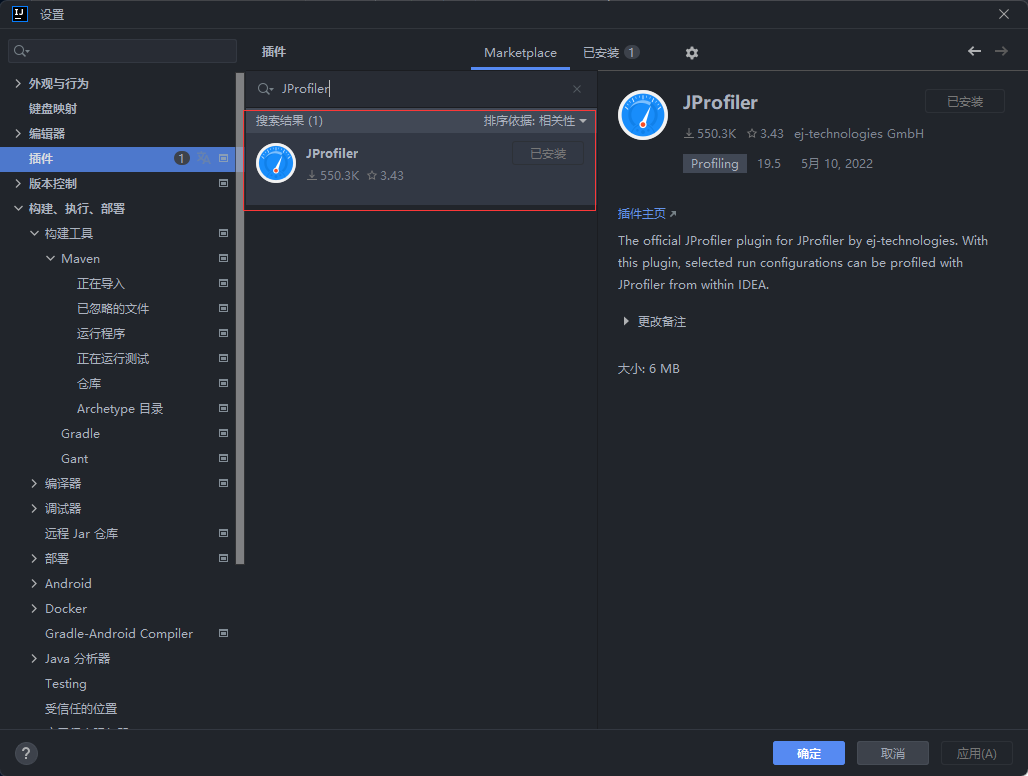
# 环境准备

## JProfiler下载和安装

下载地址:[JProfiler](https://www.ej-technologies.com/download/jprofiler/version_12)

## Idea插件下载(不下载也可以)

在插件市场搜索JProfiler即可。这里提到的不下载的原因是，可以直接通过桌面启动JProfiler，后通过在JProfiler中附加Java虚拟机实现对Java程序的监控

