Data manipulation w/ RPA

giovedì 16 maggio 2024 14:47

Variables

The most common types of variable are:

String, Integer, Array, Date and Time, Boolean, Data Tables

- Date and Time variable

It enables the user to store information about any date and time.

Generally used to calculate the number of days between two dates, store current date details, etc...

- DataTable variable

Can store tabular data in rows and columns

Typically used to migrate data from one databases to another, extract information from a website and store it locally in a spreadsheet.

- Queue Item variable

Stores an item that has been extracted from a container of items called a queue. Through this extraction the user can further use the queue items and other processes.

It's a proprietary variable of UiPath

Variables Panel: enables the user to create variables and modify them.

It's located at the bottom of the Designer panel in Studio.

If a user renames a variable, it will be renamed in all instances used.

Scope → the user specifies the region in which the variable can be visible.

Other way to define variable:

Message Box → CTRL + K → Name the variable

Assign value

- Through the Variable panel
- Searching the "Assign" activity and drop it

Arguments

Arguments are used to pass data from one workflow to another by storing the data dynamically.

Variables pass data between activities, while arguments pass data between workflows.

Arguments enable users to reuse automation workflows; hence they can be accessed outside the workflows in which they are defined and they are helpful in automation processes consisting in multiple workflows.

They have the same data types and they support the same methods and properties.

Properties: Name ; Direction ; Type ; Value

Direction → As arguments are used to pass data between workflows, the user needs to specify the direction from and to which the data is passed.

↳ Direction can be: IN ; OUT ; IN/OUT

Arguments panel enables to create arguments and modify them.

Another way to create arguments:

Message Box → CTRL + M → Name the argument

Argument Directions

↳ It's difficult to build a large project with a single workflow. As a best practice developers break the project into multiple smaller workflows that are manageable while working with multiple workflows. A developer may have to pass data between them.

The direction property of an argument determines whether a workflow will use the argument to send data to another workflow or receive data from another workflow.

There are 3 types of directions:

1) In : Direction used to receive data
2) Out : Direction used to send data
3) In/Out : Direction that allows an argument to work both ways.

Example

(-) Create 3 arguments : 1) in_int 1 ; in_int 2 with direction IN
2) out_sum with direction OUT

(-) New sequence with Assign activity
↳ out_sum = in_int 1 + in_int 2

(-) Select this sequence and select "Extract as a workflow"

(-) Create 3 variables in the main sequence : (•) INT 1 = 0
(•) INT 2 = 10
(•) RESULT

(-) Click IMPORT ARGUMENTS

↳ Pass the value of the variable in each value field

IN_INT 1 ⟶ INT 1
IN_INT 2 ⟶ INT 2
OUT_SUM ⟶ RESULT

DATA MANIPULATION

It's the process through which the data is altered using various operations in order to facilitate its usage.

Example : Convert an integer to a string using ".ToString"

(-) Search for Input Dialog in the Activity panel.

↳ Choose the variable value at runtime

(-) Assign INT_SUM = INT_FirstNumber + INT_SecondNumber

(-) Message Box = "The output is: " + INT_SUM.ToString

STRING MANIPULATION

• Concat method : concatenates the string representations of two specified objects

String.Concat (Var_0, Var_1)

• Contains method : checks whether a specified substring occurs within a string. It returns a Boolean value

VAR_NAME.Contains ("Blackrest")

• Format method : converts the value of objects to string based on the formats specified and inserts them into another string which reduces complexity and increases readability.

