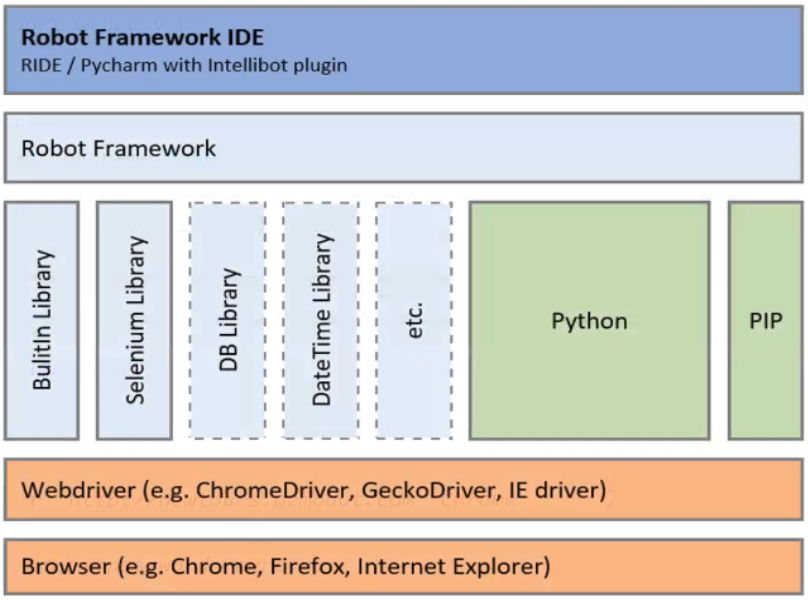
Robot Framework

1. Introduction and Installation of Robot Framework and SeleniumLibrary

- Robot Framework is a python based automation framework used for acceptance testing .
- It was initially developed by Pekka Klark at Nokia Networks
- It has Acceptance test-driven development and utilizes keyword-driven testing approach
- open-source,operating system independent.
- extends support for web, device, api and database automation.
- Require no (or very less) programming skill.
- Excellent reporting, support CI, parallel execution, remote execution.

Architecture of Robot Framework



Normally Robot framework contains multiple libraries (Builtin, Selenium, DB, DateTime, etc).These libraries are implemented either in java or python.In IDE we will write robot automation script using some keywords.when we run this script it will just go and talk to the library. Libraries can either use application interfaces directly or use lower level test tools as drivers.

Setup

- Install Python

```
python --version
pip --version
```

- Install Pycharm IDE
- Install Selenium

```
pip install selenium
pip uninstall selenium
```

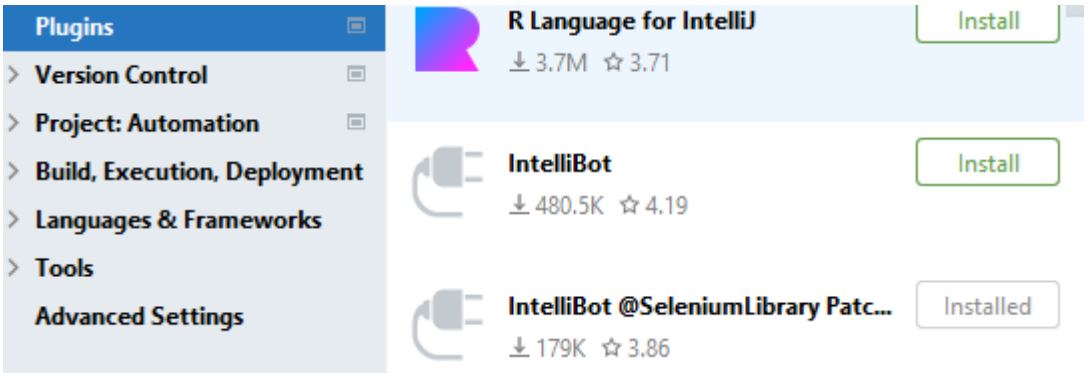
- Install robot framework

```
pip install robotframework
pip install --upgrade robotframework
pip install robotframework==2.9.2

pip uninstall robotframework

pip list
pip show robotframework
pip check robotframework
```

- Install robot framework Selenium Library

```
pip install robotframework-seleniumlibrary
pip uninstall robotframework-seleniumlibrary
```

- Check pip list in cmd prompt
- Install intellibot board with selenium library plugin in Plugins section to work with Pycharm IDE



Robot Framework is case-insensitive. The keywords Close Browser and close browser are the same command.

Note

- A test suite, in general, is a collection of test cases.Test case contains test name and test steps
- A test suite can be saved as .robot file, .txt file or .tsv file. You can follow any one and use it across in your project.

Disadvantages of Robot Framework

- Hard to maintain.
- Difficult to customize HTML reports
- Difficult to debug Errors.
- Strict Indentation rules.

2.First Test Case in Robot Framework

Folder Structure of Robot Framework Project

- **PageObject -** Contains Element Locators belonging to a specific page along with action methods.
- **Resources -** Contains the reusable Robot code files i.e,user defined keywords.
- **TestSuite -** Contains a group of Test Cases which belong to the same category.
- **Results -** Contains executed Test Results.
- **Test data -** Contains Excel,Notepad,python files for Data Driven Testing.

Multiple sections in Robot file

*** Settings ***	• It is used for importing resource files, libraries, and variable files. • Also used for defining metadata for test cases and test suites
*** Variables ***	Used for defining variables that can be used elsewhere in test data.
*** Test Cases ***	It is used to create test cases from available keywords
*** Keywords ***	Creating user keywords from available lower-level keywords

In Settings, we have **documentation, setup, teardown, tags, timeout and template**.

Timeout - This is used to set a timeout on the test case. We will keep it empty for now.

Template - This will have the keywords to be used for the test case. It is mostly used for data driven test case. The high-level user-defined keyword is specified in the template and test cases are used to pass data to the keyword.

Test Cases

```
*** Test Cases ***
TC5
  [Documentation]    Test to convert celsius to fahrenheit
  ${celcius}=    Set Variable    37
  ${fahrenheit}=    Evaluate    (${celcius}*9/5)+32
  Log    ${fahrenheit}
TC6
  [Documentation]    Test to convert List of celsius values to fahrenheit
  @{celcius}=    Create List    10    12    37    22
  FOR    ${temp}    IN    @{celcius}
     ${fahrenheit}=    Evaluate    (${temp}*9/5)+32
     Log    ${fahrenheit}
  END
```

User Keywords - nothing but Functions just like in the programming languages to make code modular.

```
*** Test Cases ***
TC1
  Function1
  Run Keyword    Function2
```

```
*** Keywords ***
Function1
  Log    This is a inner function1
Function2
  Log    This is a inner function2
```

```
*** Keywords ***
CommonFunction1
  Log    test message 1
  Log    test message 2
  Log    test message 3
CommonFunction2
  [Arguments]    ${msg1}    ${msg2}    ${msg3}
  Log    ${msg1}
  Log    ${msg2}
  Log    ${msg3}
```

```
*** Test Cases ***
TC1
  Log    test message 1
  Log    test message 2
  Log    test message 3
TC2
  Log    test message 1
  Log    test message 2
  Log    test message 3
TC3
  [Documentation]    Test having User keyword without arguments
  CommonFunction1
TC4
  [Documentation]    Test having User keyword with arguments
  CommonFunction2    test message 1    test message 2    test message 3
```

SUITE LEVEL SETTINGS

```
*** Settings ***
Documentation    this is suite level documentation
Suite Setup    Log    Suite begins
Suite Teardown    Log    Suite Ends
Test Setup    Log    test setup 1
Test Teardown    Log    test teardown 1
```

TEST LEVEL SETTINGS

```
[Documentation]    this is test level documentation
[Setup]    Log    test setup 2
[Teardown]    Log    test teardown 2
```

Note: If we have Test Setup/Teardown at suite level and SETUP/TEARDOWN at test level, SETUP/TEARDOWN methods override Test Setup/Teardown methods.

```
*** Test Cases ***
LoginTest
    create webdriver  chrome    executable_path="C:\Drivers\chromedriver_win32\chromedriver.exe"
    open browser    https://demo.nopcommerce.com/    chrome
```

TC1.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
*** Test Cases ***
LoginTest
  open browser    https://demo.nopcommerce.com/  chrome
  click link  xpath://a[@class='ico-login']
```

```
  input text  id:Email    pavanoltraining@gmail.com
  input text  id:Password   Test@123
  click element  xpath://input[@class='button-1 login-button']
  close browser
*** Keywords ***
```

Command to execute : robot -d Results TestSuite/TC1.robot

FirstTestSuite.robot

```
*** Settings ***
Documentation  Login Functionality
Library  SeleniumLibrary
*** Variables ***
${url}  https://opensource-demo.orangehrmlive.com/
${browser}  chrome
*** Test Cases ***
Verify Successful Login to OrangeHRM
  [documentation]  This test case verifies that user is able to successfully Login to OrangeHRM
  [tags]  Smoke
  Open Browser  ${url}  ${browser}
  loginToApplication
  Close Browser
*** Keywords ***
loginToApplication
  Wait Until Element Is Visible  id:txtUsername  timeout=5
  Input Text  id:txtUsername  Admin
  Input Password  id:txtPassword  admin123
  Click Element  id:btnLogin
  Element Should Be Visible  id:welcome  timeout=5
```

Command to execute : robot -d Results Tests/FirstTestCase.robot

Under TC section we refer User defined Keyword Name. Whatever steps are defined under this particular keyword those steps will be executed.This is helpful if we want to write those steps again and again.

Move User Defined Keywords and Common to Resource File

Basically the **Resource section** contains user defined keywords and commonly used functionality.

Manual test cases will be in excel, we store test cases in Test Management Tools like quality center,HP ALM, Jira Zephyr, TestLink, TestRail, etc and execute Test Cases.

Resources Folder

CommonFunctionality.robot	UserDefinedKeywords.robot
*** Settings ***	*** Settings ***
Library SeleniumLibrary	Library SeleniumLibrary
*** Variables ***	*** Keywords ***
${url}	loginToApplication

https://opensource-demo.orangehrmlive.com/
${browser} *chrome*
*** Keywords ***
Start TestCase
  Open Browser ${url} ${browser}
  Maximize Browser Window
Finish TestCase
  Close Browser
	Wait Until Element Is Visible *id:txtUsername timeout=5*
	  Input Text *id:txtUsername Admin*
	  Input Password *id:txtPassword admin123*
	  Click Element *id:btnLogin*
	  Element Should Be Visible *id:welcome timeout=5*

Test Suite Folder

FirstTestSuite.robot

```
*** Settings ***
Documentation  Login Functionality
Library  SeleniumLibrary
Resource  ../Resources/CommonFunctionality.robot
Resource  ../Resources/UserDefinedKeywords.robot
*** Variables ***
*** Test Cases ***
Verify Successful Login to OrangeHRM
 [documentation]  This test case verifies that user is able to successfully Login to OrangeHRM
 [tags]  Smoke
 Start TestCase
 loginToApplication
 Finish TestCase
```

Documentation – These Texts are are added to the HTML reports to make them more descriptive.
[documentation] – Provides more info regarding the test cases. Also added to the HTML reports.
[tags] Smoke – Adds tags to the tests

Pass Variables from Command Line

