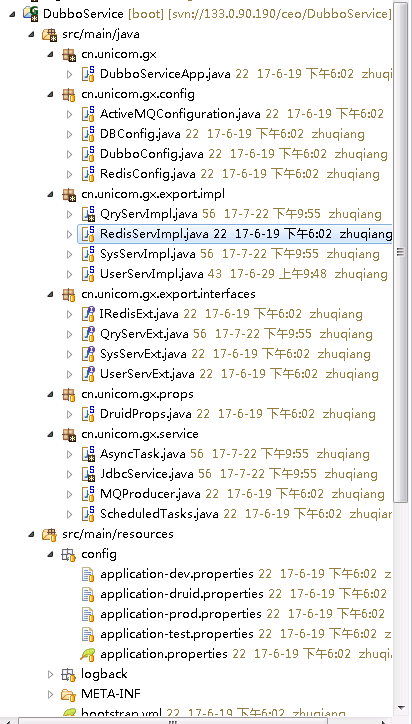
小CEO后台程序说明

1. 位置
2. 在4A环境，远程桌面登录133.0.77.60。D:\workspace-sts\DubboService
3.  使用STS打开工程
4. 程序结构

1、

2、cn.unicom.gx. DubboServiceApp 主启动服务

3、cn.unicom.gx.config.\* 数据库、dubbo、Redis、MQ的配置

4、cn.unicom.gx.export.interfaces 接口定义（详见接口文档）

5、cn.unicom.gx.export.impl 接口实现类（详见接口文档）

6、cn.unicom.gx.props DruidDataSource 配置

7、cn.unicom.gx.service 服务类，包括数据同步、异步抽取、定时服、MQ订阅生产者

8、src/main/resoures/application-prod.properties 数据源、redis、MQ等配置信息

9、META-INF dubbo配置文件

1. 编译、上传、加载服务

1、编译命令

gradle build -x test

gradle release

gradle apiJar

gradle publishToMavenLocal

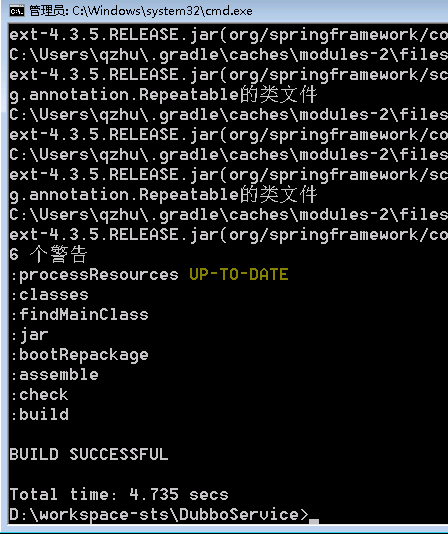
gradle copyJars

gradle copyProps

gradle copyScripts

2、复制以上编译命令，打开windows CMD 窗口，D: cd \workspace-sts\DubboService

粘贴执行。



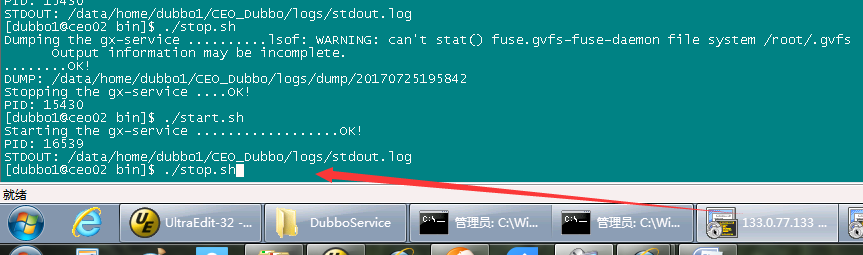
表示成功。Jar包生成在 D:\workspace-sts\DubboService\build\libs

