

## Phần 1: Cài đặt Selenium IDE

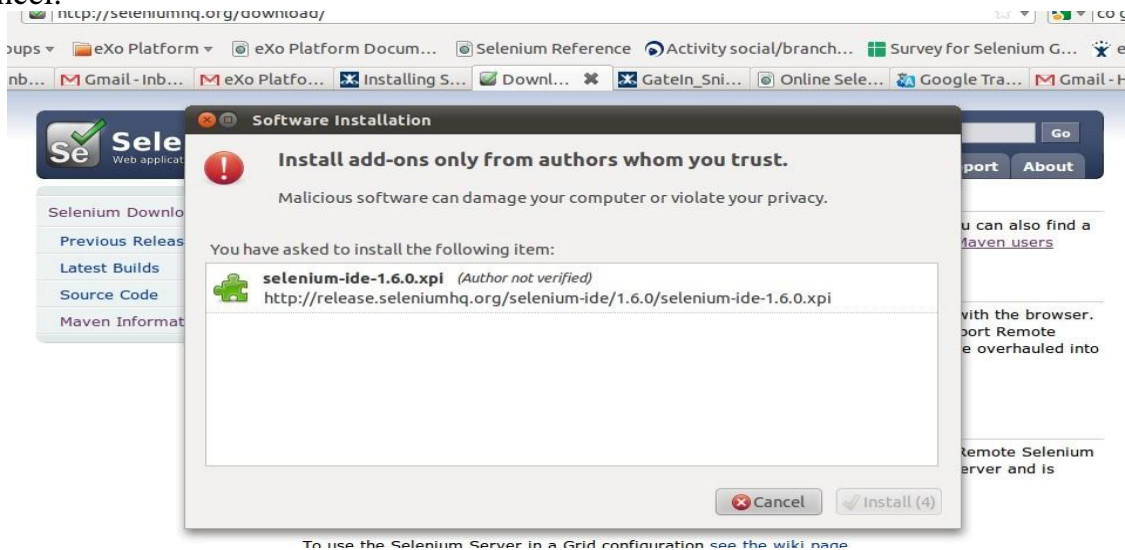
Để kiểm tra xem trên trình duyệt Firefox đã được cài selenium IDE. Bạn có thể vào bất trình duyệt này và nhấp vào Tools trên menu Bar. Quan sát trên menu đổ xuống có mục selenium IDE không? Nếu chưa có bạn hãy thực hiện những bước sau để có thể cài cho mình tiện ích này của Firefox

**Bước 1: Mở trang chủ của selenium <http://seleniumhq.org/download/>**



**Bước 2 : Cài đặt selenium IDE giống như cài đặt một plugin trên trình duyệt Firefox**

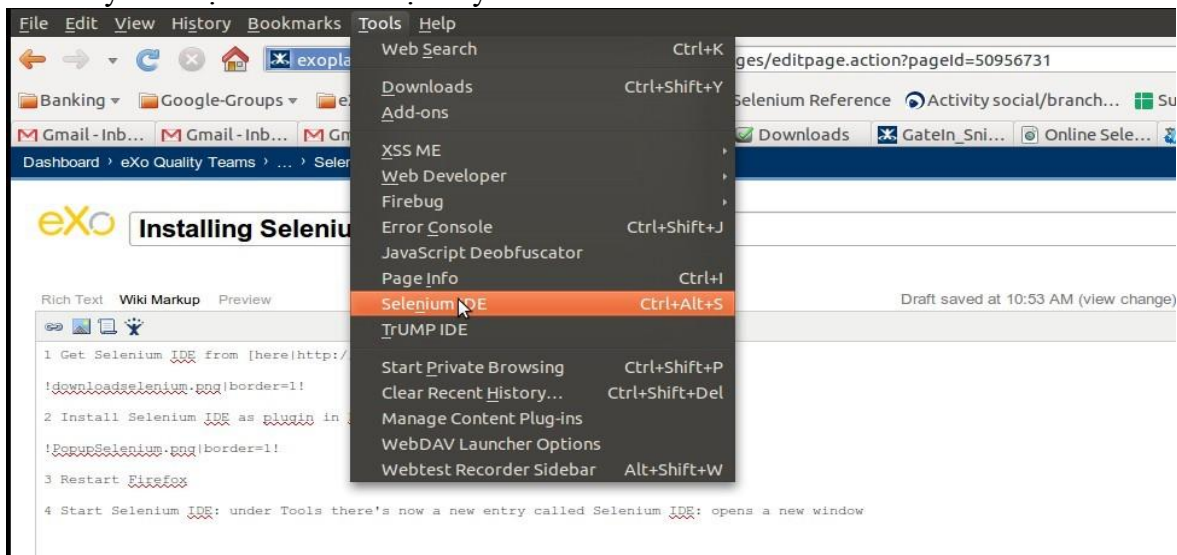
Click vào link 1.6.0 (chỉ số phiên bản này có thể thay đổi theo thời gian) nó sẽ bật ra cửa sổ Software Installation như hình dưới và chờ đợi trong vài giây sau đó click vào nút Install cạnh nút Cancel.



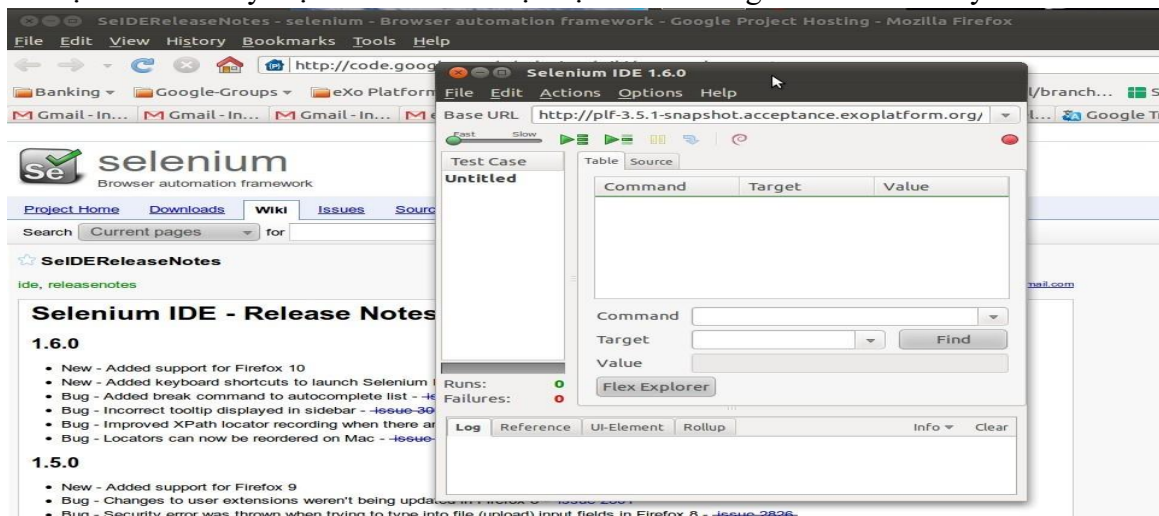
**Bước 3: Khởi động lại trình duyệt Firefox**

#### Bước 4: Lần đầu khởi động selenium IDE của bạn

Click vào Tool trên menu Bar. phía dưới bạn sẽ nhìn thấy một mục mới có tên gọi là selenium IDE. sau khi di chuyển chuột và lick lên mục này



