Need of Logging Tools

pre-requisite :: core Java , JDBC basics

confinent = the reusable class SLF4J :: Simple Logging Façade For(4) Java

Login/Signin :: giving username and password to get into App (eg: gmail login , FB login and etc..)
Logging :: keeping track of App's flow (for given task execution which lines of the code are executed)

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 state of the application execution on any given date and time.
=> Logging keeps track of the comps and code that involved in the Application's execution..

What is the difference between Logging and auditing?

=> Logging keeps track of various comps and code that are involved in the execution of the Application
[gets class names, method names, blocks , modules that are involved in the Application execution]

=> Auditing keeps track of various activities done by enduser while operating the application.
[gets the activies like user signed in , opened inbox, replied mail , ... signedout]

note: Auditing is a use-case of logging.. [The special log messages of logging that keeps track user activities in the application's execution is called Auditing]

use cases of logging

=>while performing unit testing .. if the test result negative .. we need debug the code to know the reason .. In that process log messages are useful for developers.
=>while fixing the bugs given by Tester ... the developers needs to know state of the application's execution that needs the support of log messages.
=>After releasing project .. we get production bugs from client org's end users through onsite.. the offshore team members uses the log messages to know state of execution when bug was raised through log messages while fixing bug s

After releasing project to client org
=> s/w company location is called offshore/ offsite Location
=> Client org location is called onsite/onshore location

Offshore location	Onsite Location
POLARIS (hyd)	citi bank (london)
CTS (hyd)	AMEX (USA)
TCS (blore)	Central Govt Project (Delhi)
Result (hyd)	Indian Railway (Delhi)

=> if the project is inhouse project then the offshore and onsite locations are same
=> If TCS hyd is dealing Telangana GOVT (e-governance) project the both locations will be in same city Hyd.

Just before realing project the team will be divided in two parts
part1] Onsite team (goes to client org Location to recieve,Install, maintains the project)
part2] offshore team (stays in s/w company to release the project and to fix production bugs given by client org)

offshore team is supporting team to onsite team..

What is the difference logging and debugging?

Ans) Using logging we can keep track of application flow and that flow related log message can be used for long time to fix the bugs and issues that are find in project test and in live project (production env)

Using Debugging , we understand to current code flow of execution to resolve errors and to improve code for betterment

=> Using Logging activity we can also do lots of Dubugging on the code that is executed long back

=> While maintaining projects that is there production env.. if the project is down all of sudden then the maintainance team recives exceptions related special log file to know the reasons and fix the problems..

=>while taking backup of Db s/w and bringing Db s/w back to normal state after crash. we take support of log messages
=> While performing Tx Mgmt (executing logics by applying do every thing or nothing pricipla) we need log message support. (Transaction Mgmt)

etc...

Every project contains the following 4 env...
(a) Dev env
(b) TEsting env
(c) UAT env.. (User Acceptance Test Env)
(d) Product env..

@ s/w company
@Client org..

if both s/w company and Client org are using Cloud env.. for development and production then Cloud itself acts as onsite and offshore locations.
[This cloud management can be done using AWS, Azure , Google cloud and etc..]

Using Cloud for development and production means taking s/w and hardware setup on rental basis for developing , testing project and also including installing and maintaining project after releasing the project.

note:: The code related logging will be added to project during the development .. but it will be used in different phases of Project..
note: The log messages in code will be added during the development, but will be used during the "dev","test","uat","prod" env.. of the project

=>We can do logging in any java application by using S.o.println(-) and system.err.println(-) method ,but the following limitations are there

=>we can write log messages only in console monitor which will be lost after certain amount of time.
=> We can not categorize log messages
=> we can not format log messages
=> we can not write log messages to different destinations like file, Db s/w , Mail SErver and etc..
=> we can not see old log messages after few days/hours (particular date and time log messages)
=> we can not filter log messages while retrieving ..
=> writing messages to console monitor using s.o.p(-) or s.e.p(-) is a single threaded process . so if multiple log messages at a time from the App.. they will be rendered to the app having delay..
and etc...

To overcome these problems we can use(opt) alternates for S.o.p or S.e.p based logging

Logging Tools/API:
=> Java assertions --> from Idk (sun Ms)
=> java logging api (java.util.pkg [from Idk (sun Ms)]
=> commons logging --> from apache
=> jboss logging --> from jboss (RedHat) (log4j moves Logging for Java)
=> log4j --> from apache (Best)
=> logback --> from adobe
and etc..

Scop -> System.out.println( )
S.o.p -> System.err.println(-)

=> SLF4J (simple logging facade for java) is not a task logging api .. It provides abstraction on multiple logging apis/tools /frameworks and provides unified api for logging by internally using our choice logging api..

