



Automated Web Page Testing with Selenium IDE: A Tutorial

Mary Ann May-Pumphrey

mam_p@yahoo.com

3/10/09

What is Selenium IDE?

- An open-source tool for building automated test suites of web pages
- An extension to Firefox
- Named for the antidote for Mercury poisoning (Mercury Interactive created WinRunner)

Recommended First Steps for Learning Selenium*

1. Learn how to create tests in Selenium IDE.
2. Learn how to run Selenium IDE (SIDE) tests against different browsers/platforms using Selenium RC (Remote Control) server.
3. Learn how to write Selenium tests in any one of several supported languages using Selenium RC drivers.

SIDE Test Case Structure

- 3-column HTML table
- Calls to the [Selenium API](#) (in the first column of each row of table)
- JavaScript code
 - For ***script*** argument to a Selenium command
 - In a user-created extension file
 - Potentially for any command's argument via **`javascript{code}`**

SIDE Test Case Structure

tcl		
open	http://www.deanza.edu/searchcenter/	
store	pass "no pass"	search_string
store	pass.*no[/-]pass	ss_regexp
store	10	hits
type	all_s	\${search_string}
clickAndWait	document.all.btnG	
verifyTitle	Search Results: \${search_string}	
verifyTextPresent	exact:Searched for \${search_string}.	
verifyTextPresent	Results 1 - 10 of about	
verifyValue	//input[1]	\${search_string}
storeXPathCount	//blockquote	blockquotes
storeEval	storedVars['hits']-storedVars['blockquotes']	paragraphs
while	storedVars['blockquotes'] > 0	
verifyText	//blockquote[\${blockquotes}]	regexpi:\${ss_regexp}
storeEval	storedVars['blockquotes']-1	blockquotes
endWhile		
while	storedVars['paragraphs'] > 0	
verifyText	//p[\${paragraphs}]	regexpi:\${ss_regexp}
storeEval	storedVars['paragraphs']-1	paragraphs
endWhile		

Find: regexpi

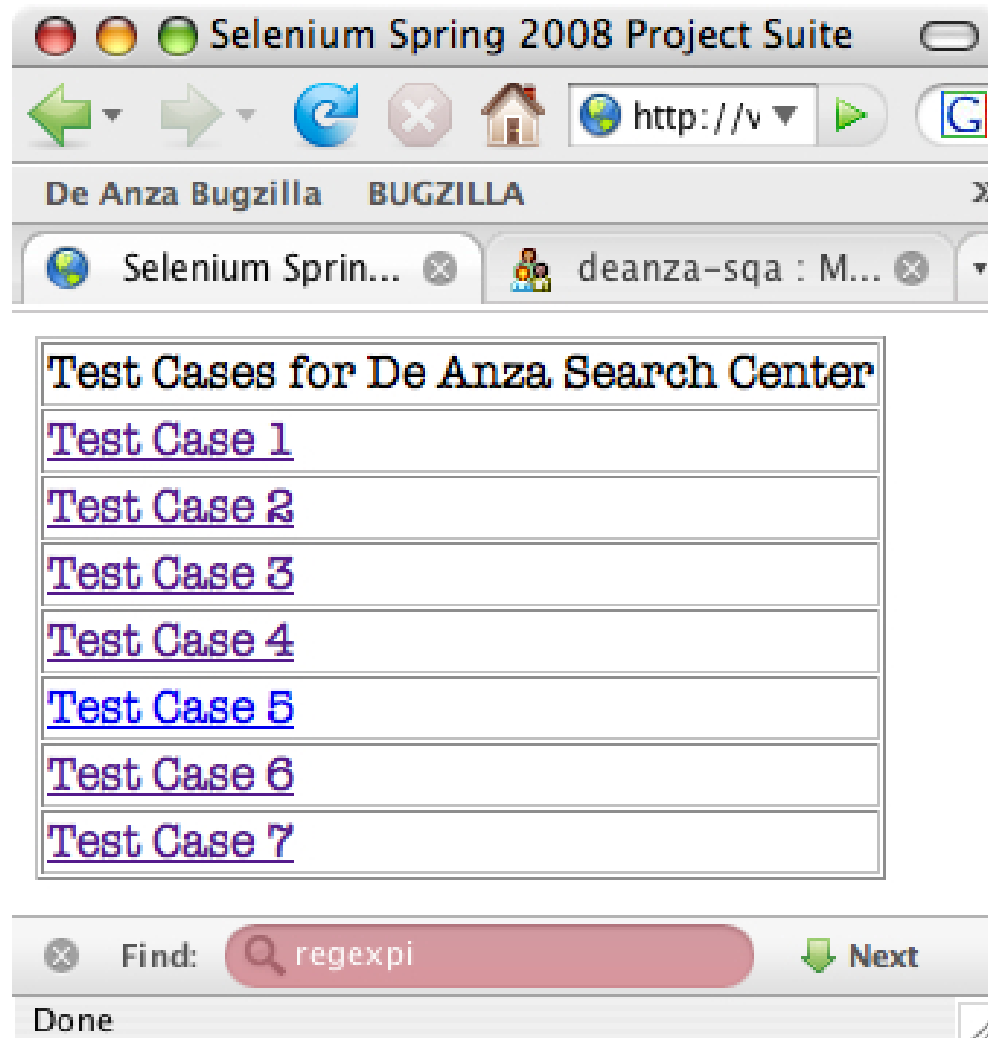
Next Previous Highlight all Match case Phrase

