# CS1699 - Lecture 15 - An Introduction to Testing Web Applications with Selenium


Image by Tom Mooring

# CS1699 - Lecture 15 - An Introduction to Testing Web Applications with Selenium


Image by Tom Mocking

# Testing Web Apps

So far, we've talked about testing systems with simple input/output.

Manual testing - Humans perform operations

Unit testing - Operating on individual methods and functions

Systems Testing - (discussed) Textual input/output

## But....

Modern programs tend to have more complicated interfaces (GUIs, web pages, etc.)

How can we test those?

# But....

Modern programs tend to have more complicated interfaces (GUIs, web pages, etc.)

How can we test those?

# The answer may surprise you!

We can actually use many of the same tools and techniques to test more complicated interfaces. Now that you understand the basics of automated testing, it's possible to take what you've learned and apply it to a more complex interface.

But this remains -
EXPECTED BEHAVIOR vs OBSERVED BEHAVIOR

We can actually use many of the same tools and techniques to test more complicated interfaces. Now that you understand the basics of automated testing, it's possible to take what you've learned and apply it to a more complex interface.

But this remains -
EXPECTED BEHAVIOR vs OBSERVED BEHAVIOR

# Testing the Web

The web is just text (HTML).

Specially formatted text, but text nonetheless.

Really, everything comes down to 1s and 0s. If your computer can display it, it can be tested.*

* It doesn't even need to be displayed, e.g. music can be tested.

Theoretically, then, we could test the web like so:

```
@Test
public void testWeb() {
  String expectedHtml = "<body><strong>Hello, world!</strong></body>";
  String pageText = getPage("http://example.com");
  assertEquals(expectedHtml, pageText);
}
```

Any downsides?

1. Change the page, change the entire test
   --> Fragile tests!
2. What about JavaScript?
3. Unreadable
4. Simplistic and low-level
   --> Kind of like programming in assembly
5. No understanding of e.g. links, textboxes

## Solution

Web Testing Framework

Think of these as a higher-level programming language for testing.

```
1 pushl %ebp #
2 movl %esp, %ebp #,
3 subl $4, %esp #,
4 movl $1, -4(%ebp) #, a
5 leal -4(%ebp), %eax #,
6 addl $1, (%eax) #, a
7 leave
8 ret
```

```
void function1() {
  int a = 1;
  a = a + 1;
}
```

Or think of them as a library of functions.

Outside of class, who writes a linked list? Or Hashmap? Just import a library instead of writing it yourself.

Remember, we want to use the same principles we used for manual and our previous automated testing. We're just working at a higher layer of abstraction.

The web is just text (HTML).

Specially formatted text, but text nonetheless.

Really, everything comes down to 1s and 0s. If your computer can display it, it can be tested.*

* It doesn't even need to be displayed, e.g. music can be tested.

Theoretically, then, we could test the web like so:

```
@Test
public void testWeb() {
  String expectedHtml = "<body><strong>Hello,
world!</strong></body>";
  String pageText = getPage("http://example.com");
  assertEquals(expectedHtml, pageText);
}
```

Any downsides?

1. Change the page, change the entire test
   --> Fragile tests!
2. What about JavaScript?
3. Unreadable
4. Simplistic and low-level
   --> Kind of like programming in assembly
5. No understanding of e.g. links, textboxes

# Solution

Web Testing Framework

Think of these as a higher-level programming language for testing.

```
1 pushl %ebp #
2 movl %esp, %ebp #,
3 subl $4, %esp #,
4 movl $1, -4(%ebp) #, A
5 leal -4(%ebp), %eax #,
6 addl $1, (%eax) #, A
7 leave
8 ret
```

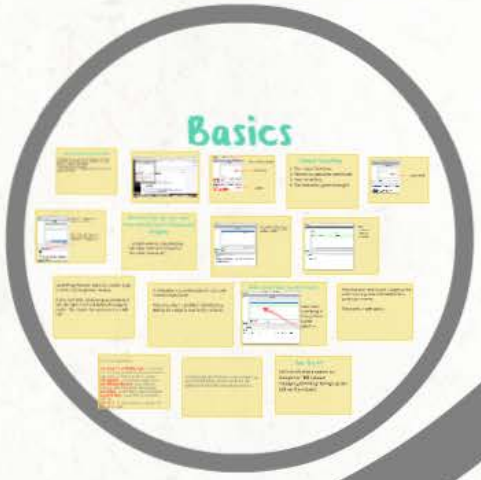
```
void function1() {
  int a = 1;
  a = a + 1;
}
```

Or think of them as a library of functions.