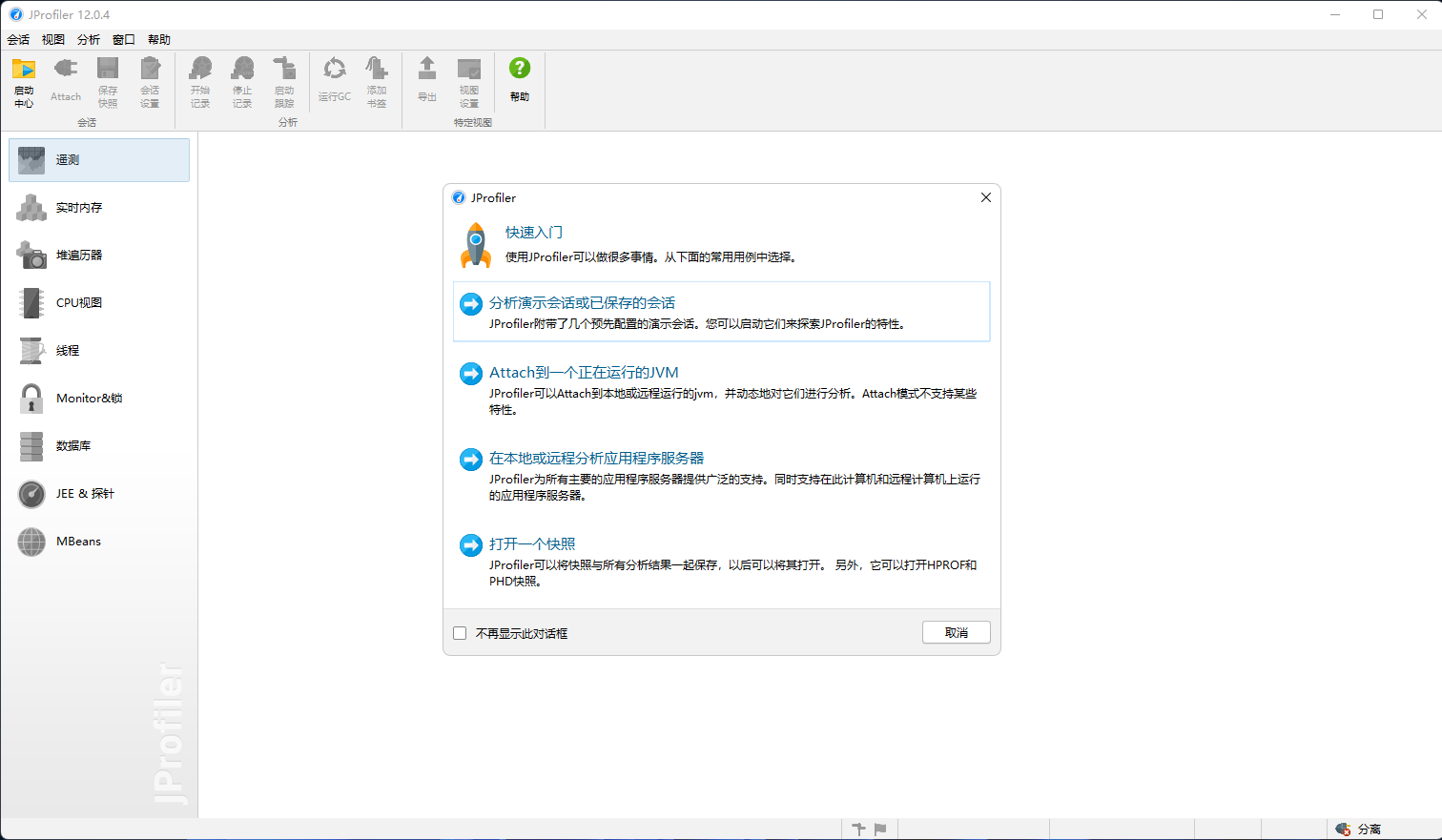


安装完成可能会遇到需要激活的的问题，大家先试用30天就行，可以尝试去网上找找激活码。

# JProfiler基础操作

附加虚拟机操作

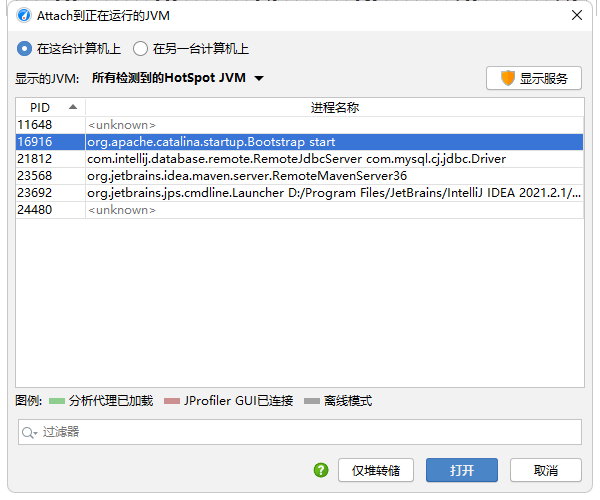
双击JProfiler图标：



选择第二即可



