WITHYOU
WITHME

RPA Blue Prism Developer Course

Written Guide

Contents

Intent

This document is designed to augment your online learning experience and provide you with the opportunity to review and revise the content that has been discussed during the individual lessons of the course

Feedback

Please contact your WithYouWithMe RPA instructor if you have feedback on this document or any of your WYWM courseware.

VBOs and IBOs

Overview

Hi and welcome to the lesson where we will be discussing VBOs and IBOs

In this lesson we'll build on our knowledge of VBOs and IBOs and walk through some more Blue Prism features that will help us do some more advanced and useful work when interacting with applications.

As usual we'll switch between the slides and Blue Prism. To get the most out of the video, make sure you follow along in Blue Prism and pause the lesson if you need to.

Details of VBOs

So far, we've learnt how to interact with applications using the object studio and application modeller to build business objects. As we've said, the objects that we've built are referred to as visual business objects or VBOs. The role of a VBO is to act as an adapter to the user interface within a specific application.

When you're building your VBOs in the application Modeller lesson, you were unwittingly developing VBOs with the following components.

A connector, which is a standard library, provided by blue prism for communicating with a particular type of application user interface.

An application control interface (or ACI), which uses the VBO's connector to expose a specific application's user interface. This was the spying tool.

And finally one or more pages , each of which implements all or part of an operation that the VBO can perform.

Whatever the interface technology, the adapter used to connect to an application is called a Visual Business Object. Each VBO implements a particular set of operations against an application's user interface. For example, a VBO might be capable of launching an application, logging in, entering a customer name to a particular screen, retrieving a result and logging off.

Blue Prism's VBOs

As we've said, Blue Prism has inbuild support for connecting to a range of common applications that include browser based HTML applications, Windows interfaces, mainframe applications accessed via terminals as well as interfaces using java.

Blue Prism has in built support and this support is provided by the list of VBOs that are available in the actions stages. You can refresh your knowledge of this by going into Blue Prism, opening a process and getting an action stage. If you select the properties and click the business object drop down.

In this drop down you can see visual business objects as well as Internal business objects. As we've mentioned before, Visual business object are those that we can manipulate and interact with in the object studio. Whereas internal business objects are used for things like manipulating collections and work queues internal to Blue Prism and these can't be manipulated.

A final point on VBOs is that we can import more as needed by clicking the file button in the top left corner of the screen, selecting import object and browsing behind the following link. C Drive - Program Files - Blue Prism Limited - Blue Prism Automate - VBO.

If you navigate to this address you'll see that there are a whole lot of addition VBOs that you can import and apply to applications as needed. Depending on the Blue Prism environment you're working in, you may need to do this now. If you can't see the MS Excel VBO in your Business Object drop down menu, you'll need to import your MS Excel VBO now.

While you're checking that, it's also worth noting that there are also a miyad of additional VBOs available online including custom VBOs created by other users. A word of caution though, the quality and reliability of custom made VBOs does tend to vary dependant on the individual that created them.

Automating Microsoft Excel

Overview

Hi and welcome to the lesson where we will be automating Microsoft Excel

In this lesson we'll build on our knowledge of the process studio, the object studio and in particular the application modeller to do some useful work in Microsoft Excel using VBOs.

As usual we'll switch between the slides and Blue Prism. To get the most out of the video, make sure you follow along in Blue Prism and pause the lesson if you need to.

Microsoft Excel

Microsoft excel is the recommended spreadsheet software for enterprise RPA. MS Excel is extremely capable and widely used by many organisations across the globe. It features extensive application programming interface functionality and Blue Prism provides a comprehensive API option in the form of the MS Excel VBO within Blue Prism.

This list of MS Excel features provided by VBOs can be accessed through the same action properties that we've used previously while using action stages. In this lesson we'll cover some general best practices in relation to the MS Excel VBO as well as other performance and application tips.

For more detail on the MS Excel VBO and it's actions, you can click the blue i button adjacent to the Business object drop down list within the properties window of the action stage. This i button will take you to an internet explorer page that will give you a bit more information about the business object you've selected and it's associated actions.

Automating Microsoft Excel

Ok, so let's jump across to Blue Prism and use the skills we've learnt thus far to start automating using MS Excel.

So we're going to create a process that looks like this and will use the MS Excel VBO to Create an instance of Microsoft Excel, create a workbook using that instance, Blue Prism will then shows us what it's doing, it'll save our workbook and then closes the instance.

There's really nothing new here so I'm not going to talk through it, you can just follow along, making sure that you're selecting the correct actions within our MS Excel VBO as they correspond to the action that we want carried out.

