Web table

Example 1: Validate if a webtable exist

boolExist = Browser(…).page(…).webtable(…).exist

Example 2: To Find number of rows and columns in a web table

intRowCnt=Browser("Google").Page("title:=.\*").WebTable(“name:= TTable").RowCount  
For r=1 to intRowCnt  
‘’ This will loop through each row and tell count of column in each row.  
intColCnt=Browser("Google").Page("title:=.\*").WebTable(“name:=TTable").ColumnCount(r)   
MsgBox intColCnt  
Next

Example 3: How to get data in a particular cell in the data table

strData= Browser("Google").Page("title:=.\*").WebTable(“name:=TTable").GetCellData(r,c)  
Where r = row number and c = column number

Example 4: Using childObject method to find objects of a particular type within a webtable

Public Function func\_findObjectinaWebTable(strObjType)  
Set objDesc=Description.Create  
objDesc("micclass").value=strobjType  
set objChild=Browser("Google").Page("title:=.\*").WebTable(“name:=TTa").ChildObjects(objDesc)  
func\_findObjectinaWebTable = objChild.count  
End Function

Example 5: Using HTML DOM to get count of objects in a webtable

Public Function func\_findObjectinaWebTable(strTagName)  
set objChild= Browser("Google").Page("title:=.\*").WebTable(“name:=TestTable").object.getElementsbyTagName(strTagName)  
func\_findObjectinaWebTable = objChild.length  
End Function  
  
Example 6: Using ChildItemCount and childItem in webtable to extract information

Public Function func\_findObjectinaWebTable(strObjType,introw,intCol)  
ob0jChildCnt=Browser("Google").Page("title:=.\*").WebTable(“name:=TestTable,"index:=0").ChildItemCount(introw,intcol,strobjType)  
If objChildCnt >0  
Childitem will return object oftype defined in arguments for childitem  
Set ChldItm = Browser("Google").Page("title:=.\*").WebTable(“name:=TestTable,"index:=0").ChildItem(introw,intcol,strobjType,0)  
If (strobjType = "Link" or strobjType ="WebEdit") Then  
ChldItm.Click  
ElseIf(stobjType = "WebCheckBox") Then  
ChldItm.Set "ON"  
End If  
End Function

## Database

## Connection Object

Before a database can be accessed by QTP, An object for database connection has to be established.

*Set adocon = createobject(“Adodb.connection”)*

Once the object is created, we need to set connection string to connect to the database. We can define a connection string with details of database including database provider, database, and user credentials for access.

### Some useful methods and properties of connection object are as follows:

### Properties:

#### adocon.connectionstring – **This property sets or returns the details used to create a connection to a data source. Includes details of driver, database,username and password.**

*Strconnection = "Driver=… "Server=****svrnme****;uid=****username****;pwd=****password****;"*

For e.g to connect to a database for excel the strConnection would be like:

*strcon = "Driver={Microsoft Excel Driver (\*.xls)};Dbqls="+xfilename +";ReadOnly=0;"*

For details of connection strings, see[**www.connectionstrings.com**](http://www.connectionstrings.com/)

#### adocon.ConnectionTimeout –**this defines the time  to wait for a connection to be established.**

#### adocon.provider –**This sets or gets connection provider name.**

#### adocon.state – **gives status whether connection is on or off.**

### Methods

#### adocon.open –**opens a database connection based on the connection string provided.**

#### **adocon.Open connectionstring,userID,password,options**

#### adocon.Execute – **execute the sql statement provided**

*adocon.execute “Select \* from table”*

#### adocon.close – **This closes the adodb connection.**

### RecordSet Object:

 Once a connection has been established, we can create recordset object to hold a set of record from database. A recorset consists of records and column

Set rs = createobject(“”Adodb.recordset”)

Some useful methods and properties of RecordSet Objects are as follows:

#### Properties:

#### BOF property **returns True  if the current record position is before the first record in the Recordset,**

#### EOF property **returns True if the current record position is after the last record in the Recordset, otherwise it returns False. For a empty recordset,i.e no records in the recordset or empty recordset, value of BOF and EOF is false. So the property can be used in a loop to validate RecordSet does not return any records.**

