Spring boot - Logging

If we do not provide any logging specific configuration, we will still see logs printed in "console". These are because of **default logging support** provided in spring boot which uses **Logback**.

Spring boot's internal logging is written with *Apache Commons Logging* so it is one and only mandatory dependency. Till, boot 1.x – we had to import it manually. Since boot 2.x, it is downloaded transitively. To be more precise, spring-boot-starter-web depends on spring-boot-starter-logging, which pulls in spring-jcl for us

Logback supports ERROR, WARN, INFO, DEBUG, or TRACE as logging level. By default, logging level is set to **INFO**. It means that code>DEBUG and TRACE messages are not visible.
To enable debug or trace logging, we can set the logging level in application.properties file. We can apply logging levels to specific packages as well. It can be done either in console or application.properties file. If the log level for a package is defined multiple times with different log levels, the lowest level will be used. TRACE is lowest and ERROR is highest.

The default log statement formatting

```
%clr{%d{yyyy-MM-dd HH:mm:ss.SSS}}{faint}

%clr{${LOG_LEVEL_PATTERN}}

%clr{${sys:PID}}{magenta}

%clr{---}{faint}

%clr{[%15.15t]}{faint}

%clr{%-40.40c{1.}}{cyan}

%clr{:}{faint} %m%n${sys:LOG_EXCEPTION_CONVERSION_WORD}
```

- Date and Time — Millisecond precision.
- Log Level — ERROR, WARN, INFO, DEBUG or TRACE.
- Process ID.
- A — separator to distinguish the start of actual log messages.
- Thread name — Enclosed in square brackets (may be truncated for console output).

- Logger name — This is usually the source class name (often abbreviated).
- The log message

Below is an example from my project:-

```
2021-03-21 13:31:03.818

INFO

1148

---

[nio-8083-exec-4]

c.p.d.controller.ServiceController

: “Service Request ID created : 4 for customer:Auroville Bakery”
```

Logging file:-

By default spring boot logs to console only. If we want to enable file logging, we can easily do it using simple property logging.file or logging.path.
When using logging.path, it will create a file named spring.log in mentioned package.

```
# Output to a temp_folder/file
logging.file=c:/temp/application.log


#logging.path=/my-folder/
```

Spring boot logging levels

When a message is logged via a Logger it is logged with a certain log level. In the application.properties file, you can define log levels of Spring Boot loggers, application loggers, Hibernate loggers, Thymeleaf loggers, and more. To set the logging level for any logger, add properties starting with logging.level.

Logging level can be one of one of TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF. The root logger can be configured using logging.level.root.

*We write as "**logging.level**.(packageName)=INFO or ERROR"*

```
#logging.level.root=WARN

logging.level.org.springframework.web=ERROR
logging.level.com.howtodoinjava=DEBUG
```

Spring boot logging patterns

To change the logging patterns,
use `logging.pattern.console` and `logging.pattern.file` properties

```
# Logging pattern for the console
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n

# Logging pattern for file
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n
```

Active profiles to load environment specific logging configuration

It is desirable to have multiple configurations for any application – where each configuration is specific to a particular runtime environment. In spring boot, you can achieve this by creating multiple `application-{profile}.properties` files in same location as `application.properties` file.

Profile specific properties always override the non-profile specific ones. If several profiles are specified, a last wins strategy applies.

If I have two environments for my application i.e. `prod` and `dev`. Then I will create two profile specific properties files.

application-**dev**.properties

```
logging.level.com.howtodoinjava=DEBUG
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n
```

application-**prod**.properties

```
logging.level.com.howtodoinjava=ERROR
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n
```

To supply profile information to application, property `spring.profiles.active` is passed to runtime.

```
$ java -jar -Dspring.profiles.active=prod spring-boot-demo.jar
```