```
4  *** Variables ***
5  ${url}  http://www.ebay.com
6  ${browser}  firefox
7
```

```
Manishs-Air:RobotFWTutorial manish$ robot -d results -v url:http://qa.ebay.com -v browser:chrome tests/eBay
/Verify_search_functionality.robot
```

Local and Global Variables

1. Set Variable

```
TC1
  ${a}=    Set Variable    50
```

2. Set Variable If

```
TC2
  ${a}=    Set Variable If    True==True    50    100
  Log    ${a}
TC3
  ${a}=    Set Variable If    True==False    50    100
  Log    ${a}
```

3. Set Global Variable

```
TC4 - Set Global Variable
  ${a}=    Set Variable    50
  Set Global Variable    ${a}
```

Data type/Number system Conversion

4. Convert To Binary

```
TC5 - Convert To Binary
  ${a}=    Set Variable    99
  ${b}=    Convert To Binary    ${a}
  Log    ${b}
```

5. Convert To Boolean

```
TC6 - Convert To Boolean
  ${a}=    Set Variable    tRuE
  ${b}=    Convert To Boolean    ${a}
  Log    ${b}
```

6. Convert To Bytes

```
TC7 - Convert To Bytes
  ${b}=    Convert To Bytes    75 65 77 65 76    int
  Log    ${b}
```

7. Convert To Hex

```
TC8 - Convert To Hex
  ${b}=    Convert To Hex    100    10
  ${b}=    Convert To Hex    100    8
```

8. Convert To Integer

```
TC9 - Convert To Integer
  ${b}=    Convert To Integer    FF    16
  ${b}=    Convert To Integer    100    8
```

9. Convert To Number

```
TC10 - Convert To Number
  ${b}=    Convert To Number    41.111    1
```

10. Convert To Octal

```
TC11 - Convert To Octal
  ${b}=    Convert To Octal    111    2
```

11. Set Log Level

```
TC13 - Info Logs
  Set Log Level    INFO
  Log    this is Info Log    INFO
  Log    <h1>this is HTML Log</h1>    HTML
  Log    this is WARN Log    WARN
  Log    this is ERROR Log    ERROR
  Log    this is DEBUG Log    DEBUG
TC14 - Debug and Trace Logs
  Set Log Level    DEBUG
  ${a}=    Set Variable    100
```

By default only Info logs are printed on the console. To print debug Logs, we need to set the Log level to DEBUG. For trace level logs, we need to set it to TRACE.

13.Suite/Test information functions

```
Set Suite Documentation
Set Suite Metadata
Set Suite Variable
Set Tags
Set Test Documentation
Set Test Message
```

14. Logs & Comments functions

```
Comment
Log
Log Many
Log To Console
Log Variables
```

15. Assertion functions

Length Should Be
Should Be Empty
Should Be Equal
Should Be Equal As Integers
Should Be Equal As Numbers
Should Be Equal As Strings
Should Be True
Should Contain
Should Contain Any
Should Contain X Times
Should End With
Should Match
Should Match Regexp
Should Not Be Empty
Should Not Be Equal
Should Not Be Equal As Integers
Should Not Be Equal As Numbers
Should Not Be Equal As Strings
Should Not Be True
Should Not Contain
Should Not Contain Any
Should Not End With
Should Not Match
Should Not Match Regexp
Should Not Start With
Should Start With
Variable Should Exist
Variable Should Not Exist

17. Import functions

Import Library
Import Resource
Import Variables

18. Execution control keywords

Call Method
Keyword Should Exist
Return From Keyword
Return From Keyword If
Run Keyword
Run Keyword And Continue On Failure
Run Keyword And Expect Error
Run Keyword And Ignore Error
Run Keyword And Return
Run Keyword And Return If
Run Keyword And Return Status
Run Keyword If
Run Keyword If All Tests Passed
Run Keyword If Any Tests Failed
Run Keyword If Test Failed
Run Keyword If Test Passed
Run Keyword If Timeout Occurred
Run Keyword Unless
Run Keywords
Repeat Keyword
Wait Until Keyword Succeeds
Sleep

19. Other Important functions

Get Count
Get Length
Get Time
Get Variable Value
Get Variables
Catenate
Evaluate
Fail
No Operation
Pass Execution
Pass Execution If
Fatal Error
Regexp Escape
Remove Tags
Set Variable

MathematicalOperations.robot

```
*** Test Cases ***
ArithmeticOperations
  addition    1    2
  Subtraction    2    5
  Multiplication    2    4
  Division    4    8
  Exponent    2    3
  Modulus    9    2
  FloorDivision    3    2
*** Keywords ***
Addition
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate    int(${num1})+int(${num2})
  log to console    ${result}
Subtraction
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate    int(${num1})-int(${num2})
  log to console    ${result}
Multiplication
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate    int(${num1})*int(${num2})
  log to console    ${result}
Division
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate    int(${num1})/int(${num2})
  log to console    ${result}
Exponent
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate    int(${num1})**int(${num2})
  log to console    ${result}
Modulus
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate
  int(${num1})%int(${num2})
  log to console    ${result}
FloorDivision
  [Arguments]    ${num1}    ${num2}
  ${result}    evaluate    int(${num1})//int(${num2})
  log to console    ${result}
```

4.Scalars, Lists, Dictionaries and Variable files

SCALAR - ${VARIABLE NAME} Value

- It holds one value at a time.

LIST - @{variable name } value1 value2

- It holds multiple values.Every value can be accessed by using an index.

DICTIONARY - &{variable name} K1=value1 K2=value2

- It holds multiple values in the form of key-value pairs.Every value can be accessed by using a key.

ENVIRONMENT - %{variable name}

- These are system variables we can see by typing **echo %os%** , **echo %path%** in cmd

FirstTestSuite.robot

```
*** Settings ***
Documentation    Suite description
Library  SeleniumLibrary
*** Variables ***
${URL}  https://opensource-demo.orangehrmlive.com/
${BROWSER}  Chrome
@{Credentials}  Admin  admin123
&{LoginData}  username=Admin  password=admin123
*** Test Cases ***
TC_001_loginTest
  OPEN BROWSER  ${URL}  ${BROWSER}
  maximize browser window
  set selenium implicit wait  5 seconds
  input text  name=username  ${Credentials}[0]
  input text  name:password
${LoginData}[password]
  CLICK ELEMENT  xpath://*[@type="submit"]
  close browser
  Log  This Test is run by %{username} on %{os}
*** Keywords ***
```

BuiltIn Variables - These Variables are provided by Robot Framework itself

Variable	Explanation
${CURDIR}	An absolute path to the directory where the test data file is located. This variable is case-sensitive.
${TEMPDIR}	An absolute path to the system temporary directory. In UNIX-like systems this is typically */tmp*, and in Windows *c:\Documents and Settings\<user>\Local Settings\Temp*.
${EXECDIR}	An absolute path to the directory where test execution was started from.
${/}	The system directory path separator. / in UNIX-like systems and \ in Windows.
${:}	The system path element separator. : in UNIX-like systems and ; in Windows.
${\n}	The system line separator. \n in UNIX-like systems and \r\n in Windows.

Scalar

```
*** Variables ***
${scalar1}      10
${scalar2}      Hello World
*** Test Cases ***
TC1 - Use Scalars
  ${sum}=  Evaluate  ${scalar1}+100
```

List

```
*** Variables ***
@{list1}      Sun  Mon  Tue  Wed  Thu  Fri
Sat
*** Test Cases ***
TC1
  Log  ${list1[3]}
```

Dictionary

```
*** Variables ***
&{dict1}        jan=1    feb=2    mar=3    apr=4    may=5    jun=6
*** Test Cases ***
TC1
   Log    ${dict1['mar']}
```

To create scalars, lists and dictionaries, we can also use keywords: Set Variable Set Variable If Set Global Variable Set Test Variable Create List Create Dictionary	*** Test Cases *** TC1 @{list2}= Create List 1 2 3 4 Log ${list2[2]} TC2 &{dict2}= Create Dictionary a=1 b=2 c=3 d=4 Log ${dict2['c']}

Refactor Selenium Webelement Locators

if there is change in the locator due to development if that locator is used in multiple places we have to change which is time consuming so locators are defined in variable section or external file for reusability and maintainability

To externalize keywords to variable section

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
${SearchTextBox}    xpath://*[@id='gh-ac']
${SearchButton}     xpath://*[@id='gh-btn']
${AdvancedSearchLink}  xpath://*[@id='gh-as-a']
*** Keywords ***
Input Search Text and Click Search
  [Arguments]  ${search_text}
  Input Text  ${SearchTextBox}  ${search_text}
  Press Keys  ${SearchButton}  RETURN
Click on Advanced Search Link
  Click Element  ${AdvancedSearchLink}
```

To externalize keywords to another python file and import

Webelements.py

```
HomePageSearchTextBox="xpath://*[@id='gh-ac']"
HomePageSearchButton="xpath://*[@id='gh-btn']"
HomePageAdvancedSearchLink="xpath://*[@id='gh-as-a']"
```

HeaderPage.robot

```
*** Settings ***
Library  SeleniumLibrary
Variables  ../Webelements.py
*** Variables ***
*** Keywords ***
Input Search Text and Click Search
  [Arguments]  ${search_text}
  Input Text  ${HomePageSearchTextBox}  ${search_text}
  Press Keys  ${HomePageSearchButton}  RETURN
Click on Advanced Search Link
  Click Element  ${HomePageAdvancedSearchLink}
```

Variable Files

config.py	
firstname="Kamal" lastname="Girdher" profile="trainer"	*** Settings *** Variables config.py *** Test Cases *** TC3 Log ${firstname}${space}${lastname}

5.Variable Scope & Arguments in Robot Framework

Variable Scope in Robot Framework

The recommendation for use of local variables is lowercase and for global variables is upper case.i.e, ${variable_demo} and ${VARIABLE_DEMO}

Global scope	Test case scope
Global variables are available everywhere in the test data. **Test suite scope** Variables with the test suite scope are available anywhere in the test suite where they are defined or imported.	Variables with the test case scope are visible in a test case and in all user keywords the test uses. **Local scope** Test cases and user keywords have a local variable scope that is not seen by other tests or keywords.

