

Can I get a Groovy SDK to invoke the...



Eric Sword 28 posts since

Jun 17, 2009

Can someone explain how IntelliJ gets the setting for `groovy.starter.conf` for a particular Groovy SDK, and if it's possible to change that setting to point to another file somehow?

Here's my situation: I am running EAP 9880 (i.e. a pre-8.1.3 release). I created a groovy SDK by pointing at the root directory of a normal Groovy 1.6.3 install. This created a global "groovy-1.6.3" library in IntelliJ and added all the jars in the `groovy-1.6.3/lib` directory to it. I have a groovy script that invokes a call on a java class in a 3rd party jar. The classes in that jar have a dependency on an old asm library, one that conflicts with the asm library that ships with groovy. I can normally run my script (and other classes that do the same thing) without a problem by invoking the embeddable version of groovy since it puts all of its third-party dependencies in another name space. If I try to run the script from within IntelliJ, the command line generated starts like this:

```
C:\Dev\java\jdk1.5.0_15\bin\java -Dgroovy.home=C:\Dev\utilities\groovy\my-groovy-1.6.3 -Dgroovy.starter.conf=C:\Dev\utilities\groovy\my-groovy-1.6.3/conf/groovy-starter.conf -Dtools.jar=C:\Dev\java\jdk1.5.0_15\lib\tools.jar -Xms128m -Xmx1124m -Didea.launcher.port=7532 "-Didea.launcher.bin.path=C:\Dev\intellij\IntelliJ IDEA 9880\bin" -Dfile.encoding=windows-1252 -classpath "C:\Dev\utilities\groovy\my-groovy-1.6.3\embeddable\groovy-all-1.6.3.jarC:\Dev\intellij\IntelliJ IDEA 9880\lib\idea_rt.jar" com.intellij.rt.execution.application.AppMain org.codehaus.groovy.tools.GroovyStarter --main groovy.ui.GroovyMain --conf C:\Dev\utilities\groovy\my-groovy-1.6.3/conf/groovy-starter.conf --classpath "C:\Dev\workspace\ie-tools-maven\core\target\test-classesC:\Dev\workspace\ie-tools-maven\core\target\classesC:\Dev\utilities\groovy\groovy-1.6.3\lib\ant-1.7.1.jarC:\Dev\utilities\groovy\groovy-1.6.3\lib\commons-cli-1.2.jarC:\Dev\utilities\groovy\groovy-1.6.3\lib\ant-junit-1.7.1.jarC:\Dev\utilities\groovy\groovy-1.6.3\lib\asm-tree-2.2.3.jarC:\Dev\utilities\groovy\groovy-1.6.3\lib\commons-logging-1.1.jarC:\Dev\utilities\groovy\groovy-1.6.3\lib\junit-3.8.2.jarC:\Dev\utilities\groovy\groovy-1.6.3\lib\asm-2.2.3.jar... [more groovy classes and then all my regular dependencies]...
```

Two execution configs are setup here (though only a single `java.exe` is launched). You can see the initial classpath for the IntelliJ `AppMain` class contains the embeddable `groovy-all-1.6.3.jar` file, but the classpath for `GroovyStarter` (the thing that actually runs the script) contains all the non-embeddable jars. The script starts running, but once it invokes a

Can I get a Groovy SDK to invoke the...

method in that 3rd party jar, I eventually get a `NoSuchMethodError` because it tries to call a method in the `asm` library that no longer exists in the `asm-2.2.3.jar` that is included from the `groovy/lib` directory. I can't have those jars in the classpath; I have to use the embeddable groovy jar. I thought that the way around the issue was to edit the global "groovy-1.6.3" library. I removed all of the individual jars and just added the embeddable one. When I run the script again, the launching command line starts like this:

```
C:\Dev\java\jdk1.5.0_15\bin\java -Dgroovy.home=C:\Dev\utilities\groovy\my-groovy-1.6.3 -Dgroovy.starter.conf=C:\Dev\utilities\groovy\my-groovy-1.6.3\conf\groovy-starter.conf -Dtools.jar=C:\Dev\java\jdk1.5.0_15\lib\tools.jar -Xms128m -Xmx1124m -Didea.launcher.port=7533 "-Didea.launcher.bin.path=C:\Dev\intellij\IntelliJ IDEA 9880\bin" -Dfile.encoding=windows-1252 -classpath "C:\Dev\utilities\groovy\my-groovy-1.6.3\embeddable\groovy-all-1.6.3.jarC:\Dev\intellij\IntelliJ IDEA 9880\lib\idea_rt.jar" com.intellij.rt.execution.application.AppMain org.codehaus.groovy.tools.GroovyStarter --main groovy.ui.GroovyMain --conf C:\Dev\utilities\groovy\my-groovy-1.6.3\conf\groovy-starter.conf --classpath "C:\Dev\workspace\ie-tools-maven\core\target\test-classesC:\Dev\workspace\ie-tools-maven\core\target\classes...[other dependency jars, but no groovy-all jar]...
```

All the individual groovy-related jars are gone, but the embeddable one is not added to the `GroovyStarter` classpath. The script will not even begin running in this case. As soon as it starts to load, I get a `NoSuchMethodError` error for a different method in the `asm` library.

