# 通过JMX监控Spring Boot应用 - 简书

http://www.jianshu.com/p/673385926ce1

在[Spring Boot应用的健康监控](http://www.jianshu.com/p/734519d3c383)一文中，我们通过Spring Boot Actuator对外暴露应用的监控信息，除了使用HTTP获取JSON格式 的数据之外，还可以通过JMX监控应用，Spring Boot也提供了对JMX监控的支持。JMX监控对外暴露的信息相同，不过是使用MBeans容器将应用数据封装管理。

接下来我们看下如何利用JMX获取应用状态信息，以及如何使用Jolokia JMX库对外暴露MBeans的HTTP访问URL。

## Get Ready

在BookPub应用的pom文件中添加jolokia-core依赖

<!-- JMX monitor -->

<dependency>

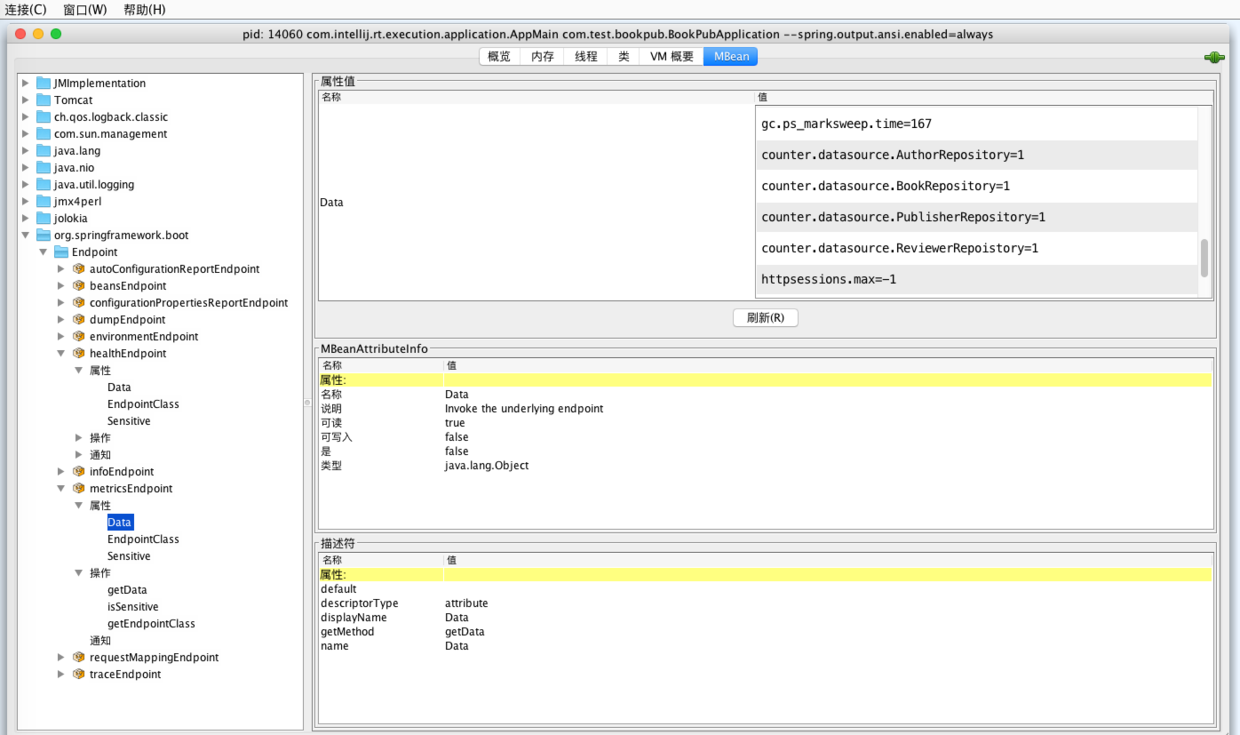
<groupId>org.jolokia</groupId>

<artifactId>jolokia-core</artifactId>

</dependency>

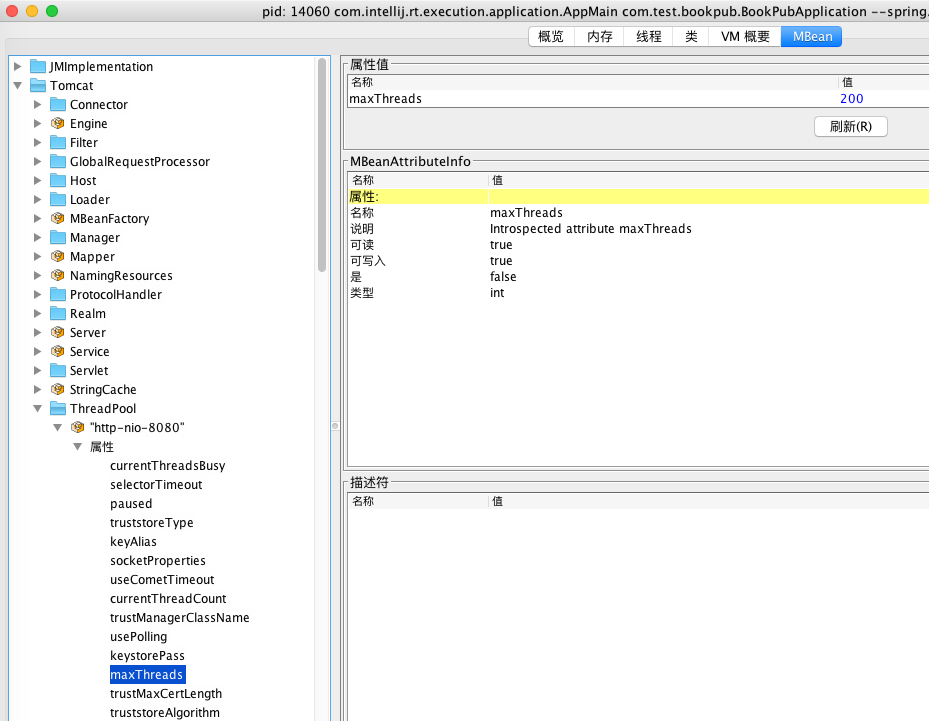
## How Do

1. 启动BookPub应用，然后在命令行中执行jconsole命令启动“Java管理和监视控制台”，然后选择org.springframework.boot节点下的Endpoint，可以看到如下信息



Java管理和监视控制台

1. 在Tomcat节点下选择ThreadPool，然后在选择http-nio-8080节点，在这个节点下选择maxThreads属性，可以看到如下信息



查看应用的最大并发线程数

1. 除了通过JMX获取信息，也暴露了对应的HTTP接口访问Mbeans对象的信息，例如，我们在postman中访问[*http://localhost:8080/jolokia/read/Tomcat:type=ThreadPool,name=%22http-nio-8080%22/maxThreads*](http://localhost:8080/jolokia/read/Tomcat:type=ThreadPool,name=%22http-nio-8080%22/maxThreads)，也可以得到对应的信息



通过HTTP访问应用的最大并发线程数

## 分析

只要添加了Spring Boot Actuator库，所有相关的endpoint和管理服务都打开了，包括JMX，我们可以通过设置endpoints.jmx.enabled=false禁止对外提供基于JMX的endpoints；或者通过设置spring.jmx.enabled=false禁止对外提供Spring MBeans。

在类路径中存在的Jolokia库会触发Spring Boot的JolokiaAutoConfiguration，这个自动配置类会自动配置可以接受/jolokia请求的JolokiaMvcEndPoint；也可以通过在application.properties中设置jolokia.config.系列的属性配置不同的Jolokia配置。完整的Jolokia配置地址在：[*https://jolokia.org/reference/html/agents.html#agent-war-init-params*](https://jolokia.org/reference/html/agents.html#agent-war-init-params)。如果你希望定制自己的Jolokia配置，则可以通过设置endpoints.jolokia.enabled=false\*让Spring Boot应用忽略自身提供的配置。

# maven-framework-project/使用JMX监控Tomcat.md at master · v5developer/maven-framework-project

https://github.com/v5developer/maven-framework-project/blob/master/spring3-jmx-example/%E4%BD%BF%E7%94%A8JMX%E7%9B%91%E6%8E%A7Tomcat.md

