



Open2Test Test Automation Framework for SilkTest - Java

Version 1.0

January 2010

DISCLAIMER

Verbatim copying and distribution of this entire article is permitted worldwide, without royalty, in any medium, provided this notice is preserved.

Table of Contents

1. Purpose of the Document.....	3
1.1. Scope	3
1.2. Overview	3
2. Framework Code Structure.....	4
3. Driver Functions.....	5
3.1. Data-Driven Test Case	5
3.2. Main Function	5
4. Action Functions.....	8
4.1. Perform Function.....	8
4.2. Check Function.....	13
4.3. Store Function	16
5. Functions for Setting the Object.....	19
5.1. Function for Setting the Context	19
5.2. Function for Setting the Object	20
6. Reporting Functions	22
6.1. Reporting Functions	22
7. String and Regular Expression	23
7.1. Functions for String Operations	23
7.2. Function for Press Key Operations	24
7.3. Condition Function.....	24
7.4. Function for Arithmetic Operations	25
7.5. Function for Converting Data Types.....	27
7.6. Loop Function	28
8. Common Functions.....	29
8.1. Function for FSO Operations	29
8.2. Function for File Operations.....	29
9. User-Defined Functions.....	33
9.1. Function for CallFunction Keyword	33

1. Purpose of the Document

The purpose of this document is to describe Open Source Test Automation Framework code in detail for Java technology.

1.1. Scope

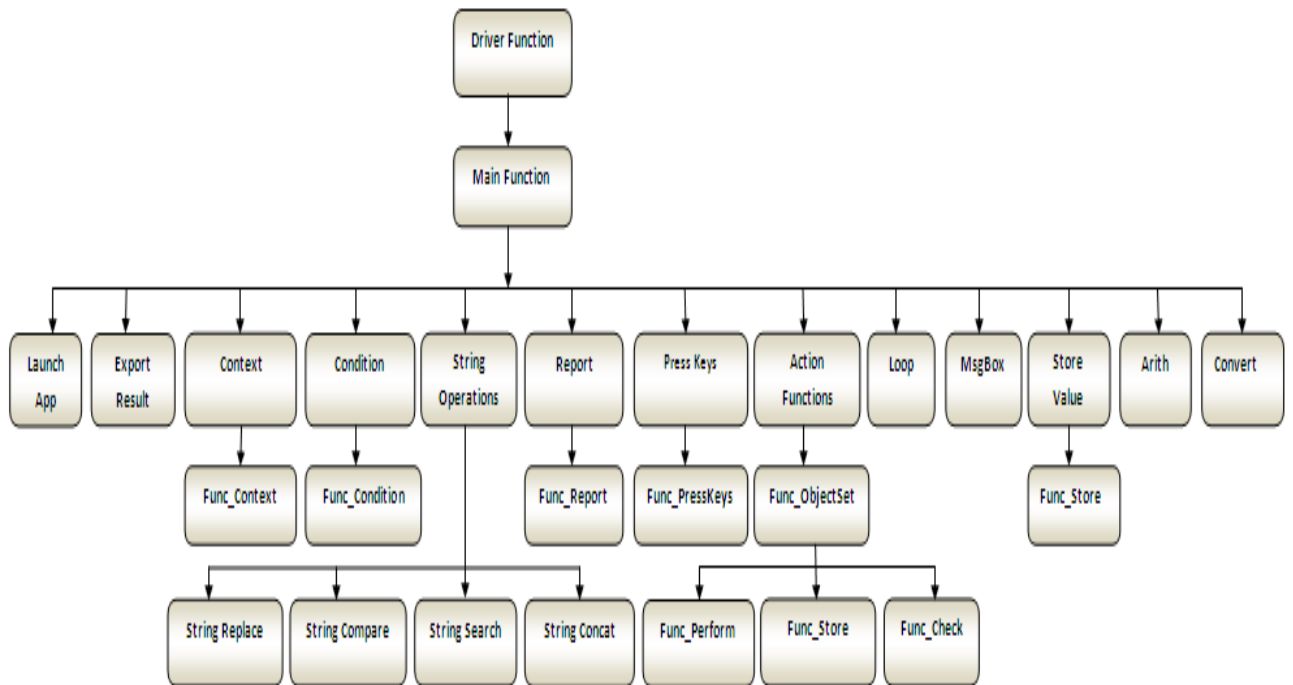
The scope of this document is to provide details about Open Source Test Automation Framework code.

1.2. Overview

This document provides details about:

- Framework Architecture
- Driver Functions
- Action Functions
- Reusable Functions
- Common Functions
- User-Defined Functions

2. Framework Code Structure



3. Driver Functions

3.1. Data-Driven Test Case

Name of the Test case:

DD_<Test Case Name>(REC_DATALIST_DD_<TestCaseName> rData)

Description: This is a data-driven test case, which is used in keyword script for calling Keyword_Driver() function.

Parameters:

a) rData - This is the record name of type REC_DATALIST_DD_<Test Case Name>, which contains a list of all column names.

Assumptions: The Automation Script is present in the external sheet, which is linked as shown below:

```
STRING gsDSNConnect= "DSN=<Data Source Name>; DBQ =<Path of the  
keyword script>; UID=; PWD=;"
```

Where DSN is Data Source Name, generally it will be 'Silk DDA Excel'.

Variables:

- a) **intj** - This variable is used to hold the row count of the keyword script.
- b) **aCellData** - This variable is used to hold the content of the first column of the keyword script.
- c) **bCellData** - This variable is used to hold the content of the second column of the keyword script.
- d) **cCellData** - This variable is used to hold the content of the third column of the keyword script.
- e) **dCellData** - This variable is used to hold the content of the fourth column of the keyword script.
- f) **eCellData** - This variable is used to hold the content of the fifth column of the keyword script.

