# Logging

Voronezh, 2018

T · · Systems

# Agenda

- Why logging?
- Frameworks
- Architecture
- Configuration
- Debug Level
- Appenders
- Layout

T · · Systems

# Why logging?

- A well-written logging code offers:
  - quick debugging,
  - easy maintenance,
  - and structured storage of an application's runtime information.

- Logging does have its drawbacks also:
  - It can slow down an application.
  - If too verbose, it can cause scrolling blindness.

# Logging Frameworks

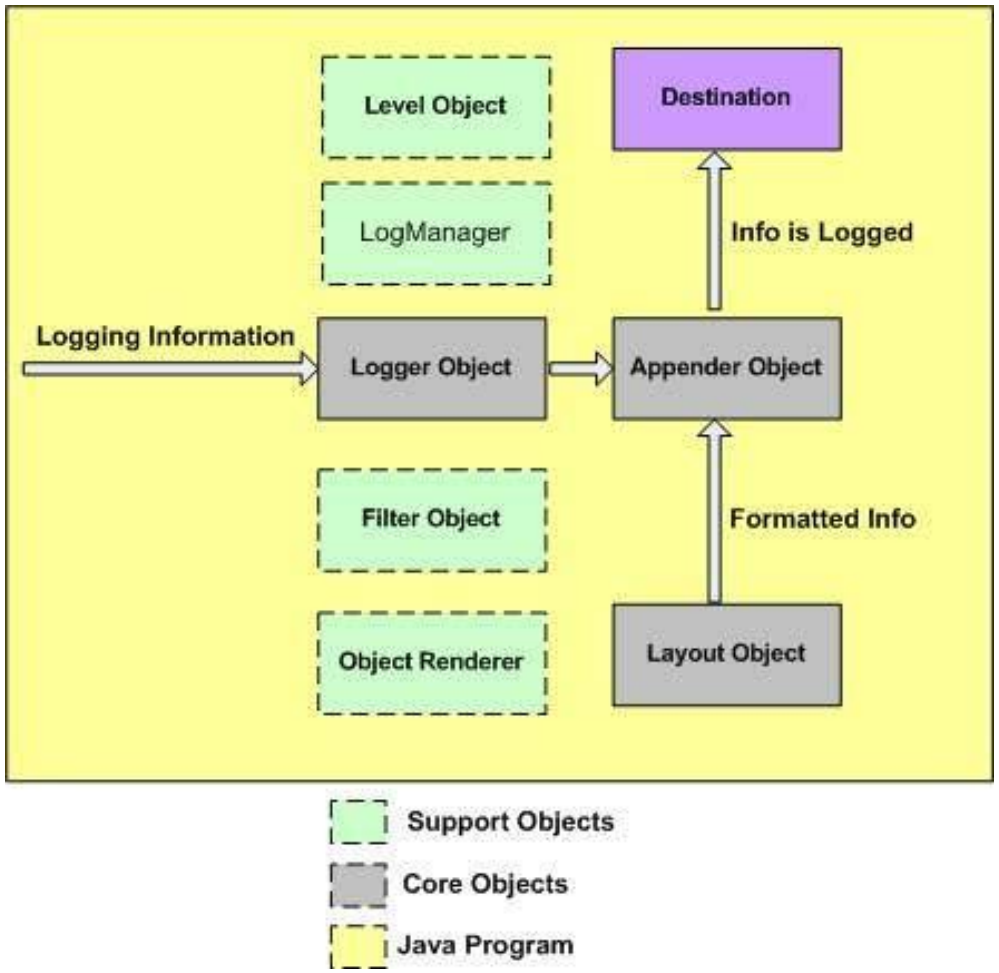
- **log4j** - используют подсевшие на него изначально и не видящие необходимости перехода
- **commons-logging** - обычно задействован в legacy-библиотеках, которые очень боятся причинить неудобства пользователем, переехав на что-нибудь получше
- **SLF4J** (Simple Logging Facade for Java) - очень популярен в библиотеках. Многие переехали на него, не выдержав ужасов commons-logging
- **Logback** - обычно современные high-performance серверы, которых не устраивает log4j
- **JUL** - тихо умирающий стандарт. Все, кто изначально пытался его использовать, переезжают на Logback

# Log4j

- **Log4j** is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License.

- Log4j has three main components:
  - **loggers**: Responsible for capturing logging information.
  - **appenders**: Responsible for publishing logging information to various preferred destinations.
  - **layouts**: Responsible for formatting logging information in different styles.

