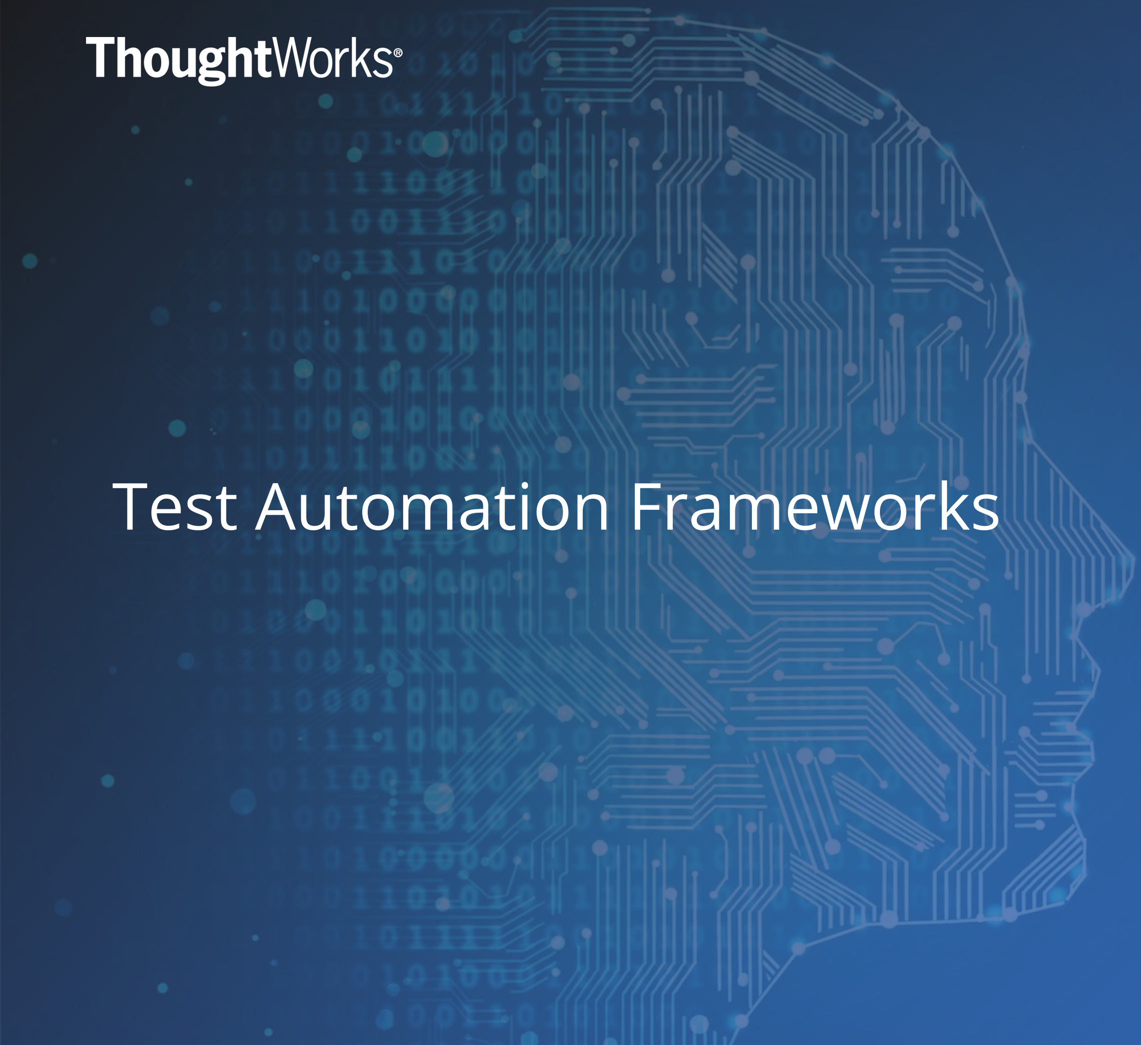
ThoughtWorks®



Test Automation Frameworks

Maven Theory - 15 - 20mins

Maven Demo - 15 mins

TestNG Theory - 10 Mins

TestNG Demo - with only one Test

Break Out room: 30 Mins

TestNG Demo for annotations, assertions - 15Mins

Breakout rooms : 30Mins

Webdriver Test Scenario 1: 30 Mins

Webdriver Breakout room :   45 Mins   - 1HR

Agenda

- Build Automation tool: **Maven**
  - **Maven Lifecycle**
  - **Maven Goals**
  - **Project Identifiers**
  - **Maven Demo**
- TestNG
  - **TestNG Annotations**
- Webdriver Test Scenario 1
- Webdriver Test Scenario 2

