# springCloud高级

## 知识补充

### 服务器中部署多少台tomcat

1. CPU核算(8核) 首先看核数
2. 内存(16-128G) 第二看内存 匹配即可

## Eureka集群搭建

### 搭建规划

一般搭建集群都是奇数台.

A:IP:7000

B:IP:7001

C:IP:7002

### 编辑yml文件

server:

port: 8020

servlet:

context-path: /

#添加服务的名称

spring:

application: #定义服务名称

name: consumer-user

eureka:

client:

service-url:

defaultZone: http://localhost:7000/eureka,http://localhost:7001/eureka,http://localhost:7002/eureka

instance:

instance-id: consumer-user-8020 #定义微服务的名称

prefer-ip-address: **true** #是否显示IP和端口

feign:

hystrix:

enabled: **true** #启动熔断器机制 !!!!!!!!!!!!!

hystrix:

command:

default:

execution:

isolation:

thread:

timeoutInMilliseconds: 3000 #设定断路器超时时间

### 关于Eureka集群说明

说明:Eureka每个节点直接都能进行相互通信.并且每个节点都是主机.

主要有一个节点存活,就可以为整合微服务提供支持.相对于其他集群而言,性能更优.

## Zuul

### Zuul的介绍

Zuul 是用户访问微服务统一的入口

主要作用是对请求进行路由和过滤2个作用.

路由功能:

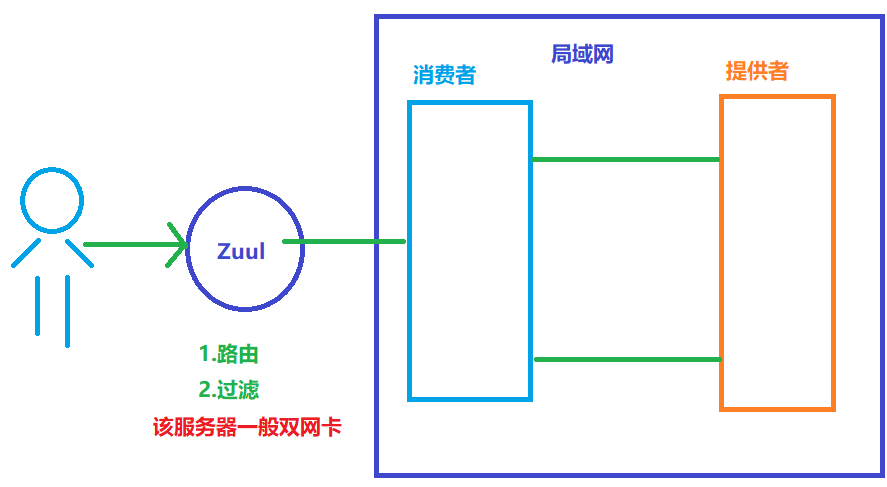
将外部的请求转发到具体的微服务实例上.

过滤功能:

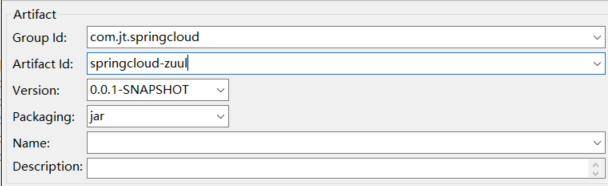
对请求进行校验/拦截

Zuul与eureka进行整合,同时能够获取全部的服务信息.后自动跳转.

### Zuul的调用原理



### 创建项目



### 导入jar包

<!--导入springBoot依赖包 -->

<parent>

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

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

<version>2.0.5.RELEASE</version>

<relativePath />

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<!--依赖管理，用于管理spring-cloud的依赖 -->

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Finchley.SR2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

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

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

<dependencies>

<!--导入springCloudjar包 -->

<dependency>

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

<artifactId>spring-cloud-stream</artifactId>

</dependency>

<!--引入springBoot jar包 -->

<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>

<!--支持热部署 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>springloaded</artifactId>

<version>1.2.8.RELEASE</version>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<!--引入springBoot监控 -->

<dependency>

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

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

</dependency>

<!--添加eureka 客户端地址 -->

<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-zuul</artifactId>

</dependency>

</dependencies>

### 编辑yml文件

server:

port: 9050

servlet:

context-path: /

spring:

application: #定义服务名称

name: springcloud-zuul

eureka:

client:

service-url:

defaultZone: http://localhost:7000/eureka,http://localhost:7001/eureka,http://localhost:7002/eureka

instance:

instance-id: springcloud-zuul-9050 #定义微服务的名称

prefer-ip-address: **true** #是否显示IP和端口

### 编辑主启动类

@SpringBootApplication(exclude=DataSourceAutoConfiguration.**class**)

