



Java Code Analysis

Continuous Java Code Quality & Code Security analysis. Start scanning for free!

SonarCloud

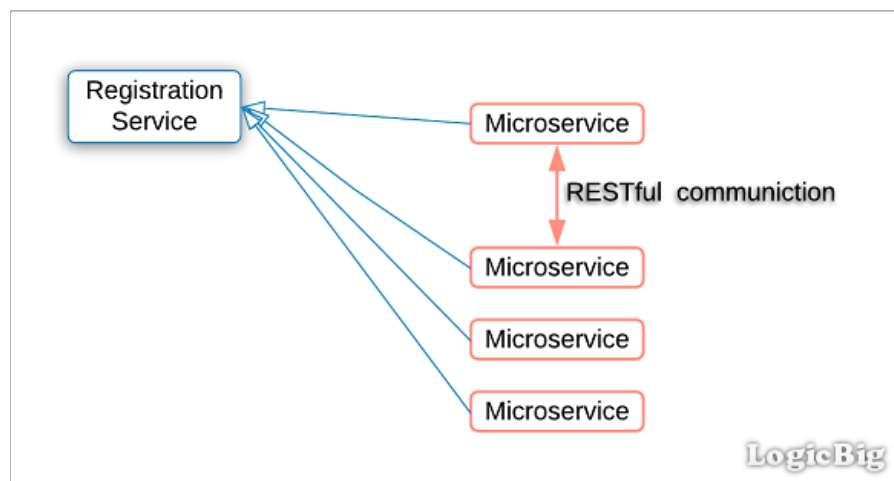
Spring Cloud - Getting Started Example

[Last Updated: Jun 20, 2020]

Following is a quick-start example of Spring Cloud. We are going to develop very simple microservices using Spring Cloud, Spring Boot and Eureka Server.

In microservice architecture, an application is composed of loosely coupled small services as opposed to a single monolithic application.

In microservice architecture a registry service is used to register the microservices so that they can be discovered.



Example

Next Page



In this example we are going to use [Eureka Server](#) as the service registry. Eureka is developed by Netflix; it is open source. Spring has integrated Eureka into dedicated Spring Cloud modules to make it easier to use it.

We are going to develop two microservices:

First is 'hello-service' which will just return a hello message.

Second service 'hello-web-client-service' will handle the request coming from a client. On receiving a request it will call 'hello-service' and will return a web page in response.

There will be three separate servers; one for Eureka and two of microservices. Also there will be three separate maven projects.

Eureka Server

Maven dependencies

pom.xml

```
<project .....>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.logicbig.example</groupId>
  <artifactId>hello-eureka-server</artifactId>
  <version>1.0-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>
  </parent>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
    </dependency>
  </dependencies>
</project>
```

```
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Finchley.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
</project>
```

Configuration

src/main/resources/application.yml

```
server:
  port: 7777
eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
```

In above configuration, the properties eureka.client.* are related to the service clients who want to register with Eureka.

The property eureka.client.register-with-eureka=false specifies that this server should not be registered to the service client itself.

The property eureka.client.fetch-registry=false specifies that the server should not fetch the registered information to itself.

