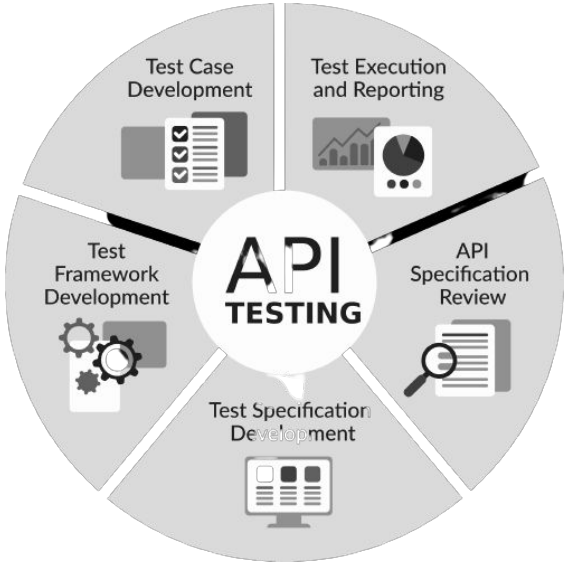
TestNG Mastery



Pramod Dutta
Lead SDET.

TestNG Mastery

TestNG

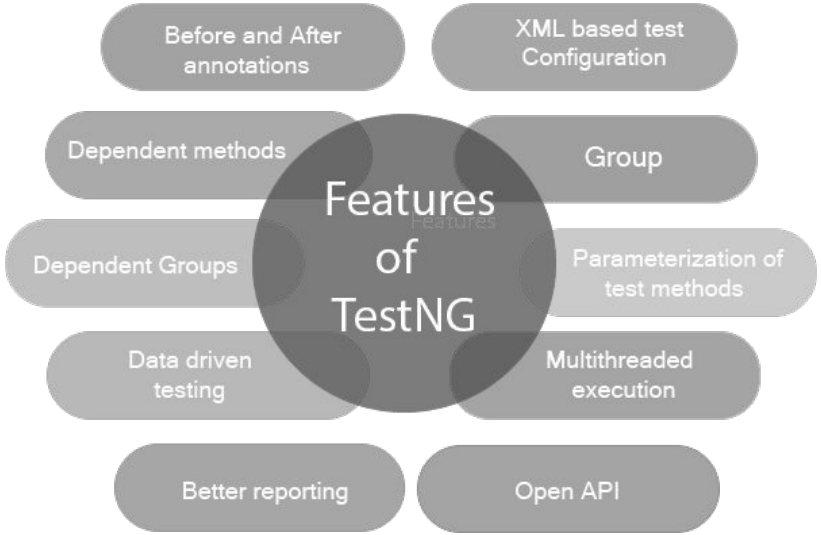
401
Response

What is TestNG?

- Easy to use **testing framework** written in Java.
- Designed for ***unit***, functional, and end-to-end integration tests.
- You can use it with the **API Testing** and **UI Testing**.
- Provides features like parallel testing, multi thread execution and annotations support.
- TestNG supports multiple plugin and can be integrated easily with existing frameworks.

Why we Need TestNG?

1. Reporting
2. Parallel Execution
3. Annotations Support
4. Multi threaded testing
5. Eazy integration
6. Open source
7. Better than JUnit Framework



How to Write Test in TestNG?

1. Create a Method and add your logic in that Method
2. Mark that Method as @Test
3. Add required Annotations like
    a. description,
    b. group,
    c. priority,
    d. enabled,

How to Write Test?

```java
package com.testng;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class Main {
    @Test
    void openGoogle(){
        WebDriver driver = new FirefoxDriver();
        driver.get("https://scrolltest.com");
        Assert.assertEquals(driver.getTitle(), expected: "Scrolltest - Software Testing & Automation");
        driver.quit();
    }
}
```

What is TestNG File?

1. It’s a XML format file.
2. It’s a file that contains the test configuration.
3. It allows us to organize test classes.
4. You can define test suites and tests.
5. You can add parameters and other parallel config support here.