[SLF4J is like universal Remote linked with multiple independent remotes which can be used various devices of the home

(very very Best)
SLF4J (framework)

=>SLF4J with Logback
=>SLF4J with Log4J
=>SLF4J with commons -logging api

Logback | Log4J | Commons-logging | other logging API

log4J (Logging for Java)

type :: Logging tool for java
version : 1.x (stable) , 2.x
vendor :: apache
open source (free and source code is availble)
jar file representing api :: log4j-x.xx.x.jar
(we can donwload from mvnrepository.com)

To download log4j s/w :: https://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.17/log4j-1.2.17.zip (not required in maven /gradel env..)
(extract the zip file)

Log4j advantages

(1) Allows to categorize log messages and we can prorities for log messages
DEBUG< INFO < WARN< ERROR <FATAL

=>use DEBUG Level for normal confirmation code flow statements like
eg: main(-) method start , main(-) method end , start b.method , end b.method and etc..

=>use INFO Level for important confirmation code flow statements like
eg: connection established with Db s/w , Log in is successful , OTP generated, Token is accessed and etc..
=>use WARN level to write log messages for code that should not used/executed but some how used and executed for temporary adjustment
=> especially useful when programmer uses deprecated apis/ poor api on temporary basis.

=>use ERROR level to write log messages from known exceptions related catch blocks blocks like catch(SQLException se), catch(IllegalArgumentException iae) and etc...

=>use FATAL level to write log messages from unknown exceptions related catch blocks blocks like catch(Exception e) or catch(Throwable e) and etc...

In Testing env..what is the difference b/w bug and issue?
=>Bug means code is there but expected funcationality is not coming (wrong logic)

Need of Logging Tools

======

==========

pre-requisite :: core Java, JDBC basics.

what is Logging

==========

of

comonent = the resuable class

SLF4J:: Simple Logging Façade For(4) Java Login/Signin :: giving username and password to get into App (eg: gmail login, FB login and etc..) Logging :: keeping track of App's flow (for given task execution which

=> The process of keeping track application's flow of execution is called logging

lines of the code are executed)

=> Using logging generated log messages we can find state of the application execution on any given date and time.

are

=> Logging keeps track of the comps and code that invloved in the Application's execution..

What is the difference between Loggingnd auditing?

=> Logging keeps track of various comps and code that are involved in the execution of the Application [gets class names, method names, blocks,modules that are involved in the Application execution]

=> Auditing keeps track of various activities done by enduser while operating the application. [gets the activies like user singned in, opened inbox, replied mail, ... signedout]

note:: Auditing is a use-caselogging.. [The special log messages of logging that keeps track user activities

use-cases of logging

=========

in the application's execution is called Auditing]

is

to

=>while performing unit testing .. if the test result negetive .. we need debug the code to know the reason.. In that process log messages are useful for developers.

=>while fixing the bugs given by Tester ... the developers needs to know state of the applicaiton's execution that needs the support of log messages. given by

team

=>After releasing project.. we get production bugs from client org's end users througn onsite.. the offshore team members uses the log messages to know state of execution when bug was raised through log messages while fixing bug s

After releasing project to client org

=> s/w company location is called offshore/ offsite Location

=> Client org location is called onsite/onshore location

Offshore location

=======

POLARIS (hyd)

CTS (hyd)

TCS (blore)

SPSoft (hyd)

Onsite Location

======

citi bank (london)

AMEX (USA)

Central Govt Project (Delhi)

Indian Railway (Delhi)

=> if the project is inhouse project then the offshore and onsite locations are same

=> if TCS hyd is dealing Telangana GOVT (e-governance) project the" both locations will be in same city Hyd.

Just before realising project the team will be divided in two parts

part1) Onsite team (goes to client org Location to recieves,install, maintain s the project)

part2) offshore team (stays in s/w company to release the project and to fix production bugs given client org)

offshore team is supporting team to onsite team..

=> While maintaining projects that is there production env.. if the project is down all of sudden

sends

then the maintainance team exceptions related special log file to know the reasons and fix the problems..

What is the difference logging and debugging?

Ans) Using logging we can keep track of application flow and that flow related log message can be used for long time to fix the bugs and issues that are find in project test and in live project (production env)

Using Debugging, we understand to current code flow of execution to resolve errors and to improve code for betterment

=> Using Logging activity we can also do lots of Dubugging on the code that is executed long back

=>while taking bacup of Db s/w and bringing Db s/w back to normal state after crash. we take support of log mesages While performing Tx Mgmt (executing logics by applying do every thing or nothing priciple) we need log message support. (Transaction Mgmt)

ane tc...

Every project: contains the following 4 env...

(a) Dev env

(b) TEsting env

@s/w company

(c) UAT env.. (User Acceptence Test Env)

(d) Product env..

@Client org..

Using Cloud for development and production measn taking s/w and hardware setup on rental basis

for developing, testing project and also reciving

to

installting and maintaning porject after releasing the project.

if both s/w company and

Client org are using Cloud env..

for development and production then Cloud itselt acts as onsite and offshore locations.