Main class

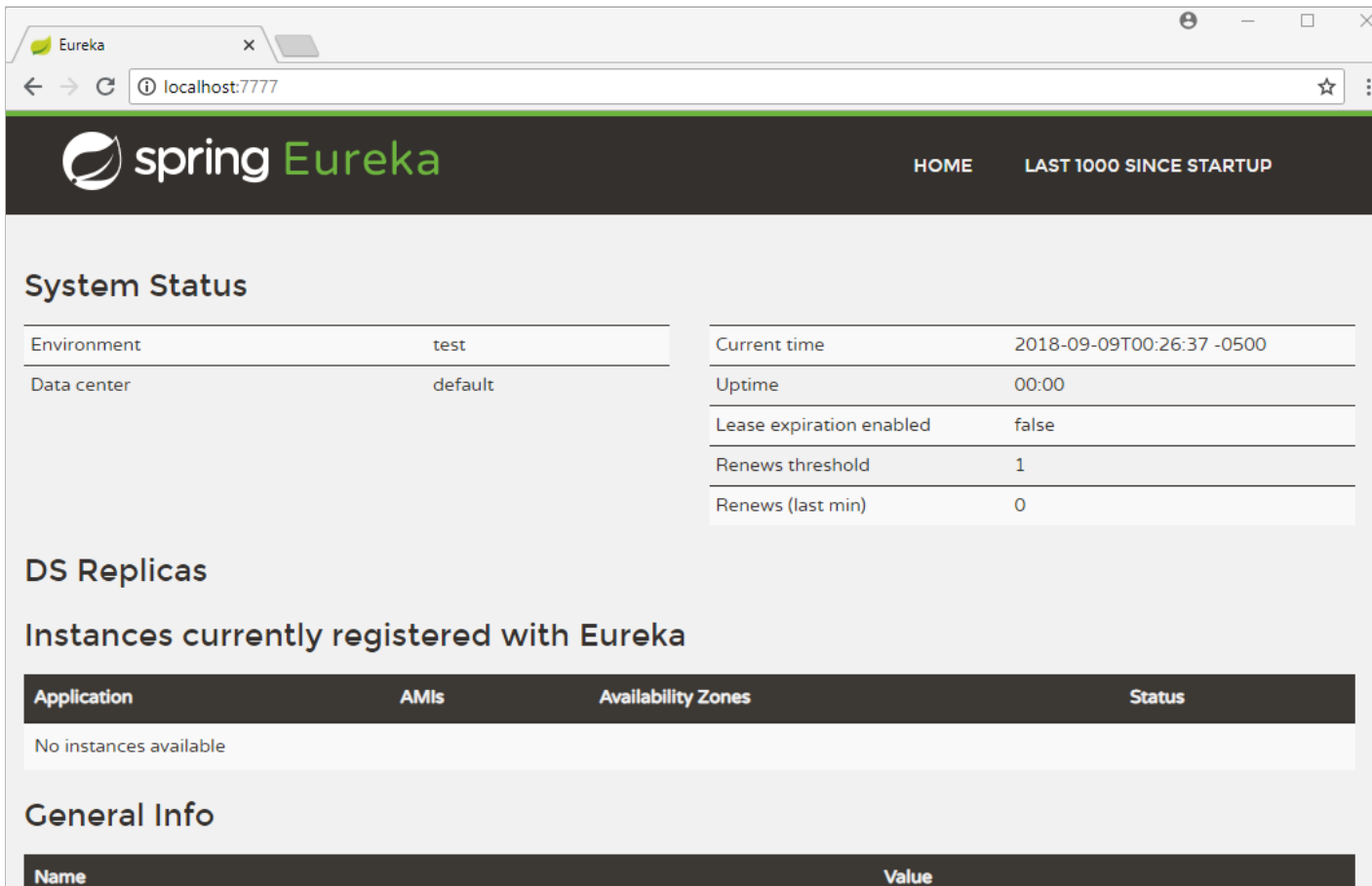
**New Merch Coming
Soon 2022**

Ad Blackjamez

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

Run above main class from your IDE. That will start the Eureka Server.

Now we can access the Eureka server at <http://localhost:7777> as shown:



The screenshot shows the Spring Eureka web interface. The browser address bar displays `localhost:7777`. The page header includes the Spring Eureka logo and navigation links for `HOME` and `LAST 1000 SINCE STARTUP`.

System Status

Environment	test	Current time	2018-09-09T00:26:37 -0500
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value
------	-------



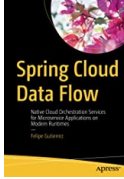



hello-service

Maven dependencies

pom.xml

```
<project .....>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.logicbig.example</groupId>
  <artifactId>hello-service</artifactId>
  <version>1.0-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>
  </parent>
  <properties>
    <java.version>1.8</java.version>
  </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>
```

 <p>\$36.79 \$60.00 Cloud Native Java: Designing Resilient Sy... (60)</p>	 <p>\$59.99 Cloud Native Spring in Action</p>
 <p>Spring Cloud Data Flow: Native Cloud Orchestrati... (4)</p>	 <p>Hands-On Microservices with Spring Boot and S... (114)</p>

