В процессе функционирования сложных приложений необходимо вести журнал сообщений и ошибок, чтобы была возможность отследить время входа и выхода пользователя из системы, возникновение исключительных ситуаций, сбоев и т. д. Существуют различные API регистрации сообщений и ошибок, среди которых на данный момент можно выделить следующие. 1. Log4J.

Apache Log4J был первым регистратором. Изначально обладает ка- чественной архитектурой, в следствие чего быстро занял доминирующие позиции и применяется в большинстве промышленных приложений. Разработан в рамках проекта Jakarta Apache.

В итоге logger регистрирует и направляет вывод события в пункт назначе- ния, определяемый направляющим элементом, в формате, заданном формати- рующим элементом.

Log4j

В современном практическом программировании представляет основной инструмент журналирования событий. Формирует журнал сообщений (отла- дочных, информационных, системных, security, сообщений об ошибках). Log4j Log4J 767 можно загрузить по адресу: http://logging.apache.org/log4j/. Перед использовани- ем необходимо зарегистрировать библиотеку log4j-[версия].jar в приложении.

Logger Основным элементом API регистрации событий и ошибок является регистратор org.apache.log4j.Logger, который управляет регистрацией сооб- щений. Вывод регистратора может быть направлен на консоль, в файл, базу данных, GUI-компонент или сокет. Это компонент приложения, принимающий и выполняющий запросы на запись в регистрационный журнал. Apache Log4J поддерживает несколько способов конфигурации. Позволяет управлять своим поведением во время исполнения. Каждый класс приложения может иметь свой собственный logger или быть прикреплен к общему для всего приложения. Регистраторы образуют иерар- хию, как и пакеты Java. Каждый логгер имеет имя, описывающее иерархию, к которой он принадлежит. Разделителем является точка. Принцип полностью аналогичен формированию имени пакета в Java. Регистратор может быть создан или получен с помощью статического мето- да getLogger(String name) или getLogger(Class name), где name — имя пакета или класса. На вершине иерархии находится корневой регистратор. Он всегда существует и у него нет имени. Ссылку на корневой регистратор можно полу- чить статическим методом getRootLogger(). У каждого регистратора есть уровень сообщения по возрастанию (TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF), который управляет выводом сообщений. Для вывода сообщений конкретного уровня используются методы trace(), debug(), info(), warn(), error(), fatal(). Чтобы вывести информацию о возникшем исключении в качестве второго параметра, в перечисленные методы нужно передать объект класса, производного от Throwable. Для вывода сообще- ния необходимо, чтобы уровень выводимого сообщения был не ниже, чем уровень регистратора (TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF), т. е. если уровень регистратора INFO, то вызов logger.debug("message") не даст никакого эффекта, т. к. DEBUG < INFO. Уровень регистратора можно указать с помощью метода setLevel(Level level), который принимает объект класса Level, содержащий одноименные константы для каждого уровня. Если уровень регистратора не указывается, то применяется уровень, унаследованный от его родителя. Уровень корневого регистратора DEBUG. Таким образом, сообще- ния, выводимые с уровнем ниже установленного, в лог не попадут. И в этом заключается основное преимущество — можно вставлять в программный код вывод информации на различных уровнях (об ошибках — на уровне ERROR, о нормальном ходе выполнения — на уровне INFO, отладочную — на уровне DEBUG), а потом гибко регулировать, что именно будет выводиться. Некоторые общие методы для вывода сообщений: ПРИЛОЖЕНИЕ 2 768 log(Priority priority, Object message, Throwable t) — выводит сообщения указанного уровня с информацией об исключительной ситуации t; log(Priority priority, Object message)  — выводит сообщения указанного уровня.