Test Automation & Advanced Selenium

Lesson 3: Working With Selenium IDE

Lesson Objectives

- Selenium IDE – An Introduction
- Installation of Selenium IDE
- Components of Selenium IDE
- Introduction to Selenium Commands – “Selenese”
- Understanding Element Locators in Selenium IDE
  - ID
  - Name
  - Link Text
  - CSS Selector
    - Tag and ID
    - Tag and class
    - Tag and attribute
    - Tag, class, and attribute
    - Inner text
  - DOM (Document Object Model)
    - getElementById
    - getElementsByName
    - dom:name
    - dom:index
  - Working with Alerts
  - Creating Test Script using Selenium IDE
  - Creating & Executing Test Suits
  - Exporting scripts to multiple languages and Formats

Copyright © Capgemini 2015. All Rights Reserved

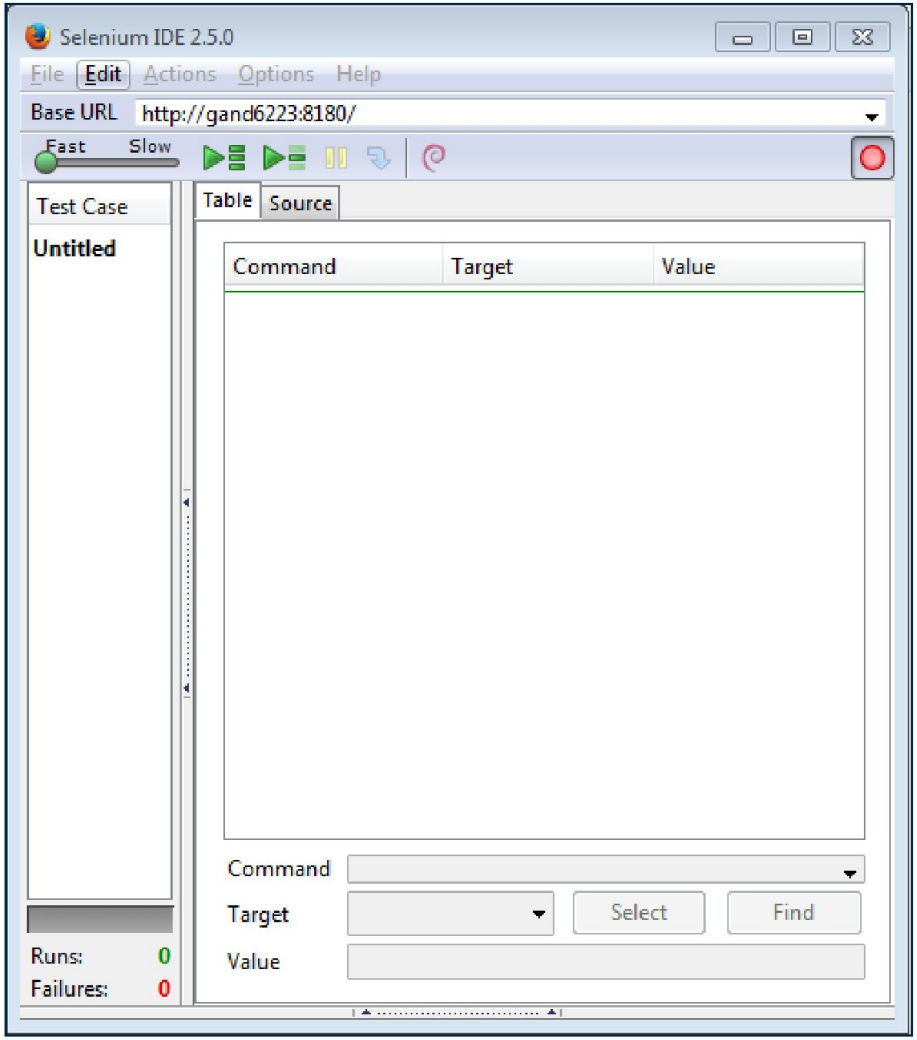
Selenium IDE – An Introduction

- Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE).
- Selenium IDE, fully-featured Integrated Development Environment (IDE)
- Installs as a plugin in Mozilla Firefox
- Enables developers to test their web applications through Selenium
- Records user interactions with the web browser and play back to test for errors
- Effortless to install and easy to learn

