**SPRING REST 1 HandsON Solutions**

**Hands on 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.

**Code:  
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);

}

}

**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 REST</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</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>

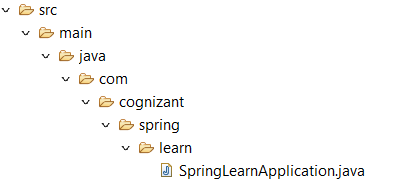
</plugins>

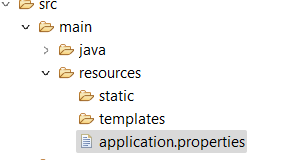
</build>

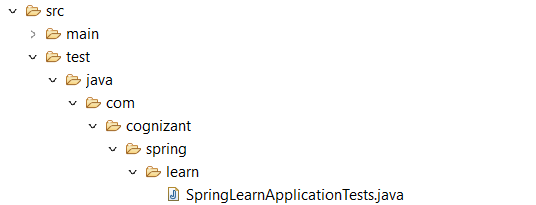
</project>

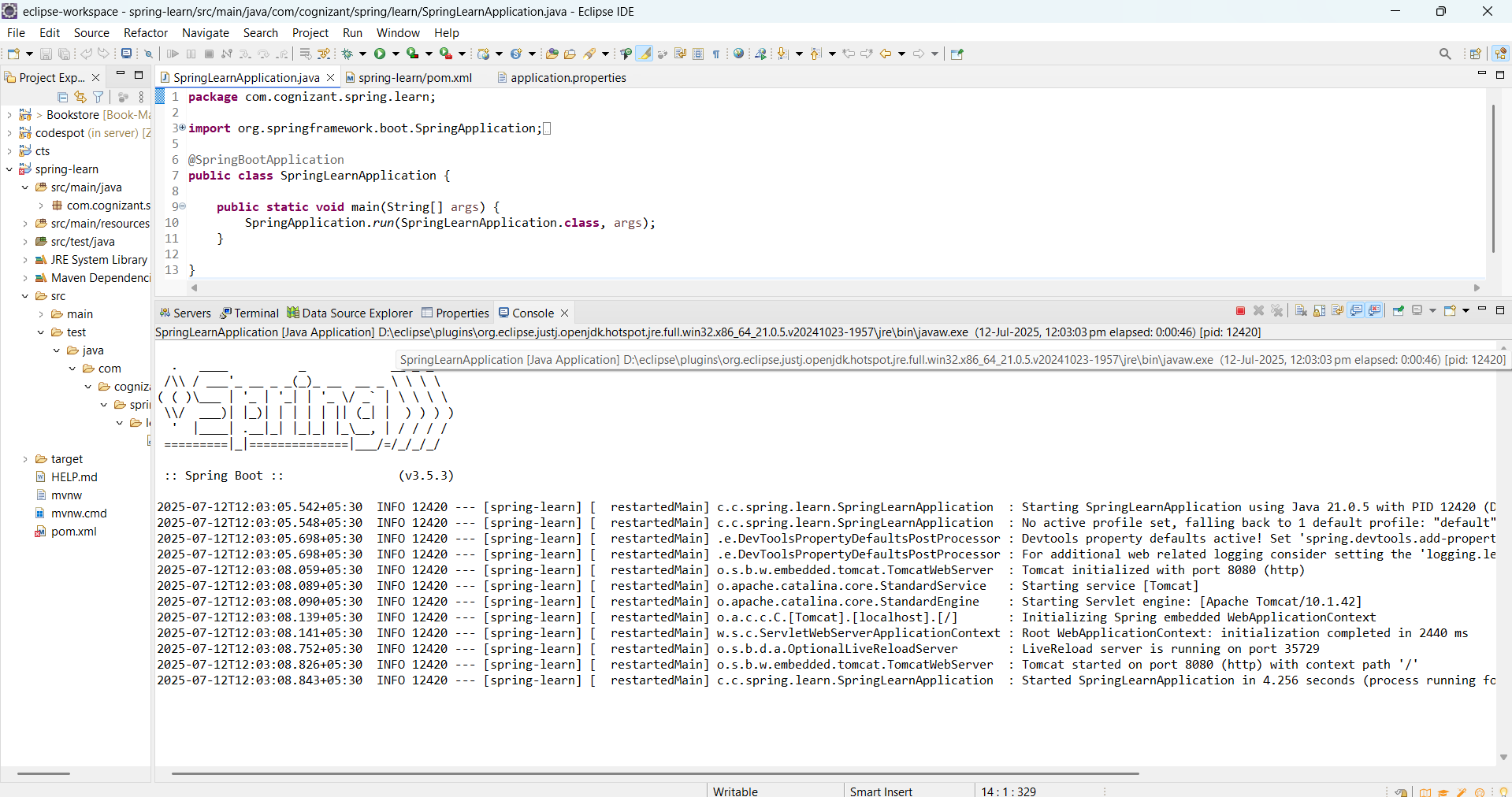
**Application.properties**  
  
spring.application.name=spring-learn

**Folders WalkThrough**

**src/main/java  
**

**src/main/resources  
**

**src/test/java  
**

**Output :  
  
Hands on 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.

<?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

        https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="dateFormat" class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy" />

</bean>

</beans>

* 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.

**Troubleshooting Tips**   
  
If the tomcat port has a conflict and the server is not starting include the below property in application.properties file in src/main/resources folder.

**Code  
SpringLearnApplication.java**  
**package** com.cognizant.spring.learn;

**import** java.util.Date;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

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

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

**import** org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

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

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

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.**class**})

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

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

*displayDate*();

}

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

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

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

**try** {

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

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

} **catch** (ParseException e) {

e.printStackTrace();

}

}

}

**Date-format.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*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

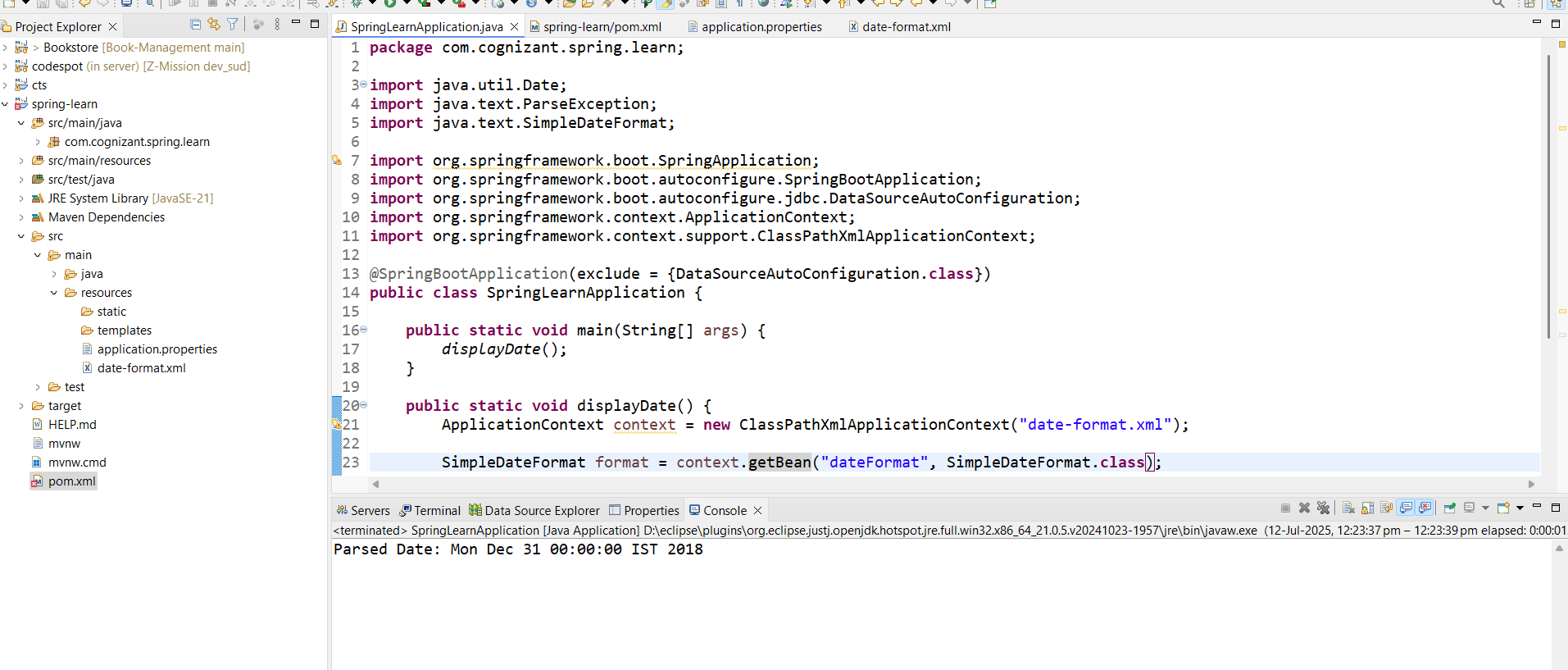
<bean id=*"dateFormat"* class=*"java.text.SimpleDateFormat"*>