Outside of class, who writes a linked list? Or Hashmap? Just import a library instead of writing it yourself.

Remember, we want to use the same principles we used for manual and our previous automated testing.  We're just working at a higher layer of abstraction.

# Testing the Web with Selenium

# Selenium

### Selenium

An open-source web testing framework.
Battle-hardened.
Works with Windows, OS X, Linux, other OSes
Works with Java, Ruby, Python, other languages
Works with most modern web browsers.
Has its own IDE.
Can also be used for quick scripting.

### Why the name Selenium?

One of their competitors was "Mercury."

Mercury poisoning is cured by Selenium pills.

### Selenium Components

1. Selenium IDE
2. Selenium API
3. Selenium WebDriver (formerly RC)
4. Selenium Grid

### Selenium IDE

Firefox add-on
Allows you to record and re-play macros

### Selenium API

Behind the scenes, this is how the code you write interacts with the Selenium engine itself. You probably won't be dealing much with this unless you're porting Selenium to a new language.

### Selenium WebDriver

Controls a browser instance and actually executes commands.

### Selenium Grid

Allows you to run Selenium tests on remote machines.

Often, development shops have dedicated testing machines so that you don't need to have your development machine occupied while the tests execute.

The point -
Selenium is a very complex, complete framework. We're really just going to be seeing the tip of the iceberg in this class.

It has other uses aside from testing, as well (web macros).

### Other Web Testing Frameworks

Apache JMeter - Used for load testing and performance testing
iMacros - More GUI-like scripting. I used it for easy scraping.
Watir - Directly drives a test instead of taking over a browser.
TestComplete - A similar tool to Selenium but mostly restricted to Microsoft stacks

### So we know what Selenium is...

Let's see how to use it!

# Selenium

An open-source web testing framework.
Battle-hardened.
Works with Windows, OS X, Linux, other OSes.
Works with Java, Ruby, Python, other languages.
Works with most modern web browsers.
Has its own IDE.
Can also be used for quick scripting.

# Why the name Selenium?

One of their competitors was "Mercury."

Mercury poisoning is cured by Selenium pills.

# Selenium Components

1. Selenium IDE
2. Selenium API
3. Selenium WebDriver (formerly RC)
4. Selenium Grid

# Selenium IDE

Firefox add-on

Allows you to record and re-play macros

# Selenium API

Behind the scenes, this is how the code you write interacts with the Selenium engine itself. You probably won't be dealing much with this unless you're porting Selenium to a new language.

# Selenium WebDriver

Controls a browser instance and actually executes commands.

# Selenium Grid

Allows you to run Selenium tests on remote machines.

Often, development shops have dedicated testing machines so that you don't need to have your development machine occupied while the tests execute.

The point -
Selenium is a very complex, complete framework. We're really just going to be seeing the tip of the iceberg in this class.

It has other uses aside from testing, as well (web macros).

# Other Web Testing Frameworks

Apache JMeter - Used for load testing and performance testing
iMacros - More GUI-like scripting.  I used it for easy scraping.
Watir - Directly drives a test instead of taking over a browser.
TestComplete - A similar tool to Selenium but mostly restricted to Microsoft stacks

So we know what Selenium is....

Let's see how to use it!

# Basics

## Setting Started with Selenium


## Simple Scripting

1. File... New Test Case
2. Record an operation (red circle)
3. Stop recording
4. Run test suite (green triangle)


## Also note that you can move steps around by just clicking and dragging

... or add comments by selecting "Add New Comment" instead of "Add New Command"


waitForPageToLoad - Waits for another page to load, with an optional timeout.

If you don't wait, Selenium goes as fast as it can, and will do a check before the page is ready. This means that your assertions will fail!

clickAndWait -> a combination of "click" and "waitForPageToLoad"

There are other "...andWait" variants since waiting for a page to load is very common.

