Testing with Selenium

Selenium (http://www.seleniumhq.org) is a widely used open-source tool for automating browsers, with growing support from browser vendors. The GitHub repository Dyalog/Selenium contains code which allows Dyalog applications to drive browsers via Selenium.

Selenium allows you to navigate to a given page, enter text into input fields, click on buttons (or any element), move the mouse and perform different types of mouse clicks, perform keypresses – simulate any action that a user could perform. Subsequently, you can verify properties and attributes of DOM elements, to test that your page is working correctly.

Setting Up

If you check out the Selenium project into the same folder as you have MiServer checked out to (for example, `/tmp/Selenium` and `/tmp/MiServer`, then you will be able to use the function `Test`, which can be found in the main miserver workspace.

Running Tests

You must start the MiServer that you wish to test in one process, and run Test from another. It is not recommended that you run `Start` and then `Test` in the same APL process. This may work in some circumstances but is not supported.

To run all existing tests on a MiServer (which is already be running), `)load` the `miserver` workspace, and then call the function Test with the folder name of the MiSite as the right argument. For example:

```
      Test 'MS3'
Development environment loaded
MiSite "c:\devt\miServer\MS3\" loaded
Starting Chrome
.........................
*** FAILED *** #28 of 40: /QA/Examples/SF/ejAccordionAdvanced: Accordion
Selection Failed
*** FAILED *** #29 of 40: /QA/Examples/SF/ejAccordionSimple: Accordion
Selection Failed
.........
*** FAILED *** #39 of 40: /QA/Examples/SF/ejTreeViewAdvanced: Node Add & Check
Failed.
.
Total of 40 samples tested in 0m35s: 3 failed.
```

A "." Is output to the session for each test, so you can see that something is happening. Each failed test will cause a message to be displayed in the session.

You can also pass a second character vector on the right; this will be used as a PCRE expression to filter the list of tests to be run – and finally, a finally, a left argument of 1 will disable error trapping and cause any failing tests to suspend so that they can be debugged:

```
      1 Test 'MS3' 'TreeViewAdvanced'
Development environment loaded
MiSite "c:\devt\miServer\MS3\" loaded
Selected: 1 of 40 tests.
test for /QA/Examples/SF/ejTreeViewAdvanced failed:
Node Add & Check Failed.
Rerun:
      Test 0
SYNTAX ERROR
Run1Test[8] ∘∘∘
             ^
```

Writing Tests

The Test function looks for files within the sites QA folder, expecting to find a structure here which is parallel to the sites page structure. If the site has a page `Examples/DC/ButtonSimple.mipage`, it will look for a file called `Examples/DC/ButtonSimple.dyalog`, which needs to be the source for a monadic function (the right argument is not currently used) which must also be called `Test`.
The test function should assume that the browser has already navigated to the page in question, and that a ref called Selenium exists. After testing the behavior of the page, the `Test` function should return an empty vector if the test succeeded, or a character vector containing a failure description.
A number of examples illustrating common types of tests can be found in the following. For more information, consult the Selenium documentation which can be found in the GitHub repository.

DC/ButtonSimple

Presses a button and waits for confirmation to appear in a div:

```
      ∇ msg←Test dummy
[1]     Selenium.Click'btnPressMe'
[2]     msg←'output'Selenium.WaitFor'Thank You!'
      ∇
```

DC/FieldSetSimple

Use SendKeys to type into two input fields which are responding to keypress events; verify the output:

```
      ∇ msg←Test dummy
[1]     'fname' 'lname'Selenium.SendKeys¨'Morten' 'Kromberg'
[2]     msg←'output'Selenium.WaitFor'Hi Morten Kromberg!'
      ∇
```

DC/ListManagerSimple

Selects two fruits by dragging them from one list box to another (ListMgrSelect is a function which has been written for the specific purpose of supporting the ListManager widget). Click on the Save button and wait for confirmation:

```
     ∇ msg←Test dummy
[1]    'fruits'Selenium.ListMgrSelect'Oranges' 'Lemons'
[2]    Selenium.Click'btnSave'
[3]    msg←'output'Selenium.WaitFor'You picked: Oranges Lemons'
     ∇
```