

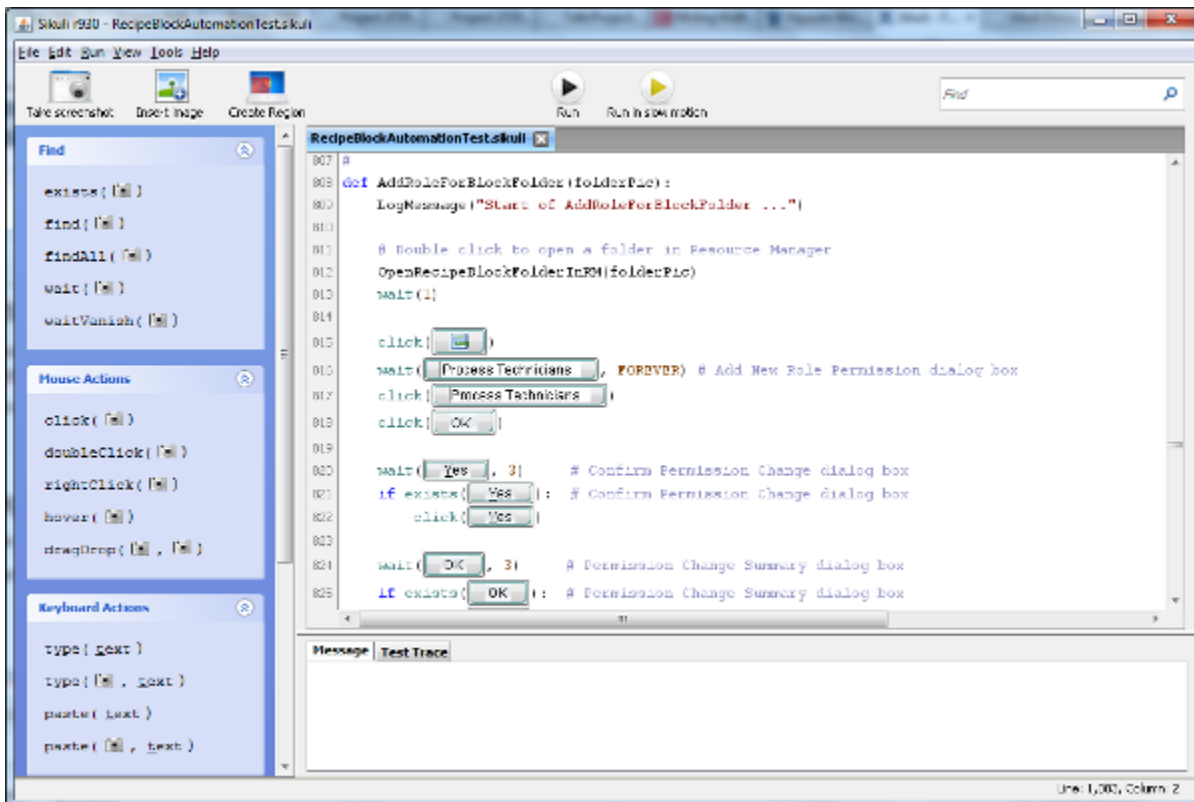
Sikuli tutorial

- Introduction
- Homepage and download:
- Sikuli Script
 - Write a Sikuli script
 - Using the Sikuli IDE
 - Pros and cons
 - Running a Sikuli Script
- Some practices
 - Hello World
 - Uncheck All Checkboxes
- Tips
- References
 - How-To: Sikuli and Robot Framework Integration

Introduction

Sikuli is a visual technology to automate graphical user interfaces (GUI) using images (screenshots). Sikuli Script automates anything you see on the screen without internal API's support. You can programmatically control a web page, a desktop application running on Windows/Linux/Mac OS X.

The new version SikuliX-1.0.0 is available at the time of this post.



