gocept.selenium

Release 3.0

Jun 30, 2022

Contents

gocept.selenium provides an API for Selenium that is suited for writing tests and integrates this with your test suite for any WSGI, Plone or Grok application.

While the testing API could be used independently, the integration is done using test layers, which are a feature of zope.testrunner.

Use gocept.pytestlayer to integrate it with pytest.

Contents:

CHAPTER 1

Setting up the environment

Download the Selenium Server JAR from seleniumhq.org and run:

```
$ java -jar /path/to/selenium-server-standalone-2.xx.xx.jar
```

This starts the server process that your tests will connect to to spawn and control the browser.

Choose the appropriate test layer (see *Integration*) and create a test case:

```
import gocept.selenium.wsgi
from mypackage import App

test_layer = gocept.selenium.wsgi.Layer(App())


class TestWSGITestCase(gocept.selenium.wsgi.TestCase):

    layer = test_layer

    def test_something(self):
        self.selenium.open('http://%s/foo.html' % self.selenium.server)
        self.selenium.assertBodyText('foo')
```

1.1 Environment variables

You can set some variables in the environment of your test runner to configure which selenium server gocept.selenium connects to. Selenium Server defaults to localhost:4444, but you can also connect to a selenium grid in your organization by using the following environment variables:

```
GOCEPT_SELENIUM_SERVER_HOST=selenium.mycompany.com
GOCEPT_SELENIUM_SERVER_PORT=8888
```

If multiple browsers are connected to your selenium grid, you can choose the browser to run the tests with like this:

```
GOCEPT_SELENIUM_BROWSER=*iexplore
```

For use with Selenium Server's webdriver interface, the browser needs to be specified differently:

```
GOCEPT_WEBDRIVER_BROWSER=firefox
```

Webdriver supports instantiating the browser directly (instead of going through the Java-based server component). If you want to do this, set:

```
GOCEPT_WEBDRIVER_REMOTE=False
```

and specify one of the browser classes defined by the Python bindings, for example:

```
GOCEPT_WEBDRIVER_BROWSER=Firefox
```

If you want to use a Firefox binary at a custom path, specify it like this:

```
GOCEPT_WEBDRIVER_FF_BINARY=<PATH>/firefox
```

By default, the selenium layer will make the HTTP server under test bind to localhost and listen to a random port chosen by the kernel (i.e. instruct it to bind to port 0). This randomly chosen port is then used to point the browser at the application. You may want to influence this behaviour, e.g. when running your selenium tests on a selenium grid:

```
GOCEPT_SELENIUM_APP_HOST=10.0.0.15
GOCEPT_SELENIUM_APP_PORT=8001
```

When you are testing an application on one machine, you can access the running application from another machine if you set `GOCEPT_SELENIUM_APP_HOST = 0.0.0.0` instead of the default `localhost`.

You can control the timeout of `waitFor` assertions and other selenium actions by setting a timeout in seconds:

```
GOCEPT_SELENIUM_TIMEOUT=10  (default: 30 seconds)
```

You can also set the speed with which the tests are run through an environment variable:

```
GOCEPT_SELENIUM_SPEED=500
```

This example will introduce a 500 millisecond pause between tests.

1.2 Jenkins integration

If you use Jenkins, you might be interested in the JUnit Attachment Plugin, and setting:

```
GOCEPT_SELENIUM_JUNIT_ATTACH=True
```

This will print information about the screenshot of a failure that the plugin can read and attach the screenshot to the test run.

In the configuration of the jenkins job you need a *Post-build Action* called *Publish JUnit test result report*. This action needs an *Additional test report feature* called *Publish test attachments* to ask Jenkins to keep the screenshots for you.

Caution: zope.testrunner is not usable for this behavior, you have to use a test runner like py.test. Newer py.test versions require you to write `junit_logging = system-out` to *pytest.ini* so the information is written to the *junit.xml* file. Run py.test with the command line option `--junitxml=junit.xml` to create this file. (That's what you'll normally do to get the test results to Jenkins.)

