Certified Tech Developer
The Ultimate Degree

# Back End I

## 1. ¿Qué es Log4j?

*Log4j es una librería desarrollada en Java por la Apache Software Foundation que permite a los desarrolladores elegir la salida y el nivel de granularidad de los mensajes o logs en tiempo de ejecución. En otras palabras, es utilizada para generar mensajes de logging de una forma limpia, sencilla, permitiendo filtrarlos por importancia y pudiendo configurar su salida tanto por consola, fichero u otras diferentes. Veamos algunas de sus ventajas y desventajas.*

*Permite tener un registro de lo que está pasando en nuestros sistemas, lo que nos posibilita entender mejor los errores.*

*La única desventaja es que a veces los archivos se hacen muy grandes y ocupan mucho espacio. Es por ello que debemos elegir bien qué tipo de información queremos almacenar.*

## 2. Arquitectura Log4j

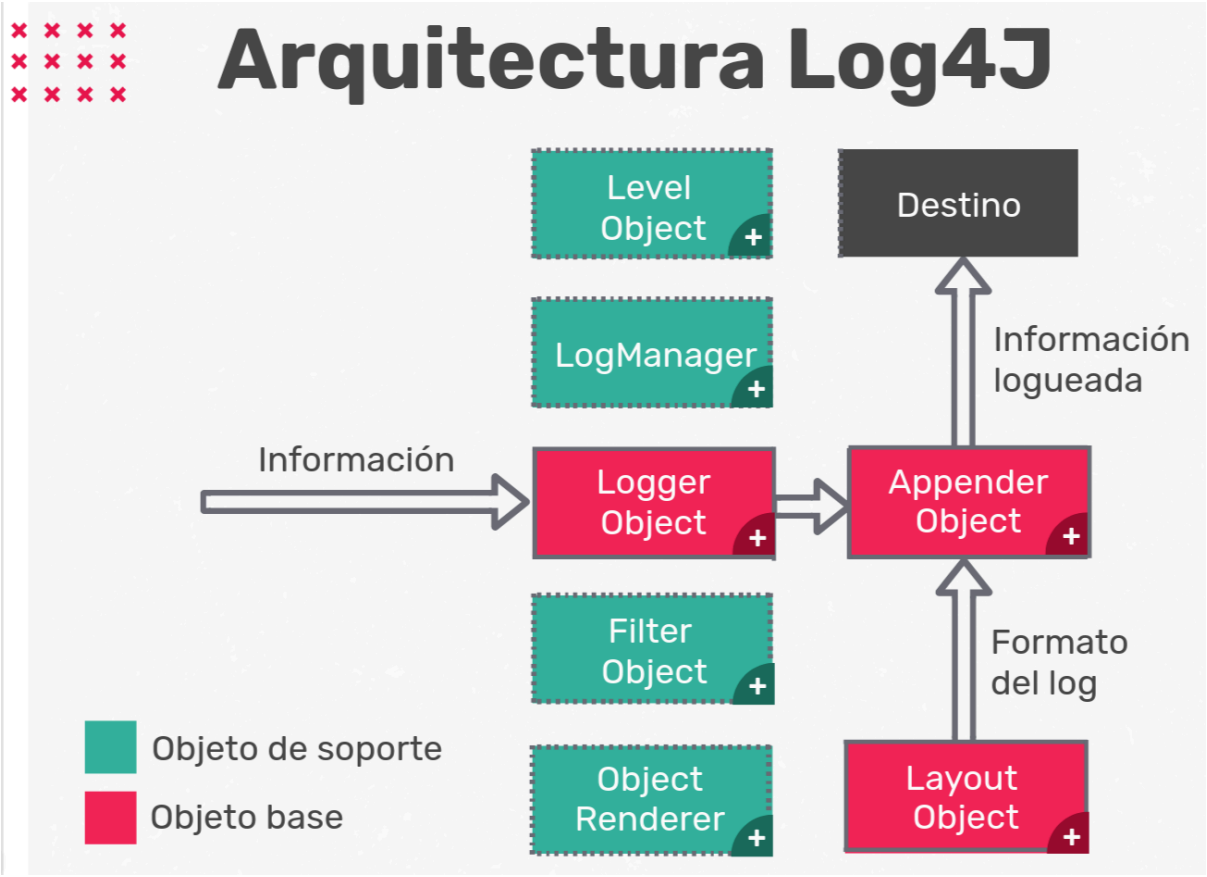
*La API de Log4j sigue una arquitectura en capas donde cada una proporciona diferentes objetos para realizar diferentes tareas. Esta estructura hace que el diseño sea flexible y fácil de ampliar en el futuro.*

*Los dos tipos de objetos disponibles con el marco Log4j son:*

> ***Objetos base:*** *Objetos de marco obligatorios y necesarios para utilizar el marco.*
> ***Objetos de soporte:*** *Objetos de marco opcionales que soportan objetos básicos para realizar tareas adicionales*

