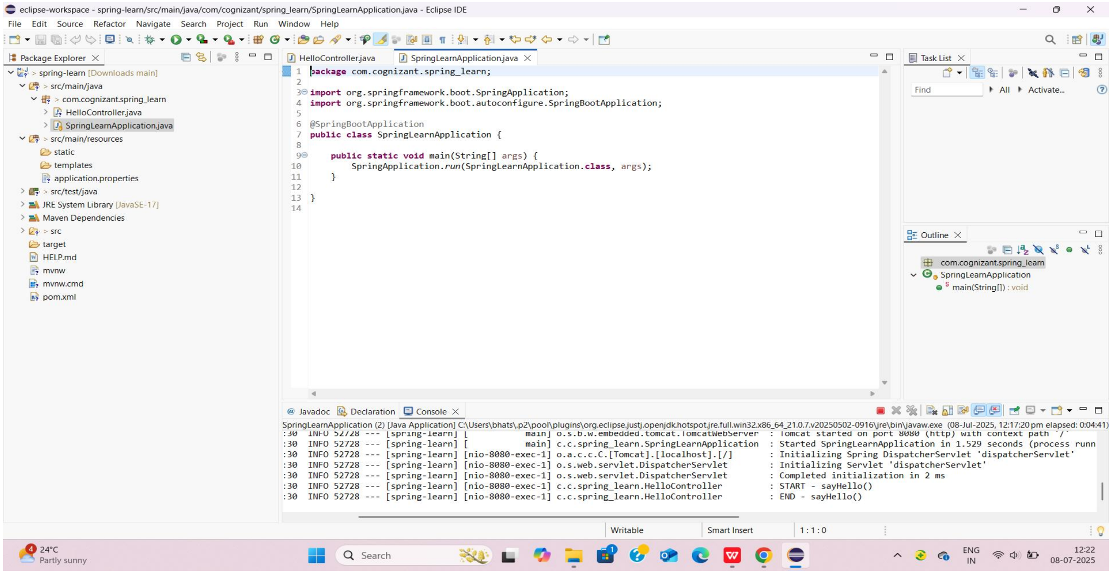
Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.



```java
package com.cognizant.spring_learn;

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

```
SpringLearnApplication (2) [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (08-Jul-2025, 12:17:20 pm elapsed: 0:04:37)
:30  INFO 52728 --- [spring-learn] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
:30  INFO 52728 --- [spring-learn] [           main] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 1.529 seconds (process runn
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] c.c.spring_learn.HelloController         : START - sayHello()
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] c.c.spring_learn.HelloController         : END - sayHello()
```



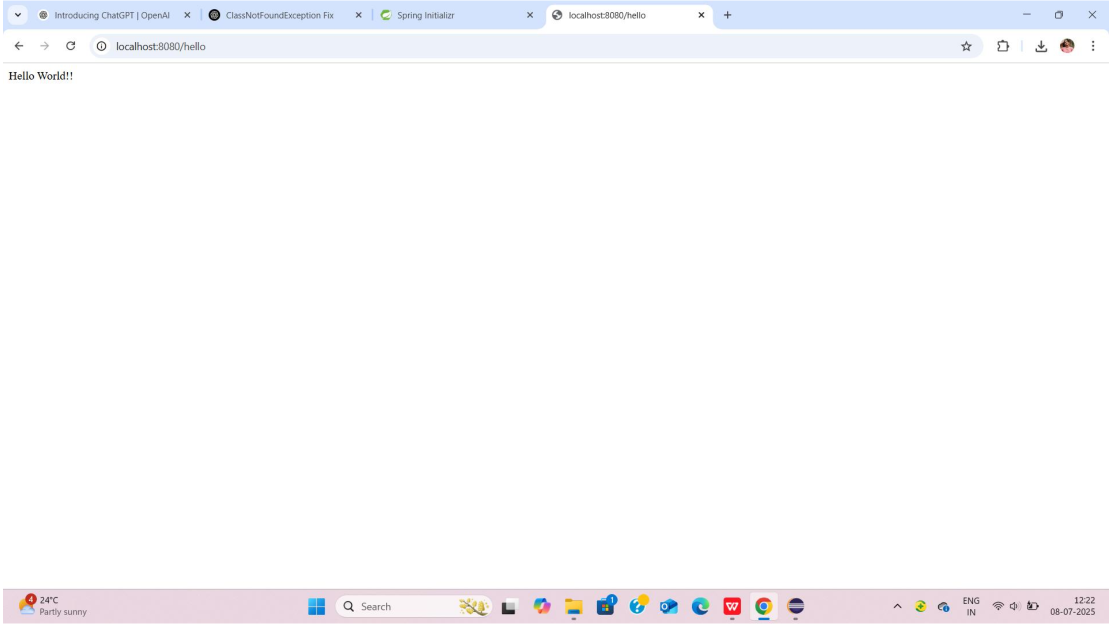
eclipse-workspace - spring-learn/src/main/java/com/cognizant/spring_learn/SpringLearnApplication.java - Eclipse IDE