Functionality:

- The data-driven test case reads the values present in the first, second, third, fourth, and fifth columns of the keyword script and calls the Keyword_Driver() function.

3.2. Main Function

Name of the function: Keyword_Driver (ARRAY[50] OF Anytype ColumnValOne, ARRAY[50] OF Anytype ColumnValTwo, ARRAY[50] OF Anytype ColumnValThree, ARRAY[50] OF Anytype ColumnValFour, ARRAY[50] OF Anytype ColumnValFive).

Description: This is the Main Function, which interprets the keywords and performs the desired actions. All the keywords used in the keyword-driven script file are processed in this function. This function is called inside the data-driven script file (.g.t file).

Parameters:

- a) ColumnValOne - This variable is used to hold the content of the first column of the keyword script.
- b) ColumnValTwo - This variable is used to hold the content of the second column of the keyword script.
- c) ColumnValThree - This variable is used to hold the content of the third column of the keyword script.
- d) ColumnValFour - This variable is used to hold the content of the fourth column of the keyword script.
- e) ColumnValFive - This variable is used to hold the content of the fifth column of the keyword script.

Assumptions: The Automation Script is present in the external sheet, which is linked as shown below.

STRING gsDSNConnect= "DSN=<Data Base Name>;DBQ=<Path of the external sheet>; UID=; PWD=;"

Variables:

- a) Automate - This array is used to store the value in the first column of the keyword script.
- b) Action - This array is used to store the value in the second column of the keyword script.
- c) Object - This array is used to store the value in the third column of the keyword script.
- d) ActionValueOne - This array is used to store the value in the fourth column of the keyword script.
- e) ActionValueTwo - This array is used to store the value in the fifth column of the keyword script.
- f) bCellData - This variable is used to store the action value in the second column after trimming the data.
- g) Flag - This variable is used in loop and condition.
- h) LoopFlag - This variable is used in looping.

Functionality:

- The Main Function reads the values in the first, second, third, fourth, and fifth columns in the script file and stores the values into the variables.
 - Whenever the value in a variable 'Automate' is 'r', the function performs the required action on application under test (AUT).
 - Whenever the value in variable 'Automate' is not present, the function will skip the current row and read the values in the next row.
 - If the keyword script sheet1 is empty, then it reports a failure, stating "Query did not return any rows: select * from "Sheet1\$" which has occurred in datadrivetc.inc file.
- Note:** Keyword script is stored in Sheet1.
- Based on the value in the variable bCellData (second column), the Main Function calls different functions.
 - If the value is other than 'perform', 'storevalue', or 'check',

the Main Function calls the respective functions. For example: If the value is 'condition', it will call Func_Condition().

- If the value is 'perform', 'storevalue', or 'check', the Main Function calls the function Func_ObjectSet() to set the object and then it calls the respective functions. For example: If the value is 'perform', the Main Function will call Func_Perform() to perform required actions on the AUT.

Note: Refer to the framework code structure diagram for a better understanding of the function calls.

4. Action Functions

4.1. Perform Function

Name of the function: Func_Perform()

Description: This function performs the set of actions on the required object in the AUT.

Parameters:

- a) Object - This parameter holds the type of object and object tag name on which the action has to be performed.
- b) ActionValueOne - This parameter is used to hold the value of the fourth column of the keyword script file, which specifies the action to be performed on the specified object.
- c) ActionValueTwo - This parameter is used to hold the path of the data sheet for type_dt and select_dt operations. This parameter holds Driver Name, Database Name, Server Name, User ID, and Password for SQL operations.
- d) objParPage - This parameter is used to hold the parent page name. This parameter will be null for common and user-defined functions, 'split' and 'SQL' operations.

Note: ObjParPage has been set in Func_Context function or Func_ObjectSet function.

Assumptions: Context is set on the current object where action has to be performed.

Variables:

- a) curObject - This variable is used to store the type of object.
- b) curObjTagName - This variable is used to store the object's tag name. It is also used to store the query during SQL operations.
- c) curObjPerform - This variable is used to store the fourth column content of the keyword script.
- d) dCellData - This variable is used to store the operation that needs to be performed on the object.
- e) dCellValue - This variable is used to store the value that is used during the perform operation.
- f) SQLReturnValue - This variable is used to store the value that is returned after the SQL operations.

Functionality:

Based on the values in curObject, the Func_Perform() function performs different actions on objects. If the value is:

1. Window

Based on the values in dCellData, the function performs different operations on objects. If the value in the variable dCellData is:

- a. close: Performs the close operation on the parent object.
- b. maximize: Maximizes the parent object.
- c. minimize: Minimizes the parent object.
- d. activate: Activates the parent object.
- e. restore: Restores the parent object to its previous size.

2. Dialog

Based on the values in `dCellData`, the function performs different operations on objects. If the value in the variable `dCellData` is:

- a. `close`: Performs the close operation on the parent object.
- b. `activate`: Activates the parent object.

3. Button:

Based on the values in `dCellData`, the function performs different operations on objects. If the value in the variable `dCellData` is:

- a. `click`: Performs the click operation on the current object.
- b. `agentclick`: If the button is not exposed, then we need to set the agent option 'OPT_ENABLED_EXPOSED' to false and perform the click operation on the button.
- c. `setfocus`: Sets the focus on the current object.

4. Toolbar:

Based on the values in `dCellData`, the function performs different operations on objects. If the value in the variable `dCellData` is:

- a. `press`: Performs a 'press' operation on the toolbar button specified.