Done

SIDE Test Suite Structure

- 1-column HTML table
- One link to each test, in the only column of each row of table

SIDE Test Suite Structure



Skills Needed to Use SIDE Effectively

- HTML
- JavaScript, including DOM (Document Object Model)
- Regular expressions
- Xpath expressions

Installation of SIDE

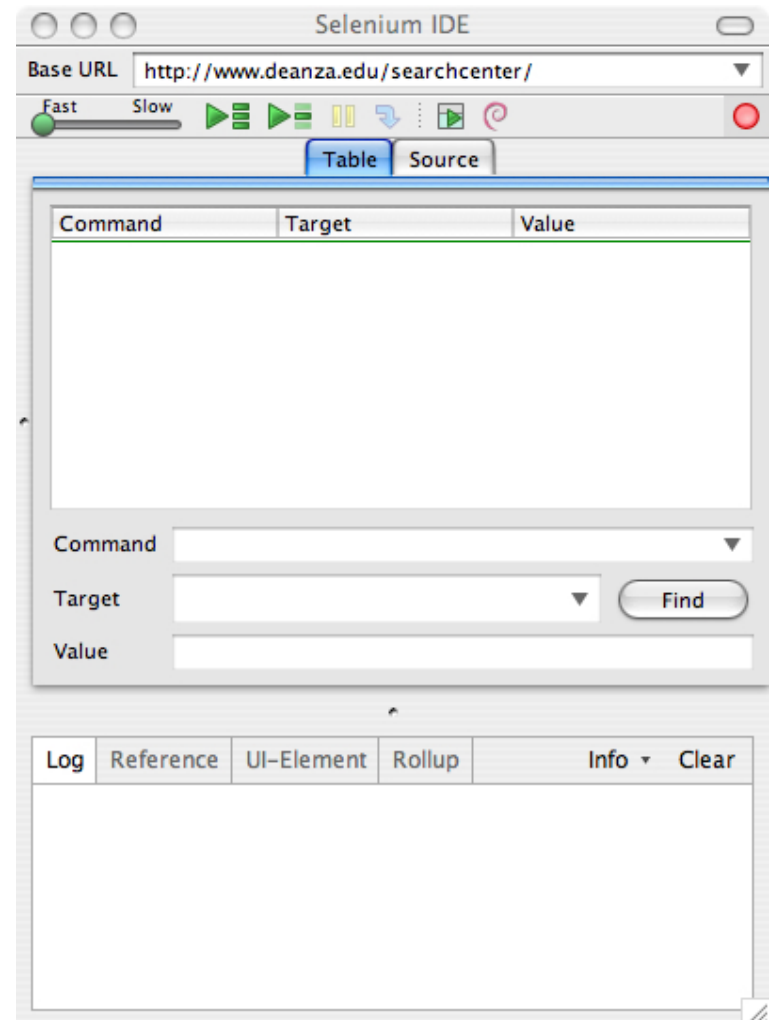
1. Point Firefox at Selenium IDE [installation page](#) on OpenQA web site.
2. Select the download link for 1.0 Beta 2.
3. If nothing happens, then select the **Edit Options** button just above the page. Then select the **Allow** button on the **Allowed Sites** pop-up, followed by the **Close** button.

Installation of SIDE

4. Select the download link for 1.0 Beta 2 again.
5. Select the **Install Now** button of the **Software Installation** window.
6. Select the **Restart Firefox** button of the **Add-ons** window.

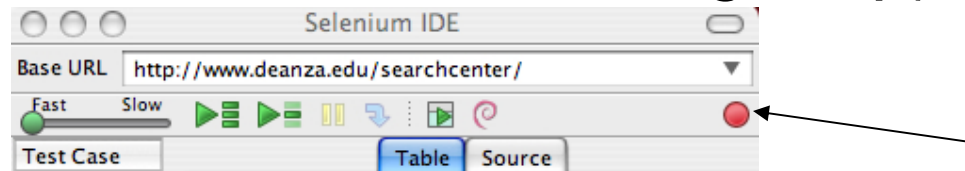
Bringing Up SIDE

Firefox's
Tools => Selenium IDE



Example: Link-Checking

1. Point Firefox browser at SUT, [De Anza A-Z Directory](http://www.deanza.edu/searchcenter/).
2. Click on the red **Record** button to start recording if needed (it is already on by default when SIDE is first brought up).

