

Spring REST using Spring Boot 3

1.spring-rest-handson

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:

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>sping-learn</artifactId>
  <version>1.0</version>
  <name>sping-learn</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>
```

```

        <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;

@SpringBootApplication
public class SpringLearnApplication {

    public static void main(String[] args) {
        System.out.println("Application starting...");
        SpringApplication.run(SpringLearnApplication.class, args);
        System.out.println("Application started successfully.");
    }
}

```

Output:

```

Application starting...
Application starting...

  ____ _
 / ___ \| | | |
/ /   \| |_| |
\ \   /| | | |
 \___/\|_|_|_|

:: Spring Boot ::                (v3.5.3)

2025-07-09T10:43:39.922+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:39.928+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:40.088+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:40.088+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:42.346+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:42.378+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:42.379+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:42.459+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:42.461+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:43.257+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:43.332+05:30 INFO 26544 --- [spring-learn]
2025-07-09T10:43:43.354+05:30 INFO 26544 --- [spring-learn]
Application started successfully.

```

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 **co**

`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

Code:

Country.java

```
package com.cognizant.springlearn;  
  
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.springlearn.Country">
        <property name="code" value="IN" />
        <property name="name" value="India" />
    </bean>

</beans>

```

SpringLearnApplication.java

```

package com.cognizant.springlearn;

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

```

```

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

@SpringBootApplication
public class SpringLearnApplication {

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

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

        // ☞ You must call this method here:
        displayCountry();
    }

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

```

Output:

```

com.cognizant.springlearn.Country - Inside Country Constructor.
com.cognizant.springlearn.Country - Inside setCode.
com.cognizant.springlearn.Country - Inside setName.
com.cognizant.springlearn.Country - Inside getCode.
com.cognizant.springlearn.Country - Inside getName.
com.cognizant.springlearn.SpringLearnApplication - Country : Country [code=IN, name=India]

```

2.spring-rest-handson

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:

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;
    }
}

```

application.properties

```

spring.application.name=spring-learn
logging.level.com.cognizant=DEBUG
server.port=8083

```

SpringLearnApplication.java

```

package com.cognizant.springlearn;

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

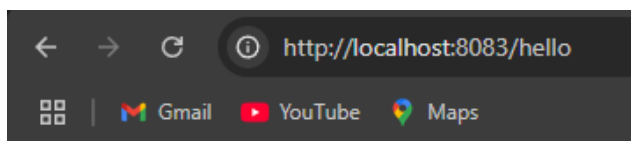
@SpringBootApplication
public class SpringLearnApplication {

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

}

```

Output:



Hello World!!

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 response?
- 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:

Country.java

```
package com.cognizant.springlearn;

import java.io.Serializable;

public class Country implements Serializable {
    private String code;
    private String name;

    public Country() {}

    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;
    }

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

```

CountryController.java

```

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class CountryController {

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

```

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.springlearn.Country">
  <property name="code" value="IN" />
  <property name="name" value="India" />
</bean>
</beans>
```

SpringLearnApplication.java

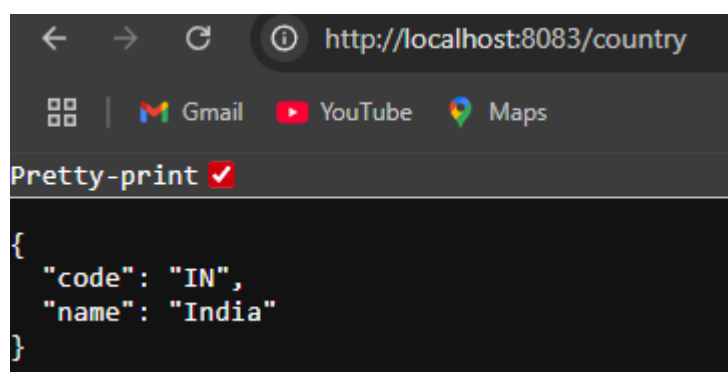
```
package com.cognizant.springlearn;

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

@SpringBootApplication
public class SpringLearnApplication {

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

Output:



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

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 Implementation: 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:

Country.java

```
package com.cognizant.springlearn;

import java.io.Serializable;

public class Country implements Serializable {
    private String code;
    private String name;

    public Country() {}

    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;
    }

    @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"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/util
        https://www.springframework.org/schema/util/spring-util.xsd">

    <util:list id="countryList" value-type="com.cognizant.springlearn.Country">
        <bean class="com.cognizant.springlearn.Country">
            <property name="code" value="IN"/>
            <property name="name" value="India"/>
        </bean>
        <bean class="com.cognizant.springlearn.Country">
            <property name="code" value="US"/>
            <property name="name" value="United States"/>
        </bean>
        <bean class="com.cognizant.springlearn.Country">
            <property name="code" value="DE"/>
            <property name="name" value="Germany"/>
        </bean>
        <bean class="com.cognizant.springlearn.Country">
            <property name="code" value="JP"/>
            <property name="name" value="Japan"/>
        </bean>
    </util:list>
</beans>

