========
Logging
========

-> The process of storing application execution details to a file is called as Logging.

-> With log messages we can understand execution flow of the application.

-> We can understand exceptions occuring in the project by seeing log messages.

Logging Frameworks
----
1) Log4J
2) Log4J2
3) LogBack
4) LogStash

Log Monitoring Tools
----
1) Putty
2) WinScp
3) ELK
4) Splunk (Licensed)

Logging Architecture
----

1) Logger : This class providing methods to generate log messages

2) Layout : It represents log message structure (format of log msg)

3) Appender : It is used to write log message to destination

4) Destiation : It can be console/file/database

Note: We will use files to store our log messages.

===============
Logging Levels
================

1) TRACE
2) DEBUG
3) INFO   (it is default log level in boot application)
4) WARN
5) ERROR

6) FATAL

=====================
Log Level Hierarchy
=====================

TRACE > DEBUG > INFO > WARN > ERROR > FATAL

=> When we set one Log level, application will print log message from that level to all higher level messages.

=> In Spring Boot by default it will use level as INFO

-> In Spring Boot by default it will use ConsoleAppender

-> To generate log msgs in log file we have set below property in application.properties file

----

```
logging.level.root = DEBUG
logging.file.name=app.log
```

```
package in.ashokit.rest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class MessageController {

private Logger logger = LoggerFactory.getLogger(MessageController.class);

@GetMapping("/welcome")
public String welcomeMsg() {
logger.debug("this is debug msg from welcome.....");
logger.info("welcomeMsg() execution started.....");

String msg = "Welcome To Ashok IT...";

try {
int i = 10 / 0;
} catch (Exception e) {
logger.error("Exception Occured" + e.getMessage());
}

logger.warn("This is warning from welcome method...");
logger.info("welcomeMsg() execution ended...");
return msg;
}

@GetMapping("/greet")
public String greetMsg() {

logger.debug("this is debug msg from greet.....");
logger.info("greetMsg() execution started...");
String msg = "Good Morning...";

logger.warn("This is warning from greet method...");

logger.info("greetMsg() execution ended...");
return msg;
}

}
```

==================
Rolling Appenders
==================

1) Size Based Rolling

2) Time Based Rolling

=> We can customize springboot application log configuration by creating logback.xml file under src/main/resources folder.

============================ logback.xml file ======================

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<appender name="Console"
class="ch.qos.logback.core.ConsoleAppender">
<encoder>
<pattern>%d [%thread] %-5level %-50logger{40} - %msg%n</pattern>
</encoder>
</appender>
<appender name="RollingFile"
class="ch.qos.logback.core.rolling.RollingFileAppender">
<file>MyApp.log</file>
<encoder>
<pattern>%d [%thread] %-5level %-50logger{40} - %msg%n</pattern>
</encoder>
<rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
<fileNamePattern>MyApp-%d{yyyy-MM-dd}.%i.log</fileNamePattern>
<maxFileSize>1MB</maxFileSize>
```

```
<maxHistory>30</maxHistory>
<totalSizeCap>10MB</totalSizeCap>
<cleanHistoryOnStart>true</cleanHistoryOnStart>
</rollingPolicy>
</appender>
<root level="INFO">
<appender-ref ref="Console" />
<appender-ref ref="RollingFile" />
</root>
</configuration>
```

==

================
Log Monitoring
================

=> It is the process of checking logs of application to understand problems occuring in the application.

=> We have several tools to perform log monitoring

1) Putty
2) WinSCP
3) ELK
4) Splunk

======
Putty
=======

=> Putty is a CLI based software

=> It is used to connect from windows machine to Linux Machine

=> To connect to linux machine we need machine details

IP : 192.168.1.2

Username : loguser

pwd : log@123

================
Putty Commands
===============

cat filename   : Get file data from top to bottm

head filename : To get first 10 lines of file

tail filename : To get last 10 lines of file grep 'Exception' filename : It will print the lines which contains Exception

========
WinScp
========

=> It is GUI based software

=> It is used to connect from windows to linux machines

=> Using this WinSCP we can upload and download files from windows to linux and vice versa

=======
Splunk
=======
=> It is commercial log monitoring software.

====
ELK
=====

E - Elastic Search

L - Log Stash

K - Kibana

=> The above 3 open source products are used for log monitoring.

1) Maven
2) Git Hub
3) Bit Bucket
4) JIRA
5) Logging Tools (LogBack)
6) Log Monitoring Tools (Putty, WinScp and Splunk)