

LECTURE 03. COLLECTIONS. PART II

Robotic Process Automation

[17 October 2023]

Elective Course, 2023-2024, Fall Semester

Camelia Chisăliță-Crețu, Lecturer PhD

Babeș-Bolyai University

Acknowledgements

This course is presented to our Faculty with the support of UiPath Romania.



Contents

- **PART I**
- **Arguments**
 - Definition. Types. direction
 - Invoke Workflow File Activity
 - Demo 1
- **Variable Categories**
- **Array**
 - Details
 - Declaration. Instantiation. Initialization
 - Demo 2
 - Demo 3
- **List**
 - Details
 - Declaration. Instantiation. Initialization
 - Operations
 - Demo 4
 - Demo 5
- **PART II**
- **Dictionary**
 - Details
 - Declaration. Instantiation. Initialization
 - Operations
 - Demo 6
 - Demo 7
- **Data Table**
 - Details
 - Declaration. Instantiation. Initialization
 - Operations
 - Demo 8
 - Demo 9
- **References**

Dictionary. Details

- **Dictionary** characteristics in UiPath:
 - **a dictionary has a flexible length;**
 - it implements the **IEnumerable** interface ==> can be iterated by using a **For Each** activity over the **keys set**;
 - it is used to store:
 - multiple related pairs (key, value) that are passed as a **single argument** between workflows;
 - data in **Orchestrator queues**.

see **Demo6 - Dictionaries**

Dictionary. Declaration. Instantiation. Initialization

- ways to **declare/instantiate/initialize** a dictionary:
 - **Variables Panel:**
 - **Name:** bookDictionary; **Type:** Dictionary<String, String>;
 - **Default:** `new Dictionary (of String, String) from {"title", "Poems"}, {"author", "M.Eminescu"}, {"publisher", "Litera"}} //Count=3`
 - **Name:** gradeDictionary; **Type:** Dictionary<String, List<String>>;
 - **Default:** `new Dictionary(of Int32, List(of String)) from {{10, new List (of String) from {"Ana", "Anca"}}, {3, new List(of String) from {"me", "you", "her"}}} //Count=2`
 - **Assign** activity:
 - `monthDictionary = new Dictionary (of Int32, List(of String)) //Count=0`
 - `sDictionary = new Dictionary(of String, String) //Count=0, pairs are added later`

Dictionary. Operations

- ways to add pairs in a dictionary:
 - **Assign** activity:
 - `bookDictionary("year") = "2019"` // overrides the value on key “year” or adds a new pair {"year", "2019"}
 - `monthDictionary(30) = new List(of String) from {"April", "June", "September"}`
 - **Add To Collection** activity:
 - for `monthDictionary = new Dictionary (of Int32, List(of String))` the properties that are set:
 - `Collection = monthDictionary(31);`
 - `Item = "March";`

Dictionaries. Example 1

The screenshot displays a UiPath Studio workflow and its execution output. The workflow consists of the following steps:

- bookDictionary (year = 2019)**: A dictionary is created with the year 2019.
- A+B Assign**: The dictionary is updated with the year 2018.
- Write Line**: The dictionary is converted to a string using `String.Join(" ", bookDictionary.Keys(1))`.
- Write Line**: The dictionary is converted to a string using `String.Join(" ", "key:" + bookDictionary.Keys(1) + " value: " + bookDictionary.Values(1))`.

The **Variables** pane at the bottom shows the variable `bookDictionary` of type `Dictionary<String, String>` with its default value: `new Dictionary (of String, String) from`.

The **Output** pane on the right shows the execution results:

- ⓘ Dictionaries execution started
- ⓘ [title, Poems] [author, M.Eminescu]
- ⓘ [publisher, Litera] [year, 2018]
- ⓘ key:author value: M.Eminescu
- ⓘ Dictionaries execution ended in: 00:00:00

see Demo6 - Dictionaries

Dictionaries. Example 2

The screenshot displays the UiPath Studio interface. The main workspace shows a workflow with a 'For Each' loop. The loop iterates over 'gradeDictionary.Keys'. Inside the loop, there is a 'Write Line' activity with the text 'key: ' + key.ToString + ' Studer'. A tooltip shows the full text: 'key: ' + key.ToString + ' Students: ' + String.Join(' ', gradeDictionary(key))

On the right, the 'Properties' pane shows the 'Common' and 'Misc' sections. The 'Output' pane shows the execution results:

- ⓘ Dictionaries execution started
- ⓘ key: 10 Students: Ana Anca
- ⓘ key: 3 Students: me you her
- ⓘ Dictionaries execution ended in: 00:00:00

At the bottom, the 'Variables' pane shows the following table:

