Logging

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Apresentação
- O que é logging
- APIs de logging para Java
- SL4J e Logback
- Laboratório

Apresentação

- Tenho 37 anos, Bagé/RS
- Analista de Tecnologia da Informação - UNIPAMPA
- Desenvolvedor Full Stack
- Bacharel em Sistemas da Informação - URCAMP
- Especialização em Engenharia de Sistemas - ESAB
- Mestrado e Doutorado em Computação - UFPEL
- Esposo da Daniela, e Pai do Miguel e Murilo
- Gosto de pesquisar, correr, ciclismo e pescaria
- Linguagens favoritas: Java, Python, C++, SQL, PLSQL, JPQL

Agora é a vez de vocês, se apresentem!

- Nome
- Idade
- Estado/Cidade
- Stack Favorita (Front, Back, Full Stack e Linguagem)
- Hobby favorito

Considerações Iniciais

- Nosso horário é das 19:00 às 22:00 **(10 min tolerância de chegada)**
- Intervalo às 20:20 de **20 minutos**
- Interaja na aula!
- Se tiver dúvidas, levante a mão no chat que explicarei novamente
- Façam as atividades práticas
- Qualquer coisa me procurem no **slack**
- Me corrija!

O que é Logging

- Utilizado para mostrar mensagens e rastrear problemas
- Usamos sem perceber, o mais comum é com System.out e System.err

Por que usar uma API de log?

- **Usar System.out e System.err traz um série de desvantagens**
  - Falta de flexibilidade: não é possível alterar o comportamento do logging
  - Mensagens sempre no mesmo destino
  - Não há como desativar o logging
  - Não existe uma formatação especial

APIs de Logging para Java

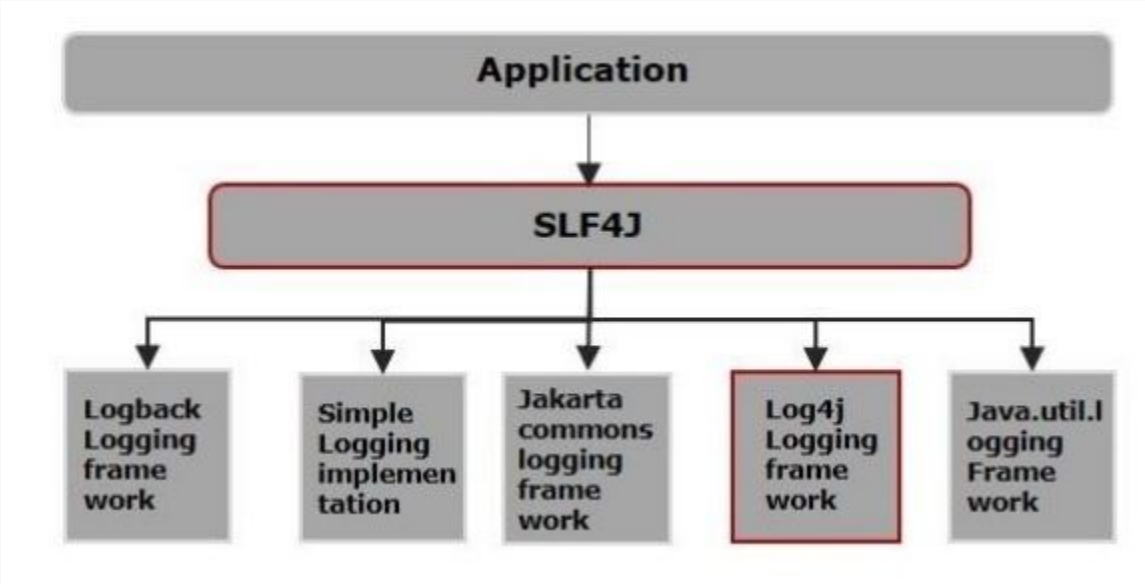
- **Java Logging API (java.util.logging)**
  - Incluída no Java desde a versão 1.4
  - Exige esforço de programação para customizá-la
- **Apache Commons Logging (JCL)**
  - É uma fachada para outras APIs
  - Possibilita transparência ao migrar para diferentes APIs de logging

APIs de Logging para Java

- **Log4J**
  - API muito difundida
  - Permite bastante flexibilidade
- **SL4J**
  - Utiliza os mesmos conceitos de Log4J
  - É preciso usar alguma API de logging integrada com o SL4J

SL4J

- **Simple Logging Facade for Java**



Funcionamento do SL4J

- **O SLF4J requer 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 SL4J
  - 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 Logback pode ser configurado através do arquivo logback.xml
- No spring boot crie o arquivo logback-spring.xml

Loggers

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

```
@Service
public class CrudFuncionarioService {


    private static final Logger LOGGER = LoggerFactory.getLogger(CrudFuncionarioService.class);
```

