СБЕРБАНК ТЕХНОЛОГИИ

Логгеры

ОБЩИЕ СВЕДЕНИЯ О ЛОГИРОВАНИИ

СБЕРБАНК ТЕХНОЛОГИИ

При создании даже небольших программ встает вопрос логирования (logging), позволяющий выводить сообщения о ходе работы в различные источники (консоль, файл и т.д.)

Имя логгера может выбираться произвольно. Чаще всего имя базируется на имени пакета и класса, для которого ведется протокол.

Информация из логов может быть использована:

- **Разработчиками** – для отслеживания стека исполнения программы, поиска ошибок, в качестве помощника при отладке;
- **Системными администраторами** – для получения информации об ошибках конфигурации, проблем в области безопасности
- **Пользователями** – для получения уведомлений из системы

ОБЩИЕ СВЕДЕНИЯ О ЛОГИРОВАНИИ

СБЕРБАНК ТЕХНОЛОГИИ

Процесс логирования:

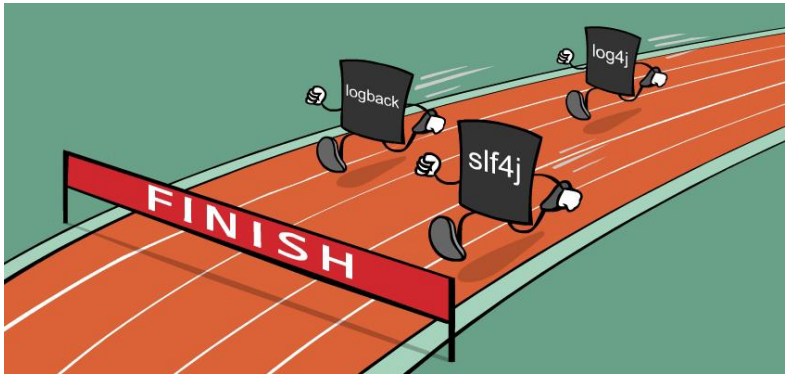
1. Сбор информации
2. Фильтрование собранной информации
3. Запись отобранной информации

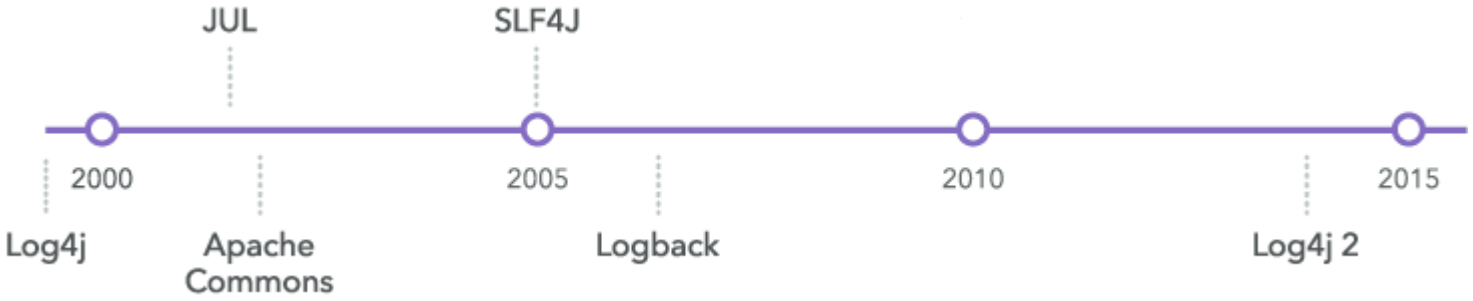
Уровни логирования – механизм, позволяющий контролировать протоколируемые сообщения. Образуют иерархию сообщений.