<constructor-arg value=*"dd/MM/yyyy"*/>

</bean>

</beans>

**Output**

****

**Hands on 3**

**Spring Core - Incorporate Logging**   
  
Incorporate logging in the Spring Boot project created in previous hands on. Refer steps below:

* Create application.properties if not yet created in src/main/resources folder
* Add below lines in application.properties

logging.level.org.springframework=info

logging.level.com.cognizant.springlearn=debug

logging.pattern.console=%d{yyMMdd}|%d{HH:mm:ss.SSS}|%-20.20thread|%5p|%-25.25logger{25}|%25M|%m%n

* In SpringLearnApplication.java include the following imports:

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

* Include the below static variable in SpringLearnApplication.java:

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

* Include info log on start and end of method. Debug log for displaying the date (refer code below)

public void displayDate() {

    LOGGER.info(“START”);

    //..

    LOGGER.debug(date);

    //..

    LOGGER.info(“END”);

}

**IMPORTANT NOTE:** Going forward all methods should incorporate logging as specified above. **Never** use System.out.println().

**Code:**

**SpringLearnApplication.java**  
**package** com.cognizant.spring.learn;

**import** java.util.Date;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

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

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

**import** org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

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

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.**class**})

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

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

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

*displayDate*();

}

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

***LOGGER***.info("START");

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

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

**try** {

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

***LOGGER***.debug("Parsed Date: {}", date.toString());

} **catch** (ParseException e) {

***LOGGER***.error("ParseException occurred", e);

}

***LOGGER***.info("END");

}

}

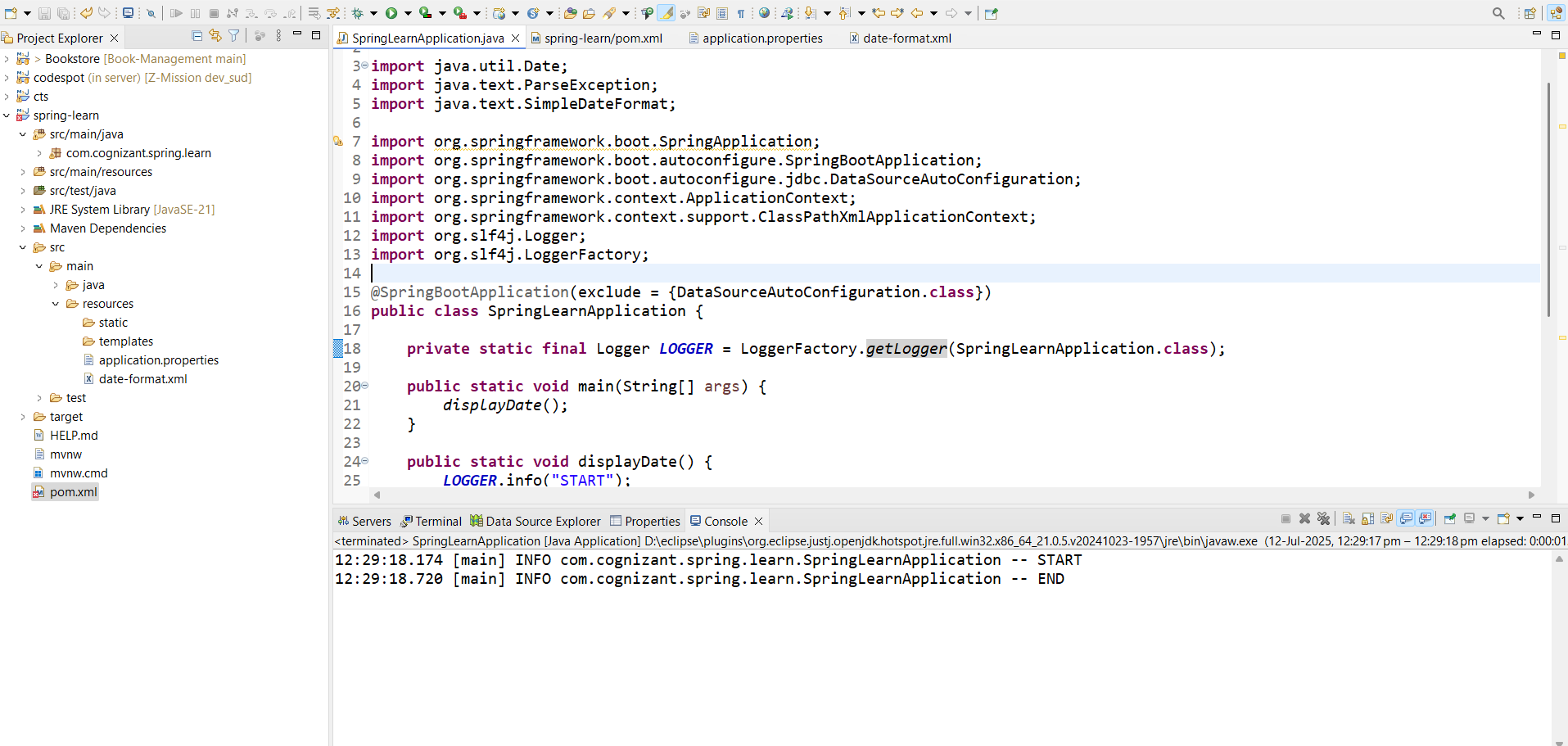
**Pom.xml new dependency**<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

**Output  
**

**Hands on 4**

**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.getBean(“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
* 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

**Code:**

**Country.java**

**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 + "]";

}

}

**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*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"country"* class=*"com.cognizant.spring.learn.Country"*>

<property name=*"code"* value=*"IN"* />

<property name=*"name"* value=*"India"* />

</bean>

</beans>

**SpringLearnApplication.java**

package com.cognizant.spring.learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})

public class SpringLearnApplication {

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

public static void main(String[] args) {

displayCountry();

}

public static void displayCountry() {

System.setProperty("logging.level.com.cognizant.spring.learn", "DEBUG");

LOGGER.info("START");

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

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

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

for (String name : context.getBeanDefinitionNames()) {

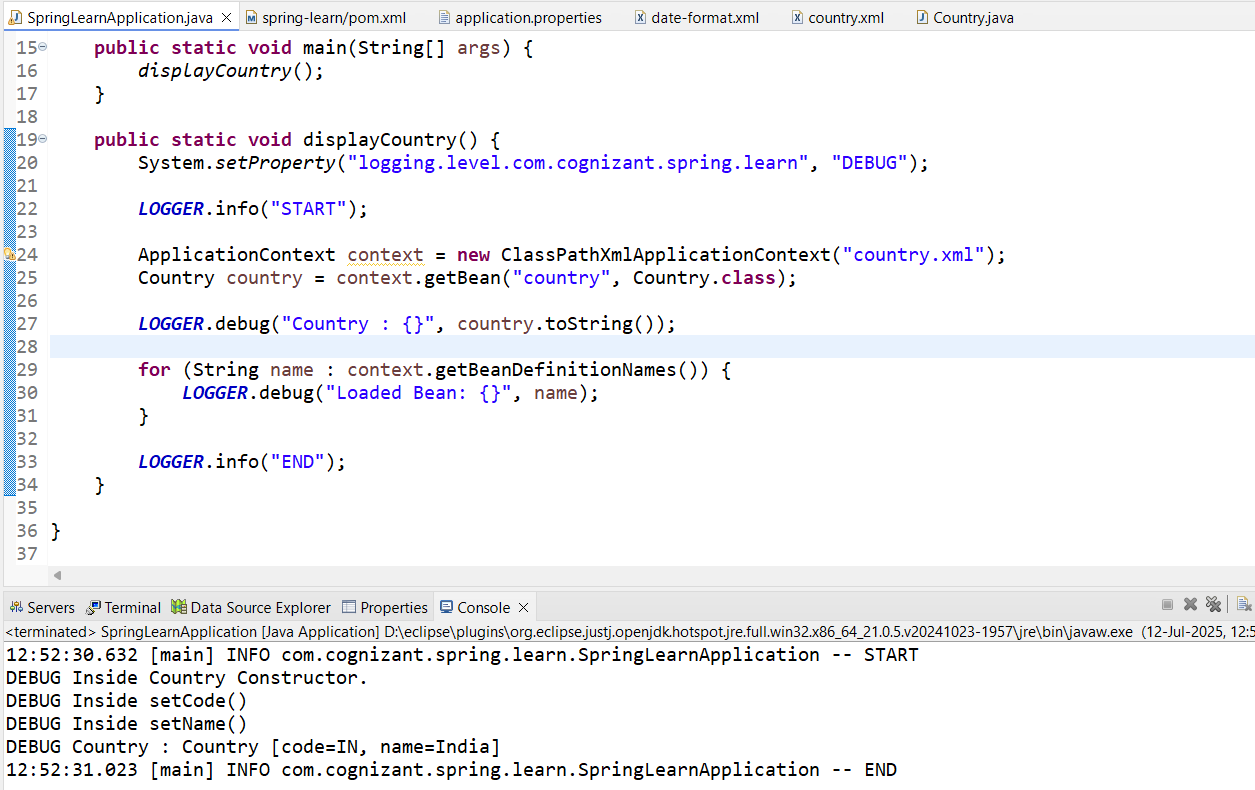
LOGGER.debug("Loaded Bean: {}", name);

