IN28MINUTES

Microservices with Spring Boot - Part 5 - Using Eureka Naming Server

Let's learn the basics of microservices and microservices architectures. We will also start looking at a basic implementation of a microservice with Spring Boot. We will create a couple of microservices and get them to talk to each other using Eureka Naming Server and Ribbon for Client Side Load Balancing.

Here is the Microservice Series Outline: Microservices with Spring Boot

- Part 1 - Getting Started with Microservices Architecture
- Part 2 - Creating Forex Microservice
- Part 3 - Creating Currency Conversion Microservice
- Part 4 - Using Ribbon for Load Balancing
- Current Part - Part 5 - Using Eureka Naming Server

This is part 5 of this series. In this part, we will focus on enabling Eureka Naming Server and have the microservices communicate with it.

You will learn

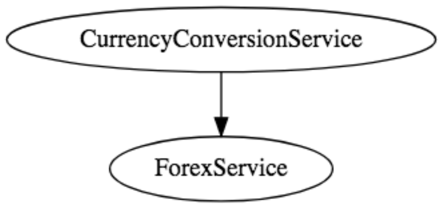
- What is the need for Naming Server?
- What is Eureka?
- How does Naming Server enable location transparancy between microservices?

10 Step Reference Courses

- Spring Framework for Beginners in 10 Steps
- Spring Boot for Beginners in 10 Steps
- Spring MVC in 10 Steps
- JPA and Hibernate in 10 Steps
- Eclipse Tutorial for Beginners in 5 Steps
- Maven Tutorial for Beginners in 5 Steps
- JUnit Tutorial for Beginners in 5 Steps
- Mockito Tutorial for Beginners in 5 Steps
- Complete in28Minutes Course Guide

Microservices Overview

In Parts 2 & 3, we created two microservices and established communication between them.



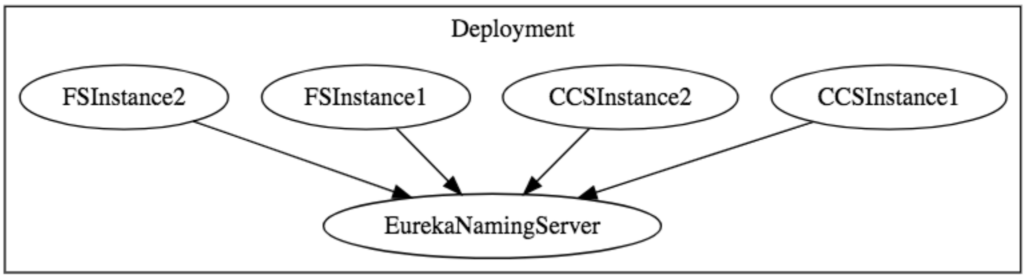
In Part 4, we used Ribbon to distribute load between the two instances of Forex Service.

However, we are hardcoding the urls of both instances of Forex Service in CCS.

```
forex-service.ribbon.listOfServers=localhost:8000,localhost:8001
```

That means every time there is a new instance of FS, we would need to change the configuration of CCS. Thats not cool.

In this part, we will use Eureka Naming Server to fix this problem.



Tools you will need

- Maven 3.0+ is your build tool
- Your favorite IDE. We use Eclipse.
- JDK 1.8+

Complete Maven Project With Code Examples

> *Our Github repository has all the code examples - https://github.com/in28minutes/spring-boot-examples/tree/master/spring-boot-basic-microservice*

Bootstrapping Eureka Naming Server with Spring Initializr

Creating Eureka Naming Server with Spring Initializr is a cake walk.

Spring Initializr http://start.spring.io/ is great tool to bootstrap your Spring Boot projects.

You can create a wide variety of projects using Spring Initializr.

Following steps have to be done for a Web Services project

- Launch Spring Initializr and choose the following
  - Choose `com.in28minutes.springboot.microservice.eureka.naming.server` as Group
  - Choose `spring-boot-microservice-eureka-naming-server` as Artifact
  - Choose following dependencies
    - Eureka
    - DevTools
- Click Generate Project.
- Import the project into Eclipse. File -> Import -> Existing Maven Project.

