# spring cloud

微服务两大阵营：

阿里系 spring cloud

通信方式：RPC（二进制） Restful（Http）

通信套件：Dubbo Spring Cloud netfilx 全家桶

注册中心：Zookeeper Netfilx Eureka

Spring Cloud Eureka

Spring cloud 对 Netfilx Eureka 做了二次封装

两个组件：

Eureka Server 注册中心

Eureka Client 注册客户端

### 部署注册中心：

使用旗舰版IntelliJ IDEA

create new project -> Spring Initializer->Next->Group:com.itlaoqi.springcloud;Artifact:registry;

Version:1.0.0.RELEASE;->Next->Spring Boot 2.0.2;Cloud Discovery;Eureka Server->Next

->保存位置->Finish

打开pom.xml 修改

<parent>

<version>2.0.1.RELEASE</version>

</parent>

<properties>

<spring-cloud.version>Finchley.RC1</spring-cloud.version>

</properties>

修改完后点击右下方悬浮窗的Maven projects need to be imported:Enable Auto-Import

打开registry->src->main->resources

将目录下的application.properties修改为application.yml然后打开

### 注意：yml中的值和：之间有个空格

写入：

spring:

application:

#应用名称，默认也是注册中心ID

name: registry

Server:

#注册中心映射端口号

port: 8761

打开自动生成的工程registry->src->main->java->com->itlaoqi->springcloud->registry目录下的RegistryApplication.java

在Application增加@EnableEurekaServer 启用Eureka服务器

@SpringBootApplication

@EnableEurekaServer //启用注册中心

public class RegistryApplication{

public static void main(String[] args){

SpringApplication.run(RegistryApplication.class,args);

}

}

点击运行按钮->在浏览器中输入localhost:8761即可打开注册中心UI

修改application.yml

增加：

eureka:

client:

#对于Eureka服务器来说，没有必要注册客户端

register-with-eureka:false

### 注册微服务

新建工程->Group:不变;Artifact:client->Next->Web->Web;Cloud Discovery->Eureka Discovery

->Next->Finish

同样在pom.xml中修改版本对应信息

在ClientApplication中写入

@SpringBootApplication

//在启动时向Eureka Server 注册微服务信息

@EnableDiscoveryClient

public class ClientApplication{

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

}

修改application.yml

spring:

application:

name:client

eureka:

client:

service-url:

defaultZone:http://localhost:9000/eureka/

在client目录下新建controller.ClientController.java

写入：

@Controller

public class ClientController{

@GetMapping("/msg")

@ResponseBody

public String msg(){

return "I am microservice";

}

}

运行

### Zuul

GateWay（网关）是微服务架构不可或缺的一部分。

GateWay为客户端提供了统一的访问入口。

Netfilx Zuul是Spring Cloud 默认使用的GateWay组件

Zuul是Netflix出品的一个路由和服务端的负载均衡器

新建工程->Group:不变;Artifact:gateway->Next->Cloud Rouing->Zuul;Cloud Discovery->Eureka Discovery

->Next->Finish

同样在pom.xml中修改版本对应信息

GateWayApplication

@SpringBootApplication

@EnableDiscoveryClient

@EnableZuulProxy//启用Zuul路由服务

public class GatewayApplication{

public static void main(String[] args ){

SpringApplication.run(GateWayApplication.class,args);

}

}

application.yml

spring:

application:

name:gateway

eureka:

client：

service-url:

defaultZone:http://localhost:8761/eureka/ #向服务器url注册自己

server:

port:9000

Zuul路由的访问规则是：

<http://xxxx:xx/service-id/....> 网关地址/服务名/服务路径

本例为：<http://localhost:9000/client/msg>

导航栏中端口下拉

Edit configurations...

Name:8001

Configuration->Main class:com.itlaoqi.springcloud.client.ClientApplication;

VM options: -Dserver.port=8001

重复上述步骤再添加8000和8002两个同样的微服务

相同的三个微服务选择不同端口都启动一下

@Controller

public class ClientController{

@Value("${server.port}")

private String port;

@GetMapping("/msg")

@ResponseBody

public String msg(){

return "I am microservice"+port;

}

可以看到两个微服务轮询

}

为了方便服务地址的访问，可以配置缩写

打开Gateway的application.yml

添加：

zuul:

` routes:

client: /c/\*\* #\*\*代表映射所有路径

访问：localhost:9000/c/msg

### Eureka高可用配置

防止注册中心瘫痪影响所有微服务运行，注册中心也采用多个

先打开到registry的application.yml中，导航栏上RegistryApplication下拉菜单选择

Edit Configuration... 弹出窗口

选中左侧导航栏中SpringBoot下的RegistryApplication，点击上方的复制小按钮复制一份，点击其中一个，修改Name:R8761;VM options:-Dserver.port=8761,点击apply。选中另一个，修改Name:R8762;VM options:-Dserver.port=8762 apply

导航栏选中R8761

application.yml

spring:

application:

#应用名称，默认也是注册中心ID

name:registry

Server:

#注册中心映射端口号

port:8761

eureka:

client:

#对于Eureka服务器来说，没有必要注册客户端

register-with-eureka:false

service-url:

defaultZone:http://localhost:8762/eureka #向8761注册自己的客户端

同理选中R8762

service-url：

defaultZone:http://localhost:8761/eureka #向8762注册自己客户端

以上实现了两个注册中心的客户端同步

修改Client的application.yml

spring:

application:

name:client

eureka:

client:

service-url:

defaultZone:http://localhost:9000/eureka/，defaultZone:http://localhost:9000/eureka/

在客户端定义多个Eureka注册中心，利用，注册多个eureka注册中心，在运行时，Eureka Client会按前后顺序自动选择

在运行客户端选择注册中心机制，我们称为客户端负载均衡机制

微服务拆分原则

可伸缩的架构设计：

x-微服务可水平扩展 每个微服务都可以独自运行，也可以多备份

y-按业务功能进行切分

z-将数据进行拆分

图书借阅项目开发

1. navicat新建数据库连接，创建会员数据库，创建会员信息表
2. 创建微服务注册中心、微服务网关
3. 创建会员微服务

新建工程->Group:com.itlaoqi.springcloud;

Artifact:member->Next->

Cloud Discovery->Eureka Discovery；web->web;SQL:JPA;SQL:MySQL;

->Next->保存代码位置->Finish

同样在pom.xml中修改版本对应信息

application.yml

spring:

application:

name:member

datasource:

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

url: jdbc:mysql://localhost:3306/sc-member?useUnicode=true&characterEncoding=

UTF-8

username:root

password:root

在com.itlaoqi.springcloud.member下创建entity.Member.java

@Entity

@Table(name="t\_member")

public class Member{

@Id

@GenerateValue(strategy=GenerationType.IDENTITY)

private Long mid;

private String name;

private String idno;

private String mobile;

private Date regdate;

private Date expdate;

//自动生成geter seter方法

}

springcloud.Member文件夹下创建repository.memberRepository

public interface MemberRepository extends JpaRepository<Member,Long>{

//自动执行select \* from t\_member where mobile=...

public List<Member>findByMobile(String mobile);

}

springcloud.Member文件夹下创建service.MemberService

@Service

public class MemberService{

@Resource

private MemberRepository memberRepository;

public Member checkByMobile(String Mobile){

List<Member> members =memberRepository.findByMobile(mobile);

if(members.size()==0){

throw new MemberNotFoundException("会员不存在");

}

return members.get(0);ch

}

}

service下创建exception.MemberNotFoundException

public class MemberNotFoundException extends RuntimeException{

public MemberNotFoundException(String msg){

super(msg);

}

}

springcloud.member文件夹下创建controller.MemberController

@Controller

public class MemberController{

@Resource

private MemberService memberService;

@GetMapping("/check")

@ResponseBody

public Map checkMobile(String mobile){

Map result =new HashMap();

try{

result.put("code","0");

result.put("message","success");

Member member=memberService.checkByMobile(mobile);

result.put("data",member)

}catch{

e.printStackTrace();

result.put("code",e.getClass().getSimpleName());

result.put("message",e.getMessage());

}

return result;

}

}

修改member项目中的application.yml

添加

server:

port: 8000

访问测试：localhost:8000/check?mobile=12345678901

查看是否有返回

修改member项目中的application.yml

添加

eureka:

clent:

service-url:

defaultZone: <http://localhost:8761:/eureka>

修改gateway下的application.yml

spring:

application:

name: gateway

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka/>

server:

port:9000

注释以下这部分

zuul:

routes:

client: /c/\*\*

启动registry注册中心，网关，以及member项目

访问测试：localhost:9000/check?mobile=12345678901

查看是否有返回，（访问的网关）

前端界面部分：

使用的jquery的WeUI

<script>

$("#btn-check").click(function(){

var mobile=$("#mobile").val();

$.ajax({

url:"http://localhost:9000/member/check",

data: {"mobile":mobile},

datatype:json,

type:"get",

success: function(json){

console.log(json);

}

})

})

</script>

在浏览器中，如果向其他域名（端口不一样也算其他域名）下的url发送请求，浏览器自身的安全策略是不允许的，我们需要设置CORS（跨域访问），来解决跨域问题。

在MemberController类上方加上

@Controller 原本就有

@CrossOrigin //在Controller上利用CrossOrigin解决跨域访问问题。

<script>

$("#btn-check").click(function(){

var mobile=$("#mobile").val();

$.ajax({

url:"http://localhost:9000/member/check",

data: {"mobile":mobile},

datatype:json,

type:"get",

success: function(json){

if(json.code==0){

$("#succMobile").text("会员:"+json.data.name);

$("#succMobile").show();

$("#errMobile").hide();

}else{

$("#succMobile").text(json.message);

$("#succMobile").hide();

$("#errMobile").show();

}

}

})

})

</script>