1. 上传

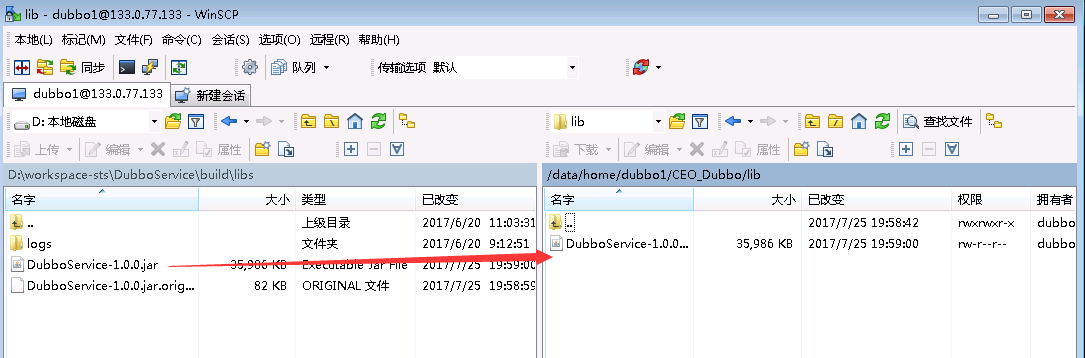
先停掉服务

dubbo1 登录 133.0.77.133 cd/data/home/dubbo1/CEO\_Dubbo/bin

执行./stop.sh



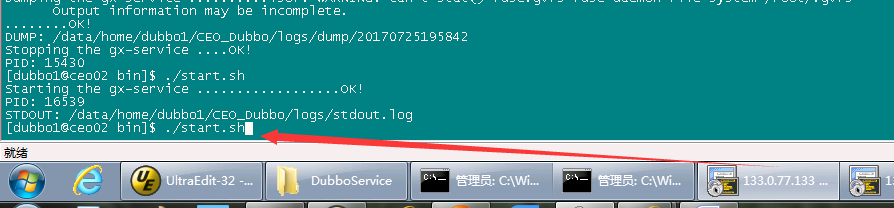
1. 使用把生产的jar包上传到目录 /data/home/dubbo1/CEO\_Dubbo/lib



1. 启动加载服务

dubbo1 登录 133.0.77.133 cd/data/home/dubbo1/CEO\_Dubbo/bin

执行./start.sh



查看日志：

dubbo1 登录 133.0.77.133 cd/data/home/dubbo1/CEO\_Dubbo/ logs

[dubbo1@ceo02 logs]$ pwd

/data/home/dubbo1/CEO\_Dubbo/logs

[dubbo1@ceo02 logs]$ tail -f stdout.log

查看controller层日志

Tail –f /data/home/taptest/dubbo/cloud/cloud20888/logs/cloud.log

1. 设计思路
2. 接口类 QryServImpl 负责为WEB接口提供调用，通过Redis或数据库表获取已经统计好的数据。
3. JdbcService ，AsyncTask负责数据抽取和统计，同步放在JdbcService，异步放在AsyncTask（有些数据处理比较费时的也放在这里）。
4. ScheduledTasks 负责定时启动数据抽取，调用jdbcService.updateData()。每分钟跑一次。
5. updateData 下包含所有需要跑的函数，函数一般的逻辑是：先看对方接口表的max(updatetime)字段是否有更新，没有updatetime字段的表，比如有些用户表，可以用sum(mobie\_phone)这种方式来判断，判断的方法是和上一次存在redis里面的datestamp\* 的内容对比，有更新再抽取。
6. 举例1：2I 同步发展人 用户表

updateData() 同步调用update\_2I()；update\_2I() 调用update\_2I\_user()。

先判断源表是否有更新select sum(DEVICE\_NUMBER) as TH1 from dm.dm\_d\_rpt\_toi\_order\_list sum(DEVICE\_NUMBER) 放入变量 datestamp

从Redis取 datestampRedis = jedisService.get(1, "datestamp2IUser")，与 datestamp 对比。如果相同则没有更新，退出函数，不相同则继续

dm.dm\_d\_rpt\_toi\_order\_list 的字段转换成TB\_2I\_MEN 的字段 批量insert。

1. 举例2：update\_2I\_all() 统计全区2I 情况

开头和例1一样判断源表是否更新，不同的是 源表有max(UPDATE\_TIME)。

数据统计结果放在jsonData 里面。转成String后放入Redis和数据库：update\_report\_redis("2IMsg-1", jsonData.toString());