5. Treeview:

Based on the values in `dCellData`, the function performs different operations on objects. If the value in the variable `dCellData` is:

- a. `select`: Selects the node present in the variable `dCellValue` using `select` method.
- b. `expand`: Expands the node present in the variable `dCellValue` using `expand` method.
- c. `collapse`: Collapses the node present in the variable `dCellValue` using `collapse` method.
- d. `collapseindex`: Collapses the node using the specified in `dCellValue` using `collapse` method.
- e. `releaseitem`: Releases the mouse over an item present in the variable `dCellValue` using `ReleaseItem` method.
- f. `releaseitemindex`: Releases the mouse over an index present in the variable `dCellValue` using `ReleaseItem` method.

6. Menu:

Based on the values in `dCellData`, the function performs different operations on objects. If the value in the variable `dCellData` is:

- a. `pick`: Picks the menu item from the menu. The menu item tag is stored in variable `dCellValue`.

7. popupmenu:

Based on the values in `dCellData`, the function performs different operations on objects. If the value in the variable `dCellData` is:

- a. `pick`: Picks the menu item from the popup menu whose `menuitem` tag is stored in variable `dCellValue`.
- b. `pickindex`: Picks the menu item using the index specified in `dCellValue` from the popup menu using the `pick` method.
- c. `select`: Selects the item present in the variable `dCellValue` from the current object using the `select` method.
- d. `setfocus`: Sets the focus in the popup menu using `SetFocus` method.

8. submenu:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. pick: Picks the menu item from the submenu whose menuitem tag is stored in variable dCellValue.

9. combobox:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. select: Selects the item present in the variable dCellValue from the current object using the select method.
- b. selectindex: Selects the item using the index specified in the variable dCellValue from the current object using the select method.
- c. set: Sets the item present in the variable dCellValue from the current object using the SetText method.
- d. select_dt: Selects an item present in the external data sheet in current object where the data sheet path is specified in the ActionValueTwo column of the keyword script.
- e. setfocus: Sets the focus in the combobox using SetFocus method.

10. tablecombobox:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. select: Selects the item present in the variable dCellValue from the current object using the select method.
- b. selectindex: Selects the index present in the variable dCellValue from the current object using the select method.

11. checkbox:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. check: Performs check operation on the current object.
- b. uncheck: Performs uncheck operation on the current object.
- c. setfocus: Sets the focus in the checkbox using SetFocus method.
- d. toggle: Performs check operation on the checkbox if it is unchecked, and performs uncheck operation if it is checked.

12. listbox:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. select: Selects the item present in the variable dCellValue from the current object using the select method.
- b. extendselect : Selects the additional items present in the variable dCellValue from the current object.
- c. click: Clicks on the specified listbox.
- d. doubleselect: Double click on the item present in the variable dCellValue from the current object.
- e. setfocus: Sets the focus in the listbox using SetFocus method.
- f. multiselect: Selects an item present in the variable dCellValue in the listbox. The listbox must already have an item selected.

13. radiobutton:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. select: Performs the select operation on the current object.
- b. setfocus: Sets the focus on the radio button using SetFocus method.

14. Scrollbar:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. scrollmax: Scrolls the scroll bar to its maximum position.
- b. scrollmin: Scrolls the scroll bar to its minimum position.
- c. setposition: Sets the position of the scroll bar to the specified position.
- d. setfocus: Sets the focus on the scroll bar using SetFocus method.

15. textbox:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. type: Types the value present in the variable dCellValue in the current object.
- b. type_dt: Types the value present in the external data sheet in the current object, where the data sheet path is specified in the ActionValueTwo column of the keyword script.
- c. set: Sets the value present in the variable dCellValue in the current object. If the dCellValue value is:
 - i. d_currenttime: Sets the current time in the current object.
 - ii. d_currentdate: Sets the current date in the current object.
 - iii. d_d: Adds or subtracts the specified number of days in the variable dCellValue3 to the current date and sets the final date obtained in the current object.
 - iv. d_m: Adds or subtracts the specified number of months in the variable dCellValue3 to the current date and sets the final date obtained in the current object.
 - v. d_y: Adds or subtracts the specified number of years in the variable dCellValue3 to the current date and sets the final date obtained in the current object.
- d. set_dt: Sets the value present in the external data sheet in the current object where the data sheet path is specified in the ActionValueTwo column of the keyword script.
- e. clear: Clears the text field.
- f. click: Performs the click operation on the current object.
- g. setfocus: Sets the focus on the text field using SetFocus method.

16. table:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

- a. select: Selects the row number present in the variable dCellValue from the current object.
- b. doubleclick: Double clicks on the specified cell value in the table using ClickCell method.

c. multiclick: Clicks on the specified cell in the table so that the multiple cells are selected in between the already selected cells and specified cell value in keyword script using the method ExtendClickCell.

Note: It is like holding the ctrl key and clicking on specified cell value.

d. setfocus: Sets the focus on the table using SetFocus method.

17. pagelist:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

a. select: Performs the select operation on the current object.

b. setfocus: Sets the focus on the pagelist using SetFocus method.

18. togglebutton:

Based on the values in dCellData, the function performs different actions on objects. If the value in the variable dCellData is:

a. click: Performs the click operation in the current object.

b. doubleclick: Performs the double click operation in the current object.

c. setstate: Performs the click operation on the toggle button depending on the value present in the variable dCellData2. If the dCellData2 value is:

i. true: Performs the click operation on the toggle button, if the toggle button is not already pressed. Otherwise, nothing is performed.

ii. false: Performs the click operation on the toggle button, if the toggle button is already pressed. Otherwise, nothing is performed.

d. setfocus: Sets the focus on the togglebutton using SetFocus method.

e. toggle: Performs the click operation on the toggle button, making it pressed, if it is not pressed. Likewise, if the toggle button is pressed, this action makes it not pressed.

