**Date : 16-12-2021**

**Demo-Session-1**

---------------------------------------------------------------------------------------------------------------------

**Level-III : Spring Cloud (Microservices) + Angular UI**

----------------------------------------------------------------------

Pre-Req:

=> Microservices (Spring Boot and Spring Cloud)

=> Typescript and Angular Basics (3 weekends)

Level-I : Spring F/w [ Not conducting ]

Level-II : Spring Boot [ MVC, Thymeleaf, Data JPA, Security...]

Level-III : Spring Cloud (Microservices) + Angular UI (current)

Microservices : Implement every service of a project as a Application.

Project = Collection of Modules

Application = Collection of Services

[Monolithic applications]

1 Gmail App = Services( Inbox, User, Sent, Drafts ) = finally 1 Project

[Microservices Applications]

1 Service = 1 Application

\*) Eureka Server :- It is a register. It holds all MS# details.

-------Eureka Server Register----------------------------------

ServiceId InstanceId HOST PORT ....

PAYMENT PAYMENT-5410 192.168.10.1 8080

USER USER-9856 192.168.10.2 9696

...etc

\*) Config Server: Multiple MS# can have common key=val, common application.properties

\*) Admin Server : Visualize all MS# production services

Started?, Config?, Prop?, Health, Beans?...etc

\*) Distributed Tracing : Zipkin and Sleuth [ Debugging MS# execution ]

\*) Message Queues : Continues Data flow from outside project.

\*) ELK : Elastic Search, Logstash and Kibana. All Log files data at one place

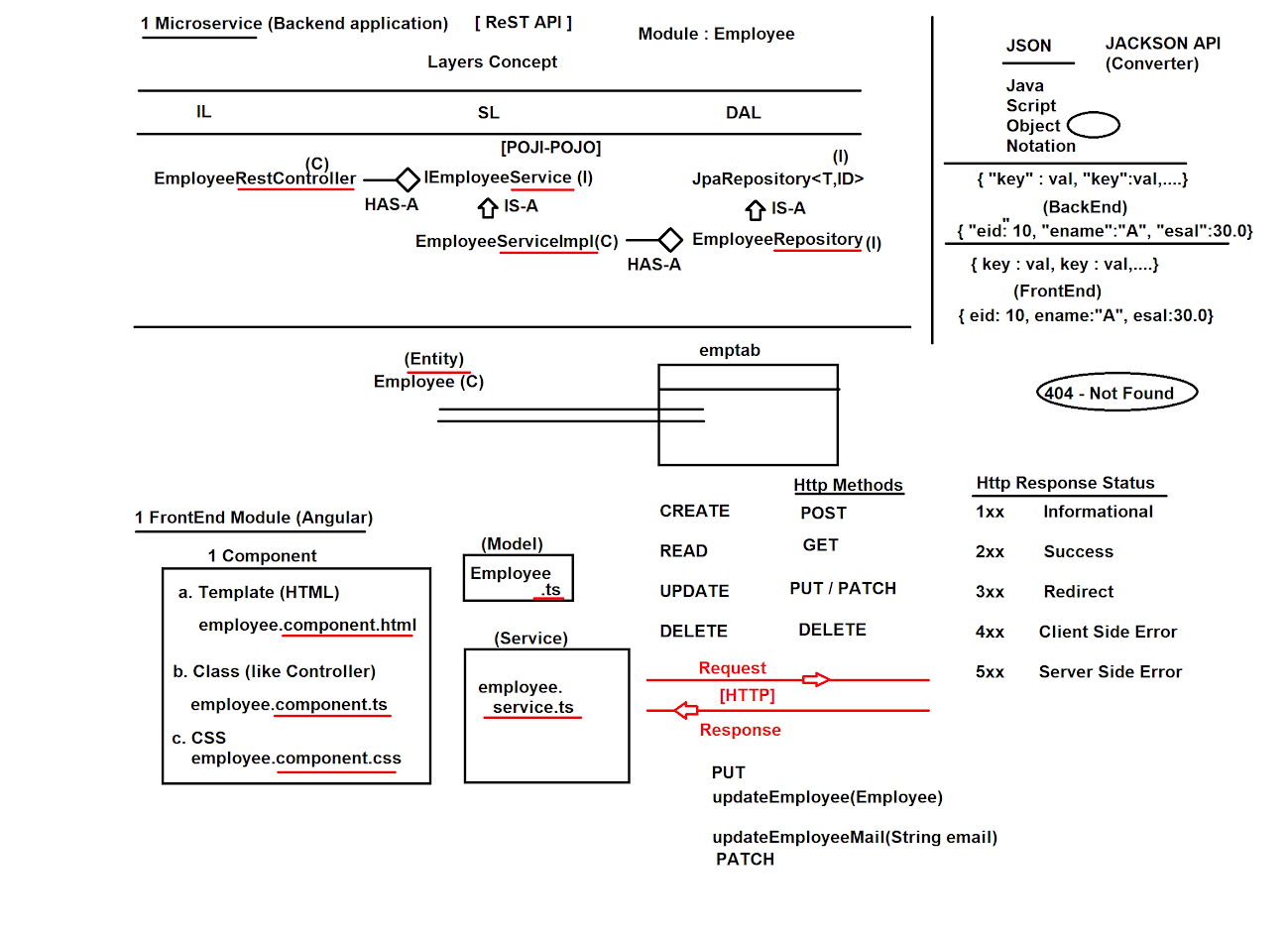
\*) API Gateway:- This is common entry and exit point. (Common IP/PORT)

Every MS# contains one IP address. We can not share all IPs to Client/UI

Duration: 4months

1-Mini project by using all concepts. (Dummy Employee Operations)

1-Main Project by using all concepts. (Supply Chain Domain/Stock Market)



---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

All Old Recorded Sessions : Tools & Spring Boot Extra Topics

-----------------------------------------------------------------------------

**Spring Boot Demos**[**https://www.youtube.com/watch?v=EA43S5R8LSc&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr**](https://www.youtube.com/watch?v=EA43S5R8LSc&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr)[**https://www.youtube.com/watch?v=ynxeJ8nP\_HM&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr**](https://www.youtube.com/watch?v=ynxeJ8nP_HM&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr) **Maven:**[**https://www.youtube.com/watch?v=9eJAvApRAI0**](https://www.youtube.com/watch?v=9eJAvApRAI0)[**https://www.youtube.com/watch?v=jOykBjSl1X0**](https://www.youtube.com/watch?v=jOykBjSl1X0)[**https://www.youtube.com/watch?v=iLwuhbnFPqo**](https://www.youtube.com/watch?v=iLwuhbnFPqo) **Gradle:**[**https://www.youtube.com/watch?v=xQQ982rgP9c**](https://www.youtube.com/watch?v=xQQ982rgP9c)[**https://www.youtube.com/watch?v=1B3KXwZYThI**](https://www.youtube.com/watch?v=1B3KXwZYThI) **AGILE**[**https://www.youtube.com/watch?v=MoxspPivr\_Q**](https://www.youtube.com/watch?v=MoxspPivr_Q) **Debug:**[**https://www.youtube.com/watch?v=HwwF4pvYWws**](https://www.youtube.com/watch?v=HwwF4pvYWws) **Log4J**[**https://www.youtube.com/watch?v=mblGoKU1aKo**](https://www.youtube.com/watch?v=mblGoKU1aKo)[**https://www.youtube.com/watch?v=5oLfHiP\_iJc**](https://www.youtube.com/watch?v=5oLfHiP_iJc) **Spring Core #1**[**https://www.youtube.com/watch?v=xusQhpQuODk&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr**](https://www.youtube.com/watch?v=xusQhpQuODk&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr) **Spring Core #2 (Autowired)**[**https://www.youtube.com/watch?v=-FlszP92JVM&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr**](https://www.youtube.com/watch?v=-FlszP92JVM&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr) **Spring Boot Runners**[**https://www.youtube.com/watch?v=8P5x4DH2WHA**](https://www.youtube.com/watch?v=8P5x4DH2WHA) **Lombok:**[**https://www.youtube.com/watch?v=kSceQH4fFi4**](https://www.youtube.com/watch?v=kSceQH4fFi4) **Spring Boot JDBC**[**https://www.youtube.com/watch?v=9wCpMk1YXnc**](https://www.youtube.com/watch?v=9wCpMk1YXnc)[**https://www.youtube.com/watch?v=-a2M-FN88bA**](https://www.youtube.com/watch?v=-a2M-FN88bA) **Spring Boot Web MVC**[**https://www.youtube.com/watch?v=O8MtjB0Ru0E**](https://www.youtube.com/watch?v=O8MtjB0Ru0E) **Spring Boot Error and Exception Handling**[**https://www.youtube.com/watch?v=AFq9eK2OoGU**](https://www.youtube.com/watch?v=AFq9eK2OoGU)[**https://www.youtube.com/watch?v=tBVAybXMKzY**](https://www.youtube.com/watch?v=tBVAybXMKzY)[**https://www.youtube.com/watch?v=M-LRfrYHWrk**](https://www.youtube.com/watch?v=M-LRfrYHWrk) **Spring Boot Batch Processing**[**https://www.youtube.com/watch?v=5dFhc5WH33c**](https://www.youtube.com/watch?v=5dFhc5WH33c)[**https://www.youtube.com/watch?v=EpxNRGcriuA**](https://www.youtube.com/watch?v=EpxNRGcriuA) **Spring Boot Multiple Databases**[**https://www.youtube.com/watch?v=nzszxQbQ5WU&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr**](https://www.youtube.com/watch?v=nzszxQbQ5WU&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr) **Spring AOP**[**https://www.youtube.com/watch?v=x2f4NzNCkCI**](https://www.youtube.com/watch?v=x2f4NzNCkCI)[**https://www.youtube.com/watch?v=8DG-J8\_WfUE**](https://www.youtube.com/watch?v=8DG-J8_WfUE) **Redis Cache:**[**https://www.youtube.com/watch?v=HBmlNMGh9O0**](https://www.youtube.com/watch?v=HBmlNMGh9O0)[**https://www.youtube.com/watch?v=IwYEdZOmY6g**](https://www.youtube.com/watch?v=IwYEdZOmY6g) **Spring Boot Captcha#1**[**https://www.youtube.com/watch?v=P\_zqWKvOi54**](https://www.youtube.com/watch?v=P_zqWKvOi54) **Spring Boot Captcha#2**[**https://www.youtube.com/watch?v=fcRs5pgtJYQ**](https://www.youtube.com/watch?v=fcRs5pgtJYQ) **Git:**[**https://www.youtube.com/watch?v=T2UHpsxJ-2o**](https://www.youtube.com/watch?v=T2UHpsxJ-2o)[**https://www.youtube.com/watch?v=38UGVeXuj3Q**](https://www.youtube.com/watch?v=38UGVeXuj3Q) **Webservices:**[**https://www.youtube.com/watch?v=fvG7FEU1Rt8**](https://www.youtube.com/watch?v=fvG7FEU1Rt8)[**https://www.youtube.com/watch?v=tX1Wtv8Gdpo**](https://www.youtube.com/watch?v=tX1Wtv8Gdpo) **Spring Boot + Angular**<https://www.youtube.com/watch?v=LE3KvvhkUkM>  
<https://www.youtube.com/watch?v=Do87D4TJ9M8>  
<https://www.youtube.com/watch?v=gkp-Mhxuxjo>  
<https://www.youtube.com/watch?v=lpPiM5MIAoQ>  
<https://www.youtube.com/watch?v=Iax4vA4MEPU>

