**Exercises on Microservices with Spring Boot 3.0**

**1.Build a User and Order Management System Problem:**

Create two microservices: • User Service to manage users. • Order Service to manage orders placed by users.

**User:**

**UserController**

package com.example.user\_service.controller;

import com.example.user\_service.entity.User;

import com.example.user\_service.repository.UserRepository;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserRepository userRepo;

@PostMapping

public User createUser(@RequestBody User user) {

return userRepo.save(user);

}

@GetMapping("/{id}")

public User getUser(@PathVariable Long id) {

return userRepo.findById(id).orElse(null);

}

@GetMapping

public List<User> getAllUsers() {

return userRepo.findAll();

}

}

**Entity**

package com.example.user\_service.entity;

import jakarta.persistence.\*;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

}

**User repo**

package com.example.user\_service.repository;

import com.example.user\_service.entity.User;

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

public interface UserRepository extends JpaRepository<User, Long> {}

**app.yml**

server:

port: 8081

spring:

datasource:

url: jdbc:mysql://localhost:3306/userdb

username: root

password: GABB2004

jpa:

hibernate:

ddl-auto: update

show-sql: true

**Order**

package com.example.order\_service.client;

import com.example.order\_service.dto.User;

import org.springframework.cloud.openfeign.FeignClient;

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

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

@FeignClient(name = "user-service", url = "http://localhost:8081")

public interface UserClient {

@GetMapping("/users/{id}")

User getUserById(@PathVariable("id") Long id);

}

**OderController**

package com.example.order\_service.controller;

import com.example.order\_service.entity.Order;

import com.example.order\_service.repository.OrderRepository;

import com.example.order\_service.client.UserClient;

import com.example.order\_service.dto.User;

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

import org.springframework.web.bind.annotation.\*;

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/orders")

public class OrderController {

@Autowired

private OrderRepository orderRepo;

@Autowired

private UserClient userClient;

@PostMapping

public Order createOrder(@RequestBody Order order) {

return orderRepo.save(order);

}

@GetMapping("/{id}")

public Map<String, Object> getOrder(@PathVariable Long id) {

Order order = orderRepo.findById(id).orElse(null);

User user = userClient.getUserById(order.getUserId());

Map<String, Object> response = new HashMap<>();

response.put("order", order);

response.put("user", user);

return response;

}

}

**Userconnection**

package com.example.order\_service.dto;

public class User {

private Long id;

private String name;

private String email;

}

**Order**

package com.example.order\_service.entity;

import jakarta.persistence.\*;

@Entity

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String productName;

private Double price;

private Long userId;

}

**App.yml**

server:

port: 8082

spring:

datasource:

url: jdbc:mysql://localhost:3306/orderdb

username: root

password: GABB2004

jpa:

hibernate:

ddl-auto: update

show-sql: true

feign:

client:

config:

default:

connectTimeout: 5000

readTimeout: 5000

**3. Implement an API Gateway Problem: Create an API Gateway to route requests to:**

• Customer Service • Billing Service

**Customer Service**

package com.example.customerservice;

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

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

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

@RestController

@RequestMapping("/api/customer")

public class CustomerController {

@GetMapping("/hello")

public String hello() {

return "Hello from Customer Service!";

}

}

**Application.yml**

server:

port: 8081

Billing

package com.example.billingservice.controller;

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

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

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

@RestController

@RequestMapping("/api/billing")

public class BillingController {

@GetMapping("/hello")

public String hello() {

return "Hello from Billing Service!";

}

}

**Application.yml**

server:

port: 8082

spring:

application:

name: billingservice

api-gateway

server:

port: 8080

spring:

redis:

host: localhost

port: 6379

cloud:

gateway:

routes:

- id: customerservice

uri: http://localhost:8081

predicates:

- Path=/customer/\*\*

filters:

- RewritePath=/customer/(?<segment>.\*), /api/customer/${segment}

- id: billingservice

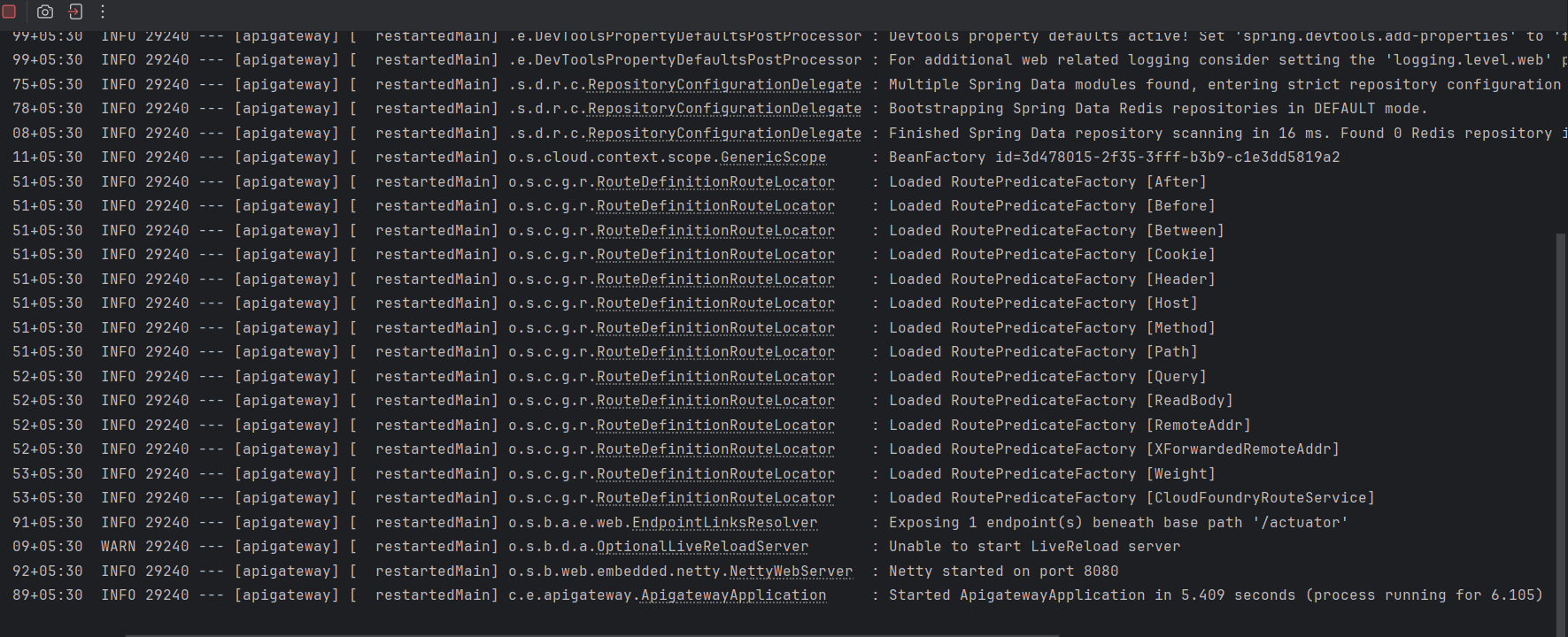
uri: http://localhost:8082

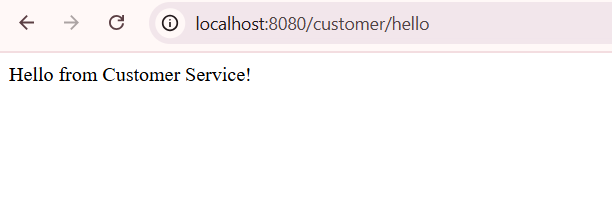
predicates:

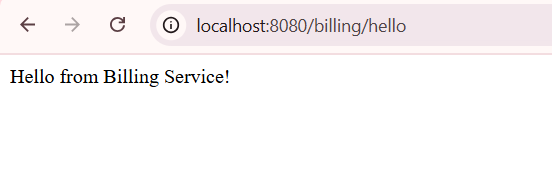
- Path=/billing/\*\*

filters:

- RewritePath=/billing/(?<segment>.\*), /api/billing/${segment}







**4. Resilient Microservices with Circuit Breaker Problem: A Payment Service calls a slow third-party API.**

**Payment**

**Paymentcontroller**

package com.example.paymentservice.controller;

import com.example.paymentservice.service.PaymentService;

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

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

@RestController

public class PaymentController {

private final PaymentService paymentService;

public PaymentController(PaymentService paymentService) {

this.paymentService = paymentService;

}

@GetMapping("/pay")

public String pay() {

return paymentService.makePayment();

}

}

**Paymentservice**

package com.example.paymentservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class PaymentserviceApplication {

public static void main(String[] args) {

SpringApplication.run(PaymentserviceApplication.class, args);

}

}

**Application.yml**

server:

port: 8080

resilience4j:

circuitbreaker:

instances:

paymentService:

registerHealthIndicator: true

slidingWindowSize: 5

permittedNumberOfCallsInHalfOpenState: 2

minimumNumberOfCalls: 3

waitDurationInOpenState: 10s

failureRateThreshold: 50

eventConsumerBufferSize: 10

management:

endpoints:

web:

exposure:

include: "\*"

**SlowapiServer**

package com.example.slowapiservice.controller;

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

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

@RestController

public class SlowApiController {

@GetMapping("/slow-api")

public String slowApi() throws InterruptedException {

Thread.sleep(4000);

return "Success from slow API";

}

}

**App.yml**

server:

port: 8081