РЕАЛИЗАЦИИ ЗАДАЧИ ЛОГИРОВАНИЯ

СБЕРБАНК ТЕХНОЛОГИИ

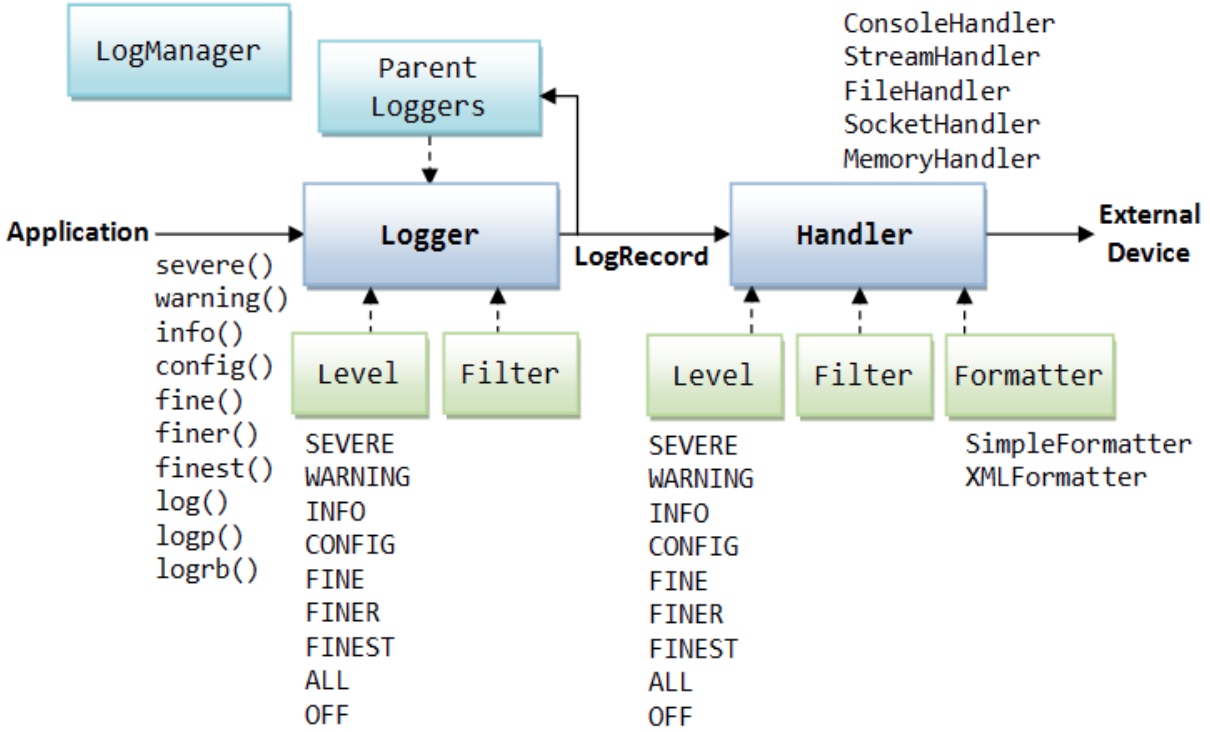
- JUL java.util.logging
- Log4j и log4j2
- JCL (Jakarta Commons Logging)
- Logback
- SLF4J (Simple Logging Facade for Java)





СБЕРБАНК ТЕХНОЛОГИИ

Начиная с JDK 1.4 Java поддерживает собственную систему логирования, которая является частью платформы (2001г)



JUL JAVA.UTIL.LOGGING

СБЕРБАНК ТЕХНОЛОГИИ

Плюсы

- Отлаженность (базируется на уже существовавших фреймворках)
- Входит в стандартный пакет, не требует подключения внешних библиотек
- Доступность для понимания

Минусы

- Неразвитость и неудобство решения, редкость обновлений
- Количество уровней логирования
- Не позволяет использовать фреймворки, на базе которых создана, в качестве своей реализации