由于我们的项目是用tomcat启动的，所以我们选择进程名为org.apache.catalina.startup.Bootstrap start 就可以了。

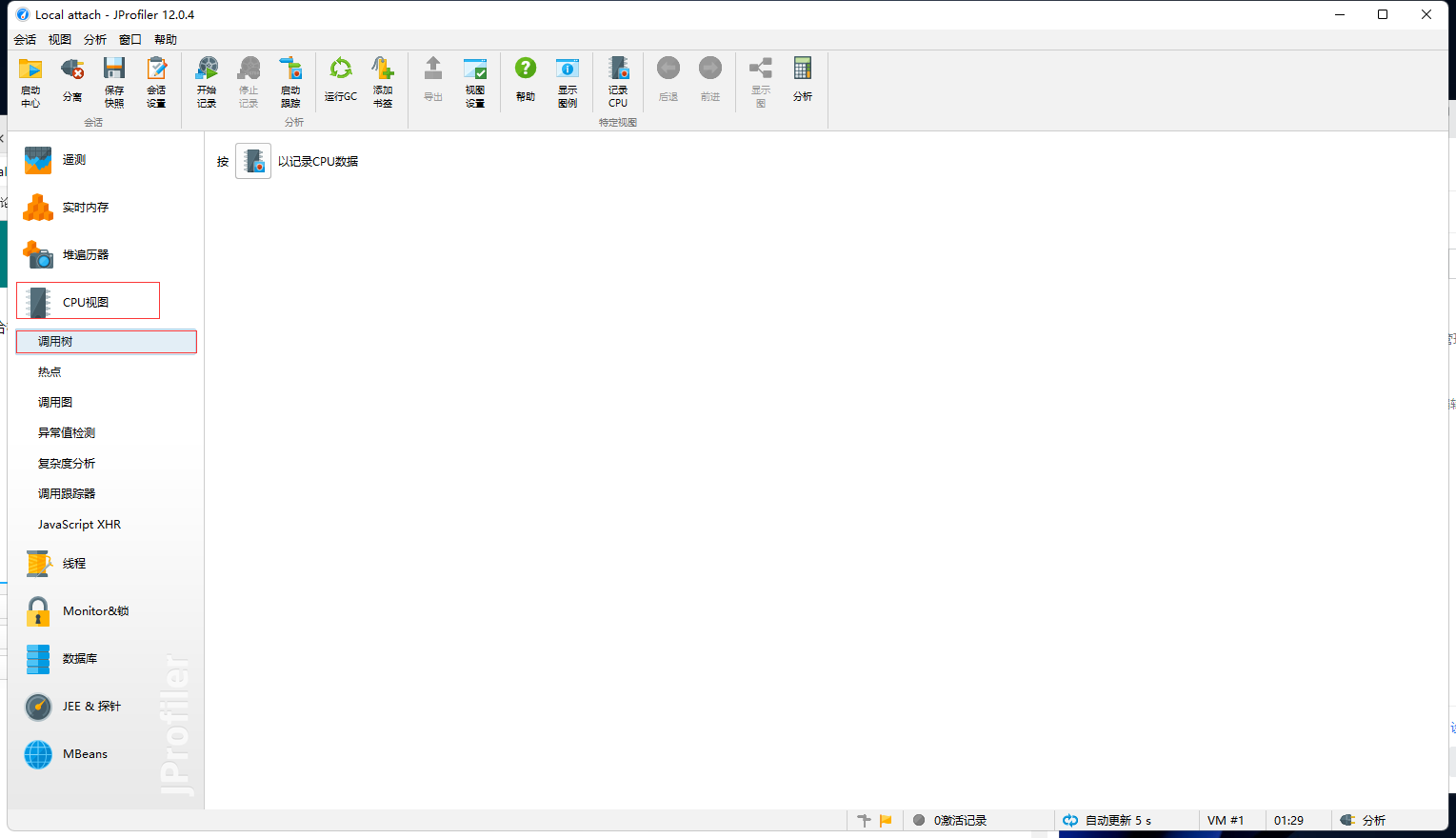


这时会弹出“如何记录方法调用？”选择推荐的即可（Recommended）

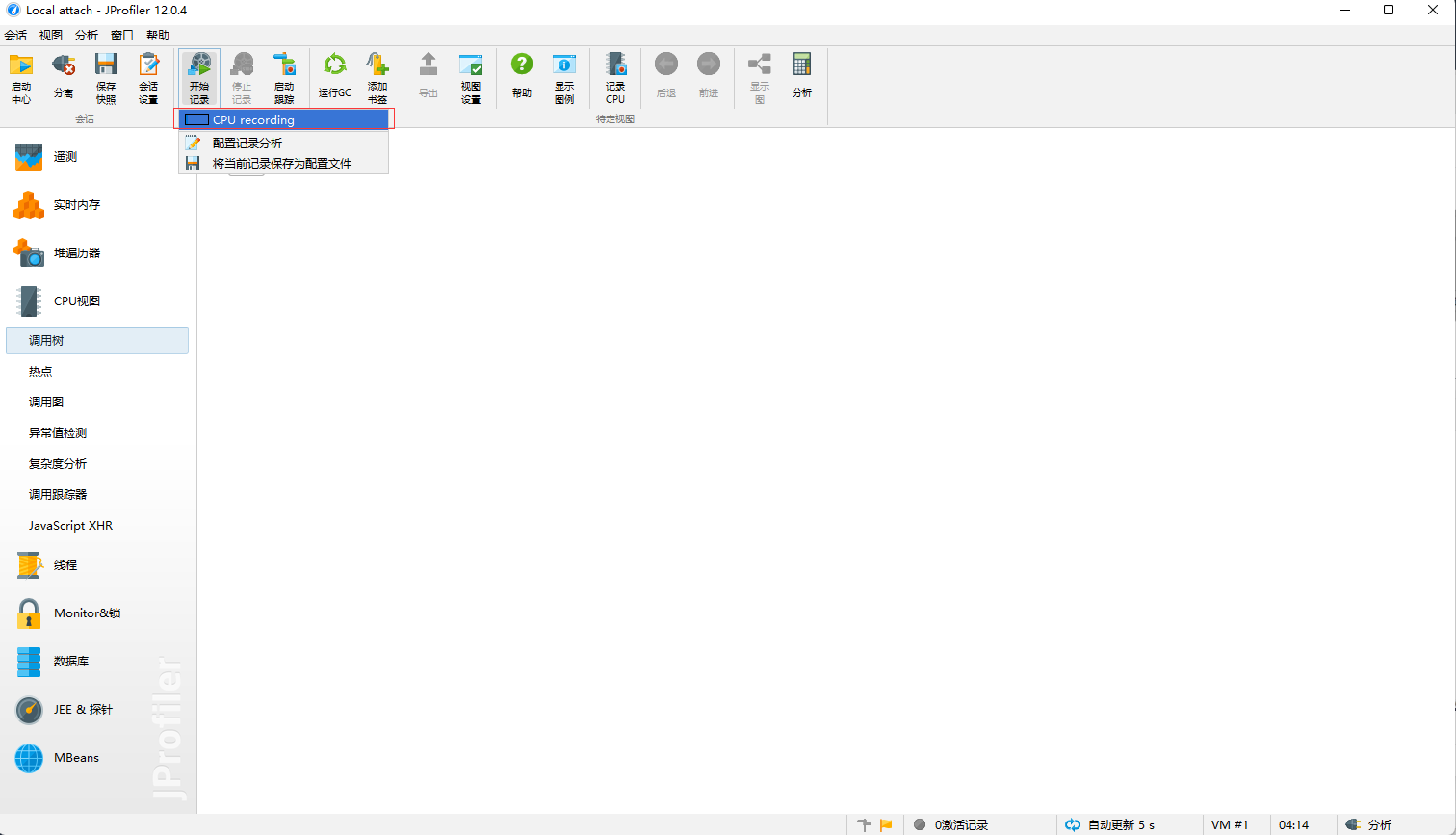