## Hello, assertions, my old friend...

Assert that something is true, just like a jUnit assertion

Note that you need to give a target to the assertions, e.g., that it should be on a particular element.

This is simple with select.

Lots of fun assertions...

**assertText / assertTextPresent** - Assert that text exists (on an element (former) or entire page (latter)). Note that this is a regex!
**assertCookie** - Assert that a cookie exists.
**assertElementPresent** - Assert that an element exists somewhere on the page.
**assertAlert** - Assert that an alert took place.
**assertEditable** - Assert that an element is editable.
**assertEval** - Evaluate some JavaScript and assert the result.

All of the asserts also have a not-variant, e.g., assertNotEditable, which checks for the opposite of what the original assert does.

## You Try It!

Let's check that a search on Google for "Bill Laboon metaprogramming" brings up my talk on the subject.

# Getting Started with Selenium

1. Download Firefox (getting Selenium IDE working in Chrome is possible, but more difficult.  Google ChromeDriver for details)
2. Go to http://docs.seleniumhq.org/
3. Download and install Selenium
4. Click on the "Se" icon in the upper right-hand corner, or go to Tools -> Selenium IDE

Firefox File Edit Actions Options Window Help

Mon 12:13 PM

SelDEReleaseNotes - selenium - Browser automation framework - Google Project Hosting

Selenium IDE 2.5.0 *

Base URL http://lit-bayou-7912.herokuapp.com/

Fast Slow

Test Case

Untitled *

Table Source

| Command | Target | Value |
|---|---|---|

Command

Target Select Find

Value

Runs: 1

Failures: 1

Log Reference Expert UI-Element Rollup Info Clear

[error] Element head not found

[info] Executing: |waitForTitle | | |

[error] locator not found: form

[error] Timed out after 30000ms

1:12-1:14 Statue in fo

1:14-1:16 D.C. skylin

1:17-1:19 Roman stat

1:20-1:21 D.C. Skylin

1:21-1:26 Capitol and

1:26-1:32 D.C. Skylin

U:--- house-of-cards-o

^X 1

```
<body>

<form a
<p>
  <labe
  <text
</texta
<p>
  <inpu
  <inpu
  <inpu
</p>
</form>

</body>
</html>
```

ium/wiki/SelDEReleaseNotes

Google

My favorites ▾ | Si

mework

Search proje

Wiki Issues Source

Search

Updated Jan 10, 2014 by Samit.Badle

elease Notes

omatic playback at a certain time or periodic intervals.