# 使用JMX监控Tomcat

## 1. 激活Tomcat的JMX远程配置

要通过JMX远程监控Tomcat，首先需要激活Tomcat的JMX远程配置。

① 先修改Tomcat的启动脚本，windows下为bin/catalina.bat（linux下为catalina.sh），添加以下内容，8999是jmxremote使用的端口号，第二个false表示不需要鉴权：

set JMX\_REMOTE\_CONFIG=-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=8999 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false

set CATALINA\_OPTS=%CATALINA\_OPTS% %JMX\_REMOTE\_CONFIG%

要注意以上语句的位置不能太后面，可以加在【if "%OS%" == "Windows\_NT" setlocal】一句后的大段的注释后面。

参考官方说明：

<http://tomcat.apache.org/tomcat-6.0-doc/monitoring.html#Enabling_JMX_Remote>

② 上面的配置是不需要鉴权的，如果需要鉴权则添加的内容为：

set JMX\_REMOTE\_CONFIG=-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=8999 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=true -Dcom.sun.management.jmxremote.password.file=../conf/jmxremote.password -Dcom.sun.management.jmxremote.access.file=../conf/jmxremote.access

set CATALINA\_OPTS=%CATALINA\_OPTS% %JMX\_REMOTE\_CONFIG%

③ 然后复制并修改授权文件

$JAVA\_HOME/jre/lib/management下有jmxremote.access和jmxremote.password的模板文件，将两个文件复制到$CATALINA\_BASE/conf目录下

◆ 修改$CATALINA\_BASE/conf/jmxremote.access 添加内容： monitorRole readonly controlRole readwrite

◆ 修改$CATALINA\_BASE/conf/jmxremote.password 添加内容： monitorRole admin controlRole admin

注意： 如果进行了以上步骤导致Tomcat启动不了，那么很可能是密码文件的权限问题

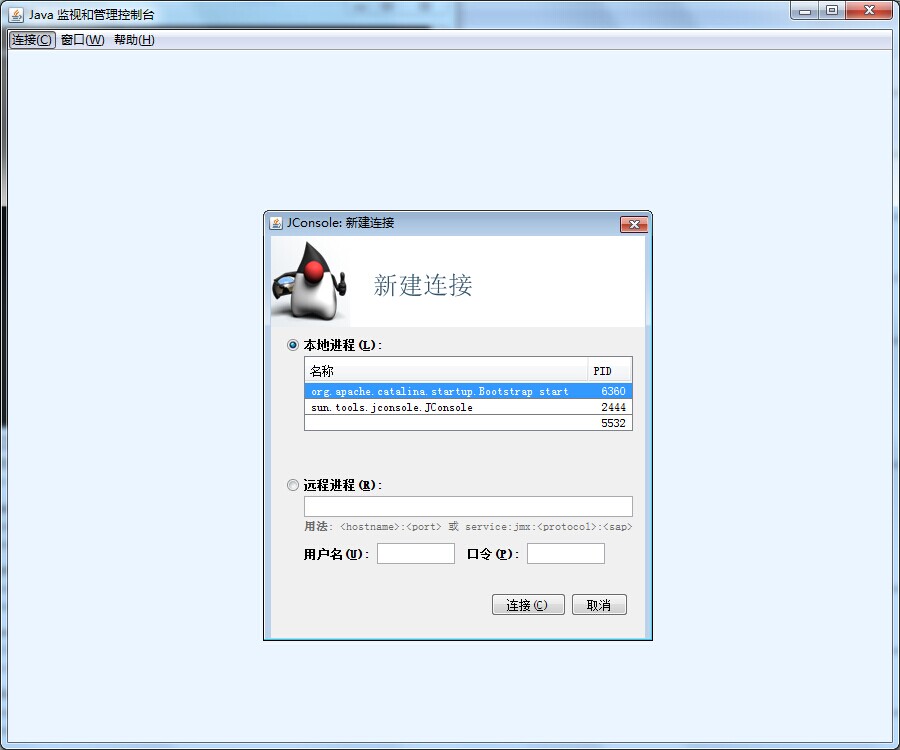
需要修改jmxremote.password文件的访问权限，只有运行Tomcat的用户才能拥有访问权限 ： Windows的NTFS文件系统下，选中文件，点右键 -->“属性”-->“安全”--> 点“高级”--> 点“更改权限”--> 去掉“从父项继承....”--> 弹出窗口中选“删除”，这样就删除了所有访问权限。再选“添加”--> “高级”--> “立即查找”，选中你的用户（或用户组，如果选用户不行那就选用户组），例administrator，点“确定"，“确定"。来到权限项目窗口，勾选“完全控制”，点“确定”，OK了。

