**Week – 04**

**Spring REST handson:**

**Hands on 1**

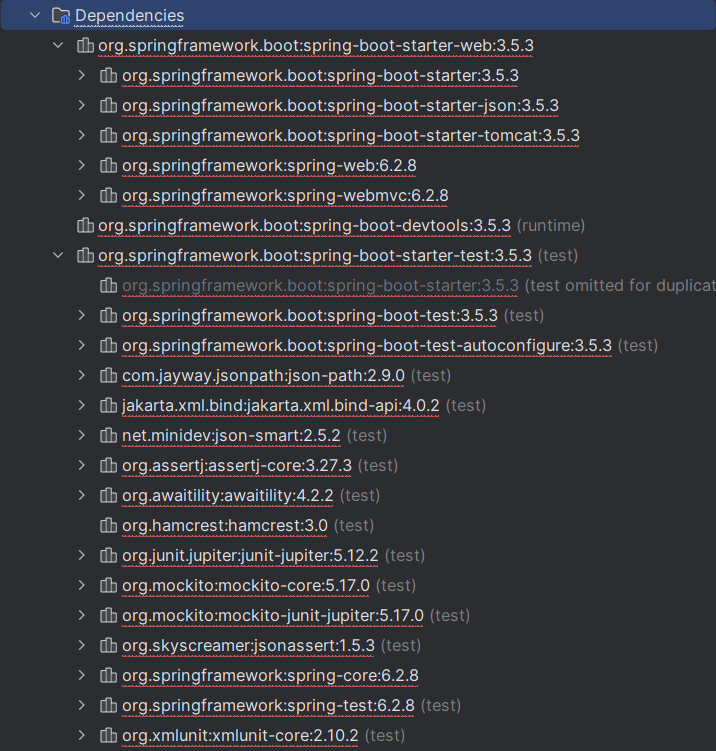
**Create a Spring Web Project using Maven**   
  
Follow steps below to create a project: 

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.

Sol.



**Hands on 4**

**Spring Core – Load Country from Spring Configuration XML**   
  
An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country. 

|  |  |
| --- | --- |
| **Code** | **Name** |
| US | United States |
| DE | Germany |
| IN | India |
| JP | Japan |

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

* Pick any one of your choice country to configure in Spring XML configuration named country.xml.
* Create a bean tag in spring configuration for country and set the property and values

    <bean id="country" class="com.cognizant.springlearn.Country">

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

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

    </bean>

* Create Country class with following aspects:
  + Instance variables for code and name
  + Implement empty parameter constructor with inclusion of debug log within the constructor with log message as “Inside Country Constructor.”
  + Generate getters and setters with inclusion of debug with relevant message within each setter and getter method.
  + Generate toString() method
* Create a method displayCountry() in SpringLearnApplication.java, which will read the country bean from spring configuration file and display the country details. ClassPathXmlApplicationContext, ApplicationContext and context.getBean(“beanId”, Country.class). Refer sample code for displayCountry() method below.

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

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

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

* Invoke displayCountry() method in main() method 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

Sol.

Pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>spring-learn</name>  
 <description>Demo project for Spring 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.spring\_learn;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class SpringLearnApplication {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(SpringLearnApplication.class);  
  
 public static void main(String[] args) {  
 *LOGGER*.debug("START");  
 *displayCountry*();  
 *LOGGER*.debug("END");  
 }  
  
 public static void displayCountry() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = (Country) context.getBean("country", Country.class);  
 *LOGGER*.debug("Country : {}", country.toString());  
 }  
}

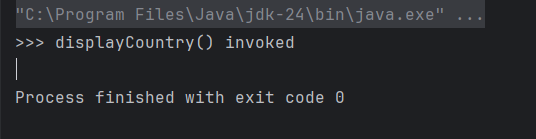
Country.java

package com.cognizant.spring\_learn;  
  
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  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="country" class="com.cognizant.spring\_learn.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
  
</beans>

Output:



**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

Sol.