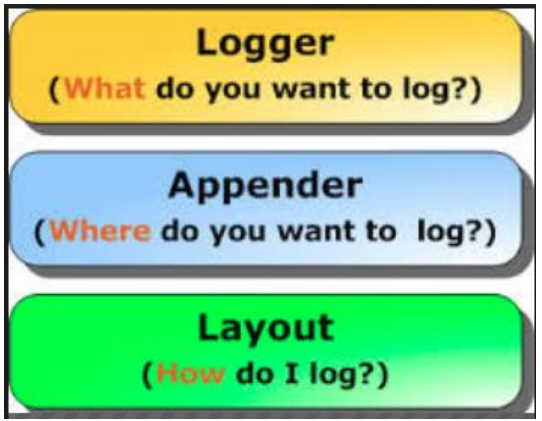
LOG4J. ОБЩИЕ СВЕДЕНИЯ

СБЕРБАНК ТЕХНОЛОГИИ

Библиотека появилась в 1999г

Разделение понятий:

- **логгера**,
- записи в лог (которую осуществляют **appenders**),
- Форматирования/компоновки записей (**layout**).



Конфигурация log4j определяет, какие appenders к каким категориям прикрепляются и сообщения какого уровня (log level) попадают в каждый **appender**.

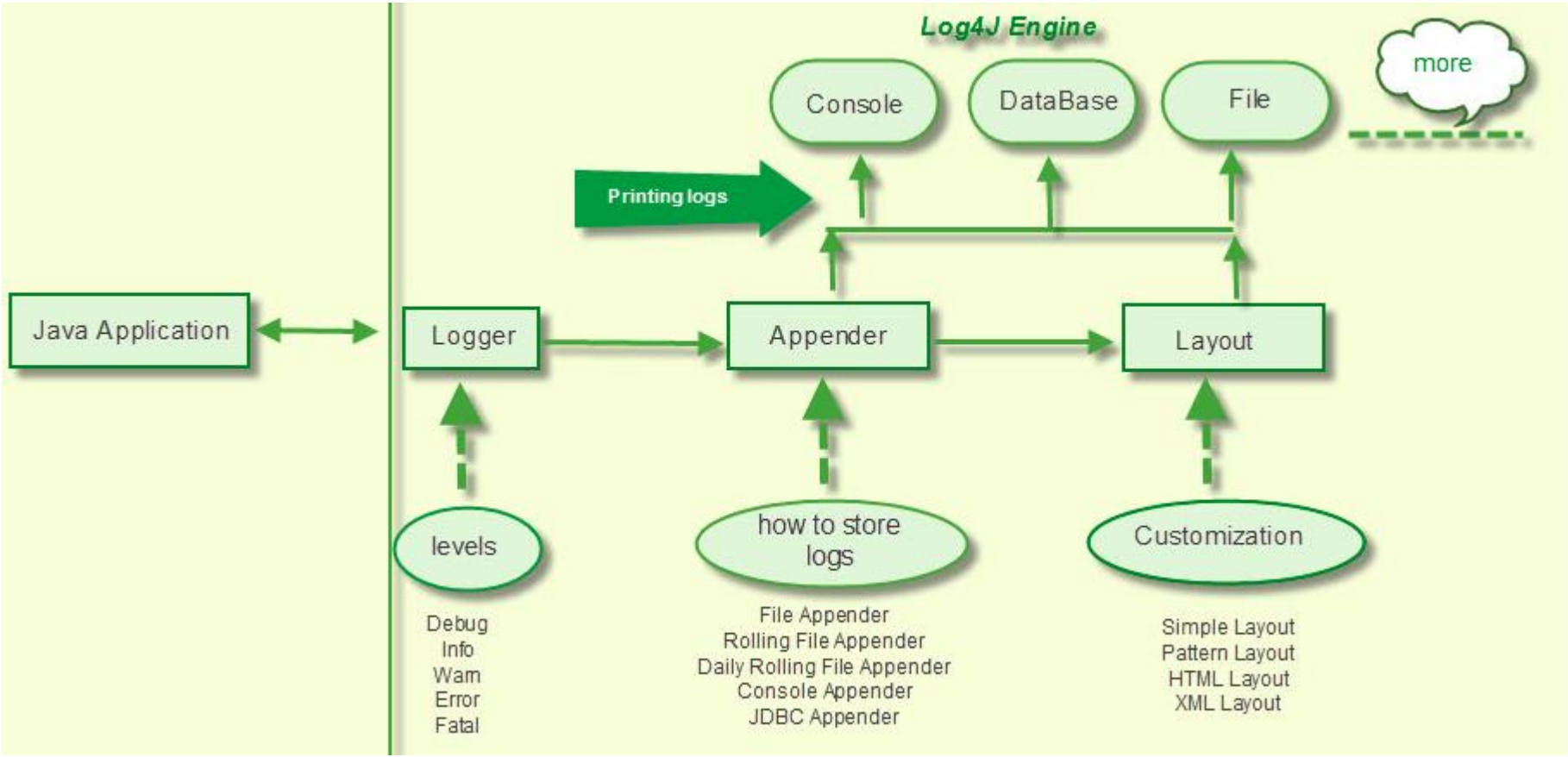
LOG4J. УРОВНИ ЛОГИРОВАНИЯ

СБЕРБАНК ТЕХНОЛОГИИ

Степень важности	Описание
ALL	Все сообщения
TRACE	Мелкое сообщение при отладке
DEBUG	Сообщения, важные при отладке
INFO	Информационное сообщение о том, что происходит в приложении
WARN	Предупреждение о возникновении нежелательной ситуации
ERROR	Ошибка, при которой приложение способно работать
FATAL	Фатальная ошибка
OFF	Нет сообщения

LOG4J. КОМПОНЕНТЫ

СБЕРБАНК ТЕХНОЛОГИИ



СБЕРБАНК ТЕХНОЛОГИИ

Плюсы

- Зрелый фреймворк с устоявшимися принципами и понятиями, удачной архитектурой
- Поддерживает несколько способов конфигурации
- Позволяет управлять своим поведением во время исполнения

Минусы

- Трудности при первом знакомстве, сложности с пониманием работы
- Необходимость подключения сторонней библиотеки
- Возможные проблемы с загрузчиком классов

SIMPLE LOGGING FACADE FOR JAVA

