OVERVIEW

Sunday, June 16, 2013 7:44 AM

This Overview will explain how everything gets connected together. Feel free to update this or any other section.

First let me say at a high level that there are three main ideas here.
Cross Browser testing
Headless Browser testing
Testing though the workstation

This is a list of what we'll need below and the what applications (list in the other Tabs) that need to be connected together to perform the below. What we want to do is list those apps that are needed for each item in this list.

1. Drive standalone HTML/HTML5 web pages Chrome
2. Drive standalone HTML/HTML5 web pages using cross browsers
3. Driver standalone HTML/HTML5 web pages using chromium headless browser
4. Drive HTML/HTML5 testing through Chromium CEF3
5. Drive HTML/HTML5 testing through Chromium CEF3 through the workstation.
6. Drive Thiefjs component testing (menus, button, combinations etc.)
7. Drive compares between HTML/HTML5 docs.
8. Tellus drives the VM configuration and Test Cases
9. TestCases
    a. Test Cases will reside in P4.
    b. Test Cases should be written in a public scripting language(currently written in Java)
10. Bindings
    a. We want to be consistent on binding usage
    b. At this point JavaScript is a commercial package(check this out please)

Ruby 1.9.3 installation

Tuesday, July 16, 2013 10:47 AM

Ruby Installation

Download: http://dl.bintray.com/oneclick/rubyinstaller/rubyinstaller-1.9.3-p448.exe?direct

> **Note:** Please use the prebuilt binary installers at RubyInstaller for Windows site. The ruby-lang.org does not have Windows installer.

Please use Ruby 1.9.3 32-bit version; poltergeist only works with 1.9. Please don't use Ruby 2.0 right now.

Ruby 1.9 comes with minitest unit testing framework (4.7.5, 2.5.1).

Ruby 1.9.3 DevKit installation

Monday, July 15, 2013 2:49 PM

Download: http://rubyinstaller.org/downloads

- Ruby 1.8.6 to 1.9.3: **tdm-32-4.5.2** ***<== please use this one***
- Ruby 2.0.0: **mingw64-32-4.7.2**
- Ruby 2.0.0 x64 (64bits): **mingw64-64-4.7.2**

That is:

- DevKit-tdm-32-4.5.2-20111229-1559-sfx.exe

Note: It is a self-extracting zip archive.

Instructions: http://github.com/oneclick/rubyinstaller/wiki/Development-Kit

1. It will auto extract
2. Enter C:\DevKit when prompted for directory to which to extract
   Note: Do not extract to path with spaces.
3. `cd C:\DevKit`

   Output:
   [INFO] found RubyInstaller v1.9.3 at C:/Ruby

   Initialization complete! Please review and modify the auto-generated
   'config.yml' file to ensure it contains the root directories to all
   of the installed Rubies you want enhanced by the DevKit.

4. **`ruby dk.rb init`**

- edit the generated `config.yml` file to include installed Rubies not automatically discovered or remove Rubies you do not want to use the DevKit with.
- ***[optional]*** `ruby dk.rb review` to review the list of Rubies to be enhanced to use the DevKit and verify the changes you made to it are correct.

Pasted from <https://github.com/oneclick/rubyinstaller/wiki/Development-Kit>

5. **`ruby dk.rb install`**

Test Installation

- Confirm your Ruby environment is correctly using the DevKit by running,

  `gem install json --platform=ruby`

- JSON should install correctly and you should see `with native extensions` in the screen messages. Next run,

  `ruby -rubygems -e "require 'json'; puts JSON.load('[42]').inspect"`

  to confirm that the json gem is working. It should show:

  `[42]`

Pasted from <https://github.com/oneclick/rubyinstaller/wiki/Development-Kit>

With Eclipse

Monday, June 17, 2013 9:41 AM

I have found a link for using Ruby in Eclipse shown below:
http://www.easyeclipse.org/site/distributions/ruby-rails.html

I have gone through the Aptana (Eclipse-based) but finding it difficult to understand. I have read the doc to install and use the plugin with Eclipse but the instructions given differs from the actual. I have referred this link to learn. I have found another IDE named RDE for working with Ruby (still have to investigate what it can do?).

Geany editor also supports Ruby syntax. (It is similar to Notepad++, but the closing braces positioning is a little nicer.)
Free IDEs:

- Aptana (Eclipse-based)
- IntelliJ (Community Edition)

They are popular ones, but take a little getting used to.

Ruby with EasyEclipse: Getting Started

Tuesday, July 16, 2013 9:16 AM

Please see Ruby with EasyEclipse: Getting Started

Ruby Quick Reference

Thursday, June 27, 2013 10:36 AM

Some notes from Ken on Ruby and running Selenium

****
High Level for the QAAutomation Team for testing web browsers on a local machine.
Doesn't require a selenium2 server

Ruby is a scripting language like python and perl

It need Rails as a framework.

Install Ruby

Install Rails in Ruby if you didn't install a complete package
    Gem install rails

Install SeleniumWebDriver
    gem install selenium-webdriver

****
High Level for the QAAutomation Team for testing web browsers on a remote machine.
That's right you guessed it.
Does require a selenium2 server and Remote web-driver.

https://code.google.com/p/selenium/wiki/RemoteWebDriver
https://code.google.com/p/selenium/wiki/RemoteWe

Java -jar selenium-server-standalone.jar
Then connect to it from Ruby

```
driver = Selenium::WebDriver.for(:remote)
```

It need Rails as a framework.

Install Ruby

Install Rails in Ruby if you didn't install a complete package
    Gem install rails

Install SeleniumWebDriver
    Gem "selenium-webdriver","~>2.33.0"

Java -jar selenium-server-standalone.jar
Then connect to it from Ruby

```
driver = Selenium::WebDriver.for(:remote)
```

https://code.google.com/p/selenium/wiki/RubyBindings

http://www.tutorialspoint.com/ruby/ruby_quick_guide.htm

Rails is a framework for Ruby

Gem is a package installer for Ruby

To install Rails if you don't already have a complete package installer
Gem install rails

Selenium-webdriver -ruby

WebDriver is a tool for writing automated tests of websites. It aims to mimic the behaviour of a real user, and as such interacts with the HTML of the application.

```
gem "selenium-webdriver", "~> 2.33.0"
```

This gem provides Ruby bindings for WebDriver and has been tested to work on MRI (1.8.7 through 1.9), JRuby and Rubinius.

RubyBindings

Thursday, June 27, 2013 10:44 AM

https://code.google.com/p/selenium/wiki/RubyBindings

Selenium-webdriver

WebDriver is a tool for writing automated tests of websites. It aims to mimic the behaviour of a real user, and as such interacts with the HTML of the application.

```
gem "selenium-webdriver", "~> 2.33.0"
```

This gem provides Ruby bindings for WebDriver and has been tested to work on MRI (1.8.7 through 1.9), JRuby and Rubinius.

Introduction

The Ruby bindings for Selenium/WebDriver are available as the selenium-webdriver gem. The web page explains how to install the selenium-webdriver gem. On Mac OSX and linux you may need to prefix the rest of the command with the sudo command if the installation fails because of security restrictions on your computer.
There are many other Selenium gems out there, but this is the only official, maintained gem. If you're looking for a slightly higher level API built on the same technology, you may want to check out watir-webdriver or capybara.
The bindings support Ruby 1.9.2 through 2.0.0, JRuby and Rubinius.

- API docs
- Changelog

The gem also includes the older selenium-client gem for use with Selenium RC. When reading the docs, keep in mind that these two namespaces refer to different APIs:

- Selenium::WebDriver - the WebDriver API
- Selenium::Client - Selenium RC API (previously released as the selenium-client gem)

The WebDriver API is the successor to the Selenium RC API. For people who don't have a significant investment in the legacy API, we recommend starting directly with Selenium::WebDriver, and focusing on the two main classes, Selenium::WebDriver::Driver andSelenium::WebDriver::Element. This is the entry point to the whole WebDriver API.
For people who already have tests written against the Selenium RC API, it's possible to use WebDriver-backed Selenium to ease the migration. The rest of this document deals with Selenium::WebDriver exclusively.

API Example

The bindings provide a slightly rubified version of the WebDriver API:

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :firefox
driver.navigate.to "http://google.com"

element = driver.find_element(:name, 'q')
element.send_keys "Hello WebDriver!"
element.submit

puts driver.title

driver.quit
```

Driver examples:

```
# execute arbitrary javascript
puts driver.execute_script("return window.location.pathname")
```

```ruby
# pass elements between Ruby and JavaScript
element = driver.execute_script("return document.body")
driver.execute_script("return arguments[0].tagName", element) #=> "BODY"

# wait for a specific element to show up
wait = Selenium::WebDriver::Wait.new(:timeout => 10) # seconds
wait.until { driver.find_element(:id => "foo") }

# switch to a frame
driver.switch_to.frame "some-frame" # name or id
driver.switch_to.frame driver.find_element(:id, 'some-frame') # frame element

# switch back to the main document
driver.switch_to.default_content

# repositionning and resizing browser window:
driver.manage.window.move_to(300, 400)
driver.manage.window.resize_to(500, 800)
driver.manage.window.maximize
```

Element examples:

```ruby
# get an attribute
class_name = element.attribute("class")

# is the element visible on the page?
element.displayed?

# click the element
element.click

# get the element location
element.location

# scroll the element into view, then return its location
element.location_once_scrolled_into_view

# get the width and height of an element
element.size

# press space on an element - see Selenium::WebDriver::Keys for possible values
element.send_keys :space

# get the text of an element
element.text
```

Advanced user interactions (see ActionBuilder):

```ruby
driver.action.key_down(:shift).
        click(element).
        double_click(second_element).
        key_up(:shift).
        drag_and_drop(element, third_element).
        perform
```

IE

Make sure that *Internet Options → Security* has the same *Protected Mode* setting (on or off, it doesn't matter as long as it is the same value) for all zones.

Chrome

Command line switches

For a list of switches, see this list.

```ruby
driver = Selenium::WebDriver.for :chrome, :switches => %w[--ignore-certificate-errors --disable-popup-blocking --disable-translate]
```

Tweaking profile preferences

For a list of prefs, see pref_names.cc.