1.3 Tips & Tricks

1.3.1 Using a custom Firefox profile

For debugging purposes it's helpful to have the Firebug debugger available in the Selenium-controlled browser. To do that, create a new Firefox profile and install Firebug into it. Then you can tell Selenium to use this profile as a profile template when running Firefox:

```
$ java -jar /path/to/selenium-server-standalone-2.xx.xx.jar -firefoxProfileTemplate ~/
↪.mozilla/firefox/<PROFILE_FOLDER>
```

When using webdriver, instead set this environment variable for running the tests (not Selenium Server):

```
GOCEPT_WEBDRIVER_FF_PROFILE=~/.mozilla/firefox/<PROFILE_FOLDER>
```

1.3.2 Using a nested X Server

On Linux systems, the Selenium-controlled browser tends to steal the window focus, which makes it impossible to do anything else while a Selenium test is running. To prevent this, use Xephyr (successor of Xnest) to start an X server contained in a window, for example:

```
#!/bin/sh
display=:1
Xephyr -host-cursor -dpi 100 -wr -screen 1400x900 $display &
export DISPLAY=$display
sleep 2
metacity &  # or any other window manager
x-terminal-emulator -e java -jar /path/to/selenium-server-standalone-2.xx.xx.jar
```

CHAPTER 2

Integration

gocept.selenium provides integration with several web frameworks. Since version 1.1, however, the actual integration functionality has been extracted to gocept.httpserverlayer, so the recommended setup is to use one layer from there that integrates with your application (see gocept.httpserverlayer documentation for details) and provides an HTTP server, and then stack the layer from gocept.selenium on top of that, to provide the Selenium integration:

```
import gocept.httpserverlayer.wsgi
import gocept.selenium
from mypackage import App

http_layer = gocept.httpserverlayer.wsgi.Layer(App())
selenium_layer = gocept.selenium.WebdriverLayer(
    name='SeleniumLayer', bases=(http_layer,))
selenese_layer = gocept.selenium.WebdriverSeleneseLayer(
    name='SeleneseLayer', bases=(selenium_layer,))

class TestWSGITestCase(gocept.selenium.WebdriverSeleneseTestCase):

    layer = selenese_layer

    def test_something(self):
        self.selenium.open('http://%s/foo.html' % self.selenium.server)
        self.selenium.assertBodyText('foo')
```

The previous set of layers that provide both the HTTP server and Selenium in one layer is still available. Different frameworks require different dependencies; this is handled via setuptools extras of gocept.selenium (e.g. for Grok integration you need to require `gocept.selenium[grok]`). Generally, you need a test layer that handles the setup, and then have your tests inherit from the appropriate `TestCase`.

2.1 WSGI

No extra requirements (simply `gocept.selenium`).

This test layer takes a WSGI callable and runs it in a temporary HTTP server:

```
import gocept.selenium.wsgi
from mypackage import App

test_layer = gocept.selenium.wsgi.Layer(App())

class WSGIExample(gocept.selenium.wsgi.TestCase):

    layer = test_layer

    def test_something(self):
        self.selenium.open('http://%s/foo.html' % self.selenium.server)
        self.selenium.assertBodyText('Hello world!')
```

2.2 Static files

No extra requirements (simply `gocept.selenium`).

This test case provides a temporary directory (as `self.documentroot`) that is served via HTTP where tests can put HTML files to examine:

```
import gocept.selenium.static

class StaticFilesExample(gocept.selenium.static.TestCase):

    def test_something(self):
        open(os.path.join(self.documentroot, 'foo.html'), 'w').write(
            'Hello World!')
            self.selenium.open('http://%s/foo.html' % self.selenium.server)
            self.selenium.assertBodyText('Hello world!')
```

2.3 Zope3 / ZTK (zope.app.wsgi)

If your ZTK application uses `zope.app.wsgi.testlayer`, see *Grok* for integrating `gocept.selenium`.

2.4 Grok

Requires `gocept.selenium[grok]`.

This test layer groks your package and sets everything up so Selenium can access the application. You will probably want to setup your app in your test setup:

```
import gocept.selenium.grok
import transaction

selenium_layer = gocept.selenium.grok.Layer(my.package)

class GrokExample(gocept.selenium.grok.TestCase):

    layer = selenium_layer

```

(continues on next page)

(continued from previous page)

```
    def setUp(self):
        super(MyTest, self).setUp()
        root = self.getRootFolder()
        root['app'] = mypackage.App()
        transaction.commit()

    def test(self):
        self.selenium.open('/app')
        self.selenium.assertBodyText('Hello world!')
```

2.5 Zope 2

Requires `gocept.selenium[zope2]`

This test layer requires `Testing.ZopeTestCase.layer.ZopeLiteLayer` and provides an HTTP server for the tests. See `gocept.selenium.zope2.tests.test_zope212` for details how to set this up.

