Testing AJAX Applications with Selenium

Patrick Lightbody

Gomez, Inc.

About Me

- Presently QA Solutions Product Manager @ Gomez, Inc.
  - Also spend part of my time evangelizing open source internally and externally.
- President of OpenSymphony Group, Inc.
- Founder of OpenQA.
- Co-creator of Struts 2.0 (aka WebWork).
- Co-creator of Selenium Remote Control.

Quick Poll

- Audience: Testers? Developers? Managers? Mix?
  - The role of developers and testers typically becomes the same or at least much more tightly integrated when you try to test AJAX.
- Continuous integration: Yes? No? Compile only? Unit tests? Functional tests?
  - CI is a much more difficult problem as applications become more rich and data “bleeds” in to the UI, which is very common in AJAX.
- Toolkit: Home-brewed? Using a framework? Using multiple frameworks?
  - Regardless of the AJAX framework you use (or lack thereof), Selenium can help.

A Growing Problem

- Browser fragmentation
  - Apple growth continues to make Safari a bigger player.
  - Firefox is an even bigger alternative browser...
  - ... but Microsoft has created the biggest fragmentation of all.
- Application fragmentation
  - We live in a “composite” world.
  - Most apps today have at least one external dependency.
  - Example: AdSense, analytics, Google Maps, “Digg This”, etc.
- Performance impact
  - Each browser has strengths and weaknesses in JavaScript execution, page layout, CSS rendering...
  - ... when combined with these composite applications, behavior and performance become difficult to determine.

