链滴

SpringCloud Alibaba 微服务实战十六 - 2.2.1.RELEASE 版本升级

作者：jianzh5

原文链接：https://ld246.com/article/1591174780962

来源网站：链滴

概述

好久没有更新SpringCloud Alibaba 系列的文章了，今天我们来将版本升级到最新的毕业版本。并且原来容器化部署的组件seata、nacos、sentinel拉出来单独部署，为我们后面k8s部署作准备。

官方推荐版本如下：

Spring Cloud Version	Spring Cloud Alibaba Version	Spring Boot Version
Spring Cloud Hoxton.SR3	2.2.1.RELEASE	2.2.5.RELEASE
Spring Cloud Hoxton.RELEASE	2.2.0.RELEASE	2.2.X.RELEASE
Spring Cloud Greenwich	2.1.2.RELEASE	2.1.X.RELEASE
Spring Cloud Finchley	2.0.2.RELEASE	2.0.X.RELEASE
Spring Cloud Edgware	1.5.1.RELEASE	1.5.X.RELEASE

这篇文章主要是讲升级过程中遇到的一些问题并讲述解决的过程与方法，如果要了解详细用法还请翻之前的文章。

主版本升级

```
<properties>
...
    <spring-boot.version>2.2.5.RELEASE</spring-boot.version>
    <alibaba-cloud.version>2.2.1.RELEASE</alibaba-cloud.version>
    <springcloud.version>Hoxton.SR3</springcloud.version>
...
</properties>
```

修改parent模块主pom文件对应的组件版本，修改完成后重新下载jar包。

nacos 1.2

nacos的升级比较容易，按照下面步骤两步即可完成。

- 初始化nacos数据库 nacos-mysql.sql
- 修改nacos配置文件 application.properties，将数据库相关配置注释放开，并修改成自己的数库配置

```
*************** Config Module Related Configurations ***************#
### If user MySQL as datasource:
spring.datasource.platform=mysql

### Count of DB:
db.num=1

### Connect URL of DB:
db.url.0=jdbc:mysql://1.1.1.1:3306/nacos?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true
db.user=root
db.password=xxxxxx
```

seata 1.2

seata初始化的过程我们之前教程中有过详细说明，但是新老版本之间差异比较大，我们这里再顺带一下，大家可以按照如下步骤完成。

- 在业务系统中创建数据表 undo_log，sql文件从下面地址获取：

https://github.com/seata/seata/blob/develop/script/client/at/db/mysql.sql

- 在你的mysql数据库中创建名为seata的库,并初始化，sql文件从下面地址获取：

https://github.com/seata/seata/blob/develop/script/server/db/mysql.sql

- seata新版本修改了artifactId，所以我们需要修改seata的依赖

```
<dependency>
   <groupId>com.alibaba.cloud</groupId>
   <artifactId>spring-cloud-starter-alibaba-seata</artifactId>
</dependency>
```

SpringCloud Alibaba 2.2.1 RELEASE 使用的是 SEATA1.1的版本，如果想体验SEATA1.2的特性，以在此基础上去掉seata的依赖，手动加入1.2版本。

```
<dependency>
   <groupId>com.alibaba.cloud</groupId>
   <artifactId>spring-cloud-starter-alibaba-seata</artifactId>
   <exclusions>
      <exclusion>
         <artifactId>seata-spring-boot-starter</artifactId>
         <groupId>io.seata</groupId>
      </exclusion>
   </exclusions>
</dependency>

<dependency>
   <groupId>io.seata</groupId>
   <artifactId>seata-spring-boot-starter</artifactId>
   <version>1.2.0</version>
</dependency>
```

- 修改seata客户端的配置

原来我们在客户端是使用 registry.conf 作为seata的配置文件，现在需要将配置移到application.yml配置中心中，具体配置大家可以参考官网文件 https://github.com/seata/seata/blob/develop/script/client/spring/application.yml， 我的配置如下：

```
seata:
  enabled: true
  application-id: ${spring.application.name}
  tx-service-group: account_service_group
  enable-auto-data-source-proxy: true
  config:
    type: nacos
    nacos:
      namespace:
      serverAddr: 10.0.23.48:8848
      group: SEATA_GROUP
```