Ads by Amazon

Core Java Tutorials

[Java 16 Features](#)
[Java 15 Features](#)
[Java 14 Features](#)
[Java 13 Features](#)
[Java 12 Features](#)
[Java 11 Features](#)
[Java 10 Features](#)
[Java 9 Module System](#)
[Java 9 Misc Features](#)
[Java 9 JShell](#)

Recent Tutorials

[Python - Tuples](#)
[Python - Lists](#)
[Python - Naming Conventions](#)

```

        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Finchley.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
</project>

```

Domain object

```

public class HelloObject {
    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}

```

A Rest Controller

```

@RestController
public class HelloController {
    private AtomicLong counter = new AtomicLong();

    @GetMapping("/hello")

```

[Python - Built In Data Structures](#)
[Python None Type](#)
[Python Boolean](#)
[Python Keywords](#)
[Python String](#)
[Python Numbers](#)
[Python Variables](#)
[Python Comments](#)
[Python Basic Syntax](#)
[Python with IntelliJ](#)
[Python language getting started](#)
[Installing Python 3.10.x on windows](#)
[Spring - Mixed validation using JSR 349 Annotations and Spring validator](#)
[Spring Annotation Based Validation](#)
[Spring Core Validation](#)
[Java Bean Validation - Creating multiple validators for the same constraint](#)
[Spring - The Use Of PropertyEditors With @Value Annotation](#)
[Spring - The Use Of PropertyEditors With XML Configuration](#)
[Spring - Use of PropertyEditors via BeanWrapper](#)
[Spring - Using BeanUtils for resolving string based method signatures](#)
[Spring - Copying properties using BeanUtils](#)
[Spring - Obtaining BeanInfo And PropertyDescriptors](#)
[Spring - Directly setting fields via DirectFieldAccess](#)
[Spring - BeanWrapper, setting nested beans](#)
[Spring - Using BeanWrapper](#)
[Spring - Transforming events](#)

```
public HelloObject getHelloWordObject() {  
    HelloObject hello = new HelloObject();  
    hello.setMessage("Hi there! you are number " + counter.incrementAndGet());  
    return hello;  
}  
}
```

src/main/resources/application.properties

```
eureka.client.serviceUrl.defaultZone=http://localhost:7777/eureka/
```

src/main/resources/bootstrap.properties

```
spring.application.name=hello-service
```

The bootstrap.properties file corresponds to bootstrap context (the parent context of the main application) which uses a different convention for locating external configuration than the main application context. Instead of application.yml (or .properties), we can use bootstrap.yml, keeping the external configuration for bootstrap and main context nicely separate.

Boot main class

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

Run above main class from your IDE.

On refreshing Eureka page you should see HELLO-SERVICE instance listed in the registry:

[Spring - Conditional Event Handling via @EventListener annotation](#)

[Spring - Using @EventListener annotation with multiple event types](#)

[Spring - Publishing and Consuming Custom Events](#)

[Java - Introduction to ResourceBundle](#)

[Spring - Injecting Resource using @Value annotation](#)

[Spring - Injecting ResourceLoader](#)

[Spring - Resource Loading](#)

[Spring - Using Spring Expression Language with @Value Annotation](#)

[Spring - Using @Value Annotation](#)

[Spring - Injecting Environment to access properties in beans](#)

[Spring - Adding New Property Source to Environment](#)

[Spring - Adding user properties by using @PropertySource](#)

[Spring - Accessing Environment Properties](#)

[Spring - Profiles](#)

[Spring - Successfully injecting circular dependencies using @Lazy Annotation](#)

[Spring - Successfully injecting circular dependencies using setter injection](#)

[Java 16 - Records Features, Quick Walk-through](#)

[Java 16 - Introduction to Records](#)

[Spring - Injecting beans into Arrays/Collections, Using @Qualifiers And Specifying the Ordering](#)

[Injecting Collections - Injecting Beans Into Arrays And Collections, ordering with Ordered Interface](#)

[Spring - Injecting beans Into Arrays and Lists, ordering with @Ordered annotation](#)

The screenshot shows the Spring Eureka web interface in a browser window. The page has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. Below the header, there's a 'System Status' section with two tables. The first table shows 'Environment: test' and 'Data center: default'. The second table shows 'Current time: 2018-09-09T00:27:32 -0500', 'Uptime: 00:01', 'Lease expiration enabled: false', 'Renews threshold: 3', and 'Renews (last min): 0'. Below this is a 'DS Replicas' section with the heading 'Instances currently registered with Eureka'. A table lists the registered instances, with 'HELLO-SERVICE' highlighted by a red arrow. The table has columns for 'Application', 'AMIs', 'Availability Zones', and 'Status'. The 'HELLO-SERVICE' row shows 'n/a (1)' for AMIs, '(1)' for Availability Zones, and 'UP (1) - JoeMchn:hello-service' for Status. At the bottom, there's a 'General Info' section with a table for 'Name' and 'Value'.

