链滴

Spring Cloud（一）入门

作者：wlgzs-sjl

原文链接：https://ld246.com/article/1606958159019

来源网站：链滴

一、微服务概述

  微服务架构是一种架构模式，或者说是一种架构风格， 它提倡将单一的应程序划分成一组小的服务，每个服务运行在其独立的自己的进程内，服务之间互相协调，互相配置，用户提供最终价值。

  服务之间采用轻量级的通信机制互相沟通，每个服务都围绕着具体的业务进构建，并且能够被独立的部署到生产环境中，另外，应尽量避免统一的，集中式的服务管理机制，对体的一个服务而言，应根据业务上下文，选择合适的语言，工具对其进行构建，可以有一个非常轻量的集中式管理来协调这些服务，可以使用不同的语言来编写服务，也可以使用不同的数据存储。

优点：

- 开发简单，开发效率提高，一个服务可能就是专一的只干一件事；
- 微服务是松耦合的，是有功能意义的服务，无论是在开发阶段或部署阶段都是独立的；
- 微服务能使用不同的语言开发；
- 每个微服务都有自己的存储能力，可以有自己的数据库，也可以有统一数据库。

缺点：

- 开发人员要处理分布式系统的复杂性；
- 多服务运维难度，随着服务的增加，运维的压力也在增大；
- 系统部署依赖；
- 服务间通信成本；
- 数据一致性等等。

微服务技术栈：

微服务条目	落地技术
服务开发	SpringBoot,Spring,SpringMVC
服务配置与管理	Netflix公司的Archaius、阿里的Diamond等
服务注册与发现	Eureka、Consul、Zookeeper等
服务调用	Rest、RPC、gRPC
服务熔断器	Hystrix、Envoy等
负载均衡	Ribbon、Nginx等
服务接口调用（客户端调用服务的简化工具）	Feign等
消息队列	Kafka、RabbitMQ、ActiveMQ等
服务配置中心管理	SpringCloudConfig、Chef等
服务路由（API网关）	Zuul等
服务监控	Zabbix、Nagios、Metrics、Specatator等
全链路追踪	Zipkin、Brave、Dapper等
服务部署	Docker、OpenStack、Kubernetes等
数据流操作开发包	SpringCloud Stream(封装与Redis，Rabbit，Kafka等发送接收消息)
事件消息总线	SpringCloud Bus

二、Spring Cloud概述

  SpringCloud利用SpringBoot的开发便利性，巧妙地简化了分布式系统基设施的开发，SpringCloud为开发人员提供了快速构建分布式系统的一些工具，包括配置管理，服务现，断路器，路由，微代理，事件总线，全局锁，决策竞选，分布式会话等等，他们都可以用Spring oot的开发风格做到一键启动和部署。

  SpringBoot可以离开SpringClooud独立使用，开发项目，但是SpringClou离不开SpringBoot，属于依赖关系。

Dubbo 和 SpringCloud 对比

	Dubbo	Spring
服务注册中心	Zookeeper	Spring Cloud Netfilx Eureka
服务调用方式	RPC	REST API
服务监控	Dubbo-monitor	Spring Boot Admin
断路器	不完善	Spring Cloud Netflix Hystrix
服务网关	无	Spring Cloud Netflix Zuul
分布式配置	无	Spring Cloud Config
服务跟踪	无	Spring Cloud Sleuth
消息总线	无	Spring Cloud Bus
数据流	无	Spring Cloud Stream
批量任务	无	Spring Cloud Task

最大区别：

  SpringCloud抛弃了Dubbo的RPC通信，采用的是基于HTTP的REST方式。
格来说，这两种方式各有优劣。虽然从一定程度上来说，后者牺牲了服务调用的性能，但也避免了原RPC带来的问题。而且REST相比RPC更为灵活，服务提供方和调用方的依赖只依靠一纸契约，不存在码级别的强依赖，这在强调快速演化的微服务环境下，显得更加合适。

三、项目构建

总体介绍：

一个父工程带着多个子Module子模块

SpringCloud父工程（Project）下依次带着3个子模块（Module）

- springcloud-api 【封装的整体entity / 接口 / 公共配置等】
- springcloud-provider-dept-8001【服务提供者】
- springcloud-consumer-dept-80【服务消费者】