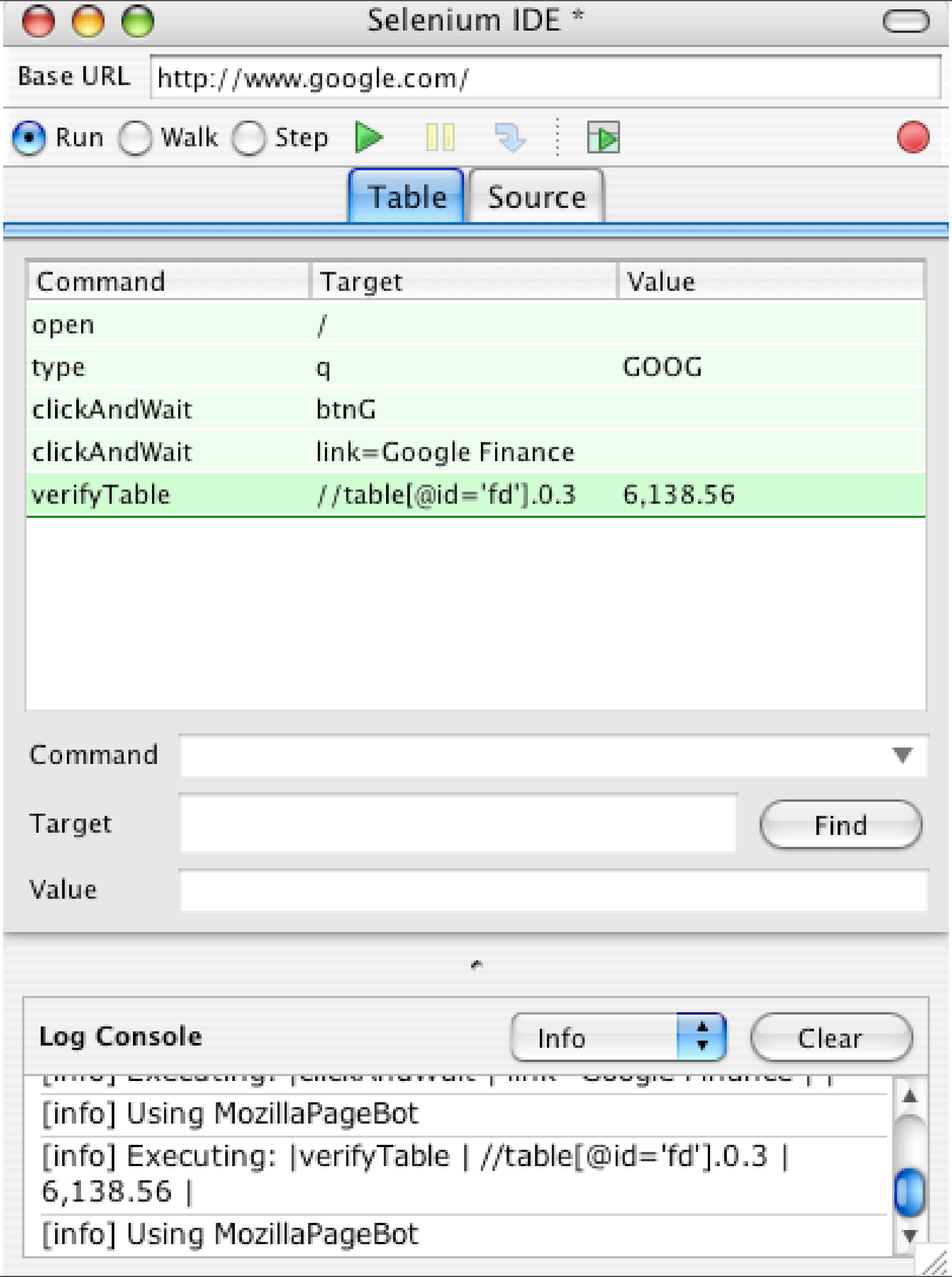
About Selenium

- A cross-platform browser automation tool.
- Written primarily in JavaScript.
- Supports tests written in JavaScript, “Selenese”, or just about any programming language.
- Has several sub-projects
  - Selenium Core
  - Selenium IDE
  - Selenium Remote Control
  - Selenium on Rails
- Is part of OpenQA, the home of many other open source QA tools.

Crash Course in Selenium

- Best way to get started with Selenium is to use it…

DEMO



Selenium IDE *

Base URL http://www.google.com/

Run Walk Step

Table Source

Command	Target	Value
open	/	
type	q	GOOG
clickAndWait	btnG	
clickAndWait	link=Google Finance	
verifyTable	//table[@id='fd'].0.3	6,138.56

