**********Documentation By Nagaraj S Kharvi**********

What Is Logging?

Logging is a vital part of all applications and brings benefits not only to us developers but also to ops and business people. Spring Boot applications need to capture relevant log data to help us diagnose and fix problems and measure business metrics.

The Spring Boot framework is preconfigured with Logback as a default implementation in its opinionated framework.

Why Is Logging Important?

The decisions on what to log and where are often strategic and are taken after considering that the application will malfunction in live environments. Logs play a key role in helping the application to recover quickly from any such failures and resume normal operations.

Spring Boot's Default Logging Configuration

Java has inbuilt logging feature (JDK Logging) which is located in java.util.logging. It has a Logger class using that we can start logging and it has concept of Log levels
* inform(String msg)
* severe(String msg)
* fine(String msg)

It has concept of Log Handlers i.e How to handle logging request
* Console Handler
* File Handler

So that's it! we can use this logs but the problem is many developers started complaining about this logs because it has
* Performance Issues
* Not flexible
* Not enough features

So many logging frameworks started coming out since
* need for a logging library
* out of the box functionality not good enough

People started creating logging libraries
Popular Libraries
* Log4J

Instead of Handlers Log4J has concept of appenders
* File
* Console
* Rolling file (Daily)
* JDBC (Log to DB)
* SMTP
* JMS

Also it has got different formatting options. So this is the end of it?
No. There was a problem when switching from Log4J and JDK logging. Thats where Facade libraries came into picture [Facade (False Appearance)]
* Apache Commons
* SLF4j - popular

Facade libraries contains no implementation it just a interfaces. These libraries provides api for standard logging then these libraries internally call another library for actual logging.
Two currently popular logging libraries
Logback
Log4J2 (Successor of Log4J with additional features and improved performance)

Both these has support for SLF4J logs
Example:

Your Code ->
Your Code ->
Your Code -> Logging Wrapper Class -> Logging library
Your Code ->
Your Code ->

When we change logging framework we just need to change only in this Logging wrapper class

Assume this example:
Lets say you are developing an application and in the end you want to integrate your application with several existing applications. In this case suppose if you are using Logback and other applications are using someother logging framework then you will end up with multiple logging frameworks doing separate logging.

So if you use SLF4J you don't have to care what logging other applications are using.

Recommended Logging Strategy
Use SLF4J
Actual Library can be Logback or Log4J2
Don't use older Log4J
Logback is slightly more popular
Log4J2 has slightly better features
- Lazy loading messages
- Async Logging

**********Documentation By Nagaraj S Kharvi**********