

LECTURE 03. COLLECTIONS. PART I

Robotic Process Automation

[15 October 2019]

Elective Course, 2019-2020, 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
- **Generic Value Type**
 - Methods
- **Variable Categories**
- **Array**
 - Details
 - Declaration. Instantiation. Initialization
 - Example 1, 2, 3
 - Demo 2
- **List**
 - Details
 - Declaration. Instantiation. Initialization
 - Example 1
- Operations
- Example 2, 3
- Demo 3
- **PART II**
- **Dictionary**
 - Details
 - Declaration. Instantiation. Initialization
 - Operations
 - Example 1, 2, 3
 - Demo 4
- **Data Table**
 - Details
 - Declaration. Instantiation. Initialization
 - Operations
 - Example 1, 2, 3
 - Demo 5
- **References**

Arguments. Definition

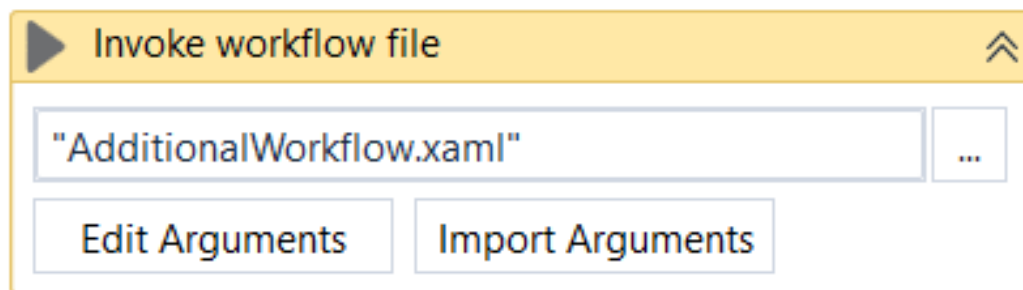
- **Arguments** are
 - used to pass data from a project to another;
 - similar to parameters in method definition;
- **advantages:**
 - increase workflow readability;
 - increase sequence/flowchart reusability in workflows;
- **arguments vs variables:**
 - **argument** – it stores data dynamically and passes it on, between *automations*, i.e., projects;
 - **variable** – it passes data between *activities*.

Arguments. Types. Direction

- **argument types** – similar to *variable types* available in UiPath;
- **argument direction** – it indicates *where* the information stored in them is supposed to go;
- possible argument directions:
 - **In** – data can be used in the current project only; it was sent by another project;
 - **Out** – data can be used outside the current project;
 - **In/Out** – **In** + **Out**;
 - **Property** – not currently used.

Arguments. Invoke Workflow File Activity

- **Invoke Workflow File** activity
 - allows to invoke a specified workflow, passing a list of input/output arguments (parameters);
- **Properties to set:**
 - **WorkflowFileName** - a string with the file path of the **.xaml** file to be invoked; the file path is relative to the current project folder;
 - **Arguments** - the parameters that are passed to the invoked workflow;
 - **Isolated** - if checked, the invoked workflow runs in a separate Windows process; this is useful when isolating a faulty workflow from the main workflow (invoker).



Demo 1

- Create a process that performs the following actions:
 - 1. *read* two integer numbers;
 - 2. *compute* the maximum value between the given numbers;
 - *define a distinct workflow having:*
 - *two input arguments;*
 - *one output argument;*
 - *invoke the defined workflow;*
 - 3. *print* the computed maximum value.

Generic Value Type

- **Generic Value** – special type, particular to UiPath;
 - it allows to hold values of different basic data types, e.g., text, numbers, date/time;
 - it is meant to simplify the use of basic activities;
- Advantages:
 - flexible use of variables;
 - no type considerations.
- E.g.:
 - `GenericValue A, B;`
 - `A=123; B="123";`
 - `Write Line: A+B ==> 123+ToInt("123")=246;`
 - `Write Line: B+A ==> "123"+ToString(123)="123123";`
 - `Write Line: "A+B="+A+B ==> "A+B="+ToString(123)+"123"="A+B+123123".`

Generic Value Type. Methods

- most of the methods allow to manipulate data stored using String-based methods;
- when cast is needed, other methods are available, e.g., ToString and ToInt.