```ruby
profile = Selenium::WebDriver::Chrome::Profile.new
```

```
profile['download.prompt_for_download'] = false
profile['download.default_directory'] = "/path/to/dir"

driver = Selenium::WebDriver.for :chrome, :profile => profile
```
See also ChromeDriver.

Remote

The RemoteWebDriver makes it easy to control a browser running on another machine. Download the jar (from Downloads) and launch the server:

```
java -jar selenium-server-standalone.jar
```
Then connect to it from Ruby
```
driver = Selenium::WebDriver.for(:remote)
```
By default, this connects to the server running on localhost:4444 and opens Firefox. To connect to another machine, use the :url option:
```
driver = Selenium::WebDriver.for(:remote, :url => "http://myserver:4444/wd/hub")
```
To launch another browser, use the :desired_capabilities option:
```
driver = Selenium::WebDriver.for(:remote, :desired_capabilities => :chrome)
```
You can also pass an instance of Selenium::WebDriver::Remote::Capabilities, e.g.:
```
caps = Selenium::WebDriver::Remote::Capabilities.htmlunit(:javascript_enabled => true)
driver = Selenium::WebDriver.for(:remote, :desired_capabilities => caps)
```
You can change arbitrary capabilities:
```
caps = Selenium::WebDriver::Remote::Capabilities.internet_explorer
caps['enablePersistentHover'] = false

driver = Selenium::WebDriver.for(:remote, :desired_capabilities => caps)
```
You may want to set the proxy settings of the remote browser (this currently only works for Firefox):
```
caps = Selenium::WebDriver::Remote::Capabilities.firefox(:proxy => Selenium::WebDriver::Proxy.new(:http =>
"myproxyaddress:8080"))
driver = Selenium::WebDriver.for(:remote, :desired_capabilities => caps)
```
Or if you have a proxy in front of the remote server:
```
client = Selenium::WebDriver::Remote::Http::Default.new
client.proxy = Selenium::Proxy.new(:http => "proxy.org:8080")

driver = Selenium::WebDriver.for(:remote, :http_client => client)
```
See `Selenium::WebDriver::Proxy` for more options.
For the remote Firefox driver you can configure the profile, see the section Tweaking Firefox preferences.

Firefox

The FirefoxDriver lets you configure the profile used.

Adding an extension

It's often useful to have Firebug available in the Firefox instance launched by WebDriver:
```
profile = Selenium::WebDriver::Firefox::Profile.new
profile.add_extension("/path/to/firebug.xpi")

driver = Selenium::WebDriver.for :firefox, :profile => profile
```

Using an existing profile

You can use an existing profile as a template for the WebDriver profile by passing the profile name (see firefox -ProfileManager to set up custom profiles.)
```
driver = Selenium::WebDriver.for(:firefox, :profile => "my-existing-profile")
```
If you want to use your default profile, pass :profile => "default"
You can also get a Profile instance for an existing profile and tweak its preferences. This does not modify the existing profile, only the one used by WebDriver.
```
default_profile = Selenium::WebDriver::Firefox::Profile.from_name "default"
default_profile.native_events = true

driver = Selenium::WebDriver.for(:firefox, :profile => default_profile)
```

Tweaking Firefox preferences

Use a proxy:
```
profile = Selenium::WebDriver::Firefox::Profile.new
proxy = Selenium::WebDriver::Proxy.new(:http => "proxy.org:8080")
profile.proxy = proxy

driver = Selenium::WebDriver.for :firefox, :profile => profile
```
Automatically download files to a given folder:
```
profile = Selenium::WebDriver::Firefox::Profile.new
```

```
profile['browser.download.dir'] = "/tmp/webdriver-downloads"
profile['browser.download.folderList'] = 2
profile['browser.helperApps.neverAsk.saveToDisk'] = "application/pdf"

driver = Selenium::WebDriver.for :firefox, :profile => profile
```

If you are using the remote driver you can still configure the Firefox profile:

```
profile = Selenium::WebDriver::Firefox::Profile.new
profile['foo.bar'] = true

capabilities = Selenium::WebDriver::Remote::Capabilities.firefox(:firefox_profile => profile)
driver = Selenium::WebDriver.for :remote, :desired_capabilities => capabilities
```

For a list of possible preferences, see this page.

Custom Firefox path

If your Firefox executable is in a non-standard location:

```
Selenium::WebDriver::Firefox.path = "/path/to/firefox"
driver = Selenium::WebDriver.for :firefox
```

SSL Certificates

The Firefox driver ignores invalid SSL certificates by default. If this is not the behaviour you want, you can do:

```
profile = Selenium::WebDriver::Firefox::Profile.new
profile.secure_ssl = true

driver = Selenium::WebDriver.for :firefox, :profile => profile
```

There is an edge case where the default SSL certificate check will not work correctly. WebDriver assumes that the certificate is untrusted whenever there's a problem, which means a certificate from a trusted issuer but with a hostname mismatch (e.g. a production certificate in a test environment) will not be correctly ovverriden. See UntrustedSSLCertificates for more on why this is. To work around it, tell the Firefox driver to not assume the issuer is untrusted:

```
profile = Selenium::WebDriver::Firefox::Profile.new
profile.assume_untrusted_certificate_issuer = false
driver = Selenium::WebDriver.for :firefox, :profile => profile
```

Not that Profile#secure_ssl remains set to the default value of true in the above example.

Native events

Native events are enabled by default on Windows. To turn them off:

```
profile = Selenium::WebDriver::Firefox::Profile.new
profile.native_events = false

driver = Selenium::WebDriver.for(:firefox, :profile => profile)
```

Experimental support for native events is available on Linux. Set profile.native_events = true to turn this on.

Opera

The OperaDriver is always run as a RemoteWebDriver server which the Ruby bindings connect to.
To get started, first download the *selenium-server-standalone* jar and set the SELENIUM_SERVER_JAR environmental variable to point to its location:

```
export SELENIUM_SERVER_JAR=/path/to/server-standalone.jar
```

Then you can simply create a new instance of Selenium::WebDriver with the :opera option:

```
driver = Selenium::WebDriver.for :opera
driver.navigate.to 'http://opera.com/'
```

Safari

A Safari driver is available as of v2.21. See SafariDriver for details on usage. From Ruby:

```
driver = Selenium::WebDriver.for :safari
driver.navigate.to "http://apple.com"
```

Timeouts

Implicit waits

WebDriver lets you configure implicit waits, so that a call to #find_element will wait for a specified amount of time before raising aNoSuchElementError:

```
driver = Selenium::WebDriver.for :firefox
driver.manage.timeouts.implicit_wait = 3 # seconds
```

Explicit waits

Use the Wait class to explicitly wait for some condition:

```
wait = Selenium::WebDriver::Wait.new(:timeout => 3)
wait.until { driver.find_element(:id => "cheese").displayed? }
```

Internal timeouts

Internally, WebDriver uses HTTP to communicate with a lot of the drivers (the JsonWireProtocol). By

default, Net::HTTP from Ruby's standard library is used, which has a default timeout of 60 seconds. If you call Driver#get on a page that takes more than 60 seconds to load, you'll see a TimeoutError raised from Net::HTTP. You can configure this timeout (before launching a browser) by doing:

```
client = Selenium::WebDriver::Remote::Http::Default.new
client.timeout = 120 # seconds
driver = Selenium::WebDriver.for(:remote, :http_client => client)
```

JavaScript dialogs

You can use webdriver to handle Javascript alert(), prompt() and confirm() dialogs. The API for all three is the same.

Note: At this time alert handling is only available in Firefox and IE (or in those browsers through the remote server), and only alerts that are generated post onload can be captured.

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :firefox
driver.navigate.to "http://mysite.com/page_with_alert.html"

driver.find_element(:name, 'element_with_alert_javascript').click
a = driver.switch_to.alert
if a.text == 'A value you are looking for'
 a.dismiss
else
 a.accept
end
```

Using Curb or your own HTTP client

For internal HTTP communication, Net::HTTP is used by default. If you e.g. have the Curb gem installed, you can switch to it by doing:

```
require 'selenium/webdriver/remote/http/curb'

client = Selenium::WebDriver::Remote::Http::Curb.new
driver = Selenium::WebDriver.for(:firefox, :http_client => client)
```

If you have the net-http-persistent gem installed, you can (as of 0.1.3) similarly use "selenium/webdriver/remote/http/persistent" to get keep-alive connections. This will significantly reduce the ephemeral ports usage of WebDriver, which is useful in some contexts. Note that this currently only works with the remote Java server (the other servers doesn't yet support keep-alive).

WebDriver-backed Selenium

If you already have tests written against the RC API, the Selenium server (since version 2.19) gives you ability to run your existing code backed by a WebDriver instance. This can help the migration to WebDriver since it allows you to mix and match the two APIs in the same test. Here's an example:

```
require 'selenium/webdriver'
require 'selenium/client'
require 'selenium/server'

server = Selenium::Server.new("selenium-server-standalone-2.19.0.jar", :background => true)
server.start

begin
 selenium = Selenium::Client::Driver.new :host    => "localhost",
                                         :port    => 4444,
                                         :url     => "http://google.com",
                                         :browser => "*webdriver"
 driver = Selenium::WebDriver.for :remote, :url => "http://localhost:4444/wd/hub/"

 selenium.start :driver => driver
 selenium.open "/"
 selenium.type 'q', 'webdriver-backed selenium'

 p driver.title == selenium.title

 selenium.stop
ensure
 server.stop
end
```

The opposite of this, a Selenium-backed WebDriver, is not available from Ruby at the moment.

Pasted from <https://code.google.com/p/selenium/wiki/RubyBindings>

Ruby zip file unzipping utility example

Tuesday, July 16, 2013 7:46 AM

(Optional)

```
# Example Ruby script to unzip zip file
# Needs `rubyzip' gem:
# gem install rubyzip
require 'zip/zipfilesystem'

if ARGV.length <= 0
    puts "Usage: ruby unzip.rb ZIPFILE"
    exit
end
target = "."
filename = ARGV[0]
Zip::ZipFile.open(filename) do |zipfile|
    zipfile.each do |file|
        puts "writing #{target}/#{file}"
        file.extract("#{target}/#{file}") {
            |entry,path| true
        }
    end
end
```

Ruby 2.0 installation

Thursday, August 01, 2013 8:27 AM

Download (Ruby 2.0 32-bit):
http://dl.bintray.com/oneclick/rubyinstaller/rubyinstaller-2.0.0-p247.exe?direct

1. Run rubyinstaller-2.0.0-p247.exe
2. Enter directory: **C:\Ruby**
   Note: So that next upgrade does not change test scripts #! line.
3. Check **Add Ruby executables to your PATH**
4. Check **Associate .rb and .rbw files with this Ruby installation**
5. Click **Install**

Ruby 2.0 DevKit installation

Thursday, August 01, 2013 8:31 AM

Download:

1. Go to http://rubyinstaller.org/downloads/
   and look for "For use with Ruby 2.0 and 2.1 (32bits version only)"
2. Remove any existing C:\DevKit directory (back it up first, if necessary)
3. Run the installer .exe
4. Enter directory: C:\DevKit
5. Click Extract
6. On command prompt:

   ```
   cd C:\DevKit
   ruby dk.rb init
   ruby dk.rb install
   ```

7. **Test Installation:** Confirm your Ruby environment is correctly using the DevKit by running:

   ```
   gem install json --platform=ruby
   ```

   JSON should install correctly and you should see with native extensions in the screen messages. Next run

   ```
   ruby -rubygems -e "require 'json'; puts JSON.load('[42]').inspect"
   ```

   to confirm that the json gem is working. If you see "[42]" in the output, then it is successful.

