**SpringCloud**

**第一篇：SpringCloud教程第1篇：Eureka**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/01/sc01-eureka.html>

eureka\_client\_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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.kin.springcloud.eureka</groupId>

<artifactId>eureka\_client</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>eureka\_client</name>

<description>Demo project for Spring Boot</description>

<parent>

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

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

<version>2.0.2.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>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-eureka</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Finchley.RELEASE</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-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

eureka\_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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.kin.springcloud.eureka</groupId>

<artifactId>euraka\_server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>euraka\_server</name>

<description>Demo project for Spring Boot</description>

<parent>

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

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

<version>2.0.2.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>

</properties>

<dependencies>

<!--eureka server -->

<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-eureka-server -->

<dependency>

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

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

<version>1.4.7.RELEASE</version>

</dependency>

<!-- spring boot test-->

<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>Finchley.RELEASE</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-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

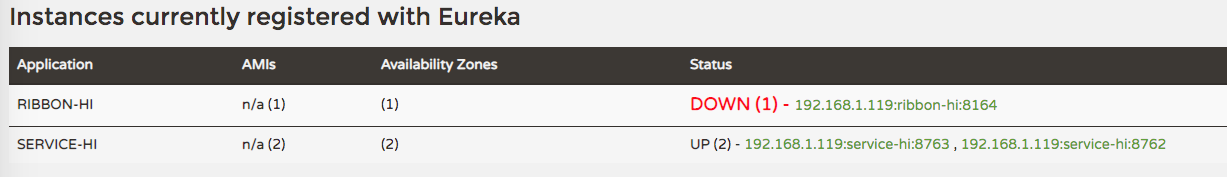
</repositories>

</project>

Eureka主要作用：在微服务架构中，业务都会被拆分成一个独立的服务。在这里，我们需要用的的组件上Spring Cloud Netflix的Eureka ,eureka是一个服务注册和发现模块。（据我理解，一个项目中分了许多模块，由于每个模块转为每一个对应的服务，因此转换为微服务之后（它需要成为eureka的client），通过使用eureka组件在服务注册中心注册服务（服务注册中心也是通过eureka组件创建，其实他也是一个client，只是在配置上使它成为了server））

访问eureka服务器图形化：直接访问服务器的ip:端口

例如：http://localhost:8761



**第二篇：SpringCloud教程第2篇：Ribbon**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/02/sc02-rest-ribbon.html>

ribbon\_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.kin.springcloud.ribbon</groupId>

<artifactId>springcloud\_ribbon</artifactId>

<version>1.0-SNAPSHOT</version>

<name>springcloud\_ribbon</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<parent>

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

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

<version>2.0.2.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>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-eureka</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-ribbon -->

<dependency>

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

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

<version>1.4.7.RELEASE</version>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Finchley.RELEASE</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-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

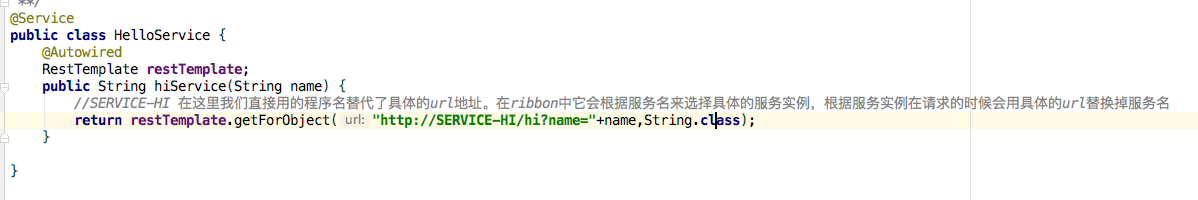
</snapshots>

</repository>

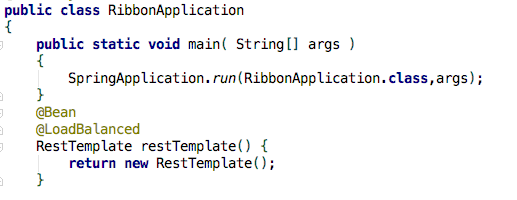
</repositories>

</project>

Ribbon主要作用是：ribbon是一个负载均衡客户端，可以很好的控制http和tcp的一些行为。Feign默认集成了ribbon。 并且能够服务与服务之间建立联系。







