Tuesday, December 20, 2016

如何指导编写一个javaagent

By: Axel Purr

想象一下这个情景，您在一个Java进程中突然需要获得所有已加载类的字节码。例如，假设您想要调试在运行时发生的某种类型的instrumentation
，而您无法定位.class文件并对其进行检查。您需要更深入地挖掘。顺便说一句，如果您刚刚开始接触字节码，这里有一篇很有用的报告，能帮助您入门——掌握Java字节码。
或者，假设您正在按照JITWatch工作，并希望将您生成的字节码与JIT生成的机器代码链接在一起。在任

何情况下，您都会遇到与类相关的问题，正如我们都知道的，关于类和类加载器的问题都很麻烦，您要准备好咖啡，大量的咖啡+++！

在本文中，我们将介绍两种方法来获取加载到JVM中的类的字节码，并学习关于javaagent和HotSpot Debugger（JDK的隐藏功能）的一两件事。

那么，让我们先来概述一下我们在本文中要解决的问题。

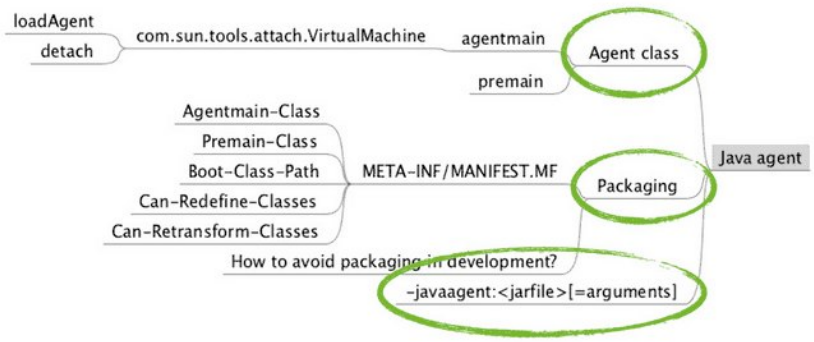
问题描述：获取加载到JVM中的所有类的字节码。

让我们开始吧！

自己动手做一个javaagent

通常，javaagent是一个JVM“插件”，一种专门精心制作的.jar文件，它能够利用JVM提供的Instrumentation API。Java 1.5提供了Instrumentation API，因此无论从哪方面来看，我们将要探讨的解决方案都是非常便捷的。

为了创建一个成功的javaagent，我们需要四个东西：一个代理类、一些元信息（告诉JVM为我们的代理类提供哪些功能）、一种使JVM加载.jar文件和代理的方式（在它开始进行与应用程序有关的工作之前）和一杯咖啡。咖啡已经放在桌上了？好吧，还有三件事要做。



上面是几星期前我参加巴塞罗那JUG会议时所作的报告——玩转Java代理中的一页幻灯片。

我必须诚实地说，这张幻灯片最初是由Java冠军、JavaOne Rockstar发言人和 XRebel产品总监Anton Arhipov制作的，因此他应该获得所有的赞誉。但谁让他现在*不在这里*呢，所以我会说这张幻灯片是我做的！

无需多言，让我们先来创建一个Java类作为我们的代理类。这里要记住的重要事情是，我们的VM将尝试定位我们的类，我们在 -javaagent 参数中将此类指定给VM，并在 main 方法之前执行他的 premain方法。

```
import java.lang.instrument.Instrumentation;
public class Agent {
  public static void premain(String args,
Instrumentation instrumentation){
    ClassLogger transformer = new
ClassLogger();

instrumentation.addTransformer(transformer);
  }
}
```

要注意，我们在 premain 方法中可以访问的 Instrumentation 参数。它是一个功能强大的API，它的其中一个优点是能够让我们注册 ClassFileTransformers。一个已注册的 ClassFileTransformer 将拦截所有应用程序类的加载，并能够访问他们的字节码。

顺便提一句，ClassFileTransformer 也可以转换应用程序类的字节码，并使JVM的加载行为与原来预期的字节完全不同。

几乎所有很酷的JVM工具都是这样运作的，包括 JRebel 和XRebel，它们会干预您的类并将类的功能正确地注入您的应用中，并且无需依赖额外的工具，甚至不用更改您的代码。另外，既然我们提到了这两个工具，您就应该对它们的迷人之处有所了解，如果您是一名Java开发人员却还没接触过JRebel或 XRebel，那么您一定会爱上它们!

