WEEK 4 – ASSIGNMENT

Superset ID: 6390124

Spring REST using Spring Boot 3 Exercises:-

Exercise 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.

pom.xml

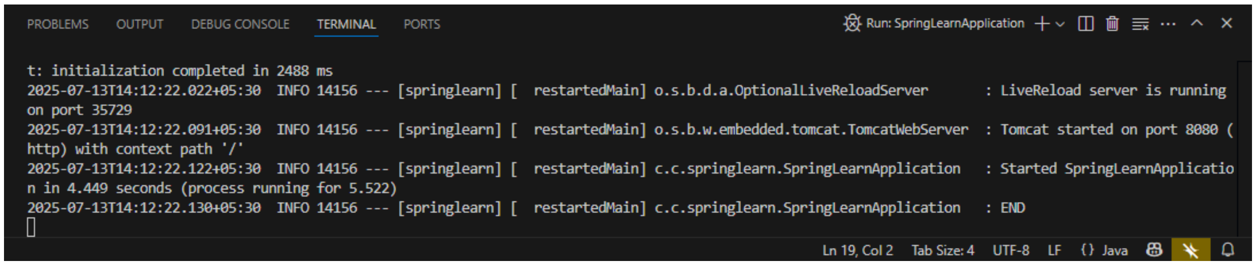
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <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>
```

```xml
<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>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 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>
```

SpringLearnApplication.java

```
package com.cognizant.springlearn;

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

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@SpringBootApplication
public class SpringLearnApplication {

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

    public static void main(String[] args) {
        LOGGER.info("START");
        SpringApplication.run(SpringLearnApplication.class, args);
        LOGGER.info("END");
    }
}
```

Output:



```
t: initialization completed in 2488 ms
2025-07-13T14:12:22.022+05:30  INFO 14156 --- [springlearn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running
on port 35729
2025-07-13T14:12:22.091+05:30  INFO 14156 --- [springlearn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (
http) with context path '/'
2025-07-13T14:12:22.122+05:30  INFO 14156 --- [springlearn] [  restartedMain] c.c.springlearn.SpringLearnApplication   : Started SpringLearnApplicatio
n in 4.449 seconds (process running for 5.522)
2025-07-13T14:12:22.130+05:30  INFO 14156 --- [springlearn] [  restartedMain] c.c.springlearn.SpringLearnApplication   : END
```

Exercise 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.

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 of SpringLearnApplication.java.
- Execute main() method and check the logs to find out which constructors and methods were invoked.

SME to provide more detailing about the following aspects:

- bean tag, id attribute, class attribute, property tag, name attribute, value attribute
- ApplicationContext, ClassPathXmlApplicationContext
- What exactly happens when context.getBean() is invoked

src/main/java/com/cognizant/springlearn/Country.java

```
package com.cognizant.springlearn;

public class Country {
```

```java
    private String code;
    private String name;

    public Country() {
        System.out.println("Inside Country Constructor.");
    }

    public String getCode() {
        System.out.println("Getting Country Code: " + code);
        return code;
    }

    public void setCode(String code) {
        System.out.println("Setting Country Code: " + code);
        this.code = code;
    }

    public String getName() {
        System.out.println("Getting Country Name: " + name);
        return name;
    }

    public void setName(String name) {
        System.out.println("Setting Country Name: " + name);
        this.name = name;
    }

    @Override
    public String toString() {
        return "Country [code=" + getCode() + ", name=" + getName() + "]";
    }
}
```

src/main/resources/application.properties

```
spring.application.name=springlearn
logging.level.com.cognizant=DEBUG
```

src/main/resources/country.xml

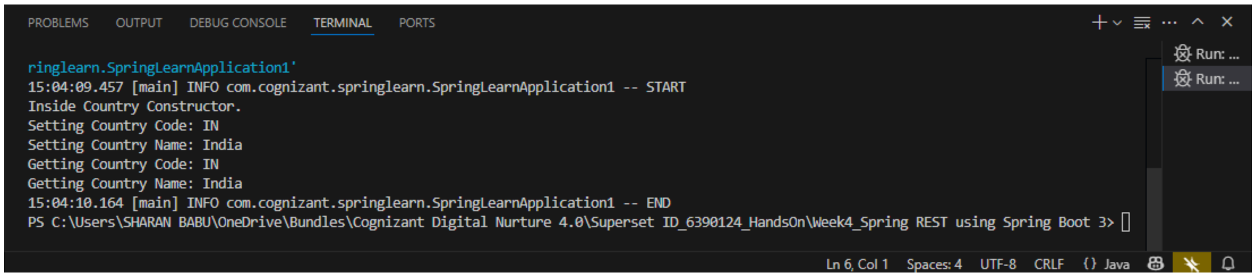
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="country" class="com.cognizant.springlearn.Country">
        <property name="code" value="IN" />
        <property name="name" value="India" />
    </bean>
```

```
</beans>
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication1.java

```java
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication1 {
    private static final Logger LOGGER =
LoggerFactory.getLogger(SpringLearnApplication1.class);

    public static void main(String[] args) {
        LOGGER.info("START");
        displayCountry();
        LOGGER.info("END");
    }

    public static void displayCountry() {
        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("country.xml")) {
            Country country = context.getBean("country", Country.class);
            LOGGER.debug("Country : {}", country.toString());
        }
    }
}
```

Output:



```
ringlearn.SpringLearnApplication1'
15:04:09.457 [main] INFO com.cognizant.springlearn.SpringLearnApplication1 -- START
Inside Country Constructor.
Setting Country Code: IN
Setting Country Name: India
Getting Country Code: IN
Getting Country Name: India
15:04:10.164 [main] INFO com.cognizant.springlearn.SpringLearnApplication1 -- END
PS C:\Users\SHARAN BABU\OneDrive\Bundles\Cognizant Digital Nurture 4.0\Superset ID_6390124_HandsOn\Week4_Spring REST using Spring Boot 3>
```

Exercise 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

src/main/java/com/cognizant/springlearn/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()");
        String message = "Hello World!!";
        LOGGER.info("END - sayHello()");
        return message;
    }
}
```

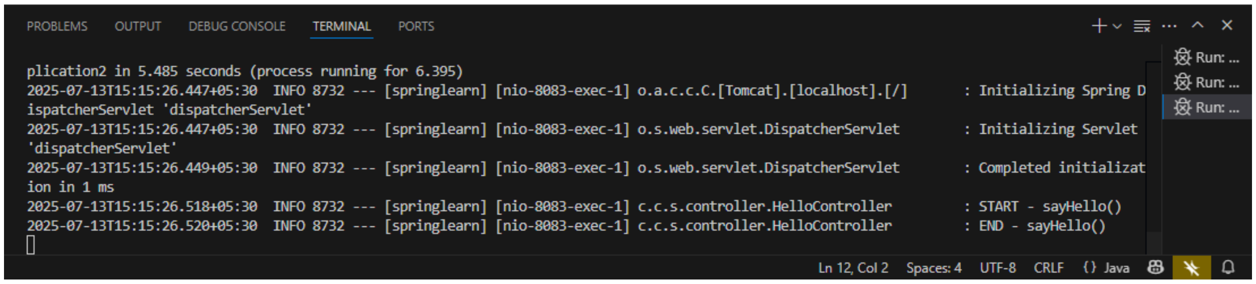
src/main/java/com/cognizant/springlearn/SpringLearnApplication2.java

```
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication2 {

    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication2.class, args);
    }
}
```

src/main/resources/application.properties

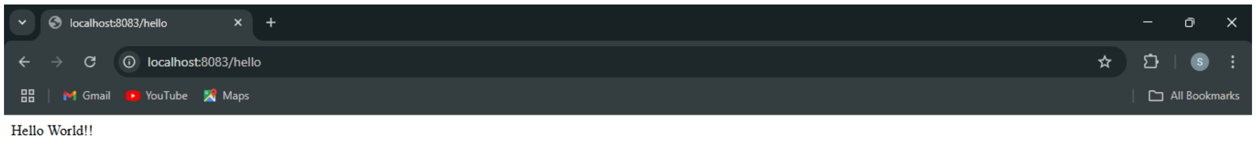
```
spring.application.name=springlearn
logging.level.com.cognizant=DEBUG
server.port=8083
```