结果供QryServImpl 使用。

QryServImpl 类的 JSONObject qry2IMsg(JSONObject jsonObject) 函数是负责为2I报表提供数据的，当参数level = 1 的时候，就是取2I 全区情况 调用

jsonRrt = qryRpt(jsonObject, "2IMsg"); qryRpt 负责到Redis 里面找出接口所需的缓存，并封装成返回数据。

"2IMsg" 参数 是与数据处理后台约定的，当初放入Redis是"2IMsg-1" -1 表示全区范围，所以只有全区权限的人才能看到。所以qryRpt 里面会判断用户的区域属性来匹配所需的缓存。如果是591的权限。那他只能取到"2Imsg591"

（该机制将来再考虑优化做成元数据表来控制。目前没有表配置）

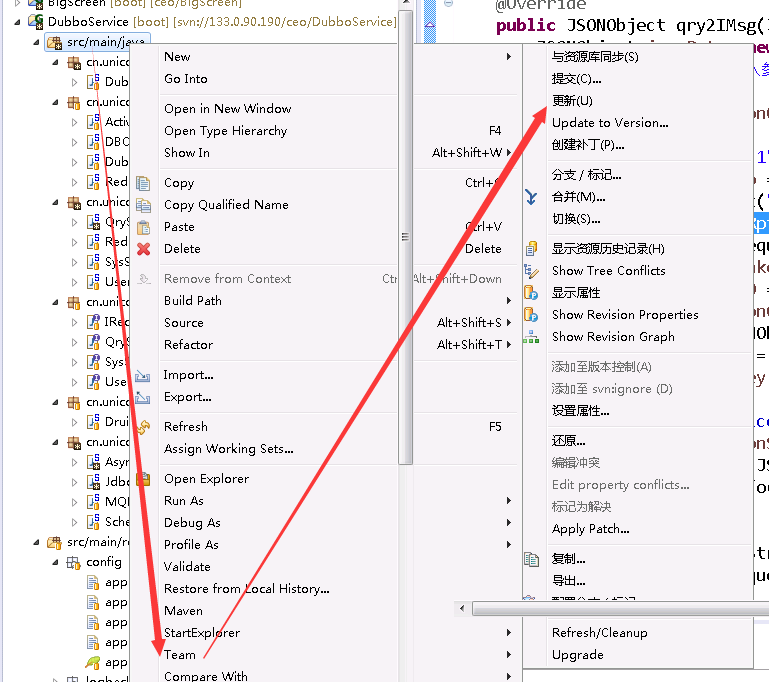
与该机制对应的还有：

菜单表：tb\_menucfg ('menu2IMsg','2I码上购订单转化情况','-1',2,1,1); 是在"2IMsg"前面加上menu

用户动作表：tb\_code\_actions values ('2IMsg-1','2I码上购订单转化情况-全区'); 这个当用户调用这个接口的时候会按这个actions记录到日志表 tb\_log\_user。

1. SVN管理

有较大改动、或时间间隔有一定时间后，需要及时保存到svn

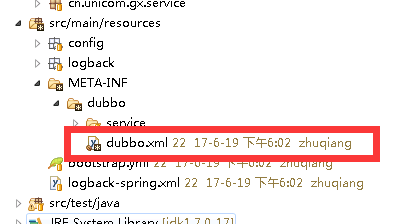


需要注意的是：一般在编辑修改前点“更新”，确保没有其他人对SVN进行改动，修改完后点提交。

提交的备注，一般有大版本改动，最好有备注说明。多人改动签上名字。

1. dubbo端口配置（新增dubbo服务时用）

修改dubbo.xml <dubbo:protocol port="${dubbo.protocol.port:26897}"/>



1. Tomcat 服务端口配置（新增dubbo服务时用）

修改application.properties server.port=8081

