readme.md

Microservices with Spring Boot and Spring Cloud

Make your microservices cloud ready with Spring Cloud

You will learn

- 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 Introduction

- Step 00 - 01 - Introduction to Microservices
- Step 00 - 02 - Challenges with Microservices
- Step 00 - 03 - Microservice Solutions REMOVE THIS
- Step 00 - 04 - Introduction to Spring Cloud TODO

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

Ports

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

URLs

Application	URL
Limits Service	http://localhost:8080/limits POST -> http://localhost:8080/actuator/refresh
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

Application	URL
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/bus/refresh

Zipkin Installation

Quick Start Page

- https://zipkin.io/pages/quickstart

Downloading Zipkin Jar

- https://search.maven.org/remote_content?g=io.zipkin.java&a=zipkin-server&v=LATEST&c=exec

Command to run

```
RABBIT_URI=amqp://localhost java -jar zipkin-server-2.5.2-exec.jar
```

VM Argument

-Dserver.port=8001

Commands

```
mkdir git-configuration-repo
cd git-configuration-repo/
git init
git add -A
git commit -m "first commit"
```

Spring Cloud Configuration

```
spring.cloud.config.failFast=true
```

More Reading about Microservices

- Design and Governance of Microservices
  - https://martinfowler.com/microservices/
- 12 Factor App
  - https://12factor.net/
  - https://dzone.com/articles/the-12-factor-app-a-java-developers-perspective
- Spring Cloud
  - http://projects.spring.io/spring-cloud/

Complete Code Example

/03.microservices/currency-conversion-service/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.in28minutes.microservices</groupId>
        <artifactId>currency-conversion-service</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>currency-conversion-service</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
                <spring-cloud.version>Finchley.M8</spring-cloud.version>
        </properties>

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

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

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-sleuth</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-sleuth-zipkin</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-bus-amqp</artifactId>
                </dependency>

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

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <dependencyManagement>
```

```
			<dependencies>
				<dependency>
					<groupId>org.springframework.cloud</groupId>
					<artifactId>spring-cloud-dependencies</artifactId>
					<version>${spring-cloud.version}</version>
					<type>pom</type>
					<scope>import</scope>
				</dependency>
			</dependencies>
		</dependencyManagement>

		<build>
			<plugins>
				<plugin>
					<groupId>org.springframework.boot</groupId>
					<artifactId>spring-boot-maven-plugin</artifactId>
				</plugin>
			</plugins>
		</build>

		<repositories>
			<repository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</repository>
			<repository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</repository>
		</repositories>

		<pluginRepositories>
			<pluginRepository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</pluginRepository>
			<pluginRepository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</pluginRepository>
		</pluginRepositories>


	</project>
```

/03.microservices/currency-conversion-service/src/main/java/com/in28minutes/microservices/currencyconversionservice/CurrencyConversionBean.java

```
package com.in28minutes.microservices.currencyconversionservice;

import java.math.BigDecimal;

public class CurrencyConversionBean {
	private Long id;
	private String from;
	private String to;
	private BigDecimal conversionMultiple;
	private BigDecimal quantity;
	private BigDecimal totalCalculatedAmount;
	private int port;

	public CurrencyConversionBean() {

	}

	public CurrencyConversionBean(Long id, String from, String to, BigDecimal conversionMultiple, BigDecimal quant
			BigDecimal totalCalculatedAmount, int port) {
		super();
		this.id = id;
		this.from = from;
		this.to = to;
		this.conversionMultiple = conversionMultiple;
		this.quantity = quantity;
		this.totalCalculatedAmount = totalCalculatedAmount;
		this.port = port;
	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getFrom() {
		return from;
	}

	public void setFrom(String from) {
		this.from = from;
	}

	public String getTo() {
		return to;
	}

	public void setTo(String to) {
		this.to = to;
	}

	public BigDecimal getConversionMultiple() {
		return conversionMultiple;
	}

	public void setConversionMultiple(BigDecimal conversionMultiple) {
		this.conversionMultiple = conversionMultiple;
	}

	public BigDecimal getQuantity() {
		return quantity;
	}

	public void setQuantity(BigDecimal quantity) {
		this.quantity = quantity;
	}

	public BigDecimal getTotalCalculatedAmount() {
		return totalCalculatedAmount;
	}

	public void setTotalCalculatedAmount(BigDecimal totalCalculatedAmount) {
		this.totalCalculatedAmount = totalCalculatedAmount;
	}

	public int getPort() {
		return port;
	}

	public void setPort(int port) {
		this.port = port;
	}
```

```
}
```

/03.microservices/currency-conversion-service/src/main/java/com/in28minutes/microservices/currencyconversionservice/CurrencyConversionController.java