```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);
    }

}
```

SpringLearnApplication (2) [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 12:17:20 pm elapsed: 0:04:41)

```
:30  INFO 52728 --- [spring-learn] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
:30  INFO 52728 --- [spring-learn] [           main] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 1.529 seconds (process runn
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] c.c.spring_learn.HelloController         : START - sayHello()
:30  INFO 52728 --- [spring-learn] [nio-8080-exec-1] c.c.spring_learn.HelloController         : END - sayHello()
```

OUTPUT



localhost:8080/hello

Hello World!!

4.REST - Country Web Service

Write a REST service that returns India country details in the earlier created spring learn application.

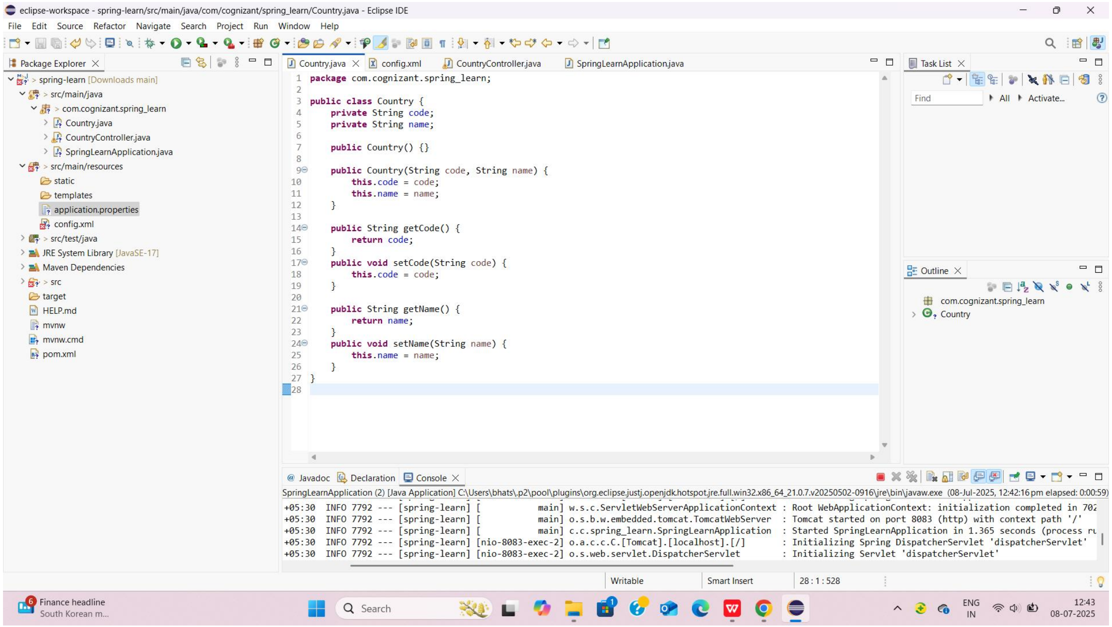
URL: /country
Controller: com.cognizant.spring-learn.controller.CountryController
Method Annotation: @RequestMapping

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:

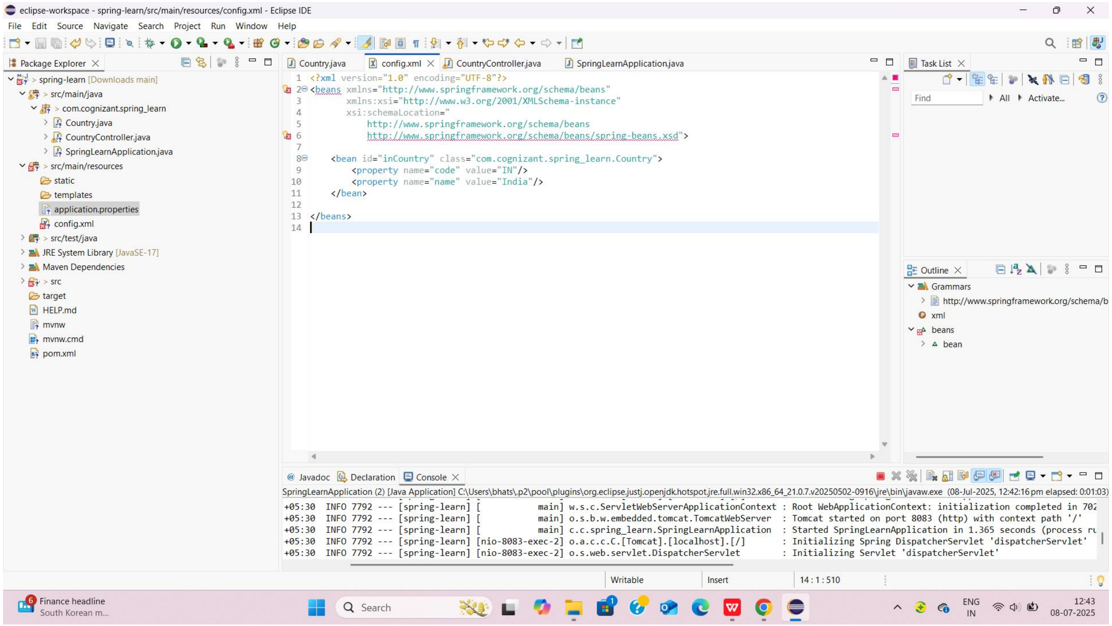
- What happens in the controller method?
- How the bean is converted into JSON reponse?
- In network tab of developer tools show the HTTP header details received
- In postman click on "Headers" tab to view the HTTP header details received.



```java
package com.cognizant.spring_learn;

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;
    }
}
```

SpringLearnApplication (2) [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 12:42:16 pm elapsed: 0:00:59)