}

LOGGER.info("END");

}

}

**Output:  
**

**2-Spring REST Hands On Solutions  
  
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

**Code:  
SpringLearnApplication**

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

**import** org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

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

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

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.**class**})

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

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

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

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

}

}

**Application.properties**

server.port=8083

**HelloCotroller.java**  
**package** com.cognizant.spring.learn.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");

String response = "Hello World!!";

***LOGGER***.debug("Response: {}", response);

***LOGGER***.info("END");

**return** response;

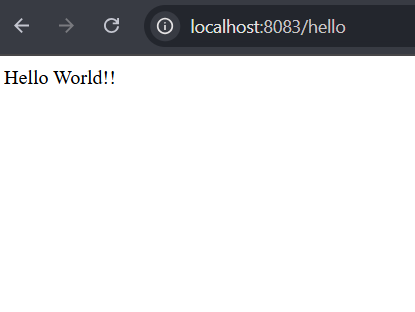
}

}

**Output:**



**Browser**



**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

**Code:**

**CountryController.java**

**import** com.cognizant.spring.learn.Country;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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 {

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

@RequestMapping("/country")

**public** Country getCountryIndia() {

***LOGGER***.info("START");

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

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

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

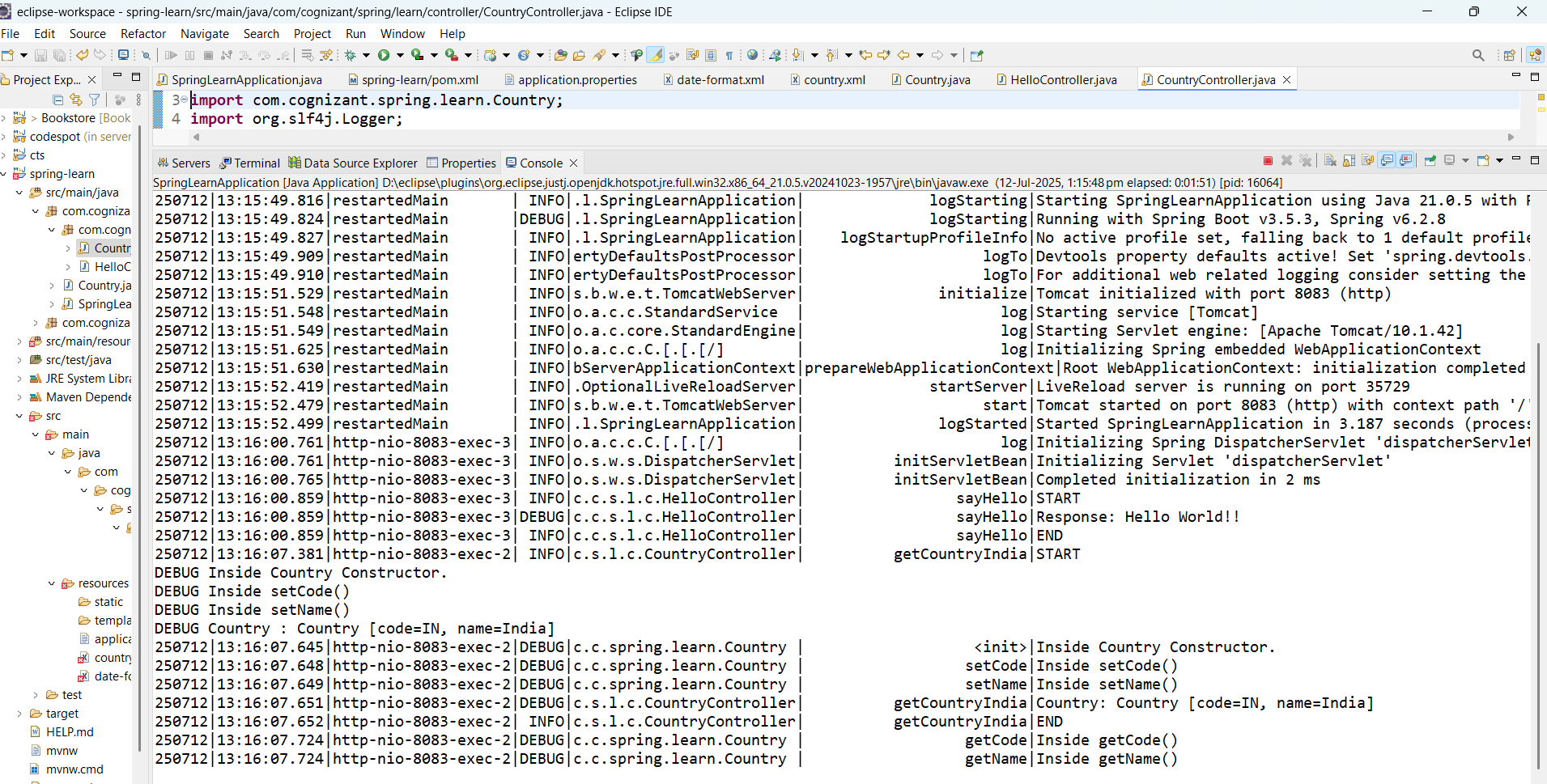
***LOGGER***.info("END");

**return** country;

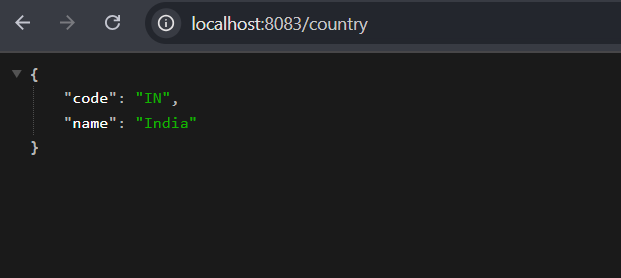
}

}

**Output**



**Browser**



**REST - Get all countries**   
  
Write a REST service that returns all the countries.  
  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation**: @GetMapping("/countries")  
**Method Name**: getAllCountries()  
**Method Implementation**: Load country list from country.xml and return  
  
**Sample Request**: http://localhost:8083/countries  
**Sample Response**:

[

  { "code": "IN", "name": "India"},

  { "code": "US", "name": "United States"},

  { "code": "JP", "name": "Japan"},

  { "code": "DE", "name": "Germany"}

]

**Code:**

**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*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"country1"* class=*"com.cognizant.spring.learn.Country"*>

