Acceptance tests for VIVO
Why ”acceptance tests”?

- Comes from a military setting.
- Also know as
  - “integration tests”
  - “system tests”
  - “end-to-end tests”
- Definitely not “unit tests”
What is Selenium?

- A browser driver
- Mimics having a user
- Commands like:
  - Direct the browser to “http://localhost:8080/vivo”
  - Click on a certain button
  - Assert element is present on the page
  - Assert text is present on the page
  - …and more…
Previous acceptance tests in VIVO

Add persons, publications, roles…

Use custom forms for awards, education, positions…

Test search results, search configuration, boosting…

Check access privileges

Exercise APIs

Total: 28 “suites”
Process with previous acceptance tests

- Use Selenium-IDE, a Firefox plugin
- Run manually
  - Follow the checklist
  - Setup for each suite
  - Run the suite
  - Observe the results
- Requires about 3 hours
Problems with previous acceptance tests

- Run manually
- Not consistent: timing issues
- Not portable: uploaded files
- Not isolated
- Difficult to write, difficult to maintain
Status of previous acceptance tests

- During the NIH grant: run by Cornell
- VIVO 1.8, 1.9: run by Cornell
- VIVO 1.10: not run
- "Not run" means "Not maintained"
Goals for new acceptance tests

- Fully automated
- Repeatable
- Portable
- Multi-browser
- Easier to write and maintain
The test harness

- Tests written in Ruby, run under RSpec
- Orthogonal:
  - Triple-stores and search index wiped after each “suite” (now “test”)
- Minimal setup:
  - TDB, so no MySQL required
  - Run in JettyRunner, so no Tomcat required
- Setup goal:
  - stub search engine, so no Solr required
Advantages of Ruby

- … or any programming language
  - Selenium also supports Python, Java, PHP, Perl, or C#
- Write sub-routines
  - Login/logout
- Do things that are not provided by the existing vocabulary
  - Wait for indexing, wait for AJAX
Converting the old tests

- 28 suites, 242 tests, and 18423 steps.
- Automated conversion, followed by curation

<table>
<thead>
<tr>
<th>Selenium-IDE</th>
<th>RSpec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suite</td>
<td>Test</td>
</tr>
<tr>
<td>Test</td>
<td>shared_examples</td>
</tr>
</tbody>
</table>
### Old tests

<table>
<thead>
<tr>
<th>open</th>
<th>/vivo/</th>
</tr>
</thead>
<tbody>
<tr>
<td>assertTitle</td>
<td>VIVO</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>link=Log in</td>
</tr>
<tr>
<td>assertTitle</td>
<td>Log in to VIVO</td>
</tr>
<tr>
<td>type</td>
<td>loginName</td>
</tr>
<tr>
<td>type</td>
<td>loginPassword</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>loginForm</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>link=Site Admin</td>
</tr>
<tr>
<td>assertTitle</td>
<td>VIVO Site Administration</td>
</tr>
<tr>
<td>select</td>
<td>VClassURI</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>//input[@value='Add individual of this class']</td>
</tr>
<tr>
<td>verifyTextPresent</td>
<td>Create a new Librarian</td>
</tr>
<tr>
<td>type</td>
<td>firstName</td>
</tr>
<tr>
<td>type</td>
<td>id=middleName</td>
</tr>
<tr>
<td>type</td>
<td>id=lastName</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>submit</td>
</tr>
<tr>
<td>assertTitle</td>
<td>Librarian, Lily Lou</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>css=header &gt; #ARG_2000028 &gt; a.add-ARG_2000028 &gt; img.add-individual</td>
</tr>
<tr>
<td>assertTitle</td>
<td>Edit</td>
</tr>
</tbody>
</table>
New tests

```ruby
shared_examples "Create a Librarian" do
  it "logs in as Admin" do
    vivo_login_from_home_page_as("testAdmin@mydomain.edu", "Password")
  end

  it "decides to add a librarian" do
    $browser.find_element(:link_text, "Site Admin").click
    expect($browser.title).to eq("VIVO Site Administration")

    $browser.find_element(:id, "VClassURI")
    browser_find_select_list(:id, "VClassURI").select_by(:text, "Librarian (vivo)")
    $browser.find_element(:id, "submit").click
    expect($browser.title).to eq("Edit")
    expect(browser_page_text).to include("Create a new Librarian")
  end

  it "creates with the basic info" do
    $browser.find_element(:id, "FirstName").send_keys("Lily")
    $browser.find_element(:id, "middleName").send_keys("Lou")
    $browser.find_element(:id, "lastName").send_keys("Librarian")
    $browser.find_element(:id, "submit").click
    expect($browser.title).to eq("Librarian, Lily Lou")
  end
end
```
What success?

- 4 of 28 running
- 2 weeks to go
How to go forward?

- Run the tests on a regular basis
  - Include in continuous integration?
  - Run the tests after each pull request?
- Write new tests for new functionality
- Improve the tests and the framework
How I plan to leave it

- [https://github.com/j2blake/vivo-acceptance-tests](https://github.com/j2blake/vivo-acceptance-tests)
- A “one-command” execution environment
- A set of passing tests
  - and perhaps some that still fail.
- Instructions on how to install and run
- Suggestions for additional work
Discussion