Spring Cloud

微服务架构解决方案

传智.燕青

# 课程目标

注意：结合源码<https://github.com/caojx-git/learn/tree/master/code/spring-cloud> 可以更快入手

1、能说出微服务架构特点。

2、能说出微服务技术栈的构成。

3、理解Spring Cloud Eureka服务治理的流程。

4、理解Spring Cloud Ribbon负载均衡的应用方法。

5、理解Spring Cloud Hystrix 断路保护的应用方法。

6、理解Spring Cloud Zuul 网关的作用及方法。

7、了解Spring Cloud Config、Spring Cloud Bus、Spring Cloud Sleuth等其它子项目的应用场景。

# 理解微服务

## 软件架构演进

软件架构的发展经历了从单体结构、垂直架构、SOA架构到微服务架构的过程。

### 单体架构



特点：

1、所有的功能集成在一个项目工程中。

2、所有的功能打一个war包部署到服务器。

3、应用与数据库分开部署。

4、通过部署应用集群和数据库集群来提高系统的性能。

优点：

1、项目架构简单，前期开发成本低，周期短，小型项目的首选。

缺点：

1、全部功能集成在一个工程中，对于大型项目不易开发、扩展及维护。

2、系统性能扩展只能通过扩展集群结点，成本高、有瓶颈。

3、技术栈受限。

### 垂直架构



特点：

1、以单体结构规模的项目为单位进行垂直划分项目即将一个大项目拆分成一个一个单体结构项目。

2、项目与项目之间的存在数据冗余，耦合性较大，比如上图中三个项目都存在客户信息。

3、项目之间的接口多为数据同步功能，如：数据库之间的数据库，通过网络接口进行数据库同步。

优点：

1、项目架构简单，前期开发成本低，周期短，小型项目的首选。

2、通过垂直拆分，原来的单体项目不至于无限扩大。

3、不同的项目可采用不同的技术。

缺点：

1、全部功能集成在一个工程中，对于大型项目不易开发、扩展及维护。

2、系统性能扩展只能通过扩展集群结点，成本高、有瓶颈。

### SOA架构



特点：

1、基于SOA的架构思想将重复公用的功能抽取为**组件**，以服务的方式给各各系统提供服务。

2、各各项目（系统）与服务之间采用webservice、rpc等方式进行通信。

3、ESB企业服务总线作为项目与服务之间通信的桥梁。

优点：

1、将重复的功能抽取为服务，提高开发效率，提高系统的可重用性、可维护性。

2、可以针对不同服务的特点制定集群及优化方案。

3、采用ESB减少系统中的接口耦合。

缺点：

1、系统与服务的界限模糊，不利于开发及维护。

2、虽然使用了ESB，但是服务的接口协议不固定，种类繁多，不利于系统维护。

3、抽取的服务的粒度过大，系统与服务之间耦合性高。

### 微服务架构



特点：

1、将系统服务层完全独立出来，并将服务层抽取为一个一个的微服务。

2、微服务遵循单一原则。

3、微服务之间采用RESTful等轻量协议传输。

优点：

1、服务拆分粒度更细，有利于资源重复利用，提高开发效率。

2、可以更加精准的制定每个服务的优化方案，提高系统可维护性。

3、微服务架构采用去中心化思想，服务之间采用RESTful等轻量协议通信，相比ESB更轻量。

4、适用于互联网时代，产品迭代周期更短。

缺点：

1、微服务过多，服务治理成本高，不利于系统维护。

2、分布式系统开发的技术成本高（容错、分布式事务等），对团队挑战大。

## 什么是微服务

为适应企业的业务发展，提高软件研发的生产力，降低软件研发的成本，软件架构也作了升级和优化，将一个独立的系统拆分成若干小的服务，每个小服务运行在不同的进程中，服务与服务之间采用http 轻量协议（比如流行的RESTful）传输数据，每个服务所拥有的功能具有独立性强、高内聚的特点，这样的设计就实现了单个服务的高内聚，服务与服务之间的低耦合效果，这一个一个的小服务就是微服务，基于这种方法设计的系统架构即微服务架构。

