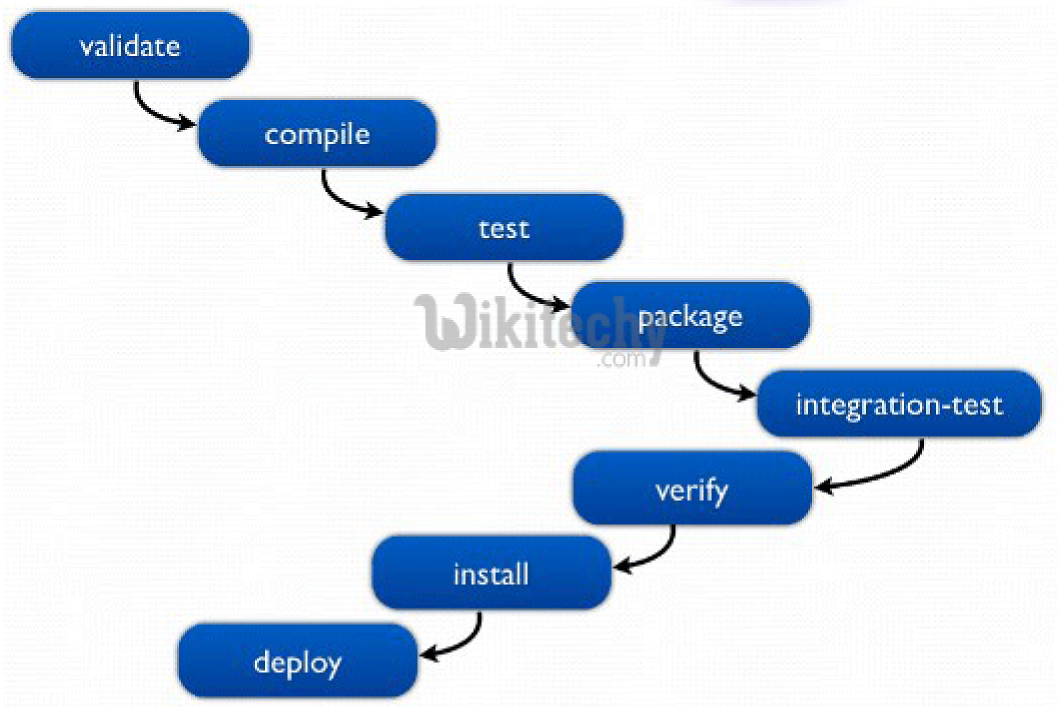
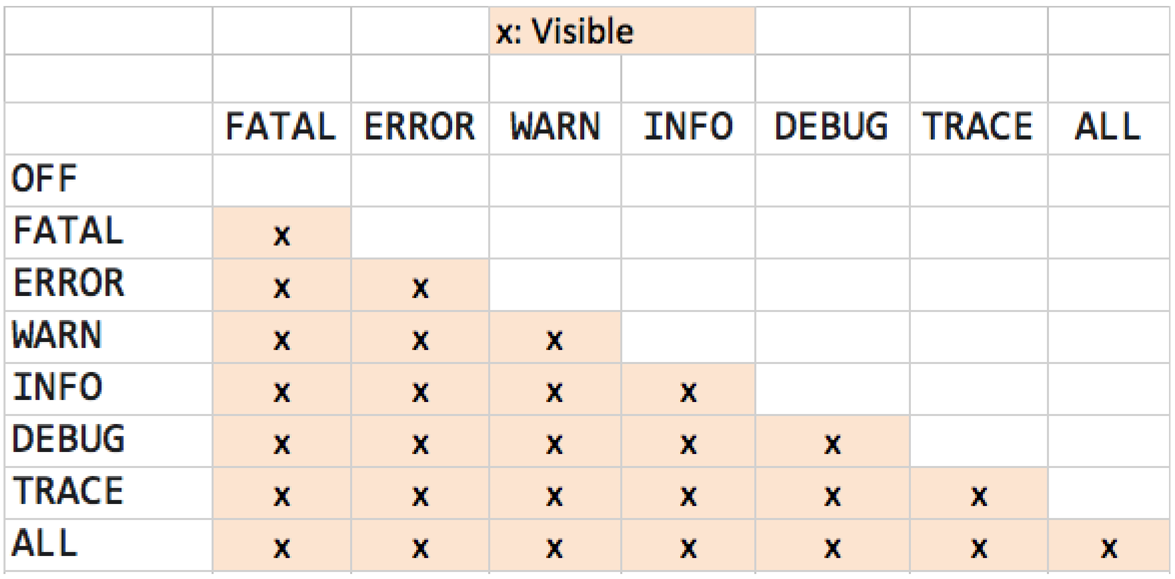
Topics: Introduction to Maven & Log4j