Application.properties

Server.port = 8083

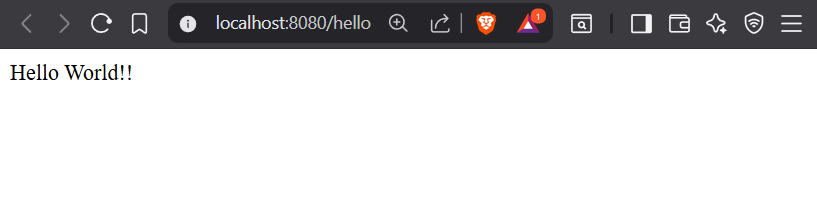
SpringLearnApplication.java

package com.cognizant.spring\_learn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringLearnApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
}

HelloController.java

package com.cognizant.spring\_learn.controller;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HelloController {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(HelloController.class);  
  
 @GetMapping("/hello")  
 public String sayHello() {  
 *LOGGER*.debug("Start: sayHello()");  
 String message = "Hello World!!";  
 *LOGGER*.debug("End: sayHello()");  
 return message;  
 }  
}

Output:



**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

Sol.

Application.properties

Server.port = 8083

Country.java

package com.cognizant.spring\_learn;  
  
public class Country {  
 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.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.Country;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class CountryController {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryController.class);  
  
 @RequestMapping("/country")  
 public Country getCountryIndia() {  
 *LOGGER*.debug("Start: getCountryIndia()");  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = context.getBean("country", Country.class);  
 *LOGGER*.debug("End: getCountryIndia()");  
 return country;  
 }  
}

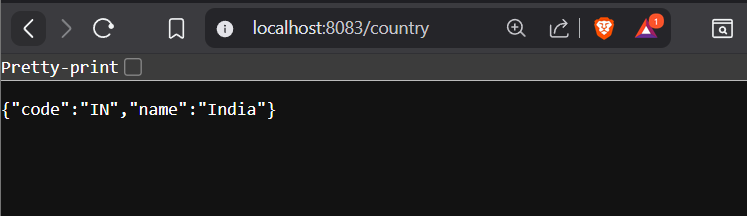
country.xml

<?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  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="country" class="com.cognizant.spring\_learn.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
 </bean>  
  
</beans>

SpringLearnApplication.java

package com.cognizant.spring\_learn;  
  
import org.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:



**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"

}

Sol.

SpringLearnApplication.java

package com.cognizant.spring\_learn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringLearnApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
}

Country.java

package com.cognizant.spring\_learn;  
  
public class Country {  
 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 + "]";  
 }  
}

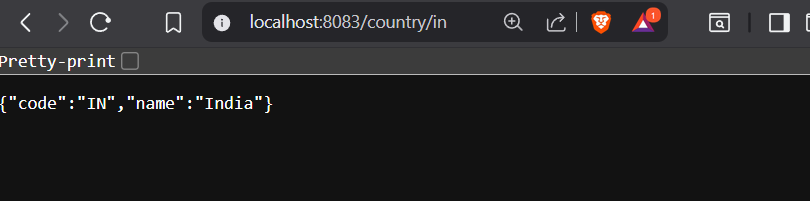
CountryService.java

package com.cognizant.spring\_learn.service;  
  
import com.cognizant.spring\_learn.Country;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 public Country getCountry(String code) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countryList = context.getBean("countryList", List.class);  
  
 return countryList.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElse(null);  
 }  
}

CountryController.java

package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.Country;  
import com.cognizant.spring\_learn.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
public class CountryController {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryController.class);  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/country/{code}")  
 public Country getCountry(@PathVariable String code) {  
 *LOGGER*.debug("Start: getCountry({})", code);  
 Country country = countryService.getCountry(code);  
 *LOGGER*.debug("End: getCountry - result: {}", country);  
 return country;  
 }  
}

Output:



**JWT Handson:**

**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.

Sol.

Application.properties

Server.port = 8090

SpringLearnApplication.java

package com.cognizant.spring\_learn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringLearnApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
}

JwtUtil.java

package com.cognizant.spring\_learn.security;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.stereotype.Component;  
  
import java.security.Key;  
import java.util.Date;  
  
@Component  
public class JwtUtil {  
  
 private final Key key = Keys.*secretKeyFor*(SignatureAlgorithm.*HS256*);  
 private final long expiration = 10 \* 60 \* 1000; // 10 minutes  
  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + expiration))  
 .signWith(key)  
 .compact();  
 }  
}