## Spring Cloud技术栈

Spring Cloud官网：http://projects.spring.io/spring-cloud/

### 微服务的技术栈



负载均衡，网关路由：高可用、集群部署，校验、请求转发、服务集成。

服务治理：服务注册、发现。

容错：避免雪崩。

监控跟踪：监控资源利用、服务响应、容器资源利用情况。

消息总线：消息队列、异步通信。

配置管理：统一配置管理。

### Spring Cloud是什么

Spring Cloud为开发人员构建微服务架构提供了完整的解决方案，SpringCloud是若干个框架的集合，它包括spring-cloud-config、spring-cloud-bus等近20个子项目，它提供了服务治理、服务网关、智能路由、负载均衡、断路器、监控跟踪、分布式消息队列、配置管理等领域的解决方案。

### Spring Cloud技术栈

微服务的兴起出现了很多优秀的公司和技术：

服务治理：Dubbo(阿里巴巴)、Dubbox(当当)、Eureka（Netflix）等 。

配置管理：Disconf（百度）、QConf（360）、Diamood（淘宝）等 。

服务跟踪：Hydra(京东)、Zipkin（Twitter）、Sleuth（Spring Cloud）等 。

Spring Cloud 提供一站式的微服务架构解决方案，如下图：



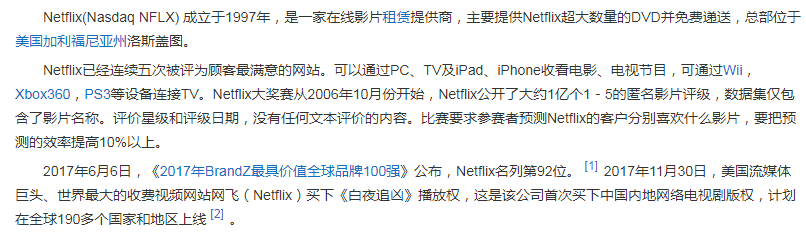
### 为什么使用Spring Cloud

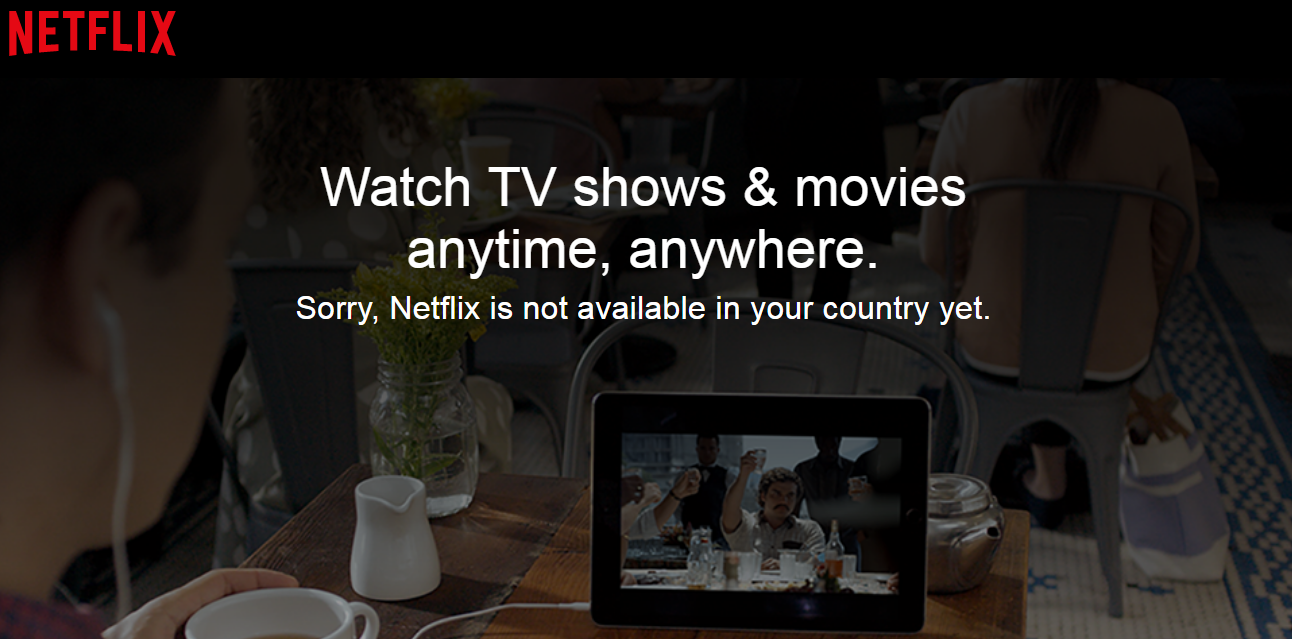
微服务架构的优点表明它可以提高我们的生产力，但是分布式系统本身的技术成本问题给互联网那些创业型公司不少的挑战，阿里、百度等巨头所提供的微服务技术只是解决其中某个问题，而整合封装这些优秀的技术恐怕是Spring最擅长的领域了，Spring Cloud也正因为此而诞生。