Output:



```
plication2 in 5.485 seconds (process running for 6.395)
2025-07-13T15:15:26.447+05:30  INFO 8732 --- [springlearn] [nio-8083-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring D
ispatcherServlet 'dispatcherServlet'
2025-07-13T15:15:26.447+05:30  INFO 8732 --- [springlearn] [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet
'dispatcherServlet'
2025-07-13T15:15:26.449+05:30  INFO 8732 --- [springlearn] [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initializat
ion in 1 ms
2025-07-13T15:15:26.518+05:30  INFO 8732 --- [springlearn] [nio-8083-exec-1] c.c.s.controller.HelloController         : START - sayHello()
2025-07-13T15:15:26.520+05:30  INFO 8732 --- [springlearn] [nio-8083-exec-1] c.c.s.controller.HelloController         : END - sayHello()
```

http://localhost:8083/hello



Hello World!!

Exercise 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

src/main/java/com/cognizant/springlearn/Country1.java
```java
package com.cognizant.springlearn;

public class Country1 {
    private String code;
    private String name;

    public Country1() {
    }

    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;
    }
}
```

src/main/resources/country1.xml
```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.springlearn.Country1">
        <property name="code" value="IN" />
        <property name="name" value="India" />
    </bean>

</beans>
```

src/main/java/com/cognizant/springlearn/controller/CountryController.java
```java
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country1;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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 Country1 getCountryIndia() {
        LOGGER.info("START - getCountryIndia()");

        Country1 country;
        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("country1.xml")) {
            country = context.getBean("country1", Country1.class);
        }

        LOGGER.info("END - getCountryIndia()");
        return country;
    }
}
```

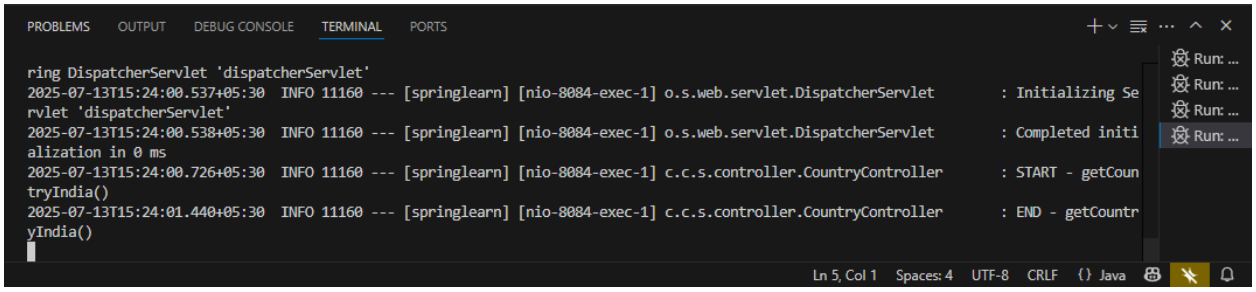
src/main/resources/application.properties

```
spring.application.name=springlearn
logging.level.com.cognizant=DEBUG
server.port=8084
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication3.java

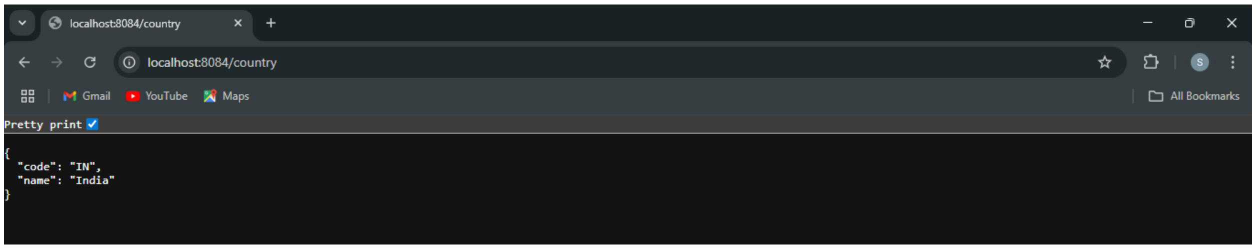
```java
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication3 {

    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication3.class, args);
    }
}
```

Output:



```
ring DispatcherServlet 'dispatcherServlet'
2025-07-13T15:24:00.537+05:30  INFO 11160 --- [springlearn] [nio-8084-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Se
rvlet 'dispatcherServlet'
2025-07-13T15:24:00.538+05:30  INFO 11160 --- [springlearn] [nio-8084-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initi
alization in 0 ms
2025-07-13T15:24:00.726+05:30  INFO 11160 --- [springlearn] [nio-8084-exec-1] c.c.s.controller.CountryController       : START - getCoun
tryIndia()
2025-07-13T15:24:01.440+05:30  INFO 11160 --- [springlearn] [nio-8084-exec-1] c.c.s.controller.CountryController       : END - getCountr
yIndia()
```

http://localhost:8084/country



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

Exercise 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"
}
```

src/main/java/com/cognizant/springlearn/Country2.java

```
package com.cognizant.springlearn;

public class Country2 {
    private String code;
    private String name;
    public Country2() {
    }

    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;
    }
}
```

src/main/resources/country2.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="countryList2" class="java.util.ArrayList">
        <constructor-arg>
            <list>
                <bean class="com.cognizant.springlearn.Country2">
                    <property name="code" value="IN"/>
                    <property name="name" value="India"/>
                </bean>
                <bean class="com.cognizant.springlearn.Country2">
                    <property name="code" value="US"/>
                    <property name="name" value="United States"/>
                </bean>
                <bean class="com.cognizant.springlearn.Country2">
                    <property name="code" value="DE"/>
                    <property name="name" value="Germany"/>
                </bean>
                <bean class="com.cognizant.springlearn.Country2">
                    <property name="code" value="JP"/>
                    <property name="name" value="Japan"/>
                </bean>
            </list>
        </constructor-arg>
    </bean>

</beans>
```

src/main/java/com/cognizant/springlearn/service/CountryService.java