AuthController.java

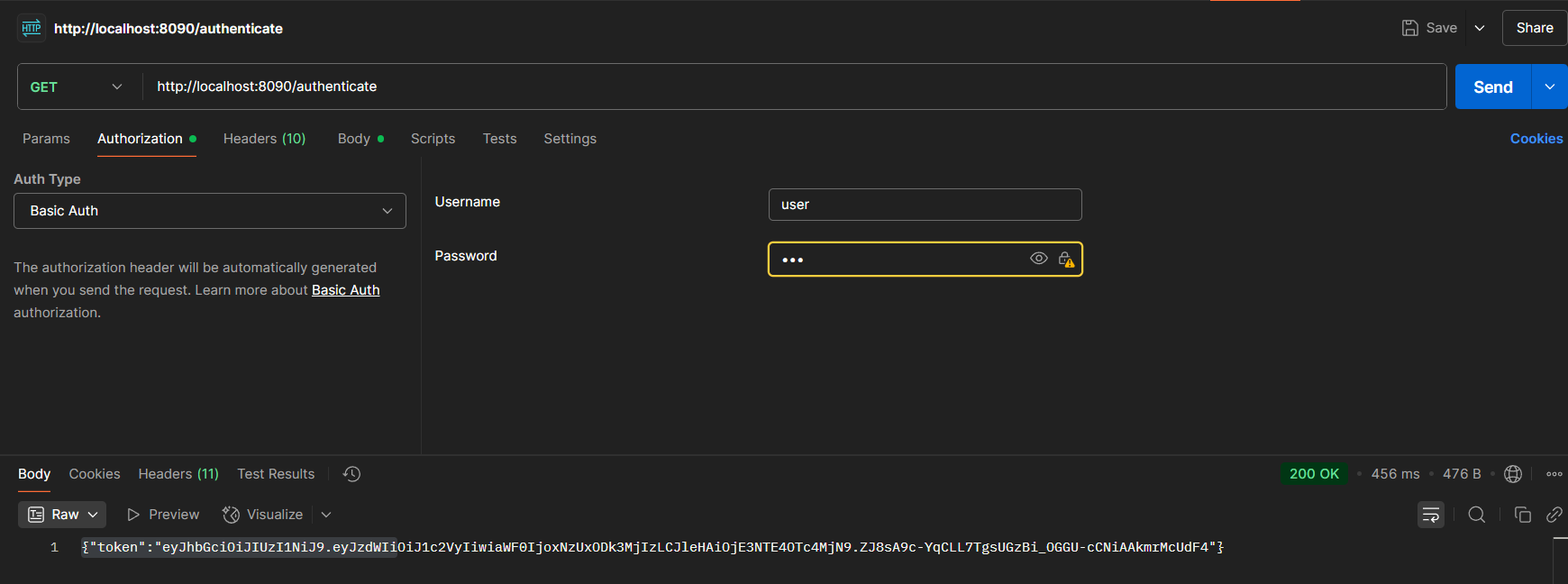
package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.security.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import java.util.Base64;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import jakarta.servlet.http.HttpServletRequest;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @GetMapping("/authenticate")  
 public ResponseEntity<?> authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization");  
  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 String credentials = new String(Base64.*getDecoder*().decode(base64Credentials));  
 String[] values = credentials.split(":", 2);  
  
 if (values.length != 2) {  
 return ResponseEntity.*status*(400).body("Invalid Basic authentication format");  
 }  
  
 String username = values[0];  
 String password = values[1];  
  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");  
 } else {  
 return ResponseEntity.*status*(401).body("Invalid credentials");  
 }  
 }  
}

SecurityConfig.java

package com.cognizant.spring\_learn.security;  
  
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;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 );  
 return http.build();  
 }  
}

Output:





**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.

Sol.

Employee.java

package com.example.employee\_api.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
 private String department;  
  
 public String getName(){  
 return this.name;  
 }  
  
 public String getDepartment(){  
 return this.department;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
}

EmployeeRepository.java

package com.example.employee\_api.repository;  
import com.example.employee\_api.model.Employee;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Long> {  
}