- Variables at test case level are defined by using **Set Variable.**

VariableScope.robot

VariableScope.robot (Global & Test Suite scope)

```
*** Settings ***
*** Variables ***
${VARIABLE_DEMO} =  This is GLOBAL variable
*** Test Cases ***
This is demo test 1
  Log  ${VARIABLE_DEMO}
This is demo test 2
  Log  ${VARIABLE_DEMO}
```

VariableScope.robot (Test Case Scope)

```
*** Settings ***
*** Variables ***
#${VARIABLE_DEMO} =  This is GLOBAL variable
*** Test Cases ***
This is demo test 1
  ${variable_demo} =  Set Variable  This is TESTCASE variable
  Log  ${VARIABLE_DEMO}
This is demo test 2
  Log  ${VARIABLE_DEMO}
This is demo test 3
  This is demo keyword
*** Keywords ***
This is demo keyword
  log  ${VARIABLE_DEMO}
```

VariableScope.robot (Test case Scope)

```
*** Settings ***
*** Variables ***
#${VARIABLE_DEMO} =  This is GLOBAL variable
*** Test Cases ***
This is demo test 1
  ${variable_demo} =  Set Variable  This is TESTCASE variable
  Log  ${VARIABLE_DEMO}
This is demo test 2
  Log  ${VARIABLE_DEMO}
```

VariableScope.robot

```
*** Settings ***
*** Variables ***
${VARIABLE_DEMO} =  This is GLOBAL variable
*** Test Cases ***
This is demo test 1
  ${variable_demo} =  Set Variable  This is TESTCASE variable
  Log  ${VARIABLE_DEMO}
This is demo test 2
  Log  ${VARIABLE_DEMO}
This is demo test 3
  This is demo keyword
*** Keywords ***
```

```
This is demo keyword
  log  ${VARIABLE_DEMO}
```

VariableScope.robot (Local Scope)

```
*** Settings ***
*** Variables ***
${VARIABLE_DEMO} =  This is GLOBAL variable
*** Test Cases ***
This is demo test 1
  ${variable_demo} = Set Variable  This is TESTCASE variable
  Log  ${VARIABLE_DEMO}
This is demo test 2
  Log  ${VARIABLE_DEMO}
This is demo test 3
  This is demo keyword
*** Keywords ***
This is demo keyword
  [Arguments]  ${variable_demo}=This is KEYWORD variable
  log  ${VARIABLE_DEMO}
```

VariableScope.robot (Local Scope)

```
*** Settings ***
*** Variables ***
#${VARIABLE_DEMO} =  This is GLOBAL variable
*** Test Cases ***
This is demo test 1
  ${variable_demo} = Set Variable  This is TESTCASE variable
  Log  ${VARIABLE_DEMO}
This is demo test 2
  Log  ${VARIABLE_DEMO}
This is demo test 3
  This is demo keyword
*** Keywords ***
This is demo keyword
  [Arguments]  ${variable_demo}=This is KEYWORD variable
  log  ${VARIABLE_DEMO}
```

Arguments in Robot Framework

We can set up or define our keywords and we can define how many arguments are required for that particular keyword .suppose we are trying to test a website which requires different username and password i.e to verify the login functionality for that website we can define login keyword and define two arguments for username and password and then we can pass different values in that username and password so we can test for valid/invalid credentials.That is the whole purpose of defining keywords with arguments

ArgumentsDemo.robot

```
*** Settings ***
*** Test Cases ***
Argument demo keyword test
  Argument demo keyword  rcv  academy
Argument demo keyword test2
  Argument demo keyword  test  testing
*** Keywords ***
Argument demo keyword
  [Arguments]  ${arg1}  ${arg2}
```

Userkeywords.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Variables ***
${url}      http://www.newtours.demoaut.com/
${browser}  chrome
*** Test Cases ***
TC1
  open browser  ${url}  ${browser}
  maximize browser window
```

```
  Log  ${arg1}
  Log  ${arg2}
```

Creating User Defined Keywords with NO Argument

```
*** Test Cases ***
TC1
  launchAppication    # User defined Keyword with NO Argument
  input text  name:userName   mercury
  input text  name:password   mercury
*** Keywords ***
launchAppication
  open browser  ${url}   ${browser}
  maximize browser window
```

```
  input text  name:userName   mercury
  input text  name:password   mercury
*** Keywords ***
```

Creating User Defined Keywords with Argument

```
*** Test Cases ***
TC1
  launchAppication   ${url}  ${browser}   # User defined Keyword with Arguments
  input text  name:userName   mercury
  input text  name:password   mercury
*** Keywords ***
launchAppication
  [Arguments]  ${appurl}  ${appbrowser}
  open browser  ${appurl}   ${appbrowser}
  maximize browser window
```

Creating User Defined Keywords with Argument & Return Value

```
*** Test Cases ***
TC1
  ${pageTitle}=   launchAppication   ${url}  ${browser}   # User defined Keyword with Arguments
  log to console  ${pageTitle}
  log  ${pageTitle}
  input text  name:userName   mercury
  input text  name:password   mercury
*** Keywords ***
launchAppication
  [Arguments]  ${appurl}  ${appbrowser}
  open browser  ${appurl}   ${appbrowser}
  maximize browser window
  ${title}=   get title
  [Return]  ${title}
```

Normally in our projects we will create a resource file where all the keywords are kept,whichever test case wants to use those keywords can just use it to achieve reusability.so we will keep Test Cases and Keywords a separate file.
Create a Resources Directory under this create resources.robot file.Inside this will write all the keywords along with settings.

External resources

A resource is more or less similar to a test suite. The only difference is that we do not write test cases in Resource files. However, we write reusable functions(called User Keywords) and we define scalars, lists and dictionaries specific to that resource

resource1.robot

```
*** Keywords ***
UserKeyword1
  Log    This is keyword 1
UserKeyword2
```

SuiteImportingResource.robot

```
*** Settings ***
Resource        resource1.robot
*** Test Cases ***
TC1
```

Log *This is keyword 2*	Run Keyword *UserKeyword1* UserKeyword1

We have used the built-in function "Run Keyword" to run our user defined keyword. Now you can easily use many Execution control functions in different ways.

Creating User Defined Keywords with Argument & Return Value in Resources File

Resources.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Keywords ***
launchAppication
  [Arguments]  ${appurl}  ${appbrowser}
  open browser  ${appurl}   ${appbrowser}
  maximize browser window
  ${title}=   get title
  [Return]  ${title}
```

While using keywords which are in a separate file we have to specify Resource also.

UserKeywords.robot

```
*** Settings ***
Library     SeleniumLibrary
Resource  ../Resources/Resources.robot
*** Variables ***
${url}     http://www.newtours.demoaut.com/
${browser}   chrome
*** Test Cases ***
TC1
  ${pageTitle}=   launchAppication   ${url}
${browser}   # User defined Keyword with
Arguments
  log to console  ${pageTitle}
  log  ${pageTitle}
  input text  name:userName   mercury
  input text  name:password   mercury
```

6.Setup & TearDown, Tagging Robot Framework

Test setup and Teardown will run irrespective of passing or failure of the steps in the test case.

- ➔ Test Setup - will run before every Test Case
- ➔ Test Teardown - will run after every Test Case
- ➔ Suite Setup - will run before Test Suite
- ➔ Suite Teardown - will run after Test Suite

SetupAndTearDown.robot

```
*** Settings ***
Suite Setup  log to console  Opening Browser
Suite Teardown  log to console  Closing Browser
Test Setup  log to console  Login to application
Test Teardown  log to console  Logout from application
*** Test Cases ***
TC1 Prepaid Recharge
  log to console  This is prepaid recharge testcase
TC2 Postpaid Recharge
  log to console  This is postpaid recharge testcase
TC3 Search
  log to console  This is Search test case
TC4 Advanced Seearch
  log to console  This is Adv Search Test case
```

BasicSearch.robot

```
*** Settings ***
Documentation  Basic Search Functionality
Resource  ../Resources/CommonFunctionality.robot
Resource  ../Resources/UserDefinedKeywords.robot
Test  Setup  CommonFunctionality.Start TestCase
Test  Teardown  CommonFunctionality.Finish TestCase
*** Variables ***
*** Test Cases ***
Verify Successful Login to OrangeHRM
  [documentation]  This test case verifies that user is able to successfully Login to OrangeHRM
  [tags]  Smoke
  loginToApplication
```

```
*** Settings ***
Suite Setup  suite_setup
Suite Teardown  suite_teardown
Test Setup  test_setup
Test Teardown  test_teardown
*** Test Cases ***
TC1
  Log  1
TC2
  Log  2
TC3
  Log  3
*** Keywords ***
suite_setup
  Log  SETUP SUITE
suite_teardown
  LOG  SUITE ENDS
test_setup
  LOG  TEST START
test_teardown
  Log  TEST ENDS
```

Tagging	**Tagging.robot**
Tags in Robot Framework are for classifying test cases. Tags are free text and they can be used at least for the following purposes: • Tags are shown in test reports, logs, and in the test data, so they provide metadata to test cases. • Statistics about test cases (total, passed, failed are automatically collected based on tags). • With tags, you can include or exclude test cases to be executed. • With tags, you can specify which test cases should be skipped.	*** Settings *** *** Test Cases *** TC1 User RegistrationTest [tags] *regression* log to console *This is user reg test* log to console *user registration test is over* TC2 LoginTest [tags] *sanity* log to console *This is login test* log to console *Login test is over*

Execution :

- ➔ robot --include=sanity TestCases\Tagging.robot
- ➔ robot --include=regression TestCases\Tagging.robot
- ➔ **To Include multiple Test Cases -** robot -i sanity -i regression TestCases\Tagging.robot
- ➔ **To exclude particular Test Case -** robot -e regression TestCases\Tagging.robot
- ➔ **To Include and Exclude -** robot -e regression -i sanity TestCases\Tagging.robot

There are certain keywords which can be used in suite teardown.

Run Keyword If All Tests Passed

Run Keyword If Any Tests Failed

There are certain keywords which are for test teardown.

Run Keyword If Test Failed

Run Keyword If Test Passed

Run Keyword If Timeout Occurred

7.Loop Statements

- Loops allow us to iterate over a sequence
- You can use Loops to
  - Loop through a list of elements
  - Repeat a single keyword several times
  - Loop through range of numbers (1-10)
- Loops begin with "FOR"
- Example:

```
FOR ${var} IN @{list}
    Keyword ${var}
END
```

Loop functions

Continue For Loop , Continue For Loop If , Exit For Loop , Exit For Loop If

ForLoopDemo.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
${url}  https://www.google.com/
${browser}  chrome
*** Test Cases ***
Test Case to demonstrate FOR Loop in Robot Framework
  [Documentation]  Test Case to demonstrate FOR Loop in Robot Framework
  Set Selenium Implicit Wait  5s
  Open Browser  ${url}  ${browser}
  Maximize Browser Window
  Input Text  name:q  RCV Academy
  Press Keys  xpath://*[@id="tsf"]//div[2]/ul/li  RETURN
  @{results_on_page}=  Get WebElements  xpath://*[@id="rso"]/div
  FOR  ${element}  IN  @{results_on_page}
      ${text}=  Get Text  ${element}
  END
  Close Browser
```

ForLoop.robot

```
*** Test Cases ***
```

Forloop1

```
  FOR  ${i}  IN RANGE  1  10
    log to console  ${i}
  END
```

Forloop2

```
  FOR  ${i}  IN  1  2  3  4  5  6  7  8  #double space - next line, single space-same line
     log to console  ${i}
  END
```

Forloop3withList

```
  @{items}  Create List  1  2  3  4  5
  FOR  ${i}  IN  @{items}
```

```
    log to console    ${i}
  END
```

Forloop4

```
  FOR   ${i}   IN  john  david  smith  scott  #double space - next line, single space-same line
    log to console    ${i}
  END
```

Forloop5withList

```
  @{nameslist}    Create List  john  david  smith  scott
  FOR   ${i}    IN  @{nameslist}
    log to console  ${i}
  END
```

Forloop6withExiting - Condition Based

```
*** Test Cases ***
TC4 - Exit for loop in between
  @{a}=    Create List    1    2    3    4    5
  FOR    ${i}    IN    @{a}
    Log    ${i}
    Exit For Loop
  END
```

```
TC5 - Conditional Exit from For Loop
  @{a}=    Create List    1    2    3    4    5
  FOR    ${i}    IN    @{a}
    Exit For Loop If    ${i}>3
    Log    ${i}
  END
```

```
  @{items}    Create List  1   2   3   4   5
  FOR   ${i}    IN  @{items}
    log to console    ${i}
    Exit For Loop If  ${i} == 2
    log to console  Exit For Loop triggered at the
second element
  END
```

8.Conditional Functions

Unlike any programming language, there is no If..Else or Switch..Case statement in robot framework. However, there is a set of functions which are used to implement conditional logic in our tests.