Generic Value Methods

Method name	Primitive data type			
	String	Int (Whole number)	Float (Decimal number)	Boolean (True/False)
Split	✓	The integer is automatically converted to string before applying the method	The float is automatically converted to string before applying the method	The boolean is automatically converted to String: "True" or "False"
Replace	✓			
Substring	✓			
Length	✓			
Contains	✓			
Trim	✓			
IndexOf	✓			
ToUpper, ToLower	✓			
ToInt	✓	✓	Floor (Rounds down)	True -> 1 False -> 0
ToString	✓	✓	✓	"True" or "False"

Variable Categories

- **Scalar** - single value of a fixed type:
 - **Integer, Boolean, Character, Date Time**, etc.;
- **Collections** – *single dimension* structures, where multiple values are viewed as an entity:
 - **Array, List, Queue;**
 - **Strings;**
 - **Dictionary;**
- **Tables** – *two dimensional* structures, where multiple values are viewed as an entity:
 - **Data Table.**

Arrays. Details

- **Array** characteristics in UiPath:
 - **an array has a fixed length**, set at declaration/instantiation/initialization time;
 - it implements the **IEnumerable** interface ==> can be iterated by using a **For Each** activity.

Arrays. Declaration. Instantiation. Initialization

- ways to **declare/instantiate/initialize** an array:
 - **Variables Panel:**
 - **Name:** colourArray; **Type:** String[]; **Default:** {"red", "green", "blue"} //Length=3
 - **Assign** activity:
 - colourArray= new String[]{"red", "green", "blue"} //Length=3
 - colourArray= new String(2){"red", "green", "blue"} //Length=3
 - colourArray= new string(2){} //values are set later, Length=3, valid indices: 0, 1, 2
- ways to **set/change the values** in arrays:
 - **Assign** activity:
 - colourArray(0)= "red" // overrides or initializes the value on index = 0

Arrays. Example 1

The screenshot displays a UiPath Studio workflow with three steps: an 'Assign' step, followed by two 'Write Line' steps. The 'Assign' step sets 'colourArray' to 'new String(2){"red"'. The first 'Write Line' step outputs 'colourArray(0)+" "+colourArr'. The second 'Write Line' step outputs 'colourArray.Length.ToString'. Below the workflow, the 'Variables' window shows the 'colourArray' variable as a 'String[]' of 'Sequence' scope. To the right, the 'Properties' and 'Output' windows are visible, showing the 'Common' and 'Misc' property tabs and a log of execution events.

Workflow Steps:

- Assign:** colourArray = new String(2){"red"}
- Write Line:** Text: colourArray(0)+" "+colourArr
- Write Line:** Text: colourArray.Length.ToString

Name	Variable type	Scope	Default
colourArray	String[]	Sequence	Enter a VB expression

Output Log:

- Arrays execution started
- red green blue
- 3
- Arrays execution ended in: 00:00:00

Arrays. Example 2

The image displays a UiPath workflow diagram and its associated interface. The workflow consists of four steps:

- A*B Assign**: `colourArray = new string(2){}`
- A*B Assign**: `colourArray(0) = "red"`
- A*B Assign**: `colourArray(1) = "green"`
- Write Line**: `Text colourArray(0) + " " + colourArr`

The **Variables** pane at the bottom shows the following details for the `colourArray` variable:

Name	Variable type	Scope	Default
colourArray	String[]	Sequence	Enter a VB expression

The **Output** pane on the right shows the following execution log:

- Arrays execution started
- red green
- 3
- Arrays execution ended in: 00:00:00

Arrays. Example 3

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

- Assign:** `intArray = new Int32(2){1,2,3}`
- Write Line:** `String.Join(" ", intArray)`
- Assign:** `intArray = intArray.AsEnumerable().Concat({4}).ToArray()`
- Write Line:** `String.Join(" ", intArray)`

The interface includes a **Properties** pane on the right with the following details:

- Common:** DisplayName: Sequence-A
- Misc:** Private: ☐

The **Output** pane shows the following execution log:

- Arrays execution started
- 1 2 3
- 1 2 3 4
- Arrays execution ended in: 00:00:00

At the bottom, a **Variables** table is shown:

Name	Variable type	Scope	Default
intArray	Int32[]	Sequence-Array St..	Enter a VB expression

Navigation tabs at the bottom include **Variables**, **Arguments**, and **Imports**. The zoom level is set to 100%.

Demo 2

- Create a process that performs the following actions:
 - 1. *instantiate* an array of 5 integer numbers;
 - 2. *set* the values to generated numbers from 1 to 10 and:
 - 2.1. *print* the generated number;
 - 2.2. *check* if the number is even:
 - 2.2.1. if TRUE then *print* number+1 ;
 - 2.2.2. if FALSE then *print* number-1 ;
 - *use For Each activity to iterate the array;*
 - *use Log activity to print the generated values;*
 - 3. *compute* the sum of numbers in the array;
 - 4. *print* the sum;
 - 5. *print* the array;
 - *use a String-based method.*

Lists. Details

- **List** characteristics in UiPath:
 - **a list has a flexible length**;
 - it implements the **IEnumerable** interface ==> can be iterated by using a **For Each** activity.