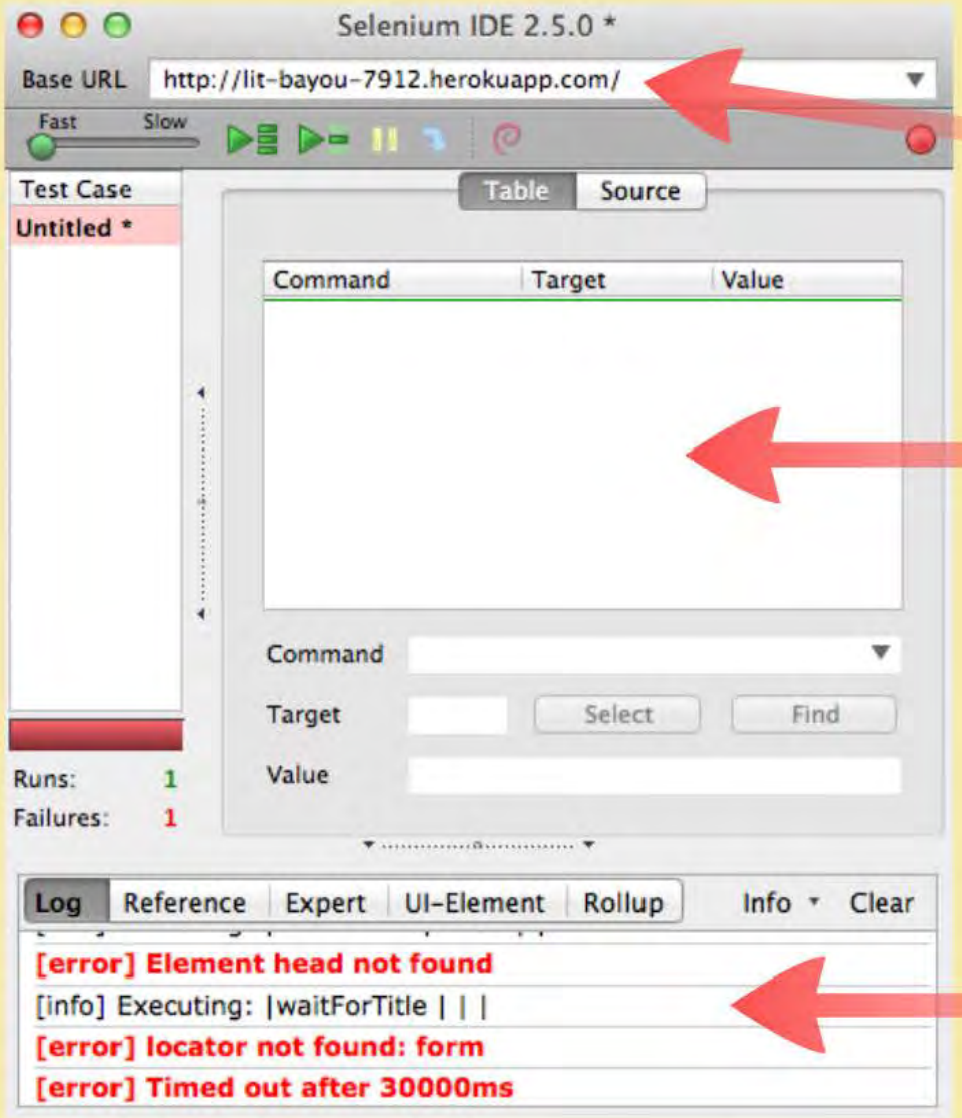
a command by clicking on the element in the browser window (http://blog.reallysimplethoughts.com/2014/01 dating-element-locators-the-easy-way/)

te from any test case (Using right click menu) - ~~issue 1987~~

- Enh - Add a new test case using a keyboard shortcut (ctrl-N or cmd+N)
- Fix - Fixed delete test case through right click menu was sometimes disabled - ~~issue 5003~~
- Fix - Fixed Selenium IDE icon is sometimes not visible - ~~issue 5712~~
- Fix - Fixed selectWindow using a variable - ~~issue 3270~~
- Some minor changes

## 2.4.0

- Enh - Base URL history, recent test cases and recent test suites can be cleared - ~~issue 6135~~
- Enh - Special key now have shorter names (http://blog.reallysimplethoughts.com/2013/09/25/using-special-keys-in-selenium-ide-part-2/)
- Enh - Support for user extensions in Webdriver playback - ~~issue 5675~~
- Fix - The recording of entering text in fields uses type instead of sendKeys.
- Enh - When developer tools are active, the last open test case or suite is automatically opened



