JVisualVM

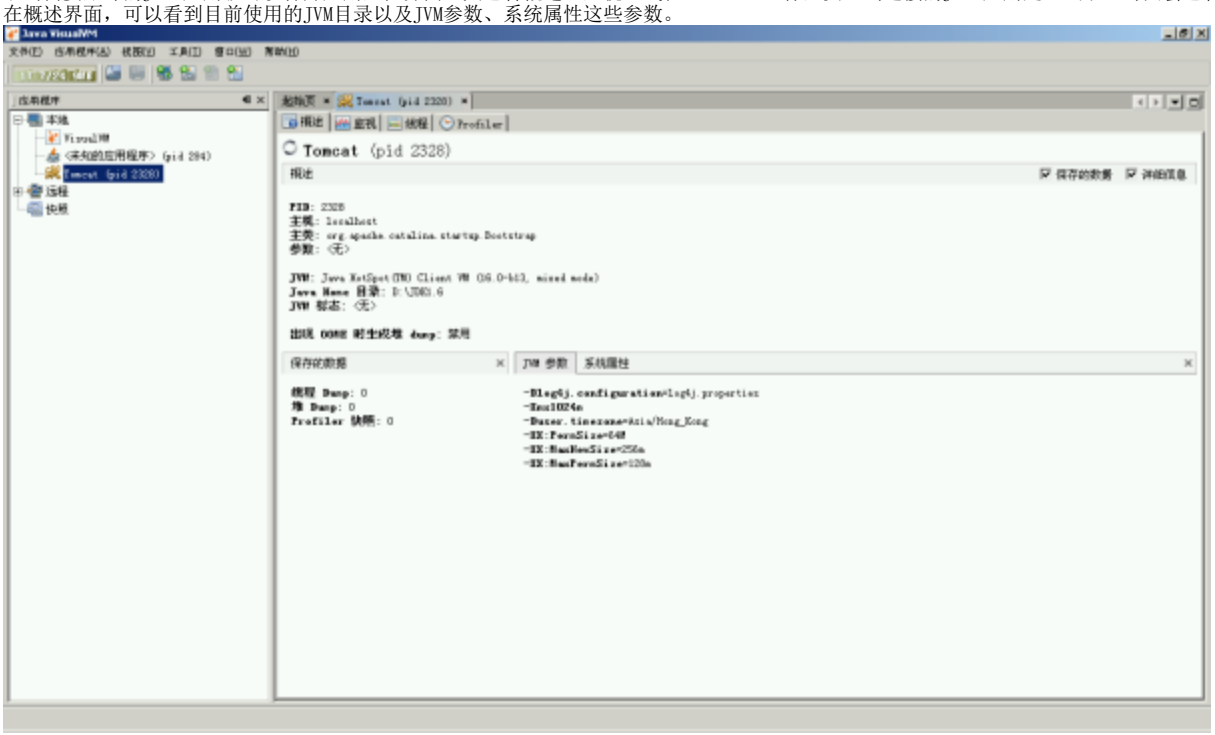
JVisualVM是集成了多个JDK命令工具的一个可视化工具，它主要用来监控JVM的运行情况，可以用它来查看和浏览Heap Dump、Thread Dump、内存对象实例情况、GC执行情况、CPU消耗以及类的装载情况。VisualVM必须运行在JDK1.6以上的VM环境下，但可以用它来监控JDK1.4以上的JVM（但是部分功能不能使用）。
JVisualVM中功能与JProfiler有一定重复，但是它的优点在于服务器不需要安装JProfiler，也不需要进行额外的配置（若要允许远程连接则需要，参看后面说明）。

启动JVisualVM

1. 双击jvisualvm.exe就可以启动，但是它有以下要求：
    a. 读写Windows的临时目录的权限
    b. 临时目录所在分区必须是NTFS格式的
2. 若提示
3. 可以在命令行提示符下，进入jvisualvm.exe所在目录然后输入： set TMP=D:\tempset TEMP=D:\tempjvisualvm.exe 启动，这样可以指定临时目录到另外一个分区中的目录。
4. 若客户端机器没有NTFS格式的分区，根据JDK文档可以在命令提示符中输入： jvisualvm -J-XX:+PerfBypassFileSystemCheck 但是可能会出现启动不报错，但是无法在本地应用中看到相应Java进程的问题，这时需要将本机添加为远程，参考以下的操作说明

