新巴巴 c88fa0ce97e84005be785e6d2023b18c:USER\_NAME fbb2016

Git yxx

Github yingxx26 y770411081

Jenkins yxx y770411081

Activiti6.0 admin test

**安装笔记**

**130 solr activemq zookeeper redis storm nginx2（分发层）kafka Hadoop**

**134 zookeeper redis dubboadmin storm nginx2（应用层）kafka Hadoop flume**

**135 zookeeper redis nginx1 storm nginx2（应用层）kafka Hadoop hbase**

**136 redis mycat Hadoop**

**137redis nginx1 mysql主Hadoop spark**

**138redis mysql从Hadoop spark flume**

**139 Hadoop hive（数据源139） hbase spark sqoop**

**140 elasticsearsh**

**Jvm调优**

**-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=d:\jvm.dump**

**-XX:+PrintGCDetails -Xms512M -Xmx512M**

**-Dcatalina.base="D:\testwork\.metadata\.plugins\org.eclipse.wst.server.core\tmp0" -Dcatalina.home="D:\utils\Tomcat 7.0" -Dwtp.deploy="D:\testwork\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps" -Djava.endorsed.dirs="D:\utils\Tomcat 7.0\endorsed"**

**慢查询日志/mysql调优**

**mysqldumpslow -s r -t 2 /var/lib/mysql/storage6-slow.log**

**--C:\ProgramData\MySQL\MySQL Server 5.7\Data\YXX-slow.log**

**---Cd 到C:\Program Files\MySQL\MySQL Server 5.7\bin**

**show variables like '%quer%';**

**set global log\_queries\_not\_using\_indexes=on;**

**set global long\_query\_time =0.01;**

**set global low\_query\_log=on;**

**/var/lib/mysql/storage6-slow.log**

**并发包只能保证效率，性能，自身的操作线程安全，不能保证复合操作 线程安全**

**如果需要业务同步，那么**

**比如concurrnethashmap的读不需要加锁，适用于读比写多的**

**Collections.synchronizedhashmap适用于多线程的原子业务操作**

**Hashtable全加锁（放弃）**

**同步块下Hashmap能用，但是，如果别的地方也要考虑多线程，而且没有复合操作，且读操作占多，此时，concurrnethashmap适用**

**Copyonwritearraylist读不需要加锁，适用于读比写多的**

**Vector放弃**

**Collections.synchronizedlist适用读写均匀**

开机启动 生效指令 chmod +x /etc/rc.d/rc.local

1，防火墙 要一个一个配置

vi /etc/sysconfig/iptables

service iptables restart

2， 启动tomcat（dubbo-admin 8090端口）

记得端口号和zookeeper要不一样，包括8080 8005 8009

dubbo-admin的Zookeeper jar包版本要和项目一样

严重：Error listenerStart Web-inf/classes中加入log4j.properties

Server.xml加上

<Context path="" docBase="/usr/local/src/apache-tomcat-7.0.47/webapps/babasport-portal-0.0.1-SNAPSHOT

" reloadable="false"/>

去掉test类和test服务

启动 /usr/local/src/apache-tomcat-7.0.47/bin/startup.sh

日志 tail -f /usr/local/src/apache-tomcat-7.0.47/logs/catalina.out

数据库访问路径修改，权限修改http://www.cnblogs.com/hongten/archive/2012/10/25/mysql.html

访问路径 http://192.168.181.130:8097/babasport-portal-0.0.1-SNAPSHOT/

3，Jdk安装

/usr/local/src/jdk1.7.0\_45

软连接 ln -s /usr/java/jdk1.7.0\_67-cloudera /usr/java/latest

4，添加tomcat环境变量  
# [root@thinkpig007 ~]# vi /etc/profile  
加入下代码  
#set environment variable  
 exportJAVA\_HOME=/usr/java/jdk1.7.0\_45  
 export CLASSPATH=.:$JAVA\_HOME/jre/lib/rt.jar:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar  
 exportTOMCAT\_HOME=/usr/local/tomcat7  
 exportCATALINA\_HOME=/usr/local/tomcat7  
 exportPATH=$PATH:$JAVA\_HOME/bin

修改应用环境变量，是配置生效  
 [root@thinkpig007 bin]# source /etc/profile

5，Zookeeper ： it is paoblbly not running 原因是环境变量的配置文件搞错

应该用全局地址/etc/profile，别的地址可能会重置

[profile、bashrc、bash\_profile之间的区别和联系](http://blog.csdn.net/chenchong08/article/details/7833242)

启动zookeeper /usr/local/src/zookeeper-3.4.6/bin/zkServer.sh start

6，fastdfs

**先启动tracker后启动storage**

**启动tracker /etc/init.d/fdfs\_trackerd start**

**重启/usr/bin/restart.sh /usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf**

**启动storage /etc/init.d/fdfs\_storaged start**

**重启/usr/bin/restart.sh /usr/bin/fdfs\_storaged /etc/fdfs/storage.conf**

**关闭防火墙 sudo systemctl stop firewalld.service**

**关闭开机启动sudo systemctl disable firewalld.service**

错误ERROR - file: tracker\_proto.c, line: 48, server:, response status 2的解决：

1. Tracker上通过命令/usr/bin/fdfs\_monitor /etc/fdfs/client.conf  
    查看服务状态
2. 防火墙

7，nginx （nginx2 /usr/servers/nginx/sbin/nginx nginx2里面的kafka是storm用的）

启动/usr/local/nginx/sbin/nginx 重启/usr/local/nginx/sbin/nginx -s reload

/usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf /usr/local/src/FastDFS\_v5.05.tar.gz

浏览器访问路径别忘端口号， 调试时要按F8

8,Redis（140单机）

/usr/local/redis/bin/redis-server /usr/local/redis/conf/6379.conf 启动服务端

/usr/local/redis/bin/redis-cli -c -p 6379 启动客户端

9，solr

**启动sh /solr/apache-tomcat-7.0.47/bin/startup.sh**

**关闭sh /solr/apache-tomcat-7.0.47/bin/shutdown.sh**

**动态查看日志tail -f /solr/apache-tomcat-7.0.47/logs/catalina.out**

**访问地址http://192.168.181.130:8100/solr**

10，activemq

**启动cd /usr/local/src/apache-activemq-5.13.0/bin**

**./activemq start**

**管理页面http://192.168.181.130:8161/**

**用户密码admin/admin**

11， 首页 http://localhost:8091/

后台 [http://localhost:8098/control/index.do](http://localhost:8080/control/index.do)

登陆 <http://localhost:8082/login.aspx>

Product 8087

12,maven

build里面的

Install 构建依赖

Package 打成war包

发布到linux

后面访问路径要加上项目名

静态文件访问不到

Server.xml加上

<Context path="" docBase="/usr/local/src/apache-tomcat-7.0.47/webapps/babasport-portal-0.0.1-SNAPSHOT

" reloadable="false"/>

13，

Mysql安装

<http://www.centoscn.com/mysql/2014/0919/3778.html>

授予远程访问权限

grant all privileges on \*.\* to 'root'@'%' identified by '123456';