Homepage and download:

The homepage of SikuliX: <http://www.sikuli.org>

Download page: <https://launchpad.net/sikuli/+download>

Sikuli Script

Sikuli Script is built as a Jython (Python for the Java platform) library. You can use any syntax of the Python language. If you are new to programming, you can still enjoy using Sikuli to automate simple repetitive tasks without learning Python. A good start might be to have a look at the [tutorials](#).

However, if you would like to write more powerful and complex scripts, which might even be structured in modules, you have to dive into the Python Language.





Write a Sikuli script

You write Sikuli scripts using its python API:







- by hand
- or in the Sikuli IDE.

Using the Sikuli IDE



Find

`exists()`
`find()`
`findAll()`
`wait()`
`waitVanish()`

Mouse Actions

`click()`
`doubleClick()`
`rightClick()`
`hover()`
`dragDrop( , )`

Keyboard Actions

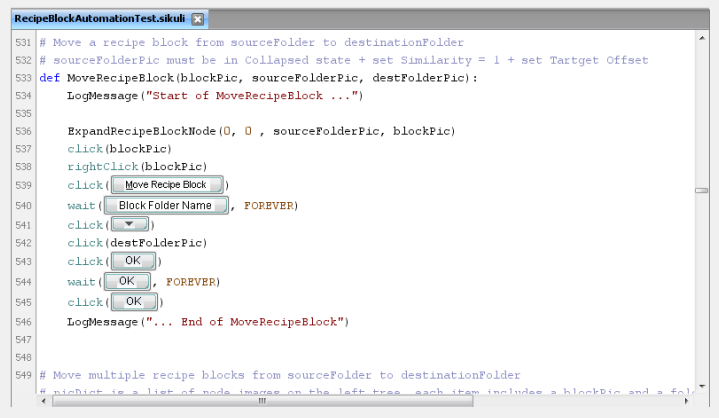
`type(text)`
`type( , text)`
`paste(text)`
`paste( , text)`

Left column: Sikuli commands, coupled with optional image capture

Most buttons do this:

- Help capture a reference image
- Insert the image and the Sikuli command into the script

- Right column: Your Sikuli script, in a special editor
- Use as a text editor
- Left column's buttons inject code at the cursor



```

RecipeBlockAutomationTest.sikuli
531 # Move a recipe block from sourceFolder to destinationFolder
532 # sourceFolderPic must be in Collapsed state + set Similarity = 1 + set Target Offset
533 def MoveRecipeBlock(blockPic, sourceFolderPic, destFolderPic):
534     LogMessage("Start of MoveRecipeBlock ...")
535
536     ExpandRecipeBlockNode(0, 0, sourceFolderPic, blockPic)
537     click(blockPic)
538     rightClick(blockPic)
539     click([Move Recipe Block])
540     wait([Block Folder Name], FOREVER)
541     click([v])
542     click(destFolderPic)
543     click([OK])
544     wait([OK], FOREVER)
545     click([OK])
546     LogMessage("... End of MoveRecipeBlock")
547
548
549 # Move multiple recipe blocks from sourceFolder to destinationFolder
550 # picDict is a list of node images on the left tree, each item includes a blockPic and a fol

```

Pros and cons

- Sikuli: awesome, but has rough edges
- IDE is best for rapid prototyping
 - not all Sikuli commands are in the IDE's left column
 - python indent issues
 - no parser warnings/errors
 - no UNDO
 - can be unstable
- Sikuli "files" are actually directories named scriptname .sikuli, containing your .png images, the scriptname .py and scriptname .html
- When loading/saving in the IDE, load/save the directory name, not scriptname .py

Running a Sikuli Script

You run Sikuli scripts in two ways:

- Click Run button in the IDE
- ***sikuli-ide.sh -r ./your_script_name.sikuli***

Invocation of Sikuli script, Example1:

sikuli-ide.sh -r ./unlock_emulator_and_run_browser.sikuli

To run the IDE and load a script in one action:

sikuli-ide.sh ./scriptname.sikuli

Some practices

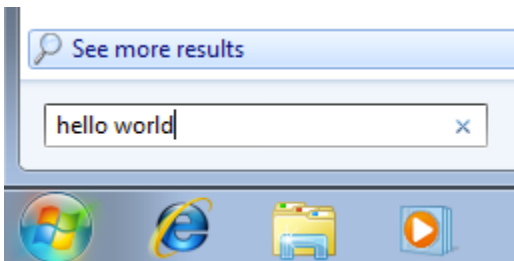
Hello World

Let us begin with a customary Hello World example!