创建父工程

新建父工程Maven项目 springcloud-parent，切记Packageing是pom模式。

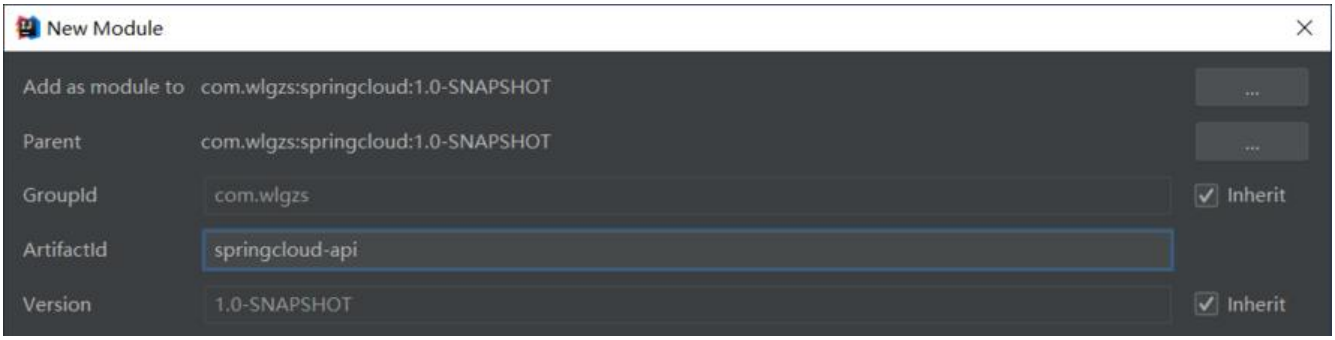
主要是定义POM文件，将后续各个子模块公用的jar包等统一提取出来，类似一个抽象父类。

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.wlgzs</groupId>
    <artifactId>springcloud</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!--打包方式 pom-->
    <packaging>pom</packaging>

    <properties>

    </properties>

    <dependencyManagement>
        <dependencies>
            <!--springCloud的依赖-->
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Greenwich.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>


            <!--SpringBoot-->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-dependencies</artifactId>
                <version>2.1.4.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>8.0.16</version>
            </dependency>
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>1.1.10</version>
            </dependency>
            <dependency>
                <groupId>org.mybatis.spring.boot</groupId>
                <artifactId>mybatis-spring-boot-starter</artifactId>
                <version>1.3.2</version>
```

```
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-core</artifactId>
            <version>1.2.3</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.12</version>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
    </dependencies>
</dependencyManagement>

</project>
```

创建api公共模块



可以观察发现，在父工程中多了一个Modules

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.wlgzs</groupId>
    <artifactId>springcloud</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>springcloud-api</module>
    </modules>

    <!--打包方式 pom-->
    <packaging>pom</packaging>
```

编写springcloud-api 的 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <parent>
        <artifactId>springcloud</artifactId>
        <groupId>com.wlgzs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>springcloud-api</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
    </dependencies>
</project>
```

编写实体类

```
package com.wlgzs.springcloud.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

/**
 * 部门表(Dept)表实体类
 */
@Data
@AllArgsConstructor
@NoArgsConstructor

public class Dept implements Serializable {

    /**
     * 部门ID
     */
    private Integer deptId;

    /**
     * 部门名称
     */
    private String deptName;
```

```
    /**
     * 数据库
     */
    private String dbSource;

    public Dept(String deptName){
        this.deptName=deptName;
    }
}
```

创建provider模块

新建springcloud-provider-dept-8001模块

编辑pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <parent>
        <artifactId>springcloud</artifactId>
        <groupId>com.wlgzs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>springcolud-provider-dept-8001</artifactId>

    <dependencies>
        <dependency>
            <groupId>com.wlgzs</groupId>
            <artifactId>springcloud-api</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
        </dependency>
        <dependency>
```

```
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-core</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-test</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
</project>
```

编辑 application.yml

```
server:
  port: 8001
mybatis:
  type-aliases-package: com.wlgzs.springcloud.entity
  configuration:
    map-underscore-to-camel-case: true
spring:
  application:
    name: springcloud-provider-dept
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db01?useUnicode=true&characterEncoding=utf-8&serverT
mezone=UTC
    username: root
    password: root
```

编写部门的dao接口

```
package com.wlgzs.springcloud.dao;

import com.wlgzs.springcloud.entity.Dept;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;


import java.util.List;

@Mapper
public interface DeptDao {
```

```
    @Insert("insert into dept (dept_name,db_source)\n" +
            "values (#{deptName},DATABASE())")
    boolean addDept(Dept dept);

    @Select("select * from dept where dept_id = #{deptId}")
    Dept queryById(int deptId);

    @Select("select * from dept")
    List<Dept> queryAll();
}
```

