Logs

¡Comencemos!

¿Qué aprenderás?

- Conocer el concepto de logs.
- Leer logs del servidor.
  Crear Logs.

Introducción

A continuación se verá lo que es un log, además de como se puede implementar y donde se almacenan. Se explicará sobre los niveles de los logs y qué función cumple cada nivel. Se realizará un sencillo ejemplo de cómo incorporar logs en una aplicación.

¡Vamos con todo!

¿Qué es un Log?

Un log es un archivo de texto plano que almacena de manera cronológica acontecimientos o cambios que han afectado a la aplicación. Un log nos permite además detectar dónde y cuándo se produjeron errores, por lo que son útiles para el desarrollador generar y obtener este tipo de información.

Estos *logs*, además poseen niveles de información.

- **DEBUG:** Son los mensajes de depuración, son útiles para obtener el comportamiento mientras se desarrolla la aplicación. Se representa de color verde.

- **INFO:** Son los mensajes que muestran información del programa durante la ejecución, por ejemplo versiones. Se representa de color verde.

- **WARN:** Son mensajes de alerta por situaciones anómalas de la ejecución, pero no afectan el funcionamiento del sistema. Se representa de color amarillo.

- **ERROR:** Son mensajes que muestran una situación de errores en la ejecución, aunque el programa siga funcionando, puede ser afectado. Por ejemplo, un fichero no existe. Se representa de color rojo.

- **FATAL:** Son mensajes críticos, son los que hacen que el programa aborte la ejecución y se detenga. Se representa de color rojo.

Generando logs

Utilizaremos la librería *slf4j*, que es una que ya está presente en spring boot, sólo queda hacer la llamada de esta y utilizarla.

Entonces, creamos un nuevo proyecto Spring, y exportamos las dependencias web. Una vez realizado esto, debemos agregar configuración en la aplicación, entonces tendremos que editar el archivo de propiedades de la aplicación (*application.properties*) en este archivo agregamos información básica del log.

Para más información de configuración de logs.

```
logging.file=app.log
logging.file.max-size=10MB
logging.level.root=INFO
logging.level.org.springframework.web=DEBUG
```

Aquí entonces, definimos cuatro configuraciones básicas.

- **Logging.file:** Se refiere a donde se van a almacenar estos logs.

- **Logging.file.max-size:** Se refiere al tamaño máximo que tendrá el archivo, si sobrepasa este tamaño, creará uno nuevo y el anterior lo compartirá.

- **Logging.level.root:** Se refiere al nivel máximo que almacenará el logger para la aplicación base (desde la carpeta origen).

- **Logging.level.org.springframework.web:** Lo mismo que el anterior, pero es para la aplicación web.

Los niveles de Logging son:

- **Debug:** Designado para informar eventos específicos útiles para depurar una aplicación. Como por ejemplo el resultado de una suma parcial en un código de calculadora.

- **Info:** Mensajes útiles para registrar el progreso de la aplicación. Como por ejemplo cuando inicia o termina un servicio.

- **Warn:** Nos alerta de situaciones que posiblemente perjudiquen el funcionamiento normal de nuestra aplicación. Como por ejemplo que utilicemos una función deprecada, que se espera sea removida en una nueva versión.

- **Error:** Designado para informar eventos de error que nos podrían permitir mantener la aplicación en ejecución. Como por ejemplo si.

- **Fatal:** Utilizado en eventos de error muy graves que obligan a la aplicación a detenerse. Como por ejemplo si ocurre una división por 0 y no atrapamos la excepción.

Probemos nuestro *logger*, en el método Main de la aplicación incorporaremos niveles de las diferentes llamadas de los *logs*.

```
package com.log.ejemploLog;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class App {
   private final static Logger logger =
LoggerFactory.getLogger(App.class);
   public static void main(String[] args) {
      SpringApplication.run(App.class, args);
      logger.debug("debug log");
      logger.info("info log");
      logger.warn("warning log");
      logger.error("error log");
   }
}
```

Ejecutamos la aplicación y nos encontraremos con que al iniciar nuestra aplicación, escribiremos nuestros logs en colores.

Entonces, observamos que nuestra consola imprime los logs que se incluyeron en el método main, donde hay dos observaciones.

1. ¿Por qué no aparece el log de debug que se incluyó?
   Es porque en las propiedades de la aplicación, indicamos que sólo se debería mostrar desde el nivel INFO los logs en la aplicación.

2. ¿Por qué sí aparecen algunos DEBUG? Esto es porque corresponden al debug desde la web.

