java远程debug的使用方法，避免本地启动的程序调试的耗时方法

发件人：杨悦<yangyue@mogutrip.cn>
时　间：2017年1月10日(星期二) 下午2:47
收件人：技术组<tech@mogutrip.cn>

Hi, ALL

开发过程中，我们大部分问题都能通过日志解决，但是有些特别细小的问题，不容易通过日志发现。
现在我们大部分人的处理这种情况的方法有2种：<1>临时增加大量日志进行观察，找出bug再删掉日志。<2>本地启动程序，本地debug
这两种方法都有不少的缺点，日志增加后还要删除，本地debug在项目比较大的时候启动耗时很长等等

根据 @张建功 同学的发现，java提供一种远程debug的方法（程序在服务器上跑，断点在本地IDEA上打，很方便）我研究了一下我们的框架，发现远程debug其实已经配置好了。
现在介绍一下使用说明，如果有不对的地方请大家指出，如果大家有别的更好的方法排查棘手问题，也请回个邮件，大家探讨下。

我们使用远程debug需要遵循如下规则，为了防止漏看，我就写在前面了
远程debug的时候，如果有client连接进行debug，这种情况下会阻塞别人的请求。
<1>prd环境绝对禁止使用远程debug
<2>不要频繁使用远程debug，简单的问题通过看log解决
<3>使用远程debug的时候，要在群里通知一下大家，避免别人的程序卡死，导致别人无目的的查问题。
<4>使用时长最好不要超过10分钟，用完了一定要记得关闭远程debug模式，不要影响其他人

remote debug使用流程
<1>将工程启动为debug模式
下图是我们工程的start.sh启动脚本(x-search,x-booking,x-update,x-blacklist,x-statistics已经确定使用的是这种架构，其余架构的工程也可以参考)
我们现在启动工程的时候用的是 "sh start.sh" 或者 "./start.sh"，这种时候启动命令没有额外的启动参数，自然JAVA_DEBUG_OPTS是空的，自然程序不会按照debug模式启动。

我们在使用启动脚本的时候可以这样使用 sh start.sh debug 或者 ./start.sh debug ,这样的话，JAVA_DEBUG_OPTS就有值了，在jvm启动的时候就会启动为debug模式了.
(我们的脚本还支持jmx，我没看，有兴趣的可以研究一下)

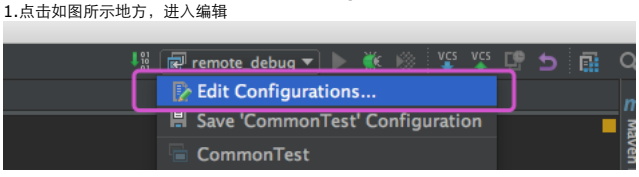
```
LIB_DIR=$DEPLOY_DIR/lib
LIB_JARS=`ls $LIB_DIR|grep .jar|awk '{print "'$LIB_DIR'/"$0}'|tr "\n" ":"`

JAVA_OPTS=" -Djava.awt.headless=true -Djava.net.preferIPv4Stack=true "
JAVA_DEBUG_OPTS=""
if [ "$1" = "debug" ]; then
    JAVA_DEBUG_OPTS=" -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n "
fi
JAVA_JMX_OPTS=""
if [ "$1" = "jmx" ]; then
    JAVA_JMX_OPTS=" -Dcom.sun.management.jmxremote.port=1099 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=fals
fi
JAVA_MEM_OPTS=""
BITS=`java -version 2>&1 | grep -i 64-bit`
if [ -n "$BITS" ]; then
    JAVA_MEM_OPTS=" -server -Xmx2g -Xms512m -Xmn256m -XX:PermSize=128m -Xss256k -XX:+DisableExplicitGC -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkE
else
    JAVA_MEM_OPTS=" -server -Xms1g -Xmx1g -XX:PermSize=128m -XX:SurvivorRatio=2 -XX:+UseParallelGC "
fi

echo -e "Starting the $SERVER_NAME ...\c"
nohup java $JAVA_OPTS $JAVA_MEM_OPTS $JAVA_DEBUG_OPTS $JAVA_JMX_OPTS -classpath $CONF_DIR:$LIB_JARS cn.mogutrip.data.update.shell.Program > $STDOUT_FI
```

启动参数介绍
-XDebug　　启用调试。
-Xnoagent　　禁用默认sun.tools.debug调试器。
-Djava.compiler=NONE　禁止 JIT 编译器的加载。
-Xrunjdwp　　加载JDWP的JPDA参考执行实例。
transport　　用于在调试程序和 VM 使用的进程之间通讯。
dt_socket　　套接字传输。
server=y/n　　VM 是否需要作为调试服务器执行。
address=8000　　调试服务器的端口号，客户端用来连接服务器的端口号。
suspend=y/n　　是否在调试客户端建立连接之后启动 VM 。