14 svn

启动/usr/local/src/tomcat-svn/apache-tomcat-7.0.47/bin/startup.sh

日志tail -f /usr/local/src/tomcat-svn/apache-tomcat-7.0.47/logs/catalina.out

访问路径http://192.168.181.130:9000/svnadmin/

15，dubbo-admin

要先启动zookeeper

启动

/usr/local/src/usr/local/src/dubboadmin/apache-tomcat-7.0.47/bin/startup.sh

tail -f /usr/local/src/usr/local/src/dubboadmin/apache-tomcat-7.0.47/logs/catalina.out

134服务器 8195

http://192.168.181.134:8195

集群

16，Zookeeper集群

服务器一 /usr/local/src/zookeeper-3.4.6/bin/zkServer.sh start

服务器二 /usr/local/src/zookeeper-3.4.6-4/bin/zkServer.sh start

服务器三 /usr/local/src/zookeeper-3.4.6-5/bin/zkServer.sh start

data目录只能有myid

端口号不占用,日志

防火墙关闭

17，activemq集群

启动

服务器一 /usr/local/src/apache-activemq-5.13.0-1/bin/activemq start

服务器二 /usr/local/src/apache-activemq-5.13.0-2/bin/activemq start

服务器三 /usr/local/src/apache-activemq-5.13.0-3/bin/activemq start

日志

tail -f /usr/local/src/apache-activemq-5.13.0-1/data/activemq.log

tail -f /usr/local/src/apache-activemq-5.13.0-2/data/activemq.log

tail -f /usr/local/src/apache-activemq-5.13.0-3/data/activemq.log

zooinspect工具看哪个是主节点

<http://192.168.181.134:8162/>

选举的端口号要一致

Zookeeperdown掉activemq要重启

18 Redis 集群

/usr/local/redis3/bin/redis-server /usr/local/redis3/cluster/6379/redis-6379.conf

启动服务端 130

/usr/local/redis3/bin/redis-cli -c -p 6379 启动客户端

/usr/local/redis3/bin/redis-server /usr/local/redis3/cluster/7112/redis-7112.conf

启动服务端 134

/usr/local/redis3/bin/redis-server /usr/local/redis3/cluster/7113/redis-7113.conf

启动服务端 135

/usr/local/redis3/bin/redis-server /usr/local/redis3/cluster/7114/redis-7114.conf

启动服务端 138

/usr/local/redis3/bin/redis-cli -c -p 7114

/usr/local/redis3/bin/redis-server /usr/local/redis3/cluster/7115/redis-7115.conf

启动服务端 137

/usr/local/redis3/bin/redis-server /usr/local/redis3/cluster/7116/redis-7116.conf

启动服务端 136

* 创建集群命令redis-trib create --replicas 1 192.168.181.130:6379 192.168.181.134:7112 192.168.181.135:7113 192.168.181.138:7114 192.168.181.137:7115 192.168.181.136:7116

19 ip地址和上网

没有ip：

方法1，（临时）不能上网：解决方法如下（修改dns配置）)vi /etc/resolv.conf

在此文件最后加入：nameserver 8.8.8.8 不要service network restart 然后ping [www.baidu.com](http://www.baidu.com/)可以ping通，不会再出现unknow host

（永久）vi /etc/sysconfig/network-scripts/ifcfg-ens33

PEERD=no

DNS1=8.8.8.8

DNS2=114.144.114.114

方法2 ，添加网关 vi /etc/sysconfig/network-scripts/ifcfg-ens33（static，gateway）

service network restart

查看路由 route -n

添加路由 route add default gw 192.168.181.2

vi /etc/resolv.conf 改成 search localdomain nameserver 192.168.181.2

方法3；永久路由 vi /etc/sysconfig/network-scripts/ifcfg-ens33

加入 192.168.181.0/24 via 192.168.181.2

0.0.0.0/0 via 192.168.181.2 然后service network restart

20， 杀死端口进程

netstat -nlp

kill -9 2895

21 ，nginx keepalived

服务器 155 192.168.181.155: http://localhost:8091/

135 slave

137 master

/usr/local/nginx/sbin/nginx

启动service keepalived start

Nginx2 /usr/servers/nginx/sbin/nginx -s reload

22 mycat

136 服务器

启动/usr/local/src/mycat/bin/mycat start

日志tail -f /usr/local/src/mycat/logs/wrapper.log

Mycat命令行界面

mysql -u root -p -h127.0.0.1 -P8066

23，storm

复制 scp -r /usr/local/src/storm root@storage8:/usr/local/src

scp /etc/profile root@storage8:/etc/profile

启动storm nimbus >/dev/null 2>&1 & ///130

storm supervisor >/dev/null 2>&1 & ///130 134 135

storm ui >/dev/null 2>&1 & ///130

storm logviewer >/dev/null 2>&1 & ///134 135

网址 <http://192.168.181.130:9999/index.html>

打包package,提交storm jar storm-helloworld-0.0.1-SNAPSHOT.jar com.roncoo.eshop.storm.WordCountTopology wordCountTopology

storm jar eshop-storm-0.0.1-SNAPSHOT.jar com.roncoo.eshop.storm.HotProductTopology HotProductTopology

Storm kill workCountTopology

storm kill HotProductTopology

zkCli.sh -server 192.168.181.130:2181 rmr /taskid-list /var/storm 如果superviser启动后就退出，删除这个目录下的文件，或者删除zookeeper下的storm rmr /storm

24,kafka 0.8.2.1 、kafka\_2.10-0.10.0.0

在cd /usr/local/src/kafka\_2.10-0.8.2.1 或者 cd /usr/local/src/kafka\_2.10-0.10.0.0 目录下 0.8版本 flume不行

启动nohup bin/kafka-server-start.sh config/server.properties &

创建主题bin/kafka-topics.sh --zookeeper 192.168.181.130:2181,192.168.181.134:2181,192.168.181.135:2181 --topic cache-message --replication-factor 1 --partitions 1 --create

创建生产者bin/kafka-console-producer.sh --broker-list 192.168.181.130:9092,192.168.181.134:9092,192.168.181.135:9092 --topic cache-message

创建消费者bin/kafka-console-consumer.sh --zookeeper 192.168.181.130:2181,192.168.181.134:2181,192.168.181.135:2181 --topic cache-message --from-beginning

25，缓存项目（eshop-inventory和eshop-cache是两个解决方案）

（1），主动更新

（超大数据量下的方案，kafka和分发策略不一致）监听kafka消息队列（一般是管理后台更新商品信息），获取到一个商品变更的消息之后，去源服务中调用接口拉去数据，更新到ehcache和redis中，先获取分布式锁，然后才能更新redis，同时更新时要比较时间版本（数据库修改时间字段）

（2）被动重建 （查询）

Nginx到lua到eshop-cache的接口，先redis，后数据库，直接返回nginx，（如果redis没有）同时推送一条消息到一个队列（blokingqueue），后台线程异步消费，后台线程负责先获取分布式锁，然后才能更新redis，同时要比较时间版本

