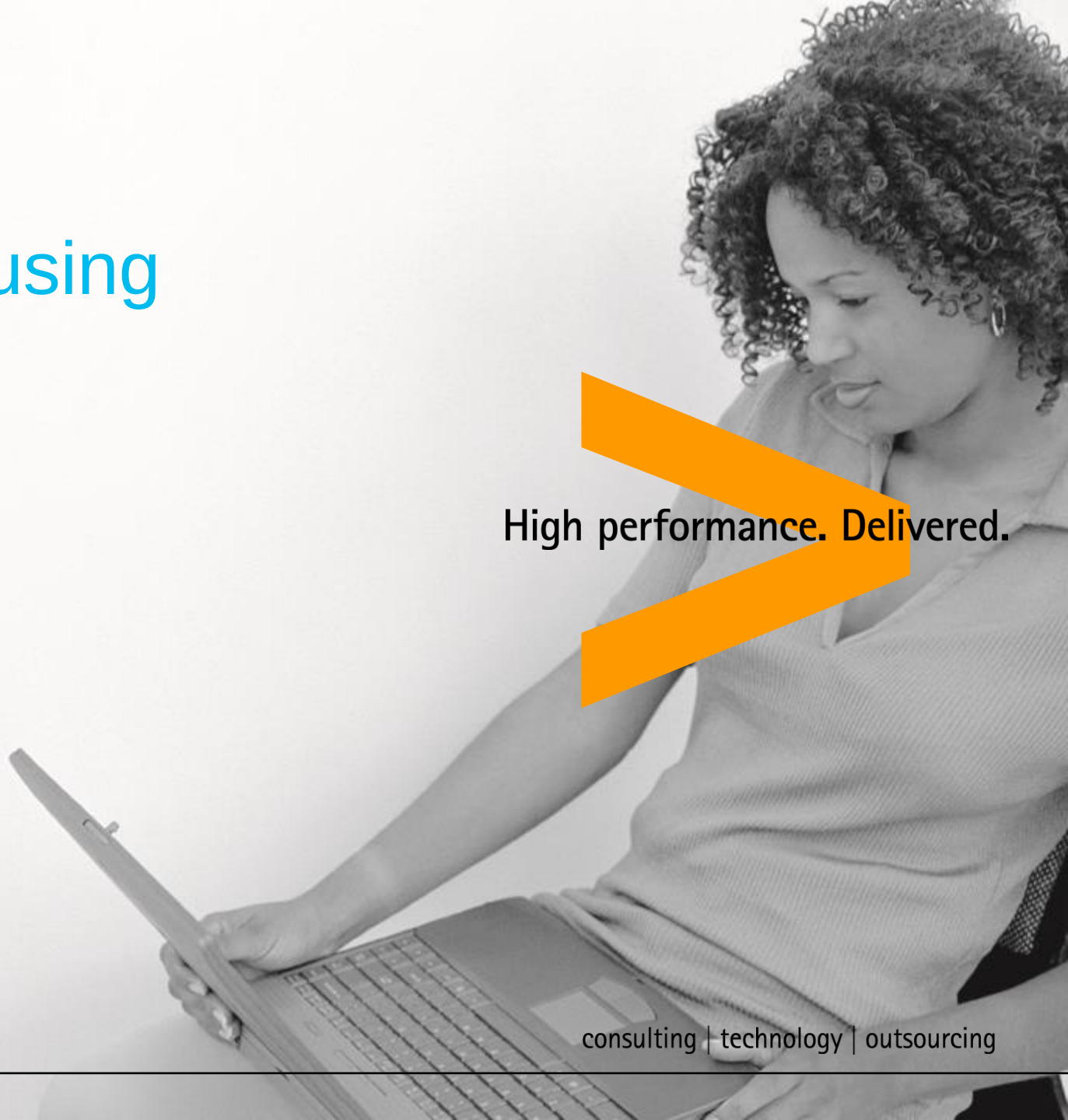
Selenium using C#



High performance. Delivered.

consulting | technology | outsourcing

Selenium Goals

- Selenium IDE
- Locators
- Selenium Web Driver
- Architecture
- Design Patterns
- Finding Elements
- Working with GUI Controls
- TestNG
- Selenium Grid
- ANT
- Jenkins Integration