@EnableZuulProxy //开启API网关

**public** **class** SpringCloud\_Zuul {

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

SpringApplication.*run*(SpringCloud\_Zuul.**class**, args);

}

}

### 路由配置

zuul:

#prefix: / #通过统一的公共前缀访问

#ignored-services: springcloud-user #禁止通过某个服务名访问

ignored-services: "\*" #禁止通过全部服务名访问

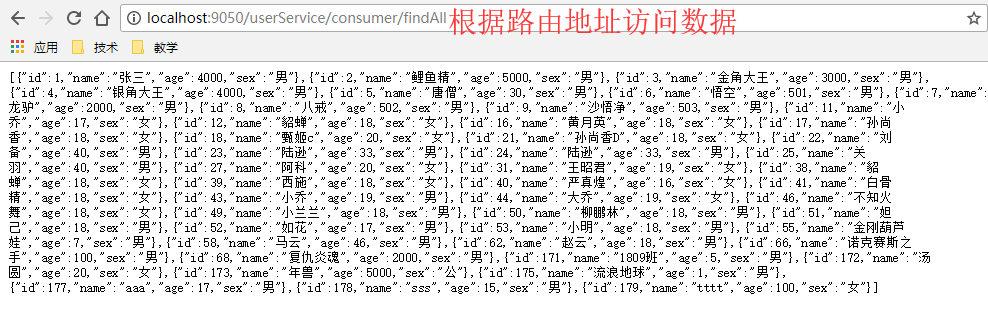
routes:

user-service:

serviceId: consumer-user #需要服务端映射路径名称

path: /userService/\*\* #编辑浏览器输入地址

### 访问测试

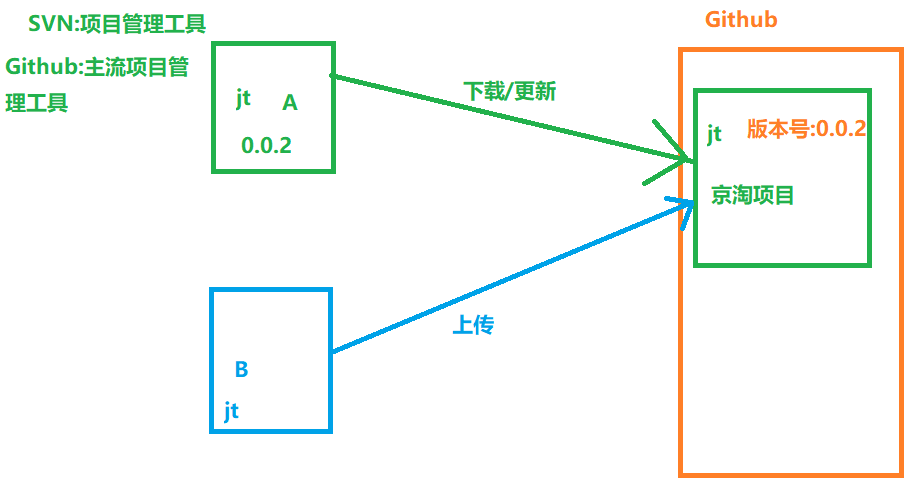


## 配置中心

### 配置中心说明

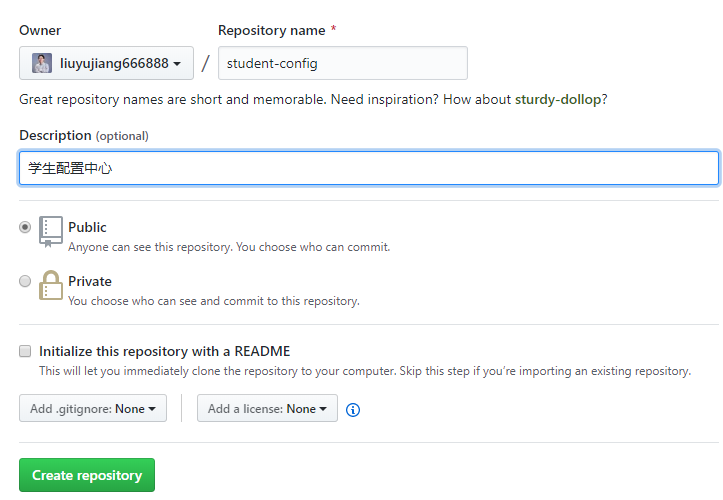
如果大型项目中,使用的配置文件非常多,如果分散管理,不利于开发.所以采用集中式的管理.

