La Clase Logger

La clase Logger de Java permite crear mensajes para el seguimiento o registro de la ejecución de una aplicación. Sirve, por ejemplo, para realizar la depuración de la aplicación si se muestran los distintos puntos o estados por lo que va pasando la ejecución con los valores tomados por variables de interés.

Esta clase Logger ofrece la ventaja de poder emitir la salida de mensajes en archivos con diferentes formatos (XML, TXT, HTML, etc).

Niveles de importancia de los mensajes

Cada mensaje generado con la clase Logger debe tener asignado un nivel de importancia a elegir entre los siguientes, ordenador de mayor importancia a menos:

SEVERE: Nivel de mensaje indicando un error serio.
WARNING: Indica un error potencial.
INFO: Para mensajes informativos.
CONFIG: Usado con mensajes relacionados con la configuración.
FINE: Proporciona información de la traza de la ejecución.
FINER: Información de traza más detallada.
FINEST: Información de traza muy detallada.

Para establecer estos niveles se debe usar la clase Level de Java, que ofrece una serie de constantes estáticas relacionadas con estos niveles: Level.**SEVERE**, Level.**WARNING**, etc.

Ejemplo:

```
public class LOGGER {
    private static Logger logger = Logger.getLogger(Pruebas.LOGGER.class .getName());

    public static void main(String[] args) {

        logger.setLevel(Level.INFO);
        logger.severe("Nivel de mensaje indicando un error serio");
        logger.warning("Texto Peligro");
        logger.info("Texto Informativo");

        logger.config("Usado con mensajes relacionados con la configuración");
        logger.fine("Proporciona información de traza de la ejecución");
        logger.finer("Información de traza más detallada");
        logger.finest("Información de traza muy detallada");
    }

}
```

Primero se declara la variable Logger
Esta clase como no ofrece un método constructor, se debe usar el método estático getLogger() que permite obtener un objeto Logger relacionado con el nombre que se asigne como parámetro.

static Logger getLogger(String name)

El “name” que se asigna por parámetro será normalmente la clase en la que se esté utilizando el Logger precedida con puntos del nombre del paquete en el que se encuentre, por ejemplo: "paquete.NombreClase". Este String se puede utilizar usando una llamada a NombreClase.class.getName(), sustituyendo NombreClase por el nombre correspondiente a la clase en la que se está haciendo uso del Logger

```
private static Logger logger = Logger.getLogger(Pruebas.LOGGER.class .getName());
```

Generar mensajes

```
logger.setLevel(Level.INFO);
logger.severe("Nivel de mensaje indicando un error serio");
logger.warning("Texto Peligro");
logger.info("Texto Informativo");
```

Por defecto, el nivel más bajo de importancia de los mensajes que se registran es INFO, por lo que los mensajes de importancia menor no se mostrarán a no ser que se modifique el nivel más bajo.

```
oct. 08, 2020 9:22:37 A. M. Pruebas.LOGGER main
SEVERE: Nivel de mensaje indicando un error serio
oct. 08, 2020 9:22:37 A. M. Pruebas.LOGGER main
WARNING: Texto Peligro
oct. 08, 2020 9:22:37 A. M. Pruebas.LOGGER main
INFO: Texto Informativo
```

Para modificar el nivel a partir del cual se registrarán los mensajes se debe usar el método setLevel():

LOG.setLevel(Level.FINE);

A este método se le debe pasar por parámetro el nivel más bajo a registrar. Para esto se dispone además de cada uno de los niveles, las constantes Level.OFF y Level.ALL para no registrar ningún mensaje o registrarlos todos respectivamente. Por ejemplo, si se desean registrar los mensajes que al menos tengan una importancia de FINE se debe ejecutar

```
logger.config("Usado con mensajes relacionados con la configuración");
logger.fine("Proporciona información de traza de la ejecución");
logger.finer("Información de traza más detallada");
logger.finest("Información de traza muy detallada");
```

```
logger.setLevel(Level.FINEST);
```

Esto es de utilidad para establecer un nivel bajo de registro de mensajes mientras la aplicación está en proceso de desarrollo para hacer un seguimiento detallado de depuración. Cuando la aplicación se publique, se deberá cambiar el nivel de registro, para generar sólo los mensajes que realmente sean importantes.

Salida de mensajes en archivos

Los mensajes de Logger parecen por defecto en la salida estándar, pero es posible almacenar dichos mensajes en archivos para que puedan ser analizados posteriormente a la ejecución de la aplicación.
Para ello, se usará el método addHandler, de la clase Logger, que permite añadir nuevos manipuladores de los mensajes, como puede ser un archivo.

```
void addHandler(Handler handler)
```

Como parámetro se debe indicar un objeto de tipo Handler que son manipuladores en los que se puede obtener el resultado de este tipo de mensajes. Algunos tipos de manipuladores que podemos usar son:

- **ConsoleHandler**: Muestra los mensajes a través de System.err que puede ser la salida estándar.
- **FileHandler**: Permite guardar los mensajes en archivos.
- **SocketHandler**: Puede enviar los mensajes a través de la red.

Si se utiliza un objeto FileHandler como Handler del Logger, se generará por defecto un archivo de tipo XML con el contenido de los mensajes. Esto puedes hacerlo con un código similar al siguiente, donde se genera el archivo Logging.xml:

```
FileHandler fileXml = new FileHandler("Logging.xml");
LOG.addHandler(fileXml);
```

El fichero XML que se obtiene tiene un contenido como el siguiente:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE log SYSTEM "logger.dtd">
- <log>
  - <record>
      <date>2020-10-08T07:39:06.057772900Z</date>
      <millis>1602142746057</millis>
      <nanos>772900</nanos>
      <sequence>0</sequence>
      <logger>Pruebas.LOGGER</logger>
      <level>SEVERE</level>
      <class>Pruebas.LOGGER</class>
      <method>main</method>
      <thread>1</thread>
      <message>Nivel de mensaje indicando un error serio</message>
    </record>
  - <record>
      <date>2020-10-08T07:39:06.130578300Z</date>
      <millis>1602142746130</millis>
      <nanos>578300</nanos>
      <sequence>1</sequence>
      <logger>Pruebas.LOGGER</logger>
      <level>WARNING</level>
      <class>Pruebas.LOGGER</class>
      <method>main</method>
      <thread>1</thread>
      <message>Texto Peligro</message>
    </record>
```

Si en lugar de obtener un archivo con contenido XML quieres almacenar los mensajes en un documento de texto plano, deberás asignarle un Formatter de tipo SimpleFormatter al objeto Handler. Puedes hacerlo de manera similar a esta:

```
FileHandler fileTxt = new FileHandler("C:/Users/IN2DAM/Desktop/Logging.txt");
SimpleFormatter formatterTxt = new SimpleFormatter();
fileTxt.setFormatter(formatterTxt);
logger.addHandler(fileTxt);

logger.severe("Nivel de mensaje indicando un error serio");
logger.warning("Texto Peligro");
logger.info("Texto Informativo");
```

Y el resultado sería lo siguiente

```
Archivo  Edición  Formato  Ver  Ayuda
oct. 08, 2020 9:37:43 A. M. Pruebas.LOGGER main
SEVERE: Nivel de mensaje indicando un error serio
oct. 08, 2020 9:37:43 A. M. Pruebas.LOGGER main
WARNING: Texto Peligro
oct. 08, 2020 9:37:43 A. M. Pruebas.LOGGER main
INFO: Texto Informativo
```