```
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication4 {
    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication4.class, args);
    }
}
```

src/main/java/com/cognizant/springlearn/controller/CountryController1.java

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country2;
import com.cognizant.springlearn.service.CountryService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
public class CountryController1 {

    private static final Logger LOGGER =
LoggerFactory.getLogger(CountryController1.class);

    @Autowired
    private CountryService countryService;

    @GetMapping("/countries/{code}")
    public Country2 getCountry(@PathVariable String code) {
        LOGGER.info("START - getCountry()");
        Country2 country = countryService.getCountry(code);
        LOGGER.info("END - getCountry()");
        return country;
    }
}
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication4.java

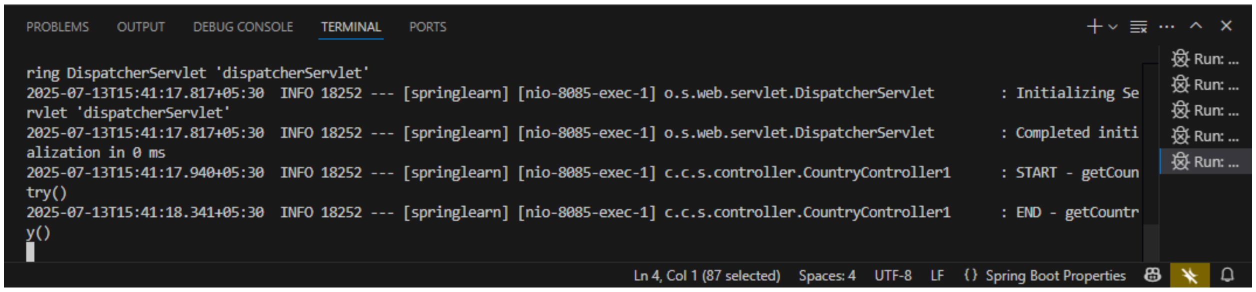
```
package com.cognizant.springlearn;
```

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

@SpringBootApplication
public class SpringLearnApplication4 {
    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication4.class, args);
    }
}
```

src/main/resources/application.properties

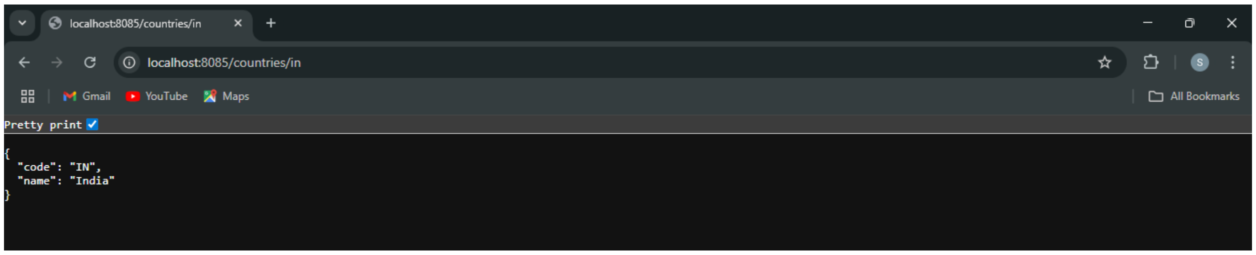
```
spring.application.name=springlearn
logging.level.com.cognizant=DEBUG
server.port=8085
```

Output:



```
ring DispatcherServlet 'dispatcherServlet'
2025-07-13T15:41:17.817+05:30  INFO 18252 --- [springlearn] [nio-8085-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Se
rvlet 'dispatcherServlet'
2025-07-13T15:41:17.817+05:30  INFO 18252 --- [springlearn] [nio-8085-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initi
alization in 0 ms
2025-07-13T15:41:17.940+05:30  INFO 18252 --- [springlearn] [nio-8085-exec-1] c.c.s.controller.CountryController1      : START - getCoun
try()
2025-07-13T15:41:18.341+05:30  INFO 18252 --- [springlearn] [nio-8085-exec-1] c.c.s.controller.CountryController1      : END - getCountr
y()
```

http://localhost:8085/countries/in



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

Exercise 6: 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.EnableWebSec
urity;
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
```

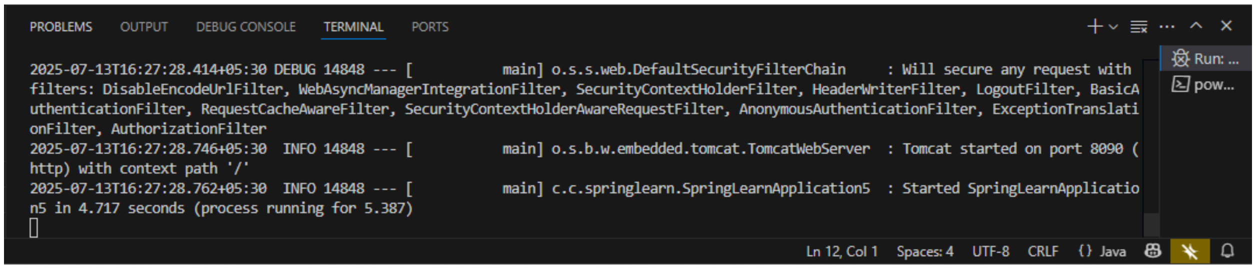
pom.xml

```
<?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

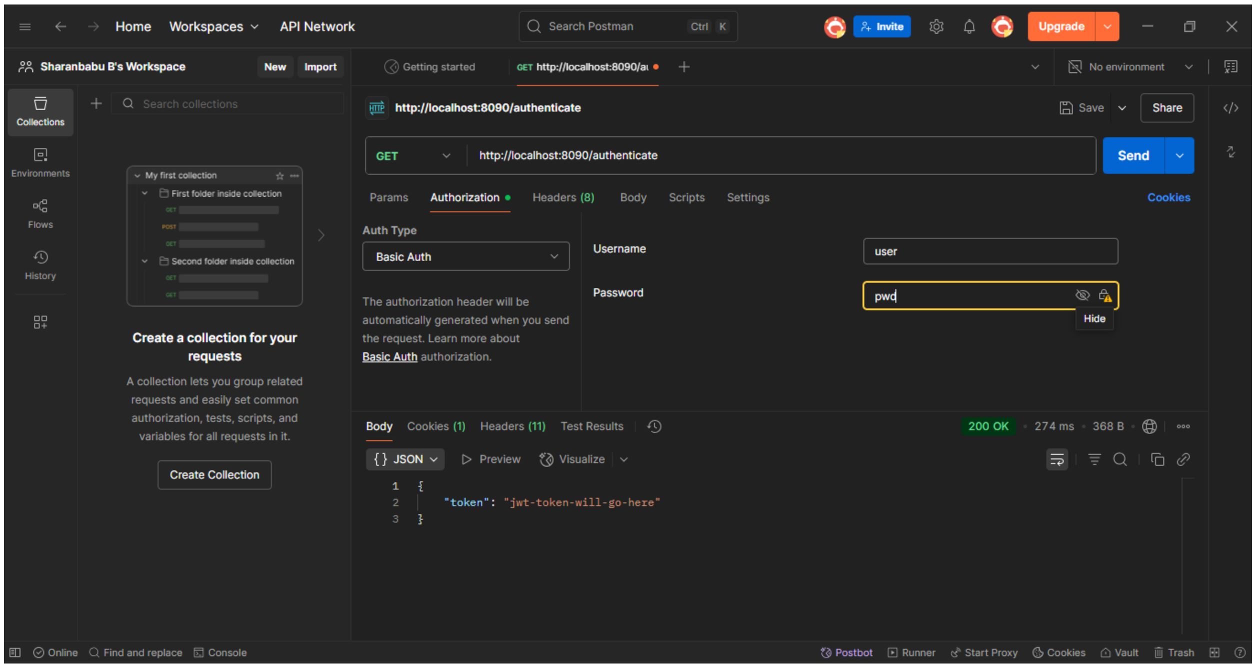
```
package com.cognizant.springlearn;

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:



```
2025-07-13T16:27:28.414+05:30 DEBUG 14848 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with
filters: DisableEncodeUrlFilter, WebAsyncManagerIntegrationFilter, SecurityContextHolderFilter, HeaderWriterFilter, LogoutFilter, BasicA
uthenticationFilter, RequestCacheAwareFilter, SecurityContextHolderAwareRequestFilter, AnonymousAuthenticationFilter, ExceptionTranslati
onFilter, AuthorizationFilter
2025-07-13T16:27:28.746+05:30  INFO 14848 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8090 (
http) with context path '/'
2025-07-13T16:27:28.762+05:30  INFO 14848 --- [           main] c.c.springlearn.SpringLearnApplication5  : Started SpringLearnApplicatio
n5 in 4.717 seconds (process running for 5.387)
```

Postman Terminal:



http://localhost:8090/authenticate

GET http://localhost:8090/authenticate

Auth Type: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.

Username: user

Password: pwd

200 OK · 274 ms · 368 B

```
{
    "token": "jwt-token-will-go-here"
}
```

Exercise 7: Display Employee List and Edit Employee form using RESTful Web Service

Problem Statement:

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.

src/main/java/com/cognizant/springlearn/controller/EmployeeController.java
```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Employee;
import com.cognizant.springlearn.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class EmployeeController {

    @Autowired
    private EmployeeService service;

    @GetMapping("/employees")
    public List<Employee> getEmployees() {
        return service.getAllEmployees();
    }

    @GetMapping("/employees/{id}")
    public Employee getEmployee(@PathVariable int id) {
        return service.getEmployeeById(id);
    }
}
```

src/main/java/com/cognizant/springlearn/service/EmployeeService.java
```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Employee;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService {

    @SuppressWarnings("unchecked")
    public List<Employee> getAllEmployees() {
        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("employee.xml")) {
            return (List<Employee>) context.getBean("employeeList");
        }
    }

    public Employee getEmployeeById(int id) {
        return getAllEmployees().stream()
                .filter(e -> e.getId() == id)
                .findFirst()
                .orElse(null);
    }
}
```

src/main/java/com/cognizant/springlearn/model/Employee.java

```
package com.cognizant.springlearn.model;

public class Employee {
    private int id;
    private String name;
    private double salary;
    private String permanent;
    private String department;

    public Employee() {
        System.out.println("Inside Employee Constructor.");
    }

    // Getters & Setters with logs
    public int getId() {
        return id;
    }

    public void setId(int id) {
        System.out.println("Setting ID: " + id);
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        System.out.println("Setting Name: " + name);
        this.name = name;
    }

    public double getSalary() {
        return salary;
```

```java
    }

    public void setSalary(double salary) {
        System.out.println("Setting Salary: " + salary);
        this.salary = salary;
    }

    public String getPermanent() {
        return permanent;
    }

    public void setPermanent(String permanent) {
        System.out.println("Setting Permanent: " + permanent);
        this.permanent = permanent;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        System.out.println("Setting Department: " + department);
        this.department = department;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ", salary=" + salary
+ ", permanent=" + permanent
                + ", department=" + department + "]";
    }
}
```

src/main/resources/employee.xml

```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="employeeList" class="java.util.ArrayList">
        <constructor-arg>
            <list>
                <bean class="com.cognizant.springlearn.model.Employee">
                    <property name="id" value="1" />
                    <property name="name" value="John" />
                    <property name="salary" value="50000" />
                    <property name="permanent" value="Yes" />
                    <property name="department" value="HR" />
                </bean>
                <bean class="com.cognizant.springlearn.model.Employee">
                    <property name="id" value="2" />
                    <property name="name" value="Alice" />
                    <property name="salary" value="60000" />
                    <property name="permanent" value="No" />
                    <property name="department" value="IT" />
                </bean>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

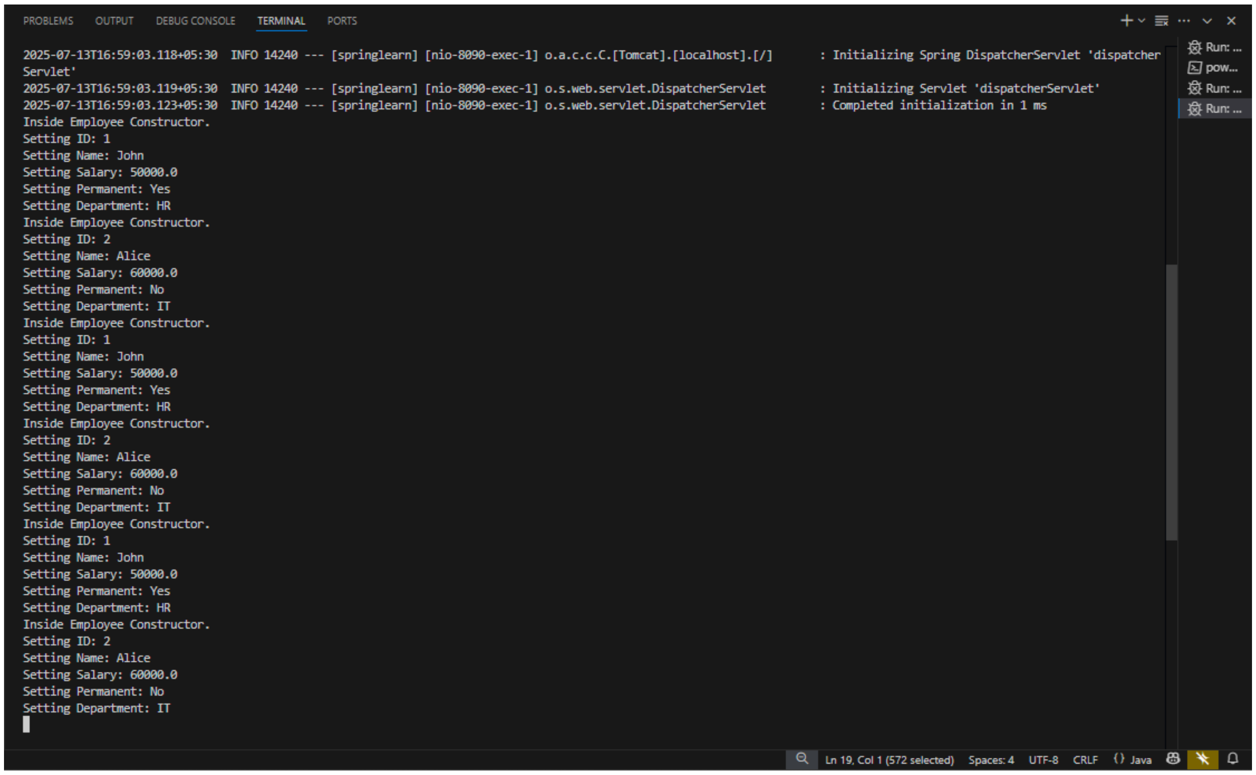
src/main/resources/application.properties

```
server.port=8090
spring.application.name=springlearn
logging.level.com.cognizant=DEBUG
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication6.java

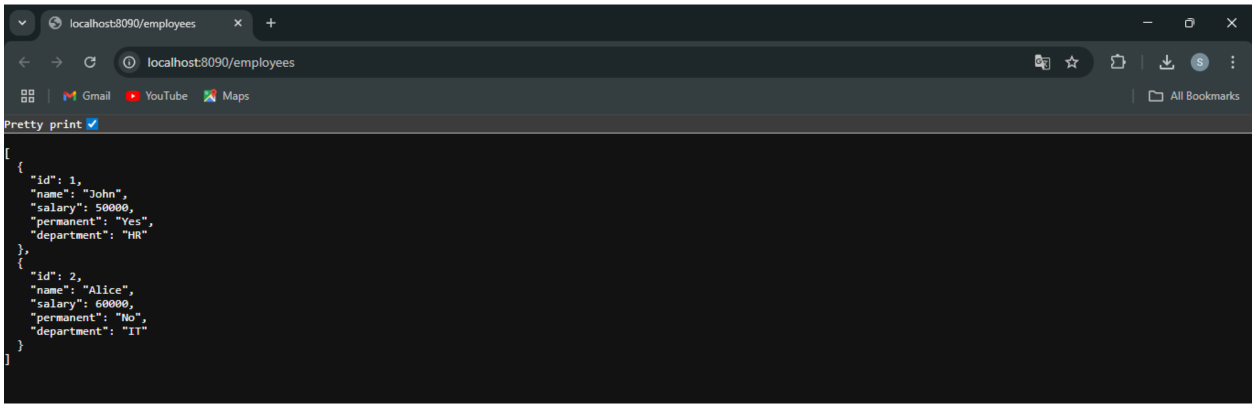
```
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication6 {