Note: For all the above keywords, Func_Perform performs the actions on the objects using the variables specified by Var:<Variable Name> in the keyword scripts. Variable Name and its value are stored in an array 'sRunVar'.

If the value of curObject is not among the above listed keywords, then the function will check for the object, which holds the value in the third column. If the value of curObject is:

a. split: Performs the split operation using GetField method and returns the required string.

b. sqlvaluecapture:

- Silk Test Opens the connection to a data base using DB_Connect function, which makes use of 'Driver Name', 'Server Name', and 'Data Base Name', which are specified in the ActionValue2 (fifth column) of the keyword script.

- Executes the query stored in `curObjTagName` using `DB_ExecuteSql` function.
 - Fetches the required value from the data base and stores it in the variable specified in the `ActionValue1` (fourth column) of the keyword script.
 - Removes the results of the SQL statement and releases the associated system resources using the `DB_FinishSql` method.
 - Disconnects SilkTest from the database system and releases all resources using `DB_DisConnect` method.
- c. `sqlexecute`:
- Silk Test Opens the connection to the data base using `DB_Connect` function. This makes use of 'Driver Name', 'Server Name', and 'Data Base Name', which are specified in the `ActionValue2` (fifth column) of the keyword script.
 - Executes the query stored in `curObjTagName` using `DB_ExecuteSql` function.
 - Removes the results of the SQL statement and releases the associated system resources using the `DB_FinishSql` method.
 - Disconnects SilkTest from the database system and releases all resources using `DB_DisConnect` method.
- d. `sqlmultiplecapture`:
- Silk Test Opens the connection to a data base using `DB_Connect` function. This function makes use of 'Driver Name', 'Server Name', and 'Data Base Name', which are specified in the `ActionValue2` (fifth column) of the keyword script.
 - Executes the query stored in `curObjTagName` using `DB_ExecuteSql` function.
 - Fetches the required multiple values from the data base and stores them in the variables specified in the `ActionValue1` (fourth column) of the keyword script.
 - Removes the results of the SQL statement and releases the associated system resources using the `DB_FinishSql` method.
 - Disconnects SilkTest from the database system and releases all resources using `DB_DisConnect` method.

4.2. Check Function

Name of the function: `Func_Check()`

Description: This function is used for all the check operations to be performed on the AUT.

Parameters:

- a) Object - This parameter holds the type of object and object tag name on which the action has to be performed.
- b) ActionValueOne - This parameter is used to hold the property of the object (fourth column of the keyword script).
- c) ActionValueTwo - This parameter holds Driver Name, Database Name, Server Name, User ID, and Password for SQL operations.
- d) objParPage - This parameter is used to hold the parent page name. This parameter will be null for SQL operations.

Assumptions: Context is set on the current object where the action has to be performed

Variables:

- a) checkObjClass - This parameter holds the object on which the specified operation needs to be performed.
- b) checkVal - This parameter holds the check or checks that need to be performed on the object.
- c) dcheckData - This parameter is used to hold the property values after splitting the contents of the variable ActionValueOne with delimiter ':'.
- d) ActualValue - Stores the runtime property value retrieved from the Application Under Test and holds the string conversion of boolean values 'true' or 'false' of the variable PropertyVal.
- e) SQLValue: Stores the value obtained after SQL operations.

Functionality:

Based on the values in checkVal, the function performs different check operations. If the checkVal value is:

- a. enabled: The value of the enabled property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyVal.
- b. exist: The value of the existence property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyVal.
- c. label: The label of the object obtained from AUT is stored in PropertyVal, which is then evaluated with dcheckData and an appropriate report is generated.
- d. select: The value of the GetSelText property of the object is stored in PropertyVal, which is then evaluated with dcheckData and an appropriate report is generated.
- e. selectindex: The value of the GetSelIndex property of the object is stored in PropertyVal, which is then evaluated with dcheckData and an appropriate report is generated.
- f. state: The value of the GetState property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyVal.

- g. text: The value of the GetText property of the object is stored in PropertyValue, which is then evaluated with dcheckData and an appropriate report is generated.
- h. expand: The value of the IsItemExpandable property of the object is stored in ActualValue, which is then evaluated with dCellValue and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.

Note: Path of the tree item is stored in the variable dcheckData.

- i. verifytext: The value of the GetCellValue property of the object is stored in PropertyValue, which is then evaluated with dcheckData and an appropriate report is generated.
- j. extendselect: The value of the IsExtendSel property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.
- k. multitext: The value of the IsMultiText property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.
- l. richtext: The value of the IsRichText property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.
- m. multiselect: The value of the IsMultiSel property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.
- n. blank: The value of the MatchStr property between the text obtained from the object and null value is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.
- o. focused: The value of the HasFocus property of the object is stored in ActualValue, which is then evaluated with dcheckData and an appropriate report is generated. ActualValue is the string conversion of boolean values 'true' or 'false' of the variable PropertyValue.
- p. rowcount: The value of the GetRowCount property of the object is stored in PropertyValue, which is then evaluated with the integer value of dcheckData and an appropriate report is generated.
- q. colcount: The value of the GetColumnCount property of the object is stored in PropertyValue, which is then evaluated with the integer value of dcheckData and an appropriate report is generated.
- r. itemcount: The value of the GetItemCount property of the object is stored in PropertyValue, which is then evaluated with the integer value of dcheckData and an appropriate report is generated.