## 代码执行时间检测

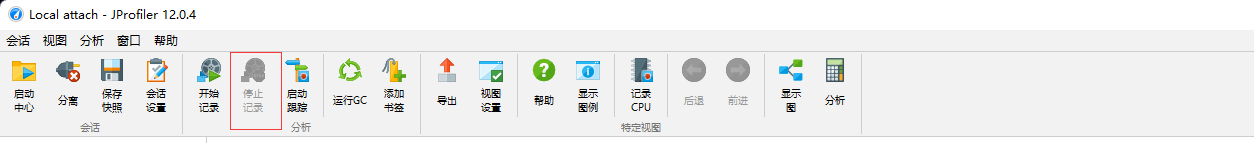
若要检测代码调用时间，请选在右边菜单中 的CPU视图->调用树



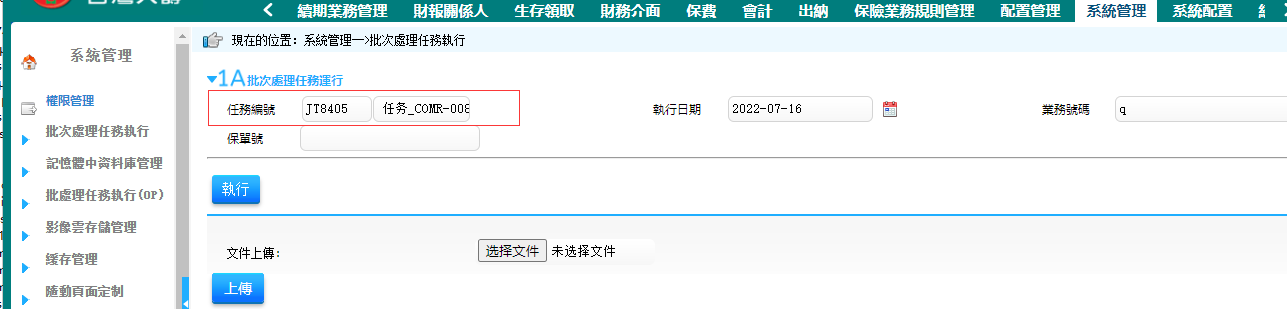
在记录方法用时前点击开始记录->CPU recoding

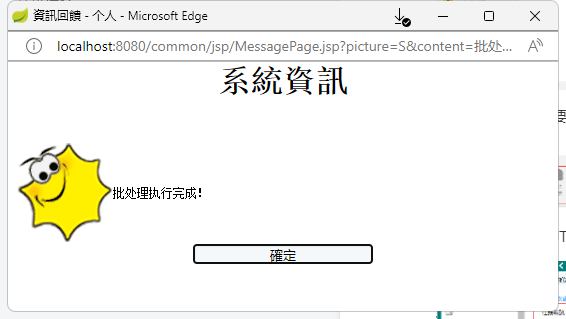