Name	Variable type	Scope	Default
gradeDictionary	Dictionary<Int32,List<String>>	Sequence	new Dictionary(of Int32, List(of String)) from {{10, new List (of St

see Demo6 - Dictionaries

Dictionaries. Example 3

The image shows a UiPath workflow diagram and its interface. The workflow consists of the following steps:

- Assign: `monthDictionary(3) = new List(of String) { "April", "June", "September" }`
- Add To Collection
- Assign: `monthDictionary(3) = new List(of String) { "April", "June", "September" }`
- For Each Key in the Dictionary:
 - ForEach: `key` in `monthDictionary.Keys`
 - Body:
 - Body (Double-click to view)

The interface includes a Properties window on the right with the following settings:

- Common**
 - DisplayName: Add To Collection
- Misc**
 - Collection: monthDictionary(31)
 - Item: "March"
 - Private: ☐
 - TypeArgume...: String

The Output window shows the following log:

- ⓘ Dictionaries execution started
- ⓘ days: 31 months: January March
- ⓘ days: 30 months: April June September
- ⓘ Dictionaries execution ended in: 00:00:01

The Variables window at the bottom shows the following table:

Name	Variable type	Scope	Default
monthDictionary	Dictionary<Int32,List<String>>	Sequence	Enter a VB expression

see Demo6 - Dictionaries

Demo 7

- Create a process that performs the following actions:
 - 1. *read* pairs of (continent, country):
 - 1.1. *build* a dictionary of countries organized by continents;
 - E.g.: {"Asia-Japan", "North America-USA", "Europe-Romania", "South America-Argentina", "North America-Canada", "Asia-China", "Australia-Australia"}
 - 2. *print* the dictionary sorted by continents;
 - 3. *write* the dictionary details into a .txt file;
 - *use Append Line activity.*

Data Table. Details

- **Data Tables** characteristics in UiPath:
 - a **data structure with flexible length**;
 - it is similar to a Excel sheet consisting of rows and columns;
 - it can be iterated by using a **For Each Row** activity;
 - it is used for:
 - storing **data from Excel sheets** and .csv files;
 - web **data scrapping**.

Row/ Column	First	Last	Club Member
0	"John"	"Doe"	Yes
1	"Jane"	"Doe"	No
2	"Jane"	"Doe"	Yes
3	"John"	"Doe"	No

see **Demo8 - DataTables**

Data Table. Declaration. Instantiation. Initialization

- ways to **declare/instantiate/initialize** a dictionary:
 - Variables Panel:**
 - Name:** studentsTable; **Type:** DataTable;
 - Read CSV** activity:
 - properties to be set:
 - FilePath = "**members.csv**";
 - IncludeColumnNames = *checked*;
 - DataTable = membersDataTable.

Row/ Column	First	Last	Club Member
0	"John"	"Doe"	Yes
1	"Jane"	"Doe"	No
2	"Jane"	"Doe"	Yes
3	"John"	"Doe"	No

see **Demo8 - DataTables**

Data Table. Operations (1)

- ways to **convert Data Table to String**:
 - **Output Data Table** activity:
 - properties to be set:
 - Input = membersDataTable;
 - Output = <a String variable>;
- ways to **access data by rows** in a data table:
 - **For Each Row** activity:
 - variable to iterate **DataRow** objects in arrays, e.g., **row**;
 - accessing a field in a **row** formed of [First, Last, Club Member] attributes:
 - firstName= row(**“first”**).ToString;
 - field name is case insensitive, e.g., first, First;

Data Table. Operations (2)

- ways to **access data by indexing rows/columns** in data table:
 - firstName = membersDataTable.Rows(1)("first").ToString;
 - status = membersDataTable.Rows(0)("club member").ToString;
 - lastName = membersTable.Rows(0)("Last").ToString;
 - lastName = membersTable.Rows(0)(1).ToString;
- ways to **filter data by using rules** in a data table:
 - **Select** method:
 - **Array of DataRow** filtered = membersTable.Select("first>'M' AND"+"[club member]='YES'");
 - the result is an **Array** iterated with **For Each** activity;
 - **Filter Data Table** activity:
 - it allows to follow a wizard that states the rules and the output columns;
 - the result is a **Data Table** iterated with **For Each Row** activity.

Data Tables. Example 1. Output Data Table

The image displays a UiPath workflow diagram and its corresponding interface. The workflow consists of the following steps:

- Read CSV**: Reads the file "members.csv".
- Output Data Table**: Outputs the data to a variable named "membersTable".
- Write Line**: Writes the value of "output" to the console.
- Write Line**: Writes the string `membersTable.Rows(0)("club member").ToString` to the console.
- Write Line**: Writes the string `membersTable.Rows(0)(1).To` to the console.