**Spring Boot + ReactJS**[**https://www.youtube.com/watch?v=kGXgibT2vyw**](https://www.youtube.com/watch?v=kGXgibT2vyw)[**https://www.youtube.com/watch?v=-UJByPVdLXo**](https://www.youtube.com/watch?v=-UJByPVdLXo)[**https://www.youtube.com/watch?v=pfRWCTTYDdM**](https://www.youtube.com/watch?v=pfRWCTTYDdM) **Log4J:**[**https://www.youtube.com/watch?v=mblGoKU1aKo**](https://www.youtube.com/watch?v=mblGoKU1aKo)[**https://www.youtube.com/watch?v=5oLfHiP\_iJc**](https://www.youtube.com/watch?v=5oLfHiP_iJc) **Spring Boot Data REST:**[**https://www.youtube.com/watch?v=uptKDrR6bLg**](https://www.youtube.com/watch?v=uptKDrR6bLg) **JUnit:**[**https://www.youtube.com/watch?v=PT9WQ\_Rz1ew**](https://www.youtube.com/watch?v=PT9WQ_Rz1ew)[**https://www.youtube.com/watch?v=Rue28g3reRI**](https://www.youtube.com/watch?v=Rue28g3reRI) **Mockito Overview:**[**https://youtu.be/6hmx--O8PYE**](https://youtu.be/6hmx--O8PYE)[**https://youtu.be/pFLFKrCknB0**](https://youtu.be/pFLFKrCknB0)[**https://youtu.be/GgWuE-k1\_nI**](https://youtu.be/GgWuE-k1_nI)[**http://youtu.be/tWdqtEXE47A**](http://youtu.be/tWdqtEXE47A) **Spring Boot UnitTest code**[**https://github.com/javabyraghu/SpringBoot2UnitTestApp**](https://github.com/javabyraghu/SpringBoot2UnitTestApp) **Apache Camel**[**https://www.youtube.com/watch?v=34TK-z9hw1c**](https://www.youtube.com/watch?v=34TK-z9hw1c)[**https://www.youtube.com/watch?v=UoUtPjsfjfU**](https://www.youtube.com/watch?v=UoUtPjsfjfU) **ELK**[**https://www.youtube.com/watch?v=uSYExRWbC9Y**](https://www.youtube.com/watch?v=uSYExRWbC9Y) **Docker:**[**https://www.youtube.com/watch?v=LmoLFcoaeQw**](https://www.youtube.com/watch?v=LmoLFcoaeQw)[**https://www.youtube.com/watch?v=6\_6MoohzdEI**](https://www.youtube.com/watch?v=6_6MoohzdEI) **Eclipse Debugging**[**https://www.youtube.com/watch?v=HwwF4pvYWws**](https://www.youtube.com/watch?v=HwwF4pvYWws) **Spring Security**[**https://www.youtube.com/watch?v=XTyQIrlyWfQ**](https://www.youtube.com/watch?v=XTyQIrlyWfQ)[**https://www.youtube.com/watch?v=7wA46kRh2u8**](https://www.youtube.com/watch?v=7wA46kRh2u8)[**https://www.youtube.com/watch?v=oLlSs-p6OEs**](https://www.youtube.com/watch?v=oLlSs-p6OEs)[**https://www.youtube.com/watch?v=cnrJ-Nnvoik**](https://www.youtube.com/watch?v=cnrJ-Nnvoik)[**https://www.youtube.com/watch?v=rgG2\_T-OB8g**](https://www.youtube.com/watch?v=rgG2_T-OB8g) **JWT Security**[**https://www.youtube.com/watch?v=feETfZbvu-k**](https://www.youtube.com/watch?v=feETfZbvu-k)[**https://www.youtube.com/watch?v=Hzkw846jIOU**](https://www.youtube.com/watch?v=Hzkw846jIOU)[**https://www.youtube.com/watch?v=bJAsHOH4lMk**](https://www.youtube.com/watch?v=bJAsHOH4lMk) **Distributed Tracing Spring Cloud Sleuth & Zipkin**[**https://www.youtube.com/watch?v=dkLsuDhDo\_g**](https://www.youtube.com/watch?v=dkLsuDhDo_g) **Cloud Deploy:**[**https://www.youtube.com/watch?v=QOwgiJWmZ9k**](https://www.youtube.com/watch?v=QOwgiJWmZ9k)[**https://www.youtube.com/watch?v=Bn2kF-g504k**](https://www.youtube.com/watch?v=Bn2kF-g504k) **Annotations:**[**https://www.youtube.com/watch?v=U\_6Na31fhsE**](https://www.youtube.com/watch?v=U_6Na31fhsE)

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Date : 17-12-2021**

**Demo-Session#2**

----------------------

**Spring Boot Microservices + Angular Project**

----------------------------------------------------------

Microservices:-

One Module/Service that runs in independent.

Ex: Inbox Module -> InboxApplication ---> Build(.jar/.war) -> 1 Server

Gmail :-

Inbox

Drafts

Sent

..etc

\*) Multiple Instances: To handle Load (more no.of request)

Instance : Run one application under server.

\*) 1 MS# = 1 UI (Model + Component + Service) + 1 ReST API ( IL + SL + DAL)

-----------------------------------------------------------------------------------------------------

Spring Boot Microservices + Angular Project

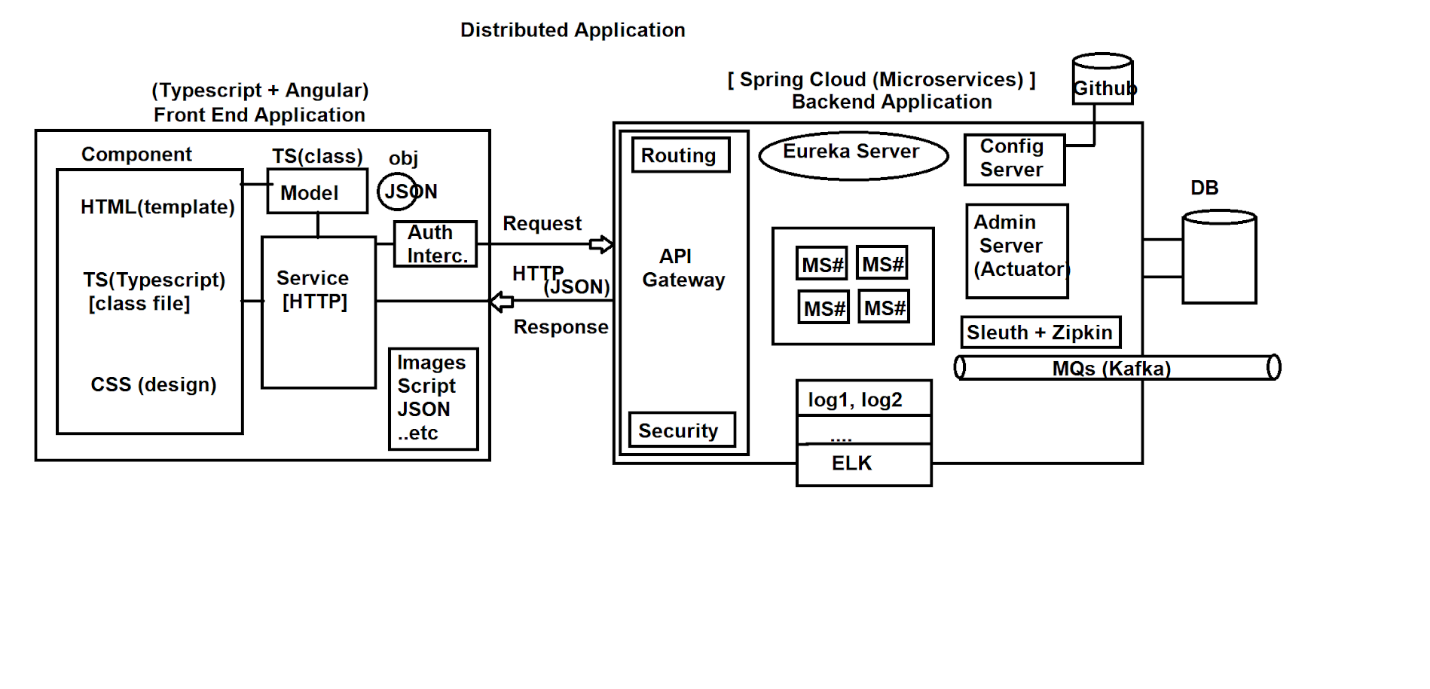
**Pre-Req:**

**Spring boot - Microservices**

**Angular (3 weekends)**

**1 Mini Project(15-20) / 1 Main Project**

**Supply Chain project / Stock Market Application**



---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Date : 20-12-2021**

**Day#1**

---------------------------------------------------------------------------------------------------------------------

**“Spring Boot + Angular” Integration**

-----------------------------------------------------------------

https://www.youtube.com/watch?v=LE3KvvhkUkM

https://www.youtube.com/watch?v=Do87D4TJ9M8

https://www.youtube.com/watch?v=gkp-Mhxuxjo

https://www.youtube.com/watch?v=lpPiM5MIAoQ

https://www.youtube.com/watch?v=Iax4vA4MEPU

----------------------------------------------------------------

**Spring Boot Microservices + Angular project**

----------------------------------------------------------------

\*) Project = Frontend + Backend

\*) Microservices App =

Microservice(REST) + Eureka Server + Config Server + Gateway App

+ MQs + Admin Server..etc

\*) 1 Module = 1 ReST API (API-Application Programming Interface)

Some classes/interfaces we define, access using URLs.

Input/Output : JSON, Primitive.

\*) Example:- Employee (Layers Design : IL/SL/DAL)

POJI - Plain Old Java Interface

POJO - Plain Old Java (Impl class) Object

Q) What is the diff b/w Controller and RestCotroller in Spring/Boot?