This time, the error comes from the Groovy internals complaining that the method can't be found. This is because the older version of the `asm` jar is loaded and it is missing a method that groovy needs. I even tried adding the `groovy-all` library directly as a dependency to the module to see if I could force it to be added to the classpath. No luck. It appears that IntelliJ will not add the `groovy-all` library to the `GroovyStarter` classpath. Even if I try to add another version of `groovy-all` (e.g. one from 1.6.0), it still won't add it to the classpath.

At this point, I got the sneaking suspicion that the included jar files are not coming from the IntelliJ global "groovy-1.6.3" library, but instead are being loaded by the settings in the `groovy-starter.conf` file. I then tried creating another `groovy-starter.conf` file with a different name and settings to load the embeddable `groovy-all` file. I passed a `-Dgroovy.starter.conf=...` option in the run config for the script to point to the new file. The option was passed through on the command line, but so was the default `-Dgroovy.starter.conf` option that pointed to the default `groovy-starter.conf` file. The `--conf`

Can I get a Groovy SDK to invoke the...

option for the GroovyStarter class didn't change; it still pointed to the default. The script failed in the same way as before.

Next I tried removing all the jars from the groovy-1.6.3/lib directory and putting in only the embeddable groovy-all directory. This worked; the script ran successfully. However, I didn't just want to muck with my main groovy setup. So I created a minimal groovy SDK setup - within a new directory, I copied over the groovy-1.6.3/conf directory. Then I created a lib directory and placed only the groovy-all jar in it. When I set the module's Groovy SDK to this new setup and ran the script, the command line looked like this:

```
C:\Dev\java\jdk1.5.0_15\bin\java -Dgroovy.home=C:\Dev\utilities\groovy\groovy-1.6.3-embed -Dgroovy.starter.conf=C:\Dev\utilities\groovy\groovy-1.6.3-embed/conf/groovy-starter.conf -Dtools.jar=C:\Dev\java\jdk1.5.0_15\lib\tools.jar -Xms128m -Xmx1124m -Didea.launcher.port=7535 "-Didea.launcher.bin.path=C:\Dev\intellij\IntelliJ IDEA 9880\bin" -Dfile.encoding=windows-1252 -classpath "C:\Dev\utilities\groovy\groovy-1.6.3-embed\lib\groovy-all-1.6.3.jarC:\Dev\intellij\IntelliJ IDEA 9880\lib\idea_rt.jar" com.intellij.rt.execution.application.AppMain org.codehaus.groovy.tools.GroovyStarter --main groovy.ui.GroovyMain --conf C:\Dev\utilities\groovy\groovy-1.6.3-embed/conf/groovy-starter.conf --classpath "C:\Dev\workspace\ie-tools-maven\core\target\test-classesC:\Dev\workspace\ie-tools-maven\core\target\classesC:\Dev\utilities\groovy\groovy-1.6.3-embed\lib...
```

The script ran successfully again. Even though the groovy-all jar does not appear on the GroovyStarter classpath (the lib directory is there instead because that's what's listed in the IntelliJ library for the SDK for some reason), I think the GroovyStarter loads the proper library anyway because the groovy-starter.conf file directs it to load all jars in the lib directory. (As an aside, I also tried changing the global library to point directly to the groovy-all jar rather than the lib directory. In that case, neither the lib directory nor the groovy-all jar was listed on the classpath for GroovyStarter, but the script still ran. I think this confirms my theory that this configuration works because GroovyStarter is loading jars based on the settings in the groovy-starter.conf file as opposed to jars listed explicitly on the classpath).

Even though that last configuration succeeded, I wasn't happy about having to make a copy of my groovy setup. That would be a pain to maintain every time I upgrade groovy.