> *Do not forget to choose Eureka in the dependencies*

Enabling Eureka

EnableEurekaServer in `SpringBootMicroserviceEurekaNamingServerApplication`.

```
@SpringBootApplication
@EnableEurekaServer
public class SpringBootMicroserviceEurekaNamingServerApplication {
```

Configure the application name and port for the Eureka Server

/spring-boot-microservice-eureka-naming-server/src/main/resources/application.properties

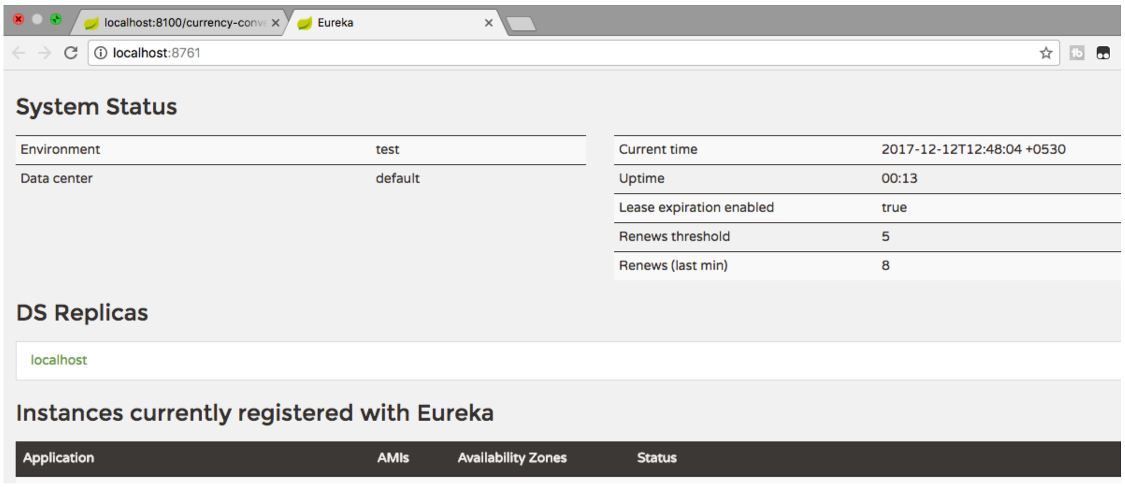
```
spring.application.name=netflix-eureka-naming-server
server.port=8761

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

Launching Eureka Naming Server

Launch `SpringBootMicroserviceEurekaNamingServerApplication` as a Java application.

You can launch up Eureka at `http://localhost:8761`

You would see that there are no instances yet connect to Eureka.



System Status

Environment	test
Data center	default

Current time	2017-12-12T12:48:04 +0530
Uptime	00:13
Lease expiration enabled	true
Renews threshold	5
Renews (last min)	8

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status

Connect FS and CCS Microservices with Eureka

Make these changes on both the microservices

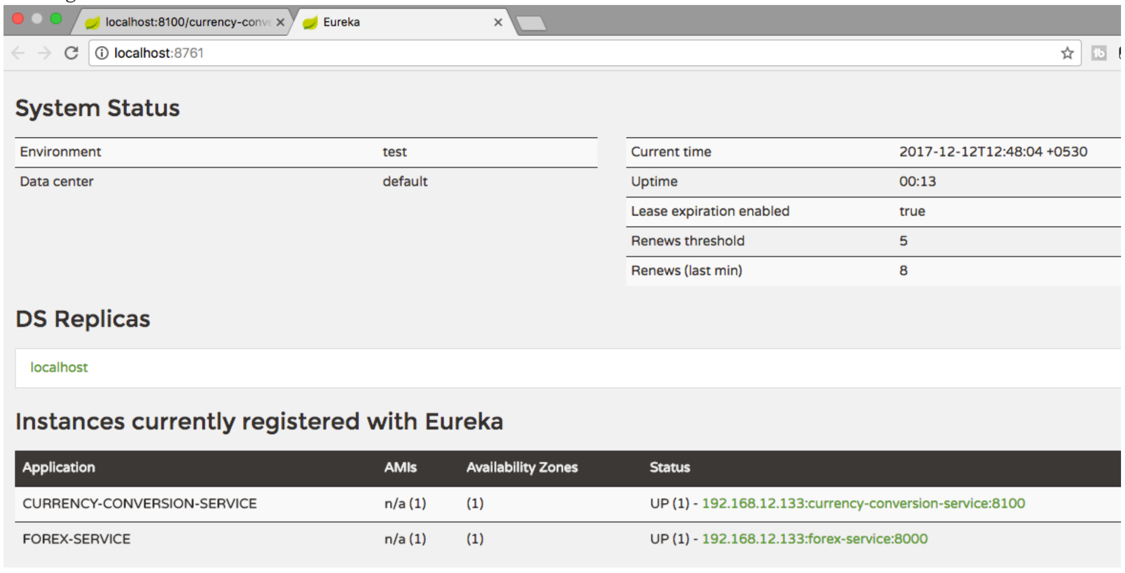
Add to pom.xml

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

Configure Eureka URL in application.properties

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

Restart all the instances of CCS and FS. You would see that the CCS and FS microservices are registered with Eureka Naming Server. Thats Cool!



