微服务架构之SpringCloud

一、简介

1.1 SpringCloud是什么

Springcloud是微服务架构的集大成者，将一系列优秀的组件进行了整合。基于springboot构建，

是一系列框架的有序集合，它利用 Spring Boot 的开发便利性简化了分布式系统的开发，比如服务发现、服务网关、服务路由、链路追踪等。Spring Cloud 并不重复造轮子，而是将市面上开发得比较好的模块集成进去，进行封装，从而减少了各模块的开发成本。换句话说：Spring Cloud 提供了构建分布式系统所需的“全家桶”。

1.2 优缺点

优点:

集大成者，Spring Cloud 包含了微服务架构的方方面面。 约定优于配置，基于注解，没有配置文件。 轻量级组件，Spring Cloud 整合的组件大多比较轻量级，且都是各自领域的佼佼者。 开发简便，Spring Cloud 对各个组件进行了大量的封装，从而简化了开发。 开发灵活，Spring Cloud 的组件都是解耦的，开发人员可以灵活按需选择组件

缺点： 项目结构复杂，每一个组件或者每一个服务都需要创建一个项目 部署门槛高，项目部署需要配合 Docker 等容器技术进行集群部署

1.3 SpringCloud常用组件

服务治理： Spring Cloud Eureka

客户端负载均衡： Spring Cloud Ribbon

服务容错保护： Spring Cloud Hystrix

声明式服务调用： Spring Cloud Feign

API 网关服务：Spring Cloud Zuul

二、微服务初体验

Eureka：专门用于给其他服务注册的称为 Eureka Server(服务注册中心)，其余注册到 Eureka Server 的服务称为 Eureka Client，Eureka的端口号默认为8761

2.1 搭建注册中心

实现服务治理，提供者，消费者都需要在注册中心中进行注册

实现步骤:

1、依赖

``

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

2、开关类配置

开关类上使用注解：@EnableEurekaServer

```
//标记这是一个注册中心服务器
```

3、全局配置文件设置

4、启动并测试

2.2 创建服务提供者

创建接口，运行并发布到注册中心

实现步骤：

1、依赖jar

``

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

2、编写代码

控制器层

3、开关类配置

在开关类上使用注解：@EnableEurekaClient //注册服务 标记这是客户端

4、全局配置文件设置

``

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
    port: 9901
spring:
    application:
        name: HelloProvider
```

5、启动并测试

运行开关类，查看注册中心页面，观察服务是否注册

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
HELLOPROVIDER	n/a (1)	(1)	UP (1) - windows10.microdone.cn:HelloProvider:9901

General Info

Name	Value
total-avail-memory	304mb
environment	test
num-of-cpus	4
current-memory-usage	43mb (14%)
server-uptime	00:40
registered-replicas	
unavailable-replicas	
available-replicas	

2.3 创建服务消费者

SpringCloud服务消费有2种方式，第一种：Feign 第二种：RestTemplate+Ribbon

Feign是一种声明式、模板化的HTTP客户端，可以进行网络请求

Feign 实现步骤：

1、依赖jar

``

```
<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-openfeign</artifactId>
</dependency>
```

2、编写代码

编写接口

编写控制器

3、配置开关类

使用