使用Spring Cloud来构建微服务架构可以省去你整合各家技术的成本，Spring Cloud为我们构建微服务架构提供了一站式的解决方案，就好比当初Spring诞生是为解决EJB企业应用开发的众多问题而提供的一站式轻量级企业应用开发解决方案一样，随着使用Spring Cloud公司数量的增加，相信微服务将被Spring Cloud一统江湖。

### Netflix公司介绍

Spring Cloud的很多技术来源于Netfix（https://netflix.github.io/），摘自百度百科的信息如下：





# 开发环境

## Java8

Spring Cloud是基于Java构建，本课程使用Java8作为基础平台。

## Maven

优秀的架构离不开优秀的项目构建工具，本课程采用Maven来构建（使用apache-maven-3.3.9-bin）。

## Spring Boot

Spring Cloud是基于Spring Boot构建，本课程使用Spring Boot 1.5.4版本。

<parent>

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

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

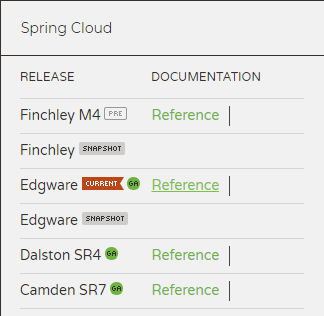
<version>1.5.4.RELEASE</version>

</parent>

## Spring Cloud

Spring Cloud为了避免和各子项目的版本名称混淆，它采用伦敦地铁站命名。

当前版本情况如下：



本课程 使用Dalston.SR3版本。

<!-- 导入Spring Cloud的依赖管理 -->

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Dalston.SR3</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

## MySQL

MySQL作为数据库的第二把交椅甚至直逼Oralce这个老大，在互联网开发中MySQL的应用是最广泛的。本课程采用MySQL5来构建架构。

## IDEA

IntelliJ IDEA功能的强大及易用性不亚于Eclipse，据统计其使用人数已直逼Eclipse，成为老大指日可待。

本课程 使用IDEA作为开发工具。

# 服务治理

## 什么是服务治理

微服务架构的缺点中最主要的就是由于微服务数量众多导致维护成本巨大，服务治理为解决此问题而产生的。服务治理的作用是让维护人员从人工维护中解放出来，由服务自维护，微服务作为服务提供方主动向服务治理中心注册，服务的消费方通过服务治理中心查询需要的服务并进行调用。

如下图：



## Spring Cloud Eureka

Spring Cloud Eureka 是对Netflix公司的Eureka的二次封装，它实现了服务治理的功能，Spring Cloud Eureka提供服务端与客户端，服务端即是服务注册中心，客户端完成服务的注册与发现。服务端和客户端均采用Java语言编写（Eureka支持多语言）。