Sau đó bạn sẽ nhìn thấy một cửa sổ mới được bật ra như trong hình vẽ dưới đây

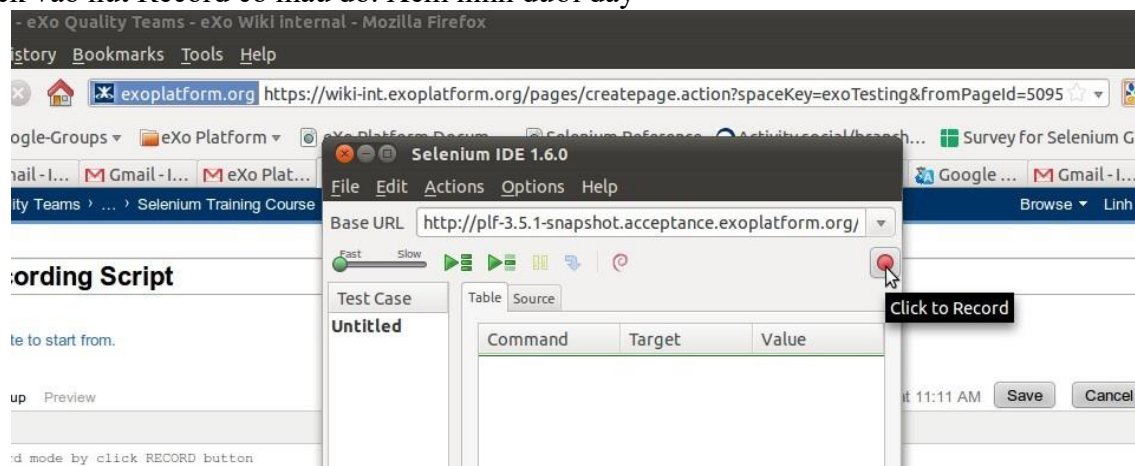


Kết quả là bạn sẽ nhận được một cửa sổ trống Selenium IDE. Chú ý rằng có một danh sách các TestCase bên trái của bảng với mỗi trường hợp đều được bắt đầu với tên *Untitled*. Bạn có thể di chuyển hay thay đổi kích thước của cửa sổ này bằng indicator bên phải

## Phần 2: Khởi động chế độ ghi và ví dụ đơn giản

### 2.1 Khởi động chế độ ghi

Theo mặc định sau khi bật selenium IDE lên thì chế độ record đã được bật. Để tắt/bật chế độ này chỉ cần click vào nút Record có màu đỏ. Xem hình dưới đây

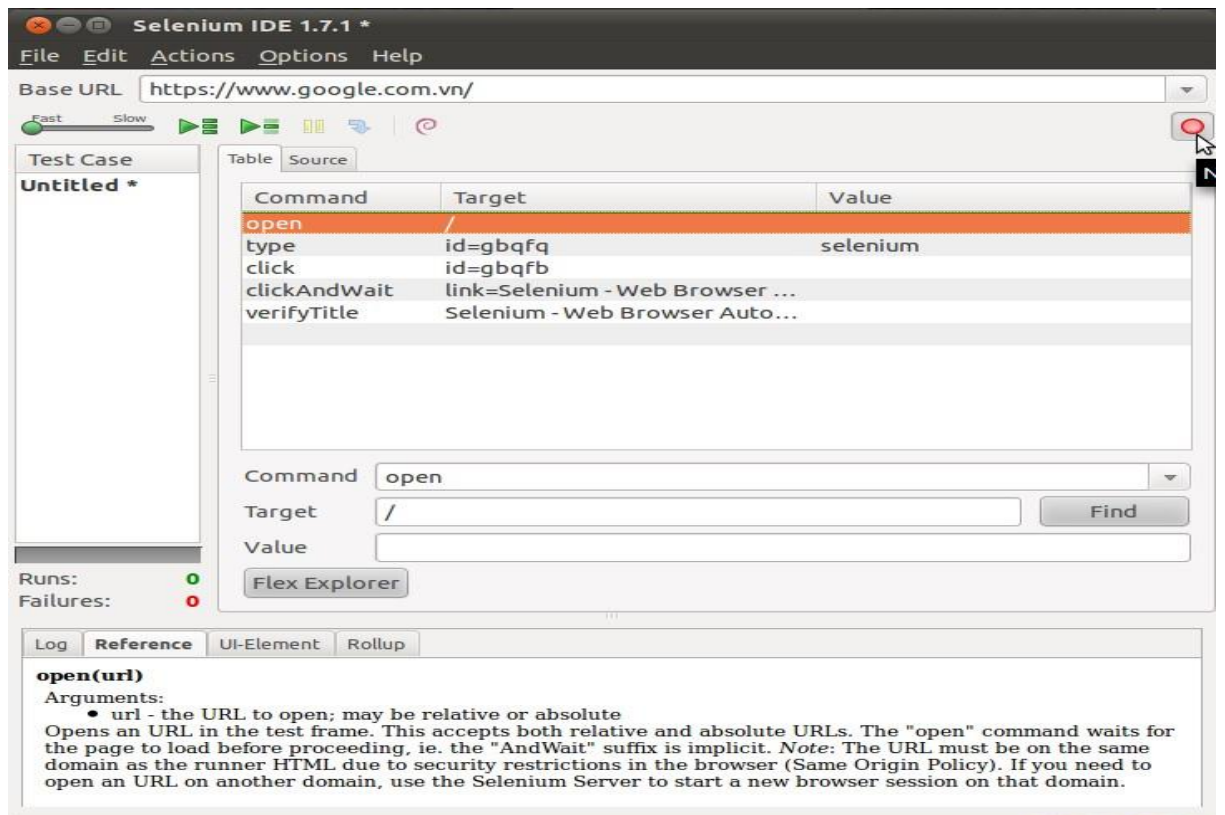
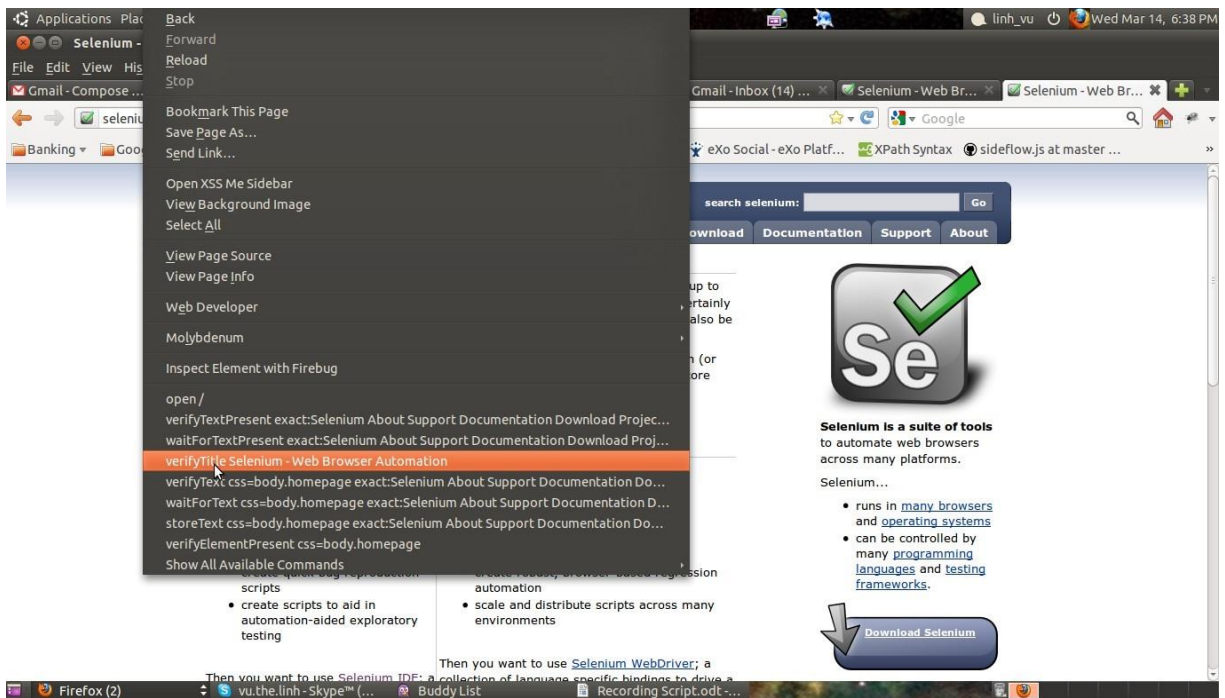


## 2.2 Ví dụ đơn giản

Bây giờ bạn đã sẵn sàng cho việc ghi lại selenium scripts đầu tiên của bạn. Dưới đây là một kịch bản đơn giản để chúng ta thực hành cách ghi một scripts bằng selenium IDE

1. Bật trình duyệt Firefox
2. Truy cập vào trang <http://www.google.com.vn>
3. Gõ từ khóa tìm kiếm là : selenium
4. Truy cập vào trang chủ của selenium trong bảng kết quả trả về bởi Google
5. Kiểm tra xem title của trang chủ của selenium có phải là: "Selenium - Web Browser Automation"

Chúng ta chỉ việc bật trình duyệt và làm việc bình thường như công việc chúng ta vẫn làm ngày hàng bằng tay và kết quả chúng ta có được một cái nhìn toàn diện như sau.