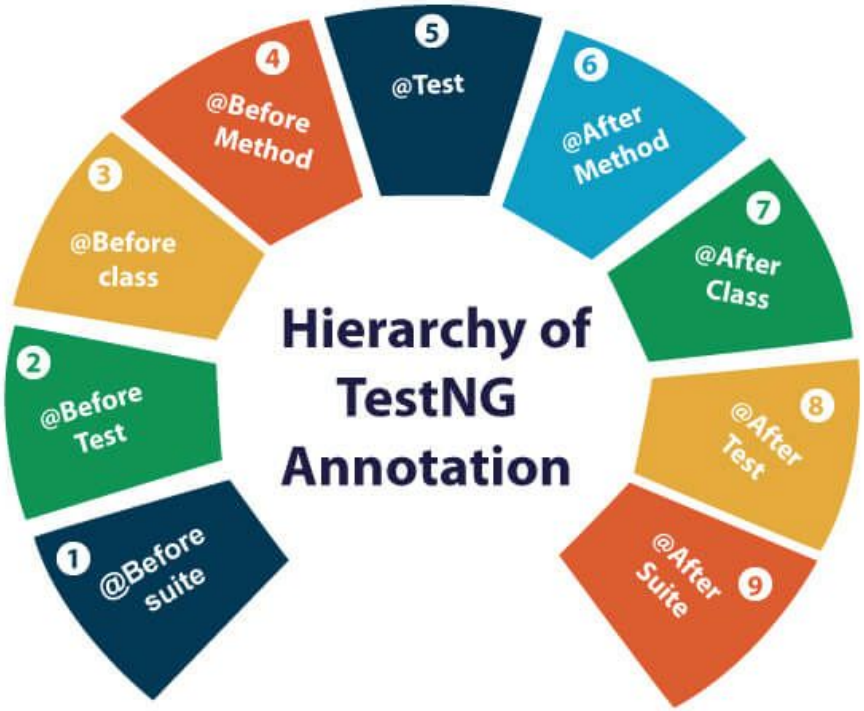
What is TestNG File?

Main.java testng.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="TS">

    <test name="TC">
        <classes>
            <class name="com.testng.Main">
            </class>
        </classes>
    </test>
</suite>
```

TestNG Annotations



TestNG Annotations

@BeforeSuite: The annotated method will be run before all tests in this suite have run.
@AfterSuite: The annotated method will be run after all tests in this suite have run.
@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.
@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.
@BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
@AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

TestNG Annotations

@**BeforeClass**: The annotated method will be run before the first test method in the current class is invoked.
@**AfterClass**: The annotated method will be run after all the test methods in the current class have been run.
@**BeforeMethod**: The annotated method will be run before each test method.
@**AfterMethod**: The annotated method will be run after each test method.

TestNG Annotations

@**Parameter** annotation on test method is to pass parameters to test methods.

@**DataProvider** annotated method is used to create test methods or test classes at runtime with different parameters.

@**Factory** can be used on a method that returns instances of test classes or on a test class constructor in conjunction with @DataProvider

TestNG Annotations

@Listeners is used at test class level to takes the array of classes that implements a plethora of implementations of ITestNGListener interface like IAlterSuiteListener, IAnnotationTransformer, IMethodInterceptor, IReporter, etc. for different purposes.

Grouping of Tests.

What is the grouping of tests?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="test_suite">
<groups>
<run>
<include name="SmokeTest"/>
</run>
</groups>
<test name="Personal Loan">
<classes>
<class name="com.javatpoint.Personal_loan"/>
</classes>
</test> <!-- Test -->
```

```
package com.javatpoint;
import org.testng.annotations.Test;
public class Personal_loan
{
 @Test(groups= {"SmokeTest"})
 public void WebLoginPersonalLoan()
 {
    System.out.println("Web Login Personal Loan");
 }
 @Test
 public void MobileLoginPersonalLoan()
 {
    System.out.println("Mobile Login Personal Loan");
 }
 @Test
 public void APILoginPersonalLoan()
 {
    System.out.println("API Login Personal Loan");
 }
}
```

Running Tests in Parallel

Parallelism and thread count can be set at **suite level** or **test level** like below.