```
@EnableDiscoveryClient //发现服务
@EnableFeignClients //基于Feign实现服务消费
```

``

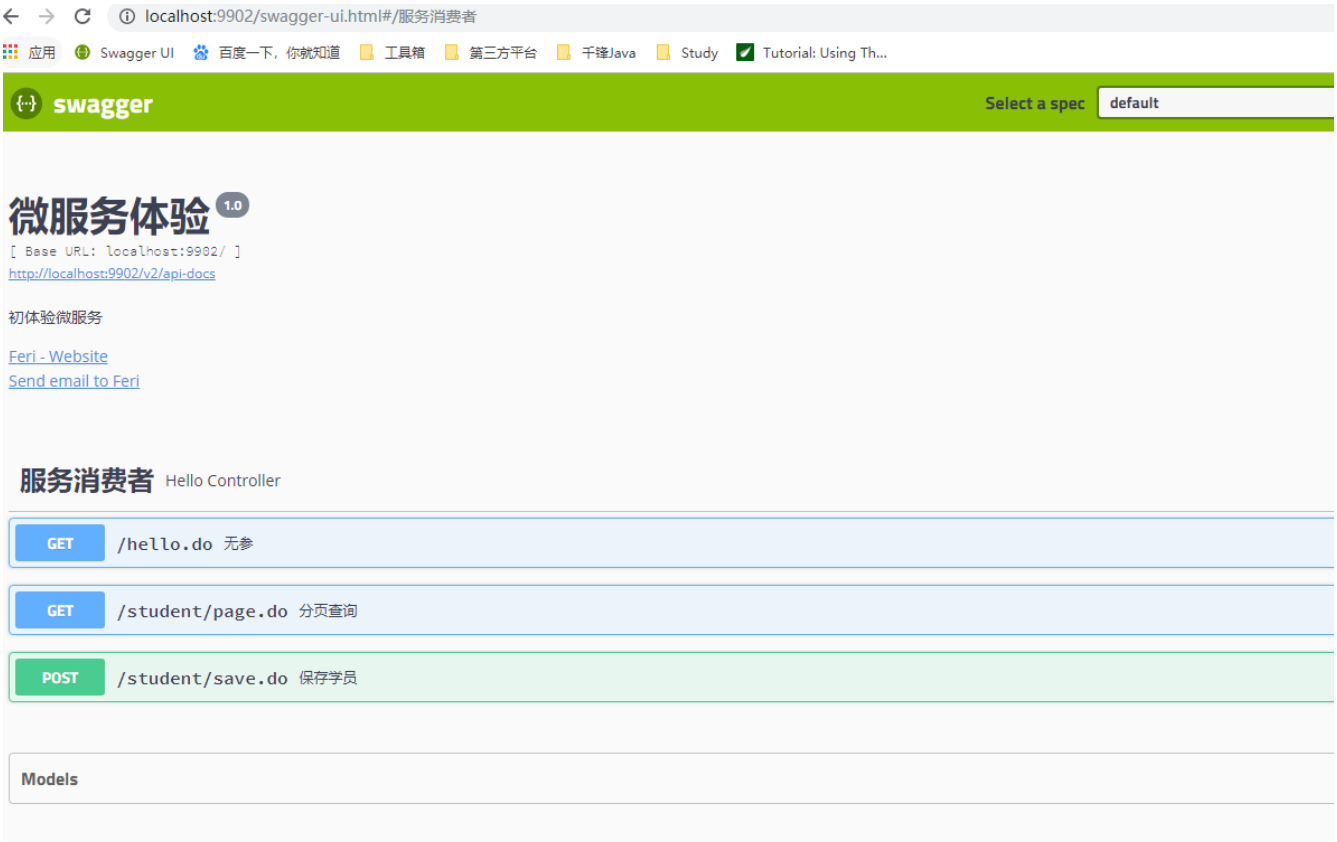
```
@SpringBootApplication
@EnableDiscoveryClient //发现服务
@EnableFeignClients //基于Feign实现服务消费
public class CloudHelloconsumerApplication {
    public static void main(String[] args) {
        SpringApplication.run(CloudHelloconsumerApplication.class, args);
    }
}
```

4、配置全局配置文件

``

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
    port: 9902
spring:
    application:
        name: HelloConsumer
```

5、运行并测试



三、微服务核心说明

3.1 Feign传递参数

常规类型：基本类型、String

需要使用注解@RequestParam

自定义类型：自定义类、集合

需要使用注解：@RequestBody

提供者：

``

```
@GetMapping("/provider/page")
public R page(@RequestParam("page") int page, @RequestParam("count") int count){
    List<Student> studentList=new ArrayList<>();
    for(int i=(page-1)*count;i<=page*count;i++){
        Student student=new Student();
        student.setId(i);
        student.setName("西亚斯："+i);
        studentList.add(student);
    }
    return R.setOK("OK",studentList);
}
@PostMapping("/provider/add")
public R save(@RequestBody Student student){
    System.out.println(student.toString());
```

```
    return R.setOK("OK",null);
}
```

消费者：

``

```
@FeignClient(name = "HelloProvider")
public interface HelloService {
    @GetMapping("provider/hello")
    R hello();
    @GetMapping("/provider/page")
    R page(@RequestParam("page") int page, @RequestParam("count") int count);
    @PostMapping("/provider/add")
    R save(@RequestBody Student student);
}
```

源码地址：https://github.com/xingpenghui/SpringCloud_1902