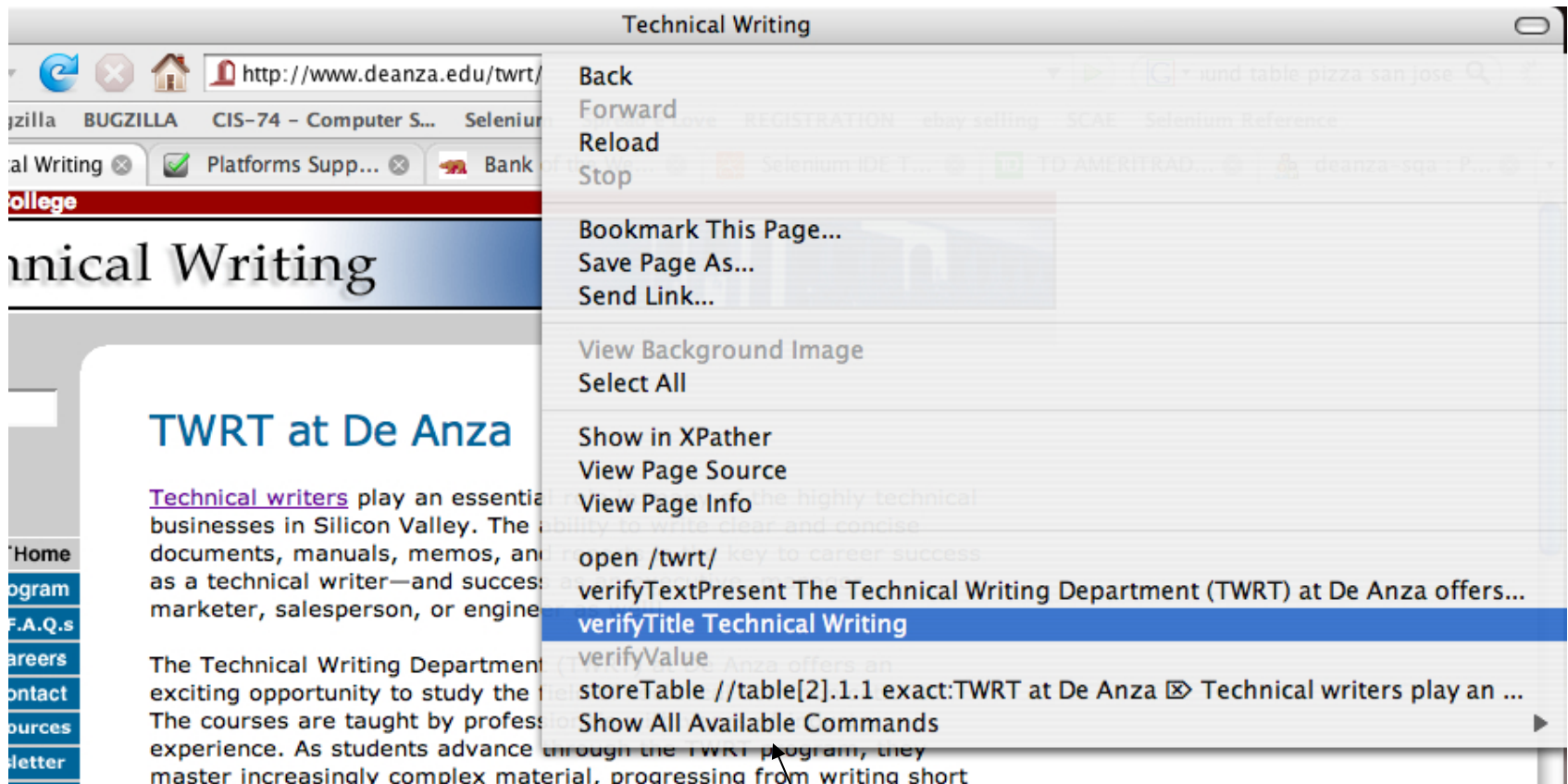


3. Click on browser window to give it focus.

Example: Link-Checking

4. Click on letter **T** from alphabetical horizontal navigation bar.
5. Select **Technical Writing** link, and wait for new page to load.
6. Select **verifyTitle Technical Writing** from context menu.

Example: Link-Checking



Always look here if desired
Selenium command isn't listed

Example: Link-Checking

7. Select browser's **Back** button.
8. Repeat the last three steps for the **Transcripts** link.
9. Click the red **Record** button to stop recording.

