Netflix Ribbon with Spring Boot

# **Introduction**

Ribbon is a client side IPC library that is battle-tested in cloud. The library includes the Netflix client side load balancers and clients for middle tier services.

Ribbon provides the following features:

* Multiple and pluggable load balancing rules
* Integration with service discovery
* Built-in failure resiliency
* Cloud enabled
* Clients integrated with load balancers
* [Archaius](https://github.com/Netflix/archaius) configuration driven client factory

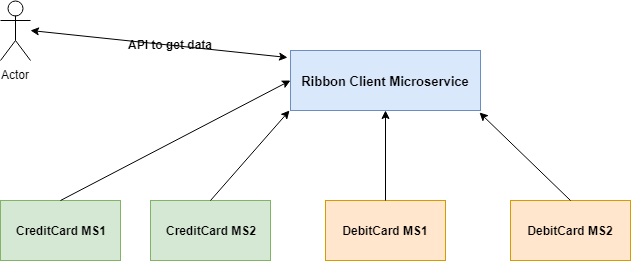
There are three sub projects:

* ribbon-core: includes load balancer and client interface definitions, common load balancer implementations, integration of client with load balancers and client factory.
* ribbon-eureka: includes load balancer implementations based on [Eureka](https://github.com/Netflix/eureka) client, which is the library for service registration and discovery.
* ribbon-httpclient: includes the JSR-311 based implementation of REST client integrated with load balancers.

**Ribbon is a client-side load balancer** that gives you a lot of control over the behavior of HTTP and TCP clients. A central concept in Ribbon is that of the named client. Each load balancer is part of an ensemble of components that work together to contact a remote server on demand, and the ensemble has a name that you give it as an application developer (for example, by using the @FeignClient annotation). On demand, Spring Cloud creates a new ensemble as an ApplicationContext for each named client by using RibbonClientConfiguration. This contains (amongst other things) an ILoadBalancer, a RestClient, and a ServerListFilter.

Before moving into the code structure, let us consider an example.

There is a microservice for debit card and for credit card. Now we want to run both the micro services in different ports. There is another micro service called card services which can access both debit and credit card micro services. In this case, there may be multiple instances of credit and debit card micro service, when we access through card service api, it will use Netflix Ribbon load balancer api to access the micro services so that it can easily handle fault tolerance. The diagram is given below.



# **Debit Card Microservice**

## **Maven Configuration (pom.xml)**

<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>debitcard</groupId>

<artifactId>debitcard</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>debitcard</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.0.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# **Spring Boot Layer**

**DebitCardBootApplication.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.boot.SpringApplication;

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

@SpringBootApplication

**public** **class** DebitCardBootApplication {

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

SpringApplication.*run*(DebitCardBootApplication.**class**, args);

}

}

**BootConfig.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@ComponentScan(basePackages= {"com.ddlab.rnd.\*","com.ddlab.boot.\*"})

@Configuration

**public** **class** BootConfig {

}

**Resource/Controller layer**

**package** com.ddlab.boot.controller;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/home")

@CrossOrigin

**public** **class** DebitCardController {

@Value("${server.port}")

String port;

@GetMapping(path = "/name", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** ResponseEntity<String> getDebitCardName() {

**return** **new** ResponseEntity<String>("HDFC Debit Card service running in " + port, HttpStatus.***OK***);

}

}

## **Spring Boot Configuration**

**application.properties**

spring.application.name=dc

server.servlet.context-path=/dc

server.port=8081

#server.port=8082

server.error.whitelabel.enabled=false

The same code is for CreditCard Microservice. The application.properties for CreditCard service is given below.

application.properties for creditcard microservice

spring.application.name=cc

server.servlet.context-path=/cc

server.port=8091

#server.port=8092

server.error.whitelabel.enabled=false

# **Card Service Ribbon Client Microservice**

## **Maven Configuration(pom.xml)**

<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>cardserviceRibbon</groupId>

<artifactId>cardserviceRibbon</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>cardserviceRibbon</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.0.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-web</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-netflix-ribbon</artifactId>**

**</dependency>**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**<dependencyManagement>**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>Finchley.RELEASE</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

**</dependencies>**

**</dependencyManagement>**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

## **Spring Boot Layer**

**CardServiceBootApplication.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.boot.SpringApplication;

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

**import** org.springframework.cloud.netflix.ribbon.RibbonClient;

**import** com.ddlab.rnd.config.Configuration;

@SpringBootApplication

@RibbonClient(name = "justANameToIgnore", configuration = Configuration.**class**)