Build tools

Build tools are commonly known as programs that automate the process of building an executable application from source code

The activities include:

- Every project requires certain dependencies
  - How do we fetch these dependencies
  - The project dependencies are themselves dependent on other dependencies
- Compiling source code to the form of binary code.
- Packaging that binary code.
- Running the tests.

Example of Build Tools for Java

- Maven
- Gradle
- Ant

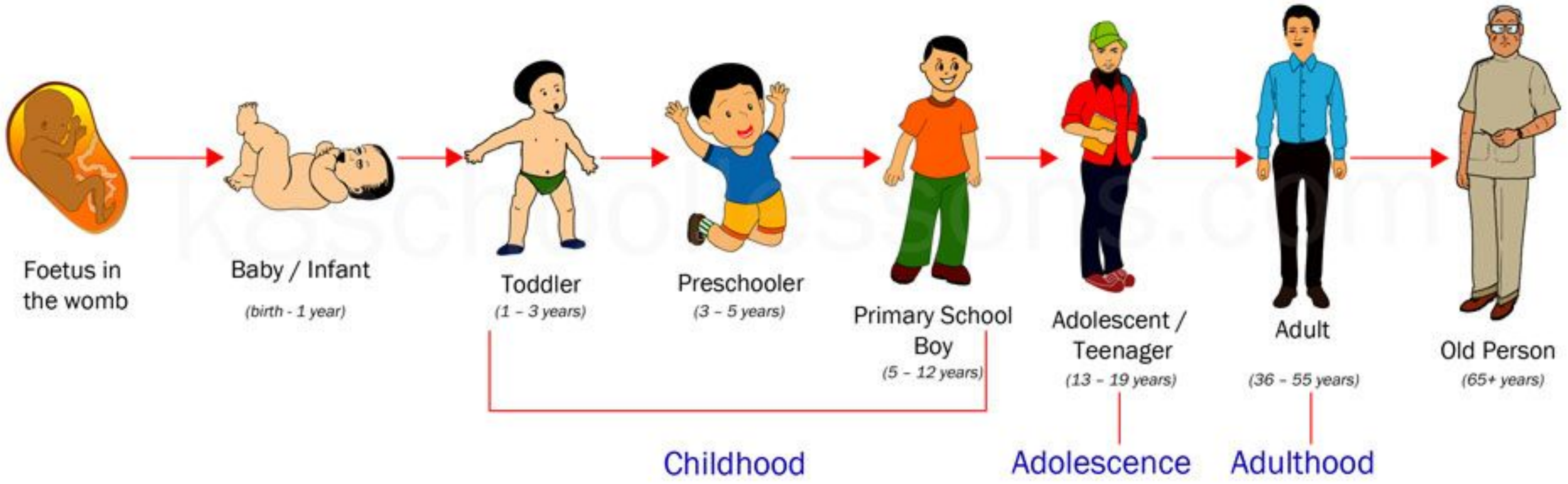
Maven

Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation

During the process, Maven takes care of the following elements:

- Builds
- Dependencies
- Reports
- Distribution
- Releases
- Mailing list
- Goals - Test

Human Life cycle



Maven: Lifecycle Phases

A Maven phase represents a stage in the Maven build lifecycle

- *validate* – checks the correctness of the project
- *compile* – compiles the provided source code into binary artifacts
- *test* – executes unit tests
- *package* – packages compiled code into an archive file
- *integration-test* – executes additional tests, which require the packaging
- *verify* – checks if the package is valid
- *install* – installs the package file into the local Maven repository
- *deploy* – deploys the package file to a remote server or repository

Maven: Goals

Each phase is a sequence of goals. When we run a phase – all goals bound to this phase are executed in order