#### MaxCount **Property returns the maximum value of records to be returned in a recordset.** **rs.MaxCount = 20 will return 20 rows of data in recordset.**

#### Methods:

#### rs.cancel – **cancels an existing execution.**

#### rs.clone – **returns a clone of existing recorset and assigns to an object**

#### **set rsclone = rs.clone**

#### rs.Close - **closes instance of recordset**

#### rs.open – **opens a recordset based on query specified.**

#### **rs.open sqlquery, adocon**

*where sqlquery is query executed and adocon is connection object.*

#### rs.move – **moves the pointer in a recordset by specified count as defined in numrec rs.move numrec, start.**

#### **Also  movenext,moveprevious, movefirst, movelast can be used to move to specified location in recordset.**

#### rs.fields.count**gives number of items in the fields collection.**

#### rs.field.item(i) **returns specified item from the collection.**

## **Methods to Access XML DOM:**

### 1. Creating an object for xmlDOM

*set  xmldom = Createobject("MSXML.DOMDocument")*

*xmldom.async = “False”*

### 2. Loading a file using xml DOM

*Xmldom.Load(FileName)*

### 3. To Find number of parent nodes in a xml

*Intlen = Xmldom.childnodes.length*

### 4. To find names of parent nodes

*For i=0 to intlen-1*

*strParNodeName = xmldom.childnodes(i).Name*

*Next*

### 5. How to search a node with tagName

*Set nodelist = xmldom.getElementsByTagName(strTagName)*

This gives a collection of nodes with tag name as strTagName.

The tagName value "\*" will returns all elements in the document.

### 6. How to extract information from nodelist using GetElementbyTagName

*Set nodelist = xmldom.getElementsByTagName(strTagName)*

*For i= 0 to nodelist.length – 1*

*strText = nodelist.item(i).xml;*

*msgbox strText*

*Next*

### So points to remember in this are as follows:

#### **We can get all elements with the tagname using xmldom.getElementsByTagName**

#### **Once we have the list, we can access each of the nodes in the list using item(index).property**

#### **To access attribute property value for a node , use strText = nodelist.item(i).getattribute("category")**

#### **To get xml content of a node, use strText = nodelist.item(i).xml**

#### **Use strText = nodelist.item(i).nodeName to get the name of required node.**

#### **Use strText = nodelist.item(i).text to get value in a particular node**

### Sample Code explaining the concept

*FileName = "d:\xmldom.xml"*

*''Create an instance of xml dom*

*set  xmldom = Createobject("MSXML.DOMDocument")*

*''set async = "False"*

*xmldom.async = "False"*

*xmldom.Load(FileName)*

*intlen = xmldom.childnodes.length*

*msgbox intlen*

*Set nodelist = xmldom.getElementsByTagName("title")*

*msgbox nodelist.length*

*If nodelist.length > 0 then*

*For each x in nodelist*

*AttName=x.getattribute("lang")*

*myname = x.Text*

*msgbox AttName +" "+ myname*

*Next*

*Else*

*msgbox " field not found."*

*End If*

### 7. Using Selectnodes method to extract information

*FileName = "d:\xmldom.xml"*

*''Create an instance of xml dom*

*set  xmldom = Createobject("MSXML.DOMDocument")*

*''set async = "False"*

*xmldom.async = "False"*

*xmldom.Load(FileName)*

*intlen = xmldom.childnodes.length*

*msgbox intlen*

*Set nodelist = xmldom.selectNodes("/bookstore/book/title")*

*msgbox nodelist.length*

*If nodelist.length > 0 then*

*For each x in nodelist*

*AttName=x.getattribute("lang")*

*myname = x.Text*

*msgbox AttName +" "+ myname*

*Next*

*Else*

*msgbox " field not found."*

*End If*

### 8. Using SelectSingleNode to extract information

*FileName = "d:\xmldom.xml"*

*''Create an instance of xml dom*

*set  xmldom = Createobject("MSXML.DOMDocument")*

*''set async = "False"*

*xmldom.async = "False"*

*xmldom.Load(FileName)*

*''This will give 1st item property matching the node structure*

*Msgbox xmldom.selectSingleNode("/bookstore/book/title").Text*

*''This will give 1st item property matching the node structure with filter as attribute value*

*Msgbox xmldom.selectSingleNode("/bookstore/book/title[@lang = 'ke']").Text*

*''This will give 1st item property matching the node structure with filter as text  value*

*Msgbox xmldom.selectSingleNode("/bookstore/book/title[.= 'Learning XML']").Text*

## QCUtil Object

QCUtil Object is used to access QC OTA.

