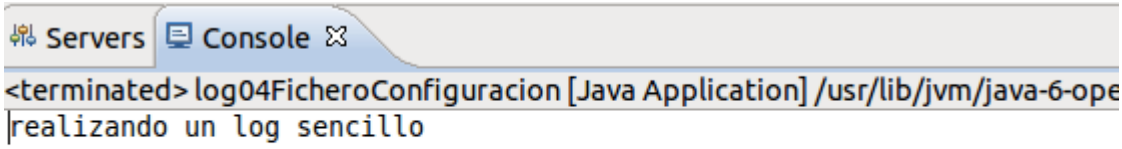
En el libro hemos usado Log4J como api de logs . Muchas aplicaciones que se desarrollan hoy en dia usan este api y se apoyan de forma fuerte en la clase PatternLayout para definir la estructura de los textos que son impresos por el log como se muestra a continuación.

En el fichero de log4j.properties el PatternLayout se define en la siguiente linea de código:

log4j.appender.miappender.layout.conversionPattern=%m%n

Este tipo de patron que hemos utilizado nos muestra el mensaje de log (%m) y el retorno de carro (%n) como se muestra a continuación.



Ahora bien en la mayor parte de las situaciones nos gustaría tener información adicional sobre cuando y que clase realiza las operaciones de Log. Para ello Log4j nos provee de un conjunto sólido de opciones .Vamos a revisarlas:

%M : Muestra el método que ha generado la linea actual de Log

%L: La linea exacta del programa en la cual se genero el log

%p: El nivel de log asociado a la petición

%t :El hilo en el cual se esta ejecutando la petición que realiza log

%F El nombre del fichero que realizo la petición de log

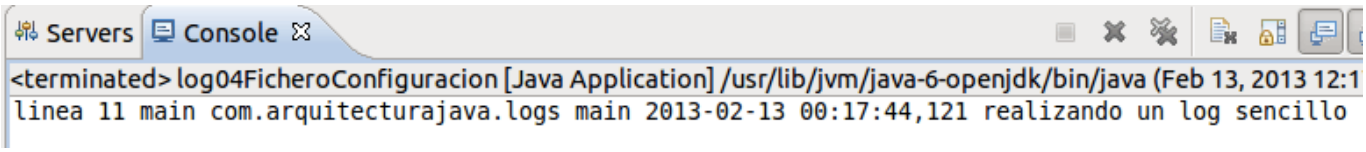
%d :La fecha en la que se realizo el log

%c : El logger concreto que ha salvado esa información.

Por ejemplo podríamos tener un patrón de conversión como el siguiente que nos aporta una información mas clara

log4j.appender.miappender.layout.conversionPattern= linea %L %M %c %t %d %m %n

De esta forma la información sobre los mensajes de Log es mucho mas clara como podemos ver .



Ahora bien aunque podemos usar cualquier patron o combinación de patrones no todos exigen el mismo esfuerzo al motor de logs . Existen situaciones en la que el uso de alguno de los patrones degrada con claridad el rendimiento de nuestra aplicación .En el siguiente post abordaremos estos problemas.