You will learn how to capture a screenshot of a GUI element and write a Sikuli Script to do two things:

1. Click on that element
2. Type a string in that element

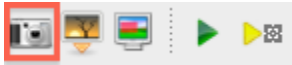
The goal of the Hello World script is to automatically type "Hello World" into the *Start* menu search box, like this:



Now, open the Sikuli IDE. We begin by taking the screenshot of our target, the *Start* menu symbol that is usually located in the lower-left corner of the desktop. Using this screenshot, we can tell Sikuli script what to click on.

To simulate a mouse click on the *Start* symbol, we are going to use the `click` function. To tell Sikuli how the *Start* symbol look like, we need to capture its image on the screen.

Sikuli IDE provides two methods to capture screen images. The first method is to click on the camera button in the toolbar. This will bring you to the screen capturing mode.

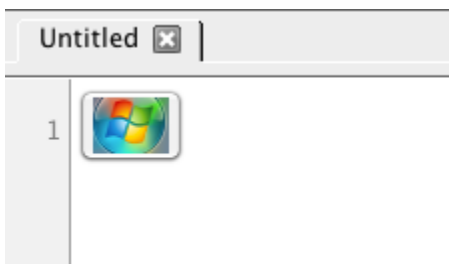


The second method is to press a hot-key (Ctrl + Shift + 2). Often the target whose image you wish to capture may be covered by the Sikuli IDE's window. You can minimize the IDE's window and use this hot-key to switch to the capturing mode.

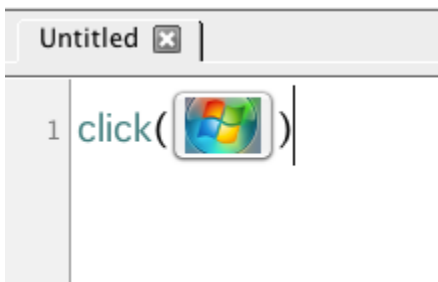
In the screen capturing mode, the screen will look darker and freeze momentarily. The entire desktop becomes like a canvas where you can draw a rectangle around the target you want to capture an image of. In this case, the target is the *Start* symbol. The cross of red dotted lines shows the center of the rectangle you just drew.



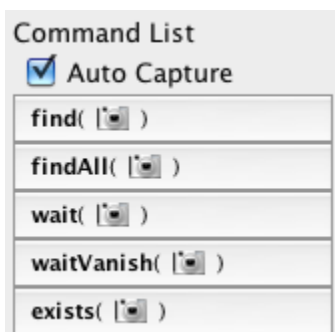
After you have drawn (or selected) a rectangle, the image within the rectangle will be captured and inserted into the script editor at the current cursor position.



Now, you can write the click function using this image as an argument to tell Sikuli to click on start symbol.

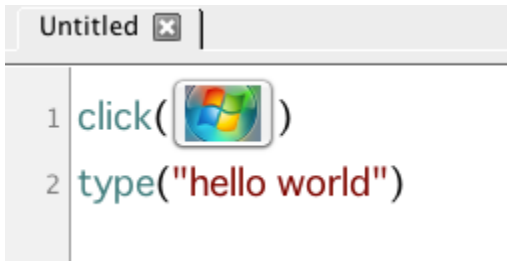


As a convenience, Sikuli IDE provides a *Command List* on the left panel. It shows a list of the most often used functions. Camera icons in the functions indicate these functions expect a captured image as an argument.