## Phần 3: Thực thi, dừng và gỡ lỗi trên kịch bản selenium



### 3.1 Thực Thi

**IDE** cho phép nhiều tùy chọn để chạy các TestCase của bạn. Bạn có thể chạy một Testcase hoặc toàn bộ Testcase ( gọi là *Test Suite*) cùng một lúc, dừng lại và tiếp tục chạy kịch bản ở một đoạn bất kỳ, chạy một lệnh duy nhất mà bạn hiện đang phát triển. Thực thi các Testcase là rất linh hoạt trong IDE.

#### 3.1.1 Thực thi một Test case



Run: Nhấp vào nút *Run* để chạy thử nghiệm hiện đang được chọn. Khi chỉ có một thử nghiệm duy nhất được nạp vào thì nút này và nút *Run All* có tác dụng tương tự.

#### 3.1.2 Thực thi một Test suite



Run All: Nhấp vào nút *Run All* để chạy tất cả các TestCase đang được mở

### 3.2 Tạm Dừng



Cho phép bạn dừng thử nghiệm tại một đoạn bất kỳ nào bạn muốn.



Sau khi click vào biểu tượng *Pause* nó sẽ chuyển sang biểu tượng *Resume*. Với nút này cho phép bạn có thể tiếp tục chạy thử nghiệm của bạn tại đoạn bạn đã dừng.

### 3.3 Gỡ lỗi

Log	Reference	UI-Element	Rollup	Info	Clear
[info]	Executing:	waitForPageToLoad			
[info]	Executing:	clickAndWait   xpath=id('menu_download')/a			
[info]	Executing:	assertTitle   Downloads			
[info]	Executing:	verifyText   xpath=id('mainContent')/h2   Downloads			

Khi bạn chạy TestCase, Các thông báo lỗi và các thông báo thông tin sẽ được hiện thị một cách tự động trong cửa sổ này, ngay cả khi bạn không chọn tab Log. Các thông báo này thường hữu ích để gỡ lỗi TestCase. Chú ý nút Clear để xóa Log cho mỗi lần chạy lại thử nghiệm.

Gỡ lỗi là việc bạn tìm và sửa lỗi xảy ra trên kịch bản của bạn. Đây là công việc bình thường khi bạn muốn phát triển một thử nghiệm.

#### 3.3.1 Breakpoints and Startpoints / Điểm ngắt và điểm tiếp tục

##### **Breakpoints**

Sel-IDE hỗ trợ thiết lập các điểm ngắt, ngừng các hoạt động của một TestCase, từ bất kỳ điểm nào trong trường hợp thử nghiệm. Đó là, người ta có thể chạy một lệnh cụ thể ở giữa các TestCase và kiểm tra các TestCase hoạt động tại điểm đó. Để làm điều này, thiết lập một lệnh breakpoint ngay trước khi lệnh mà bạn muốn kiểm tra.

Để thiết lập một breakpoint, chọn một lệnh, nhấp chuột phải, và từ trình đơn ngữ cảnh, chọn *Toggle Breakpoint*. Sau đó nhấp vào nút Run để chạy các TestCase của bạn từ đầu đến điểm dừng.

#### *Startpoints*

Việc đặt startpoint hữu ích để chạy một TestCase từ một nơi nào đó ở giữa của các TestCase hoặc đặt nó sau một Breakpoint để tiếp tục từ vị trí đó.

Để thiết lập một StartPoint, chọn một lệnh, nhấp chuột phải, và từ trình đơn ngữ cảnh, chọn *Set/Clear Start Point*. Sau đó nhấp vào nút Run để thực hiện đầu TestCase tại StartPoint đó.

### *3.3.2 Thực thi từng lệnh một*

Để thực hiện một TestCase một lệnh tại một thời gian (“step through” it), hãy làm theo các bước sau:

1. Bắt đầu các TestCase với nút Run từ thanh công cụ.



2. Ngay lập tức tạm dừng các TestCase thực hiện với nút Pause.



3. Nhiều lần chọn nút Bước.

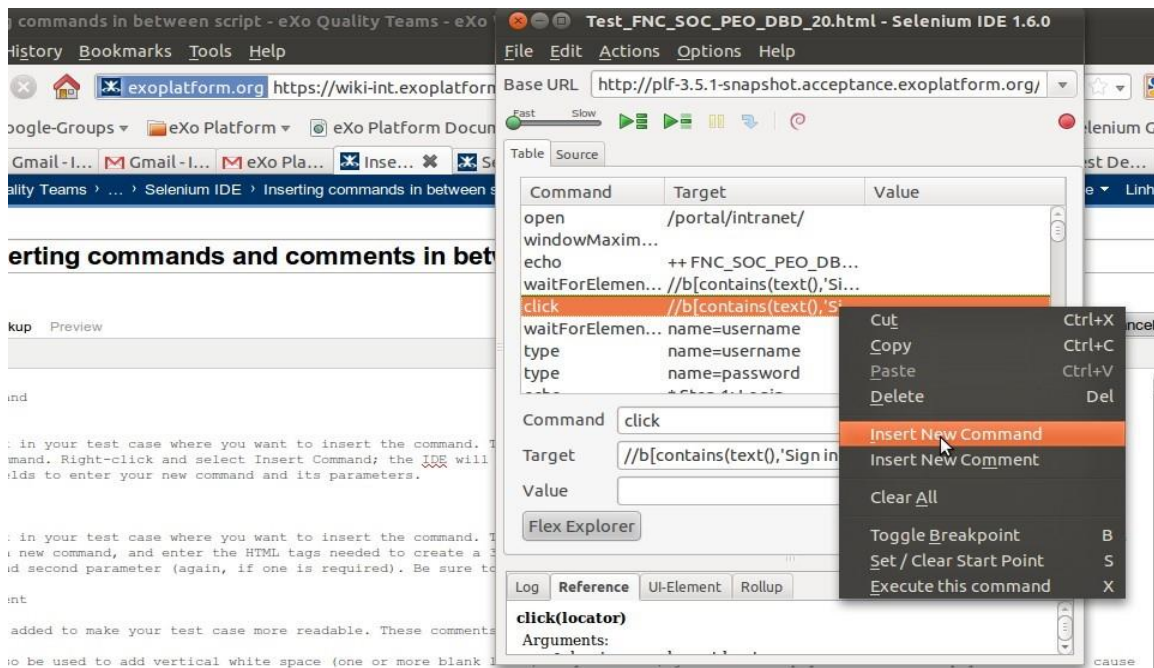


## **Phần 4: Chèn câu lệnh và nhận xét**

### *4.1 Chèn Câu Lệnh*

#### **Chèn ở dạng bảng.**

Chọn điểm trong TestCase của bạn nơi bạn muốn chèn lệnh. Để làm điều này, nhấp chuột trái trên lệnh, nơi bạn muốn chèn một lệnh mới. Nhấn chuột phải và chọn lệnh *Insert*, IDE sẽ thêm một dòng trống trước dòng bạn đã chọn. Bây giờ sử dụng các lệnh chỉnh sửa văn bản để nhập lệnh mới của bạn và các thông số của nó.



