Zelenium: Browser Testing for Zope

presented to the
2005 North American Plone Symposium
2005/07/21

Tres Seaver
Palladion Software
tseaver@palladion.com

Another Testing Framework?

- Gaps in coverage
- Different audiences need different kinds of tests

Current Test Coverage

- Unit tests exercise components in isolation
- Integration tests exercise “assemblies”
- Functional tests exercise “slices” of the system, according to usage
- System tests exercise the configured system as a whole
- The further we go “up”, the poorer our coverage (generally)

Qui custodiet custodiens?

- Tests verify system functionality
  - who verifies tests?
- Nearer the “surface” of an application, user verification becomes more important
- Traditional spellings are aimed at programmers, not users

Browser Tests Address Gaps

- Web applications are increasingly pushing more behavior into the browser
  - “AJAX” (Javascript + XML/RPC)
  - “Deferred page assembly”
- Traditional testing cannot exercise this functionality well
- Server-side testing which “emulates” browsers may yield fals confidence

Other Advantages

- Cross-browser compatibility tests
  - Browsers are a major source of bugs!
- Test specifications users understand
  - Shared understanding increases acceptance, productivity
  - “FIT” project results
- Bug reporting
  - Blue sky: record user reproducing bug, generate test case

Walkthrough: Testing the CMF

- http://localhost:8081/cmf_tests/workflow

Navigating the Selenium UI

- “Dashboard” consisting of
  - Test Suite <iframe>
  - Test Case <iframe>
  - Control Panel <form>
  - “Application-under-Test” (AUT) <iframe>

Anatomy of a Test Case

- Each test case is a simple HTML page, containing a 3-column table
  - First row is ignored (useful for documentation)
  - Subsequent rows consist of triples: VERB | TARGET | DATA
  - Each row is either an “action” or an “assertion”
- Triples are spelled using a FIT-inspired language, “Selenese”

Selenese Action Verbs

- click, clickAndWait
  - target may be any “clickable” item
- select, selectAndWait
  - target is normally a <select> widget
- type
  - target is an <input> or <textarea> widget
- open
  - target is a URL (as if typed in location bar)
  - Avoid overuse: users don't type there!

Selenese Assertion Verbs

- verifyTextPresent, verifyTextNotPresent
- verifyElementPresent, verifyElementNotPresent
- 'assert*' variants halt the test on failure; 'verify*' variants record failure and continue

Generating Test Cases

- tcpwatch records “wire-level” information
  - Artifacts make “intent” of user hard to infer
  - Ideally, browser-based “gesture” recording might help
- Zelenium provides generator.py
  - Generated test cases often require large-scale fixups

Authoring Test Cases

- Authoring tests can be specification
  - “Fleshing out” use cases
- Simple HTML format, easy to manage in text editor
  - or with tools like Composer

Wrapping Selenium for Zope

- Maik Roeder's Plone wrapper
  - Selenium core application mapped to skins
  - Designed to ship with Plone
  - Favors test cases generated from Python
- Zelenium
  - No Plone / CMF dependency
  - CMF will have them soon
  - Favors “static” test cases
    - don't want to test the tests!

Zelenium Features

- Allows prototyping test cases in the ZMI
- Generates “test suite” tables
- Allows recursive test suites
- Allows including test cases from the filesystem

Zelenium Features (cont'd)

- Capture results, including server-side data
  - '?auto=true' query string trigger
  - Results captured in an object which generates summary report
- Generate test cases from tcpwatch logs
  - Generated versions often need tweaking
- Export test suites as ZIP files
  - Optionally, include Selenium core

Setting up the Test Environment

- Install Zelenium / ExternalEditor products
- Add Zuite instance
- Populate with File instances
- Point at filesystem using property

Issue: Avoiding Test Artifacts

- “Throwaway” site
  - But may need some “known state”
- Teardown code
  - Messy, easy to omit something
- DemoStorage can provide best of both:
  - Underlying storage can have “known state”
  - Teardown is simply restarting appserver

Configuring DemoStorage

- Wrap <demostorage> around normal storage
  - <zodb_db main>
    mount-point /
    <demostorage>
     <zeoclient>
      server localhost:8100
      storage 1
      name zeostorage
      var $INSTANCE/demo_var
    </zeoclient>
   </demostorage>
  </zodb_db>

DemoStorage and ZEO

- “ZEO: don't leave home without it”
  - allows you to make persistent changes to underlying storage
  - debugging on the fly
  - Zope 2.7.6 / ZODB3 3.2.8 fixes bug in DemoStorage-around-ZEO interaction

Resources

- “Selenium site”, http://selenium.thoughtworks.com
- “Zelenium product”, http://www.zope.org/Members/tseaver/Zelenium
- “FIT: Framework for Integrated Test”, http://fit.c2.com/wiki.cgi
- Tres Seaver, tseaver@palladion.com