    private static final Logger LOGGER =
LoggerFactory.getLogger(SpringLearnApplication6.class);

    public static void main(String[] args) {
        LOGGER.info("START");
        SpringApplication.run(SpringLearnApplication6.class, args);
        LOGGER.info("END");
    }
}
```

Output:



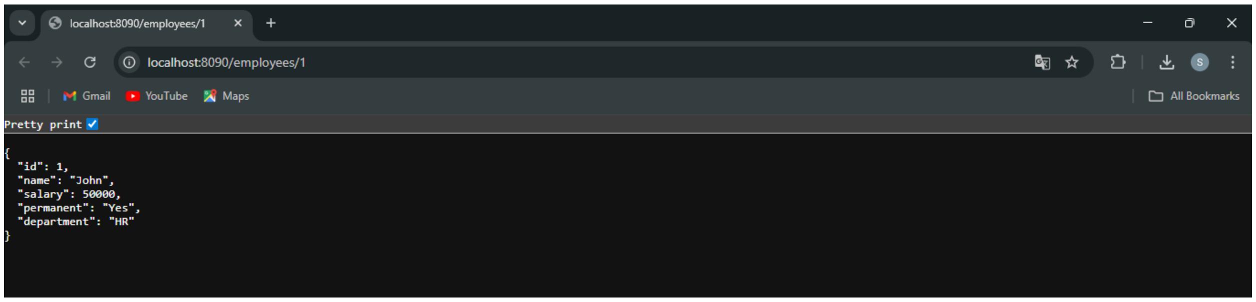
```
2025-07-13T16:59:03.118+05:30  INFO 14240 --- [springlearn] [nio-8090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcher
Servlet'
2025-07-13T16:59:03.119+05:30  INFO 14240 --- [springlearn] [nio-8090-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-13T16:59:03.123+05:30  INFO 14240 --- [springlearn] [nio-8090-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
Inside Employee Constructor.
Setting ID: 1
Setting Name: John
Setting Salary: 50000.0
Setting Permanent: Yes
Setting Department: HR
Inside Employee Constructor.
Setting ID: 2
Setting Name: Alice
Setting Salary: 60000.0
Setting Permanent: No
Setting Department: IT
Inside Employee Constructor.
Setting ID: 1
Setting Name: John
Setting Salary: 50000.0
Setting Permanent: Yes
Setting Department: HR
Inside Employee Constructor.
Setting ID: 2
Setting Name: Alice
Setting Salary: 60000.0
Setting Permanent: No
Setting Department: IT
Inside Employee Constructor.
Setting ID: 1
Setting Name: John
Setting Salary: 50000.0
Setting Permanent: Yes
Setting Department: HR
Inside Employee Constructor.
Setting ID: 2
Setting Name: Alice
Setting Salary: 60000.0
Setting Permanent: No
Setting Department: IT
```

http://localhost:8090/employees



```
[
  {
    "id": 1,
    "name": "John",
    "salary": 50000,
    "permanent": "Yes",
    "department": "HR"
  },
  {
    "id": 2,
    "name": "Alice",
    "salary": 60000,
    "permanent": "No",
    "department": "IT"
  }
]
```

http://localhost:8090/employees/1



```
{
  "id": 1,
  "name": "John",
  "salary": 50000,
  "permanent": "Yes",
  "department": "HR"
}
```

Exercise 8: 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

src/main/resources/employee1.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">

    <!-- Department Beans -->
    <bean id="department1"
class="com.cognizant.springlearn.model.Department">
        <property name="id" value="1" />
        <property name="name" value="HR" />
    </bean>

    <bean id="department2"
class="com.cognizant.springlearn.model.Department">
        <property name="id" value="2" />
        <property name="name" value="Finance" />
    </bean>

    <!-- Skill Beans -->
    <bean id="skill1" class="com.cognizant.springlearn.model.Skill">
        <property name="id" value="1" />
        <property name="name" value="Java" />
    </bean>

    <bean id="skill2" class="com.cognizant.springlearn.model.Skill">
        <property name="id" value="2" />
        <property name="name" value="Angular" />
    </bean>

    <!-- Employee List -->
    <bean id="employeeList1" class="java.util.ArrayList">
```

```
        <constructor-arg>
            <list>
                <bean class="com.cognizant.springlearn.model.Employee1">
                    <property name="id" value="1"/>
                    <property name="name" value="John"/>
                    <property name="salary" value="30000"/>
                    <property name="permanent" value="true"/>
                    <property name="dateOfBirth">
                        <value>1990/01/01</value>
                    </property>
                    <property name="department" ref="department1"/>
                    <property name="skills">
                        <list>
                            <ref bean="skill1"/>
                            <ref bean="skill2"/>
                        </list>
                    </property>
                </bean>

                <!-- You can add more <bean> for Employee1 here -->

            </list>
        </constructor-arg>
    </bean>

</beans>
```

src/main/java/com/cognizant/springlearn/model/Employee1.java

```
package com.cognizant.springlearn.model;

import java.util.Date;
import java.util.List;

public class Employee1 {
    private int id;
    private String name;
    private double salary;
    private boolean permanent;
    private Date dateOfBirth;
    private Department department;
    private List<Skill> skills;

    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> getSkills() {
    return skills;
}

public void setSkills(List<Skill> skills) {
    this.skills = skills;
}
```

```
    @Override
    public String toString() {
        return "Employee1{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", salary=" + salary +
                ", permanent=" + permanent +
                ", dateOfBirth=" + dateOfBirth +
                ", department=" + department +
                ", skills=" + skills +
                '}';
    }
}
```