```

CountryService.java

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

import com.cognizant.springlearn.Country;
import org.springframework.stereotype.Service;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

@Service
public class CountryService {

    public Country getCountry(String code) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("country.xml");
        List<Country> countryList = context.getBean("countryList", List.class);

        return countryList.stream()
            .filter(c -> c.getCode().equalsIgnoreCase(code))
            .findFirst()
            .orElse(null);
    }
}
```

CountryController.java

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

import com.cognizant.springlearn.Country;
import com.cognizant.springlearn.service.CountryService;

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

@RestController
public class CountryController {

    @Autowired
    private CountryService countryService;

    @GetMapping("/countries/{code}")
    public Country getCountry(@PathVariable String code) {
        return countryService.getCountry(code);
    }
}
```

SpringLearnApplication.java

```
package com.cognizant.springlearn;

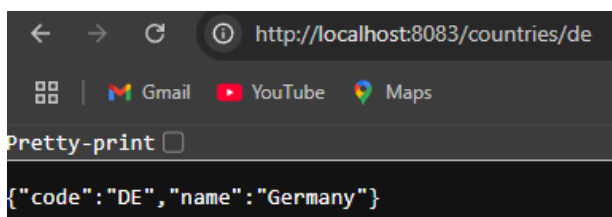
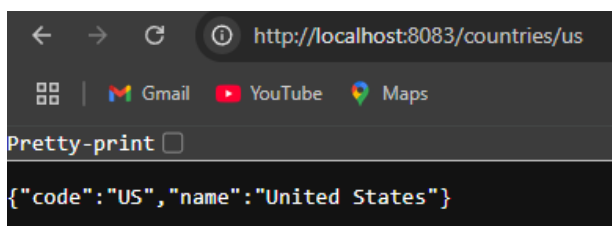
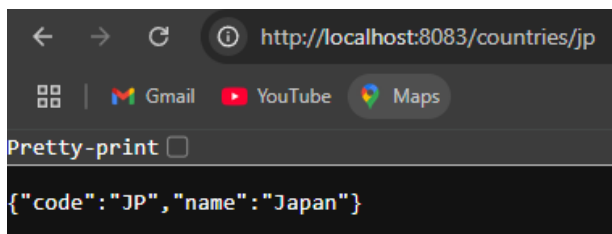
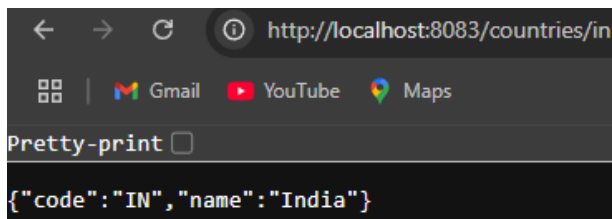
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringLearnApplication {

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

}
```

Output:



5.JWT-handson

Create authentication service that returns JWT

Create authentication controller and configure it in SecurityConfig

AuthenticationController.java

- Create new rest controller named AuthenticationController in controller package
- Include method authenticate with "/authenticate" as the URL with @GetMapping.
- To read the Authorization value from HTTP Header, include a parameter for authenticate method as specified below. Spring takes care of reading the Authorization value from HTTP Header and pass it as parameter.

```
@RequestHeader("Authorization") String authHeader
```

- The return type of this method should be Map<String, String>
- Include start and end logger in this method
- Include a debug log for displaying the authHeader parameter
- Create a new HashMap<String, String> and assign it to a map.
- Put a new item into the map with key as "token" and value as empty string.

SecurityConfig.java

- In the second configure method, include authenticate URL just after the countries URL defined earlier. Refer code below:

```
.antMatchers("/countries").hasRole("USER")  
.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")
```

- The above configuration sets that users of both roles can access /authenticate URL.

