# SpringCloud微服务课程说明

SpringCloud微服务课程由每特教育|蚂蚁课堂出品，该课程是基于SpringBoot2.0基础之上进行讲解+jdk1.8以上，更多分布式与微服务技术请关注蚂蚁课堂。

主讲老师:97后架构师-余胜军

余老师联系方式 QQ 644064779（已经加满）、QQ1051546329（未满）  
余老师微信号 yushengjun644

每特教育官方粉丝群:每特教育 193086273、SpringCloud微服务讨论群805985159

蚂蚁课堂官方网站www.itmayeidu.com

**上海每特教育科技有限公司|苏州特每信息科技有限公司版权所有**

SpringCloud中文翻译：https://springcloud.cc/spring-cloud-dalston.html

# 网站架构演变过程

## 传统架构

传统的SSH架构，分为三层架构 web控制层、业务逻辑层、数据库访问层。

传统架构也就是单点应用，就是大家在刚开始初学JavaEE技术的时候SSH架构或者SSM架构，业务没有进行拆分，都写同一个项目工程里面，一般是适合于个人或者是小团队开发。

这种架构模式，一旦有一个模块导致服务不可用，可能会影响整个项目。

## 分布式架构

分布式架构基于传统架构演变过来，将传统的单体项目以项目模块进行拆分，拆分为会员项目、订单项目、支付项目、优惠券项目等，从而降低耦合度，这种项目架构模式慢慢开始适合于互联网公司规模人数开发。

## SOA架构

SOA架构代表面向与服务架构，俗称服务化，通俗的理解为面向与业务逻辑层开发，将共同的业务逻辑抽取出来形成一个服务，提供给其他服务接口进行调用，服务与服务之间调用使用rpc远程技术。

SOA架构特点:

1.SOA架构中通常使用XML方式实现通讯，在高并发情况下XML比较冗余会带来极大的影响，所以最后微服务架构中采用JSON替代xml方式

2.SOA架构的底层实现通过WebService和ESB（xml与中间件混合物），Web Service技术是SOA服务化的一种实现方式，WebService底层采用soap协议进行通讯，soap协议就是Http或者是Https通道传输XML数据实现的协议。

## 微服务架构

### 微服务架构产生的原因

微服务架构基于SOA架构演变过来的

在传统的WebService架构中有如下问题:

1. 依赖中心化服务发现机制
2. 使用Soap通讯协议，通常使用XML格式来序列化通讯数据，xml格式非常喜欢重，比较占宽带传输。
3. 服务化管理和治理设施不完善

### 漫谈微服务架构

## 什么是微服务

微服务架是从SOA架构演变过来，比SOA架构粒度会更加精细，让专业的人去做专业的事情（专注），目的提高效率，每个服务于服务之间互不影响，微服务架构中，每个服务必须独立部署，互不影响，微服务架构更加体现轻巧、轻量级，是适合于互联网公司敏捷开发。

## 微服务架构特征

微服务架构倡导应用程序设计程多个独立、可配置、可运行和可微服务的子服务。

服务与服务通讯协议采用Http协议，使用restful风格API形式来进行通讯，数据交换格式轻量级json格式通讯，整个传输过程中，采用二进制，所以http协议可以跨语言平台，并且可以和其他不同的语言进行相互的通讯，所以很多开放平台都采用http协议接口。

## 微服务架构如何拆分

1.微服务把每一个职责单一功能存放在独立的服务中

2.每个服务运行在单独的进程中

3.每个服务有自己独立数据库存储、实际上有自己独立的缓存、数据库、消息队列等资源。

### 微服务架构与SOA架构区别

1.微服务架构基于 SOA架构 演变过来，继承 SOA架构的优点，在微服务架构中去除 SOA 架构中的 ESB 消息总线，采用 http+json（restful）进行传输。

2.微服务架构比 SOA 架构粒度会更加精细，让专业的人去做专业的事情（专注），目的提高效率，每个服务于服务之间互不影响，微服务架构中，每个服务必须独立部署，微服务架构更加轻巧，轻量级。