Copyright © Capgemini 2015. All Rights Reserved

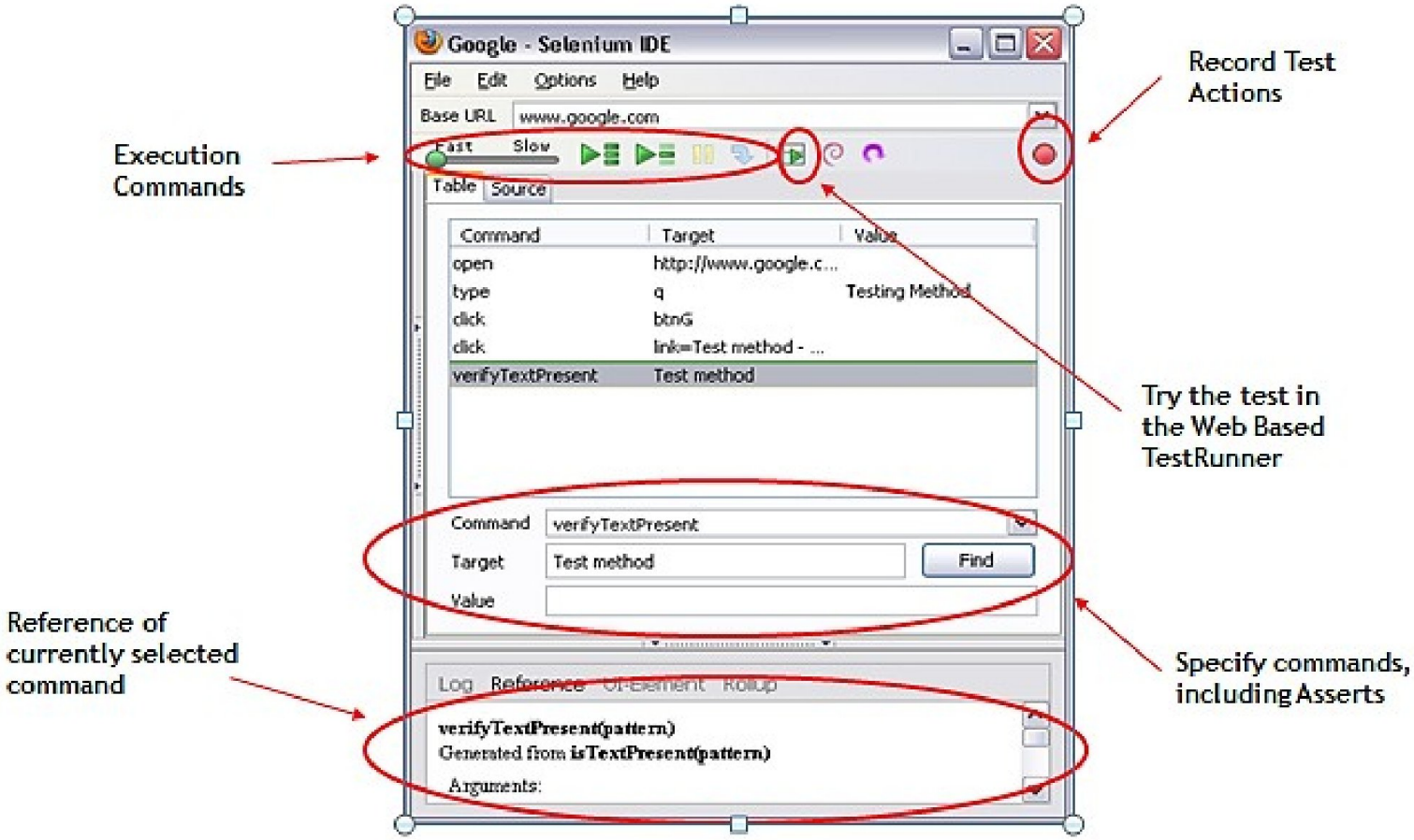
Installation of Selenium IDE

- Download Selenium IDE
- Copy Selenium IDE file to extensions folder of Mozilla Firefox.
- To facilitate opening of .xpi file for selenium you can drag and drop the .xpi file on the Mozilla browser to install it and get it in the add-ons of the browser



Copyright © Capgemini 2015. All Rights Reserved

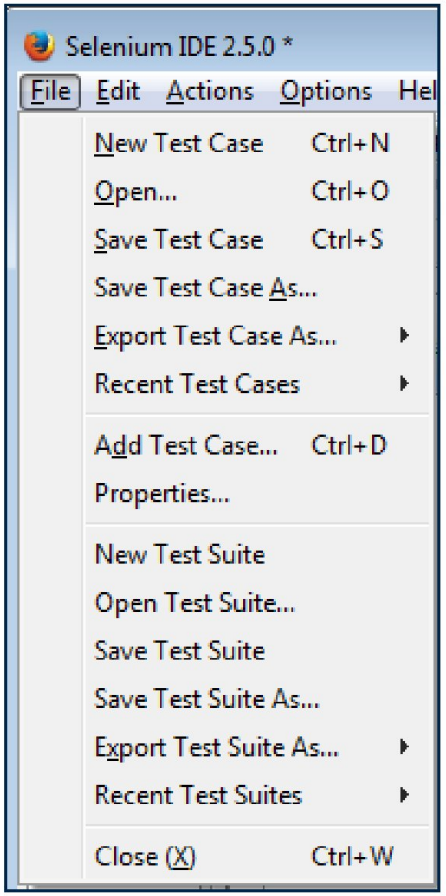
Components of Selenium IDE



Copyright © Capgemini 2015. All Rights Reserved

Components of Selenium IDE (Cont.)

- Menu Bar
  - Positioned at the upper most of the Selenium IDE window
- File Menu
  - Allows user to:
  - Create new test case, open existing test case, save the current test case
  - Export Test Case As option exports and converts only the currently opened Selenium IDE test case
  - Export Test Case As and Export Test Suite As in any of the associated programming language compatible with Selenium RC and WebDriver
  - Export Test Suite As option exports and converts all the test cases associated with the currently opened IDE test suite
  - Close the test case



Copyright © Capgemini 2015. All Rights Reserved

Components of Selenium IDE (Cont.)

Tool Bar

- Contains buttons for controlling the execution of your test cases

- Step feature for debugging your test cases
-

Speed Control: Controls how fast your test case runs
- Run All: Runs the entire test suite when a test suite with multiple test cases is loaded
- Run: Runs the currently selected test. When only a single test is loaded this button and the Run All button have the same effect
- Pause/Resume: Allows stopping and re-starting of a running test case

Copyright © Capgemini 2015. All Rights Reserved

Components of Selenium IDE (Cont.)

Tool Bar

- Step: Allows you to “step” through a test case by running it one command at a time. Use for debugging test cases

- Apply Rollup Rules: This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action. Detailed documentation on rollup rules can be found in the UI-Element Documentation on the Help menu

- Record: Records the user’s browser actions.

Copyright © Capgemini 2015. All Rights Reserved

Components of Selenium IDE (Cont.)

- Test Case Pane
- Script is displayed in the test case pane
- It has two tabs:
  - Displays the command and their parameters in a readable “Table” format
  - “Source” displays the test case in the native format in which the file will be stored
    - By default, this is HTML although it can be changed to a programming language such as Java or C#, or a scripting language like Python

Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

Copyright © Capgemini 2015. All Rights Reserved

Components of Selenium IDE (Cont.)

