**Objectives**

* Demonstrate integration of RESTful Web Service of type GET and test the service using postman.
  + REST Web Service architecture with Controller, Service and Dao, service methods.

NOTE: There is no Quiz for this session

# Cognizant Digital Nurture 4.0 Deep Skilling

## 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.

**Solution :**

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 to be changed to get the data from a RESTful Web Service developed in Spring. The following are the high-level activities that need to be done to accomplish this:

1. **Create static employee list data using Spring XML configuration**
   * Define a list of Employee beans in an XML config (e.g., employee.xml)
2. **Create a REST Service that reads data from XML configuration and returns it**
   * Define a @RestController in Spring that reads the list of employees from the XML file and returns it via a GET endpoint
3. **Make changes in Angular component to consume the created REST Service**
   * Modify the Angular EmployeeComponent to use HttpClient to make a GET request to the Spring REST endpoint and bind the response to the UI

Once the above activities are completed:

* Clicking on the **Edit** button against each employee should display the **Edit Employee** form with values retrieved from RESTful Web Service
* This will also involve similar activities (like XML definition, REST endpoint, Angular HTTP call) for editing employee data

**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

**Solution :**

### Update employee.xml in src/main/resources:

* Create one or two more **departments**
* Create four **employee** beans
* Reuse existing **skills**
* Create an ArrayList<Employee> containing all four employees

**Sample XML:**

<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">

<!-- Department Beans -->

<bean id="dep1" class="com.cognizant.model.Department">

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

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

</bean>

<bean id="dep2" class="com.cognizant.model.Department">

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

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

</bean>

<!-- Employee Beans -->

<bean id="emp1" class="com.cognizant.model.Employee">

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

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

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

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

<property name="department" ref="dep1" />

</bean>

<!-- Define more emp2, emp3, emp4 similarly -->

<!-- Employee List -->

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

<constructor-arg>

<list>

<ref bean="emp1"/>

<ref bean="emp2"/>

<ref bean="emp3"/>

<ref bean="emp4"/>

</list>

</constructor-arg>

</bean>

</beans>

### Update EmployeeDao Class:

@Component

public class EmployeeDao {

private static List<Employee> EMPLOYEE\_LIST;

@Autowired

public EmployeeDao(ApplicationContext context) {

EMPLOYEE\_LIST = context.getBean("employeeList", ArrayList.class);

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

**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.

**Solution :**

#### Update EmployeeService Class:

* Change annotation from @Component to @Service
* Create method getAllEmployees() that returns list of employees
* Annotate the method with @Transactional

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

#### Update EmployeeController Class:

* Add a method getAllEmployees()
* Use @GetMapping("/employees")
* Call employeeService.getAllEmployees() and return result

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

}

**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.

**Solution :**

Create a new REST service to get all the departments.

#### 1. Update DepartmentDao:

* Create a static variable DEPARTMENT\_LIST of type ArrayList<Department>
* In the constructor, load the department list from Spring XML configuration

java

CopyEdit

@Component

public class DepartmentDao {

private static List<Department> DEPARTMENT\_LIST;

@Autowired

public DepartmentDao(ApplicationContext context) {

DEPARTMENT\_LIST = context.getBean("departmentList", ArrayList.class);

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

#### 2. Create DepartmentService:

* Annotate with @Service
* Autowire DepartmentDao
* Create method getAllDepartments() with @Transactional annotation

java

CopyEdit

@Service

public class DepartmentService {

@Autowired

private DepartmentDao departmentDao;

@Transactional

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

#### 3. Create DepartmentController:

* Annotate with @RestController
* Autowire DepartmentService
* Create @GetMapping("/departments") method named getAllDepartments()

java

CopyEdit

@RestController

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@GetMapping("/departments")

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

}

#### 4. Test the Service:

* Run the Spring Boot application
* Use Postman to test:

bash

CopyEdit

GET http://localhost:8083/departments

* Confirm that the response returns a JSON list of departments
* Check the application logs to verify that the department REST service is being triggered