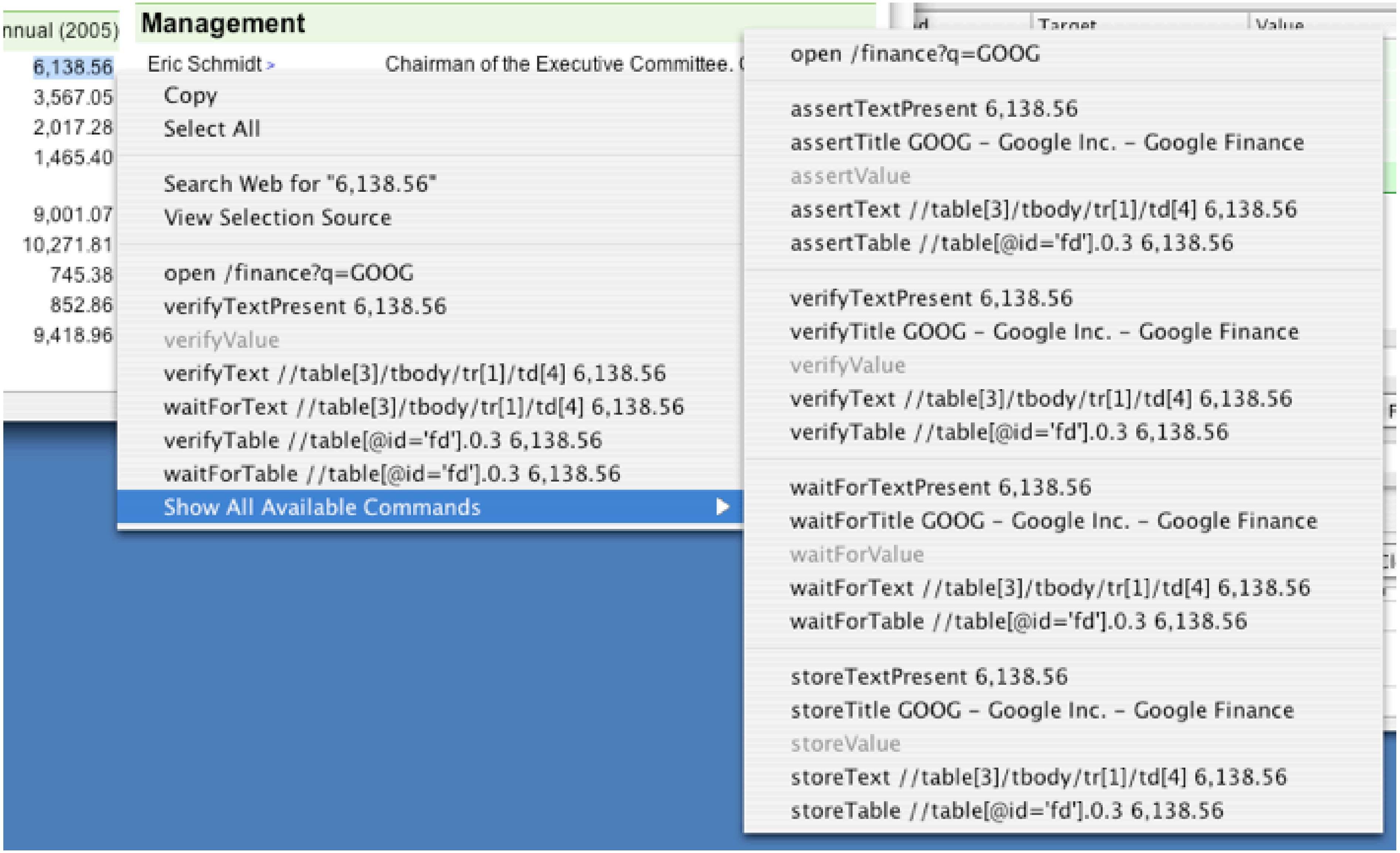
Command

Target

Value

Log Console Info Clear

```
[info] Using MozillaPageBot
[info] Executing: |verifyTable | //table[@id='fd'].0.3 |
6,138.56 |
[info] Using MozillaPageBot
```



```
Copy
Select All

Search Web for "6,138.56"
View Selection Source

open /finance?q=GOOG
verifyTextPresent 6,138.56
verifyValue
verifyText //table[3]/tbody/tr[1]/td[4] 6,138.56
waitForText //table[3]/tbody/tr[1]/td[4] 6,138.56
verifyTable //table[@id='fd'].0.3 6,138.56
waitForTable //table[@id='fd'].0.3 6,138.56
Show All Available Commands
```

```
open /finance?q=GOOG

assertTextPresent 6,138.56
assertTitle GOOG - Google Inc. - Google Finance
assertValue
assertText //table[3]/tbody/tr[1]/td[4] 6,138.56
assertTable //table[@id='fd'].0.3 6,138.56

verifyTextPresent 6,138.56
verifyTitle GOOG - Google Inc. - Google Finance
verifyValue
verifyText //table[3]/tbody/tr[1]/td[4] 6,138.56
verifyTable //table[@id='fd'].0.3 6,138.56

waitForTextPresent 6,138.56
waitForTitle GOOG - Google Inc. - Google Finance
waitForValue
waitForText //table[3]/tbody/tr[1]/td[4] 6,138.56
waitForTable //table[@id='fd'].0.3 6,138.56

storeTextPresent 6,138.56
storeTitle GOOG - Google Inc. - Google Finance
storeValue
storeText //table[3]/tbody/tr[1]/td[4] 6,138.56
storeTable //table[@id='fd'].0.3 6,138.56
```

Selenese Language

- The default language of Selenium.
- A simple language that is structured like Fit (rows inside a table make up commands).
- Has three core components
  - Actions - the things that actually control the browser
  - Accessors - how you work with data in the browser
  - Element Locators - how you identify data in the browser
- Has limited support for variables, but no control structure.