src/main/java/com/cognizant/springlearn/model/Department.java

```
package com.cognizant.springlearn.model;

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 + '\'' +
                '}';
    }
}
```

src/main/java/com/cognizant/springlearn/model/Skill.java

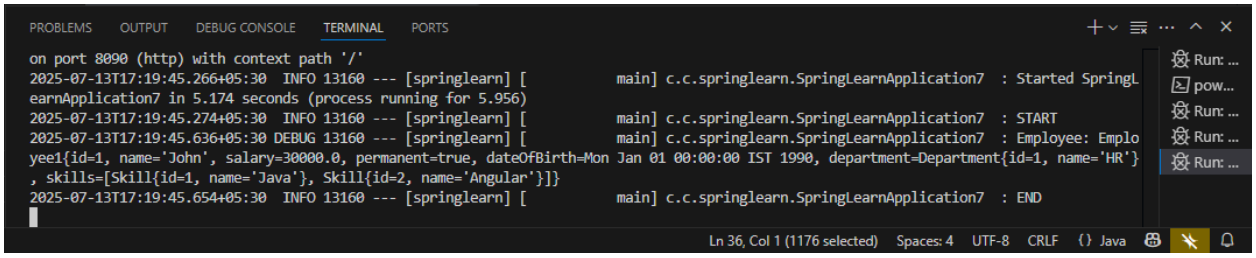
```
package com.cognizant.springlearn.model;
```

```
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;
    }

    @Override
    public String toString() {
        return "Skill{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication7.java

```
package com.cognizant.springlearn;

import com.cognizant.springlearn.model.Employee1;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.SpringApplication;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

@SpringBootApplication
public class SpringLearnApplication7 {
    private static final Logger LOGGER =
LoggerFactory.getLogger(SpringLearnApplication7.class);

    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication7.class, args);
        displayEmployees();
    }

    @SuppressWarnings("unchecked")
    public static void displayEmployees() {
        LOGGER.info("START");

        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("employee1.xml")) {
            List<Employee1> employees = (List<Employee1>)
context.getBean("employeeList1");
            for (Employee1 emp : employees) {
                LOGGER.debug("Employee: {}", emp);
            }
        }

        LOGGER.info("END");
    }

}
```

Output:



```
on port 8090 (http) with context path '/'
2025-07-13T17:19:45.266+05:30  INFO 13160 --- [springlearn] [           main] c.c.springlearn.SpringLearnApplication7  : Started SpringL
earnApplication7 in 5.174 seconds (process running for 5.956)
2025-07-13T17:19:45.274+05:30  INFO 13160 --- [springlearn] [           main] c.c.springlearn.SpringLearnApplication7  : START
2025-07-13T17:19:45.636+05:30 DEBUG 13160 --- [springlearn] [           main] c.c.springlearn.SpringLearnApplication7  : Employee: Emplo
yee1{id=1, name='John', salary=30000.0, permanent=true, dateOfBirth=Mon Jan 01 00:00:00 IST 1990, department=Department{id=1, name='HR'}
, skills=[Skill{id=1, name='Java'}, Skill{id=2, name='Angular'}]}
2025-07-13T17:19:45.654+05:30  INFO 13160 --- [springlearn] [           main] c.c.springlearn.SpringLearnApplication7  : END
```

Exercise 9: 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.

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>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
        </dependency>
        <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>
```

```xml
            <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/controller/EmployeeController1.java

```java
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Employee2;
import com.cognizant.springlearn.service.EmployeeService1;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class EmployeeController1 {
    private static final Logger LOGGER =
LoggerFactory.getLogger(EmployeeController1.class);

    @Autowired
    private EmployeeService1 employeeService;

    // Changed endpoint to avoid conflict with another controller
    @GetMapping("/employees-v2")
    public List<Employee2> getAllEmployees() {
        LOGGER.debug("START: getAllEmployees");
        List<Employee2> list = employeeService.getAllEmployees();
        LOGGER.debug("END: getAllEmployees");
        return list;
    }
}
```

src/main/java/com/cognizant/springlearn/service/EmployeeService1.java

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.dao.EmployeeDao;
import com.cognizant.springlearn.Employee2;
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 EmployeeService1 {

    @Autowired
    private EmployeeDao employeeDao;

    @Transactional
    public List<Employee2> getAllEmployees() {
        return employeeDao.getAllEmployees();
    }
}
```

src/main/java/com/cognizant/springlearn/dao/EmployeeDao.java

```
package com.cognizant.springlearn.dao;

import org.springframework.stereotype.Repository;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.ArrayList;
import java.util.List;
import com.cognizant.springlearn.Employee2;

@Repository
public class EmployeeDao {

    private static ArrayList<Employee2> EMPLOYEE_LIST;

    @SuppressWarnings("unchecked")
    public EmployeeDao() {
        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("employee1.xml")) {
            EMPLOYEE_LIST = (ArrayList<Employee2>)
context.getBean("employeeList1");
        } catch (Exception e) {
            System.err.println("Error loading employee1.xml: " +
e.getMessage());
            e.printStackTrace();
        }
    }

    public List<Employee2> getAllEmployees() {
        return EMPLOYEE_LIST;
    }
}
```

src/main/java/com/cognizant/springlearn/Employee2.java

```
package com.cognizant.springlearn;

import java.util.Date;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Employee2 {

    private static final Logger LOGGER =
LoggerFactory.getLogger(Employee2.class);

    private int id;
    private String name;
    private double salary;
    private boolean permanent;
    private Date dateOfBirth;
    private Department1 department;
    private List<Skill1> skillList;

    public Employee2() {
        LOGGER.debug("Inside Employee2 Constructor");
    }

    public int getId() {
        LOGGER.debug("Getting Id: {}", id);
        return id;
    }

    public void setId(int id) {
        LOGGER.debug("Setting Id: {}", id);
        this.id = id;
    }

    public String getName() {
        LOGGER.debug("Getting Name: {}", name);
        return name;
    }

    public void setName(String name) {
        LOGGER.debug("Setting Name: {}", name);
        this.name = name;
    }

    public double getSalary() {
        LOGGER.debug("Getting Salary: {}", salary);
        return salary;
    }

    public void setSalary(double salary) {
        LOGGER.debug("Setting Salary: {}", salary);
        this.salary = salary;
    }

    public boolean isPermanent() {
        LOGGER.debug("Getting Permanent: {}", permanent);
        return permanent;
    }

    public void setPermanent(boolean permanent) {
        LOGGER.debug("Setting Permanent: {}", permanent);
        this.permanent = permanent;
    }

    public Date getDateOfBirth() {
        LOGGER.debug("Getting DOB: {}", dateOfBirth);
        return dateOfBirth;
    }

    public void setDateOfBirth(Date dateOfBirth) {
        LOGGER.debug("Setting DOB: {}", dateOfBirth);
        this.dateOfBirth = dateOfBirth;
    }

    public Department1 getDepartment() {
        LOGGER.debug("Getting Department: {}", department);
        return department;
    }

    public void setDepartment(Department1 department) {
        LOGGER.debug("Setting Department: {}", department);
        this.department = department;
    }

    public List<Skill1> getSkillList() {
        LOGGER.debug("Getting Skill List: {}", skillList);
        return skillList;
    }

    public void setSkillList(List<Skill1> skillList) {
        LOGGER.debug("Setting Skill List: {}", skillList);
        this.skillList = skillList;
    }

    @Override
    public String toString() {
        return "Employee2 [id=" + id + ", name=" + name + ", salary=" +
salary +
                ", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth +
                ", department=" + department + ", skillList=" + skillList +
"]";
    }
}
```

src/main/java/com/cognizant/springlearn/Department1.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Department1 {

    private static final Logger LOGGER =
LoggerFactory.getLogger(Department1.class);

    private int id;
    private String name;

    public Department1() {
        LOGGER.debug("Inside Department1 Constructor");
    }

    public int getId() {
        LOGGER.debug("Getting Department Id: {}", id);
        return id;
    }

    public void setId(int id) {
        LOGGER.debug("Setting Department Id: {}", id);
        this.id = id;
    }

    public String getName() {
        LOGGER.debug("Getting Department Name: {}", name);
        return name;
    }

    public void setName(String name) {
        LOGGER.debug("Setting Department Name: {}", name);
        this.name = name;
    }