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd"
<suite name="TS" parallel="tests" thread-count="5">
```

Running Tests in Parallel

Methods run test methods in parallel in different threads. All dependent methods will be run in different threads, respecting the priority of tests.

Tests run <test> tags in parallel in separate threads.

Classes run test classes in parallel in separate threads, but test methods in those test classes will run in the same thread.

Instances run instances of test methods/classes in parallel in different threads.

Running Tests in Parallel

@DataProvider, the parallelism can be controlled using the attribute @DataProvider(**parallel = true**)

Listeners

Listeners are TestNG annotations that **literally “listen” to the events in a script** and modify TestNG behaviour accordingly

IAnnotationTransformer
IExecutionListener
IHookable
IInvokedMethodListener
IMethodInterceptor
IReporter
ISuiteListener
ITestListener

ReportNG

Easy to use Reporting

```
<dependency>
        <groupId>org.testng</groupId>
        <artifactId>reportng</artifactId>
        <version>1.2.2</version>
        <scope>test</scope>
    </dependency>
```

Listeners

Demo of IExecutionListener

CustomListener.java

```java
package com.thetestingacademy.testng.Listener;

import org.testng.IExecutionListener;

public class CustomListener implements IExecutionListener {
    @Override
    public void onExecutionFinish() {
        long endTime= System.currentTimeMillis();
        System.out.println("****   *** Finished execution at- "+ endTime +"

    }

    @Override
    public void onExecutionStart() {
        long startTime= System.currentTimeMillis();
        System.out.println(" ****   *** Started execution at - "+ startTime

    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="All Test Suite">
    <listeners>
        <listener class-name="com.thetestingacademy.testng.Listener.CustomLis
    </listeners>
    <test name="LearnTestNG">

        <groups>
            <run>
                <include name = "sanity"></include>
<!--                <exclude name="smoke"></exclude>-->
            </run>
        </groups>
        <classes>
            <class name="com.thetestingacademy.testng.Groups"></class>
        </classes>
    </test>
</suite>
```

Data Provider CSV File

```
<dependency>

<groupId>com.codoid.products</gr
oupId>

<artifactId>fillo</artifactId>
   <version>1.15</version>
</dependency>
```

User, pass x4

Passed as array to function()

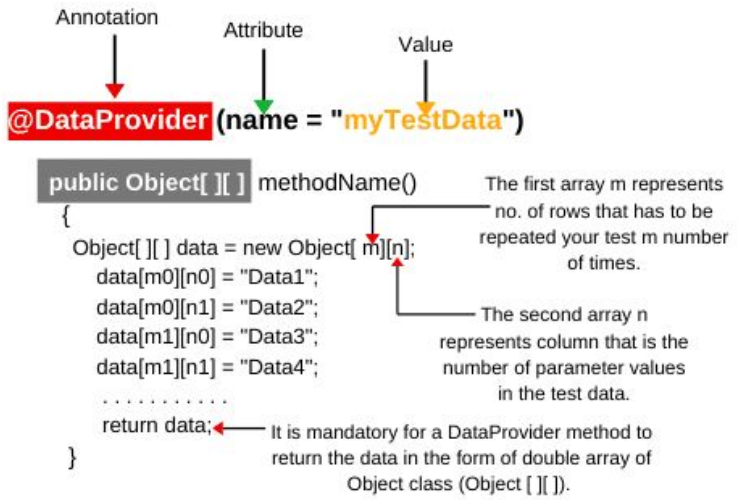


Fig: TestNG DataProvider Annotation | A complete code structure

Data Provider CSV File

```
Run All
public class BasicTest2 {

    @Test(dataProvider = "dp1")
    Run | Debug
    public void TestLogin(String[] s) throws Exception {
        System.out.println(s[0] +" >> "+s[1]);
    }