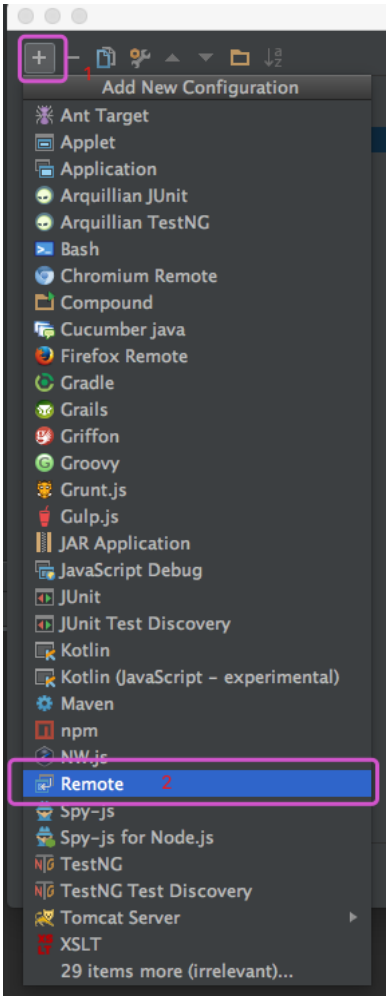
着重说明
address：我们同一个server可能部署了多个服务，他们的调试端口我估计默认的都是8000，同一台server同时debug多个项目时，需要临时修改start.sh中这个调试端口
suspend：如果设置为y，它会阻塞程序运行，直到有客户端连接到对应的监听端口(这里是8000)，程序才真正开始执行。我们有时候会抱怨程序一闪而过，还没来得及在本地加载上代码程序就执行完了，这种情况就可以使用suspend参数。我们一般情况下都是rest和dubbo服务，所以设置为n，不用等待客户端连接就启动jvm

貌似jdk1.7以上可以这么写-agentlib:jdwp=transport=dt_socket,address=9999,server=y,suspend=y
我没试过，有兴趣的可以试一下

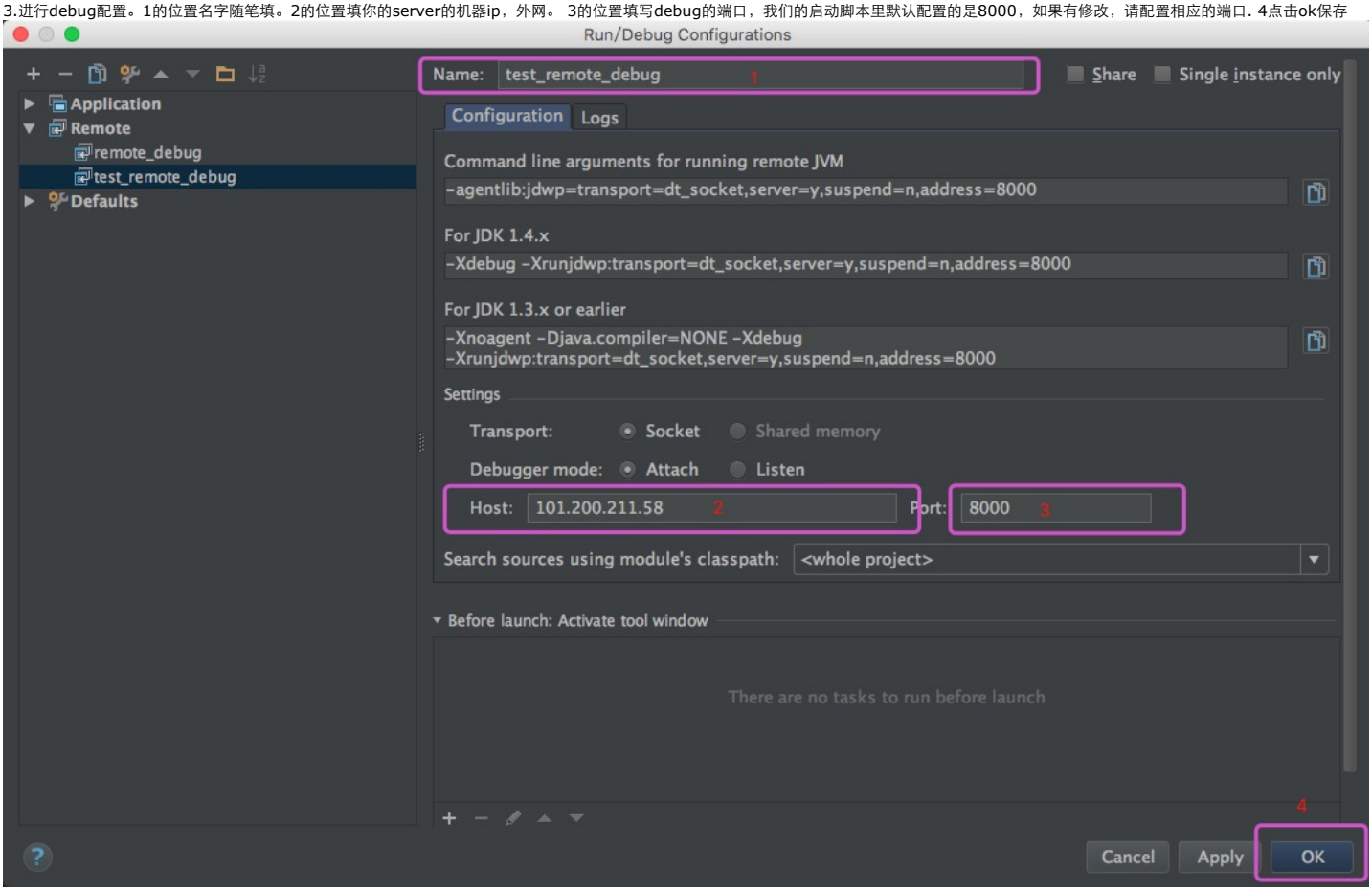
<2>本地使用IDEA来连接server进行远程debug
1.点击如图所示地方，进入编辑