```
Run Keyword If ${condition} == “Some Data” Keyword1
... ELSE IF ${condition} == "Some Other Data" Keyword2
... ELSE Keyword3
```

Set Variable If	**Return From Keyword If**
Get Variable Value	**Run Keyword And Return If**
Run Keyword If	**Run Keyword If All Tests Passed**
Run Keyword Unless	**Run Keyword If Any Tests Failed**
Continue For Loop If	**Run Keyword If Test Failed**
Exit For Loop If	**Run Keyword If Test Passed**
Pass Execution If	**Run Keyword If Timeout Occurred**

Conditional Functions.robot

```
*** Settings ***
Resource        resource1.robot
*** Test Cases ***
TC1 - Set Variable If
  #If condition returns true
  ${cond}=   Set Variable   True
  ${a}=   Set Variable If   ${cond}==True   10
  Log   ${a}
  #If condition returns false
  ${a}=   Set Variable If   ${cond}==False   10   0
  Log   ${a}
TC2 - Run Keyword If
  ${cond}=   Set Variable   True
  Run Keyword If   ${cond}==True   UserKeyword1   ELSE   UserKeyword2
TC3 - Run Keyword Unless
  FOR   ${i}   IN RANGE   1   10
    Log   ------
    Run Keyword Unless   ${i}>5   Log   Iteration=${i}
  END
TC4 - Continue For Loop If
  FOR   ${i}   IN RANGE   1   10
    Log   Starting ${i}
    Continue For Loop If   ${i}>5
    Log   Ending ${i}
  END
```

```
TC5 - Exit For Loop If
  FOR    ${i}    IN RANGE    1    10
     Log    Starting ${i}
     Exit For Loop If    ${i}>5
     Log    Ending ${i}
  END
TC6 - Pass Execution If
  ${i}=    Set Variable    10
  Pass Execution If    ${i}>5    Passing the execution
  Fail    Forcefully failing the test
TC7 - Return From Keyword If
  ${b}=    Userkeyword3
#TC8 - Run Keyword And Return If
  #we will see the usage later
*** Keywords ***
Userkeyword3
  ${a}=    Set Variable    10
  Return From Keyword If    ${a}<5    Hello
  Return From Keyword If    ${a}>5    Hi
```

IfElseDemo.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
${url}  https://www.saucedemo.com/
${browser}  chrome
*** Test Cases ***
Test Case to demonstrate IF/ELSE in Robot Framework
  [Documentation]  Test Case to demonstrate IF/ELSE in Robot Framework
  Open Browser  ${url}  ${browser}
  Maximize Browser Window
  Input Text  id:user-name  standard_user
  Input Text  id:password  secret_sauce
  Click Button  xpath://*[@id="login_button_container"]/div/form/input[3]
  ${items_on_page}=  Get Element Count  xpath://*[@class="inventory_list"]/div
  Run Keyword If  ${items_on_page} == 10  Test Keyword 1
  ...  ELSE IF  ${items_on_page} < 10 and ${items_on_page} > 6  Test Keyword 2
  ...  ELSE  Test Keyword 3
*** Keywords ***
Test Keyword 1
  Log To Console  Executed Keyword1 - Found Items as expected
```

```
    Close Browser
Test Keyword 2
    Log To Console  Executed Keyword2 - Found less than expected Items
    Close Browser
Test Keyword 3
    Log To Console  Executed Keyword3 - Exception
    Close Browser
```

9.Working with Browser and Navigations.

WORKING WITH BROWSER

Open Browser - Opens a new browser instance to the optional url

Examples:

```
Open Browser  http://example.com  Chrome
Open Browser  http://example.com  Firefox  alias=RCV
Open Browser  about:blank
Open Browser  browser=Chrome
```

WORKING WITH BROWSER

- **Close browser** - Closes the current browser
- **Close All Browsers** - Closes all open browsers and resets the browser cache. Should be used in Test or Suite teardown to ensure all open browsers are closed
- **Get Browser Ids** - Returns index of all active browser as list
- **Maximise Browser Window** - Maximizes current browser window
- **Get Browser Alias** - Returns aliases of all active browser that has an alias as NormalizedDict
- **Switch Browser** - Switches between active browsers using index_or_alias

WorkingWithBrowser.robot

```
*** Settings ***
Library    SeleniumLibrary
*** Variables ***
*** Test Cases ***
MultipleBrowsersTest
    [Documentation]    TC to demonstrate Browser Operation keywords in Robot Framework
    open browser    http://google.com/    chrome    alias=KMR
    maximize browser window
    open browser    https://www.amazon.com/    chrome    alias=KMMR
    maximize browser window
```

```
    @{browser_ID}=    get browser ids
    FOR    ${i}    IN    @{browser_ID}
        log to console    ${i}
    END
    switch browser  1
    ${title1}=   get title
    log to console  ${title1}
    switch browser  2
    ${title2}=  get title
    log to console  ${title2}
    &{a}=       get browser aliases
    FOR    ${alias}   IN @{a}
      Log to console    ${alias} # logs BrowserA and BrowserB
    END
#      close browser
    close all browsers
```

*** Keywords ***

If we have opened 4 browsers we will have 4 indexes and so on

Navigations

Browser related keywords are **Go To, Go Back, Get Location**

NavigationalKeywords.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
NavigationTest
  open browser  https://www.google.com/
chrome
  ${loc}=    get location
  log to console  ${loc}
  sleep  3
  go to  https://www.bing.com/
 log to console  ${loc}
  sleep  3
  go back
  ${loc}=    get location
  log to console  ${loc}
  sleep  3
  close browser
```

Note:

- Go To Navigates the current browser window to the provided url.
- Go Back Simulates the user clicking the back button on their browser.

10.Working with Webelements

WORKING WITH WEBELEMENTS

- **Get Element Attribute** - Returns the value of attribute from the element locator
- **Get Element Count** - Returns the number of elements matching locator
- **Get Element Size** - Returns width and height of the element identified by locator
- **Get WebElement** - Returns the first WebElement matching the given locator
- **Get WebElements** - Returns a list of WebElement objects matching the locator
- **Capture Element** Screenshot - Captures a screenshot from the element and embeds it into log file
- **Assign Id To Element** - Assigns a temporary id to the element specified by locator
- **Clear Element Text** - Clears the value of the text-input-element identified by locator
- **Double Click Element** - Double clicks the element identified by locator
- **Cover Element** - Will cover elements identified by locator with a blue div without breaking page layout
- **Click Element At Coordinates** - Click the element locator at xoffset/yoffset

WORKING WITH WEBELEMENTS

- **Element Attribute Value Should Be** - Verifies element identified by locator contains expected attribute value
- **Element Should Be Disabled** - Verifies that element identified by locator is disabled
- **Element Should Be Visible** - Verifies that the element identified by locator is visible
- **Element Should Not Be Visible** - Verifies that the element identified by locator is NOT visible
- **Element Should Contain** - Verifies that element locator contains text expected
- **Element Should Not Contain** - Verifies that element locator does not contain text expected
- **Element Text Should Be** - Verifies that element locator contains exact the text expected
- **Element Text Should Not Be** - Verifies that element locator does not contain exact the text not_expected
- **Element Should Be Enabled** - Verifies that element identified by locator is enabled
- **Element Should Be Focused** - Verifies that element identified by locator is focused

WorkingWithWebelements.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
Test Case to demonstrate WebElement Operation Keywords
  [Documentation]  Test Case to demonstrate WebElement Operation Keywords
  Open Browser  http://google.com  Chrome  alias=ChromeRCV
  Maximize Browser Window
 ${attr}=  Get Element Attribute  xpath://*[@id="tsf"]//div[3]/center/input[1]  class
```

```
${count}=  Get Element Count  xpath://*[@id="tsf"]//div[3]/center/input[1]
${width}  ${height}=  Get Element Size  xpath://*[@id="tsf"]//div[3]/center/input[1]
Get WebElement  xpath://*[@id="tsf"]//div[3]/center/input[1]
@{webelements}=  Get WebElements  xpath://*[@id="tsf"]//div[3]/center/input[1]
 Capture Element Screenshot  xpath://input[@name='q']
 Assign Id To Element  name:btnK  RCVid
 Page Should Contain Element     RCVid
 Element Should Be Focused  xpath://input[@name='q']
 Element Should Be Visible  xpath://input[@name='q']
Input Text  xpath://input[@name='q']  RCVAcademy
Clear Element Text  xpath://input[@name='q']
Cover Element  xpath://*[@id="tsf"]/div[2]/div[1]/div[3]/center/input[2]
Element Attribute Value Should Be  name:btnK  value  Google Search
Element Should Be Enabled  name:btnK
Element Should Not Be Visible  name:btnKgf
Element Should Contain  xpath://*[@id="fsl"]/a[3]  How Search
Element Should Not Contain  xpath://*[@id="fsl"]/a[3]  Google Search
Element Text Should Be  xpath://*[@id="fsl"]/a[3]  How Search works
Element Text Should Not Be  xpath://*[@id="fsl"]/a[3]  How Search
Double Click Element  xpath://*[@id="gb_70"]
Click Element At Coordinates  xpath://*[@id="identifierNext"]/div[2]  15  10
#Element Should Be Disabled  name:btnK
Close Browser
```

Inputbox.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
${browser}  chrome
${url}  https://demo.nopcommerce.com
*** Test Cases ***
TestingInputBox
  open browser  ${url}  ${browser}
  maximize browser window
  title should be  nopCommerce demo store
  click link  xpath://a[@class='ico-login']
  ${"email_txt"}  set variable  id:Email
  element should be visible  ${"email_txt"}
  element should be enabled  ${"email_txt"}
  input text  ${"email_txt"}  JohnDavid@gmail.com
  sleep  5
  clear element text  ${"email_txt"}
  sleep  3
  close browser
*** Keywords ***
```

- Visibility status
- Enabled status
- Provide value
- Clearing value
- Verify Title of the Page

11.Capture Screenshots

CaptureScreen.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Test Cases ***
LoginTC
  open browser    https://opensource-demo.orangehrmlive.com/      chrome
  maximize browser window
  input text  id:txtUsername  Admin
  input text  id:txtPassword  admin123
  # capture element screenshot  xpath://*[@id="divLogo"]/img
C:/Users/admin/PycharmProjects/Automation/logo.png
  # capture page screenshot  C:/Users/admin/PycharmProjects/Automation/LoginTC.png
  capture element screenshot  xpath://*[@id="divLogo"]/img    logo.png
  capture page screenshot     LoginTC.png
```

12.Selenium Speeds Timeouts & Waits

Selenium Speeds Timeouts

Get Selenium Timeout - Gets the timeout that is used by various keywords

Set Selenium Timeout - Sets the timeout that is used by various keywords

Get Selenium Speed - Gets the delay that is waited after each Selenium command

Set Selenium Speed - Sets the delay that is waited after each Selenium command

Waits in selenium

Wait commands in selenium direct test script to pause for certain time before throwing exception

There are 3 types of wait in Selenium

- Implicit wait
- Explicit wait
- Fluent wait

Both Implicit and Explicit waits are dynamic waits.