    @Override
    public String toString() {
        return "Department1 [id=" + id + ", name=" + name + "]";
    }
}
```

src/main/java/com/cognizant/springlearn/Skill1.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Skill1 {

    private static final Logger LOGGER =
LoggerFactory.getLogger(Skill1.class);

    private int id;
    private String name;

    public Skill1() {
        LOGGER.debug("Inside Skill1 Constructor");
    }

    public int getId() {
        LOGGER.debug("Getting Skill Id: {}", id);
        return id;
    }

    public void setId(int id) {
        LOGGER.debug("Setting Skill Id: {}", id);
        this.id = id;
    }

    public String getName() {
        LOGGER.debug("Getting Skill Name: {}", name);
        return name;
    }

    public void setName(String name) {
        LOGGER.debug("Setting Skill Name: {}", name);
        this.name = name;
    }

    @Override
    public String toString() {
        return "Skill1 [id=" + id + ", name=" + name + "]";
    }
}
```

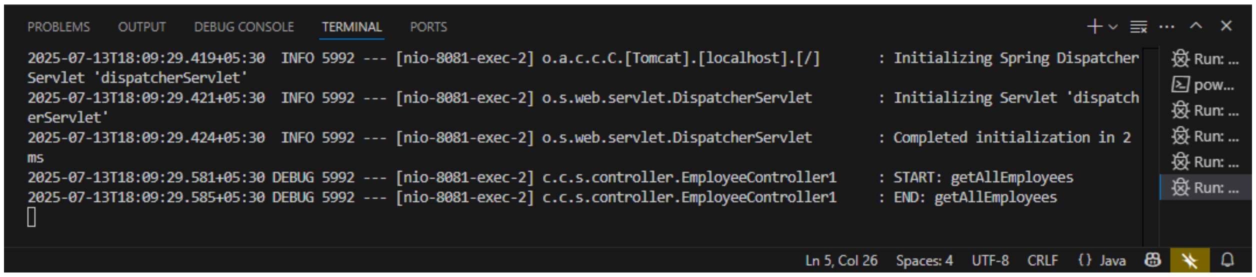
src/main/resources/application.properties

```
server.port=8081
logging.level.com.cognizant=DEBUG
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication8.java

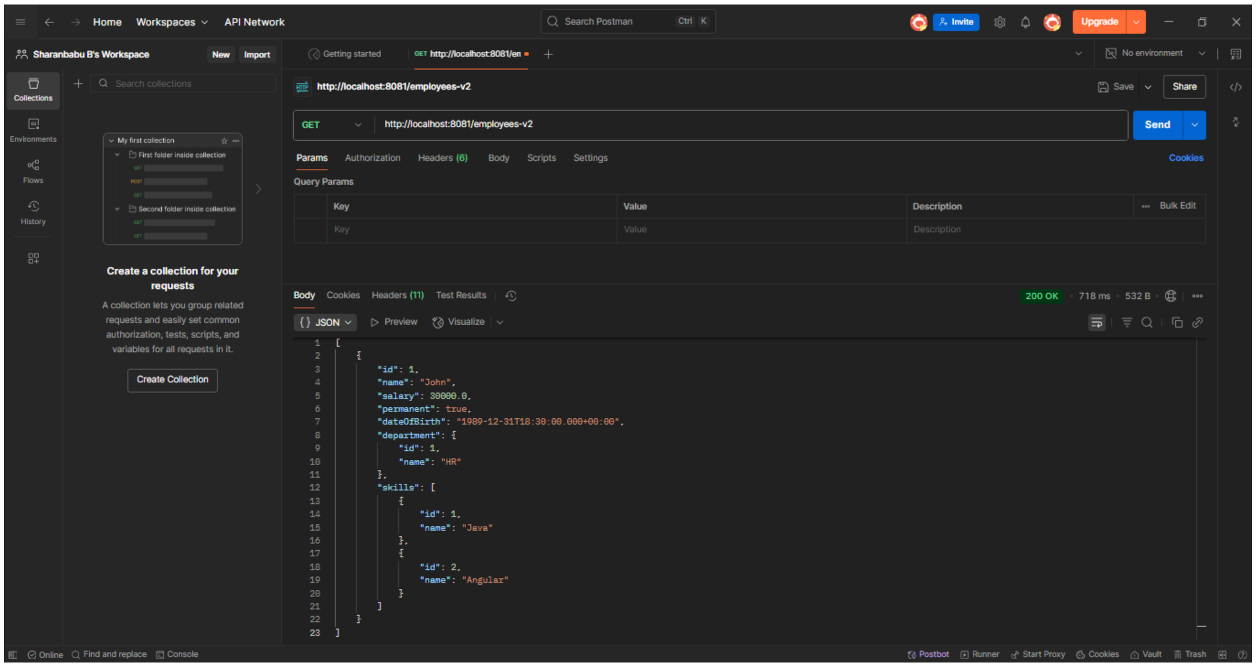
```
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication8 {

    private static final Logger LOGGER =
LoggerFactory.getLogger(SpringLearnApplication8.class);

    public static void main(String[] args) {
        LOGGER.info("START");
        SpringApplication.run(SpringLearnApplication8.class, args);
        LOGGER.info("END");
    }
}
```

Output:



```
2025-07-13T18:09:29.419+05:30  INFO 5992 --- [nio-8081-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Dispatcher
Servlet 'dispatcherServlet'
2025-07-13T18:09:29.421+05:30  INFO 5992 --- [nio-8081-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatch
erServlet'
2025-07-13T18:09:29.424+05:30  INFO 5992 --- [nio-8081-exec-2] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2
ms
2025-07-13T18:09:29.581+05:30 DEBUG 5992 --- [nio-8081-exec-2] c.c.s.controller.EmployeeController1     : START: getAllEmployees
2025-07-13T18:09:29.585+05:30 DEBUG 5992 --- [nio-8081-exec-2] c.c.s.controller.EmployeeController1     : END: getAllEmployees
```

Postman Terminal:



GET http://localhost:8081/employees-v2

200 OK · 718 ms · 532 B

