
第 6 章 WebSphere Product Center 记录器

WebSphere Product Center 提供了预先配置的文件，这些文件生成日志，接着，这些日志可用来对 WebSphere Product Center 中的问题进行故障诊断。本文档提供了日志记录机制的概述并说明了如何设置日志文件。

WebSphere Product Center 服务日志配置文件

下列文件控制着整个 WebSphere Product Center 中的各个子系统。生成的日志的位置是在每个文件中定义的。

注意：所有路径都是相对于 \$TOP 的。

/etc/logs/eventprocessor.log.xml

/etc/logs/scheduler.log.xml

/etc/logs/system.log.xml

/etc/logs/appsvr.log.xml

/etc/logs/workflowengine.log.xml

运行时生成的日志

可以查看运行时生成的日志以了解所发生的错误，如果问题与 WebSphere Product Center 或内部支持基础结构相关，这样做就有助于进行故障诊断。

WebSphere Product Center 生成的日志文件存储在 \$TOP/logs/*.log 中。

配置日志文件

可以根据需要来编辑 WebSphere Product Center 日志文件的属性（即，位置、最大大小和格式）。下列各节描述了一些元素，这些元素用来配置日志以及提供配置 WebSphere Product Center 日志文件时可以使用的值的列表。

更改位置

注意：仅适用于文件和滚动追加器。

要更改所生成的日志文件的位置，请更改指定的日志配置文件的参数。

例如：

```
<param name="File" value="\${TOP}/logs/webserver_db.log" />
```

更改文件大小

注意： 仅适用于滚动追加器。

可以将日志文件的大小设置为指定的存储大小，达到该大小后，日志文件将开始回绕并清除最前面的输出。要控制开始截断文件的时间，请更改日志文件大小参数值。

例如：

```
<param name="maxFileSize" value="10MB" />
```

更改文件备份选项

注意： 仅适用于滚动追加器。

可以定义记录器以便为日志文件保留指定数目的备份。在达到最大值之后，将删除最旧的文件。

例如：

```
<param name="maxBackupIndex" value="2" />
```

更改转换模式

可以通过重新定义转换模式来更改日志的布局配置。

例如：

```
<param name="ConversionPattern" value=
"%d [%t] %-5p %c (%F:%L) %x- %m%n"/>
```

转换模式与 C 语言中的 `printf` 函数的转换模式紧密相关。转换模式由文字文本和称为 *转换说明符* 的格式控制表达式组成。

注意： 可以在转换模式中任意插入任何文字文本。

转换说明符

每个转换说明符都以百分号 “%” 开头并且后跟可选的 *格式修饰符* 和 *转换字符*。

%（*格式修饰符*）（*转换字符*）

例如，

`%-5p [%t]: %m%n`

- 格式修饰符控制诸如字段宽度、填充、左对齐和右对齐之类的内容
- 转换字符指定数据的类型（例如，类别、优先级、日期和线程名）。

缺省情况下，相关信息是按原样输出的。但是，在格式修饰符的帮助下，可以更改最小字段宽度、最大字段宽度和对齐。

可选的格式修饰符被放到百分号与转换字符之间。在示例中，转换说明符 `%-5p` 表示日志记录事件的优先级应该向左对齐，并且宽度为 5 个字符。

第一个可选格式修饰符是左对齐标志，它仅仅是减号（-）字符。后面跟着的是可选的最小字段宽度修饰符。这是一个十进制常量，它表示要输出的最小字符数。如果数据项不需要那么多的字符，则在左边或右边进行填充，直到达到最小宽度为止。

缺省情况是在左边进行填充（向右对齐），但可以通过左对齐标志来指定右填充。填充字符是空格。如果数据项的长度大于最小字段宽度，则将扩充字段以容纳该数据。值不会被截断。

可以使用最大字段宽度修饰符来更改此行为，该修饰符是由句点后面跟十进制常量指定的。如果数据项的长度大于最大字段宽度，则将从数据项的开头而不是从末尾除去额外的字符。

例如，如果最大字段宽度是 8，并且数据项的长度是 10 个字符，则将删除数据项的前两个字符。

注意：此行为与 C 语言中的 `printf` 函数不同，在该函数中，截断是在末尾进行的。

下列页面提供用来定义转换说明符的值。

格式修饰符

以下是类别转换说明符的各种格式修饰符示例。

格式修饰符	向左对齐	最小值 宽度	最大宽度	注释
<code>%20c</code>	False	20	无	如果类别名的长度不足 20 个字符，则在左边填充空格。
<code>%-20c</code>	True	20	无	如果类别名的长度不足 20 个字符，则在右边填充空格。