    @DataProvider()
    public String[][] dp1() {
        String[][] data= new String[][] {
            {"hyr","123"},
            {"pqr","456"},
            {"xyz","789"}
        };
        return data;
    }
```

TestNG – Parallel Test Execution

TestNG parallel execution of tests, classes and suites with examples. Learn how to run testng tests and suites in parallel or single test in multiple threads.

Reduces execution time

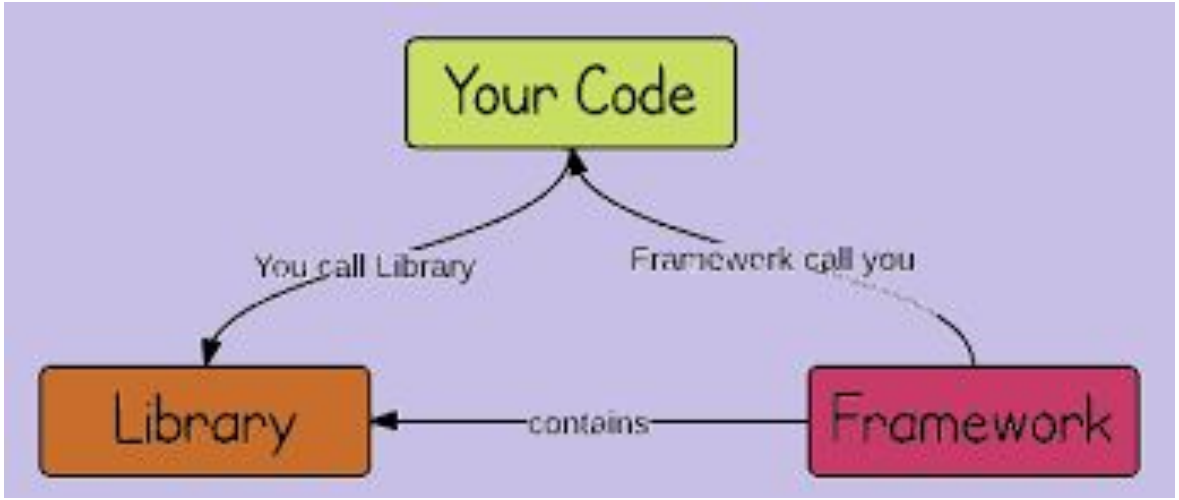
Allows multi-threaded tests

Test Level
Class Level
Method Level
D**ata Provider**
Thread-Count

Why Framework

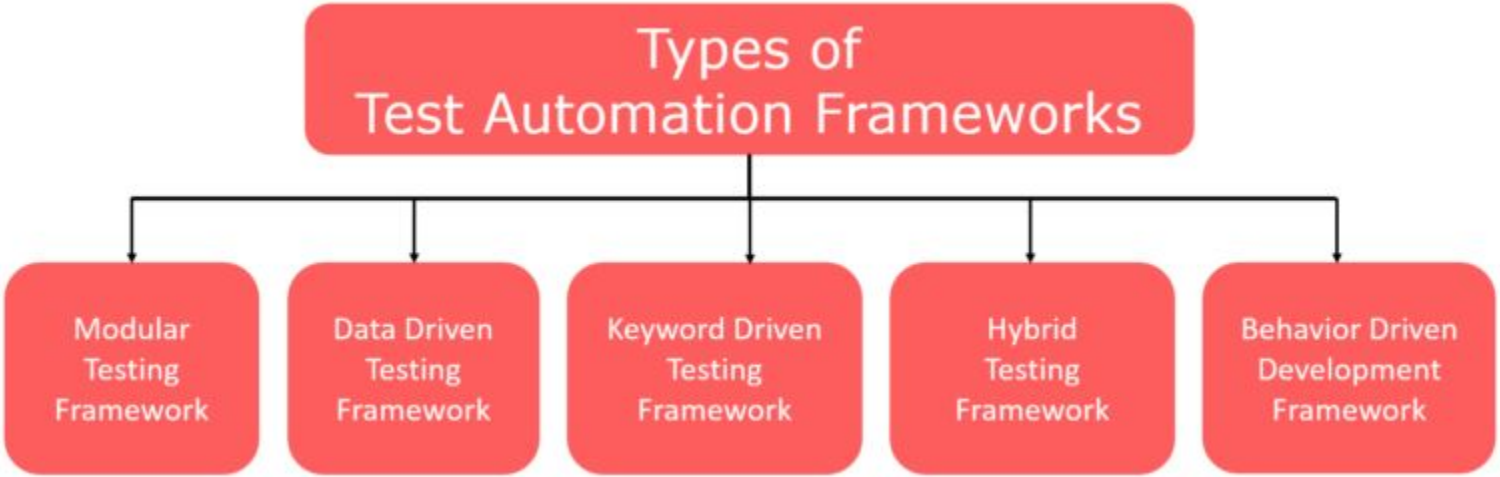
Why Framework

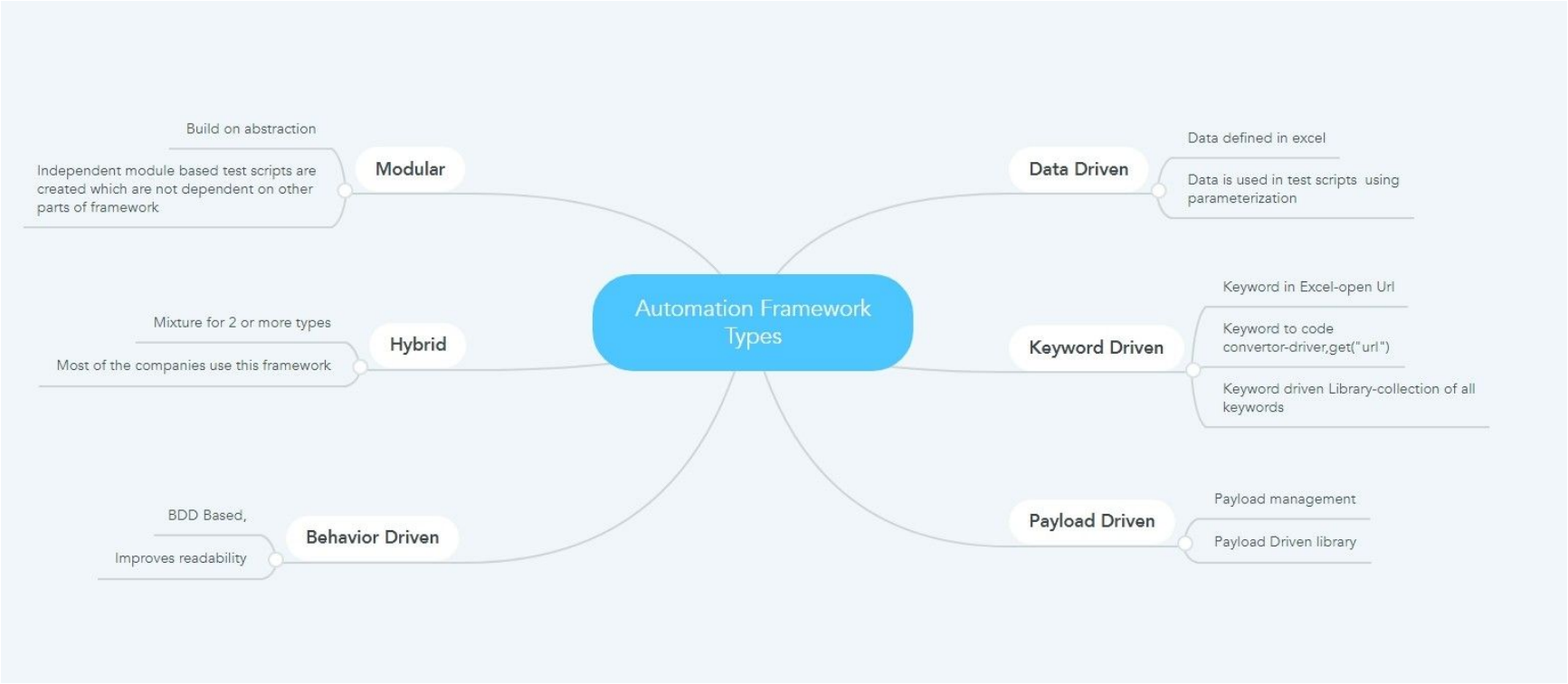
A framework defines the organization's way of doing things - a 'Single Standard'



Framework vs Library

Basis	LIBRARY	FRAMEWORK
Meaning	Library is collection of reusable functions used by computer program	Framework is a piece of code that dictates the arcitecture of project
