Shows how to install and run Zelenium tests.

${table_of_content}

Installation

- Install business template erp5_ui_test.

Running Interactively through the web

Navigate to <host>/erp5/portal_tests and hit "Go" button in the top left corner. A list of all available tests should appear instead of the "Go" button. Select the test you want to run and now you can control the execution in top-right window.

If you want to run only your zuite, navigate yourself to<host>/erp5/portal_tests/<your_test_zuite> and hit the "Go" button from here. If you don't know the name of your zuite by heart you can get there via <host>/erp5/portal_tests/manage_main then click on your test zuite and then on "View" among the top tabs.

Sit back and watch a ghost work on your machine.

Running From Command Line

The advised way to launch Zelenium tests is through the web.

There is though an option to run zelenium tests from the command line thus by your favorite Continuous Integration tool. SlapOS ERP5 provides a wrapper for setting up correct python path and testing environment (databases and caches) different from production environment. The wrapper is available in your <instance-home>/<zope-slappart>/bin under the name runUnitTest.

```
$ cd ~/srv/runner/instance/<slappart>/bin
$ ./runUnitTest --save <tested-business-template>:<Reference> # first test run
$ ./runUnitTest --load <tested-business-template>:<Reference> # subsequent runs
```

First run of your test

Current test runner luckily does not make difference between unit and functional tests. All tests returned by function *test_suite* inside the script specified by *Reference* will be run. See below minimal *Portal Component* script to run Zelenium tests specified in portal_test/lte_request_ui_zuite.

```
import unittest

from Products.ERP5Type.tests.ERP5TypeFunctionalTestCase import ERP5TypeFunctionalTestCase

class TestRenderLTERequest(ERP5TypeFunctionalTestCase):
  foreground = 0
  run_only = "lte_request_ui_zuite"

  def getBusinessTemplateList(self):
    return (
      'erp5_lte_request',
      'erp5_web_renderjs_ui',
      'erp5_ui_test_core',
    )

def test_suite():
  suite = unittest.TestSuite()
  suite.addTest(unittest.makeSuite(TestRenderLTERequest))
  return suite
```

This script is exported with erp5_lte_request *business template* and saved under *Portal Component* with attributes

```
ID: test.erp5.testFunctionalLTERequest
Reference: testFunctionalLTERequest
```

and that is exactly the reference you have to use when running the test. So to run this particular test you would write

```
./runUnitTest --save erp5_lte_request:testFunctionalLTERequest
```

Subsequent runs

You saw that every run (without --save and --load) was creating the whole environment from scratch. It installed all *Test Dependencies* defined in the business template you are testing and populated database. In order to preserve the environment use --save and to reuse it again use --load. The --save option saves the environment into one place no matter which *business template* you are testing or which *test* you are running. So be aware of loading incomplete environments when you change to testing a different *business template*.

In order to see more options of UI Tests (eg. invoking Firefox on the current DISPLAY instead of on Xvfb, specifying a test suite etc.) try --help option.

Running on production

Please be aware that this could be very dangerous and you might loose production data. Some tests might not be well written and they could erase data. Though, sometimes it is useful to run your tests directly on production. Production instance is defined by its ZODB Datafile usually located at ~/srv/runner/instance/slappart3/srv/zodb/root.fs. Business templates and python sources are by default shared between tests and production because both points on the default path. This path can be changed manually in production instance in "Business Templates" -> "Import/Export". To run for example your functional unit test on production to see if it is working

```
./runUnitTest --live_instance=~/srv/runner/instance/slappart3/srv/zodb/root.fs erp_lte_request:testFunctionalLTERequest
```

Troubleshooting

You may see an error on "type" commands (the left column):

Selenium failure. Please report to selenium-devel@lists.public.thoughtworks.org, with details from the logs at the base of the page. UniversalFileRead privilege has been denied for this script

This is not at all a bug in selenium, to test file uploads, we use this hack. To enable it in Firefox, type about:config in the location bar and set signed.applets.codebase_principal_support to true. Warning, this can represent a security risk for your browser

${related_subject_list}