System Status

Environment	test
Data center	default

Current time	2017-12-12T12:48:04 +0530
Uptime	00:13
Lease expiration enabled	true
Renews threshold	5
Renews (last min)	8

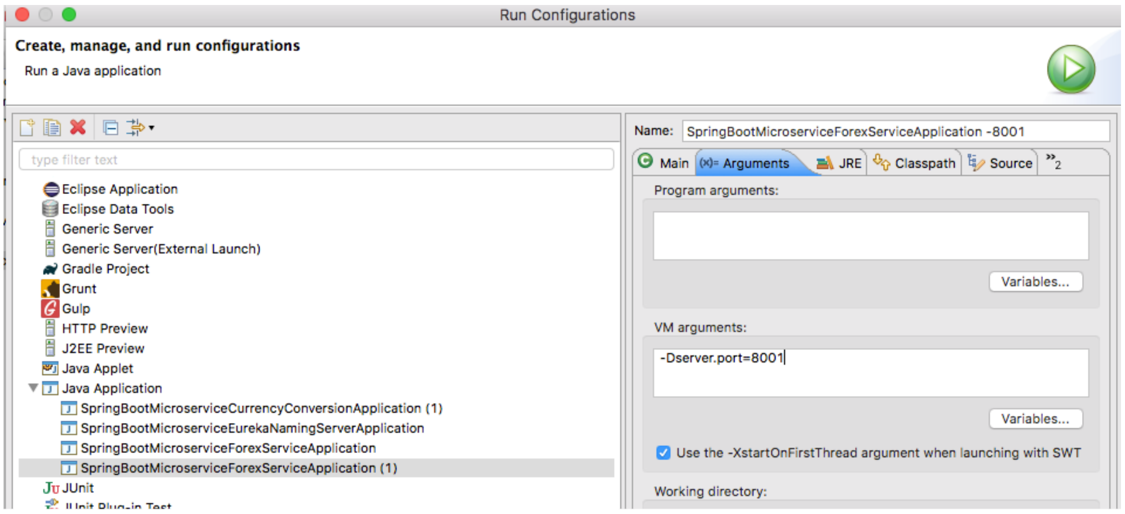
DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CURRENCY-CONVERSION-SERVICE	n/a (1)	(1)	UP (1) - 192.168.12.133:currency-conversion-service:8100
FOREX-SERVICE	n/a (1)	(1)	UP (1) - 192.168.12.133:forex-service:8000

Screenshot shows how to launch an additional instance of Forex Service on 8081.

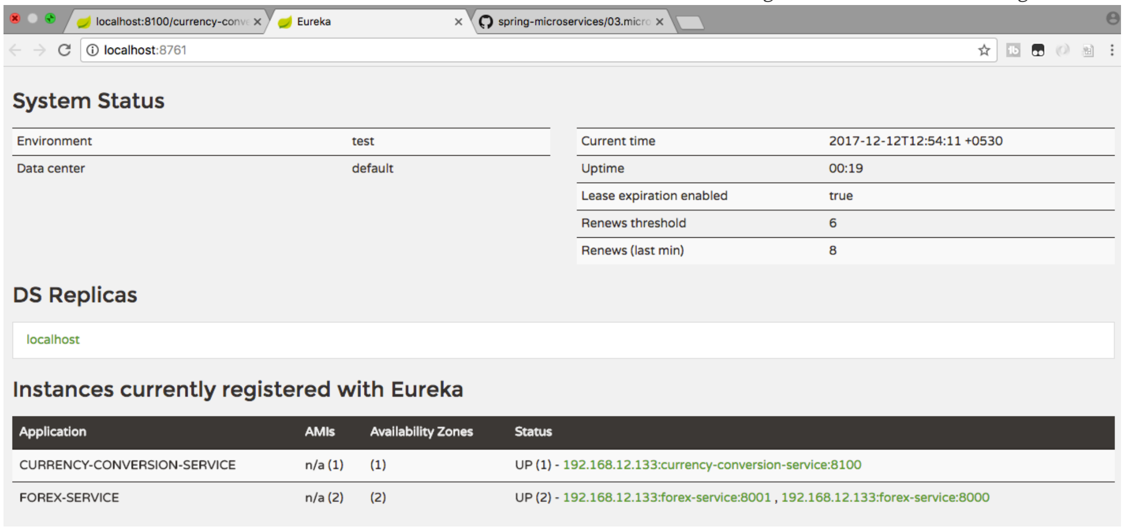


Name: SpringBootMicroserviceForexServiceApplication -8001

VM arguments: -Dserver.port=8001

☑ Use the -XstartOnFirstThread argument when launching with SWT

You would see that one instance of CCS and two instances of FS microservices are registered with Eureka Naming Server.



System Status

Environment	test
Data center	default

Current time	2017-12-12T12:54:11 +0530
Uptime	00:19
Lease expiration enabled	true
Renews threshold	6
Renews (last min)	8

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CURRENCY-CONVERSION-SERVICE	n/a (1)	(1)	UP (1) - 192.168.12.133:currency-conversion-service:8100
FOREX-SERVICE	n/a (2)	(2)	UP (2) - 192.168.12.133:forex-service:8001 , 192.168.12.133:forex-service:8000

Routing Ribbon Requests Through Eureka

All that you would need to do is to remove this configuration

Remove this configuration in application.properties