Inversion of control	With a library , you are in-charge, means you can choose where and when you want to insert or use the library.	In a framework, the framework is in-charge, not you, means a framework tells you where to put a specific part of your code
Function	They are important in program linking and binding process	In a framework , provided standard way to build and deploy applications
Flexiblity	Libraries are more flexible with greater degree of control	Frameworks are enforced structure and standards.
Example	React.js, Jquery is a javascript library	Angular js, Vue js is javascript framework


Types of
Test Automation Frameworks
Modular Testing Framework
Data Driven Testing Framework
Keyword Driven Testing Framework
Hybrid Testing Framework
Behavior Driven Development Framework


Automation Framework Types
Modular
Build on abstraction
Independent module based test scripts are created which are not dependent on other parts of framework
Hybrid
Mixture for 2 or more types
Most of the companies use this framework
Behavior Driven
BDD Based,
Improves readability
Data Driven
Data defined in excel
Data is used in test scripts using parameterization
Keyword Driven
Keyword in Excel-open Url
Keyword to code convertor-driver,get("url")
Keyword driven Library-collection of all keywords
Payload Driven
Payload management
Payload Driven library

Modular

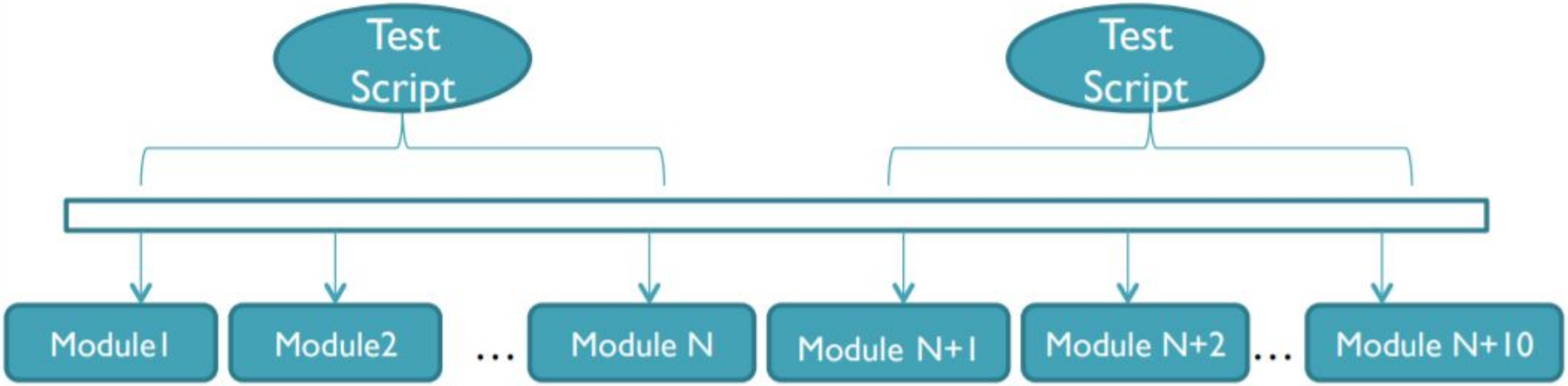
Modular frameworks divides the test scripts into small modules where modules are small scripts written to perform certain tasks.

Modular framework is like creation of small, independent scripts that represents modules, sections and functions of the application under test

individual test scripts can be combined to make larger test scripts by using a master script to achieve the required scenarios.

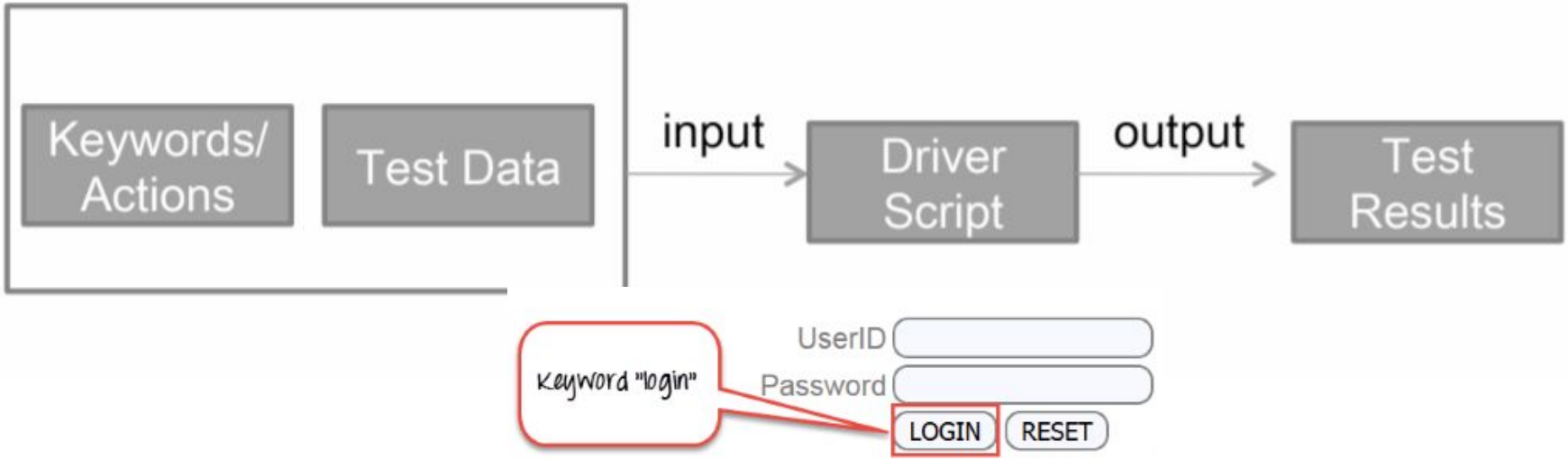
Master script is used to invoke the individual modules to run end to end test scenarios

Modular


Test Script
Test Script
Module1
Module2
...
Module N
Module N+1
Module N+2
...
Module N+10

Keyword Driven