Reference

http://github.com/oneclick/rubyinstaller/wiki/Development-Kit

Ruby minitest 5.0.6 installation

Thursday, August 01, 2013 8:46 AM

gem install minitest-5.0.6.gem

Ruby selenium-webdriver installation

Monday, July 15, 2013 2:44 PM

Usually, installation is like this:

```
gem install selenium-webdriver
```

But it has documentation installation error:

Parsing documentation for nokogiri-1.6.0-x86-mingw32
unable to convert "\x90" from ASCII-8BIT to UTF-8 for lib/nokogiri/1.9/nokogiri.so, skipping
unable to convert "\x90" from ASCII-8BIT to UTF-8 for lib/nokogiri/2.0/nokogiri.so, skipping

Can avoid those error by not installing documentation:

```
gem install selenium-webdriver --no-rdoc --no-ri
```

To download selenium-webdriver gem file:

http://rubygems.org/downloads/selenium-webdriver-2.40.0.gem

Dependencies

multi_js
- http://rubygems.org/downloads/multi_json-1.8.4.gem

rubyzip
- https://rubygems.org/downloads/rubyzip-1.1.0.gem

childprocess
- https://rubygems.org/downloads/childprocess-0.5.1.gem

  ffi
  - http://rubygems.org/downloads/ffi-1.9.3.gem

    DevKit (for Ruby)
    - http://rubyinstaller.org/downloads
      Instructions: https://github.com/oneclick/rubyinstaller/wiki/Development-Kit

websocket 1.0.4
- https://rubygems.org/downloads/websocket-1.0.4.gem

Selenium Ruby Binding documentation

Wednesday, July 17, 2013 3:18 PM

Ruby selenium-webdriver

https://code.google.com/p/selenium/wiki/RubyBindings

Selenium Ruby remote Chrome example

Monday, July 08, 2013 1:07 PM

Ruby selenium-webdriver with Remote Chrome Browser

Prerequisite:

1. ChromeDriver.exe must be in PATH on remote machine
2. Remote machine must be running selenium-server-standalone.jar -port 6770

Example

```
require "selenium-webdriver"

caps = Selenium::WebDriver::Remote::Capabilities.chrome
driver = Selenium::WebDriver.for :remote, :url =>
"http://172.17.245.70:4444/wd/hub", :desired_capabilities => caps

driver.navigate.to "http://google.com"

element = driver.find_element(:name, 'q')
element.send_keys "Hello WebDriver!"
element.submit

puts driver.title

driver.quit
```

Selenium Ruby default Chrome search paths

Thursday, July 11, 2013 3:50 PM

Ruby selenium-webdriver

By default, selenium-webdriver searches for chrome.exe in the following paths:

```
def self.possible_paths
  [
    registry_path,
    "#{ENV['USERPROFILE']}\\Local Settings\\Application Data\\Google\\Chrome\\Application\\chrome.exe",
    "#{ENV['USERPROFILE']}\\AppData\\Local\\Google\\Chrome\\Application\\chrome.exe",
    "#{Platform.home}\\Local Settings\\Application Data\\Google\\Chrome\\Application\\chrome.exe",
    "#{Platform.home}\\AppData\\Local\\Google\\Chrome\\Application\\chrome.exe",
  ].compact
end
```

Pasted from <http://rubydoc.info/gems/selenium-webdriver/0.0.28/Selenium/WebDriver/Chrome/Launcher/WindowsLauncher>

Selenium Ruby set binary path

Monday, July 15, 2013 1:09 PM

Firefox:

```
Selenium::WebDriver::Firefox.path = "/some/path"
```

Chrome:

```
Selenium::WebDriver::Chrome.path = "/some/path"
```

Pasted from <https://code.google.com/p/selenium/issues/detail?id=769>

Ruby Selenium::WebDriver::Remote::Capabilities.chrome()

Wednesday, October 22, 2014 8:48 AM

chrome(opts = {})

In Ruby, Selenium::WebDriver::Remote::Capabilities.chrome is:
a Selenium::WebDriver::Remote::Capabilities object. It has private instance variable, @capabilities, of type Hash. Default capabilities:

```
{
 browser_name: "chrome",
 version: "",
 platform: :any,
 javascript_enabled: true,
 css_selectors_enabled: true,
 takes_screenshot: false,
 native_events: false,
 rotatable: false,
 firefox_profile: nil,
 proxy: nil,
}
```

The opts hash will merge (individual keys added) into the private @capabilities hash:

```
capabilities = {
  'platformName' => 'iOS',
  'platformVersion' => '7.1',
  'deviceName' => 'iPhone 5s',
  'app' => '/Users/username/Library/Developer/Xcode/DerivedData/WebViewApp-
gpovzqchsajoybbnamepcbgjguhj/Build/Products/Debug-
iphonesimulator/WebViewApp.app'
}
caps = Selenium::WebDriver::Remote::Capabilities.iphone capabilities
```

Class: Selenium::WebDriver::Remote::Capabilities Private

Inherits:

Objectshow all

Defined in:

rb/lib/selenium/webdriver/remote/capabilities.rb

Overview

This class is part of a private API. You should avoid using this class if possible, as it may be removed or be changed in the future.

Specification of the desired and/or actual capabilities of the browser that the server is being asked to create.

Constant Summary

DEFAULTS =

This constant is part of a private API. You should avoid using this constant if possible, as it may be removed or be changed in the future.

```
{
  :browser_name        => "",
  :version             => "",
  :platform            => :any,
  :javascript_enabled  => false,
  :css_selectors_enabled => false,
  :takes_screenshot    => false,
  :native_events       => false,
  :rotatable           => false,
  :firefox_profile     => nil,
  :proxy               => nil
}
```

Class Method Summary(collapse)

- + (Object) **android**(opts = {}) private
- + (Object) **chrome**(opts = {}) private
- + (Object) **firefox**(opts = {}) private
- + (Object) **htmlunit**(opts = {}) private
- + (Object) **internet_explorer**(opts = {}) (also: ie) private
- + (Object) **ipad**(opts = {}) private
- + (Object) **iphone**(opts = {}) private
- + (Object) **json_create**(data) private
- + (Object) **opera**(opts = {}) private
- + (Object) **safari**(opts = {}) private

Instance Method Summary(collapse)

- - (Object) **==**(other) (also: #eql?) private
- - (Object) **[]**(key) private
- - (Object) **[]=**(key, value) private
  Allows setting arbitrary capabilities.
- - (Object) **as_json**(opts = nil) private
- - (Capabilities) **initialize**(opts = {}) constructor
  Firefox-specific options:.
- - (Object) **merge!**(other) private

- - (Object) **proxy=**(proxy) private
- - (Object) **to_json**(*args) private

Constructor Details

- (Capabilities) **initialize**(opts = {})
Firefox-specific options:
Parameters:
- **:browser_name** (Hash) —
  a customizable set of options
- **:version** (Hash) —
  a customizable set of options
- **:platform** (Hash) —
  a customizable set of options
- **:javascript_enabled** (Hash) —
  a customizable set of options
- **:css_selectors_enabled** (Hash) —
  a customizable set of options
- **:takes_screenshot** (Hash) —
  a customizable set of options
- **:native_events** (Hash) —
  a customizable set of options
- **:proxy** (Hash) —
  a customizable set of options
- **:firefox_profile** (Hash) —
  a customizable set of options

[View source]

Capybara

Sunday, June 16, 2013 7:46 AM

Capybara is a library written in the Ruby programming language which makes it easy to simulate how a user interacts with your application.

It can work with the following drivers through the same API (if the driver supports those features):

- selenium-webdriver (for UI browsers, such as Chrome)
- poltergeist (for headless browser, PhantomJS)

https://github.com/thoughtbot/capybara-webkit

1. I'm using Ruby 1.9.3.
2. Put phantomjs.exe directory in the PATH
3. Installed DevKit
4. >gem install capybara
5. >gem install poltergeist

Test script:

```
require 'capybara'
require 'capybara/poltergeist'
Capybara.javascript_driver = :poltergeist

@session = Capybara::Session.new(:poltergeist)
@session.visit 'http://testerstories.com/test_app'
puts "The current URL is #{@session.current_url}"
```

Reference

Capybara API documentation

Capybara installation

Monday, July 15, 2013 2:37 PM

Usually, installation is like this:

```
gem install capybara
```

But it has documentation installation error:

Parsing documentation for nokogiri-1.6.0-x86-mingw32
unable to convert "\x90" from ASCII-8BIT to UTF-8 for lib/nokogiri/1.9/nokogiri.so, skipping
unable to convert "\x90" from ASCII-8BIT to UTF-8 for lib/nokogiri/2.0/nokogiri.so, skipping

So this avoids those error by not installing documentation:

```
gem install capybara --no-rdoc --no-ri
```

Capybara getting started

Monday, July 15, 2013 3:04 PM

Getting started with Capybara

1. Ruby Installation

Please see: Ruby installation

2. Ruby Gems Installations

a) Capybara installation
b) Ruby selenium-webdriver installation

3. PhantomJS installation

a) Please see: PhantomJS installation

4. For Ruby with PhatomJS

a) Need this first: Ruby DevKit installation
b) Please see: Ruby poltergeist installation

Note: Ruby poltergesit is not working for me right now. (though phantomjs by itself is working.)

Capybara API documentation

Wednesday, July 17, 2013 3:17 PM

Capybara Documentation: http://rdoc.info/github/jnicklas/capybara

Note: In the documentation, usually "page" and "session" is used interchangeably.

Capybara simple Firefox example

Tuesday, July 16, 2013 12:58 PM

```
# This example starts login on example page
# This is non-unit test script.
require 'capybara'

@session = Capybara::Session.new(:selenium)
@session.visit 'http://testerstories.com/test_app'

puts "The current URL is #{@session.current_url}"
puts "The current path is #{@session.current_path}"

response = @session.body
p(response.scan("<title>Test Site for Automation</title>"))


@session.click_on('Form Examples')

# -or-
#@session.click_link('Form Examples')

puts "Log In Button Found:
#{@session.has_css?('input[id="btnSubmit"]').to_s}"
puts "Customer Code Field Found:
#{@session.has_xpath?('//input[@id="customerCode"]').to_s}"

# textbox
@session.fill_in 'customerCode', :with => 'Test'
@session.fill_in 'loginName', :with => 'jnyman'
@session.fill_in 'password', :with => 'tester2'

# checkbox
@session.check 'shunt'
@session.check 'harpoon'

# radiobuttons
@session.choose 'mtc'

# listbox
@session.select 'Tachyonic Antitravel', :from => 'concepts'
```

Capybara and selenium-webdriver API difference

Wednesday, July 17, 2013 2:11 PM

For Ruby, Capybara API is higher level than selenium-webdriver API.

To use Capybara API:

```
require 'capybara'
```

To use Selenium Ruby Binding (selenium-webdriver):

```
require 'selenium-webdriver'
```

Capybara can use its `:selenium` driver, which internally use Selenium WebDriver. For example:

```
@session = Capybara::Session.new(:selenium)
```

This allows Capybara to drive headed browser through webdriver.