Command, Target and Value

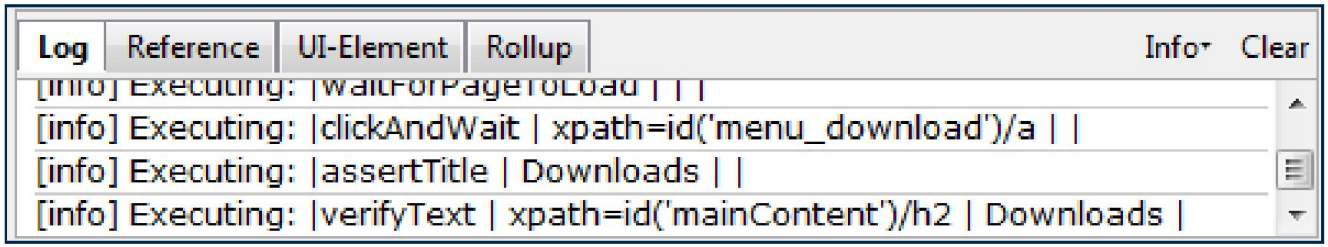
- Command displays the currently selected command along with its parameters
- Target displays unique tag value for the selected field
- Value contains the data in case of text box fields

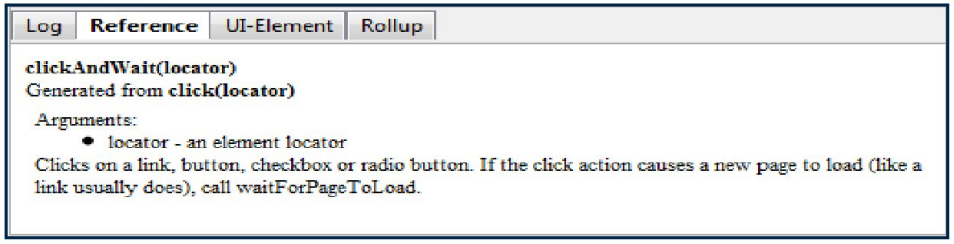
Copyright © Capgemini 2015. All Rights Reserved

Components of Selenium IDE (Cont.)

Log/Reference/UI-Element/Rollup Pane

- Bottom pane is used
- Log: Error messages and information messages showing the progress are displayed automatically
- Reference : Displays documentation on the current command in table mode
- UI-Element and Rollup: Detailed information on these two panes (which cover advanced features) can be found in the UI-Element Documentation on the Help menu of Selenium-IDE





Copyright © Capgemini 2015. All Rights Reserved

Introduction to Selenium Commands – "Selenese"

- Set of commands used by selenium to automate the web application testing
- Combination of Selenese commands creates test script

3A's

Selenese is a combination of as 3A's:

- Action: Used to change the state of the AUT(Application under Test)

  E.g.: Click , close , doubleclick
- Accessor: Check the state of application & save state in some variable

  E.g.: storeTitle , storeElementPresent
- Assertions: Verifies the state of the application matches it with the expected state and generates the True/False result

  E.g. : verifyText, assertText

Copyright © Capgemini 2015. All Rights Reserved

Understanding Element Locators in Selenium IDE

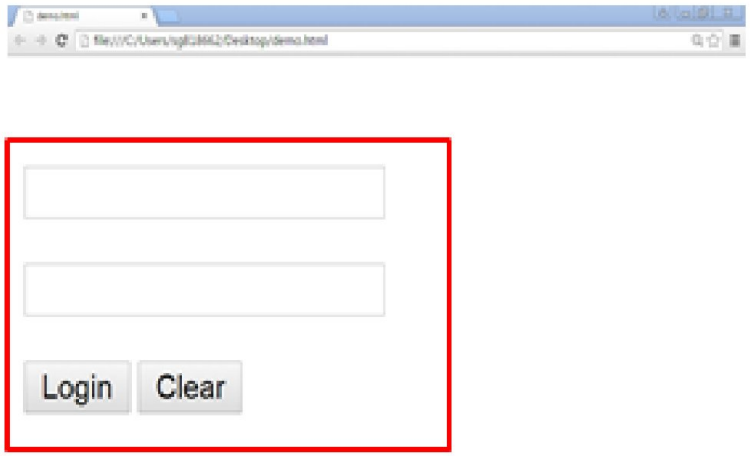
- Types of Element Locators
- ID
- Name
- Link Text
- CSS Selector
  - Tag and ID
  - Tag and class
  - Tag and attribute
  - Tag, class, and attribute
  - Inner text
- DOM (Document Object Model)
  - getElementById
  - getElementsByName
  - dom:name
  - dom: index
- XPath