Testing

curl command:

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

Expected Response:

```
{"token":""}
```

Log verification:

Check if Authorization header value is displayed with "Basic" prefix and Base64 encoding of "user:pwd"

Code:

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>sping-jwt</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>sping-jwt</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-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-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>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>

</dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>

```

SecurityConfig.java

```

package com.cognizant.springjwt.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
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;

@Configuration
public class SecurityConfig {

    @Bean
    public InMemoryUserDetailsManager userDetailsService() {
        UserDetails user = User.withUsername("user")
            .password("{noop}pwd")
            .roles("USER")
            .build();
    }
}

```

```

        UserDetails admin = User.withUsername("admin")
            .password("{noop}admin123")
            .roles("ADMIN")
            .build();

        return new InMemoryUserDetailsManager(user, admin);
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/login", "/logout", "/css/**").permitAll()
                .requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")
                .anyRequest().authenticated()
            )
            .formLogin(form -> form
                .loginPage("/login")
                .defaultSuccessUrl("/home", true)
                .permitAll()
            )
            .logout(logout -> logout
                .logoutUrl("/logout")
                .logoutSuccessUrl("/login?logout")
                .invalidateHttpSession(true)
                .clearAuthentication(true)
            );

        return http.build();
    }
}

```

AuthenticationController.java

```

package com.cognizant.springjwt.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;

@RestController
public class AuthenticationController {

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

    @GetMapping("/authenticate")

```

```

    public Map<String, String> authenticate(@RequestHeader("Authorization") String
authHeader) {
        LOGGER.info("START - authenticate()");
        LOGGER.debug("Authorization Header: {}", authHeader);

        Map<String, String> map = new HashMap<>();
        map.put("token", "");
        LOGGER.info("END - authenticate()");
        return map;
    }
}

```

LoginController.java

```

package com.cognizant.springjwt.controller;

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

@Controller
public class LoginController {

    @GetMapping("/login")
    public String loginPage() {
        return "login"; // maps to login.html
    }

    @GetMapping("/home")
    public String homePage() {
        return "home"; // maps to home.html
    }
}

```

application.properties

```

spring.application.name=sping-jwt
server.port=8090
spring.thymeleaf.cache=false
logging.level.com.cognizant=DEBUG

```

login.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Login</title>
</head>
<body>
<h2>Please login</h2>
<form th:action="@{/login}" method="post">
    <label>Username: <input type="text" name="username"/></label><br/>
    <label>Password: <input type="password" name="password"/></label><br/>
    <button type="submit">Login</button>
</form>
<p th:if="${param.error}">Invalid username or password</p>
<p th:if="${param.logout}">You have been logged out</p>
</body>
</html>
```

home.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Home</title>
</head>
<body>
<h2>Welcome! You are logged in.</h2>
<a th:href="@{/logout}">Logout</a>
</body>
</html>
```

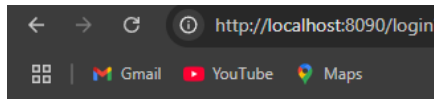
SpringJwtApplication.java

```
package com.cognizant.springjwt;

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

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

Output:



Please login

Username:

Password:

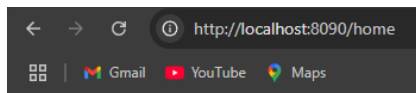
Login

Please login

Username:

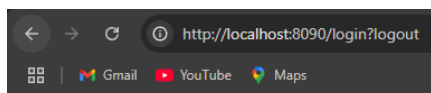
Password:

Login



Welcome! You are logged in.

[Logout](#)



Please login

Username:

Password:

Login

You have been logged out

```
{"token":""}
```

Read Authorization header and decode the username and password

Steps to read and decode header:

- Create a new private method in AuthenticationController with below method signature

```
private String getUser(String authHeader)
```

- Get the Base64 encoded text after "Basic "
- Decode it using the library available in Java 8 API. Refer code below.

```
Base64.getDecoder().decode(encodedCredentials)
```

- The above call returns a byte array, which can be passed as parameter to string constructor to convert to string.
- Get the text until colon on the string created in previous step to get the user
- Return the user obtained in previous step
- Include appropriate debug logs within this method
- Invoke the getUser() method from authenticate method
- Execute the curl command used in the previous step and check the logs if the user information is obtained successfully.

Code:

AuthenticationController.java

```
package com.cognizant.springjwt.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.*;

import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

@RestController
public class AuthenticationController {

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

    @GetMapping("/authenticate")
```



```

    public Map<String, String> authenticate(@RequestHeader("Authorization") String
authHeader) {
        LOGGER.info("START - authenticate()");
        LOGGER.debug("Authorization Header: {}", authHeader);

        String username = getUser(authHeader);
        LOGGER.debug("Decoded Username: {}", username);

        Map<String, String> map = new HashMap<>();
        map.put("token", ""); // placeholder
        LOGGER.info("END - authenticate()");
        return map;
    }

    // Private method to decode username from Authorization header
    private String getUser(String authHeader) {
        LOGGER.debug("START - getUser()");

        // Remove "Basic " prefix
        String encodedCredentials = authHeader.substring("Basic ".length());
        LOGGER.debug("Encoded Credentials: {}", encodedCredentials);

        // Decode from Base64
        byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);
        String decodedString = new String(decodedBytes);
        LOGGER.debug("Decoded Credentials: {}", decodedString); // Format:
user:pwd

        // Extract username before colon
        String username = decodedString.split(":")[0];
        LOGGER.debug("Extracted Username: {}", username);

        LOGGER.debug("END - getUser()");
        return username;
    }
}