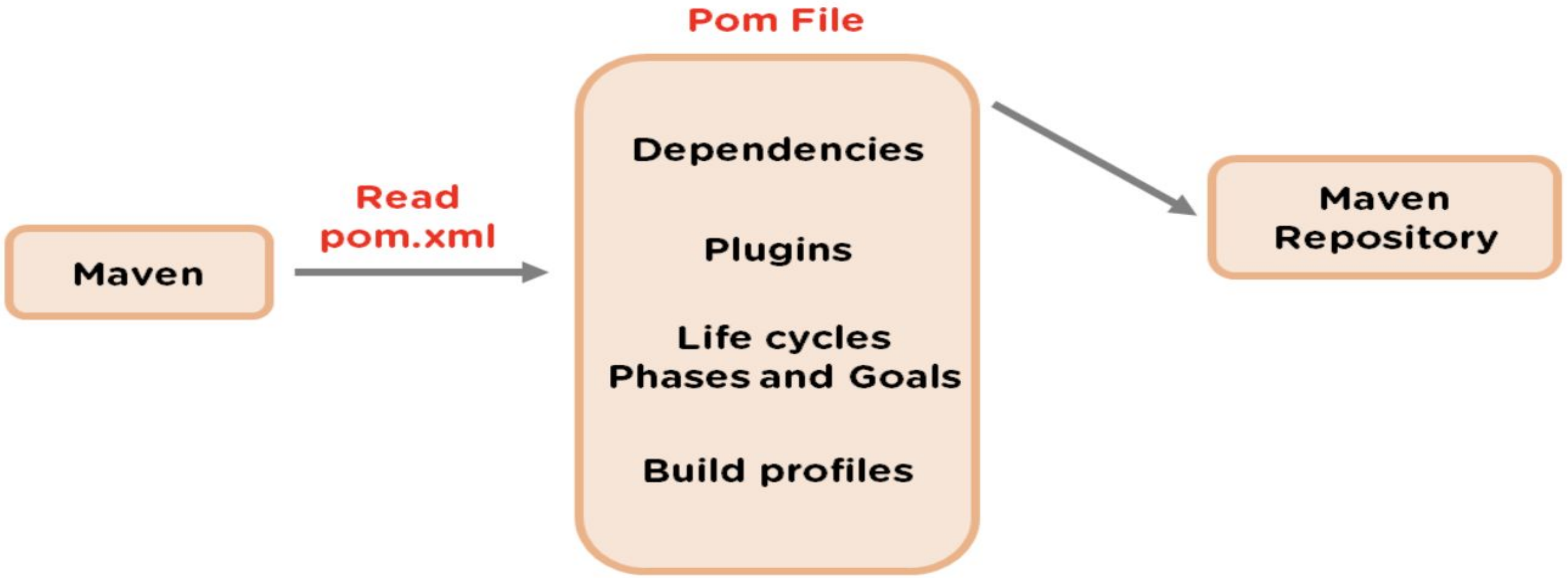
phases and default goals bound to them

- *compiler:compile* – the *compile* goal from the *compiler* plugin is bound to the *compile* phase
- *compiler:testCompile* is bound to the *test-compile* phase
- *surefire:test* is bound to *test* phase
- *install:install* is bound to *install* phase
- *jar:jar* and *war:war* is bound to *package* phase

Maven: Project Identifiers

- *groupId* – a unique base name of the company or group that created the project
- *artifactId* – a unique name of the project
- *version* – a version of the project
- *packaging* – a packaging method (e.g. *WAR*/*JAR*/*ZIP*)

Maven architecture



Demo

- Demo Creation of Maven Project
- Maven POM file
  - Dependencies
- Walkthrough of Lifecycle

TestNG Introduction

- ❑ Definition of TestNG as per its documentation is as follows −

TestNG is a testing framework inspired from JUnit and NUnit, but introducing

some new functionalities that make it more powerful and easier to use.