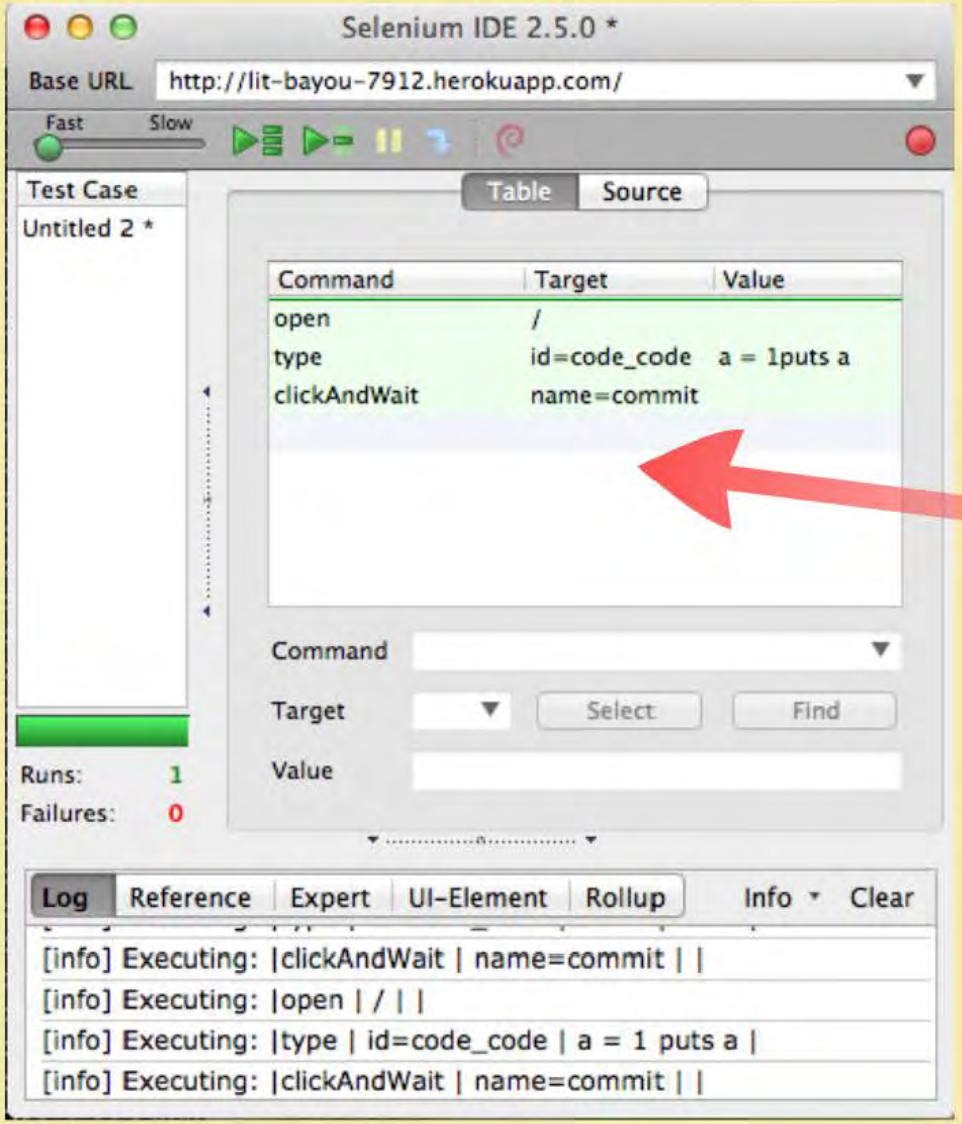

URL to start script at

Commands

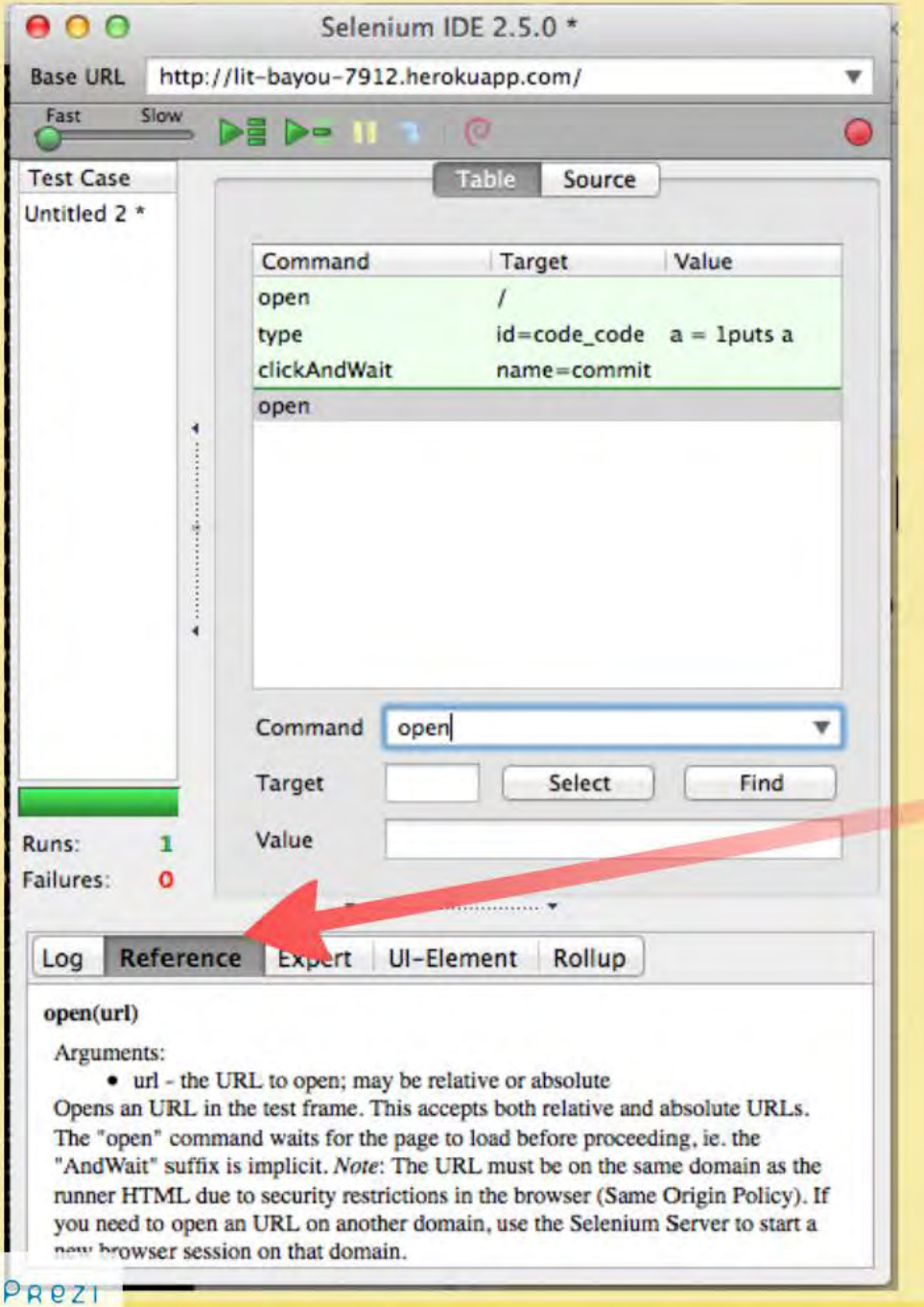
Logging

# Simple Scripting

1. File... New Test Case