Example: Link-Checking

The screenshot shows the Selenium IDE interface with the following components:

- Base URL:** `http://www.deanza.edu/searchcenter/`
- Execution Controls:** A slider between 'Fast' and 'Slow', and buttons for 'Run', 'Stop', 'Pause', 'Undo', 'Redo', and 'Refresh'.
- Table View:** A table with columns 'Command', 'Target', and 'Value'. It contains a sequence of commands for link checking. An arrow labeled 'Captured code' points to the 'Value' column.
- Command Entry Fields:** Fields for 'Command', 'Target', and 'Value' with a 'Find' button.
- Log Panel:** A panel at the bottom with tabs for 'Log', 'Reference', 'UI-Element', and 'Rollup'. It includes 'Info' and 'Clear' buttons.

Command	Target	Value
open	<code>http://www.deanza.edu/directory/dir-az.html</code>	
click	<code>link=T</code>	
clickAndWait	<code>link=Technical Writing</code>	
verifyTitle	<code>Technical Writing</code>	
click	<code>link=Transcripts</code>	
verifyTitle	<code>Transcripts</code>	

Example: Link-Checking

- At this point, all of the user actions except the clicks on the **Back** button have been recorded.
- To take care of these clicks, add calls to the **goBackAndWait** command.

Inserting Selenese Commands from Table View

1. Select **Table** tab of Selenium IDE.
2. Select third **click/clickAndWait**.
3. Bring up the context menu of IDE and select **Insert New Command**.

Inserting Selenese Commands from Table View

The screenshot shows the Selenium IDE interface with the Table View selected. A context menu is open over the 'click' command row, which has 'link=Transcripts' as its target. The menu includes options like Cut, Copy, Paste, Delete, Insert New Command, Insert New Comment, Clear All, Toggle Breakpoint, Set / Clear Start Point, and Execute this command. Annotations with arrows point to the 'Insertion point' (the row being edited), 'Arg-1' (the Command column), 'Arg-2' (the Value column), and the 'Context menu'.

Command	Target	Value
open	http://www.deanza.edu/directory/dir-az.html	
click	link=T	
clickAndWait	link=Technical Writing	
verifyTitle	Technical Writing	
click	link=Transcripts	
verifyTitle	Transcripts	

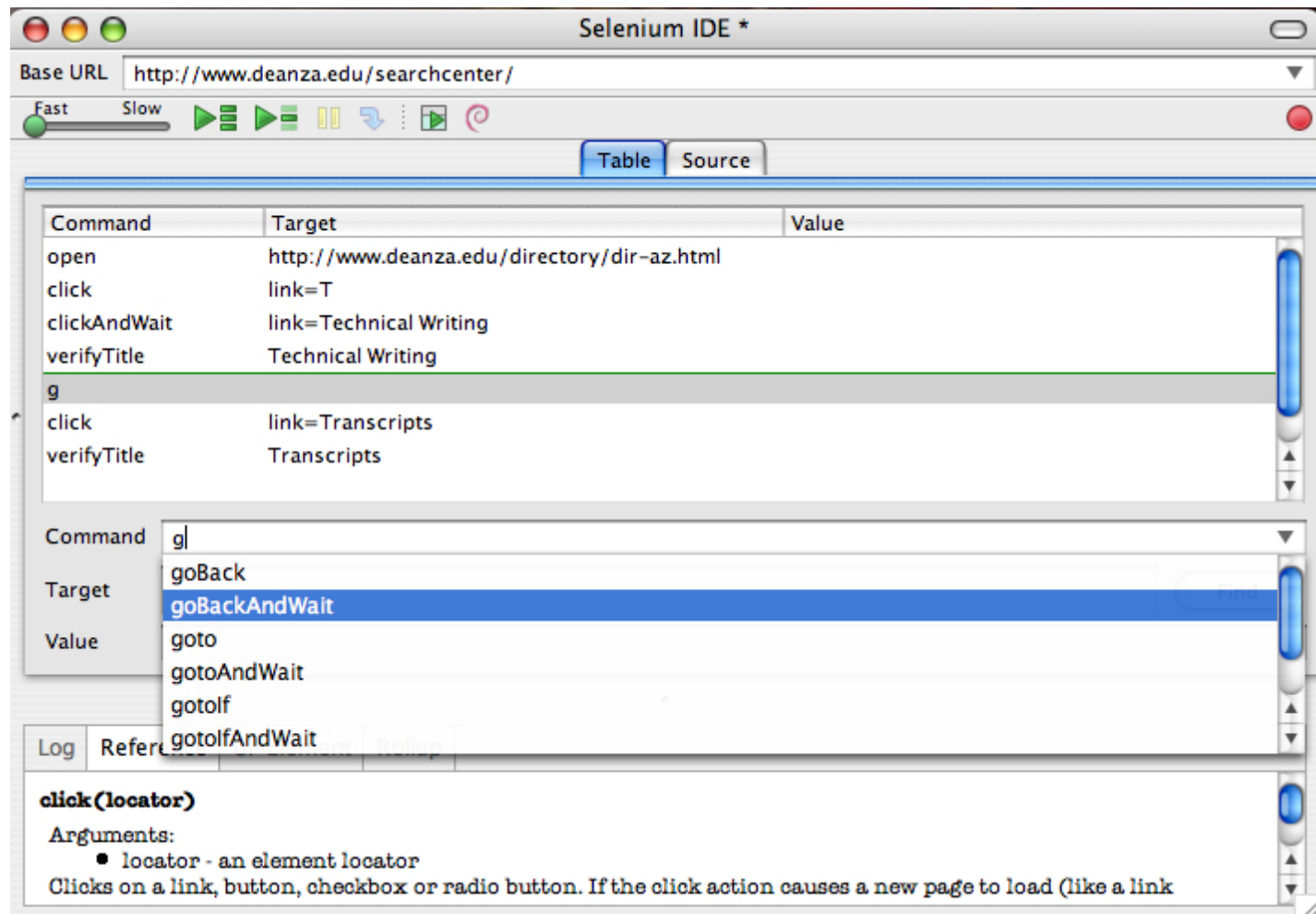
Annotations:

- Insertion point: Points to the 'click' row in the table.
- Arg-1: Points to the 'Command' column header.
- Arg-2: Points to the 'Value' column header.
- Context menu: Points to the menu that appears when right-clicking on a row.

Inserting Selenese Commands from Table View

4. Type "g" into **Command** field.
5. Select **goBackAndWait** from command menu.

Inserting Selenese Commands from Table View



Replaying a Test

1. Select **File=>Save Test Case As** to save the test.
2. Drag the speed slider to **Slow**.
3. Select the second green right-pointing arrow (*Play from the beginning or start point*).

Replaying a Test

The screenshot shows the Selenium IDE interface with the following components and annotations:

- Speed slider:** Located at the top left, with a slider bar and a play button icon.
- Playback glyph:** A small icon (a green play button) located next to the speed slider.
- Currently executing line:** A blue vertical bar on the right side of the test table, indicating the current position in the test sequence.
- Erase log contents:** A button labeled "Clear" located at the bottom right of the log panel.
- Results:** The log panel at the bottom, showing the execution history of the test.

The test table contains the following commands:

Command	Target	Value
open	http://www.deanza.edu/directory/dir-az.html	
click	link=T	
clickAndWait	link=Technical Writing	
verifyTitle	Technical Writing	
goBackAndWait		
click	link=Transcripts	
verifyTitle	Transcripts	

The log panel shows the following results:

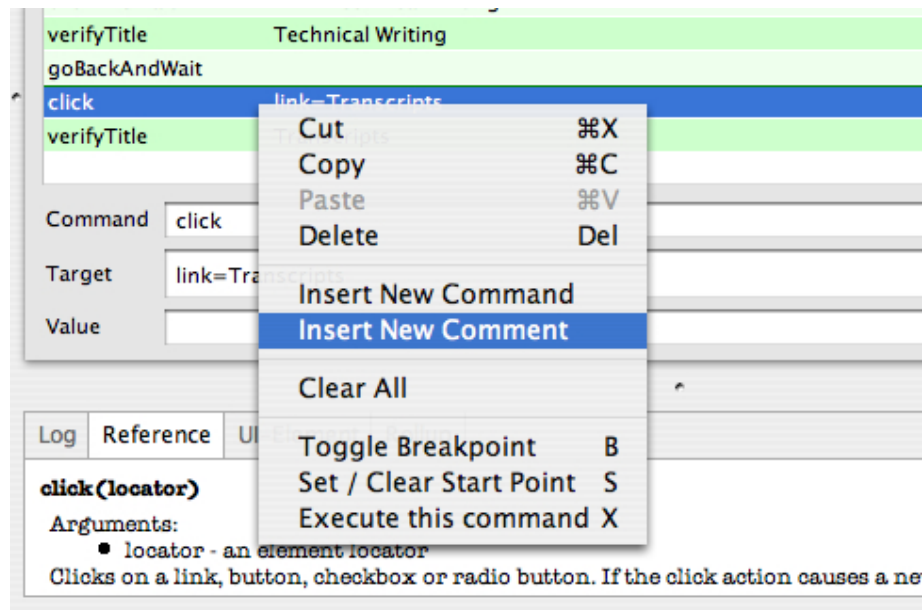
```
[info] Executing: |open | http://www.deanza.edu/directory/dir-az.html | |
[info] Executing: |click | link=T | |
[info] Executing: |clickAndWait | link=Technical Writing | |
```

Analyzing the Results

- From **Table** view, observe the green vs. red shading.
- From **Table** or **Source** view, select the **Log** tab and look for errors (bold red font).