2.6 Zope 2 via WSGI

If your Zope 2 setup supports it, you can use the WSGI integration instead of a specialised Zope 2 integration to run your tests.

You might see the following exception when running tests:

```
File ".../repoze.retry-1.0-py2.7.egg/repoze/retry/__init__.py", line 55, in __call__
  cl = int(cl)
 ValueError: invalid literal for int() with base 10: ''
```

To fix it you can use an additional middleware around your WSGI application: `gocept.selenium.wsgi.CleanerMiddleware`. It also fixes an issue with `wsgiref`. See comments in the code for more information.

2.7 Zope 2 / Plone with plone.testing

Requires `gocept.selenium[plonetesting]`.

`gocept.selenium` provides a `plone.testing.Layer` at `gocept.selenium.plonetesting.SELENIUM` that you can mix and match with your other layers, see `gocept.selenium.plonetesting.testing` with `gocept.selenium.plonetesting.tests.zope2`, and `gocept.selenium.plonetesting.testing_plone` with `gocept.selenium.plonetesting.tests.plone{3,4}` for details how to set this up.

2.8 Converting Selenese HTML files

Selenium tests can be written in HTML tables.

Their syntax is a bit clunky. But their development and debugging is eased a lot by using Selenium IDE Firefox extension. Selenium IDE provides both initial recording of tests and stepping through those tests. However, HTML tests have a main drawback: they are hard to include in a continuous integration system.

`gocept.selenium` provides a script that converts a set of Selenium HTML tests into a Python module with a `TestCase` (based on `gocept.selenium` and `plone.testing`).

Using the `converthtmltests` script, the developer can use HTML tests – written, debugged and maintained with the Selenium tools – while being able to easily include those Selenium tests in a continuous integration system.

2.8.1 Usage

```
converthtmltests -l LAYER [options] directory

options:
  -f FILE, --file=FILE  write tests to FILE
  -l LAYER, --layer=LAYER
                        full python import path to layer instance
```

The script gathers and converts all Selenium HTML tests found in the mentioned directory.

The user must refer to a `plone.testing` layer by specifying its Python import path. That layer is set on the test case generated in the Python module.

An output file can be specified. In case no output file name is specified, the module produced is named `tests_all_selenium.py`.

CHAPTER 3

API reference

3.1 Selenese API

3.1.1 General information

The `Selenese` object available as `self.selenium` for each TestCase provides methods to control the browser, and to make assertions about things the browser sees.

For a detailed list of commands and assertions please consult the Selenium Reference.

Assertions come in several flavours:

- Return the value `self.selenium.getText('id=foo')`
- Assert `self.selenium.assertText('id=foo', 'blabla')`
- Negated Assert `self.selenium.assertNotText('id=foo', 'blabla')`
- Wait `self.selenium.waitForElementPresent('id=foo')`
- Negated Wait `self.selenium.waitForNotElementPresent('id=foo')`

3.2 Webdriver API

Starting with version 2, gocept.selenium also includes integration with Selenium's webdriver backend, the plan being to keep our own API as backwards-compatible as possible during the 2.x release series and switching to a modernized API only with version 3.

This means that we've set out to implement the Selenese API on top of webdriver and while this has proven to be possible to a large extent, some details of the Selenese API don't make any sense or are too different to be worth implementing in a webdriver environment.

Here's how to set this up (see *Integration* for details):

```
import gocept.httpserverlayer.wsgi
import gocept.selenium
from mypackage import App

http_layer = gocept.httpserverlayer.wsgi.Layer(App())
webdriver_layer = gocept.selenium.WebdriverLayer(
    name='WSGILayer', bases=(http_layer,))
test_layer = gocept.selenium.WebdriverSeleneseLayer(
    name='WebdriverTestLayer', bases=(webdriver_layer))


class TestWSGITestCase(gocept.selenium.WebdriverSeleneseTestCase):

    layer = test_layer

    def test_something(self):
        self.selenium.open('http://%s/foo.html' % self.selenium.server)
        self.selenium.assertBodyText('foo')
```

Here's a list of backwards-incompatibilities between using WebdriverSeleneseLayer and the (old) SeleniumRC-backed gocept.selenium.RCLayer:

- `getEval` behaves differently.
  - `getEval` adds a `return` statement in front of the code, i.e. to run Javascript code which is not an expression, use `runScript`
  - `getEval` has access to different globals now: `browserbot` is no longer defined, while `window` and `document` refer directly to the window under test.
  - `getEval` returns the dictionary representation of objects instead of the rather uninformative `[object Object]`.
- The browser name syntax has changed: specify Firefox as "firefox", not "firefox*" (concerns the environment variable for setting the browser, which used to be `GOCEPT_SELENIUM_BROWSER` and is `GOCEPT_WEBDRIVER_BROWSER` for webdriver). See the WebDriver wiki for possible browser names.
- With Selenium Remote-Control one had to change the base Firefox profile to be used on the server side (by passing `-firefoxProfileTemplate` to `selenium-server.jar`). With WebDriver this has moved to the client side, so you can select a profile by setting the path to an existing Firefox profile as the environment variable `GOCEPT_SELENIUM_FF_PROFILE`.
- Selenese methods that don't work yet:
  - `highlight`
  - `getSpeed`
  - `setSpeed`
  - `getAllWindowNames`
  - `getAllWindowTitles`
  - `selectPopUp`
  - `deselectPopUp`
- Selenese methods with changed behaviour:
  - `open`: dropped the `ignoreResponseCode` parameter
  - `assertOrdered` only works with relative xpath locators, not with any element locators anymore.

- Selenese methods that have been removed and are not coming back:
  - `addCustomRequestHeader`
  - `addLocationStrategy`
  - `addScript`
  - `allowNativeXpath`
  - `answerOnNextPrompt`
  - `assignId`
  - `captureNetworkTraffic`
  - `chooseCancelOnNextConfirmation`
  - `chooseOkOnNextConfirmation`
  - `fireEvent`
  - `focus`
  - `getMouseSpeed`
  - `getTable`
  - `ignoreAttributesWithoutValue`
  - `removeScript`
  - `retrieveLastRemoteControlLogs`
  - `setBrowserLogLevel`
  - `setContext`
  - `setCursorPosition`
  - `setMouseSpeed`
  - `useXpathLibrary`
  - `waitForFrameToLoad`
- Locator patterns that can no longer be used:
  - option: id
  - frame: relative, dom

On the other hand, here are some new features that only WebdriverSeleneseLayer offers:

- Locator `js` (or `dom` or anything that starts with `document`): Find an element by evaluating a javascript expression. Example: `getText('js=document.getElementsByClassName("foo")')`
- Convenience locator `jquery` (when your site already loads `jQuery`). Example: `getText('jquery=.foo')` (this is the equivalent of `getText('js=window.jQuery(".foo")[0]')`)

3.3 Test helpers

3.3.1 assertScreenshot

Note: `assertScreenshot` *needs* PIL. You might consider to require the *screenshot* extra in your setup.py like so: `gocept.selenium[screenshot]`

The `assertScreenshot` method allows you to validate the rendering of a HTML element in the browser. A screenshot of the element is saved in a given directory and in your test `assertScreenshot` takes a picture of the currently rendered element and compares it with the one saved in disk. The test will fail, if the screenshot and the taken picture do not match (within a given threshold).

`assertScreenshot` takes the following arguments:

name A name for the screenshot (which will be appended with *.png*).

locator A locator to the element, which will be captured.

threshold If the difference[2] in percent between the saved and current image is greater than the threshold, a failure is triggered. (defaults to 1)

There is a capture mode available to help you in retrieving your master screenshot (which will be left on disk for comparison). When writing your test, set `capture_screenshot` on the *Selenese* object (see general-information) to `True` and the test run will save the screenshot to disk instead of comparing it. Before you check in your newly created screenshot, you should watch it to make sure, it looks like you expected it. Setting `capture_screenshot` to `False` will compare the screenshot on disk with a newly created temporary image during the next test run.

If `assertScreenshot` fails, paths to the following images are provided to you in the error message:

original The path to the original image (the master image).

current The path to the image taken in the current test run (from the browser).

diff The path to an image highlighting the differences between original and current.

If you would like to open the image showing the differences in an image viewer, set the environment variable `SHOW_DIFF_IMG` before running the test.

3.3.2 Skipping tests for certain browsers

There are cases when a test should does not pass on certain browsers. This is either due to the application using browser features which are not supported by the browser, or due to selenium not working well with the browser. To aid in skipping tests in these cases, there is a test decorator `gocept.selenium.skipUnlessBrowser(name, version=None)`:

```
>>> class TestClass(...):
...
...     @gocept.selenium.skipUnlessBrowser('Firefox', '>=16.0')
...     def test_fancy_things(self):
...         ...
```

Note: `skipUnlessBrowser` *only* supports skipping test methods. It cannot be used as class decorator.