3.SOA 架构中可能数据库存储会发生共享，微服务强调独每个服务都是单独数据库，保证每个服务于服务之间互不影响。

4.项目体现特征微服务架构比 SOA 架构更加适合与互联网公司敏捷开发、快速迭代版本，因为粒度非常精细。

# SpringCloud微服务框架

## 为什么选择SpringCloud

因为SpringCloud出现,对微服务技术提供了非常大的帮助，因为SpringCloud 提供了一套完整的微服务解决方案，不像其他框架只是解决了微服务中某个问题。

服务治理: 阿里巴巴开源的Dubbo和当当网在其基础上扩展的Dubbox、Eureka、Apache 的Consul等

分布式配置中心: 百度的disconf、Netfix的Archaius、360的QConf、SpringCloud、携程的阿波罗等。

分布式任务:xxl-job、elastic-job、springcloud的task等。

服务跟踪:京东的hyra、springcloud的sleuth等

## SpringCloud简介

SpringCloud是基于**SpringBoot基础之上开发的微服务框架**，SpringCloud是一套目前非常完整的微服务解决方案框架，其内容包含服务治理、注册中心、配置管理、断路器、智能路由、微代理、控制总线、全局锁、分布式会话等。

SpringCloud包含众多的子项目

SpringCloud config 分布式配置中心

SpringCloud netflix 核心组件

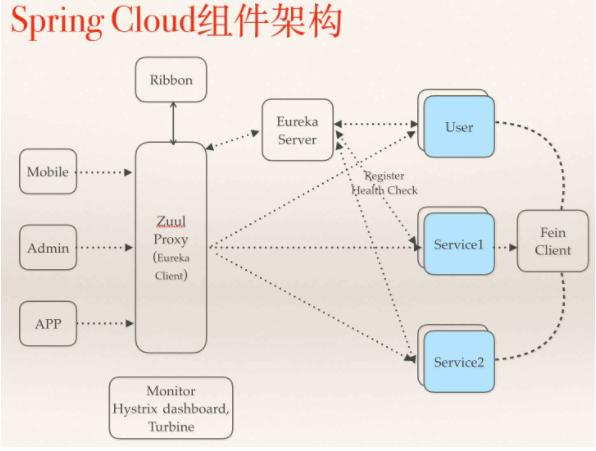
Eureka:服务治理 注册中心

Hystrix:服务保护框架

Ribbon:客户端负载均衡器

Feign：基于ribbon和hystrix的声明式服务调用组件

Zuul: 网关组件,提供智能路由、访问过滤等功能。



# 服务治理SpringCloud Eureka

## 什么是服务治理

在传统rpc远程调用中，服务与服务依赖关系，管理比较复杂，所以需要使用服务治理，管理服务与服务之间依赖关系，可以实现服务调用、负载均衡、容错等，实现服务发现与注册。

## 服务注册与发现

在服务注册与发现中，有一个注册中心，当服务器启动的时候，会把当前自己服务器的信息 比如 服务地址通讯地址等以别名方式注册到注册中心上。

另一方（消费者|服务提供者），以该别名的方式去注册中心上获取到实际的服务通讯地址，让后在实现本地rpc调用远程。

## 搭建注册中心

### 常用注册中心框架

### 注册中心环境搭建

#### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <!-- 管理依赖 -->  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Finchley.M7</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <dependencies>  <!--SpringCloud eureka-server -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  </dependency>  </dependencies>  <!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/libs-milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories> |