*Los objetos base incluyen los siguientes tipos de objetos:*

*Logger object*
*Appender object*




| *OBJETOS DE SOPORTE* | |
|---|---|
| ***OBJETO*** | ***DESCRIPCIÓN*** |
| **Level Object** | El objeto de nivel define la granularidad y prioridad de cualquier información de registro. Hay siete niveles de registro definidos en la API: OFF, DEBUG, INFO, ERROR, WARN, FATAL y ALL. |
| **LogManager** | *Administra el marco de registro. Es responsable de leer los parámetros de configuración inicial de una configuración de todo el sistema, un archivo de configuración o una **class** de configuración.* |
| **Filter Object** | *El objeto de filtro se utiliza para analizar la información de registro y tomar otras decisiones sobre si esa información debe registrarse o no.* |
| **Object Renderer** | *Se especializa en proporcionar una representación de cadena de varios objetos pasados a la infraestructura de registro. Los objetos layout utilizan este objeto para preparar la información de registro final.* |

| OBJETOS BASE | |
|---|---|
| **OBJETO** | **DESCRIPCIÓN** |
| **Logger Object** | *La capa de nivel superior es el Logger que proporciona el objeto Logger. Este es responsable de capturar la información de registro y se almacena en una jerarquía de espacio de nombres.* |
| **Appender Object** | *Esta es una capa de nivel inferior que proporciona el objeto appender. Este es responsable de publicar información de registro.* |
| **Layout Object** | *La capa de diseño proporciona objetos que se utilizan para formatear la información del registro en diferentes estilos. Los objetos de diseño juegan un papel importante en la publicación de información de registro de una manera que sea legible.* |

### 3. Niveles de registro

| NIVELES DE REGISTRO | |
|---|---|
| **NIVEL** | **DESCRIPCIÓN** |
| **OFF** | *Este es el nivel de mínimo detalle, deshabilita todos los logs.* |
| **FATAL** | *Se utiliza para mensajes críticos del sistema, generalmente después de guardar el mensaje, el programa se cierra.* |
| **ERROR** | *Indica eventos de error que aún podrían permitir que la aplicación continúe ejecutándose.* |
| **WARN** | *Se utiliza para mensajes de alerta sobre eventos.* |
| **INFO** | *Se refiere a mensajes informativos que resaltan el progreso de la aplicación en un nivel aproximado.* |
| **DEBUG** | *Designa los eventos informativos detallados más útiles para depurar una aplicación.* |
| **TRACE** | *Se utiliza para mostrar mensajes con un mayor nivel de detalle que debug.* |
| **ALL** | *Es el nivel de máximo detalle, habilita todos los logs.* |

**Ejemplo**

*En el siguiente ejemplo veremos cómo instanciar un objeto logger que nos permitirá loguear mensajes en diferentes prioridades. Los más usados son error, warning, debug e info.*

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

public class TestLog {
  private static final Logger logger = Logger.getLogger(TestLog.class);
  public static void main(String[] args) {
    logger.info("Empezamos nuestro metodo MAIN");
    try {
        String variable = "Hola";
        int division = 1 / 0;
    } catch (Exception e) {
        logger.error("Error por dividir por cero ", e);
    }
    logger.warn("Advertencia el metodo MAIN esta por finalizar");
    logger.debug("Esto va a mostrarse solo si el infoLogger esta en DEBUG");
    logger.info("Finalizamos el thread MAIN");
  }
}
```

## 4. PRÁCTICA

*Te proponemos resolver la siguiente actividad: Crear una aplicación donde tengamos dos clases: León y Tigre. Cada una debe tener dos atributos:*

*nombre* ***String***
*edad* ***int***

*y para la clase León vamos a agregar el atributo:*

*esAlfa* ***boolean***

*Para los dos animales vamos a crear un método correr, que va a loguear un info de que está corriendo y vamos a crear otro método que calcule si es mayor a 10 años y ser alfa, en caso de serlo, deberá loguear un info con la información.*

*También deberemos arrojar una* ***Exception*** *si la edad del animal es menor a cero y agregar un log de error. Creamos una clase main, donde creamos leones y tigres que cumplan que al correr y esMayorA10 ejecutan los métodos:*

***public void*** *correr()*
***public void*** *esMayorA10()*

*También debemos chequear que los **logs** existan.*

**La salida debe ser algo como:**

```
[2021-07-18 18:27:46] [ INFO ] [Leon:37] El León Simba está corriendo
[2021-07-18 18:27:46] [ INFO ] [Leon:37] El León Bom está corriendo
[2021-07-18 18:27:46] [ ERROR] [Leon:45] La edad no puede ser negativa
[2021-07-18 18:27:46] [ ERROR] [Test:30] La edad del León Bom es incorrecta
java.lang.Exception
        at com.main.Leon.esMayorA10(Leon.java:46)
        at com.main.Test.main(Test.java:28)
[2021-07-18 18:27:46] [ INFO ] [Tigre:28] El Tigre está corriendo
[2021-07-18 18:27:46] [ INFO ] [Tigre:28] El Tigre está corriendo
```