Is it possible to kill a Java Virtual Machine from another Virtual Machine?

Asked 16 years, 3 months ago Modified 5 years, 4 months ago

Viewed 4k times

4

I have a Java application that launches another java application. The launcher has a watchdog timer and receives periodic notifications from the second VM. However, if no notifications are received then the second virtual machine should be killed and the launcher will perform some additional clean-up activities.

The question is, is there any way to do this using only java? so far I have to use some native methods to perform this operation and it is somehow ugly.

Thanks!

java process-management

Share

Improve this question

Follow

edited Sep 25, 2008 at 22:32

asked Sep 15, 2008 at 15:09

Mario Ortegón
18.9k ● 18 ● 72 ● 81

9 Answers

Sorted by: Highest score (default)

6

I may be missing something but can't you call the `destroy()` method on the `Process` object returned by `Runtime.exec()`?

Share Improve this answer

Follow

answered Sep 15, 2008 at 15:24

David Webb
193k ● 57 ● 316 ● 302

3

You can use java.lang.Process to do what you want. Once you have created the nested process and have a reference to the Process instance, you can get references to its standard out and err streams. You can periodically monitor those, and call .destroy() if you want to close the process. The whole thing might look something like this:

```
Process nestedProcess = new ProcessBuilder("java mysub
InputStream nestedStdOut = nestedProcess.getInputStrea
know
InputStream nestedStdErr = nestedProcess.getErrorStrea
while (true) {
    /*
        TODO: read from the std out or std err (or get
way)
```

```
        Then put the real "kill-me" logic here instead
    */
    if (false) {
        nestedProcess.destroy();
        //perform post-destruction cleanup here
        return;
    }

    Thread.currentThread().sleep(1000L); //wait for a
}
```

Hope this helps,

Sean

Share Improve this answer Follow

edited Sep 15, 2008 at 15:47

answered Sep 15, 2008 at 15:42

Sean Reilly

21.8k ● 4 ● 50 ● 63

2

You could also publish a service (via burlap, hessian, etc) on the second JVM that calls System.exit() and consume it from the watchdog JVM. If you only want to shut the second JVM down when it stops sending those periodic notifications, it might not be in a state to respond to the service call.

Calling shell commands with java.lang.Runtime.exec() is probably your best bet.

Share Improve this answer

answered Sep 15, 2008 at 15:26

Follow

user5933

2

The usual way to do this is to call `Process.destroy()`... however it is an incomplete solution since when using the sun JVM on *nix destroy maps onto a SIGTERM which is not guaranteed to terminate the process (for that you need SIGKILL as well). The net result is that you can't do real process management using Java.

There are some open bugs about this issue see: link text

Share Improve this answer

Follow

edited Nov 13, 2023 at 19:27

Community Bot

1 • 1

answered Sep 17, 2008 at 8:30

Rob

31 • 2

1

OK the twist of the gist is as follows:

I was using the Process API to close the second virtual machine, but it wouldn't work.

The reason is that my second application is an Eclipse RCP Application, and I launched it using the eclipse.exe launcher included.

However, that means that the Process API destroy() method will target the eclipse.exe process. Killing this

process leaves the Java Process unscathed. So, one of my colleagues here wrote a small application that will kill the right application.

So one of the solutions to use the Process API (and remove redundant middle steps) is to get away with the Eclipse launcher, having my first virtual machine duplicate all its functionality.

I guess I will have to get to work.

Share Improve this answer

Follow

answered Sep 17, 2008 at 7:17

Mario Ortegón

18.9k ● 18 ● 72 ● 81

1

`java.lang.Process` has a `waitFor()` method to wait for a process to die, and a `destroy()` method to kill the subprocess.

Share Improve this answer

Follow

edited Aug 24, 2019 at 14:38

deHaar

18.5k ● 11 ● 43 ● 54

answered Sep 15, 2008 at 15:47

Bill Smith

0

You can have the java code detect the platform at runtime and fire off the platform's kill process command. This is really an refinement on your current solution.

There's also Process.destroy(), if you're using the ProcessBuilder API

Share Improve this answer

Follow

edited Sep 15, 2008 at 15:24

answered Sep 15, 2008 at 15:17

sblundy

61.4k • 22 • 123 • 124

0

Not exactly process management, but you could start an rmi server in the java virtual machine you are launching, and bind a remote instance with a method that does whatever cleanup required and calls System.exit(). The first vm could then call that remote method to shutdown the second vm.

Share Improve this answer

Follow

answered Sep 15, 2008 at 15:29

killdash10

716 • 5 • 12

0

You should be able to do that `java.lang.Runtime.exec` and shell commands.

Share Improve this answer

Follow

edited Aug 24, 2019 at 14:39

deHaar

18.5k • 11 • 43 • 54

answered Sep 15, 2008 at 15:14

James A. N. Stauffer
2,669 3 22 32