```java
package com.in28minutes.microservices.currencyconversionservice;

import java.math.BigDecimal;
import java.util.HashMap;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class CurrencyConversionController {

        private Logger logger = LoggerFactory.getLogger(this.getClass());

        @Autowired
        private CurrencyExchangeServiceProxy proxy;

        @GetMapping("/currency-converter/from/{from}/to/{to}/quantity/{quantity}")
        public CurrencyConversionBean convertCurrency(@PathVariable String from, @PathVariable String to,
                        @PathVariable BigDecimal quantity) {

                // Feign - Problem 1
                Map<String, String> uriVariables = new HashMap<>();
                uriVariables.put("from", from);
                uriVariables.put("to", to);

                ResponseEntity<CurrencyConversionBean> responseEntity = new RestTemplate().getForEntity(
                                "http://localhost:8000/currency-exchange/from/{from}/to/{to}", CurrencyConversionBean.
                                uriVariables);

                CurrencyConversionBean response = responseEntity.getBody();

                return new CurrencyConversionBean(response.getId(), from, to, response.getConversionMultiple(), quanti
                                quantity.multiply(response.getConversionMultiple()), response.getPort());
        }

        @GetMapping("/currency-converter-feign/from/{from}/to/{to}/quantity/{quantity}")
        public CurrencyConversionBean convertCurrencyFeign(@PathVariable String from, @PathVariable String to,
                        @PathVariable BigDecimal quantity) {

                CurrencyConversionBean response = proxy.retrieveExchangeValue(from, to);

                logger.info("{}", response);

                return new CurrencyConversionBean(response.getId(), from, to, response.getConversionMultiple(), quanti
                                quantity.multiply(response.getConversionMultiple()), response.getPort());
        }

}
```

/03.microservices/currency-conversion-service/src/main/java/com/in28minutes/microservices/currencyconversionservice/CurrencyConversionServiceApplication.java

```
package com.in28minutes.microservices.currencyconversionservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;

import brave.sampler.Sampler;

@SpringBootApplication
@EnableFeignClients("com.in28minutes.microservices.currencyconversionservice")
@EnableDiscoveryClient
public class CurrencyConversionServiceApplication {

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

        @Bean
        public Sampler defaultSampler() {
                return Sampler.ALWAYS_SAMPLE;
        }

}
```

/03.microservices/currency-conversion-service/src/main/java/com/in28minutes/microservices/currencyconversionservice/CurrencyExchangeServiceProxy.java

```
package com.in28minutes.microservices.currencyconversionservice;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.cloud.netflix.ribbon.RibbonClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

//@FeignClient(name="currency-exchange-service", url="localhost:8000")
//@FeignClient(name="currency-exchange-service")
@FeignClient(name="netflix-zuul-api-gateway-server")
@RibbonClient(name="currency-exchange-service")
public interface CurrencyExchangeServiceProxy {
        //@GetMapping("/currency-exchange/from/{from}/to/{to}")
        @GetMapping("/currency-exchange-service/currency-exchange/from/{from}/to/{to}")
        public CurrencyConversionBean retrieveExchangeValue
                (@PathVariable("from") String from, @PathVariable("to") String to);
}
```

/03.microservices/currency-conversion-service/src/main/resources/application.properties

```
spring.application.name=currency-conversion-service
server.port=8100
eureka.client.service-url.default-zone=http://localhost:8761/eureka
#currency-exchange-service.ribbon.listOfServers=http://localhost:8000,http://localhost:8001
```

/03.microservices/currency-conversion-service/src/test/java/com/in28minutes/microservices/currencyconversionservice/CurrencyConversionServiceApplicationTests.java