2. Record an operation (red circle)
3. Stop recording
4. Run test suite (green triangle)




Our code

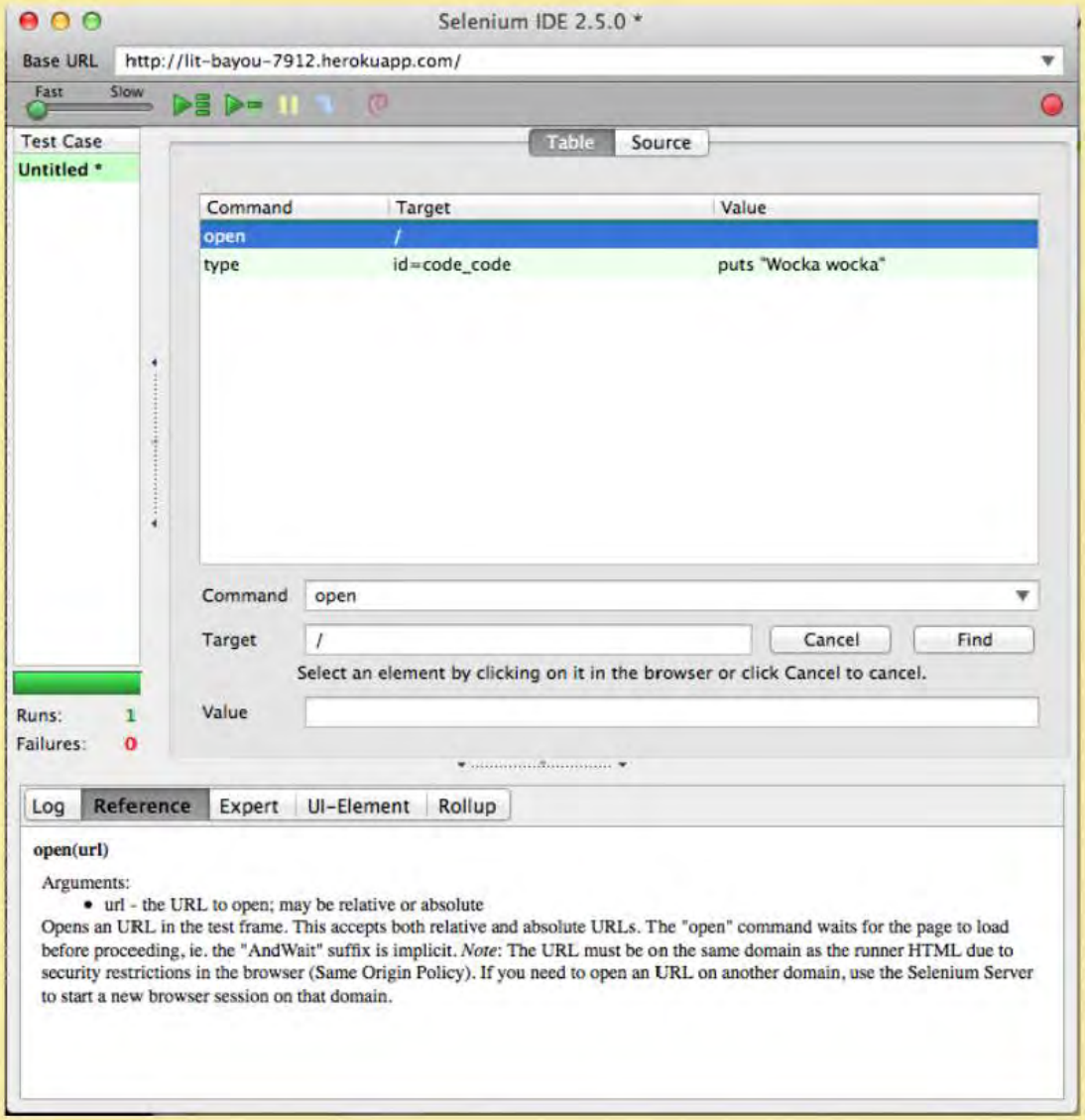



