# 1、负载均衡

## 1、服务端负载均衡

### 我们通常说的负载均衡都是服务端的负载均衡，其中分为硬件负载均衡和软件负载均衡，

#### 硬件负载均衡主要是服务器节点之间专门安装用于负载均衡的设备，如F5，

#### 软件负载均衡则通过在服务器上安装具有负载均衡的功能或模块来完成请求分发，比如Nginx等。

### 原理:当客户段发送请求到负载均衡设备的时候，该设备按照某种算法（轮询，权重负载，按照流量负载）

## 2、客户端负载均衡

### 客户端负载均衡所有的客户端节点都维护着自己要访问的服务器清单，而这些服务器清单来自于服务注册中心，

# 2、GET请求（restTemplate.getForEntity）

## 1、服务提供者Controller代码

@RestController  
public class HomeController {  
  
  
 @Autowired  
 private DiscoveryClient client;  
  
 @RequestMapping(value = "hello",method = RequestMethod.***GET***)  
 @ResponseBody  
 public String index(String str){  
 ServiceInstance serviceInstance = client.getLocalServiceInstance();  
 System.***out***.println(  
 "host："+serviceInstance.getHost()+"/n " +  
 "service\_id :"+serviceInstance.getServiceId());  
 return str;  
 }  
  
}

## 2、服务消费者代码，请求提供者

@RestController  
public class ConsumerController {  
  
  
 @Autowired  
 RestTemplate restTemplate;  
  
 @ResponseBody  
 @RequestMapping(value = "/ribbon-consumer", method = RequestMethod.***GET***)  
 public String helloConsumer() {  
 return restTemplate.getForEntity("http://HELLO-SERVICE/hello?str=healejean",String.class).getBody();  
 }  
}

## 3、浏览器查看 http://localhost:9000/ribbon-consumer



## 4、占位符{1}和请求获取对象

### 1、服务提供者方法

@RequestMapping(value = "user\_url",method = RequestMethod.***GET***)  
@ResponseBody  
public User index(User user){  
 ServiceInstance serviceInstance = client.getLocalServiceInstance();  
 System.***out***.println(  
 "host："+serviceInstance.getHost()+"/n " +  
 "service\_id :"+serviceInstance.getServiceId());  
  
 user.setId("1");  
 return user;  
}

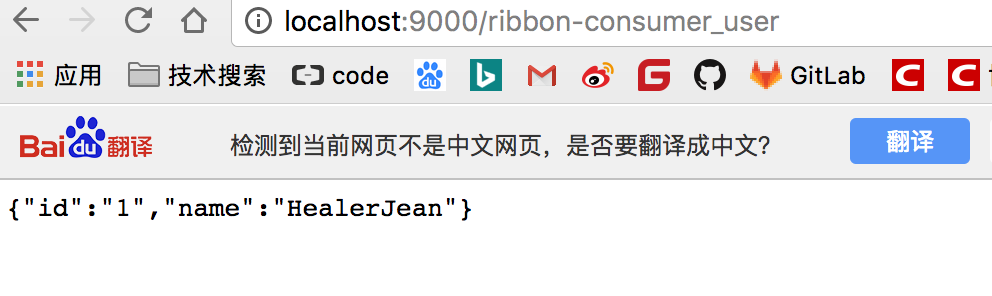
### 2、服务消费者获取对象

#### 解释：

#### 1、这里的Uerl对象内容和上面的内容是一样的，但是位置不是同一个的位置

#### 2、{1} 表示占位符，1表示第一个、第三位参数是一个Object数组

@ResponseBody  
@RequestMapping(value = "/ribbon-consumer\_user", method = RequestMethod.***GET***)  
public User userUrl() {  
 ResponseEntity<User> responseEntity = restTemplate.getForEntity("http://HELLO-SERVICE/user\_url?name={1}",User.class,"HealerJean");  
 User body = responseEntity.getBody();  
 return body;  
}