With Selenium API in Ruby, one would start a session like this:

```
@session = Selenium::WebDriver.for :firefox
```

For example, to set loginName field value:

In Capybara style:

```
@session.fill_in 'loginName', :with => 'jnyman'
```

In Selenium style:

```
element = @session.find_element(:name, 'loginName')
element.send_keys "jnyman"
element.submit
```

Capybara Examples:

Capybara

Selenium Ruby Binding Examples:

Selenium Ruby Binding

References

Capybara API documentation:

http://rdoc.info/github/jnicklas/capybara

Selenium Ruby Binding documentation:

https://code.google.com/p/selenium/wiki/RubyBindings

Ruby poltergeist installation

Sunday, June 16, 2013 7:48 AM

Need this first:

Ruby DevKit installation

```
gem install poltergeist
```

Ruby poltergeist compatibility

Tuesday, July 16, 2013 10:34 AM

Compatibility

Poltergeist runs on MRI 1.9, JRuby 1.9 and Rubinius 1.9. Poltergeist and PhantomJS are currently supported on Mac OS X, Linux, and Windows platforms.

Ruby 1.8 is no longer supported.

Reference

https://github.com/jonleighton/poltergeist

Ruby poltergeist issue: "wrong exec option symbol: pgroup"

Tuesday, July 23, 2013 8:18 AM

Capybara poltergeist driver for phantomjs

Issue Description

Issue: Using poltergeist gets error "wrong exec option symbol: pgroup"

This issue only happens on Windows, not Linux.

Solution

The fix has been merged into master (in June 2013).

Download of fix

Download a gem built from project git master: poltergeist-1.3.0.gem under,

H:\Department\QualityAssurance\QAAutomation\Selenium\Ruby\

Installation Instructions

Please uninstall any existing poltergeist, first:

```
Dir> gem uninstall poltergeist
```

To install it,

```
Dir> gem install poltergeist-1.3.0.gem
```

Issue Discussion

https://github.com/jonleighton/poltergeist/pull/329

Git Diff

https://github.com/jonleighton/poltergeist/pull/329/files

Ruby poltergeist save screenshot example

Wednesday, July 24, 2013 11:11 AM

To take screenshot (snapshot):

```
@session.save_screenshot('screenshot.png')
```

Example

```
require 'capybara'
require 'capybara/poltergeist'
Capybara.javascript_driver = :poltergeist
@session = Capybara::Session.new(:poltergeist)
@session.visit 'http://testerstories.com/test_app'

puts "The current URL is #{@session.current_url}"
puts "The current path is #{@session.current_path}"

response = @session.body
p(response.scan("<title>Test Site for Automation</title>"))


@session.click_on('Form Examples')

# -or-
#@session.click_link('Form Examples')

puts "Log In Button Found:
#{@session.has_css?('input[id="btnSubmit"]').to_s}"
puts "Customer Code Field Found:
#{@session.has_xpath?('//input[@id="customerCode"]').to_s}"

@session.fill_in 'customerCode', :with => 'Test'
@session.fill_in 'loginName', :with => 'jnyman'
@session.fill_in 'password', :with => 'tester2'

@session.check 'shunt' # checkbox
@session.check 'harpoon'

@session.choose 'mtc'  # dropdown list
@session.select 'Tachyonic Antitravel', :from => 'concepts'

@session.save_screenshot('screenshot.png')
```

Phantomjs

Sunday, June 16, 2013 7:46 AM

Link

How can I get PhantomJS to trigger the Ajax call?
I use PhantomJS 1.9.1, Capybara 2.1.0 and Poltergeist 1.3.0

Pasted from <http://stackoverflow.com/questions/17401678/capybara-with-phantomjs-poltergeist-selecting-value-from-drop-down-doesnt-tr>

PhantomJS 1.9.1 with

Poltergeist - A PhantomJS driver for Capybara

http://phantomjs.org/release-1.8.html

The obvious downside to Selenium is that it requires a full graphical desktop for any and all tests. However, Selenium can now control PhantomJS in the same way that it does any other browser.

Used for headless testing of web applications. It is also used in Icahbod by Bryant Rolfe etc.

It integrates Ghost Driver an implementation of WebDriver Wire Protocal (written in JSON).
What's JSON? An alternate to XML used mainly for serial messaging.

PhantomJS installation

Monday, July 15, 2013 2:54 PM

Download: http://phantomjs.org/download.html

Download phantomjs-1.9.1-windows.zip (6.8 MB) and extract (unzip) the content.

The executable phantomjs.exe is ready to use.

Reference:
http://phantomjs.org

Thiefjs

Sunday, June 16, 2013 8:00 AM

Link to Thief.js:

http://thief.factset.com/versions/2.11.0/docs/

WebDriver Overview

Sunday, June 16, 2013 7:49 AM

WebDriver

The app(Script) connects directly to the browser on the same machine.

WebDriverRemote

Sunday, July 07, 2013 9:48 AM

RemoteWebDriver ALSO SEE RUBY RubyBindings this OneNote,

Makes it easy to control a browser running on another machine.

Information about the RemoteWebDriver
WebDriver

Updated Jan 14, 2013 by luke.semerau

Remote WebDriver

This is information about using the client implementation of the RemoteWebDriver. This is the code that is used within your tests. For information on how to set up the server-side, please take a look at the RemoteWebDriverServer page.

Installing

Copy "webdriver-all.jar" and all the associated JARs to your CLASSAPATH. This will give you the remote webdriver client, which is generally what you need. Please consult the RemoteWebDriverServer for information on how to set up the server-side of the remote webdriver.

Pros

- Separates where the tests are running from where the browser is.
- Allows tests to be run with browsers not available on the current OS (because the browser can be elsewhere)

Cons

- Requires an external servlet container to be running
- You may find problems with line endings when getting text from the remote server
- Introduces extra latency to tests, particularly when exceptions are thrown.

Using

This is probably best demonstrated with some code:

```
// We could use any driver for our tests...
DesiredCapabilities capabilities = new DesiredCapabilities();

// ... but only if it supports javascript
capabilities.setJavascriptEnabled(true);

// Get a handle to the driver. This will throw an exception
// if a matching driver cannot be located
WebDriver driver = new RemoteWebDriver(capabilities);

// Query the driver to find out more information
Capabilities actualCapabilities = ((RemoteWebDriver) driver).getCapabilities();

// And now use it
driver.get("http://www.google.com");
```

One nice feature of the remote webdriver is that exceptions often have an attached screen shot, encoded as a Base64 PNG. In order to get this screenshot, you need to write code similar to:

```
public String extractScreenShot(WebDriverException e) {
 Throwable cause = e.getCause();
 if (cause instanceof ScreenshotException) {
  return ((ScreenshotException) cause).getBase64EncodedScreenshot();
 }
```

```
 return null;
}
```

Pasted from <https://code.google.com/p/selenium/wiki/RemoteWebDriver>

WebDriverRemote commonly referred to as RemoteWebDriver
There are two modes
Client mode - language bindings connect to the remote instance. FireFox, OperaDriver and the RemoteWebDriver client normally work

Server mode - language bindings are responsible for setting up the server, which the driver in the browser can connect to. ChromeDriver works this way.

Chromium CEF3 Phase 1 – Prework

Introduction:

The Chromium Embedded Framework (CEF) is an open source framework for embedding a web browser control based on Google Chrome; it is a convenient way to implement an HTML5 based GUI in a desktop application or to provide browser capabilities to an application. It comes with bindings for C, C++, Delphi, Java, .NET, and Python and runs on Linux, Mac OS X, and Windows.

There are three versions of CEF to date:

- CEF1 - Single process implementation using the Chromium WebKit API.
- CEF2 - Multi process implementation built on the Chromium browser (abandoned)
- CEF3 - Multi process implementation using the Chromium Content API.

What can CEF framework do?

It can do the same thing as Chrome browser and can be used in the same way as that browser:

- Embedding an HTML5-compliant Web browser control in an existing native application.
- Creating a light-weight native "shell" application that hosts a user interface developed primarily using Web technologies.
- Rendering Web content "off-screen" in applications that have their own custom drawing frameworks.
- Acting as a host for automated testing of existing Web properties and applications.
- but just needs to set a different browser executable binary file path (maybe in configuration)

Team mission:

Factset is working on integrating the multi-process version of CEF into the Factset Workstation. Our team needs to create support processes for the testing of Chromium CEF3 elements in our Tellus Framework, and automate the testing of the FactSet version of Chromium CEF3.

What do we need to do in Phase 1?

1. Getting familiar with Chromium CEF3

   Reading Chromium CEF3 documents and RPDs

   http://infonet.factset.com/view/Projects/Browser/CefMultiProcessPlan
   http://is.factset.com/RPD/summary.aspx?messageId=6254463
   http://is.factset.com/rpd/Summary.aspx?messageId=10862612
   http://is.factset.com/RPD/summary.aspx?messageId=9256929
   http://is.factset.com/RPD/summary.aspx?messageid=11252825
   http://magpcss.org/ceforum/viewtopic.php?f=10&t=645
   http://code.google.com/p/chromiumembedded/wiki/Architecture#CEF3

2. Practice with Chromium CEF3

   (1). Installed CEF3Client in your local machine.

You can get CEF3Client.zip from H:\Department\QualityAssurance\QAAutomation\CEF\ (file://pc.factset.com/dfs/Department/QualityAssurance/QAAutomation/CEF)

(2). Try to run following Demo code

```
#!C:|Ruby|bin|ruby.exe

require "selenium-webdriver"
Selenium::WebDriver::Chrome.path = "C:||CEFClient||cefclient.exe"
driver = Selenium::WebDriver.for :chrome
driver.navigate.to "http://google.com"

element = driver.find_element(:name, 'q')
element.send_keys "Hello WebDriver!"
element.submit

title = driver.title

puts title          # => "Google"

driver.quit
```

(Note: Assuming the CEF client is located at c:\CEFClient\cefclient.exe

3. Getting familiar with "Ruby selenium-webdriver" (onenote://H:/Department/QualityAssurance/QAAutomation/Selenium/SelForFactSet/SelForFactSet)

Chromium CEF3 testing automation boundaries:

Tellus UI:

- Adding browser type "CEF" and add a version "3" to TellusDev UI (distinguish from regular Chrome browser)

Framework:

- Using Selenium as platform
- Update framework files as necessary. e.g. Adding the cefclient to the framework, fixing the "cefclient crash after, etc.
- Run ruby testcases on Tellus DEV

Functional:

- Develop ruby testcases to support CEF3 test as necessary

Chromium CEF3 Phase 2

Team mission:

Factset is working on integrating the multi-process version of CEF into the Factset Workstation. Our team

needs to create support processes for the testing of Chromium CEF3 elements in our Tellus Framework, and automate the testing of the FactSet version of Chromium CEF3.

What do we need to do in Phase 2?

In phase 2, we need to define how to support the effort defined by Chromium CEF3 developer Malay/ Gonzo and company

1. Get Factset prototype sample code from developer
2. Configure cef3 test environment
3. Install prototype sample code to test environment
4. Define test approach for existing feature and new cef3 feature
5. Develop demo script, and run testing

What we have done (Status) in Phase 2 (worked with Joseph)?