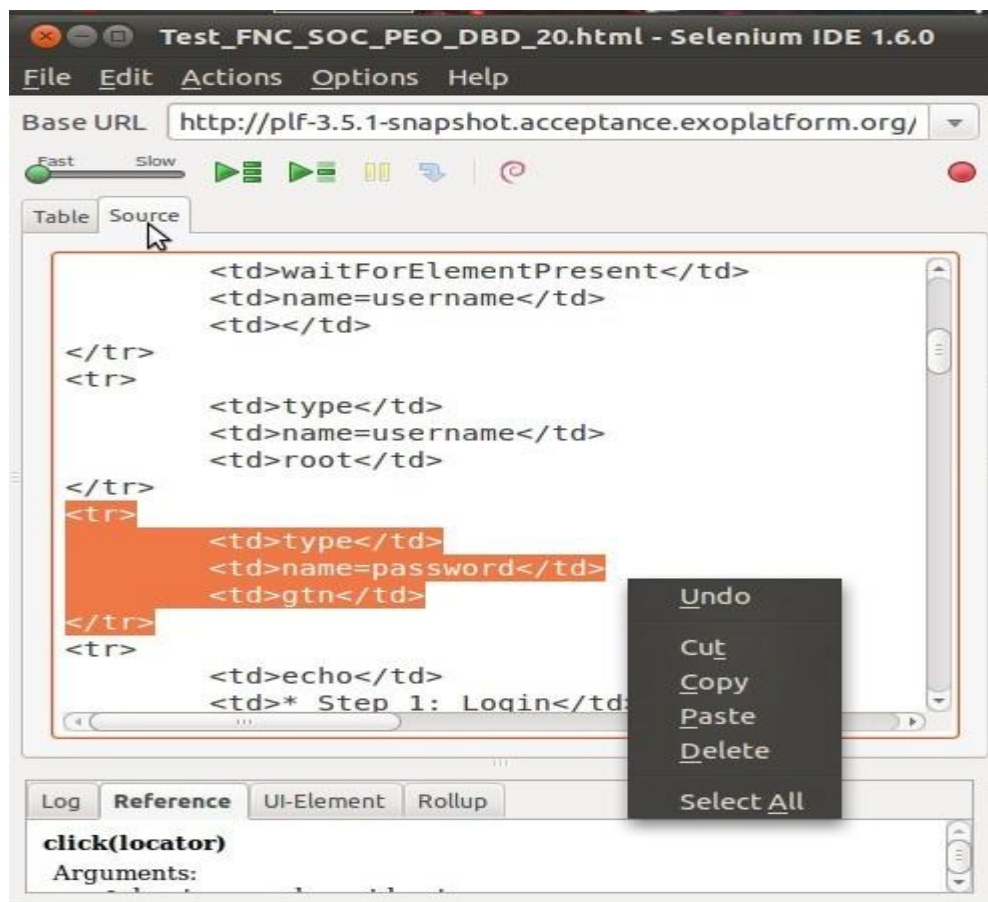
### Chèn ở trong mã nguồn

Chọn điểm trong TestCase của bạn nơi bạn muốn chèn lệnh. Để làm điều này, nhập vào các thẻ HTML cần thiết để tạo ra bảng có chứa 1 dòng và 3 cột.

Cột đầu tiên ứng với tham số command (tên câu lệnh)

Cột thứ hai ứng với tham số Target (vị trí phần tử trên trang web)

Cột thứ 3 là ứng với tham số Value (giá trị)



## 4.2 Chèn nhận xét

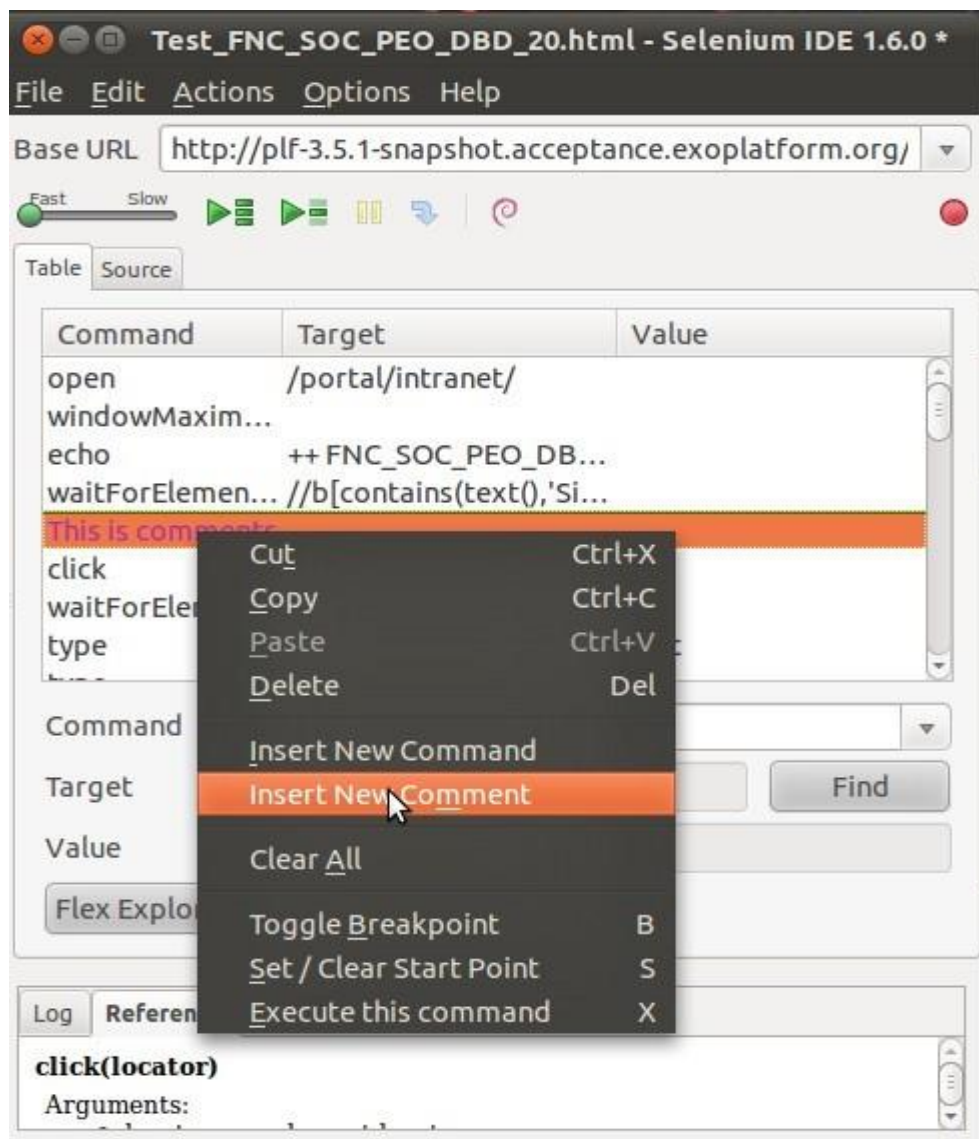
Nhận xét (comment) có thể được thêm vào để làm TestCase của bạn dễ đọc hơn. Những comments được bỏ qua khi các TestCase được chạy.

Khác với khi thêm một command nếu thêm một comment trống thì nó không gây lỗi cho scripts của bạn.

### Chèn dạng bảng

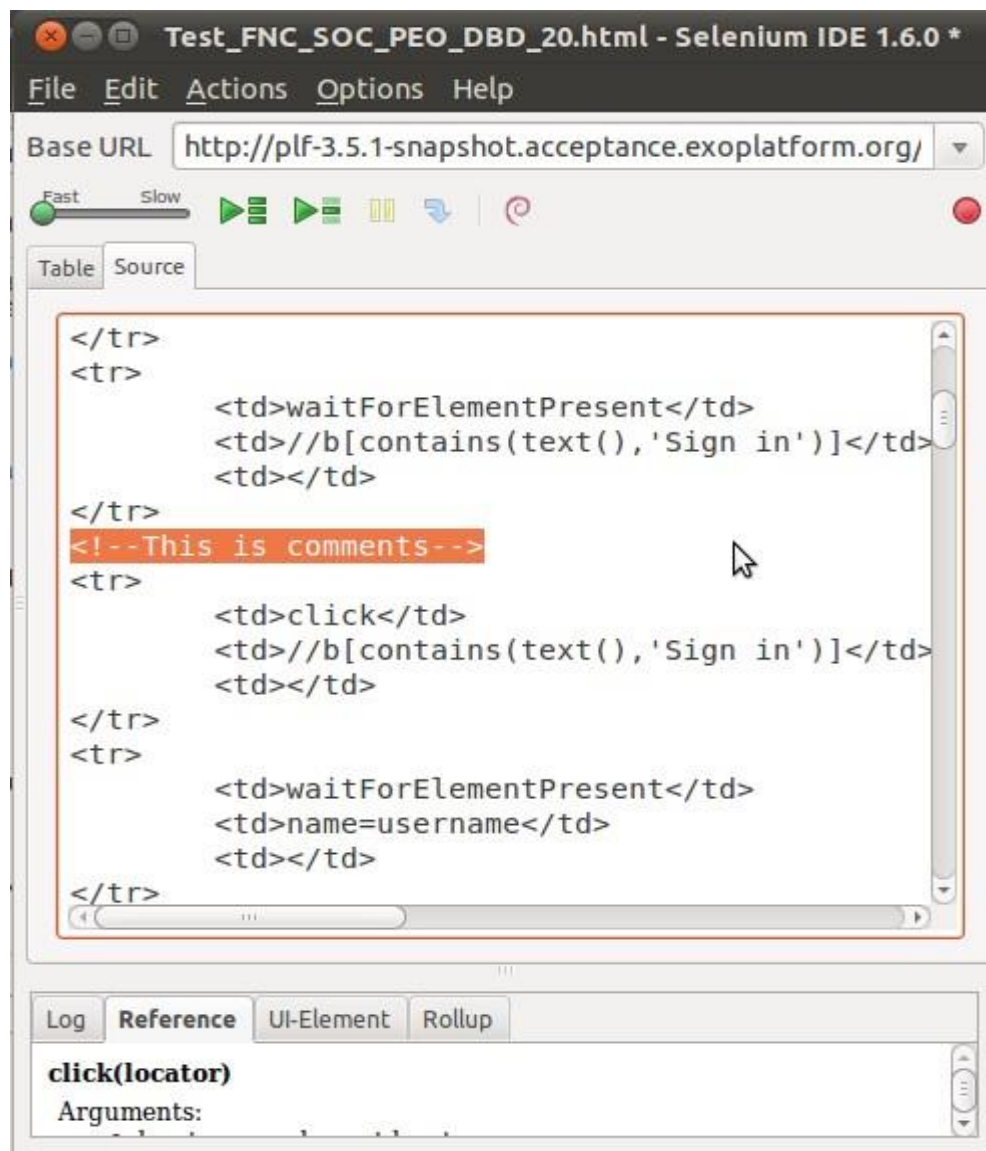
Chọn một dòng trong TestCase của bạn nơi bạn muốn chèn. Nhấp chuột phải vào và chọn *Insert Comment*. Comment của bạn sẽ xuất hiện trong văn bản màu tím.