```
+05:30  INFO 7792 --- [spring-learn] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 702
+05:30  INFO 7792 --- [spring-learn] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8083 (http) with context path '/'
+05:30  INFO 7792 --- [spring-learn] [           main] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 1.365 seconds (process ru
+05:30  INFO 7792 --- [spring-learn] [nio-8083-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
+05:30  INFO 7792 --- [spring-learn] [nio-8083-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
```

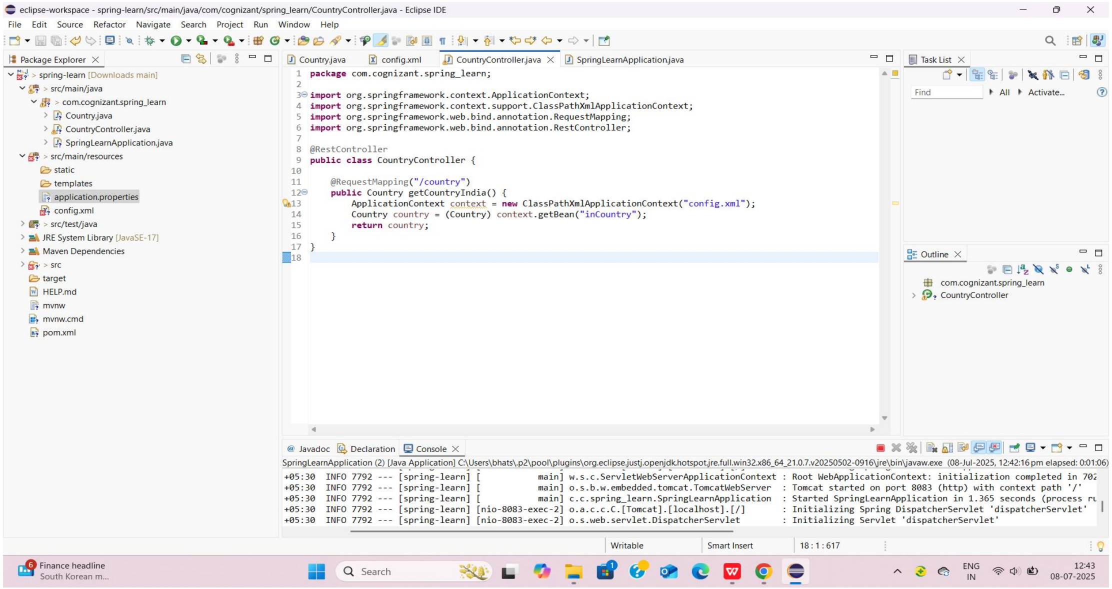


eclipse-workspace - spring-learn/src/main/resources/config.xml - Eclipse IDE

```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="inCountry" class="com.cognizant.spring_learn.Country">
        <property name="code" value="IN"/>
        <property name="name" value="India"/>