在完成记录后记得要点击“停止记录”



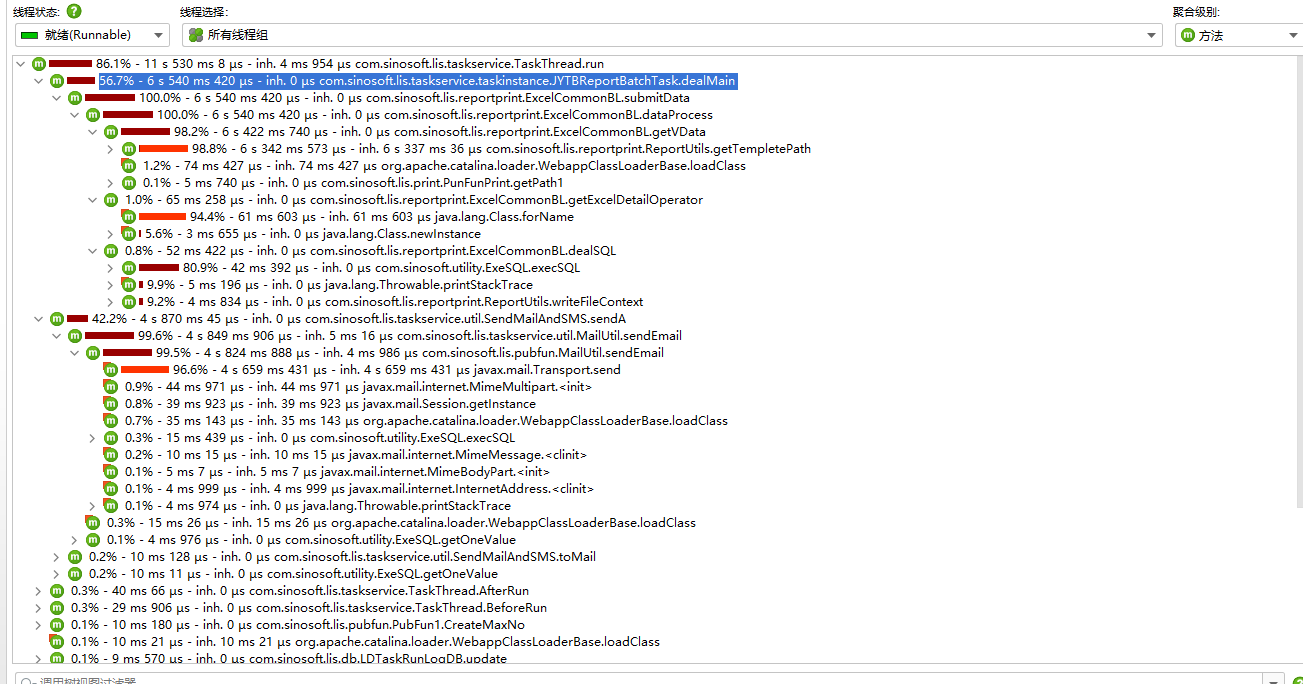
例如我测试批处理JT8405，我在点击执行按钮前就需要点击JProfiler中的“开始记录”按钮，等到执行完成出现后才点击 “停止记录”，点击“停止记录”的原因是别其他程序中其他代码的干扰。若要重新记录直接点击开始记录就可以，不用关心清楚之前的数据问题。