### Chèn dạng mã nguồn

Chọn điểm trong TestCase của bạn nơi bạn muốn chèn. Thêm một bình luận kiểu HTML, tức là, <- nhận xét của bạn ở đây ->.



### 4.3 Sửa Command/Comment

#### Sửa dạng bảng

Đơn giản chỉ cần chọn dòng được thay đổi và chỉnh sửa nó bằng cách sử dụng Command, Target, và các lĩnh vực giá.

## Sửa dạng mã nguồn

Chỉ cần sửa đổi dòng mà bạn muốn lệnh, tham số, hoặc bình luận giống như bạn chỉnh sửa văn bản trên một trình soạn thảo.

# Phần 5: XPATHS và cài đặt firebug add-on

Xpath là ngôn ngữ truy vấn để lựa chọn nodes và trích chọn thông tin nodes trên các tài liệu HTML/XML.

## 5.1 Các kỹ thuật xác định Xpath.

### 5.1.1 Xác định Xpath theo đường dẫn tuyệt đối

Để xác định một node theo phương pháp này chúng ta sẽ bắt đầu bằng node gốc (root) và kết thúc tại node mà ta mong muốn. Node cần xác định này là hậu duệ, con cháu thứ n của root.

Phương pháp này sử dụng toàn bộ đường dẫn đầy đủ, nó rất hữu ích để lựa chọn một node cụ thể. Từ vị trí gốc chúng ta dùng một dấu gạch chéo lên (/) (forward slash) hoặc hai dấu forward slash (//) phụ thuộc vào cách lựa chọn node.

```
/html/body/div/div[3]/div[3]/form/div/div[4]/div[3]/div/ul/li[2]/a/span
```

```
//html/body/div//form/div/div[4]/li[2]/a/span
```

### 5.1.2 Xác định Xpath theo thuộc tính của node

Để làm điều này bạn phải sử dụng một tiền tố là biểu tượng @ đứng trước một thuộc tính của node.

```
//div[@title='selenium training']
```

```
//a[@href='/selenium/course/']
```

### 5.1.2 Xác định Xpath theo mối quan hệ.

Các biểu thức XPath cũng có thể được tạo ra bằng cách sử dụng đường dẫn vị trí tương đối. Thay vì bắt đầu từ phần tử gốc, bạn chỉ đơn giản là bắt đầu từ node mà bạn muốn và đi từ đó tới vị trí bạn muốn.

```
//div[@id='gt-src-wrap']/div/textarea
```

## 5.2 Những mẫu Xpath hữu ích.

### 5.2.1 Text()

Xác định vị trí của node dựa vào nội dung văn bản của node.

```
//a[text()='Sign In']
```

### 5.2.2 starts-with

Nhiều trang web bạn sẽ thấy những giá trị thuộc tính biến đổi (dynamic value) khiến ta khó khăn trong việc xác định node. Giải pháp đơn giản cho vấn đề này sử dụng từ khóa *starts-with*.

Ví dụ: Một node của bạn có id động

```
<input id="selenium-12345" />
```

Trong đó 12345 là giá trị động sẽ thay đổi. Bạn có thể sử dụng Xpath sau đây

```
//input[starts-with(@id, 'selenium-')]
```

### 5.2.3 sibling

Xác định một node dựa vào mối quan hệ anh chị, em ruột. Dạng này rất hữu ích cho form và tables.

```
//td[text()='Anh']/following-sibling::td[@tite='Em'] //tr[text()='Em']/preceding-sibling::a[@tite='Anh']
```

### 5.2.4 contains

Nếu một node mà giá trị của nó được bao quanh bởi nhiều giá trị bị biến đổi hay những phần giá trị mà chúng ta không quan tâm thì từ khóa **contains** rất hữu ích.

Ví dụ: `<span class="anh yeu em">`

Ta có thể xác định thẻ span này ta có thể dựa trên **yeu** mà không quan tâm tới giá trị **anh** và **em** bằng cách sử dụng mẫu Xpath sau:

```
//span[contains(@class, 'yeu')]
```

Sử dụng cách xác định vị trí node theo CSS sẽ giúp cải thiện hơn tốc độ scripts của bạn

```
css=span.yeu
```

**Note:** Trong tài liệu này chỉ liệt kê ra các trường hợp hữu ích và hay dùng. Để hiểu kỹ hơn về các bạn có thể tham khảo [http://www.w3schools.com/xpath/xpath\\_axes.asp](http://www.w3schools.com/xpath/xpath_axes.asp).

### **5.3 Cài đặt Firebug.**

Cũng giống như selenium, Firebug là một add-on trên trình duyệt FF. Tiện ích này sẽ giúp cho chúng ta có thể sửa, xem và gỡ lỗi CSS, HTML, DOM và Javascripts. Nó hữu ích cho việc giúp chúng ta tìm Xpath.

Sau đây là các bước để cài đặt

Bước 1: Truy cập vào <https://addons.mozilla.org/en-US/firefox/addon/firebug/>

Bước 2: Nhấp vào nút “Add to Firefox”





## Phần 6: Các câu lệnh wait

### 6.1 Câu lệnh có hậu tố “AndWait”.

Những câu lệnh thuộc nhóm Action sẽ luôn có một lệnh có thêm hậu tố AndWait phía sau tương ứng với câu lệnh đó, ví dụ click thì có clickAndWait. Điều này sẽ giúp cho selenium hiểu được là cần chờ một thời gian bằng thời gian timeout để tải lại trang web sau hành động đó.

Những câu lệnh có hậu tố AndWait thường được sử dụng mà lệnh đó là nguyên nhân dẫn đến việc di chuyển sang một trang mới hay phải tải lại trang hiện tại (navigation/refresh).

Nếu như hành động AndWait không gây nên navigation/refresh nhưng bạn cố tình thêm AndWait vào sau lệnh của bạn thì kịch bản của bạn sẽ dừng lại vì lý do timeout.

### 6.2 Câu lệnh có tiền tố “waitFor” *dùng* trong các ứng dụng AJAX

Trong các trang web có sử dụng ứng dụng AJAX thì dữ liệu được lấy từ máy chủ nhưng không làm mới lại trang, nếu bạn dùng “AndWait” nó sẽ không làm việc vì lý do nó không refresh lại trang. Nếu dùng lệnh pause để tạm dừng một thời gian nhất định nào đó cũng không phải là một giải pháp tối ưu vì thời gian hoàn tất công việc của ứng dụng Ajax có thể sớm hoặc muộn hơn khoảng thời gian pause. Các tốt nhất là chờ đợi các yếu tố cần xuất hiện một cách bằng một khoảng thời gian linh động và tiếp tục thực hiện các lệnh sau đó.

