Аникин Д.А. Логирование для отладки и профилирования JAVA-приложений // Международный журнал информационных технологий и энергоэффективности. – 2023. – Т. 8 № 5(31) ч.2. с. 86–91

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.432.2

ЛОГИРОВАНИЕ ДЛЯ ОТЛАДКИ И ПРОФИЛИРОВАНИЯ JAVA-ПРИЛОЖЕНИЙ

Аникин Д.А.
ФГБУО ВО «МИРЭА - Российский технологический университет», Москва, Россия (119454, г. Москва, пр. Вернадского, 78), e-mail: danil-anikin-98@mail.ru

Настоящая статья посвящена обзору инструментов логирования Java-приложений и рассмотрению методов отладки и профилирования с помощью рассмотренных технологий. В ходе разработки программ необходимо иметь инструмент, благодаря которому проводится отслеживание состояний пользовательских сессий во время работы приложения, автоматизированные инструменты управления логами направлены на решение указанной проблемы. В результате их анализа были выявлены общие принципы и руководства по реализации логирования в разрабатываемых программах.

Ключевые слова: Информатика, логирование, джава, программирование, отладка и профилирование.

LOGGING FOR DEBUGGING AND PROFILING JAVA APPLICATIONS

Anikin D.A.
MIREA - Russian Technological University, Moscow, Russia (119454, Moscow, Vernadskogo Ave., 78), e-mail: danil-anikin-98@mail.ru

This article is devoted to an overview of Java application logging tools and a review of debugging and profiling methods using the technologies considered. During the development of programs, it is necessary to have a tool through which the states of user sessions are monitored during the operation of the application, automated log management tools are aimed at solving this problem. As a result of their analysis, general principles and guidelines for the implementation of logging in the developed programs were identified.

Keywords: Computer science, logging, java, programming, debugging and profiling.

Разработка программных приложений для различных средств электронно-вычислительной техники – это огромная часть современной жизни и тренды развития науки и общества явно указывают на то, что область действия информационных технологий будет продолжать расширять собственную зону влияния. При разработке приложений разработчики сталкиваются с проблемой выявления ошибок и отладки кода, далеко не всегда программисты безошибочно реализуют необходимую бизнес-логику и в ходе работы возникают ситуации, которые не были предусмотрены изначальным замыслом. В такой момент к обязанностям разработчика добавляется отслеживание выполнения кода и поиск ошибок, допущенных при написании, определению, что именно и в какой момент привело к некорректному поведению программы [5].

В данной статье будут рассмотрены принципы использования логирования для отладки и профилирования на примере Java-приложений, а также разобраны различные фреймворки

и инструменты, предназначенные для решения задачи мониторинга и анализа исполнения приложений. Кроме того, будут рассмотрены преимущества и недостатки логирования при отладке Java-приложений и будут представлены рекомендации, основанные на опыте использования различных фреймворков. Улучшение производительности разработчиков является одной из важнейших задач в производстве программного обеспечения, и в этой статье будут представлены рекомендации для быстрой и эффективной разработки Java-приложений.

Как уже было установлено, в задачи разработчика входит не только написание кода, но и отслеживание его выполнения и поиск ошибок. Одним из наиболее популярных и распространённых решений этой проблемы является логирование. Логирование - это процесс ведения журнала событий, которые происходят в компьютерной системе. Логи – это отдельные файлы, которые и составляют журнал логирования [1]. Они должны быть структурированы и подходить под единый шаблон, выведенный в рамках разработки приложения.

При обработке пользовательских сессий может возникнуть множество различных ситуаций, каждая из которых должна быть записана в журнал логов. Так как журнал является тем средством, которое впоследствии будет проанализировано программистом с целью проверки корректности работы или поиска ошибки, то одним из важнейших факторов является читаемость и удобство восприятия логов. Во всех высоконагруженных системах с большим количеством параллельных сессий лог-файлы будут стремительно наполняться большим количеством записей, анализ которых в скором времени станет очень трудной задачей. С целью повышения читаемости и контроля количества сохраняемой информации была разработана концепция уровней логирования.

Уровни логирования явным образом указывают на тип, который записывается в лог-файл. Каждый уровень следует принимать только в подходящем контексте, в противном случае это может привести к еще более запутанному наполнению журнала [3]. Иерархия уровней представлена на Рисунке 1.

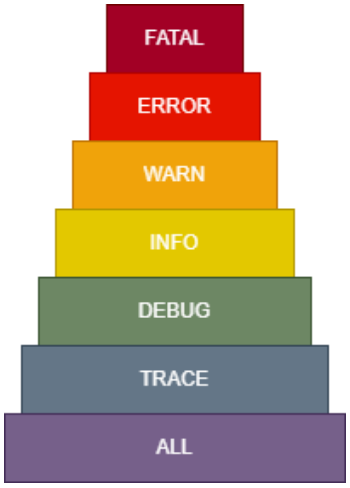

Рисунок 1 – Иерархия уровней логирования

ALL: включает в себя все уровни логирования. Он может быть полезен при обработке большого объёма данных, когда необходимо получить полную картину действий приложения.