So I tried one last test. I had a project library that pointed at the groovy-all jar in my local

Can I get a Groovy SDK to invoke the...

maven repository. I tried selecting that as the Groovy SDK for the module. Since only the jar existed in that location, there was no surrounding "Groovy SDK" directories (e.g. conf, bin, embeddable, etc). This is the command line it generated:

```
C:\Dev\java\jdk1.5.0_15\bin\java "-Dgroovy.home=C:\Documents and Settings\esword
\.m2\repository\org\codehaus\groovy\groovy-all\1.6.3" "-Dgroovy.starter.conf=C:
\Dev\intellij\IntelliJ IDEA 9880\plugins\Groovy\lib\groovy-starter.conf" -
Dtools.jar=C:\Dev\java\jdk1.5.0_15\lib\tools.jar -Xms128m -Xmx1124m -
Didea.launcher.port=7533 "-Didea.launcher.bin.path=C:\Dev\intellij\IntelliJ IDEA 9880\bin"
-Dfile.encoding=windows-1252 -classpath "C:\Dev\java\jdk1.5.0_15\jre\lib\charsets.jarC:
\Dev\java\jdk1.5.0_15\jre\lib\deploy.jarC:\Dev\java\jdk1.5.0_15\jre\lib\javaws.jar...[more
jre jars here]...C:\Dev\workspace\ie-tools-maven\core\target\classesC:\Documents
and Settings\esword\.m2\repository\org\codehaus\groovy\groovy-all\1.6.3\groovy-
all-1.6.3.jar...[all my other dependency jars]...C:\Dev\intellij\IntelliJ IDEA 9880\lib\idea_rt.jar"
com.intellij.rt.execution.application.AppMain org.codehaus.groovy.tools.GroovyStarter
--main groovy.ui.GroovyMain --conf "C:\Dev\intellij\IntelliJ IDEA 9880\plugins\Groovy
\lib\groovy-starter.conf" --classpath "C:\Dev\workspace\ie-tools-maven\core\target\test-
classesC:\Dev\workspace\ie-tools-maven\core\target\classes...[all my other dependency
jars]..."
```

Weird... This time all of the normal dependencies for my module are listed in the first classpath definition (the one for AppMain), including the groovy-all jar in my maven repository. Then GroovyStarter is listed. As with most previous tests above, the classpath for it does not contain the groovy-all jar. The groovy.starter.conf property is set to a file within IntelliJ's plugins directory. That file is the same as the default groovy-starter.conf file that ships with groovy (i.e. it tells GroovyStarter to load all jars in the "\${groovy.home}/lib" directory). Except in this case, there is no "\${groovy.home}/lib" directory. The script ran successfully with this config as well.

So I now know a couple of ways to get the embeddable version of groovy to load for running scripts. The problem is that neither of which is obvious and it took a lot of trial and error to figure out what was going on. My questions are:

1) How does IntelliJ decide what to put on the classpaths in the launch configs and how do those classpaths affect what jars are loaded when running GroovyStarter?

Can I get a Groovy SDK to invoke the...

- 2) How does IntelliJ set the groovy.starter.conf and --conf options for any particular Groovy SDK setting?
- 3) Can I change the those settings to point to another file so that I don't have to create a custom groovy SDK install to run scripts and tests.
- 4) How does IntelliJ determine what global or project libraries are listed in the "Facet Groovy" settings dialog as options for choosing a Groovy SDK? It appears that any library that contains a jar with "groovy" in the name is listed?

Thanks for any answers.

e

Tags: groovy, embeddable, groovystarter, groovy-starter



[Peter Gromov](#) 624 posts since

Sep 12, 2002 1. **Re: Can I get a Groovy SDK to invoke the embeddable groovy-all jar?** Jul 16, 2009 10:24 PM

I've created a JIRA issue:

<http://www.jetbrains.net/jira/browse/IDEADEV-38621>.

A Groovy plugin version that should fix this problem is attached there,

please check if it does the fix.

Short answers to your questions:

1) How does IntelliJ decide what to put on the classpaths in the launch configs and how do those classpaths affect what jars are loaded when running GroovyStarter?

It takes the jars from Groovy SDK library.

Can I get a Groovy SDK to invoke the...

2) How does IntelliJ set the `groovy.starter.conf` and `--conf` options for any particular Groovy SDK setting?

It deduces the `.conf` file location from the jars. If it doesn't exist,

it takes an internal `.conf` file.

3) Can I change the those settings to point to another file so that I don't have to create a custom groovy SDK install to run scripts and tests.

Currently - no. Do you really need it?

4) How does IntelliJ determine what global or project libraries are listed in the "Facet Groovy" settings dialog as options for choosing a Groovy SDK? It appears that any library that contains a jar with "groovy" in the name is listed?

You're totally right.