Keyword Driven Framework is a functional automation testing framework that divides test cases into four different parts in order to separate coding from test cases and test steps for better automation

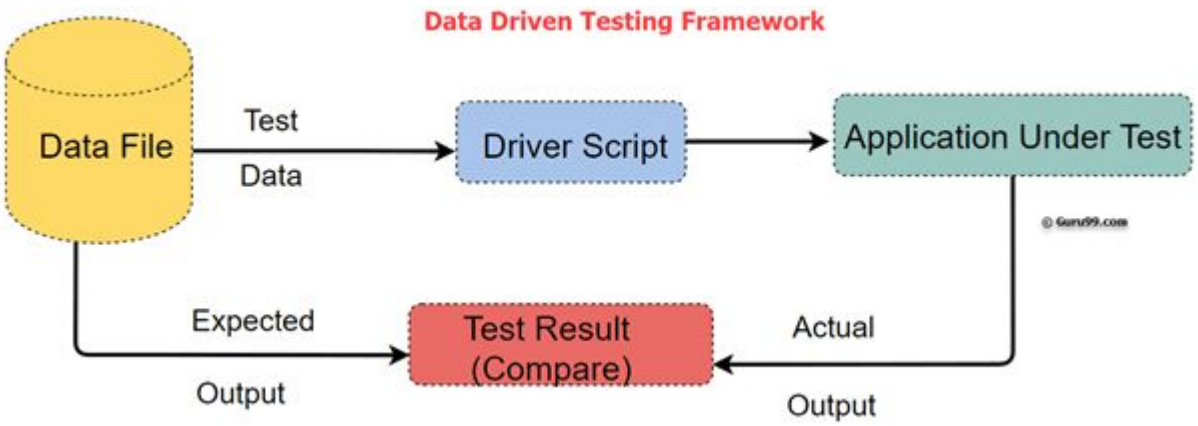


Keyword Driven

ID	Name	Step_ID	Description	Keyword
1	Login to Application	Step 1	Open browser	openBrowser
		Step 2	Navigate to URL	navigate
		Step 3	Enter Email	enterEmail
		Step 4	Enter Password	enterPassword
		Step 5	Click on Sign in button	clickSignIn
		Step 6	Click on Logout button	logout
		Step 7	Close browser	closeBrowser

Data Driven

Data driven testing allows testers to input a single test script that can execute tests for all test data from a table and expect the test output in the same table

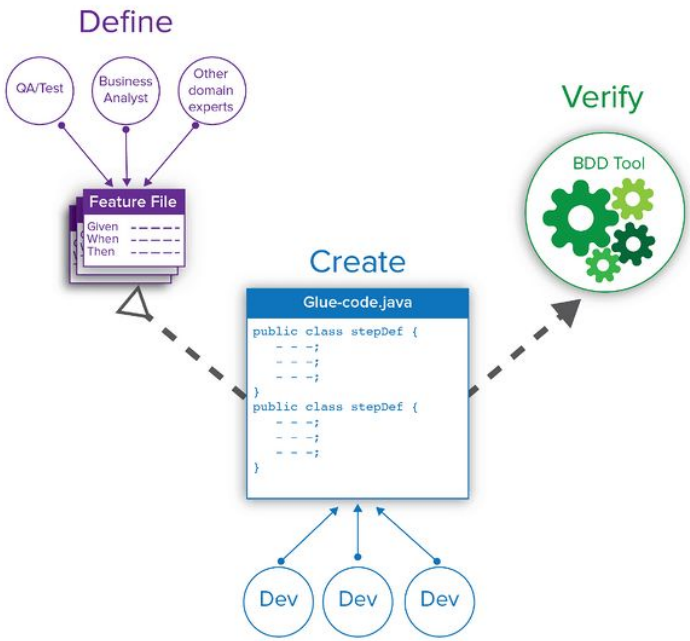


Behavior Driven

Runner File

Feature Files(Gherkin Language)

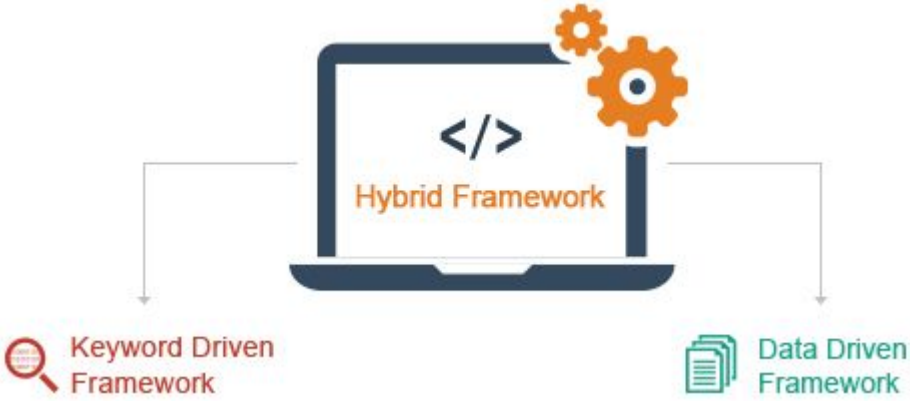
GivenWhenThen Methods/Test Scripts



Hybrid

Combination of multiple types

=>Data Driven+BDD+Keyword



Understand Requirements

- Is Framework Needed?
- Select Programming Language and tool
- Choose Your Framework Design.
- Create your framework
- Implement CI/CD

Framework Components

- Manage Dependencies/Projects(Maven/Gradle/PIP/NPM/Nuget)
- Manage Data(Excel/json files/prop files/xml files)
- Manage Payload and Endpoints(json strings/jsonmaps/pojos/serialize/deserialize/gson)
- Manage Tests (testng and allure for this)(precondition/postcondition/set/config/teardown/steps,description,priority,severity/execution)
- Reuse Components(Keywords, Abstraction, Inheritance, Generics, Configs. Specs. setups/helpers)
- Logger-Report Loggers(Testng. allure), text loggers(log4J)
- Reports-Test summary, percentage, steps, description, failure reason, logs-allure
- Utils -String manipulators, Json Manipulators, Data Manipulators, Readers/Writers/custom code/tools
- CI/CT-Version Controlling-github/git, Continuous Integration and Testing-jenkins/teamcity/travisci

Thanks, for attending Class

I hope you liked it.

Fin.