<property name=*"code"* value=*"IN"*/>

<property name=*"name"* value=*"India"*/>

</bean>

<bean id=*"country2"* class=*"com.cognizant.spring.learn.Country"*>

<property name=*"code"* value=*"US"*/>

<property name=*"name"* value=*"United States"*/>

</bean>

<bean id=*"country3"* class=*"com.cognizant.spring.learn.Country"*>

<property name=*"code"* value=*"JP"*/>

<property name=*"name"* value=*"Japan"*/>

</bean>

<bean id=*"country4"* class=*"com.cognizant.spring.learn.Country"*>

<property name=*"code"* value=*"DE"*/>

<property name=*"name"* value=*"Germany"*/>

</bean>

<bean id=*"countryList"* class=*"java.util.ArrayList"*>

<constructor-arg>

<list>

<ref bean=*"country1"*/>

<ref bean=*"country2"*/>

<ref bean=*"country3"*/>

<ref bean=*"country4"*/>

</list>

</constructor-arg>

</bean>

</beans>

**CountryController.java**

**package** com.cognizant.spring.learn.controller;

**import** java.util.\*;

**import** com.cognizant.spring.learn.Country;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

**import** org.springframework.web.bind.annotation.\*;

@RestController

**public** **class** CountryController {

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

@RequestMapping("/country")

**public** Country getCountryIndia() {

***LOGGER***.info("START");

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

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

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

***LOGGER***.info("END");

**return** country;

}

@GetMapping("/countries")

**public** List<Country> getAllCountries() {

***LOGGER***.info("START - getAllCountries");

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

List<Country> countries = (List<Country>) context.getBean("countryList");

***LOGGER***.debug("Countries: {}", countries);

***LOGGER***.info("END - getAllCountries");

**return** countries;

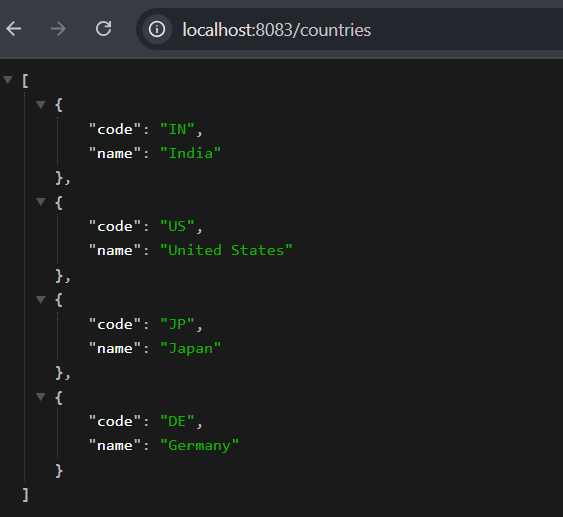
}

}

**Output**

****

**Browser**

****

**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"

}

**Code:**

**SpringLearnApplication.java**

package com.cognizant.spring.learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

public static void displayCountry() {

System.setProperty("logging.level.com.cognizant.spring.learn", "DEBUG");

LOGGER.info("START");

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

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

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

for (String name : context.getBeanDefinitionNames()) {

LOGGER.debug("Loaded Bean: {}", name);

}

LOGGER.info("END");

}

}

**CountryService.java**

package com.cognizant.spring.learn.service;

import com.cognizant.spring.learn.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

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

List<Country> countries = (List<Country>) context.getBean("countryList");

return countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(null); // Or throw an exception if not found

}

}

**CountryController.java**

**package** com.cognizant.spring.learn.controller;

**import** java.util.\*;

**import** com.cognizant.spring.learn.Country;

**import** com.cognizant.spring.learn.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.ClassPathXmlApplicationContext;

**import** org.springframework.web.bind.annotation.\*;

@RestController

**public** **class** CountryController {

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

@RequestMapping("/country")

**public** Country getCountryIndia() {

***LOGGER***.info("START");

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

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

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

***LOGGER***.info("END");

**return** country;

}

@GetMapping("/countries")

**public** List<Country> getAllCountries() {

***LOGGER***.info("START - getAllCountries");

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

List<Country> countries = (List<Country>) context.getBean("countryList");

***LOGGER***.debug("Countries: {}", countries);

***LOGGER***.info("END - getAllCountries");

**return** countries;

}

@Autowired

**private** CountryService countryService;

@GetMapping("/countries/{code}")

**public** Country getCountry(@PathVariable String code) {

***LOGGER***.info("START - getCountry");

Country country = countryService.getCountry(code);

***LOGGER***.debug("Country found: {}", country);

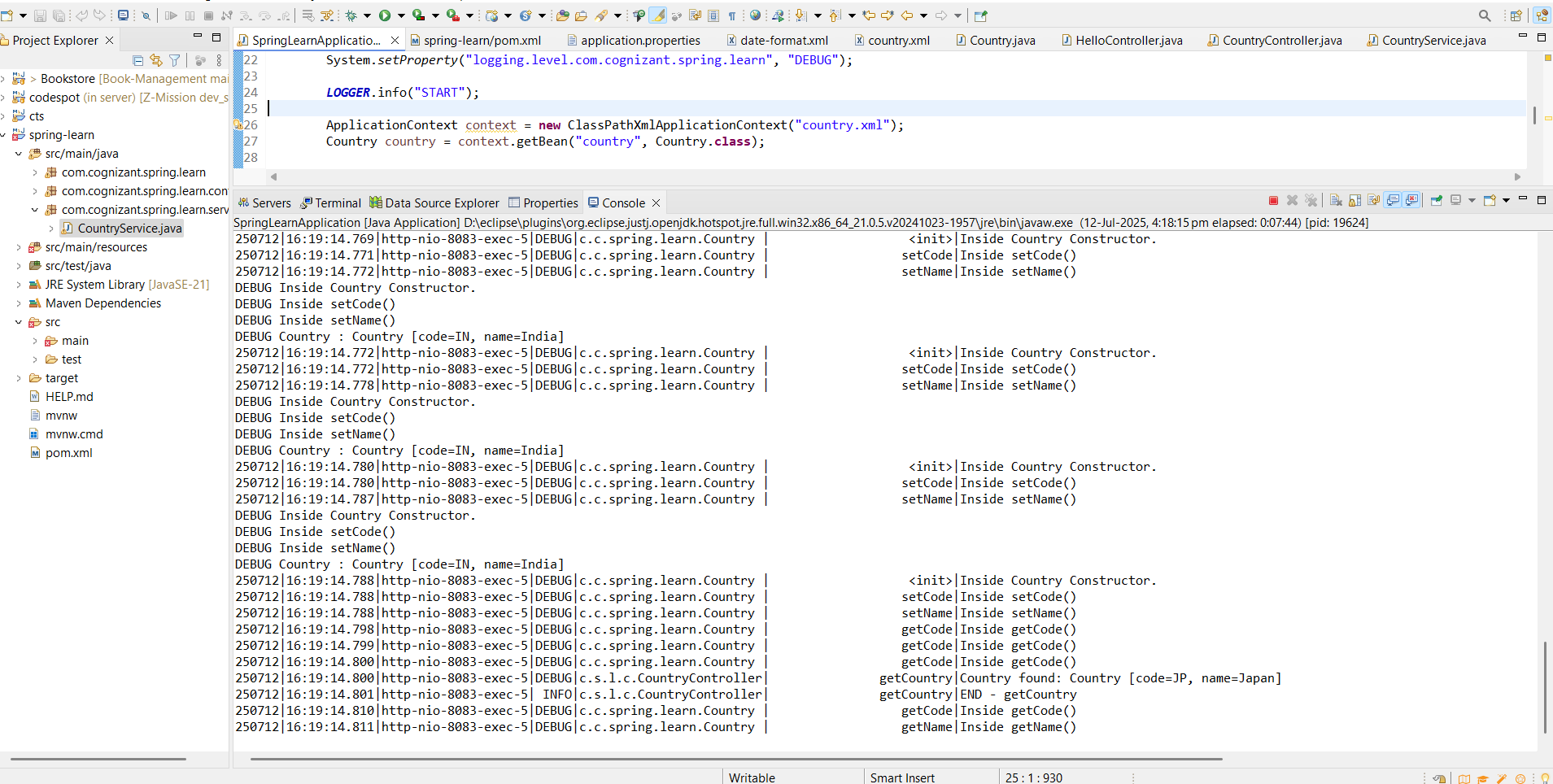
***LOGGER***.info("END - getCountry");

**return** country;

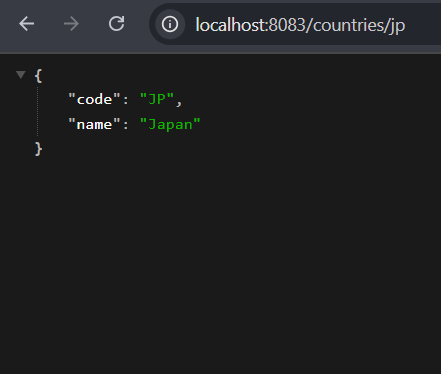
}

}

**Output**

****

**Browser**

****

**3-Spring REST Handson Solutions**

**Problem Statement - Display Employee List and Edit Employee form using RESTful Web Service**   
  
In the previous angular module, we developed a screen that lists employees and it was populated with hard coded values. Now this angular application has be changed to get the data from RESTful Web Service developed in Spring. The following are the high level activities that needs to be done to accomplish this: 

* Create static employee list data using spring xml configuration
* Create a REST Service that reads data from xml configuration and returns it
* Make changes in angular component to consume the created REST Service

Once above activities are completed, clicking on the Edit button against each employee should display Edit Employee form with values retrieved from RESTful Web Service. This will also involve activities similar to the one specified above.  
  