**public** **class** CardServiceBootApplication {

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

SpringApplication.*run*(CardServiceBootApplication.**class**, args);

}

}

**BootConfig.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@ComponentScan(basePackages= {"com.ddlab.rnd.\*","com.ddlab.boot.\*"})

@Configuration

**public** **class** BootConfig {

}

## **Ribbon Configuration Layer**

**Configuration.java**

**package** com.ddlab.rnd.config;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

**import** com.netflix.client.config.IClientConfig;

**import** com.netflix.loadbalancer.AvailabilityFilteringRule;

**import** com.netflix.loadbalancer.IPing;

**import** com.netflix.loadbalancer.IRule;

**import** com.netflix.loadbalancer.PingUrl;

**public** **class** Configuration {

@Autowired IClientConfig ribbonClientConfig;

@Bean

**public** IPing ribbonPing(IClientConfig config) {

**return** **new** PingUrl();

}

@Bean

**public** IRule ribbonRule(IClientConfig config) {

**return** **new** AvailabilityFilteringRule();

}

}

## **Controller layer**

**CardServiceController.java**

**package** com.ddlab.boot.controller;

**import** java.net.URI;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.cloud.client.ServiceInstance;

**import** org.springframework.cloud.client.loadbalancer.LoadBalanced;

**import** org.springframework.cloud.client.loadbalancer.LoadBalancerClient;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.http.MediaType;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** org.springframework.web.client.RestTemplate;

@RestController

@RequestMapping("/cards")

@CrossOrigin

**public** **class** CardServiceController {

**@LoadBalanced**

@Bean

RestTemplate restTemplate() {

**return** **new** RestTemplate();

}

**@Autowired RestTemplate restTemplate;**

@GetMapping(path = "/dcard/name", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String getDebitCardName() {

**return** **this**.restTemplate.getForObject("http://debitcard/dc/home/name", String.**class**);

}

@GetMapping(path = "/ccard/name", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String getCreditCardName() {

**return** **this**.restTemplate.getForObject("http://creditcard/cc/home/name", String.**class**);

}

**@Autowired private LoadBalancerClient loadBalancer;**

@GetMapping(path = "/info", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String doStuff() {

ServiceInstance instance = loadBalancer.choose("creditcard");

URI storesUri =

URI.*create*(String.*format*("http://%s:%s", instance.getHost(), instance.getPort()));

**return** storesUri.toString();

}

}

## **Ribbon Client Microservice Configuration**

**Application.yml**

spring:

application:

name: Ribbon-Client

**debitcard**:

ribbon:

eureka:

enabled: **false**

listOfServers: localhost:8081,localhost:8082

ServerListRefreshInterval: 1000

**creditcard:**

ribbon:

eureka:

enabled: **false**

listOfServers: localhost:8091,localhost:8092

ServerListRefreshInterval: 1000

server:

port: 8090

# **How to Use**

* 1. Start Debit Card Micro service in both 8081 and 8082 ports.
  2. Start CreditCard Micro service in both 8091 and 8092 ports.
  3. Start Ribbon Client Micro service CardServiceRibbon in port 8090 port.

Use the following URLS in browser.

**For Debit Card**

http://localhost:8081/dc/home/name

http://localhost:8082/dc/home/name

**For Credit Card**

http://localhost:8091/cc/home/name

http://localhost:8092/cc/home/name

**For Card Service Ribbon**

http://localhost:8090/cards/dcard/name

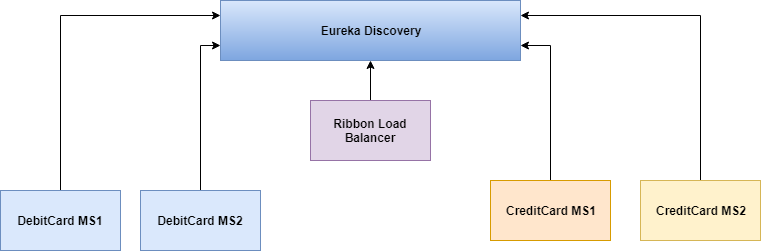
http://localhost:8090/cards/ccard/name

<http://localhost:8090/cards/info>

Spring Boot with Eureka and Ribbon

# **Introduction**

In this case micro services become more complex. We use both Eureka as discovery client and Ribbon as client side load balancer. Let us consider the same example with Eureka and Ribbon. The diagram is given below.



