#### [Consuming Spring RESTful Web Service in Angular](javascript:void(0);)

**Objectives**

* Demonstrate integration of RESTful Web Service of type GET with Angular front end
  + REST Web Service architecture with Controller, Service and Dao, service methods, invoking service using http client in angular service, cross origin
    - Cross Origin reference - https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html

NOTE: There is no Quiz for this session

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

* Create static employee list data using spring xml configuration

* Create a REST Service that reads data from xml configuration and returns it

* Make changes in angular component to consume the created REST Service

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

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

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

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

**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 chrome browser or curl or postman.

**Integrate Employee List angular component with Spring REST**   
  
Implement steps below to make the employee list component get data from REST service:

* Open the angular-learning project in Visual Studio Code
* Open file employee.service.ts
* Ensure that necessary imports are in place.

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs';

* Define the constructor and inject the HttpClient

constructor(private httpClient: HttpClient) { }

* Define a new attribute employeeApiUrl which will contain the URL for employees

private employeeApiUrl = 'http://localhost:8090/employees';

* Comment the existing code in getAllEmployees() method
* Define the getAllEmployees() method as defined below

  public getAllEmployees(): Observable<any[]> {

    return this.\_httpClient.get(this.employeeApiUrl);

  }

* Open edit-employee-template.component.ts and incorporate following changes:
  + Inject EmployeeService in constructor
  + Include following code in ngOnInit() method, which set the employee list for the component to be loaded dynamically from the service.

this.employeeService.getAllEmployees().subscribe(data => {

  this.employees = data;

});

* Execution of this code will fail. In browser console, one can see the cross-origin error.

Explain on the following aspects:

* The need to have Observable and Subscribe is to handle the asynchronous nature of the service call.
* The importance of cross-origin and why this error happens.

**Implement REST Service with Cross Origin support**  
  
To resolve the cross origin issue in the previous hands on, follow steps below:

* Open EmployeeController and define the below annotation at the class level

@CrossOrigin("http://localhost:4200")

* Try executing the angular application and see whether employee list is getting loaded

**Defining cross origin as global configuration  
​​​​​​​**

* The cross origin definition can also be defined at application level. Include a new class in com.fis.spring-learn package. This is a new way spring configuration can be included as Java source files:

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class WebConfig implements WebMvcConfigurer {

    @Override

    public void addCorsMappings(CorsRegistry registry) {

        registry.addMapping("/\*\*").allowedMethods("GET").allowedOrigins("http://localhost:4200");

    }

}

* Comment / remove @CrossOrigin annotation defined earlier.
* To accept all websites and all HTTP methods it can also be defined as below:

registry.addMapping("/\*\*").allowedMethods("\*").allowedOrigins("\*");

**Integrate display of Edit Employee angular component with Spring REST**   
  
On clicking Edit in Employee List screen, we need to retrieve the specific employee data to populate the form fields in Edit Employee component.  
  
To get a single employee detail a REST Service needs to be created.  
  
Follow steps below to accomplish this activity:

* Incorporate the following as part of REST Service:
  + Create new method getEmployee(int id) in EmployeeDao. This method should find the matching employee from EMPLOYEE\_LIST static variable and return the matching Employee.
  + Create new method getEmployee(int id) in EmployeeService, which invokes the EmployeeDao.getEmployee() method.
  + Create new method getEmployee(int id) in EmployeeController with URL "/employees/{id}", which invokes the EmployeeService.getEmployee() method.
  + Test the method using chrome browser or curl or postman.
* Changes in angular application:
  + Modify EmployeeService.getEmployee(id : number) method
    - Invoke the REST service by concatenating api URL with "/" and id.
  + In edit employee template component ngOnInit() method comment the existing hard coding and implement the following steps:
    - Get the employee id from router parameter
    - Invoke EmployeeService.getEmployee() passing the id obtained from router parameter

After making above changes check if the edit employee page form is populated with respective data.  
  
Try making above changes in edit employee reactive form. Changes needs to be done in ngOnInit() of the edit employee reactive component.

**Create REST service for department**

Create a new service to get all the departments to populate the department drop down in the Edit Employee screen of angular application.

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
* Make following changes in angular application
  + Create a new service for department with url defintion, httpClient injection and method to get all departments
  + Inject the department service in edit employee component
  + In ngOnInit() of Edit Employee Reactive Form component, invoke getAllDepartments() of department service and assign it to department array property.
  + Modify the department definition to FormControl instead of FormGroup. Refer code below.

'department': new FormControl('')

* Also ensure that the department in HTML template is defined as specified below:

    <div>

        <h3>Department</h3>

        <select formControlName="department">

            <option \*ngFor="let department of departments"

                    [ngValue]="department"

                    [selected]="department.id === employee.department.id">

                {{department.name}}

            </option>

        </select>

    </div>

* Run the angular application and check if the department is populated and the respective department is selected based on the id set.
* Also verify if department REST service is called by looking into the logs.