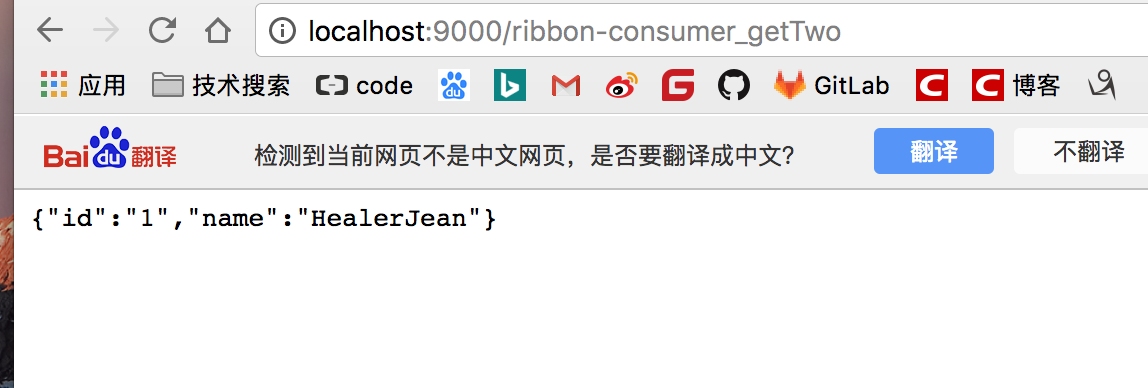
## 5、 map传参占位符{name}

@ResponseBody  
@RequestMapping(value = "/ribbon-consumer\_userMap", method = RequestMethod.***GET***)  
public User userUrlMap() {  
 Map<String,Object> map = new HashMap<>();  
 map.put("name","HealerJean");  
ResponseEntity<User> responseEntity = restTemplate.getForEntity("http://HELLO-SERVICE/user\_url?name={name}",User.class,map);  
 User body = responseEntity.getBody();  
 return body;  
}



## 6、Get的第二种请求方式 restTemplate.getForObject,传参形式和上面的getForEntity是一样的,只不过不到需要.getBody了

/\*\*  
 \* 4、get的第二中请求方式  
 \*/  
  
@ResponseBody  
@RequestMapping(value = "/ribbon-consumer\_getTwo", method = RequestMethod.***GET***)  
public User getTwo() {  
 User body = restTemplate.getForObject("http://HELLO-SERVICE/user\_url?name={1}",User.class,"HealerJean");  
 return body;  
}



# 3、post请求(传递的方式其实相当于表单)

## 1、第一种方式 postForEntity

### 1、服务提供者的post方法

@RequestMapping(value = "user\_urlPost",method = RequestMethod.***POST***)  
@ResponseBody  
public User urlPost(User user){  
 ServiceInstance serviceInstance = client.getLocalServiceInstance();  
 System.***out***.println(  
 "host："+serviceInstance.getHost()+"/n " +  
 "service\_id :"+serviceInstance.getServiceId());  
  
 user.setId("1");  
 return user;  
}

### 2、发起post的请求

/\*\*  
 \* 5、post请求  
 \*/  
  
@ResponseBody  
@RequestMapping(value = "/ribbon-consumer\_postFirst", method = RequestMethod.***GET***)  
public User postFirst() {  
 User user = new User();  
 user.setName("HealerJean");  
 ResponseEntity<User> responseEntity =

restTemplate.postForEntity("http://HELLO-SERVICE/user\_urlPost",user,User.class);  
 User body = responseEntity.getBody();  
 return body;  
}

## 2.、第二种方式 postForObject

# 4、put请求(restTemplate.put不保证正确)

## 解释：

### 1、1、传递对象和id过去

/\*\*  
 \* 6、put请求  
 \*/  
  
@ResponseBody  
@RequestMapping(value = "/ribbon-consumer\_put", method = RequestMethod.***GET***)  
public User put() {  
 User user = new User();  
 user.setName("HealerJean");  
 String id= "2";  
  
 restTemplate.put("http://HELLO-SERVICE/user\_url/{1}",user,id);  
 return user;  
}

# 5、delete请求（restTemplate.delete）

## 解释：

## 1、传递的参数主要是id

/\*\*  
 \* delete 请求  
 \*/  
@ResponseBody  
@RequestMapping(value = "/ribbon-consumer\_delete", method = RequestMethod.***GET***)  
public User delete() {  
 String id= "2";  
 restTemplate.delete("http://HELLO-SERVICE/user\_url/{1}",id);  
 return null;  
}