```

Output:

```

INFO AuthenticationController - START - authenticate()
DEBUG AuthenticationController - Authorization Header: Basic dXNlcjpwd2Q=
DEBUG AuthenticationController - START - getUser()
DEBUG AuthenticationController - Encoded Credentials: dXNlcjpwd2Q=
DEBUG AuthenticationController - Decoded Credentials: user:pwd
DEBUG AuthenticationController - Extracted Username: user
DEBUG AuthenticationController - END - getUser()
DEBUG AuthenticationController - Decoded Username: user
INFO AuthenticationController - END - authenticate()

```

```
{"token": ""}
```

Generate token based on the user

Steps to generate token:

- Include JWT library by including the following maven dependency.

```
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt</artifactId>
  <version>0.9.0</version>
</dependency>
```

- After inclusion in pom.xml, run the maven package command line and update the project in Eclipse. View the dependency tree and check if the library is added.
- Create a new method in AuthenticationController with below method signature:

```
private String generateJwt(String user)
```

- Generate the token based on the code specified below.

```
JwtBuilder builder = Jwts.builder();
builder.setSubject(user);

// Set the token issue time as current time
builder.setIssuedAt(new Date());

// Set the token expiry as 20 minutes from now
builder.setExpiration(new Date((new Date()).getTime() + 1200000));
builder.signWith(SignatureAlgorithm.HS256, "secretkey");

String token = builder.compact();

return token;
```

- Import reference for the above code

```
import io.jsonwebtoken.JwtBuilder;
```

```
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
```

- Invoke this method from authenticate() method passing the user obtained from getUser() method.
- Add the token into the map using put method.
- Include appropriate logs

Execute the curl command for authenticate and check if the generated token is returned.

Code:

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
    http://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>sping-jwt</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>sping-jwt</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-security</artifactId>
    </dependency>
```

```

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.0</version>
</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>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-logging</artifactId>
</dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>

```

AuthenticationController.java

```
package com.cognizant.springjwt.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.*;

import java.util.Base64;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import io.jsonwebtoken.JwtBuilder;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

@RestController
public class AuthenticationController {

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

    @GetMapping("/authenticate")
    public Map<String, String> authenticate(@RequestHeader("Authorization") String
authHeader) {
        LOGGER.info("START - authenticate()");
        LOGGER.debug("Authorization Header: {}", authHeader);

        String username = getUser(authHeader);
        LOGGER.debug("Decoded Username: {}", username);

        String token = generateJwt(username);
        LOGGER.debug("Generated Token: {}", token);

        Map<String, String> map = new HashMap<>();
        map.put("token", token);

        LOGGER.info("END - authenticate()");
        return map;
    }

    private String getUser(String authHeader) {
        LOGGER.debug("START - getUser()");

        String encodedCredentials = authHeader.substring("Basic ".length());
        LOGGER.debug("Encoded Credentials: {}", encodedCredentials);

        byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);
        String decodedString = new String(decodedBytes); // user:pwd
        LOGGER.debug("Decoded Credentials: {}", decodedString);

        String username = decodedString.split(":")[0];
        LOGGER.debug("Extracted Username: {}", username);

        LOGGER.debug("END - getUser()");
        return username;
    }
}
```

```
// Generate JWT Token
private String generateJwt(String user) {
    LOGGER.debug("START - generateJwt() for user: {}", user);

    JwtBuilder builder = Jwts.builder();
    builder.setSubject(user);
    builder.setIssuedAt(new Date());
    builder.setExpiration(new Date(System.currentTimeMillis() + 20 * 60 *
1000)); // 20 minutes
    builder.signWith(SignatureAlgorithm.HS256, "secretkey");

    String token = builder.compact();

    LOGGER.debug("END - generateJwt()");
    return token;
}
}
```

Output:

```
{
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1c2VyIiwiaWF0IjoxNzEwODI4LCJle
}
```

