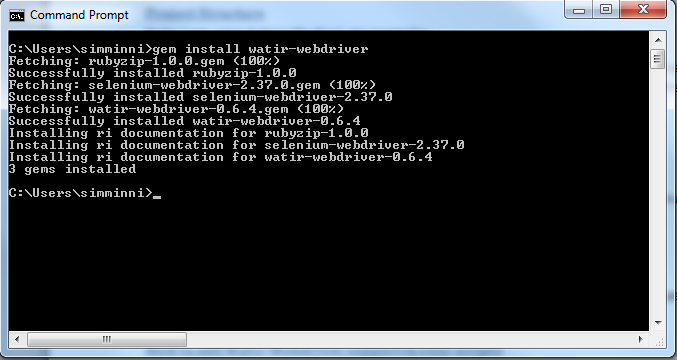
## Selenium-Ruby-Watir-Webdriver

### How to add Watir-Webdriver support in your scripts

First of all you need to install Watir-Webdriver gem on your machine. It’s pretty simple.

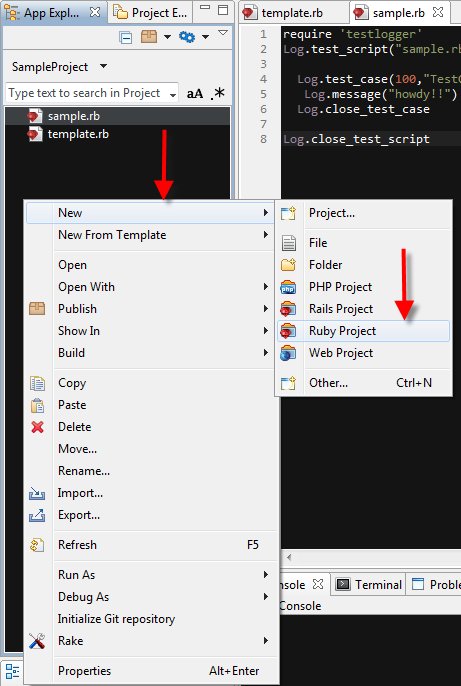
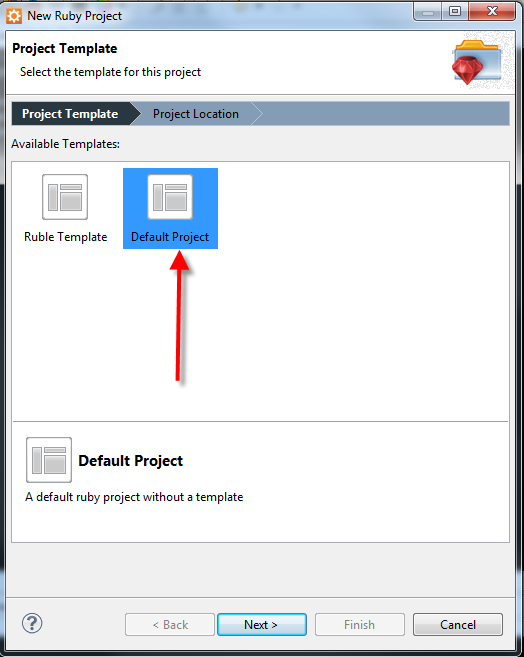
**Installing watir-webdriver gem**