- s. **itemexist**: The value of the GetContents property of the object is stored in PropertyVal, which checks whether the value of dcheckData is present or not in the list of values stored in the PropertyVal using ListFind property. An appropriate report is then generated.
- t. **sqlcheckpoint**: Captures value from the database by executing the specified query and stores it in a variable name SQLValue. This value is then evaluated with the value in the variable dcheckData and an appropriate report is generated.

4.3. Store Function

Name of the function: Func_Store()

Description: This function is used to store any property value of a particular object in a variable.

Parameters:

- a) Object - This parameter holds the type of object and object tag name on which the action has to be performed.
- b) ActionValueOne - This parameter is used to store the property value of an object (fourth column of the keyword script).
- c) objParPage - This parameter is used to hold the parent page name. This parameter will be null for random strings or random number generation operations.

Assumptions: Context is set on the current object where the operation has to be performed.

Variables:

- a) curObject - This is the object on which the specified operation needs to be performed.
- b) curObjTagName - This parameter is used to hold the tag of the current object.
- c) storeVar - This variable is used to hold the property value of an object.
- d) storeVal - This variable is used to hold the variable name in which the property of an object is stored.

Note: Variable Name in 'storeVal' is obtained by splitting ActionValueOne contents with the delimiter ':'.

Functionality:

Based on the values in storeVar variable, the function performs different actions. If the value storeVar is:

- i. **itemscout**: "GetItemcount" property value of the object is stored in the variable storeVal specified in the fourth column of the keyword script.
- ii. **rowcount**: "GetRowCount" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.

-
- iii. colcount: "GetColumnCount" property of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - iv. enabled: "IsEnabled" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - v. exist: "Exists" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - vi. text: "GetText" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - vii. getstate: "GetState" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - viii. getitem: "GetItemText" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - ix. selecteditem: "GetSelText" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - x. getindex: "GetSelIndex" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xi. itemexist: The value of the GetContents property of the object is stored in PropertyVal. It finds the value of dcheckData present or not in the list of values stored in the PropertyVal using ListFind property. Strings of boolean values are stored in the variable storeVal specified in the fourth column in the keyword script.
 - xii. getcell: "GetCellvalue" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xiii. getsubitem: "GetSubItems" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xiv. expand: "IsItemExpandable" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xv. expand: "IsItemExpandable" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xvi. getcount: "GetPageCount" property value of the Pagelist object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xvii. pagename: "GetPageName" property value of the Pagelist object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xviii. caption: "GetCaption" property value of the object is stored in the variable storeVal specified in the fourth column in the keyword script.
 - xix. getposition: "GetPosition" property value of the object, as well as string format of Boolean value, is stored in the

variable storeVal specified in the fourth column in the keyword script.

- xx. random;num;<no. of digit>: Generates the random numbers with specified number of digits using the method RandStr and stores the number in the variable name specified in the ActionValue1 (fourth column) of the keyword script.
- xxi. random;str;<no. of characters>: Generates the random strings with length equal to specified number of characters using the method RandStr and stores the string in the variable name specified in the ActionValue1 (fourth column) of the keyword script.
- xxii. random;alphanum;<no. of characters>;<no. of digits>: Generates the random strings with length equal to specified number of characters and random numbers with specified number of digits using the method RandStr and stores the string in the variable name specified in the ActionValue1 (fourth column) of the keyword script.

Note: If any of the above properties are not valid for any of the objects, then an error is thrown stating that the property is not supported.

5. Functions for Setting the Object

5.1. Function for Setting the Context

Name of the function: Func_Context()

Description: This function is used to set the focus on the object on which some action needs to be performed.

Parameters:

- a) Object - This parameter holds the type of object and Object Name on which the action has to be performed.
- b) ActionValueOne - This parameter is used to hold the value of the fourth column of the keyword script.

Assumptions: The current object exists in the application under test.

Variables:

- a) curContext: This variable is used store the type of object that needs to be activated.
- b) dContextData: This variable is used store the type of child object that needs to be activated.
- c) objParPage: This variable is used store the parent page name.

Functionality:

Based on the values in curContext, Func_Context() sets the context on objects. If the value of curContext is:

1) window:

Func_Context function sets the context on the window name present in the variable 'curWnd'. Current window is set in variable objParPage, if dCellvalue is null.

2) dialog:

Func_Context function sets the context on the dialog name present in the variable 'curWnd'. Current dialog is set in variable objParPage, if dCellvalue is null.

Based on the values in dContextData, Func_Context() sets the context on child objects inside the main window or dialog. If the value of dContextData is:

a) pagelist:

Func_Context function sets the context on the specified pagelist present inside the window or dialog.

b) customwin:

Func_Context function sets the context on the specified custom window present inside the window or dialog.

c) toolbar:

Func_Context function sets the context on the specified toolbar present inside the window.

5.2. Function for Setting the Object

Name of the function: Func_ObjectSet()

Description: This function is used to set the child object on which some action is to be performed.

Parameters:

- a) Object - This parameter holds the type of object and object tag name on which the action has to be performed.
- b) objParPage - This parameter is used to hold parent page name.

Assumptions: The AUT is already up and running.

Variables:

- a) curObject: This variable is used to store the type of object that needs to be set.
- b) curObjTagName: This variable is used to store the tag of the current object that needs to be set.
- c) curObjTagName2: This variable is used to store the tag of the child object within one more object that needs to be set.
- d) objParPage: This variable is used to store the current object class that is set.
- e) objParPage1: This variable is used to store the current object class temporarily.

