Descriptive programming

Descriptive programming is used to execute operations on an Object in the AUT whose definition is not stored in the Object Repository. Using this mechanism, you can bypass identification from the Object Repository and supply the Object Description in the statement itself.

An object name is simply used to map an object in script with its description in an object repository. Meaning if you change the object name in your script and object repository, the script should run.

We can use descriptive programming when:

* When the testers want to perform an operation on an object that is not present in the object repository
* When objects in the application are very dynamic in nature.
* When the Object Repository grows big, it results in poor Performance as the size of the Object Repository increThis means thatases.
* When the framework is built, such that it has been decided not to use Object Repository at all.
* When testers want to perform an action on the application at run-time without having the knowledge of object's unique properties.

# Types of descriptive programming

1. Static descriptive programming
2. Dynamic descriptive programming

## Static descriptive programming

In Static Method, for object identification, you specify an object’s property in the following format

property:=values,

This format is called property value pair and is enclosed in inverted commas

If your object uses multiple descriptions for identification, you can specify those using commas

"nativeclass:=Edit", "attached text:=Agent Name:"

**Example: Registration using descriptive programming**

SystemUtil.Run "msedge.exe","https://magento.softwaretestingboard.com/customer/account/create/"  
Browser("Create New Customer Account").Page("Create New Customer Account").WebEdit("html tag:=INPUT","title:=First Name").**Set** "Johnny"  
Browser("Create New Customer Account").Page("Create New Customer Account").WebEdit("html tag:=INPUT","title:=Last Name").**Set** "Mark"  
Browser("Create New Customer Account").Page("Create New Customer Account").WebEdit("html tag:=INPUT","title:=Email").**Set** "johnnymarksep178@yopmail.com"  
Browser("Create New Customer Account").Page("Create New Customer Account").WebEdit("html tag:=INPUT","title:=Password").**Set** "Abcd@123"  
Browser("Create New Customer Account").Page("Create New Customer Account").WebEdit("html tag:=INPUT","title:=Confirm Password").**Set** "Abcd@123"  
Browser("Create New Customer Account").Page("Create New Customer Account").WebButton("html tag:=BUTTON","title:=Create an Account").Click  
confText = Browser("Create New Customer Account").Page("Create New Customer Account").WebElement("html tag:=DIV","innerhtml:=Thank you.\*").GetROProperty("innerhtml")  
**msgbox** confText

## Dynamic Descriptive programming

The second method of doing the same action is using Dynamic Descriptive programming

In case your script uses the descriptive programming object candidate multiple times, it will be very tiresome to specify all the property value pairs for each statement

In such cases, you can make use of **Description Class** provided by UFT.

Syntax:

Set MyDescription = Description.Create()

MyDescription("property").Value = "property-value"

Example:

**Set** getAllLinks = **Description**.Create()  
getAllLinks("html tag").value = "A"  
getAllLinks("text").value = "\w.\*"  
**set** allLinks = Browser("Create New Customer Account").Page("Create New Customer Account").ChildObjects(getAllLinks)  
**For** i = 0 **to** allLinks.Count - 1       
   *'get the name of all the links in the page*  
   x = allLinks(i).GetROProperty("text")   
   print x   
**Next**

# Ordinal Identifier

Descriptive programming is used to write the script based on ordinal identifiers, which will enable UFT to act on those objects when two or more objects have the same properties.

Set Obj = Browser("title:=.\*google.\*").Page("micclass:=Page")

Obj.WebEdit("name:=Test","location:=0").Set "ABC"

Obj.WebEdit("name:=Test","location:=1").Set "123"

# Recovery Scenarios