Adding Comments from Table View

Bring up the context menu of IDE and select **Insert New Comment**.



Adding Comments from Table View

The screenshot shows the Selenium IDE interface in 'Table' view. The Base URL is `http://www.deanza.edu/searchcenter/`. The table contains the following commands:

Command	Target	Value
open		<code>http://www.deanza.edu/directory/dir-az.html</code>
click		<code>link=T</code>
<i>Test of Technical Writing link's target</i>		
clickAndWait		<code>link=Technical Writing</code>
verifyTitle		<code>Technical Writing</code>
goBackAndWait		
<i>Test of Transcripts link's target</i>		
click		<code>link=Transcripts</code>

Below the table, the 'Command' field is set to `Test of Transcripts link's target`. The 'Target' and 'Value' fields are empty. A 'Find' button is located to the right of the Target field.

At the bottom, the 'Log' tab is active, showing the following log entry:

```
clickAndWait(locator)
Generated from click(locator)
Arguments:
  • locator - an element locator
```

Comments

Assertions

Prefixes	Sample Suffixes
assert	Alert NotAlert
verify	Confirmation NotConfirmation
waitFor	TextPresent TextNotPresent Title NotTitle

Selenium IDE Variables

- Created primarily via store* commands:
 - store (expression,variableName)
 - storeAlert (variableName)
 - storeAllButtons (variableName)
 - and dozens of others!
- Accessed via **`${variable}`** syntax, e.g.,
 - **`echo variable = ${variable}`**

Patterns, Locators, Scripts

Most arguments required by the Selenium API fall into three categories:

Patterns

Locators

Scripts

Patterns: glob (the default)

glob:*pattern*

* (anything/nothing)

? (any single character)

Example: **verifyTitle glob:Technical *Writing**

More robust test now--it allows additional characters, such as extraneous spaces, to be present between the words.

Patterns: regexp

regexp:*regexp*

regexpi:*regexp* (case-insensitive match)

All JavaScript's regular expression metacharacters are supported, including:

- * (0 or more of preceding character)

- + (1 or more of preceding character)

- ? (0 or 1 of preceding character)

- {**n**} (*n* of preceding character)

- a|b** (alternation: a OR b)

- [aeiou]** (character class: any one of the chars)

Patterns: regexp

Example: Verifying search results for course CIS-200W

/ Allow 0 to ~ spaces between department & dash */*
verifyTextPresent **regexp:CIS *-200W-**

/ Disallow course numbers 000-199 and 300-999 */*
verifyTextNotPresent **-[0-13-9][0-9][0-9].?-**

/ Disallow course numbers 201-209 */*
verifyTextNotPresent **regexp:-20[1-9].?-**

Patterns: exact

exact:*string*

Seems a bit pointless!

The default **glob:** pattern-matching without any special characters does the same thing.

Locators: identifier=

identifier=*identifier*

- Specifies first element with matching **id** attribute if one exists; otherwise, first element with matching **name** attribute.
- Default locator tag if identifier doesn't start with **document** (DOM locator) or **//** (Xpath).

/* Enter a stored search string into input field */

type **all_s** **\${search_string}**

Locators: id=

id=*id*

Specifies first element with matching **id** attribute.

/ Enter a stored search string into input field */*

type **id=all_s** **\${search_string}**

Locators: name=

name=*name*

Specifies first element with matching **name** attribute.

/ Enter a stored search string into input field */*

type **name=q** **\${search_string}**

Locators: dom=

dom=*javascriptExpression*

document.*restOfJavascriptExpression*

/ Choose item from drop-down with specified label */*

select

document.schsearch.Uniq_Course_ID

Computer Information Systems (CIS)

Locators: xpath=

xpath=*xpathExpression*

//restOfXPathExpression

/ Select checkbox with value attribute of “Tu” */*

click *//input[@value='Tu']*

/ Select checkbox with id attribute containing “Thu” */*

click *//input[contains(@id,'Thu')]*

Locators: xpath=

For more info on using **xpath** locators with Selenium, including how the Firefox extension **xpather** can help, see [Help with Xpath](#) on the Openqa Wiki.

Locators: link=

link=*textPattern*

Note that the argument is a **textPattern**, not a **textString**!