### GitHub使用原理

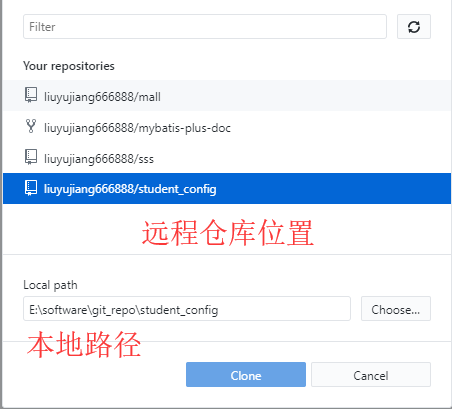


1. 通过远程仓库控制版本.
2. 用户上传代码后会更新版本信息
3. 当用户更新时,会从远程仓库更新数据.

### 新建远程仓库

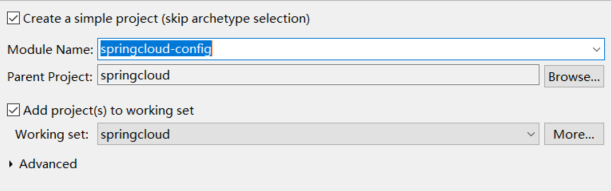


### 工具连接仓库



## 配置中心

### 创建配置中心的服务端



### 添加jar包

<dependencies>

<!--springcloud配置中心 -->

<dependency>

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

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

<!--eclipse中整合github中防止冲突 -->

<dependency>

<groupId>org.eclipse.jgit</groupId>

<artifactId>org.eclipse.jgit</artifactId>

</dependency>

</dependencies>

### 编辑yml配置文件

说明:通过配置中心可以连接远程仓库.

server:

port: 9080

spring:

application:

name: springcloud-config #定义服务的名称

cloud:

config:

server:

git:

uri: https://github.com/liuyujiang666888/student\_config.git

#连接远程github账户

### 编辑主启动类

@SpringBootApplication(exclude= {DataSourceAutoConfiguration.**class**})//排除数据源启动,否则会报错

@EnableConfigServer //开启配置中心设置

**public** **class** SpringCloud\_Config {

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

SpringApplication.*run*(SpringCloud\_Config.**class**, args);

}

}

## 定义配置中心的客户端

### 编辑客户端yml文件

--- #分割线 8000 定义为prod 8005端口定义dev

server:

port: 8000

servlet:

context-path: /

spring:

profiles: prod #定义为生产环境

datasource:

#引入druid数据源

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

url: jdbc:mysql://127.0.0.1:3306/jtdb?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

username: root

password: root

application: #定义服务名称

name: provider-user

eureka:

client:

service-url:

defaultZone: http://localhost:7000/eureka,http://localhost:7001/eureka,http://localhost:7002/eureka

instance:

instance-id: provider-user-8000 #定义微服务的名称

prefer-ip-address: **true** #是否显示IP和端口

#mybatis-plush配置

mybatis-plus:

type-aliases-package: com.jt.springcloud.pojo

mapper-locations: classpath:/mybatis/mappers/\*.xml

configuration:

map-underscore-to-camel-case: **true** #自动开启驼峰规则映射

--- #8005端口定义dev

server:

port: 8005

servlet:

context-path: /

spring:

profiles: dev #定义为研发环境

datasource:

#引入druid数据源

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

url: jdbc:mysql://127.0.0.1:3306/jtdb?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

username: root

password: root

application: #定义服务名称

name: provider-user

eureka:

client:

service-url:

defaultZone: http://localhost:7000/eureka,http://localhost:7001/eureka,http://localhost:7002/eureka

instance:

instance-id: provider-user-8005 #定义微服务的名称

prefer-ip-address: **true** #是否显示IP和端口

#mybatis-plush配置

mybatis-plus:

type-aliases-package: com.jt.springcloud.pojo

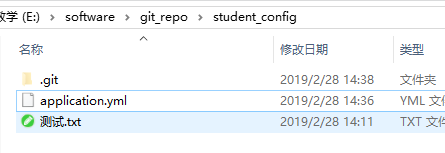
mapper-locations: classpath:/mybatis/mappers/\*.xml

configuration:

map-underscore-to-camel-case: **true** #自动开启驼峰规则映射

### 上传yml文件

说明:将修改好的配置文件通过工具,上传到远程仓库中.方便之后的用户联网访问.



### 添加客户端jar包

<!--springcloud客户端配置 -->

<dependency>

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

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

</dependency>

<!--eclipse中整合github中防止冲突 -->

<dependency>

<groupId>org.eclipse.jgit</groupId>

<artifactId>org.eclipse.jgit</artifactId>

</dependency>

### 编辑bootstrap配置

spring:

application:

name: provider-user #一般见名知意和配置文件一样

cloud:

config:

name: application #加载远程配置中心文件名称

profile: dev #动态加载生产环境配置文件

label: master #获取master默认分支

uri: http://localhost:9080/

### 测试结果

当程序启动时,选择的是dev环境时.会自动的连接github中的配置文件的开发环境.以后做环境的切换,比较方便.