СБЕРБАНК ТЕХНОЛОГИИ

SLF4J - абстрагирующий фреймворк логирования. Поддерживает (абстрагирует) большое количество фреймворков логирования - **java.util.logging, Log4J, Commons Logging, Logback**.

Структура библиотеки:

- Общая часть библиотеки, API
- Набор модулей, реализующих схемы «передачи» логов – SLF4J >> Log4J, SLF4J >> Commons Logging, SLF4J >> java.util.logging, SLF4J >> NOP, SLF4J >> Simple.

SLF4J

СБЕРБАНК ТЕХНОЛОГИИ

Плюсы

- Универсальная качественно проработанная «обертка» над логерами
- Поддержка параметризованных сообщений
- Отсутствие проблем с динамическим связыванием и classloader’ом
- Поддерживает контексты NDC и MDC

Минусы

- Необходимость подключения сторонней библиотеки (и адаптеров)

- Взаимодействие с логгером осуществляется через API предоставляемый оберткой SLF4J.
- Logback концептуально является наследником Log4J. Уровни совпадают с log4j.
- Основными элементы конфигурации: Logger, Appenders, Layouts и Filters.

Примеры нововведений: зависимость **уровня логирования** от определенных параметров, что позволяет воспроизводить ошибки в режиме промышленной эксплуатации, выставляя уровень детального логирования не всему приложению, а только действиям, совершаемым указанным пользователем. *Подробнее далее…*

ПОЧЕМУ СТОИТ ВЫБРАТЬ LOGBACK?