{"serviceId":"productInfoService","productId":5}

——localhost:8080/getProductInfo?productId=5

26 ,hadoop(分布式框架，分布式存储)

第二次启动 可以不用格式化

rm -rf /usr/local/src/hadoop/tmp/dfs

2.6启动journalnode（分别在在storage、storage4、storage5上执行）

cd /usr/local/src/hadoop/

sbin/hadoop-daemon.sh start journalnode

2.7格式化HDFS #在storage9 上执行命令:

cd /usr/local/src/hadoop/

hdfs namenode -format

scp -r tmp/ root@storage8:/usr/local/src/hadoop/

2.8格式化ZKFC(在storage9 上执行即可)

hdfs zkfc -formatZK

2.9启动HDFS(在storage9 上执行)

sbin/start-dfs.sh

2.10启动YARN storage7

cd /usr/local/src/hadoop/

sbin/start-yarn.sh

27：phoenix

启动 /usr/local/src/phoenix/bin/sqlline.py storage,storage4,storage5:2181

建立hbase视图映射：create view "test3"(userid varchar primary key, "info"."name" varchar);

建立hbase表映射：create table "test4"(userid varchar primary key, "info"."name" varchar, "info"."age" varchar);

建立二级 索引 create index test3\_name on "test3"("info"."name");

插入更新 upsert into "test4" values ('1994','yxxxx','21');

28，hbase（方便java客户端） 别忘了在windows本地hosts增加集群节点

先启动hadoop，见24

启动hbase，在139上运行：

/usr/local/src/hbase/bin/start-hbase.sh

6.通过浏览器访问hbase管理页面

192.168.181.139:16010

7.为保证集群的可靠性，要启动多个HMaster

hbase-daemon.sh start master

8.打开命令行界面

/usr/local/src/hbase/bin/hbase shell

Hbase启动失败（）storage  zkCli.sh -server 192.168.181.130:2181 rmr /hbase

有异常storage9 /usr/local/src/hadoop/bin/hadoop fs -rmr /hbase/WALs

29． hadoop异常

删除storage storage4 storage5 zookeeper上的data文件里的version

rm -rf /usr/local/src/zookeeper-3.4.6-4/data/version-2

###注意：严格按照下面的步骤  
  2.5启动zookeeper集群（分别在itcast05、itcast06、tcast07上启动zk）  
   cd /itcast/zookeeper-3.4.5/bin/  
   ./zkServer.sh start  
   #查看状态：一个leader，两个follower  
   ./zkServer.sh status  
  2.6启动journalnode（分别在在itcast05、itcast06、tcast07上执行）  
   cd /itcast/hadoop-2.4.1  
   sbin/hadoop-daemon.sh start journalnode  
   #运行jps命令检验，itcast05、itcast06、itcast07上多了JournalNode进程  
  2.7格式化HDFS  
   #在itcast01上执行命令:  
   hdfs namenode -format  
   #格式化后会在根据core-site.xml中的hadoop.tmp.dir配置生成个文件，这里我配置的是/itcast/hadoop-2.4.1/tmp，然后将/itcast/hadoop-2.4.1/tmp拷贝到itcast02的/itcast/hadoop-2.4.1/下。  
   scp -r tmp/ itcast02:/itcast/hadoop-2.4.1/  
  2.8格式化ZK(在itcast01上执行即可)  
   hdfs zkfc -formatZK  
  2.9启动HDFS(在itcast01上执行)  
   sbin/start-dfs.sh

  2.10启动YARN(#####注意#####：是在itcast03上执行start-yarn.sh，把namenode和resourcemanager分开是因为性能问题，因为他们都要占用大量资源，所以把他们分开了，他们分开了就要分别在不同的机器上启动)  
   sbin/start-yarn.sh

30, hive（翻译mapreduce ，数据分析）

/usr/local/src/hadoop/bin/hdfs dfs -chmod -R 777 /tmp

* /usr/local/src/hive/bin/hive
* 启动hive远程
* nohup /usr/local/src/hive/bin/hive --service hiveserver2 &
* 建表 create external table t\_mygirls(id int ,name string ,age int ,size string) row format delimited fields terminated by “\t”;
* 数据.dat文件
* 上传Hadoop fs -put mygirls.dat /user/hive/warehouse/yun12.db/t\_mygirls
* 删除/usr/local/src/hadoop/bin/hadoop fs -rmr /user/hive/warehouse/yun12.db/t\_mygirls/mygirls.dat
* 初始化元数据 **alter database hive default character set latin1;** flush privileges;
* grant all privileges on \*.\* to 'root'@'%' identified by '123456';
* /usr/local/src/hive/bin/schematool -dbType mysql -initSchema
* 查询在一定要cd /usr/local/src/hive/bin下 ./hive启动 select \* from t\_mygirls；
* Shell脚本/usr/local/src/hive/bin/hive -S -e 'select \* from t\_order';
* load data local inpath '/usr/local/src/order.data' into table t\_order

hive --hiveconf hive.root.logger=DEBUG,console -e 'select count(1) from links'

31， spark

* 启动 /usr/local/src/spark/sbin/start-all.sh

启动shell /usr/local/src/spark/bin/spark-shell --master spark://storage9:7077 --executor-memory 2g --total-executor-cores 4

* 提交例子/usr/local/src/spark/bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://storage9:7077 /usr/local/src/spark/lib/spark-examples-1.6.2-hadoop2.6.0.jar 100

从hdfs读取 /usr/local/src/hadoop/bin/hdfs dfs -ls /

hadoop dfsadmin -safemode leave

/usr/local/src/hadoop/bin/hdfs dfs -mkdir /wc

/usr/local/src/hadoop/bin/hdfs dfs -put word.txt /wc/1.log

cd /usr/local/src/spark/sbin ./start-all.sh

/usr/local/src/spark/bin/spark-shell --master spark://storage9:7077

Sc.textFile(“hdfs://storage9:9000/wc”).flatMap(\_.split(“ “)).map((\_,1)).reduceByKey(\_+\_).sortBy(\_.\_2,false).collect

/usr/local/src/spark/bin/spark-submit --master spark://storage9:7077 --class WordCount /usr/local/src/hello-1.0.jar hdfs://storage9:9000/wc

hdfs://storage9:9000/out1

cd /usr/local/src/spark/bin下 运行 ./spark-shell

val lines = sc.textFile("/usr/local/src/helloSpark")

lines.count()

