webtest-selenium Documentation

Release 0.1

Gael Pasgrimaud

Sep 29, 2017

Contents

class `webtest_selenium.`**`SeleniumApp`**(*app=None*, *url=None*, *timeout=30000*, *extra_environ=None*, *relative_to=None*, ***kwargs*)

See `webtest.TestApp`

SeleniumApp only support `GET` requests

`browser`

The current `Selenium`

`close`()

Close selenium and the WSGI server if needed

`webtest_selenium.`**`selenium`**(*obj*)

A callable usable as:

- class decorator
- function decorator
- contextmanager

CHAPTER 1

Response API

Some of the return values return instances of these classes:

CHAPTER 2

Environment variables

Those value are used if found in environment:

- SELENIUM_HOST: Default to 127.0.0.1
- SELENIUM_PORT: Default to 4444
- SELENIUM_BIND: IP used to bind extra servers (WSGI Server/File server). Default to 127.0.0.1
- SELENIUM_DRIVER: The driver used to start the browser. Usualy something in *chrome, *firefox, *googlechrome. Default to *googlechrome. You can get the full list by running:

```
$ java -jar selenium-server.jar -interactive
cmd=getNewBrowserSession
```

- SELENIUM_KEEP_OPEN: If exist then browser session are not closed so you can introspect the problem on failure.
- SELENIUM_JAR: If selenium is not running then this jar is used to run selenium.

CHAPTER 3

Examples

Testing a wsgi application

```
class TestApp(unittest.TestCase):

    def setUp(self):
        self.app = webtest.TestApp(application)

    def _test_forms(self):
        resp = self.app.get('/forms.html')
        self.assertSetEqual(set([0, "myform1", 1, 2, "myform3"]), set(resp.forms))
        self.assertEqual(resp.forms[0], resp.forms["myform1"])
        self.assertEqual(resp.forms[2], resp.forms["myform3"])

    def test_webtest(self):
        resp = self.app.get('/',
                            {'redirect': '/message.html?message=submited'})
        resp.mustcontain('It Works!')
        form = resp.forms['myform']
        form.lint()

        self.assertEqual(form['mytext'].value, '')
        resp.mustcontain(no='Form submited')

        with webtest_selenium.selenium(resp) as sresp:
            if sresp:
                sform = sresp.forms['myform']
                sform['mytext'] = 'foo'
                sresp = sform.submit(name='go', timeout=0)
                sresp.mustcontain('Form submited')

        if resp.updated:
            resp.mustcontain('Form submited')
            form = resp.forms['myform']
            self.assertEqual(form['mytext'].value, 'foo')

        resp = form.submit(name='go')
        resp = resp.follow()
        resp.mustcontain('<pre>submited</pre>')

        self._test_forms()

    @webtest_selenium.selenium
    def test_selenium(self):
        resp = self.app.get('/',
                            {'redirect': '/message.html?message=submited'})
        resp.mustcontain('It Works!')
        form = resp.forms['myform']
        form.lint()

        form['mytext'] = 'foo'
        self.assertEqual(form['mytext'].value, 'foo')

        # file upload are only supported with *firefox *chrome drivers
        filename = os.path.join(files, 'html', 'index.html')
        file = form['myfile']
        file.value = (filename,)

        form['myradio'] = 'true'
        self.assertEqual(form['myradio'].value, 'true')
        check = form.get('mycheckbox', index=0)
        check.value = 'true'
        self.assertEqual(check.value, 'true')
        form['myselect'] = 'value2'
        form['myselect'] = 'value2'
        self.assertEqual(form['myselect'].value, 'value2')
        form['mymultiselect'] = ['value1', 'value3']
        self.assertEqual(form['mymultiselect'].value, ['value1', 'value3'])

        # there is an ajax hook on the page
        resp = form.submit(name='go', timeout=0)
        resp.mustcontain('Form submited')

        # but we can submit the form to get the non-javascript behavior
        resp = form.submit()
        resp = resp.follow()
        resp.mustcontain('<pre>submited</pre>')

        self._test_forms()
```

Testing the jquery.ui website

```
class TestJQueryUI(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        cls.app = webtest_selenium.SeleniumApp(url='http://jqueryui.com/')

    def setUp(self):
        self.resp = self.app.get('http://jqueryui.com/demos/')

    def test_datepicker(self):
        resp = self.resp.click('Datepicker')
        field = resp.doc.datepicker
        field.fireEvent('focus')
        resp.doc.link('16').wait_and_click()
        self.assertIn('/16/', field.value)

    def test_dropable(self):
        resp = self.resp.click('Droppable')
        draggable = resp.doc.draggable
        droppable = resp.doc.droppable
        self.assertFalse(droppable.hasClass('ui-state-highlight'))
        draggable.drag_and_drop(droppable)
        self.assertTrue(droppable.hasClass('ui-state-highlight'))

        resp.doc.link('Shopping Cart Demo').click()
        cart = resp.doc.css('#cart ol.ui-droppable')
        cart.wait()
        item = resp.doc.xpath('//li[.="Lolcat Shirt"]')
        self.assertNotIn(item, cart)
        item.drag_and_drop(cart)
        self.assertIn(item, cart)

    @classmethod
    def tearDownClass(cls):
        cls.app.close()
```

Python Module Index

Index