如下图显示了Eureka Server与Eureka Client的关系：



## 架构



## 实战

### 流程图



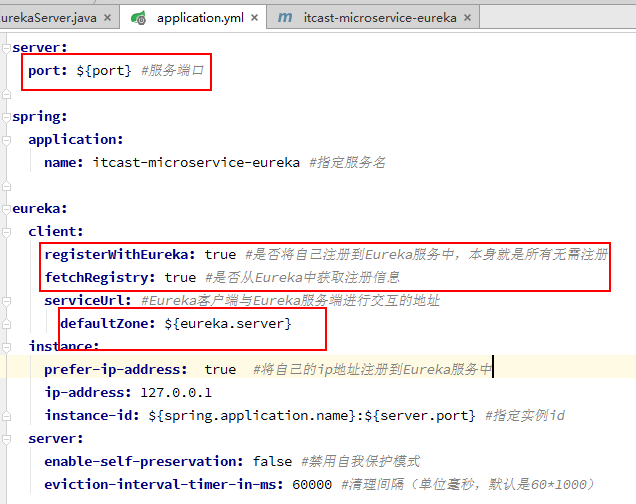
### 开发并部署Eureka Server

1、创建Spring Boot工程

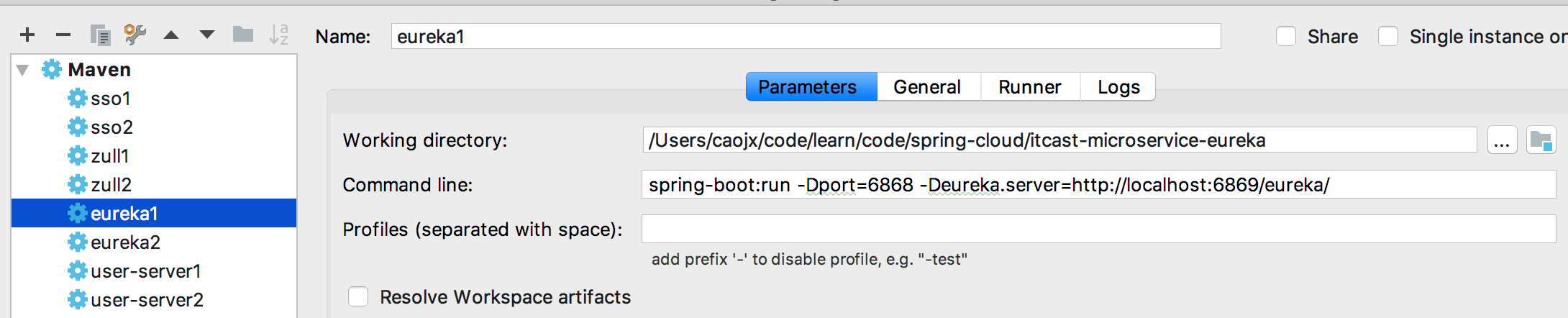
2、在pom.xml中添加依赖（spring boot 、spring cloud、Eureka Server）

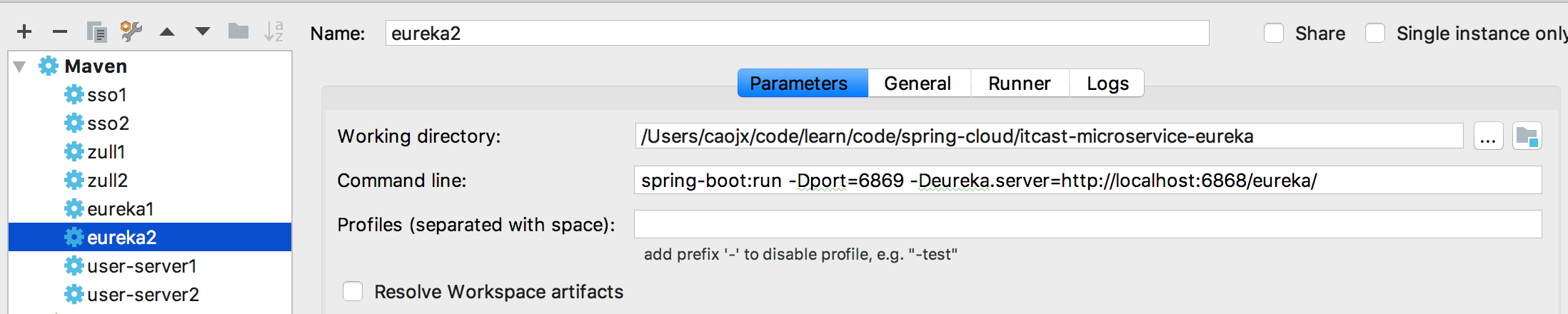
3、配置application.yml

4、部署两台Eureka Server，并且互相注册，实现高可用。



注意：如果没有设置Eureka Server的复制结点eureka默认会找8761端口。





### 开发用户信息服务

用户信息服务实现用户信息查询、用户注册、用户信息修改等功能。

开发用户信息服务采用Spring Boot、SpringMVC、Mybatis架构，数据库采用MySqL5。

### 服务注册

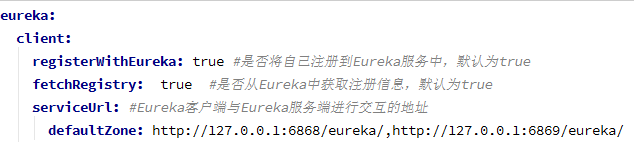
服务注册完成将用户信息服务注册到Eureka Server中，供服务消费方查询。