    </bean>

</beans>

```

SpringLearnApplication (2) [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 12:42:16 pm elapsed: 0:01:03)

```
+05:30  INFO 7792 --- [spring-learn] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 702
+05:30  INFO 7792 --- [spring-learn] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8083 (http) with context path '/'
+05:30  INFO 7792 --- [spring-learn] [           main] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 1.365 seconds (process ru
+05:30  INFO 7792 --- [spring-learn] [nio-8083-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
+05:30  INFO 7792 --- [spring-learn] [nio-8083-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
```



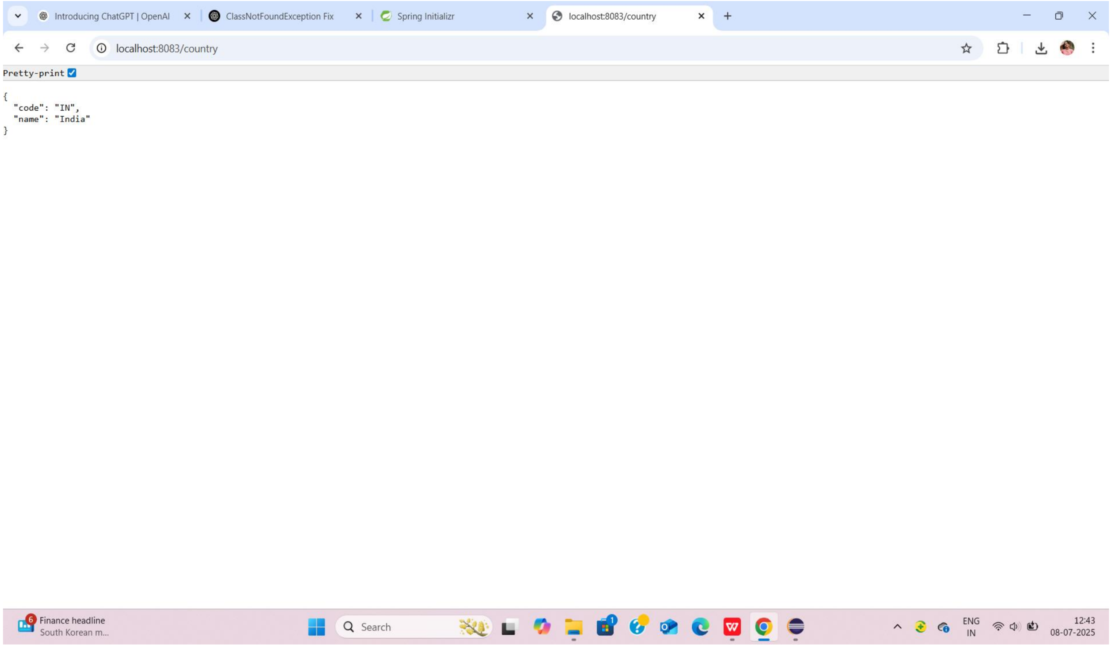
eclipse-workspace - spring-learn/src/main/java/com/cognizant/spring_learn/CountryController.java - Eclipse IDE