**第三篇：SpringCloud教程第3篇：feign**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/03/sc03-feign.html>

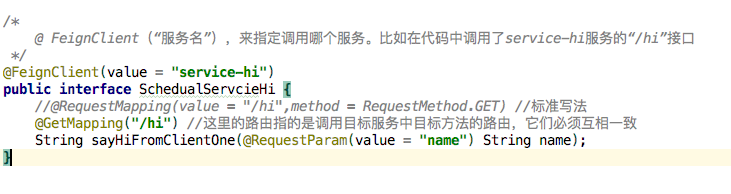
Feign默认集成了Ribbon，并和Eureka结合，默认实现了负载均衡的效果。

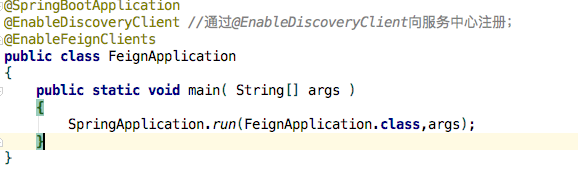
Springcloud\_feign\_pom.xml:

<?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.kin.springcloud.feign</groupId>  
 <artifactId>springcloud\_feign</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.2.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <name>springcloud\_feign</name>  
 <!-- FIXME change it to the project's website -->  
 <url>http://www.example.com</url>  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka</artifactId>  
 <version>1.4.7.RELEASE</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-feign -->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-feign</artifactId>  
 <version>1.4.7.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Finchley.RELEASE</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
 <build>  
 <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->  
 <plugins>  
 <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean\_Lifecycle -->  
 <plugin>  
 <artifactId>maven-clean-plugin</artifactId>  
 <version>3.1.0</version>  
 </plugin>  
 <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_jar\_packaging -->  
 <plugin>  
 <artifactId>maven-resources-plugin</artifactId>  
 <version>3.0.2</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.0</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.22.1</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-jar-plugin</artifactId>  
 <version>3.0.2</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-install-plugin</artifactId>  
 <version>2.5.2</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>2.8.2</version>  
 </plugin>  
 <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->  
 <plugin>  
 <artifactId>maven-site-plugin</artifactId>  
 <version>3.7.1</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-project-info-reports-plugin</artifactId>  
 <version>3.0.0</version>  
 </plugin>  
 </plugins>  
 </pluginManagement>  
 </build>  
</project>

Feign的主要作用是：实现比ribbon更加好的实现负载均衡，其实是因为他默认集成了ribbon，并和euraka结合。Feign是一个声明式的伪Http客户端，它使得写Http客户端变得更简单。使用Feign，只需要创建一个接口并注解。它具有可插拔的注解特性，可使用Feign 注解和JAX-RS注解。

效果实现与第二章基本一致。





**第四篇：SpringCloud教程第4篇：Hystrix**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/04/sc04-hystrix.html>

由于网络原因或者自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，导致服务瘫痪。服务与服务之间的依赖性，故障会传播，会对整个微服务系统造成灾难性的严重后果，这就是服务故障的“雪崩”效应。

Netflix开源了Hystrix组件，实现了断路器模式，SpringCloud对这一组件进行了整合。断路由：在访问某一个服务的时候，该服务故障或者没有启动时，缺少返回值，会影响其他功能，因此断路由在此添加一个默认返回值解决问题。

当 service-hi 工程不可用的时候，service-ribbon调用 service-hi的API接口时，会执行快速失败，直接返回一组字符串，而不是等待响应超时，这很好的控制了容器的线程阻塞。默认值是由开发人员定义的。

以下有两个实现方法：

1. 改造ribbon：

ribbon\_pom.xml:

添加依赖：

<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-hystrix -->

<dependency>

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

<artifactId>spring-cloud-starter-hystrix</artifactId>

<version>1.4.7.RELEASE</version>

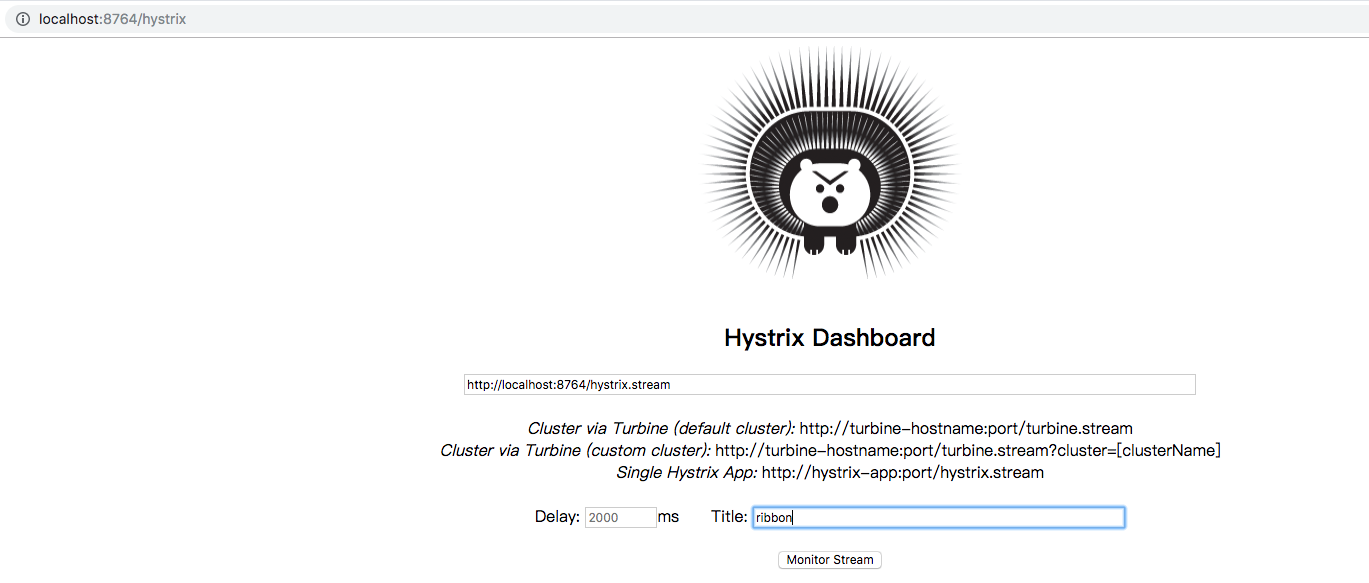
</dependency>

1. 改造Feign

Feign是自带断路器的，在D版本的Spring Cloud中，它没有默认打开。需要在配置文件中配置打开它

Application.yml配置文件添加：feign.hystrix.enabled=true

Hystrix Dashboard (断路器：Hystrix 仪表盘) *hystrix-dashboard用于查看当前断路由状态，有图形界面访问访问http://localhost:当前工程端口号/hystrix*





Tips：根据图片内容输入一直提示连接失败需要修改主程序：添加以下代码

@Bean  
**public** ServletRegistrationBean getServlet***() {*** HystrixMetricsStreamServlet streamServlet = **new** HystrixMetricsStreamServlet***()***;  
 ServletRegistrationBean registrationBean = **new** ServletRegistrationBean***(***streamServlet***)***;  
 registrationBean.setLoadOnStartup***(***1***)***;  
 registrationBean.addUrlMappings***(*"/hystrix.stream"*)***;  
 registrationBean.setName***(*"HystrixMetricsStreamServlet"*)***;  
 **return** registrationBean;  
***}***

基于service-ribbon 改造，Feign的改造和这一样,首选在pom.xml引入spring-cloud-starter-hystrix-dashboard的起步依赖：

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-actuator -->

<dependency>

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

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

<version>2.2.0.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-hystrix-dashboard -->

<dependency>

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

<artifactId>spring-cloud-starter-hystrix-dashboard</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