So now we're in Blue Prism we create our Excel Process and when we get to the main page we start setting it up

You would have noticed that when I was setting up my actions stages I also created some corresponding data items that were required due to the inputs and outputs of my action. It's a good time to get into the habit of always checking the properties window of a stage for the inputs and outputs that are required, especially action stages.

In this case a few of my action stages asked me for my handle as either an input or an output. When my create instance action stage is run it produces an output called handle and it gives it a number value. The handle is used by Blue Prism to identify a created instance of excel and each time an instance is created it will have a different handle number. Blue prism can then use this handle in subsequent actions to ensure it's working on the correct instance of Excel.

After we've created an instance we create a new workbook in excel and all Blue Prism needs to know here is the handle, which is an input. Once it's created a new workbook and given it a name, that name will become an output and will get stored in the data item we created. You may have noticed that I just clicked this blue button when creating the data item. This is a quick and easy way to create data items without needing to drag and drop every time.

I could have also selected the Open workbook action within the MS Excel VBO and this would have allowed me to open a workbook, but I would have needed to give Blue Prism a file path for the workbook that I would have liked to open.

I put a show workbook stage in there as well so we can see what's happening. It's worth noting that I don't actually need to include this stage as Blue Prism will interact just fine with excel below, without showing us what it's doing. This is called operating below the Graphical user interface (or GUI) layer.

And our save as and close instance stages are fairly self explanatory. So let's reset and run our process to see how it works.

Great, as we cycle through you would have noticed our handle data item gets populated as well as our workbook name and the new workbook.

This process is designed to show you some of the functionality of our MS Excel VBO but be sure the pause the lesson here and experiment with the MS Excel VBO and the range of actions that are available.

We can also attach and detach from excel or a particular workbook, we can get a worksheet as a collection (fast) which pulls the active worksheet within the current version of excel and puts it in a collection, we can write a collection to a workbook using the write collection action, we can import a csv file using the import csv action, we can go to a particular cell using the go to cell action and as we saw we can save workbooks or save workbooks as.

Make sure you have a play with some of these actions in your version of Blue Prism, they will come handy later in the course and during your assignment.

Managing Performance

Finally we'll discuss managing performance. The performance of excel automations using the MS excel VBOs can vary depending on how the object is implemented and executed. For example, displaying the workbook on the screen has benefits in that we can see clearly what's going on which aids development and testing, but it is not necessary when our process is in the production environment and having our process show us what it's doing induces an increased load on our computer's processor. To improve performance it's a good idea to remove any show action stages prior to a bot being deployed into the production environment.

Data security should also be considered and displaying a workbook on a screen is one less barrier to a data breach, so for this reason as well its best practice not to show a workbook while running an excel process in the production environment.

As you'll be aware, the speed at which a process runs and interacts with objects varies significantly between the control room and the process studio. This is especially the case when using the MS excel VBO.

For this reason, performance testing should be carried out in the control room, where the process will be run during normal business operations, and not in the process studio where we tested the functionality of our bot. This ensures that we're measuring peak performance as opposed to degraded performance in the process studio.

Some memory issues can occur when working with large amounts of data, especially if this data is being pulled from an application like MS Excel and stored in a collection. We can reduce the memory usage load by minimising the data in a collection where possible. We can use an action called "Get workbook range as collection" which will only pull data within a specific range within a workbook rather than all the data available.

We should also look to avoid producing copies of the same collection or copies of the data within it. This will also aid in reducing memory usage problems.

Environmental Variables

Overview

Hi and welcome to the lesson where we will be discussing Environmental Variables

In this lesson we'll build on our working knowledge of how Blue Prism processes work in industry by introducing the concept of environmental variables and demonstrating their use in the processes we've built so far.

As usual we'll switch between the slides and Blue Prism. To get the most out of the video, make sure you follow along in Blue Prism and pause the lesson if you need to.

Overview of Environmental Variables

Environmental variables are used in most processes in industry so it's important that we have an understanding of how we apply them to our processes.

First let's refresh our understanding of a variable. In terms of a Blue Prism process, a variable is a value that is considered dynamic, and by this we mean it will change on a frequent basis throughout the life of our process.

Examples of variables that we've looked at so far are usernames and passwords, but other variables could include a URL of a browser based application, a file path to collect a file, a file path to save a file or an email address that is used to notify a user.

The examples we just listed are know as Environmental Variables because they relate to the environment in which we're working.

Environment variables should be used to store any process or application information required to work the business process that may be subject to change either between environments (development/test/production) or over the life of the process.

The advantaged of environmental variables is that they can be created and maintained by a developer with elevated access privileges and once created they can be used and selected by entry-level developers that may not have the same access privileges. This also aligns with the IT security and risk management frameworks that you will find in many organisations.

We use environment variables so that minor changes to the configuration of a solution can be made without the need to make development changes to the objects or processes.