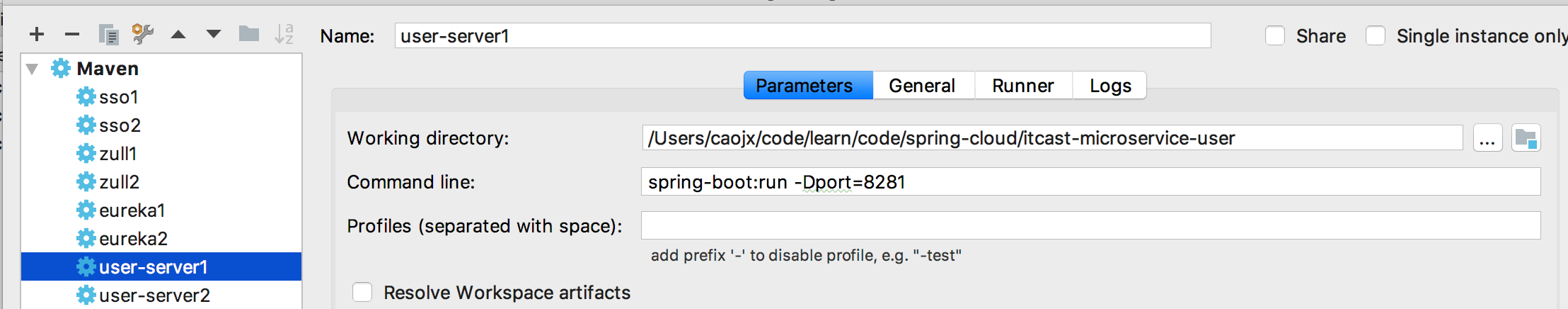
1、在用户信息服务中配置Eureka服务中心地址及用户信息服务名称。

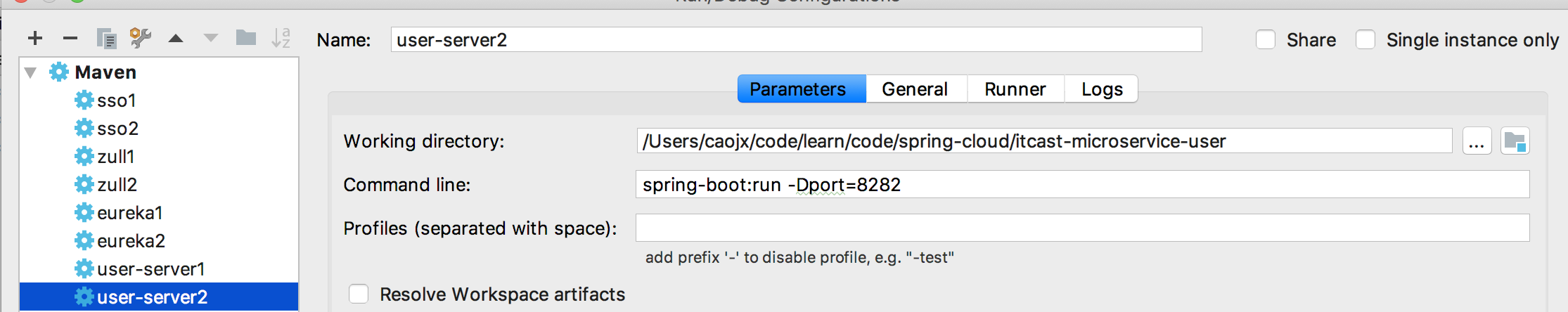
2、在SpringBoot的启动类中添加注解@EnableDiscoveryClient

3、注意Eureka服务中心地址配置两个，因为有两台Eureka服务中心。

4、用户服务注册中心启动两个服务，为后边测试负载均衡准备。







# 负载均衡

## 什么是负载均衡

负载均衡是微服务架构中必须使用的技术，通过负载均衡来实现系统的高可用、集群扩容等功能。负载均衡可通过硬件设备及软件来实现，硬件比如：F5、Array等 ，软件比如：LVS、Nginx等 。