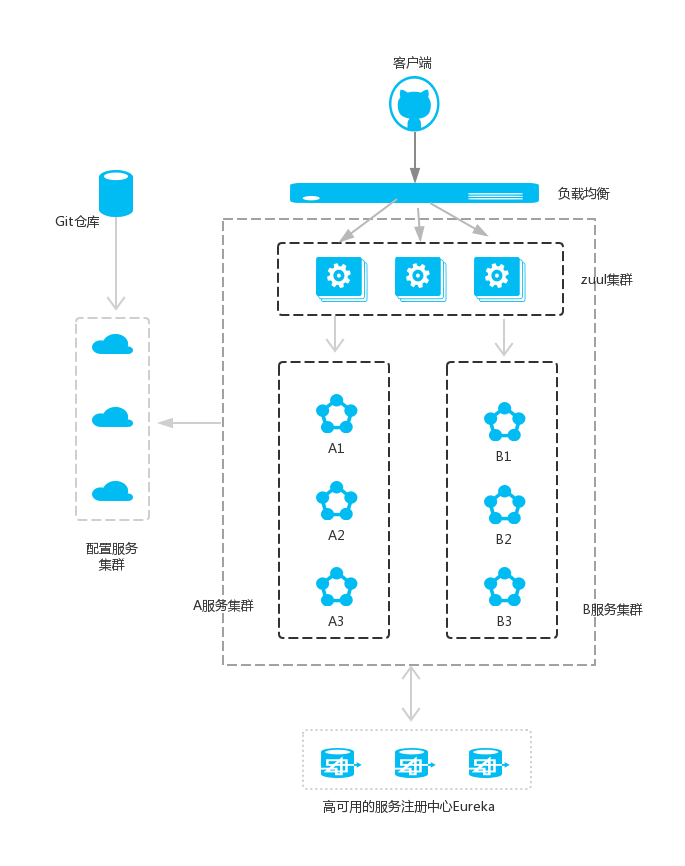


加入该注解即可访问hystrix仪表

**第五篇：SpringCloud教程第5篇：Zuul（F版本）**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2018/08/05/sc-f5-zuul.html>

在微服务架构中，需要几个基础的服务治理组件，包括服务注册与发现、服务消费、负载均衡、断路器、智能路由、配置管理等，由这几个基础组件相互协作，共同组建了一个简单的微服务系统。



Zuul的主要功能是路由转发和过滤器。路由功能是微服务的一部分，比如／api/user转发到到user服务，/api/shop转发到到shop服务。zuul默认和Ribbon结合实现了负载均衡的功能。

Springcloud\_zuul\_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.kin.springcloud.zuul</groupId>  
 <artifactId>springcloud\_zuul</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.2.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <name>springcloud\_zuul</name>  
 <!-- FIXME change it to the project's website -->  
 <url>http://www.example.com</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
  
  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-zuul -->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-zuul</artifactId>  
 <version>1.4.7.RELEASE</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka</artifactId>  
 <version>1.4.7.RELEASE</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-ribbon -->  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Finchley.RELEASE</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-milestones</id>  
 <name>Spring Milestones</name>  
 <url>https://repo.spring.io/milestone</url>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 </repository>  
 </repositories>  
</project>

zuul不仅只是路由，并且还能过滤，做一些安全验证。继续改造工程；

/\*

filterType：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型，具体如下：

pre：路由之前

routing：路由之时

post： 路由之后

error：发送错误调用

filterOrder：过滤的顺序

shouldFilter：这里可以写逻辑判断，是否要过滤，本文true,永远过滤。

run：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问。

\*/

@Component

public class MyFilter extends ZuulFilter {

private static Logger log = LoggerFactory.getLogger(MyFilter.class);

@Override

public String filterType() {

return "pre";

}

@Override

public int filterOrder() {

return 0;

}

@Override

public boolean shouldFilter() {

return true;

}

@Override

public Object run() {

RequestContext ctx = RequestContext.getCurrentContext();

HttpServletRequest request = ctx.getRequest();

log.info(String.format("%s >>> %s", request.getMethod(), request.getRequestURL().toString()));

Object accessToken = request.getParameter("token");

if(accessToken == null) {

log.warn("token is empty");

ctx.setSendZuulResponse(false);

ctx.setResponseStatusCode(401);

try {

ctx.getResponse().getWriter().write("token is empty");

}catch (Exception e){}

return null;

}

log.info("ok");

return null;

}

}

**第六篇：SpringCloud教程第6篇：config**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/06/sc06-config.html>

主要作用：在分布式系统中，由于服务数量巨多，为了方便服务配置文件统一管理，实时更新，所以需要分布式配置中心组件。在Spring Cloud中，有分布式配置中心组件spring cloud config ，它支持配置服务放在配置服务的内存中（即本地），也支持放在远程Git仓库中。在spring cloud config 组件中，分两个角色，一是config server，二是config client。

**第七篇：SpringCloud教程第7篇：高可用的分布式配置中心**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/07/sc07-config.html>

主要作用：上一篇文章讲述了一个服务如何从配置中心读取文件，配置中心如何从远程git读取配置文件，当服务实例很多时，都从配置中心读取文件，这时可以考虑将配置中心做成一个微服务，将其集群化，从而达到高可用

**第八篇：SpringCloud教程第8篇：Spring Cloud Bus**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2017/06/08/sc08-bus.html>

主要作用：Spring Cloud Bus 将分布式的节点用轻量的消息代理连接起来。它可以用于广播配置文件的更改或者服务之间的通讯，也可以用于监控。本文要讲述的是用Spring Cloud Bus实现通知微服务架构的配置文件的更改。（需要使用到rabbitMQ）