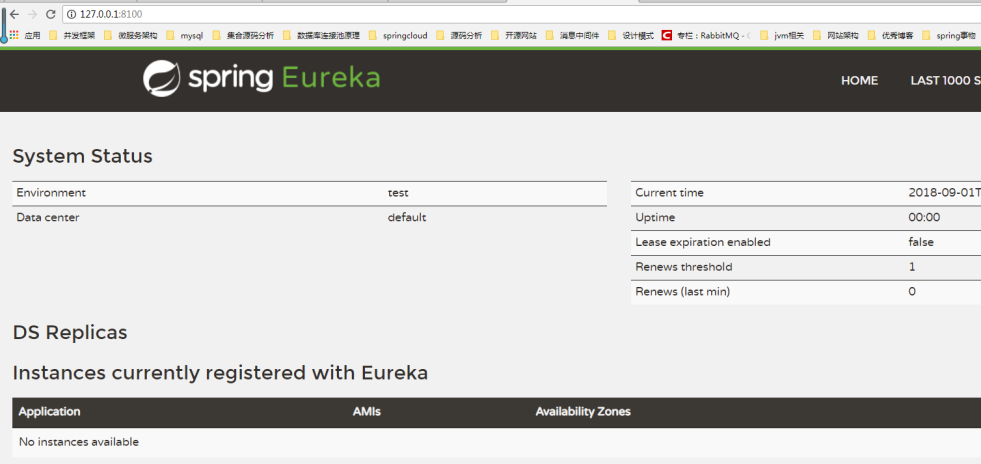
#### application.yml

|  |
| --- |
| ###服务端口号  server:  port: 8100  ###eureka 基本信息配置  eureka:  instance:  ###注册到eurekaip地址  hostname: 127.0.0.1  client:  serviceUrl:  defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/  ###因为自己是为注册中心，不需要自己注册自己  register-with-eureka: **false**  ###因为自己是为注册中心，不需要检索服务  fetch-registry: **false** |

#### 启动Eureka服务

|  |
| --- |
| @EnableEurekaServer  @SpringBootApplication  **public** **class** AppEureka {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppEureka.**class**, args);  }  } |

@EnableEurekaServer作用:开启eurekaServer



### 注册服务提供者

项目案例：实现会员服务（提供者）springcloud-2.0-member调用订单服务(消费者)springcloud-2.0-order

#### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <!-- 管理依赖 -->  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Finchley.M7</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <dependencies>  <!-- SpringBoot整合Web组件 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- SpringBoot整合eureka客户端 -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  </dependencies>  <!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/libs-milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories> |

#### application.yml

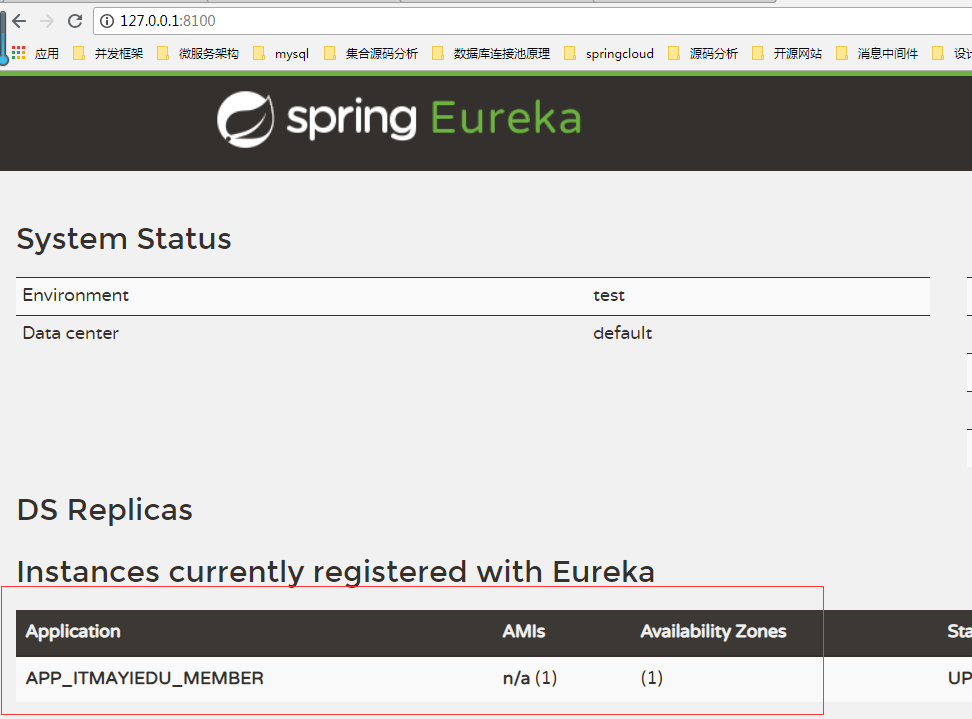
|  |
| --- |
| ###服务启动端口号  server:  port: 8000  ###服务名称(服务注册到eureka名称)  spring:  application:  name: app-itmayiedu-member  ###服务注册到eureka地址  eureka:  client:  service-url:  defaultZone: http://localhost:8100/eureka    ###因为该应用为注册中心，不会注册自己  register-with-eureka: **true**  ###是否需要从eureka上获取注册信息  fetch-registry: **true** |

