UT-02. Actividad 2.5 Logging

1. Objetivos

Instrumentar la aplicación con logs de traza.

- SLF4J Fachada o abstracción común para encajar diferentes sistemas de log
- Log4J Framework más famoso de Log. Lo "encajaremos" con SLF4J

1.1 Referencias

Simple Logging Facade for Java (SLF4J) http://www.slf4j.org/

Apache Log4j https://logging.apache.org/log4j/1.2/manual.html

2. Proyecto de ejemplo

2.1 Propiedades de proyecto

Nombre: Logger4j

Tipo: Maven > Java EE 7

Servlets:

es.cifpcm.infra.logging.ServletB

es.cifpcm.logger.web.ServletA

2.2 Dependencias

http://stackoverflow.com/questions/4311026/how-to-get-slf4j-hello-world-working-with-log4j

Vamos a utilizar Apache Log4J como framework de Log, pero no lo vamos a utilizar directamente, sino a través de Simple Logging Facade for Java (SLF4J).

Vamos a utilizar los Logger como si fueran SLF4J, pero internamente se va a llamar al framework Log4J. Esto es transparenta para nosotros, simplemente se consigue añadiendo los Jar apropiados.

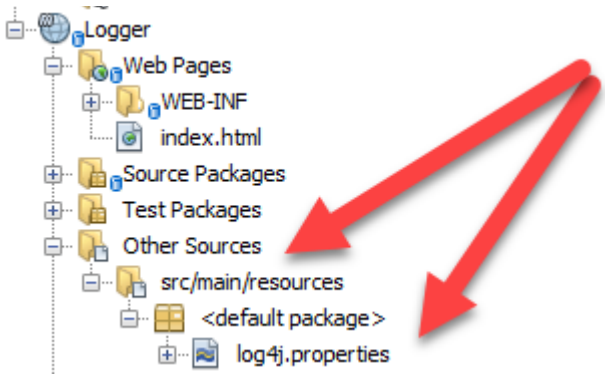
Añadiremos las siguientes dependencias:

Jar	Descripción	Explicación
slf4j-api-1.7.20.jar	API de la fachada SLF4J	Interfaces de SLF4J

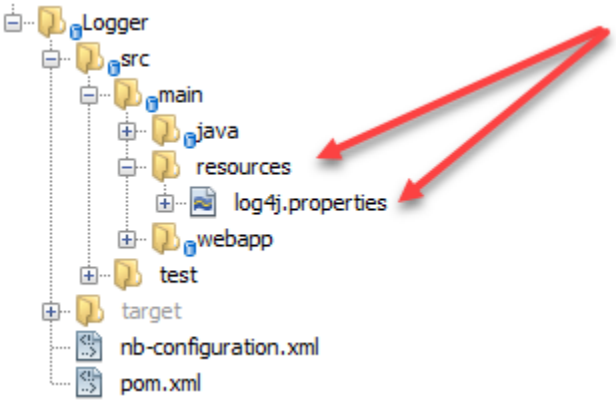
Jar	Descripción	Explicación
slf4j-log4j12-1.7.21.jar	Binding a Log4j 1.2	Transforma las llamadas SLF4J en llamadas a Log4J
Log4j-1.2.17.jar	Log4J 1.2	Interfaces e implementación de Log4J

2.3 Fichero de propiedades

Crear un fichero de propiedades de nombre log4j (log4j.properties) en el paquete por defecto de "Other Sources"



Vista de proyecto



Vista de ficheros

2.3.1 Fichero de propiedades

El root Logger se establecerá a nivel DEBUG y "sacará" mensajes a traves de los Appender file (fichero log) y stdout (pantalla/salida estándar)

```
# Root logger option

log4j.rootLogger=DEBUG, file, stdout
```

Se creará un appender de nombre **file**, para escribir en fichero (RollingFileAppender va escribiendo en ficheros de un tamaño establecido)

```
# Direct log messages to a log file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=c:/temp/log/logger.log

log4j.appender.file.MaxFileSize=10MB

log4j.appender.file.MaxBackupIndex=10

log4j.appender.file.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

Se creará otro de nombre **stdout**, para registrar en la salida estándar:

```
# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

2.4 Código de los Servlet

En primer lugar definimos una variable miembro para el Logger y la inicializamos a partir de la LoggerFactory, pidiendo un Logger con el nombre de la clase del Servlet.

Utilizando este objeto logger en los diferentes métodos de las clases podremos ir registrando todo lo que pasa durante el desarrollo de nuestro programa. Utilizaremos varios niveles:

- DEBUG. Cualquier traza que queramos generar para desarrollo.

- INFO. Eventos importantes que queramos registrar (comienzo de una operación larga, de una conexión aun sistema externo, etc.)
- ERROR. Registrar errores y excepciones.