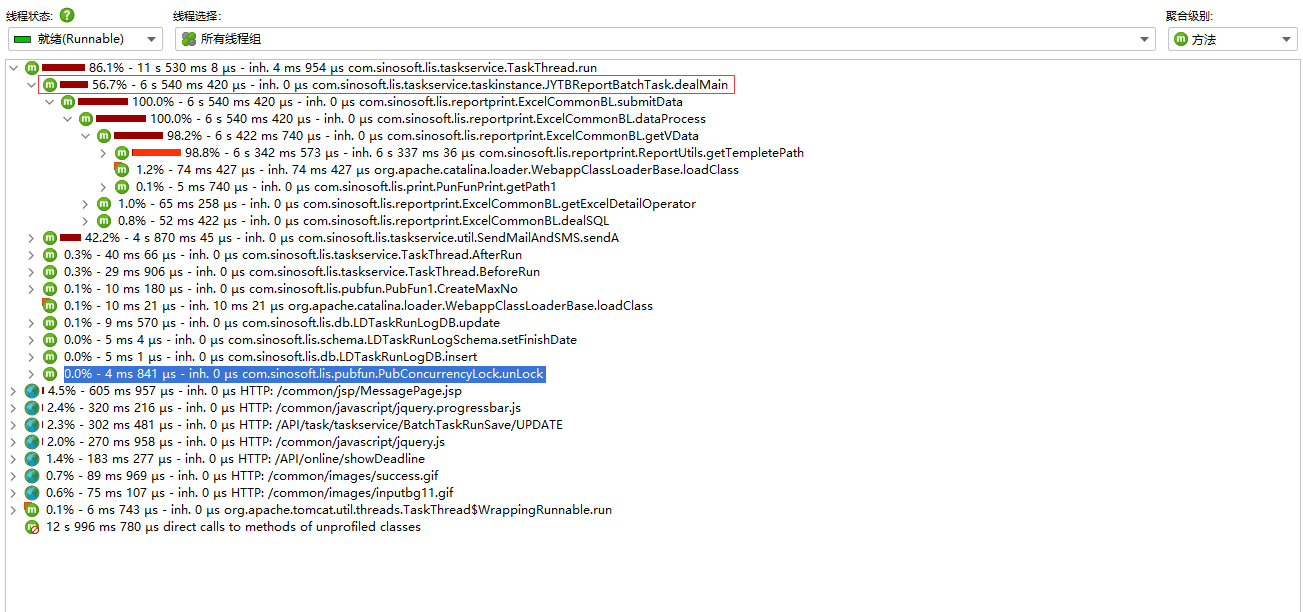


### 分析结果

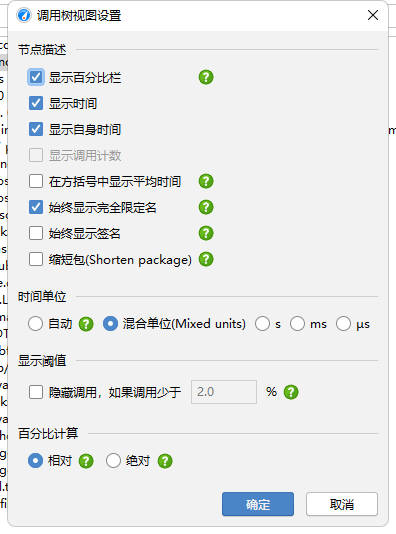
点击三角展开调用树，沿着调用树一路找到自己需要检测执行时间的代码的方法。



例如我执行批处理JT8405所涉及到的类和方法。



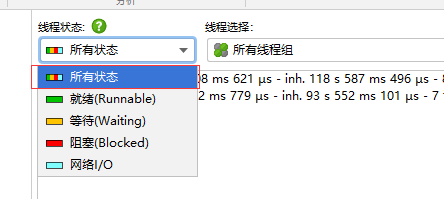
大家看到可能和我展示不一样，需要在“视图设置”设置一下。按照我的设置一下就好

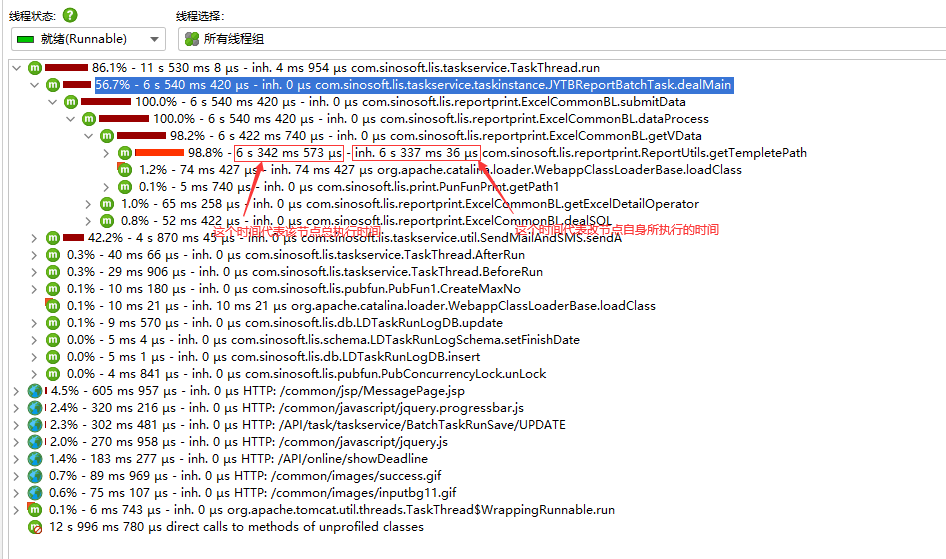


图中所说的总执行时间包括自身执行的时间以及所调用的其他函数或方法的时间。如果发现显示的时间与实际执行的时间相差很多，请将展示的线程状态调整为所有状态。

线程状态“就绪(Runnable)”只包括由操作系统调度运行线程的时间。 这意味着花费在等待网络Socket和其他线程上的时间不包括在内。