Copyright © Capgemini 2015. All Rights Reserved

Locating by ID

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

Id = loginForm



Copyright © Capgemini 2015. All Rights Reserved

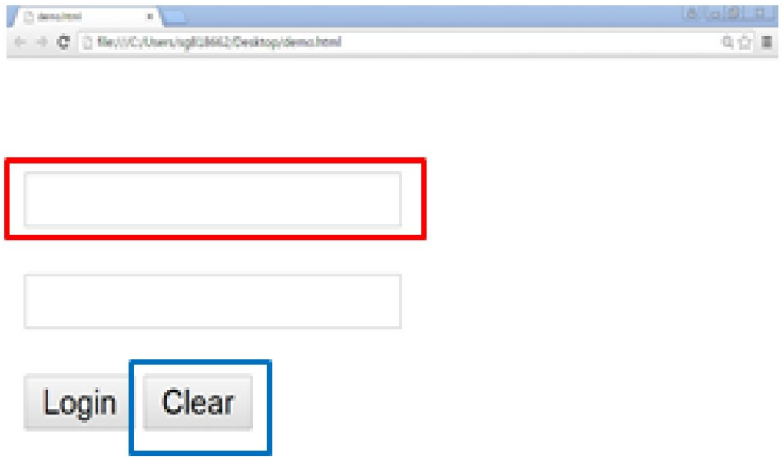
Locating by NAME

name=username

name=continue value=Clear

name=continue Clear

name=continue type=button




```
1 <html>
2 <body>
3 <form id= "loginForm" >
4 <input name= "username" type= "text" />
5 <input name= "password" type= "password" />
6 <input name= "continue" type= "submit" value= "Login" />
7 <input name= "continue" type= "button" value= "Clear" />
8 </form>
9 </body>
10 <html>
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by Link Text

link=Link1

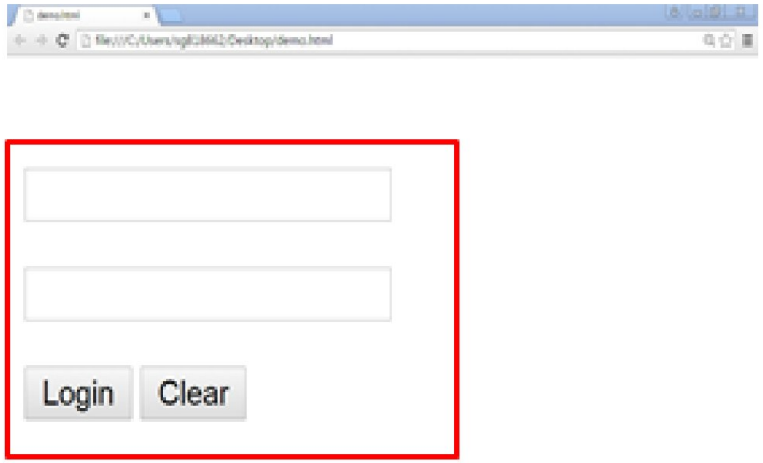
link=Link3

```
1 <html>
2 <body>
3 <a href="#">Link1</a>
4 <a href="#">Link2</a>
5 <a href="#">Link3</a>
6 <form id= "loginForm" >
7 <input name= "username" type= "text" />
8 <input name= "password" type= "password" />
9 <input name= "continue" type= "submit" value= "Login" />
10 <input name= "continue" type= "button" value= "Clear" />
11 </form>
12 </body>
13 <html>
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by CSS Selector - Tag and ID

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

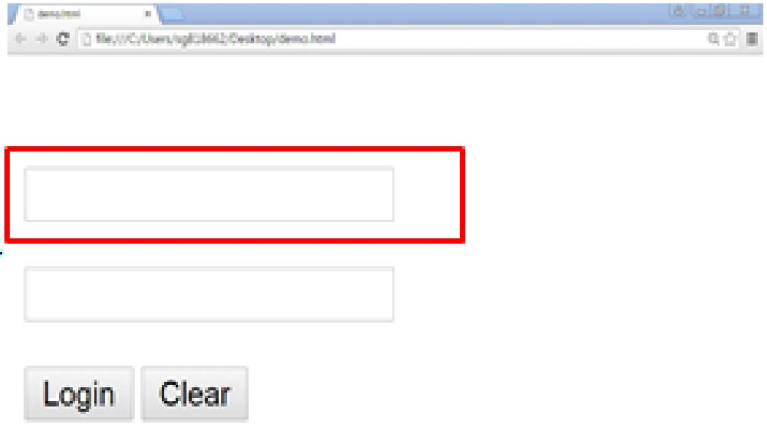


css = form# loginForm

Copyright © Capgemini 2015. All Rights Reserved

Locating by CSS Selector - Tag and Class

```
<html>
<body>
<form id= "loginForm"  >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

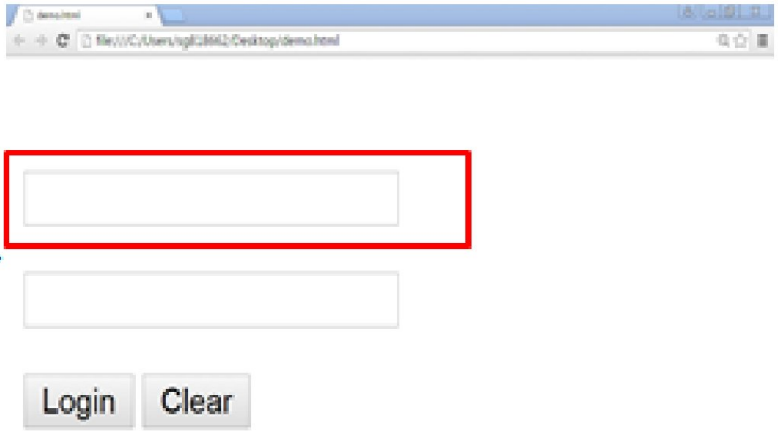



```
css = input.inputtext
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by CSS Selector - Tag and Attribute

```
<html>
<body>
<form id= "loginForm"  >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

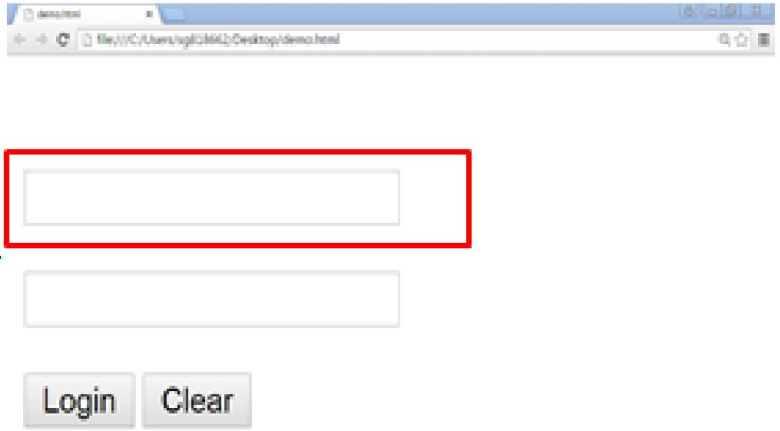



```
css = input[name=username]
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by CSS Selector – Tag, Class and Attribute