СБЕРБАНК ТЕХНОЛОГИИ

- более **быстрая** реализация в сравнении с log4j
- **нативная** интеграция с Slf4j **нивелирует оверхед** от обертки
- конфигурирование как с помощью **XML** так и на **Groovy**
- продвинутое **восстановление** после сбоев IO-операций
- автоматическое **удаление** и **сжатие** архивных файлов.
- автоматическое **обновление в рантайме** при изменении файла конфигурации

и т.д.

APPENDER

СБЕРБАНК ТЕХНОЛОГИИ

Logback делегирует процедуру **записи лога** компонентам, называемым **appenders** (реализует интерфейс ch.qos.logback.core.Appender)

- ConsoleAppender
- FileAppender
- RollingFileAppender – TimeBasedRollingPolicy / SizeBasedTriggeringPolicy/
- ServerSocketAppender and SSLServerSocketAppender
- SMTPAppender
- DBAppender
- SiftingAppender
- AsyncAppender
- и другие + возможность написать свой Appender

ENCODER

СБЕРБАНК ТЕХНОЛОГИИ

Кодеры отвечают за преобразование события в массив байтов, а также за выписку этого байтового массива в OutputStream.

- **PatternLayoutEncoder -** форматирует строку согласно паттерн-лэйауту.

LAYOUT

СБЕРБАНК ТЕХНОЛОГИИ

Отвечают за форматирование строки

- **PatternLayout**

```
%d %-5level [%thread] %logger{0}: %msg%n
```

  - можно задать в паттерне общеиспользуемые значения (дата и время, уровень логирования, название треда и т.п.),
  - можно ограничить размер блока сообщения – как минимум, так и максимум
  - Вычисление выражений, группировка, подсветка текста и т. п.

- **HTMLLayout** –

  выводит логи в html формате

RelativeTime	Thread	MDC	Level	Logger	Message
0	main		INFO	main.apollo.13.Launcher	Preparing for takeoff
0	main		DEBUG	main.apollo.13.Launcher	Engines ready
0	main		DEBUG	main.apollo.13.Launcher	T minus 10 and counting
0	main		DEBUG	main.apollo.13.Launcher	10..9..8..7..6..5..4..3..2
0	main		DEBUG	main.apollo.13.Launcher	Ignition
0	main		ERROR	main.apollo.13.Shuttle	Houston, we have a problem

```
apollo.shuttle.ShuttleException: oops...
    apollo.shuttle.Shuttle.oops(Shuttle.java:12)
    main.ShuttleManager.main(ShuttleManager.java:33)
```

FILTERS

СБЕРБАНК ТЕХНОЛОГИИ

Позволяют разными способами отсеивать логи по определенным признакам

- **LevelFilter и ThresholdFilter** – по уровню логирования (WARN|INFO|DEBUG etc.).
- **EvaluatorFilter** – проверяет выполение условия

  <expression>e.level.toInt() >= WARN.toInt() && !(e.mdc?.get("req.userAgent") =~ /Googlebot|msnbot|Yahoo/) </expression>
- **Matchers** – проверка по регуляркам
- **DuplicateMessageFilter –** проверяет сообщения-дубли
- и др.

MAPPED DIAGNOSTIC CONTEXT

СБЕРБАНК ТЕХНОЛОГИИ

Одной из целей logback является аудит и отладка сложных распределенных приложений. Большинство распределенных систем реального мира имеют дело с несколькими клиентами одновременно.В типичной многопоточной реализации такой системы разные потоки будут обрабатывать разные клиенты.

MDC - отладочный контекст, в рамках которого производится запись диагностической информации. Например, данные о клиенте или среде выполнения. Информацию из MDC можно использовать в Layout, Appender или Filter.

РАЗДЕЛЕНИЕ ЛОГОВ

СБЕРБАНК ТЕХНОЛОГИИ

Самый простой способ – по выборке контекста (обычно используют имена классов, включая пакеты).

Продвинутый способ, разделение по переменным значениям, например, из MDC с помощью **SiftingAppender**.