**Microservices with Spring Boot and Spring Cloud**

**Make your microservices cloud ready with Spring Cloud:**

* Establishing Communication between Microservices.
* Centralized Microservice Configuration with Spring Cloud Config Server.
* Using Spring Cloud Bus to exchange messages about Configuration updates.
* Simplify communication with other Microservices using Feign REST Client.
* Implement client side load balancing with Ribbon.
* Implement dynamic scaling using Eureka Naming Server and Ribbon.
* Implement API Gateway with Zuul.
* Implement Distributed tracing with Spring Cloud Sleuth and Zipkin.
* Implement Fault Tolerance with Zipkin.

## **Microservices Step by Step:**

#### **Spring Cloud Config Server**

* Step 00 - 04 - Introduction to Limits Microservice and Spring Cloud Config Server.
* Step 01 - Setting up Limits Microservice.
* Step 02 - Creating a hard coded limits service.
* Step 03 - Enhance limits service to pick up configuration from application properties.
* Step 04 - Setting up Spring Cloud Config Server.
* Step 05 - Installing Git.
* Step 06 - Creating Local Git Repository.
* Step 07 - Connect Spring Cloud Config Server to Local Git Repository.
* Step 08 - Configuration for Multiple Environments in Git Repository.
* Step 09 - Connect Limits Service to Spring Cloud Config Server.
* Step 10 - Configuring Profiles for Limits Service.
* Step 11 - A review of Spring Cloud Config Server.

#### **Implementing 2 Microservices with Eureka Naming Server, Ribbon and Feign**

* Step 12 - Introduction to Currency Conversion and Currency Exchange Microservices TODO.
* Step 13 - Setting up Currency Exchange Microservice.
* Step 14 - Create a simple hard coded currency exchange service.
* Step 15 - Setting up Dynamic Port in the the Response.
* Step 16 - Configure JPA and Initialized Data.
* Step 17 - Create a JPA Repository.
* Step 18 - Setting up Currency Conversion Microservice.
* Step 19 - Creating a service for currency conversion.
* Step 20 - Invoking Currency Exchange Microservice from Currency Conversion Microservice.
* Step 21 - Using Feign REST Client for Service Invocation.
* Step 22 - Setting up client side load balancing with Ribbon.
* Step 23 - Running client side load balancing with Ribbon.
* Step 24 - Understand the need for a Naming Server.
* Step 25 - Setting up Eureka Naming Server.
* Step 26 - Connecting Currency Conversion Microservice to Eureka.
* Step 27 - Connecting Currency Exchange Microservice to Eureka.
* Step 28 - Distributing calls using Eureka and Ribbon.
* Step 29 - A review of implementing Eureka, Ribbon and Feign.

#### **API Gateways and Distributed Tracing**

* Step 30 - Introduction to API Gateways.
* Step 31 - Setting up Zuul API Gateway.
* Step 32 - Implementing Zuul Logging Filter.
* Step 33 - Executing a request through Zuul API Gateway.
* Step 34 - Setting up Zuul API Gateway between microservice invocations.
* Step 35 - Introduction to Distributed Tracing.
* Step 36 - Implementing Spring Cloud Sleuth.
* Step 37 - Introduction to Distributed Tracing with Zipkin.
* Step 38 - Installing Rabbit MQ.
* Step 39 - Setting up Distributed Tracing with Zipkin.
* Step 40 - Connecting microservices to Zipkin.
* Step 41 - Using Zipkin UI Dashboard to trace requests.

#### **Spring Cloud Bus and Hysterix**

* Step 42 - Understanding the need for Spring Cloud Bus
* Step 43 - Implementing Spring Cloud Bus
* Step 44 - Fault Tolerance with Hystrix

**Application port:**

|  |  |
| --- | --- |
| Application | Port |
| Limits Service | 8080, 8081, ... |
| Spring Cloud Config Server | 8888 |
| Currency Exchange Service | 8000, 8001, 8002, .. |
| Currency Conversion Service | 8100, 8101, 8102, ... |
| Netflix Eureka Naming Server | 8761 |
| Netflix Zuul API Gateway Server | 8765 |
| Zipkin Distributed Tracing Server | 9411 |

## **Application URLs**

|  |  |
| --- | --- |
| Application | URL |
| Limits Service | <http://localhost:8080/limits> <http://localhost:8080/actuator/refresh> (POST) |
| Spring Cloud Config Server | <http://localhost:8888/limits-service/default> <http://localhost:8888/limits-service/dev> |
| Currency Converter Service - Direct Call | <http://localhost:8100/currency-converter/from/USD/to/INR/quantity/10> |
| Currency Converter Service - Feign | <http://localhost:8100/currency-converter-feign/from/EUR/to/INR/quantity/10000> |
| Currency Exchange Service | <http://localhost:8000/currency-exchange/from/EUR/to/INR> <http://localhost:8001/currency-exchange/from/USD/to/INR> |
| Eureka | <http://localhost:8761/> |
| Zuul - Currency Exchange & Exchange Services | <http://localhost:8765/currency-exchange-service/currency-exchange/from/EUR/to/INR> <http://localhost:8765/currency-conversion-service/currency-converter-feign/from/USD/to/INR/quantity/10> |
| Zipkin | <http://localhost:9411/zipkin/> |
| Spring Cloud Bus Refresh | <http://localhost:8080/actuator/bus-refresh> (POST) |

## **VM Argument**

-Dserver.port=8001

## **Spring Cloud Configuration**

spring.cloud.config.failFast = true

## **Command to run Zipkin**

java -jar zipkin-server-2.12.9-exec.jar

**Why the use dependency:**

**limits-service:**

* 1. spring-boot-starter-actuator (The next one which I would add in is actuator. Actuator provides monitoring and management around).
  2. spring-cloud-starter-config (SPRING CLOUD CONFIG Client, that connects to a spring Cloud config server to fetch the application's configuration).