# **Eureka Server**

## **Maven Configuration (pom.xml)**

<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>eureka-server</groupId>

<artifactId>eureka-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>eureka-server</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.0.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>**

**</dependency>**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**<dependencyManagement>**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>Finchley.RELEASE</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

**</dependencies>**

**</dependencyManagement>**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

## **Spring Boot Application Layer**

**EurekaServerApplication.java**

**package** com.ddlab.rnd.eureka;

**import** org.springframework.boot.SpringApplication;

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

**import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

**@EnableEurekaServer**

**public** **class** EurekaServerApplication {

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

SpringApplication.*run*(EurekaServerApplication.**class**, args);

}

}

## **Spring Boot Eureka Configuration**

**application.properties**

#spring.application.name=eureka

#server.servlet.context-path=/eureka

#server.port=8090

server.error.whitelabel.enabled=false

server.port=8761

#You comment below two lines, you will see Eureka server as a client and

#It will be displayed as UNKNOWN

#The below the correct configuration

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

#

#logging.level.com.netflix.eureka=OFF

#logging.level.com.netflix.discovery=OFF

Eureka configuration is bit straight forward.

# **DebitCard Microservice**

## **Maven Configuration (pom.xml)**

<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>debitcard</groupId>

<artifactId>debitcard</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>debitcard</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.0.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- For Eureka -->

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>**

**</dependency>**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**<!-- For Eureka -->**

**<dependencyManagement>**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>Finchley.RELEASE</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

**</dependencies>**

**</dependencyManagement>**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# **Spring Boot Layer**

**DebitCardBootApplication.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.boot.SpringApplication;

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

@SpringBootApplication

**public** **class** DebitCardBootApplication {

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

SpringApplication.*run*(DebitCardBootApplication.**class**, args);

}

}

**BootConfig.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@ComponentScan(basePackages= {"com.ddlab.rnd.\*","com.ddlab.boot.\*"})

@Configuration

**public** **class** BootConfig {

}

**Resource/Controller layer**

**package** com.ddlab.boot.controller;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/home")

@CrossOrigin

**public** **class** DebitCardController {

@Value("${server.port}")

String port;

@GetMapping(path = "/name", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** ResponseEntity<String> getDebitCardName() {

**return** **new** ResponseEntity<String>("HDFC Debit Card service running in " + port, HttpStatus.***OK***);

}

}

## **Spring Boot Configuration for Eureka**

**application.properties**

spring.application.name=dc

server.servlet.context-path=/dc

#server.port=8081

server.port=8082

server.error.whitelabel.enabled=false

#Eureka Configuration

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

**bootstrap.properties**

**spring.application.name=dc**

The source code for CreditCard microservice is same. However, I provide below the configuration details for better understanding.

## **Spring Boot Configuration for Eureka for CreditCard**

**application.properties**

spring.application.name=cc

server.servlet.context-path=/cc

#server.port=8091

server.port=8092

server.error.whitelabel.enabled=false

#Eureka Configuration

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

**bootstrap.properties**

**spring.application.name=cc**

# **Card Service Ribbon Client Microservice**

## **Maven Configuration(pom.xml)**

<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>cardserviceRibbon</groupId>

<artifactId>cardserviceRibbon</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>cardserviceRibbon</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.0.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</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>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**<dependencyManagement>**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>Finchley.RELEASE</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

**</dependencies>**

**</dependencyManagement>**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

## **Spring Boot Layer**

**CardServiceBootApplication.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.boot.SpringApplication;

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

**import** org.springframework.cloud.netflix.ribbon.RibbonClient;

**import** com.ddlab.rnd.config.Configuration;

@SpringBootApplication

@RibbonClient(name = "justANameToIgnore", configuration = Configuration.**class**)

**public** **class** CardServiceBootApplication {

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

SpringApplication.*run*(CardServiceBootApplication.**class**, args);

}

}

**BootConfig.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@ComponentScan(basePackages= {"com.ddlab.rnd.\*","com.ddlab.boot.\*"})

@Configuration

**public** **class** BootConfig {

}

## **Ribbon Configuration Layer**

**Configuration.java**

**package** com.ddlab.rnd.config;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

**import** com.netflix.client.config.IClientConfig;

**import** com.netflix.loadbalancer.AvailabilityFilteringRule;

**import** com.netflix.loadbalancer.IPing;

**import** com.netflix.loadbalancer.IRule;

**import** com.netflix.loadbalancer.PingUrl;

**public** **class** Configuration {

@Autowired IClientConfig ribbonClientConfig;