```
package com.in28minutes.microservices.currencyconversionservice;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class CurrencyConversionServiceApplicationTests {

        @Test
        public void contextLoads() {
        }

}
```

/03.microservices/currency-exchange-service/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.in28minutes.microservices</groupId>
        <artifactId>currency-exchange-service</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>currency-exchange-service</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
                <spring-cloud.version>Finchley.M8</spring-cloud.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-config</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-sleuth</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-sleuth-zipkin</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-bus-amqp</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
                <dependency>
                        <groupId>com.h2database</groupId>
```

```
				<artifactId>h2</artifactId>
			</dependency>


			<dependency>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-starter-test</artifactId>
				<scope>test</scope>
			</dependency>
		</dependencies>

		<dependencyManagement>
			<dependencies>
				<dependency>
					<groupId>org.springframework.cloud</groupId>
					<artifactId>spring-cloud-dependencies</artifactId>
					<version>${spring-cloud.version}</version>
					<type>pom</type>
					<scope>import</scope>
				</dependency>
			</dependencies>
		</dependencyManagement>

		<build>
			<plugins>
				<plugin>
					<groupId>org.springframework.boot</groupId>
					<artifactId>spring-boot-maven-plugin</artifactId>
				</plugin>
			</plugins>
		</build>

		<repositories>
			<repository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</repository>
			<repository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</repository>
		</repositories>

		<pluginRepositories>
			<pluginRepository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</pluginRepository>
			<pluginRepository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</pluginRepository>
		</pluginRepositories>


	</project>
```

/03.microservices/currency-exchange-service/src/main/java/com/in28minutes/microservices/currencyexchangeservice/CurrencyExchangeController.java

```java
package com.in28minutes.microservices.currencyexchangeservice;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.env.Environment;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class CurrencyExchangeController {

	private Logger logger = LoggerFactory.getLogger(this.getClass());

	@Autowired
	private Environment environment;

	@Autowired
	private ExchangeValueRepository repository;

	@GetMapping("/currency-exchange/from/{from}/to/{to}")
	public ExchangeValue retrieveExchangeValue
		(@PathVariable String from, @PathVariable String to){

		ExchangeValue exchangeValue =
				repository.findByFromAndTo(from, to);

		exchangeValue.setPort(
				Integer.parseInt(environment.getProperty("local.server.port")));

		logger.info("{}", exchangeValue);

		return exchangeValue;
	}
}
```

/03.microservices/currency-exchange-service/src/main/java/com/in28minutes/microservices/currencyexchangeservice/CurrencyExchangeServiceApplication.java

```java
package com.in28minutes.microservices.currencyexchangeservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.context.annotation.Bean;

import brave.sampler.Sampler;

@SpringBootApplication
@EnableDiscoveryClient
public class CurrencyExchangeServiceApplication {

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

	@Bean
	public Sampler defaultSampler(){
		return Sampler.ALWAYS_SAMPLE;
	}

}
```

/03.microservices/currency-exchange-service/src/main/java/com/in28minutes/microservices/currencyexchangeservice/ExchangeValue.java

```java
package com.in28minutes.microservices.currencyexchangeservice;

import java.math.BigDecimal;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class ExchangeValue {

	@Id
	private Long id;

	@Column(name="currency_from")
	private String from;

	@Column(name="currency_to")
	private String to;

	private BigDecimal conversionMultiple;
	private int port;

	public ExchangeValue() {

	}


	public ExchangeValue(Long id, String from, String to, BigDecimal conversionMultiple) {
		super();
		this.id = id;
		this.from = from;
		this.to = to;
		this.conversionMultiple = conversionMultiple;
	}

	public Long getId() {
		return id;
	}

	public String getFrom() {
		return from;
	}

	public String getTo() {
		return to;
	}

	public BigDecimal getConversionMultiple() {
		return conversionMultiple;
	}

	public int getPort() {
		return port;
	}

	public void setPort(int port) {
		this.port = port;
	}

}
```