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



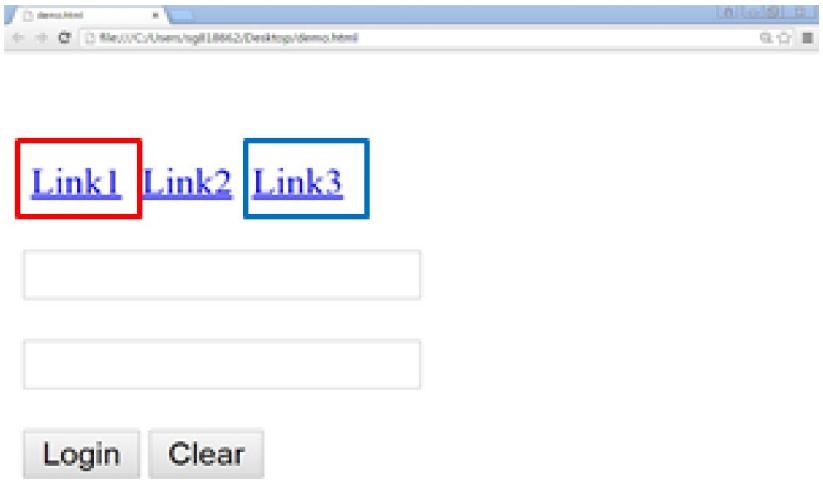

```
css = input.inputtext[name=username]
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by CSS Selector – Inner text

css=a:contains("Link1")

css=a:contains("Link3")

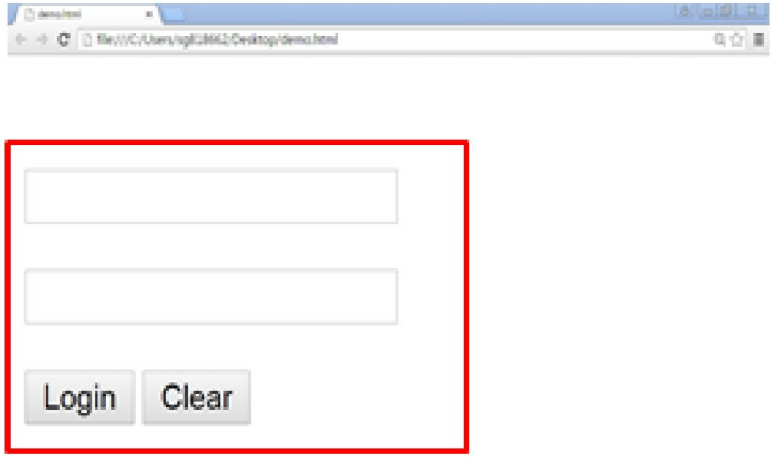



```
1 <html>
2 <body>
3 <a href="#">Link1</a>
4 <a href="#">Link2</a>
5 <a href="#">Link3</a>
6 <form id= "loginForm" >
7 <input name= "username" type= "text" />
8 <input name= "password" type= "password" />
9 <input name= "continue" type= "submit" value= "Login" />
10 <input name= "continue" type= "button" value= "Clear" />
11 </form>
12 </body>
13 <html>
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by DOM – getElementsById

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

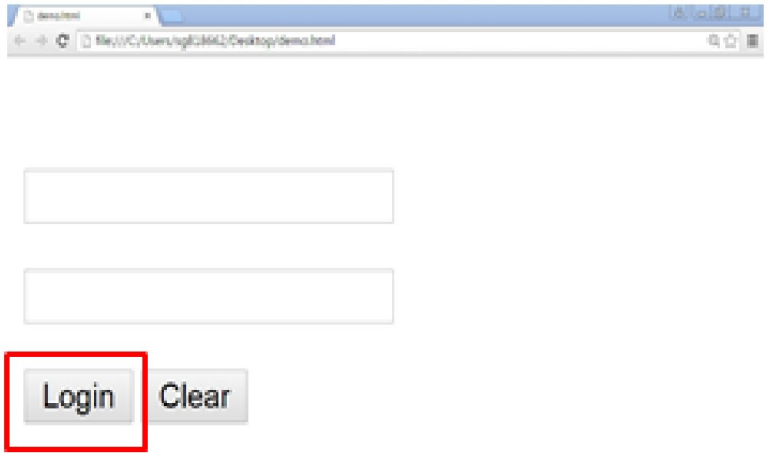



```
document.getElementById(" loginForm ")
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by DOM – getElementsByName

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

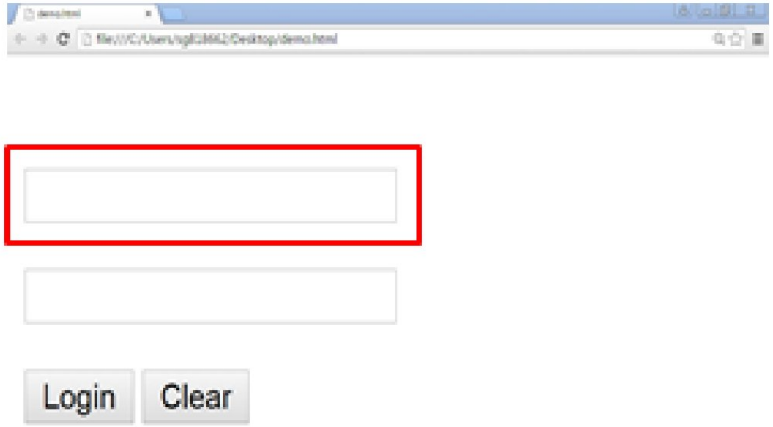



```
document.getElementsByName(" continue ")[1]
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by DOM – DOM:Name