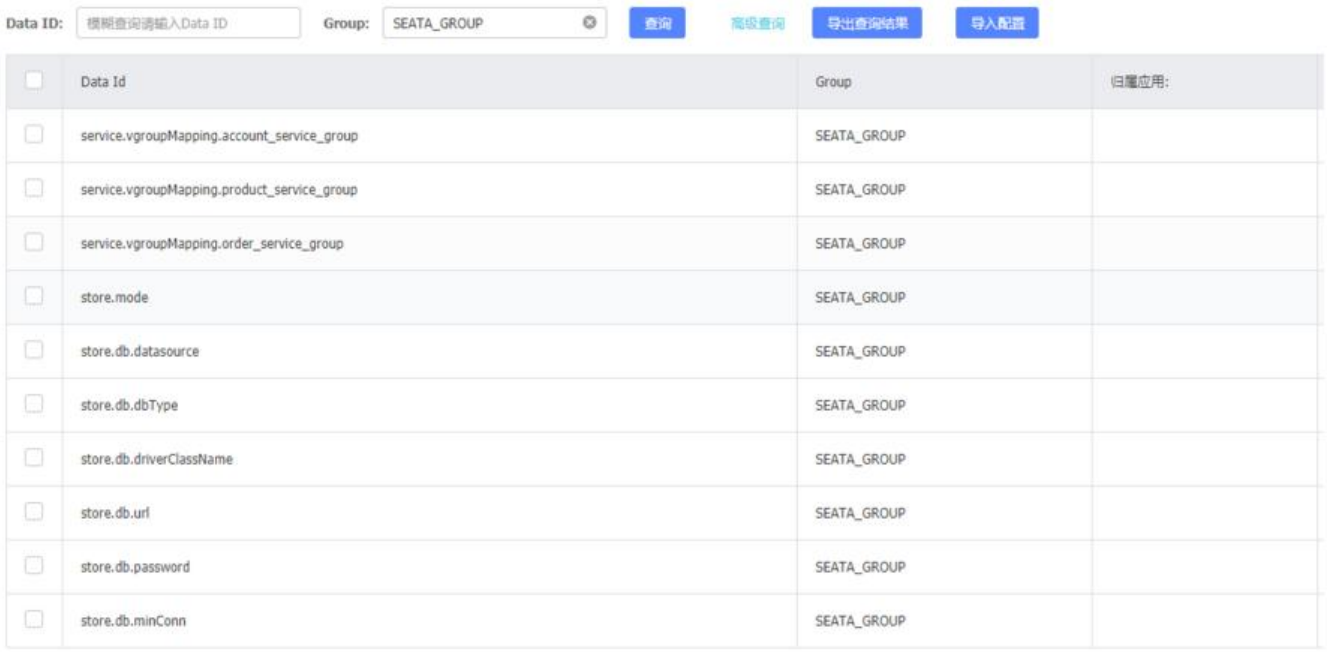
```
    userName: "nacos"
    password: "nacos"
  registry:
    type: nacos
    nacos:
      application: seata-server
      server-addr: 10.0.23.48:8848
      namespace:
      userName: "nacos"
      password: "nacos"
```

其他模块大家自行修改。

● 将seata的配置推送到nacos，这里采用的是db模式，操作步骤请参照官方说明：https://github.com/seata/seata/tree/develop/script/config-center

```
service.vgroupMapping.account_service_group=default
service.vgroupMapping.product_service_group=default
service.vgroupMapping.order_service_group=default
store.mode=db
store.db.datasource=druid
store.db.dbType=mysql
store.db.driverClassName=com.mysql.jdbc.Driver
store.db.url=jdbc:mysql://10.0.23.48:3306/seata?useUnicode=true
store.db.user=root
store.db.password=xxxxxx
store.db.minConn=5
store.db.maxConn=30
store.db.globalTable=global_table
store.db.branchTable=branch_table
store.db.queryLimit=100
store.db.lockTable=lock_table
store.db.maxWait=5000
```

修改完成后在git中执行shell命令 sh nacos-config.sh -h 10.0.23.48 -p 8848 -g SEATA_GROUP -u acos -w nacos，执行完成后配置文件就被推送到了nacos



● 修改seata服务端配置 registry.conf， 完成后启动seata服务端

```
registry {
  type = "nacos"
  nacos {
    application = "seata-server"
    serverAddr = "10.0.23.48:8848"
    namespace = ""
    cluster = "default"
    username = "nacos"
    password = "nacos"
  }
}
config {
  type = "nacos"
  nacos {
    serverAddr = "10.0.23.48:8848"
    namespace = ""
    group = "SEATA_GROUP"
    username = ""
    password = ""
  }
}
```