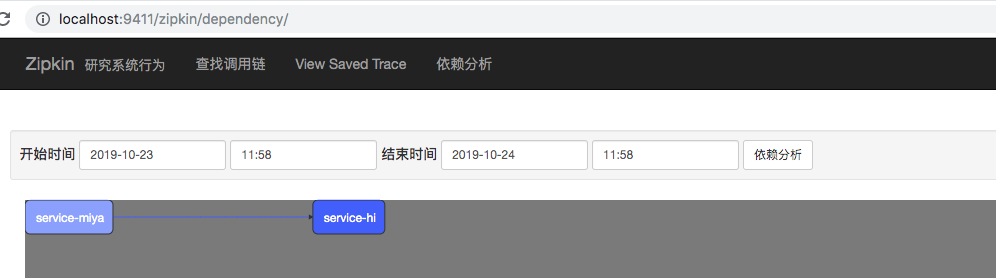
**第九篇：SpringCloud教程第9篇：Sleuth（F版本）**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2018/08/09/sc-f9-sleuth.html>

Spring Cloud Sleuth 主要功能就是在分布式系统中提供追踪解决方案，并且兼容支持了 zipkin，你只需要在pom文件中引入相应的依赖即可。

在spring Cloud为F版本的时候，已经不需要自己构建Zipkin Server了，只需要下载jar即可，下载地址：<https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/>

 java -jar zipkin-server-2.11.1-exec.jar



运行即可访问localhost:9411

其中案例中service-hi与service-miya pom文件一致：

root\_pom:

***<?*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.kin.springcloud.sleuth***</*groupId*>  
 <*artifactId*>***springcloud\_sleuth***</*artifactId*>  
 <*version*>***1.0-SNAPSHOT***</*version*>  
  
 <*name*>***springcloud\_sleuth***</*name*>*** *<!--* ***FIXME change it to the project's website*** *-->* ***<*url*>***http://www.example.com***</*url*>  
 <*parent*>  
 <*groupId*>***org.springframework.boot***</*groupId*>  
 <*artifactId*>***spring-boot-starter-parent***</*artifactId*>  
 <*version*>***2.0.3.RELEASE***</*version*>  
 <*relativePath*/>  
 </*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.RELEASE***</*spring-cloud.version*>  
 </*properties*>  
  
 <*dependencies*>  
 <*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*>  
  
  
</*project*>***

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 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  
 <*modelVersion*>***4.0.0***</*modelVersion*>  
  
 <*groupId*>***com.kin.springcloud.zipkin\_service\_hi***</*groupId*>  
 <*artifactId*>***springcloud\_zipkin\_service\_hi***</*artifactId*>  
 <*version*>***0.0.1-SNAPSHOT***</*version*>  
 <*name*>***springcloud\_zipkin\_service\_hi***</*name*>  
 <*description*>***Demo project for Spring Boot***</*description*>  
 <*parent*>  
 <*groupId*>***com.kin.springcloud.sleuth***</*groupId*>  
 <*artifactId*>***springcloud\_sleuth***</*artifactId*>  
 <*version*>***1.0-SNAPSHOT***</*version*>  
 </*parent*>  
 <*dependencies*>  
  
 <*dependency*>  
 <*groupId*>***org.springframework.boot***</*groupId*>  
 <*artifactId*>***spring-boot-starter-web***</*artifactId*>  
 </*dependency*>*** *<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-zipkin -->* ***<*dependency*>  
 <*groupId*>***org.springframework.cloud***</*groupId*>  
 <*artifactId*>***spring-cloud-starter-zipkin***</*artifactId*>  
 <*version*>***2.0.0.RELEASE***</*version*>  
 </*dependency*>  
  
  
  
  
 </*dependencies*>  
  
 <*build*>  
 <*plugins*>  
 <*plugin*>  
 <*groupId*>***org.springframework.boot***</*groupId*>  
 <*artifactId*>***spring-boot-maven-plugin***</*artifactId*>  
 </*plugin*>  
 </*plugins*>  
 </*build*>  
  
</*project*>***

**第十篇：SpringCloud教程第10篇：高可用的服务注册中心（F版本）**

详细创建步骤参考：<https://www.fangzhipeng.com/springcloud/2018/08/10/sc-f10-eureka.html>

Eureka-eserver peer1 8761,Eureka-eserver peer2 8769相互感应，当有服务注册时，两个Eureka-eserver是对等的，它们都存有相同的信息，这就是通过服务器的冗余来增加可靠性，当有一台服务器宕机了，服务并不会终止，因为另一台服务存有相同的数据。