#### 服务接口

|  |
| --- |
| @RestController  **public** **class** MemberController {  @RequestMapping("/getMember")  **public** String getMember() {  **return** "this is getMember";  }  } |

#### 启动会员服务

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  **public** **class** AppMember {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppMember.**class**, args);  }  } |



可以看到会员服务成功注册到eureka服务上

### 服务消费者

#### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <!-- 管理依赖 -->  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Finchley.M7</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <dependencies>  <!-- SpringBoot整合Web组件 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- SpringBoot整合eureka客户端 -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  </dependencies>  <!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/libs-milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories> |

#### Application.yml配置

|  |
| --- |
| ###服务启动端口号  server:  port: 8001  ###服务名称(服务注册到eureka名称)  spring:  application:  name: app-itmayiedu-order  ###服务注册到eureka地址  eureka:  client:  service-url:  defaultZone: http://localhost:8100/eureka    ###因为该应用为注册中心，不会注册自己  register-with-eureka: **true**  ###是否需要从eureka上获取注册信息  fetch-registry: **true** |

#### 使用rest方式调用服务

|  |
| --- |
| @RestController  **public** **class** OrderController {  @Autowired  **private** RestTemplate restTemplate;  @RequestMapping("/getorder")  **public** String getOrder() {  // order 使用rpc 远程调用技术 调用 会员服务  String memberUrl = "http://app-itmayiedu-member/getMember";  String result = restTemplate.getForObject(memberUrl, String.**class**);  System.***out***.println("会员服务调用订单服务,result:" + result);  **return** result;  }  } |

#### 启动消费者服务

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  **public** **class** AppOrder {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppOrder.**class**, args);  }  @Bean  @LoadBalanced  RestTemplate restTemplate() {  **return** **new** RestTemplate();  }  } |

@LoadBalanced就能让这个RestTemplate在请求时拥有客户端负载均衡的能力

## 高可用注册中心

在微服务中，注册中心非常核心，可以实现服务治理，如果一旦注册出现故障的时候，可能会导致整个微服务无法访问，在这时候就需要对注册中心实现高可用集群模式。

### Eureka高可用原理

默认情况下Eureka是让服务注册中心，不注册自己

|  |
| --- |
| ###因为该应用为注册中心，不会注册自己  register-with-eureka: **true**  ###不需要去注册中心上检索服务  fetch-registry: **true** |

Eureka高可用实际上将自己作为服务向其他服务注册中心注册自己，这样就可以形成一组相互注册的服务注册中心，从而实现服务清单的互相同步，达到高可用效果。

### Eureka集群环境搭建

#### Eureka01配置

|  |
| --- |
| ###服务端口号  server:  port: 8100  ###eureka 基本信息配置  spring:  application:  name: eureka-server  eureka:  instance:  ###注册到eurekaip地址  hostname: 127.0.0.1  client:  serviceUrl:  defaultZone: http://127.0.0.1:8200/eureka/  ###因为自己是为注册中心，不需要自己注册自己  register-with-eureka: **true**  ###因为自己是为注册中心，不需要检索服务  fetch-registry: **true** |