● 经过上面几步配置seata就升级完成了，请大家自行测试

```
2020-06-03 14:58:40,404 INFO  OrderServiceImpl:53 - ORDER XID is: 10.0.23.48:8091:2013351648
2020-06-03 14:58:42,130 INFO  DefaultGlobalTransaction:143 - [10.0.23.48:8091:2013351648] commit status: Committed
2020-06-03 14:58:42,805 INFO  RmMessageListener:79 - onMessage:xid=10.0.23.48:8091:2013351648,branchId=2013351650,branchType=AT,resour
2020-06-03 14:58:42,806 INFO  AbstractRMHandler:97 - Branch committing: 10.0.23.48:8091:2013351648 2013351650 jdbc:mysql://10.0.23.48:
2020-06-03 14:58:42,809 INFO  AbstractRMHandler:105 - Branch commit result: PhaseTwo_Committed
```

sentinel 1.7

使用sentinel的时候，很多同学都会遇到在sentinel控制台不显示api管理菜单，不能读取nacos限流置等情况，大家可以在gateway模块启动的时候加上启动参数 -Dcsp.sentinel.app.type=1，重启sentnel控制台即可正常显示。

在之前的文章中曾经提到过老版本的Sentinel在与网关集成时限流不生效的问题，原因是因为sentine获取到的网关id并不是我们配置的account-service，而是加了 CompositeDiscoveryClient_ 前缀，文说明如下：

限流不生效

各位在使用过程中如果发现网关层限流不生效，可以以debug模式启动网关服务，然后对网关过滤器 `SentinelGatewayFilter` 中的filter方法进行调试，我发现sentinel获取到的网关id并不是我们配置的 `account-service` ，而是加了 `CompositeDiscoveryClient_` 前缀，如下图所示：

```
public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  exchange: "SecurityContex
    Route route = (Route)exchange.getAttribute(ServerWebExchangeUtils.GATEWAY_ROUTE_ATTR);  route: "Route{:
    Mono<Void> asyncResult = chain.filter(exchange);  asyncResult: "MonoDefer"  chain: FilteringWebHandler[
    String apiName;
    if (route != null) {
        String routeId = route.getId();  routeId: "CompositeDiscoveryClient_account-service"  route: "Route
        Object[] params = this.paramParser.parseParameterFor(routeId, exchange, (r) -> {  paramParser: Gate
            return r.getResourceMode() == 0;
        });
        apiName = (String)Optional.ofNullable(GatewayCallbackManager.getRequestOriginParser()).map((f) ->
            return (String)f.apply(exchange);
        }).orElse( other: "");
        asyncResult = asyncResult.transform(new SentinelReactorTransformer(new EntryConfig(routeId,  resource
```

所以我们需要修改 `gateway-sentinel-flow` 的配置，给我们的resource 也加上前缀，修改完的配置如下：

在新版本中已经没有这个问题了，所以我们可以在网关限流配置文件中将前缀删除。删除后的配置如：

auth-service

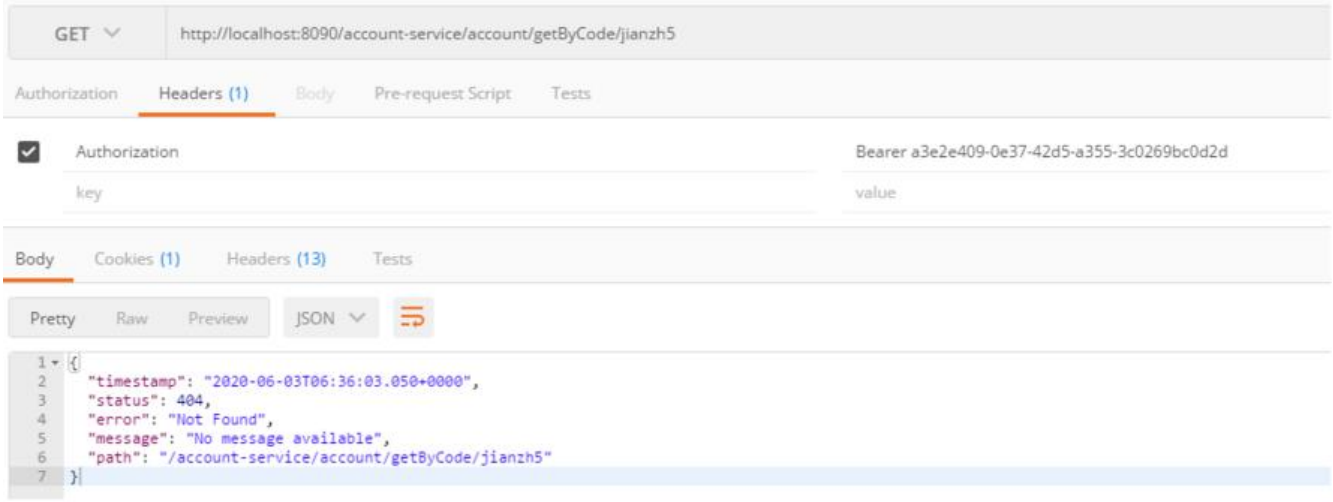
使用postman去auth-service获取access_token的时候如果出现如下错误org.springframework.security.core.authority.SimpleGrantedAuthority; local class incompatible: stream classdesc serialVersionUID = 510, local class serialVersionUID = 520

```
Caused by: java.io.InvalidClassException: org.springframework.security.core.authority.SimpleGrantedAuthority; local class incompatible: stream classdesc seri
    at java.io.ObjectStreamClass.initNonProxy(ObjectStreamClass.java:616) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:1630) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1521) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1781) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1353) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:373) ~[?:1.8.0_112]
    at java.util.ArrayList.readObject(ArrayList.java:791) ~[?:1.8.0_112] <4 internal calls>
    at java.io.ObjectStreamClass.invokeReadObject(ObjectStreamClass.java:1058) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1909) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1808) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1353) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:2018) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1942) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1808) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1353) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:2018) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1942) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1808) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1353) ~[?:1.8.0_112]
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:373) ~[?:1.8.0_112]
    at org.springframework.security.oauth2.common.util.SerializationUtils.deserialize(SerializationUtils.java:43) ~[spring-security-oauth2-2.3.4.RELEASE.jar:
    ... 110 more
2020-06-03 12:49:03,302 WARN  TokenEndpoint:169 - Handling error: NullPointerException, null
```

