

FAQ

[1] How to do the initial setup of my logging framework

Log4j

Jakarta Commons Logging

JDK 1.4 Logging

[2] How to start Eclipse with JDK 1.4?

[3] How to export/import the Log4E configuration?

[4] I don't like the German/Chinese translation. How to get rid of it?

[1] How to do the initial setup of my logging framework

If you are already familiar with your logging framework you might want to skip this section.

This chapter gives you a slight idea of how to make the initial setup of your logger. This is NOT supported by Log4E at the moment and was not the intended use at the beginning of this project.

Log4E does not ship any logging framework which means that you have to download and install it for yourself!

Examples are available for Log4j, Commons Logging, JDK 1.4 Logging (again: you have to do this for yourself). Log4j

1. Download Log4j at <http://logging.apache.org/> and copy the log4j.jar to your lib directory (put it in your classpath).

2. Create a new file 'log4j.properties' or 'log4j.xml' (case sensitive) and put it in your classpath. To be more concrete: Put it in your source directory, the file will be copied by Eclipse automatically to your build directory.

For example: .../MyEclipseProject/
.../MyEclipseProject/src/log4j.properties
.../MyEclipseProject/src/com/mycompany/myapp/...

2a. Alternatively, you can use the method `org.apache.log4j.PropertyConfigurator.configure("C:/.../log4j.properties")`
or
`org.apache.log4j.xml.DOMConfigurator.configure("C:/.../log4j.xml");`

3. Edit the 'log4j.properties' or 'log4j.xml' to declare your own categories, log levels and appenders (which means the output like standard out or a log file).

```
log4j.properties example: #####
# Categories and levels
#####
```

```
log4j.rootCategory=ERROR, FileApp, ConApp
log4j.category.de.jayefem=DEBUG
```

```
#####
# Appenders
#####
```

```
# ConApp is set to be a ConsoleAppender.
log4j.appender.ConApp=org.apache.log4j.ConsoleAppender
# ConApp uses PatternLayout.
log4j.appender.ConApp.layout=org.apache.log4j.PatternLayout
# Define Pattern
```

```
log4j.appender.ConApp.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
```

```
# FileApp
```

```
log4j.appender.FileApp=org.apache.log4j.RollingFileAppender
log4j.appender.FileApp.File=D:/proj/Devel/Java/de.jayefem.log4e/log/log4e.log
log4j.appender.FileApp.MaxFileSize=500KB
# Keep one backup file
log4j.appender.FileApp.MaxBackupIndex=1
log4j.appender.FileApp.layout=org.apache.log4j.PatternLayout
log4j.appender.FileApp.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
```

"de.jayefem", "ConApp", "FileApp" and the path to the logfile are selfdefined. All other words are keywords of Log4j.

```
log4j.xml example:<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd" >
```

```
<log4j:configuration debug="false"
  xmlns:log4j="http://jakarta.apache.org/log4j/">

  <!-- ===== Appenders ===== -->

  <appender name="Console" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value="%d [%t] %-5p %c - %m%n" />
    </layout>
  </appender>

  <appender name="DefaultFileAppender"
    class="org.apache.log4j.RollingFileAppender">
    <!-- absolute path to log file -->
    <param name="File" value="C:/log4e.log" />
    <!-- setting this to false will cause the file to
      be truncated with each restart -->
    <param name="Append" value="true" />
    <!-- MaxFileSize -->
    <param name="MaxFileSize" value="2000KB" />
    <!-- Oldest file will be deleted if
      more than n files were generated -->
    <param name="MaxBackupIndex" value="2" />
    <!-- Layout -->
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value="%d [%t] %-5p %c - %m%n" />
    </layout>
  </appender>

  <!-- ===== Categories ===== -->

  <!-- Application category -->
  <category name="de.jayefem" additivity="true">
    <priority value="info" />
  </category>

  <!-- The root category -->
  <root>
    <priority value="warn" />
    <appender-ref ref="Console"/>
    <appender-ref ref="DefaultFileAppender" />
  </root>
</log4j:configuration>
```

4. That's all. Have fun.

Note that there are much more possibilities to configure Log4j. Check <http://logging.apache.org/> for more. Jakarta Commons Logging

Jakarta Commons Logging Framework is a wrapper for all common logging frameworks. If you want to use it, you have to install it AND the underlying logging framework. To install the Commons Logging download it from <http://jakarta.apache.org/commons/logging/> and put the commons-logging.jar in your classpath.

The Commons Logging Frameworks uses Log4j by default. When Log4j isn't found in classpath and JDK 1.4 or higher is being used, the JDK 1.4 logger will be used. If none of the above applies, Commons Logging will fall back to the internal SimpleLog.