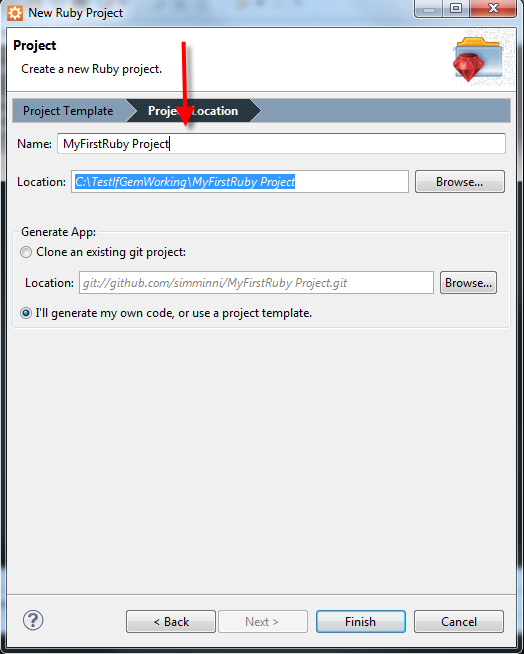
1. Open command prompt.
2. Type “gem install watir-webdriver”
3. You should see a confirmation message that it is installed. Note: If you don’t have selenium-webdriver gem already installed, let me tell you, watir-webdriver has selenium-webdriver as a dependent gem. All the dependent gems for watir-webdriver gem would also get installed automatically when you install watir-webdriver. Remember? In the tutorial [Selenium History](http://www.questionselenium.com/2013/11/selenium-history.html), we have seen that water-webdriver implements selenium-webdriver. This is the reason why watir-webdriver is dependent on selenium-webdriver.



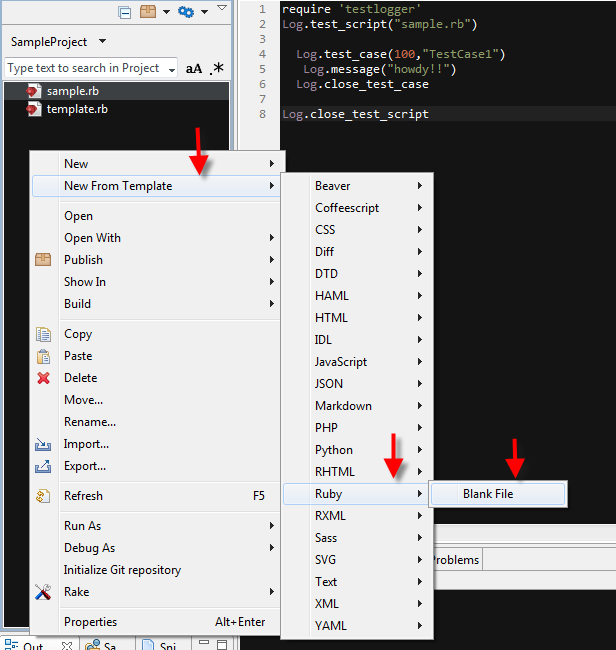
**Steps to include the support for watir-webdriver in your scripts**

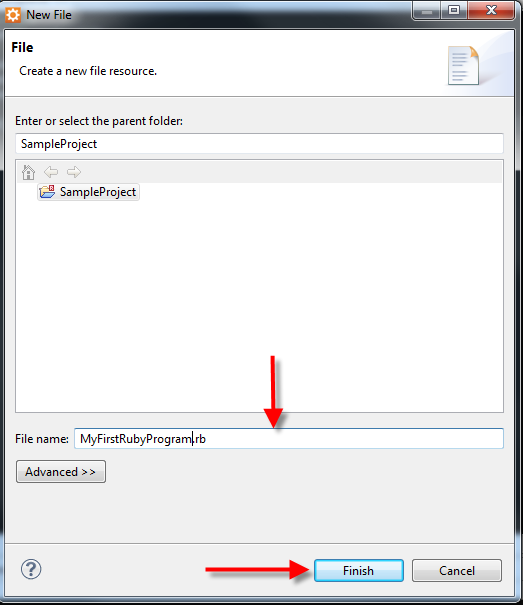
1. Open Aptana Studio
2. Create a new project if you don’t have one already
   1. Right click on left hand pane of Aptana
   2. Move cursor over “New”
   3. Move cursor over “Ruby Project”
   4. Select “Default Project” from New Project window. Click Next button.
   5. Let the defaults stay for all the fields but enter a suitable project name in Name filed. Click Finish button.