要分析应用程序在等待外部资源，例如:数据库或REST Services所花费的时间， 请切换到“网络I/O”线程状态。“阻塞(Blocked)”和 “等待(Waiting)”线程状态对于分析线程间通信非常有用。





如果涉及到SQL执行可以尝试开启数据的JDBC记录

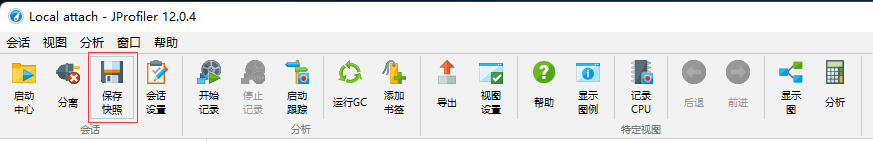


更多详细信息可以点击帮助



### 导出结果

点击工具栏的“保存快照”即可将结果保存下来。



## 其他

idea中安装完成JProfiler的插件后可以直接点击JProfiler的按钮,即可启动tomcat和JProfiler并同时完成附加操作。后面的操作就都是一样了。

descript

获取更加详细操作可以点击帮助按钮，获得点击该链接[JProfiler帮助文档](https://www.ej-technologies.com/resources/jprofiler/v/12.0/help_zh_CN/doc/main/introduction.html)