A)

[Monolithic Application]

Controller always connected to UI technologies given by Java

(ex: JSP, Thymeleaf, JSF..etc)

[Distributed Applications]

RestController is UI independent, you can even connect with other language applications

(Client Applications) Works with Global Data format ie JSON/XML

@RestController = @Controller + @ResponseBody

----------------------------------------------------------------

**Example:- Employee (T=Employee, ID=Long)**

**1. Repository** (All basic Database Operations code is generated)

package in.nareshit.raghu.repo;

public interface EmployeeRepository

extends JpaRepository<Employee,Long> {

}

**2. Service Interface**

package in.nareshit.raghu.service;

public interface IEmployeeService {

}

**3. Service Impl**

package in.nareshit.raghu.service.impl;

@Service

public class EmployeeServiceImpl

implements IEmployeeService

{

@Autowired

private EmployeeRepository repo; //HAS-A

}

4. RestController

@RestController

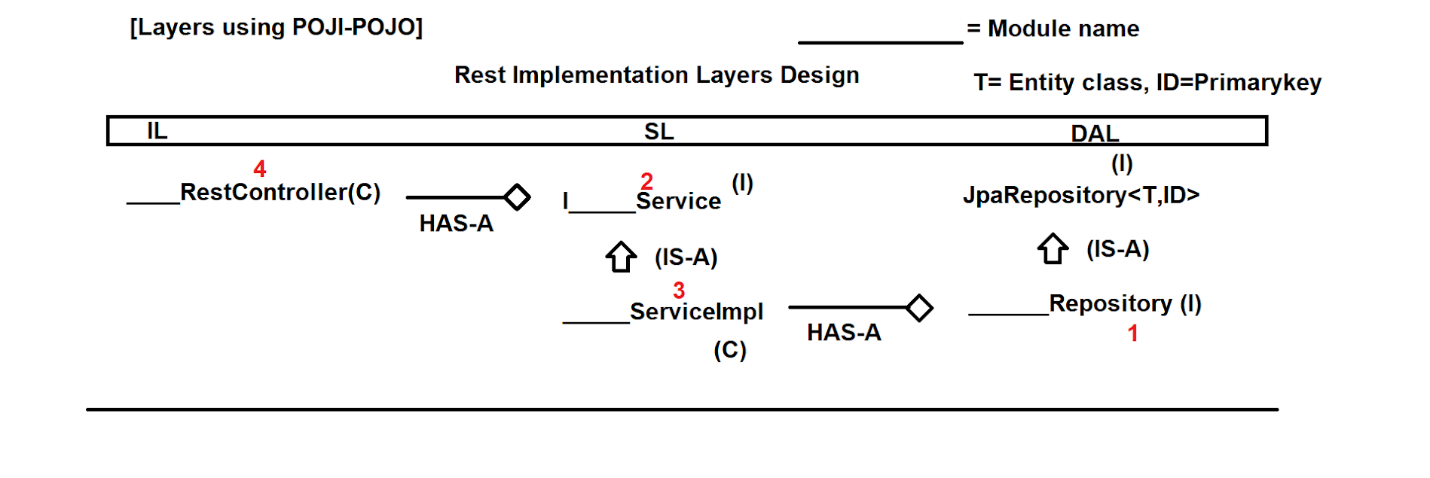
@RequestMapping("/employee")

public class EmployeeRestController {

@Autowired

private IEmployeeService service;

}



---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Date : 21-12-2021**

**Day#2**

-------------------------------------------------------

**1) Project setup**

a. S/w Install

b. Project creation

c. onetime setup files

**2) Project Code**

a. Entity

b. Repository

c. ServiceInterface

d. ServiceImpl

e. RestController

**3) Test using POSTMAN**

**4) Logging and Swagger Config**

---------------------------------------------------------------------------------------------------------------------

**1) Project setup**

a. S/w Install

>> JDK Min : 8/11/latest

>> MySQL Setup (Community Installer)

https://dev.mysql.com/downloads/installer/

>> STS : https://spring.io/tools

> Download as JAR Format > Double click to Extract

> open sts-4.11.1.RELEASE > Click on STS ICON

> Choose workspace D:\PROJECT8PM\_DEC\_2021 > Finish

-DB Setup--------------------

mysql> show databases;

mysql> create database empboot8pm;

mysql> use empboot8pm;

mysql> show tables;

mysql> drop database empboot8pm;

-----------------------------

--STS -- Setup JDK --------------

> Window > preferences > Search with Installed JRE > Click on same

> Remove Default one

> Click on Add > Standard VM > Next > Directory

> Select Location of JDK

C:\Program Files\Java\jdk-11.0.12

> Add / Finish / Apply and Close

-----------------------------------

**b. Project creation**

> File > New > Spring Starter Project

> Enter Details > Next > Choose Dependencies

Web, Lombok, Devtools, Data JPA, MySQL, Lombok.

**Lombok Video:-**

https://www.youtube.com/watch?v=kSceQH4fFi4

ctrl+ and ctrl - for font size adjust

**c. onetime setup files**

---application.properties-----------

# Server Config

server.port=8080

**# Database Config**

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/empboot8pm

spring.datasource.username=root

spring.datasource.password=root

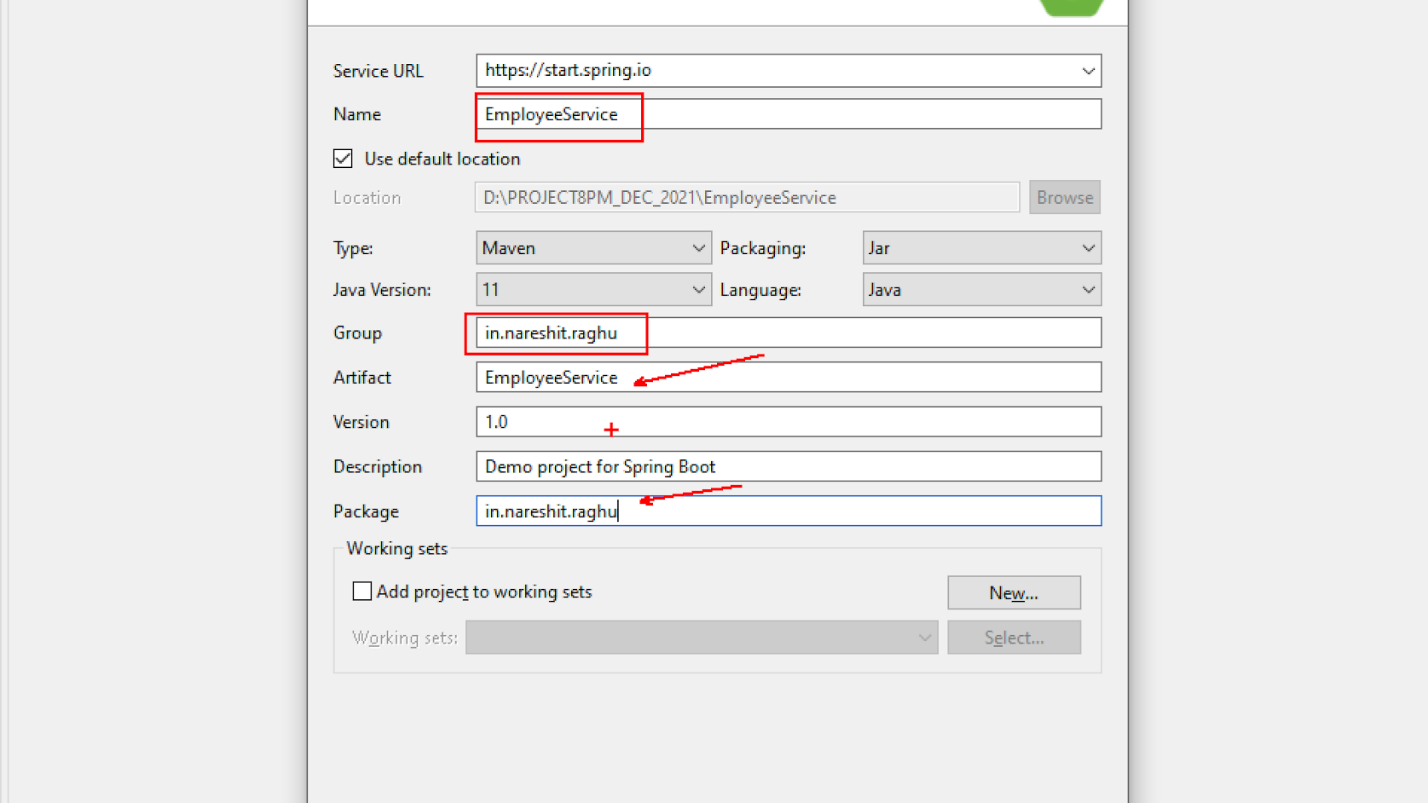
**# ORM Config**

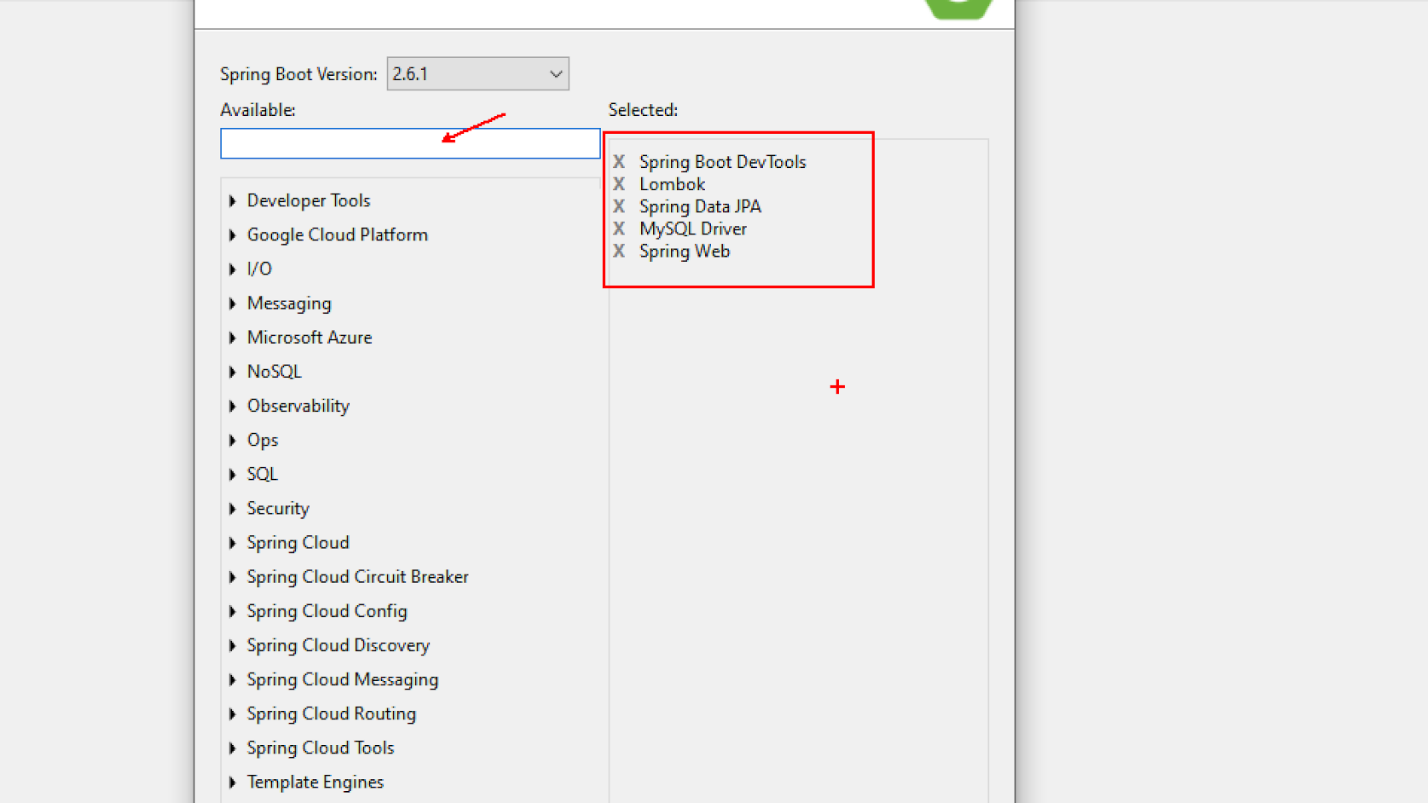
spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=create

