06.04Apr. Dependencies & Testing

3 April 2023 08:36 PM

Dependencies

Executions Files
Direct Execution = .EXE, .cmd, .bat, .jar, etc File --> PATH
Silent Execution = .dll- Dynamic Link Library, .jar --> CLASSPATH

- Set CLASSPATH is used to Point to Dependencies(Library)
- Execute on Other Program wants to execute

$ java mspaint

$ java -cp c:\abc.jar notepad

$ java -cp c:\abc.jar;c:\xyz.jar wordpad

$ set classpath=%classpath%;c:\abc.jar;c:\xyz.jar;c:\pqr.jar;
$ java ms-word

Maven --> POM.xml

TestNG
- JUnit --> TestNG
- Unit Test Framework Test Next Generation
- Multiple tools Configuration file testng.xml
- @ --> Annotations ==> Use Cases
- TDD = Test Driven Development methodology

1. Write a Test Case | Blue
   i. Fail --> Refactor(Technical) | RED
      Pass --> Report Generate (Non-Technical) | GREEN

TestNG Manage:
---
1. **IDE** - Eclipse
2. **CLI** - JAR File (download jar, set classpath)
3. **Build** - Maven
4. **CI**- Jenkins

How TestNG Framework Works

Test Data --> Test Cases(Scenario) --> Build Tool(Maven) --> Generate Report --> Management/Client/Architectures

3,2 --> Test Cases(3+2, 3-2) --> Build Tool(Maven - mvn test) --> Generate Report(**xml**, h**tml**) --> Management/Client/Architectures/CRM/ERP/Email/Notification Service

Current User Home Directory/Folder
%userprofile% Windows, Unix(Linux/Mac) = ~

java -cp <JAR-PATHs> <output-Address>

Windows

java -cp "%userprofile%/.m2/repository/org/testng/testng/7.4.0/testng-7.4.0.jar;%userprofile%/.m2/repository/com/beust/jcommander/1.81/jcommander-1.81.jar;%userprofile%/.m2/repository/org/webjars/jquery/3.5.1/jquery-3.5.1.jar;target/classes;target/test-classes" org.testng.TestNG -testclass "com.baeldung.testng.DateSerializerServiceUnitTest"

Unix(Mac/Linux)

java -cp "~/.m2/repository/org/testng/testng/7.4.0/testng-7.4.0.jar;~/.m2/repository/com/beust/jcommander/1.81/jcommander-1.81.jar;~/.m2/repository/org/webjars/jquery/3.5.1/jquery-3.5.1.jar;target/classes;target/test-classes" org.testng.TestNG -testclass "com.baeldung.testng.DateSerializerServiceUnitTest"

Repository(Inter related Files = Project)
1. Source Code = Only Source Code(.java/ .py / .go) --> VCS Tool = **Github**
   BUILD Tool(Apache Maven)
2. Output Files = Artifact(.war, .jar , .msi) --> Artifactory Tool = JFrog/Azure Artifact
3. Complete OS = Image(iso, vbox, vdi,etc) --> Virtualization/Container Tool = Vagrant/**Docker**

Use Cases:
- System Requirements
- Helps to understand
  - the end-user actions
  - System Behaviour
- Pictorial action displayed

Officer1
Officer2 | Clerk
Officer3

Continues Integration/CD/CD/CM/ Tool
Open Source Jenkins
AWS - CodePipeline
Azure - Pipeline

Using TestNG to Test Cases:
$ mvn -P ExecuteSingleTest test
$ mvn -P ExecuteTestSuite test