Application	AMIs	Availability Zones	Status
HELLO-SERVICE	n/a (1)	(1)	UP (1) - JoeMchn:hello-service



[Spring - Injecting beans into Arrays and Collections, selecting elements with @Qualifier annotation](#)

[Spring - Injecting multiple Beans Into Arrays and Collections](#)

[Spring - Arrays and Collections As Beans](#)

[Spring - Using @ComponentScan#excludeFilters to exclude classes from scanning based on annotations](#)

[Spring - Using @ComponentScan#includeFilters to scan non component classes based on annotations](#)

[Spring - Implementing ApplicationContextAware Interface](#)

[Spring - Using excludeFilters attribute of @ComponentScan to exclude component classes](#)

[Spring - Using @ComponentScan to scan non component classes via includeFilters attribute](#)

[Spring - Using Filters To Customize Scanning with @ComponentScan](#)

[Spring - Using basePackageClasses Attribute of @ComponentScan](#)

[Spring - Specifying packages to be scanned with basePackages attribute of @ComponentScan](#)

[JUnit - How to test user command line Input in Java?](#)

[Spring - Session based Prototype Bean Example](#)

[Spring - Prototype Bean Example](#)

[Spring - Singleton Bean Example](#)

[Spring - Receiving lifecycle callbacks by implementing InitializingBean and DisposableBean](#)

[Spring - Receiving lifecycle callbacks by using 'initMethod' and 'destroyMethod' of @Bean annotation](#)

[Spring - Implicit Constructor Injection In @Configuration Class](#)

hello-web-client-service

pom.xml

```
<project .....>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.logicbig.example</groupId>
  <artifactId>hello-web-client-service</artifactId>
  <version>1.0-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
```



```

    <version>2.0.4.RELEASE</version>
</parent>
<properties>
    <java.version>1.8</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Finchley.SR1</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
</project>

```

A Spring MVC Controller

[Spring - Using @Autowired annotation on arbitrary methods](#)

[Spring - Using @Inject annotation on setter methods](#)

[Spring Core - Using @Autowire annotation On a setter method](#)

[Spring - Defining Injection point by using @Inject annotation](#)

[Spring - Defining injection point by using @Autowire annotation](#)

[Spring - Resolving ambiguity by using @Inject and @Named annotations](#)

[Spring - Resolving ambiguity by using @Inject and @Qualifier Annotations](#)

[Spring - Autowiring By Name, Using Autowire.BY_NAME](#)

[Spring - Autowiring By Type mode, Using Autowire.BY_TYPE](#)

[Spring - Default Auto-wiring mode, Autowire.NO Example](#)

[Elements of @Bean Annotation](#)

[Spring - Using Multiple @Configuration Classes](#)

[Spring - Dependency Injection In @Configuration Classes](#)

[Reactor - Programmatically generate Flux via Consumer<SynchronousSink<T>>](#)

[Spring Boot - Testing With @SpyBean](#)

[Mockito - Creating Spy Of Real Objects](#)

[Reactor - Creating Flux Instance From Iterable](#)

[Reactor - Create Flux Instance From Array](#)

[Reactor - Creating Flux Instance Which Emits Range Of Integer](#)

[Reactor - Creating Flux and Mono with empty\(\)](#)

[Java - Find Files in classpath under a Folder And SubFolder](#)