-------------------------------------------





---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Date : 22-12-2021**

**DAY#3**

------------------------------------------------------------

**All PDFs Link:**

**https://www.mediafire.com/file/w5x9w5vcmkwkkdv/RaghuSirNareshITJavaPdfs.zip/file**

1) Project setup

a. S/w Install

b. Project creation

c. onetime setup files

2) Project Code

a. Entity

b. Repository

c. ServiceInterface

d. ServiceImpl

e. RestController

3) Test using POSTMAN

4) Logging and Swagger Config

=====================================================================

\*) One MS# Application we develop using Spring REST Concept, with below files in order

a. Entity : A Class that is connected with Database table

b. Repository : Database Operations (insert,update,delete and select)

c. ServiceInterface

d. ServiceImpl : Logics, calculations, Transactions..etc

e. RestController : Webservices Operations (JSON/PathVariables)

==Example module==================

Employee (eid,ename,esal,gen,dept,address)

a. Employee.java

b. EmployeeRepository.java

c. IEmployeeService.java

d. EmployeeServiceImpl.java

e. EmployeeRestController.java

Task#

Student (sid,sname,sfee,gen,course,address)

a. Student.java

b. StudentRepository.java

c. IStudentService.java

d. StudentServiceImpl.java

e. StudentRestController.java

ctrl+shift+O : to get imports

===========Employee Module files================================

**1. Entity**

package in.nareshit.raghu.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Data;

@Data

@Entity

@Table(name="emptab")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name="eid")

private Long empId;

@Column(name="ename")

private String empName;

@Column(name="esal")

private Double empSal;

@Column(name="egen")

private String empGen;

@Column(name="edept")

private String empDept;

@Column(name="eaddr")

private String empAddr;

}

**2. Repository**

package in.nareshit.raghu.repo;

//ctrl+shift+O

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.entity.Employee;

public interface EmployeeRepository

extends JpaRepository<Employee, Long>

{

}

**3. Service interface**

package in.nareshit.raghu.service;

public interface IEmployeeService {

}

**4. Service Impl**

package in.nareshit.raghu.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import in.nareshit.raghu.repo.EmployeeRepository;

import in.nareshit.raghu.service.IEmployeeService;

@Service

public class EmployeeServiceImpl

implements IEmployeeService

{

@Autowired

private EmployeeRepository repo; //HAS-A

}

**5. Rest controller**

package in.nareshit.raghu.rest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import in.nareshit.raghu.service.IEmployeeService;

@RestController

@RequestMapping("/api/employee")

public class EmployeeRestController {

@Autowired

private IEmployeeService service; //HAS-A

}

=============Operations===========================================

**1. CREATE (Backend) Flow**

**S#1** Angular application will send Request using URL and Data (JSON Format)

**S#2** At RestController, @RequestBody reads JSON data and convert into Object Format

Syntax:

@RequestBody ClassName objectName

Ex:

@RequestBody Employee employee

**S#3** IL, calls service layer method and pass employee object

**S#4** Call, Repository method save(obj) [pre-defined method]

**S#5** No Code is required for save at Repository

it call make INSERT call to Database

(Dialect)

INSERT INTO EMPTAB VALUES(----)

**S#6** After save, it returns full object with ID Generated.

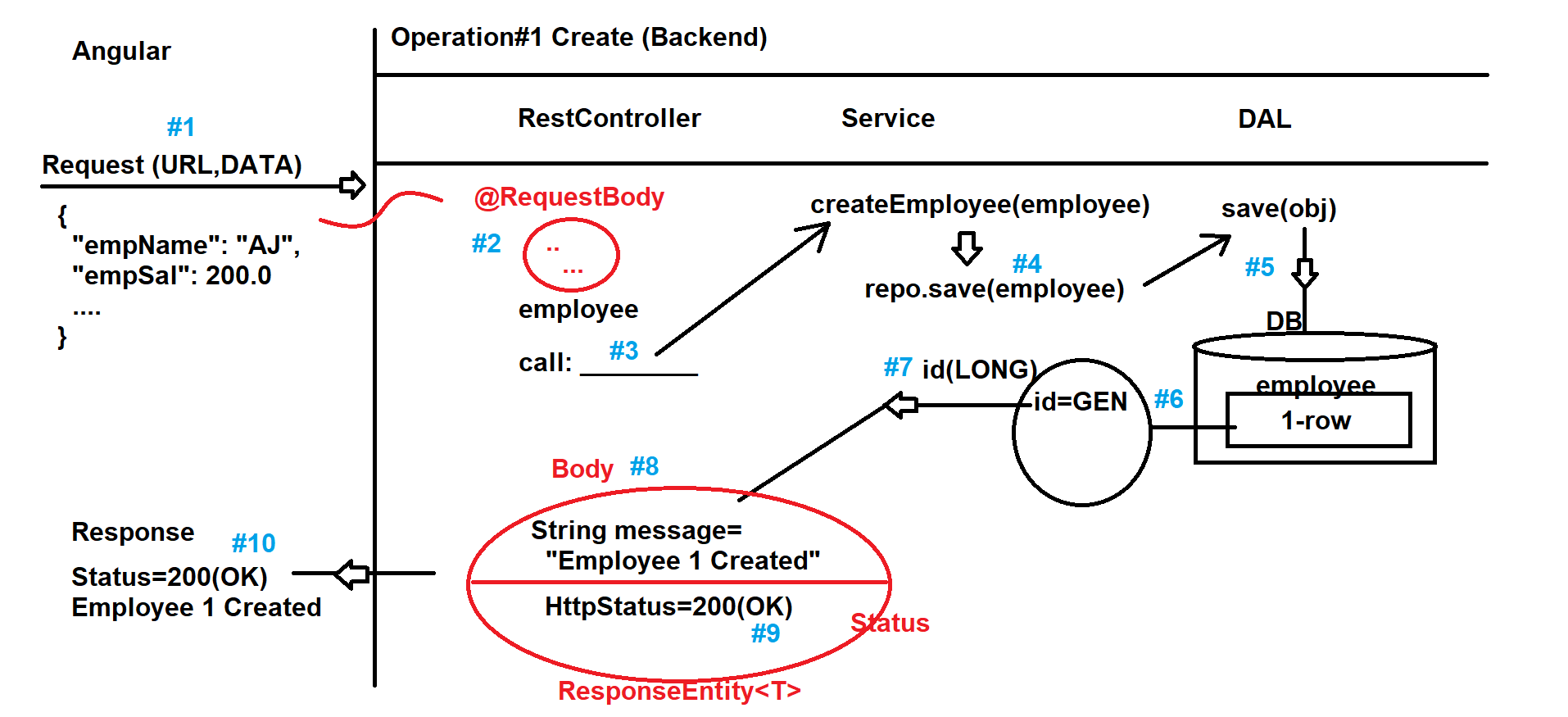
**S#7** We return only ID from service to RestController

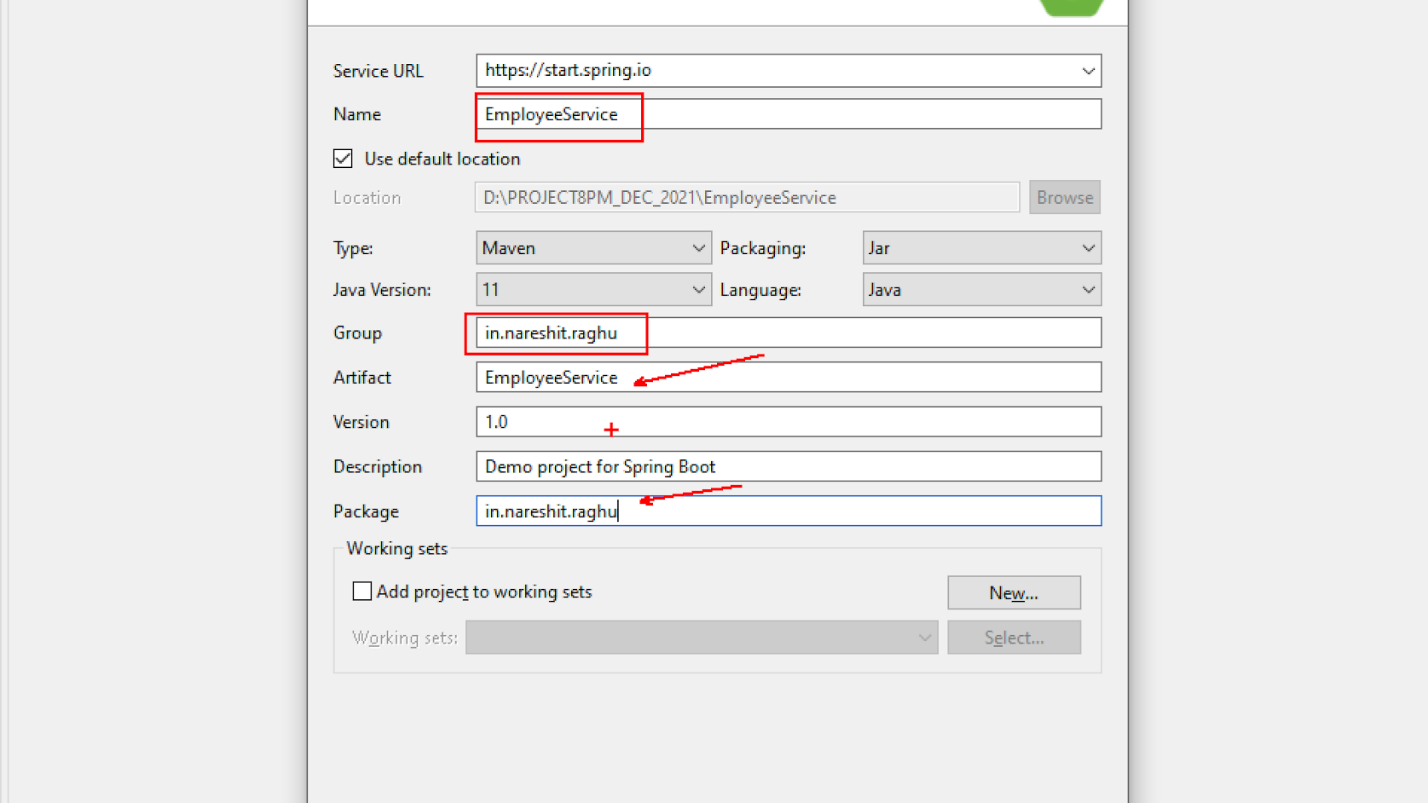
**S#8** At RestController create a full Response Object with Body and HttpStatus