NOTE: There is no specific activity as part of this hands on, refer the next hands ons that covers above three activities in detail.

**Create static employee list data using spring xml configuration**   
  
Follow steps below to accomplish this activity: 

* Incorporate the following in employee.xml:
  + Create one or two more departments
  + Create four more instances of Employee.  (use employee sample data from angular)
  + Reuse existing skills instead of creating new ones
  + Include all four employee instances in an ArrayList.

* In EmployeeDao, incorporate the following:
  + Create static variable with name EMPLOYEE\_LIST of type ArrayList<Employee>
  + Include constructor that reads employee list from xml config and set the EMPLOYEE\_LIST
  + Create method getAllEmployees() that returns the EMPLOYEE\_LIST

**Code:**

**SpringLearnApplication:**

package com.cognizant.spring.learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cognizant.spring.learn.dao.EmployeeDao;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})

public class SpringLearnApplication {

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

public static void main(String[] args) {

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

//displayCountry();

EmployeeDao dao = new EmployeeDao();

List<Employee> employees = dao.getAllEmployees();

employees.forEach(System.out::println);

}

public static void displayCountry() {

System.setProperty("logging.level.com.cognizant.spring.learn", "DEBUG");

LOGGER.info("START");

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

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

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

for (String name : context.getBeanDefinitionNames()) {

LOGGER.debug("Loaded Bean: {}", name);

}

LOGGER.info("END");

}

}

**Employee.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*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- Departments -->

<bean id=*"dept1"* class=*"com.cognizant.spring.learn.Department"*>

<property name=*"id"* value=*"1"*/>

<property name=*"name"* value=*"Human Resources"*/>

</bean>

<bean id=*"dept2"* class=*"com.cognizant.spring.learn.Department"*>

<property name=*"id"* value=*"2"*/>

<property name=*"name"* value=*"Finance"*/>

</bean>

<!-- Skills -->

<bean id=*"skill1"* class=*"com.cognizant.spring.learn.Skill"*>

<property name=*"id"* value=*"1"*/>

<property name=*"name"* value=*"Java"*/>

</bean>

<bean id=*"skill2"* class=*"com.cognizant.spring.learn.Skill"*>

<property name=*"id"* value=*"2"*/>

<property name=*"name"* value=*"Angular"*/>

</bean>

<!-- Date parser -->

<bean id=*"dateFormat"* class=*"java.text.SimpleDateFormat"*>

<constructor-arg value=*"dd/MM/yyyy"*/>

</bean>

<!-- Pre-parsed date beans -->

<bean id=*"date1"* factory-bean=*"dateFormat"* factory-method=*"parse"*>

<constructor-arg value=*"01/01/2000"*/>

</bean>

<bean id=*"date2"* factory-bean=*"dateFormat"* factory-method=*"parse"*>

<constructor-arg value=*"15/02/1995"*/>

</bean>

<!-- Employees -->

<bean id=*"employee1"* class=*"com.cognizant.spring.learn.Employee"*>

<property name=*"id"* value=*"101"*/>

<property name=*"name"* value=*"John"*/>

<property name=*"salary"* value=*"50000"*/>

<property name=*"permanent"* value=*"true"*/>

<property name=*"dateOfBirth"* ref=*"date1"*/>

<property name=*"department"* ref=*"dept1"*/>

<property name=*"skillList"*>

<list>

<ref bean=*"skill1"*/>

<ref bean=*"skill2"*/>

</list>

</property>

</bean>

<bean id=*"employee2"* class=*"com.cognizant.spring.learn.Employee"*>

<property name=*"id"* value=*"102"*/>

<property name=*"name"* value=*"Alice"*/>

<property name=*"salary"* value=*"60000"*/>

<property name=*"permanent"* value=*"false"*/>

<property name=*"dateOfBirth"* ref=*"date2"*/>

<property name=*"department"* ref=*"dept2"*/>

<property name=*"skillList"*>

<list>

<ref bean=*"skill1"*/>

</list>

</property>

</bean>

<!-- Employee List -->

<bean id=*"employeeList"* class=*"java.util.ArrayList"*>

<constructor-arg>

<list>

<ref bean=*"employee1"*/>

<ref bean=*"employee2"*/>

<!-- add employee3 and employee4 here if created -->

</list>

</constructor-arg>

</bean>

</beans>

**Employee.java**

**package** com.cognizant.spring.learn;

**import** java.util.Date;

**import** java.util.List;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** **double** salary;

**private** **boolean** permanent;

**private** Date dateOfBirth;

**private** Department department;

**private** List<Skill> skillList;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

**public** **boolean** isPermanent() {

**return** permanent;

}

**public** **void** setPermanent(**boolean** permanent) {

**this**.permanent = permanent;

}

**public** Date getDateOfBirth() {

**return** dateOfBirth;

}

**public** **void** setDateOfBirth(Date dateOfBirth) {

**this**.dateOfBirth = dateOfBirth;

}

**public** Department getDepartment() {

**return** department;

}

**public** **void** setDepartment(Department department) {

**this**.department = department;

}

**public** List<Skill> getSkillList() {

**return** skillList;

}

**public** **void** setSkillList(List<Skill> skillList) {

**this**.skillList = skillList;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id +

", name=" + name +

", salary=" + salary +

", permanent=" + permanent +

", dateOfBirth=" + dateOfBirth +

", department=" + department +

", skills=" + skillList + "]";

}

}

**Skill.java**

**package** com.cognizant.spring.learn;

**public** **class** Skill {

**private** **int** id;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**Department.java**

**package** com.cognizant.spring.learn;

**public** **class** Department {

**private** **int** id;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**EmployeeDao.java**

**package** com.cognizant.spring.learn.dao;

**import** com.cognizant.spring.learn.Employee;

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

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

**import** java.util.List;

**public** **class** EmployeeDao {

**private** **static** List<Employee> *EMPLOYEE\_LIST*;

**public** EmployeeDao() {

ApplicationContext context = **new** ClassPathXmlApplicationContext("employee.xml");

*EMPLOYEE\_LIST* = (List<Employee>) context.getBean("employeeList");