Below are the properties of QCUtil object that can be used for extracting information from QC.

### Properties of QCUtil Object

#### 1. QCUtil.CurrentRun: **This give reference to the current test run in QC.We can then get  information of currentrun  like name as shown below:**

*set objRun = QCUtil.CurrentRun*

*msgbox objRun.Name*

#### 2. QCUtil.CurrentTest: **This give reference to the current test in QC. We can extract information like name of the test, adding attachment to current test using this property**

*set objTest = QCUtil.CurrentRun*

*msgbox objTest.Name*

#### 3. QCUtil.CurrentTestSet:**This give reference to the current test set in QC. We can  extract information similar as above for testset using this property.**

set CurrentTSTest = QCUtil.CurrentTestSet

#### 4. QCUtil.CurrentTestSetTest: **This gives reference to current test execution instance in the testset.**

*set CurrentTSTestSet = QCUtil.CurrentTestSetTest*

#### 5. IsConnected Property**: returns true/false based on QTP currently connected to QC project.**

blnQCsts = QCUtil.IsConnected

#### 6. QCConnection Property**: This gives an instance of current run session and can access the structure of QC using this property**

*Set QCConnection = QCUtil.QCConnection*

## Examples of working with QCUtil Object

### 1.Adding a defect in QC using QCUtil Object:

*‘Create instance of QCConnection*

*Set QCConnection = QCUtil.QCConnection*

*‘Create an instance of BugFactory*

*Set DefFactory = QCConnection.BugFactory*

*'Add a new defect*

*Set Bug = DefFactory.AddItem(Nothing)*

*‘Provide mandatory details for the defect*

*Bug.Status = “New”*

*Bug.Summary = “Module Detected new defect summary”*

*Bug.DetectedBy = “njoshi”*

*Bug.AssignedTo = “dev001”*

*Bug.Post*

*Set DefFactory = nothing*

*Set QCConnection = nothing.*

### 2. Adding Attachment to QC

*Dim ObjCurrentTest,ObjAttch*

*‘ We can add attachment to currentTest, current run, testset, and testsettest by using repective ‘properties   
Set ObjTest = QCUtil.CurrentTest.Attachments  
Set ObjAttachFile = ObjTest.AddItem(Null)  
ObjAttachFile.FileName = FileName ‘ Provide path of the file that needs to be attached to QC   
ObjAttachFile.Type = 1  
ObjAttachFile.Post  
ObjAttachFile.Refresh*

### 3. Validating If QC is connected properly:

*if QCUtil.IsConnected then*

*msgbox “QC is connected”*

*Else*

*Msgbox “QC is not connected”*

*EndIf*

### 4. Connecting to QC through QTP

*Set qtApp = CreateObject ("QuickTest.Application")*

*If  qtApp.launched <> True then*

*qtApp.Launch*

*End If*

*qtApp.Visible = "true"*

*If Not qtApp.TDConnection.IsConnected Then*

*qtApp.TDConnection.Connect QCurl, DomainName, ProjectName, UserName, Password, False*

*End If*

### How to use Regular Expressions in QTP

Regular expressions are used to identify objects and text strings with varying values. Regular Expression are strings that defines search phrase based on special character provided in the expression.This is useful when expected value of any object property is regularly changing but in a fix pattern.

## **Some useful points to remember about Regular Expressions are:**

* Regular Expression is useful for following scenarios in QTP:
* Defining the property values of an object in dialog boxes or in programmatic descriptions
* Defining expected values for checkpoints
* Defining pop-up window conditions in a recovery scenario.
* Can create regular expression for strings only.
* Period (.), hyphen (-), asterisk (\*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\) are special characters used to create regular expression.
* When one of above mentioned special characters is preceded by a backslash (\), QTP treats it as a literal character.
* By default, the value of all Property objects added to a Properties collection are treated as regular expressions.

