Framework : It is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing.

Advantage of Test Automation framework : Reusability of code, Maximum coverage, Recovery scenario, Low cost maintenance, Minimal manual intervention, Easy Reporting

**Keyword driven framework:** The keyword driven frameworks requires the development of data tables and keywords, independent of the test automation.  In a keyword driven test, the functionality of the application under test is documented in a table as well as step by step instructions for each test.

**Hybrid Testing Framework:** Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.

**Behavior Driven Development Framework:** Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc.

**Data driven frame work :** Data Driven Testing Framework is a framework which is driven by test data, which means that test data is the important factor here. The basic expectation with this kind of test automation framework is that your test scripts should be built in such a way that it should work for different sets of data without any changes to the test script.

n this Framework , while Test case logic resides in Test Scripts, the Test Data is separated and kept outside the Test Scripts. Test Data is read from the external files (Excel File) and are loaded into the variables inside the Test Script. Variables are used both for Input values and for Verification values. act.sendKeys(Keys.ENTER);

Consider a scenario where you have a large number of user credentials for an application. You need to login to the application with each of these credentials to find out which of the user credentials are working properly and which are not. In order to complete this task, you can create a script in QTP which would read each of the user credentials and try to login to the application and at the end would report the results.

### Example-

Developing the Flight Reservation Login script using this method will involve two steps.

**Step 1)** Create a Test - Data file which could be Excel , CSV , or any other database source.

**Step 2)**Develop Test Script and make references to your Test- Data source.

**SystemUtil.Run "flight4a.exe","","","open"**

**Dialog("Login").WinEdit("Agent Name:").Set DataTable("AgentName", dtGlobalSheet)**

**Dialog("Login").WinEdit("Password:").Set DataTable("Password", dtGlobalSheet)**

**Dialog("Login").WinButton("OK").Click**

**'Check Flight Reservation Window has loaded**

**Window("Flight Reservation").Check CheckPoint("Flight Reservation")**

I used to write the test scripts in a way where the test data was “hard-coded” with the test script. i.e. the data and the script were together as a single unit. you would need to change the test data in the script after each run which is not such a good idea because **the basic premise of automation is “you have to aim for least amount of manual intervention possible”**.

Normal framework : Browser(“”).page(””).Webedit(“”).set ”value”

Data driven framework: var <get data from external source somehow> Browser(“”).page(””).Webedit(“”).set var

**Advantages**: Changes to the Test Scripts do not affect the Test Data

Test Cases can be executed with multiple Sets of Data

A Variety of Test Scenarios can be executed by just varying the Test Data in the External Data File

**DisAdvantages**: More time is needed to plan and prepare both Test Scripts and Test Data

* The **components** in a data driven framework are**: Test script, Data files, Shared Functional library(if exists or could be a linear program), Object repository (Again, this component will not exist if descriptive programming was used to create objects)**

The test results will show a “Passed’ or ‘Failed’ status for each test run.

Apart from the data table that comes by default, we can use any external excel file as an input sheet.

**Locators tell Selenium IDE which GUI elements (like Text Box, Buttons, Check Boxes etc. ) its needs to operate on.**

**Different types of locators:**

**ID :**id=id of the element**,**

**Name : name = name of the element, name with filters :** *name=*name\_of\_the\_elementfilter*=*value\_of\_filter

**Link text : link-link\_text ,**

**CSS Selector(Tag and Id, Tag and class, Tag and attribute Tag,class and attribute),**

**These are used to identify an element based on a combination of HTML tag, id, class, and attributes**. **Locating by CSS Selector is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.**

|  |  |
| --- | --- |
| css=*tag*#*id* | tag = the HTML tag of the element being accessed  # = the hash sign. This should always be present when using a CSS Selector with ID  id = the ID of the element being accessed |
| css=tag.class | . = the dot sign. This should always be present when using a CSS Selector with class |

**DOM( getelementbyId, getelementbyName, dom:name, dom:Index), Xpath**

|  |  |
| --- | --- |
| document.getElementById("*id of the element*") | id of the element = this is the value of the ID attribute of the element to be accessed. This value should always be enclosed in a pair of parentheses (""). |

**Testing Annotations:**

|  |  |
| --- | --- |
| @BeforeSuite | The annotated method will be run only once before all tests in this suite have run. |
| @AfterSuite | The annotated method will be run only once after all tests in this suite have run. |
| @BeforeClass | The annotated method will be run only once before the first test method in the current class is invoked. |
| @AfterClass | The annotated method will be run only once after all the test methods in the current class 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. |

Select class of Selenium WebDriver provides useful methods for interacting with select options. User can perform operations on a select dropdown and also de-select operation using the below methods.