1. Create a new ruby file
   1. Right click on left hand pane of Aptana
   2. Move cursor over “New From Template ”
   3. Move cursor over “Ruby”
   4. Move cursor over “Blank File”
   5. Enter a suitable name to your script. Make sure you have “.rb” as extension while adding the name to the script. Click finish.

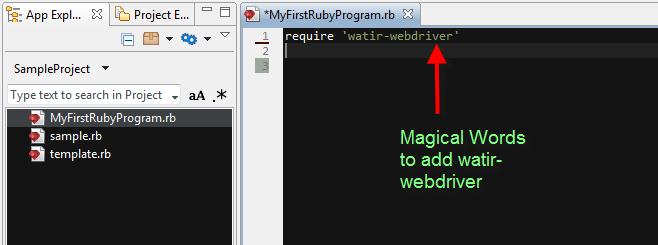




1. Open script that you have just created from left hand pane.
2. Type the following line of code on top of your script

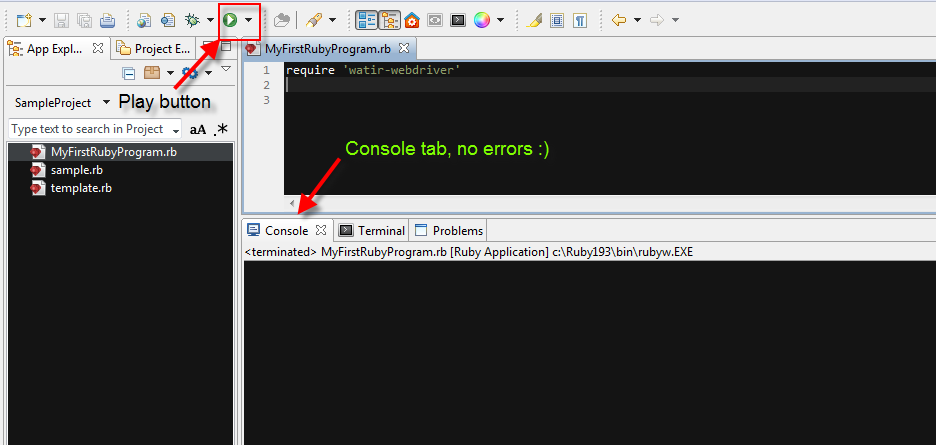
require “watir-webdriver”

Note: In ruby if we install a gem(library), we can add it to any of our scripts using “require” keyword followed by gem(library) name in quotes.



1. Now to verify that watir-webdriver is added to your script, just click on **run** button on Aptana Studio tool panel.

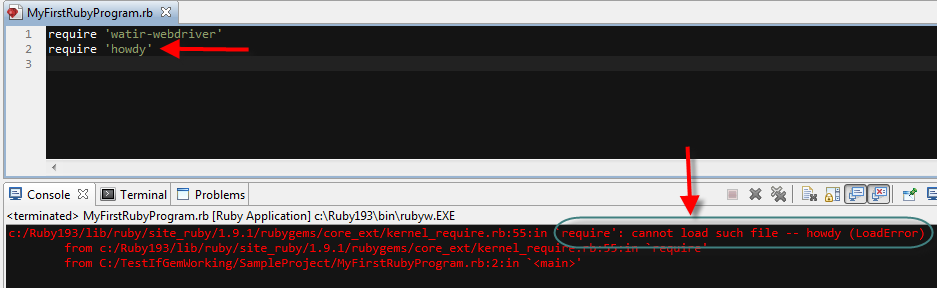
If you see any errors in console, this would mean you did something wrong while configuring or installing watir-webdriver.



1. You don’t believe if technique mention in step 6 is working or not, do you ?

Ok let’s clarify our doubt.

Type ‘require howdy’ and click on play button. You would see some errors in console. Now you can confidently believe that technique at step 6 is working and is a proof that watir-webdriver is successfully added to script.



### Some of simple operations automated

**Opening a browser with given URL**

We would like to show how you can do this in 3 browsers: Firefox, Chrome and Internet Explorer. If you don’t have these browsers installed, please do that before proceeding any further.

