Managing Log Files

- Writing to the Correct Log
- Separate Logs of Different Origin/Plugins into Different Log Files
- Identifying App Origin in Log Files
- Large catalina.out File

Writing to the Correct Log

Let's start with the basics first. In order to write into the log files correctly, we should make use of the **LogUtil** (Source code: https://github.com/jogetworkflow/jw-community/blob/6.0-SNAPSHOT/wflow-commons/src/main/java/org/joget/commons/util/LogUtil.java) utility class.

We should not use the following to print out log in writing our own plugins.

```
System.out.println("Execution is successful");
```

This is because this line of message would appear in **catalina.out** but **not** in the default Joget's log file, **joget.log**.

Instead, we should make use of these methods provided by **LogUtil**. Check out the some sample in the codes used by **Email Tool** (Source code: https://github.com/jogetworkflow/jw-community/blob/6.0-SNAPSHOT/wflow-core/src/main/java/org/joget/apps/app/lib/EmailTool.java#L227)

```
LogUtil.info(EmailTool.class.getName(), "EmailTool: Sending email from=" + email.getFromAddress().toString() +
", to=" + to + "cc=" + cc + ", bcc=" + bcc + ", subject=" + email.getSubject());
LogUtil.info(EmailTool.class.getName(), "EmailTool: Sending email completed for subject=" + email.getSubject());
LogUtil.error(EmailTool.class.getName(), ex, "");
```

Separate Logs of Different Origin/Plugins into Different Log Files

You may have already noticed that by default, we have log file named as **email.log** as Email Tool and related plugins are writing into this specific file. We may also consider this approach in breaking down the number of lines being written into a single log file for better troubleshooting.

Navigate to the "[JogetFolder]\apache-tomcat-8.5.23\webapps\jw\WEB-INF\classes\log4j.properties" configuration file and check out the use of **R2** tag to see how **EmailTool, UserNotificationAuditTrail**, and **ExportFormEmailTool** are writing into **email.log** file.

log4j.properties

```
log4j.logger.org.joget.apps.app.lib.EmailTool=DEBUG, R2
log4j.logger.org.joget.apps.app.lib.UserNotificationAuditTrail=DEBUG, R2
log4j.logger.org.joget.plugin.enterprise.ExportFormEmailTool=DEBUG, R2

# A1 is set to be a ConsoleAppender.
log4j.appender.A1=org.apache.log4j.ConsoleAppender

# A1 uses PatternLayout.
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%-5p %d{dd MMM yyyy HH:mm:ss} %c %x - %m%n

# R is set to be DailyRollingFileAppender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=${catalina.home}/logs/joget.log
log4j.appender.R.DatePattern='.'yyyyMMdd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%-5p %d{dd MMM yyyy HH:mm:ss} %c %x - %m%n

# R2 is set to be DailyRollingFileAppender
log4j.appender.R2=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R2.File=${catalina.home}/logs/email.log
log4j.appender.R2.DatePattern='.'yyyyMMdd
log4j.appender.R2.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.R2.layout.ConversionPattern=%-5p %d{dd MMM yyyy HH:mm:ss} %-50c - %m%throwable{0}%n
```

Identifying App Origin in Log Files

In the section above, we talked about using **LogUtil** to write into the log files and how to write into separate files too. When we have too many apps running in the same copy of Joget, sometimes it is trace certain line of messages to the origin of Joget apps that trigger them.

For example, let's look at these log messages.

Sample log messages

```
ERROR 17 Jun 2019 17:29:39 org.joget.apps.app.lib.EmailTool  - org.apache.commons.mail.EmailException: Sending the email to the following server failed : smtp.outlook.comtest:587
org.apache.commons.mail.EmailException: Sending the email to the following server failed : smtp.outlook.comtest:587
        at org.apache.commons.mail.Email.sendMimeMessage(Email.java:1421)
        at org.apache.commons.mail.Email.send(Email.java:1448)
        at org.joget.apps.app.lib.EmailTool$1.run(EmailTool.java:239)
        at java.lang.Thread.run(Thread.java:748)
        at org.joget.commons.util.PluginThread.run(PluginThread.java:22)
```

There is no way that we can tell from which Joget app the EmailTool is triggered. However, in a Process Tool's execute method, we can obtain the appDef object which contains the Joget app information. Check out the sample code below.

```
public Object execute(Map properties) {

        AppDefinition appDef = (AppDefinition) properties.get("appDef");
        String appInfoAndMessage = appDef.toString() + "- Something happened";
        LogUtil.error(EmailTool.class.getName(), ex, appInfoAndMessage);

}
```

This way, we would be able to trace to the app that triggers and writes the line of message in the log file.

Where / How to obtain App Definition?

The object "appDef" is available in the following type of plugins.

- Form Post Submission Processing Tool
- Process Tool

In other plugin types, we can try to obtain the App Definition object by using the following codes.

```
import org.joget.apps.app.service.AppUtil;

AppDefinition appDef = AppUtil.getCurrentAppDefinition();
```

Large catalina.out File

We can consider to LogRotate the log files. Please see the following links:-

- https://dzone.com/articles/how-rotate-tomcat-catalinaout
- https://www.thegeekstuff.com/2010/07/logrotate-examples

As for **joget.log**, we are already using Log4J for rotation as seen in the **log4j.properties** file snippet above.