EmployeeService.java

package com.example.employee\_api.service;  
  
import com.example.employee\_api.model.Employee;  
import com.example.employee\_api.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeRepository repository;  
  
 public List<Employee> getAll() {  
 return repository.findAll();  
 }  
  
 public Optional<Employee> getById(Long id) {  
 return repository.findById(id);  
 }  
  
 public Employee create(Employee employee) {  
 return repository.save(employee);  
 }  
  
 public Employee update(Long id, Employee newEmp) {  
 return repository.findById(id).map(emp -> {  
 emp.setName(newEmp.getName());  
 emp.setDepartment(newEmp.getDepartment());  
 return repository.save(emp);  
 }).orElseThrow(() -> new RuntimeException("Employee not found"));  
 }  
  
 public void delete(Long id) {  
 repository.deleteById(id);  
 }  
}

EmployeeController.java

package com.example.employee\_api.controller;  
import com.example.employee\_api.model.Employee;  
import com.example.employee\_api.service.EmployeeService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/employees")  
@CrossOrigin(origins = "\*")  
public class EmployeeController {  
  
 @Autowired  
 private EmployeeService service;  
  
 @GetMapping  
 public List<Employee> getAllEmployees() {  
 return service.getAll();  
 }  
  
 @GetMapping("/{id}")  
 public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {  
 return service.getById(id)  
 .map(ResponseEntity::*ok*)  
 .orElse(ResponseEntity.*notFound*().build());  
 }  
  
 @PostMapping  
 public ResponseEntity<Employee> createEmployee(@RequestBody Employee employee) {  
 return ResponseEntity.*ok*(service.create(employee));  
 }  
  
 @PutMapping("/{id}")  
 public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employee) {  
 return ResponseEntity.*ok*(service.update(id, employee));  
 }  
  
 @DeleteMapping("/{id}")  
 public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {  
 service.delete(id);  
 return ResponseEntity.*noContent*().build();  
 }  
}

