**TRABAJO PRÁCTICO: LOG4J**

**PARTE A:**

**Ejercicio** **1**

Investigue y explique con sus palabras que produce la etiqueta **Loggers** en el archivo de configuración de log4j.

Respuesta:

La etiqueta <Loggers> en este archivo de configuración de Log4j define los registradores de la aplicación y sus configuraciones. En este caso, solo hay un registrador raíz definido dentro de la etiqueta <Loggers>, que es el registrador raíz de la aplicación. Este registrador raíz se configura con un nivel de registro de "debug" y se le asigna un apéndice (appender) llamado "console", que enviará los mensajes de registro a la consola. Esto significa que todos los mensajes de registro de nivel "debug" y superior serán registrados y enviados a la consola

**Ejercicio** **2**

Investigue y explique con sus palabras que produce la etiqueta **Appenders** en el archivo de configuración de log4j.

Respuesta:

La etiqueta <Appenders> en este archivo de configuración de Log4j define los apéndices disponibles para enviar los mensajes de registro a diferentes destinos. En este caso, se define un único apéndice llamado "console", que está configurado para enviar los mensajes de registro a la salida estándar (consola). El formato de los mensajes de registro se especifica utilizando un patrón de diseño (PatternLayout), que incluye la fecha, el hilo de ejecución, el nivel de registro, el nombre del registrador, el mensaje y un salto de línea.

**Ejercicio** **3**

Investigue y explique con sus palabras que produce la etiqueta **Root Level** en el archivo de configuración de log4j.

Respuesta:

La etiqueta <Root> junto con la etiqueta <Root Level> en este archivo de configuración de Log4j define el nivel de registro raíz para toda la aplicación. El nivel de registro raíz se establece en "debug", lo que significa que todos los mensajes de registro de nivel "debug" y superior serán registrados y enviados al apéndice definido. En este caso, el nivel de registro raíz afecta al registrador raíz de la aplicación, que es el único registrador definido en este archivo.

**Ejercicio 4**

Investigue y documente: qué es Apache Flume ?

Respuesta:

Apache Flume es una herramienta de código abierto de Apache diseñada para la ingestión de datos de alta velocidad y alta fiabilidad en sistemas de almacenamiento de datos distribuidos, como Hadoop HDFS, Apache HBase y Apache Kafka, entre otros. Flume permite la transferencia eficiente de grandes volúmenes de datos desde múltiples fuentes a través de una arquitectura de canalización de datos flexible y escalable. Es utilizado en entornos de big data para recopilar datos de diferentes orígenes, como registros de servidores, datos de sensores y transmisiones de redes sociales, y almacenarlos en sistemas de almacenamiento de datos para análisis posterior.

**PARTE B:**

**Ejercicio**

Investigue, documente y aplique un ejemplo de la clase ThreadContext.

Respuesta:

La clase ThreadContext es una parte importante de Apache Log4j 2 que proporciona un contexto de hilo para los mensajes de registro. Permite asociar información adicional con cada hilo en ejecución y recuperarla fácilmente cuando se generan registros. Esto es útil en entornos multi-hilo donde es importante rastrear y correlacionar los registros con las operaciones específicas que se están realizando en cada hilo.

Ejemplo:

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.ThreadContext;

public class ThreadContextExample {

private static final Logger logger = LogManager.getLogger(ThreadContextExample.class);

public static void main(String[] args) {

// Añadir información al contexto del hilo actual

ThreadContext.put("userId", "12345");

ThreadContext.put("username", "john\_doe");

// Generar algunos registros

logger.info("Inicio de la aplicación.");

// Simular alguna operación

performSomeOperation();

// Eliminar información del contexto del hilo actual

ThreadContext.clearAll();

}

private static void performSomeOperation() {

// Obtener información del contexto del hilo actual

String userId = ThreadContext.get("userId");

String username = ThreadContext.get("username");

// Generar registros con la información del contexto del hilo actual

logger.info("Operación realizada por el usuario con ID: {} y nombre de usuario: {}", userId, username);

}

}

En este ejemplo:

1. Importamos las clases necesarias de Log4j.
2. Configuramos un Logger para la clase ThreadContextExample.
3. Utilizamos ThreadContext.put() para agregar información (como ID de usuario y nombre de usuario) al contexto del hilo actual.
4. Generamos algunos registros utilizando el Logger
5. Simulamos alguna operación (performSomeOperation()) donde recuperamos la información del contexto del hilo actual utilizando ThreadContext.get().
6. Finalmente, limpiamos el contexto del hilo actual con ThreadContext.clearAll().

El resultado es que los registros generados incluyen la información específica del contexto del hilo actual, lo que facilita la correlación de los registros con las operaciones y usuarios específicos en un entorno multi-hilo.