MÓDULO 1 - EXTRA
LOGGING

UNIPDS
EDUCAÇÃO

Observabilidade e Logs

Precisamos saber o que acontece internamente no sistema.

- **Métricas:** valores numéricos ao longo do tempo
- **Traces:** as etapas de uma requisição por todo o sistema
- **Logs:** o que e quando aconteceu

Vamos focar aqui!

java.util.logging

- Além do `System.out.println`
- Diagnosticar erros e entender a causa raiz de falhas
- Monitorar o fluxo e acompanhar o comportamento normal da aplicação
- Debugar e investigar o estado da aplicação em detalhes durante o desenvolvimento
- Auditar e rastrear ações específicas no sistema.

Níveis de Logs

- **SEVERE**: Erros muito graves que podem impedir a aplicação de continuar.
- **WARNING**: Situações inesperadas ou erros que devem ser investigados.
- **INFO**: Informações gerais sobre o progresso da aplicação.
- **CONFIG**: Configurações específicas que estão sendo usadas.
- **FINE** e **FINER**: Informações de depuração mais detalhadas.
- **FINEST**: informações de debug.

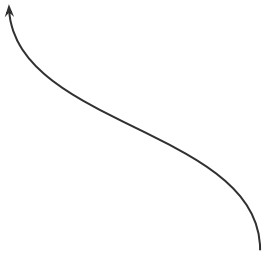
Usando o Logger

```
import java.util.logging.Logger;

public class ServidorItensCardapioComSocket {

    private static final Logger logger =
                Logger.getLogger(ServidorItensCardapioComSocket.class.getName());

     //...

     System.out.println("Subiu servidor!");
     logger.info("Subiu servidor!");

     //...

     logger.finest(request);
     logger.fine("\n\nChegou um novo request");

}
```

A convenção é usar o nome da classe para o Logger

Logando com um Supplier

```
logger.finer(() -> "Method: " + method);
logger.finer(() -> "Request URI: " + requestURI);
logger.finer(() -> "HTTP Version: " + httpVersion);
```



Só efetua a concatenação se o nível FINER estiver habilitado

Mais logs

```
if ("/itensCardapio.json".equals(requestURI)) {
   logger.fine("Chamou arquivo itensCardapio.json");

   //...
} else if ("GET".equals(method) && "/itens-cardapio".equals(requestURI)) {
   logger.fine("Chamou listagem de itens de cardápio");

   //...
} else if ("GET".equals(method) && "/itens-cardapio/total".equals(requestURI)) {
   logger.fine("Chamou total de itens de cardápio");

  //...
} else if ("POST".equals(method) && "/itens-cardapio".equals(requestURI)) {
   logger.fine("Chamou adição de itens de cardápio");

   //...
} else {
   logger.warning(() -> "URI não encontrada: " + requestURI);

   clientOut.println("HTTP/1.1 404 Not Found");
}
```

Logando exceções

```
try (clientSocket) {
    // Lê socket...

    try {

        // Trata URIs...

    } catch (Exception ex) {
        logger.log(Level.SEVERE, ex,
                           () -> "Erro ao tratar " + method + " " + requestURI);
         clientOut.println("HTTP/1.1 500 Internal Server Error");
    }

} catch (Exception ex) {
    // logger.severe("Erro no servidor");
    logger.log(Level.SEVERE, "Erro fatal no servidor", ex);
    throw new RuntimeException(ex);
}
```

Logs resultantes

```
Aug 29, 2025 9:23:57 PM mx.florinda.cardapio.ServidorItensCardapioComSocket
main
INFO: Subiu servidor!
Aug 29, 2025 9:25:12 PM mx.florinda.cardapio.ServidorItensCardapioComSocket
trataRequisicao
SEVERE: Erro ao tratar GET /itens-cardapio
java.lang.RuntimeException: java.sql.SQLSyntaxErrorException: Unknown database
'cardapio'
    at mx.florinda.cardapio.SQLDatabase.listaItensCardapio(SQLDatabase.java:38)
    at mx.florinda.cardapio.ServidorItensCardapioComSocket.trataReq...java:89)
    at mx.florinda.cardapio.ServidorItensCardapioComSocket.lambda$...java:35)
    at java.base/java.util.concurrent.ThreadPoolExecutor.runWorke...java:1144)
    at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.ru...java:642)
    at java.base/java.lang.Thread.run(Thread.java:1583)
Caused by: java.sql.SQLSyntaxErrorException: Unknown database 'cardapio'
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQL...java:112)
    ... 5 more
```

