DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Especialização Back

Exercícios com o professor

Nosso cliente "Banco Digital" nos informou que o serviço de recuperação de usuários é consumido por outros microsserviços e, portanto, de suma importância. Vamos implementar nesse exercício um servidor local eureka para visualizar os serviços online e um circuit breaker para redirecionar as requisições do (possível) serviço offline, a fim de evitar falhas em cascata. Para tanto, vamos utilizar o endpoint anteriormente criado na nossa atividade de mesa. Esse microsserviço será anotado como "eureka client" para que seja visualizado pelo servidor eureka e circuit breaker.

Dependências do servidor Eureka

- Spring Cloud Starter Netflix Eureka Server;
- Spring Boot Starter Web 2.6.3;

Instruções do servidor

1. Adicione a anotação @EnableEurekaServer na sua aplicação principal, aquela anotada com @SpringBootApplication (certifique-se de manter as duas anotações juntas)
2. Dentro do pacote "resources", renomeie o arquivo "application.properties" para "application.yml" e adicione as seguintes informações (altere o nome service-registry para um nome de sua escolha):

```yaml
server:
  port: 8761
spring:
  application:
    name: service-registry
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
  server:
    wait-time-in-ms-when-sync-empty: 0
```

3. Builde o projeto (mvn clean install -DskipTests) e execute seu servidor;
4. Para acessar o eureka, navegue para "**localhost:8761**"

Você deve visualizar em seu navegador a página do spring Eureka e o nome do seu servidor na lista "Instances currently registered with Eureka".

Dependências adicionais da API principal

- Spring Cloud Starter Netflix Eureka Client;
- Spring Boot Starter Actuator;

Instruções da API

1. Adicione a anotação @EnableDiscoveryClient ao seu controller, juntamente com o @RestController. Isso garantirá que sua API seja descoberta automaticamente pelo servidor Eureka, e que seu endpoint seja visível para nosso próximo passo (API gateway/circuit breaker);

2. Dentro do pacote “resources”, renomeie o arquivo “application.properties” para “application.yml” e adicione as seguintes informações:

```yaml
spring:
  application:
    name: digital-banking

server:
  port: 8001
  servlet:
    context-path: /digitalbank/v1/

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    serviceUrl:
      defaultZone: http://localhost:8761/eureka
```

3. Em “spring: application: name:”, insira o nome desejado para a sua API;
4. Em “server: port:”, insira a porta 8001;
5. em “server: servlet: context-path”, insira um path contextual de sua escolha para o acesso ao endpoint.
6. Builde o projeto (mvn clean install -DskipTests) e execute o projeto;
7. Certifique-se do funcionamento correto do seu endpoint em **localhost:8001/[seu-path]/[seu-endpoint]**;
8. Certifique-se de que o nome da sua API definido anteriormente se encontra no contexto do servidor Eureka em localhost:8761;

Isso se deve pelo fato de que a requisição deverá passar primeiro pela nossa API gateway com circuit breaker, para depois ser redirecionada para esse novo endereço. O contexto “defaultZone” indica onde encontrar o caminho do servidor Eureka previamente estabelecido, garantindo que o registro da nossa API no servidor seja feita de forma automática. A implementação do circuit breaker vem a seguir.

Dependências da API Gateway

- Spring Boot Starter WebFlux;
- Spring Boot Starter Security;

- Spring Boot Starter Actuator;
- Spring Cloud Starter Gateway;
- Spring Cloud Starter Circuitbreaker Reactor Resilience4j;
- Spring Cloud Starter Netflix Eureka Client;
- Resilience4j Spring Boot 2

Instruções da API

1. Construa um endpoint para o fallback da API principal. Esta API deve retornar uma mensagem informando "*Servidor encontra-se temporariamente indisponível*";
2. Juntamente com a anotação @GetMapping, o endpoint deve conter a anotação @CircuitBreaker (name = "[nome-do-seu-circuit-breaker]")
3. Construa um GatewayFilter com a anotação @Component para fim de autenticação da API principal; **(Aqui cabe explicação e instrução do professor, vide a classe de autenticação "PreFiltroUsuario" no código de exemplo)**

4. Adicione os seguintes parâmetros no arquivo “application.yml”:

```yaml
server:
  port: 8080
spring:
  application:
    name: api-gateway
    eureka:
      client:
        serverUrl:
          defaultZone: http://localhost:8761/eureka
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
          lower-case-service-id: true
      routes:
        - id: apigateway
          uri: lb://DIGITAL-BANKING
          predicates:
            - Path=/digitalbank/v1/**
          filters:
            - name: CircuitBreaker
              args:
                name: apigateway
                fallbackUri: forward:/fallback/usuarioFallback
            - name: PreFiltroUsuario
```

5. Em “server: port”, adicione a porta de entrada 8080;
6. Em “spring: application: name:”, insira o seu nome desejado para a sua API gateway;
7. Em “routes: - id”, insira o nome desejado para o ID da sua rota;
8. Em “routes: uri”, insira “**lb://[*nome*-da-sua-API-principal]**”, configurada no exercício anterior;
9. Em “routes: predicates”, insira o path contextual que você configurou no exercício anterior;
10. Nesse passo, “filters” refere-se ao GatewayFilterFactory utilizado para rotear as requisições que passarão pelo nosso gateway.

10.1. O primeiro filtro, "CircuitBreaker", refere-se à anotação que colocamos anteriormente no nosso endpoint, no passo 2.

10.2. Em "filters: args: name:", insira o nome colocado na sua anotação, também no passo 2;

10.3. Em "filters: args: fallbackUri", insira **forward:/[seu-path]/[seu-endpoint]**, criado no passo 1;

11. O segundo filtro refere-se ao nosso AbstractGatewayFilterFactory de autenticação, onde vamos requisitar as informações do funcionário do banco para acessarmos o endpoint **(Aqui cabe explicação e instrução do professor, vide a classe de autenticação "PreFiltroUsuario" no código de exemplo)**;

Instruções dos serviços

Vamos agora executar nossos 3 serviços em conjunto.

1. Inicie seu servidor Eureka e certifique-se do seu funcionamento acessando **localhost:8761**;
2. Inicie sua API principal e certifique-se do seu funcionamento acessando **localhost:*8001*/[seu-path]/[seu-endpoint]**;
3. Inicie sua API Gateway e certifique-se do seu funcionamento acessando **localhost:*8080*/[seu-path]/[seu-endpoint]**;

O intuito aqui é manter o caminho com porta 8001 oculta, e acessar nosso endpoint apenas pelo caminho com porta 8080, ou seja, através da nossa API Gateway.

4. Derrube seu serviço da API principal e acesse novamente o caminho **localhost:*8080*/[seu-path]/[seu-endpoint]**;
5. Certifique-se de que sua API gateway ativou o circuit breaker e redirecionou a requisição para o endpoint fallback, informando então em tela a mensagem que informa que o servidor encontra-se indisponível.

Nesse ponto, poderíamos implementar qualquer serviço de circuit breaker através do nosso endpoint de fallback.