Java Avançado

Log em Aplicações
com SLF4J e Backlog

SoftBlue
www.softblue.com.br

Todos os direitos de cópia reservados. Não é permitida a distribuição física ou eletrônica deste material sem a permissão expressa e por escrito do autor.

Tópicos Abordados

- O que é logging
- Por que usar uma API
- APIs de logging para Java
- SLF4J e Backlog
  - Configuração
  - Loggers
  - Log Levels
  - Appenders
  - Layouts

Logging

- Utilizado para mostrar mensagens e rastrear problemas
- Pessoas usam mesmo sem perceber
  - *System.out*
  - *System.err*

Por que usar uma API de logging?

- Usar *System.out* e *System.err* traz uma série de desvantagens
  - Falta de flexibilidade
    - Não é possível alterar o comportamento do logging sem mexer no código
  - Mensagens sempre no mesmo destino
  - Não há como desativar o logging
    - "Tudo ou nada"
  - Não há formatação especial

APIs de Logging para Java

- Java Logging API (java.util.logging)
  - Incluída no Java desde a versão 1.4
  - Exige esforço de programação para customizá-la
- Apache Commons Logging (JCL)
  - É uma fachada para outras APIs
  - Possibilita transparência ao migrar para diferentes APIs de logging
- Log4j
  - API muito difundida
  - Permite bastante flexibilidade

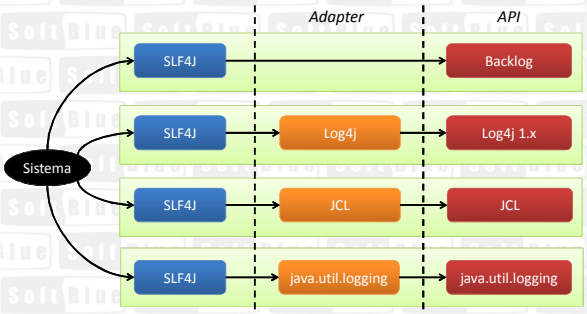
Log4j, SLF4J e Backlog

- Apesar de ser muito utilizada, o desenvolvimento do Log4j 1.x desacelerou
- Isto abriu brechas para a criação do SLF4J
  - O SLF4J usa os mesmos conceitos do Log4j
  - Resolve uma série de problemas do Log4j
- O SLF4J é apenas uma fachada
  - É preciso usar alguma API de logging integrada com o SLF4J
  - A API Backlog é a que oferece a melhor performance
- O Log4j 2.x é uma iniciativa de tentar ganhar mercado novamente

SLF4J

- **S**imple **L**ogging **F**acade for **J**ava
- O SFL4J é uma API que vem ganhando mercado muito rapidamente
  - Muitos frameworks usam SLF4J internamente
- Site oficial
  - http://www.slf4j.org
- Precisa trabalhar em conjunto com uma API de logging

Funcionamento do SLF4J



Funcionamento do SLF4J

- O SLF4J requer alguns JARs no classpath da aplicação para funcionar
  - JAR do SLF4J
  - JAR do adapter (caso necessário)
  - JAR(s) da API de logging a ser utilizada em conjunto com o SLF4J
- Para determinar qual API de log será utilizada, basta adicionar o JAR no classpath

Funcionamento do Logback

- Para usar o Logback, basta adicionar os JARs necessários no classpath da aplicação
  - O uso do logback integrado com o SLF4J requer que o JAR do SLF4J também esteja no classpath
- O Logback pode ser configurado através do arquivo **logback.xml**

Loggers

- São os objetos utilizados quando há a necessidade de gerar uma informação de log
- Um logger é uma fonte de mensagens de log

```
public class TesteLog {
    private static final Logger LOGGER =
        LoggerFactory.getLogger(LogTest.class);

    public static void main(String[] args) {
        LOGGER.info("Mensagem de log!");
    }
}
```

Um logger pode ser também uma string → "br.com.softblue.LogTest"