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

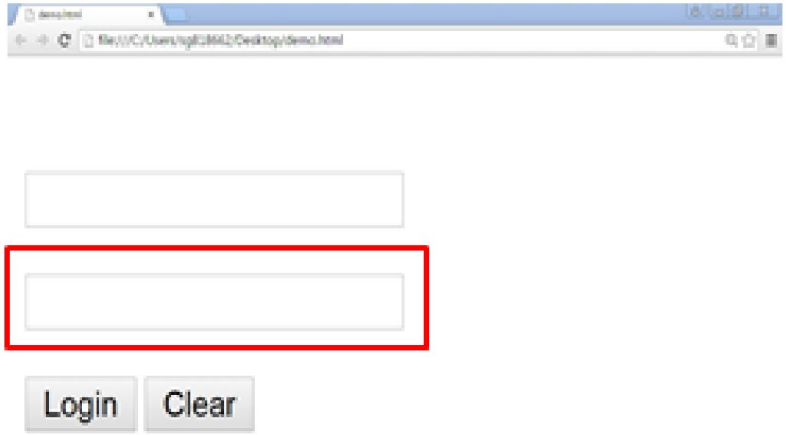



```
document.forms[" loginForm "].elements[" username "]
```


Copyright © Capgemini 2015. All Rights Reserved

Locating by DOM – DOM:Index

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

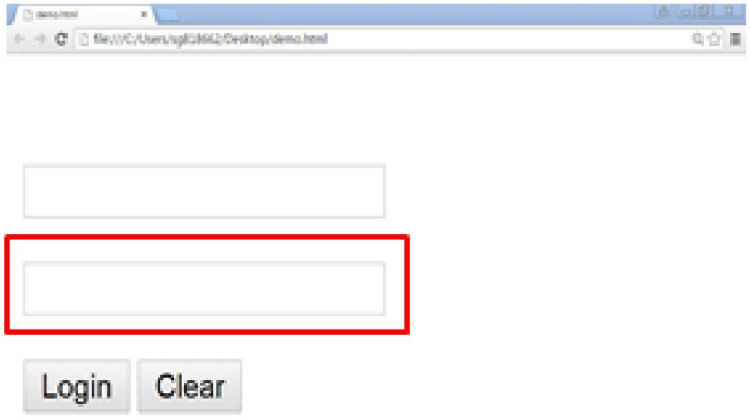


document.forms[1].elements[2]

Copyright © Capgemini 2015. All Rights Reserved

Locating by XPath

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



//*[@id="loginForm"]/input[2]

//html/body/form/input[2]

Copyright © Capgemini 2015. All Rights Reserved

Working With Alerts

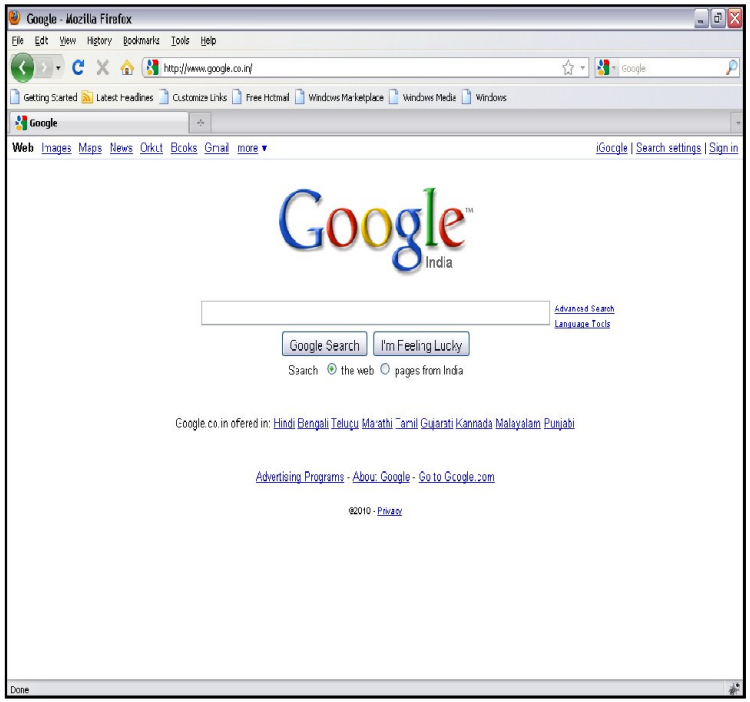
- Simplest form of pop-up windows
- Most common Selenium IDE commands used in handling alerts are the following :

assertAlert assertNotAlert	retrieves the message of the alert and asserts it to a string value that you specified
assertAlertPresent assertAlertNotPresent	asserts if an Alert is present or not
storeAlert	retrieves the alert message and stores it in a variable that you will specify
storeAlertPresent	returns TRUE if an alert is present; FALSE if otherwise
verifyAlert verifyNotAlert	retrieves the message of the alert and verifies if it is equal to the string value that you specified
verifyAlertPresent verifyAlertNotPresent	verifies if an Alert is present or not

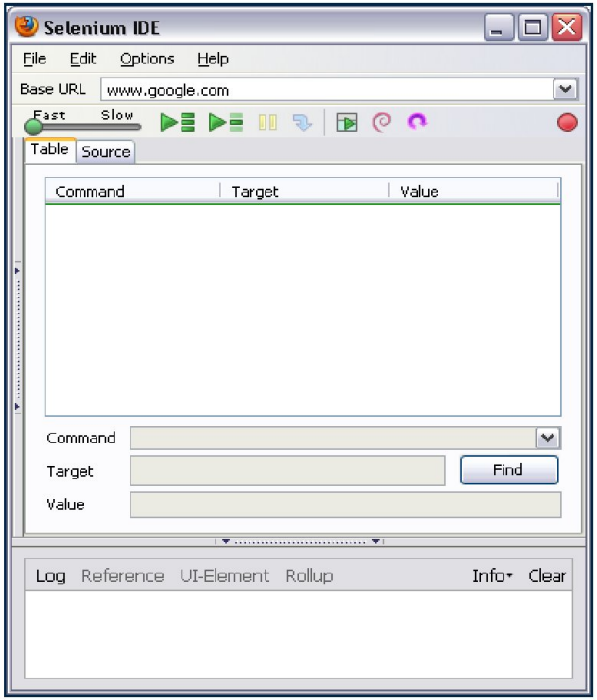
Copyright © Capgemini 2015. All Rights Reserved