}

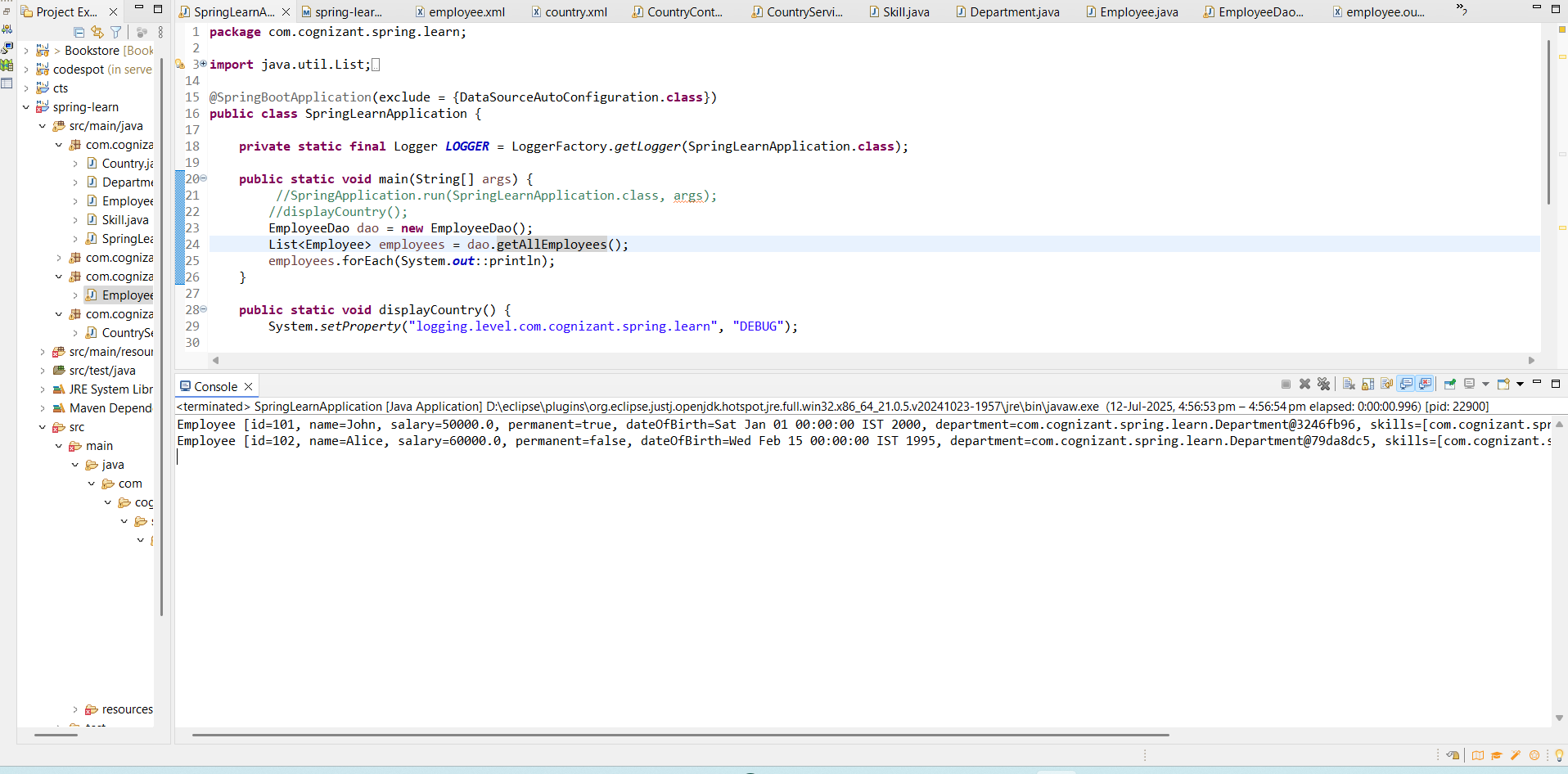
**public** List<Employee> getAllEmployees() {

**return** *EMPLOYEE\_LIST*;

}

}

**Output**

****

**Create REST service to gets all employees**   
  
Follow steps below to accomplish this activity:  

* In EmployeeService, incorporate the following:
  + Change the annotation for this class from @Component to @Service
  + Create method getAllEmployees() that invokes employeeDao.getAllEmployees() and return the employee list
  + Define @Transactional annotation for this method.

* In EmployeeController, incorporate the following:
  + Include a new get method with name getAllEmployees() that returns the employee list
  + Mark this method as GetMapping annotation with the URL as '/employees'
  + Within this method invoke employeeService.getAllEmployees() and return the same.

​​​​​​

* Test ​the service using postman.

**Code:**

**SpringLearnApplication.java**

package com.cognizant.spring.learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cognizant.spring.learn.dao.EmployeeDao;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

//displayCountry();

//EmployeeDao dao = new EmployeeDao();

//List<Employee> employees = dao.getAllEmployees();

// employees.forEach(System.out::println);

}

public static void displayCountry() {

System.setProperty("logging.level.com.cognizant.spring.learn", "DEBUG");

LOGGER.info("START");

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

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

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

for (String name : context.getBeanDefinitionNames()) {

LOGGER.debug("Loaded Bean: {}", name);

}

LOGGER.info("END");

}

}

**EmployeeService.java**

**package** com.cognizant.spring.learn.service;

**import** com.cognizant.spring.learn.dao.EmployeeDao;

**import** com.cognizant.spring.learn.Employee;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** java.util.List;

@Service

**public** **class** EmployeeService {

@Autowired

**private** EmployeeDao employeeDao;

@Transactional

**public** List<Employee> getAllEmployees() {

**return** employeeDao.getAllEmployees();

}

}

**EmployeeController.java**

**package** com.cognizant.spring.learn.controller;

**import** com.cognizant.spring.learn.Employee;

**import** com.cognizant.spring.learn.service.EmployeeService;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** java.util.List;

@RestController

**public** **class** EmployeeController {

@Autowired

**private** EmployeeService employeeService;

@GetMapping("/employees")

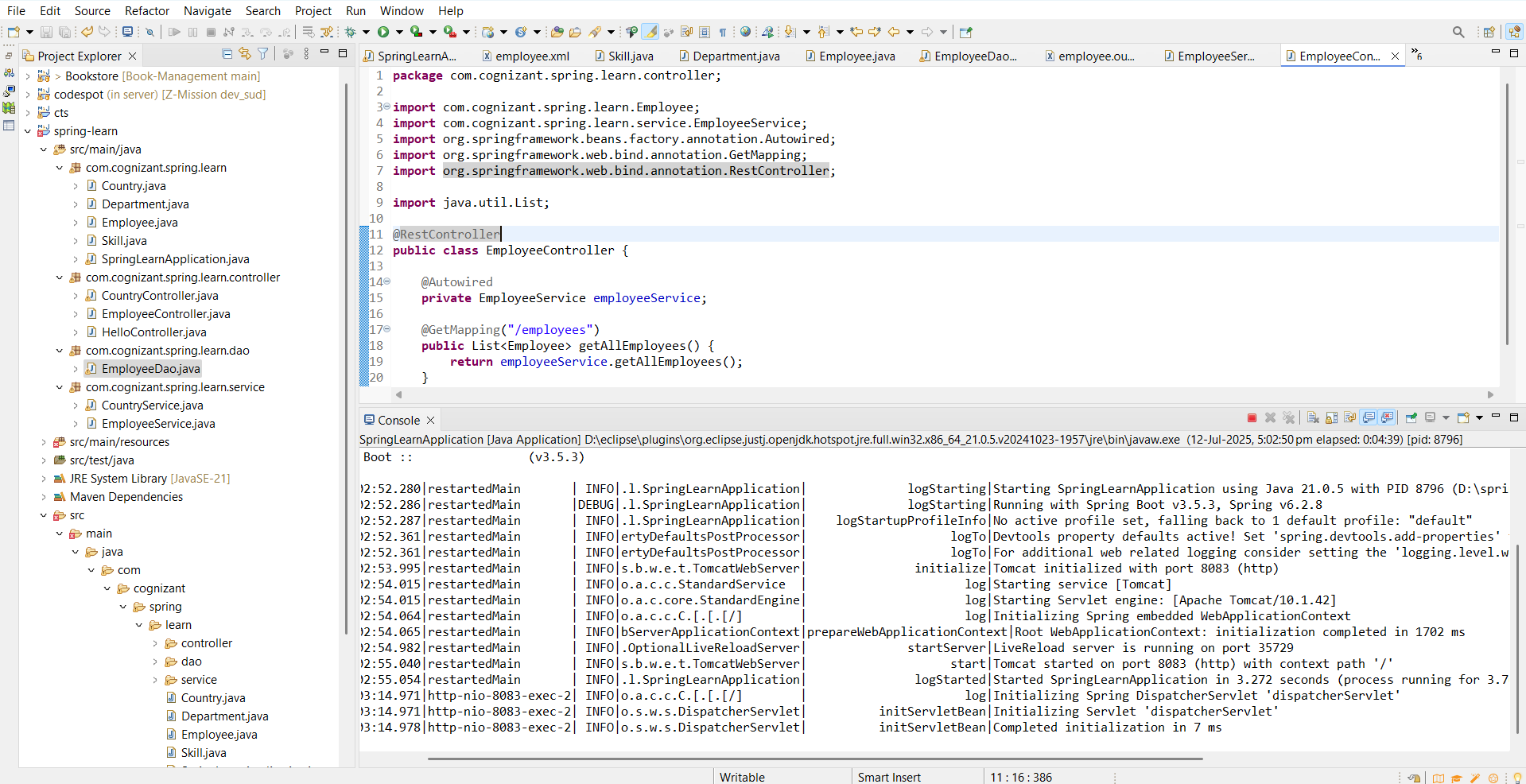
**public** List<Employee> getAllEmployees() {

**return** employeeService.getAllEmployees();

}

}

**Output**

****

**Browser**

****

**Create REST service for department**

Create a new service to get all the departments.