Master RPD:11745810 - Automate the testing of Chromium CEF3 supported Factset Workstation

1. Get Factset prototype sample code from developer
   Status:
   Yes. We got CEF3 supported Factset prototype sample code (a preproduction build of workstation with cef3 support) from Gonzo Berman on Feb 10, 2014 (note from Gonzo: I have not hooked up the chrome remote-debugging switch for Selenium like we discussed earlier. Right now the remote-debugging port is hardcoded to 8088.).
   We always keep installing latest CEF3 enabled workstation build.

2. Configure cef3 test environment
   Stratus:
   We installed Java, selenium 2, Selenium-webdriver, Chromedrive, Watir driver and all required tools to our test environment.
   (1) Configured a static VM machine (testcompleteb01) for Cef3 testing
   (2) Configured FPE "QA single use" VM for Cef3 testing
   (3) Configured local machine (FYIN) using inhouse account) for Cef3 testing

3. Install prototype sample code
   Status:
   Installed first preproduction build of cef3 supported workstation on VM on Feb 10, 2014
   Keep touching with developer, once the new test build is ready, we download latest build and install it to our test environment.

4. Define test approach for existing feature new cef3 feature
   Status:
   (1) For existing feature on workstation:
   - We used TestComplete and existing testplan/testcase to test. We downloaded the latest CEF3 Factset workstation, installed it to VM, and ran testcases "TestSystemWithNewBuild.sj. We got correct results.
   - I think we may use TestComplete to test cef3 workstation's existing feature/functions, this will save a lot of jobs. However, further investigation is needed to confirm.

   (2) New cef3 feature:
   - Tested and confirmed that webpage and Webkit Inspector (built-in developer tool) can be opened from Factset workstation.
   - Tested and confirmed CEF3 testpage is displayed on localhost:8088 Remote Debugging port
   - Run ruby script test on FPE QA single VM, confirmed that we can successfully connect to remote-debugging-port, and can find webpage and return web ID

from CEF3 workstation.

(3) Approaches and issues:

- Tried node-module, Chrome-Remote-Interface and ran JS script test on VM. It works (switches window and loads Web page inside workstation), but it is low level approach and lacks documentation. It is not recommended.
- Tried Selenium/ChromeDriver (with port 4444) and run ruby script test on VM. It has "unable to connect to renderer" and "couldn't switch windows" issue (the low-level socket connection starts connection to the window's URL but does not complete connection.)
- Tried approach -- using direct Websocket connection in ruby, but it still couldn't switch windows within workstation. It was redirected to the system default browser instead. (Its purpose was a quick double-checking of the WebSocket issue that ChromeDriver has.) As this approach was for double-checking WebSocket, we will not devote more time on direct ruby WebSocket issue.

5. Develop demo script, and run testing

Status:

We have developed Demo scripts for each approach. We will continue working on solving "unable to connect to renderer" issue, and finding a better approach to test cef3 feature.

Chromium Data PassThrough

Thursday, May 22, 2014 2:26 PM

Links

http://serverfault.com/questions/19914/google-chrome-passthrough-windows-authentication

Single sign on - authentication passthrough

Chrome Version : 5.0.375.99 beta
URLs (if applicable) :
Other browsers tested:
Add OK or FAIL after other browsers where you have tested this issue:
Safari 4:
Firefox 3.x:
IE 7:
IE 8:
What steps will reproduce the problem?
1.Go to URL of a site in the Intranet sites list in Internet Settings (
Windows) and pass-through identification is activated. (No Login Prompt)
2. Remove URL from Intranet Sites, repeat with login prompt
3. Re-add URL, repeat 1 with no login prompt.
What is the expected result?
Pass-through identification not expected to work.
What happens instead?
No Login prompt, uses local Windows user credentials when visiting a site
requiring Windows authentication.
Please provide any additional information below. Attach a screenshot if
possible.
New feature was not listed in change log.

Pasted from <https://groups.google.com/a/chromium.org/forum/#!msg/chromium-bugs/FNjnQknxtfs/M1dw9wykKQsJ>

Reported by d...@google.com, Feb 1, 2012
Chrome Version : 18.0.1025.1
OS Version: Goobuntu
URLs (if applicable) : docs.google.com
Other browsers tested: n/a
What steps will reproduce the problem?
1. Open docs.google.com in a new tab.
2. Open the Chrome process explorer (shift-esc). Find the docs process in the list, observe its memory use, maybe about 90MB.
3. Repeatedly open documents from that doclist, allow them to load, then close them, leaving only doclist. Never close docli st.
What is the expected result?
Once the extra tabs have been closed, memory use should fall back to something like it was when the doclist was initially ope ned.
What happens instead?
The memory used grows each time a tab is opened, and doesn't fall back when it is closed. After 20 or so tabs had been opene d and closed, memory usage was approaching 1GB. It seems like the memory used by these tabs is effectively never released when they are closed, and will only be released when the last tab on that render pro cess closes.
In various scenarios, docs can use both a shared worker and a hosted chrome app. This behavior has been observed even when b oth those factors are removed. The behavior reproduces quite reliably. This may be related to issue 112184.
UserAgentString: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.1 Safari/535.19

Pasted from <https://code.google.com/p/chromium/issues/detail?id=112386>

Language Selection

Sunday, June 16, 2013 7:54 AM

Language selection for test case development for Selenium2(WebDriver)

Choices
There are three possibilities for language selection and these are based on available bindings for Selenium2.
Those are Java, Ruby and Python. Each has it own pros and cons as pointed out by our team members.

Let me summarize here with the higher level important items to keep in mind

- It should be easy to learn by someone that has as technical background.
  - Java has the largest learning curve
  - Python is easier then Java, but not as easy as Ruby
- A technical person should be able to view the code and get the gist of it.
  - Ruby is the easiest
- The performance should be good
  - Python is slightly better than Ruby
- What is SOE using?
  - They are moving more toward JavaScript and Ruby. Did I mention that they are also becoming big users of JQuery?
  - They are also a big user of Google applications which is a big user of Python.
- JavaScript bindings for Selenium2 is just not there yet. If is for Selenium1 Remote but that still is very limited and requires more apps just to install it. Not a good choice.
- That leaves us with Ruby and Python.
  - Ruby is easier to learn, but performance is slower
  - Python is harder to learn, better performance than Ruby, and has a larger code base than Ruby.

    OK, so what's the choice? If it were just our QA AUTOMATION team then I would say Python. But since other folks will be involved too I have to lean with Rudy.

    Bottom Line is that the scripting language for Selenium2 will be Rudy. But if it causes too many unknown problems we'll use Python as a backup.

    -Ken

Selenium Ruby test suite package

Monday, July 15, 2013 6:52 PM

Ruby Selenium test job packaging

A zip file containing test scripts needs to be copied to VM to run. Unzip it to a temporary directory, and use `rake test' (Ruby make) command to run it.

Tool Requirements

This solution requires the following additional Ruby gems:

rake (Ruby make)

```
Dir> gem install rake
```

VM has preinstalled unzipping software, 7-Zip.

-or-

rubyzip (*only if* using Ruby script to unzip the zip file)

```
Dir> gem install rubyzip
```

Zip File Content

A job zip file contains:

Rakefile

Rake (Ruby make) configuration file. It will read the manifest and execute every test case in it.

Note: It may be possible for the Rakefile to be installed on the system instead of being in the package, if the manifest is sufficient to declare the content of the test package. However, being in the package allows for customized test behavior per package.

Test case files (./test/TestCase.rb)*

All files with pattern TestCase*.rb under test subdirectory will be executed. Each .rb file is a test case. And each method in the file is a test step.

The tests are assumed to have no inter-dependency.

Example Files

Selenium test1.rb test case example
Selenium test2.rb test case example
Selenium test3.rb test case example
Selenium Ruby test package Rakefile example

Unzipping the zip file

There are two possible solutions:

1. Unzipping utility, 7-Zip
2. Ruby script
   This requires 'rubyzip' gem:

```
Dir> gem install rubyzip
```

7-Zip example

```
C:\Dev\try_ruby\tmp>7z.exe x -y job1.zip

7-Zip [64] 9.20  Copyright (c) 1999-2010 Igor Pavlov  2010-11-18

Processing archive: job1.zip

Extracting  Manifest.txt
Extracting  Rakefile
Extracting  test1.rb
Extracting  test2.rb

Everything is Ok

Files: 4
Size:       921
Compressed: 1091
```

Ruby unzip script example

```
Dir> ruby unzip.rb job1.zip
```

Running the test

Use `rake' command to run tests:

```
Dir> rake
```

It will automatically run the target, `test', by default.

In the actual Tellus system, custom output would be directed to a log and notification center.

> **Note:** If a step fails (assertion), the rest of the steps will still run.

Example:

```
C:\Dev\try_ruby>rake
C:/Ruby193/bin/ruby.exe -I"lib;test" -I"C:/Ruby193/lib/ruby/1.9.1"
"C:/Ruby193/l
ib/ruby/1.9.1/rake/rake_test_loader.rb" "test/test1.rb" "test/test2.rb"
"test/te
st3.rb"
Run options: --seed 16799

# Running tests:
```

```
Step 2-1 started
FStep 2-2 started
Step 2-2 ended
.Step 1-1 started
Step 1-1 ended
.Step 1-2 started
Step 1-2 ended
.Step 3-2 started
Step 3-2 ended
.Step 3-1 started
Step 3-1 ended
.

Finished tests in 0.027000s, 222.2222 tests/s, 222.2222 assertions/s.

  1) Failure:
test_0001_can_be_created_with_a_specific_size(Array)
[C:/Dev/try_ruby/test/test2
.rb:8]:
Expected: 11
  Actual: 10

6 tests, 6 assertions, 1 failures, 0 errors, 0 skips
rake aborted!
Command failed with status (1): [C:/Ruby193/bin/ruby.exe -I"lib;test" -
I"C:...]

Tasks: TOP => default => test
(See full trace by running task with --trace)
```

Note:

The message at the end:

```
rake aborted!
Command failed with status (1):
```

Is due to the failed test (step 1 of test case 2). It could be disregarded.

Selenium Ruby test1.rb test case example

Monday, July 15, 2013 8:43 PM

```
require 'minitest/spec'
require 'minitest/autorun'

describe Array do
  it "can be created with no arguments" do
    puts "Step 1-1 started"
    Array.new.must_be_instance_of Array
    puts "Step 1-1 ended"
  end

  it "can be created with a specific size" do
    puts "Step 1-2 started"
    #sleep 1
    Array.new(10).size.must_equal 10 # fail
    puts "Step 1-2 ended"
  end
end
```

Selenium Ruby test2.rb test case example

Monday, July 15, 2013 8:44 PM

```
require 'minitest/spec'
require 'minitest/autorun'

describe Array do
  it "can be created with a specific size" do
    puts "Step 2-1 started"
    #sleep 1
    Array.new(10).size.must_equal 11 # fail
    puts "Step 2-1 ended"
  end

  it "can be created with no arguments" do
    puts "Step 2-2 started"
    Array.new.must_be_instance_of Array
    puts "Step 2-2 ended"
  end

end
```

Selenium Ruby test3.rb test case example

Monday, July 15, 2013 8:44 PM

```
require 'minitest/spec'
require 'minitest/autorun'

describe Array do
  it "can be created with no arguments" do
    puts "Step 3-1 started"
    Array.new.must_be_instance_of Array
    puts "Step 3-1 ended"
  end

  it "can be created with a specific size" do
    puts "Step 3-2 started"
    #sleep 3
    Array.new(10).size.must_equal 10
    puts "Step 3-2 ended"
  end
end
```

Selenium Ruby test package Rakefile example

Monday, July 15, 2013 8:48 PM

```
# Rakefile
task :default => :test

require 'rake/testtask'

Rake::TestTask.new do |t|
    t.libs << "test"
    t.test_files = FileList["test/TestCase*.rb"]
    t.verbose = true
end
```

Test case structure

Tuesday, July 16, 2013 8:01 AM

A test case is a Ruby minitest unit test (a .rb file).

