Redis:

你在那里使用的Redis，是如何使用的。希望你把我们用Redis的代码的流程必须记清楚。

1：记录短信验证码

2：替换Mybatis的二级缓存解决脏读，穿透，击穿，雪崩等问题。

3：用Redis来做秒杀(事务，watch,分布式锁)

==============================================================

Spring Cloud微服务:

集群：两个厨师一起向外界提供相同的服务。

分布式：把我们过去的单体架构，单一项目拆解成多个架构，合并向外界提供服务。

白案厨师 订单系统

红案厨师 商品系统

拉面厨师 库存系统

分布式：

Dubbo+ZooKeeper (RPC)

Spring Cloud+ Eureka (Rest)

WebService(天气预报 物流信息 短信接口)



在spring cloud微服务架构中，总是存在数据的生产者和数据的消费者。消费者总是要从生产者中获取数据，或者进行数据的交互。

一：完成一个生产者和一个消费者的数据交互。

中文官网地址：<https://www.springcloud.cc/>

远程调用方式：

1： HttpClient 不管你是哪种调用方式，底层都是HttpClient。

是最灵活的，也是功能最强大的。

2： RestTemplate 封装了HttpClinet,基于Ribbon。

3： Feign 伪客户端方式远程调用。

开发：

1：需要一个服务的提供者，代码跟你过去的代码一模一样。

2：需要一个服务的消费者

1：导入web, cloud包

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

2：在启动类中加入

@Bean

public RestTemplate restTemplate(){

return new RestTemplate();

}

3：yml

server:

port: 9001

#用户自定义的变量 自定义的变量在代码中可以通过 ${user.userServicURL} 自动获取属性值

user:

userSericeURL: <http://localhost:9000/>

4:控制器

@RestController

public class CustomerController {

@Resource

private RestTemplate restTemplate;

@Value("${user.userSericeURL}")

private String userSericeURL;

@GetMapping("/customer/findAll")

public List findAll(){

return restTemplate.getForObject(userSericeURL+"findAll", List.class);

}

}

RestTemplate的几种调用方式：

总共12种 getObject getForEntity postForObject postForEntity

1：getForObject（URL,TYPE）

返回的直接就是Java对象。

2：ResponseEntity entity = restTemplate.getForEntity(userSericeURL+"findAll", List.class);

//get For Entity返回的数据是大于你getForoObject的

//getForObject仅仅只返回要你的数据

//getForEntity的body是你的数据

//还包含了你当前整个请求状态，请求的数据的类型 和时间 和你的请求头和status code等等信息

System.out.println(entity);

return (List)entity.getBody();

单纯的getForObject有三种。getForEntity也有三种。

@GetMapping("findById/{uid}")

public Userinfo findById(@PathVariable Integer uid){

return restTemplate.getForObject(userSericeURL+"findById?uid="+uid,Userinfo.class);

}

@GetMapping("findById1/{uid}")

public Userinfo findById1(@PathVariable Integer uid){

return restTemplate.getForObject(userSericeURL+"findById?uid={uid}", Userinfo.class,uid);

}

@GetMapping("findById2/{uid}")

public Userinfo findById2(@PathVariable Integer uid){

Map map = new HashMap();

map.put(uid, uid);

return restTemplate.getForObject(userSericeURL+"findById?uid={uid}", Userinfo.class,map);

}

二 Eureka 注册中心，服务发现组件

1：所有的实例都应该把自己注册到注册中心Eureka。

2: Eureka能够发现所有的实例的服务

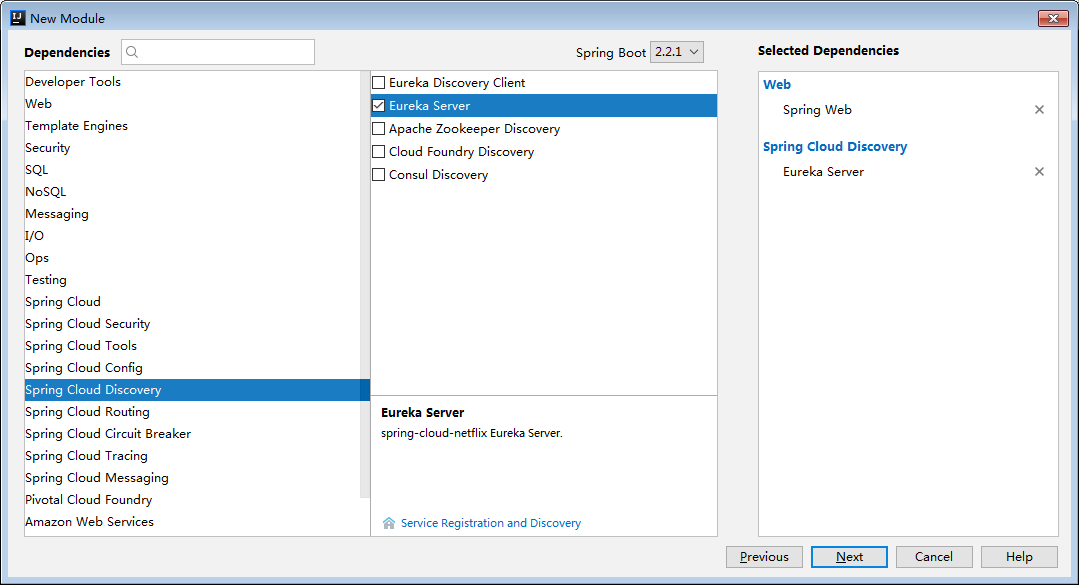
3： 提供了实例的移除。

4：维系了心跳机制

5：有健康检查

搭建Eureka服务器：

1：新建项目，选中EurekaServer



<dependency>

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

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

2:配置yml

server:

port: 9999

#给当前这个服务的实例提供一个名称

spring:

application:

name: eureka

eureka:

client:

register-with-eureka: false #是否把自己注册到eureka 因为本身就是eureka 而且因为只有一个eureka所以是false

fetch-registry: false #是否从eureka上检索服务和API列表

service-url:

defaultZone: http://localhost:9999/eureka/ #当前eureka服务的实例的地址

instance:

prefer-ip-address: true #是否允许以ip进行注册

3：启动

@EnableEurekaServer类加载