Follow steps below to achieve this:

* Create a new REST Service, define below list of classes and respective methods:
  + DepartmentController
    - getAllDepartments() with URL "/departments", this method will return array of departments
  + DepartmentService
    - getAllDepartments()
  + DepartmentDao
    - getAllDepartments() - Create a static variable DEPARTMENT\_LIST, this should be populated from spring xml configuration
* Test ​the service using postman.
* Also verify if department REST service is called by looking into the logs.

**Code:  
SpringLearnApplication.java**

package com.cognizant.spring.learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cognizant.spring.learn.dao.EmployeeDao;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

//displayCountry();

//EmployeeDao dao = new EmployeeDao();

// List<Employee> employees = dao.getAllEmployees();

// employees.forEach(System.out::println);

}

public static void displayCountry() {

System.setProperty("logging.level.com.cognizant.spring.learn", "DEBUG");

LOGGER.info("START");

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

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

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

for (String name : context.getBeanDefinitionNames()) {

LOGGER.debug("Loaded Bean: {}", name);

}

LOGGER.info("END");

}

}  
**Department.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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Departments -->

<bean id="dept1" class="com.cognizant.spring.learn.Department">

<property name="id" value="1"/>

<property name="name" value="Human Resources"/>

</bean>

<bean id="dept2" class="com.cognizant.spring.learn.Department">

<property name="id" value="2"/>

<property name="name" value="Finance"/>

</bean>

<bean id="departmentList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="dept1"/>

<ref bean="dept2"/>

</list>

</constructor-arg>

</bean>

</beans>  
**Department.java**package com.cognizant.spring.learn;

public class Department {

private int id;

private String name;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**DepartmentDao.java**

package com.cognizant.spring.learn.dao;

import com.cognizant.spring.learn.Department;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public class DepartmentDao {

private static List<Department> DEPARTMENT\_LIST;

public DepartmentDao() {

ApplicationContext context = new ClassPathXmlApplicationContext("department.xml");

DEPARTMENT\_LIST = (List<Department>) context.getBean("departmentList");

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

**DepartmentService.java**

package com.cognizant.spring.learn.service;

import com.cognizant.spring.learn.Department;

import com.cognizant.spring.learn.dao.DepartmentDao;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class DepartmentService {

@Autowired

private DepartmentDao departmentDao;

@Transactional

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

**DepartmentController.java**

package com.cognizant.spring.learn.controller;

import com.cognizant.spring.learn.Department;

import com.cognizant.spring.learn.service.DepartmentService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

public class DepartmentController {

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

@Autowired

private DepartmentService departmentService;

@GetMapping("/departments")

public List<Department> getAllDepartments() {

LOGGER.info("START: getAllDepartments()");

List<Department> departments = departmentService.getAllDepartments();

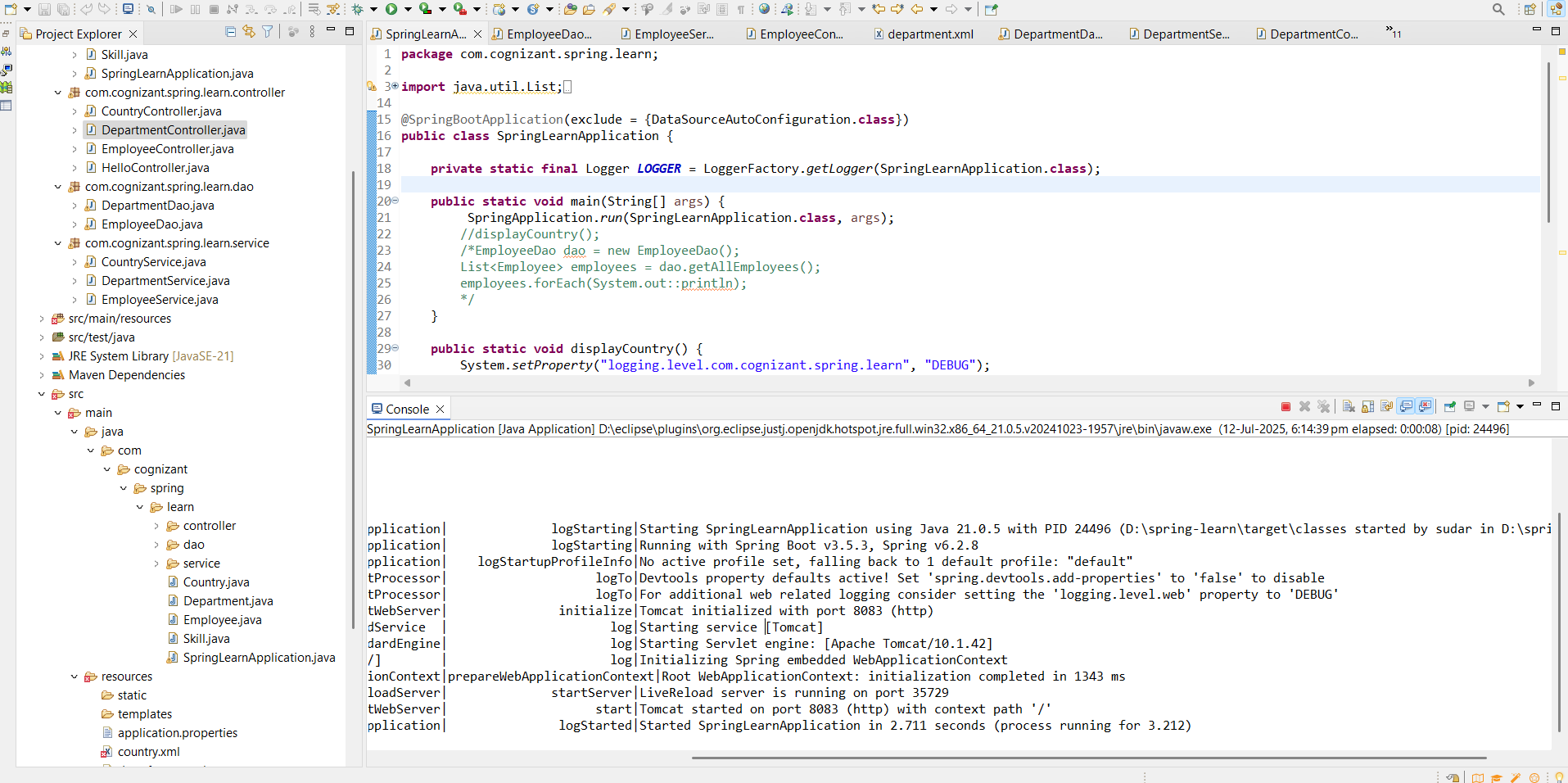
LOGGER.info("END: getAllDepartments()");

return departments;

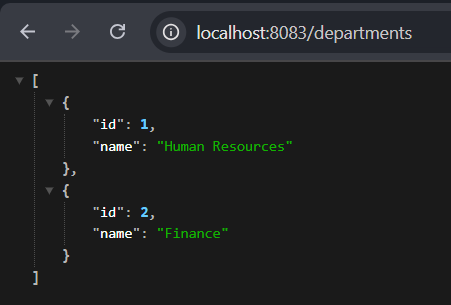
}

}

**Output**

****

**Browser**

****

**5-Spring REST Handson Solutions   
  
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.

**src/main/java/com/cognizant/springlearn/controller/AuthenticationController.java**

package com.cognizant.springlearn.controller;

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 {

    @GetMapping("/authenticate")

    public Map<String, String> authenticate() {

        Map<String, String> response = new HashMap<>();

        response.put("token", "jwt-token-will-go-here");

        return response;

    }

}

**src/main/java/com/cognizant/springlearn/config/SecurityConfig.java**

package com.cognizant.springlearn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import static org.springframework.security.config.Customizer.withDefaults;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

    @Bean

    public PasswordEncoder passwordEncoder() {

        return new BCryptPasswordEncoder(); // Hashing enabled

    }

    @Bean

    public InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

        UserDetails user = User.withUsername("user")

                .password(passwordEncoder.encode("pwd")) // Encrypt password

                .roles("USER")

                .build();

        return new InMemoryUserDetailsManager(user);

    }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http

                .authorizeHttpRequests(auth -> auth

                        .requestMatchers("/authenticate").authenticated()

                        .anyRequest().permitAll())

                .httpBasic(withDefaults())

                .csrf(csrf -> csrf.disable()); // Disable CSRF for testing

        return http.build();

    }

}

**src/main/java/com/cognizant/springlearn/util/JwtUtil.java**

package com.cognizant.springlearn.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\_KEY = "secret";

