Nacos 服务注册与发现

Nacos Discovery 可以帮助您将服务自动注册到 Nacos 服务端并且能够 动态感知和刷新某个服务实例的服务列表。

服务注册到Nacos

创建nacos-discovery-provider项目

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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>me.guopop</groupId>
    <artifactId>nacos-discovery-provider</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>nacos-discovery-provider</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>

        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-
discovery</artifactId>
            <version>2.2.5.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
```

```
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

application.yml

```
server:
  port: 8001

spring:
  application:
    name: nacos-discovery-provider
  cloud:
    nacos:
      discovery:
        server-addr: 39.105.47.81:8848
        username: nacos
        password: nacos
```

启动类

```
package me.guopop.nacosdiscoveryprovider;

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

@EnableDiscoveryClient
@SpringBootApplication
public class NacosDiscoveryProviderApplication {

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

启动项目

```
Initializing ExecutorService 'Nacos-Watch-Task-Scheduler'
Tomcat started on port(s): 8001 (http) with context path ''
nacos registry, DEFAULT_GROUP nacos-discovery-provider 192.168.1.17:8001 register finished
Started NacosDiscoveryProviderApplication in 6.248 seconds (JVM running for 6.732)
```

服务注册成功！！！

Nacos Discovery 整合@LoadBalanced RestTemplate

创建nacos-discovery-consumer

pom.xml 同上

application.yml

```yaml
server:
  port: 8002

spring:
  application:
    name: nacos-discovery-consumer
  cloud:
    nacos:
      discovery:
        server-addr: 39.105.47.81:8848
        username: nacos
        password: nacos
```

启动类

```java
package me.guopop.nacosdiscoveryconsumer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@EnableDiscoveryClient
@SpringBootApplication
public class NacosDiscoveryConsumerApplication {

    @LoadBalanced
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
```

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

}
```

RestTemplateController.java

```
package me.guopop.nacosdiscoveryconsumer;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

/**
 * @author guopop
 * @date 2021/3/11 10:52
 */
@RestController
public class RestTemplateController {

    @LoadBalanced
    @Autowired
    public RestTemplate restTemplate;

    @GetMapping("/hello/{message}")
    public String hello(@PathVariable String message) {
        return restTemplate.getForObject("http://nacos-discovery-
provider/hello/" + message, String.class);
    }
}
```

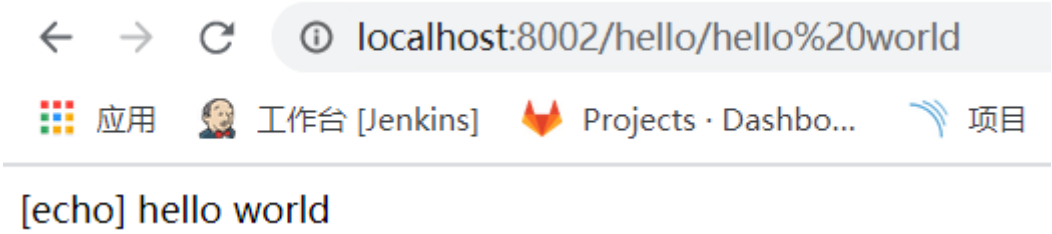
nacos-discovery-provider项目新增HelloController.java

```
package me.guopop.nacosdiscoveryprovider;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author guopop
 * @date 2021/3/11 10:56
 */
@RestController
public class HelloController {

    @GetMapping("/hello/{message}")
    public String hello(@PathVariable String message) {
        return "[echo] " + message;
    }
}
```

启动两个项目

服务注册成功！！！

访问http://localhost:8002/hello/hello%20world



服务间调用成功！！！

Nacos Discovery 整合 Spring Cloud OpenFeign

nacos-discovery-consumer项目pom.xml

```
<!--新增依赖-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
    <version>2.2.5.RELEASE</version>
</dependency>
```