We use the "Specs" style of test script, which is more "English-like" and "descriptive".

Example test case

```
require "capybara"
require "selenium-webdriver"
require 'minitest/spec'
require 'minitest/autorun'

# test case
describe "when you are at Google page" do

    def setup
        # could inject config in future
        @url = "http://172.17.245.70:6700/wd/hub" # testcompleteb01
        #@url = "http://localhost:4444/wd/hub"
        #@url = "http://tellusdevb02.pc.factset.com:4444/wd/hub"
    end

    def teardown
        @driver.quit
        output_result @title
    end

    before do
        @caps = Selenium::WebDriver::Remote::Capabilities.chrome
        @driver = Selenium::WebDriver.for :remote, :url => @url,
            :desired_capabilities => @caps
    end

    # test step 1
    it "should match title on page" do

        @driver.navigate.to "http://google.com"

        element = @driver.find_element(:name, 'q')
        element.send_keys "Hello WebDriver!"
        element.submit

        @title = @driver.title

        # assertion match title
        @title.must_equal "Google"

    end

    def output_result(result)
```

```
        puts "Content-type: text/html"
        puts ""
        puts "{result: '#{result}'}"
    end

end
```

Running the test

```
ruby test.rb
```

Reference

https://github.com/seattlerb/minitest

Selenium Framework scripts location

Tuesday, July 16, 2013 9:37 AM

Location for Selenium Framework and test scripts:
- Production: //depot/smqa/Tellus/**Selenium**/
- Development: //depot/smqa/Tellus/**SeleniumDev**/

RPD:9581379

Ruby gems needed on VM

Friday, July 26, 2013 4:37 PM

JQuery

Thursday, June 27, 2013 10:20 AM

Link to Praveens Jquery PPT, it's in the jquery.zip file

H:\Department\QualityAssurance\QAAutomation\Automation\Praveen Projects\jquery.zip

- http://jquery.com
- http://api.jquery.com
- www.tutorilaspoint.com

Cucumber

Friday, July 12, 2013 10:30 AM

Cucumber is a tool for running automated tests written in plain language. Because they're written in plain language, they can be read by anyone on your team. Because they can be read by anyone, you can use them to help improve communication, collaboration and trust on your team.

Cucumber Ruby Installation

Friday, May 01, 2015 4:31 AM

Installing Ruby

Follow the setup as per the following first: http://infonet.factset.com/view/QualityAssurance/SeleniumTraining

Place the downloaded chromedriver in C:\Ruby193\bin

Launch command prompt and change directory to C:\Cucumber-*<GroupName>* by entering the command - cd C:\Cucumber-*<GroupName>* eg. cd C:\Cucumber-Analytics

Use the following commands to install all the required gems

gem install rake bundler yard watir-webdriver
gem install testgen
gem install bundler

To create a cucumber project use the following command in the command prompt:

testgen project *<project name>* --pageobject-driver=watir
Eg.
testgen project cucumber-demo --pageobject-driver=watir

Change into the newly created cucumber-demo directory by entering the following command - cd C:\Cucumber-*<GroupName>*\cucumber-demo

Use the following command:

bundle install

Launch the Project in Aptana

Launch Aptana
~~Enter the Workspace name as C:\Cucumber_<GroupName>~~

Use underscore in the workspace name as using hyphen may throw an error
Enter the Workspace name as C:\Cucumber_*<GroupName>*

Go to File-> Import

Expand General node -> Existing Folder as New Project
Browse and Select the respective project eg. C:\Cucumber-*<GroupName>*\cucumber-demo
Click Finish

Right click on features folder ->New->File
Provide the filename with .feature as the extension

Getting Started with Cucumber-jasmine for NodeJS

Friday, May 01, 2015 7:58 AM

jasmine-cucumber

Cucumber-*like* JavaScript API that works with Protractor and Jasmine. This is different from the *"official"* cucumber.js.

This is the library that Eyme used for her Cucumber Topical demo.

Installing node.js

1. Download the installer version 0.10.29 from http://nodejs.org/download/
2. Click **Windows Installer** (either 32-bit or 64-bit edition)
3. Run the downloaded MSI installer

Follow the instructions here: http://infonet.factset.com/view/QualityAssurance/ProtractorTraining

Installing protractor-jasmine-cucumber

```
cd <Your home> (usually C:\Users\username)
npm install protractor-jasmine-cucumber
```

For Analysts:

Launch the project in Webstorm
Select Create New Project
Browser to the Cucumber-Protractor project placed in the group folder (eg. Cucumber-Analytics)
Click Yes

Edit Configurations -> Change **C:\Cucumber-Protractor\conf.js** to the respective location where the project is placed
eg. **C:\Cucumber-Analytics\Cucumber-Protractor\conf.js**

For Scripters:

Example

protractor-conf.js

```
var cucumber = require('protractor-jasmine-cucumber');
exports.config = {
    baseUrl: 'http://localhost:3000',
    framework: 'cucumber',
    rootElement: 'body', // location of ng-app directive
    seleniumAddress: 'http://localhost:4444/wd/hub',
    capabilities: {
        'browserName': 'chrome'
    },
    suites: {
        suite1 : cucumber.injectFiles('*-specs.js', '*-steps*.js')
    },
    onPrepare: function() {
        var jasmine = require('jasmine-node');
        require('jasmine-reporters');
        jasmine.getEnv().addReporter(new jasmine.JUnitXmlReporter('C:/test', true, true));
    }
};
```

cucumber-spec.js

```
var feature = require('protractor-jasmine-cucumber').feature;
feature('Home Page')
        .scenario('Initial View of the Home Page')
        .when('There are 3 widgets to rent')
        .then('I see a list of 3 widgets for rent')
```

cucumber-step.js

```
var featureSteps = require('protractor-jasmine-cucumber').featureSteps;
featureSteps('Home Page')
        .when('There are 3 widgets to rent')
        .then('I see a list of 3 widgets for rent')
```

Executing Test

```
protractor protractor-conf.js
```

Cucumber.js Installation

Friday, March 06, 2015 12:13 PM

Note: This works with **chai** testing framework, but *not* Jasmine. (This is *not* the API that Eyme used for the Cucumber Topical demo.) Please see jasmine-cucumber for a cucumber-like API that works with Jasmine.

Installing node.js

1. Download the installer from http://nodejs.org/download/
2. Click **Windows Installer** (either 32-bit or 64-bit edition)
3. Run the downloaded MSI installer

Installing Protractor

```
npm install -g protractor
```

That will install `protractor` executable in this directory:

C:\Users*username*\AppData\Roaming\npm\

or on some machines, under C:\Program Files\...

Add that directory in your **PATH** environment variable.

Reference: https://github.com/angular/protractor

Installing Selenium Server from Protractor

Note: The following Standalone Server from Protractor is not needed if you are going to run against an existing Selenium Server or Selenium Grid.

The following will install selenium standalone server and chromedriver:

```
webdriver-manager update
```

To start Selenium server:

```
webdriver-manager start
```

It will start standalone Selenium Server on port 4444, by default.

Reference: https://github.com/angular/protractor/blob/master/docs/getting-started.md

Installing Cucumber.js and its Dependencies

```
npm install -g cucumber protractor
cd %NODE_PATH%
npm install expect.js
npm install chai-as-promised
```

Start Selenium Server

Using Selenium Jar:

In selenium.bat:

```
SET DIR=C:\Selenium
java -jar "%DIR%\selenium-server-standalone-2.43.1.jar" ^
 -Dwebdriver.chrome.driver=%DIR%\chromedriver.exe ^
 -Dwebdriver.ie.driver=%DIR%\IEDriverServer.exe ^
 -debug
```

Then run it:
`selenium.bat`

-OR-
(Optional) Install Selenium server using Node.js
`webdriver-manager update --standalone`

Then start Selenium on default port 4444:
`webdriver-manager start`
(To shutdown, press Ctrl+C)

Test Script Files

acceptance.conf.js

```
exports.config = {
  baseUrl: 'http://localhost:3000',
  rootElement: 'body', // location of ng-app directive
  seleniumAddress: 'http://localhost:4444/wd/hub',
  capabilities: {
    'browserName': 'chrome'
  },
  specs: ['specs/*.feature'],
  framework: 'cucumber',
  cucumberOpts: {
    // Optionally specify steps file;
    // by default, it runs all .js under spec/
    // path is relative to the conf.js
    require: 'specs/homepageSteps.js',
    format: 'pretty'
  }
};
```

homepage.feature

```
Feature: Home Page

  Scenario: Initial View of the Home Page
    Given There are 3 widgets to rent
    When I go to the home page
    Then I see a list of 3 widgets for rent.
```

homepageSteps.js

```
var chai = require('chai');
var chaiAsPromised = require('chai-as-promised');
chai.use(chaiAsPromised);
var expect = chai.expect;

module.exports = function() {
   // TODO: Add steps
};
```

Running Test

```
C:\Dir\widgets>protractor acceptance.conf.js
Using the selenium server at http://localhost:4444/wd/hub
[launcher] Running 1 instances of WebDriver

Feature: Home Page



  Scenario: Initial View of the Home Page    # specs\homepage.feature:3
    Given There are 3 widgets to rent        # specs\homepage.feature:4
    When I go to the home page               # specs\homepage.feature:5
    Then I see a list of 3 widgets for rent. # specs\homepage.feature:6


1 scenario (1 undefined)
3 steps (3 undefined)

You can implement step definitions for undefined steps with these snippets:

this.Given(/^There are (\d+) widgets to rent$/, function (arg1, callback) {
  // Write code here that turns the phrase above into concrete actions
  callback.pending();
});

this.When(/^I go to the home page$/, function (callback) {
  // Write code here that turns the phrase above into concrete actions
  callback.pending();
});

this.Then(/^I see a list of (\d+) widgets for rent\.$/, function (arg1, callback
) {
  // Write code here that turns the phrase above into concrete actions
  callback.pending();
});

[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 passed
```

Note: It will run all .js files under specs directory.