TRACE: представляет наиболее подробную информацию о работе программы, которая может быть полезна во время разработки и отладки. Могут быть записаны все состояния объектов, значения переменных, ошибки и другие данные, которые помогут разработчику понять, что происходит во время исполнения.

DEBUG: уровень для информации, которая пригодится только при отладке программы. Этот уровень используется для записи дополнительных данных, которые помогут разработчику разобраться в процессе выполнения программы. Это может быть информация о значениях переменных, результатах выполнения условных операторов, вызовах функций и т.д.

INFO: уровень для информации, которая может помочь при анализе работы программы и выявлении её проблем. Этот уровень используется для сообщений о промежуточных результатах работ, в том числе о запуске и остановке, успешных операциях, вызове функций и методов, а также о состоянии программы в момент её выполнения.

WARNING: уровень для сообщений об ошибках или непредвиденных ситуациях, которые были обработаны программой. Этот уровень используется для сообщений о незначительных проблемах, которые не привели к аварийной остановке, но требуют внимания и устранения.

ERROR: уровень для сообщений об ошибках, которые не позволяют продолжить работу программы. Этот уровень используется для сообщений о критических сбоях, которые приводят к завершению работы программы.

FATAL: самый высокий уровень, он используется для сообщений о критических ошибках, которые могут привести к потере данных, повреждению системы или другим серьезным последствиям.

Java - один из самых популярных языков программирования в мире, который имеет широкую функциональность и позволяет создавать мультиплатформенное программное обеспечение. Именно на его примере будет рассмотрено практическое применение и реализация различных фреймворков для ведения логов.

Логирование в Java-приложениях имеет свои преимущества и недостатки [2]. Среди основных достоинств можно выделить:

- Диагностика и устранение проблем. Запись событий в лог-файлы позволяет разработчикам узнать, что происходило в приложении во время возникновения ошибок или других проблем. Это помогает быстрее и эффективнее диагностировать и устранять ошибки.
- Мониторинг и анализ работы приложения. Лог-файлы могут содержать информацию о работе приложения, такую как пропущенные элементы, время ответа и т.д., которая может быть очень полезна при анализе работы приложения.
- Обнаружение угроз безопасности. Логирование приложений может обнаружить запуск несанкционированных процессов или другие аномалии, которые могут свидетельствовать о попытке взлома приложения.
- Улучшение пользовательского опыта. Логирование приложений может помочь разработчикам понять, как пользователи используют приложение, и определить, какие части наиболее часто вызывают проблемы или создают неудобства.

Однако использование логов часто чревато следующими негативными последствиями:

- Снижение производительности. Запись большого количества данных в лог-файлы может повысить нагрузку на приложение, что может привести к снижению его производительности.
- Непоследовательность. Логирование приложений может быть непоследовательным и зависеть от настроек или ошибок конфигурации, что может затруднить понимание того, что происходит в приложении.
- Угроза конфиденциальности. Лог-файлы могут содержать конфиденциальную информацию о пользователях, такую как: логины, пароли, данные банковских карт и т.д. Если эти данные попадут в чужие руки, это может привести к серьезным последствиям.
- Переполнение хранилища данных. Если приложение не управляет размером лог-файлов или их количеством, то это может привести к переполнению хранилища данных, что в свою очередь может замедлить работу приложения.

Поэтому важно проанализировать потребности разработки и выбрать подходящий фреймворк или инструмент для логирования. Для автоматизации процесса разработки системы логирования Java предоставляет различные фреймворки, в которых реализован практически весь функционал, необходимый для составления и обработки log-файлов. Наиболее популярными являются:

JUL (Java Util Logging) - стандартный механизм логирования для Java. JUL обладает всеми базовыми функциями и позволяет записывать логи в файл или консоль. Однако он не позволяет настраивать уровни логирования в реальном времени.

Log4j - одна из самых популярных библиотек для логирования Java приложений. Позволяет выбрать уровень логирования, настроить различный вывод записей, начиная с консольного и файлового, заканчивая системами управления базами данных. Она используется во множестве проектов, однако на данный момент является устаревшей.

Log4j2 - обновленная версия Log4j с расширенным функционалом. Она была переработана с целью обеспечения большей производительности и поддержки асинхронного логирования, а также для достижения более простой конфигурации и использования.

Logback - обладает большими возможностями для настройки и управления логированием, так что её можно настроить для разных типов приложений и сред. Она также поддерживает асинхронное логирование, администрирование через JMX и расширенные функции форматирования.

