Апендеры в Log4j

JSP & Servlets
5 уровень, 5 лекция

ОТКРЫТА

Список апендеров

Логгер позволяет сохранять данные сразу в несколько файлов. Такие выходные потоки данных называются аппендерами (от append). Существует довольно много стандартных апендеров, поэтому мы рассмотрим только самые популярные из них:

	Апендеры	**Описание**
1	Console	Выводит данные в консоль
2	File	Выводит данные в файл
3	DailyRollingFile	Выводит данные в файл, файл циклически перезаписывается
4	Async	Позволяет писать данные в другой аппендер асинхронно
5	Socket	Пишет данные в определенный сокет
6	JDBC	Пишет сообщения в базу по протоколу JDBC
7	JPA	Пишет сообщения в базу по протоколу JPA
8	HTTP	Шлет события по HTTP-протоколу на удаленный сервер
9	SMTP	Складирует сообщения в буфер, а потом шлет в виде email’а

Хорошая документация по все апендерам есть на их официальном сайте

А ниже мы рассмотрим самые популярные и простые из них

ConsoleAppender

Самый простой аппендер – это `ConsoleApender`. Как вы уже догадались, он пишет свои сообщения просто в консоль. У него есть аж несколько интересных нам параметров:

	Атрибуты	
1	name	Имя аппендера
2	filter	Позволяет отфильтровать часть сообщений
3	layout	Задает форматирование сообщений при выводе
4	target	Задает, куда пишем: `SYTEM_OUT` или `SYSTEM_ERR`

Сконфигурировать его очень просто:

```
<?xml version="1.0" encoding="UTF-8"?>
    <Configuration status="warn" name="MyApp" packages="">
    <Appenders>
        <Console name="STDOUT" target="SYSTEM_OUT">
        <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>
      <Loggers>
        <Root level="error">
        <AppenderRef ref="STDOUT"/>
        </Root>
    </Loggers>
    </Configuration>
```

FileAppender

Самый полезный апендер – это `FileAppender`. В отличии от `ConsoleAppender` он пишет свои сообщения в файл. Что очень полезно, когда ваше приложение работает где-нибудь на сервере. У него куча параметров, т.к. он должен уметь писать файлы в разных операционных системах.

Но мы рассмотрим только самые популярные из них

1	name	Задает имя аппендера
2	filter	Позволяет отфильтровать часть сообщений
3	layout	Задает форматирование сообщений при выводе
4	fileName	Задает имя файла, куда писать сообщения
5	append	Если `true`, то сообщения будут дописаны в старый лог, если `false` – лог-файл будет пересоздаваться каждый раз при запуске приложения.
6	bufferSize	Задает размер буфера в байтах
7	immediateFlush	Если `true`, то каждое сообщение сразу реально пишется на диск (без буфера). Лог начинает работать медленно, но это спасает от потери данных при падении программы.

Вы уже умеете хорошо работать с файлами, так что в этих настройках для вас ничего нового. Сконфигурировать такой логгер даже проще чем консольный:

```
<?xml version="1.0" encoding="UTF-8"?>
    <Configuration status="warn" name="MyApp" packages="">
    <Appenders>
        <File name="MyFile" fileName="logs/app.log">
        <PatternLayout>
            <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
        </PatternLayout>
        </File>
    </Appenders>
    <Loggers>
        <Root level="error">
        <AppenderRef ref="MyFile"/>
        </Root>

```

```
15      </Loggers>
        </Configuration>
```

RollingFileAppender

Самый популярный аппендер – это `RollingFileAppender`. В отличии от `FileAppender` он позволяет разбить лог на много маленьких файлов. Это очень актуально для больших логов. Кроме того, он позволяет задать правила, что делать со старыми файлами после того, как начали писаться новые.

А у этого аппендера почти под сотню различных настроек. Подробнее с ними вы можете ознакомиться по ссылке

Рассмотрим самые популярные атрибуты этого аппендера:

	Атрибуты	
1	name	Задает имя аппендера
2	filter	Позволяет отфильтровать часть сообщений
3	layout	Задает форматирование сообщений при выводе
4	fileName	Задает имя файла, куда писать сообщения
5	filePattern	Задает шаблон имен для архивных файлов, которые больше не пишутся
6	policy	Задает условие, когда файл должен начать перезаписываться
7	strategy	Описывает, что делать со старыми файлами: архивировать, историю за сколько дней хранить и т.п.

Вот хороший пример:

```
1   <Configuration status="warn" name="MyApp" packages="">
2     <Appenders>
3       <RollingFile name="RollingFile" fileName="logs/app.log" filePattern="logs/app-%d{MM-dd-yyyy}-%i.l
4       <PatternLayout>
5           <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
6       </PatternLayout>
7       <Policies>
8           <TimeBasedTriggeringPolicy />
9           <SizeBasedTriggeringPolicy size="250 MB"/>
10      </Policies>
11      </RollingFile>
12    </Appenders>
13    <Loggers>
14      <Root level="error">
15      <AppenderRef ref="RollingFile"/>
16      </Root>
17    </Loggers>
18  </Configuration>
```

С помощью параметра `filePattern` задан шаблон для архивов старых логов. Так же тут есть два при срабатывании которых начнется писаться новый файл:

- `TimeBasedTriggeringPolicy` – сработает, если начнутся новые сутки (поменяется текущая дата)
- `SizeBasedTriggeringPolicy` – сработает, если размер файла достигнет 250Мб

Предыдущая лекция
Следующая лекция >

+9

Комментарии популярные новые старые

JavaCoder

Введите текст комментария

У ЭТОЙ СТРАНИЦЫ ЕЩЕ НЕТ НИ ОДНОГО КОММЕНТАРИЯ

проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store