open <URL> - Opens either an absolute or relative URL

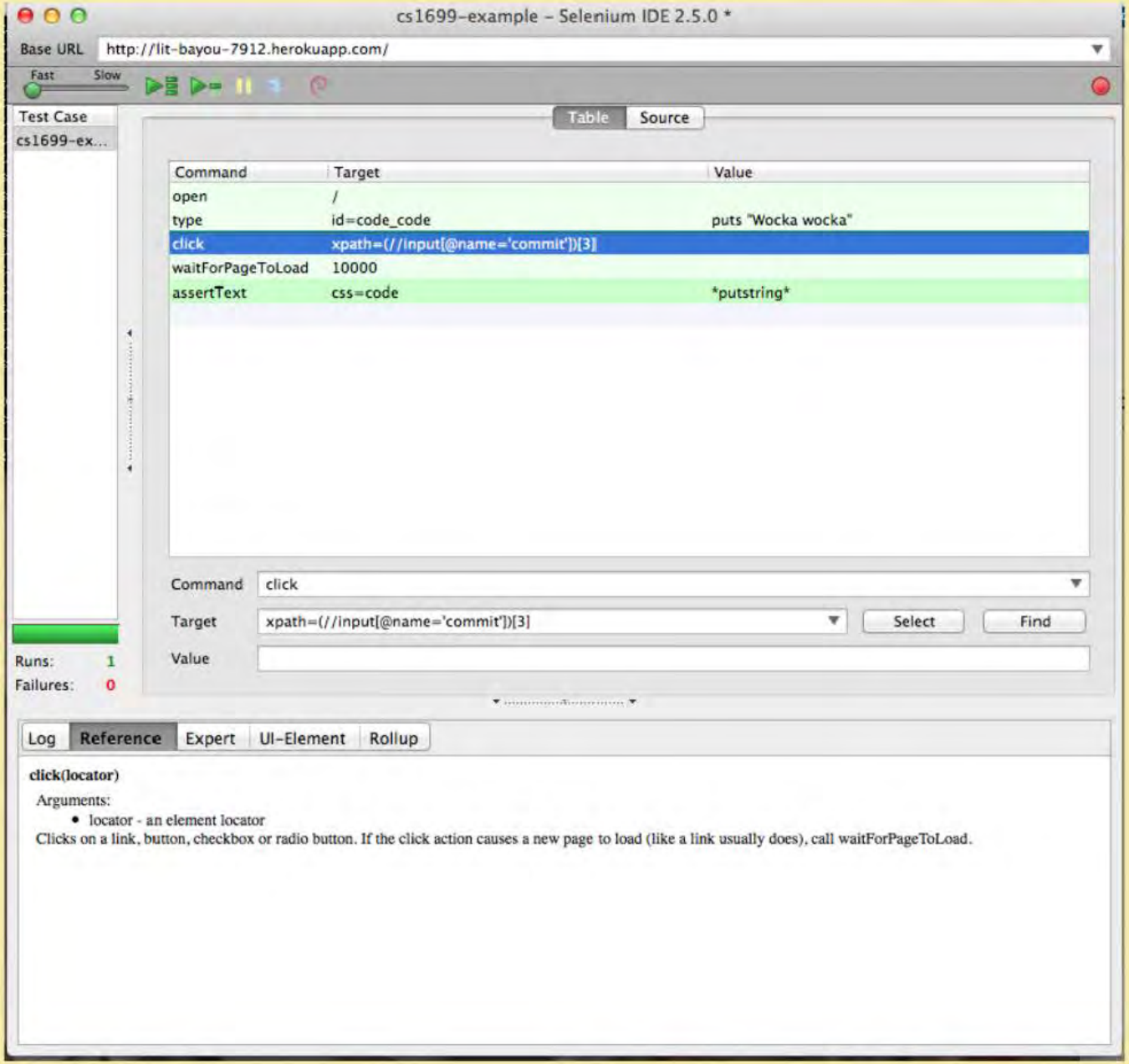
Note that you can check what an operation does by clicking on the "Reference" tab at the bottom