3.3.3 Downloading files

By default `selenium` does not support to download files because this is done via native operating system dialogues which `selenium` cannot intercept.

[2] The difference is computed as normalised root mean square deviation of the two images.

The only way seems to be to instruct the browser to always store downloads of a certain MIME type in the download directory.

This is implemented in `gocept.selenium` for PDF files when using `Firefox`. The download directory is accessible as a `pathlib.Path` via `self.layer['selenium_download_dir']`. It gets cleared at the end of every test.

CHAPTER 4

Development

4.1 Developing gocept.selenium

Author gocept <mail@gocept.com>

Online documentation https://goceptselenium.readthedocs.org/

PyPI page https://pypi.python.org/pypi/gocept.selenium/

Issue tracker https://github.com/gocept/gocept.selenium/issues

Source code https://github.com/gocept/gocept.selenium

Current change log https://raw.githubusercontent.com/gocept/gocept.selenium/master/CHANGES.rst

4.1.1 Documentation

In order to build the Sphinx documentation, run the following commands:

```
$ python3 -m venv .
$ bin/pip install Sphinx
$ bin/sphinx-build doc doc/build
```

The generated HTML gets stored in `doc/build`.

4.2 Changelog

4.2.1 7.1 (2022-06-30)

- Fix deprecation warnings.
- Depend on `webdriver-manager` to get drivers automatically updated.

- Add support for `edge` headless mode and test it on GHA.

4.2.2 7.0 (2022-06-28)

- Remove `.screenshot.ZeroDimensionError`. Where it was previously raised now the whole screenshot is saved.

4.2.3 6.1 (2021-05-04)

- Fix links and typos in documentation.
- Fix tests running on `Google Chrome`. (Issue #20)
- Google Chrome: Add temporary download directory support. It is accessible as a `pathlib.Path` via `self.layer['selenium_download_dir']`.
- Google Chrome: Add support for head mode.
- Add beta support for `Microsoft Edge`. (It currently supports all features besides headless mode and download directory support but is only tested using the tests of `gocept.selenium`. **Caution:** Edge does not seem to be really stable on Mac OS, after some test runs it sometimes refuses to start and requires to delete its preferences etc.)
- Add access to the selected browser and headless mode via a getitem call on the layer.

4.2.4 6.0 (2020-12-16)

Backwards incompatible changes

- Drop support for Python 2, 3.5 and 3.6.
- Remove `requirements.txt`.

Features

- Add support for Python 3.8 and 3.9.
- Firefox: Add a temporary download directory for PDF files. It is accessible as a `pathlib.Path` via `self.layer['selenium_download_dir']`.

4.2.5 5.2 (2020-10-28)

- Delete localStorage on testTearDown of `.webdriver.Layer`.
- Wait for elements the time defined as timeout instead of always 5 seconds.

4.2.6 5.1 (2019-11-14)

- Catch ElementClickInterceptedException when an element is clicked.
- Migrate to Github.

4.2.7 5.0 (2019-05-02)

- Fix *UserWarning* in *selenium* >= *3*, that screenshot name should end with *.png*.

4.2.8 5.0a1 (2019-03-05)

Backwards incompatible changes

- Remove support for Selenium 1, in particular `RCTestCase` and `RCLayer`.
- Remove support for Selenium 2.
- Remove support for a remote selenium server. *gocept.selenium* now uses the local implementation, starting its own browser.

Features

- Add support for Python 3.6 and 3.7.
- Selenium updated to version 3.
- Add firefox headless support.
- Add new defaults for gocept.selenium.webdriver.Layer.
- Add experimental support for chromedriver in headless mode only.

Other changes

- Remove bootstrap.py, add requirements.txt.
- Fix more deprecation warnings.
- `selenese_pattern_equals()` in *wd_selenese.py* now returns a bool.

4.2.9 4.0 (2018-11-09)

- Drop support for Zope 2.
- Depend on `gocept.httpserverlayer >= 3`.
- Depend on `plone.testing >= 7.0`.
- Fix deprecation warnings.

4.2.10 3.1.1 (2017-03-07)

- Fix `wd_selense.Selenese.selectFrame('index=0')` to forward the index as number to the underlying WebDriver so it is actually treated as an index.