String.Format ("{0} lives in {1}", Var_0, Var_1)

• IndexOf method : returns the zero-based index of the first occurrence of a character in a string

VAR = "Smith" ; VAR.IndexOf (":") = 2

• Join method : concatenates the elements in a collection and

DISPLAYS THEM AS A STRING

ARR_VAR = {"WELCOME", "TO"}

↳ STRING.JOIN(" ", ARR_VAR) = "WELCOME TO"

1) REPLACE METHOD : IDENTIFIES A SEQUENCE OF CHARACTERS IN A TEXT AND REPLACES IT WITH A GIVEN STRING

NAME = "SMITH JONES" → NAME.REPLACE("SMITH", "BRANDON")

↳ NAME = "BRANDON JONES"

2) SPLIT METHOD : SPLITS A STRING INTO SUBSTRINGS BASED ON A CERTAIN CRITERIA. THIS SEPARATOR CAN BE A SPACE, A COMMA, A FULL STOP.

VAR = "WELCOME TO UIPATH" → VAR.SPLIT(" ")(2) = UIPATH

↳ INDEX OF SUBSTRING TO TAKE

3) SUBSTRING METHOD : EXTRACTS A SUBSTRING FROM A STRING USING THE STARTING INDEX AND THE LENGTH. USED TO ISOLATE OR SEPARATE A SUBSTRING FROM THE ORIGINAL STRING.

VAR = "WELCOME TO UIPATH", VAR.SUBSTRING(0, 7) = "WELCOME"

STRING ACTIVITIES IN STUDIO

MODIFY TEXT ACTIVITY : UPDATES A TEXT VALUE USING MODIFICATIONS INCLUDING FIND AND REPLACE, TRIM, COMBINING OR CONCATENATING WITH ANOTHER TEXT VALUE AND CHANGE TO UPPER OR LOWER CASE.

REGULAR EXPRESSION (REGEX)

IT'S A SPECIFIC SEARCH PATTERN THAT CAN BE USED TO EASILY MATCH, LOCATE AND MANAGE TEXT.

IT CAN BE USED FOR :

- INPUT VALIDATION
- STRING PARSING
- DATA SCRAPING
- STRING MANIPULATION

REGEX BUILDER INTEGRATED ACTIVITIES

1) MATCHES : SEARCHES AN INPUT STRING FOR ALL OCCURRENCES AND RETURNS ALL THE SUCCESSFUL MATCHES.
THIS ACTIVITY CAN BE USED TO RETRIEVE ALL THE ENTITIES AND USE THEM FURTHER.

2) IS MATCH : INDICATES WHETHER THE SPECIFIED REGULAR EXPRESSION FINDS

A match in the specified input string, it returns only a true or false value.
This activity can serve as a condition for another activity.

3) **Replace :** replaces strings that match a regular expression pattern with a specified replacement string.
This activity can be used for data quality purposes.

c) RegEx Builder wizard

The wizard helps ease the process of building and testing Regular Expression search criteria.
The RegEx wizard can be opened from the body of each of any these 3 activities

The user interface of the builder has 3 sections :
- Test Text
- Type value and quantifiers
- Full expression

1) **Test Text :** the user can test the chosen search criteria against the text on which RegEx is applied.

2) **Type value and quantifiers :** user can choose the RegEx expression's type, value and quantifiers.

a) Type drop down allows searching for a given text or one expression from many. It can be configured to search only at the beginning or at the end. It also offers pre-built expressions for emails, URL, US date and US phone numbers.

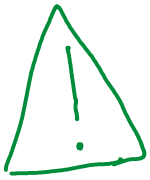
b) The value field contains exactly the text that needs to be retrieved.

c) The quantifiers drop down list enables the user to select the type of results that should be displayed.

- Exactly : select an exact amount of consecutive occurrences the user wants to find.
- Any : highlights any number of consecutive found matches starting from zero.
- At least one : highlights any number of consecutive found matches starting from one.
- Zero or one : only highlights a single consecutive occurrence of the term.

e) Between X and Y Times : Highlights the number of consecutive occurrences selected

3) Full Expression : Displays the current RegEx expression in its raw form.



In the properties panel of the "Matches Activity" select the input message and the output variable

---

Data Table Manipulation

A DataTable is an in-memory representation of a single database table that has a collection of rows and columns.

Columns are identified through capital letters and rows through numbers.

In Studio there are various ways to create DataTables.

