

Apache
CON

Apache Commons Exec

Reliably Executing External Processes

Siegfried Goeschl



Leading the Wave
of Open Source

History

- Code originated from Apache Ant
- Started 2005 by Niclas Gustavsson
- Dormant in the Commons Sandbox
- Resurrected in 2008
- In 2009 release of version 1.0
- Version 1.1 just hit the road



Common Use Cases

- ImageMagick to convert images
- Foxit Reader for printing PDFs
- WinSCP for secure file transfer



Why To Use Commons Exec

**“How to bring down
an Application
Server?!”**

**Launch an
external process!**



The Problems Ahead

- JDK provides only low-level API
 - ProcessBuilder & Runtime
 - Difficult to use
 - Error prone
- When you mess up
 - Synchronization Issues
 - Resource Depletion



Synchronization Issues

- `Process.waitFor` can wait forever
 - Launched process may never terminate
- Process output must be promptly consumed to avoid deadlock
 - Writes into a fixed-size buffer
- `Process.waitFor` does not clear thread interrupt flag when it throws `InterruptedException`



Resource Depletion Ahead

- Process resources are not automatically freed until finalization
 - Leaves `stdin`, `stdout`, `stderr` open
- `Process.destroy()` might not work
 - OpenVMS
- `Process.destroy()` doesn't kill process grandchildren depending on OS
 - Affects all Windows platforms





What is the one thing you
want your audience to
remember?!

Apache Commons Exec

`Runtime.exec()`
`ProcessBuilder.start()`



How I Became Contributor

“Siegfried, can we actually print a PDF invoice?!”

Sure!





The First Print Job

```
String line = "AcroRd32.exe /p /h " + file.getAbsolutePath();  
CommandLine commandLine = CommandLine.parse(line);  
DefaultExecutor executor = new DefaultExecutor();  
int exitValue = executor.execute(commandLine);
```

Blows up because the exit code
'1' is always returned!

Handling the Exit Value

```
String line = "AcroRd32.exe /p /h " + file.getAbsolutePath();  
CommandLine commandLine = CommandLine.parse(line);  
DefaultExecutor executor = new DefaultExecutor();  
executor.setExitValue(1);  
int exitValue = executor.execute(commandLine);
```

Got stuck when printer
was out of paper!





Tame the Runaway Process

```
String line = "AcroRd32.exe /p /h " + file.getAbsolutePath();  
CommandLine commandLine = CommandLine.parse(line);  
DefaultExecutor executor = new DefaultExecutor();  
executor.setExitValue(1);  
ExecuteWatchdog watchdog = new ExecuteWatchdog(60000);  
executor.setWatchdog(watchdog);  
int exitValue = executor.execute(commandLine);
```

Failed miserably when printing
'C:\\Document And Settings\\documents\\432432.pdf'



Quoting Is Your Friend

```
String fileName = file.getAbsolutePath();  
String line = "AcroRd32.exe /p /h \"" + fileName + "\"";  
CommandLine commandLine = CommandLine.parse(line);  
DefaultExecutor executor = new DefaultExecutor();  
executor.setExitValue(1);  
ExecuteWatchdog watchdog = new ExecuteWatchdog(60000);  
executor.setWatchdog(watchdog);  
int exitValue = executor.execute(commandLine);
```

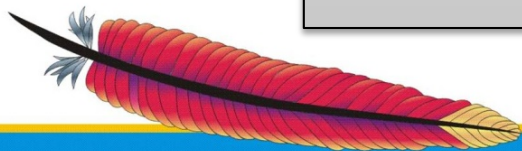
Building the command
line sucks!



Incremental Command Line

```
CommandLine cmdLine = new CommandLine("AcroRd32.exe");  
cmdLine.addArgument("/p");  
cmdLine.addArgument("/h");  
cmdLine.addArgument("${file}");  
Map map = new HashMap();  
map.put("file", new File(., "invoice.pdf"));  
cmdLine.setSubstitutionMap(map);  
DefaultExecutor executor = new DefaultExecutor();  
executor.setExitValue(1);  
ExecuteWatchdog watchdog = new ExecuteWatchdog(60000);  
executor.setWatchdog(watchdog);  
int exitValue = executor.execute(cmdLine);
```

Would be nice to print
in the background!



```
CommandLine cmdLine = new CommandLine("AcroRd32.exe");  
cmdLine.addArgument("/p");  
cmdLine.addArgument("/h");  
cmdLine.addArgument("${file}");  
Map map = new HashMap();  
map.put("file", new File("invoice.pdf"));  
commandLine.setSubstitutionMap(map);
```

```
DefaultExecuteResultHandler resultHandler =  
    new DefaultExecuteResultHandler();
```

```
ExecuteWatchdog watchdog = new ExecuteWatchdog(60*1000);  
Executor executor = new DefaultExecutor();  
executor.setExitValue(1);  
executor.setWatchdog(watchdog);  
executor.execute(cmdLine, resultHandler);
```

```
// some time later ...  
int exitValue = resultHandler.waitFor();
```




Tips and Tricks (1)

- Creating complex command line
 - `CommandLine.parse()` is fragile when mixing single & double quotes
 - Build the command line incrementally
 - You can control quoting per argument
 - Use “printargs” script for debugging

Tips and Tricks (2)

- Redirecting streams
 - Redirection is implemented by the shell
- Killing a process
 - Killing a process works (mostly)
 - Its child processes might not be killed at all depending on your OS (Windows)
 - If in doubt avoid convenience scripts to start a process





Conclusion

- Using plain Java API is error prone
 - Deadlocks
 - Resource Depletion
- Use commons-exec instead
 - Automatic stream pumping
 - Killing of run-away processes
 - Asynchronous processing



Resources

- <http://commons.apache.org/exec/>
- <http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html>
- <http://kylecartmell.com/?p=9>
- http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4890847
- http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4770092