```
forex-service.ribbon.listOfServers=localhost:8000,localhost:8001
```

Restart the CCS instance.

Eureka in Action

Currently we have the following services up and running

- Currency Conversion Micro Service (CCS) on 8100
- Two instances of Forex MicroService on 8000 and 8001
- Eureka Server launched

Now you would see that the requests to CCS would get distributed between the two instances of Forex Microservice by Ribbon through Eureka.

Request 1

GET to http://localhost:8100/currency-converter-feign/from/EUR/to/INR/quantity/10000

```
{
  id: 10002,
  from: "EUR",
  to: "INR",
  conversionMultiple: 75,
  quantity: 10000,
  totalCalculatedAmount: 750000,
  port: 8000,
}
```

Request 2

GET to http://localhost:8100/currency-converter-feign/from/EUR/to/INR/quantity/10000

```
  {
    id: 10002,
    from: "EUR",
    to: "INR",
    conversionMultiple: 75,
    quantity: 10000,
    totalCalculatedAmount: 750000,
    port: 8001,
  }
```

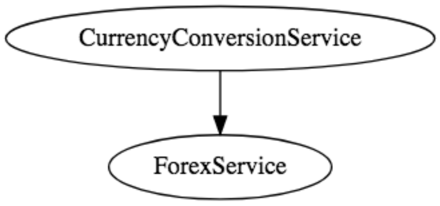
You can see that the port numbers in the two responses are different.

> *Exercise : Launch up another instance of Forex Service on 8002. You would see that load gets automatically routed to it as well*

Cool! Thats awesome. Isn't it.

Summary

We have now created two microservices and established communication between them.



We are using Ribbon to distribute load between the two instances of Forex Service and Eureka as the naming server.

When we launch new instances of Forex Service, you would see that load is automatically distribute to them.

The idea behind these series of 5 articles was to give a flavor of Spring Boot and Spring Cloud with Microservices.

There is a lot more ground to conver with Microservices. Until next time, Cheers!

Next Steps

- Join 100,000 Learners and Become a Spring Boot Expert – 5 Awesome Courses on Microservices, API's, Web Services with Spring and Spring Boot. Start Learning Now
- Learn Basics of Spring Boot - Spring Boot vs Spring vs Spring MVC, Auto Configuration, Spring Boot Starter Projects, Spring Boot Starter Parent, Spring Boot Initializr
- Learn RESTful and SOAP Web Services with Spring Boot
- Learn Microservices with Spring Boot and Spring Cloud
- Watch Spring Framework Interview Guide - 200+ Questions & Answers

Complete Code Example

/spring-boot-microservice-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.springboot.microservice.example.currency-conversion</groupId>
```

```xml
<artifactId>spring-boot-microservice-currency-conversion</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>spring-boot-microservice-currency-conversion</name>
<description>Microservices with Spring Boot and Spring Cloud - Currency Conversion Service</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.boot</groupId>
    <artifactId>spring-boot-starter-web</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-netflix-ribbon</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</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>
```

/spring-boot-microservice-currency-conversion-service/src/main/java/com/in28minutes/springboot/microservice/example/currencycon

```
package com.in28minutes.springboot.microservice.example.currencyconversion;
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
      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;
  }

}
```

/spring-boot-microservice-currency-conversion-service/src/main/java/com/in28minutes/springboot/microservice/example/currencycon

```
package com.in28minutes.springboot.microservice.example.currencyconversion;

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) {

    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.class,
        uriVariables);

    CurrencyConversionBean response = responseEntity.getBody();

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

}
```

