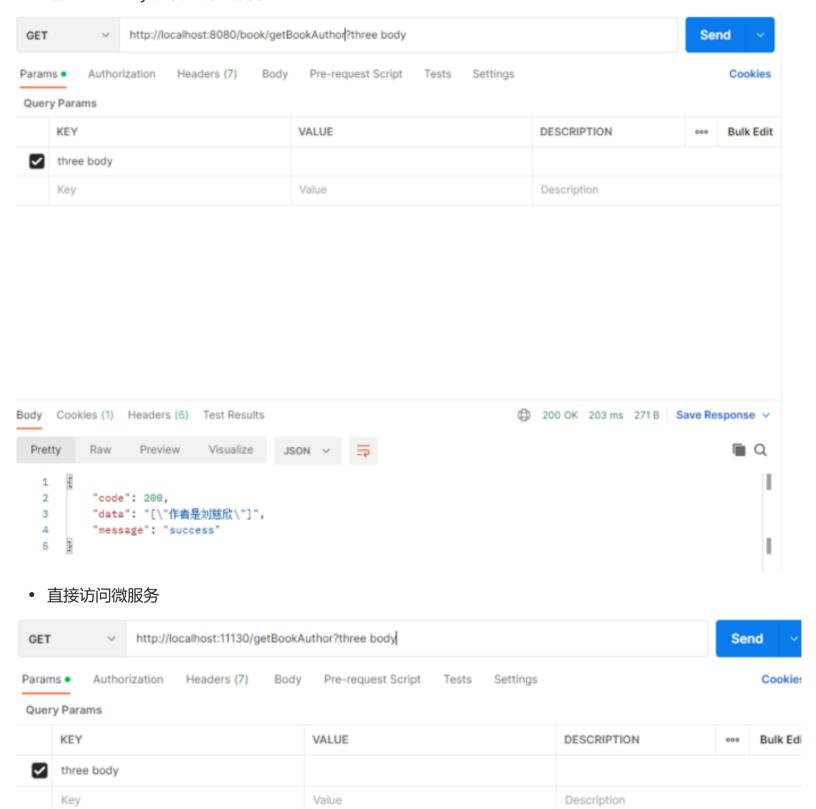
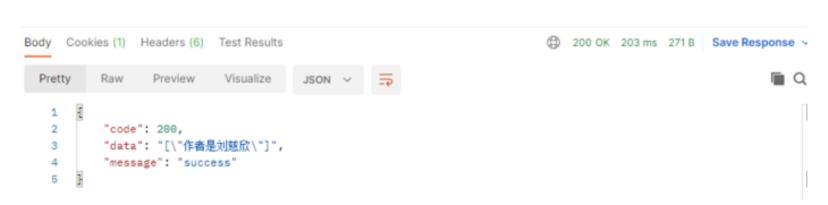
# HW7

# 微服务展示

• 通过 Gateway 转发访问微服务





• 查看注册的服务

# System Status

Environment	N/A		
Data center	N/A		
Current time	2022-11-17T22:11:15 +0800		
Uptime	00:44		
Lease expiration enabled	true		
Renews threshold	5		
Renews (last min)	8		

HOME

# **DS Replicas**

localhost

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BOOK-SERVICE	n/a (1)	(1)	UP (1) - Wave:book-service:11130
GATEWAY	n/a (1)	(1)	UP (1) - Wave:gateway:8080

# 架构说明

#### **MicroService**

#### 部署方式

部署在 11130 端口, 在 yaml 中说明服务注册的相关参数, 在服务启动后会自动注册到 eureka。

### 使用方式

- 直接访问微服务所在的位置
- 访问 Gateway, 通过在请求中添加相关服务信息, 使 Gateway 将其转发到对应服务

### **Gateway**

#### 部署方式

Gateway 部署在8080端口,在 yaml 中说明服务注册的相关参数,在服务启动后会自动注册到 eureka。

#### 作用

Gateway 是在 Spring 生态系统之上构建的 API 网关服务,根据 web 请求和匹配条件,定位到真正的服务节点,并在这个转发过程的前后,进行一些精细化控制。

使用了 Gateway 后,客户端需要访问微服务架构下的服务时,不再需要关心这些服务的 IP 和 port,只需要发送对应的请求给 Gateway, 其便会将其处理后转发给对应的微服务,这将微服务与前端解耦。另一方面 Gateway 还可以提供过滤器,对格式不正确的访问请求直接过滤,减轻后端访问压力。

# **Service Registry**

#### 部署方式

选用 Eureka 作为服务治理组件, 部署在 8040 端口。

#### 作用

选用 Eureka 作为服务治理组件,提供服务注册和服务发现功能,微服务提供者在服务启动后向 Eureka 发起 Service Registry,在其运行过程中需要定时向 Eureka 发送心跳证明其还存活,当其停止服务后,向 Eureka 发起 cancel 请求。

通过 Service Registry,服务消费者(在本次项目中为 Gateway)便可以根据服务的名字拉取服务的信息,且这种方式不需要与微服务的域名/端口进行绑定,便于微服务的更改。