4.2.11 3.1 (2016-11-11)

- Support selenium versions >= 2.53. https://bitbucket.org/gocept/gocept.selenium/issues/12
- Require a selenium version < 3.0 as this version removed the support for Selenium RC.

4.2.12 3.0 (2016-06-07)

- Drop support for:
  - `zope.app.testing` (extras_require: [ztk])
  - `Testing.ZopeTestCase` (extras_require: [zope2])
  - `plone.app.testing` (extras_require: [test_plonetestingz2])
  - `Products.PloneTestCase` (extras_require: [plonetestcase])
- Remove the empty `script` extras_require.
- Drop support for Python 2.4, 2.5, 2.6. Now only supporting Python 2.7.
- Currently only supporting a `selenium` version < 2.53 as this version breaks using a custom Firefox. See https://github.com/SeleniumHQ/selenium/issues/1965
- Add `.wd_selense.Selenese.selectParentFrame()` to select the parent of a frame or an iframe.

4.2.13 2.5.4 (2016-04-12)

- Fix using a local Firefox using `GOCEPT_WEBDRIVER_REMOTE=False` as the environment setting.

4.2.14 2.5.3 (2016-04-11)

- Update tests to *gocept.httpserverlayer >= 1.4*.

4.2.15 2.5.2 (2016-04-11)

- Add documentation for the Jenkins integration of screenshots made from test failures. (#13936)
- Webdriver: Add a loop with time-out to `click` in order to deal with `StaleElementReferenceException` and `NoSuchElementException`.

4.2.16 2.5.1 (2015-08-27)

- Webdriver: `waitFor` retries an assertion when `NoSuchElementException` was raised. (This is useful for assertions like `waitForVisible`.)

4.2.17 2.5.0 (2015-08-05)

- Add `clear` to webdriver to delete the contents of an input field.

4.2.18 2.4.1 (2015-06-23)

- Write junit annotations when a screenshot was taken for assertions beside `assertScreenshot()`. (#13678)

4.2.19 2.4.0 (2015-03-27)

- Added `getCssCount` and `getXpathCount`, so tests can get a baseline before an action.
- Fix `getSelectedValue` for webdriver.

4.2.20 2.3.0 (2015-03-09)

- Webdriver: `waitFor` will now retry the assertion when `StaleElementReferenceException` was raised, instead of yielding the error. (This could happen for assertions like `waitForAttribute`, which would retrieve the DOM node and *then* ask for it's attribute. Thus the node can be changed in-between, which leads to the error.)

4.2.21 2.2.2 (2015-01-09)

- Improve environment variable handling implementation.

4.2.22 2.2.1 (2015-01-07)

- Fix handling firefox profile in remote=false mode.

4.2.23 2.2.0 (2015-01-07)

- Allow launching the browser directly when using Webdriver (set `GOCEPT_WEBDRIVER_REMOTE=False` and the browser name accordingly).
- Add optional `movement` parameter to `dragAndDropToObject` that moves the mouse a little before releasing the button, so one gets more realistic behaviour when needed (Webdriver only, RC does not seem to have this issue).
- Add `js` and `jquery` locators (Webdriver only).

4.2.24 2.1.9 (2014-11-06)

- Fixed capitalisation of Selenese's `chooseOkOnNextConfirmation`. (Backwards incompatibility should be OK as it can never have worked before, anyway.)

4.2.25 2.1.8 (2014-09-04)

- No longer stop whole test run if an exception occures during `testSetUp` of `.seleniumrc.Layer` (#13375)

4.2.26 2.1.7 (2014-08-12)

- Remove `window.gocept_selenium_abort_all_xhr` again, this solution is incomplete, since we can only inject this during `open()` – when the browser then navigates to a different page, the injection is lost.

4.2.27 2.1.6 (2014-08-06)

- Inject JS function `window.gocept_selenium_abort_all_xhr` during `open()`, which is useful to call during test teardown to avoid spurious XHR requests to still be performed after the actual test has already ended. (Implemented in Webdriver only, but could be backported to RC if needed).

4.2.28 2.1.5 (2014-07-26)

- Webdriver: Only create a firefox profile when the selected browser is firefox (#11763).

4.2.29 2.1.4 (2014-07-09)

- Restore Python 2.6 compatibility of tests accidently broken in release 2.1.3.
- Adjust *isElementPresent* of WebDriver to work with PhantomJS, since it may raise a general WebDriverException if the element was not found.