```
INFO AuthenticationController - START - authenticate()
DEBUG AuthenticationController - Authorization Header: Basic dXNlcjpwd2Q=
DEBUG AuthenticationController - START - getUser()
DEBUG AuthenticationController - Encoded Credentials: dXNlcjpwd2Q=
DEBUG AuthenticationController - Decoded Credentials: user\:pwd
DEBUG AuthenticationController - Extracted Username: user
DEBUG AuthenticationController - END - getUser()
DEBUG AuthenticationController - Decoded Username: user
DEBUG AuthenticationController - START - generateJwt() for user: user
DEBUG AuthenticationController - END - generateJwt()
DEBUG AuthenticationController - Generated Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1c2VyIiwiaWF0IjoxNzEwODI4LCJle
AuthenticationController - END - authenticate()
```

3.Spring-rest-handson

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.

Code:

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.example</groupId>
    <artifactId>employee-rest-xml</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>employee-rest-xml</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-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!-- Spring Context to support XML beans -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
    </dependency>

    <!-- JAXB for XML parsing -->
    <dependency>
        <groupId>jakarta.xml.bind</groupId>
        <artifactId>jakarta.xml.bind-api</artifactId>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
    </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

```



```
        </plugins>
    </build>

</project>
```

employees.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<employees>
    <employee>
        <id>101</id>
        <name>John Doe</name>
        <designation>Developer</designation>
        <salary>50000</salary>
    </employee>
    <employee>
        <id>102</id>
        <name>Jane Smith</name>
        <designation>Manager</designation>
        <salary>70000</salary>
    </employee>
</employees>
```

spring-beans.xml

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

Employee.java

```
package com.example.employee.model;

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

    // Getters and 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; }

    public String getDesignation() { return designation; }
    public void setDesignation(String designation) { this.designation = designation; }

    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
}
```

EmployeeService.java

```
package com.example.employee.service;

import com.example.employee.model.Employee;
import jakarta.annotation.PostConstruct;
import org.springframework.stereotype.Service;

import javax.xml.parsers.DocumentBuilderFactory;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import org.w3c.dom.*;

@Service
public class EmployeeService {

    private List<Employee> employees = new ArrayList<>();

    @PostConstruct
    public void init() {
        try {
            InputStream is =
getClass().getClassLoader().getResourceAsStream("employees.xml");
```

```

        Document doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(is);
        NodeList nodes = doc.getElementsByTagName("employee");
        for (int i = 0; i < nodes.getLength(); i++) {
            Element e = (Element) nodes.item(i);
            Employee emp = new Employee();

emp.setId(Integer.parseInt(e.getElementsByTagName("id").item(0).getTextContent()))
;

emp.setName(e.getElementsByTagName("name").item(0).getTextContent());

emp.setDesignation(e.getElementsByTagName("designation").item(0).getTextContent())
;

emp.setSalary(Double.parseDouble(e.getElementsByTagName("salary").item(0).getTextC
ontent()));
            employees.add(emp);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public List<Employee> getAllEmployees() {
    return employees;
}

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

public void updateEmployee(Employee updatedEmp) {
    for (int i = 0; i < employees.size(); i++) {
        if (employees.get(i).getId() == updatedEmp.getId()) {
            employees.set(i, updatedEmp);
            break;
        }
    }
}
}
}

```

EmployeeController.java

```

package com.example.emplyee.controller;

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

```

```

import java.util.List;

@RestController
@RequestMapping("/api/employees")
@CrossOrigin(origins = "*") // enable Angular access
public class EmployeeController {

    @Autowired
    private EmployeeService service;

    @GetMapping
    public List<Employee> getAll() {
        return service.getAllEmployees();
    }

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

    @PutMapping("/{id}")
    public String update(@PathVariable int id, @RequestBody Employee emp) {
        service.updateEmployee(emp);
        return "Employee updated!";
    }
}

```

EmployeeRestXmlApplication.java

```

package com.example.employee;

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

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

```

application.properties

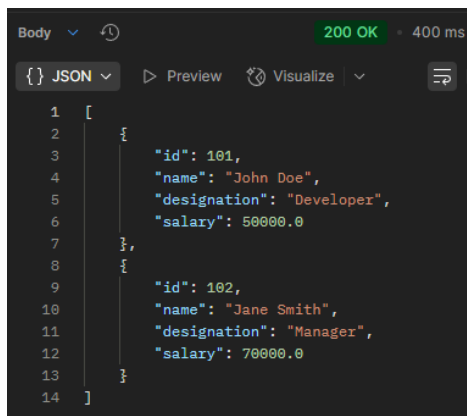
```

spring.application.name=employee-rest-xml
server.port=8080
spring.main.allow-bean-definition-overriding=true

