Logging in Spring Boot

We need to track what's happening & where it is happening in your application, For that we need to used to Logging to track all details in our application.

Logging In Spring Boot
application
what happening in your application
Track
logging

Spring Boot by-default provides spring logback/SLF4j framework in the applications.

SLF4j - API standards or specification

logback - implementation or execution

we can also use

log4j

Logging Level- Importance of log messages

1. **ERROR** -
    a. should be used when the application hits an issue preventing one or more functionalities from properly functioning.
    b. The ERROR log level can be used when one of the payment systems is not available, but there is still the option to check out the basket in the e-commerce application or when your social media logging option is not working for some reason.
    c. You can also see the ERROR log level associated with exceptions. (eg. database exception)
2. WARN
    a. the log level that indicates that something unexpected happened in the application.
    b. For example a problem, or a situation that might disturb one of the processes, but the whole application is still working. (for depreciated method, might chance of memory insufficient,)
3. INFO
    a. the standard log level indicating that something happened, application processed a request, etc. (start-up messages, or to indicate particular flow is done)
    b. The information logged using the INFO log level should be purely informative and not looking into them on a regular basis shouldn’t result in missing any important information.
4. DEBUG
    a. less granular compared to the TRACE level, but still more than you will need in everyday use.
    b. The DEBUG log level should be used for information that may be needed for deeper diagnostics and troubleshooting.
5. TRACE
    a. log events with this level are the most fine-grained and are usually not needed unless you need to have the full visibility of what is happening in your application and inside the third-party libraries that you use.
    b. You can expect the TRACE logging level to be very verbose.

By-default ERROR,WARN & INFO are enabled

To enabled remaining logging levels - use below commands

1. logging.level.-package-name

to enable from all processess logging, use package name - root

Eg. - logging.level.root = TRACE

for our package

Eg. - logging.level.in.api = TRACE

We can customized our logging based on configuration.xml file

```
<configuration>

            <!-- appender for console logging -->
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleA
    <!-- encoders are assigned the type
         ch.qos.logback.classic.encoder.PatternLayoutEncoder
    <encoder>
      <pattern>%d{dd/LL/YYYY hh:mm:ss.SSS: a} [%thread] %-5le
    </encoder>
  </appender>
            <!-- appender for file logging -->
  <appender name="FILE_SOUT" class="ch.qos.logback.core.rolli
    <file>logs/backend-project.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.Tim
            <fileNamePattern>logs/backend-project-%d{yyyy-MM-
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>%d{dd/LL/YYYY hh:mm:ss.SSS: a} [%thread]
        </encoder>
  </appender>

  <!-- Logger for File logging -->
```

```xml
  <logger name="in.api" level="TRACE" addivity="false">
    <appender-ref ref="FILE_SOUT"/>
  </logger>

  <!-- Logger for Console logging -->
  <root level="INFO">
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

Refer logging docs :

https://logback.qos.ch/manual/introduction.html