监控本地 Java 应用

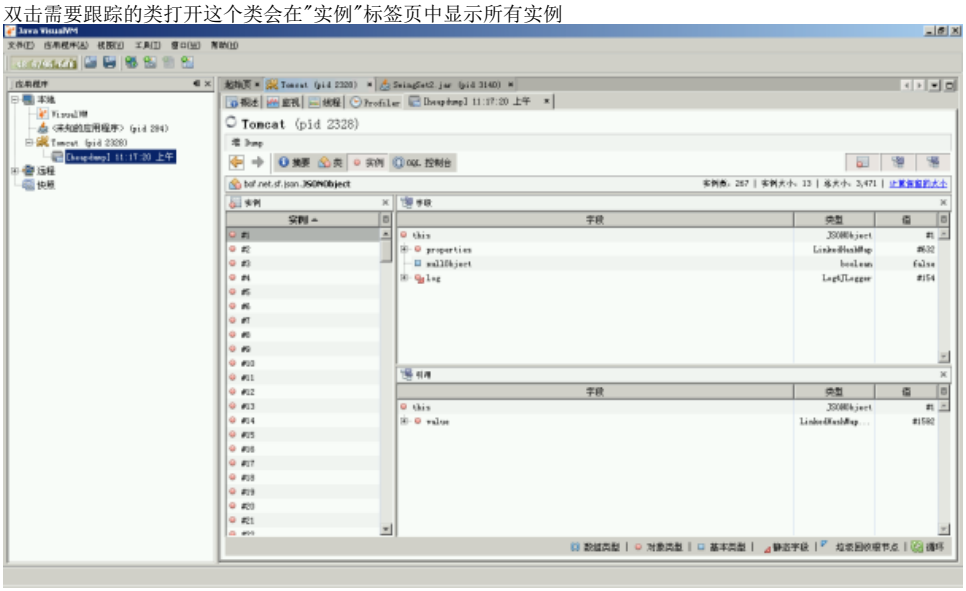
JVisualVM 本身是一个 Java 应用，所以打开 JVisualVM 看到的第一个可监控应用就是它本身。JVisualVM还会列举出当前机器上运行的所有Java应用，它会显示这个Java应用的MainClass和进程号（pid）。
双击需要跟踪的Java应用就可以打开应用监控界面，共包含概述、监视、线程、Profiler标签页（当连接的Java应用是1.4或1.5时只会包含概述和监视）。
在概述界面，可以看到目前使用的JVM目录以及JVM参数、系统属性这些参数。



在监视标签页中可以看到当前JVM所占用的CPU时间，堆和PermGen中内存占用情况，在实际监控中主要应该关注堆的内存占用情况。它还提供了"堆 Dump"功能可生成一份当前时刻的堆镜像，生成镜像后可以跟踪当前JVM中包含的各个实例：

1. 在左边树上双击打开heapdump
2. 在"类"标签页中通过在"类名过滤器"输入可以快速找到需要的类

3. 双击需要跟踪的类打开这个类会在"实例"标签页中显示所有实例



4. 这个界面中左边所有实例列表，右上方是当前这个实例中包含的所有属性，右下方是当前实例被哪些引用。在实际使用分析过程中主要是从右下方的树分析实例是否有被引用没有释放导致内存溢出。
5. 可以通过右上方的"计算保留的大小"得到该对象及其子孙属性实际占用的内存大小，但是计算花费时间比较多并有可能计算不太精确
6. OQL（Object Query Language）控制台是提供输入并执行OQL查询语句，OQL可以帮助快速的查找指定的实例（具体语法请参考Java网站）

在线程标签页中可以执行"线程 Dump"生成threaddump文件，这个功能主要用于分析服务器停止响应（例如无法打开报表、刷新数据等）。
在Profiler标签页中可以对该Java应用进行CPU和内存的性能分析。但是由于功能不是非常强大，建议使用JProfiler进行跟踪。