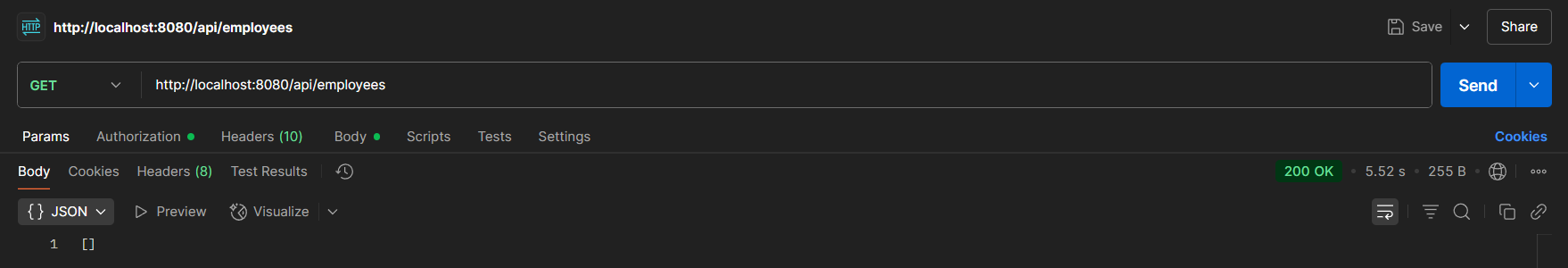
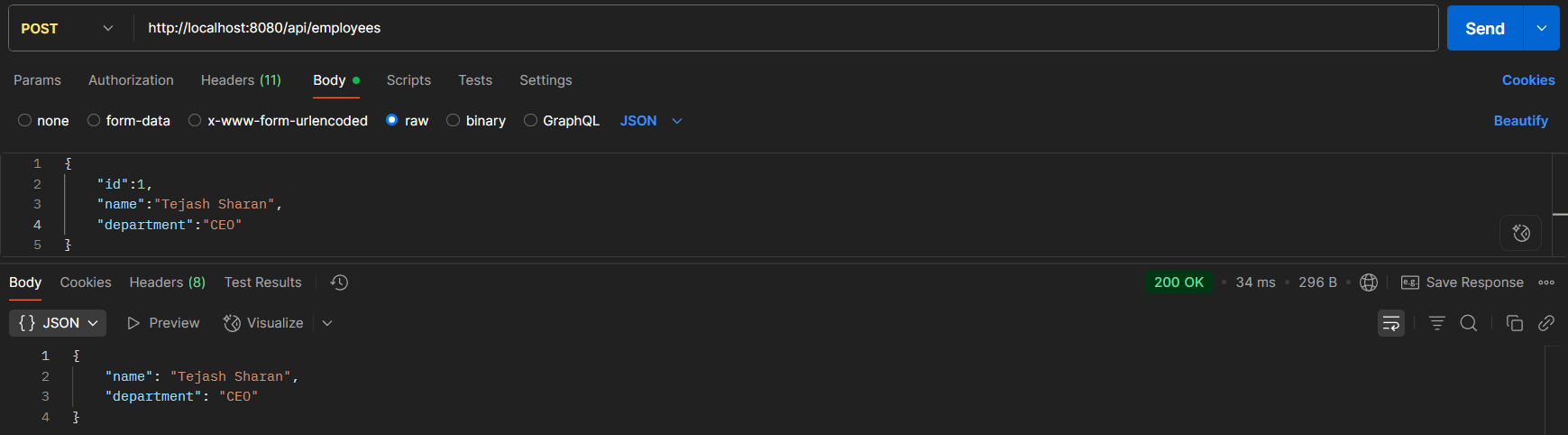
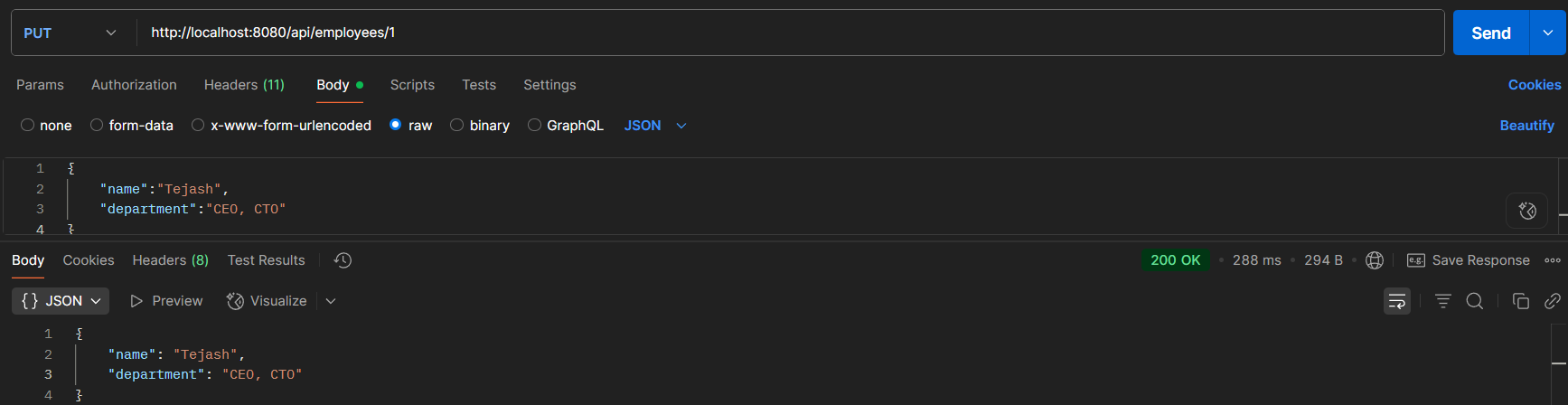
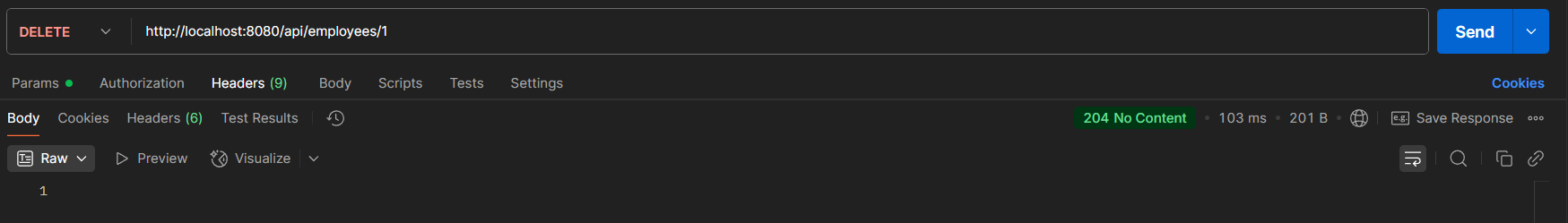
application.properties

spring.application.name=employee-api  
  
spring.datasource.url=jdbc:h2:mem:employeedb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.h2.console.enabled=true  
spring.jpa.hibernate.ddl-auto=update

EmployeeApiApplication.java

package com.example.employee\_api;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class EmployeeApiApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EmployeeApiApplication.class, args);  
 }  
  
}

output:

**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

Sol.