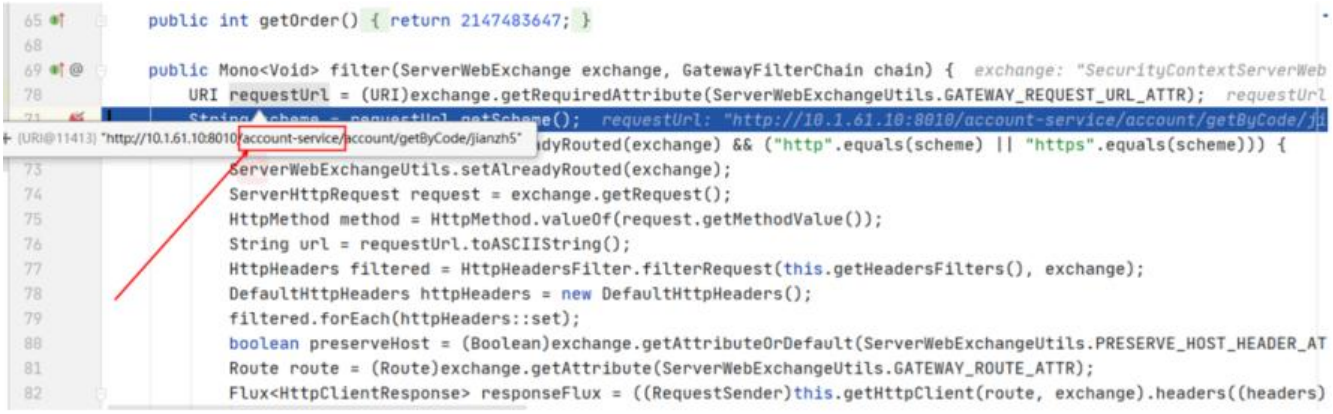
出现这个问题的原因是原来数据库已经存储了账号对应的access_token了，但是SpringSecurity不支跨版本的序列化，解决方法也很简单，只要把你的用户对应的access_token给删除，重新生成即可。

SpringCloud Gateway

使用PostMan做集成测试时发现接口调用一直提示404 Not Found 错误。



通过对源码文件org.springframework.cloud.gateway.filter.NettyRoutingFilter 的调试，发现新版的SpringCloud Gateway在做转发时还保留我们配置的前缀。



如上图所示，转发后还带上前缀导致不能找到对应的请求路径，所以出现404异常。

要解决这个问题我们需要在转发前删掉这一前缀，刚好SprtingCloud Gateway提供了StripPrefix GatewayFilter filter，可以用来解决这一问题：

The StripPrefix GatewayFilter factory takes one parameter, parts. The parts parameter indicate the number of parts in the path to strip from the request before sending it downstream.

详情请参看：https://cloud.spring.io/spring-cloud-static/spring-cloud-gateway/2.2.2.RELEASE/eference/html/#the-stripprefix-gatewayfilter-factory

既然知道了异常原因也找到了解决方法，那就很简单了。只需要在网关模块配置映射时加上默认过滤即可：

```
spring:
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
      routes:
      - id: account-service
        uri: lb://account-service
```

```
      predicates:
        - Path=/account-service/**
      ...
    # 解决前缀转发问题
    default-filters:
      - StripPrefix=1
```

经过以上几步我们的SpringCloud alibaba 升级完成，升级过程中会遇到各种各样的问题，面对问题时候大家不要急躁，多通过代码调试定位问题，定位到问题后我相信利用好搜索引擎是可以解决的。