### Below are the various regular expressions used in QTP.

#### Matching Any Single Character (.) **eg abc. Will match abc followed by any character.**

#### Matching Any Single Character in a List ( [xy] ) **e.g [ab] will match either a or b**

#### Matching Any Single Character Not in a List ( [^xy] ) **e.g 1[^23] will match all values between 11 to 19 except 12 and 13.**

#### Matching Any Single Character within a Range ( [x-y] ) **e.g : 1[1-3] will match 11,12, and 13.**

#### Matching Any AlphaNumeric Character Including the Underscore ( \w )

#### Matching Any Non-AlphaNumeric Character (\W) **will match any special character other than underscore. Please note case of W in this case.**

#### Matching Zero or More Specific Characters ( \* ) **This matches zero or more occurrences of the preceding character. e.g ca\* will match caa,caaaa,c and so on. Similarly c.\* will match c, cs,caaa, and so on, since preceding character here is “.”.**

#### Matching One or More Specific Characters ( + ) **Only difference from \* is it will match for minimum one character. e.g ta+r will match taar,tar but not tr as in above case.**

#### Matching Zero or One Specific Character ( ? ) **A question mark (?) instructs QTP to match zero or one occurrences of the preceding character. For example: te?r matches ter and tr, but nothing else**

#### Matching One of Several Regular Expressions ( | )  **e.g new|day will match either of new or day. If we write ne(w|d)ay, it will match neway or neday.**

#### Matching the Beginning of a Line ( ^ ) **This will match only if match is found at beginning of line.**

#### Matching the End of a Line ( $ )  **This will match only if match is found at end of line.**

#### Matching a word at boundary(\b) **e.g new\b will match testnew but not in knewit.**

#### Matches a digit character(\d)  **Matches a digit value.**

#### Matching a non-digit character(\D) **Matches a non digit value**

**RegExp**

VBScript provides a RegExp object to handle regular expressions.This object allows you to find regular expression matches in strings, and replace regex matches in strings with other strings.

## **Properties of RegExp Object**

Following are the properties of regexp object:

#### IgnoreCase:  **By default regular expression is case sensitive, to make it case insensitive, set value as “True”**

#### Pattern: **We can define the pattern of regular expression in the pattern property. Pattern can include literal and meta characters like .,\*,/d as discussed in**[**the earlier post on how to create regular expressions in QTP**](http://qaautomationqtp.blogspot.in/2013/05/how-to-use-regular-expressions-in-qtp.html)

#### Global: **To return or replace all matches, set Global as True. If set as False, finds the first match only**

‘’Defining a regular expression

Set newRegExp = New RegExp

newRegExp.IgnoreCase = True

newRegExp.Global = True

newRegExp.Pattern = ".\*an"

### Methods of RegExp object

#### Execute method:**executes a match against the specified string. Returns a Matchescollection, which contains a Match object for each match. The Matchobject can also contain SubMatches collection.**

#### **Execute method has following properties:**

#### Item

**§  Item.value**– value of item in the collection

**§  Firstindex** – First instance of match location

#### Count

Count of instances matching the regular expression

**Replace method:** replaces the part of the string found in a match with another string.Does replace the pattern defined for object with the replace value.

***Syntax:*** *regExpobj.replace(teststring,”newpattern”)*

#### Test method: **executes an attempted match and returns True or False based on match found.**

## **Sample Code explaining properties and methods of RegExp Object.**

*Set newRegExp = New RegExp*

*newRegExp.IgnoreCase = True*

*newRegExp.Global = True*

*newRegExp.Pattern = ".\*an"*

*testString = "this ant is an insect"*

*Set colmatch = newRegExp.Execute(teststring)*

*For each match in colmatch*

*msgbox match.Value*

*msgbox colmatch.count*

*Next*

*boolMatch = newRegExp.Test(teststring)*

*msgbox boolMatch*

*newRegExp.pattern ="a"*

*strString = newRegExp.replace(teststring,"b")*

*msgbox strstring*