启动sparksql

/usr/local/src/spark/bin/spark-shell --master spark://storage9:7077 --driver-class-path /usr/local/src/spark/lib/mysql-connector-java-5.1.35-bin.jar

sqlContext.sql("show databases").show

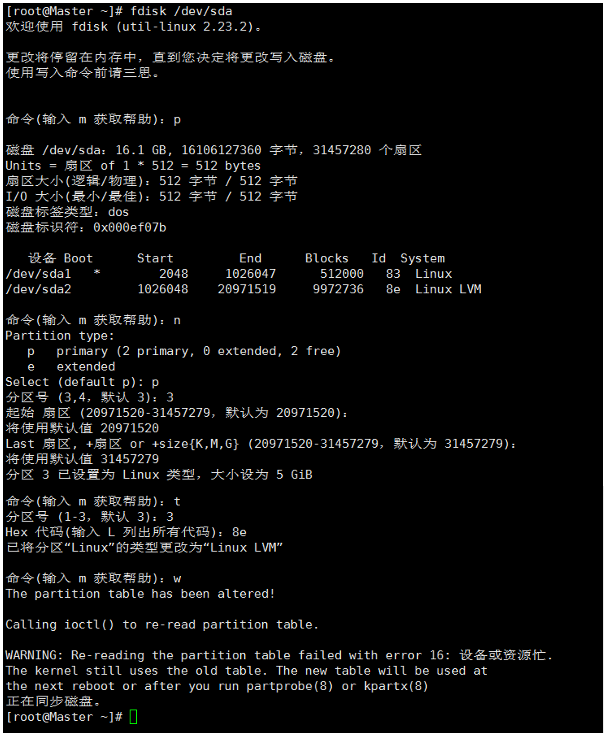
32, shell脚本(权限)

chmod a+x xxx.sh

139 /usr/local/src/test1.sh

33， 扩容

fdisk /dev/sda



重启reboot

mkfs.ext3 /dev/sda3 格式化

vgdisplay 查看

lvm

pvcreate /dev/sda3

vgextend cl /dev/sda3  //将初始化过的分区加入到虚拟卷组cl

vgdisplay -v

lvextend -l+100%FREE /dev/mapper/cl-root

pvdisplay   //查看卷容量

xfs\_growfs /dev/mapper/cl-root扩展容量。

du -sh /usr/local/src/spark/\* 查看文件大小

34，sqoop

查看表sqoop list-tables --username root --password '123456' --connect jdbc:mysql://192.168.181.139:3306/hive?characterEncoding=UTF-8

导入hdfs 。。sqoop import --username root --password '123456' --connect jdbc:mysql://192.168.181.139:3306/hive?characterEncoding=UTF-8 --table user\_movie -m 4

导入hive。。 sqoop import --username root --password '123456' --connect jdbc:mysql://192.168.181.139:3306/hive?characterEncoding=UTF-8 --table user\_movie -m 4 --hive-import

sqoop import --username root --password '123456' --connect jdbc:mysql://192.168.181.1:3306/itcast?characterEncoding=UTF-8 --table brand -m 4 --hive-import

导入hive（用sql）1

sqoop import --connect jdbc:mysql://192.168.181.1:3306/test?characterEncoding=UTF-8 --username root --password 123456 --query 'select \* from aaa where $CONDITIONS' --hive-table aaa --hive-import --target-dir /user/hive/warehouse/aaa --split-by orderid

导入hive（用sql）2

sqoop import --connect jdbc:mysql://192.168.181.1:3306/test?characterEncoding=UTF-8 --username root --password 123456 --query 'SELECT user\_id,bill\_id,order\_id,t1.pay\_type\_id,bill\_amt,created\_time,t2.pay\_type\_name from (SELECT \* from t\_user\_order\_bill)t1 join (SELECT \* from t\_payment)t2 on(t1.pay\_type\_id=t2.pay\_type\_id) where $CONDITIONS' --hive-table ods\_t\_user\_order\_bill --hive-import --target-dir /user/hive/warehouse/ods\_t\_user\_order\_bill --split-by user\_id

导出hive

sqoop export --username root --password '123456' --connect jdbc:mysql://192.168.181.1:3306/echart?characterEncoding=UTF-8 --table t\_user\_copy --export-dir /user/hive/warehouse/t\_user\_copy --input-fields-terminated-by '\001'

job：

sqoop create-hive-table --username root --password '123456' --connect jdbc:mysql://192.168.181.1:3306/itcast?characterEncoding=UTF-8 --table buyer

sqoop job --create jobname -- import --username root --password '123456' --connect jdbc:mysql://192.168.181.1:3306/itcast?characterEncoding=UTF-8 --table buyer\_copy --hive-import    --hive-table buyer --incremental lastmodified --check-column lastmodified --last-value '2018-08-04 00:00:00'

　　sqoop job --list　　　　　　　　　   列出所有的job

　　sqoop job --show jobname　　　　显示jobname的信息

　　sqoop job --delete jobname 　　　删除jobname

　　sqoop job --exec  jobname  　　　执行jobname

35， 图形界面

yum groupinstall "GNOME 桌面"

36，创建用户，授权

创建yxx用户组及yxx用户

groupadd yxx

useradd yxx -g yxx -p /usr/local/src/yxx

更改yxx文件夹及内部文件的所属用户及组为yxx: yxx

chown -R yxx:yxx /usr/local/src/yxx

elasticsearch为你elasticsearch的目录名称

解决方法：删除用户不用userdel yxx

改用userdel -rf yxx

elasticsearch 启动命令

切换到yxx用户再启动

su yxx#切换账户

cd /usr/local/src/yxx/elasticsearch-5.5.2/bin #进入你的elasticsearch目录下的bin目录

./elasticsearch

37: elasticsearch 坑

./bin/elasticsearch-plugin install <https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v5.5.2/elasticsearch-analysis-ik-5.5.2.zip>

启动su yxx

/usr/local/src/yxx/elasticsearch-5.5.2/bin/elasticsearch

如有报错，删除/usr/local/src/yxx/elasticsearch-5.5.2/logs/

访问： http://192.168.181.140: 9200/

启动header插件

cd /usr/local/src/yxx/elasticsearch-head-master

npm run start

访问： http://192.168.181.140: 9100/

//启动essql

//cd /usr/local/src/yxx/site-server

//node node-server.js &

//访问： <http://192.168.181.140:8088/>

Kill ： ps -ef|grep elasticsearch |grep -v grep|awk '{print $2}' |xargs kill -9

修改vi /etc/sysctl.conf配置文件，

cat /etc/sysctl.conf | grep vm.max\_map\_count

vm.max\_map\_count=262144

如果不存在则添加

echo "vm.max\_map\_count=262144" >>/etc/sysctl.conf

sysctl -p

3.2、修改/etc/profile

通过vi /etc/profile修改，在最后加入以下内容

ulimit -n 65536

ulimit -u 65536

source /etc/profile

38： flume

Flume.txt

39， pyspark

启动

/usr/local/src/spark/bin/pyspark --master spark://storage9:7077 --driver-class-path /usr/local/src/spark/lib/mysql-connector-java-5.1.35-bin.jar

/usr/local/src/spark/bin/spark-submit /usr/local/src/pylearn/sparkcontext.py

安装numpy

yum -y install blas blas-devel lapack-devel lapack

yum -y install seaborn scipy

yum -y install freetype freetype-devel libpng libpng-devel

yum -y install epel-release

yum -y install python-pip

yum -y install pytz

yum -y install python-dateutil

pip install --upgrade pip

40：ORACLE

授权：

cmd

sqlplus sys/密码  as sysdba -》sqlplus sys/123456  as sysdba

grant create any procedure to root;

grant execute any procedure to root;