如下图是负载均衡的架构图：



用户请求先到达负载均衡器（也相当于一个服务），负载均衡器根据负载均衡算法将请求转发到微服务。负载均衡算法有：轮训、随机、加权轮训、加权随机、地址哈希等方法，负载均衡器维护一份服务列表，根据负载均衡算法将请求转发到相应的微服务上，所以负载均衡可以为微服务集群分担请求，降低系统的压力。

## Spring Cloud Ribbon

Spring Cloud Ribbon是基于客户端的负载均衡工具，负载均衡分为服务端负载均衡和客户端负载均衡，3.1小节的图形指的是服务端负载均衡，客户端负载均衡与服务端负载均衡的区别在于客户端要维护一份服务列表，Ribbon从Eureka Server获取服务列表，Ribbon根据负载均衡算法直接请求到具体的微服务，中间省去了负载均衡服务。

如下图是Ribbon负载均衡的流程图：



1、在消费微服务中使用Ribbon实现负载均衡，Ribbon先从Eureka Server中获取服务列表。

2、Ribbon根据负载均衡的算法进行负载均衡，将请求转发到其它微服务。

## 实战

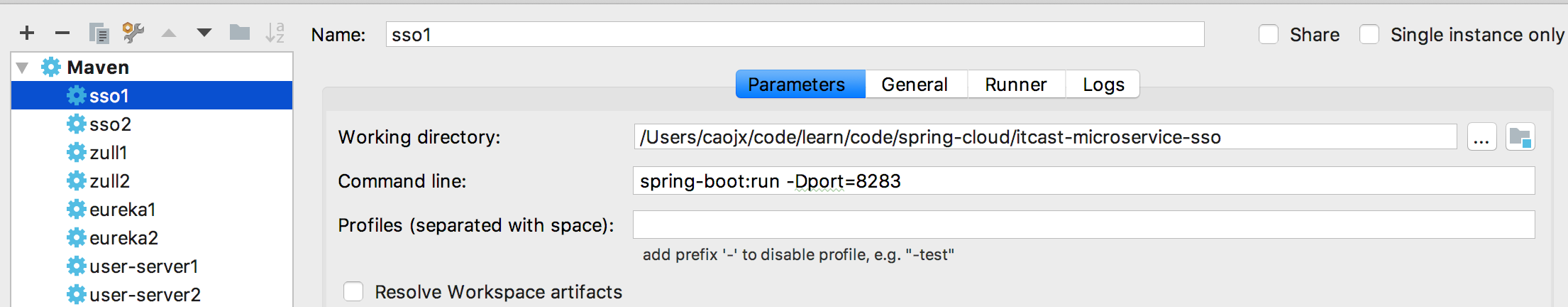


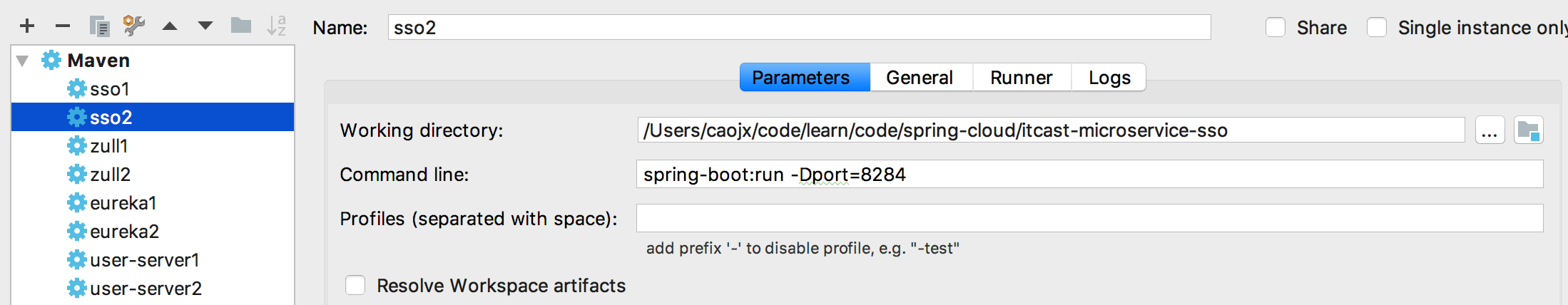
### 开发单点登录服务

单点登录服务提供用户登录、用户退出等功能。

单点登录服务需要调用用户信息服务查询用户信息。

单点登录服务采用Spring Boot、SpringMVC开发。





### 服务发现

单点登录服务需要从EurekaServer查询用户信息服务。

1、配置Eureka Server服务地址。

2、在SpringBoot的启动类中添加注解@EnableDiscoveryClient



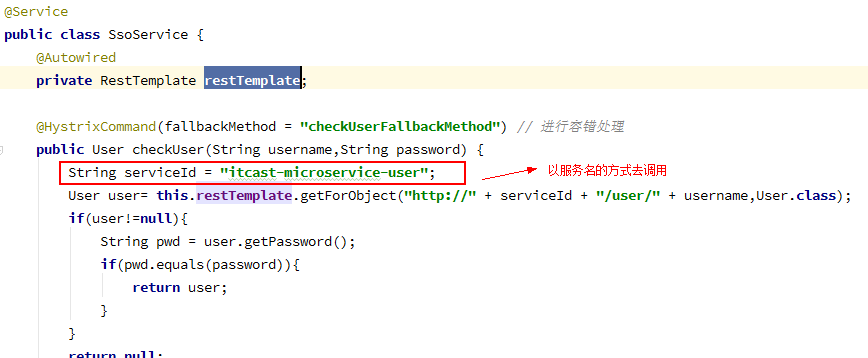