```java
package com.cognizant.spring_learn;

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 {

    @RequestMapping("/country")
    public Country getCountryIndia() {
        ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");
        Country country = (Country) context.getBean("inCountry");
        return country;
    }
}

```

SpringLearnApplication (2) [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 12:42:16 pm elapsed: 0:01:06)

```
+05:30  INFO 7792 --- [spring-learn] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 702
+05:30  INFO 7792 --- [spring-learn] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8083 (http) with context path '/'
+05:30  INFO 7792 --- [spring-learn] [           main] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 1.365 seconds (process ru
+05:30  INFO 7792 --- [spring-learn] [nio-8083-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
+05:30  INFO 7792 --- [spring-learn] [nio-8083-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
```

OUTPUT



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

5.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.

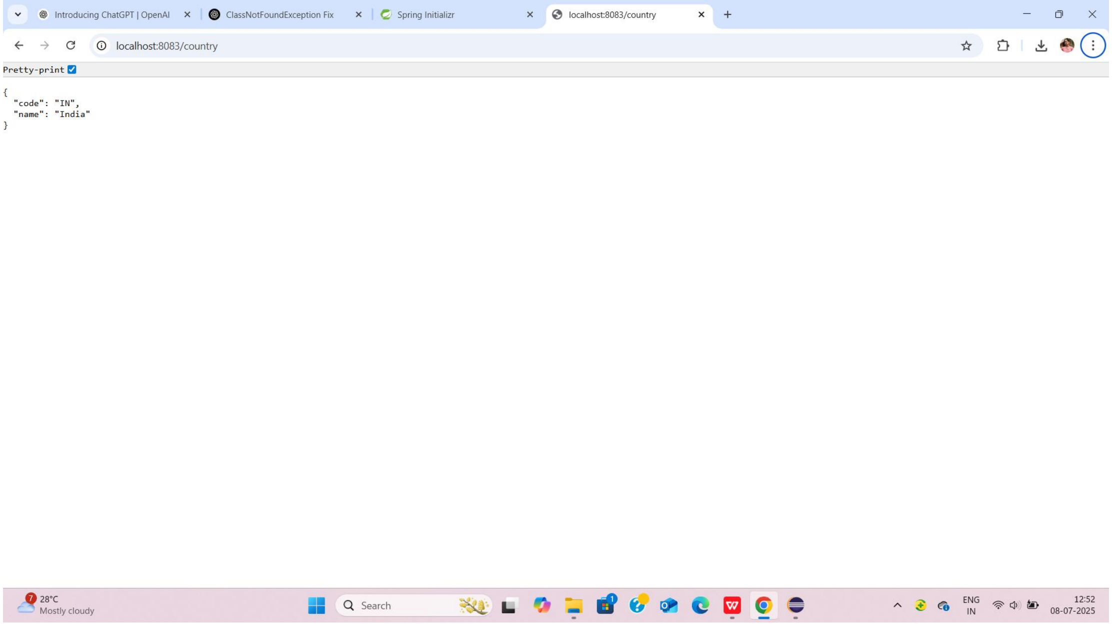
- Lambda expression can also be used instead of iterating the country list

Sample Request: http://localhost:8083/country/in

Sample Response:{

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

Output:



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

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

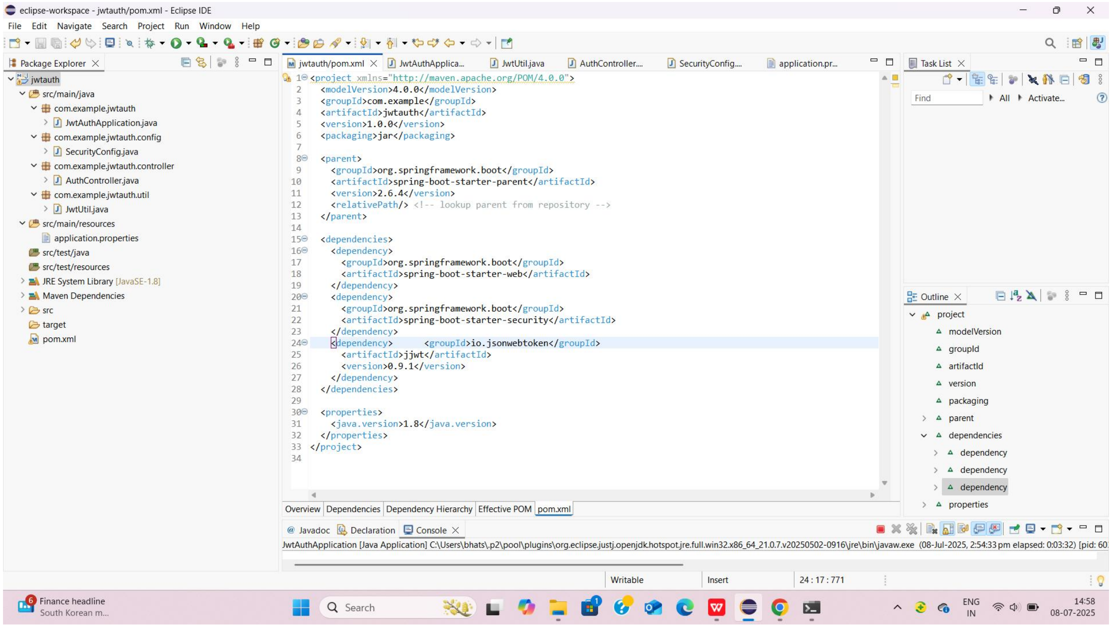
```
curl -s -u user:pwd http://localhost:8090/authenticate
```

Response