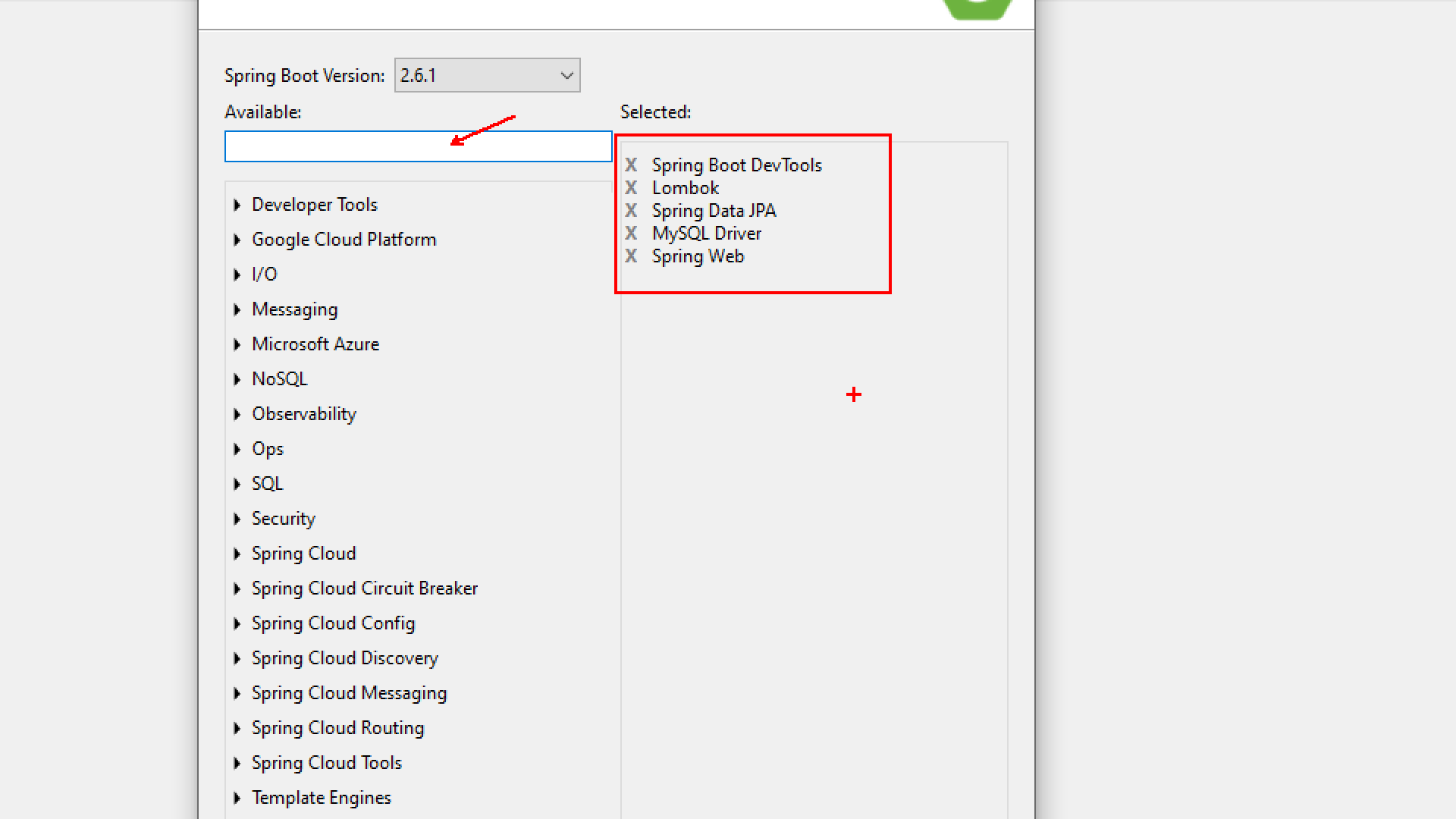
**S#9** Provide HttpStatus to indicate what happen while processing request at backend

ResponseEntity (body,Status)

**S#10** same response is given back to Angular







---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Date : 23-12-2021**

---------------------------------------------------------------------------------------------------------------------

**Spring Boot + Angular**

https://www.youtube.com/watch?v=LE3KvvhkUkM

https://www.youtube.com/watch?v=Do87D4TJ9M8

https://www.youtube.com/watch?v=gkp-Mhxuxjo

https://www.youtube.com/watch?v=lpPiM5MIAoQ

https://www.youtube.com/watch?v=Iax4vA4MEPU

Operation#1 coding part

1. Entity (done)

2. Repository (done)

3. IService

Add a method in IEmployeeService

Long createEmployee(Employee employee);

4. ServiceImpl

Implement method in EmployeeServiceImpl

using repo.save(obj) and return id

public Long createEmployee(Employee employee) {

employee = repo.save(employee);

return employee.getEmpId();

}

5. RestController

Input : JSON --> Object (@RequestBody)

Output : ResponseEntity(message,Status)

===POSTMAN ===================================================

(used for Testing purpose only)

Download and Install

https://www.postman.com/downloads/

> Double click > Next > Next > Finish

> Do not Register > Skip Register in bottom

http://localhost:8080/api/employee/create

===================Frontend s/w======================

a. Node JS

https://nodejs.org/en/download/

> Download > Next > next > finish (done)

> open cmd :

npm -version

node -v

b. Install Angular

https://angular.io/guide/setup-local

(Install angular using cmd)

> npm install -g @angular/cli

> ng version

c. Visual Studio Code

https://code.visualstudio.com/download

> Next > next > finish

> Open Visual Studio code

> File > Open Folder > create a new Folder Ex: D:\Project8PM\_DEC2021

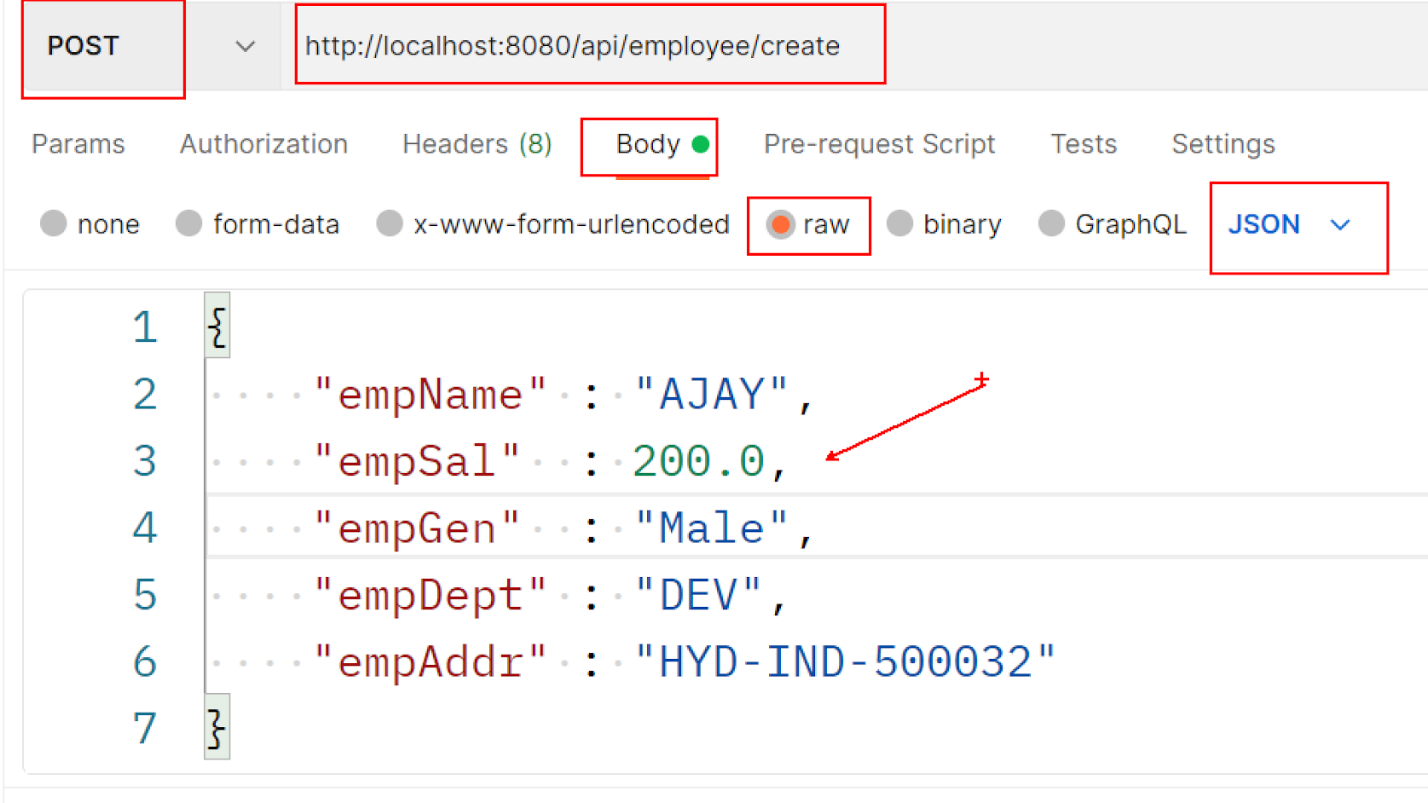
> Select same > finish/select

> open cmd > goto location of D:\Project8PM\_DEC2021

D:\>d:

D:\>cd Project8PM\_DEC2021

D:\Project8PM\_DEC2021\>ng new employee-frontend-app

--------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Date : 24-12-2021**

------------------------------------------------------------

[JAVABYRAGHU@GMAIL.COM](mailto:JAVABYRAGHU@GMAIL.COM)

**MS# Backend REST API Operations**

Operation#1 create (done)

Operation#2 Fetch All

Operation#3 Fetch one by id

Operation#4 Remove one by id

Operation#5 Update

------------------------Operation#2 Fetch All-------------------------

Database(table) --> Rows => List<T> ==> JSON Format [ {},{},{},{} ]

S#1 Angular Application makes request like

http://localhost:8080/api/employee/all

S#2 RestController executes a method based on path matching, which makes call to service

getAllEmployees()

S#3 Service makes call to Repository findAll() which is a pre-defined method

S#4 FindAll() communicates to Database and generates one SELECT SQL Query

S#5 Example SQL: select \* from employee

S#6 Database returns data to Repository which converts into List<T>

S#7 Same List is given back to service layer

S#8 RestController will create ResponseEntity that contains List as Body and Status as 200

S#9 Internally @ResponseBody annotation is executed which converts ReturnType List<T> into JSON Format

This annotation is given default by @RestController

S#10 JSON Response is given back to client machine

==coding steps=====================================

a. add a method in IEmployeeService

List<Employee> findAllEmployees();

b. implement method in EmployeeServiceImpl

public List<Employee> findAllEmployees() {

List<Employee> list = repo.findAll();

return list;

}

c. Rest Controller add a method, that returns ResponseEntity

@GetMapping("/all")

public ResponseEntity<List<Employee>> findAllEmployees() {

List<Employee> list = service.findAllEmployees();

return new ResponseEntity<List<Employee>>(list,HttpStatus.OK);

}

Problem:

SQLSyntaxErrorException: Unknown database 'empboot8pm'

Solution:

mysql> create database empboot8pm;

mysql> use empboot8pm;

mysql> show tables ;

==============Operation#3 Fetch one by id=======================

Coding Steps:

a. IEmployeeService

Employee findOneEmployee(Long id);

b. EmployeeServiceImpl

public Employee findOneEmployee(Long id) {

Optional<Employee> opt = repo.findById(id);

if(opt.isPresent())

return opt.get();

else

return null;

}

c. EmployeeRestController

Q) What is Optional<T> ? why it is used?