Để làm được điều này chúng ta sử dụng các lệnh có tiền tố “waitForXXX” như là “waitForElementPresent” hoặc là “waitForVisible”.....

Lệnh “waitFor” sẽ dành một khoảng thời gian để chờ cho điều kiện XXX trở nên đúng. Nó sẽ ngay lập tức thực hiện các lệnh phía sau nếu như điều kiện trở thành đúng và nếu như thời gian chờ đợi một điều kiện đúng sai ra mà lớn hơn thời gian thiết lập cho timeout kịch bản của bạn sẽ bị dừng lại

## Phần 7: Verification and Assertions

### 7.1 Sự khác nhau giữa verifyXXX và assertXXX.

Lựa chọn giữa “assert” và “verify” rất hữu ích, thuận tiện cho việc quản lý nguyên nhân dẫn tới hành động thất bại (failures).

Chúng ta thường rất khi kiểm tra đoạn đầu của kịch bản, ví dụ ta nên kiểm tra xem sau khi di chuyển hay mở ra một trang mới thì nó có đúng như thế không để tiếp tục thực hiện các hành động sau đó. Nếu như việc kiểm tra này thất bại nghĩa là bạn không ở trên trang chính xác mà bạn mong muốn thì bạn có thể hủy bỏ trường hợp thử nghiệm này của bạn ngay từ đầu. Mặt khác bạn cũng có thể tiếp tục chạy kịch bản của bạn nếu như kết quả mong đợi mà bạn kiểm tra không gây ảnh hưởng đến những bước sau này.

“assert” sẽ làm dừng kịch bản của bạn và hủy bỏ toàn bộ các thao tác sau đó trong khi “verify” cho dù có kết trả lại của nó là fail thì nó vẫn tiếp tục chạy kịch bản của bạn. Bởi thế chúng ta nên sử dụng “assert” để xác minh, kiểm tra kết quả đầu ra mà nó mang tính logic.

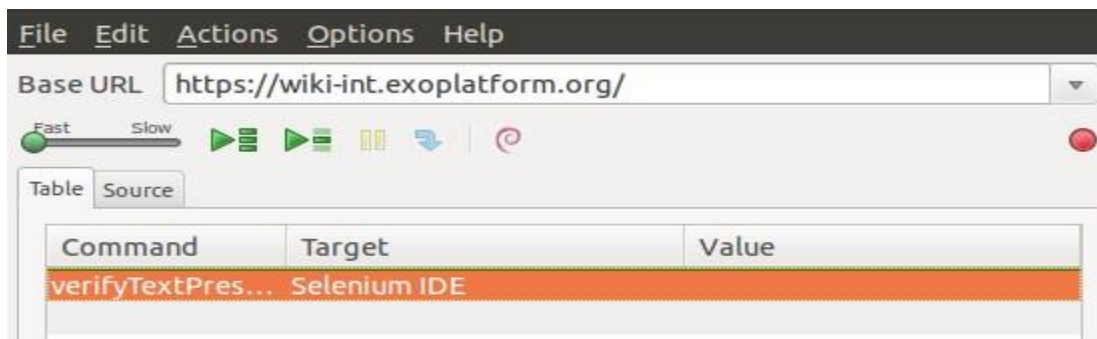
Từ đó chúng ta nên xem xét cho những lệnh assert/verify một cách hợp lý sau mỗi lệnh thuộc nhóm Action.

## 7.2 Một số lệnh verify thường dùng.

### 7.2.1 *verifyTextPresent*

Lệnh VerifyTextPresent được sử dụng để kiểm tra một đoạn văn bản cụ thể tồn tại một nơi nào đó trên trang. Chỉ có duy nhất một đối số duy nhất là mẫu văn bản để được kiểm tra.

Ví dụ:



Với lệnh này nó sẽ tìm kiếm chuỗi văn bản “Selenium IDE” có xuất hiện ở một nơi nào đó trên trang hiện đang thử nghiệm không. Sử dụng hàm này bạn chỉ cần quan tâm nó văn bản cần kiểm tra có xuất hiện không, chứ không kiểm tra nó xuất hiện tại một vị trí nào cụ thể.

### 7.2.2 *verifyElementPresent*

Sử dụng lệnh này bạn sẽ kiểm tra được sự xuất hiện của một phần tử nào đó trên giao diện người dùng hơn là quan tâm đến giá trị hay nội dung của nó. Nó không dùng cho kiểm tra văn

bản, chỉ để kiểm tra việc hiển thị của các HTML tag. Nó thường dùng phổ biến cho việc kiểm tra sự xuất hiện của một hình ảnh.



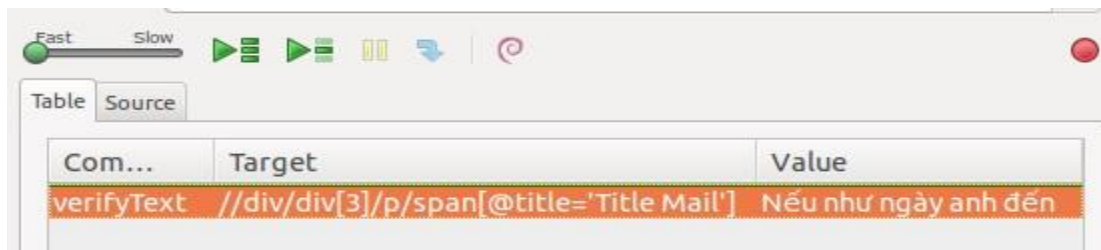
Câu lệnh ở hình minh họa trên sẽ kiểm tra sự hiển thị một hình ảnh được quy định bởi thẻ **img** trong ngôn ngữ HTML, thẻ này được thiết kế theo sau thẻ **div** và một thẻ **p**, tham số trong mục *Target* giúp cho selenium định vị được vị trí của ảnh cần kiểm tra.

Ngoài ra nó còn được sử dụng để kiểm tra sự tồn tại của bất kỳ thẻ HTML nào trong trang bạn đang thử nghiệm như là *link*, các thẻ **div**, thẻ **a** ....

Command	Target	Value
verifyElementPresent	//div/p	
verifyElementPresent	//div/a	
verifyElementPresent	id=Login	
verifyElementPresent	link=Go to Marketing Research	
verifyElementPresent	//a[ 2]	
verifyElementPresent	//head/title	

### 7.2.3 verifyText

Sử dụng câu lệnh này sẽ giúp bạn kiểm tra được cả nội dung văn bản và vị trí UI của văn bản đó. Lệnh này phải sử dụng một biểu thức định vị vị trí (locator). Bạn dùng Xpath hoặc DOM, CSS để xác định locator và kiểm tra được văn bản cụ thể tại một vị trí cụ thể trên trang mà bạn đang thử nghiệm.



### 7.3 Nên dùng “verify” hay “assert”.

Khi nào bạn nên sử dụng lệnh verify và khi nào bạn nên sử dụng lệnh assert? Câu trả lời là nó phụ thuộc vào bạn, phụ thuộc vào văn cảnh cũng như mà các tình huống mà bạn gặp phải. Sự khác biệt duy nhất ở đây là điều bạn muốn (kết quả mong đợi) khi kiểm tra, bạn có muốn thử nghiệm của bạn chấm dứt hay tiếp tục chạy hoặc có thể đơn giản là đánh dấu những chỗ bạn đã kiểm tra không thành công trước đó.

Nếu bạn sử dụng lệnh assert, kịch bản của bạn sẽ dừng lại ngay lập tức và không chạy bất kỳ thêm một lệnh nào khác và khi đó bạn có thể kiểm tra trực tiếp kịch bản của bạn nơi mà nó không thực hiện thành công việc assert. Một số công cụ như TestNG hay JUnit có những chức năng bổ sung giúp bạn có thể đánh dấu lại nhưng đoạn thử nghiệm không thành công này. Tuy nhiên có một nhược điểm là sẽ có nhiều chức năng không được kiểm tra sau và ta không biết được thông tin nào về những chức năng ấy cho tới khi vấn đề gặp phải được giải quyết để kịch bản của bạn chạy qua được lệnh assert đã gây nên việc chấm dứt kịch bản này.