Gerando mensagens de log

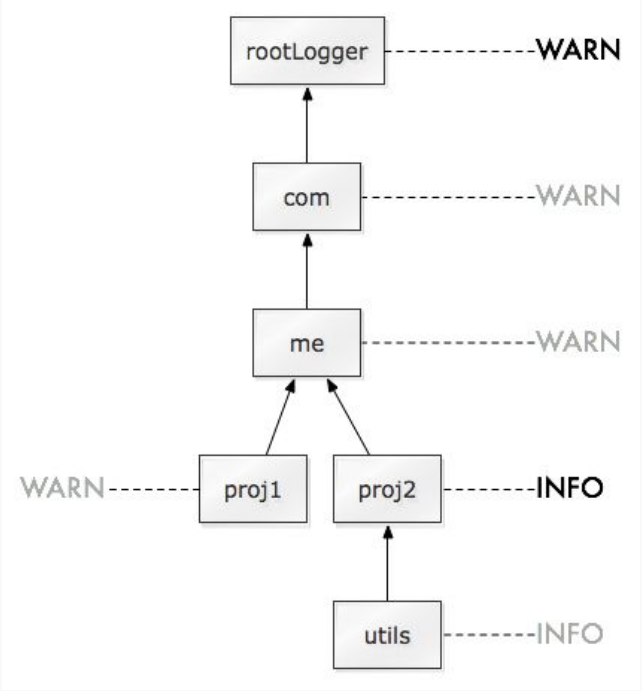
- O objeto Logger possui métodos para gerar os mais diferentes tipo de mensagens
  - **trace(), debug(), info(), warn() e error()**

```
LOGGER.info("Mensagem de log");

LOGGER.debug("O usuário {} conectou", "danilo");
```

Hierarquia dos Loogers

- Os loggers funcionam de forma hierárquica



Log Levels

- ERROR - Os logs de erros são problemas sérios que afetam uma parte significativa do sistema ou alguma parte do seu aplicativo que não funcionou. As exceções são consideradas logs de nível ERROR. Outros exemplos de logs de erros são falhas de conexão de banco de dados e erros de configuração.
- WARN - Os logs de aviso são usados para indicar possíveis problemas que podem causar erros e devem ser monitorados caso falhem. Obviamente, o contexto de um aviso é subjetivo para o desenvolvedor e a situação, portanto, os logs de aviso podem variar de sistema para sistema.
- INFO - é o nível de log padrão definido pelo Spring Boot. Se nenhuma configuração for feita, o nível de log será automaticamente definido como INFO. Esses tipos de logs são informações que normalmente não são necessárias, mas são úteis em situações como depuração de código de produção ou determinação de quando determinados dados são manipulados.

Log Levels

- DEBUG - Os logs DEBUG incluem informações mais detalhadas e específicas que não são necessárias em situações normais. Isso geralmente é definido como um nível de log quando um desenvolvedor está tentando rastrear profundamente um problema ou um bug que é difícil de rastrear.
- TRACE - TRACE é uma versão mais granular do DEBUG. Os logs de TRACE são exaustivos, imagine registrar cada operação que o sistema está fazendo, desde iniciar um serviço, inicializar novas variáveis e chamar o método

Appenders

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

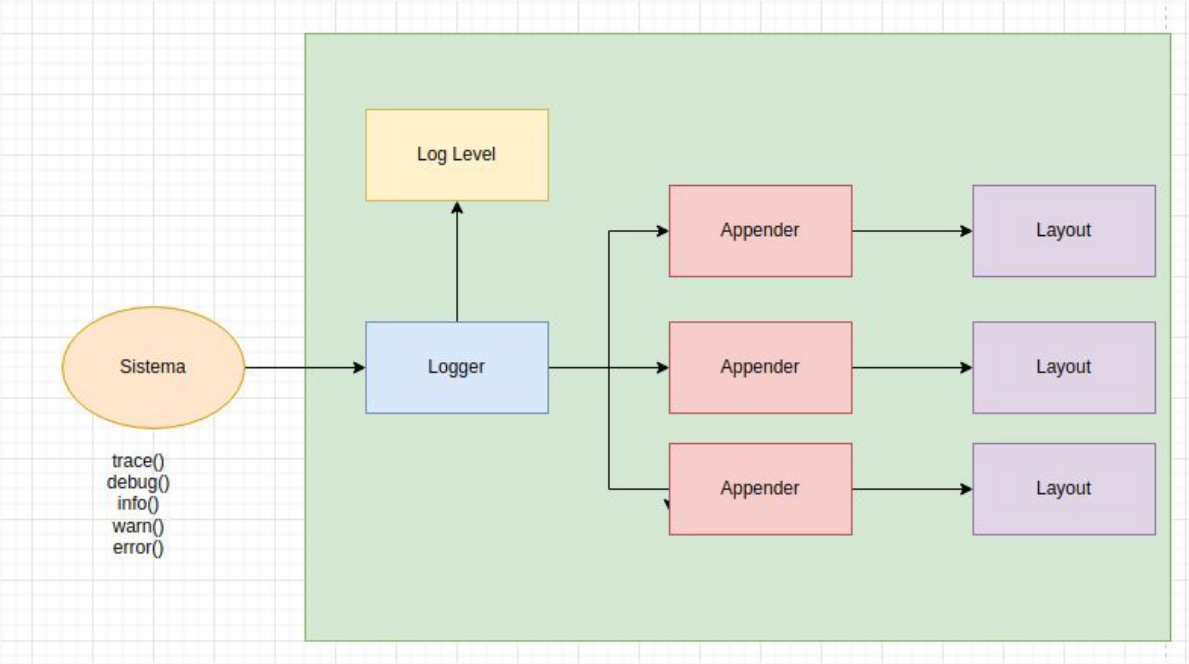
Appender	Descrição
ConsoleAppender	Direciona os logs para o console
FileAppender	Direciona os logs para um arquivo
RollingFileAppender	Direciona os logs para um arquivo e permite definir políticas sobre tamanho máximo do arquivo, número de arquivos de backup e etc.

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
JsonLayout	Formata a saída em JSON

Juntando as partes



Laboratório

- Tarefa 1 - Projeto Agenda Clamed
  - Material de apoio
    - Vídeo 1 - (https://youtu.be/7g5OdQb0u4g)
    - Github - ()

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>