[This cloud management can be done using AWS, Azure, Google cloud and etc..]

note:: The code related logging will be added to project during the devleopment .. but it will be used in different phases of Project.. note:: The log messages to code will be added during the development, but will be used during the "dev","test", "uat", "prod" env.. of the project

=>WE can do logging in any java application by using S.o.println(-) and system.err.println(-) method, but the following limitations are there

=>we can write log messages only to console monitor which will be lost after certain amount of time. => We can not categorize log messages

=> we can not format log messages

=> we can not write log messages to different destionations like file, Db s/w, Mail Server and etc..

=> we can not see old log messages after few days/hours (particular date and time log message => we can not filter log messages while retrieving ..

=> writing messages to console monitor using s.o.p(-) or s.e.p(-) is a single threaded process. so if multiple log messages at a time from the App.. they will be rendered to the app having delay.. are generated

and etc..

use

To overcome these problems we can multipel alternates for s.o.p or S.e.p based logging

=> Java assertions --> from Jdk (sun Ms)

in

=> java logging api given java.util pkg (from Jdk (sun Ms))

S.o.p-> System.out.println(-)

S.e.p -> System.err.println(-)

Logging

Tools/API

=> commons logging --> from apache

=> jboss logging -> from jboss (RedHat)

(Log4j means Logging for Java)

=>log4j --> from apache (Best)

=> logback --> from adobe

and etc..

=> SLF4J (simple logging facade for java) is not a basic logging api .. it provides abstraction on multiple logging apis/tools/frameworks and provides unified api for logging by internally using our choice logging

api..

(SL4FJ is like universal Remote linked with

multiple indepedent remotes which can be used various devices of the home

(very very Best) SLF4J (framework)

Commons-logging

Logback

Log4J

=>SLF4J with Logback

=>SLF4J with Log4J

=>SLF4J with commons-logging api

other logging API

Log4J (Logging for Java)

======

type:: Logging tool for java

version 1.x (stable), 2.x

vendor :: apache

open source (free and source code is avaible)

jar file representing api :: log4j-<ver>.jar

(we can donwload form mynrepository.com)

To download log4j s/w :: http://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.17/log4j-1.2.17.zip (not required

(extract the zip file)

Log4j advantages

======================

add

(1) Allows to categorize log message and we can prorites for log messages

DEBUG< INFO < WARN< ERROR <FATAL

=>use DEBUG Level for normal confirmation code flow statements

like

eg: main(-) method start, main(-) method end, start b.method, end b.method and etc..

=>use INFO Level for important confirmation code flow statements

like

eg: connection established with Db s/w, Log in is successful, OTP generated, Token is accessed and etc..

be

in maven /gradel env..)

=>use WARN level to write log messages for code that should not used/executed but some how used and executed for temporary adjustment

=> especially useful when programmer uses deprecated apis/ poor api on temporary basis.

=>use ERROR level to write log messages from known exceptions reated catch blocks blocks like catch(SQLException se), catch (IllegaAgumentException iae) and etc...

=>use FATAL level to write log messages from unknown exceptions reated catch blocks blocks like catch(Exception e) or catch(Throwable e) and etc...

In Testing env..what is the difference b/w bug and issue?

=>Bug means code is there but expected funcationality is not coming (wrong logic)

we.

eg: when click on home hyper link it is going to about us page

=>issue means the feature /functionality itself missing

eg:: home

hyperlink itself is missing

what is the diffrence b/w bug and error?

Ans) bug means wrong logics giving wrong results with out giving/throwing exceptions (no abnormal termintation

of application execution)

eg:: home hyperlink is taking u to about us page

error/failure means wrong logics throwing exceptions and stopping the application execution..

eg: StackOverflow, stackunderflow and tc..

(2) Allows to write/record log messages to different destinations like console, files, Db s/w, mail server and etc..

(3) Allow to format the log messages using different layouts (HtmlLayout, XmlLayout and etc...)

(4) Allows to retrieve log messages by applying filters..

ALL<DEBUG<INFO<WARN<ERROR<FATAL<OFF

S

=>if the logger level to retrive log message is INFO then we get all log message whose priority level

is INFO and higher (INFO,WARN, ERROR,FATAL messages)

=>if the logger level to retrive log messagesis ERROR then we get all log messages whose priority level

is ERROR and higher (ERROR,FATAL messages)

=>if the logger level to retrive log message is ALL then we get all log messages

=>if the logger level to retrive log message is OFF then logging will be disabled on the application.

app

note:: In realtime for every two log fiiles will be maintained

a) Common log file (records all log message end to end) (Logger level is ALL/DEBUG)

b) Exception log file (recodrs only ERROR and FATAL level log message ..useful when system/project is down)

to

(5) can change the inputs of the App related log4j either using properties or xml file

(6) Log4J can write log messages to files (or) console (or).. as parellel process.

(7) it is indurstry standard..

(Logger level is ERROR)

and etc.

VIILL