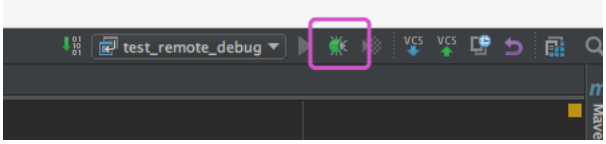
2.先点击"+"号，然后选择remote



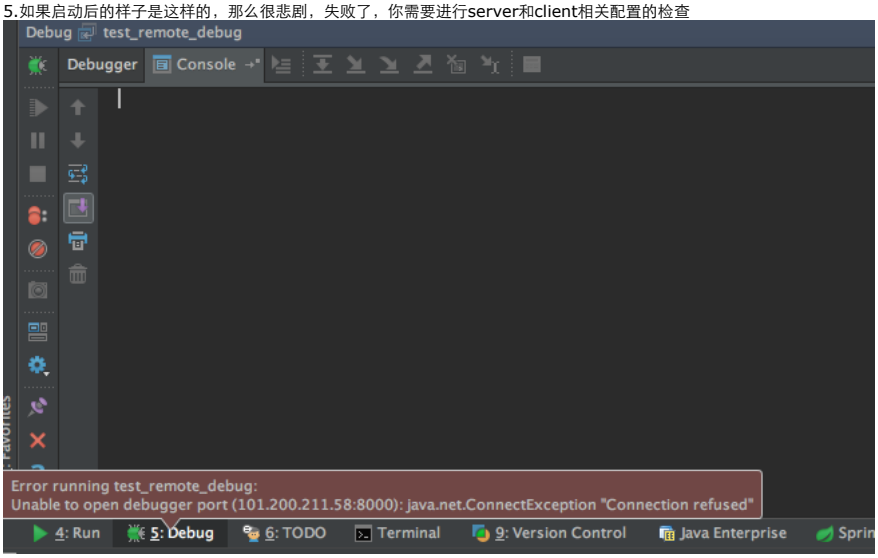
3.进行debug配置。1的位置名字随笔填。2的位置填你的server的机器ip，外网。 3的位置填写debug的端口，我们的启动脚本里默认配置的是8000，如果有修改，请配置相应的端口. 4点击ok保存



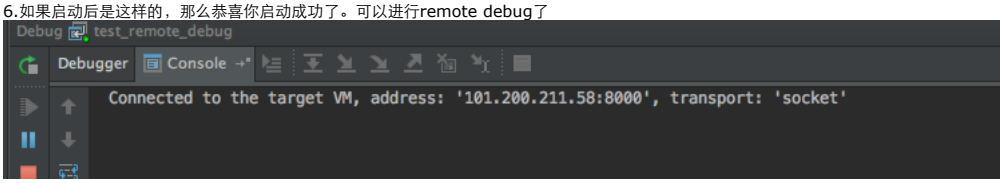
4.再下拉菜单选择我们刚才填的那个名字，就是那个远程debug的配置。点击图中所示的臭虫，进行debug