Functionality:

Based on the values in curContext, Func_ObjectSet() function sets the current objects class and then stores it in a variable by name objParPage. If the value of curContext is:

- i. textbox: Sets the object with class JavaJFCTextField whose tag is mentioned in curObjTagName as the current object.
- ii. button: Sets the object with class JavaJFCPushButton whose tag is mentioned in curObjTagName as the current object.
- iii. radiobutton: Sets the object with class JavaJFCRadioList whose tag is mentioned in curObjTagName as the current object.
- iv. treeview: Sets the object with class JavaJFCTreeView whose tag is mentioned in curObjTagName as the current object.
- v. menu: Sets the object with class JavaJFCMenu whose tag is mentioned in curObjTagName as the current object.
- vi. combobox: Sets the object with class JavaJFCComboBox whose tag is mentioned in curObjTagName as the current object.
- vii. listbox: Sets the object with class JavaJFCListBox whose tag is mentioned in curObjTagName as the current object.
- viii. checkbox: Sets the object with class JavaJFCCheckBox whose tag is mentioned in curObjTagName as the current object.

- ix. **popupmenu:** Sets the object with class `JavaJFCPopupMenu` whose tag is mentioned in `curObjTagName` as the current object.
- x. **table:** Sets the object with class `JavaJFCTable` whose tag is mentioned in `curObjTagName` as the current object.
- xi. **tablecombobox:** Sets the object with class `JavaJFCTable` whose tag is mentioned in `curObjTagName` as the current object first and then sets the object with class `JavaJFCComboBox` whose tag is mentioned in the `curObjTagName2` as the current object.
- xii. **toolbar:** Sets the object with class `JavaJFCToolBar` whose tag is mentioned in `curObjTagName` as the current object.
- xiii. **submenu:** Sets the object with class `JavaJFCMenu` twice whose tags is mentioned in `curObjTagName` and `curObjTagName2` as the current object.
- xiv. **statictext:** Sets the object with class `JavaJFCStaticText` whose tag is mentioned in `curObjTagName` as the current object.
- xv. **togglebutton:** Sets the object with class `JavaJFCToggleButton` whose tag is mentioned in `curObjTagName` as the current object.
- xvi. **scrollbar:** Sets the object with class `JavaJFCScrollBar` whose tag is mentioned in `curObjTagName` as the current object.

6. Reporting Functions

6.1. Reporting Functions

Name of the function: Func_Report()

Description: This function is used for generating the customized report with specified user inputs through the keyword script.

Parameters: Object - This parameter is used to store the third column contents of the keyword script.

Assumptions: NA

Variables:

- a) strReportType - This variable is used to hold the type of report to generate either pass or failed report.
- b) strExpectedString - This variable is used to hold the expected string.
- c) strExpectedValue - This variable is used to hold the expected value.
- d) strActualString - This variable is used to hold the actual string.
- e) strActualValue - - This variable is used to hold the actual value.

Functionality:

Based on the values in the object, the function will generate different reports. If the value is:

- i. pass: Generates a report with status as passed and with expected message text in the result file.
- ii. fail: Generates a report with status as failed and with expected message text in the result file.

7. String and Regular Expression

7.1. Functions for String Operations

Name of the function: Func_StringOperations()

Description: This function is used for all string operations.

Parameters:

- a) StrOperation - This variable holds the value of the second column content of the keyword script.
- b) Object - This variable holds the value of the third column content of the keyword script.
- c) ActionValueOne - This variable holds the value of the fourth column content of the keyword script.

Assumptions: None

Variables:

- a) strMainString: This variable is used to store the main string, which is obtained from the third column content of the keyword script after splitting with delimiter ';'.
- b) strSubString: This variable is used to store the substring, which is obtained from the third column content of the keyword script after splitting with delimiter ';'.
- c) strReplaceString: This variable is used to store replacement string value, which is obtained from the third column of the keyword script after splitting with delimiter ';'.

Functionality:

Based on the values in StrOperation, the function performs different actions. If the value in variable StrOperation is:

i) strreplace:

- 1. Searches for the substring (strSubString) in the main string (strMainString) and replaces it with strReplaceString.
- 2. Stores the replaced main string in the variable name specified in the fourth column of the keyword script.

ii) strcompare:

- 1. Compares the two strings stored in variables strMainString and strSubString using MatchStr method.
- 2. Stores the string values of Boolean value True or False in the variable name specified in the fourth column of the keyword script.

iii) strsearch:

- 1. Searches for the substring (strSubString) in the main string (strMainString).
- 2. Stores the required message, whether the substring exists or not, in the main string in the variable name specified in the fourth column of the keyword script.

iv) strconcat:

1. Concatenates all the strings present in the third column of the keyword script.
2. Stores the concatenated string in the variable name specified in the fourth column of the keyword script.

7.2. Function for Press Key Operations

Name of the function: Func_presskey()

Description: This function is used to perform press key operations.

Parameters:

- a) Object - This variable is used to hold the value of the third column content in the keyword script; it holds the label of the key to be pressed.

ActionValueOne: This variable is used to hold the value of the fourth column content of the keyword script. Also, this variable specifies on which object the key has to be pressed, either on the dialog box or the main window.

Assumptions: NA

Variables:

- a. curObject: This variable is used to hold the fourth column content of the keyword script.
If the value in the variable curObject is:
 - i. window - Presses the key when window is active.
 - ii. dialog - Presses the key when dialog is active.
- b. curObjTagName - This variable holds the current object name.
- c. curObjPerform - This variable holds key to be pressed on the current object.

Functionality:

Depending on the values stored in the variables curObject, curObjTagName, and curObjPerform, the function Func_PressKeys performs the required action on the current object.