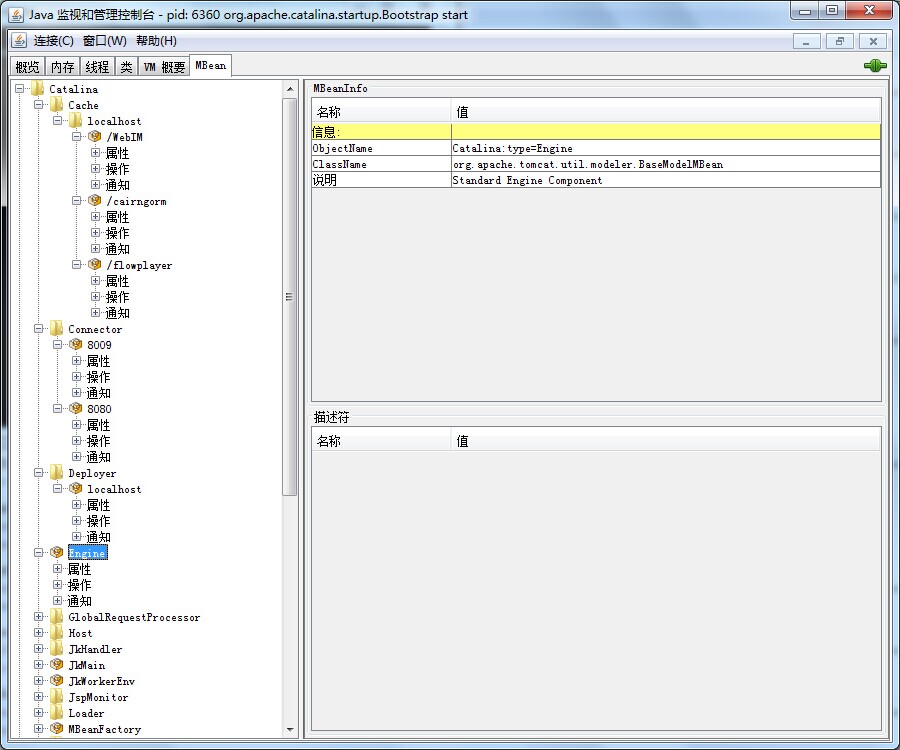
官方的提示： The password file should be read-only and only accessible by the operating system user Tomcat is running as.