In **C#** the Select class is actually SelectElement.

 The Select class ([java](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/support/ui/Select.html) and [python](http://selenium.googlecode.com/svn/trunk/docs/api/py/webdriver_support/selenium.webdriver.support.select.html) documentation) includes utility methods that allow you to perform common tasks.

<select id="mySelectID">

<option value="Value">Option</option>

<option value="NotValue">Not Option</option>

</select>

**SOFTWARE TESTING:**

Software testing is a process to test an application. It is a process to check that the application is bug free and It can be stated as a process as validating and verifying the product.

1. Testing under different conditions (positive and negative conditions)
2. Over a period: Perform testing in cyclic way until we deliver an application to the client.
3. Evaluating the result: Comparing the actual result with Expected result.

Actual Result: Result coming from application

Expected result: Result we get from client requirement.

Difference between Product and Project:

Product: Product is a software which is developed first and then the application will be modified according to the market requirements and sell it.

Project: Project is a software developed on the basis of the client requirement: Take requirement from client and create a software and deliver it to the client.

Difference between Manual and Automation Testing:

Manual: In this testing is done by user manually without using any tool. Here all cases are running by the tester and manually prepare execution report.

Tester verify each and every condition manually and update in results.

Automation: In this testing is done by using any automation tool like QTP, Selenium, LoadRunner etc.

In automation testing we are going to create automation scripts, and the automation scripts are run by using automation tools.

Types of applications:

We have 3 types of applications. They are: Window based applications

Web based applications

Client server based applications

Window based/Desktop based: These application run only in personal computer or work stations, so when we test

**APPIUM**:

Latest version : 1.4

It is an open source tool for automating native, mobile web and hybrid applications.

It is an mobile automation tool for mobile.

It supports both android and IOS platforms.

It’s a cross platform. It allows us to write tests against different platforms (android& IOS) using the same API. Only small change in the startup code and can use it for both IOS and android (Code reuse).

It supports multi languages. Selenium supports 7 languages (Java, Javascript, C#, Ruby, Python, PHP).

Appium also supports multi languages (Java, Robot framework, C#, PHP, Ruby, Perl, Node, Python)

Types of applications:

It supports: Native Applications

Web applications/View

Hybrid applications

Native applications: These are the apps which are runs on a device and which doesn’t run on browser. Ex: Whatsapp

Web application/View: These apps are which runs only on the browser. Ex: Any blog

Hybrid application: These are the apps which runs on both browser and device. Ex: Facebook, Gmail etc.

We need 13 tools for Appium installation. They are :

Java :

Eclipse : It is an IDE for writing the code (Selenium)

Selenium Jar file:

Java client:

Appium server:

Node JS:

Android SDK bundle: We need it so that we can interact with the android libraries. We need it while we work with android emulators.

Microsoft .net: It is a backend support

GenyMotion (optional): While we work with emulators and simulators, it will allow us to create multiple simulator with different configurations.

APK file:

Android Device:

PDA Device: it helps us to work with drivers.

Steps for installation:

* Download and set path for android SDK.
* Install NodeJS
* Download and install Appium
* Download Java-client jar and add into eclipse.
* PADNET in mobile and in desktop.
* Download GenyMotion and create AVD.

SDLC:

Unit testing: It is a testing of individual program or module.

Integrated testing: It is a testing for accurate flow of information for two or more systems.

System testing: Full pledge test includes stress, Load, Volume, Recovery, Performance and Security test.

Acceptance test: It includes QAT and UAT.