跑题了，让我们回到记录字节码！现在，我们已经注册了一个ClassLogger 转换器，让我们来看看它是如何实现的。

```
public class ClassLogger implements
ClassFileTransformer {
  @Override
  public byte[] transform(ClassLoader
loader,
                          String className,
                          Class<?>
classBeingRedefined,
                          ProtectionDomain
protectionDomain,
                          byte[]
classfileBuffer) throws
IllegalClassFormatException {
    try {
      Path path = Paths.get("classes/" +
className + ".class");
      Files.write(path, classfileBuffer);
```

```
    } catch (Throwable ignored) { //
ignored, don't do this at home kids
    } finally { return classfileBuffer; }
  }
}
```

正如您看到的，代码在这个例子中非常微不足道。transform 方法可以访问应用程序的类名称以及该类的主体对应的字节。在我们的例子中，我们只是将字节放到了一个文件中。

很遗憾，编码时间就此结束，最后一部分工作是打包一个jar文件，并提供一个清单文件，用于指定我们的代理为 Premain-Class。

下面是Gradle
构建文件部分，其中会用到一个诀窍：

```
public class ClassLogger implements
ClassFileTransformer {
  @Override
  public byte[] transform(ClassLoader
loader,
                          String className,
                          Class<?>
classBeingRedefined,
                          ProtectionDomain
protectionDomain,
                          byte[]
classfileBuffer) throws
IllegalClassFormatException {
    try {
      Path path = Paths.get("classes/" +
className + ".class");
      Files.write(path, classfileBuffer);
    } catch (Throwable ignored) { //
ignored, don't do this at home kids
    } finally { return classfileBuffer; }
  }
}
```

所有位和片段都已为您备好，以便您开始创建您的

javaagent
jar，在您开始享受拦截类加载以记录类字节的强大功能之前，您所要做的只是提供 -javaagent 参数。这一步简单至极：

```
java
-jar myapp.jar
```

```
java
-javaagent:/path/to/agent.jar
-jar myapp.jar
```

当myapp.jar结束运行时，在类的目录下，您会发现一堆 .class 文件需要您检查。我手边没有代理的源代码，但在这个方便的Github存储库中，您可以找到一个完整的javaagent
项目，虽然有点复杂，但都遵循同样的原理。您可以看一看，它非常酷。

像专家一样使用HSDB

刚才，我们探讨了如何创建一个简单的javaagent，它可拦截类加载，并将所有加载类的字节写入我们文件系统的文件中。这是我用来解决获得所有加载类的字节码这一问题的默认方法。Javaagent确实功能极为强大，事实上它也能解决许多类似问题。

但现在我想分享另一种检查类的字节码的方法，并使用另一个很酷的工具（HSDB）。我近期在巴塞罗那举行的SpringIO大会上刚刚了解到这种工具，这要感谢Thomas ，很高兴能在此与大家分享。