/03.microservices/currency-exchange-service/src/main/java/com/in28minutes/microservices/currencyexchangeservice/ExchangeValueRepository.java

```
package com.in28minutes.microservices.currencyexchangeservice;

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

public interface ExchangeValueRepository extends
                JpaRepository<ExchangeValue, Long>{
        ExchangeValue findByFromAndTo(String from, String to);
}
```

/03.microservices/currency-exchange-service/src/main/resources/application.properties

```
spring.application.name=currency-exchange-service
server.port=8000

spring.jpa.show-sql=true
spring.h2.console.enabled=true

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

/03.microservices/currency-exchange-service/src/main/resources/data.sql

```
insert into exchange_value(id,currency_from,currency_to,conversion_multiple,port)
values(10001,'USD','INR',65,0);
insert into exchange_value(id,currency_from,currency_to,conversion_multiple,port)
values(10002,'EUR','INR',75,0);
insert into exchange_value(id,currency_from,currency_to,conversion_multiple,port)
values(10003,'AUD','INR',25,0);
```

/03.microservices/currency-exchange-service/src/test/java/com/in28minutes/microservices/currencyexchangeservice/CurrencyExchangeServiceApplicationTests.java

```
package com.in28minutes.microservices.currencyexchangeservice;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class CurrencyExchangeServiceApplicationTests {

        @Test
        public void contextLoads() {
        }

}
```

/03.microservices/git-localconfig-repo/limits-service-dev.properties

```
limits-service.minimum=1
```

/03.microservices/git-localconfig-repo/limits-service-qa.properties

```
limits-service.minimum=2
limits-service.maximum=222
```

/03.microservices/git-localconfig-repo/limits-service.properties

```
limits-service.minimum=8
limits-service.maximum=888
management.endpoints.web.exposure.include=*
```

/03.microservices/limits-service/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.in28minutes.microservices</groupId>
        <artifactId>limits-service</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>limits-service</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
                <spring-cloud.version>Finchley.M8</spring-cloud.version>
        </properties>

        <dependencies>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-config</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-bus-amqp</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <dependencyManagement>
                <dependencies>
                        <dependency>
                                <groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-dependencies</artifactId>
                                <version>${spring-cloud.version}</version>
                                <type>pom</type>
                        <scope>import</scope>
                    </dependency>
                </dependencies>
        </dependencyManagement>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

        <repositories>
                <repository>
                        <id>spring-snapshots</id>
                        <name>Spring Snapshots</name>
                        <url>https://repo.spring.io/snapshot</url>
                        <snapshots>
                                <enabled>true</enabled>
                        </snapshots>
                </repository>
                <repository>
                        <id>spring-milestones</id>
                        <name>Spring Milestones</name>
                        <url>https://repo.spring.io/milestone</url>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
        </repositories>

        <pluginRepositories>
                <pluginRepository>
                        <id>spring-snapshots</id>
                        <name>Spring Snapshots</name>
                        <url>https://repo.spring.io/snapshot</url>
                        <snapshots>
                                <enabled>true</enabled>
                        </snapshots>
                </pluginRepository>
                <pluginRepository>
                        <id>spring-milestones</id>
                        <name>Spring Milestones</name>
                        <url>https://repo.spring.io/milestone</url>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </pluginRepository>
        </pluginRepositories>


</project>
```

/03.microservices/limits-service/src/main/java/com/in28minutes/microservices/limitsservice/bean/LimitConfiguration.java

```
package com.in28minutes.microservices.limitsservice.bean;

public class LimitConfiguration {
        private int maximum;
        private int minimum;

        protected LimitConfiguration() {

        }

        public LimitConfiguration(int maximum, int minimum) {
                super();
                this.maximum = maximum;
                this.minimum = minimum;
        }
```

```java
	public int getMaximum() {
		return maximum;
	}

	public int getMinimum() {
		return minimum;
	}

}
```

/03.microservices/limits-service/src/main/java/com/in28minutes/microservices/limitsservice/Configuration.java

```java
package com.in28minutes.microservices.limitsservice;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Component
@ConfigurationProperties("limits-service")
public class Configuration {

	private int minimum;
	private int maximum;

	public void setMinimum(int minimum) {
		this.minimum = minimum;
	}

	public void setMaximum(int maximum) {
		this.maximum = maximum;
	}

	public int getMinimum() {
		return minimum;
	}

	public int getMaximum() {
		return maximum;
	}

}
```

