



DT ALM Newsletter; Issue 7

Introduction

In this issue of the newsletter we are going to take a look at Borland SilkTest and some of the features it has.

What is SilkTest?

SilkTest is a test automation tool which enables you to perform functional tests on your applications via their Graphical User Interfaces (GUI's).

Why would I want to use it?

By automating your functional testing you would be able to do either of the following (or a little of both):

1. Reduce the costs associated with testing
If you have a dedicated resource for manually testing applications via their GUI's then SilkTest could significantly reduce the amount of resource required.
2. Increase the coverage of your testing to improve quality
If you feel that you do not have significant resource to comprehensively test your applications, SilkTest can help you get more done with the resource you have at your disposal.

How do I use it?

SilkTest has a simple user interface that makes it very easy to create new test projects.



The idea is that by working from left to right with the large buttons shown above, you can get testing within minutes. The open project option is fairly self-explanatory (if your project doesn't exist it will allow you to create a new one) but let's explain what the rest do:

Enable Extensions – This is how you tell SilkTest which application you want it to test. It will supply you with a list of all the applications that are currently running so providing your application is running you simply select it from the list.

Set Recovery System – The Recovery System is a useful feature of SilkTest that will return your application to a state if a test fails so that subsequent tests can be run. Whilst SilkTest sets its own base state, you can define your own states to save having to perform similar actions for a large number of tests (e.g. login).

Record Testcase – Once the Recovery System is set, you are then able to record testcases. Once you have started recording, simply test your application and SilkTest will record the actions you take. All you then need to do is stop the recorder when you have finished your test.

Run Testcase – If you elect to run the testcase you have just recorded, SilkTest will perform a carbon copy of the actions you undertook whilst the recorder was on.

Explore Results – After a testcase has been run, the results are displayed by default. However you can use the explore results button to look at results files from other testcases.

Using 4Test

Whilst the six steps detailed above are enough to get you up and running, if you want to produce a reusable set of automated tests, you will need to do a little more work. All code in SilkTest works in its own scripting language called 4Test.

Unlike other scripting languages, 4Test has been designed specifically for writing test scripts. However if you prefer not to use 4Test, SilkTest will also work with C++.

Consider a simple login testcase. If you were to use the recorder it would record the following actions in your testcase:

1. Username = "MyUser"
2. Password = "MyPassword"
3. Click "Login"

Suppose you wanted the testcase to be more flexible. You could modify the script so that it takes two parameters for Username and Password. You could then use your login testcase for multiple user accounts. This would be useful if you wanted to test your application's permission settings, and needed to use a number of accounts to ensure that the permissions are set accordingly.

Interaction

Whilst some automated test suites are co-ordinate based, SilkTest interacts with your application at the object level.

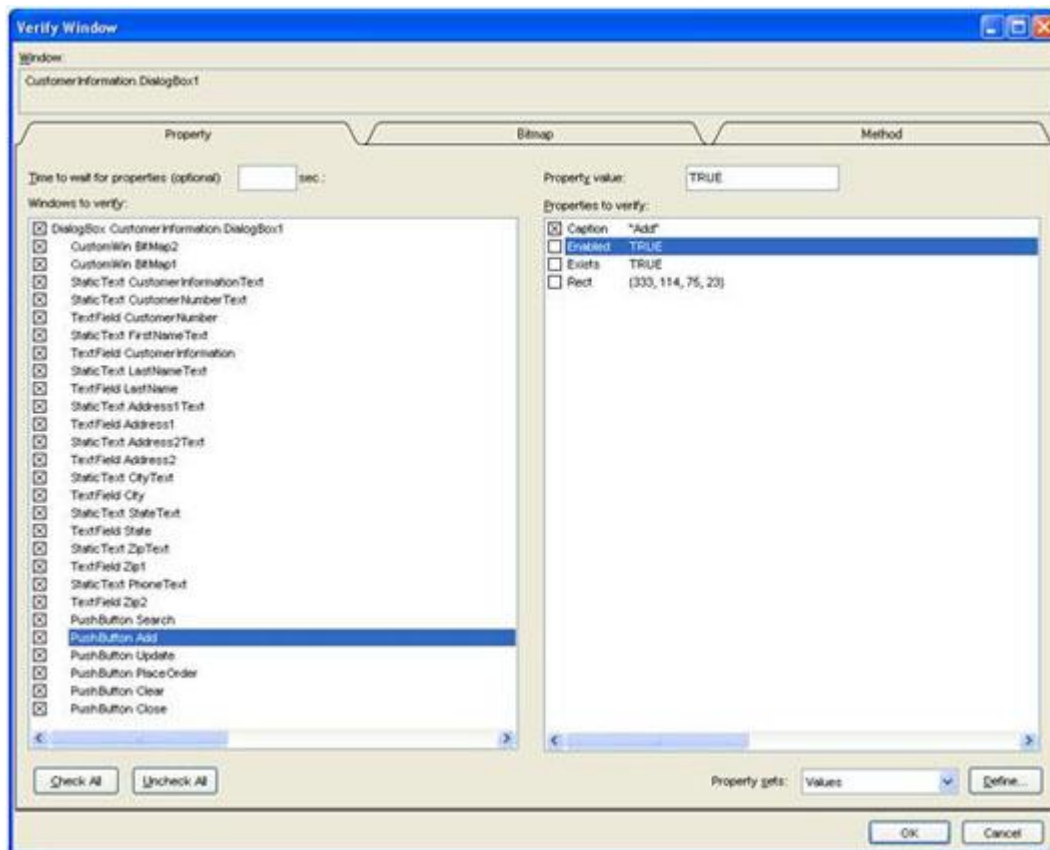
This means that if you were to move or resize a button for example, provided you haven't renamed it SilkTest would still be able to find it and interact with it as described. It also means that you can run your tests on a variety of different screen resolutions and they will still work – unless you've added co-ordinates to your scripts.

SilkTest treats every control within your application as a separate window and 4Test is an object-oriented scripting language making it very easy to write well structured and re-usable scripts.

Verification

One of the great strengths of SilkTest is the ease in which you can verify that your application is behaving as expected.

While recording a testcase, if you move the mouse over the part of your GUI that you wish to verify and press <ctrl> + <alt>, SilkTest will gather all the properties for that part of your application. For example if the mouse is over a button it will gather all the properties for that button (Caption, Exists, Enabled & Size), if the mouse is over a form containing several buttons, it will gather all the properties for all the buttons as well as the form itself.



The above screenshot shows the verify window. You can see that every window that has been identified is checked and the caption property for “Add” is checked also. You can uncheck windows & properties you aren’t interested in and vice-versa.

When you come to run your test, SilkTest will compare the checked properties you have recorded against the values they have at runtime and pass or fail the test accordingly.

If you are consistently verifying the same kind of properties throughout your application, you can define a property set. You can then select this property set in the bottom-right corner of the verify window and SilkTest will automatically check the relevant properties for you rather than you having to manually check/uncheck them each time.

Verification is a very useful tool and can get you testing a large amount of the application very quickly. Smoke tests for example can be automated within minutes in SilkTest making them quicker and more consistent for each successive build of your application.

Conclusion

SilkTest is a very useful test automation suite. It has the flexibility to enable you to create quick and simple or structured and reusable automated test scripts depending on your goals and technical confidence.