/ “Assessment and Placement Information” link */*

click

link=regex:Assessment +.* +Placement +Information

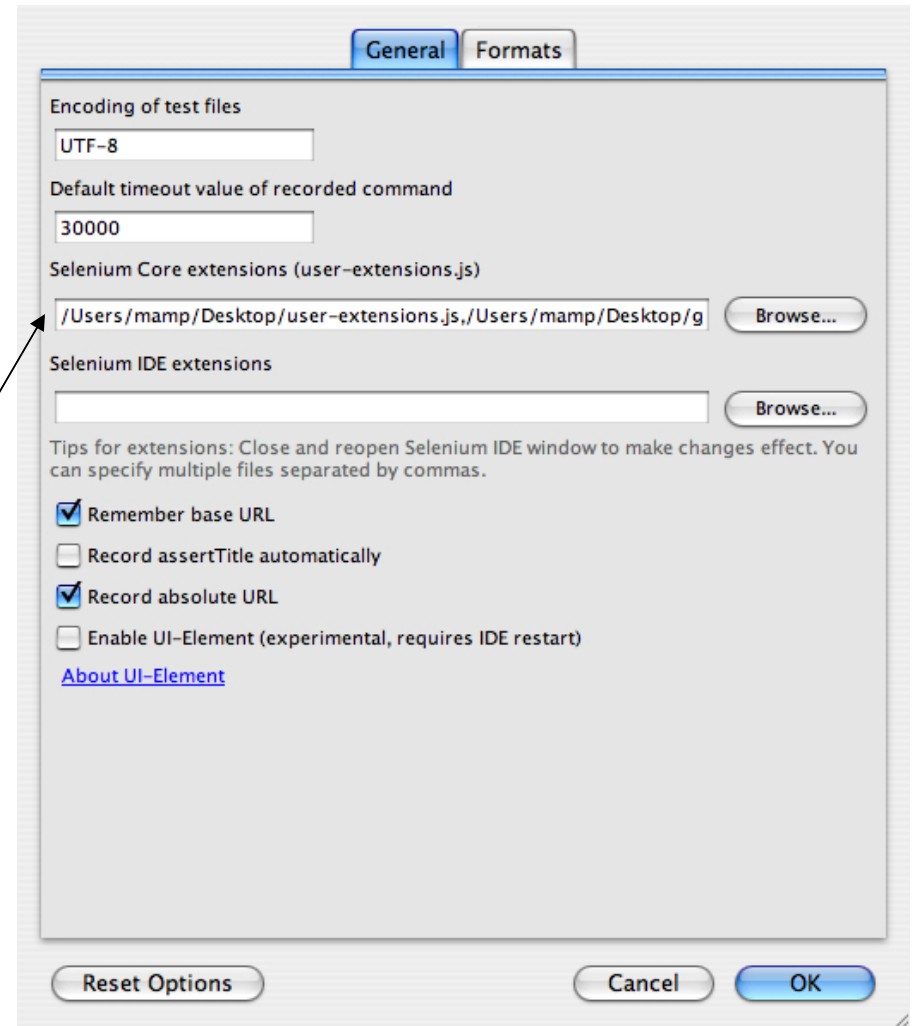
Scripts

storeEval (<i>script,</i> <i>variableName</i>)	storeEval this.browserbot.bodyText().match(/ of about +([0-9]+)\.)/[1] hits
storeExpression (<i>expression,</i> <i>variableName</i>)	storeExpression javascript {this.browserbot.bodyTe xt().match(/of about +([0-9]+)\.)/[1]} hits

Extensions to SIDE

Write your own or download one such as [goto_sel_ide.js](#) which provides loops and conditionals.

Specify via
Options=>Options
window



SIDE Gotchas/Shortcomings

- Log file cannot be saved other than via copy/paste.
- User clicks on links are sometimes recorded as **click** rather than **clickAndWait**.
- Test suite functionality is minimal.
- No built-in subroutines, loops, or conditionals.
- Xpath implementation seems buggy.