GRANT CREATE SEQUENCE TO root;

创建序列

create sequence seq\_id

increment by 1

start with 1

maxvalue 999999999;

序列代码

@Id

@SequenceGenerator(name="gen",sequenceName="SEQ\_ID" ,allocationSize=1)

@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="gen")

@Column(name = "id", unique = true, nullable = false)

查看序列

select \* from user\_sequences;

mysql：

grant create routine on fcxxall.\* to root

grant alter routine on fcxxall.\* to root

grant execute on fcxxall.\* to root

远程连接

D:\app\yxx\product\11.2.0\dbhome\_1\NETWORK\ADMIN 找到 tnsnames.ora

(1) oracle用户登录系统。  
(2)sqlplus / as sysdba登陆数据库。  
(3)执行alter system set REMOTE\_LOGIN\_PASSWORDFILE=SHARED SCOPE=SPFILE;   
(4)shutdown immediate;  
(5)startup;

41：linux定时任务cron

编写脚本testcron.sh

编写定时任务crontab -u root -e

加入 \* \* \* \* \* root testcron.sh

重启service crond restart

42：hdp

安装注意：vim /etc/python/cert-verification.cfg 修改disable

启动 137 ambari-server start

133-137 ambari-agent start

访问192.168.181.137:8080

43：cdh

141启动/etc/init.d/cloudera-scm-server start

查看内存free -m

定时清理/user/hdfs/.Trash

142命令目录 比如hbase /opt/cloudera/parcels/CDH/lib/hbase/bin

或者直接比如hive

切换用户执行su - hdfs hive

或者设置hdfs权限检查

开启hive on spark

1 hive.execution.engine 2 sparkonyarn

44：hue

别忘记+file

Workflow 报错， 缺少用户组执行权限 hdfs dfs -chmod -R 777 /user

Oozie启动 hive服务 ，sparkonyarn

Sqoop

Shell调用 sqoop.sh

#!/bin/bash

sqoop --options-file sqoop.txt

sqoop.txt:

import

--username

root

--password

123456

--connect

jdbc:mysql://192.168.181.1:3306/test?characterEncoding=UTF-8

--table

ddd

--fields-terminated-by

','

--delete-target-dir

--hive-import

--hive-database

default

--hive-table

Ddd

45：脚本，坏的解释器

sed -i 's/\r$//' build.sh

46，压力测试

/bin/ab -r -c 2000 -n 10000 <http://192.168.181.1:8080/showXmch.action>

47，react

创建项目 cd e:/React 执行 create-react-app myapp

添加 cd e:/React/myapp

yarn add react-router-dom axios less less-loader

yarn eject 暴露

webpack.config.dev.js修改less配置（原来的搜索less），复制到webpack.config.prod.js

yarn add antd

yarn add babel-plugin-import

webpack.config.prod.js

修改['import', { libraryName: 'antd', style: true }

降级less yarn add [less@^2.7.3](mailto:less@%5e2.7.3)

yarn add jsonp --save

yarn add redux react-redux

yarn add react-loadable

yarn add react-transition-group

yarn add redux-immutable

yarn add redux-thunk

yarn add styled-components[@^3.5.0](mailto:less@%5e2.7.3)

yarn add axios

启动 cd e:/React/myapp yarn start

设置默认浏览器 open in broser 插件，setting设置下

48:css

Position固定

overFlow auto 超出滚动

iframe ， target 导航

display: inline-block; 同一行显示

hover（css） 鼠标悬浮导航 配合'mouseenter'（js事件）

$('.containor').on('mouseenter', **function**() {

$(".nav\_right").removeClass('hide');

vertical-align: middle;垂直居中

自动计算 ，对应的居中

@import './../../style/default.less';

.home-wrap{

background-color: @colorM;

height: calc(62vh);

display: flex;

align-items: center; 垂直居中

justify-content: center; 水平居中

font-size: 20px;

}

border-top: 9px solid @colorM;

border-left: 12px solid transparent; 实心 透明

border-right: 12px solid transparent;

49：rabbitmq

关闭防火墙

systemctl stop firewalld.service

启动：rabbitmq-plugins enable rabbitmq\_management

rabbitmq-server

报错解决：chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie

添加用户并授权

rabbitmqctl add\_user root 123456

rabbitmqctl set\_user\_tags root administrator

rabbitmqctl set\_permissions -p / root '.\*' '.\*' '.\*'

访问 http://192.168.181.140:15672/

50：springcloud 接口

调用方：

Applicaton配置

@EnableDiscoveryClient  
@EnableFeignClients

**application.properties 添加eureka.client.serviceUrl.defaultZone**=**http://localhost:8761/eureka/**

Control里调用**orderMsgOutService**.sendordermsgbyobj(orderAll);

Service 层

@FeignClient(value = **"youfanmsgqueueserv"**)//服务方的**spring.application.name**  
**public interface** OrderMsgOutService {  
 @RequestMapping(value = **"/sendordermsg"**,method = RequestMethod.***GET***)  
 **public void** sendordermsg(@RequestParam(value = **"msg"**) String msg);  
 @RequestMapping(value = **"/sendordermsgbyobj"**,method = RequestMethod.***POST***)  
 **public void** sendordermsgbyobj(@RequestBody OrderAll orderAll);  
}

服务方：

Applicaton配置

@EnableEurekaClient  
@EnableDiscoveryClient

@RestController  
**public class** OrderMsgOutControl {  
 @Autowired  
 **private** OrderSender **orderSender**;  
 @RequestMapping(value = **"/sendordermsg"**,method = RequestMethod.***GET***)  
 **public void** sendordermsg(@RequestParam String msg) {  
 **orderSender**.send(msg);  
 }  
 @RequestMapping(value = **"/sendordermsgbyobj"**,method = RequestMethod.***POST***)  
 **public void** sendordermsgbyobj(@RequestBody OrderAll orderAll) {  
 **orderSender**.send(orderAll);  
 }  
}

51: Hystrix

调用方再加一层

**package** com.youfan.hystrix;

*….*@Service  
**public class** ProductTypeServicehy {  
 @Autowired  
 ProductTypeService **productTypeService** ;  
 @HystrixCommand(fallbackMethod = **"listproductTypeFallback"**)  
 **public** List<ProductType> listproductType(){  
 **return productTypeService**.listproductType();  
 }  
 **public** List<ProductType> listproductTypeFallback(){  
 ProductType productType = **new** ProductType();  
 productType.setId(1);  
 productType.setParentid(-1);  
 productType.setProducttypedescription(**"吃的"**);  
 productType.setProducttypename(**"食品18"**);  
 productType.setTypegrade(**"1"**);  
 List<ProductType> list = **new** ArrayList<ProductType>();  
 list.add(productType);  
 **return** list;  
 }  
}

然后 在control里调用

@RequestMapping(value = **"/indexproduct"**,method = RequestMethod.***GET***)  
**public** String sayHi(Model model , @RequestParam **int** productytpeid){  
 List<ProductType> list = **productTypeServicehy**.listproductType();  
 model.addAttribute(**"productypelist"**,list);  
 System.***out***.println(list);