Selenese Actions

- Are where your command actually does something.
- Most action typically take one or two arguments: an element locator and possibly a value.
- All actions have an additional “AndWait” sister-action.
- Examples:
  - check some_checkbox
  - open http://www.google.com
  - type username fred_flintstone

Selense Accessors

- Are always “data related”.
- Typically take only one argument: an element locator.
- Have <u>seven</u> permutations:
  - store (locator, variable)
  - verify and verifyNot (locator, pattern)
  - assert and assertNot (locator, pattern)
  - waitFor and waitForNot (locator, pattern)
- Examples:
  - verifyValue username fred_flintstone
  - waitForElementPresent some_div
  - assertVisible error_box

Selenese Element Locators

- Are how you actually access data to be acted upon or accessed.
- Have a syntax of:
  - [locator_type =] locator_value
- Have support for seven different types. They return an element...
  - id - ... with the specified id
  - name - ... with the specified name
  - identifier - ... with the specified id or name
  - dom - ... that is returned by the evaluated JS expression
  - xpath - ... that is represented by the given XPath expression
  - link - ... that is an href and surrounds the specified text
  - css - that is represented by the given CSS selector

Default Locator Strategy

- dom, if the locator starts with “document.”
  - Example: click document.forms[0].elements[4]
- xpath, if the locator starts with “//”
  - Example: verifyElementPresent //img[contains(@src, ‘close.gif’)]
- identifier, for all others
  - Example: click btnG

Selense Variable Support

- Allows for basic logic in your scripts.
- Use the storeXxx permutation of the accessors:
  - storeValue nameField firstname
  - storeEval ‘Mr’ title
  - assertTextPresent ${title} ${firstName}
- Does <u>not</u> pretend to be full-featured... if you need complex tests, you probably need a more complex language.

Testing AJAX

- The A in AJAX makes testing much more interesting.
- We've seen the "AndWait" variations of commands...
- ... but what about when there never is another page load (Google Maps, Yahoo Mail, and at least partly almost every new web app)?