启动类添加@EnableFeignClients

```
package me.guopop.nacosdiscoveryconsumer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@EnableFeignClients
@EnableDiscoveryClient
@SpringBootApplication
public class NacosDiscoveryConsumerApplication {

    @LoadBalanced
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
```

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

}
```

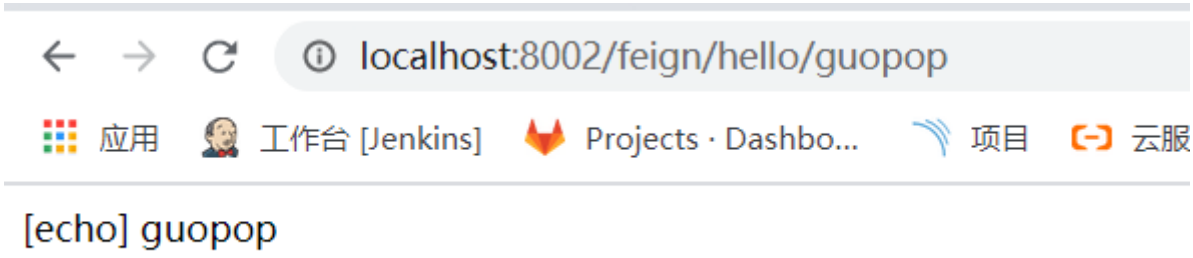
OpenFeignController.java

```
package me.guopop.nacosdiscoveryconsumer;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author guopop
 * @date 2021/3/11 11:09
 */
@RestController
public class OpenFeignController {

    @Autowired
    private HelloService helloService;

    @GetMapping("/feign/hello/{message}")
    public String hello(@PathVariable String message) {
        return helloService.hello(message);
    }
}
```

HelloService.java

```
package me.guopop.nacosdiscoveryconsumer;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

/**
 * @author guopop
 * @date 2021/3/11 11:10
 */
@FeignClient("nacos-discovery-provider")
public interface HelloService {

    @GetMapping("/hello/{message}")
    String hello(@PathVariable("message") String message);
}
```

启动项目



服务间openFeign调用成功！！！

Nacos Discovery 详细配置

配置项	Key	默认值	说明
服务端地址	spring.cloud.nacos.discovery.server-addr		Nacos Server 启动监听的 ip 地 址和端口
服务名	spring.cloud.nacos.discovery.service	${spring.application.name}	注册的服务名
权重	spring.cloud.nacos.discovery.weight	1	取值范围 1 到 100，数值越大，权重越大
网卡名	spring.cloud.nacos.discovery.network-interface		当 IP 未配置时，注册的 IP 为此网 卡所对应的 IP地址，如果此项也未 配置，则默认取第一块网卡的地址
注册的 IP 地址	spring.cloud.nacos.discovery.ip		优先级最高
注册的端口	spring.cloud.nacos.discovery.port	-1	默认情况下不用配置，会自动探测
命名空间	spring.cloud.nacos.discovery.namespace		常用场景之一是不同环境的注册的区分隔离，例如开发测试环境和生产环境的资源（如配置、服务）隔 离等
AccessKey	spring.cloud.nacos.discovery.access-key		当要上阿里云时，阿里云上面的一个云账号名
SecretKey	spring.cloud.nacos.discovery.secret-key		当要上阿里云时，阿里云上面的一个云账号密码

配置项	Key	默认值	说明
Metadata	spring.cloud.nacos.discovery.metadata		使用Map格式配置，用户可以根据自己的需要自定义一些和服务相关 的元数据信息
日志文件名	spring.cloud.nacos.discovery.log-name		
集群	spring.cloud.nacos.discovery.cluster-name	DEFAULT	Nacos 集群名称
接入点	spring.cloud.nacos.discovery.endpoint		地域的某个服务的入口域名，通过此域名可以动态地拿到服务端地址
是否集成 Ribbon	ribbon.nacos.enabled	true	一般都设置成 true 即可
是否开启 Nacos Watch	spring.cloud.nacos.discovery.watch.enabled		

Nacos Discovery Actuator Endpoint

Todo