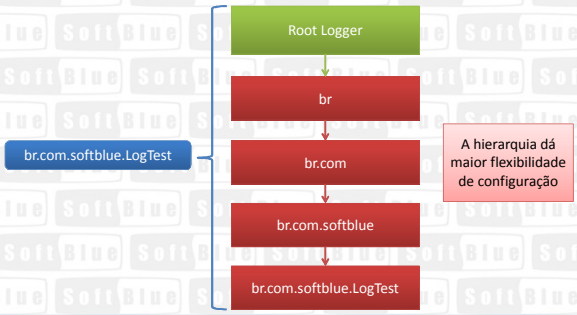
Gerando mensagens de log

- O objeto *Logger* possui métodos para gerar os mais diversos tipos de mensagens
  - **trace()**, **debug()**, **info()**, **warn()** e **error()**

```
LOGGER.info("Mensagem de log");
LOGGER.debug("O usuário {} conectou", "XYZ");
LOGGER.debug("Tentativa {} de {}", 1, 3);
```

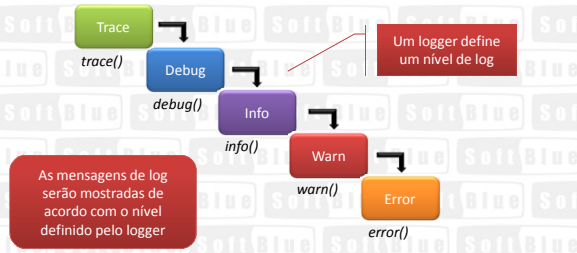
Hierarquia dos Loggers

- Os loggers funcionam de forma hierárquica

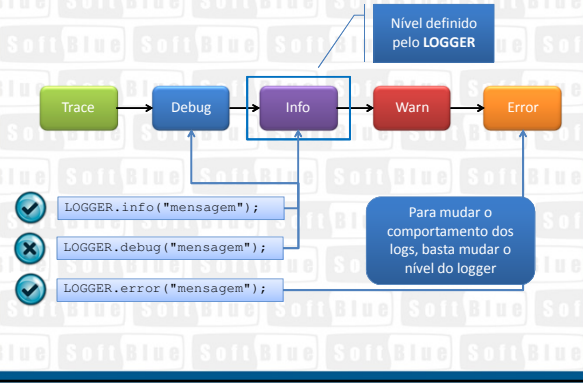


Log Levels

- Existem vários níveis de log, que funcionam de acordo com uma hierarquia



Log Levels



Appenders

- Representam o destino das mensagens de log
- Um logger pode ter um ou mais appenders associados
- Alguns appenders importantes:

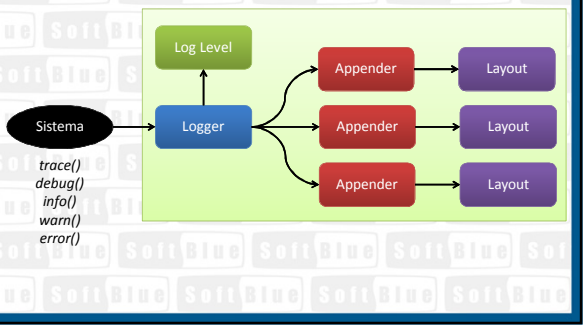
Appender	Descrição
ConsoleAppender	Direciona os logs para o console
FileAppender	Direciona os logs para um arquivo
RollingFileAppender	Direciona os logs para um arquivo e permite definir políticas sobre tamanho máximo do arquivo, número de arquivos de backup, etc.

Layouts

- Formatam os dados dos logs
- Um appender deve ter um layout associado
- Alguns appenders importantes:

Layouts	Descrição
PatternLayout	Formata a saída com base em um padrão de conversão
HTMLLayout	Formata a saída em tabela HTML
XMLLayout	Formata a saída em XML

Juntando as Partes



Colocando em Prática...

Agora que você já aprendeu a teoria, acesse as vídeo-aulas práticas e pratique os assuntos abordados neste módulo!

Clique aqui para acessar as vídeo-aulas práticas