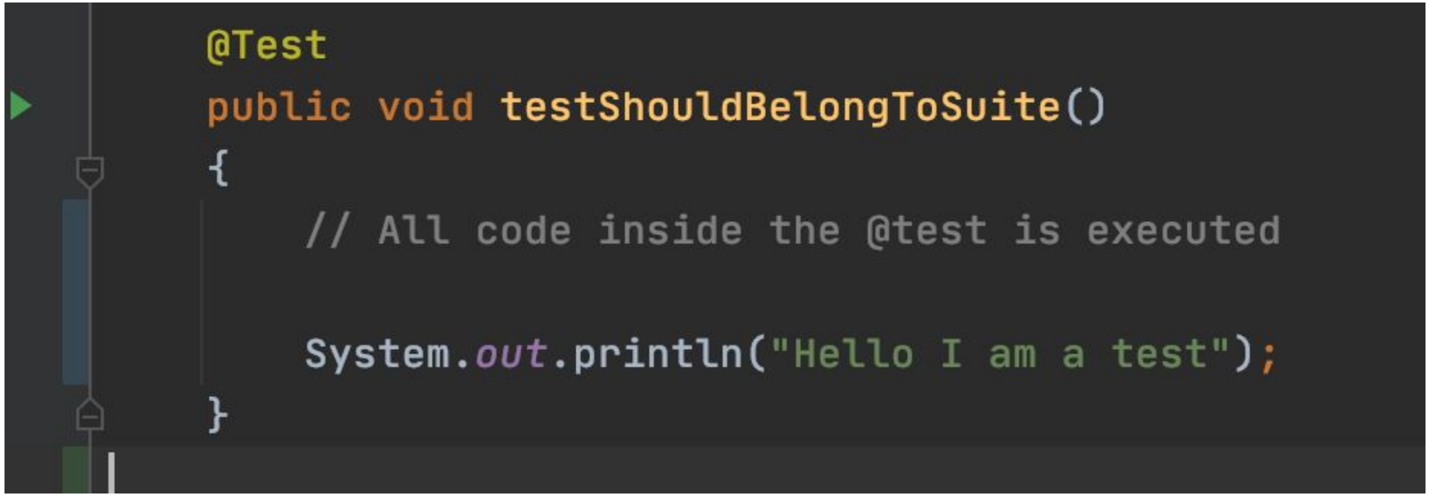
- ❑ What is Annotation? An annotation is a tag or meta-data that provides additional information about the class, interface, or method in TestNG.

Popular Test Frameworks

- JUnit
- TestNG
- Nunit
- PyUnit
- NoseTest
- Test-Unit
- Cucumber
- Gauge
- Serenity