Put simply, you can think of environmental variables in much the same way as we think of inputs to a process. The only difference being that we can set an environmental variable and it will continue to be

inputted until we change it, as opposed to inputs to a process that need to be provided each time a process commences when run from the control room.

You can imagine that if you are a process controller in industry, you wouldn't want to be forever inputting usernames and passwords or the number of loops each time you create and run a session. You'd want this information to be stored somewhere and only changes as the need arose. This is precisely why we use environmental variables as opposed to inputs in the vast majority of cases.

Using Environmental Variables

So let's jump across to Blue Prism and use our notepad object and process to write some text in Blue Prism, but this time let's input the text using an environmental variable rather than inputting it directly into a data item on the process or object layer.

First let's open our Notepad process and step through our Notepad process to familiarise ourselves with what this process does.

So we can see that our process will launch, write to and close notepad and we've also got our data item over here. We have this data item on our process layer and at the moment it has a message that we've going to send down to our object layer, via in input to our write action.

If you need to refresh yourself with how this works, make sure you take the time to review the inputs and outputs lesson.

Ok, so let's reset and step through our process, first you can see that we go into our launch action,,,, then back up to our process layer, then down to our write action and you'll note the value being inputted to our data item when we step off the stage stage, we write our text to notepad from the value in the data item we just sent down to our object layer, when then go back up to our process layer before heading back to the object layer and our terminate action, where notepad is closed.

Great, we're pretty comfortable with how all that works but imagine if the message that we want to write has changed, or if the password to an application has changed, or a file path has been amended. We could updated this in our data item on our process layer each time but if we're always going in and out of our processes to make minor changes we increase the likelihood of errors, and if our bot is running in a high reliability environment we want to minimise the likelihood of errors being created as much as possible.

Another consideration is that depending on the company that you're working for, any changes, including something as minor as a password update, may require our process to be reassessed and reapproved. This process would be time consuming and a waste of resources so we try and minimise any changes to our process where possible.

This is where environmental variables come in to play. We mentioned before that we can use an input to a process to send information from the control room, inputted by our process controller, into our process

layer. The only problem with this is that it requires our process controller to have all the information they need at their disposal which is not always possible and is definitely not scalable.

So instead, we can create an environmental variable and have that information stored in a data item. Better yet, we can store an environmental variable in multiple data items in multiple process. Let's go through that now.

First we need to create an environmental variable and we do that in the system tab. We them go to processes and click on environmental variables. You can see I have some environmental variables in here already, but to add another I can simply click add variable and I need to fill in some basic details similar to what's asked in a data item.

I'll give it the name Notepad Message, it'll be a text data type as we know, I'll give it a brief description,,,,, and I'll input the value that I want my data item to store, this case, let's make it a file path that is likely to change on a fairly frequent basis. C:\Windows\System32\notepad.exe

Once that's done I click apply.

Before we leave this page you'll notice that I can also highlight and remove variables if I want to, and I can also look at where these other variables are referenced and it will tell me each of the processes that they're used.

Great, so let's go back into our notepad process and use the environmental variable called Notepad message we just created.

To do this we can open the data item that we were using before and to select the environmental variable we'd like to store in this data item we simply select the exposure drop down box. In this list we can see that we have the option to select Statistic which allows us to store the contents of a data item in a database, We have environment which allows us to store an environmental variable in this data item, and we have session which allows us to display the value of this data item in the control room.

Because we just set up an environmental variable we'll select this.

When we select Environment you will notice that the name of this data item changes and now includes a drop down menu. This is where we select the environmental variable we'd like to store in this data item. As you can see they look pretty familiar.

Now we'll select Notepad message and you can see that our data type and initial value field are pre populated and greyed out. This data item is now displaying exactly what we put in our environmental variable.

So let's click ok, reset our process, run it again and note that the message has changed on our notepad.

Great. Now if I want to change the text that's written, I can do this without the need to go into my process.

Let me demonstrate by saving and closing this process. Going into the system tab and adjusting the text of my notepad message, clicking apply and now if I check back with my notepad process I can see that when I open my data item the message has changed to reflect the update to my environmental variable.

RPA Solution Developer Documentation

Overview

Hi and welcome to the lesson where we will be discussing RPA Solution Development Documents

In this lesson we'll be discussing the documents that play a role in RPA solution development at a high level so we have an understanding of what's required from a documentation perspective when developing automations and deploying them into the production environment.

This lesson is also supported by a Blue Prism solution and associated delivery documentation that you can reference in the materials tab of this lesson. The delivery documentation utilises three Blue Prism templates that illustrate the content and details required at each stage of delivery. In the real world, an enterprise's local Blue Prism framework will prescribe the documentation and templates required when you're out in industry.

