im4java Frequently Asked Questions

Table of contents

Questions

1. General Questions

1.1. What exactly do you mean with JNI-hazard?

Running native code using JNI from within java always imposes additional risks. The JVM is no longer sandboxed, so there might be some security issues. In addition, there could be errors like memory leaks or memory corruption within the JNI-layer (JMagick) or within the native code (ImageMagick).

This is especially dangerous for long running processes (typically web-application-servers). Memory corruption or a segmentation fault (maybe triggered by a intentionally manipulated image) might bring down the whole server.

On the other side, in reality the situation is not as bad as it sounds above. JMagick is well tested, and for standard use-cases it prooves to be very stable.

Some additional information is available on the JMagick-users mailing list where this topic has been discussed to some detail.

1.2. The library does not use ImageMagicks convert command, but the one supplied by Microsoft. What can I do?

Read the Developer's Guide, section Setting up the Environment.

1.3. The program xxx supports option -foo, but im4java does not. Why? And what can I do about it?

Keeping up with actively devolped tools is not easy, im4java will always lag behind in the support of new commandline options. But adding support for a new option is straightforward:

1. Download the source-distribution of im4java.
2. Edit the file `input/XXXinterface.txt` and add your new option. The syntax of the file should be clear.
3. Run `make src jar`. That's it!
4. Post the diff of `input/XXXinterface.txt` on the jmagick-users mailing-list. I will add it to the next version of im4java.

1.4. Running my code just throws a CommandException. What can I do?

In case there is a CommandException, you should always print the associated error-text:

```
          try {
            ...
          } catch (CommandException ce) {
            ce.printStackTrace();
            ArrayList<String> cmdError = ce.getErrorText();
            for (String line:cmdError) {
              System.err.println(line);
            }
          } catch (Exception e) {
            e.printStackTrace();
          }
```

The error-text ist the stderr of the used command. Hopefully it tells you why it failed.

2. Implementation related

2.1. To be asked.

To be answered.