@Bean

**public** IPing ribbonPing(IClientConfig config) {

**return** **new** PingUrl();

}

@Bean

**public** IRule ribbonRule(IClientConfig config) {

**return** **new** AvailabilityFilteringRule();

}

}

## **Controller layer**

**CardServiceController.java**

**package** com.ddlab.boot.controller;

**import** java.net.URI;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.cloud.client.ServiceInstance;

**import** org.springframework.cloud.client.loadbalancer.LoadBalanced;

**import** org.springframework.cloud.client.loadbalancer.LoadBalancerClient;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.http.MediaType;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** org.springframework.web.client.RestTemplate;

@RestController

@RequestMapping("/cards")

@CrossOrigin

**public** **class** CardServiceController {

**@LoadBalanced**

@Bean

RestTemplate restTemplate() {

**return** **new** RestTemplate();

}

**@Autowired RestTemplate restTemplate;**

@GetMapping(path = "/dcard/name", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String getDebitCardName() {

**return** **this**.restTemplate.getForObject("http://debitcard/dc/home/name", String.**class**);

}

@GetMapping(path = "/ccard/name", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String getCreditCardName() {

**return** **this**.restTemplate.getForObject("http://creditcard/cc/home/name", String.**class**);

}

**@Autowired private LoadBalancerClient loadBalancer;**

@GetMapping(path = "/info", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String doStuff() {

ServiceInstance instance = loadBalancer.choose("creditcard");

URI storesUri =

URI.*create*(String.*format*("http://%s:%s", instance.getHost(), instance.getPort()));

**return** storesUri.toString();

}

}

## **EurekaRibbonCardController.java**

**package** com.ddlab.boot.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.cloud.client.ServiceInstance;

**import** org.springframework.cloud.client.loadbalancer.LoadBalancerClient;

**import** org.springframework.http.MediaType;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** org.springframework.web.client.RestTemplate;

@RestController

@RequestMapping("/eurocards")

@CrossOrigin

**public** **class** EurekaRibbonCardController {

**@Autowired private LoadBalancerClient loadBalancer;**

**private** String getDCLoadBalancerURL() {

**ServiceInstance dcInstance = loadBalancer.choose("DC");**

String dcURL = dcInstance.getUri().toString();

System.***out***.println("Base URL : " + dcURL);

dcURL = dcURL + "/dc/home/name";

**return** dcURL;

}

**private** String getCCLoadBalancerURL() {

**ServiceInstance ccInstance = loadBalancer.choose("CC");**

String ccURL = ccInstance.getUri().toString();

System.***out***.println("Base URL : " + ccURL);

ccURL = ccURL + "/cc/home/name";

**return** ccURL;

}

@GetMapping(path = "/eureka/ribbon/names", produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String doStuff() {

RestTemplate restTemplate = **new** RestTemplate();

String response = **null**;

StringBuilder allResponse = **new** StringBuilder();

**try** {

String dcURL = getDCLoadBalancerURL();

**response = restTemplate.getForObject(dcURL, String.class);**

allResponse.append("Debit Card Details \n").append(response).append("\n\n");

String ccURL = getCCLoadBalancerURL();

**response = restTemplate.getForObject(ccURL, String.class);**

allResponse.append("Credit Card Details \n").append(response).append("\n\n");

} **catch** (Exception ex) {

System.***out***.println(ex);

}

System.***out***.println("Finally response ::: " + response);

**return** allResponse.toString();

}

}

## **Ribbon Client Microservice Configuration**

**Application.yml**

spring:

application:

name: Ribbon-Client

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

debitcard:

ribbon:

eureka:

enabled: **false**

listOfServers: localhost:8081,localhost:8082

ServerListRefreshInterval: 1000

creditcard:

ribbon:

eureka:

enabled: **false**

listOfServers: localhost:8091,localhost:8092

ServerListRefreshInterval: 1000

server:

port: 8090

**bootstrap.properties**

spring.application.name=Ribbon-Client

# **How to use**

* 1. Start Eureka Server in 8671 port.
  2. Start Debit Card Micro service in both 8081 and 8082 ports.
  3. Start CreditCard Micro service in both 8091 and 8092 ports.
  4. Start Ribbon Client Micro service CardServiceRibbon in port 8090 port.

Access the following URLs.

<http://localhost:8761/> (For Eureka Server)

<http://localhost:8090/eurocards/eureka/ribbon/names>

Output

Debit Card Details

HDFC Debit Card service running in 8081

Credit Card Details

Standard Chatered Card service running in 8091

If you access the Eureka server(<http://localhost:8761/>), you find the following output as screenshot.

