|  |  |  |
| --- | --- | --- |
| ●Introduction to Spring Cloud and Microservices Operations  (Registry, Inter-Service communication, Load balancing, Circuit breaker, Monitoring etc.,),  Spring Batch | Day2: Hands-on Registry, Inter-service communication, Load balancing, circuit breaker, API Gateway | Spring Batch Self-learning |

Service Discovery

Circuit Breaker

Load Balancing

Activity:

What are:

Service Discovery

Circuit Breaker

Load Balancing

Inter-Service communication

Feign client

RestTemplate

Demo:

1. Lets create a restful webservice for “Employee”
2. Create another restful webservice and access the employee service

Employee

id

name

address

Repository

Controller

Port 8080 is in use.

To find which service is running in 8080, go to cmd:

netstat -aon

find the pid of the process that is listening to 8080 port.

Kill the process using tast manager.

------change the port

server.port=8383

we have created the employee service.

Now, lets create another service and access employee service from there.

---------------Employee service-----------------

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.13</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.ust.demo</groupId>

<artifactId>18-jul-employee-h2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>18-jul-employee-h2</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-boot-starter</artifactId>

<version>3.0.0</version>

</dependency>

<!-- <dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency> -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

server.port=8383

spring.datasource.username=sa

spring.datasource.password=password

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

spring.datasource.driver-class-name=org.h2.Driver

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

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

package com.ust.demo.julemployeeh2.model;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Employee {

@Id

private Integer id;

private String name;

private String address;

public Employee() {}

public Employee(Integer id, String name, String address) {

super();

this.id = id;

this.name = name;

this.address = address;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", address=" + address + "]";

}

}

package com.ust.demo.julemployeeh2.model;

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

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer>

{

}

package com.ust.demo.julemployeeh2.controller;

import java.util.List;

import java.util.Optional;

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

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.RequestMapping;

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

import com.ust.demo.julemployeeh2.model.Employee;

import com.ust.demo.julemployeeh2.model.EmployeeRepository;

@RestController

@RequestMapping("/employee")

public class EmployeeController {

@Autowired

private EmployeeRepository er;

@PostMapping

public Employee addEmployee(@RequestBody Employee employee) {

return er.save(employee);

}

@GetMapping

public List<Employee> retrieveAllEmployees() {

return er.findAll();

}

@GetMapping("/{id}")

public Employee findEmployeeById(@PathVariable Integer id) {

Optional<Employee> temp = er.findById(id);

Employee employee=null;

if(temp.isPresent())

{

employee=temp.get();

}

return employee;

}

@PutMapping("/{id}")

public Employee updateEmployee(@PathVariable Integer id, @RequestBody Employee employee) {

Employee temp=null;

temp=findEmployeeById(id);

if(temp!=null)

{

er.save(employee);

temp=employee;

}

return temp;

}

@DeleteMapping("/{id}")

public Employee delete(@PathVariable Integer id) {

Employee temp=null;

temp=findEmployeeById(id);

if(temp!=null)

{

er.delete(temp);

}

return temp;

}

}

--------------------------------------end-------------------

Create main service and access employee service

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.13</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.ust.demo</groupId>

<artifactId>18-jul-main-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>18-jul-main-service</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

server.port=8384

package com.ust.demo;

public class Employee {

private Integer id;

private String name;

private String address;

public Employee() {}

public Employee(Integer id, String name, String address) {

super();

this.id = id;

this.name = name;

this.address = address;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", address=" + address + "]";

}

}

package com.ust.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.http.ResponseEntity;

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

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

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

import org.springframework.web.client.RestTemplate;

@SpringBootApplication

@RestController

@RequestMapping("/main")

public class Application {

@GetMapping

public String home()

{

//access the employee service and find employee by id

//http://localhost:8383/employee/13

RestTemplate rt=new RestTemplate();

ResponseEntity<Employee> result = rt.getForEntity("http://localhost:8383/employee/13", Employee.class);

return "Hello worldd: "+result.getBody();

}

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

Add the following dependency:

<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-commons -->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-commons</artifactId>

<version>2.0.2.RELEASE</version>

</dependency>

Instead of creating an object of RestTemplate, we can autowire it.

Create a Configuration class and define a bean for RestTemplate class

@Configuration

public class MyConfiguration {

@Bean

@LoadBalanced

public RestTemplate rt()

{

return new RestTemplate();

}

}

package com.ust.demo;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import org.springframework.http.ResponseEntity;

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

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

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

import org.springframework.web.client.RestTemplate;

@SpringBootApplication

@RestController

@RequestMapping("/main")

public class Application {

@Autowired

private RestTemplate rt;

@GetMapping

public String home()

{

//access the employee service and find employee by id

//http://localhost:8383/employee/13

// RestTemplate rt=new RestTemplate();

ResponseEntity<Employee> result = rt.getForEntity("http://localhost:8383/employee/13", Employee.class);

return "Hello worldd: "+result.getBody();

}

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

Task:

Use RestTemplate to access the employee service using:

GET

POST

PUT

DELETE

<https://dzone.com/articles/service-discovery-and-load-balancing-using-eureka>