Ngược lại khi bạn sử dụng verify nó sẽ không chấm dứt kịch bản của bạn và vẫn tiếp tục chạy tiếp. Nếu như thử nghiệm của bạn chỉ sử dụng những lệnh verify thì bản thử nghiệm của bạn sẽ được báo cáo trả về là chạy thành công. Thế là bạn sẽ mất thời gian nhiều hơn cho việc ngồi xem báo cáo được phản hồi về từ TestNG hay Junit, số lượng thử nghiệm của bạn lên tới con số khổng lồ thì bạn nghĩ xem bạn sẽ phải vật lộn với bảng báo cáo như thế nào?

Kết luận là ta nên dùng assert vì nó có thể phản hồi ngay lập tức những thông tin gây ra cái chết cho kịch bản của bạn ^.^

## Phần 8: Javascripts và selenium

JS thường được sử dụng trong selenium IDE dưới hai dạng scripts và Non- Scripts (thường là biểu thức).



## 8.1 Cách sử dụng Javascripts dạng scripts.

Một vài câu lệnh trong selenium, tham số đầu vào của lệnh chỉ định một đoạn kịch bản JS như là *assertEval*, *verifyEval*, *storeEval*, và *waitForEval*. Tham số này không cần cú pháp gì đặc biệt. Người dùng chỉ việc đặt một đoạn mã JS vào phần *Target* (bởi vì cú pháp trong câu lệnh của selenium thường lấy Target là tham số đầu tiên)

Tất cả các biến được tạo ra trong trường hợp thử nghiệm của bạn được lưu trữ trong JavaScript là một mảng kết hợp. Nhưng mảng này chỉ có chuỗi ký tự, không sử dụng chỉ mục. Bất cứ khi nào bạn muốn truy cập hay thao tác với một biến **trong một đoạn mã JavaScript**, bạn phải đề cập đến nó như *storedVars* [*tên biến của bạn*].

Ví dụ dưới đây minh họa cách một đoạn mã JavaScript có thể được sử dụng để thực hiện một số phép tính đơn giản:

Command	Target	Value
store	10	hits
storeXPathCount	//blockquote blockquotes	
storeEval	storedVars['hits']-storedVars['blockquotes']	paragraphs

Ví dụ tiếp theo minh họa cách một đoạn mã JavaScript có thể biến đổi dữ liệu văn bản của bạn từ chữ *HOA* thành chuỗi văn bản chữ *thường*, trong trường hợp này ta sẽ phải sử dụng 2 phương thức *toUpperCase method* và *toLowerCase* của đối tượng *String*.

Command	Target	Value
store	Edith Wharton	name
storeEval	storedVars['name'].toUpperCase()	uc
storeEval	storedVars['name'].toLowerCase()	lc

## 8.2 Cách sử dụng Javascripts dạng Non-scripts.


JavaScript cũng có thể được sử dụng để giúp tạo ra giá trị cho các tham số trong lệnh của selenium. Tuy nhiên trong trường hợp này cú pháp sẽ đặc biệt hơn chút. Tất cả các đoạn mã JS sẽ được đặt trong dấu { và } và phía trước có nhãn *javascript*.

*javascript{mã js của bạn ở đây}*

Dưới đây là một ví dụ mà giá trị của thông số thứ hai của lệnh selenium được tạo ra bởi mã js bằng cách sử dụng cú pháp đặc biệt này

Command	Target	Value
store	league of nations	searchString
type	q	javascript {storedVars[ 'searchString'].toUpperCase()}
echo	The Selenese Print Command	

Selenium có một lệnh đơn giản cho phép bạn in ra giá trị đầu ra của js đó là lệnh **echo**. Nó rất hữu ích cho việc cung cấp thông tin trong quá trình phát triển kịch bản của bạn. Nó sẽ in giá trị này trong của sổ Log trong quá trình chạy. Nó sẽ giúp ta tìm ra được các vấn đề tồn tại ẩn trong kịch bản như sai sót về thuật toán hoặc sự nhầm lẫn trong việc tính toán dựa vào công thức trong mã js.

Store	javascript{var d=new Date(); d.toString();}	dateToday
Store	javascript{var d=new Date(); d.setDate(d.getDate() + 1); d.toString();}	dateTomorrow
Store	javascript{var d=new Date(); d.setDate(d.getDate() + 40); d.toString();}	date40days
Store	Plain text string	variableName 

Echo	javascript{storedVars['dateTomorrow']}	
Type	textField	javascript{'Tomorrow=' + storedVars['dateTomorrow']}
Echo	javascript{'Tomorrow=' + storedVars['dateTomorrow']}	
Type	fullName	javascript{storedVars['fullName'].toUpperCase()}

## Phần 9: Biểu thức chính quy trong Selenium

Nhiều câu lệnh selenium sử dụng cái biểu mẫu chính quy (regular expressions) trong các tham số. Nó cho phép bạn có thể kiểm tra tính đúng đắn của định dạng dữ liệu. Có 3 loại patterns mà bạn có thể sử dụng trong kịch bản của bạn: globs, exact and [regular expressions](#).

## 9.1 glob:patter

Glob trong selenium không được phong phú như trong Linux ( shell scripts). Nó chỉ hỗ trợ 3 ký tự đặc biệt sau: \*, ? và [ ].

*\* : Phù hợp với mọi số lượng ký tự.*

*? : Phù hợp với một ký tự đơn lẻ.*

*[ ] : gọi là một lớp ký tự. Cho phép bạn tạo match với bất kỳ ký tự nào bên trong.*

Ví dụ:

[0-9]: matches mọi chữ số.

[a-zA-Z]: matches với mọi ký tự trong bảng chữ cái

Để xác định một glob trong một lệnh selen, tiền tố mẫu glob: string. Ví dụ nếu bạn muốn tìm kiếm cho văn bản màu hoặc màu sắc bạn có thể sử dụng Colo \* r glob như hình dưới đây.

Command	Target	Value
clickAndWait	link=search	
verifyTextPresent	glob: colo*r	

Tuy nhiên bạn có thể bỏ tiền tố glob: trong selenium vì kiểu patterns này là mặc định.

## 9.2 regexp:regexp và regexpi:regexpi

Match một chuỗi ký tự sử dụng biểu thức chính quy (regular-expression) bạn có thể sử dụng được toàn bộ tính năng của biểu thức chính quy trong JavaScript.

Trong 3 kiểu Patterns thì regular-expression thường được sử dụng phổ biến hơn cả. Selenium hỗ trợ tập hợp đầy đủ các mẫu regex như Javascript hỗ trợ, chính vì thế bạn sẽ không còn bị

hạn chế bởi \*,? và [] như với kiểu glob. Để sử dụng các biểu mẫu RegEx bạn cần thêm tiền tố *regexp:* hoặc *regexpi:*

Ví dụ sau đây sẽ kiểm tra dữ liệu nhập vào với “id” chứa chuỗi ‘javascript’, ‘JavaScript’ hoặc ‘Javascript’.

Command	Target	Value
clickAndWait	link=search	
verifyValue	id=name	regexp:[Jj]ava([Ss]cript)

Dưới đây là một vài RegEx patterns phổ biến:

1. `regexp:(0[1-9]|1[012])[- /.](0[1-9]|12)[0-9]3[01])[- /.](19|20)\d\d` Phù hợp với định dạng ‘mm/dd/yyyy’ được phân cách bởi ‘-’, ‘/’, ‘.’
2. `regexpi:^[A-Z0-9+_.-]+@[A-Z0-9.-]+$` Phù hợp với định dạng email
3. `regexp:^[0-9]{5}(\?:-[0-9]{4})?$`  
Phù hợp với định dạng mã ZIP (U.S. postal code).
4. `regexp:^(https?|ftp|file)://.+`  
Phù hợp với định dạng URL

### 9.3 exact:string

Phù hợp với một chuỗi chính xác, đúng nguyên văn, mà không có bất kỳ ký tự đại diện nào.