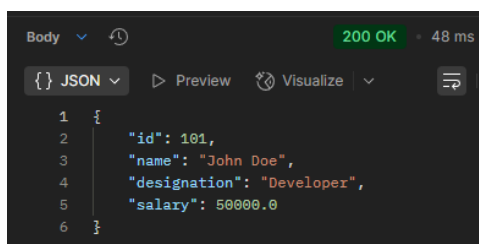
```

Output(Using Postman):

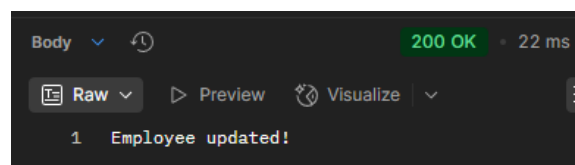
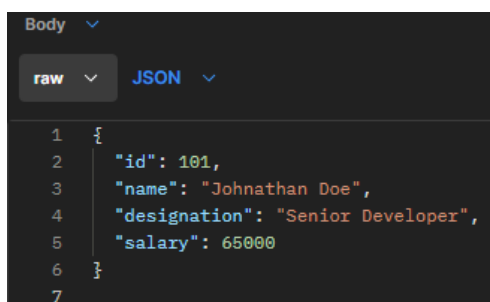
GET <http://localhost:8080/api/employees>



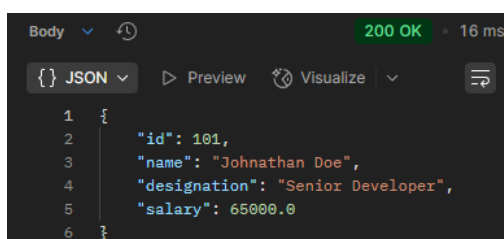
GET <http://localhost:8080/api/employees/101>



PUT <http://localhost:8080/api/employees/101>



GET <http://localhost:8080/api/employees/101>



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:

Employee.java

```
package com.example.employee.model;

import java.util.List;

public class Employee {
    private int id;
    private String name;
    private String designation;
    private double salary;
    private String department;
    private List<String> skills;

    public Employee() {}

    // Getters and 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; }

    public String getDesignation() { return designation; }
```

```

    public void setDesignation(String designation) { this.designation =
designation; }

    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }

    public String getDepartment() { return department; }
    public void setDepartment(String department) { this.department = department; }

    public List<String> getSkills() { return skills; }
    public void setSkills(List<String> skills) { this.skills = skills; }
}

```

EmployeeDao.java

```

package com.example.employee.dao;

import com.example.employee.model.Employee;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.List;

@Repository
public class EmployeeDao {

    private List<Employee> employees;

    public EmployeeDao() {
        ApplicationContext context = new
ClassPathXmlApplicationContext("employees.xml");
        this.employees = new ArrayList<>();
        this.employees.add(context.getBean("emp1", Employee.class));
        this.employees.add(context.getBean("emp2", Employee.class));
    }

    public List<Employee> getAllEmployees() {
        return employees;
    }

    public Employee getById(int id) {
        for (Employee emp : employees) {
            if (emp.getId() == id) {
                return emp;
            }
        }
        return null;
    }
}

```

```

    }

    public void updateEmployee(Employee updatedEmp) {
        for (int i = 0; i < employees.size(); i++) {
            if (employees.get(i).getId() == updatedEmp.getId()) {
                employees.set(i, updatedEmp);
                break;
            }
        }
    }
}

```

EmployeeController.java

```

package com.example.employee.controller;

import com.example.employee.model.Employee;
import com.example.employee.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> getAllEmployees() {
        return service.getAllEmployees();
    }

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

    @PutMapping("/employees")
    public void updateEmployee(@RequestBody Employee emp) {
        service.updateEmployee(emp);
    }
}

```


EmployeeService.java

```
package com.example.employee.service;

import com.example.employee.dao.EmployeeDao;
import com.example.employee.model.Employee;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService {

    @Autowired
    private EmployeeDao dao;

    public List<Employee> getAllEmployees() {
        return dao.getAllEmployees();
    }

    public Employee getEmployeeById(int id) {
        return dao.getById(id);
    }

    public void updateEmployee(Employee emp) {
        dao.updateEmployee(emp);
    }
}
```

EmployeeRestXmlApplication.java

```
package com.example.employee;