监控远程Java应用

当需要监控远程的Java应用时，需要远程的Java应用修改启动参数：

1. 在远程的Java应用的启动参数（参考JProfiler中的增加启动参数的说明）中增加-Dcom.sun.management.jmxremote.port=8999 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false 端口号请根据实际情况选择一个空闲端口
2. 若远程的Java应用为JDK 1.6或以上可以在远程服务器中启动jstatd，这样不需要修改远程Java应用的启动参数 在D:\JDK1.6\bin目录中增加一个文本文件jstatd.all.policy，修改内容为：
   ```
   grant codebase "file:${java.home}/../lib/tools.jar" {
         permission java.security.AllPermission;
   };
   grant codebase "file:${java.home}/lib/tools.jar" {
         permission java.security.AllPermission;
   };
   ```
   然后在命令提示符中输入jstatd -J-Djava.security.policy=jstatd.all.policy启动
3. 在服务器启用了远程监控功能后，在客户端的JVisualVM中就可以在在左边的树"远程"节点中右键"添加远程主机"
4. 添加主机后，若远程服务器使用了jstatd会自动添加列出所有已经启动的Java进程。若使用的是启动参数方式，则在主机中右键"添加JMX连接"并输入正确的端口号
5. 添加完成后就可以双击左边树上的应用打开这个应用的概述界面

但是远程监控功能相对较弱并且部分功能会无法使用，可以考虑直接在服务器上启动JVisualVM监控。当服务器是Unix/Linux等非Windows环境时，可以通过Xmanager连接到服务器上，这样可以将直接在服务器上启动JVivusalVM

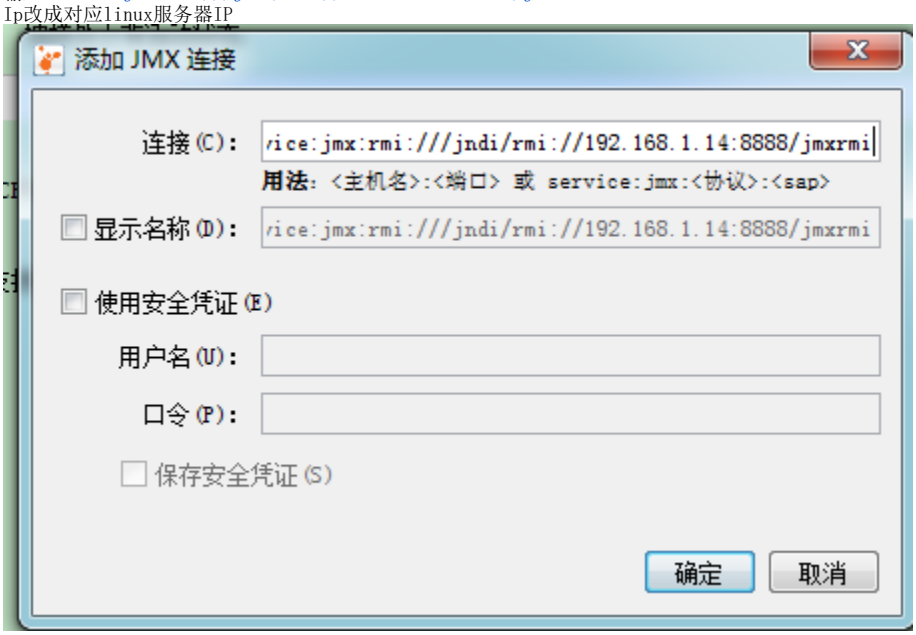
监控Linux

在应用服务器上增加JVM参数"-Djava.rmi.server.hostname=192.168.1.14 -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=8888 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false" hostname需要修改为服务器的实际IP

? 不能识别的附件

在window打开jvisualvm选择"添加JMX连接"

输入service:jmx:rmi:///jndi/rmi://192.168.1.14:8888/jmxrmi
Ip改成对应linux服务器IP



点击确认连接就可以看到远程机器的信息了