DEMO

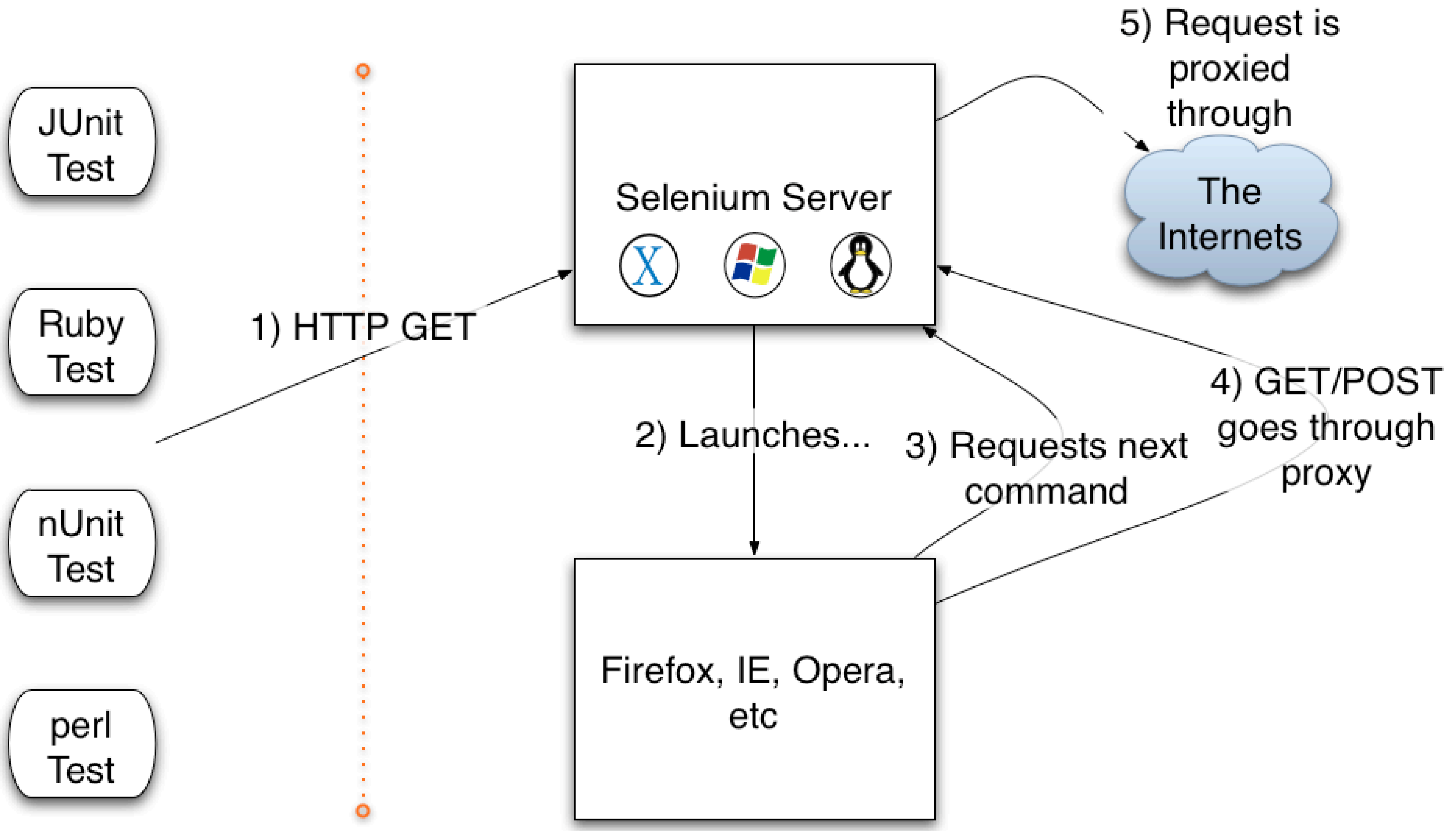
AJAX Toolkits

- You can test any application written on any AJAX toolkit with Selenium, but...
- Some toolkits make it easier than others.
- Dojo
  - Caution: Selenium won't know what your widget IDs are, or how to control them!
- Scriptaculous
  - Tip: scriptaculous does use HTML templates for some of the generated UI (in-place editor), so place a wrapping div with an ID to help.
- You can compensate for the more difficult frameworks by writing your own user extensions.

Advanced Selenium with Selenium RC

- Selenium RC solves three problems:
  - Cross-site scripting restrictions are painful in this composite world (ie: Hotmail -> MS Passport).
  - Not easy to automate the process of running your tests on many browsers (continuous integration).
  - Selenese is a very basic language that offers no reuse and no control structure.
  - Selenium IDE only works on Firefox and Selenium Core requires you to modify your AUT.
- Consists of two parts:
  - A single, standalone Selenium Server
  - Client drivers for C#, Java, perl, PHP, Python, and Ruby.

Selenium RC: How it Works



Getting Started with Selenium RC

- Treat like a daemon process (httpd, sendmail, etc).
- java -jar selenium-server.jar
- The client drivers simply issue HTTP GET commands to the server

```
cmd=getNewBrowserSession&1=*pifirefox&2=http://www.google.com
cmd=open&1=/
cmd=type&1=q&2=GOOG
cmd=clickAndWait&1=btnG
cmd=clickAndWait&1=link=Google Finance
cmd=verifyTable&1=//table[@id='fd'].0.3&2=6,138.56
```

Tips and Tricks

- Give your elements IDs! (Design for testability)
- Make application state easy to reset. Invest in fixtures.
- Use good tools: Firebug, Selenium IDE, XPath Checker.
- When in doubt, try in Selenium IDE.
- Use all the features of Selenium IDE:
  - autocomplete helps you learn the commands
  - “Logs” tab help you debug issues and get help in the forums
  - “Reference” tab documents every single command
  - Find button helps you determine if your locator is correct

Tips and Tricks (Cont.)

- Refactor tests: your test will evolve just like code
- Avoid tight coupling to the page:
  - Bad: //table[3]/tbody/tr[1]/td[4]
  - Bad: session_199238237132_search_results
  - Bad: //img[@src = 'http://staging.acme.com/images/logo.png']
  - Bad: //a[@href = 'http://staging.acme.com/login']
  - Good: //td[text() = 'ISBN XYZ']
  - Good: //div[contains(@id, '_search_results')]
  - Good: //img[contains(@src, 'logo.png')]
  - Good: link=Login
- Don't blindly trust Selenium IDE's scripts - they might work now, but only you can ensure they work later!

Commercial Options

- HostedQA: http://www.hostedqa.com
  - **RealityCheck** - run your Selenium scripts on any browser/OS
  - **RealityView** - check out your site design on any browser/OS
- Built on top of Selenium (where Selenium RC came from!)
- Takes screenshots and a movie of each step along the way.
- Supports advanced test refactoring and analysis.
- Pick up a card from me for a free promo code.

Questions

Questions?

You can also email me at

plightbody@gomez.com

if you have additional questions.