7.3. Condition Function

Name of the function: Func_Condition()

Description: This function is used to evaluate the expression according to the inputs given in the keyword script.

Parameters:

- a) Object - This variable is used to hold the third column contents in the keyword script.
- b) ActionValueOne - This variable is used to hold the third column contents in the keyword script.

Assumptions: NA

Variables:

- a) sExpression1 - This variable is used to store the first element to be evaluated.
- b) sExpression2 - This variable is used to store the second element to be evaluated.
- c) iPos - This variable is used to store the position of the 'Var', which decides whether the variable is present in sExpression1 or sExpression2.
- d) iPosInt: This variable is used to store the position of the 'Int'. This decides whether the integer value is present in sExpression1 or sExpression2.
- e) RowValue - This variable is used to store the start row or end row of the keyword script.

Functionality:

Depending on the value passed in the variable object, which is then split by the delimiter ';'.

If the second item after the split is:

- i. equals - When sExpression1 is equal to sExpression2, then the 'start row' number is assigned; otherwise, the 'end row' number is assigned to the variable RowValue.
- ii. greaterthan - When sExpression1 is greater than sExpression2, then the 'start row' number is assigned; otherwise, the 'end row' number is assigned to the variable RowValue.
- iii. lesserthan - When sExpression1 is less than sExpression2, then the 'start row' number is assigned; otherwise, the 'end row' number is assigned to the variable RowValue.
- iv. greaterorequal - When sExpression1 is greater than or equal to sExpression2, then the 'start row' number is assigned; otherwise, the 'end row' number is assigned to the variable RowValue.
- v. lesserorequal - When sExpression1 is less than or equal to sExpression2, then the 'start row' number is assigned; otherwise, the 'end row' number is assigned to the variable RowValue.
- vi. not - When sExpression1 is not equal to sExpression2, then the 'start row' number is assigned; otherwise, the 'end row' number is assigned to the variable RowValue.

Note: The 'start row' is obtained from the fourth column separated by delimiter ':' (first element) of the keyword script, and 'end row' is obtained from fourth column separated by delimiter ':' (second element) of the keyword script.

7.4. Function for Arithmetic Operations

Name of the function: Func_arith()

Description: This function is used to perform arithmetic operations

Parameters:

- a) Object - This parameter is used to store the arithmetic expression specified in the third column content of the keyword script.
- b) ActionValueOne - This parameter is used to hold the variable name in which the result of the arithmetic operation is stored.

Assumptions: NA

Variables:

- a) strParameterOne - This variable is used to hold the third column content of the keyword script.
- b) ArithType - This variable is used to hold the arithmetic operator to perform the arithmetic operations.
- c) Result - This variable is used to hold the result of the arithmetic operation.
- d) strParameterTwo - This variable used is used to hold the values during arithmetic operations.
- e) strParameterThree - This variable is used to hold the values during arithmetic operations.

Functionality:

- This function will search for operator type in the variable 'strParameterOne'. Depending on the type of operator present in the variable strParameterOne, it assigns the required operator symbol to the variable ArithType. This function performs the required arithmetic operation and the result is then assigned to the variable name present in the fourth column of the keyword script. If the value in the variable ArithType is:

- i. '+': 'strParameterTwo' variable is used to store values present in the variable 'strParameterOne' after splitting with the delimiter '+'. Then it will retrieve the values of the variable name, if they start with 'Var', and store them in the variable 'strParameterThree'. This function performs the addition operation and stores the final result in the variable 'Result'. The content of the variable 'Result' is assigned to the variable name present in the fourth column of the keyword script.

Note: Initially, the 'Result' variable is initialized with zero for the addition arithmetic operation.

- ii. '-': temp variable is used to store the value present in the variable strParameterOne after splitting with the delimiter.

'-'. Then content of the variable temp is assigned to the variable 'Result'. 'strParameterTwo' variable is used to store all the values present after the delimiter '-'. If the variable 'strParameterTwo' contains 'Var', then it retrieves the required values from the corresponding variable name and assign them to the variable name 'strParameterThree'. Using the variables 'Result', 'strParameterTwo', and 'strParameterThree',

the function performs the required subtraction operation and stores the final result in the variable 'Result'. The content of the variable 'Result' is assigned to the variable name present in the fourth column of the keyword script.

iii. '*': 'strParameterTwo' variable is used to store values present in the variable 'strParameterOne' after splitting with the delimiter '*'. Then it will retrieve the values of the variable name, if they start with 'Var', and store them in the variable 'strParameterThree'. Using the variables 'Result', 'strParameterTwo', and 'strParameterThree', the function performs the required multiplication operation and stores the final result in the variable 'Result'. The content of the variable 'Result' is assigned to the variable name present in the fourth column of the keyword script.

Note: Initially, the 'Result' variable is initialized with one for the multiplication arithmetic operation.

iv. '/': 'strParameterTwo' variable is used to store the numerator value, and 'strParameterThree' variable is used to store the denominator value. Using the variables 'strParameterTwo' and 'strParameterThree', the function performs the division arithmetic operation and stores the final result in the variable 'Result'. The content of the variable 'Result' is assigned to the variable name present in the fourth column of the keyword script.

7.5. Function for Converting Data Types

Name of the function: Func_Convert()

Description: This function is used to convert the input value to the specified data type.

Parameters:

- a. Object - This variable is used to store the input values specified in the keyword script.
- b. ActionValueOne - This variable holds the value of the fourth column contents in the keyword script

Assumptions: NA

Variables:

- a) Conversion_type: This variable is used to store the variable to be converted.
- b) Conversion_string: This variable is used to store the converted value.