Además de los anteriores tenemos muchos más. Asociado a cada nivel de Log hay un método del logger.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    logger.info("Entrando en processRequest de la app: {}", request.getContextPath());

    logger.debug("Parámetro mes: {}", request.getParameter("mes"));

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
```

DEBUG

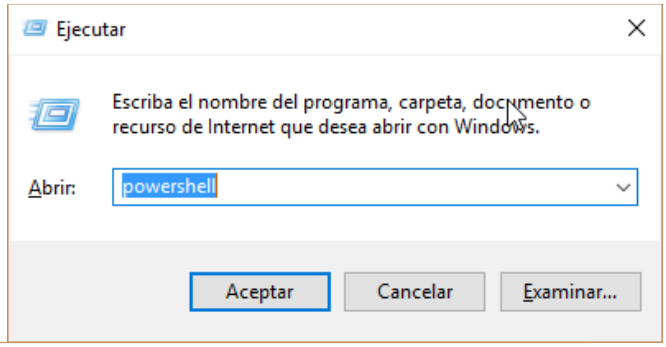
INFO

3. Pruebas a realizar

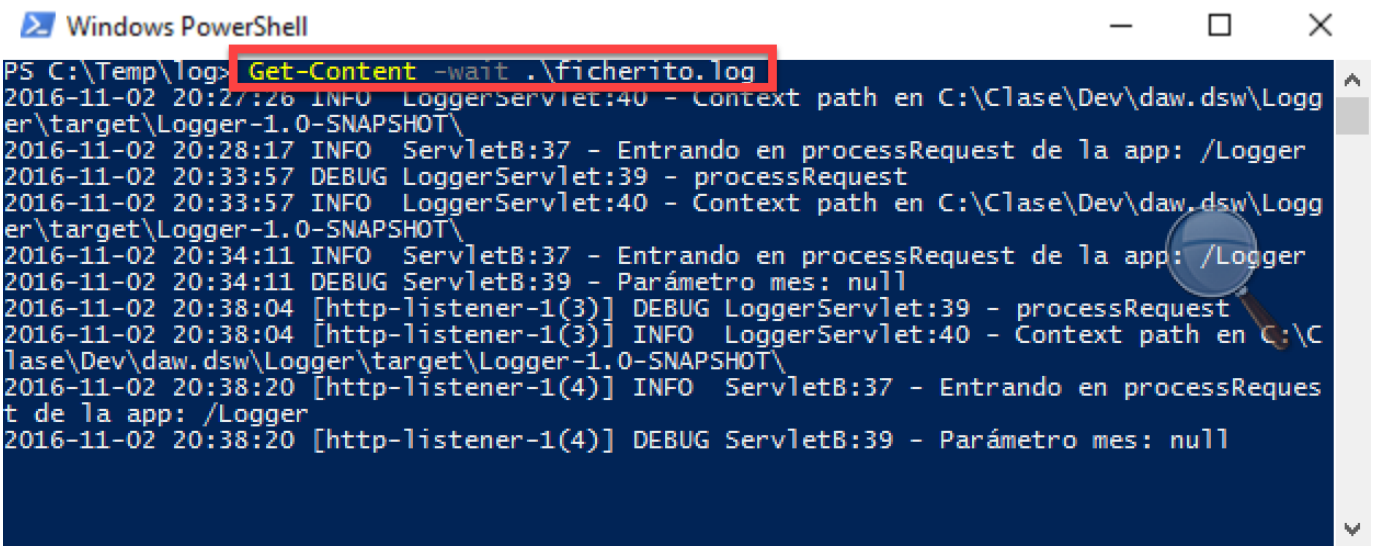
Añadir trazas de log de nivel DEBUG e INFO al método processRequest de ambos servlet.

Para verificar los cambios en los log de disco vamos a utilizar Windows Powershell.

Para arrancar Windows Powershell: winKey + R → powershell



Para ir viendo cómo cambia un archivo (lo que sería un tail en Linux) utilizaremos el comando Get-Content con la opción -wait.



3.1 Cambiar el nivel a uno de los logger

Definir el logger del servletA como nivel INFO. Comprobar que desaparecen los mensajes de DEBUG de ese servlet.

3.2 Inhabilitar el logger del servletB

Inhabilitar el logger del servletB, estableciendo el nivel de log a OFF

3.2.1 Añadir un nuevo appender “ficherito” que escriba en un archivo “ficherito.log“

Cambiar el nivel del rootLogger a INFO

Crear el appender y establecer para el logger es.cifpcm.infra este appender a nivel DEBUG

3.2.2 Modificar el patrón de conversión (ficherito) para que incluya el hilo (thread de ejecución)

En el appender de nombre “ficherito” cambiar el patrón para añadir el nombre del hilo de ejecución.