Documentation Overview

The three templates in this course are the solution design document, (or SDD), the process design instruction (PDI), and the object design instruction (ODI).

As we know, a Blue Prism process need to be much more than a series of manual steps executed automatically. It needs to be robust enough to work unattended and be able to recover from unexpected situations and work concurrently on multiple machines. This key element of development is often missed when students learn to design and develop bots but is absolutely critical when ensuring the overall robustness of an automation.

Another common oversight when designing a process is to assume that it will never encounter an real problems, and that it will stay on the happy path. Again, this is unrealistic and recover steps should always be included to cater for a process flow that takes the unhappy path.

The term defence coding describes a way of coding where the develop is always expecting and planning for things to go wrong and ensures that the final product is a resilient and robust robot that is capable of dealing with all exceptions that it will encounter.

Another key tennant of solution development is to build working components that combine the building blocks in a bottom up approach. Objects should be free of business process logic and object pages should be small and execute simple and discrete tasks. The function of an object is to provide processes with the tools to manipulate an application and the more generic the object logic is, the easier it will be to test and increasing the likelihood of it being reused in other processes.

Note that we will dive deeper into this concept in the Blue Prism Excellence course, in particular the building for scalability lesson.

The SDD

The Solution Design Document (or SDD) will complement the Process definition document (or PDD) and describe how the process we've developed will successfully automate the process described in our PDD.

The SDD is therefore created by the development team but it is created for the business process owner the IT department and will typically have the RPA Analyst assisting from a project management perspective. More detailed information about the roles and responsibilities within an RPA team is available in the WYWM RPA Analyst course.

The SDD is a comprehensive document containing not only high level details of how the RPA solution will automate a process but also includes the details of other deliverables that are required such as web services, database tables, webforms etc.

The SDD will also include other details such as security, scheduling, alerting and management information. The SDD is intended to convey to the business, the details of the automated process so they understand the working of our robot and ultimately approve it's deployment into the production environment.

Critically, the SDD should not go into the granular details of the developed solution, as this level of details will likely create too much ambiguity give the proposed reader will typically not have any RPA development experience.

The granular detail that we omit from the SDD should be included in the Process Design Instruction (PDI) and the Object Design Instruction (ODI).

Finally, the SDD includes the following sections an you can review these in the template provided in the material tab of this lesson.

It has a Solution Overview, which provides and overview of the end to end solution
It has an object model which provide an overview of the objects that will make up our solution
It includes Operating control and alerting processes
Data security and credentials information that annotates how sensitive data is handled
And finally we have business and technical assumptions which should be validated as soon as possible after the SDD is produced and approved.

To provide some additional details we include a solution diagram similar to what we saw in the PDD. This diagram provides some more details about the critical elements of our solution. Unlike the process diagram in the PDD however, the SDD is written and developed from an RPA perspective and starts to

include reference that annotate how the solution will be structured in terms of work queues, exceptions etc.

We also include an object model diagram as part of our SDD. This is included in chaper three of our template and shows our process's components and object as well as any objects and external web services that will be called on during our solution's operation. It's worth noting that this level of detail is not always available at the time the SDD is produced and this may need to be added to in future.

The PDI

The next document is the PDI. This document is a means of informing the development team on how and what to build and is completed by the solution designer. The PDI will describe in detail a single process and the components, business objects, work queues and credentials it uses to support the overall solution.

The PDI is essentially a blueprint from which the action solution is developed. The granular detail of the solution that was excluded from the SDD should be included in the the PDI, as the intended reader of this document will have RPA development experience and will be the person building the automation in Blue Prism.

It's worth noting that the PDI is generally dynamic but an initial PDI should be completed prior to the commencement of development to avoid scope creep during the course of the RPA delivery process. For this reason accuracy and document version control should be maintained to a high standard.

The PDI sits within the remit of the design and development team and includes a process diagram, which if produced correctly will look very similar to the actions and stages we see in the process studio.

It has a process description which related back to the process diagram and uses all the same terminology. It will also include data that articulates the details of work queues, sessions and environmental variables. And it provides details on referrals and exceptions that have been planned for.

Again, an example PDI and a templated PDI is available in the materials tab of this lesson.

The ODI

Finally, the Object Design Instruction or ODI is created as a blueprint from which business objects are developed. From this perspective the ODI and the PDI serve a very similar purpose, however where the PDi referred to the development of a process, the ODI specifies how the objects that are being called upon during that process are created.

The ODI will specify how to build each object and each action within that object, a list of input and output parameters will also be specified. The ODI is typically created in the form of a spreadsheet, specifying

the name of the object, the required action stages and the associated inputs and outputs to each action. It will also have a section on notes for some additional information to be annotated.