[Java - How to find enum by ordinal?](#)

```

@Controller
public class HelloWebClientController {
    @Autowired
    private DiscoveryClient discoveryClient;

    @GetMapping("/")
    public String handleRequest(Model model) {
        //accessing hello-service
        List<ServiceInstance> instances = discoveryClient.getInstances("hello-service");
        if (instances != null && instances.size() > 0) {//todo: replace with a load balancing mechanism
            ServiceInstance serviceInstance = instances.get(0);
            String url = serviceInstance.getUri().toString();
            url = url + "/hello";
            RestTemplate restTemplate = new RestTemplate();
            HelloObject helloObject = restTemplate.getForObject(url,
                HelloObject.class);
            model.addAttribute("msg", helloObject.getMessage());
            model.addAttribute("time", LocalDateTime.now());
        }
        return "hello-page";
    }
}

```

src/main/resources/templates/hello-page.html

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">

<body>
<h2>Hello Page</h2>
<div th:text="{msg}" />
<div>Time: <span th:text="{time}" /></div>
</body>
</html>

```

[Java 15 - Sealed Classes \(preview\)](#)

[Spring Boot Primefaces Integration](#)

[Spring Boot JSF Integration](#)

[Reactor - Creating an instance using Flux.just\(\) Mono.just\(\)](#)

[Spring Boot - Testing With @MockBean](#)

[Java - How to delete old files under a folder if number of files are over a specified limit?](#)

[Mockito - BDD Style Verification using then\(\) - should\(\)](#)

[Mockito - BDD Style Stubbing with given\(\) - willReturn\(\)](#)

[Cassandra - Mapping Java Objects using Mapper](#)

[Java - How to convert Calendar to LocalDateTime?](#)

[Mockito - verifyNoMoreInteractions\(\) and verifyNoInteractions\(\)](#)

[Mockito - Verifying Multiple Number of Method Invocations](#)

[Mockito - Verify Method Calls With Argument Matchers](#)

[Kafka - ConsumerRebalanceListener Example](#)

[Kafka - Understanding Partition Rebalancing](#)

[Kafka Manual Commit - commitSync\(\) Example](#)

[Kafka Manual Commit - CommitAsync With Callback and Specified Offset](#)

[Kafka Manual Commit - commitAsync With Callback Example](#)

[Java - Introduction to Java 8 Date and Time API](#)

[Reactor - Understanding Flux/Mono's retryWhen\(\)](#)

[Reactor - Retrying Flux/Mono Sequence](#)

[Spring - Injecting Prototype Bean Using Java 8 Functions](#)

[Mockito - Argument Matchers Example](#)

[Mockito - Argument Matchers](#)

src/main/resources/application.properties

```
server.port=9080
eureka.client.serviceUrl.defaultZone=http://localhost:7777/eureka/
```

src/main/resources/bootstrap.properties

```
spring.application.name=hello-service
```

Boot main class

```
@SpringBootApplication
@EnableDiscoveryClient
public class HelloWebClientServiceMain {

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

Run above class from your IDE.

Refresh Eureka web page again:

[Kafka Manual Commit - CommitAsync\(\) Example](#)[Java - How to Indent multiline String?](#)[Mockito - Verifying Method Calls](#)[Spring MVC - Post Request With Simple Form Submission](#)[Java - Parsing String To Numeric Primitives](#)[Mockito - Stubbing methods with exceptions](#)[Java - Avoiding possible NullPointerException with method call chain](#)[Java - Autoboxing And Unboxing, How to avoid NullPointerException with unboxing?](#)[Kafka - Auto Committing Offsets](#)[Kafka - Understanding Offset Commits](#)[Kafka - Publishing records With null keys and no assigned partitions](#)[Java Collections - How to find frequency of each element in a collection?](#)[How to convert java.util.Map To Java Bean?](#)[Java - How to repeat a string n number of times?](#)[Git - Merging Branches](#)[Java - How to convert Iterator To List?](#)[Spring Boot - Unit Testing Application Arguments](#)[How to find the longest and the shortest String in Java?](#)[How to find first and last element of Java 8 stream?](#)[Mockito - Stubbing consecutive method calls](#)[Mockito - Stubbing a method's return Value](#)[Kafka - Using Keys For Partition Assignment](#)[Spring Boot - Using @TestConfiguration In static nested class](#)[Spring Boot - Using @TestConfiguration to define beans for tests](#)[Java Collections - Why Arrays.asList\(\) does not work for primitive arrays?](#)

The screenshot shows the Eureka web interface in a browser window. The address bar shows 'localhost:7777'. The page is divided into three main sections: System Status, DS Replicas, and General Info.