/03.microservices/limits-service/src/main/java/com/in28minutes/microservices/limitsservice/LimitsConfigurationController.java

```java
package com.in28minutes.microservices.limitsservice;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.in28minutes.microservices.limitsservice.bean.LimitConfiguration;
import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;

@RestController
public class LimitsConfigurationController {

	@Autowired
	private Configuration configuration;

	@GetMapping("/limits")
	public LimitConfiguration retrieveLimitsFromConfigurations() {
		LimitConfiguration limitConfiguration = new LimitConfiguration(configuration.getMaximum(),
				configuration.getMinimum());
		return limitConfiguration;
	}
	@GetMapping("/fault-tolerance-example")
	@HystrixCommand(fallbackMethod="fallbackRetrieveConfiguration")
	public LimitConfiguration retrieveConfiguration() {
		throw new RuntimeException("Not available");
	}

	public LimitConfiguration fallbackRetrieveConfiguration() {
		return new LimitConfiguration(999, 9);
	}

}
```

/03.microservices/limits-service/src/main/java/com/in28minutes/microservices/limitsservice/LimitsServiceApplication.java

```
package com.in28minutes.microservices.limitsservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.hystrix.EnableHystrix;

@SpringBootApplication
@EnableHystrix
public class LimitsServiceApplication {
	public static void main(String[] args) {
		SpringApplication.run(LimitsServiceApplication.class, args);
	}
}
```

/03.microservices/limits-service/src/main/resources/bootstrap.properties

```
spring.application.name=limits-service
spring.cloud.config.uri=http://localhost:8888
spring.profiles.active=qa
management.endpoints.web.exposure.include=*
```

/03.microservices/limits-service/src/test/java/com/in28minutes/microservices/limitsservice/LimitsServiceApplicationTests.java

```
package com.in28minutes.microservices.limitsservice;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class LimitsServiceApplicationTests {

	@Test
	public void contextLoads() {
	}

}
```

/03.microservices/netflix-eureka-naming-server/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>com.in28minutes.microservices</groupId>
	<artifactId>netflix-eureka-naming-server</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>netflix-eureka-naming-server</name>
	<description>Demo project for Spring Boot</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.0.0.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
		<java.version>1.8</java.version>
		<spring-cloud.version>Finchley.M8</spring-cloud.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-config</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<dependencyManagement>
		<dependencies>
			<dependency>
				<groupId>org.springframework.cloud</groupId>
				<artifactId>spring-cloud-dependencies</artifactId>
				<version>${spring-cloud.version}</version>
				<type>pom</type>
				<scope>import</scope>
			</dependency>
		</dependencies>
	</dependencyManagement>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

	<repositories>
		<repository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</repository>
		<repository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
				</snapshots>
			</repository>
		</repositories>

		<pluginRepositories>
			<pluginRepository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</pluginRepository>
			<pluginRepository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</pluginRepository>
		</pluginRepositories>


	</project>
```

/03.microservices/netflix-eureka-naming-server/src/main/java/com/in28minutes/microservices/netflixeurekanamingserver/NetflixEurekaNamingServerApplication.java

```java
package com.in28minutes.microservices.netflixeurekanamingserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class NetflixEurekaNamingServerApplication {

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

/03.microservices/netflix-eureka-naming-server/src/main/resources/application.properties

```
spring.application.name=netflix-eureka-naming-server
server.port=8761

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

/03.microservices/netflix-eureka-naming-server/src/test/java/com/in28minutes/microservices/netflixeurekanamingserver/NetflixEurekaNamingServerApplicationTests.java

```java
package com.in28minutes.microservices.netflixeurekanamingserver;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
```