Locate the `click()` function in the list and click on it. If **Auto Capture** is on (default), you will be directed to the screen capturing mode in which you can capture an image of an interface target to be inserted into the `click()` function as an argument.

The next step is to tell Sikuli to enter the string "Hello World" into search box, which can be done with a simple `type` function.

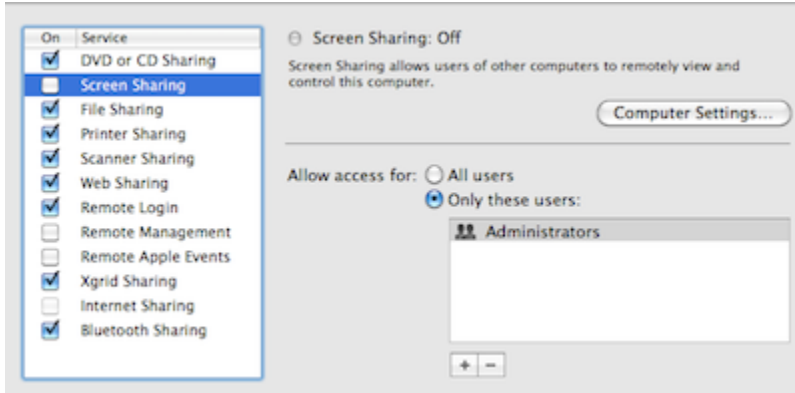


This function will type the string given in the argument into whichever input control that has the focus. After clicking on the *Start* symbol, we can expect the search box will be the input that has the focus.

Congratulations! You have just completed your first Sikuli Script. Press the run button to see this script in action!

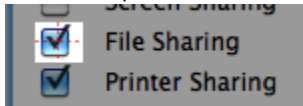
Uncheck All Checkboxes

In this tutorial, we will demonstrate how to use a for loop to interact with multiple instances of a GUI component. Suppose we want to uncheck all the check boxes in a window, such as the Sharing preferences window shown below:

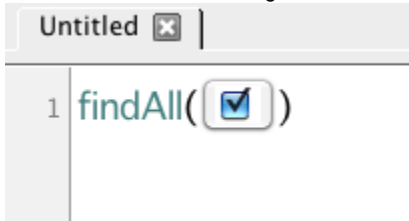


Unfortunately, there is no "uncheck all" function available. The solution? Write a Sikuli Script to look for ALL the checked items and uncheck them automatically. The function needed for this operation is `findAll()`.

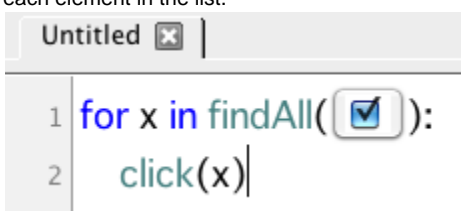
First, let's capture the screenshot image of a checked item.



Then, we can insert the image into the `findAll()` function.



`findAll()` searches the entire screen for all the matching visual patterns and returns a list of locations of those similar patterns. This capability allows us to obtain all the checked items are on the screen. Then, we can simply write a for loop in standard Python syntax and call `click()` on each element in the list.



When this script is executed, Sikuli will find all the items that are currently checked and click on each item one by one in the loop.

Tips

During using the powerful GUI test tool, I faced some uncomfortable problems. I would like to propose some tips to overcome them:

1. Avoid being eager to write a very long scenario. This would be a disaster at your test execution if there is a fault occurred.
 2. The entire automated test scenarios should be split into small segments (or methods). This can help save your time during test writing and make it easy to trace your test execution as well as maintenance.
 3. A new log should be added each time you get in/out of a method.
 4. Avoid looking for dialog box titles which depends on OS theme. Instead, looks for specific text on the dialog box.
-

References

How-To: Sikuli and Robot Framework Integration

<http://blog.mykhailo.com/2011/02/how-to-sikuli-and-robot-framework.html>