=> Spring core -> ✓

springBoot => { JDBC = } ; JPA ; Data JPA }

springBoot - profiles } ✓

=> Logging is

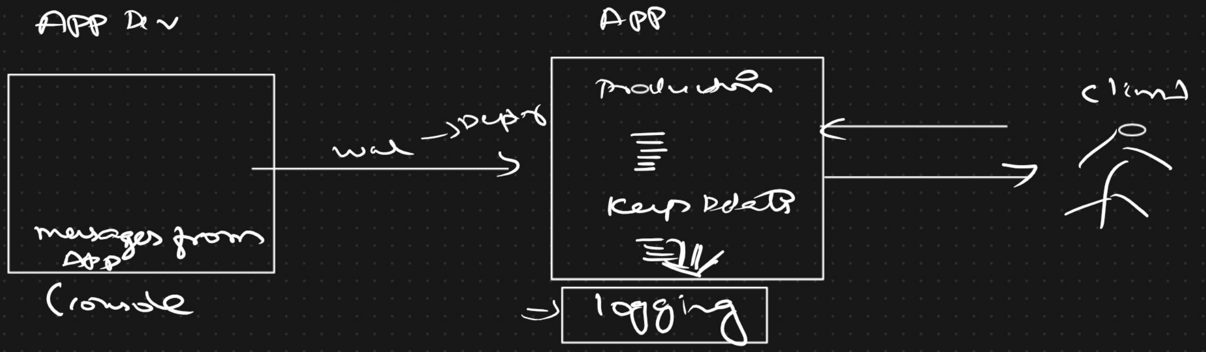
↳ messages from app (success, failure, ----) }

What is Logging?
 The process of keeping track of application's flow of execution is called "Logging".

-->using logging generated log messages we can find the state of the application execution in any given date and time

 -->logging keeps track of components and code that are involved in application execution.

Note: Auditing is one of the use case of "logging".
 The special log messages of logging that keeps track of user activities w.r.t application execution is called "Auditing".



resolve your => Bug

client

Develop APP } Testing ⟶ UAT ⟶ Production

onsite

Logging : log4j → log4j2 } SL4J

⇓

log4j Apache

JDBC → MySQL Driver

=> Javalogging → util }

=> Commons logging

≡

≡

jboss

≡

SL4J
 -> It stands for Standard Simple logging facade for Java
 -> It provides abstraction on mulitple logging api/tools/framework and provides
unified api for logging by internally using our choice of logging api

log4J

Log4j Advantages
================
1. Allows to categorize the log messages and we can add priorities for log messages.
DEBUG<INFO<WARN<ERROR<FATAL --> Log4j

TRACE<DEBUG<INFO<WARN<ERROR<FATAL --> Log4j2

Use DEBUG level for normal confirmation code flow statements
eg: main() method start,main() end ,start of b.method and end of b.method etc...

Use INFO level for important confirmation of code flow statements
eg: connection established with DB s/w, login successfull, OTP generated,.....

Use WARN level to write log messages for code that should not used/executed but some home used and executed.
eg: especially useful when we used deprecated api's/poor api's on temporary basis.

Use ERROR level to write log messages from know exceptions related to catch blocks like (SQLException e),catch(IllegalArgumentException e)

Use FATAL level to write log messages from unknow exceptions related catch blocks like (Exception e),catch(Throwable t) etc.

=> log messages :- can be stored

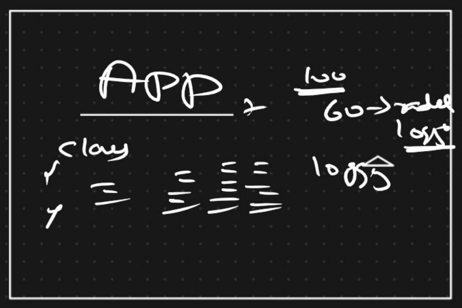
→ console , File , email , Database etc .....