④ 重新启动Tomcat，在Windows命令行输入“netstat -a”查看配置的端口号是否已打开，如果打开，说明上面的配置成功了。

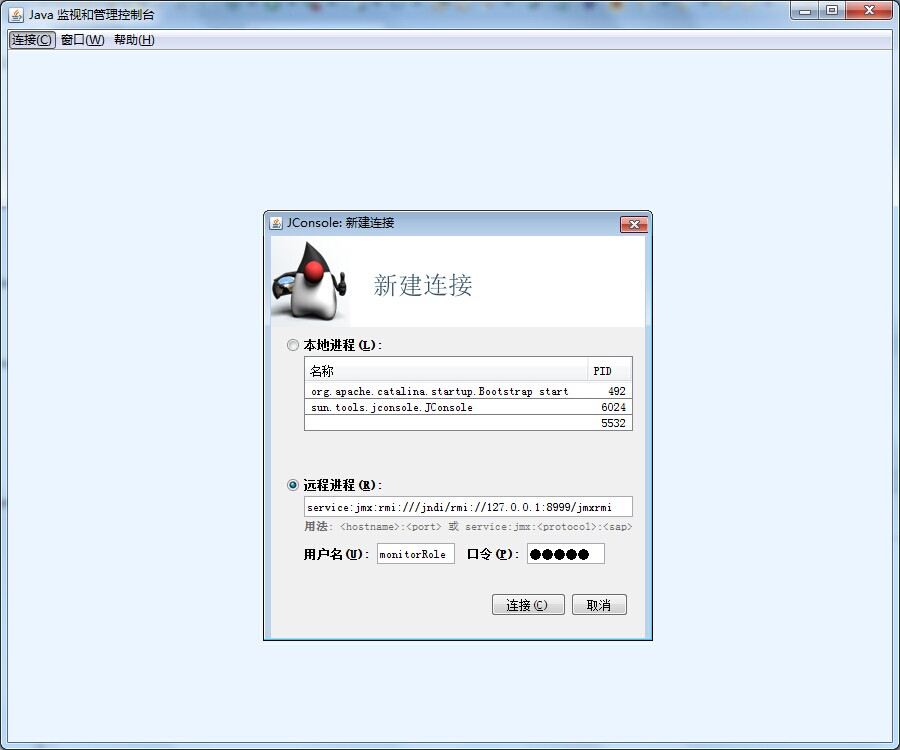
⑤ 使用jconsole测试JMX。

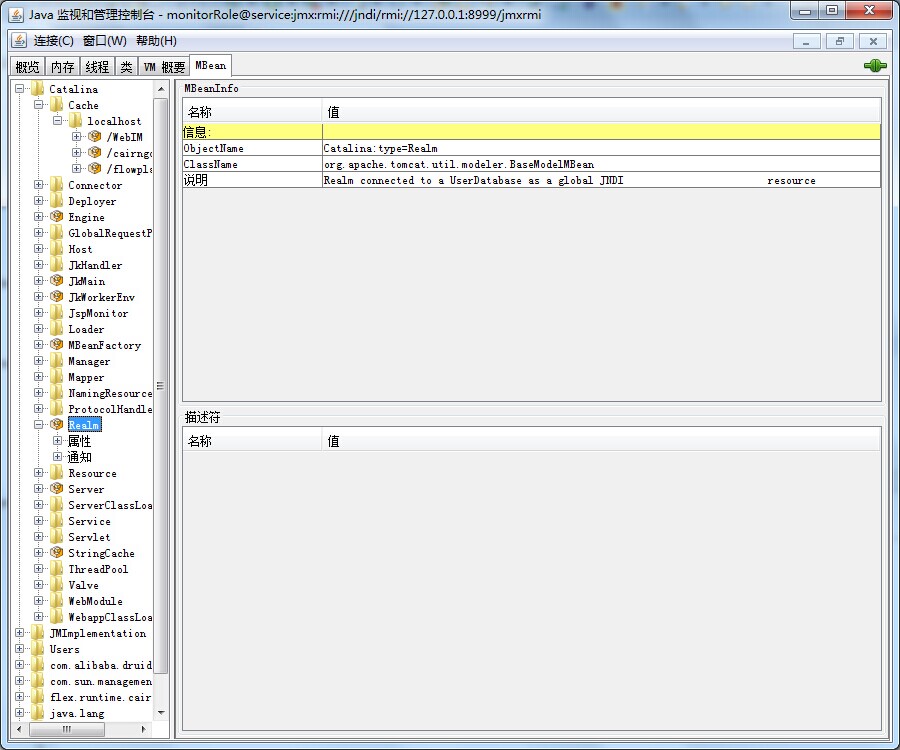
运行$JAVA\_HOME/bin目录下的jconsole.exe，打开J2SE监视和管理控制台，然后建立连接，如果是本地的Tomcat则直接选择然后点击连接，如果是远程的，则进入远程选项卡，填写地址、端口号、用户名、口令即可连接。。Mbean属性页中给出了相应的数据，Catalina中是tomcat的，java.lang是jvm的。对于加粗的黑体属性值，需双击一下才可看内容。

[](https://github.com/v5developer/maven-framework-project/blob/master/spring3-jmx-example/src/main/resources/jmx-tomcat-1.jpg)

[](https://github.com/v5developer/maven-framework-project/blob/master/spring3-jmx-example/src/main/resources/jmx-tomcat-2.jpg)

使用远程连接查看tomcat

[](https://github.com/v5developer/maven-framework-project/blob/master/spring3-jmx-example/src/main/resources/jmx-tomcat-3.jpg)