Creating Test Scripts using Selenium IDE

- Go to the Web Page for which you want to carry out the test



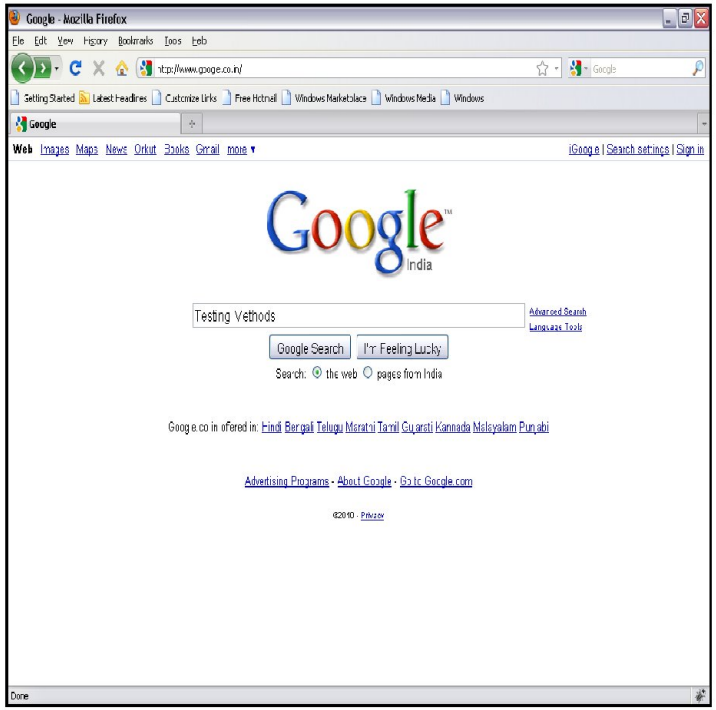
- Hit the record button on IDE



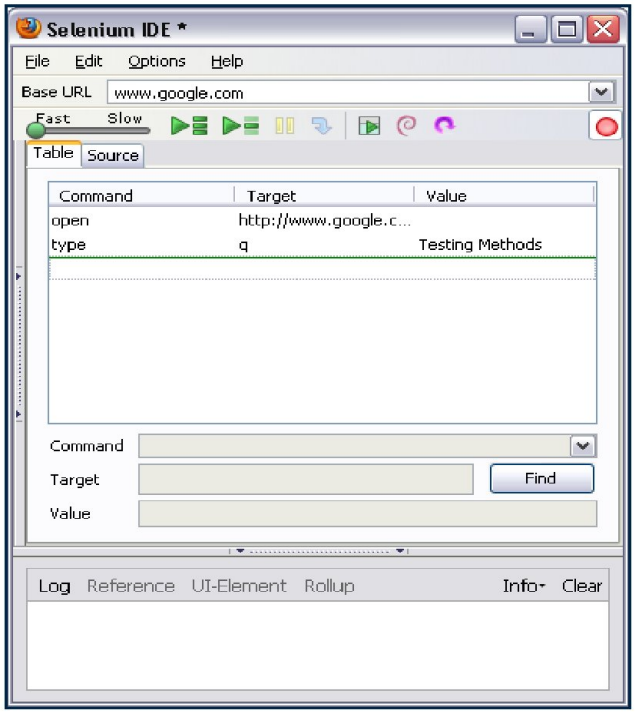
Copyright © Capgemini 2015. All Rights Reserved

Creating Test Scripts using Selenium IDE(Cont.)