/spring-boot-microservice-currency-conversion-service/src/main/java/com/in28minutes/springboot/microservice/example/currencycon

```
package com.in28minutes.springboot.microservice.example.currencyconversion;

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="forex-service")
@RibbonClient(name="forex-service")
public interface CurrencyExchangeServiceProxy {
  @GetMapping("/currency-exchange/from/{from}/to/{to}")
  public CurrencyConversionBean retrieveExchangeValue
    (@PathVariable("from") String from, @PathVariable("to") String to);
}
```

/spring-boot-microservice-currency-conversion-service/src/main/java/com/in28minutes/springboot/microservice/example/currencycon

```
package com.in28minutes.springboot.microservice.example.currencyconversion;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.feign.EnableFeignClients;

@SpringBootApplication
@EnableFeignClients("com.in28minutes.springboot.microservice.example.currencyconversion")
@EnableDiscoveryClient
public class SpringBootMicroserviceCurrencyConversionApplication {

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

/spring-boot-microservice-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
```

/spring-boot-microservice-currency-conversion-service/src/test/java/com/in28minutes/springboot/microservice/example/currencycon

```
package com.in28minutes.springboot.microservice.example.currencyconversion;

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 SpringBootMicroserviceCurrencyConversionApplicationTests {

  @Test
  public void contextLoads() {
  }

}
```

/spring-boot-microservice-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.springboot.microservice.eureka.naming.server</groupId>
  <artifactId>spring-boot-microservice-eureka-naming-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>spring-boot-microservice-eureka-naming-server</name>
  <description>Microservices with Spring Boot and Spring Cloud - Eureka Naming Server</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-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>
```

/spring-boot-microservice-eureka-naming-server/src/main/java/com/in28minutes/springboot/microservice/eureka/naming/server

```java
package com.in28minutes.springboot.microservice.eureka.naming.server;

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

@SpringBootApplication
@EnableEurekaServer
public class SpringBootMicroserviceEurekaNamingServerApplication {

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

/spring-boot-microservice-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
```

/spring-boot-microservice-eureka-naming-server/src/test/java/com/in28minutes/springboot/microservice/eureka/naming/server/

```
package com.in28minutes.springboot.microservice.eureka.naming.server;

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 SpringBootMicroserviceEurekaNamingServerApplicationTests {

  @Test
  public void contextLoads() {
  }

}
```

/spring-boot-microservice-forex-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.springboot.microservice.example.forex</groupId>
  <artifactId>spring-boot-microservice-forex-service</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>spring-boot-microservice-forex-service</name>
  <description>Microservices with Spring Boot and Spring Cloud - Forex Service</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.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</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>
```

/spring-boot-microservice-forex-service/src/main/java/com/in28minutes/springboot/microservice/example/forex/Excha

```
package com.in28minutes.springboot.microservice.example.forex;
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;
  }

}
```

/spring-boot-microservice-forex-service/src/main/java/com/in28minutes/springboot/microservice/example/forex/Excha

```
package com.in28minutes.springboot.microservice.example.forex;
import org.springframework.data.jpa.repository.JpaRepository;

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

/spring-boot-microservice-forex-service/src/main/java/com/in28minutes/springboot/microservice/example/forex/Forex

```
package com.in28minutes.springboot.microservice.example.forex;
import java.math.BigDecimal;

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 ForexController {

  @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")));

    return exchangeValue;
  }
}
```

/spring-boot-microservice-forex-service/src/main/java/com/in28minutes/springboot/microservice/example/forex/Spring

```
package com.in28minutes.springboot.microservice.example.forex;

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

@SpringBootApplication
@EnableDiscoveryClient
public class SpringBootMicroserviceForexServiceApplication {

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

/spring-boot-microservice-forex-service/src/main/resources/application.properties

```
spring.application.name=forex-service
server.port=8000
```

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

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

/spring-boot-microservice-forex-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);
```

/spring-boot-microservice-forex-service/src/test/java/com/in28minutes/springboot/microservice/example/forex/SpringI

```
package com.in28minutes.springboot.microservice.example.forex;

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 SpringBootMicroserviceForexServiceApplicationTests {

  @Test
  public void contextLoads() {
  }

}
```

> *Subscribe to get amazing offers on all our courses.*

> *Find out how in28Minutes reached 100,000 Learners on Udemy in 2 years. The in28minutes Way - Our approach to creating awesome learning experiences.*