[](https://github.com/v5developer/maven-framework-project/blob/master/spring3-jmx-example/src/main/resources/jmx-tomcat-4.jpg)

## 2. 使用JMX监控Tomcat示例代码

String jmxURL = "service:jmx:rmi:///jndi/rmi://192.168.10.93:8999/jmxrmi";

JMXServiceURL serviceURL = new JMXServiceURL(jmxURL);

Map map = new HashMap();

// 用户名密码，在jmxremote.password文件中查看

String[] credentials = new String[] { "monitorRole", "admin" };

map.put("jmx.remote.credentials", credentials);

JMXConnector connector = JMXConnectorFactory.connect(serviceURL, map);

MBeanServerConnection mbsc = connector.getMBeanServerConnection();

// 端口最好是动态取得

ObjectName threadObjName = new ObjectName("Catalina:type=ThreadPool,name=http-8080");

MBeanInfo mbInfo = mbsc.getMBeanInfo(threadObjName);

// tomcat的线程数对应的属性值

String attrName = "currentThreadCount";

MBeanAttributeInfo[] mbAttributes = mbInfo.getAttributes();

System.out.println("currentThreadCount:" + mbsc.getAttribute(threadObjName, attrName));

# Maven build 中启动 JMX Agent | Leegorous

http://leegorous.net/blog/2011/08/23/run-maven-build-with-jmx-agent/

为了监控 JVM，研究了下 [VisualVM](http://visualvm.java.net/)，但是不知道怎么导出里面的数据，可导出就比较方便做数据比对，而且也不知道怎么控制显示的历史信息跨度。而这两点在 [JConsole](http://download.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html) 中都可以。于是要折腾 JConsole，一开始还不怎么顺利，虽然看上去只是几项配置。  
因为在开发过程当中都是使用 mvn jetty:run 启动的。原来以为是这样：

[?](http://leegorous.net/blog/2011/08/23/run-maven-build-with-jmx-agent/)

|  |  |
| --- | --- |
| 1  2  3 | mvn jetty:run -Dcom.sun.management.jmxremote.port=9410 \      -Dcom.sun.management.jmxremote.authenticate=false \      -Dcom.sun.management.jmxremote.ssl=false |

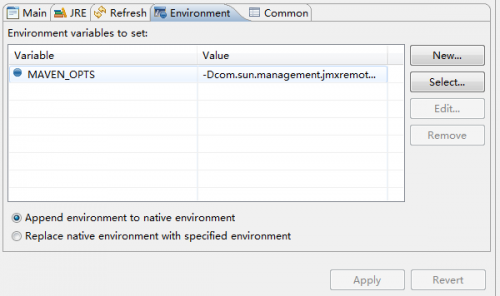
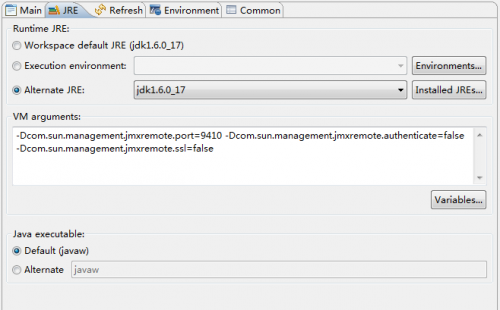
然后使用 **jconsole 127.0.0.1:9410** ，提示是 Connection refused

后来了解到，如果 JMX Agent 启动成功的话，会产生一个端口监听。  
再执行 **netstat -a | grep 9410** ，毫无反应。

网上乱 google jconsole, jmx agent, jmx remote, jmx connection refused 等关键字都没什么头绪。  
找到一篇 [关于 jconsole 的错误排查](http://blogs.oracle.com/jmxetc/entry/troubleshooting_connection_problems_in_jconsole) 的不错的文章，但是情况不吻合，这里唯一区别的是不是直接 java 启动，而是通过 mvn ，曾经使用过调试，需要添加 MAVEN\_OPTS 的参数的。一试成功。

[?](http://leegorous.net/blog/2011/08/23/run-maven-build-with-jmx-agent/)