System Status

Environment	test	Current time	2018-09-09T00:34:18 -0500
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	5
		Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
HELLO-SERVICE	n/a (1)	(1)	UP (1) - JoeMchn:hello-service
HELLO-WEB-CLIENT-SERVICE	n/a (1)	(1)	UP (1) - JoeMchn:hello-web-client-service:9080

A red arrow points to the 'HELLO-WEB-CLIENT-SERVICE' row.

General Info

Name	Value
total-avail-memory	447mb

Final output

Now make request to 'hello-web-client-service' by entering localhost:9080 in web-browser:

[Java Collections - Only put Map key/value if the specified key does not exist](#)

[Getting Started with Mockito](#)

[How to connect a Database server in IntelliJ Community Edition?](#)

[Java HashMap - Understanding equals\(\) and hashCode\(\) methods](#)

[Java IO - How to write lines To a file and read lines from a files?](#)

[Java Collections - How to find distinct elements count in collections and arrays?](#)

[Spring Boot - Web Application Testing With Embedded Server And TestRestTemplate](#)

[Java - How to find Available Runtime Memory?](#)

[Kafka - Understanding Consumer Group with examples](#)

[Spring Boot - Unit Testing Web Application With Embedded Server](#)

[Java - Different ways to Set Nested Field Value By Reflection](#)

[Java - Different ways to Set Field Value by Reflection](#)

[Installing Python 2.7 on windows](#)

[Installing Cassandra And Intro To CQLSH](#)

[Cassandra - Getting Started with Java](#)

[Java 14 - Switch Expressions And Statements Examples](#)

[Kafka - Manually Assign Partition To A Consumer](#)

[Kafka - Understanding Topic Partitions](#)

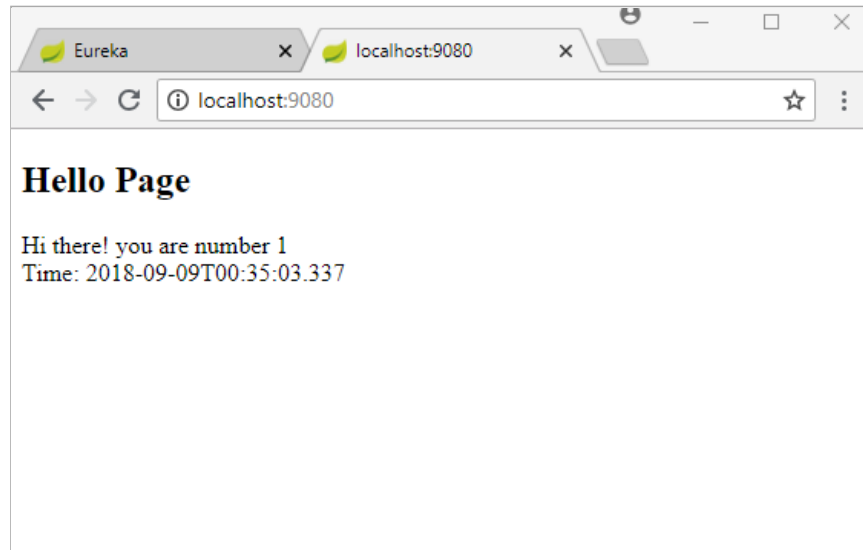
[Spring Boot - Unit Testing Web Application With Mock MVC](#)

[Kafka - Introduction to Kafka Admin API](#)

[Java 14 - Helpful NullPointerException](#)

[Java 14 - Pattern Matching for instanceof](#)

[Spring Boot - Application Unit Testing with @SpringBootTest](#)



好用的梯子VPN – 一键连接，翻越国界

看视频，玩游戏，聊天交友，自由上网 webjiasu.com

Example Project

Dependencies and Technologies Used:

- Spring Boot 2.0.4.RELEASE
Corresponding Spring Version 5.0.8.RELEASE
- Spring Cloud Finchley.SR1
- spring-cloud-starter-netflix-eureka-server 2.0.1.RELEASE: Spring Cloud Starter Netflix Eureka Server.
- spring-boot-starter-web : Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container.
- spring-cloud-starter-netflix-eureka-client 2.0.1.RELEASE: Spring Cloud Starter Netflix Eureka Client.
- spring-boot-starter-thymeleaf : Starter for building MVC web applications using Thymeleaf views.
Uses [org.thymeleaf:thymeleaf-spring5](#) version 3.0.9.RELEASE
- JDK 1.8
- Maven 3.5.4

