Logger

API para la escritura, notificación, registro y propagación de errores en aplicaciones Java. Consta de una clase abstracta para el control centralizado del log en los diferentes niveles de severidad.

- Escritura en el log del servidor a través de la API log4j.
- Notificación vía WS para registro de errores en DDBB, según implementación del servicio de Log de Registro de Errores .
- Propagación de excepciones al usuario (front-end) a través de la implementación del método abstracto, el cual debe propagar de cara al usuario por el uso de las herramientas de mensajería utilizadas por el framework del front-end de cada aplicativo.

public void addMessage(final String idMessage, final int severity, final String resume, final String detail)

Log

Implementación de los diferentes métodos de llamado a escritura en el servidor de aplicaciones a través de la API log4j, según los diferentes niveles de severidad:

- Debug: Se realizará la escritura en el log sólo si está habilitada en este modo.
- Info: registra en el log del servidor cualquier notificación en este nivel de severidad.
- Warn: registra en el log del servidor cualquier notificación en este nivel de severidad.
- **Error**: registra en el log del servidor y a través del servicio de registro de log cualquier notificación en este nivel de severidad.
- **Fatal**: registra en el log del servidor y a través del servicio de registro de log cualquier notificación en este nivel de severidad.

NOTA: Los dos últimos métodos harán uso de la notificación vía WS para registro en DDBB.

Esta funcionalidad hará uso de la configuración definida, log.json. Tal como se puede observar, dicha configuración está escrita en formato.

Notificación

A través del consumo del servicio de registro de errores en base de datos, se hará registro de los niveles de severidad Error y Fatal.

Propagación de excepciones

En todo momento, se realizará el llamado a la implementación del método

```
/***
 * Método abastracto para ser implementado en la clase desendiente, el cual
```

```
 * tiene el propósito de notificar visualmente al usuario acerca los eventos que
 * están siendo reportados. En la implementación de este método, se usará
 * cualquier tecnología de mensajería con la cual esté familiarizado el
 * front-end del aplicativo, permitiendo así la portabilidad a cualquier
 * framework
 */
public abstract void addMessage(final String idMessage, final int severity, final String resume,
        final String detail);
```

el cual al ser un método abstracto será de obligatoria implementación en la clase que extiende de la abstracta com.ada.utilidades.situ.log.ALogger para garantizar de forma limpia, la notificación de los eventos presentados hasta el usuario.

Un ejemplo de la implementación de la clase ALogger puede ser Log, previa adición de la API `logger` al `.classpath` del sistema cliente.

NOTA: En el archivo de configuración del servidor o en una variable de entorno la llave `LOG_CONFIGURATION_PATH`, con valor por ejemplo de `C:\co\ada\situ\log.json`, se deberá configurar la ruta de ubicación del archivo log.json. El cual es el insumo principal del proceso para darle control a las excepciones.

<< regresar