Implicit Wait

- Implicit wait tells the WebDriver maximum wait time when searching for elements before throwing exception
- Implicit wait is global, it is applicable to all the webelements in the script
- Since Implicit wait applies to all webelements you do not specify "ExpectedConditions" on the element to be located

Set Selenium Implicit Wait - Sets the implicit wait value used by Selenium

Get Selenium Implicit Wait - Gets the implicit wait value used by Selenium

Set Browser Implicit Wait - Same as Set Selenium Implicit Wait but only affects the current browser

Wait Until Element Contains - Waits until the element locator contains text

Wait Until Element Does Not Contain - Waits until the element locator does not contain text

Wait Until Element Is Enabled - Waits until the element locator is enabled

Wait Until Element Is Not Visible - Waits until the element locator is not visible

Wait Until Element Is Visible - Waits until the element locator is visible

Explicit Wait

- Explicit waits tell the WebDriver to halt the execution until a particular condition is met or the maximum time has passed
- Explicit wait time is applicable only to those web elements which are specified by the user
- Explicit wait requires the expected conditions to be specified for elements, like "Wait Until Element Is Enabled".

Wait Until Page Contains - Waits until text appears on the current page

Wait Until Page Contains Element - Waits until the element locator appears on the current page

Wait Until Page Does Not Contain - Waits until text disappears from the current page

Wait Until Page Does Not Contain Element - Waits until the element locator disappears from the current page

Wait Until Location Is - Waits until the current URL is expected

Wait Until Location Is Not - Waits until the current URL is not location

Wait Until Location Contains - Waits until the current URL contains expected

Wait Until Location Does Not Contain - Waits until the current URL does not contain location

Fluent Wait - Fluent wait tells the WebDriver maximum wait time when searching for elements before

throwing an exception.Fluent Wait looks for a webelement repeatedly at regular intervals until timeout happens or until the object is found.With Fluent Wait, it is possible to set a default polling period as needed. You can configure the wait to ignore any exceptions during the polling period

If we want to pause or delay the script for some time we will go to sleep.if we wait at multiple places we will specify a sleep statement at multiple places.To avoid this and apply delay time in every statement we need to go for selenium speed.

If the particular element does not appear on the web page it should wait for a maximum time called selenium timeout.Default time is 5 secs.

If there is an invalid locator immediately there is a nosuchelementfound exception is raised.if want to wait for that element for sometime we can use implicit wait.Implicit wait is applicable for every statement in the script.If the element if found within the mentioned implicit time it will execute go and execute other statements.Implicit wait is more efficient than other things.

SpeedTests.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
RegTest
  ${spead}=  get selenium speed
  log to console  ${spead}
  open browser
http://demowebshop.tricentis.com/register
chrome
  maximize browser window
  set selenium speed  2 seconds
  select radio button  Gender  M
  input text  name:FirstName  David
  input text  name:LastName  John
  input text  name:Email  anhc@gmail.com
  input text  name:Password  davidjohn
  input text  name:ConfirmPassword
davidjohn
  ${spead}=  get selenium speed
  log to console  ${spead}
```

TimeOutTests.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
RegTest
  open browser
http://demowebshop.tricentis.com/register
chrome
  maximize browser window
  ${time}=  get selenium timeout
  log to console  ${time}
  set selenium timeout  10 seconds
  wait until page contains  Registration  # 5
secs
  select radio button  Gender  M
  input text  name:FirstName  David
  input text  name:LastName  John
  input text  name:Email  anhc@gmail.com
  input text  name:Password  davidjohn
  input text  name:ConfirmPassword
davidjohn
```

ImplicitWaitTests.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
RegTest
  open browser  http://demowebshop.tricentis.com/register  chrome
  maximize browser window
```

```
  ${implicttime}=    get selenium implicit wait
  log to console  ${implicttime}
  set selenium implicit wait  10 seconds
  ${implicttime}=    get selenium implicit wait
  log to console  ${implicttime}
  select radio button  Gender    M
  input text  name:FirstName1    David
  input text  name:LastName    John
  input text  name:Email    anhc@gmail.com
  input text  name:Password    davidjohn
  input text  name:ConfirmPassword    davidjohn
```

ExplicitWait.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
*** Test Cases ***
Test Case to demonstrate explicit wait in Robot Framework
  [Documentation]  Test Case to demonstrate explicit wait in Robot Framework
  Open Browser  https://www.sugarcrm.com/au/request-demo/  chrome
  Maximize Browser Window
  Scroll Element Into View  xpath://*[@id="menu-item-115"]/a
  Wait Until Page Contains  With our customers
  Wait Until Page Contains Element  xpath://*[@id="menu-item-19419"]/a
  Wait Until Page Does Not Contain  With our customerss
  Wait Until Page Does Not Contain Element  xpath://*[@id="menu-item-19419"]/abch
  Wait Until Location Is  https://www.sugarcrm.com/au/request-demo/
  Wait Until Location Is Not  https://www.sugarcrm.com/au/request-demo/abc
  Wait Until Location Contains  au
  Wait Until Location Does Not Contain  aud
  Wait Until Element Contains  xpath://*[@id="menu-item-19419"]/a  Deployment Options
timeout=10s  error=Found Text
  Wait Until Element Does Not Contain  xpath://*[@id="menu-item-16789"]/a  random text
  Wait Until Element Is Enabled  xpath://*[@id="menu-item-19419"]/a
  Wait Until Element Is Not Visible  xpath://*[@id="menu-item-19419"]/a/b
  Wait Until Element Is Visible  xpath://*[@id="menu-item-19419"]/a
  Close Browser
```

13.Radio Buttons,Checkboxes,Dropdowns and List Box

Page Should Contain Radio Button - Verifies radio button locator is found from current page

Page Should Not Contain Radio Button - Verifies radio button locator is not found from current page

Select Radio Button - Sets the radio buttongroup group_name to value

Radio Button Should Not Be Selected - Verifies radio button group group_name has no selection

Radio Button Should Be Set To - Sets the radio button group group_name to value

Select Checkbox - Selects the checkbox identified by locator

Unselect Checkbox - Removes the selection of checkbox identified by locator

Checkbox Should Be Selected - Verifies checkbox locator is selected/checked

Checkbox Should Not Be Selected - Verifies checkbox locator is not selected/checked

Page Should Contain Checkbox - Verifies checkbox locator is found from the current page

Page Should Not Contain Checkbox - Verifies checkbox locator is not found from the current page

WorkingWithRadioButtons.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
TC to demonstrate Radio button operation
keywords
  [Documentation]  TC to demonstrate Radio
button operation keywords
  Open Browser
https://www.sugarcrm.com/au/request-demo/
Chrome
  Maximize Browser Window
  Sleep  4s
  Page Should Contain Radio Button
xpath://*[@id="doi0"]
  Page Should Not Contain Radio Button
xpath://*[@id="doi00"]
  Radio Button Should Not Be Selected  doi
  Scroll Element Into View
xpath://*[@id="field1"]/div/input
  Select Radio Button  doi  0
  Sleep  4s
  Radio Button Should Be Set To  doi  0
  Close Browser
```

Radiobuttons.robot

```
*** Settings ***
Library    SeleniumLibrary
*** Variables ***
${browser}  chrome
${url}
http://www.practiceselenium.com/practice-form.
html
*** Test Cases ***
Testing Radio Buttons and Check Boxes
  open browser  ${url}  ${browser}
  maximize browser window
  set selenium speed  2seconds
            Selecting Radio buttons
  select radio button  sex  Female
  select radio button  exp  5
            Selecting Checkbox
  select checkbox  BlackTea
  select checkbox  RedTea
  unselect checkbox  BlackTea
```

WorkingWithCheckboxes.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
```

```
TC to demonstrate working with checkboxes in Robot FW
  [Documentation]  TC to demonstrate working with checkboxes in Robot FW
  Open Browser  https://www.sugarcrm.com/au/request-demo/  Chrome
  Maximize Browser Window
  Sleep  4s
  Scroll Element Into View  xpath://*[@id="field1"]/div/input
  Page Should Contain Checkbox  id:interest_market_c0
  Page Should Not Contain Checkbox  id:interest_market_c00
  Select Checkbox  id:interest_market_c0
  Checkbox Should Be Selected  id:interest_market_c0
  Capture Page Screenshot
  Sleep  4s
  Checkbox Should Not Be Selected  id:interest_sell_c0
  Unselect Checkbox  id:interest_market_c0
  Checkbox Should Not Be Selected  id:interest_market_c0
  Capture Page Screenshot
  Sleep  4s
  Close Browser
```

Handling Dropdowns & List Box

Get List Items - Returns all labels or values of selection list locator
Get Selected List Label - Returns the label of selected option from selection list locator
Get Selected List Value - Returns the value of selected option from selection list locator
Select From List By Index - Selects options from selection list locator by indexes
Select From List By Label - Selects options from selection list locator by labels
Select From List By Value - Selects options from selection list locator by values
Select All From List - Selects all options from multi-selection list locator
Get Selected List Labels - Returns labels of selected options from selection list locator
Get Selected List Values - Returns values of selected options from selection list locator
Unselect From List By Index - Unselects options from selection list locator by indexes
Unselect From List By Label - Unselects options from selection list locator by labels
Unselect From List By Value - Unselects options from selection list locator by values
Unselect All From List - Unselects all options from multi-selection list locator
List Selection Should Be - Verifies selection list locator has expected options selected
List Should Have No Selections - Verifies selection list locator has no options selected
Page Should Contain List - Verifies selection list locator is found from current page
Page Should Not Contain List - Verifies selection list locator is not found from current page

dropdowns.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Variables ***
${browser}  chrome
${url} http://www.practiceselenium.com/practice-form.html
```