Configuração de Logs

Java já tem a configuração da JDK em `conf/logging.properties`

- define um nível `INFO` como padrão global
- define um `ConsoleHandler` com um `SimpleFormatter`
- também define um `FileHandler` com `XMLFormatter` que gera um arquivo no diretório do usuário com um nome tipo `java0.log`

Podemos definir um customizado em `src/main/resources/logging.properties`

```
.level = FINE
handlers = java.util.logging.ConsoleHandler,java.util.logging.FileHandler
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.FileHandler.level = FINE
java.util.logging.FileHandler.formatter = java.util.logging.SimpleFormatter
java.util.logging.FileHandler.pattern = log.%g.txt
```

Definir argumento da JVM para apontar para a customização dos logs:
`-Djava.util.logging.config.file=src/main/resources/logging.properties`

https://docs.oracle.com/javase/8/docs/api/java/util/logging/package-summary.html

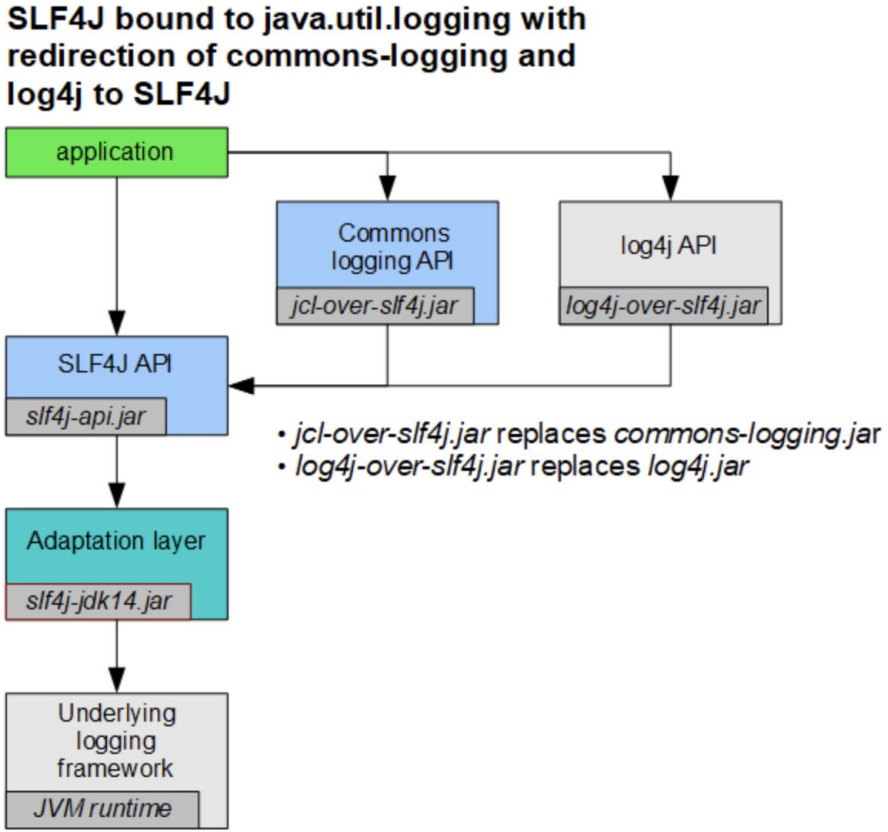
Bibliotecas de Logs no Java e o SLF4J

No ecossistema Java há diferentes libs de logs além do Java Util Logging (JUL):

- Log4J e Log4J 2
- Apache Commons Logging
- Logback...

Se cada biblioteca usa a sua lib de logs, como configurar todas?

O **SLF4J (Simple Logging Façade for Java)** fornece uma API para ser usada e adapters para libs reais.



https://www.slf4j.org/