[Installing and Running Kafka](#)

[Kafka - Getting Started](#)

[Spring Boot - Different Ways To Pass Application Properties](#)

[Reactor - Transforming into Publishers and Delaying any Error with flatMapDelayError\(\) and flatMapSequentialDelayError\(\)](#)

[Reactor - Transforming into Publisher and Maintaining the source order with flatMapSequential\(\)](#)

[Reactor - Transforming into Publishers and then flattening by using flatMap operation](#)

[Reactor - Using transform Operation](#)

[Reactor - Mapping items](#)

[Reactor - Getting Started](#)

[Java 13 - Text Blocks \(JEP 355 - preview\)](#)

[Spring Cloud - Hystrix CircuitBreaker, Thread Local Context Propagation](#)

[Spring Cloud - Circuit Breaker Hystrix Event Listener](#)

Getting started with Spring Cloud + Microservices

spring-cloud-getting-started
hello-eureka-server
src
pom.xml
hello-service
src
pom.xml
hello-web-client-service
src
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.logicbig.example</groupId>
    <artifactId>hello-eureka-server</artifactId>
    <version>1.0-SNAPSHOT</version>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.4.RELEASE</version>
    </parent>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
```

外网VPN，突破中国网络封锁 – 高速连接，不限流量

全球服务器部署，高速翻墙，访问全网，连接更快更稳定，全球IP自由切换 webjiasu.com

Project Structure

spring-cloud-getting-started
hello-eureka-server
src
pom.xml
hello-service
src
pom.xml
hello-web-client-service
src

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.logicbig.example</groupId>
    <artifactId>hello-eureka-server</artifactId>
    <version>1.0-SNAPSHOT</version>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.4.RELEASE</version>
    </parent>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Finchley.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
</project>
```

UP

```
package com.logicbig.example;

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

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

UP

```
server:
  port: 7777
eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
```

UP

```
<?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.logicbig.example</groupId>
    <artifactId>hello-service</artifactId>
    <version>1.0-SNAPSHOT</version>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```



```

    <version>2.0.4.RELEASE</version>
  </parent>
  <properties>
    <java.version>1.8</java.version>
  </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>
  </dependencies>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>Finchley.SR1</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>

```

UP

```

package com.logicbig.example;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.concurrent.atomic.AtomicLong;

@RestController

```

```
public class HelloController {  
    private AtomicLong counter = new AtomicLong();  
  
    @GetMapping("/hello")  
    public HelloObject getHelloWordObject() {  
        HelloObject hello = new HelloObject();  
        hello.setMessage("Hi there! you are number " + counter.incrementAndGet());  
        return hello;  
    }  
}
```

[UP](#)

```
package com.logicbig.example;  
  
public class HelloObject {  
    private String message;  
  
    public String getMessage() {  
        return message;  
    }  
  
    public void setMessage(String message) {  
        this.message = message;  
    }  
}
```

[UP](#)

```
package com.logicbig.example;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication
```

```
@EnableDiscoveryClient
public class HelloServiceMain{

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

UP

```
eureka.client.serviceUrl.defaultZone=http://localhost:7777/eureka/
```

UP

```
spring.application.name=hello-service
```

UP

```
<?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.logicbig.example</groupId>
    <artifactId>hello-web-client-service</artifactId>
    <version>1.0-SNAPSHOT</version>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.4.RELEASE</version>
    </parent>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
```

```

        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Finchley.SR1</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
</project>

```

UP

```

package com.logicbig.example;

public class HelloObject {
    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}

```

```
}  
}
```

UP

```
package com.logicbig.example;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.cloud.client.ServiceInstance;  
import org.springframework.cloud.client.discovery.DiscoveryClient;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.client.RestTemplate;  
import java.time.LocalDateTime;  
import java.util.List;  
  
@Controller  
public class HelloWebClientController {  
    @Autowired  
    private DiscoveryClient discoveryClient;  
  