Fill out homepageSteps.js

```
// Add this.When(), this.Then():
var chai = require('chai');
var chaiAsPromised = require('chai-as-promised');
chai.use(chaiAsPromised);

var expect = chai.expect;

module.exports = function() {
    this.Given(/^There are (\d+) widgets to rent$/,
             function (arg1, callback) {
      // Write code here that turns the phrase above
      // into concrete actions
      callback.pending();
    });

    this.When(/^I go to the home page$/, function (callback) {
      // Write code here that turns the phrase above
```

```
      // into concrete actions
      callback.pending();
    });

    this.Then(/^I see a list of (\d+) widgets for rent\.$/,
            function (arg1, callback) {
      // Write code here that turns the phrase above
      // into concrete actions
      callback.pending();
    });
};
```

Run again

```
protractor acceptance.conf.js
```

Output

```
Using the selenium server at http://localhost:4444/wd/hub
[launcher] Running 1 instances of WebDriver

Feature: Home Page



  Scenario: Initial View of the Home Page    # specs\homepage.feature:3
    Given There are 3 widgets to rent        # specs\homepage.feature:4
    When I go to the home page               # specs\homepage.feature:5
    Then I see a list of 3 widgets for rent. # specs\homepage.feature:6


1 scenario (1 pending)
3 steps (1 pending, 2 skipped)
[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 passed
```

Reference

Tutorial:
https://docs.google.com/presentation/d/1UNAkIr4dZK6leWhNJ1J96kPQlbwAWdgDsBJab9-NE4c/present?pli=1&ueb=true&slide=id.p

Protractor project itself uses cucumber.js with chai:
https://github.com/angular/protractor/blob/master/docs/frameworks.md (near the bottom of the page)

Cucumber.js Home:
https://github.com/cucumber/cucumber-js

Twiki:
http://infonet.factset.com/view/QualityAssurance/ProtractorTraining
http://infonet.factset.com/view/QualityAssurance/SeleniumTraining

Apps Requirements

Friday, July 12, 2013 2:56 PM

List needed apps here for browser, headless browser, Ichabod, HTML5 and others here. Also list those apps that will have to be called from the Ruby Test Scripts.

- What Selenium apps and supporting apps need to be on the VMs.
- Ruby - What drivers/apps it needs to have installed. What apps ruby test cases will have to call.
- Browser Testing - This may have to be broken up by Browser.
- Headless Browser Testing -
- Ichabod -
- HTML5 testing.
- Add Others Here

InstallProcess

Monday, July 15, 2013 7:40 PM

- I'm trying to nail now the order that would run Phantomjs on the VM
    - Install Ruby 1.9.3,
    - install DevKit(ruby dk.rb init, ruby dk.rb install),
    - gem install capybara,
    - gem install poltergeist,
    - phontomJS is just an executable that needs to be in the path.

IS Morph Windows Authentication Setup

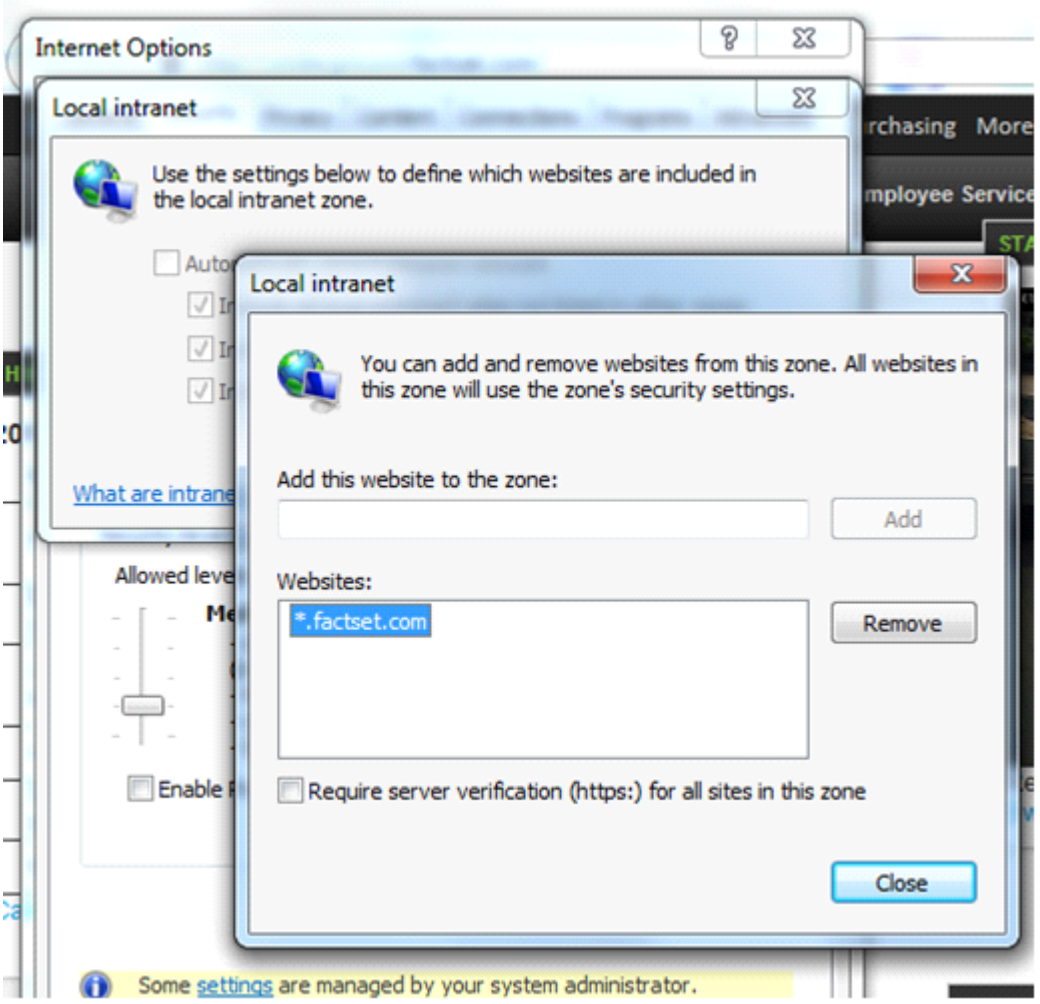
Tuesday, February 24, 2015 10:16 AM

Agenda:

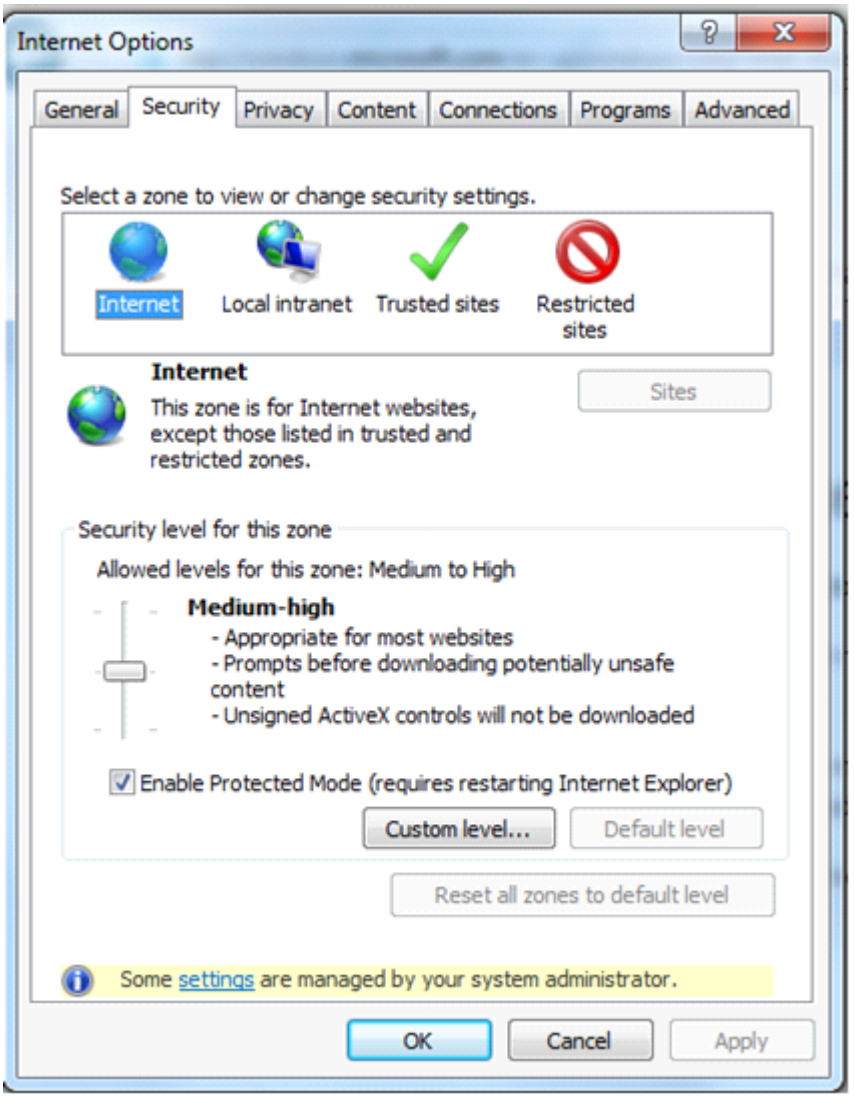
1. Ensure after updates that require reboots and when new nodes/hubs are added the browsers are reset to have *.factset.com as a trusted site.
    - Joseph Hwang determined that only firefox can be set via the script, but IE and Chrome cannot.

Info I need

- Show me the setting to confirm
    - `*.factset.com` in **Local Intranet** sites
    - Reg key or GPO?
    -



    - **Protected Mode** enabled for all 4 zones



- List of servers that need this setting with domain:
  - Tellusgridb03
  - tellusgridb01
- Off domain and Short term: We can set these manually.
- On domain and Long term: I'll get these PCs into a new organizational unit and apply this setting as group policy.

Joseph LaPorte | **Senior Systems Engineer, Windows Team** | ✉ jlaporte@factset.com | ☎ 203.810.1075 | FactSet Research Systems Inc.

OverView

Friday, July 19, 2013 9:31 AM

We will keep track of the Selenium Project here. I will create sub pages for short and long term tasks.

Sel2ShortTermGoal

Friday, July 19, 2013 9:33 AM

Last Update 7/19/2013 9:40 am EST

Selenium2 Short Term Goal (Tellus Developmemt Environment)

The user selects one of two test cases. If the use selects BowserHTML and submits that then that browser based test case will be executed on the PlayList VM and the results show up on the Tellus results page. If the use selects HeadLessBowserHTML and submits that then that headless browser based test case will be executed on the PlayList VM and the results show up on the Tellus results page.

What needs to be done to achieve this goal? I'll list them; maybe not number them as they are not by priority.