4.2.30 2.1.3 (2014-07-07)

- Webdriver: No longer screenshotting while waiting for the condition to become true when using a `waitFor*` method.

4.2.31 2.1.2 (2014-06-25)

- Remove seleniumrc variable from Layer on teardown for symmetry.
- Fix *isVisible* of WebDriver, so it also returns False if a parent element is hidden.

4.2.32 2.1.1 (2014-04-28)

- Close temporary files when making screenshots. This fixes some occurrences of "Too many open files".

4.2.33 2.1.0 (2013-12-20)

- Make timeout configurable via environment variable `GOCEPT_SELENIUM_TIMEOUT` (#10497).
- Apply `setTimeout` to the `open()` timeout, too (#10750).
- Add environment variable `GOCEPT_SELENIUM_JUNIT_ATTACH` to support the "JUnit Attachments Plugin" for Jenkins.

internal:

- Move instantiating Selenese object from testSetUp to layer setUp. This *should* not change the behaviour for clients (we take care to reset the configured timeout in testSetUp as before), but take care.
- Fix URL to GROK toolkit versions.

4.2.34 2.0.0 (2013-10-02)

- Marking 2.0 stable, yay.

4.2.35 2.0.0b6 (2013-10-02)

- Save screenshots of assertion failures with mode 644 (world-readable), which is useful for build servers.

4.2.36 2.0.0b5 (2013-10-01)

- Implement `setWindowSize` for both RC and Webdriver.
- Implement `getAllWindowIds` in RC-Selenese.

4.2.37 2.0.0b4 (2013-04-26)

- If a test fails because of an empty body, taking automatically a screenshot failed and concealing the original error message. This is now fixed. (#12341)

4.2.38 2.0.0b3 (2013-04-10)

- Improved documentation, in particular with respect to the changes by integrating webdriver.
- If an `AssertionError` occures in a test using webdriver, a screenshot is taken automatically and the path is presented to the user. (#12247)
- Made a test for `assertScreenshot` pass on systems with a different browser default font.

4.2.39 2.0.0b2 (2013-03-01)

- Stabilize webdriver/selenese API functions *waitForPageToLoad()* and *isTextPresent* to not raise errors when the elements vanish in between.

4.2.40 2.0.0b1 (2013-02-14)

- Extract StaticFilesLayer to gocept.httpserverlayer.
- Added *assertScreenshot* to visually compare rendered elements with a master screenshot.

4.2.41 2.0.0a2 (2013-01-09)

- Add layer that uses Webdriver as the Selenium backend instead of the old Remote Control.

4.2.42 1.1.2 (2012-12-21)

- Fix: Initialise the WSGI layer in the correct order to actually allow the configured WSGI app to be remembered.
- Fix: updated some imports after the extraction of gocept.httpserverlayer.

4.2.43 1.1.1 (2012-12-19)

- Update StaticFilesLayer to the new httpserverlayer API.

4.2.44 1.1 (2012-12-19)

- Extract HTTP server integration into separate package, gocept.httpserverlayer

4.2.45 1.0 (2012-11-03)

- Marking the API as stable.

4.2.46 0.17 (2012-11-01)

- Added `gocept.selenium.skipUnlessBrowser` decorator to skip tests unless ceratins browser requirements are met.
- Fix: The static test server did not shutdown in some situations.

4.2.47 0.16 (2012-10-10)

- Fixed selenese popup tests.
- Open a random port for the server process by default: When the environment variable *GOCEPT_SELENIUM_APP_PORT* is not set, a random free port is bound. This allows parallel testing, for instance (#11323).

4.2.48 0.15 (2012-09-14)

- WSGI-Layer is comptabile with Python 2.5.
- Encoding support in converthtmltests (Patch by Tom Gross <tom@toms-projekte.de>).
- XHTML support for selenium tables (Patch by Tom Gross <tom@toms-projekte.de>).

4.2.49 0.14 (2012-06-06)

- API expansion: Added `assertCssCount`. Thus requiring selenium >= 2.0.
- Added Trove classifiers to package metadata.
- Moved code to Mercurial.

4.2.50 0.13.2 (2012-03-15)

- Fixed WSGI flavor: There was a `RuntimeError` in tear down if the WSGI server was shut down correctly.

4.2.51 0.13.1 (2012-03-15)

- Updated URL of bug tracker.
- *script* extra no longer requires *elementtree* on Python >= 2.5.