Employee.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="skill1" class="java.lang.String">  
 <constructor-arg value="Java"/>  
 </bean>  
 <bean id="skill2" class="java.lang.String">  
 <constructor-arg value="Angular"/>  
 </bean>  
 <bean id="skill3" class="java.lang.String">  
 <constructor-arg value="SQL"/>  
 </bean>  
  
 <bean id="department1" class="java.lang.String">  
 <constructor-arg value="Engineering"/>  
 </bean>  
 <bean id="department2" class="java.lang.String">  
 <constructor-arg value="HR"/>  
 </bean>  
 <bean id="department3" class="java.lang.String">  
 <constructor-arg value="Sales"/>  
 </bean>  
  
 <bean id="employee1" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="1"/>  
 <property name="name" value="John Doe"/>  
 <property name="department" ref="department1"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill1"/>  
 <ref bean="skill2"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employee2" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="2"/>  
 <property name="name" value="Jane Smith"/>  
 <property name="department" ref="department2"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill2"/>  
 <ref bean="skill3"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employee3" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="3"/>  
 <property name="name" value="Mike Johnson"/>  
 <property name="department" ref="department3"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill1"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employee4" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="4"/>  
 <property name="name" value="Emily Brown"/>  
 <property name="department" ref="department1"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill3"/>  
 <ref bean="skill1"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employeeList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <ref bean="employee1"/>  
 <ref bean="employee2"/>  
 <ref bean="employee3"/>  
 <ref bean="employee4"/>  
 </list>  
 </constructor-arg>  
 </bean>  
  
 <bean id="employeeDao" class="com.example.employee\_api.dao.EmployeeDao">  
 <constructor-arg ref="employeeList"/>  
 </bean>  
  
</beans>

Employee.java

package com.example.employee\_api.model;  
  
import java.util.List;  
  
public class Employee {  
 private int id;  
 private String name;  
 private String department;  
 private List<String> skills;  
  
 public int getId(){  
 return this.id;  
 }  
  
 public void setId(int id){  
 this.id = id;  
 }  
  
 public String getName(){  
 return this.name;  
 }  
  
 public String getDepartment(){  
 return this.department;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 public List<String> getSkills() {  
 return this.skills;  
 }  
  
 public void setSkills(List<String> skills) {  
 this.skills = skills;  
 }  
}

EmployeeDao.java

package com.example.employee\_api.dao;  
  
  
import com.example.employee\_api.model.Employee;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class EmployeeDao {  
  
 public static List<Employee> *EMPLOYEE\_LIST* = new ArrayList<>();  
  
 public EmployeeDao(List<Employee> employeeListFromXml) {  
 *EMPLOYEE\_LIST* = employeeListFromXml;  
 }  
  
 public List<Employee> getAllEmployees() {  
 return *EMPLOYEE\_LIST*;  
 }  
}

AppConfig.java

package com.example.employee\_api.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.ImportResource;  
  
@Configuration  
@ImportResource("classpath:employee.xml")   
public class AppConfig {  
}

**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.

Sol.

Employee.java

package com.example.employee\_api.model;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import java.util.List;  
  
@Entity  
public class Employee {  
 @Id  
 private int id;  
 private String name;  
 private String department;  
 private List<String> skills;  
  
 public int getId(){  
 return this.id;  
 }  
  
 public void setId(int id){  
 this.id = id;  
 }  
  
 public String getName(){  
 return this.name;  
 }  
  
 public String getDepartment(){  
 return this.department;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 public List<String> getSkills() {  
 return this.skills;  
 }  
  
 public void setSkills(List<String> skills) {  
 this.skills = skills;  
 }  
}

EmployeeDao.java

package com.example.employee\_api.dao;  
  
import com.example.employee\_api.model.Employee;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class EmployeeDao {  
  
 public static List<Employee> *EMPLOYEE\_LIST* = new ArrayList<>();  
  
 public EmployeeDao(List<Employee> employeeListFromXml) {  
 *EMPLOYEE\_LIST* = employeeListFromXml;  
 }  
  
 public List<Employee> getAllEmployees() {  
 return *EMPLOYEE\_LIST*;  
 }  
}

EmployeeController.java

package com.example.employee\_api.controller;  
  
import com.example.employee\_api.model.Employee;  
import com.example.employee\_api.service.EmployeeService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api")  
public class EmployeeController {  
  