# Also note that you can move steps around by just clicking and dragging

... or add comments by selecting "Add New Comment" instead of "Add New Command"




type <target> <value> will type a string in the appropriate target (textbox or similar)




click
<element>
- clicks on
an element

waitForPageToLoad - Waits for another page to load, with an optional timeout.

If you don't wait, Selenium goes as fast as it can, and will do a check before the page is ready.  This means that your assertions will fail!

clickAndWait -> a combination of "click" and "waitForPageToLoad"

There are other "...andWait" variants since waiting for a page to load is very common.

# Hello, assertions, my old friend...

cs1699-example - Selenium IDE 2.5.0 *

Base URL http://lit-bayou-7912.herokuapp.com/

Fast Slow

Test Case

cs1699-ex...

Table Source

| Command | Target | Value |
|---|---|---|
| open | / | |
| type | id=code_code | puts "Wocka wocka" |
| click | xpath=(//input[@name='commit'])[3] | |
| waitForPageToLoad | 10000 | |
| assertText | css=code | *putstring* |

Command click

Target xpath=(//input[@name='commit'])[3] Select Find

Value

Runs: 1

Failures: 0

Log Reference Expert UI-Element Rollup

**click(locator)**

Arguments:

- locator - an element locator

Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.

Assert that something is true, just like a JUnit assertion

Note that you need to give a target to the assertions, e.g., that it should be on a particular element.

This is simple with select.

## Lots of fun assertions...

**assertText / assertTextPresent** - Assert that text exists (on an element (former) or entire page (latter)). Note that this is a regex!
**assertCookie** - Assert that a cookie exists.
**assertElementPresent** - Assert that an element exists somewhere on the page.
**assertAlert** - Assert that an alert took place.
**assertEditable** - Assert that an element is editable.
**assertEval** - Evaluate some JavaScript and assert the result.

All of the asserts also have a not-variant, e.g., assertNotEditable, which checks for the opposite of what the original assert does.

# You Try It!

Let's check that a search on Google for "Bill Laboon metaprogramming" brings up my talk on the subject.

# Advanced Selenium

Next Class!

# Next Class!

# CS1699 - Lecture 15 - An Introduction to Testing Web Applications with Selenium


Image by Tom Mocking