52：ribbon

新建服务作为中间层；**youfanrobbinserver**

**设置spring.application.name**=**youfanrobbinserver**

**代码**

**package** com.youfan.control;  
*…*@RestController  
**public class** ProducttypoutControl {  
 */\*\*  
 \* 注入RestTemplate  
 \*/* @Autowired  
 RestTemplate **restTemplate**;  
 @RequestMapping(value = **"/listproducttype"**,method = RequestMethod.***GET***)  
 **public** List<ProductType> listproducttype(){  
 String url=**"http://YOUFANPRODUCTTYPE/listproducttype"**;

// **YOUFANPRODUCTTYPE为原来的服务层spring.application.name**  
 List<ProductType> list = **restTemplate**.getForObject(url,List.**class**);  
 System.***out***.print(**"robbin--------------------------"**);  
 **return** list;  
 }  
}

**调用方 修改 ：**

@FeignClient(value = **"youfanrobbinserver"**)//改为 ribbon服务的 **spring.application.name**  
**public interface** ProductTypeService {  
 @RequestMapping(value = **"/listproducttype"**,method = RequestMethod.***GET***)  
 List<ProductType> listproductType();  
}

53:git

* 1. **git config –-global user.name ‘yxx’**
  2. **git config –-global user.email ‘790230828@qq.com’**
  3. **ssh-keygen -t rsa -C '790230828@qq.com'**

Git log操作日志

Git reset –hard ….

git push -f mypro master 强制（回滚用）

git status 提示

git init 初始化本地仓库

git add t2.txt

git config --list查看

git commit -m ‘add t2.txt’