**Note:** Một số trang web hữu ích để học RegEx

<http://www.regular-expressions.info/javascriptexample.html> <http://www.regular-expressions.info/reference.html>  
<http://www.zytrax.com/tech/web/regex.htm>  
[http://gnosis.cx/publish/programming/regular\\_expressions.html](http://gnosis.cx/publish/programming/regular_expressions.html)

## Phần 10: Xử lý JavaScript Alerts

Selenium IDE không hỗ trợ ghi trên cửa sổ pop-up bởi vì khi một cửa sổ mới được mở ra nó sẽ không nằm trong thẻ body của cửa sổ chính nơi bạn đang làm việc. Selenium IDE, Dom và những plugin khác của Firefox cũng không làm việc luôn. Do đó bạn phải thao tác bằng tay để ghi lại những sự kiện của bạn trên cửa sổ pop-up này.

Có nhiều phương pháp khác nhau để giải quyết với các loại cửa sổ pop up khác nhau. Sau đây là một số phương pháp mà tôi đã sử dụng để xử lý JavaScript Alerts.

```
selenium.chooseCancelOnNextConfirmation()  
selenium.chooseOkOnNextConfirmation()
```

Hai lệnh này sẽ được dùng khi hành động của bạn sẽ tạo nên một cảnh báo. Ví dụ khi bạn xóa một đối tượng nào đó sẽ có một cảnh báo bằng js hiện lên với mục đích giúp bạn cân nhắc việc có chắc chắn xóa nó không. Cửa sổ này sẽ có hai lựa chọn là “OK” và “Cancel”.

Với trường hợp bạn đồng ý xóa thì lệnh `selenium.chooseOkOnNextConfirmation()` được cài là mặc định. Ngược lại khi bạn muốn thoát khỏi việc xóa đối tượng này. Bạn sẽ phải bấm vào nút Cancel tùy nhiên Selenium sẽ không hỗ trợ việc này mà bạn phải tự điền lệnh `selenium.chooseCancelOnNextConfirmation()` trước lệnh xóa.

```
selenium.chooseCancelOnNextConfirmation()  
selenium.click(buttonLocator)
```

## Phần 11: Sử dụng User-Extensions.js

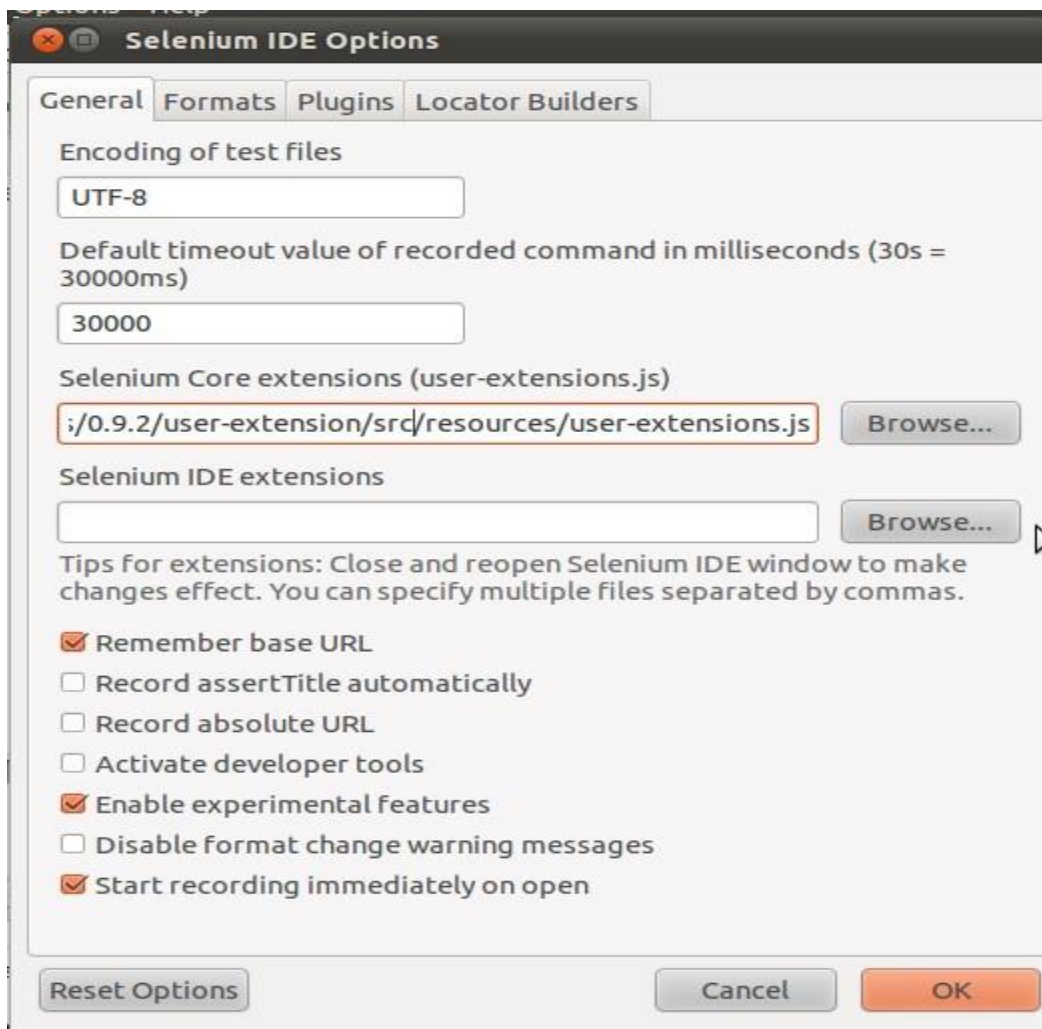
Để mở rộng hơn các hàm mới được sử dụng trong selenium IDE. Bạn sẽ tự định nghĩa những hàm/ chức năng này bằng ngôn ngữ js.

### 11.1 Cách tạo và sử dụng tệp tin user-extensions.js

Bạn có thể dùng một trình soạn thảo bất kỳ như notepad để mở một tệp tin mới và lưu nó lại dưới dạng mở rộng là .js và tên của tệp tin này chúng ta nên đặt là user-extensions (không bắt buộc).

Để sử dụng được nó bạn mở selenium IDE từ menu bar chọn Options/Options... và cài đặt theo hình dưới đây.





## 11.2 Cách định nghĩa một hàm mới trong user-extensions.js

Dưới đây là một ví dụ

Bạn mở tệp tin user-extensions.js mà bạn vừa tạo trước đó và viết một đoạn mã js như sau

```
Selenium.prototype.doHelloWorld = function() {  
    alert("Hello world !");  
}
```

Sau đó bạn nhớ lưu lại tệp tin này và khởi động lại selenium của bạn. Kết quả bạn sẽ có một hàm mới được liệt kê trong list các lệnh của selenium IDE với tên gọi là helloworld. Khi bạn chạy lệnh này mà nó bật lên một pop-up với nội dung là Hello world ! thì xin chúc

mừng bạn đã thành công trong việc tạo một lệnh mới có thể dùng được trong selenium IDE

### 11.3 Định nghĩa hàm typeRandomEmail

```
selenium.prototype.doTypeRandomEmail = function randomString(locator,string_length) {  
  
    var element = this.page().findElement(locator);  
  
    var chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";  
    var randomstring = "";  
    for (var i=0; i<string_length; i++) {  
        var rnum = Math.floor(Math.random() * chars.length);  
        randomstring += chars.substring(rnum,rnum+1);  
    }  
    //return randomstring;  
    var valueToType =randomstring+"@exoplatform.com";  
  
    this.page().replaceText(element, valueToType);  
}
```

### 11.4 Sử dụng vòng lặp và câu lệnh rẽ nhánh trong selenium

Bạn có thể dùng vòng lặp cũng như câu lệnh rẽ nhánh trong selenium IDE. Đây là điều phức tạp với những bạn không có kiến thức về lập trình

**Note:** Một số trang web hữu ích cho bạn tìm hiểu về vấn đề này

[http://www.theautomatedtester.co.uk/tutorials/selenium/selenium\\_extension.htm](http://www.theautomatedtester.co.uk/tutorials/selenium/selenium_extension.htm)

<http://code.google.com/p/selenium->

[fasttrack/downloads/detail?name=userextensions.js&can=2&q=](http://code.google.com/p/selenium-fasttrack/downloads/detail?name=userextensions.js&can=2&q=)

[http://www.qualitytrainings.com/How do you set user extensions in Selenium IDE](http://www.qualitytrainings.com/How_do_you_set_user_extensions_in_Selenium_IDE)