1. maven lifecycle



2. logging basics



			x: Visible				
	FATAL	ERROR	WARN	INFO	DEBUG	TRACE	ALL
OFF							
FATAL	x						
ERROR	x	x					
WARN	x	x	x				
INFO	x	x	x	x			
DEBUG	x	x	x	x	x		
TRACE	x	x	x	x	x	x	
ALL	x	x	x	x	x	x	x

Maven
======
Common problems and activities

=> mutiple jar , jar hell
=> How to find dependencies of one jar to another?
    versioning issues

=> Project structure, how to standerized project st
=> building , publishing and deployemnt?

Maven solve all above problem!

Maven hello World

Donwload maven http://maven.apache.org/download.cgi

set M2_HOME
C:\tools\maven3.1

PATH
%M2_HOME%\bin
----

build life cycle:

```
validate ->
        compile ->
                test->
                        package->
                                install (to local repo)->
                                        deploy (does not deploy to server, but to remote repo)
```

Create an core java project

Checking installation
    mvn -version

    reposotories:
        Local , Remote

    Command:
    mvn archetype:generate

    choose a number? 106 (go with it)
        Different archetype?model how u want to structure your probject!

    Version (version of archtype?)
        choose latest

    GroupId?
        Similer to package com.demo

    artififact id?
        name of application

        MavanTestApp (name of jar)
        version no?

    After crreating project
    ----

mvn compile (it also run junit test cases)
mvn package

Maven project in one go:

```
mvn archetype:create -DarchetypeGroupId=org.apache.maven.archetypes -DgroupId=com.demo.works -
                                        DartifactId=SampleProject
```

=> it will create an class and test case

=> then compile:mvn compile

update POM telling to use java 1.5

```
<build>
            <plugins>
            <plugin>
                        <artifactId>maven-compiler-plugin</artifactId>
                                    <configuration>
                                    <source>1.8</source>
                               <target>1.8</target>
                             </configuration>
            </plugin>
</plugins>
</build>
```

then run test case;
----
mvn test

convert this project to eclipse project
----
mvn eclipse:eclipse

Now browse this project in ide

problem:
----
unbound classpath repo error!: eclipse dont know maven repo

M2_REPO find .m2 dir

m2 eclipse plugin:

logging basics
====================

Logging?

=> logging is essential for debugging and for maintaing our application

=> We must know what is going in our application, specially when error come SOP and printing exception message is not good?

Writing system.out.println(" ");//Should not be used for debugging
Why?
----
as it is very hard to remove those unnessary Sop once coding is done

It may produce serious problem in production enveronment
headach for admin peoples

=> Real advantage of logging is that it can be enable/disable and debugging
messages can be directed to the file

Logging framewrok?

Log 4j
log back
Commons logging
java.util.logging

=> most commonly used one is log4j

=> we should not fix ourself with any one specific logging framework as we have to change as required.... go for facade ...use Simple Logging Facade for Java

SLF4j
=======

=> The Simple Logging Facade for Java or (SLF4J) serves as a simple facade or abstraction for various logging frameworks, e.g. java.util.logging, log4j and logback, allowing the end user to plug in the desired logging framework at deployment time

Levels of logging

ALL----------->log everything
DEBUG
INFO
WARN
ERROR
FATAL
OFF----------->Log nothing

maven dependencies:

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.2</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.2</version>
</dependency>
```

Ref:
-----
http://www.mkyong.com/logging/log4j-log4j-properties-examples/
http://www.mkyong.com/logging/log4j-hello-world-example/

Hello World:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Applications
 {
private static final Logger logger=LoggerFactory.getLogger(Applications.class);

        public static void main(String[] args)
        {

        System.out.println("Hello world logging");
        logger.info("stating logging!!!!");
        System.out.println("Hello world logging");
        logger.info("finished logging!!!!");
        }
}
```

Ex:

```
private static final Logger logger=LoggerFactory.getLogger(Applications.class);

logger.info("start logging");

String no="4x";
try
{
        Integer.parseInt(no);
}
catch(NumberFormatException ex)
 {
        logger.error("connot formet :"+no+" to and no....");
 }
```