git remote add mypro git@github.com:yingxx26/t2.git 建立关联

git remote 查看

git remote rm rigin删除关联

git push -u mypro master 发远端

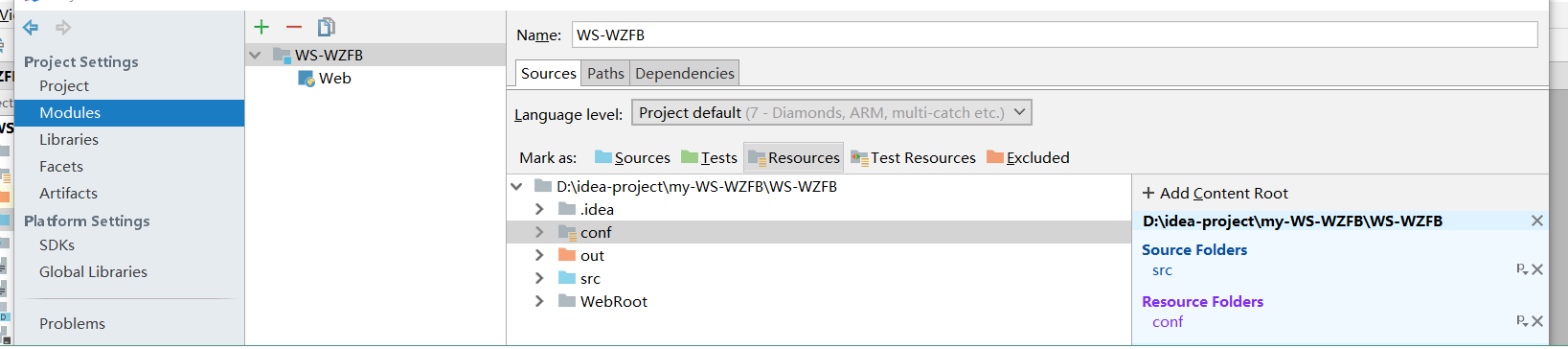
git fetch 查看远端

git pull mypro master拉的更新

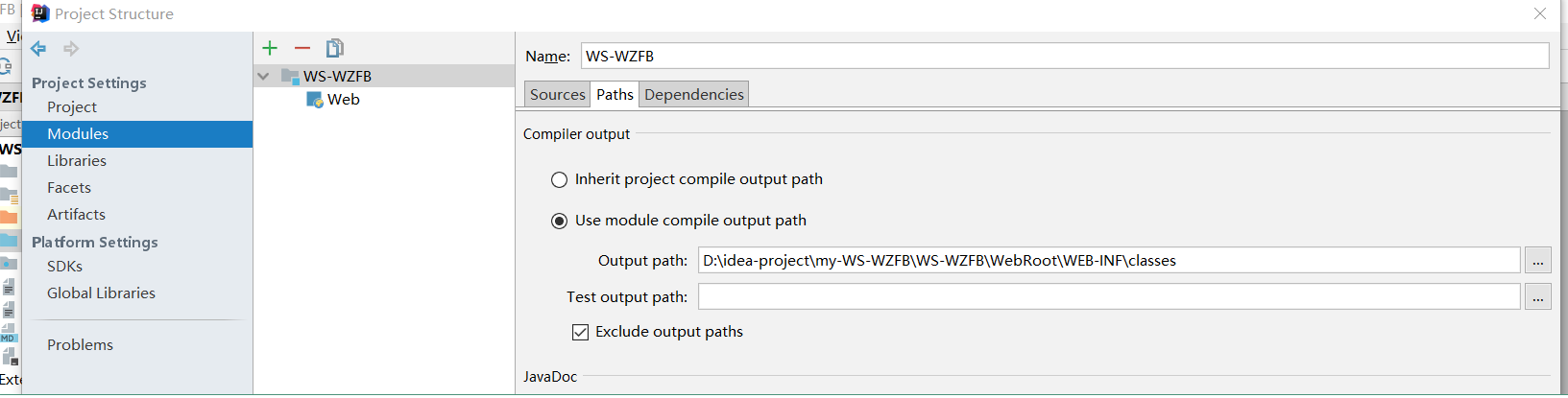
git checkout

54:idea

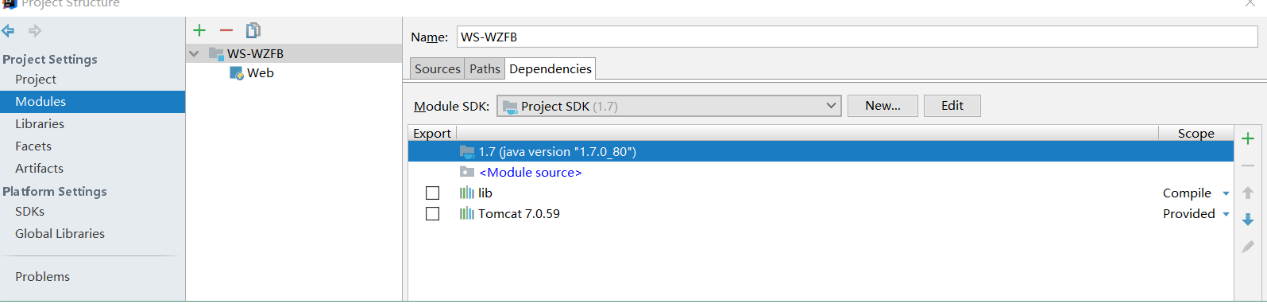
设置目录属性



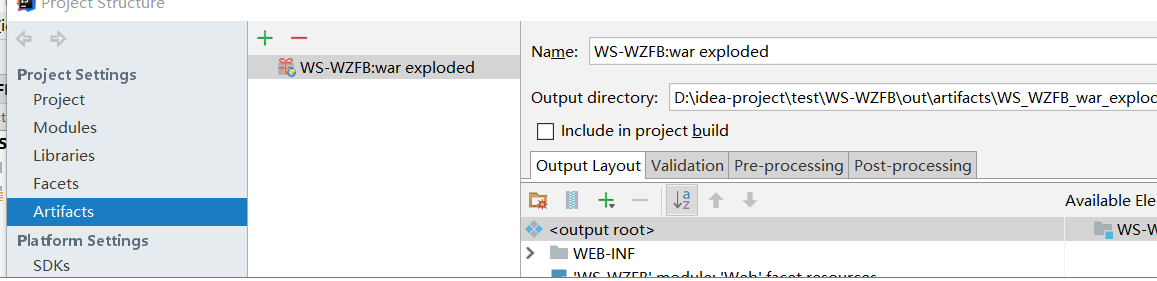
设置编译路径



设置依赖包



添加artifacts



快捷键：

Sout ： System.out.println();

Ctrl+ alt+v : 补全返回值

Psvm ： Main函数

Ctrl+alt+m ： 封装方法

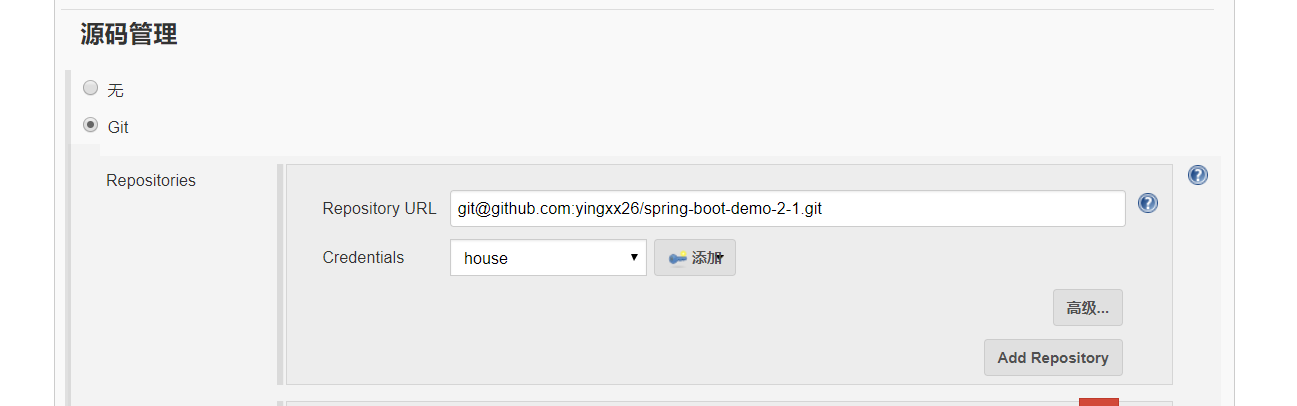
Ctrl+shift+T ：test测试类

@RunWith(SpringRunner.**class**)  
@SpringBootTest(classes = LyCartApplication.**class**)  
**public class** RedisTest {  
  
 @Autowired  
 **private** StringRedisTemplate **stringRedisTemplate**;  
  
 @Test  
 **public void** test(){  
 BoundHashOperations<String,Object,Object> hashOperations = **this**.**stringRedisTemplate**.boundHashOps(**"test"**);  
 hashOperations.put(**"3"**,**"4"**);  
 System.***out***.println(**"获取设置的绑定key值:"** + hashOperations.getKey());  
 }  
}

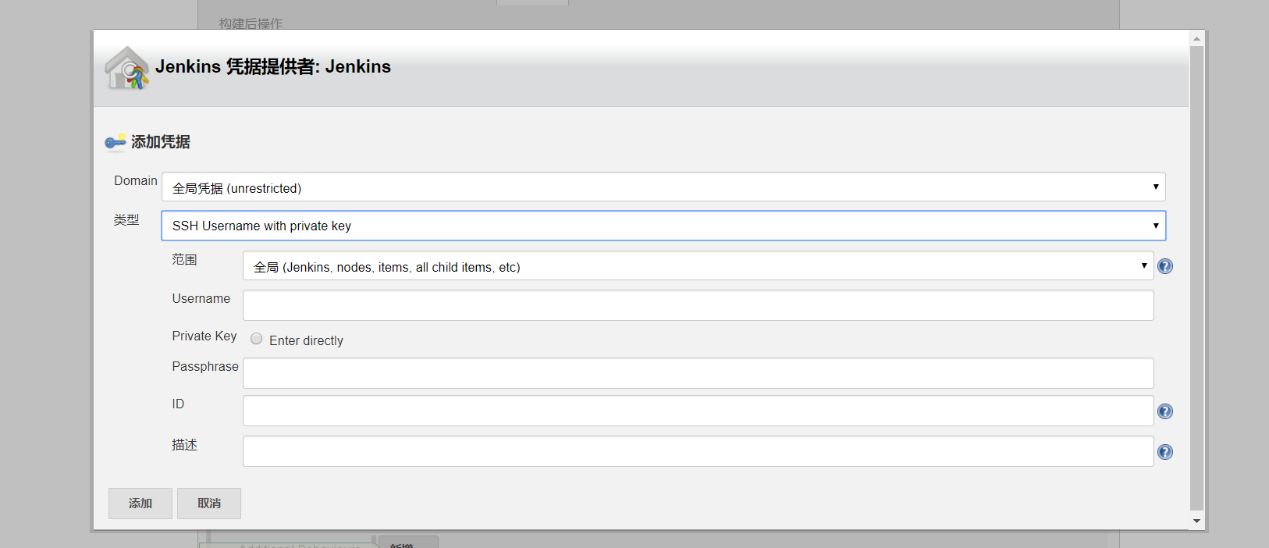
1. Ctrl+Alt+L，格式化代码
2. Ctrl+Alt+I，将选中的代码进行自动缩进编排，这个功能在编辑 JSP 文件时也可以工作
3. Ctrl+Alt+O，优化导入的类和包

55 : Jenkins

添加证书



选择私钥，用户名自定义



56 : github博客

用markdown编写.md文件，

保存到F:\Program Files\Git-blog\Hexo\source\\_posts

在F:\Program Files\Git-blog\Hexo目录右键 选择git bash here

输入指令hexo d -g

刷新 页面 <https://yingxx26.github.io/>

更换主题

<https://hexo.io/themes/> 选中一个 复制网址

在F:\Program Files\Git-blog\Hexo\themes目录下 git clone 网址

修改F:\Program Files\Git-blog\Hexo\\_config.yml 的theme:

hexo g

hexo s

hexo clean

hexo g -d

57 : vue

调试，代码加入debugger，然后在浏览器f12

vue ui

npm install 安装依赖

npm run dev或者 npm run serve 启动

main.js入口-> app.vue（不是必须）->router.js(通过app.vue里面的name找到router.js里的path，找到对应组件)

menu-》router-》pages

页面结构

<template> </template>

<script></script>

<style></style>

58：乐优商城

1，Fegin使用：

调用方 实现 服务方的接口

调用方

@FeignClient(value = **"item-service"**) 服务方的服务名  
**public interface** CategoryClient **extends** CategoryApi {  
}