    @GetMapping("/")  
    public String handleRequest(Model model) {  
        //accessing hello-service  
        List<ServiceInstance> instances = discoveryClient.getInstances("hello-service");  
        if (instances != null && instances.size() > 0) {//todo: replace with a load balancing mechanism  
            ServiceInstance serviceInstance = instances.get(0);  
            String url = serviceInstance.getUri().toString();  
            url = url + "/hello";  
            RestTemplate restTemplate = new RestTemplate();  
            HelloObject helloObject = restTemplate.getForObject(url,  
                HelloObject.class);  
            model.addAttribute("msg", helloObject.getMessage());  
            model.addAttribute("time", LocalDateTime.now());  
        }  
        return "hello-page";  
    }  
}
```

```
}  
}
```

UP

```
package com.logicbig.example;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class HelloWebClientServiceMain {  
  
    public static void main(String[] args) {  
        SpringApplication.run(HelloWebClientServiceMain.class, args);  
    }  
}
```

UP

```
server.port=9080  
eureka.client.serviceUrl.defaultZone=http://localhost:7777/eureka/
```

UP

```
spring.application.name=hello-web-client-service
```

UP



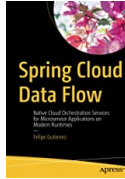





```
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml"  
    xmlns:th="http://www.thymeleaf.org">
```

```

<body>
<h2>Hello Page</h2>
<div th:text="{msg}" />
<div>Time: <span th:text="{time}" /></div>
</body>
</html>

```

UP

 <p>Cloud Native Java: Designing Resilient Systems with Spring Boot, Spring Cloud...</p> <p>\$36.79 \$69.99</p> <p>(60)</p>	 <p>Cloud Native Spring in Action</p> <p>\$59.99</p>	 <p>Spring Cloud Data Flow: Native Cloud Orchestration Services for Microserv...</p> <p>\$59.67 \$69.99</p> <p>(4)</p>	 <p>Hands-On Microservices with Spring Boot and Spring Cloud: Build and deploy ...</p> <p>\$62.99</p> <p>(114)</p>
 <p>Cloud Native Microservices with Spring and Kubernetes: Design and Build Mod...</p> <p>\$29.95</p> <p>(41)</p>	 <p>Mastering Microservices with Java: Build enterprise microservices with Sprin...</p> <p>\$46.69</p> <p>(21)</p>	 <p>Microsoft Azure for Java Developers: Deploying Java Applications through Azure WebApp, Azure Kubernetes Ser...</p> <p>\$52.71</p>	 <p>Spring Microservices</p> <p>\$32.01 \$64.99</p> <p>(37)</p>

Ads by Amazon

See Also

[Using RestTemplate as a Load Balancer Client with Netflix Ribbon](#)

[Load Balancing And Running Multiple Instances of a Microservice](#)

[Spring Cloud Config with File System Backend](#)

[Spring Cloud Config with Git Backend](#)

[Circuit Breaker Hystrix Basics](#)

[Hystrix Circuit Breaker, getting failure exception in fallback method](#)

[Hystrix Circuit Breaker, Setting Configuration Properties Using @HystrixProperty](#)[Circuit Breaker, Specifying Hystrix configuration in application.properties file](#)[Circuit Breaker Hystrix, concurrent requests and default thread pool size](#)[Circuit Breaker Hystrix, Changing Default Thread Pool Properties](#)[Circuit Breaker Hystrix Event Listener](#)[Hystrix CircuitBreaker, Thread Local Context Propagation](#)

Core Java Tutorials

[Java 16 Features](#)[Java 15 Features](#)[Java 14 Features](#)[Java 13 Features](#)[Java 12 Features](#)[Java 11 Features](#)[Java 10 Features](#)[Java 9 Module System](#)[Java 9 Misc Features](#)[Java 9 JShell](#)

Recent Tutorials

[Python - Tuples](#)[Python - Lists](#)[Python - Naming Conventions](#)[Python - Built In Data Structures](#)[Python None Type](#)[Python Boolean](#)[Python Keywords](#)[Python String](#)[Python Numbers](#)[Python Variables](#)[Python Comments](#)[Python Basic Syntax](#)[Python with IntelliJ](#)[Python language getting started](#)[Installing Python 3.10.x on windows](#)

Share 

[Next Page](#)