### Ribbon编程

1、定义RestTemplate对象

2、配置 @LoadBalanced

3、设置负载均衡算法。

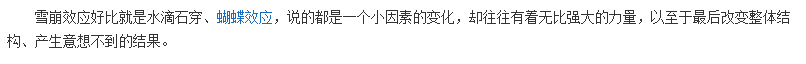
4、使用RestTemplate调用微服务。



# 容错保护

## 什么是容错保护

容错保护是指微服务在执行过程中出现错误并从错误中恢复的能力。微服务容错性不好很容易导致雪崩效应，什么是雪崩效应？摘自百度百科中的定义：



微服务的雪崩效应表现在服务与服务之间调用，当其中一个服务无法提供服务可能导致其它服务也死掉，比如：单点登录服务调用用户信息服务查询用户信息，由于用户信息服务无法提供服务导致单点登录服务一直等待，从而导致用户登录、用户退出功能无法使用，像这样由一个服务所引起的一连串的多个服务无法提供服务即是微服务的雪崩效应。

## Spring Cloud Hystrix

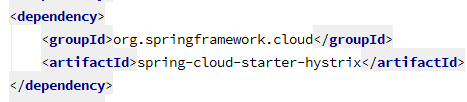
Spring Cloud Hystrix 是基于Netflix的开源框架Hystrix的整合，它实现了断路器、线程隔离、信号隔离等容错功能。

下图是Hystrix断路器示意图：



## 实战

1.要在服务消费方添加hystrix。



2、使用Hystrix实现容错。



3、在Spring boot的启动类上添加@enableHystrix注解



# 服务网关

## 什么是服务网关

服务网关是在微服务前边设置一道屏障，请求先到服务网关，网关会对请求进行过虑、校验、路由等处理。有了服务网关可以提高微服务的安全性，校验不通过的请求将被拒绝访问。

前边介绍的Ribbon客户端负载均衡技术可以不用经过网关，因为通常使用Ribbon完成微服务与微服务之间的内部调用，而对那些对外提供服务的微服务，比如：用户登录、提交订单等，则必须经过网关来保证微服务的安全。