调用方的service层注入

@Service  
**public class** GoodsServiceImpl **implements** GoodsService {  
 @Autowired  
 **private** GoodsClient **goodsClient**;  
 @Autowired  
 **private** BrandClient **brandClient**;

服务方 暴露 和自己controller层同样的方法和参数

@RequestMapping(**"category"**)  
**public interface** CategoryApi {  
@GetMapping(**"names"**)  
 ResponseEntity<List<String>> queryNameByIds(@RequestParam(**"ids"**)List<Long> ids);   
 @GetMapping(**"all"**)  
 ResponseEntity<List<Category>> queryCategoryByIds(@RequestParam(**"ids"**)List<Long> ids);  
}

2，fegin的ribbon和hystrix使用





3,启动 portal前台

live-server --port=9002

用户 **username1**

密码 **abcdefg1**

4，若是访问不了应该是防火墙问题

59，权限系统

Apollo

F:\apollo

右键git

访问<http://localhost:8070> 输入用户名apollo，密码admin后登录

60 权限功能

在application.yml中配置参数

auth:   tokenValiditySeconds: 1200  #token存储到redis的过期时间   clientId: XcWebApp   clientSecret: XcWebApp   cookieDomain: localhost   cookieMaxAge: ‐1

3.3.4.2 资源服务授权配置

基本上所有微服务都是资源服务，这里我们在课程管理服务上配置授权控制，当配置了授权控制后如要访问课程信 息则必须提供令牌。

1、配置公钥

认证服务生成令牌采用非对称加密算法，认证服务采用私钥加密生成令牌，对外向资源服务提供公钥，资源服务使 用公钥 来校验令牌的合法性。 将公钥拷贝到 publickey.txt文件中，将此文件拷贝到资源服务工程的classpath下

2、添加依赖

4、在conﬁg包下创建ResourceServerConﬁg类：

<dependency> <groupId>org.springframework.cloud</groupId>      <artifactId>spring‐cloud‐starter‐oauth2</artifactId>      </dependency>

@Configuration

@EnableResourceServer @EnableGlobalMethodSecurity(prePostEnabled = true, securedEnabled = true)

//激活方法上的 PreAuthorize注解 public class ResourceServerConfig extends ResourceServerConfigurerAdapter {       //公钥     private static final String PUBLIC\_KEY = "publickey.txt";

北京市昌平区建材城西路金燕龙办公楼一层 电话：400-618-9090

  3.3.4.3 资源服务授权测试

这里我们测试课程图片查询

get http://localhost:31200/course/coursepic/list/4028e58161bd3b380161bd3bcd2f0000

请求时没有携带令牌则报错：

      //定义JwtTokenStore，使用jwt令牌     @Bean     public TokenStore tokenStore(JwtAccessTokenConverter jwtAccessTokenConverter) {         return new JwtTokenStore(jwtAccessTokenConverter);     }       //定义JJwtAccessTokenConverter，使用jwt令牌     @Bean     public JwtAccessTokenConverter jwtAccessTokenConverter() {         JwtAccessTokenConverter converter = new JwtAccessTokenConverter();         converter.setVerifierKey(getPubKey());         return converter;     }     /\*\*      \* 获取非对称加密公钥 Key      \* @return 公钥 Key      \*/     private String getPubKey() {         Resource resource = new ClassPathResource(PUBLIC\_KEY);         try {             InputStreamReader inputStreamReader = new  InputStreamReader(resource.getInputStream());             BufferedReader br = new BufferedReader(inputStreamReader);             return br.lines().collect(Collectors.joining("\n"));         } catch (IOException ioe) {             return null;         }     }     //Http安全配置，对每个到达系统的http请求链接进行校验     @Override     public void configure(HttpSecurity http) throws Exception {                 //所有请求必须认证通过                 http.authorizeRequests().anyRequest().authenticated();     } }

修改授权配置类ResourceServerConﬁg的conﬁgure方法：

针对swagger-ui的请求路径进行放行：

 @Override public void configure(HttpSecurity http) throws Exception {     //所有请求必须认证通过     http.authorizeRequests()             //下边的路径放行     .antMatchers("/v2/api‐docs", "/swagger‐resources/configuration/ui",             "/swagger‐resources","/swagger‐resources/configuration/security",             "/swagger‐ui.html","/webjars/\*\*").permitAll()     .anyRequest().authenticated(); }

注意：

通过上边的配置虽然可以访问swagger-ui，但是无法进行单元测试，除非去掉认证的配置或在上边配置中添加所有 请求均放行（"/\*\*"）。

61 微服务之间进行调用时需携带JWT。

定义拦截器

package com.xuecheng.framework.interceptor;

public class FeignClientInterceptor implements RequestInterceptor {

@Override     public void apply(RequestTemplate requestTemplate) {

 try {             //使用RequestContextHolder工具获取request相关变量             ServletRequestAttributes attributes = (ServletRequestAttributes)  RequestContextHolder.getRequestAttributes();             if(attributes!=null){                 //取出request                 HttpServletRequest request = attributes.getRequest();                 Enumeration<String> headerNames = request.getHeaderNames();                 if (headerNames != null) {                     while (headerNames.hasMoreElements()) {                         String name = headerNames.nextElement();

 String values = request.getHeader(name);

 if(name.equals("authorization")){                             //System.out.println("name="+name+"values="+values);                             requestTemplate.header(name, values);                         }                       }                 }             }           }catch (Exception e) {             e.printStackTrace();         }     }

4.2.2 使用Feign拦截器

本例子中课程管理调用cms需要携带jwt，所以需要在课程管理中定义Feign拦截器bean，在启动类中定义bean如 下：

@Bean  public FeignClientInterceptor feignClientInterceptor(){      return new FeignClientInterceptor();  }

62 elasticsearch6

su yxx

/usr/local/src/yxx/elasticsearch6/elasticsearch-6.4.2/bin/elasticsearch

http://192.168.181.140: 9200/

cd /usr/local/src/yxx/elasticsearch-head-master

npm run start

访问： http://192.168.181.140: 9100/

63 上传jar到 maven

mvn install:install-file -Dfile=G:\x-pack-sql-jdbc-6.4.0.jar -DgroupId=org.elasticsearch.plugin -DartifactId=x-pack-sql-jdbc -Dversion=6.4.0 -Dpackaging=jar

64 Activiti6.0

/usr/local/src/apache-tomcat-8.0.50/bin/startup.sh

tail -f /usr/local/src/apache-tomcat-8.0.50/logs/catalina.out

192.168.181.140:8080/activiti-app