An introduction to

Robot Framework

http://robotframework.org
Fast facts

• Generic test automation framework
  – Utilizes the keyword-driven testing approach
  – Suitable for both “normal” test automation and ATDD

• Implemented with Python
  – Runs also on Jython (JVM) and IronPython (.NET)
  – Can be extended natively using Python or Java
  – Other languages supported via a remote interface

• Open source
  – Hosted on GitHub, Apache 2 license
  – Sponsored by Nokia Networks
  – Rich ecosystem and very active community
High level architecture

- Test Data
  - Test data syntax
- Robot Framework
  - Test library API
- Test Libraries
- Test Tools
  - Application interfaces
- System Under Test
Simple keyword-driven syntax

*** Test Cases ***

Valid Login

Open Browser To Login Page
Input Username Demo
Input Password Mode
Submit Credentials
Welcome Page Should Be Open

[Teardown] Close Browser
### Data-driven tests

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>USER NAME</th>
<th>PASSWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Username</td>
<td>invalid</td>
<td><code>${VALID PWD}</code></td>
</tr>
<tr>
<td>Invalid Password</td>
<td>invalid</td>
<td>invalid</td>
</tr>
<tr>
<td>Both Invalid</td>
<td>invalid</td>
<td>whatever</td>
</tr>
<tr>
<td>Empty Username</td>
<td><code>${EMPTY}</code></td>
<td><code>${VALID PWD}</code></td>
</tr>
<tr>
<td>Empty Password</td>
<td><code>${VALID USER}</code></td>
<td><code>${EMPTY}</code></td>
</tr>
<tr>
<td>Both Empty</td>
<td><code>${EMPTY}</code></td>
<td><code>${EMPTY}</code></td>
</tr>
</tbody>
</table>
*** Test Cases ***

Valid Login

*Given* browser is opened to login page
*When* user "demo" logs in with password "mode"
*Then* welcome page should be open
Higher level keywords

*** Keywords ***

Open Browser To Login Page
Open Browser  ${LOGIN URL}  ${BROWSER}
Maximize Browser Window
Set Selenium Speed  ${DELAY}
Login Page Should Be Open

Login Page Should Be Open
Title Should Be  Login Page

Input Username
[Arguments]  ${username}
Input Text  username_field  ${username}

Input Password
[Arguments]  ${password}
Input Text  password_field  ${password}
Simple test library API

class Selenium2Library:

    def input_text(self, locator, text):
        """Types given `text` into text field `locator`."""
        print "Typing text `%s` into `%s`." % (text, locator)
        element = self._element_find(locator)
        element.clear()
        element.send_keys(text)

    def title_should_be(self, title):
        """Verifies that current page title equals `title`."""
        actual = self.get_title()
        if actual != title:
            raise AssertionError("Title should have been '%s' but was '%s'." % (title, actual))
        print "Page title is '%s'." % title

# Above is real Selenium2Library code but slightly simplified.
Variables

• Easy to create:

  *** Variables ***
  ${BROWSER}$  Firefox
  ${HOST}$    localhost:7272
  ${LOGIN URL}$  http://${HOST}/
  ${WELCOME URL}$  http://${HOST}/welcome.html

• Override from the command line:

  --variable BROWSER:IE
Tagging

- Free metadata to categorize test cases
- Statistics by tags collected automatically
- Select test cases to be executed
- Specify which test cases are considered critical
Clear reports
Detailed logs

TEST SUITE: Higher Level Login
- Full Name: Login Tests.Higher Level Login
- Source: /home/jth/workspace/seleniumlib/demo/login_tests/higher_level_login.html
- Start / End / Elapsed: 20090415 07:36:29.500 / 20090415 07:36:55.480 / 00:00:25.980
- Overall Status: FAIL
- Message: 3 critical tests, 0 passed, 3 failed
  3 tests total, 0 passed, 3 failed

TEST CASE: Higher Level Valid Login
- Full Name: Login Tests.Higher Level Login.Higher Level Valid Login
- Tags: regression, smoke
- Start / End / Elapsed: 20090415 07:36:29.533 / 20090415 07:36:36.412 / 00:00:06.879
- Status: FAIL (critical)
- Message: Location should have been 'http://localhost:7272/welcome.html' but was 'http://localhost:7272/error.html'

  SETUP: resource.Open Login Page
  KEYWORD: resource.Input Username demo
  KEYWORD: resource.Input Password mode
  KEYWORD: resource.Click Login Button

  KEYWORD: resource.Welcome Page Should Be Open
  Start / End / Elapsed: 20090415 07:36:36.340 / 20090415 07:36:36.374 / 00:00:00.034
  KEYWORD: SeleniumLibrary.Location Should Be ${WELCOME URL}
  Documentation: Verifies that current URL is exactly 'url'.
  Start / End / Elapsed: 20090415 07:36:36.341 / 20090415 07:36:36.374 / 00:00:00.033
  07:36:36.373 INFO Verifying current location is 'http://localhost:7272/welcome.html'.
  07:36:36.374 FAIL Location should have been 'http://localhost:7272/welcome.html' but was 'http://localhost:7272/error.html'

  TEARDOWN: SeleniumLibrary.Close Browser
Different test libraries

- **Standard libraries**
  - Included in normal installation
  - OperatingSystem, Screenshot, String, Telnet, XML, …

- **External libraries**
  - Must be installed separately
  - Selenium2Library, SwingLibrary, DatabaseLibrary, AutoItLibrary, SSHLibrary, HTTPLibrary, …

- **Project and team specific libraries**
Editor support

- RIDE

- Plugins for Eclipse, IntelliJ/PyCharm, SubLime, TextMate, Vim, Emacs, Brackets, Atom, ...
Easy integration

• Test suites are created from files and directories
  – Trivial to store into any version control system
• Simple command line interface
  – Easy to start test execution by external tools
• Output also in XML format
  – All information in machine readable format
  – Outputs from different test runs can be combined
• Plugins for common CI and build tools
  – Jenkins, Ant, Maven
For more information

- Ecosystem front page
  - http://robotframework.org

- Project pages
  - https://github.com/robotframework/robotframework

- Quick Start Guide
  - https://github.com/robotframework/QuickStartGuide

- User Guide
  - http://robotframework.org/robotframework/#user-guide

- Demo projects
  - http://robotframework.org/#documentation