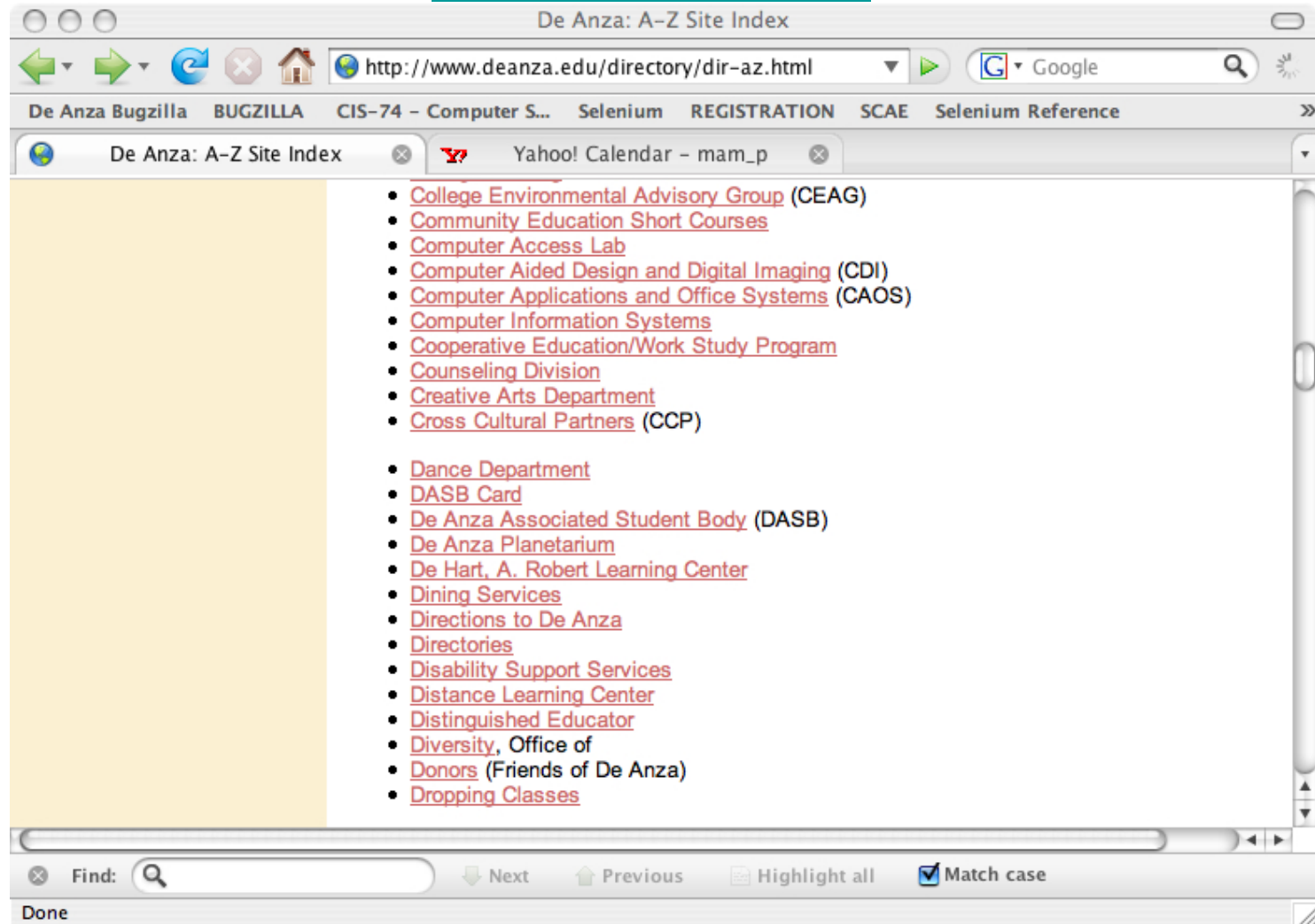
Execution on Other Platforms

1. [Download](#) the Selenium RC server.
2. `java -jar selenium-server.jar -htmlSuite
browserString startURL suiteFile resultFile`

Example:

```
java -jar selenium-server.jar -htmlSuite "*chrome"  
http://www.ebay.com  
~/Desktop/De*/SIDE-S2008/testsuite/testsuite.html  
/tmp/results.html
```

Demo: Checking ***All*** De Anza A-Z Links



Demo: Checking ***All*** De Anza A-Z Links

JavaScript extension file (the ***golden*** file):

```
var a_z = new Array();  
// One pair/link: link-text followed by landing page title  
a_z.push("About De Anza","About De Anza");  
a_z.push("Calendar","Academic Calendar/Final Exam Schedule");  
a_z.push("Academic Freedom Policy","Academic Freedom");  
.  
.  
.  
a_z.push("Astronomy Department","regex:De Anza College  
Physical Sciences, Math., & Engineering Division  
(PSME).*http://nebula.deanza.edu");
```

Demo: Checking ***All*** De Anza A-Z Links

Test case:

Requires
extension→

deanza_a-z_dir		
open	http://www.deanza.edu/directory/dir-az.html	
storeEval	a_z.length	numLinks
storeExpression	0	index
while	(\${index} < \${numLinks})	
storeEval	a_z[\${index}]	linkText
storeEval	a_z[\${index}+1]	title
echo	Checking link: \${linkText}	
clickAndWait	link=\${linkText}	
pause	1000	
verifyTitle	\${title}	
goBackAndWait		
storeEval	\${index}+2	index
endWhile		

Find: 12943 Next Previous Highlight all Match case

In-Depth SIDE Courses

- **De Anza College CIS-140:** *Automated Web Page Testing with Selenium IDE*
- **Santa Clara Adult Education's High Tech Academy:** *Software Test Automation using Selenium IDE*
- **Portnov QA School**

Questions? Want a copy of
these slides?

mam_p@yahoo.com