- Enter the text on Web Page and submit

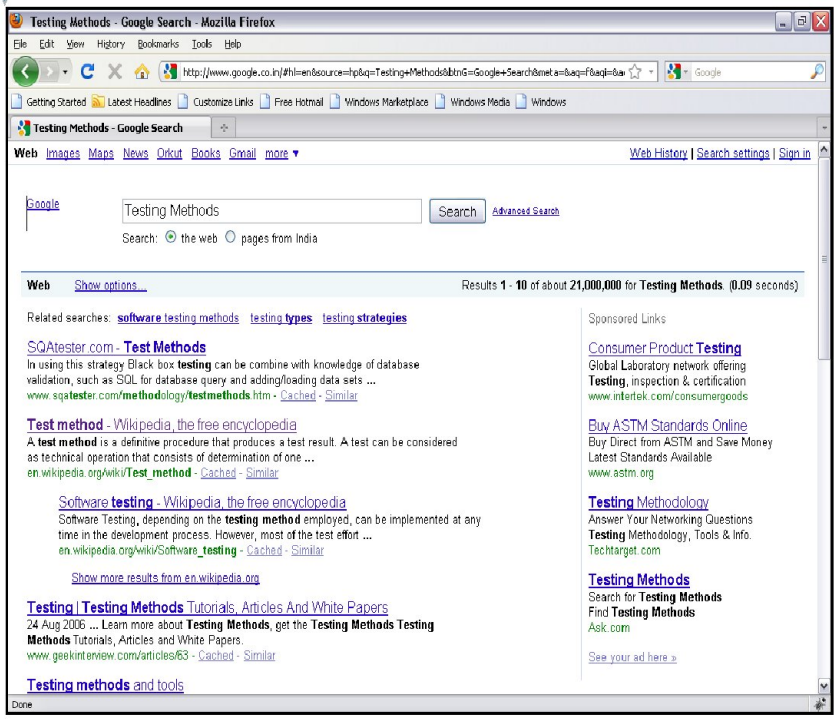


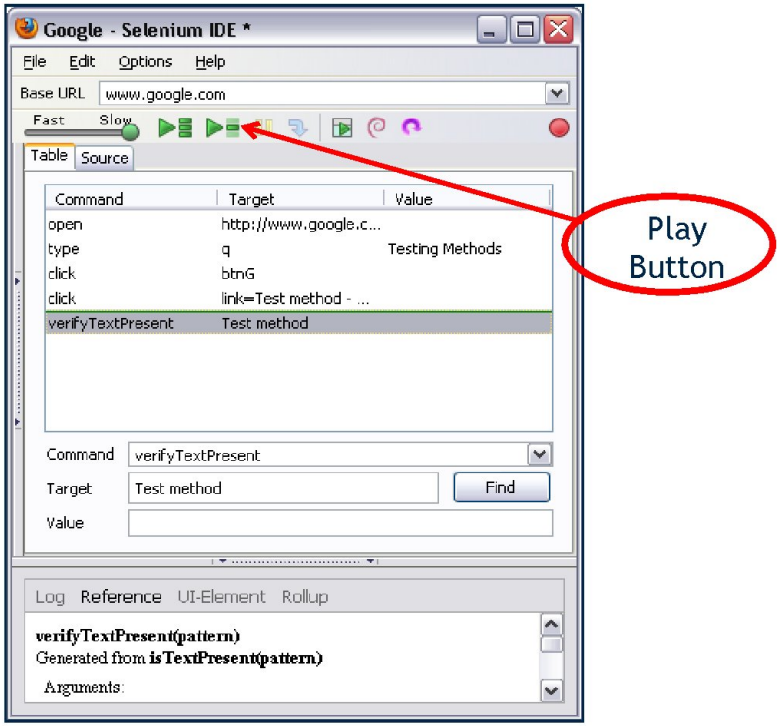
- IDE should be updated, stop the recorder and add the assertions



Copyright © Capgemini 2015. All Rights Reserved

Creating Test Scripts using Selenium IDE(Cont.)





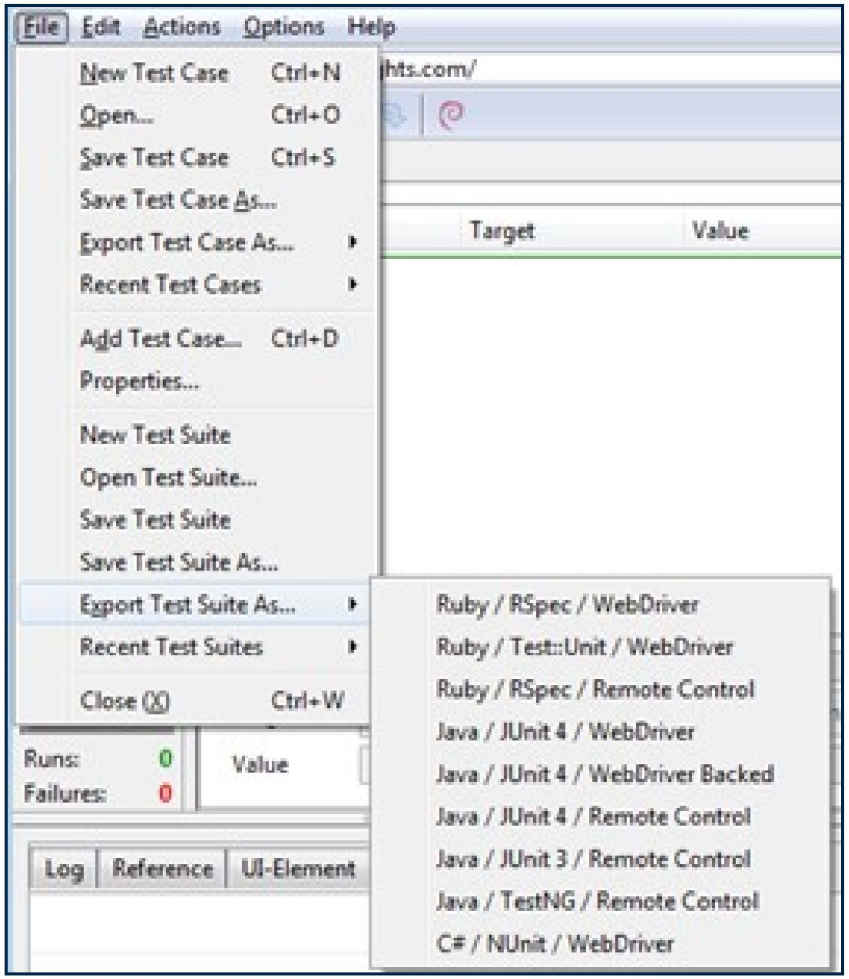
Hit the play button to play the recorded scripts

Copyright © Capgemini 2015. All Rights Reserved

Exporting scripts to multiple languages and Formats

- Test cases can be exported only to the following formats:

- .cs (C# source code)
- .java (Java source code)
- .py (Python source code)
- .rb (Ruby source code)



Copyright © Capgemini 2015. All Rights Reserved

Summary

- In this lesson, you have learnt
  - Selenium IDE (Integrated Development Environment) is the simplest tool in the Selenium Suite.
  - Menu bar is used in creating, modifying, and exporting test cases into formats useable by Selenium RC and WebDriver
  - The default format for Selenese commands is HTML.
  - The Test Case Pane shows the list of currently opened test cases and a concise summary of test runs
  - Locators tell Selenium IDE which GUI elements (say Text Box, Buttons, Check Boxes etc.) its needs to operate on
  - The choice of locator depends largely on your Application Under Test

Copyright © Capgemini 2015. All Rights Reserved

Review Question

- Question 1
  - Select the Locator which is NOT part of Selenium IDE
  - CSS Selector
  - XPath
  - getElementsById
  - getElementsByXpath
- Question 2: True/False
  - The choice of locator depends largely on your Application Under Test.
- Question 3: Fill in the Blanks
  - Error messages and information messages showing the progress are displayed automatically in _____ pane.

Copyright © Capgemini 2015. All Rights Reserved