3 → Things ⇒ logging (log4J)

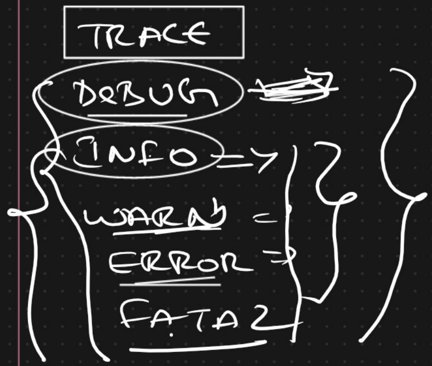
1 → logger → object → Base object → It enables logging for a class
2 → Appender (where to store) → FileAppender, ConsoleApp ----
3 → Layout (How to store format) xml, Html - pattern.

class Entity

state, setter, getter } logging



⇒ → pre-defined methods by Based - TYPE



rootlogger ⇒ ~~INFO~~ WARN

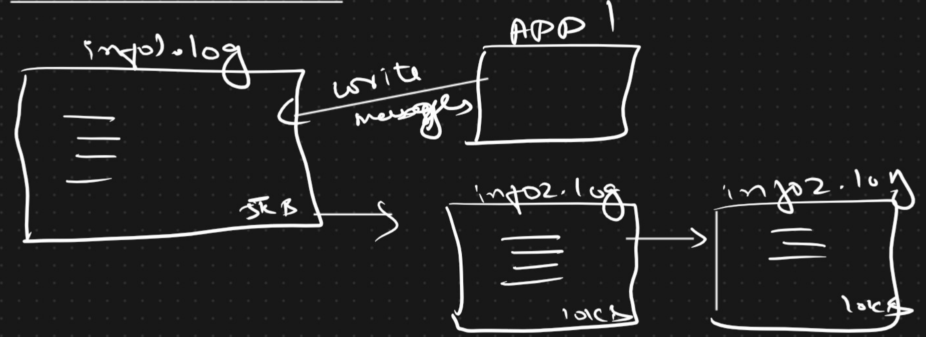
Logger =
logger.debug(" ")

which ⇒ logger ⇒

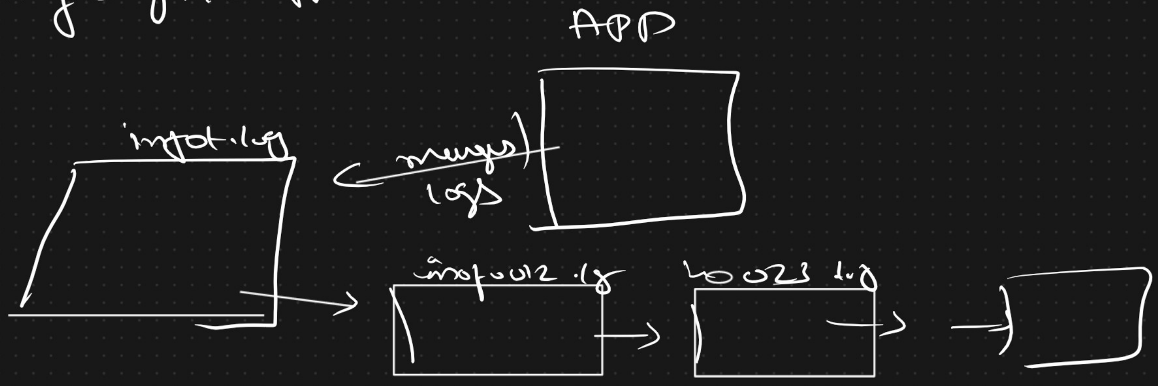
Appender ⇒ where to store ⇒ console
↳ FileAppender
↓
RollingFileAppender

RollingFileAppender

DailyRollingFileAppender



DailyRollingFileAppender


APP
infot.log
messages
logs
info0012.log
40023.log