Spring Cloud Service Discovery

Eureka ile Service Discovery

- Spring Cloud service discovery kabiliyeti için Netflix OSS'nin **Eureka** çözümünü kullanma imkanı sunar

Eureka Service Registry Server

pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
    </dependency>
</dependencies>
```

Eureka Service Registry Server

```
@EnableEurekaServer
@SpringBootApplication
public class ServiceRegistryApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServiceRegistryApplication.class, args);
    }
}
```

Eureka Service Registry Server

application.properties

```
server.port=8761

eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Default eureka server port değeridir

Classpath'de discovery client dependency'leri varsa server'ın kendi kendini register etmemesi için gereklidir

Eureka Service Discovery Client

pom.xml

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

Eureka Service Discovery Client


```
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

spring.application.name=client-service
```

Eureka Service Discovery Client

Eklenmesi şart değildir, classpath'de netflix-eureka-client starter olduğu için eureka client otomatik devreye girer

@EnableDiscoveryClient ise eureka dışında herhangi bir mevcut client kütüphanesi için kullanılır

```
@EnableEurekaClient
@SpringBootApplication
public class ClientApplication {
    ...
}
```

- Client kendisini server'a tanıtırken kendisine ait **host, port ve health indicator URL bilgileri**ni iletir
- Server her bir client instance'ı **periyodik poll** eder
- Heart beat'i başarısız olanları bir süre sonra **deregister** eder

EurekaClient Arayüzü Üzerinden Servis Lookup

Service instance'ları arasından uygun olan bir tanesinin bilgilerini döner

Uygun service instance'ını tespit için round-robin algoritması kullanılır

```
@Autowired
private EurekaClient eurekaClient;

...

InstanceInfo info = eurekaClient.getNextServerFromEureka(
                                        "helloservice", false);

URI serviceUrl = URI.create("http://" + info.getHostName()
                                    + ":" + info.getPort() + "/hello");

restTemplate.getForObject(serviceUrl, String.class);
```

Feign ile Servis Erişimi

- Feign **dekleratif** biçimde **web servis client** oluşturmayı sağlar
- Servis çağrısı arka tarafta **Ribbon load balancer** kullanılarak gerçekleştirilir
- Servis instance'ı da **Eureka client** vasıtası ile bulunur

```
@FeignClient("helloservice")
public interface HelloServiceClient {

    @GetMapping("/hello")
    public String hello();

}
```

Feign ile Servis Erişimi

pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>
</dependencies>
```

Feign ile Servis Erişimi

```
@SpringBootApplication
@EnableFeignClients
public class ClientApplication {
...
}
```

Load Balanced RestTemplate Kullanımı

- **@LoadBalanced** anotasyonu ile bir RestTemplate bean'ı tanımlanarak da servislere erişmek mümkündür

```
@Configuration
public class ClientConfig {

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate(
                            RestTemplateBuilder restTemplateBuilder) {
        return restTemplateBuilder.build();
    }
}
```

Load Balanced RestTemplate Kullanımı

```
@Autowired
private RestTemplate restTemplate;

...

return restTemplate.getForObject(
                        "http://helloservice/hello", String.class);
```

URI içerisinde hostname yerine erişilecek servisin adı kullanılmalıdır

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com