1) Build DataTable : This activity builds a reusable structure for a DataTable using a dedicated window. It allows the definition of several columns and type of data for each column. The user can configure each column with specific options

2) Read Range : This activity reads the data from an existing file, copies the content of a worksheet or a selection from that worksheet and stores it in a DataTable variable.
The DataTable variable can be created directly from the Properties panel using Ctrl+K shortcut.

3) Read CSV : This activity reads the content of a CSV known as a comma-separated values file and stores it in a DataTable variable.

4) Data Scraping : It enables the user to extract structured data from a browser, application or document and store it in a DataTable.

Data Table Activities

- Add Data Column : Adds a column to an existing DataTable.
The input data can be of DataColumn type or the column can be added empty by specifying the data type and configuring the options.

- Add Data Row : input data can be of DataRow type or can be entered as an Array Row by matching each object with the data type of each column

- Lookup Data Table : it allows searching for a provided value in a specified DataTable and returns the RowIndex at which it was found, or it can be configured to return the value from a cell with given coordinates that is RowIndex and Target Column.

- Filter Data Table : Allows filtering a DataTable through a Filter Wizard, using various conditions. This activity can be configured to create a new DataTable for the output of the activity, or to keep the existing one and filter out or delete the entries that do not match the filtering conditions.

- Join DataTables : It combines data from two tables by using values common to each other. It is one of the most useful activities in business scenarios, where working with more than one DataTable is very common.

- Merge Data Table : It is used to merge a specified DataTable with the current DataTable. It provides four types of actions to take when merging : Adding, Ignoring, Returning an error and Manual input

- Generate Data Table : creates a DataTable variable from unstructured data, by letting the user indicate the row and column separations. This option is extremely useful when data is captured from scanned documents or web scraping.

- For Each Row : Similar to For each loop as it iterates through all the rows in a DataTable and performs the same action.

- Clear Data Table : clear all the data entries in a DataTable. It can be very useful with DataTables that are used as intermediary tables for data moved from one to another.

- Output Data Table : used to write a specified DataTable into [illegible] in a CSV format. This can serve as an

A string variable ... a ...

intermediate step in a process or as a final step when, after several manipulations, a DataTable contains only several numbers.

•) Remove Data : it's a group of two activities used to remove rows or columns from an existing DataTable. Rows can be identified by their index number or by inputting the DataRow as an object and the columns can be alternatively identified by their names.

•) Remove Duplicate Rows

•) Sort DataTable : used to sort DataTables using a specific column as the criterion. Besides indicating which column to use in this respect, the user can choose whether the sorting will be done in ascending or descending order.

Collections Manipulation

A collection variable represents a set of similar data type items in a single unit used for grouping and managing them.

Different types of Collection variables are : Arrays, Lists, Dictionaries

•) Add to Collection : adds an item to a specified collection

•) Remove from Collection: removes an item from a specifics collection and can output a Boolean variable that confirms the success of the removal operation

•) Exists in Collection : indicates whether a given item is present in a collection by outputting a Boolean as a result.

•) Clear Collection : clears a specified collections of an items.

• Lists = data structures consisting of objects of the same data type

1) Add and remove items
2) Search for an item
4) Sorting the objects
5) Extracting items and converting them

3) Looping through the items to other data types.

- Create new list: type = List<T> and choose String
  default = New List(Of String) From {"Jack", "Ali", ...}
- Create temp list: same type, default empty
- Invoke method =>
  - Target Type = (Null)
  - Target Object = Student_List
  - Method Name = Sort / Reverse
- Assign: TempList = Student_List.GetRange(0, 5)
- Message Box: String.Join(",", TempList.ToArray)

• Dictionaries

is a collection of words and their meanings or definitions.
They consist of key and value pairs in which the keys are unique

- Adding and deleting (key, value) pairs
- Retrieving the value associated with a key
- Re-assigning new values to existing keys

- Initializing
- Adding
- Removing
- Retrieving