# Log4j Architecture




T · · Systems

# Configuration

- The **log4j.properties** file is a log4j configuration file which keeps properties in key-value pairs.
- By default, the LogManager looks for a file named **log4j.properties** in the **CLASSPATH**.
- To **configure logging**:
  - The level of the root logger is defined as **DEBUG**. The **DEBUG** attaches the appender named X to it.
  - Set the appender named X to be a valid appender.
  - Set the layout for the appender X.

# log4j.properties

```
# Define the root logger with appender X
log4j.rootLogger = DEBUG, X

# Set the appender named X to be a File appender
log4j.appender.X=org.apache.log4j.FileAppender

# Define the layout for X appender
log4j.appender.X.layout=org.apache.log4j.PatternLayout
log4j.appender.X.layout.conversionPattern=%m%n
```

T · · Systems

# Debug Level (1)

- **FATAL** - очень критичная ошибка в приложении, приложение не может работать после данной ошибки, реакция на сообщение должна быть максимально быстрой.

- **SERVE** - критичная ошибка в приложении, данная ситуация является внештатной, но приложение может работать дальше, реакция на сообщение должна быть максимально быстрой.

- **ERROR** - ошибка в приложении, приложение может работать дальше без всяких проблем, возможно проблема с неправильными входными данными или доступом к внешним сервисам (БД, legacy), данная ошибка предусматривалась при разработке.

- **WARN** - некритичная ошибка, приложение может работать дальше без всяких проблем, данная ошибка предусматривалась при разработке, возможно одна из функций приложения дала сбой, который можно исправить.

# Debug Level (2)

- **INFO** - важная информация о работе приложения, например запуск остановка приложения или использование конфигурационных файлов или аутентификация пользователя в системе.

- **DEBUG** - отладочная информация работы приложения, например техническая информация полученная при работе в внешними системами, или информация о вызове методов объектов, со списком параметров.

- **TRACE** - трассировка выполнения приложения, например информация о вызываемых методах и времени их работы, также на данном уровне пишется информация о времени вызова внешних сервисов (БД, legacy).

# Appenders (1)

- **ConsoleAppender** - наиболее часто используемый во время разработки аппендер. Выводит сообщения на консоль.

- **FileAppender** - просто записывает логируемые сообщения в файл. К недостаткам этого аппендера следует отнести то что размер файла лога постоянно растёт и может поучиться один огромный файл.

- **DailyRollingFileAppender** - тоже записывает сообщения в файл но каждый день создаёт новый файл с таким же именем.

- **RollingFileAppender** - этот аппендер тоже записывает сообщения в файл. и для создаёт новые файлы. но не каждый день как предыдущий а при достижении опередлённого размера (по умолчнию 10 МБ), старые файлы переименовывает - добавляет к имени файла индекс;1, 2, 3 и т.д. Максимальный размер индекса задаётся настройкой maxBackupIndex. При достижении инднекса maxBackupIndex старые файлы перетираются новыми. Таким образом размер логов можно строго ограничить.

# Appenders (1)

- **SMTPAppender** - посылает сообщения по электронной почте.
- **NTEventLogAppender-** Пишет логи в виндовый журнал. обязательно положите NTEventLogAppender.dll в директорию перечисленную в PATH.
- **SyslogAppender** пишет логи в Syslog - такие логи широко испольуется в IP сетях. Стандарт для Unix и Linix систем.
- **JDBCAppender** записывает сообщения в БД через jdbc. Уже буферезирован. Как написано в javadoc этот аппендер могут полностью заменить в следующих версия log4j.
- **LF5Appender** записывает сообщения на ui консоль написанную на Swing.

T · · Systems

# Layout

- DateLayout
- HTMLLayout
- PatternLayout
- SimpleLayout
- XMLLayout

# PatternLayout Example (1)

- **log4j.properties**:

```
# Define the root logger with appender file
log = /usr/home/log4j
log4j.rootLogger = DEBUG, FILE

# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

# Define the layout for file appender
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern=%d{yyyy-MM-dd}-%t-%x-%-5p-%-10c:%m%n
```

# PatternLayout Example (2)

- **Log4jExample.class**:

```
import org.apache.log4j.Logger;

public class Log4jExample{
    static Logger LOG =
         Logger.getLogger(Log4jExample.class.getName());

    public static void main(String[] args)throws Exception{
       LOG.debug("Hello this is an debug message");
       LOG.info("Hello this is an info message");
    }
}
```

# PatternLayout Example (3)

- **/usr/home/log4j/log.out**:

```
2010-03-23-main--DEBUG-Log4jExample:Hello this is an debug message
2010-03-23-main--INFO-Log4jExample:Hello this is an info message
```

Спасибо за внимание!

T · · Systems