```
*** Test Cases ***
Handling DropDowns and Lists
  open browser  ${url}    ${browser}
  maximize browser window
  select from list by label  continents   Asia
  sleep  3
  select from list by index  continents   5
  #select from list by value  continents  value
                                                    list box
  select from list by label  selenium_commands    Switch Commands
  select from list by label  selenium_commands    WebElement Commands
  sleep  3
  unselect from list by label     selenium_commands   Switch Commands
```

14.Alerts/Popups and Frames

Alerts - Sometimes when we are automating test cases we may see alert windows/Popup windows.Alert is not a web element it is a special type of **web object**.we cannot inspect anything on this alert window.we need to handle this in different ways.

Handle Alert - Handles the current alert and returns its message.

Input Text Into Alert - Types the given text into an input field in an alert.

Alert Should Be Present - Verifies that an alert is present and by default, accepts it

Alert Should Not Be Present - Verifies that no alert is present.

Alerts.robot

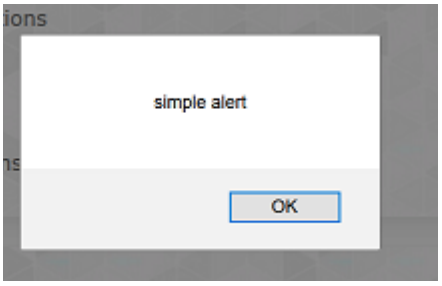
```
*** Settings ***
Library     SeleniumLibrary
*** Test Cases ***
HandlingAlerts
   open browser  https://testautomationpractice.blogspot.com/  chrome
   click element  xpath://*[@id="HTML9"]/div[1]/button  # opens alert
   sleep    3
   handle alert  accept
   handle alert    dismiss
   handle alert  leave
   alert should not be present  Press a button!
  # alert should  be present  Press a button!
```

HandlingAlerts.robot

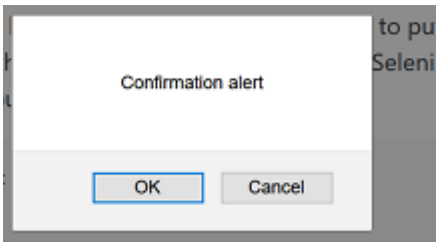
```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
*** Test Cases ***
Test Case to demostrate how to handle Alerts in Robot Framework
   [Documentation]  Test Case to demostrate how to handle Alerts in Robot Framework
   Open Browser  https://www.w3schools.com/js/tryit.asp?filename=tryjs_alert  chrome
   Select Frame  id:iframeResult
   Maximize Browser Window
   Click Button  xpath://body/button
   Sleep  2s
   Handle Alert  ACCEPT  timeout=5 s
   Go To  https://www.w3schools.com/js/tryit.asp?filename=tryjs_alert
   Select Frame  iframeResult
   Maximize Browser Window
   Click Button  xpath://body/button
   Sleep  2s
   ${message1}=  Handle Alert  action=ACCEPT  timeout=2s
   Go To  https://www.w3schools.com/js/tryit.asp?filename=tryjs_confirm
   Select Frame  iframeResult
   Click Button  xpath://body/button
```

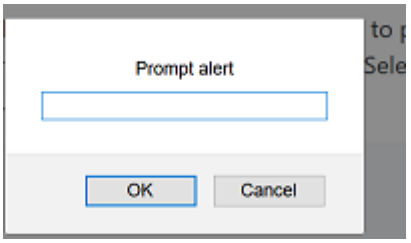
```
  Sleep  2s
  ${message2}=  Handle Alert  DISMISS  2s
 Go To  https://www.w3schools.com/js/tryit.asp?filename=tryjs_prompt
 Select Frame  iframeResult
 Sleep  2s
 Click Button  xpath://body/button
  Input Text Into Alert  RCVAcademy  action=DISMISS
  Sleep  2s
  Go To  https://www.w3schools.com/js/tryit.asp?filename=tryjs_alert2
  Select Frame  iframeResult
  Click Button  xpath://body/button
  Sleep  2s
  Alert Should Be Present  text=Hello How are you?  action=ACCEPT
  Go To  https://www.sugarcrm.com/au/request-demo/
  Sleep  2s
  Alert Should Not Be Present  action=ACCEPT, timeout=2s
  Close Browser
```

Alert.robot




```
*** Settings ***
Library         SeleniumLibrary
*** Test Cases ***
TC1
  Open Browser    https://www.google.com    chrome
  Execute Javascript    alert("Create a simple alert.")
  Sleep    2s
  Handle Alert    ACCEPT
  Sleep    1s
  Execute Javascript    var x= confirm("Confirmation alert.")
  Sleep    2s
  Handle Alert    DISMISS
  Sleep    1s
  Execute Javascript    var y= prompt("Prompt alert.")
  Sleep    2s
  Input Text Into Alert    hello there    ACCEPT
```





Frames - Generally the webpage is divided into multiple sections called Frames/Iframes. Sometimes we may see one webpage displayed on another webpage which is embedded in it.

- **Windows - Frames, Webapplications - Iframes**

The reason for working with frames is if there are any elements which are present inside the Frame/Iframe we cannot directly interact with it / do any validations on that element.if we want to interact we need to switch to a particular frame and we need to interact.

Select Frame - Sets frame identified by locator as the current frame

Unselect Frame - Sets the main frame as the current frame.

Frame Should Contain - Verifies that frame identified by locator contains text

Current Frame Should Contain - Verifies that the current frame contains text

Current Frame Should Not Contain - Verifies that the current frame does not contain text

Frames.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Test Cases ***
Testing Frames
   open browser  https://seleniumhq.github.io/selenium/docs/api/java/index     chrome
   maximize browser window
   select frame  packageListFrame      #id name  xpath
   click link  org.openqa.selenium
   unselect frame
   sleep  3
   select frame  packageFrame
   click link  WebDriver
   unselect frame
   sleep  3
   select frame  classFrame
   click link  Index
   unselect frame
   sleep  3
   close browser
```

WorkingWithFrames.robot

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
${url}  http://google.com
${browser}  chrome
*** Test Cases ***
TC to demonstrate working with frames in Robot FW
   [Documentation]  TC to demonstrate working with frames in Robot FW
   Open Browser  https://www.w3schools.com/js/tryit.asp?filename=tryjs_alert  Chrome
   Maximize Browser Window
   Select Frame  id:iframeResult
   Current Frame Should Contain  JavaScript Alert
   Current Frame Should Not Contain  JavaScript Confirm Box Two
   Unselect Frame
   Frame Should Contain  id:iframeResult  JavaScript Alert
   Close Browser
```

15.WebTable and Links

Working with HTML Table

Count Number of Rows in a Table, Count Number of Columns in a Table, Get data from a Table, Validations on a Table.

TableDemo.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Test Cases ***
TableValidations
  open browser  https://testautomationpractice.blogspot.com/     chrome
  maximize browser window
  ${rows}=    get element count  xpath://table[@name='BookTable']/tbody/tr
  ${cols}=    get element count  xpath://table[@name='BookTable']/tbody/tr[1]/th
  log to console  ${rows}
  log to console  ${cols}
  ${data}=    get text    xpath://table[@name='BookTable']/tbody/tr[5]/td[1]
  log to console  ${data}
  table column should contain  xpath://table[@name='BookTable']   2   Author
  table row should contain  xpath://table[@name='BookTable']  4   Learn JS
  table cell should contain  xpath://table[@name='BookTable']    5   2   Mukesh
  table header should contain  xpath://table[@name='BookTable']   BookName
  close browser
```

- Go and Observe report.html

Links

➔ Count Number of Links on Web Page, Extract all the Links from Page

GetAllLinks.robot

```
*** Settings ***
Library     SeleniumLibrary

*** Test Cases ***
GetAllLinksTest
    open browser    http://www.newtours.demoaut.com/     chrome
    maximize browser window
    ${AllLinksCount}=    get element count  xpath://a
    log to console  ${AllLinksCount}

    @{LinkItems}    create list

    : FOR    ${i}    IN RANGE    1    ${AllLinksCount}+1
    \    ${linkText}=  get text  xpath:(//a)[${i}]
    \    log to console  ${linkText}
```

16.Mouse Operations & Scrolling Page

```
*** Settings ***
Library  SeleniumLibrary
*** Variables ***
*** Test Cases ***
Test Case to demonstrate mouse operations in Robot Framework
  [Documentation]  Test Case to demonstrate mouse operations in Robot Framework
  Open Browser  https://www.sugarcrm.com/au/request-demo/  chrome
  Maximize Browser Window
  Scroll Element Into View  id:interest_market_c0
  Mouse Down  xpath://*[@id="field25"]/div/input
  Sleep  2s
  Mouse Up  xpath://*[@id="field25"]/div/input
  Sleep  2s
  Scroll Element Into View  xpath://*[@id="menu-item-115"]/a
  Sleep  4s
  Mouse Down On Link  xpath://*[@id="menu-item-107"]/a
 Sleep  4s
 Mouse Over  xpath://*[@id="menu-item-19419"]/a
 Sleep  4s
 Mouse Out  xpath://*[@id="menu-item-19419"]/a
 Sleep  4s
  Mouse Down On Image  xpath://footer//div[1]/a/img
 Sleep  4s
  #Drag and drop example
 Go to  https://demoqa.com/droppable/
  Drag And Drop  xpath://*[@id="draggable"]/p  xpath://*[@id="droppable"]
  Sleep  4s
  #Right click on element
  Open Context Menu  xpath://*[@id="sidebar"]/aside[2]/ul/li[2]/a
  Sleep  2s
  Close Browser
```

MouseOperations.robot

```
*** Test Cases ***
MouseActions
```

Right click/open open context menu

```
  open browser  http://swisnl.github.io/jQuery-contextMenu/demo.html  chrome
  maximize browser window
  open context menu  xpath://span[@class='context-menu-one btn btn-neutral']
  sleep  3
```

Double click action

```
go to  https://testautomationpractice.blogspot.com/
maximize browser window
double click element  xpath://button[contains(text(),'Copy Text')]
sleep   3
```

Drag and Drop

```
go to   http://www.dhtmlgoodies.com/scripts/drag-drop-custom/demo-drag-drop-3.html
maximize browser window
drag and drop  id:box6  id:box106
sleep  5
close browser
```

Scrolling Page

Scrolling Page using JavaScriptExecutor

Scrolling page till it reach a pixel number, Scrolling page till find element on page, Scroll page till the bottom

ScrollingPage.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Test Cases ***
ScrollingTestr
  open browser  https://www.countries-ofthe-world.com/flags-of-the-world.html  chrome
  maximize browser window
  #execute javascript  window.scrollTo(0,1500)
  #scroll element into view  xpath://*[@id="content"]/div[2]/div[2]/table[1]/tbody/tr[105]/td[1]/img
  execute javascript  window.scrollTo(0,document.body.scrollHeight)
  sleep   3
  execute javascript  window.scrollTo(0,-document.body.scrollHeight)    # starting position
```

17.Handling Tabbed Windows

Working with Windows

Get Window Handles - Returns all current window handles as a list

Get Window Identifiers - Returns and logs id attributes of all known browser windows

Get Window Names - Returns and logs names of all known browser windows

Get Window Titles - Returns and logs titles of all known browser windows

Set Window Position - Sets window position using x and y coordinates

Get Window Position - Returns current window position

Set Window Size - Sets current windows size to given width and height

Get Window Size - Returns current window width and height as integers

Select Window - DEPRECATED in SeleniumLibrary 4.0. use Switch Window

Switch Window - Switches to browser window matching locator

Close Window - Closes currently opened and selected browser window/tab

TabbedWindows.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Test Cases ***
TabbedWindowsTest
   open browser  http://demo.automationtesting.in/Windows.html     chrome
   click element  xpath://*[@id="Tabbed"]/a/button
   switch window  title=Sakinalium | Home
   click element  xpath://*[@id="container"]/header/div/div/div[2]/ul/li[4]/a
   sleep   3
   close all browsers
*** Settings ***
Library  SeleniumLibrary
*** Test Cases ***
TC to demostrate Browser Window Operation Keywords in Robot FW
   [Documentation]  TC to demostrate Browser Window Operation Keywords in Robot FW
   Open Browser  http://salesforce.com  Chrome  alias=ChromeRCV
   Maximize Browser Window
   Sleep  4s
   Wait Until Element Is Visible
xpath://*[@id="main"]/div[6]/div/div[5]/div/div[2]/div/div/div[2]/div[1]/div[2]/div[2]/div/div[2]/div[1]/div[6
]/div/div/div/a
   Click Link
xpath://*[@id="main"]/div[6]/div/div[5]/div/div[2]/div/div/div[2]/div[1]/div[2]/div[2]/div/div[2]/div[1]/div[6
```

```
]/div/div/div/a
  @{WindowHandles}=  Get Window Handles
  Sleep  4s
  @{WindowIdentifier}=  Get Window Identifiers
  Sleep  4s
 @{WindowNames}=  Get Window Names
 Sleep  4s
 @{WindowTitle}=  Get Window Titles
  Sleep  4s
 Set Window Position  100  200
 ${x}  ${y}=  Get Window Position
  Log  ${x}
  Log  ${y}
  Sleep  4s
  Set Window Size  800  600
 ${width}  ${height}=  Get Window Size
  Log  ${width}
  Log  ${height}
  Sleep  4s
 Switch Window  @{WindowHandles}[1]
 Log  @{WindowHandles}[1]
  Sleep  4s
 Close Window
 Sleep  4s
 Switch Window  @{WindowHandles}[0]
 Close Window
```