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

                .setExpiration(new Date(System.currentTimeMillis() + 10 \* 60 \* 1000)) // 10 min

                .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

                .compact();

    }

}

**src/main/resources/application.properties**

server.port=8090

logging.level.com.cognizant=DEBUG

logging.level.org.springframework.security=DEBUG

**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>springlearn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>springlearn</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>io.jsonwebtoken</groupId>

            <artifactId>jjwt</artifactId>

            <version>0.9.1</version>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-security</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>org.junit.jupiter</groupId>

            <artifactId>junit-jupiter</artifactId>

            <version>5.10.2</version> <!-- Use latest compatible version -->

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

    <plugins>

        <!-- Plugin to allow running Java class with main method -->

        <plugin>

            <groupId>org.codehaus.mojo</groupId>

            <artifactId>exec-maven-plugin</artifactId>

            <version>3.1.0</version>

        </plugin>

        <!-- Spring Boot plugin for packaging -->

        <plugin>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-maven-plugin</artifactId>

        </plugin>

    </plugins>

    </build>

</project>

**src/main/java/com/cognizant/springlearn/SpringLearnApplication5.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication5 {

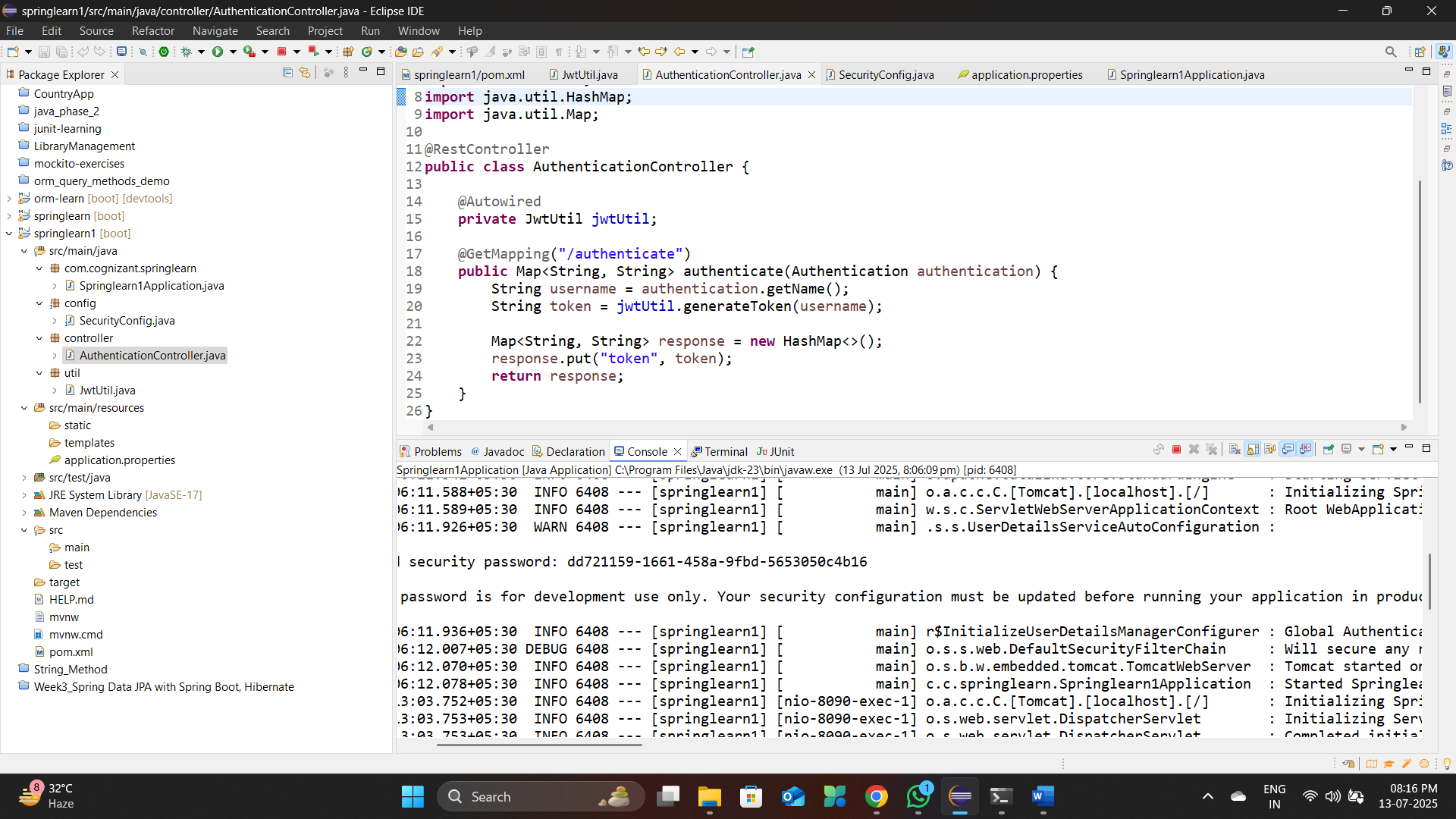
    public static void main(String[] args) {

        SpringApplication.run(SpringLearnApplication5.class, args);

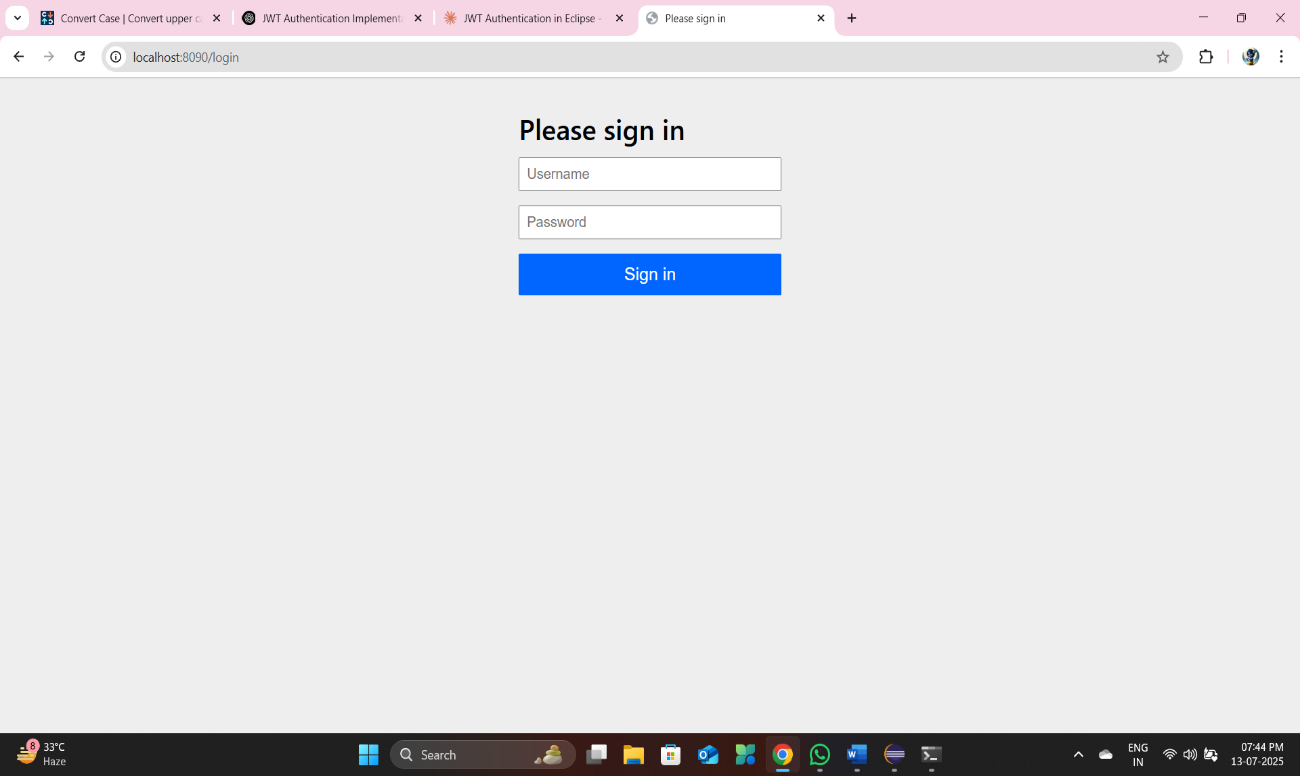
    }

}

**Output:**



**BROWSER**

****