```
{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJl
eHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}
```

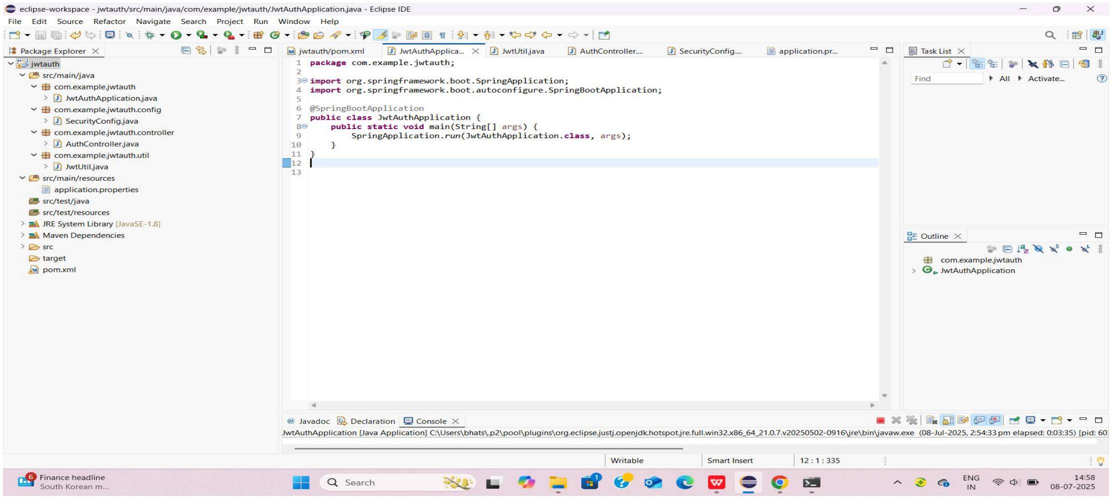
This can be incorporated as three major steps:

- 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



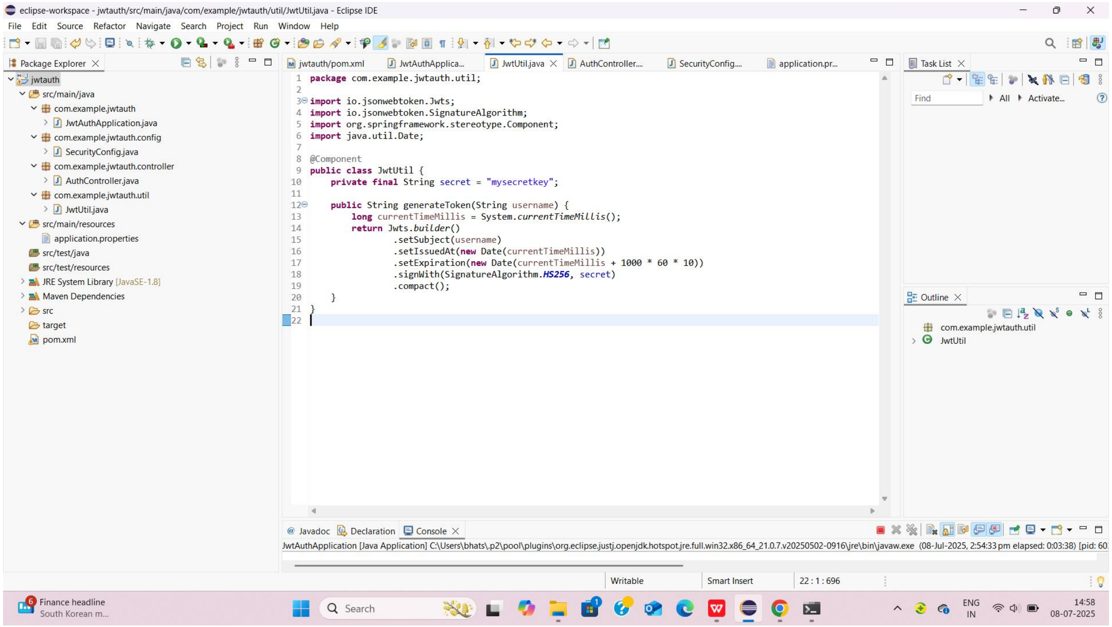
```
<project xmlns="http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>jwtauth</artifactId>
  <version>1.0.0</version>
  <packaging>jar</packaging>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>      <groupId>io.jsonwebtoken</groupId>
      <artifactId>jjwt</artifactId>
      <version>0.9.1</version>
    </dependency>
  </dependencies>

  <properties>
    <java.version>1.8</java.version>
  </properties>
</project>

```