Ahora bien, vamos a revisar el archivo que se generó para estos logs, entonces, revisamos nuestro directorio de raíz, donde observamos lo siguiente:

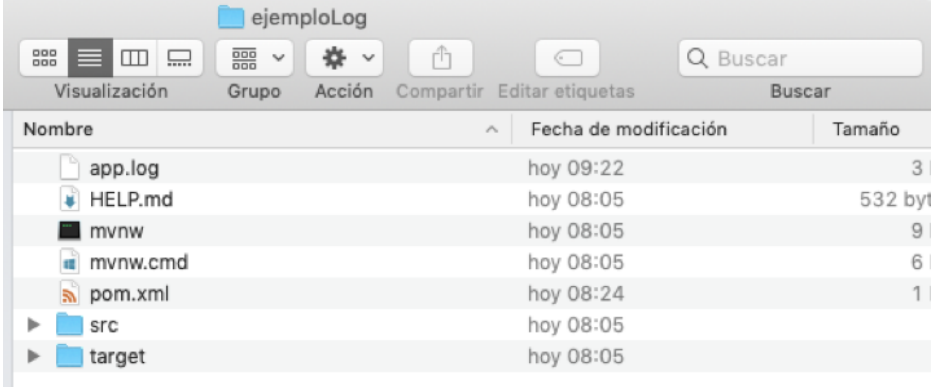

Imagen 1. Raíz de nuestro proyecto.
Fuente: Desafío Latam.

Abriendo el archivo *log*, vemos que se genera texto plano, con la información presentada en la consola, es exactamente igual a la que se ve al iniciar la aplicación.

```
2019-06-05 09:22:34.020  INFO 54359 --- [main] com.log.ejemploLog.App                    : Starting App on MacBook-Pro-de-Cristian.local with PID 54359 (/Users/
cristianfariasgonzalez/proyectos_java/spring/ejemploLog/target/classes started by cristianfariasgonzalez in /Users/cristianfariasgonzalez/proyectos_java/spring/ejemploLog)
2019-06-05 09:22:34.023  INFO 54359 --- [main] com.log.ejemploLog.App                    : No active profile set, falling back to default profiles: default
2019-06-05 09:22:34.987  INFO 54359 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2019-06-05 09:22:35.015  INFO 54359 --- [main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-06-05 09:22:35.016  INFO 54359 --- [main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.19]
2019-06-05 09:22:35.099  INFO 54359 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-06-05 09:22:35.099 DEBUG 54359 --- [main] o.s.web.context.ContextLoader            : Published root WebApplicationContext as ServletContext attribute with name
[org.springframework.web.context.WebApplicationContext.ROOT]
2019-06-05 09:22:35.099  INFO 54359 --- [main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1036 ms
2019-06-05 09:22:35.230 DEBUG 54359 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Patterns [/**/favicon.ico] in 'faviconHandlerMapping'
2019-06-05 09:22:35.386  INFO 54359 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-06-05 09:22:35.400 DEBUG 54359 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : ControllerAdvice beans: 0 @ModelAttribute, 0 @InitBinder, 1 RequestBodyAdvice, 1
ResponseBodyAdvice
2019-06-05 09:22:35.460 DEBUG 54359 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : 2 mappings in 'requestMappingHandlerMapping'
2019-06-05 09:22:35.474 DEBUG 54359 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Patterns [/webjars/**, /**] in 'resourceHandlerMapping'
2019-06-05 09:22:35.484 DEBUG 54359 --- [main] .m.m.a.ExceptionHandlerExceptionResolver : ControllerAdvice beans: 0 @ExceptionHandler, 1 ResponseBodyAdvice
2019-06-05 09:22:35.645  INFO 54359 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2019-06-05 09:22:35.650  INFO 54359 --- [main] com.log.ejemploLog.App                    : Started App in 2.029 seconds (JVM running for 2.713)
2019-06-05 09:22:35.652  INFO 54359 --- [main] com.log.ejemploLog.App                    : info log
2019-06-05 09:22:35.652  WARN 54359 --- [main] com.log.ejemploLog.App                    : warning log
2019-06-05 09:22:35.653 ERROR 54359 --- [main] com.log.ejemploLog.App                    : error log
```

Imagen 2. Revisando el contenido del Log.
Fuente: Desafío Latam.

Conclusiones

Siempre un archivo de log será un texto plano, es común ver que los archivos de log son archivos con extensión log (por buenas prácticas). Todos los servicios que se instalan en el computador poseen su propio log de información, de los cuales todos tienen los mismos niveles, ya sea DEBUG, INFO, WARN o ERROR. Y dependerá del nivel definido para que se almacene en los archivos.