18.Data Driven Testing using script

Data Driven Testing means we can execute our test case multiple times with different sets of data.we can specify the data in the same **script, excel, csv format**, etc.Data source can be anything.so ultimate goal is we have to read the data from different sources and we can send data to that application and we can perform data driven testing.

Under Resources directory create login_resources robot file

login_resources.robot

```
*** Settings ***
Library     SeleniumLibrary
*** Variables ***
${LOGIN URL}    https://admin-demo.nopcommerce.com
${BROWSER}      chrome
*** Keywords ***
Open my Browser
   open browser    ${LOGIN URL}    ${BROWSER}
   maximize browser window
Close Browsers
   close all browsers
Open Login Page
   go to  ${LOGIN URL}
Input username
   [Arguments]     ${username}
   input text  id:Email    ${username}
Input pwd
   [Arguments]  ${password}
   input text  id:Password     ${password}
click login button
   click element  xpath://input[@class='button-1 login-button']
click logout link
   click link  Logout
Error message should be visible
   page should contain  Login was unsuccessful
Dashboard page should be visible
   page should contain  Dashboard
```

Note

Suite Setup is executed before the actual test case is started.**Suite Teardown** is executed after completion of all the test cases.**T**hese are executed only once.Because every test case will use multiple sets of data multiple times.**Test Template** is used for DDT.It is about whatever the test steps we are going to repeat multiple times with different combinationations of data.we have to specify that inside the settings along with keyword.User defined keywords are defined in keywords section.

Under TestSuite Folder create **DDT1.robot**

DDT1.robot

```
*** Settings ***
Library     SeleniumLibrary
Resource  ../Resources/login_resources.robot
Suite Setup  Open my Browser
Suite Teardown  Close Browsers
Test Template  Invalid login
*** Test Cases ***       username                    password
Right user empty pass    admin@yourstore.com        ${EMPTY}
Right user wrong pass    admin@yourstore.com        xyz
Wrong user right pass    adm@yourstore.com          admin
Wrong user empty pass    adm@yourstore.com          ${EMPTY}
Wrong user wrong pass    adm@yourstore.com          xyz
*** Keywords ***
Invalid login
  [Arguments]  ${username}    ${password}
  Input username  ${username}
  Input pwd  ${password}
  click login button
  Error message should be visible
```

19.Data driven testing using excel and csv

If we want to read the data in excel or csv file we need additional library Robot framework Datadriver Library installed in plugins in pycharm.

https://pypi.org/project/robotframework-datadriver/

Use login_resources robot file in Resources Directory used in **Data Driven Testing using script**

Create **LoginData.xlsx** & **LoginData.csv** under **TestData** folder for data driven testing

${username}	${password}
admin@yourstore.com	adm
adm@yourstore.com	admin
adm123@yourstore.com	adm

${username};${password};
admin@yourstore.com;adm;
adm@yourstore.com;admin;
adm123@yourstore.com;adm;

Under TestSuite Folder create **DDT2_Excel.robot** & **DDT2_CSV.robot**

DDT2_Excel.robot

```
*** Settings ***
Library      SeleniumLibrary
Resource     ../Resources/login_resources.robot
Library      DataDriver  ../TestData/LoginData.xlsx
sheet_name=Sheet1
Suite Setup  Open my Browser
Suite Teardown  Close Browsers
Test Template  Invalid login
*** Test Cases ***
LoginTestwithExcel using ${username} and
${password}
*** Keywords ***
Invalid login
  [Arguments]  ${username}    ${password}
  Input username  ${username}
  Input pwd  ${password}
  click login button
  Error message should be visible
```

DDT2_CSV.robot

```
*** Settings ***
Library      SeleniumLibrary
Resource     ../Resources/login_resources.robot
Library      DataDriver  ../TestData/LoginData.csv
Suite Setup  Open my Browser
Suite Teardown  Close Browsers
Test Template  Invalid login
*** Test Cases ***
LoginTestwithCsv using ${username} and
${password}
*** Keywords ***
Invalid login
  [Arguments]  ${username}    ${password}
  Input username  ${username}
  Input pwd  ${password}
  click login button
  Error message should be visible
```

Note

- If particular steps are required to be repeated for every input or different sets of data we use **Test Template**
- After execution we will get a warning because of the same data Just ignore or otherwise change data and run.
- Robot Framework will identify values only with a semicolon(;) delimiter

20.Database Testing

Normally in database Testing we will validate whether the data is inserted properly or not,Table is created properly or not, data is updating or not, data is deleting or not, number of rows are there or not.

To perform database testing in a robot framework we need to install 2 Packages or Libraries in the project interpreter.

- **robotframework-databaselibrary** & **PyMySQL**

It is again dependent on the database we are choosing

- suppose for mysql - pymysql Library, sqlserver - pymssql, oracle - pyoracle

To see the keywords for database testing in robot framework go to official website

https://franz-see.github.io/Robotframework-Database-Library/api/0.5/DatabaseLibrary.html

Under TestData folder create **mydb_person_insertData.sql** for Multiple records and in TestSuite Folder create **DBTesting.robot**

mydb_person_insertData.sql

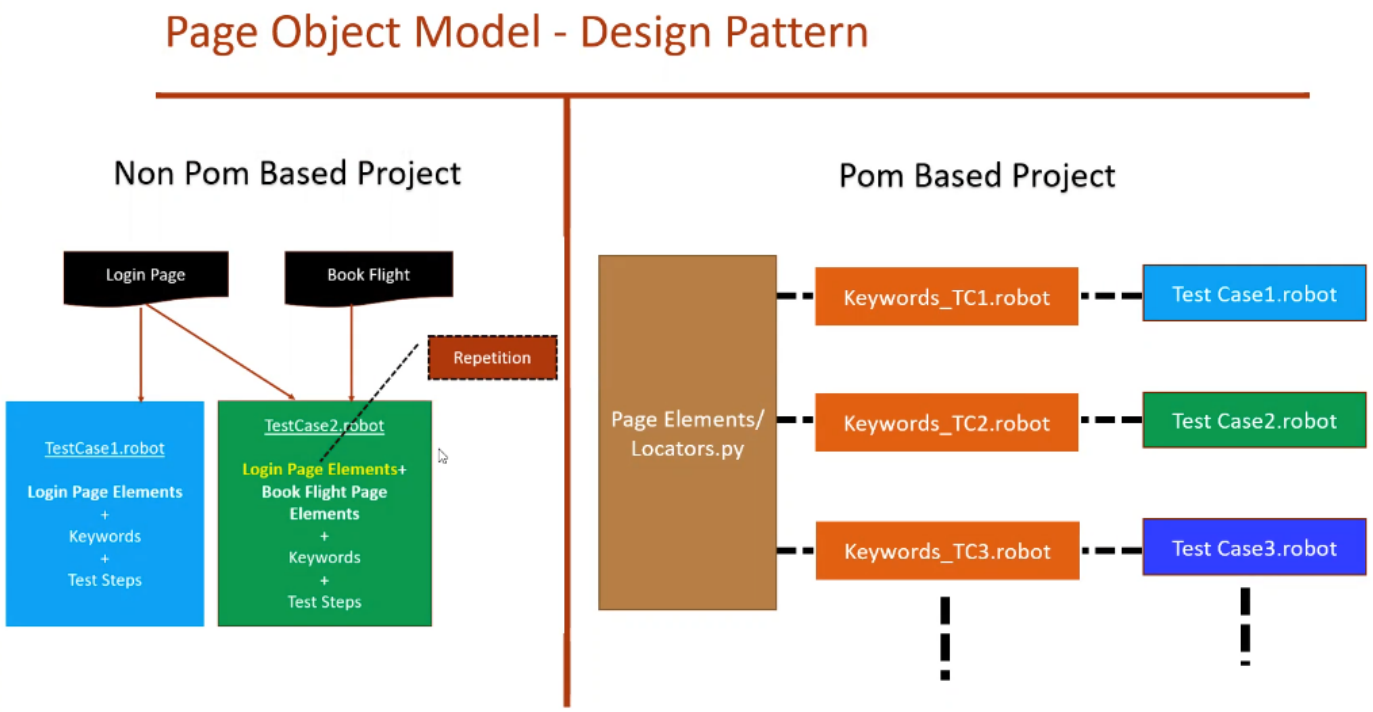
- INSERT INTO mydb.person values(101,"John","Canady");
- INSERT INTO mydb.person values(102,"David","Canady");
- INSERT INTO mydb.person values(103,"Smith","Canady");
- INSERT INTO mydb.person values(104,"Marry","Canady");
- INSERT INTO mydb.person values(105,"Tye","Canady");

DBTesting.robot

```
*** Settings ***
Library     DatabaseLibrary
Library     OperatingSystem
Suite Setup     Connect To Database  pymysql  ${DBName}  ${DBUser}  ${DBPass}  ${DBHost}  ${DBPort}
Suite Teardown    Disconnect From Database
*** Variables ***
${DBHost}       127.0.0.1
${DBName}       mydb
${DBPass}       root
${DBPort}       3306
${DBUser}       root
*** Test Cases ***
Create person table
  ${output}=   Execute SQL String  Create table person(id integer,first_name varchar(20),last_name varchar(20));
  log to console  ${output}
  should be equal as strings  ${output}  None
Inserting Data in person Table
#    Single Record
#    ${output}=   Execute SQL String  Insert into person values(101,"John","canady");
```

```
  Multiple records
  ${output}=  Execute SQL Script  ./TestData/mydb_person_insertData.sql
  log to console  ${output}
  should be equal as strings  ${output}  None
Check David record present in Person Table
  check if exists in database  select id from mydb.person where first_name="David";
Check Jio record Not present in Person Table
  check if not exists in database  select id from mydb.person where first_name="Jio";
Check Person Table exists in mydb database
  table must exist  person
Verify Row Count is Zero
  row count is 0  SELECT * FROM mydb.person WHERE first_name = 'xyz';
Verify Row Count is Equal to Some Value
 row count is equal to x  SELECT * FROM mydb.person WHERE first_name = 'David';  1
Verify Row Count is Greater than Some Value
  row count is greater than x  SELECT * FROM mydb.person WHERE first_name = 'David';  0
Verify Row Count is less than Some Value
  row count is less than x  SELECT * FROM mydb.person WHERE first_name = 'David';  5
Update record in person table
   ${output}=  Execute SQL String  Update mydb.person set first_name="Jio" where id=104;
   log to console  ${output}
   should be equal as strings  ${output}  None
Retrieve Records from Person Table
    @{queryResults}=  query  Select * from mydb.person;
    log to console  many @{queryResults}
Delete Records from person table
   ${output}=  Execute SQL String  Delete from mydb.person;
  should be equal as strings  ${output}  None
```

21.PageObject Model



Page object model is not a framework it is a pattern also called Page object repository or Page object factory.This pattern is talking about how we can manage/maintain the page objects/elements for multiple Test Cases.

