Microservice-Based Java Application Documentation

-S. Snekha (Advisory Intern - CEDA)

1. Introduction

This document provides an overview of a microservice-based Java application consisting of two microservices: **'Intern'** and **'Training'** which can **communicate** with each other. The application is built using Spring Boot and integrates with **Swagger** for API documentation and **Eureka** for service discovery.

2. Architecture Overview

The application consists of two microservices:

1. **Intern Microservice** – Handles CRUD operations for intern records.
2. **Training Microservice** – Manages training records and links them to interns.

The services communicate with each other using **RestTemplate** and are registered with **Eureka** for easy service discovery.

3. Endpoints Documentation

Intern Microservice

- **POST /api/intern/add_interns**
  Creates a new intern record.
- **GET /api/intern/list**
  Returns a list of all interns.
- **GET /api/intern/interns_search/{id}**
  Searches interns by name.
- **DELETE /api/intern/delete_interns/{id}**
  Deletes an intern by ID.
- **PUT /api/intern/edit_interns/{id}**
  Updates an intern record by ID.
- **GET /api/intern/trainings**
  Fetches training data from the Training Microservice.

Training Microservice

- **POST /api/trainings/add_trainings**
  Creates a new training record.
- **GET /api/trainings/training_list**
  Returns a list of all trainings.
- **GET /api/trainings/trainings_search/{id}**
  Searches trainings by title.
- **DELETE /api/trainings/delete_trainings/{id}**
  Deletes a training by ID.

- **PUT /api/trainings/edit_trainings/{id}**
  Updates a training record by ID.
- **GET /api/trainings/interns**
  Fetches intern data from the Intern Microservice.

4. Swagger and Eureka Integration

Swagger is integrated to provide interactive API documentation. Eureka is used for service discovery, allowing the microservices to dynamically register and locate each other.

5. Code Structure

The project follows the standard Spring Boot structure with the following key components:
- **Controller** – Handles incoming HTTP requests.
- **Repository** – Interfaces with the database.
- **Model** – Defines the data structure for interns and trainings.

6. Data Model

Intern Entity:
- **'id'** – Long
-**'name'** – String
-**'practice'** – String
-**'attended'** – Boolean

Training Entity:
-**'id'** – Long
-**'title'** – String
-**'description'** – String
-**'completed'** – Boolean

7. Deployment Instructions

1. Build the project using Maven:
 'mvn clean' 'mvninstall'
2. Start the Eureka Server:
 'mvn spring-boot:run'
3. Start the Intern and Training services:
 'mvn spring-boot:run'

8. Error Handling

The application handles errors gracefully using HTTP status codes:
-'**404 Not Found**' – Resource not found.
-'**400 Bad Request**' – Invalid input.
-'**500 Internal Server Error**' – Internal server error.