SLF4J (Simple Logging Facade for Java) - это фреймворк для протоколирования, который является абстракцией над такими библиотеками, как Log4j, Logback и JUL. SLF4J представляет собой обобщенный интерфейс для различных систем и не зависит от конкретной реализации [4].

Рассмотренные библиотеки позволяют снизить влияние технических недостатков записи логов, однако без грамотного управления ими любая система в конечном счете придёт к неорганизованным и перегруженным журналам. Для сохранения порядка следует придерживаться лучших практик организации логирования [7]:

- Запись в журнале лога должна чётко отражать реальную ситуацию и иметь соответствующий ей уровень. Не следует рядовые исключения обработки содержимого формы помечать уровнем ERROR, также как и нежелательно помечать критические ошибки уровнем WARN.

- При разработке веб-приложений следует логировать входящий http-запрос и исходящий htttp-ответ, это позволит определить, что в возникновении исключительной ситуации виновен сервис, который обрабатывает запрос [6].
- Не стоит записывать все события и все промежуточные состояния в журнал, так как на операции чтения и записи затрачиваются вычислительные ресурсы машины, это приводит к понижению скорости работы системы, а файлы увеличиваются в размерах непропорционально полезной информации, которая представлена в них.
- Стоит записывать сообщения, напрямую связанные с исполнением бизнес-логики приложения. Это позволяет отследить корректность решения поставленной задачи.
- Использовать уникальные идентификаторы пользовательских сессий, запросов, это позволяет упростить поиск и фильтрацию записей.
- Использовать понятный и легко интерпретируемый формат. Стандартом отрасли на данный момент является JSON-строка, которая позволяет хранить в себе состояние возможных объектов. Читаемый вид позволяет упростить анализ и восприятие журнала.
- Не записывать конфиденциальную информацию. Утечка персональных данных пользователя может привести к сильному репутационному удару компании или же привести к прямому урону жизни пользователей, чьи данные были потеряны.

В результате анализа концепции логирования и рассмотрения готовых фреймворков для её реализации были выявлены советы по их использованию и организации ведения логов в разрабатываемом приложении. Указанные практики позволяют обеспечить эффективное логирование и анализ работы приложений, повысить безопасность, стабильность и производительность системы. Были рассмотрены как преимущества, так и недостатки, но, несмотря на них, сохранение истории работы приложения является необходимым средством, без которого невозможно дальнейшее развитие информационных систем.

Список литературы

1. Anton, Chuvakin Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management / Chuvakin Anton, Schmidt Kevin. — Amsterdam. — Netherlands : , 2012. — 413 с. — Текст : непосредственный.
2. Java Logging Technology // Java documentation URL: https://docs.oracle.com/javase/8/docs/technotes/guides/logging/index.html (дата обращения: 10.05.2023).
3. Introduction to java logging // Baeldung URL: https://www.baeldung.com/java-logging-intro (дата обращения: 10.05.2023).
4. SLF4J Documentation // slf4j URL: https://www.slf4j.org/docs.html (дата обращения: 10.05.2023).
5. Peters, T. (1993). The history and development of transaction log analysis. Library Hi Tech., 42(11), 41−66
6. Robert, C, Martin Clean Code / C, Martin Robert. — Massachusetts : Courier in Stoughton, 2009. — 462 с. — Текст : непосредственный.
7. Robert, C,Martin Clean Architecture / C,Martin Robert. — Massachusetts : Courier in Stoughton, 2018. — 429 с. — Текст : непосредственный.

Аникин Д.А. Логирование для отладки и профилирования JAVA-приложений // Международный журнал информационных технологий и энергоэффективности. – 2023. – Т. 8 № 5(31) ч.2. с. 86–91

References

1. Anton, Chuvakin Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management / Chuvakin Anton, Schmidt Kevin. — Amsterdam. — Netherlands : , 2012. — p. 413— Text: direct.
2. Java Logging Technology // Java documentation URL: https://docs.oracle.com/javase/8/docs/technotes/guides/logging/index.html (дата обращения: 10.05.2023).
3. Introduction to java logging // Baeldung URL: https://www.baeldung.com/java-logging-intro (accessed on: 10.05.2023).
4. SLF4J Documentation // slf4j URL: https://www.slf4j.org/docs.html (accessed on: 10.05.2023).
5. Peters, T. (1993). The history and development of transaction log analysis. Library Hi Tech., 42(11), 41−66
6. Robert, C, Martin Clean Code / C, Martin Robert. — Massachusetts : Courier in Stoughton, 2009. — p. 462— Text: immediate.
7. Robert, C,Martin Clean Architecture / C,Martin Robert. — Massachusetts : Courier in Stoughton, 2018. — p. 429— Text: immediate.