Lists. Declaration. Instantiation. Initialization

- ways to declare/instantiate/initialize a list:
 - **Variables Panel:**
 - **Name:** weekDaysList; **Type:** List<String>;
 - **Default:** `new List(of String) from {"Monday", "Tuesday"} //Length=2`
 - **Assign** activity:
 - `weekDaysList = new List(of String)from{"Monday", "Tuesday"}//Length=2`
 - `weekDaysList = new List(of String) //Length=0, values are added later`
- way to change values already set in a list:
 - **Assign** activity:
 - `weekDaysList(0)= "Friday" // overrides the value on index = 0`

Lists. Example 1

UiPath Studio interface showing a workflow for creating and displaying a list.

Main Workflow:

- Assign** (A*B): Assigns `new List(of String) from {"Monday", "Tuesday"}` to `weekDaysList`.
- Assign** (A*B): Assigns `new List(of String)` to `aList`.
- Write Line**: Displays the text `String.Join(" ", weekDaysList)`.

Variables Table:

Name	Variable type	Scope	Default
weekDaysList	List<String>	Sequence	new List(of String) from {"Monday", "Tuesday"}
aList	List<String>	Sequence	Enter a VB expression

Properties Panel (UiPath.Core.Activities.CommentOut):

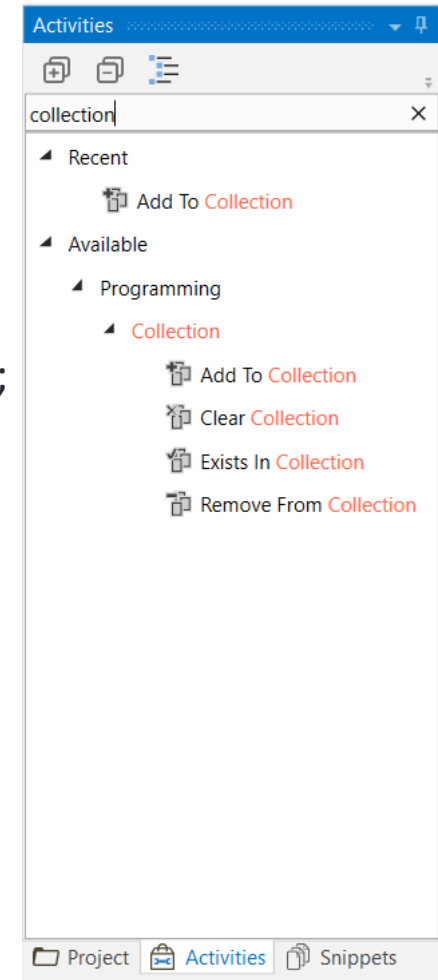
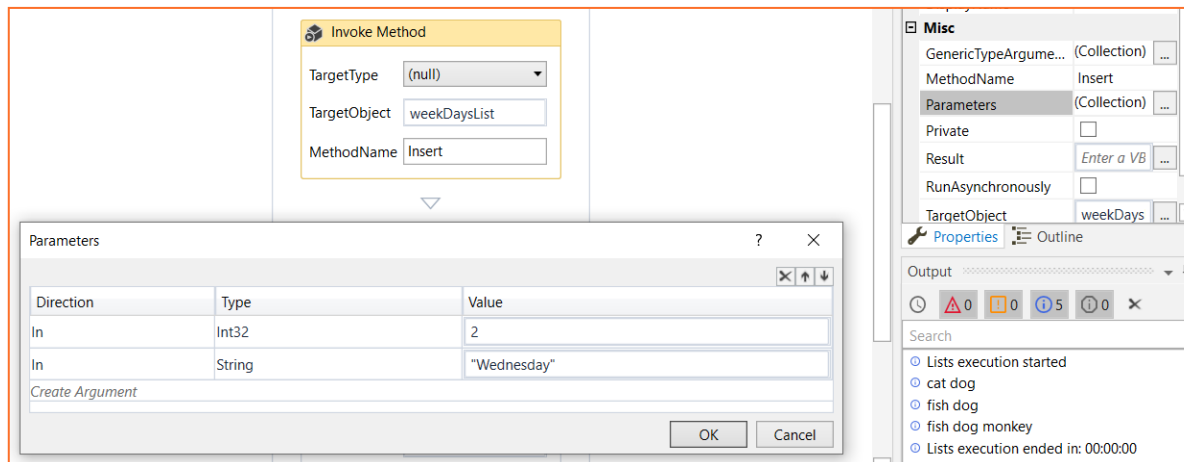
- Common**: DisplayName, Comment O
- Misc**: Private (checkbox)

Output Panel:

- Lists execution started
- Monday Tuesday
- Lists execution ended in: 00:00:00

Lists. Operations

- ways to apply predefined operations on lists:
 - Invoke Method** activity:
 - the user needs to know the signature of the use method;
 - it can be used to *add, insert, remove, etc.* an item from a list;
 - Collection** package of activities:
 - Add To Collection;**
 - Clear Collection;**
 - Exists In Collection;**
 - Remove From Collection;**



Lists. Example 2A. Set/update an item

Expand All Collapse All

System.Activities.Statements.Sequence

Common

DisplayName Sequence-Li

Misc

Private ☐

Properties **Outline**

Output

⌚ ⚠ 0 ⚠ 0 ⓘ 5 ⓘ 0 ✕

Search

- Lists execution started
- cat dog
- fish dog
- fish dog monkey
- Lists execution ended in: 00:00:00

Main

A*B Assign

aList = new List(of String)

new List(of String) from{"cat", "dog"}

Write Line

Text String.Join(" ",aList)

A*B Assign

aList(0) = "fish"

Name	Variable type	Scope	Default
aList	List<String>	Sequence	Enter a VB expression
item	String	Sequence-List Exa...	"monkey"

Lists. Example 2B. Add an item

- **Invoke Method** activity can be used to *add, insert, remove, etc.* an item to/into/from a list;

The screenshot displays the UiPath Studio interface. In the center, an 'Invoke Method' activity is configured with the following settings:

- TargetType: (null)
- TargetObject: aList
- MethodName: Add

Below the 'Invoke Method' activity is a 'Write Line' activity with the text: `String.Join(" ", aList)`.

A 'Parameters' dialog box is open in the foreground, showing the following table:

Direction	Type	Value
In	String	item

Below the table is a 'Create Argument' section with an empty text box. The dialog has 'OK' and 'Cancel' buttons.

On the right side of the interface, the 'Properties' pane shows the following settings for the 'Invoke Method' activity:

- Common: DisplayName: Invoke Method
- Misc: GenericTypeArgument: (Collection) ...
- MethodName: Add
- Parameters: (Collection) ...
- Private: ☐
- Result: Enter a VB ...
- RunAsynchronously: ☐
- TargetObject: aList ...

The 'Output' pane at the bottom right shows the following log entries:

- Lists execution started
- cat dog
- fish dog
- fish dog monkey
- Lists execution ended in: 00:00:00

At the bottom of the screen, a table lists the variables used in the workflow:

Name	Variable type	Scope	Default
aList	List<String>	Sequence	Enter a VB expression
item	String	Sequence-List Exa...	"monkey"

Lists. Example 3. Collection Package

- **Add To Collection** activity is used add an item to the list;

The screenshot displays the UiPath Studio interface. The central workspace shows a workflow with two activities: 'Add To Collection' (highlighted in yellow) and 'Write Line'. The 'Write Line' activity's 'Text' property is set to `String.Join(" ", weekDaysList)`. Below the workspace is a table defining variables:

Name	Variable type	Scope	Default
value	String	Sequence-List Exa...	"Thursday"
weekDaysList	List<String>	Sequence	new List(of String) from {"Monday", "Tuesday"}
aList	List<String>	Sequence	Enter a VB expression

Below the table is a link labeled 'Create Variable'. On the right, the 'Properties' pane shows the 'Common' tab for the 'Add To Collection' activity, with 'Collection' set to 'weekDays' and 'Item' set to 'value'. The 'Output' pane at the bottom right shows a list of execution logs:

- Lists execution started
- Monday Tuesday Thursday
- Monday Tuesday Wednesday Thursday
- Tuesday Wednesday Thursday
- Lists execution ended in: 00:00:00

Demo 3

- Create a process that performs the following actions:
 - 1. *create* a list of the following items: {"still water", "sparkling water", "tea", "coffee", "wine", "juice", "green tea", "black tea"};
 - 2. *generate* an index value (from 0 to 7);
 - 3. *print* the message "*Do you like to drink some juice/green tea/...?*" and enter the answer (*yes* or *no*):
 - 3.1. if "YES" move the item to the front of the list;
 - 3.2. if "NO" move the item to the end of the list;
 - 4. *repeat* step 3. *four* times;
 - 5. *print* the bill, i.e., the list of accepted drinks.

References

- UiPath Academy - <https://academy.uipath.com>
 - Awareness Training;
 - Level 1 – Foundation Training, Lesson 3;
- UiPath Docs - <https://docs.uipath.com/studio>
 - Arguments - <https://docs.uipath.com/studio/docs/managing-arguments>
 - Invoke Workflow File Activity - <https://docs.uipath.com/activities/docs/invoke-workflow-file>