创建Service服务层接口

```
package com.wlgzs.springcloud.service;

import com.wlgzs.springcloud.entity.Dept;

import java.util.List;

public interface DeptService {
    boolean addDept(Dept dept);

    Dept queryById(int id);

    List<Dept> queryAll();
}
```

ServiceImpl实现类

```
package com.wlgzs.springcloud.service;

import com.wlgzs.springcloud.dao.DeptDao;
import com.wlgzs.springcloud.entity.Dept;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.Resource;
import java.util.List;


@Service
public class DeptServiceImpl implements DeptService{
    @Resource
    private DeptDao deptDao;

    public boolean addDept(Dept dept) {
        return deptDao.addDept(dept);
    }

    public Dept queryById(int id) {
```

```
        Dept dept=deptDao.queryById(id);
        return dept;
    }

    public List<Dept> queryAll() {
        List<Dept> depts=deptDao.queryAll();
        return depts;
    }
}
```

DeptController提供REST服务

```
package com.wlgzs.springcloud.controller;

import com.wlgzs.springcloud.entity.Dept;
import com.wlgzs.springcloud.service.DeptService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class DeptController {
    @Autowired
    DeptService deptService;

    @PostMapping("/")
    public boolean addDept(@RequestBody Dept dept){
        return deptService.addDept(dept);
    }

    @GetMapping("/{id}")
    public Dept getDeptById(@PathVariable int id){
        return deptService.queryById(id);
    }

    @GetMapping("/")
    public List getAll(){
        return deptService.queryAll();
    }
}
```

编写DeptProvider的主启动类

```
package com.wlgzs.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
```

```
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
public class DeptProvider_8001 {
    public static void main(String[] args) {
        SpringApplication.run(DeptProvider_8001.class,args);
    }
}
```

启动测试

创建consumer模块

新建springcloud-consumer-dept-80模块

编辑pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <parent>
        <artifactId>springcloud</artifactId>
        <groupId>com.wlgzs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>springcloud-consumer-dept-80</artifactId>

    <dependencies>
        <dependency>
            <groupId>com.wlgzs</groupId>
            <artifactId>springcloud-api</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
        </dependency>
    </dependencies>
</project>
```

application.yml 配置文件

```
server:
  port: 80
```

新建一个ConfifigBean包注入 RestTemplate

```
package com.wlgzs.springcloud.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class ConfigBean {

    @Bean
    public RestTemplate getRestTemplate(){
        return new RestTemplate();
    }

}
```

创建Controller包，编写DeptConsumerController类

```
package com.wlgzs.springcloud.controller;

import com.wlgzs.springcloud.entity.Dept;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

import java.util.List;

/**
 * @author wlgzs-sjl
 * @date 2020/11/21 22:17
 */
@RestController
public class DeptConsumerController {
    //理解：消费者，不应该有service层
    // 使用RestTemplate访问restful接口非常的简单粗暴且无脑
    //  （url，requestMap，ResponseBean.class） 这三个参数分别代表
    // REST请求地址，请求参数，Http响应转换 被 转换成的对象类型

    @Autowired
    private RestTemplate restTemplate;

    private static final String REST_URL_PREFIX = "http://localhost:8001";

    @GetMapping("/{id}")
```

```
    public Dept get(@PathVariable("id") int id){
        return restTemplate.getForObject(REST_URL_PREFIX+"/"+id,Dept.class);
    }

    @GetMapping("/")
    public List getAll(){
        return restTemplate.getForObject(REST_URL_PREFIX,List.class);
    }

    @PostMapping("/")
    public boolean add(Dept dept){
        return restTemplate.postForObject(REST_URL_PREFIX,dept,Boolean.class);
    }


}
```

了解RestTemplate

  RestTemplate提供了多种便捷访问远程Http服务的方法，是一种简单便捷访问restful服务模板类，是Spring提供的用于访问Rest服务的客户端模板工具集

  使用RestTemplate访问restful接口非常的简单粗暴且无脑（url，requsetMap，ResponseBean.class） 这三个参数分别代表REST请求地址，请求参数，Http响应转换 被 转换的对象类型。

主启动类

```
package com.wlgzs.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DeptConsumer_80 {
    public static void main(String[] args) {
        SpringApplication.run(DeptConsumer_80.class,args);
    }
}
```

参考B站教程狂神说Javahttps://www.bilibili.com/video/BV1jJ411S7xr