A) It is a classes added in JDK 1.8/Java SE 8.

Used to handle null values and avoid NullPointerException.

\*) static path : /pathName

\*) dynamic Path : /{keyName} --> indicates a value comes at runtime.

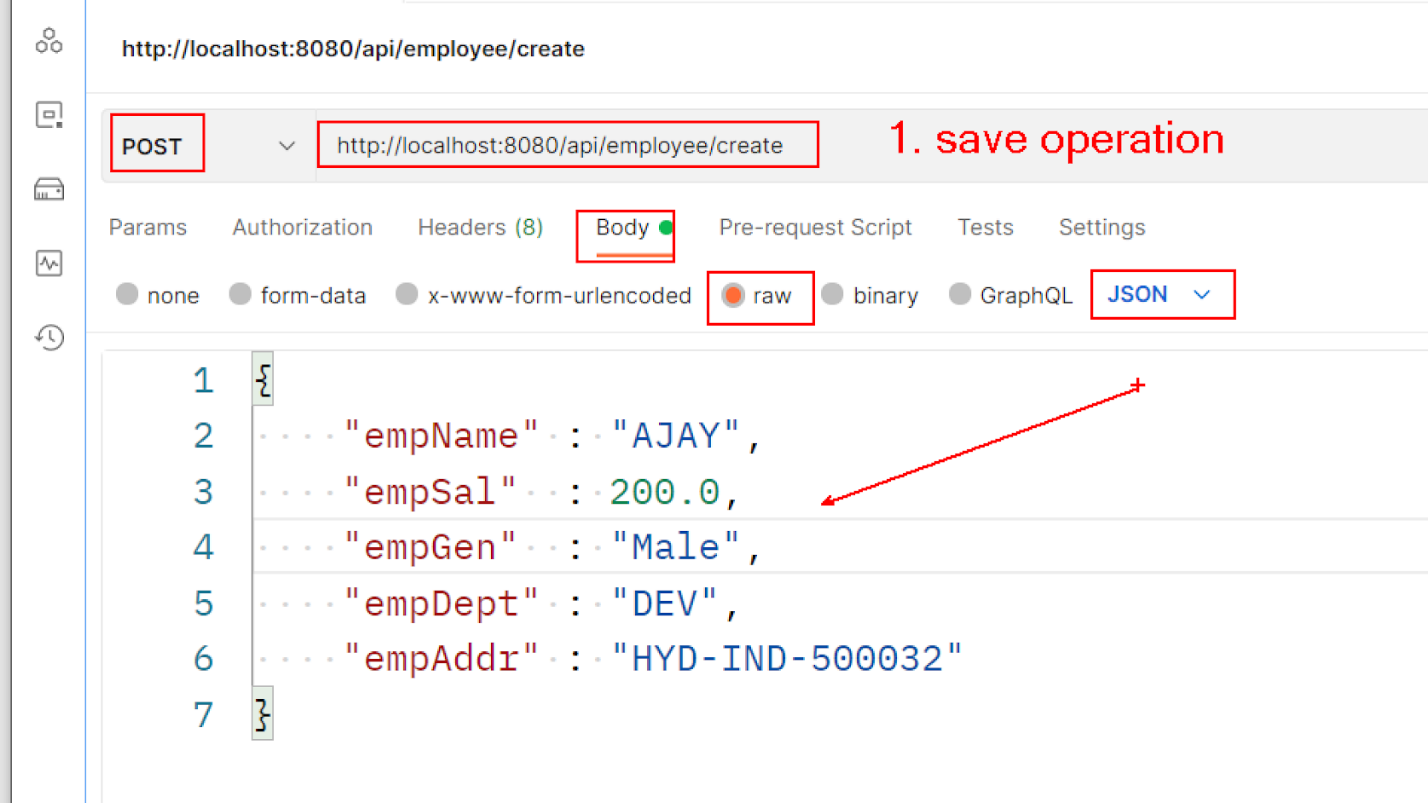
Query Params:-

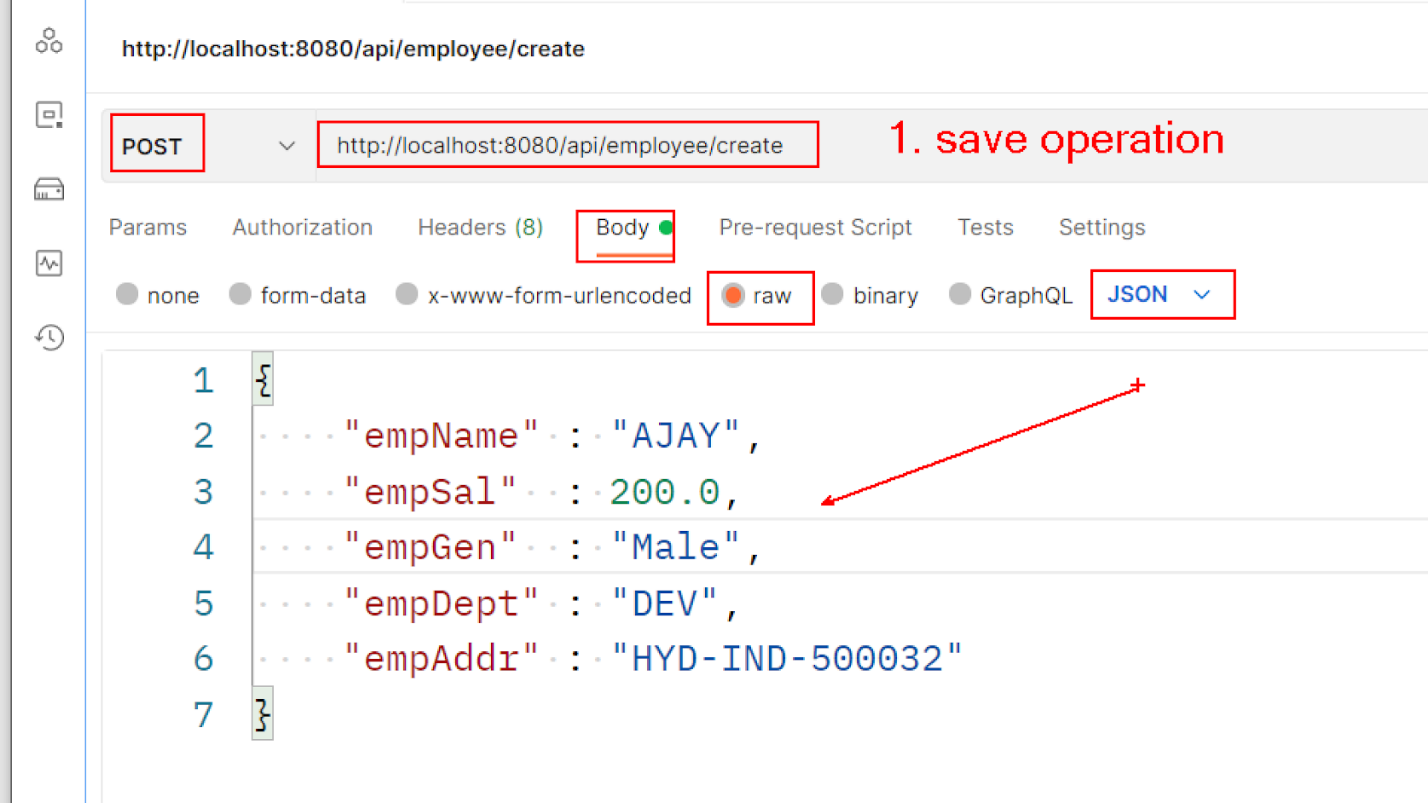
http://localhost:8080/employee/find?name=ABC&id=101

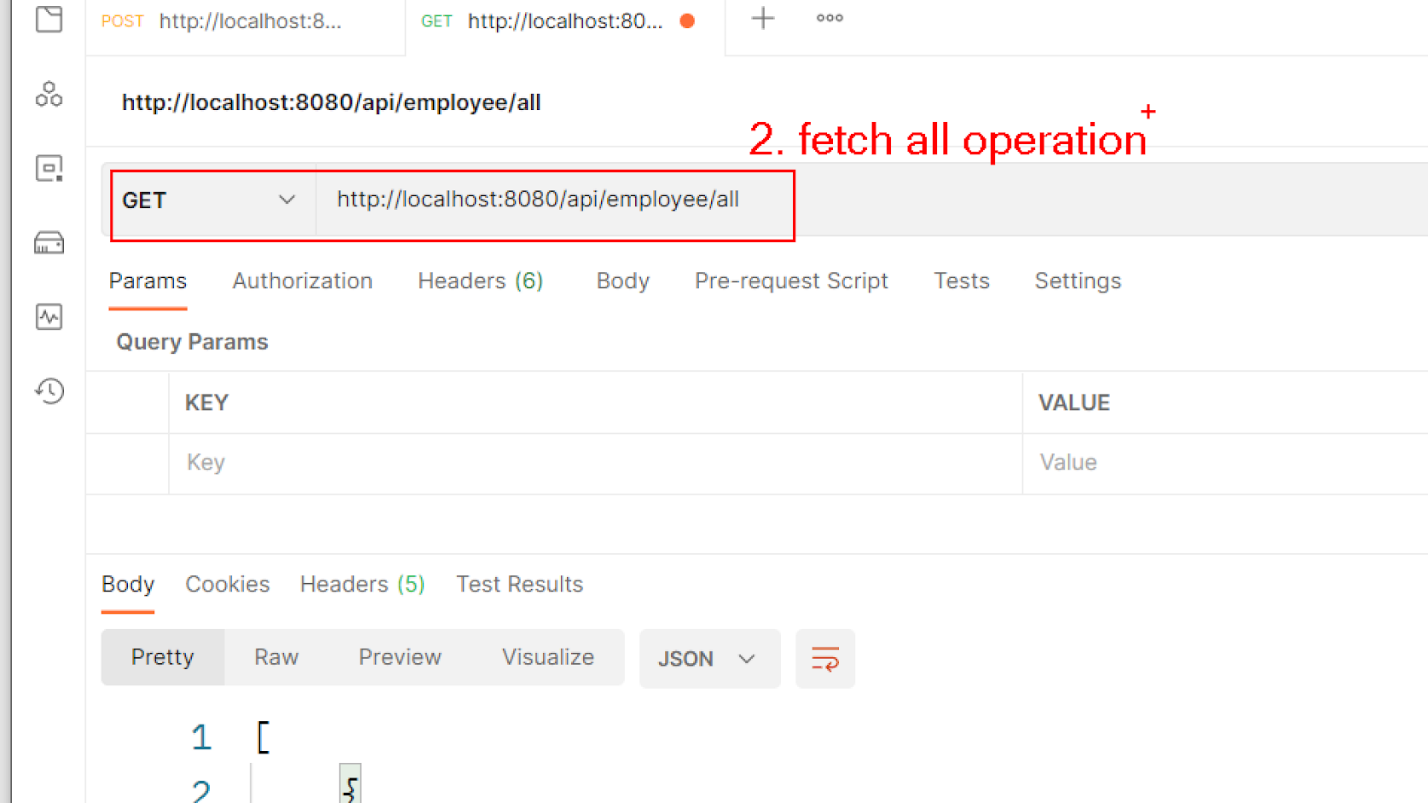
PathVariables:- (Clean URL, Short URL, no Symbols like = and ?, &, Follows order,

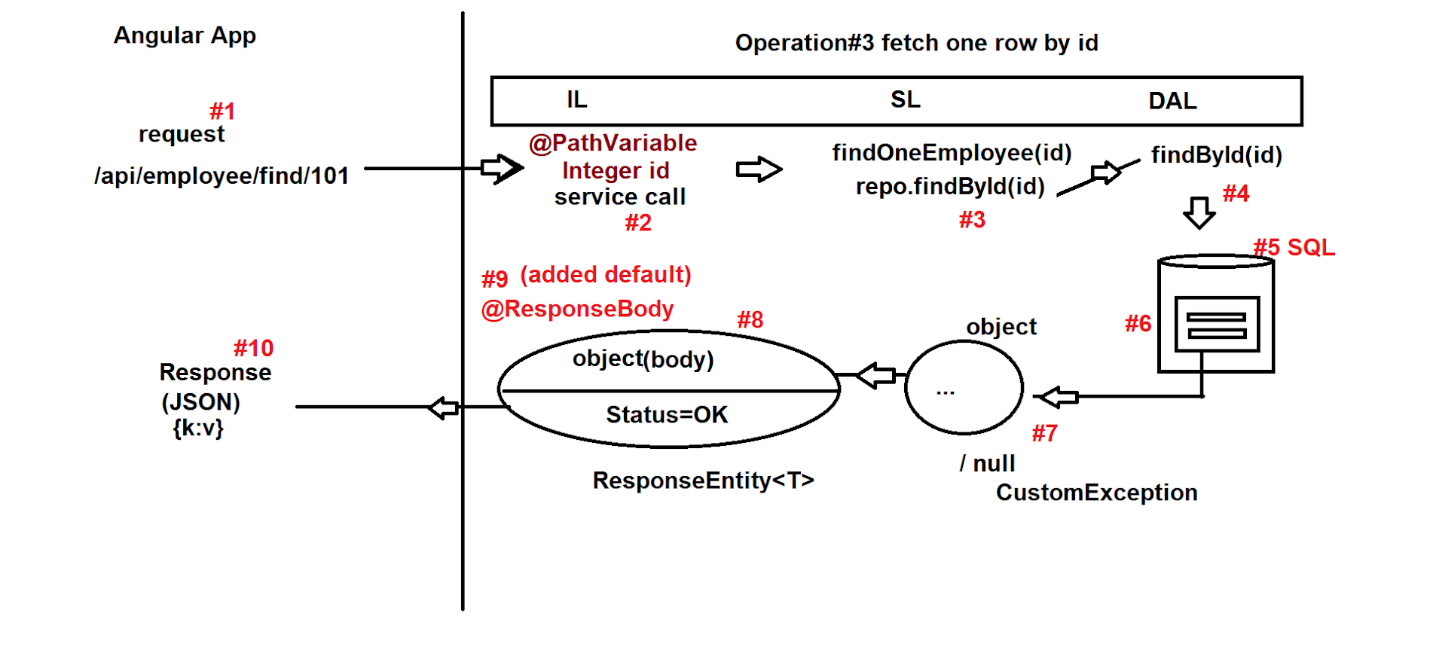
Faster in execution)

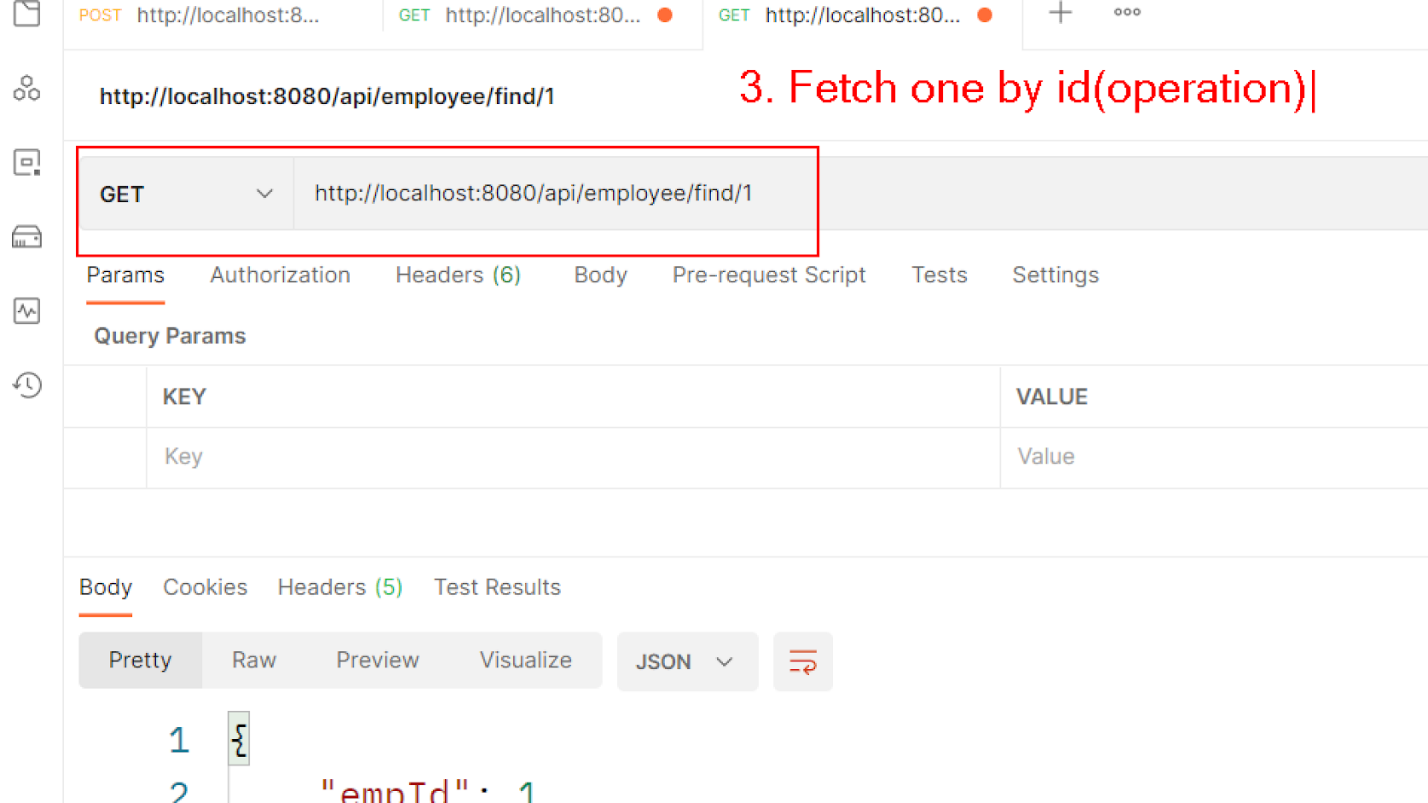
<http://localhost:8080/employee/find/ABC/101>











---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

Date : 27-12-2021 – Global Exception Handling in SpringBoot

---------------------------------------------------------------------------­-

**Spring Boot Error and Exception Handling**

https://www.youtube.com/watch?v=AFq9eK2OoGU

https://www.youtube.com/watch?v=tBVAybXMKzY

https://www.youtube.com/watch?v=M-LRfrYHWrk

\*) Microservice and Angular Design Part

\*) Spring ReST Layers Design (IL,SL,DAL)

\*) Angular Files Overview

\*) Angular , Spring S/w Installation

\*) Mini Project (Backend code)

Employee Module

a. Entity

b. Repository

c. Service Interface

d. ServiceImpl

e. RestController

\*) Operations (Backend)

a. Save Operation (JSON->Object)

b. Fetch All (DB->Rows -> List<T>-> JSON)

c. Fetch one Row (DB->Row -> Object/null -> JSON)

===========Custom Exception =========================================

a. Define Exception class

b. Throw at Service based on condition

c. Catch at Restcontroller

d.\*\* Rethrow for Global Hanlding

=========Global Exception Handler=====================================

e. Catch at Global Handler from any Controller method for generic code

f. Read Exception Message and Return with Response Code :500

Q) What is the diff b/w using Exception(C) and RuntimeException(C) for

creating custom exception?

A)

Q) Is exception also one class?

A) YES

Q) Is exception reference holds object?

A) YES

\*) Symbol ? used at Generics to select dataType based on conditional execution.