Demo: TestNG @Test Annotation

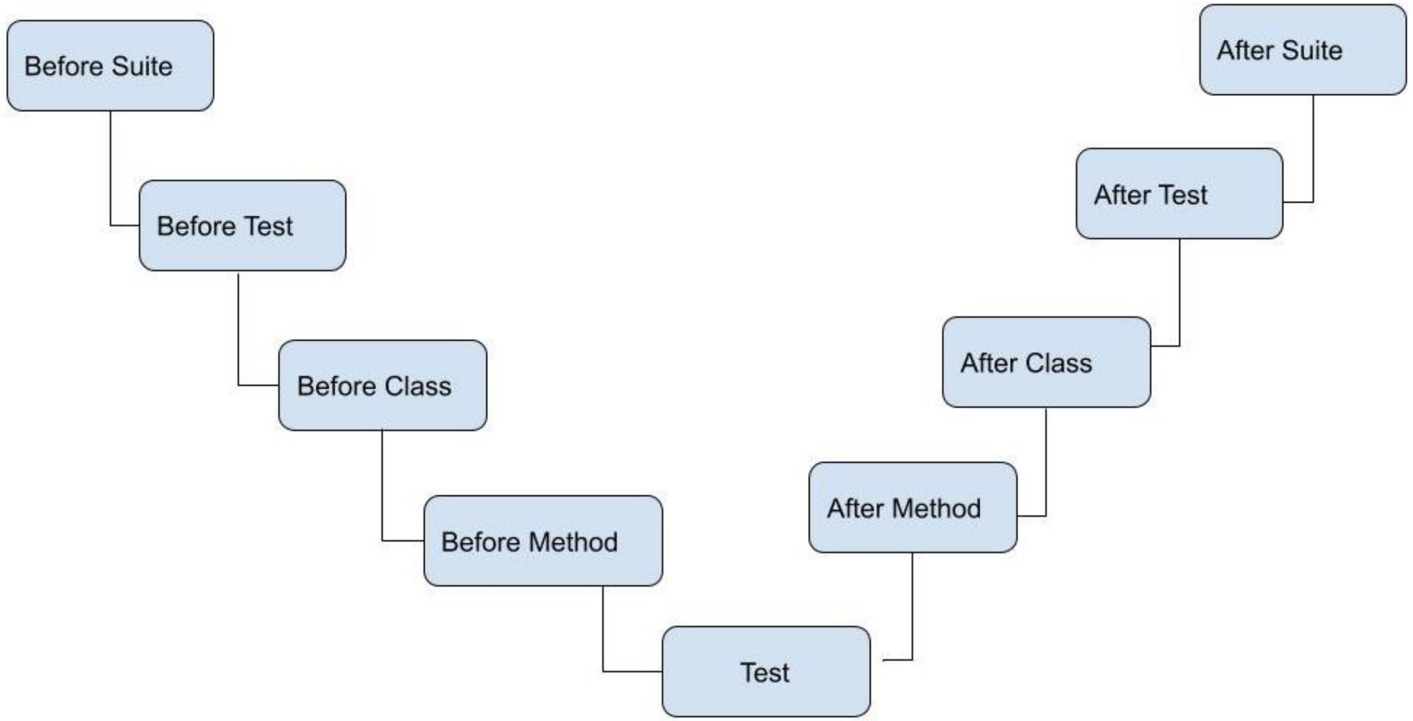
- A test method is a Java method annotated by @Test in your source.



```
@Test
public void testShouldBelongToSuite()
{
    // All code inside the @test is executed

    System.out.println("Hello I am a test");
}
```

TestNG Annotation Hierarchy



Annotation: @BeforeMethod

@BeforeMethod annotated method gets executed just before executing the @Test annotated method.

- ❏ TestNG checks if there is any @BeforeMethod, which is related to @Test method if present, then TestNG executes it.
- ❏ @BeforeMethod will be executed for every @Test method present in the class, @BeforeMethod declared in one class will not affect @Test present in the Other classes

Annotations: @AfterMethod

@AfterMethod will be executed once the @Test block completes it's execution.

❑ @AfterMethod will run for every @Test method in the TestNG test class. If there is no @Test method, then @AfterMethod will be ignored by TestNG.

❑ @AfterMethod will have cleaning code for the method like deletion of details and logging out, or refreshing pages

Test class Based Annotations

@Test, @BeforeMethod, @AfterMethod, @BeforeClass,

@AfterClass, @BeforeGroups,@AfterGroups,@Parameters,@DataProvider

TestNG.xml based Annotations

@beforeSuit,@AfterSuit,@BeforeTest,@AfterSuit

TestNG.XML:

Manage Test Executions

Assertions

- *Assertions are a way to verify that the expected result and the actual result matched or not.*

```
Assert.Method(actual, expected)

Method Can be: assertEquals, assertTrue, assertFalse
```

- ***Actual**: The actual value that the tester gets like if the tester's assertion is on the title of the page then what was the actual title of the page goes here.*
- ***Expected**: The value that you expect like if the tester's assertion is on the title of the page then what value of title do you expect goes here.*

Assertions: Example

Assert.assertEqual(String actual,String expected) :- It takes two string arguments and checks whether both are equal, if not it will fail the test.

Assert.assertTrue(condition, message) :- It takes one boolean argument and String message. It Asserts that a condition is true. If it isn't, an AssertionError, with the given message, is thrown

Assert.assertFalse(condition, message) :- It takes one boolean argument and String message. It Asserts that a condition is false. If it isn't, an AssertionError, with the given message, is thrown

Demo: TestNG

Demo: Webdriver Test Scenario 1

TEST Scenario: 1

1. Launch Spree website
2. click on login
3. create new account for a user
4. Verify the email address on account page

Automation Framework

- Common Libraries
- Test Data - Excel
- Reporting libraries
- Test Automation scripts
- Reporting
- Configurations
- Object repository