%30c	不适用	无	30	如果类别名的长度超过 30 个字符，则从开头进行截断。
%20.30c	False	20	30	如果类别名的长度不足 20 个字符，则在左边填充空格。但是，如果类别名的长度超过 30 个字符，则从开头进行截断。
%-20.30c	True	20	30	如果类别名的长度不足 20 个字符，则在右边填充空格。但是，如果类别名的长度超过 30 个字符，则从开头进行截断。

转换字符

以下是识别的转换字符的列表：

转换字符	作用
c	<p>用来输出日志记录事件的类别。（可选）在类别转换说明符后面可以跟随着精度说明符，后者是括在花括号中的十进制常量。</p> <p>如果给出了精度说明符，则将只打印类别名最右边的相应数目的组件。缺省情况下，将打印整个类别名。</p> <p>例如，对于类别名 “a.b.c”，模式 %c{2} 将输出 “b.c”。</p>
d	<p>用来输出日志记录事件的日期。在日期转换说明符后面可以跟随着括在花括号内的日期格式说明符。</p> <p>例如，%d{HH:mm:ss,SSS} 或 %d{dd MMM yyyy HH:mm:ss,SSS}。如果未给出日期格式说明符，则采用 ISO8601 格式。</p> <p>日期格式说明符允许使用与 SimpleDateFormat 的时间模式字符串相同的语法。尽管是标准 JDK 的部件，但 SimpleDateFormat 的性能相当不好。</p> <p>要获得较好的结果，建议使用 log4j 日期格式化符。可以使用字符串 “ABSOLUTE”、“DATE” 和 “ISO8601”（它们分别指定 AbsoluteDateFormat、DateTimeDateFormat 和 ISO8601DateFormat）中的一个字符串来指定这些日期格式化符。例如，%d</p>

	{ISO8601} 或 %d{ABSOLUTE}。 这些指示的日期格式化符的执行效果显著优于 SimpleDateFormat。
m	用来输出 WebSphere Product Center 提供的与日志记录事件相关联的消息。
n	输出一个或多个依赖于平台的行分隔符。 此转换字符实际上提供了与使用不可移植行分隔字符串（如 “\n” 或 “\r\n”）相同的性能。因此，这是指定行分隔符的首选方法。
p	用来输出日志记录事件的优先级。
r	用来输出从启动 WebSphere Product Center 开始到创建日志记录事件为止经过的毫秒数。
t	用来输出生成了日志记录事件的线程的名称。
x	用来输出与生成了日志记录事件的线程相关联的 NDC（嵌套的诊断上下文）。
%	序列 %% 输出单个百分号。

WebSphere Product Center 记录设置文件

下列示例演示 WebSphere Product Center 的日志文件的定义方式。粗体条目设置 WebSphere Product Center 日志文件的配置。

```
<!-- basic ASYNC appender -->
<appender name="ASYNC" class="org.apache.log4j.AsyncAppender">
<appender-ref ref="DEFAULT"/>
</appender>
```

```
<!-- basic CONSOLE appender. This is the same as doing system.out-->
<appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value=

 "[%t] %-5p %c (%F:%L) %x- %m%n"/>
</layout>
</appender>
```

```
<!-- simple FILE appender. The file will be opened and if append is true --
>
```

```

<!--          it will not be truncated          -->
<appender name="DEFAULT" class="org.apache.log4j.FileAppender">
  <param name="File" value="${TOP}/logs/tomcat_default.log " />
  <param name="Append" value="true" />
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value=

"%d [%t] %-5p %c (%F:%L) %x- %m%n"/>
  </layout>
</appender>

<!-- Rolling FILE appender. The file will be opened and if append is true --
>
<!--          it will not be truncated          -->
<!--          maxFileSize: How big before you rotate      -->
<!--          maxBackupIndex: How many backups do you keep? -->
<appender name="DB" class="org.apache.log4j.RollingFileAppender">
  <param name="File" value="${TOP}/logs/tomcat_db.log " />
  <param name="Append" value="true" />
  <param name="maxFileSize" value="10MB" />
  <param name="maxBackupIndex" value="2" />
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value=

"%d [%t] %-5p %c (%F:%L) %x- %m%n"/>
  </layout>
</appender>

<!-- For the austin.db category, you want to have only a few logs kept hence
-->
<!-- the rollingappender -->
<category name="austin.db" additivity=" false">
  <priority value="INFO" />
  <appender-ref ref="DB" />
</category>

<!-- ROOT CATEGORY -->
<!-- MUST ALWAYS BE LAST ENTRY AND HAVE AN APPENDER--
>
<!-- If a logging event is not caught by any other logger it will be handled
by this-->
<!-- rule. -->
<root>
  <priority value="error"/>
  <appender-ref ref="DEFAULT"/>
</root>

</log4j:configuration>

```