 @Autowired  
 private EmployeeService employeeService;  
  
 @GetMapping("/employees")  
 public List<Employee> getAllEmployees() {  
 return employeeService.getAllEmployees();  
 }  
}

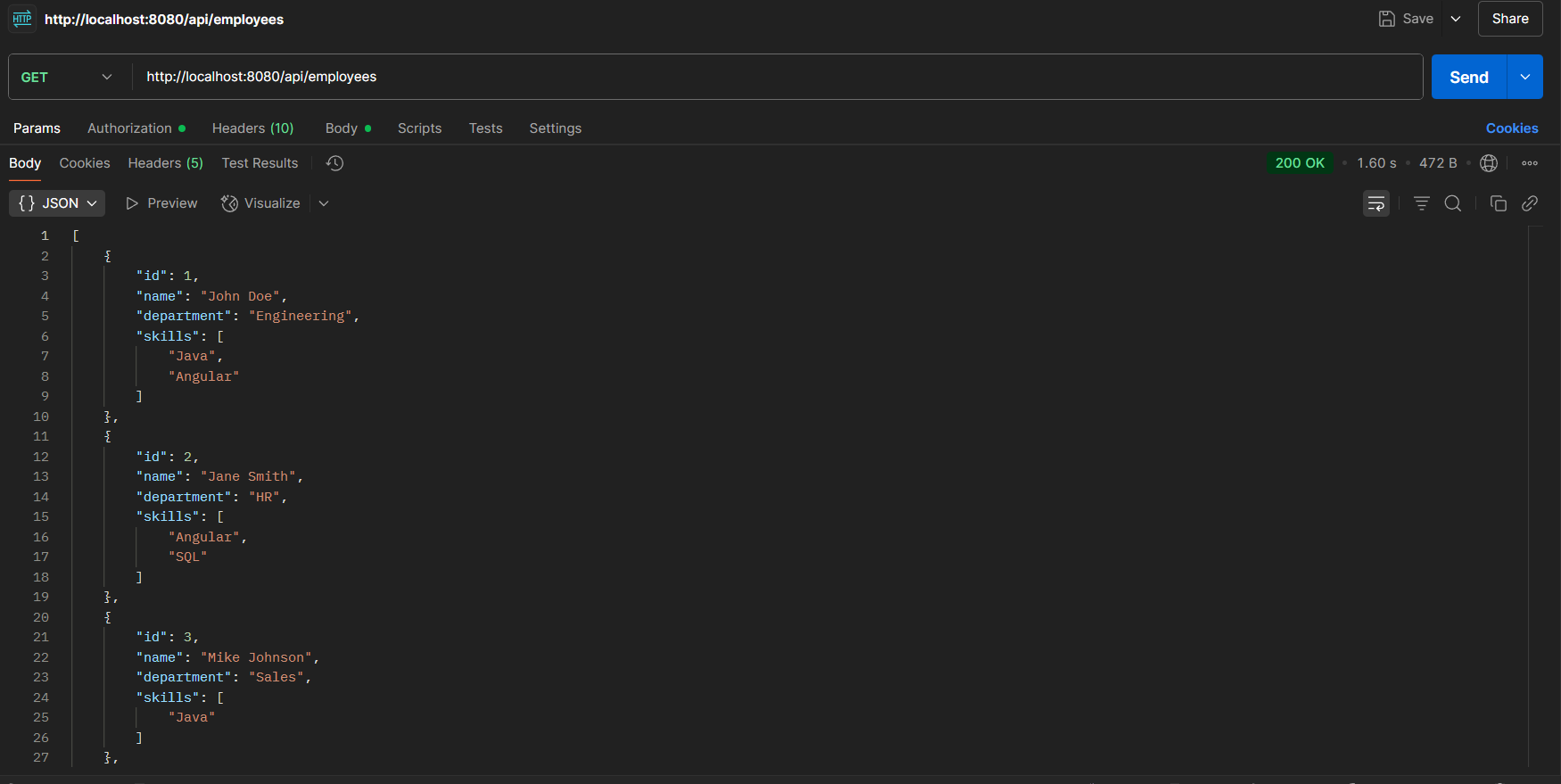
EmployeeService.java

package com.example.employee\_api.service;  
  
import com.example.employee\_api.dao.EmployeeDao;  
import com.example.employee\_api.model.Employee;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
  
@Service  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeDao employeeDao;  
  
 @Transactional  
 public List<Employee> getAllEmployees() {  
 return employeeDao.getAllEmployees();  
 }  
}

AppConfig.java

package com.example.employee\_api.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.ImportResource;  
  
@Configuration  
@ImportResource("classpath:employee.xml")  
public class AppConfig {  
}

Output:



**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.

Sol.