Functionality:

- This function searches for 'Var' in the variable Conversion_string.
- If 'Var' is present, the Conversion_string variable will be assigned the environment value of RunVar; otherwise, the Conversion_string variable will be assigned the value separated with delimiter ":" in the third column of the keyword script.
- After conversion, it will store the value in the variable name mentioned in the fourth column of the keyword script.

- Based on the values in `Conversion_type`, the function will perform different actions. If the value in `Conversion_type` is:
 - i. `'ucase'`: It will convert the value in `Conversion_string` into uppercase and store it in the variable name in the fourth column of the keyword script.
 - ii. `'lcase'`: It will convert the value in `Conversion_string` into lowercase and store it in the variable name present in the fourth column of the keyword script.
 - iii. `'len'`: It will get the length of the string value in `Conversion_string` and store it in the variable name present in the fourth column of the keyword script.
 - iv. `'trim'`: It will remove the extra spaces in the string value in the variable `Conversion_string` and store it in the variable name present in the fourth column of the keyword script.

7.6. Loop Function

Description: This function is used to repeat a set of statements for a specified number of times.

Variables:

- a) `Loopflag` - This variable is used to hold either zero or one.
- b) `intCrement` - This variable is used to hold the index value of the loop, which is set to one.
- c) `intStartRow` - This variable is used to store the start row of the loop.
- d) `intEndRow` - This variable is used to store the end row of the loop.
- e) `intNoOfTimes` - This variable is used to store the loop count.

Functionality:

- Before the loop function is called, `Loopflag` variable is set to one.
- Variable `intCrement` is initialized with value one.
- Start row of the loop is stored in the variable `intStartRow`.
- End row of the loop is stored in the variable `intEndRow`. Set of statements in the keyword script from start row to end row gets executed until a specified number of times, which is stored in the variable `intNoOfTimes`.

8. Common Functions

8.1. Function for FSO Operations

Name of the function: Func_CommonFunctions()

Description: This function is used to perform a set of operations using the file system objects.

Parameters:

- a) Object - This holds the type of object being used for FSO, such as file or folder.
- b) ActionValueOne: This variable holds the details to be used while using FSO.

Assumptions: NA

Variables:

- a) cCellData - This variable stores the type of object(Ex: file or folder) to be used.

Based on the values in the variable cCellData, this function performs different actions. If the value in cCellData is:

- i. file: It will call the function Func_File() while passing the variable ActionValueOne as argument.
- ii. folder: It will call the function Func_Folder() while passing the variable ActionValueOne as argument.

8.2 Function for File Operations

Name of the function: Func_File()

Description: This function is used for working with files by using FSO.

Parameters:

- a) ActionValueOne - This holds the details to be used while using FSO.

Assumptions: NA

Variables:

- a) checkVal - This variable is used to store the details of the operation to be performed.
- b) SourceFilePath - This variable is used to store the source file path.
- c) DestinationFilePath - This variable is used to store the destination file path.
- d) hFile - This is used to store the header file name during the file operations.

Based on the values in the checkVal, the function will perform different actions. If the value in checkVal is:

i) create:

- If the file path specified in the variable `SourceFilePath` is already present, then a report is generated stating that the file already exists.
- If the file is not present, then a new file is created in the specified path.

ii) delete:

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.
- If the file is present, then the file is deleted from the specified path.

iii) copy:

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.
- If the file is present, then it is copied to the location present in the variable `DestinationFilePath`.

iv) move:

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.
- If the file is present, it is moved to the location present in the variable `DestinationFilePath`.

v) write:

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.
- If the file is present, then required text is written in the file.

vi) read:

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.
- If the file is present, then required contents of the file is read from the file.

vii) readline

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.
- If the file is present, then the required line is read from the file.

viii) Append

- If the file path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the file does not exist.

- If the file is present, then it opens the file in append mode (i.e., write = true) and appends the required value in the file.

8.3 Function for Folder Operations

Name of the function: Func_Folder()

Description: This function is used to work on folders using FSO.

Parameters:

- a) ActionValueOne - This holds the details to be used while using FSO.

Variables:

- a) checkVal - This variable is used to store the details of the operation to be performed.
- b) SourceFolderPath - This variable is used to store the source folder path.
- c) DestinationFolderPath - This variable is used to store the destination folder path.

Based on the values in the checkVal, the function will perform different actions. If the value in checkVal is:

i) create:

- If the folder path specified in variable SourceFilePath is already present, then a report is generated stating that the folder already exists.
- If the folder is not present, then a new folder is created with the specified path in the variable SourceFilePath.

ii) delete:

- If the folder path specified in the variable SourceFilePath is not present then, a report is generated stating that the folder does not exist.
- If the folder is present then, that folder is deleted from specified path in variable SourceFilePath.

iii) copy:

- If the folder path specified in the variable SourceFilePath is not present, then a report is generated stating that the folder does not exist.
- If the folder is present, then it is copied to the path mentioned in the variable DestinationFilePath.

iv) move:

- If the folder path specified in the variable `SourceFilePath` is not present, then a report is generated stating that the folder does not exist.
- If the folder is present, it is moved to the path mentioned in the variable `DestinationFilePath`.

9. User-Defined Functions

9.1. Function for CallFunction Keyword

Name of the function: Func_FunctionCall()

Description: This is the template for users to create their own functions.

Parameters:

- a) Object - The function name to be used.
- b) ActionValueOne - The parameters to be used with the called function.

Assumptions: NA

Variables:

- a) curFunctionName - This variable is used to store the function name.
- b) arrParameters - This variable is used to store the action parameters.

COPYRIGHT

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.