The **Variables** pane at the bottom shows the following variables:

Name	Variable type	Scope	Default
membersTable	DataTable	Sequence	Enter a VB expression
output	String	Sequence	Enter a VB expression

The **Properties** pane on the right shows the following settings:

- Common**: DisplayName: Output Data Table
- Input**: DataTable: membersTable
- Misc**: Private: ☐
- Output**: Text: output

The **Output** pane shows the following output:

```
First,Last,Club Member
Jay,Gavin,Yes
Zachary,Craig,No
Sherry,Hooks,Yes
Marcella,Knipp,Yes
Melvin,White,No
Nancy,Fox,Yes
Kristina,Turner,No
Craig,Saucier,No
Raymond,Wilson,No
Kent,Danley,No
Yes
Gavin
```

see Demo8 - DataTables

Data Tables. Example 2A. Select method

The screenshot displays a UiPath workflow designed to filter and display data from a Data Table. The workflow consists of the following steps:

- Assign:** A yellow box labeled "A*B Assign" contains the expression `filteredDataRows = membersTable.Select`.
- Select:** A grey box contains the SQL query: `membersTable.Select("first>'M' AND"+"[club member]='YES'")`.
- For Each:** A loop block with the header "For Each" and a body section. The loop iterates over `filteredDataRows`, with each iteration item labeled `item`.
- Body:** Inside the loop, there is a "Body" container with a "Write Line" activity. The "Text" property of "Write Line" is set to `item("first").ToString`.

At the bottom of the workflow editor, a table provides details for the `filteredDataRows` variable:

Name	Variable type	Scope	Default
filteredDataRows	DataRow[]	Sequence	Enter a VB expression

The right-hand side of the interface shows the "Properties" and "Output" panels. The "Properties" panel includes sections for "Common" (with "DisplayName" set to "Assign") and "Misc" (with "Private" unchecked, "To" set to `filteredDataRows`, and "Value" set to `membersTable.Select`). The "Output" panel displays a list of names: Marcella, Nancy, Sherry, Zachary, Kristina, Craig, Raymond, Kent, and a message indicating "DataTables execution ended in: 00:00:00".

see Demo8 - DataTables

Data Tables. Example 2B. Filter Data Table

The screenshot displays the UiPath Studio interface for a workflow titled 'Filter Data Table'. The workflow is structured as follows:

- Filter Data Table** (Activity): Contains a 'Filter Wizard...' button.
- For Each Row** (Loop): Iterates over 'filteredDataTable' using the variable 'row'.
- Body** (Container): Contains a 'Write Line' activity with the text 'row(\"first\").ToString()'.

The right sidebar shows the 'Properties' window for the 'Filter Data Table' activity. The 'Common' tab is selected, showing the following properties:

- DisplayName**: Filter Data Table
- Input**: DataTable (membersTable)
- Misc**: Private (checkbox)
- Options**: FilterRowsMode (Keep), SelectColumn... (Keep)
- Output**: DataTable (filteredDataTabl)

The bottom status bar shows the 'Variables' tab with the following table:

Name	Variable type	Scope	Default
filteredDataTable	DataTable	Sequence	Enter a VB expression

see Demo8 - DataTables

Data Tables. Example 2B. Filter Data Table - Wizard

Filter Wizard

Input DataTable: membersTable Output DataTable: filteredDataTable

Filter Rows Output Columns

Rows Filtering Mode

☒ Keep ☐ Remove

	Column	Operation	Value		
	first	Does Not Start With	M	x	+
And	club member	=	No	x	+

Filter Wizard

Input DataTable: membersTable Output DataTable: filteredDataTable

Filter Rows Output Columns

Columns Selection Mode

☒ Keep ☐ Remove

Column		
first	x	+
last	x	+
club member	x	+

OK Cancel

see Demo8 - DataTables

Demo 9

- Create a process that performs the following actions:
 - 1. *read* “Students.csv” file with the following structure: Student, Specialisation, Group;
 - *use Read CSV activity;*
 - 2. *print* the .csv file content;
 - *use Output Data Table activity;*
 - 3. *enter* a specialisation;
 - 4. *filter* students by specialisation and order them by group;
 - *use various ways:*
 - `filteredSortedDataTable = (From row In studentsDataTable.Select("specialisation='"+spec+"'") Order By Convert.ToInt32(row("group")), row("student") Select row).ToArray.CopyToDatatable()`
 - `filteredDataTable = studentsDataTable.Select("specialisation='"+spec+"'")`
 - `sortedDataTable <== Sort Data Table` activity
 - 5. *save* the resulted data into a .csv file.
 - *use Write CSV activity to write a Data Table object to a .csv file.*
- see Demo9 - StudentsDataTable*

References

- UiPath Docs - <https://docs.uipath.com/>
- UiPath Studio Docs - <https://docs.uipath.com/studio/standalone/2023.4>
- UiPath Forum - <https://forum.uipath.com/>
- UiPath Academy - <https://academy.uipath.com/>