Employee.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="skill1" class="java.lang.String">  
 <constructor-arg value="Java"/>  
 </bean>  
 <bean id="skill2" class="java.lang.String">  
 <constructor-arg value="Angular"/>  
 </bean>  
 <bean id="skill3" class="java.lang.String">  
 <constructor-arg value="SQL"/>  
 </bean>  
  
 <bean id="department1" class="java.lang.String">  
 <constructor-arg value="Engineering"/>  
 </bean>  
 <bean id="department2" class="java.lang.String">  
 <constructor-arg value="HR"/>  
 </bean>  
 <bean id="department3" class="java.lang.String">  
 <constructor-arg value="Sales"/>  
 </bean>  
  
 <bean id="departmentList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <ref bean="department1"/>  
 <ref bean="department2"/>  
 <ref bean="department3"/>  
 </list>  
 </constructor-arg>  
 </bean>  
  
 <bean id="departmentDao" class="com.example.employee\_api.dao.DepartmentDao">  
 <constructor-arg ref="departmentList"/>  
 </bean>  
  
  
 <bean id="employee1" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="1"/>  
 <property name="name" value="John Doe"/>  
 <property name="department" ref="department1"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill1"/>  
 <ref bean="skill2"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employee2" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="2"/>  
 <property name="name" value="Jane Smith"/>  
 <property name="department" ref="department2"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill2"/>  
 <ref bean="skill3"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employee3" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="3"/>  
 <property name="name" value="Mike Johnson"/>  
 <property name="department" ref="department3"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill1"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employee4" class="com.example.employee\_api.model.Employee">  
 <property name="id" value="4"/>  
 <property name="name" value="Emily Brown"/>  
 <property name="department" ref="department1"/>  
 <property name="skills">  
 <list>  
 <ref bean="skill3"/>  
 <ref bean="skill1"/>  
 </list>  
 </property>  
 </bean>  
  
 <bean id="employeeList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <ref bean="employee1"/>  
 <ref bean="employee2"/>  
 <ref bean="employee3"/>  
 <ref bean="employee4"/>  
 </list>  
 </constructor-arg>  
 </bean>  
  
 <bean id="employeeDao" class="com.example.employee\_api.dao.EmployeeDao">  
 <constructor-arg ref="employeeList"/>  
 </bean>  
  
</beans>

DepartmentDao.java

package com.example.employee\_api.dao;  
  
import java.util.List;  
  
public class DepartmentDao {  
 public static List<String> *DEPARTMENT\_LIST*;  
  
 public DepartmentDao(List<String> departmentsFromXml) {  
 *DEPARTMENT\_LIST* = departmentsFromXml;  
 }  
  
 public List<String> getAllDepartments() {  
 return *DEPARTMENT\_LIST*;  
 }  
}

DepartmentController.java

package com.example.employee\_api.controller;  
  
import com.example.employee\_api.service.DepartmentService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api")  
public class DepartmentController {  
  
 private final DepartmentService departmentService;  
  
 @Autowired  
 public DepartmentController(DepartmentService departmentService) {  
 this.departmentService = departmentService;  
 }  
  
 @GetMapping("/departments")  
 public List<String> getAllDepartments() {  
 System.*out*.println("GET /departments called");  
 return departmentService.getAllDepartments();  
 }  
}

DepartmentService.java

package com.example.employee\_api.service;  
  
import com.example.employee\_api.dao.DepartmentDao;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
  
@Service  
public class DepartmentService {  
  
 private final DepartmentDao departmentDao;  
  
 @Autowired  
 public DepartmentService(DepartmentDao departmentDao) {  
 this.departmentDao = departmentDao;  
 }  
  
 @Transactional  
 public List<String> getAllDepartments() {  
 return departmentDao.getAllDepartments();  
 }  
}

Output:

