access a rest api from another rest api or from mvc project also.

We will be able to represent a rest api (using URL) as a java interface that can be autowired to any other controller

1. Create a spring boot project

Spring web

OpenFeign

1. Go to Application.java and below @SpringBootApplication annotation,

@EnableFeignClients

1. Create an interface that represent a rest api (FoodApi)

package com.ust.spring.api;

import java.io.IOException;

import java.util.List;

import org.springframework.cloud.openfeign.FeignClient;

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

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

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

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

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

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

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

import org.springframework.web.multipart.MultipartFile;

import com.ust.spring.entity.Food;

@FeignClient(name = "food-api",url = "http://localhost:8080/food")

public interface FoodApi {

@PostMapping

public Food addFood(@RequestBody Food food);

@PostMapping("/photo")

public Food addFood1(@RequestParam("name")String name,@RequestParam("price") Double price,@RequestParam("photo") MultipartFile file) throws IOException;

@GetMapping

public List<Food> retrieveAllFoods();

@GetMapping("/{id}")

public Food findFoodById(@PathVariable("id") Integer id);

@PutMapping

public Food updateFood(@RequestBody Food food);

@DeleteMapping("/{id}")

public Food deleteFood(@PathVariable("id") Integer id);

}

1. Create a MyController and autowire the api

package com.ust.spring.controller;

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

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

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

import com.ust.spring.api.FoodApi;

import com.ust.spring.entity.Food;

@RestController

public class MyController {

@Autowired

private FoodApi api;

@GetMapping("/home")

public String home()

{

Food food = api.findFoodById(28);

if(food!=null)

{

return food.getName();

}

return "hello world from 07-Jun";

}

}

H2 database

Employee

id

firstName

lastName

department

1. Create a spring boot application “07-jun-h2-demo”

Spring web

Spring data jpa

H2 database

1. In application.properties:

#spring.datasource.url=jdbc:h2:mem:jag

spring.datasource.url=jdbc:h2:file:./data/sampledata

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.h2.console.path=/h2

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

1. Create entity, repo, service, controller as usual
2. Access the database console using

<http://localhost:8080/h2>

use the same URL, driver, username and password to login to h2 console

1. Create a spring boot project

Spring web

Open Feign

1. In pom.xml we need to add the following dependencies:

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

1. application.properties

spring.mvc.view.prefix=/views/

spring.mvc.view.suffix=.jsp

server.port=8383

1. create folder as per prefix:

webapp/views/ folder

inside this folder, create “employee.jsp”

1. controller:

To use PostgreSQL as database in JPA, use the following:

Pom.xml

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.5.1</version>

</dependency>

Application.properties:

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres

spring.datasource.driver-class-name=org.postgresql.Driver

spring.datasource.username=postgres

spring.datasource.password=password

spring.jpa.properties.dialect= org.hibernate.dialect.PostgreSQL10Dialect