- The two Ruby test cases have to be written and reside in the Selenium structure in Perforce.(Shiva, Bhupender assisted Joseph)
  - BrowserHTML will execute a HTML test case using Ruby and WebDriver
    - It will write results to a log file
  - HeadLessBrowerHTML will execute a HTML test case using Ruby and PhantomJS (and everything else in-between, DevKit, capybara, poltergeist)
    - It will write results to a log file
- The two test cases are present in the Tellus Dev UI (Frank)
- The User makes their "run it now" selections from the Tellus Dev UI and then clicks the Submit button. (Frank)
  - Parameters are saved in the HMTL file(or a replacement for that)
- Tellus framework sends the VM Playlist to FocalPoint and waits for the IP address.(Vasu, Gautham)
- The Test Case and Test Case List is sent to the VM in the proper directory structure (Vasu, Gautham)
- Tellus executes a Ruby script on the VM that will execute the Ruby script in the list.(Gautham, Vasu assisted by Joseph)
  - This needs to be setup to handle one or more Ruby test cases.
- When finished executing the test case the log(s) are returned to the Tellus controller. (Vasu, Gautham)
- The results are parsed written to the Database and the Tellus results web page. (Vasu, Gautham, Frank, Joseph)

GroupEndToEndGuildeLines

Monday, July 29, 2013 8:01 AM

Automation Person

The Automation person receives the requirements from the FM/TL to Automate a test case.
The Automation person creates the test case locally using Ruby scripting language.
The Automation person needs to identify where the HTML (others) to be tested comes from and where they need to be sent to.
The Automation person will insure that they are using the correct driver(s)(WebDriver or Poltergeist)
The Automation person will insure that they are logging the proper error messages to the XML log file.
The Automation person moves the code to the Development environment.
The Automation person will test on the Development environment and then show the FM/TL
Once the FM/TL agree the test case will be moved to Production and exercised there
Once the FM/TL agree the test case can be moved to the CRON Calendar.

The User

The user selects one of two types(for starters) of test cases.
If the use selects BowserHTML and submits that then that browser based test case will be executed on the PlayList VM and the results show up on the Tellus results page.
If the use selects HeadLessBowserHTML and submits that then that headless browser based test case will be executed on the PlayList VM and the results show up on the Tellus results page.

The Framework

The Framework Person insures that the Tellus framework manages the deploying and undeploying of VMs as before.
The Framework Person insures that the proper Test Cases are retrieved from P4 and send to the VMs under test.
The Framework Person insures that the test cases can be executed from a list of test cases on the VM under test.
The Framework Person insures that the results are retrieved from the VM under test.
The Framework Person insures that the results are posted to the database.

The Tellus UI

The UI person will insure that the test cases and browser choices are present in the new Tellus UI.
The UI person will insure that the proper selections are present in the UI and recorded in the HTML or Database when selected .
The UI person will insure that the proper results are posted to the new Tellus UI
The UI person will insure the results are linked to the CRON Calendar.

General Flow

- The two Ruby test cases have to be written and reside in the Selenium structure in Perforce .(Shiva, Bhupender assisted Joseph)
  - BrowserHTML will execute a HTML test case using Ruby and WebDriver
    - It will write results to a log file
  - HeadLessBrowerHTML will execute a HTML test case using Ruby and PhantomJS (and everything else in-between, DevKit, capybara, poltergeist)
    - It will write results to a log file
- The two test cases are present in the Tellus Dev UI (Frank)
- The User makes their "run it now" selections from the Tellus Dev UI and then clicks the Submit button. (Frank)
  - Parameters are saved in the HMTL file(or a replacement for that)
- Tellus framework sends the VM Playlist to FocalPoint and waits for the IP address. (Vasu, Gautham)
- The Test Case and Test Case List is sent to the VM in the proper directory structure (Vasu, Gautham)
- Tellus executes a Ruby script on the VM that will execute the Ruby script in the list. (Gautham, Vasu assisted by Joseph)
  - This needs to be setup to handle one or more Ruby test cases.
- When finished executing the test case the log(s) are returned to the Tellus controller. (Vasu, Gautham)
- The results are parsed written to the Database and the Tellus results web page. (Vasu, Gautham, Frank, Joseph)

FlowEndToEnd

Monday, July 29, 2013 8:47 AM

General Flow

- The two Ruby test cases have to be written and reside in the Selenium structure in Perforce.(Shiva, Bhupender assisted Joseph)
  - BrowserHTML will execute a HTML test case using Ruby and WebDriver
    - It will write results to a log file
  - HeadLessBrowerHTML will execute a HTML test case using Ruby and PhantomJS (and everything else in-between, DevKit, capybara, poltergeist)
    - It will write results to a log file
- The two test cases are present in the Tellus Dev UI (Frank)
- The User makes their "run it now" selections from the Tellus Dev UI and then clicks the Submit button. (Frank)
  - Parameters are saved in the HMTL file(or a replacement for that)
- Tellus framework sends the VM Playlist to FocalPoint and waits for the IP address.(Vasu, Gautham)
- The Test Case and Test Case List is sent to the VM in the proper directory structure (Vasu, Gautham)
- Tellus executes a Ruby script on the VM that will execute the Ruby script in the list.(Gautham, Vasu assisted by Joseph)
  - This needs to be setup to handle one or more Ruby test cases.
- When finished executing the test case the log(s) are returned to the Tellus controller. (Vasu, Gautham)
- The results are parsed written to the Database and the Tellus results web page. (Vasu, Gautham, Frank, Joseph)

Automation Person (Bhupender, Shiva)

- The Automation person receives the requirements from the FM/TL to Automate a test case.
- The Automation person creates the test case locally using Ruby scripting language.
- The Automation person needs to identify where the HTML (others) to be tested comes from and sent to.
- The Automation person will insure that they are using the correct driver(s)(WebDriver or Poltergeist)
- The Automation person will insure that they are logging the proper error messages to the XML log file.
- The Automation person moves the code to the Development environment.
- The Automation person will test on the Development environment and then show the FM/TL
- Once the FM/TL agree the test case will be moved to Production and exercised there
- Once the FM/TL agree the test case can be moved to the CRON Calendar.

The User: Tellus UI

The user goes to the new Tellus UI and configures either a adhoc or a scheduled test.

The user selects one of two types(for starters) of test cases.

If the use selects BowserHTML and submits that then that browser based test case will be executed on the PlayList VM and the results show up on the Tellus results page.

If the use selects HeadLessBowserHTML and submits that then that headless browser based test case will be executed on the PlayList VM and the results show up on the Tellus results page.

The Framework (Gautham, Vasu)

- The Framework Person insures that the Tellus framework manages the deploying and undeploying of VMs as before.
- The Framework Person insures that the proper Test Cases are retrieved from P4 and send to the VMs under test.
- The Framework Person insures that a list of test cases are created and will be sent with the test cases to the VM under test.
- The Framework Person insures that the test cases can be executed from a list of test cases on the VM under test.
- The Framework Person insures that the results are retrieved from the VM under test.
- The Framework Person insures that the results are posted to the database.

The Tellus UI (Frank, Joseph)

- The UI person will insure that the test cases and browser choices are present in the new Tellus UI.
- The UI person will insure that the proper selections are present in the UI and recorded in the HTML or Database when selected.
- The UI person will insure that the proper results are posted to the new Tellus UI
- The UI person will insure the results are linked to the CRON Calendar.
- The UI person will insure that only proper time slots can be utilized in the CRON Calendar.

SeleniumDevelopment

Wednesday, July 31, 2013 9:32 AM

- Developers High Level
  - Ruby & drivers
  - Browser Testing
  - Headless Browser Testing
  - ThiefJS
  - Tellus Selenium2 Infrastructure
  - Chromium CEF3
  - Debugging
  - Selenium

- Framework high level
  - Ruby & drivers
  - Browser Testing (understanding)
  - Headless Browser Testing (understanding)
  - ThiefJS (understanding)
  - Tellus Selenium2 Infrastructure
  - Chromium CEF3 (understanging)
  - Ruby Controller(Rake, MiniTest)
  - Selenium logging to DB
  - Debugging
  - Selenium

- Tellus UI high level
  - Tellus DB
  - Tellus Redesign
    - CRON Calendar
    - Results Page
    - Progress Page
    - Configuration
  - Debugging
  - Headless Browser Testing
  - ThiefJS
  - Tellus Selenium2 Infrastructure
  - Chromium CEF3
  - Ruby Controller(Rake, MiniTest)
  - Selenium logging to DB
  - Selenium

Lower Levels

- Ruby
  - Installation
  - GEM
  - Require
  - Logging
  - Rake unless a better option

- Bindings/Drivers/Apps
    - Bindings/Drivers
        - WebDriver
        - Capybara
        - Poltergeist
        - Watir (pronounced Water)
    - Apps
        - Phantomjs
        - Ichabod

- FactSet Components
    - ThiefJS
    - Chromium CEF3
    - HTML5
        - What it is
        - How to test
        - Location of source code

Ruby: Optimize existing CGI to FastCGI

Friday, August 09, 2013 5:32 PM

I will update this page soon.
Frank

Tools List

Sunday, September 15, 2013 5:44 AM

Selenium2 tools

This is the list of current known drivers/apps/technology we are going to use. The list could grow or become shorter as me move forward.

- Ruby Test case development
  - Aptana Studio 3 – Ruby coding
  - Esclipse - Ruby coding
  - Notepad ++ - Ruby coding
- Logging
  - Logger.rb (Internal app created by Shiva)
- Drivers required for code written above
  - WebDriver
  - Capyabara
  - Poltergeist
  - Watir
- App for headless browser
  - PhantomJS
- Ruby test case controller
  - Rake (modified by us)
  - Mini
- Modular Based Test Cases
  - Human readable
    - Refactoring
  - Human Writable
    - Convert text based skeletons to code: Rspec maybe
    - SpreadSheet
- Grouping
  - Bundler

AngularJS 5/9/14

Friday, May 09, 2014 11:04 AM

AngularJS modeling

Rest API is a URL

Use nouns not verbs

Restangular

BreezeJS

Learn.breezejs.com

ActiveRecord for AngularJS

ORM object-relational mapping.

Ruby

PouchDB

Resources

WebAPIdesign
Data persistence in angularjs2.0
http://bit.ly/zen-ng-data

https://docs.google.com/presentation/d/1J--MwiYn_uxWy7r3XDwr9gm4JCx5sEFWXE7dmjQzz7c/present?pli=1&ueb=true#slide=id.p

Appium General Info

Thursday, July 24, 2014 9:44 AM

From Joseph

- **Appium**:
  - Performed proof-of-concept demo: drive Android app in Emulator (test script in Ruby)
  - Test scripts are standard **Selenium** test scripts (with app info passed to driver as a few options)
  - Java test script is also working with Android Emulator
  - Verified: (Ruby) test script works with real Android device
  - [In Progress] Testing hybrid Android app
  - About Appium:
    - Is implemented as a Selenium server, and bridges to mobile device
    - Supports native and **hybrid** apps--implication is **one test script for both desktop *and* mobile** "responsive" Web app (with minor handling of differences)
    - Has **recorder** (but seems to be available only on Mac)
    - Supports iOS, Android (and FirefoxOS)
    - Supports multiple standard languages, including Java, JavaScript, Ruby, etc.
    - Appium is open source (Apache 2.0 License)
    - Works with apps as-is; apps do not need to be "instrumented" (linked with "agent" library) like MonkeyTalk
    - Project home: http://appium.io/
  - Looked into hybrid mobile app