==========Operation Delete======================================

1. IEmployeeService

void deleteOneEmployee(Long id);

2. EmployeeServiceImpl

public void deleteOneEmployee(Long id) {

//repo.deleteById(id);

repo.delete(findOneEmployee(id));

}

GET : Fetch Data

POST : Create new data

PUT : Modify existed data

DELETE : Remove existed data

3. RestController

@DeleteMapping("/remove/{id}")

public ResponseEntity<String> deleteEmployee(

@PathVariable Long id

)

{

ResponseEntity<String> resp = null;

try {

service.deleteOneEmployee(id);

resp = new ResponseEntity<String>("Employee Deleted",HttpStatus.OK);

} catch (EmployeeNotFoundException e) {

e.printStackTrace();

resp = new ResponseEntity<String>(e.getMessage(),HttpStatus.INTERNAL\_SERVER\_ERROR);

}

return resp;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_GLOBAL EXCEPTION HANDLER CODING STEPS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Define a Public class with any name and add Annotation @RestControllerAdvice

2. Define a method for one exception type and apply @ExceptionHandler(ExceptionClassName.class)

3. Read Exception object as method param and return ResponseEntity<> ...

4. at RestController Methods modify code as "throw e" inside catch block.

package in.nareshit.raghu.custom.handler;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import in.nareshit.raghu.exception.EmployeeNotFoundException;

/\*\*

\* common catch block code

\*

\*/

@RestControllerAdvice

public class MyCustomExceptionHandler {

@ExceptionHandler(EmployeeNotFoundException.class)

public ResponseEntity<String> handleEmployeeNotFoundException(

EmployeeNotFoundException e

)

{

return new ResponseEntity<String>(

e.getMessage(),

HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

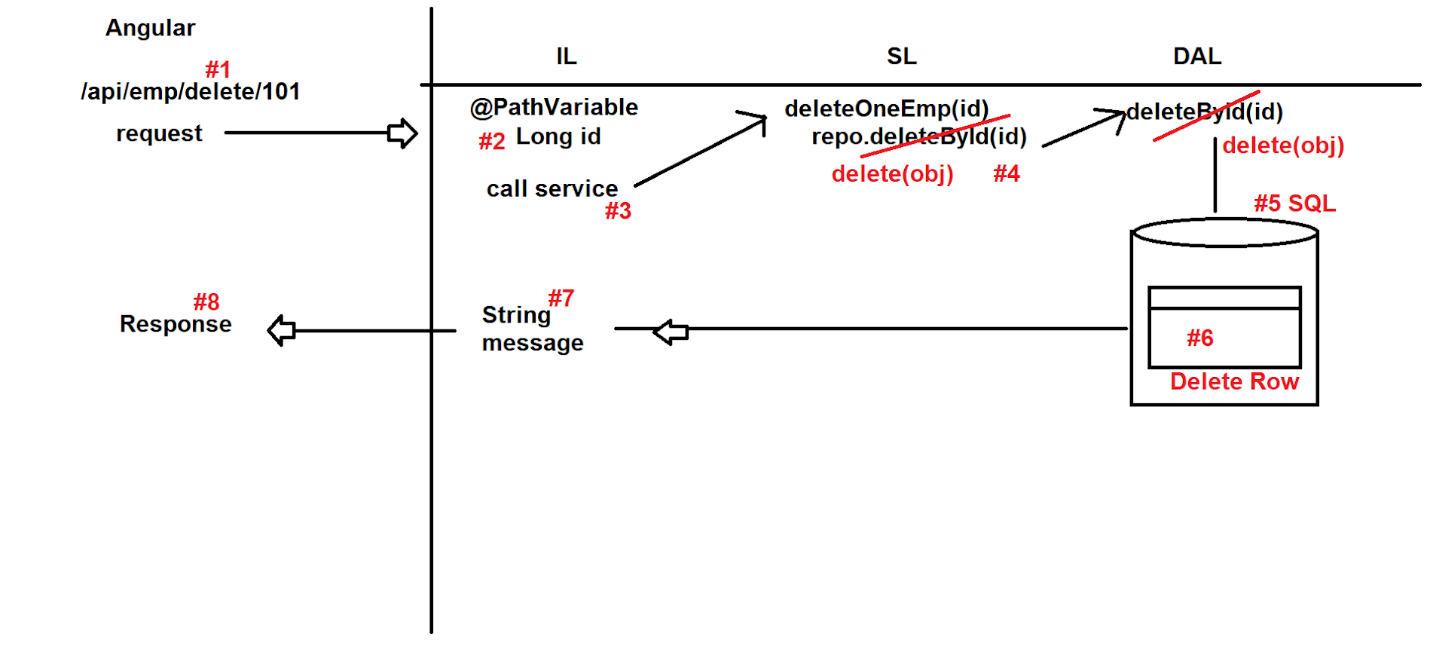
\*) At rest Controller

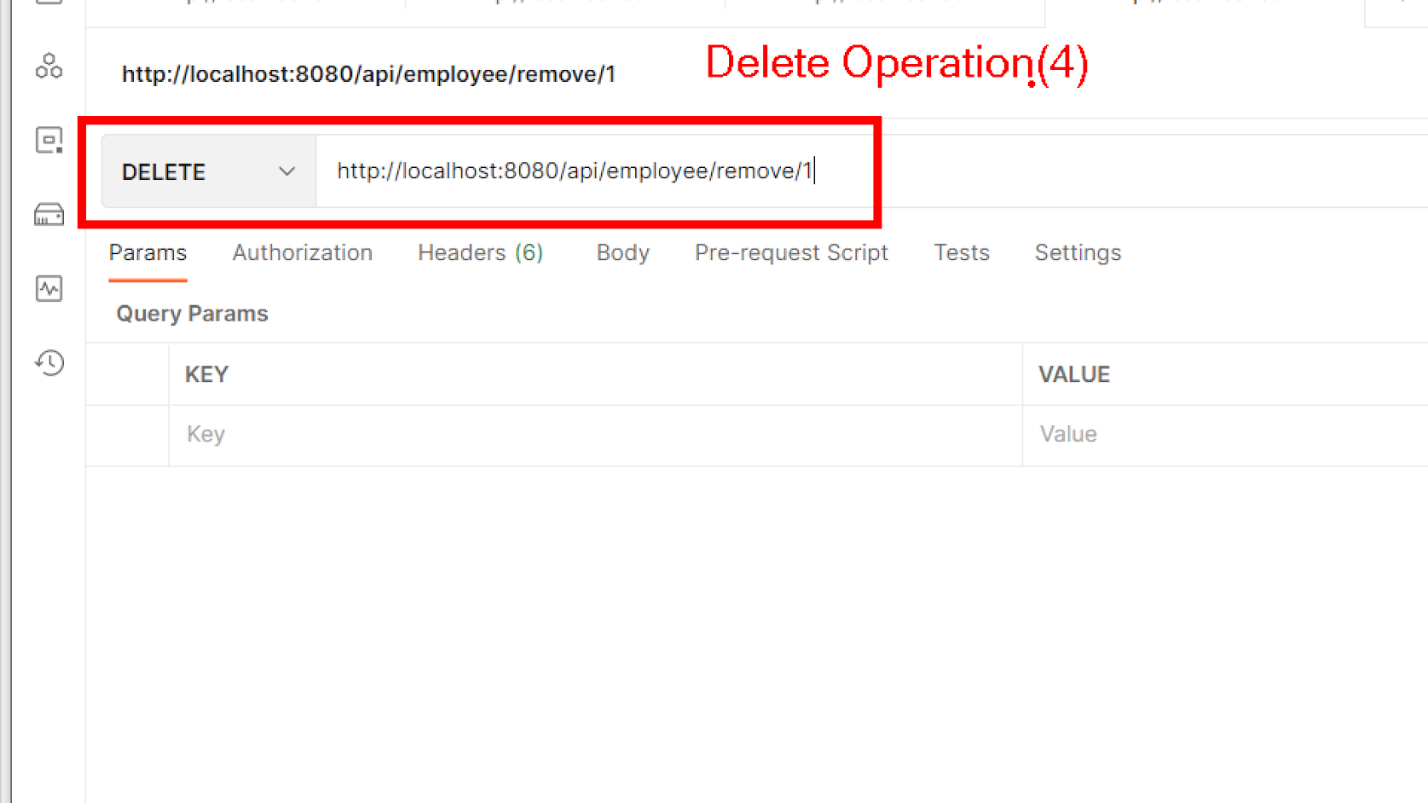
} catch (EmployeeNotFoundException e) {

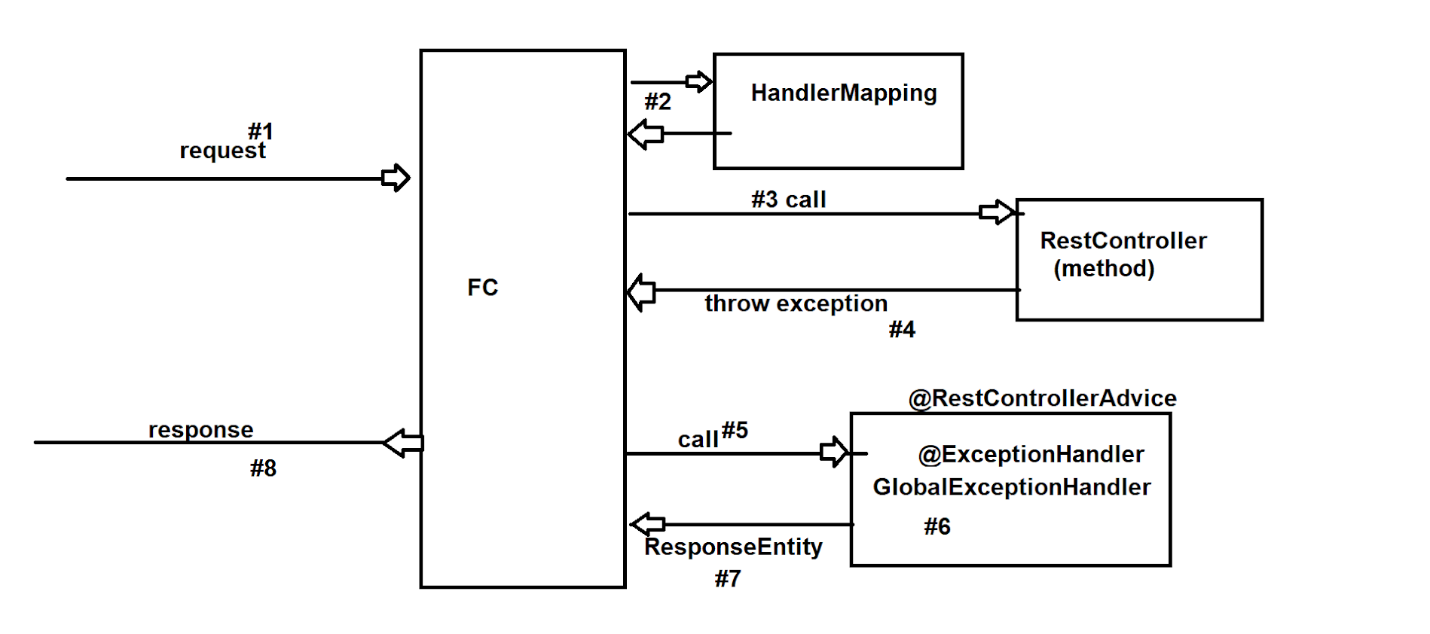
e.printStackTrace();

throw e;

}







---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------