**Opening a browser with given URL – Chrome**

Remember in **Selenium History** we said Selenium uses browser based browser-driver. We need to set the path to this browser-driver. It is an executable file. You can download it from <http://chromedriver.storage.googleapis.com/index.html> . Note Google might change the location of this executable. In case you don’t find them there, let me know.

Click on 2.6 link or any other latest version you see there. Then click chromedriver\_win32.zip. Your executable would be downloaded in form of a zip file. Extract and save the executable to a location. Note: There are chances that you might run into Version Compatibility problem. By Version Compatibility problem, I mean the browser-driver you downloaded sometimes may not be supporting the version of browser installed on your machine. In that case I would suggest you try different versions available at the download link given above ☺ I have place executable/browser-driver under C drive.

**Code:**

#adding watir-webdriver support

require 'watir-webdriver'

#setting browser-driver executable location

Selenium::WebDriver::Chrome.driver\_path = File.expand\_path('C:/chromedriver.exe')

# Instantiated chrome browser. Below code will launch a chrome browser

oPage = Watir::Browser.new :chrome

#specify which URL to navigate in the opened browser

oPage.goto "www.google.com"

#print title of the page to console

oPage.title

#close the browser

oPage.close

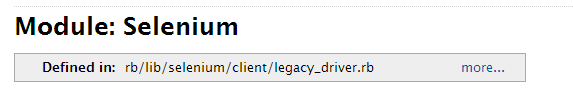
#or oPage.quit

Explanation:

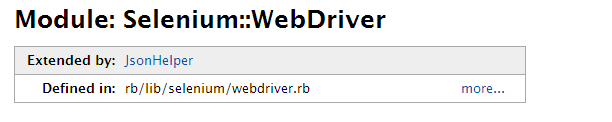
1. **Selenium::WebDriver::Chrome.driver\_path**

In the above syntax,

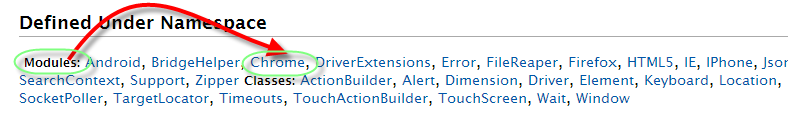
* + **Selenium** is a module (building block of ruby programming language, refer: [modules: tutorialpoint](http://www.tutorialspoint.com/ruby/ruby_modules.htm) for more information). Open [selenium documentation](http://selenium.googlecode.com/svn/trunk/docs/api/rb/Selenium.html) and you can find it is clearly stated that Selenium is a module.



* + **WebDriver** is also a module. It is a sub module of Selenium. Yes you can have a module inside a module.  
    Open [selenium documentation](http://selenium.googlecode.com/svn/trunk/docs/api/rb/Selenium.html) and click on WebDriver under “Defined under Namespace” section (You are seeing WebDriver under this section because it is a sub module). You will be navigated to [WebDriver](http://selenium.googlecode.com/svn/trunk/docs/api/rb/Selenium/WebDriver.html) page. Here at the top of the page you can see that Selenium::WebDirver termed as module.

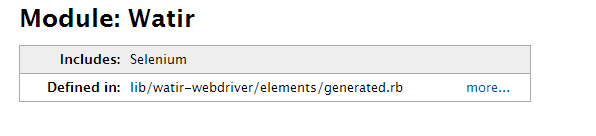


* + **Chrome** is also a module? Yes it is☺. Like I said before we can have a module inside a module, this child module can also have module inside it. This chain can go for ever. This is to give modularity to our code. Modules are building blocks of ruby program that helps in structuring the code.



* + **driver\_path** is a **method** undermodule Chrome. This method is used to tell watir-webdriver where the chrome’s browser-driver is located. Note that we are calling Selenium::WebDriver API call here. Watir-WebDriver would internally use Selenium-WebDriver and it would take care of looking for this attribute to launch a browser.