#### Eureka02配置

|  |
| --- |
| ###服务端口号  server:  port: 8200  ###eureka 基本信息配置  spring:  application:  name: eureka-server  eureka:  instance:  ###注册到eurekaip地址  hostname: 127.0.0.1  client:  serviceUrl:  defaultZone: http://127.0.0.1:8100/eureka/  ###因为自己是为注册中心，不需要自己注册自己  register-with-eureka: **true**  ###因为自己是为注册中心，不需要检索服务  fetch-registry: **true** |

#### 客户端集成Eureka集群

|  |
| --- |
| server:  port: 8000  spring:  application:  name: app-itmayiedu-member  #eureka:  # client:  # service-url:  # defaultZone: http://localhost:8100/eureka  ###集群地址  eureka:  client:  service-url:  defaultZone: http://localhost:8100/eureka,http://localhost:8200/eureka  register-with-eureka: **true**  fetch-registry: **true** |

#### Maven配置

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <!-- 管理依赖 -->  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Finchley.M7</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <dependencies>  <!--SpringCloud eureka-server -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  </dependency>  </dependencies>  <!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/libs-milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories> |

### 使用Consul来替换Eureka

#### Consul简介

Consul 是一套开源的分布式服务发现和配置管理系统，由 HashiCorp 公司用 Go 语言开发。

它具有很多优点。包括： 基于 raft 协议，比较简洁； 支持健康检查, 同时支持 HTTP 和 DNS 协议 支持跨数据中心的 WAN 集群 提供图形界面 跨平台，支持 Linux、Mac、Windows

Consul 下载地址https://springcloud.cc/spring-cloud-consul.html

#### Consul环境搭建

[官方下载地址](https://www.consul.io/downloads.html" \t "https://blog.csdn.net/achenyuan/article/details/_blank)下载window版，解压得到一个可执行文件。   
设置环境变量，让我们直接在cmd里可直接使用consul使命。在path后面添加consul所在目录例如D:\soft\consul\_1.1.0\_windows\_amd64

启动consul命

consul agent -dev -ui -node=cy

-dev开发服务器模式启动，-node结点名为cy，-ui可以用界面访问，默认能访问。

测试访问地址:http://localhost:8500

#### Consul客户端

##### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <!-- 管理依赖 -->  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Finchley.M7</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <dependencies>  <!-- SpringBoot整合Web组件 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!--SpringCloud consul-server -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-consul-discovery</artifactId>  </dependency>  </dependencies>  <!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/libs-milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories> |

##### 客户端配置文件

|  |
| --- |
| ###eureka 服务端口号  server:  port: 8502  spring:  application:  name: consul-order  ####consul注册中心地址  cloud:  consul:  host: localhost  port: 8500  discovery:  hostname: 192.168.18.220 |

### 使用Zookeeper来替换Eureka

### Eureka源码分析

# 客户端负载均衡器

## 客户端负载均衡器

## RestTemplate

## 负载均衡器源码分析

## 负载均衡器重试机制

# 服务保护机制SpringCloud Hystrix

## 服务降级、熔断、限流概念

## Hystrix工作原理

## Hystrix属性配置

## Hystrix仪表盘

## Turbine集群监控

# 声明式服务调用SpringCloud Fegin

## Fegin操作用法

## Fegin参数绑定

## Fegin继承特性

## Ribbon配置

## Hystrix配置

# Api网关服务SrpingCloud Zuul

## 快速入门

## 路由详解

## 过滤器详解

## 动态加载

# 分布式配置中心SrpingCloud config

## 快速入门

## Config架构

## 服务端详解

## 客户端详解

# 分布式服务跟踪SpringCloud sleuth

快速入门

跟踪原理

抽样收集

与Logstash整合

与Zipkin整合