**SYSTEM TESTING**

**ACCEPTANCE TESTING**

**UNIT TESTING**

**INTEGRATION TESTING**

Unit Testing:

1. Unit testing is used for testing individual units/components of a software.
2. It is a component of test-driven development(TDD)
3. It is done during the development stage.
4. White box approach (Testing of internal program language) is approved in unit testing whereas in black box only functional testing is done.
5. The purpose of doing unit testing is to validate that each unit of the software performed as designed.
6. It is normally performed by the developer themselves or its peers. In rare case it may also be performed by independent software testers.

Tasks:

1. Unit Test plan:

* Prepare
* Review
* Rework
* Baseline

1. Unit Test Scripts/Cases:

* Prepare
* Review
* Rework
* Baseline

1. Unit Test

* Perform

**Perfecto Mobile:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operating System** | **Developed by** | **Popularity** | **Latest Version** |
| Android | Google Inc. | High | Android 7 Nougat,  Marshmallow 6.x,  Lollipop 5.x,  Kitkat 4.4x, 4.3, 4.2, 4.1(Jelly Bean) |
| iOS | Apple | High | iOS 10.3 Beta 2 |
| Blackberry | Blackberry Ltd. | Low | 10.3.3.x |
| Windows | Microsoft Inc. | Medium | Windows 10 |
| Symbian | Symbian Foundation | Low | ---- |

We have 3 types of mobile applications:

1. Mobile web
2. Native app.
3. Hybrid app.

The classification is based upon the development efforts and App redistribution strategy.

**Mobile Web:**

These apps are not real applications, these are the apps which we can open through the web browser in our mobile.

Benefits:

1. It is easy accessible.
2. Easy development
3. Easy Update: Just update in one location and all the users automatically have access to the latest version of the site.
4. No installation is required as compared to Native and Hybrid apps.

**Native App:**

**Manual Testing:**

Manual testing is a process where the whole process is done manually in order to find the defects without using any tool or automation scripts.

A test document will be prepared to maintain the coverage of complete test.

Below are the testing techniques that can be performed manually during the test life cycle.

1. Unit testing
2. System testing
3. Integration testing
4. Acceptance testing
5. White box testing
6. Black box testing

Unit testing:

It is a technique used to test individual modules to check if there are any issues by the developer itself. It is concerned with the functional correctness of the standalone modules.

The main aim of this unit testing is to identify, analyze and fix the defects.

Advantages:

1. It reduces bugs when the change in existing functionality.
2. It reduces cost as if the defects are recognized at the earliest stage.
3. Improves design and allows better refactoring of code.
4. Unit tests, when integrated with build gives the quality of the build as well.

Unit Testing Life Cycle:

1. Check in the code to repository
2. Check out the code from repository
3. Make changes
4. Execute unit tests
5. Fix defects and re-execute unit tests
6. Code review

Unit testing techniques:

For performing unit testing we have few techniques. They are: Black box testing, White box testing and Grey box testing.

White box testing/ glass/clear/open/structural box testing:

It is a technique which evaluates the code and internal structure of the program.

It can be performed by both testers and developers. It helps to understand which line of code is actually executed and which is not.

Steps to perform white box testing:

1. Understand the functionality of the application through its source code. Since it involves in inner working of the application, the tester must be very aware of the programming languages that are used in the application for testing.

Security is one of the main objective of performing testing of a software.

Tester must be able to find security issues and prevent attacks from hackers who may effect the application either knowingly or unknowingly.

1. Create the tests and execute them.

What we will test in the white box testing:

1. Internal security holes
2. Broken or poorly structured paths in the coding process
3. Expected output
4. Functionality of conditional loops
5. Testing each statement, object and function on an individual basis.

White box testing can be done at integration, functional and unit levels of software development.

Our basic goal of performing white box testing is to verify the work flow for an application.

White box testing techniques:

We have code analysis technique for perming white box testing. Code coverage analysis, it eliminates the gaps in a test case suite. We have few coverage techniques.

Statement Coverage: It requires every possible statement in the code to be tested at least once during the test process.

Branch coverage:

Black box testing: Used to test the user interface, input, and output.

Grey box testing: Used to execute tests, risks and assessment methods.