292129f934 ▾ ···

gradle-vagrant-plugin / src / main / groovy / com / bmuschko / gradle / vagrant / process / GDKExternalProcessExecutor.groovy

bmuschko Prep for plugin portal. History

1 contributor

67 lines (57 sloc) | 2.29 KB ···

```groovy
/*
 * Copyright 2013 the original author or authors.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.bmuschko.gradle.vagrant.process

import groovy.util.logging.Slf4j

@Slf4j
class GDKExternalProcessExecutor implements ExternalProcessExecutor {
    boolean printToConsole

    GDKExternalProcessExecutor(boolean printToConsole = true) {
        this.printToConsole = printToConsole
    }

    @Override
    ExternalProcessExecutionResult execute(List<String> commands) throws IOException {
        printCommandLineArgs(commands)
        Process process = commands.execute()
        handleProcess(process)
    }

    @Override
    ExternalProcessExecutionResult execute(List<String> commands, List envp, File dir) throws IOException {
        printCommandLineArgs(commands, envp)
        Process process = commands.execute(envp, dir)
        handleProcess(process)
    }

    private void printCommandLineArgs(List<String> commands, List envp = null) {
        log.info "Executing external command: '${commands.join(' ')}' with environment variables ${envp ?: '[]'}"
    }

    private ExternalProcessExecutionResult handleProcess(Process process) {
        def out = new StringBuilder()
        def err = new StringBuilder()

        if(printToConsole) {
            // Process.consumeProcessOutput(System.out, System.err) didn't seem to flush the output to the console on Windows
            process.in.eachLine { line ->
                out <<= line
                println line
            }

            process.err.eachLine { line ->
                err <<= line
                println line
            }
        }

        process.waitFor()
        String text = printToConsole ? (out <<= err).toString() : process.text
        new ExternalProcessExecutionResult(exitValue: process.exitValue(), text: text)
    }
}
```