It is also possible to specify the logging framework directly:

1. Create a new file 'commons-logging.properties' and put it in your classpath. To be more concrete: Put it in your source directory, the file will be copied by Eclipse automatically to your build directory.

For example: .../MyEclipseProject/
 .../MyEclipseProject/src/commons-logging.properties
 .../MyEclipseProject/src/com/mycompany/myapp/...

2. Edit the 'commons-logging.properties'.

```
commons-logging.properties example: #
#org.apache.commons.logging.LogFactory=
# org.apache.commons.logging.impl.LogFactoryImpl
# SimpleLog
#org.apache.commons.logging.Log = org.apache.commons.logging.impl.SimpleLog
# JDK 1.4 logger
#org.apache.commons.logging.Log=org.apache.commons.logging.impl.Jdk14Logger
# Avalon Toolkit
#org.apache.commons.logging.Log=org.apache.commons.logging.impl.LogKitLogger
# Log4j
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
```

As though it is not recommended to use Simplelog because it is not threadsafe, here's an example how to set it up:

1. Create a new file 'simplelog.properties' and put it in your classpath. To be more concrete: Put it in your source directory, the file will be copied by Eclipse automatically to your build directory.

For example: .../MyEclipseProject/
 .../MyEclipseProject/src/simplelog.properties
 .../MyEclipseProject/src/com/mycompany/myapp/...

2. Edit the 'simplelog.properties' to declare your own categories and log levels.

```
simplelog.properties example: # Default logging detail level for all instances of SimpleLog. Must be one of
# ("trace", "debug", "info", "warn", "error", or "fatal").
```

```
# If not specified,
# defaults to "info".
```

```
org.apache.commons.logging.simplelog.defaultlog=warn
```

```
# Logging detail level for a SimpleLog instance named "xxxxx". Must be one of
# ("trace", "debug", "info", "warn", "error", or "fatal").
```

```
# If not specified, the default logging detail level is used.
```

```
org.apache.commons.logging.simplelog.log.de.jayefem.log4e=debug
```

```
# Set to true if you want the Log instance name to be included in output
# messages. Defaults to false.
```

```
org.apache.commons.logging.simplelog.showlogname=false
```

```
# Set to true if you want the last component of the name to be included in
# output messages. Defaults to true.
```

```
org.apache.commons.logging.simplelog.showShortLogname=true
```

Set to true if you want the current date and time to be included in output
 # messages. Default is false.
 org.apache.commons.logging.simplelog.showdatetime=true

See <http://jakarta.apache.org/commons/logging/> for more. JDK 1.4 Logging 1. Use JDK 1.4 or higher :-)

2. Create a new file 'logging.properties'.

3. Invoke your application with:

```
java -Djava.util.logging.config.file=D:\your\path\to\logging.properties com.mycompany.myproject.MyClass
```

```
logging.properties example: # handlers
handlers=java.util.logging.FileHandler, java.util.logging.ConsoleHandler
```

```
# general level
.level=INFO
```

```
# file handler
java.util.logging.FileHandler.pattern = %h/java%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter
```

```
# console handler
java.util.logging.ConsoleHandler.level = FINEST
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
test.de.jayefem.log4e.logkits.JDK1_4_Logging.level = FINEST
```

Since I am not an expert of JDK 1.4 logging there might be better ways to configure the logging framework. Suggestions are welcome.

See <http://java.sun.com/j2se/1.4.2/docs/api/java/util/logging/package-summary.html> for more information.

[2] How to start Eclipse with JDK 1.4?

Start it with paramter '-vm'. E.g. on Windows: eclipse.exe -vm C:\j2re1.4.2_03\bin\javaw.exe

[3] How to export/import the Log4E configuration?

the template definitions are saved in:

```
<path_to>\workspace\.metadata\plugins\de.jayefem.log4e\log4e-profiles.xml
```

the preferences are saved in:

```
<path_to>\workspace\.metadata\plugins\org.eclipse.core.runtime\.settings\de.jayefem.log4e.prefs
```

<path_to> is the path where your workspace is stored. See "File->Switch Workspace..." to find out where your current workspace is.

Import:

- Shut down Eclipse
- Check if "log4e-profiles.xml" already exists. If so, you have to merge the exported file with the existing file for yourself.
- Copy "log4e-profiles.xml" and "de.jayefem.log4e.prefs" to the appropriate workspace folders

[4] I don't like the German/Chinese translation. How to get rid of it?

Start Eclipse with these parameters:

```
-nl en_US
```

E.g.: C:\eclipse\eclipse.exe -nl en_US