HSDB
位于一个 sa-jdi.jar 文件中，您可以在本地的HotSpot JDK分布的 lib 目录下找到它。

为了在您所选的Java流程上发挥HSDB的强大功能，您必须以特权访问运行它，并为它提供您想要查询的

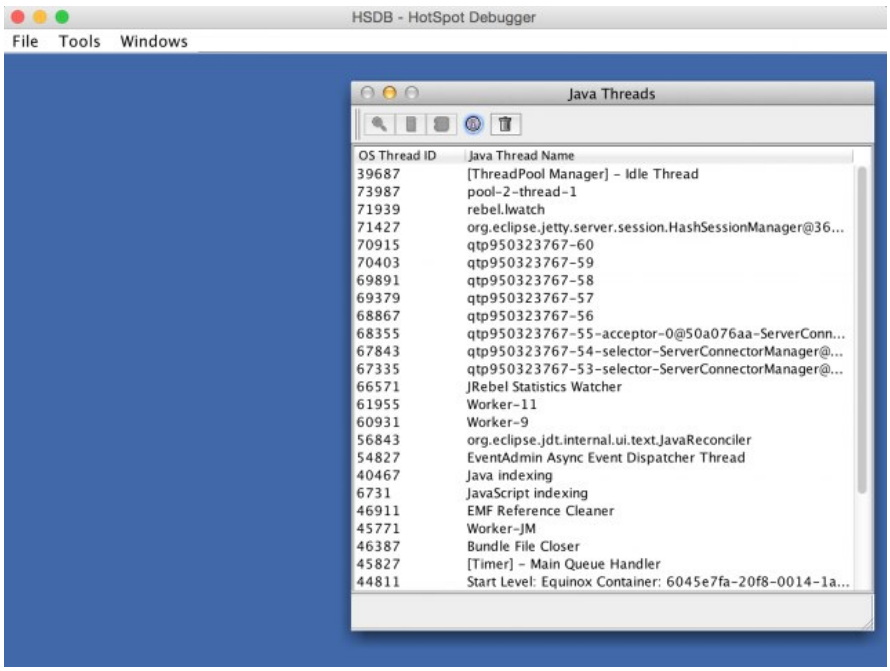
Java程序的PID：

shelajev@shrimp /Library/Java/JavaVirtualMachines
/jdk1.8.0_40.jdk/Contents/Home

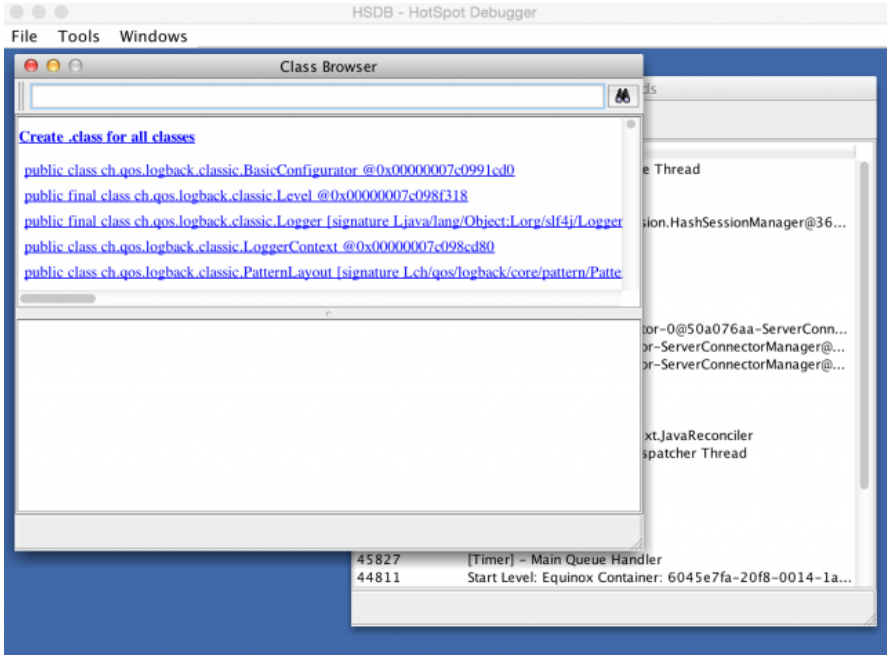
$
sudo java -cp lib/sa-jdi.jar sun.jvm.hotspot.HSDB

从主菜单中选择File
=> Attach to HotSpot process …
并输入您Java流程的PID。顺便提一句，获取Java流程PID的最好方法之一是使用 jps
——另一个内置的JDK工具。

在建立连接之后，我们可以看到目标Java流程中的线程 列表（我选择Eclipse作为我的Java流程）。



如果您有时间的话，看一下这个列表非常有用，但与我们的问题更具相关性的是获得类文件的视图，，可通过 Tools
=> Class browser。



您在这个新窗口的顶部可以看到Create .class for all classes （为所有类创建.class）的链接。点击此链接将生成所有已加载的应用程序类文件，我们用更快的速度解决了同样的问题，虽然这种方法与之前的javaagent方式相比有些怪异。

HotSpot Debugger功能当然不只局限于生成类文件。它是一款十分强大的工具，而它的其他功能我会另找时间详述。在本文的开头，我们提出了一个获得类文件的问题，最终我们得到了解决方法。欢迎来到字节码的世界！

结论

在本文中，我们探讨了Java开发人员使用的两种超级强大的工具——javaagent和HotSpot Debugger。这两种工具都具有很多功能，也都有能力访问JVM的加载类，我们只不过粗浅地尝试了它们的一种能力，希望您在阅读本文后对它们有所了解，并可以进一步探索它们的强大功能。欢迎您与我分享您的进展！

您会怎样解决上述问题？请在下方的评论栏踊跃留言，或通过Twitter与我联系@shelajev。

OLEG
SHELAJEV 开发人员倡导者

Oleg Šelajev是一名工程师、作家、演讲者、讲师和ZeroTurnaround上的倡导者。他主要从事测试、编码、写作、大会演讲、撰写博文和报告等工作。他目前正在攻读动态系统升级和代码演变专业的博士学位。Oleg是塔尔图大学的兼职讲师，他喜欢参加Java/JVM开发大会并发言，例如JavaOne、JavaZone、JFokus等大会活动。在业余时间里，Oleg喜欢下国际象棋，水平达到半大师级，也喜欢玩拼图和解决各种难题。

Join the discussion