Consul and Spring Boot

In this lesson, we'll discuss Consul and Spring boot.

WE'LL COVER THE FOLLOWING

Introduction

The Consul integration in Spring Boot is comparable to the integration of Eureka (see Eureka: Service Discovery).

There is a configuration file `application.properties`. Here is the relevant section:

```
spring.application.name=catalog
spring.cloud.consul.host=consul
spring.cloud.consul.port=8500
spring.cloud.consul.discovery.preferIpAddress=true
spring.cloud.consul.discovery.instanceId=\${spring.application.name}:
\${spring.application.instance_id:\${random.value}}
```

The section configures the following values:

- `spring.application.name` defines the name under which the application is registered in Consul.
- `spring.cloud.consul.host` and `spring.cloud.consul.port` determine at which port and on which host Consul can be accessed.
- Via `spring.cloud.consul.discovery.preferIpAddress` the services register

with their IP address and not with the host name. This circumvents problems that arise because host names cannot be resolved in the Docker environment.

- `spring.cloud.consul.discovery.instanceId` assigns an unambiguous ID to each microservice instance for discriminating between instances for load balancing.

Code dependencies

In addition, a dependency to `spring-cloud-starter-consul-discovery` has to be inserted in `pom.xml` for the build. Moreover, the main class of the application that has the annotation `@SpringBootApplication` has to be annotated with `@EnableDiscoveryClient`.

Health check with Spring Boot Actuator

Finally, the microservice must provide a health check. Spring Boot contains the Actuator module for this, which offers a health check as well as metrics. The health check is available at the URL `/health`. This is exactly the URL Consul requests. It is enough to add a dependency to `spring-boot-starter-actuator` in the `pom.xml`. Specific health checks may need to be developed if the application depends on additional resources.

Consul and Ribbon

Of course, a microservice must also use Consul to communicate with other microservices. For this, the Consul example uses the Ribbon library analogous to the Netflix example (see Load Balancing: Ribbon).

Ribbon has been modified in the Spring Cloud project so that it can also deal with Consul. Because the microservices use Ribbon, the rest of the code is unchanged compared to the Netflix example.

QUIZ

1 Which host can Consul be accessed on in the following configuration file?

Which host can Consul be accessed on in the following configuration file?

```
spring.application.name=catalog
spring.cloud.consul.host=consul_host
spring.cloud.consul.port=9500
spring.cloud.consul.discovery.preferIpAddress=true
spring.cloud.consul.discovery.instanceId=\${spring.applicatio
n.name}:
\${spring.application.instance_id:\${random.value}}
```


In the next lesson, we'll discuss DNS and registrator.