## Spring Cloud Zuul

Spring Cloud Zuul是整合Netflix公司的Zuul开源项目实现的微服务网关，它实现了请求路由、负载均衡、校验过虑等 功能。



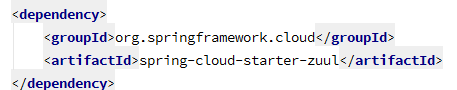
## 实战



1、部署用户信息服务A、单点登录服务B,每个服务部署至少两台机器。

2、将用户信息服务A、单点登录服务B注册到EurekaServer中。

3、开发并部署zuul。

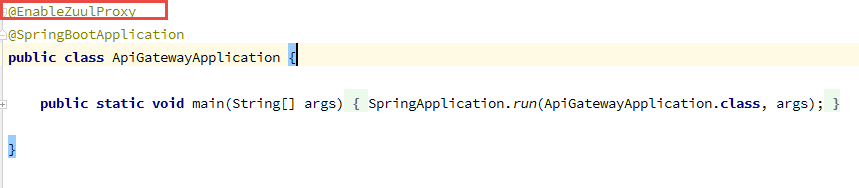


4、在zuul中配置路由

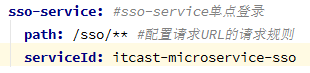


5、可以定义filter，需要集成zuul提供filter类，进行校验拦截。

6、在spring boot的启动类中配置注解

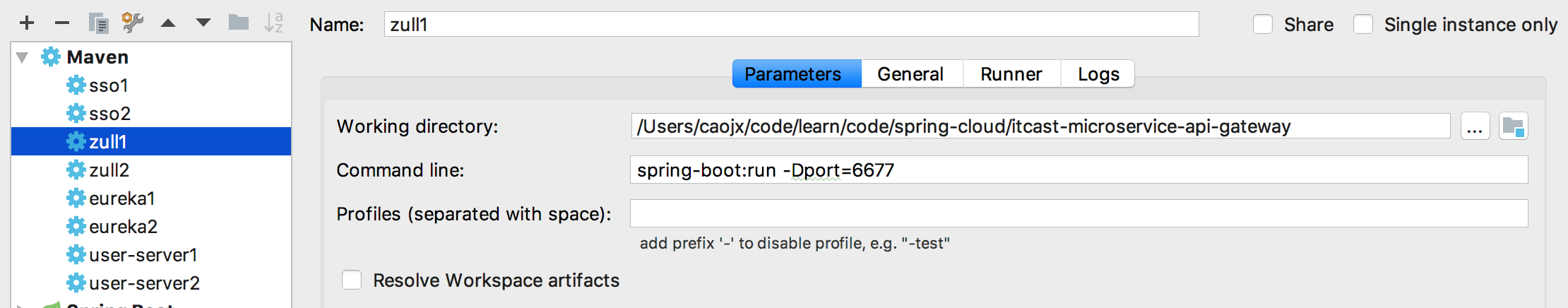


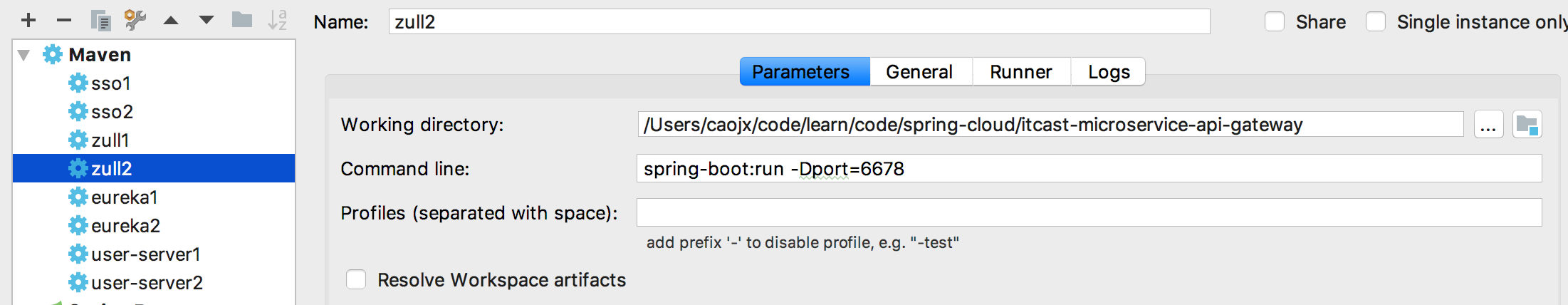
7、根据上边的路由配置访问微服务。



凡是以/sso/打头的请求，路由到 itcast-microservice-sso微服务。







## 总结

zuul网关，不仅提供对外服务访问 ，微服务也可以通过zuul请求其它的微服务。