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

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

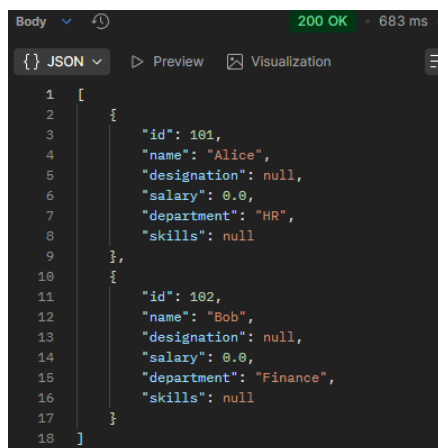
employees.xml

```
<?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="emp1" class="com.example.employee.model.Employee">  
        <property name="id" value="101"/>  
        <property name="name" value="Alice"/>  
        <property name="department" value="HR"/>  
    </bean>  
  
    <bean id="emp2" class="com.example.employee.model.Employee">  
        <property name="id" value="102"/>  
        <property name="name" value="Bob"/>  
        <property name="department" value="Finance"/>  
    </bean>  
  
</beans>
```

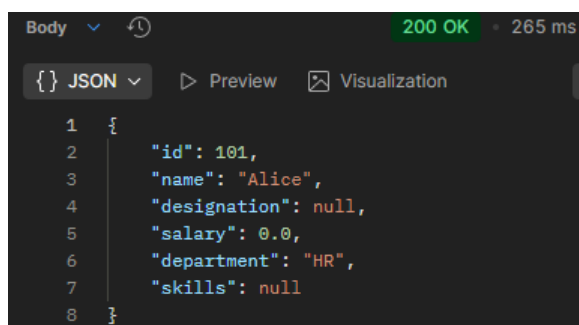
Output:

GET <http://localhost:8080/employees>



```
Body 200 OK - 683 ms  
{ } JSON Preview Visualization  
1  [  
2    {  
3      "id": 101,  
4      "name": "Alice",  
5      "designation": null,  
6      "salary": 0.0,  
7      "department": "HR",  
8      "skills": null  
9    },  
10   {  
11     "id": 102,  
12     "name": "Bob",  
13     "designation": null,  
14     "salary": 0.0,  
15     "department": "Finance",  
16     "skills": null  
17   }  
18 ]
```

GET <http://localhost:8080/employees/101>







```
Body 200 OK - 265 ms  
{ } JSON Preview Visualization  
1  {  
2    "id": 101,  
3    "name": "Alice",  
4    "designation": null,  
5    "salary": 0.0,  
6    "department": "HR",  
7    "skills": null  
8  }
```

PUT <http://localhost:8080/employees>

```
1  {  
2    "id": 101,  
3    "name": "Updated Name",  
4    "salary": 55000  
5  }  
6
```

GET <http://localhost:8080/employees/101>

Body  200 OK • 9 ms •

{ } JSON  Preview  Visualization 

```
1  {  
2    "id": 101,  
3    "name": "Updated Name",  
4    "designation": null,  
5    "salary": 55000.0,  
6    "department": null,  
7    "skills": null  
8  }
```

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:

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.example</groupId>
  <artifactId>employee-rest-xml</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>employee-rest-xml</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>
        <!-- Spring Boot Web -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- Spring Boot Data JPA -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <!-- MySQL Connector for MySQL 8.0.42 -->
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <version>8.0.33</version>
        </dependency>

        <!-- Jakarta Persistence API (for JPA annotations) -->
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.1.0</version>
        </dependency>

        <!-- Spring Boot Test (optional for unit testing) -->
        <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.datasource.url=jdbc:mysql://localhost:3306/employeeedb
spring.datasource.username=root
spring.datasource.password=$rutiSLD#07
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
server.port=8080
```

Employee.java

```
package com.example.employee.model;

import jakarta.persistence.*;

@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private String department;
    private double salary;

    // 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;
    }

    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
    }
}
```

```

    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
}

```

EmployeeDao.java

```

package com.example.employee.dao;

import com.example.employee.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeDao extends JpaRepository<Employee, Integer> {
}

```

EmployeeService.java

```

package com.example.employee.service;

import com.example.employee.dao.EmployeeDao;
import com.example.employee.model.Employee;
import jakarta.transaction.Transactional;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService {

    private final EmployeeDao employeeDao;

    public EmployeeService(EmployeeDao employeeDao) {
        this.employeeDao = employeeDao;
    }

    @Transactional
    public List<Employee> getAllEmployees() {
        return employeeDao.findAll();
    }
}

```

EmployeeController.java

```
package com.example.employee.controller;

import com.example.employee.model.Employee;
import com.example.employee.service.EmployeeService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class EmployeeController {

    private final EmployeeService employeeService;

    public EmployeeController(EmployeeService employeeService) {
        this.employeeService = employeeService;
    }

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

EmployeeRestXmlApplication.java

```
package com.example.employee;

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

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

Database Related Query

CREATE DATABASE employeedb;


```
CREATE TABLE employee (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    name VARCHAR(100),  
    department VARCHAR(100),  
    salary DOUBLE  
);
```

```
INSERT INTO employee (name, department, salary) VALUES  
( 'Alice', 'HR', 45000.00),  
( 'Bob', 'IT', 60000.00),  
( 'Charlie', 'Finance', 55000.00);
```

Output:

GET <http://localhost:8080/employees>



The screenshot shows a REST client interface with a status bar at the top indicating a '200 OK' response in 1.40 seconds. The response body is displayed in JSON format, showing an array of three employee objects. The first object represents Alice in the HR department with a salary of 45000.00. The second object represents Bob in the IT department with a salary of 60000.00. The third object represents Charlie in the Finance department with a salary of 55000.00.

