Managing processes (SystemUtil)

The `SystemUtil` object is a wrapper that provides a few methods to launch and terminate processes, and to block and unblock user input. the following sections, we will describe these methods and explain hov they work.

Getting ready

From the **File** menu, navigate to **New** | **Test**, or use the *Ctrl + N* shortcut.

How to do it…

Proceed with the following steps:

1. `Run`: It invokes an application with parameters. There is an optic to define the work directory, operation, and mode at the openin (maximized and so on). The syntax is as follows:

```
SystemUtil.Run file, [params], [dir], [op], [mode]
```

   For example:

```
SystemUtil.Run "iexplore.exe", "google.com",,, 3 'Show maximized
```

   By `operation [op]`, it means which action is to be performed with the file supplied as a string argument with the possible values of open, edit, explore, find, or print. If omitted, the open action is performed. Sending edit with the pathname to a text file will open the default text editor (by default, `Notepad.exe`); explore will open Windows Explorer at the given path; find will open Windows Explorer in search for the files that match the pattern; and finally, print will send the file to the default printer. Of course, correspondingly, if the file is not editable or printable, the statement fails. Using open with a nonexecutable file will open it in its default associated application.

2. `CloseDescendentProcesses`: This closes all processes that were launched by UFT during the run session. It returns the number of processes terminated:

```
SystemUtil.CloseDescendentProcesses()
```

3. `CloseProcessByName`: This closes all processes with the name passed as an argument. It returns the number of processes terminated. The syntax is as follows:

```
SystemUtil.CloseProcessByName process_name
```

   For example, to close all IE browsers, we may use the following code:

```
i = SystemUtil.CloseProcessByName("iexplore.exe")
```

4. `CloseProcessByWndTitle`: This closes all processes that launched windows with the title passed as an argument. It returns the number of processes terminated. The syntax is:

```
SystemUtil.CloseProcessByWndTitle window_title
```

   For example, to close a Notepad window with an open file named

`MyFile.txt`, we will use the following:

```
i = SystemUtil.CloseProcessByWndTitle("MyFile.txt")
```

To close all notepad windows with `MyFile_` as a prefix in the title and with the additional variable text that follows, we will use a regular expression and indicate that this is the case by sending `True` as a second argument:

```
i = SystemUtil.CloseProcessByWndTitle("MyFile.+\.txt", True)
```

5. `CloseProcessByHwnd`: It closes a process that launched a window with the handle (`hwnd`) passed as argument. It returns `true` if found and closed, or `false` otherwise. The syntax is as follows:

```
SystemUtil.CloseProcessByHwnd window_handle
```

For example, to close a window for which we know its handle, we will use code similar to the following:

```
SystemUtil.Run "notepad.exe"
hwnd = Window("regexpwndtitle:=.*Notepad").GetROProperty("hwnd")
print hwnd
SystemUtil.CloseProcessByHwnd(hwnd)
```

6. `CloseProcessById`: This closes a process with a specific ID. It returns `true` if found and terminated, or `false` otherwise. The syntax is:

```
SystemUtil.CloseProcessById process_id
```

We can retrieve the process ID for a given window and then terminate its associated process as follows:

```
SystemUtil.Run "notepad.exe"
pid = Window("regexpwndtitle:=.*Notepad").GetROProperty("process id
print pid
SystemUtil.CloseProcessById(pid)
```

7. `BlockInput`: This disables the keyboard and mouse for user input during the run session, in order to avoid an accidental interruption by a user. Input is resumed when the run session ends or is paused (for example, runtime error and breakpoint); the *Alt* + *Ctrl* + *Del* combination is pressed or else a critical system error will occur. The syntax is as follows:

```
SystemUtil.BlockInput()
```

8. `UnblockInput`: This re-enables the keyboard and mouse for user input during the run session. The syntax is as follows:

```
SystemUtil.UnblockInput()
```