Zelenium tests and common utilities are stored in*erp5_ui_test* business templates. Many other tests could also be found in *erp5_*_ui_test*. Zuites are stored in portal_tests and are organized in Zuite recursively. A zuite is an set of tests stored as Page Templates (see also HowToRunZeleniumTests).

${table_of_content}

Useful resource

- XPath Checker
- Selenium
- Selenium HTML Command Referece(-ish)

Writing Zelenium Tests

Zelenium, how the name hints, is Selenium test tool built into Zope itself. It is accessible at <instance>/erp5/portal_tests/manage_main. Make sure that you have business template*erp5_ui_test* installed.

To create a test, you need to create a*Zuite* first using drop-down menu on top-right corner. Zuite is a kind of folder and contains all runnable tests. As the ID, use "<module>_ui_zuite", for example "renderjs_ui_listbox_zuite". Then navigate inside your new Zuite and create a *Page Template* with ID starting at "test" such as "testMyModuleFormSubmit".

Example TestCase

Below is an example of test code that you could save as a Page Template :

```
<html xmlns:tal="http://xml.zope.org/namespaces/tal"
      xmlns:metal="http://xml.zope.org/namespaces/metal">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Test My Module UI</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">My Module Zuite</td></tr>
</thead><tbody>
<tal:block metal:use-macro="here/Zuite_CommonTemplate/macros/init" />
<tr>
  <td>open</td>
  <td>${base_url}/web_site_module/my_module_website/</td>
  <td></td>
</tr>
<tr>
  <td>assertElementPresent</td>
  <td>//a[contains(text(), 'Add')]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//a[@data-i18n="Submit" and @type="submit"]</td>
  <td></td>
</tr>
<tr>
  <td>verifyTextPresent</td>
  <td>OK</td>
  <td></td>
</tr>
</tbody></table>
</body></html>
```

Standard Zelenium Commands

Selector could be xPath expression (in form of "//expression") or simplified selector.

- **XPath** selector works with all commands so it is the safest bet. It starts with**//**.
- **Simplified selector** refers to subset of element types by default thus it is enough to specify uniquely identified attribute of such an element. For example for **type** and **select** commands the selector can be "name=<input-name>".

Text (or regexp) can be either glob or regular expression form.

- **glob** is UNIX style pattern matching and prefixed by*glob:* (eg. click, link=glob:person[0-9]id or assertText, glob:Hello*Welcome)
- **regexp** is traditional regular expression prefixed by regexp: (examples as above but with different prefix)

Command Name	Target (1. argument)	Value (2. argument)	Description
assertText	selector	text (or regexp)	Verify textual content (TextNode in javascript) of a concrete element identified by selector.
assertTextPresent	text (or regexp)	-	Verify that given text is present anywhere in the document. Useful when using selector is impossible due to inpredictability.
assertElementPresent	selector	-	Verify that given element is present in the document.
verifyValue	selector	text (or regexp)	Verify **current** value of an input field. Works even for select. That means dynamic value inputted by javascript or the user.
type	simplified-selector	text	Type test into editable input/textarea field.
select	simplified-selector	value=<option-value>	Select an option in a select box. Seems that option can be selected only using value and not label.
pause	value [ms]	-	Please do not use. It pauses execution for given amount of mili-seconds. Use WaitForElement instead.
click(AndWait)	selector	-	Click a click-able element (and wait for a page-reload - so this extension is useless for SinglePage applications).
fireEvent	selector	event name	Fire an event on matched element (eg. fireEvent, dom=document.forms[1], submit).
store	value	variable name	Stores a static value in a variable. Later the variable can be accessed by ${variable_name} or in javascript (command storeEval) as storedVars['variable_name'].
storeText	selector	variable name	Stores a text presented in an element matched by the selector.
storeEval	javascript expression	variable name	Executes javascript expression or builtin javascript script and stores the resulting value in variable name.
verifyVisible	selector	-	Verify element's visibility on the page.

Assert versus **Verify** is subtle but important. Verify continues the test if the condition fails. They are completely interchangeable in the way that every assert* has its verify* alias.

Selenium User Extensions for ERP5

We have the following extensions.

- assertPortalStatusMessage that checks the value of a portal status message.
- assertFloat that basically acts as assertText, but converts both values to float before comparting them, with this 1 000.00 is equals to 1000.

Tips and Tricks

Beware responsive changes

When you `waitForElement` it might never show up because it is hidden in the smallest screens. Even though the tests are running at 1280x1024x24 it is better to optimize your tests for 1024x768 to be super-sure I think.

Type a date

When using command `type` for `input[@type="date"]` the value must be in form `YYYY-MM-DD`. Keep in mind the leading 0 to make single-digit numbers follow DD or MM pattern.

Selecting in a list field

When using `selectAndWait` command and the element is already selected, selenium hangs. The solution is to use `assertSelected` before using `selectAndWait`, so if the selection is not what you expect it to be, selectAndWait will not be executed.

Keep in Mind

Reindex in the middle of a test

You should avoid doing useless reindex in the middle of tests. Every time you use it, this means the scenario you are checking would surely not work without reindex. It is usually better when user can do many actions without the need to wait for indexation. Though, indexing is still useful in various cases, like checking catalog search. In such case you could do as follow :

```
<!-- reindex -->
<tr>
  <td>store</td>
  <td>javascript{selenium.browserbot.getCurrentWindow().location.href}</td>
  <td>current_location</td>
</tr>

<tr>
  <td>open</td>
  <td tal:content="string:${here/portal_url}/Zuite_waitForActivities"/>
  <td/>
</tr>

<tr>
  <td>assertTextPresent</td>
  <td>Done</td>
  <td/>
</tr>

<tr>
  <td>open</td>
  <td>${current_location}</td>
  <td></td>
</tr>
```

- Tests should not rely on previous test runs
- Test should have setUp and tearDown scripts which can prepare and clean up environment respectively
- Test must not contains any hard coded values like user names, organizations, etc ... Instead configuration of former can be moved to a Python script which can be used at test rendering time to dynamically generate test
- Test should use a non manager account but a real ERP5 account so security is taken into account whenever test is run
- Test must be created if feasible in a way so that they can be reused and plugged into a production instance, run in it without modifying production data
- Common code can be grouped into a "library" of macros

Zelenium Hints

- Take care when using 'open' rather than 'openAndWait'. This can cause random test failures whenever only 'open' (fire and go one test execution) is used.
- When you have to use "type" in selenium always assert if all fields and the save button is present, before use any type into one page or any selection. This will prevent the tests become stopped waiting for confirmation in dialog is some action is not possible.

Compatibility Notes

Zelenium 0.8 uses Selenium 0.6, and SVN trunk of Zelenium uses Selenium 0.8.3. And there are some incompatibilities.

- getLocation() returns full URL instead of path.
- We define getAbsoluteLocation() in erp5_ui_test_core. If you use assertAbsoluteLocation() or verifyAbsoluteLocation() instead of assertLocation() or verifyLocation(), the test should work on both version.
- isEditable() raises an error if the specified element isn't an input element.
- We can't use assertNotEditable() or verifyNotEditable() for asserting "it isn't an input field".
- Use
 to specify a new line in type(), assertText(), verifyText() etc. instead of a real new line in HTML code.
- There is no way to write a test with new lines that supports both version.

${related_subject_list}