```java
@SpringBootTest
public class NetflixEurekaNamingServerApplicationTests {

	@Test
	public void contextLoads() {
	}

}
```

/03.microservices/netflix-zuul-api-gateway-server/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>com.in28minutes.microservices</groupId>
	<artifactId>netflix-zuul-api-gateway-server</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>netflix-zuul-api-gateway-server</name>
	<description>Demo project for Spring Boot</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.0.0.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
		<java.version>1.8</java.version>
		<spring-cloud.version>Finchley.M8</spring-cloud.version>
	</properties>

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

		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-zuul</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-sleuth</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-sleuth-zipkin</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-bus-amqp</artifactId>
		</dependency>


		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<dependencyManagement>
		<dependencies>
```

```
				<dependency>
					<groupId>org.springframework.cloud</groupId>
					<artifactId>spring-cloud-dependencies</artifactId>
					<version>${spring-cloud.version}</version>
					<type>pom</type>
					<scope>import</scope>
				</dependency>
			</dependencies>
		</dependencyManagement>

		<build>
			<plugins>
				<plugin>
					<groupId>org.springframework.boot</groupId>
					<artifactId>spring-boot-maven-plugin</artifactId>
				</plugin>
			</plugins>
		</build>

		<repositories>
			<repository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</repository>
			<repository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</repository>
		</repositories>

		<pluginRepositories>
			<pluginRepository>
				<id>spring-snapshots</id>
				<name>Spring Snapshots</name>
				<url>https://repo.spring.io/snapshot</url>
				<snapshots>
					<enabled>true</enabled>
				</snapshots>
			</pluginRepository>
			<pluginRepository>
				<id>spring-milestones</id>
				<name>Spring Milestones</name>
				<url>https://repo.spring.io/milestone</url>
				<snapshots>
					<enabled>false</enabled>
				</snapshots>
			</pluginRepository>
		</pluginRepositories>


	</project>
```

/03.microservices/netflix-zuul-api-gateway-server/src/main/java/com/in28minutes/microservices/netflixzuulapigatewayserver/NetflixZuulApiGatewayServerApplication.java

```
package com.in28minutes.microservices.netflixzuulapigatewayserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;
import org.springframework.context.annotation.Bean;

import brave.sampler.Sampler;
```

```java
@EnableZuulProxy
@EnableDiscoveryClient
@SpringBootApplication
public class NetflixZuulApiGatewayServerApplication {

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

        @Bean
        public Sampler defaultSampler(){
                return Sampler.ALWAYS_SAMPLE;
        }
}
```

/03.microservices/netflix-zuul-api-gateway-server/src/main/java/com/in28minutes/microservices/netflixzuulapigatewayserver/ZuulLoggingFilter.java

```java
package com.in28minutes.microservices.netflixzuulapigatewayserver;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;

@Component
public class ZuulLoggingFilter extends ZuulFilter{

        private Logger logger = LoggerFactory.getLogger(this.getClass());

        @Override
        public boolean shouldFilter() {
                return true;
        }

        @Override
        public Object run() {
                HttpServletRequest request =
                                RequestContext.getCurrentContext().getRequest();
                logger.info("request -> {} request uri -> {}",
                                request, request.getRequestURI());
                return null;
        }

        @Override
        public String filterType() {
                return "pre";
        }

        @Override
        public int filterOrder() {
                return 1;
        }
}
```

/03.microservices/netflix-zuul-api-gateway-server/src/main/resources/application.properties

```
spring.application.name=netflix-zuul-api-gateway-server
server.port=8765
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

/03.microservices/netflix-zuul-api-gateway-server/src/test/java/com/in28minutes/microservices/netflixzuulapigatewayserver/NetflixZuulApiGatewayServerApplicationTests.java

```
package com.in28minutes.microservices.netflixzuulapigatewayserver;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class NetflixZuulApiGatewayServerApplicationTests {

        @Test
        public void contextLoads() {
        }

}
```

/03.microservices/spring-cloud-config-server/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.in28minutes.microservices</groupId>
        <artifactId>spring-cloud-config-server</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>spring-cloud-config-server</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
                <spring-cloud.version>Finchley.M8</spring-cloud.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-config-server</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-bus-amqp</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <dependencyManagement>
                <dependencies>
                        <dependency>
                                <groupId>org.springframework.cloud</groupId>
```

```
				<artifactId>spring-cloud-dependencies</artifactId>
				<version>${spring-cloud.version}</version>
				<type>pom</type>
				<scope>import</scope>
			</dependency>
		</dependencies>
	</dependencyManagement>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

	<repositories>
		<repository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</repository>
		<repository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</repository>
	</repositories>

	<pluginRepositories>
		<pluginRepository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</pluginRepository>
		<pluginRepository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</pluginRepository>
	</pluginRepositories>


</project>
```