```
JwtAuthApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (08-Jul-2025, 2:54:33 pm elapsed: 0:03:32) [pid: 60
```



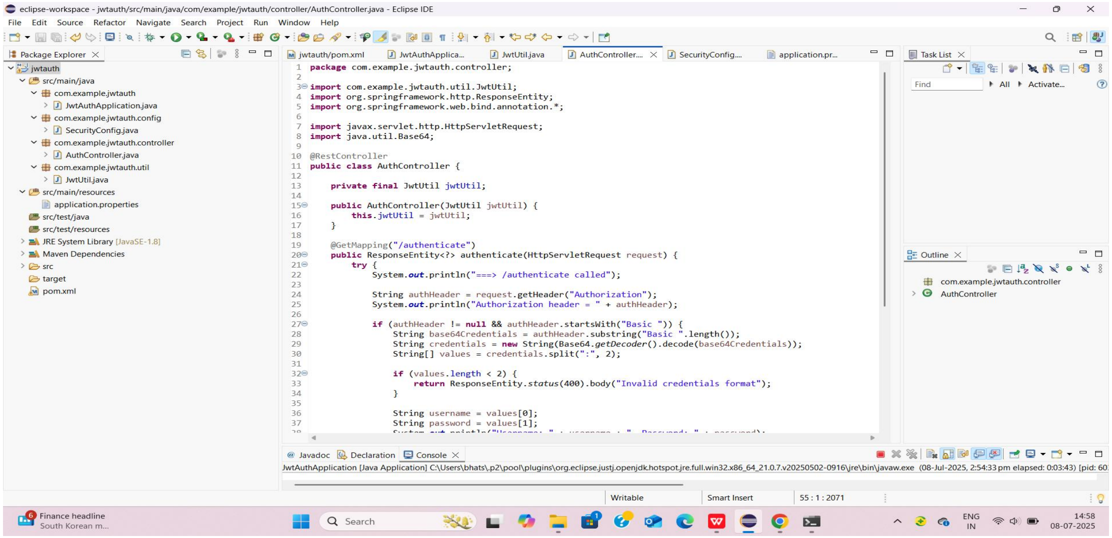
```java
package com.example.jwtauth;

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

@SpringBootApplication
public class JwtAuthApplication {
    public static void main(String[] args) {
        SpringApplication.run(JwtAuthApplication.class, args);
    }
}

```

```
JwtAuthApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (08-Jul-2025, 2:54:33 pm elapsed: 0:03:35) [pid: 60
```



```java
package com.example.jwtauth.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 = "mysecretkey";

    public String generateToken(String username) {
        long currentTimeMillis = System.currentTimeMillis();
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date(currentTimeMillis))
                .setExpiration(new Date(currentTimeMillis + 1000 * 60 * 10))
                .signWith(SignatureAlgorithm.HS256, secret)
                .compact();
    }
}

```

```
JwtAuthApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (08-Jul-2025, 2:54:33 pm elapsed: 0:03:38) [pid: 60
```



```java
package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.util.Base64;

@RestController
public class AuthController {

    private final JwtUtil jwtUtil;

    public AuthController(JwtUtil jwtUtil) {
        this.jwtUtil = jwtUtil;
    }

    @GetMapping("/authenticate")
    public ResponseEntity<?> authenticate(HttpServletRequest request) {
        try {
            System.out.println("===> /authenticate called");

            String authHeader = request.getHeader("Authorization");
            System.out.println("Authorization header = " + authHeader);

            if (authHeader != null && authHeader.startsWith("Basic ")) {
                String base64Credentials = authHeader.substring("Basic ".length());
                String credentials = new String(Base64.getDecoder().decode(base64Credentials));
                String[] values = credentials.split(":", 2);

                if (values.length < 2) {
                    return ResponseEntity.status(400).body("Invalid credentials format");
                }

                String username = values[0];
                String password = values[1];
                System.out.println("Username: " + username + ", Password: " + password);
```

```
JwtAuthApplication [Java Application] C:\Users\bhats\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (08-Jul-2025, 2:54:33 pm elapsed: 0:03:43) [pid: 60
```