

Настройка уровней логирования

JSP & Servlets
5 уровень, 3 лекция

ОТКРЫТА

4.1 Список уровней логирования

Вы написали свою программу, залили ее на сервер и тут у вас сразу начинают появляться вопросы:

- Как сделать так, чтобы `debug()` метод не отработывал во время работы в `production`?
- В логах слишком много информации, хотелось бы оставить только сообщения об ошибках?
- Как увидеть детальный лог по одной части приложения?

Разумеется, создатели логов столкнулись с тем же самым еще десятки лет назад. Не буду рассказывать, как эту проблему решили в языке C, но в языке Java ее решили очень красиво.

Лог фильтрует данные, перед тем как записывать информацию в файл. Можно очень быстро снизить/увеличить детализацию лога с помощью настройки уровня логирования. Эти уровни описаны в таблице ниже:

	Уровень	Примечание
1	ALL	Писать в лог все сообщения
2	TRACE	Мелкое сообщение при отладке
3	DEBUG	Сообщения важные при отладке
4	INFO	Простые сообщения
5	WARN	Писать только fatal, error и warning
6	ERROR	Писать только ошибки и фатальные ошибки
7	FATAL	Писать только фатальные ошибки
8	OFF	Не писать в лог сообщения

Эти уровни используются при фильтрации сообщений. Если выставить уровень логирования в `WARN`, то все сообщения, менее важные, чем `WARN` будут отброшены: `TRACE`, `DEBUG`, `INFO`. Если выставить уровень фильтрации в `FATAL`, то будут отброшены даже `ERROR`’ы.

Есть еще два уровня важности, которые используются при фильтрации – это `OFF` (отбросить все сообщения) и `ALL` – писать все сообщения (не отбрасывать ничего).

4.2 Пример настройки лога

Давайте рассмотрим простой пример настройки лога. Для этого нам понадобится файл `log4j.properties`, который можно разместить в папке `resources`. Добавим в него такое содержание:

1	# Root logger option
2	log4j.rootLogger= WARN , stdout
3	

```
4 # Direct log messages to stdout
5 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
6 log4j.appender.stdout.Target=System.out
7 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
8 log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}
```

Вот тут в самой первой строке мы и задаем уровень логирования – **WARN**. А это значит, что сообщения, которые пишутся в логгер со статусом **DEBUG** и **INFO** будут проигнорированы.

- Указываем какой тип аппендера будем использовать – **ConsoleAppender**
- Указываем, куда будем писать лог – **System.out**
- Задаем класс, который будет управлять форматом записи – **PatternLayout**
- Задаем формат записи для всех сообщений – дата и время

4.3 Популярные ошибки в логировании

И еще один важный момент – популярные ошибки в логировании. Вариантом что-то сделать не так много, но можно выделить несколько частых ошибок:

1. **Избыток логирования.** Не стоит логировать каждый шаг, который чисто теоретически может быть важным. Есть правило: **логи могут нагружать работоспособность не более, чем на 10%**. Иначе будут проблемы с производительностью.
2. **Логирование всех данных в один файл.** Это приведет к тому, что в определенный момент чтение/запись в него будет очень сложной, не говоря о том, что есть ограничения по размеру файлов в определенных системах.
3. **Использование неверных уровней логирования.** У каждого уровня логирования есть четкие границы, и их стоит соблюдать. Если граница расплывчатая, можно договориться какой из уровней использовать.

< Предыдущая лекция

Следующая лекция >



Комментарии (3)

популярные

новые

старые

JavaCoder

Введите текст комментария

Сергей Уровень 65, Воронеж, Россия

4 октября, 21:12 ⋮

Оставляю [эту ссылку](#) здесь, где всё максимально понятно. Много примеров, настроек и т.д. В статье как я понимаю используется устаревшая версия log4j, ссылка ведёт на актуальную версию 2.x. Соответственно пример в статье уже не актуален.

Ответить

👍 +1 🧡

Andrey Panchenko Моет полы в Яндекс

15 сентября, 15:37 ⋮

Чувак [написал](#) более практично, если кто хочет попробовать в деле.

Ответить

👍 0 🧡

Алексей Уровень 79, Москва

26 июля, 21:32 ⋮

Маловато информации про log4j.properties, примеров бы побольше с тем как и где его размещать.

Ответить

👍 +11 🧡

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные 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