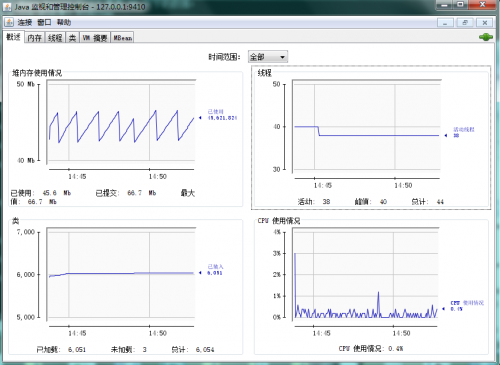
|  |  |
| --- | --- |
| 1  2  3 | set MAVEN\_OPTS=-Xmx512m -Dcom.sun.management.jmxremote.port=9400 \      -Dcom.sun.management.jmxremote.authenticate=false \      -Dcom.sun.management.jmxremote.ssl=false %MAVEN\_OPTS% |

再进一步，通常是在 eclipse 的 m2e 插件启动 jetty:run，如下图设置  
[](http://leegorous.net/blog/wp-content/uploads/2011/08/m2e-set-jmx.png)  
经过多次验证，基本肯定这里存在 bug 的，设置 MAVEN\_OPTS 是无效的，而且不报错。  
既然这是作为 JDK 参数的，再尝试将参数添加到 jdk 的启动参数里面，居然报错了～  
[](http://leegorous.net/blog/wp-content/uploads/2011/08/m2e-set-jdk-with-jmx.png)

[?](http://leegorous.net/blog/2011/08/23/run-maven-build-with-jmx-agent/)

|  |  |
| --- | --- |
| 1 | Exception thrown by the agent : java.net.MalformedURLException: Local host name unknown: java.net.UnknownHostException: LEX-WS: LEX-WS |

遂往 hosts 添加 **127.0.0.1 LEX-WS**  
启动依然报错。  
郁闷，重启一下 eclipse，在启动，居然好了。莫非动态修改 hosts 文件 jvm 里是无法动态跟上的？

好，现在一切就绪，再次执行 **jconsole 127.0.0.1:9410**  
[](http://leegorous.net/blog/wp-content/uploads/2011/08/jconsole-connected.png)  
这是在本地模拟远程的情况。在远程的配置下，还需要加入一个 **-Djava.rmi.server.hostname=<host.name.or.ip>**，否则，可能会出现连接没有拒绝，但是总是连接不成功的情况，那也是相当的郁闷的。

PS，这样配置之后，也是可以在 visualvm 里通过添加 JMX Connection 来做监控的，而且多了个线程监控。

文档信息

* 版权声明：保持署名-非商用-共同方式共享 | [Creative Commons BY-NC-SA 3.0](http://creativecommons.org/licenses/by-nc-sa/3.0/)
* 原文地址：<http://leegorous.net/blog/2011/08/23/run-maven-build-with-jmx-agent/>
* 付费支持：[通过支付宝](https://me.alipay.com/leegorous)

# jmx - Automatic start of jconsole from eclipse - Stack Overflow

http://stackoverflow.com/questions/14702593/automatic-start-of-jconsole-from-eclipse

-Djava.rmi.server.hostname=localhost -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.port=10001

This is not about automatic start, but it is about connecting the program running in eclipse through jvisualvm.

In eclipse, in the Main class java file do the below, then run as java application.

Right click --> Run As --> Run Configuration --> Arguments --> VM argumanets --> Add the below

-Djava.rmi.server.hostname=hostname -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.port=10001

Then open java visaul vm from JDK bin directory. In that

File --> Add JMX Connection --> In connection text box --> localhost:10001

This will start your profiling now.

# Maven pom.xml

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<jvmArguments>-Djava.rmi.server.hostname=localhost -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xdebug -Xrunjdwp:transport=dt\_socket,server=y,suspend=n,address=5005</jvmArguments>

<arguments>

<argument>--spring.profiles.active=dev</argument>

</arguments>

</configuration>

</plugin>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<jvmArguments>-XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xdebug

-Xrunjdwp:transport=dt\_socket,server=y,suspend=n,address=5005

</jvmArguments>

<arguments>

<argument>--spring.profiles.active=dev</argument>

</arguments>

</configuration>

</plugin>

</plugins>

</build>