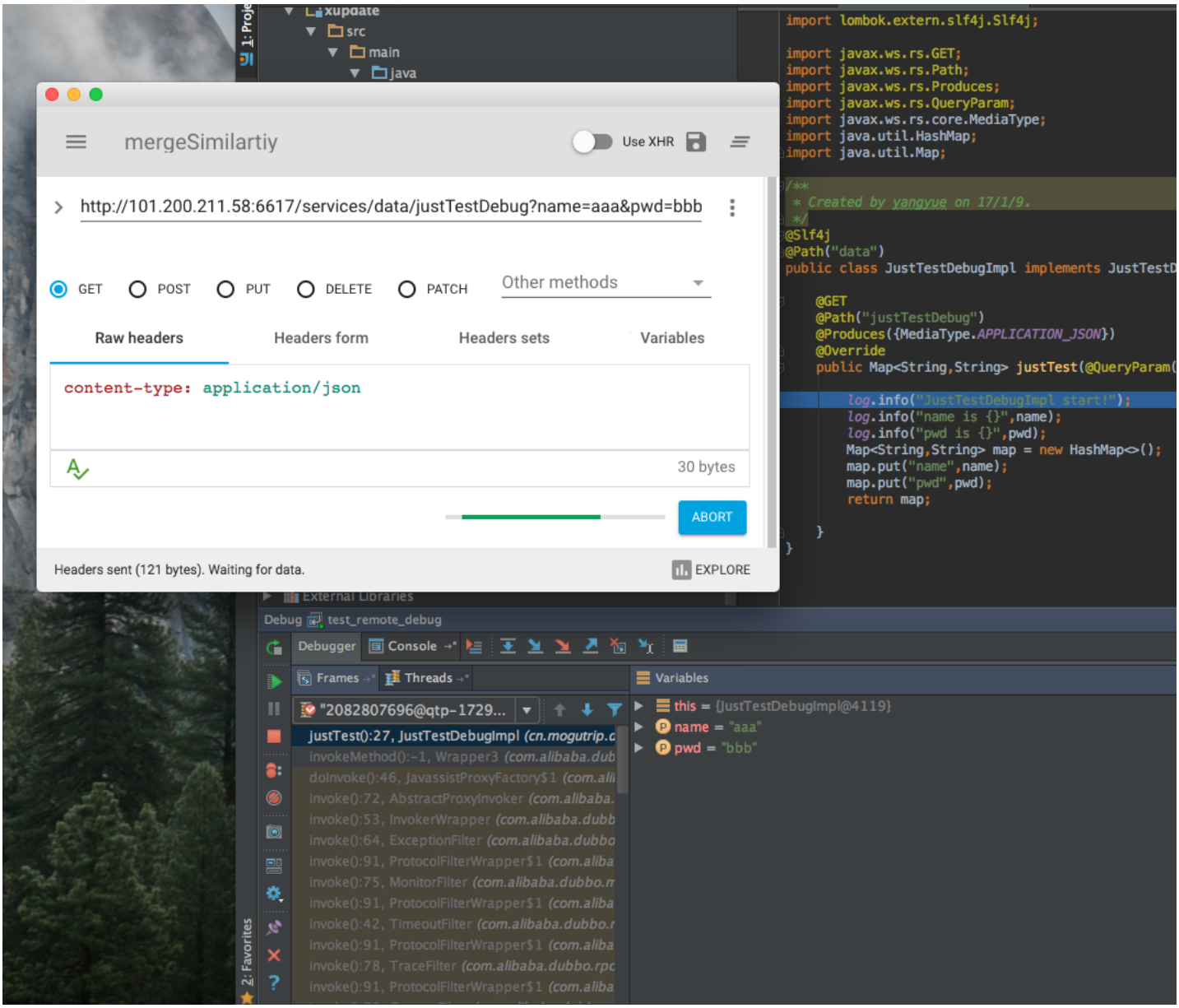
5.如果启动后的样子是这样的，那么很悲剧，失败了，你需要进行server和client相关配置的检查



6.如果启动后是这样的，那么恭喜你启动成功了。可以进行remote debug了



7.向服务器发送一个post请求，我们本地的idea已经成功拦截到请求了。可以再本地进行debug了。和本地debug一样，可以setvalue，执行自定义代码等等，本地debug操作技巧就不做介绍了



8.注意事项。
我们这种rest的请求，最后return回httpclient的时候，一定要点击下面的三角按钮，resume Program。如果连续用F8执行下一行，最后程序不会return到httpclient，并且程序会卡死，不接受新请求。