

MeterMojo

```
{{package org.apache.jmeter;

import java.io.BufferedReader; import java.io.File; import java.io.FileReader; import java.io.FileWriter; import java.io.IOException; import java.lang.Thread.
UncaughtExceptionHandler; import java.security.Permission; import java.text.DateFormat; import java.text.SimpleDateFormat; import java.util.Arrays;
import java.util.Date; import java.util.List; import java.util.regex.Pattern;

import org.apache.commons.io.IOUtils; import org.apache.maven.plugin.AbstractMojo; import org.apache.maven.plugin.MojoExecutionException; import
org.apache.maven.plugin.MojoFailureException; import org.apache.tools.ant.DirectoryScanner;

/**

    • JMeter Maven plugin.
    • @author Tim McCune
    • @goal jmeter
    */
    public class JMeterMojo extends AbstractMojo {

private static final Pattern PAT_ERROR = Pattern.compile(".*\\s+ERROR
S+.*");

/**

    • @parameter
    */
    private List<String> includes;

/**

    • @parameter
    */
    private List<String> excludes;

/**

    • @parameter expression="${basedir}/src/test/jmeter"
    */
    private File srcDir;

/**

    • @parameter expression="jmeter-reports"
    */
    private File reportDir;

/**

    • @parameter expression="${basedir}/src/test/jmeter/jmeter.properties"
    */
    private File jmeterProps;

/**

    • @parameter
    */
    private boolean remote;

private File workDir;
private File saveServiceProps;
private File jmeterLog;
private DateFormat fmt = new SimpleDateFormat("yyMMdd");

/**
```

- Run all JMeter tests.

```

*/
private void executeTest(File test) throws [MojoExecutionException] {
    /... cut out from mail
    try {
        // This mess is necessary because the only way to know when JMeter
        // is done is to wait for all of the threads that it spawned to exit.
        new JMeter().start(args.toArray(new String[]{}));
        [BufferedReader] in = new [BufferedReader](new [FileReader](jmeterLog));
        while (!checkForEndOfTest(in)) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                break;
            }
        }
        in.close();
    } catch (ExitException e) {
        if (e.getCode() != 0) {
            throw new [MojoExecutionException]("Test failed", e);
        }
    } finally {
        System.setSecurityManager(oldManager);
        Thread.setDefaultUncaughtExceptionHandler(oldHandler);
    }
    } catch (IOException e) {
        throw new [MojoExecutionException]("Can't execute test", e);
    }
}

```

```

private boolean checkForEndOfTest(BufferedReader in) throws [MojoExecutionException] {
    boolean testEnded = false;
    try {
        String line;
        while ( (line = in.readLine()) != null) {
            if (line.indexOf("Test has ended") != -1) {
                testEnded = true;
                break;
            }
        }
    } catch (IOException e) {
        throw new [MojoExecutionException]("Can't read log file", e);
    }
    return testEnded;
}

```

```

private void checkForErrors() throws [MojoExecutionException], [MojoFailureException] {
    try {
        [BufferedReader] in = new [BufferedReader](new [FileReader](jmeterLog));
        String line;
        while ( (line = in.readLine()) != null) {
            if (PAT_ERROR.matcher(line).find()) {
                throw new [MojoFailureException]("There were test errors");
            }
        }
        in.close();
    } catch (IOException e) {
        throw new [MojoExecutionException]("Can't read log file", e);
    }
}

```

```

private void initSystemProps() throws [MojoExecutionException] {
    workDir = new File("target" + File.separator + "jmeter");
    workDir.mkdirs();
    createSaveServiceProps();
    jmeterLog = new File(workDir, "jmeter.log");
    try {
        System.setProperty("log_file", jmeterLog.getCanonicalPath());
    } catch (IOException e) {
        throw new [MojoExecutionException]("Can't get canonical path for log file", e);
    }
}

```

/**

- This mess is necessary because JMeter must load this info from a file.
- Resources won't work.

*/

```

private void createSaveServiceProps() throws MojoExecutionException {
    saveServiceProps = new File(workDir, "saveservice.properties");
    try {
        FileWriter out = new FileWriter(saveServiceProps);
        IOUtils.copy(Thread.currentThread().getContextClassLoader()
            .getResourceAsStream("saveservice.properties"), out);
        out.flush();
        out.close();
        System.setProperty("saveservice_properties",
            File.separator + "target" + File.separator + "jmeter" + File.separator +
            "saveservice.properties");
    } catch (IOException e) {
        throw new MojoExecutionException("Could not create temporary saveservice.properties", e);
    }
}

```

/**

- Executes a single JMeter test by building up a list of command line parameters to pass to JMeter.start().

```

*/
private void executeTest(File test) throws MojoExecutionException {
    try {
        getLog().info("Executing test: " + test.getCanonicalPath());
        String reportFileName = test.getName().substring(0,
            test.getName().lastIndexOf(".")) + "-" +
            fmt.format(new Date()) + ".xml";
        List<String> args = Arrays.asList("-n",
            "-t", test.getCanonicalPath(),
            "-l", reportDir.toString() + File.separator + reportFileName,
            "-p", jmeterProps.toString(),
            "-d", System.getProperty("user.dir"));
        if (remote) {
            args.add("-r");
        }
        // This mess is necessary because JMeter likes to use System.exit.
        // We need to trap the exit call.
        [SecurityManager] oldManager=System.getSecurityManager();
        System.setSecurityManager(new [SecurityManager]() {
            @Override public void checkExit(int status) { throw new [ExitException](status);}
            @Override public void checkPermission(Permission perm, Object context) {}
            @Override public void checkPermission(Permission perm) {}
        });
        [UncaughtExceptionHandler] oldHandler = Thread.getDefaultUncaughtExceptionHandler();
        Thread.setDefaultUncaughtExceptionHandler(new [UncaughtExceptionHandler]() {
            public void uncaughtException(Thread t, Throwable e) {
                if (e instanceof [ExitException] && (([ExitException]) e).getCode() == 0) {
                    return; //Ignore
                }
                getLog().error("Error in thread " + t.getName());
            }
        });
        try {
            // This mess is necessary because the only way to know when JMeter
            // is done is to wait for all of the threads that it spawned to exit.
            int startThreadCount = Thread.activeCount();
            new JMeter().start(args.toArray(new String[]{}));
            int activeThreadCount;
            while ( (activeThreadCount = Thread.activeCount()) > startThreadCount) {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    break;
                }
            }
            catch (ExitException e) {
                if (e.getCode() != 0) {
                    throw new MojoExecutionException("Test failed", e);
                }
            } finally {
                System.setSecurityManager(oldManager);
                Thread.setDefaultUncaughtExceptionHandler(oldHandler);
            }
            catch (IOException e) {
                throw new MojoExecutionException("Can't execute test", e);
            }
        }
    }
}

```

```
private static class ExitException extends SecurityException {  
    private static final long serialVersionUID = 5544099211927987521L;  
  
    public int _rc;  
  
    public ExitException(int rc) {  
        super(Integer.toString(rc));  
        _rc = rc;  
    }  
  
    public int getCode() {  
        return _rc;  
    }  
}  
  
}}
```