4.2.52 0.13 (2012-01-30)

- Added a selenese assert type 'list' and added it to the window management query methods.
- API expansion: added `openWindow`.
- API change: filter the result of `getAllWindowNames` to ignore 'null'.
- backwards-compatible API change: `selectWindow` now selects the main window also when passed the window id `None` or no argument at all.
- pinned compatible ZTK version to 1.0.1, grok version to 1.2.1, generally pinned all software packages used to consistent versions for this package's own testing

4.2.53 0.12 (2011-11-29)

- API expansion: added `getAllWindow*` and `selectWindow`.

4.2.54 0.11 (2011-09-15)

- Added some notes how to test a Zope 2 WSGI application.
- Described how to test a Zope 2/Plone application if using *plone.testing* to set up test layers.

4.2.55 0.10.1 (2011-02-02)

- Improvements on the README.
- Wrote a quick start section for packages using ZTK but using `zope.app.wsgi.testlayer` instead of `zope.app.testing`.
- Allowed to use *regexp* as pattern prefix for regular expressions additionally to *regex* to be compatible with the docstring and the Selenium documentation.

4.2.56 0.10 (2011-01-18)

- Script that generates python tests from Selenium HTML tables. Reused from KSS project, courtesy of Jeroen Vloothuis, original author.
- Using a URL of *Selenium RC* in README where version 1.0.3 can be downloaded (instead of 1.0.1) which works fine with Firefox on Mac OS X, too.

4.2.57 0.9 (2010-12-28)

- Provide integration with the recent testlayer approach (zope.app.appsetup/zope.app.wsgi) used by Grok (#8260).
- Provide integration with plone.testing
- Make browser and RC server configurable (#6484).
- Show current test case in command log (#7876).
- Raise readable error when connection to RC server fails (#6489).
- Quit browser when the testrunner terminates (#6485).

4.2.58 0.8 (2010-10-22)

- Fixed tests for the StaticFilesLayer to pass with Python 2.4 through 2.7.
- API expansion: `getSelectOptions`

4.2.59 0.7 (2010-08-16)

- API expansion: `getElementHeight|Width`, `getCookie*` and a few others.
- lots of action methods (`mouse*` among others)

4.2.60 0.6 (2010-08-09)

- assertXpathCount now also takes ints (#7681).
- API expansion: add `isChecked` to verify checkboxes, `runScript`, `clickAt`, `getLocation`, `getSelectedValue`, `getSelectedIndex`.
- The `pause` method uses float division now. Pauses where implicitly rounded to full seconds before when an int was passed.
- The name of the factored test layer contains the module of the bases now. The name is used by zope.testrunner distinguish layers. Before this fix selenium layers factored from base layers with the same names but in different modules would be considered equal by zope.testrunner.
- The factored ZTK layer cleanly shuts down the http server in tearDown now. This allows to run different selenium layers in one go.

4.2.61 0.5 (2010-08-03)

- Add a static files test layer for running selenium tests against a set of static (HTML) files.
- Patterns now also work with multiline strings, i. e. 'foo*' will match 'foonbar' (#7790).

4.2.62 0.4.2 (2010-05-20)

- API expansion: `*keyDown`, `*keyUp`, `keyPress`.

4.2.63 0.4.1 (2010-04-01)

- API expansion: added `getSelectedLabel`.
- Ignore the code of a server's response when calling *open*. The default behaviour of SeleniumRC changed between 1.0.1 and 1.0.2 but we want the old behaviour by default.

4.2.64 0.4 (2010-03-30)

- API expansion: add `getLocation` to retrieve currently loaded URL in browser.
- API expansion: added `waitForPopUp`, `selectPopUp`, `deselectPopUp` and `close`.
- API expansion: added `verifyAlertPresent`, `verifyAlertNotPresent` and `waitForAlertPresent`.

- Usability: raise a better readable exception when an unimplemented selenese method is called.
- Usability: raise failure exceptions that convey the name of the failed assertion in spite of some lambdas wrapped around it.

4.2.65 0.3 (2010-01-12)

- Extracted 'host' and 'port' as class attributes of gocept.selenium.ztk.Layer so subclasses can override them; stopped hardcoding 8087 as the server port.

4.2.66 0.2.1 (2009-12-18)

- Fix incomplete sdist release on PyPI.

4.2.67 0.2 (2009-12-18)

- Make Zope 2 test server reachable from the outside.
- Implemented getTitle/assertTitle/waitForTitle/etc.

4.2.68 0.1 (2009-11-08)

- first release