Eureka Server：

<http://localhost:8761/>

Product Service list

<http://localhost:8771/api/v1/product/list>

Product Service find

<http://localhost:8771/api/v1/product/find?id=1>

Order save (ribbon和feign)

<http://localhost:8781/api/v1/order/save?product_id=6&user_id=4>

Order list (feign)

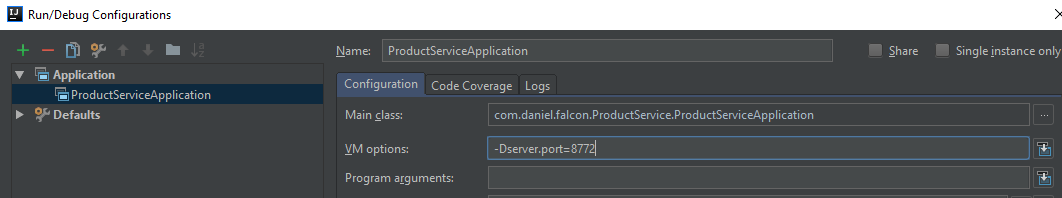
<http://localhost:8781/api/v1/order/list>

启动顺序: Eureka Server 🡪 Product 🡪 Order

千万不可以并行操作,后一个必须等前一个完全结束后,才能开始.

Dell EMC的labtop好像Order无法向Product发送消息(Connection timeout)

使用IntelliJ打开多个Product

1. 8771一定要使用application.yal中进行配置
2. 设置VM Options：-Dserver.port=8772
3. 
4. Apply
5. 直接点绿色箭头



1. 在Eureka Server上check一下

测试feign的调用timeout功能

出现问题

1. 在ProductController.java – findById, 打开sleep函数。
2. 重启ProductService
3. 关闭ProductOrderFeign中的application.yml， 关闭 #修改调用超时时间。
4. 重启ProductOrderFeign
5. 发送 http://localhost:8781/api/v1/order/save?product\_id=6&user\_id=4

出现Read timed out executing GET <http://product-service/api/v1/product/find?id=6>

解决问题

1. 然后打开ProductOrderFeign中的application.yml， 关闭 #修改调用超时时间
2. 重启ProductOrderFeign
3. 发送 http://localhost:8781/api/v1/order/save?product\_id=6&user\_id=4
4. {"id":0,"productName":"\"椅子 data from port=8771\"","tradeNo":"a2782788-cca3-405a-b9fe-daf56f92b3ff","price":253,"createTime":"2018-11-05T15:09:27.680+0000","userId":4,"userName":null}

测试Hystrix降级功能

1. 启动Eureka Server 🡪 Product 🡪 OrderHystrix
2. 发送 http://localhost:8781/api/v1/order/save?product\_id=6&user\_id=4，确保正常功能
3. 关闭ProductService
4. 再发送<http://localhost:8781/api/v1/order/save?product_id=6&user_id=4>
5. 显示{"msg":"抢购人数太多，您被挤出来了，稍等重试","code":-1}

并且在log中出现：“feign 调用product-service findbyid 异常”

注意：如果通过implements方式实现fallback方法，必须在yml配置文件中，添加

feign:

hystrix:

enabled: true

不然，ProductClient直接使用子类的fallback方法，即使ProductService正常， 也会直接跳过。

Windows安装redis

<https://blog.csdn.net/u012343297/article/details/78839063>

启动redis

#> cd ~/Work/code/redis64-2.8.17

#> ./redis-server.exe redis.windows.conf

Linux安装,启动redis

1. 将课程中的redis-4.0.11.tar.gz拷贝到linux机器
2. #> tar -zxvf redis-4.0.11.tar.gz
3. #> cd /home/redis-4.0.11
4. #> make
5. #> cd src
6. ./redis-server ../redis.conf

Hystrix降级并使用redis缓存机制，控制发短信次数（20s一次）

1. 启动redis

#> cd ~/Work/code/redis64-2.8.17

#> ./redis-server.exe redis.windows.conf

1. 启动Eureka Server 🡪 Product 🡪 OrderHystrixRedis
2. 发送 http://localhost:8781/api/v1/order/save?product\_id=6&user\_id=4，确保正常功能
3. 关闭ProductService
4. 再发送<http://localhost:8781/api/v1/order/save?product_id=6&user_id=4>
5. 显示{"msg":"抢购人数太多，您被挤出来了，稍等重试","code":-1}

并且在log中出现：

feign 调用product-service findbyid 异常

紧急短信，用户下单失败，请离开查找原因,ip地址是=0:0:0:0:0:0:0:1

1. 再发送<http://localhost:8781/api/v1/order/save?product_id=6&user_id=4>

Log中显示：

feign 调用product-service findbyid 异常

已经发送过短信，20秒内不重复发送

说明Redis缓存work了。

Hystrix资源隔离

<https://blog.csdn.net/liuchuanhong1/article/details/73718794>

Hystrix默认的隔离策略是：Thread， 并且默认的Timeout时间是1ms。

如果ProduceService需要2s响应时间，Feign的timeout时间改为4s，Hystrix也会执行降级操作，这是因为没有修改Hystrix的timeout时间。

测试Hystrix timeout功能

1. 在ProductController.java – findById, 打开sleep函数。
2. 重启ProductService
3. 在OrderServiceHystrix/OrderServiceHystrixRedis中的application.yml， 关闭 #设置超时时间。
4. Feign timeout时间设置为4s
5. 重启OrderServiceHystrix/OrderServiceHystrixRedis
6. 发送 http://localhost:8781/api/v1/order/save?product\_id=6&user\_id=4

显示{"msg":"抢购人数太多，您被挤出来了，稍等重试","code":-1}

并且在log中出现：

feign 调用product-service findbyid 异常

1. 可以看到Feign timeout时间设置无效
2. 在OrderServiceHystrix/OrderServiceHystrixRedis中的application.yml， 关打开#设置超时时间。
3. 重启OrderServiceHystrix/OrderServiceHystrixRedis
4. 消息显示正常。

Api-gateway IP

<http://localhost:9000/apigateway/order/api/v1/order/save?product_id=6&user_id=4>

api-gateway屏蔽功能

1. 修改application.yml配置

增加ignored-patterns: /\*-service/\*\*，重启

1. 马上出现404 ， “No message available”

解决api-gateway无法获取cookie信息

1. 默认情况下，api-gateway会ignore cookie信息
2. 修改application.yml配置，添加sensitive-headers: （将sensitive-headers置空），这意味着，我们的配置希望api-gateway不屏蔽任何的header信息。

测试api-gateway的Auth的token认证

1. 使用apigatewayAuth作为网关
2. 使用restlet，发一个request，header中不带token的，回复401

<http://localhost:9000/apigateway/order/api/v1/order/save?product_id=6&user_id=4>

1. 在header中添加token，发送同样的request，能够回复正常数据。
2. header中不带token，但是拍URL的parameter中添加token，发送，能回复正常数据。

<http://localhost:9000/apigateway/order/api/v1/order/save?product_id=6&user_id=4&token=daniel>

1. 对于/apigateway/order/api/v1/order/list，api-gateway的Auth的token认证同样需要。只要重复2-4即可。