```
1  [  
2      {  
3          "id": 1,  
4          "name": "Alice",  
5          "department": "HR",  
6          "salary": 45000.0  
7      },  
8      {  
9          "id": 2,  
10         "name": "Bob",  
11         "department": "IT",  
12         "salary": 60000.0  
13     },  
14     {  
15         "id": 3,  
16         "name": "Charlie",  
17         "department": "Finance",  
18         "salary": 55000.0  
19     }  
20 ]
```

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:

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.example</groupId>
  <artifactId>department-rest-api</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>department-rest-api</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.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!-- Spring Context (for XML configuration support) -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
    </dependency>

    <!-- Logging (optional, for SLF4J logger) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-logging</artifactId>
    </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <annotationProcessorPaths>
                        <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </path>
                    </annotationProcessorPaths>
                </configuration>
            </plugin>
        </plugins>
    </build>

```

```

        </plugin>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>

```

Department.java

```

package com.example.department.model;

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

    // Constructors
    public Department() {}
    public Department(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;
    }
}

```

departments.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="departmentList" class="java.util.ArrayList">
        <constructor-arg>
            <list>
                <bean class="com.example.department.model.Department">
                    <constructor-arg value="1"/>
                    <constructor-arg value="HR"/>
                </bean>
                <bean class="com.example.department.model.Department">
                    <constructor-arg value="2"/>
                    <constructor-arg value="IT"/>
                </bean>
                <bean class="com.example.department.model.Department">
                    <constructor-arg value="3"/>
                    <constructor-arg value="Finance"/>
                </bean>
            </list>
        </constructor-arg>
    </bean>

</beans>
```

DepartmentDao.java

```
package com.example.department.dao;

import com.example.department.model.Department;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class DepartmentDao {

    public static List<Department> DEPARTMENT_LIST;

    @Autowired
    public DepartmentDao(@Qualifier("departmentList") List<Department>
departments) {
        DEPARTMENT_LIST = departments;
    }
}
```

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

DepartmentService.java

```
package com.example.department.service;  
  
import com.example.department.dao.DepartmentDao;  
import com.example.department.model.Department;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class DepartmentService {  
  
    private final DepartmentDao departmentDao;  
  
    public DepartmentService(DepartmentDao departmentDao) {  
        this.departmentDao = departmentDao;  
    }  
  
    public List<Department> getAllDepartments() {  
        return departmentDao.getAllDepartments();  
    }  
}
```

DepartmentController.java

```
package com.example.department.controller;  
  
import com.example.department.model.Department;  
import com.example.department.service.DepartmentService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.*;  
  
import java.util.List;
```

```

@RestController
public class DepartmentController {

    private static final Logger logger =
LoggerFactory.getLogger(DepartmentController.class);
    private final DepartmentService departmentService;

    public DepartmentController(DepartmentService departmentService) {
        this.departmentService = departmentService;
    }

    @GetMapping("/departments")
    public List<Department> getAllDepartments() {
        logger.info("Department REST service called.");
        return departmentService.getAllDepartments();
    }
}

```

application.properties

```

spring.main.allow-bean-definition-overriding=true
logging.level.root=INFO
server.port=8080

```

DepartmentRestApiApplication.java

```

package com.example.department;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ImportResource;

@SpringBootApplication
@ImportResource("classpath:departments.xml")
public class DepartmentRestApiApplication {
    public static void main(String[] args) {
        SpringApplication.run(DepartmentRestApiApplication.class, args);
    }
}

```

Output:

GET <http://localhost:8080/departments>



The screenshot shows a web browser's developer console with the 'Body' tab selected. The status bar at the top indicates a '200 OK' response with a response time of '397 ms'. The response body is displayed as a JSON array with three objects, each representing a department. The JSON is formatted with line numbers 1 through 14 on the left side of the editor.

```
1  [  
2    {  
3      "id": 1,  
4      "name": "HR"  
5    },  
6    {  
7      "id": 2,  
8      "name": "IT"  
9    },  
10   {  
11     "id": 3,  
12     "name": "Finance"  
13   }  
14 ]
```