Suppose we have multiple test cases in our project, for every test case we have to identify certain number of elements on the webpage and those elements along with the test methods we will include everything in the python test.Instead of including everything in one file we will just separate those page elements in another file .This is called as a Page Object Model Pattern.

We will maintain test cases and page elements in two different classes .

Non POM Based Project

Let say 3 different pages (Login, search, advanced search) in application.similarly we have 3 test cases (Login test case, search test case, advanced search test case) along with page elements

Problems in Normal Approach :

1. Same elements we are identifying in every test case (Duplication of Elements)
2. Suppose automated 10 Test Cases all 10 TC used Login Page element and suppose if tomorrow if the page element got modified or some attribute got changed then i need to go and modify those element in every Test Case (Updating Elements)

Page object model pattern Approach

It is not a framework it is a design pattern.In this pattern we will maintain the page elements in separate file i.e,we will maintain different page elements in different files or different page elements in a single file.For every Test case different keywords are required so keywords are defined in a separate files .Now in the Test Case we will define only Test steps .suppose when we create on Test case which contain only the test steps and those test steps will use keywords from keyword file and keywords page elements in separate file

The advantage is no repetition, Re-usability and Updation

We will create PageObjects dir, Resources dir, TestCases python packages

In **PageObjects** we will have Locators.py.Under **Resources** we will have LoginKeywords.robot and RegistrationKeywords.robot.Under **TestCases** we will have LoginTest and Registration Test

Locators.py

```
# Login Page Elements
txt_loginUserName="name:userName"
txt_password="name:password"
btn_signIn="name:login"
#Registration Page Elements
link_Reg="link:REGISTER"
txt_firstName="name:firstName"
txt_lastName="name:lastName"
txt_phone="name:phone"
txt_email="name:userName"
txt_add1="name:address1"
txt_add2="name:address2"
txt_city="name:city"
txt_state="name:state"
txt_postCode="name:postalCode"
drp_country="name:country"
txt_userName="name:email"
txt_Password="name:password"
txt_conformedPassword="name:confirmPassword"
btn_submit="xpath://input[@name='register']"
```

Locators.py is a python file so we used variables but if it is a robot file we will use Resource

LoginKeywords.robot

```
*** Settings ***
Library     SeleniumLibrary
Variables   ../PageObjects/Locators.py
*** Keywords ***
Open my Browser
  [Arguments]  ${SiteUrl}   ${Browser}
  open browser   ${SiteUrl}   ${Browser}
  Maximize Browser Window
Enter UserName
  [Arguments]    ${username}
       Input   Text          ${txt_loginUserName}
${username}
Enter Password
  [Arguments]    ${password}
  Input Text   ${txt_password}   ${password}
```

RegistrationKeywords.robot

```
*** Settings ***
Library     SeleniumLibrary
Variables   ../PageObjects/Locators.py
*** Keywords ***
Open my Browser
  [Arguments]  ${SiteUrl}   ${Browser}
  open browser   ${SiteUrl}   ${Browser}
  Maximize Browser Window
Click Register Link
 click link  ${link_Reg}
Enter FirstName
  [Arguments]    ${firstName}
  Input Text   ${txt_firstName}   ${firstName}
Enter LastName
  [Arguments]    ${lastName}
```

```
Click SignIn
  click button    ${btn_signIn}
Verify Succesful Login
  title should be  Find a Flight: Mercury Tours:
close my browsers
  close all browsers
```

LoginTest.robot

```
*** Settings ***
Library     SeleniumLibrary
Resource    ../Resources/LoginKeywords.robot
*** Variables ***
${Browser}      chrome
${SiteUrl}      http://newtours.demoaut.com/
${user}      tutorial
${pwd}      tutorial
*** Test Cases ***
LoginTest
  Open my Browser      ${SiteUrl}    ${Browser}
  Enter UserName     ${user}
  Enter Password  ${pwd}
  Click SignIn
  sleep  3 seconds
  Verify Succesful Login
  close my browsers
```

RegistrationTest.robot

```
*** Settings ***
Library     SeleniumLibrary
Resource
../Resources/RegistrationKeywords.robot
*** Variables ***
${Browser}      chrome
${SiteUrl}      http://newtours.demoaut.com
*** Test Cases ***
RegistrationTest
  Open my Browser      ${SiteUrl}    ${Browser}
  Click Register Link
  Enter FirstName     David
  Enter LastName      John
  Enter Phone         1234567890
  Enter Email         davidJohn@gmail.com
  Input Text    ${txt_lastName}    ${lastName}
Enter Phone
  [Arguments]    ${phone}
  Input Text    ${txt_phone}    ${phone}
Enter Email
  [Arguments]    ${email}
  Input Text    ${txt_email}    ${email}
Enter Address1
  [Arguments]    ${add1}
  Input Text    ${txt_add1}    ${add1}
Enter Address2
  [Arguments]    ${add2}
  Input Text    ${txt_add2}    ${add2}
Enter City
  [Arguments]    ${city}
  Input Text    ${txt_city}    ${city}
Enter State
  [Arguments]    ${state}
  Input Text    ${txt_state}    ${state}
Enter Postal Code
  [Arguments]    ${postalcode}
  Input Text    ${txt_postCode}    ${postalcode}
Select Country
  [Arguments]    ${country}
  select from list by label  ${drp_country}
${country}
Enter User Name
  [Arguments]    ${username}
  Input Text    ${txt_userName}    ${username}
Enter Password
  [Arguments]    ${password}
  Input Text    ${txt_Password}    ${password}
Enter Confirmed Password
  [Arguments]    ${confpassword}
  Input Text    ${txt_conformedPassword}
${confpassword}
Click Submit
  click button    ${btn_submit}
Verify Succesful Registration
  page should contain  Thank you for registering.
```

Enter Address1 *Toronto* Enter Address2 *Canada* Enter City *Toronto* Enter State *Brampton* Enter Postal Code *L3S 1E7* Select Country *CANADA* Enter User Name *johnxyz* Enter Password *johnxyz* Enter Confirmed Password *johnxyz* Click Submit Verify Succesful Registration close my browsers	close my browsers close all browsers

22.Parallel Execution

Executing Test Suites in Robot Framework

- How to Run Test Suites
- How to Run Tests Parallely using Robot framework - pabot
- How to save Results in Folder
- How to Run Tests using Single Windows Bat File

How to Run Test Suites - Sequential Execution

Approach 1 - Specify Folder

cd C:\Users\admin\PycharmProjects\Automation>robot TestCases\

Approach 2 - Using Regular Expression

cd C:\Users\admin\PycharmProjects\Automation\robot TestCases*.robot **- All Test Cases are Executed**

➔ C:\Users\admin\PycharmProjects\Automation\robot TestCases\Reg*.robot **- only Registration Test is executed**

How to Run Tests Parallely using Robot framework - pabot

To run outside of PyCharm i.e in CLI we need to install in cmd that is the purpose of installing outside of PyCharm.

- **Pre -Req :** Install Below package in cmd & PyCharm in project interpreter
- **CLI:** pip install -U **robotframework-pabot**

pabot

A parallel executor for Robot Framework tests. Split one execution into multiple and save test execution time.

pabot is implemented on top of robot .

Command to execute TestCases Parallely

cd C:\Users\admin\PycharmProjects\Automation>pabot - -processes 2 TestCases*.robot

How to save Results in Folder

cd C:\Users\admin\PycharmProjects\Automation>pabot - -processes 2 - -outputdir Results TestCases*.robot

2 represents no of TC for executing Parallelly.The Results folder stores executed test results.

How to Run Tests using Single Windows Bat File

We can also run Test Cases using bat file.we are manually writing command by using Pycharm instead of doing this we can just put this command inside the bat file and with a single click our TestCase will execute.

By using bat file we can run outside of Pycharm also,this is the advantage we have

In the project level create run.bat file and write the command

run.bat

cd C:\Users\admin\PycharmProjects\Automation

pabot - -processes 2 - -outputdir Results TestCases*.robot

Right click on the bat file and select **Run cmd script.**it opens cmd and Executes Test cases parallelly.

Later we can run the same bat file using Jenkins and other continuous integration tools

Note

Execute Test Cases on Remote Server on Jenkins

23.Headless Browser Testing in Robot Framework

- Instead of chrome mention headlesschrome for Headless Browser Testing
- Instead of firefox mention headlessfirefox for Headless Browser Testing

Normally when we are executing our TestCases it will launch the browser it will interact with the elements and everything will go like a flow.As a user i can see all the operations whatever is happening on the UI. Headless browsing means we can perform all the operations at the backend i.e, the user cannot see what is happening on the UI.Advantage is Faster Execution.

Taking the Login and Registration Tests from **PageObject Model**

LoginTest.robot

```
*** Settings ***
Library     SeleniumLibrary
Resource  ../Resources/LoginKeywords.robot
*** Variables ***
${Browser}      headlessfirefox
${SiteUrl}      http://newtours.demoaut.com/
${user}      tutorial
${pwd}      tutorial
*** Test Cases ***
LoginTest
  Open my Browser     ${SiteUrl}  ${Browser}
  Enter UserName  ${user}
  Enter Password  ${pwd}
  Click SignIn
  sleep  3 seconds
  Verify Succesfull Login
  close my browsers
```

RegistrationTest.robot

```
*** Settings ***
Library     SeleniumLibrary
Resource
../Resources/RegistrationKeywords.robot
*** Variables ***
${Browser}  headlesschrome
${SiteUrl}  http://www.newtours.demoaut.com
*** Test Cases ***
RegistrationTest
  Open my Browser     ${SiteUrl}  ${Browser}
  Click Register Link
  Enter FirstName     David
  Enter LastName      John
  Enter Phone         1234567890
  Enter Email         davidJohn@gmail.com
  Enter Address1      Toronto
  Enter Address2      Canada
  Enter City          Toronto
  Enter State         Brampton
  Enter Postal Code   L3S 1E7
  Select Country      CANADA
  Enter User Name     johnxyz
  Enter Password      johnxyz
  Enter Confirmed Password    johnxyz
  Click Submit
  Verify Succesful Registration
  close my browsers
```