```
[
    {
        "id": 1,
        "name": "John",
        "salary": 30000.0,
        "permanent": true,
        "dateOfBirth": "1989-12-31T18:30:00.000+00:00",
        "department": {
            "id": 1,
            "name": "HR"
        },
        "skills": [
            {
                "id": 1,
                "name": "Java"
            },
            {
                "id": 2,
                "name": "Angular"
            }
        ]
    }
]
```

Exercise 10: 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.

src/main/resources/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
                           http://www.springframework.org/schema/beans/spring
-beans.xsd">

    <bean id="departmentList" class="java.util.ArrayList">
        <constructor-arg>
            <list>
                <bean class="com.cognizant.springlearn.model.Department1">
                    <property name="id" value="1"/>
                    <property name="name" value="Human Resources"/>
                </bean>
                <bean class="com.cognizant.springlearn.model.Department1">
                    <property name="id" value="2"/>
                    <property name="name" value="Finance"/>
                </bean>
            </list>
        </constructor-arg>
    </bean>

</beans>
```

src/main/resources/application.properties
```
server.port=8090
logging.level.com.cognizant=DEBUG
```

src/main/java/com/cognizant/springlearn/controller/DepartmentController.java
```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Department1;
import com.cognizant.springlearn.service.DepartmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class DepartmentController {

    @Autowired
    private DepartmentService departmentService;

    @GetMapping("/departments")
    public List<Department1> getAllDepartments() {
        return departmentService.getAllDepartments();
    }
}
```

src/main/java/com/cognizant/springlearn/service/DepartmentService.java

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.dao.DepartmentDao;
import com.cognizant.springlearn.model.Department1;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class DepartmentService {

    @Autowired
    private DepartmentDao departmentDao;

    public List<Department1> getAllDepartments() {
        return departmentDao.getAllDepartments();
    }
}
```

src/main/java/com/cognizant/springlearn/dao/DepartmentDao.java

```
package com.cognizant.springlearn.dao;

import com.cognizant.springlearn.model.Department1;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class DepartmentDao {

    private static List<Department1> DEPARTMENT_LIST;

    @SuppressWarnings("unchecked")
    public DepartmentDao() {
        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("department.xml")) {
            DEPARTMENT_LIST = (List<Department1>)
context.getBean("departmentList");
        }
    }

    public List<Department1> getAllDepartments() {
        return DEPARTMENT_LIST;
    }
}
```

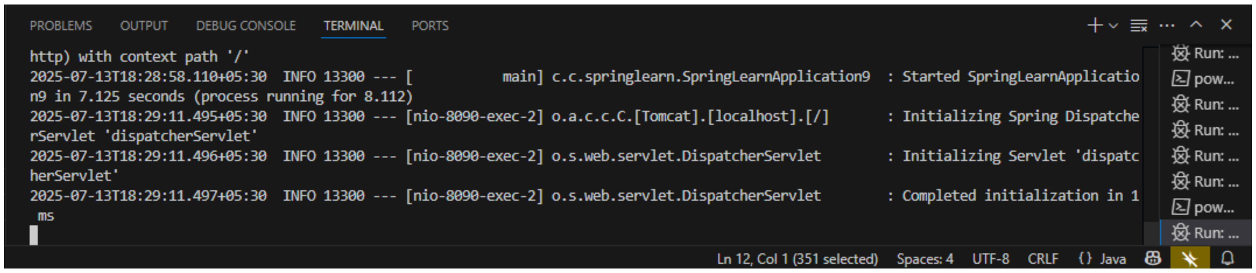
src/main/java/com/cognizant/springlearn/model/Department1.java

```
package com.cognizant.springlearn.model;

public class Department1 {
    private int id;
    private String name;

    public Department1() {
    }

    public Department1(int id, String name) {
        this.id = id;
        this.name = name;
    }

    // Getters & Setters
    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 "Department1 [id=" + id + ", name=" + name + "]";
    }
}
```

src/main/java/com/cognizant/springlearn/SpringLearnApplication9.java

```
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication9 {
    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication9.class, args);
    }
}
```

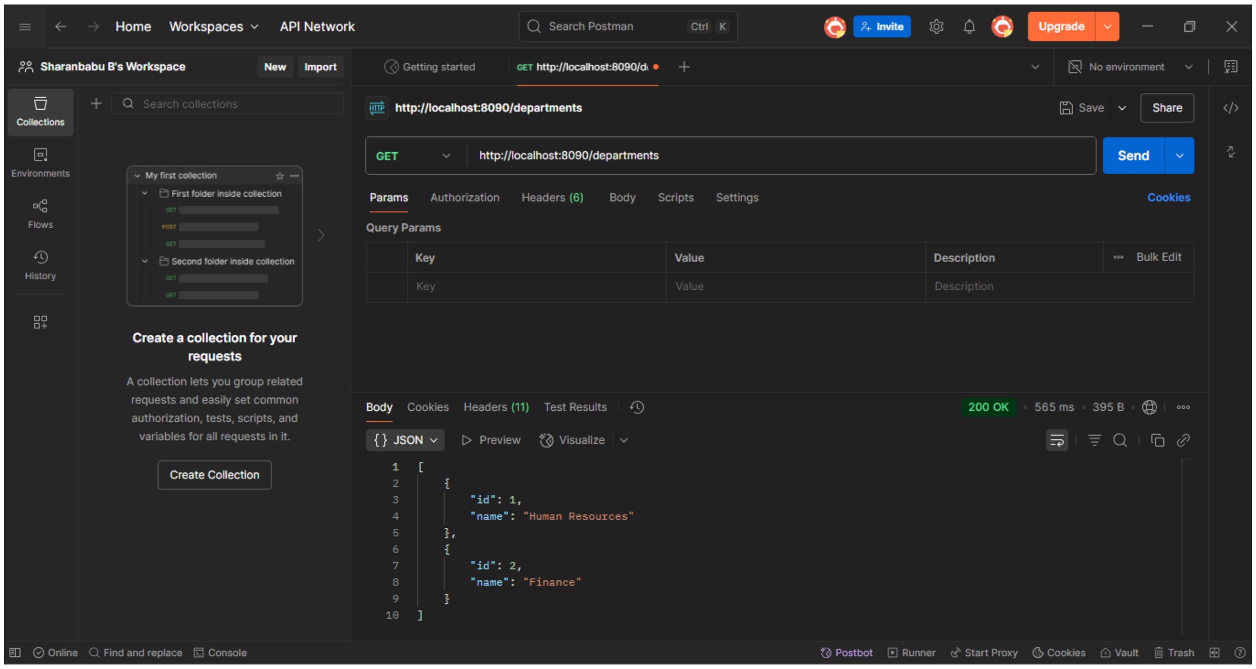
Output:



```
http) with context path '/'
2025-07-13T18:28:58.110+05:30  INFO 13300 --- [           main] c.c.springlearn.SpringLearnApplication9  : Started SpringLearnApplicatio
n9 in 7.125 seconds (process running for 8.112)
2025-07-13T18:29:11.495+05:30  INFO 13300 --- [nio-8090-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Dispatche
rServlet 'dispatcherServlet'
2025-07-13T18:29:11.496+05:30  INFO 13300 --- [nio-8090-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatc
herServlet'
2025-07-13T18:29:11.497+05:30  INFO 13300 --- [nio-8090-exec-2] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1
 ms
```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Ln 12, Col 1 (351 selected) Spaces: 4 UTF-8 CRLF {} Java

Postman Terminal:



GET http://localhost:8090/departments

200 OK · 565 ms · 395 B

```
[
    {
        "id": 1,
        "name": "Human Resources"
    },
    {
        "id": 2,
        "name": "Finance"
    }
]
```