/03.microservices/spring-cloud-config-server/src/main/java/com/in28minutes/microservices/springcloudconfigserver/SpringCloudConfigServerApplication.java

```java
package com.in28minutes.microservices.springcloudconfigserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@EnableConfigServer
@SpringBootApplication
public class SpringCloudConfigServerApplication {

	public static void main(String[] args) {
```

```
                SpringApplication.run(SpringCloudConfigServerApplication.class, args);
        }
}
```

/03.microservices/spring-cloud-config-server/src/main/resources/application.properties

```
spring.application.name=spring-cloud-config-server
server.port=8888
spring.cloud.config.server.git.uri=file:///in28Minutes/git/spring-micro-services/03.microservices/git-localconfig-repo
```

/03.microservices/spring-cloud-config-server/src/test/java/com/in28minutes/microservices/springcloudconfigserver/SpringCloudConfigServerApplicationTests.java

```
package com.in28minutes.microservices.springcloudconfigserver;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringCloudConfigServerApplicationTests {

        @Test
        public void contextLoads() {
        }

}
```

/03.microservices/zipkin-distributed-tracing-server/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.in28minutes.microservices</groupId>
        <artifactId>zipkin-distributed-tracing-server</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>zipkin-distributed-tracing-server</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
                <spring-cloud.version>Finchley.M8</spring-cloud.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-sleuth-zipkin-stream</artifactId>
                </dependency>
```

```
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-stream-rabbit</artifactId>
		</dependency>

		<dependency>
			<groupId>io.zipkin.java</groupId>
			<artifactId>zipkin-autoconfigure-ui</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<dependencyManagement>
		<dependencies>
			<dependency>
				<groupId>org.springframework.cloud</groupId>
				<artifactId>spring-cloud-dependencies</artifactId>
				<version>${spring-cloud.version}</version>
				<type>pom</type>
				<scope>import</scope>
			</dependency>
		</dependencies>
	</dependencyManagement>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

	<repositories>
		<repository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</repository>
		<repository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</repository>
	</repositories>

	<pluginRepositories>
		<pluginRepository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</pluginRepository>
		<pluginRepository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</pluginRepository>
	</pluginRepositories>
```

```
</project>
```

/03.microservices/zipkin-distributed-tracing-server/src/main/java/com/in28minutes/microservices/zipkindistributedtracingserver/ZipkinDistributedTracingServerApplication.java

```
package com.in28minutes.microservices.zipkindistributedtracingserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import zipkin.server.EnableZipkinServer;

@EnableZipkinServer
@SpringBootApplication
public class ZipkinDistributedTracingServerApplication {

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

/03.microservices/zipkin-distributed-tracing-server/src/main/resources/application.properties

```
spring.application.name=zipkin-distributed-tracing-server
server.port=9411
```

/03.microservices/zipkin-distributed-tracing-server/src/test/java/com/in28minutes/microservices/zipkindistributedtracingserver/ZipkinDistributedTracingServerApplicationTests.java

```
package com.in28minutes.microservices.zipkindistributedtracingserver;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class ZipkinDistributedTracingServerApplicationTests {

        @Test
        public void contextLoads() {
        }

}
```

TODO - Microservices Demo

- Step 91 - Spring Cloud Config Server Demo - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step11.zip
- Step 92 - Two Microservices - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step20.zip
- Step 93 - Auto Scaling with Eureka Naming Server, Ribbon and Feign - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step29.zip
- Step 94 - Zuul API Gateway Demo - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step34.zip

- Step 95 - Distributed Tracing with Zipkin - Demo - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step41.zip
- Step 96 - Spring Cloud Bus Demo - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step43.zip
- Step 97 - Fault Tolerance with Hysterix Demo - https://github.com/in28minutes/spring-microservices/blob/master/03.microservices/step44.zip