1. **oPage = Watir::Browser.new :chrome**
   * **oPage,** this is just a variable. For more information on ruby variables, go through [variables: tutorialspoint](http://www.tutorialspoint.com/ruby/ruby_variables.htm) We would be using this variable to store reference to chrome browser.
   * **Watir** is a module which is packed and supplied to you as watir-webdriver gem. Once you required this gem in your script, you can make watir API calls.   
     Open [watir documentation](http://rubydoc.info/gems/watir-webdriver/0.1.7/Watir) and now you can identify if Watir is a module or not.



* + **Browser** is a class. For which we are creating a new instance using **.new** and parameterized constructor. You can find this under [official documentation](http://rubydoc.info/gems/watir-webdriver/0.1.7/Watir/Browser). Note that this parameter is optional. By default this is set to :firefox



* + **:chrome**

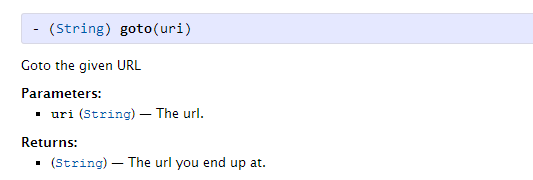
This is nothing but a symbol. More info on symbols can be found [here](http://www.ruby-doc.org/core-1.9.3/Symbol.html).

We can also use a string “chrome” in place of symbol :chrome but since using symbol is recommended [in official documentation](https://code.google.com/p/selenium/wiki/RubyBindings), we are opting it. You can try out using a symbol in place of symbol and notice that would work.



1. **oPage.goto “www.google.com”**

* **oPage:** User defined variable holding the reference to Browser instance
* **goto:** Method under Class: [Browser](http://rubydoc.info/gems/watir-webdriver/0.1.7/Watir/Browser) **.** Let us have a look at its API. In the above statement we have passed the URL as “www.google.com”



1. **oPage.title**

title: returns the title of the page loaded into browser window.

1. **oPage.close**

close: Closes active browser tab. At times multiple tabs are opened when clicked on links present at homepage. We tend to switch the control to the child tab to perform some operation. If you switch the control to a different tab then if you want to close the child tab alone, close is the function you need to call on Browser object.

1. **oPage.quit**

quit: Closes all the tabs opened during automation run. Call this method when you are sure that you want all tabs to be terminated, usually at the end of a test run.

**Opening a browser with given URL – Internet Explorer**

Like we required a browser-driver for chrome, we also require one for Internet Explorer.

You can find 32 bit browser-driver [here](https://code.google.com/p/selenium/downloads/detail?name=IEDriverServer_Win32_2.37.0.zip) and 64 bit browser driver [here](https://code.google.com/p/selenium/downloads/detail?name=IEDriverServer_x64_2.37.0.zip).

Once you have appropriate version of browser-driver, all we need to change is two lines of code form previous program.

1. Change the location parameter to Internet Explorer browser-driver.  
   Ex: Selenium::WebDriver::Chrome.driver\_path = File.expand\_path('C:/ IEDriverServer.exe')
2. Change the parameter from **:chrome** to **:ie** while initializing the Browser object.   
   Ex: oPage = Watir::Browser.new :ie

With the above changes made, run the code. You should see an IE browser opened followed by automation code we written.

**Opening a browser with given URL – Firefox**

Like we required a browser-driver for Chrome and IE, we don’t need one for Firefox. This means we can remove the line of code where we are setting location to browser-driver.

Open [Browser class API](http://rubydoc.info/gems/watir-webdriver/0.1.7/Watir/Browser) under Watir module. Look at initialize method (constructor). Note that the default parameter is set to “firefox”. This mean if you are not passing any value while creating a browser object, by default firefox browser would open.It is optional to pass :firefox .

**oPage = Watir::Browser.new** is valid

**oPage = Watir::Browser.new :firefox** is also valid