Getting Started with Selenium IDE

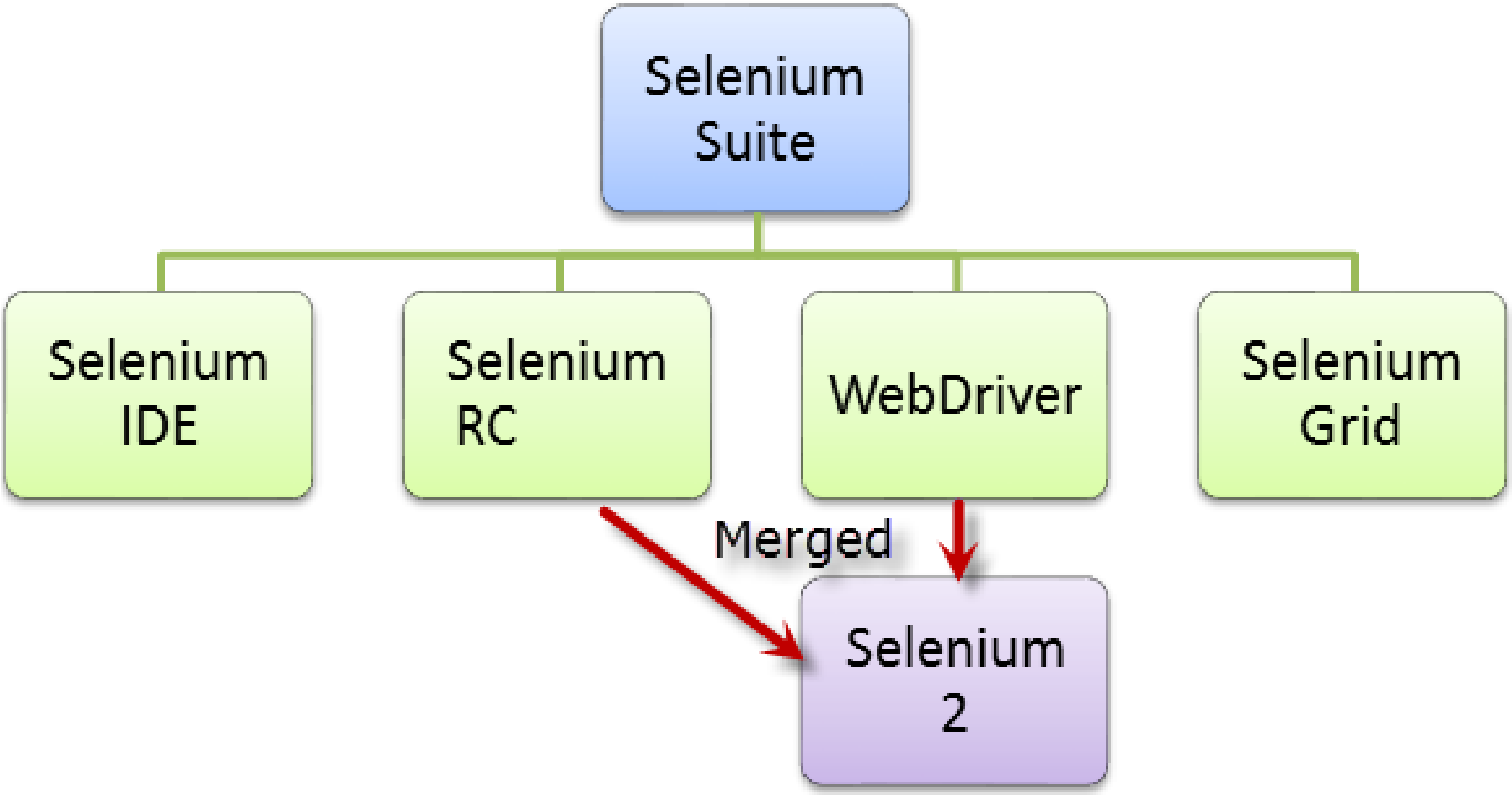
- Selenium is one of the most well known testing frameworks in the world that is in use.
- It is an open source project that allows testers and developers alike to develop functional tests to drive the browser.
- It can be used to record workflows so that developers can prevent future regressions of code.
- Selenium can work on any browser that supports JavaScript, since Selenium has been built using JavaScript.

Getting Started with Selenium IDE

- A set of tools that supports rapid development of test automation for web-based applications.
- Can be recorded and written as HTML
- Support for a number of programming languages: Java, C#, Perl, PHP, Python, Ruby
- Cross browsers support: IE, Firefox, Opera, Safari and Google Chrome
- Cross platform support: Windows, Linux, and Macintosh.

Getting Started with Selenium IDE

- Invented in 2004 by Jason R. Huggins and team.
- Originally named JavaScript Functional Tester [JSFT] 100% Javascript and HTML
- Designed to make test writing easy
- Open source browser based integration test framework built originally by ThoughtWorks
- Selenium is open source software, released under the Apache 2.0 license and can be downloaded and used without charge.

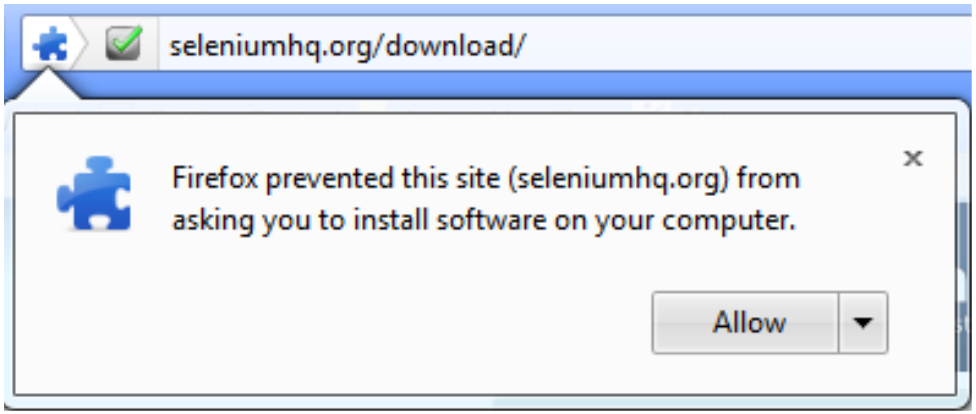

Selenium Suite
Selenium IDE
Selenium RC
WebDriver
Selenium Grid
Merged
Selenium 2

Getting Started with Selenium IDE

- Selenium IDE is a Firefox Add-on developed originally by Shinya Kasatani as a way to use the original Selenium Core code without having to copy Selenium Core onto the server.
- Selenium Core is the key JavaScript modules that allow Selenium to drive the browser.
- It has been developed using JavaScript so that it can interact with DOM (Document Object Model) using native JavaScript calls.
- Selenium IDE was developed to allow testers and developers to record their actions as they follow the workflow that they need to test.

Installing Selenium IDE

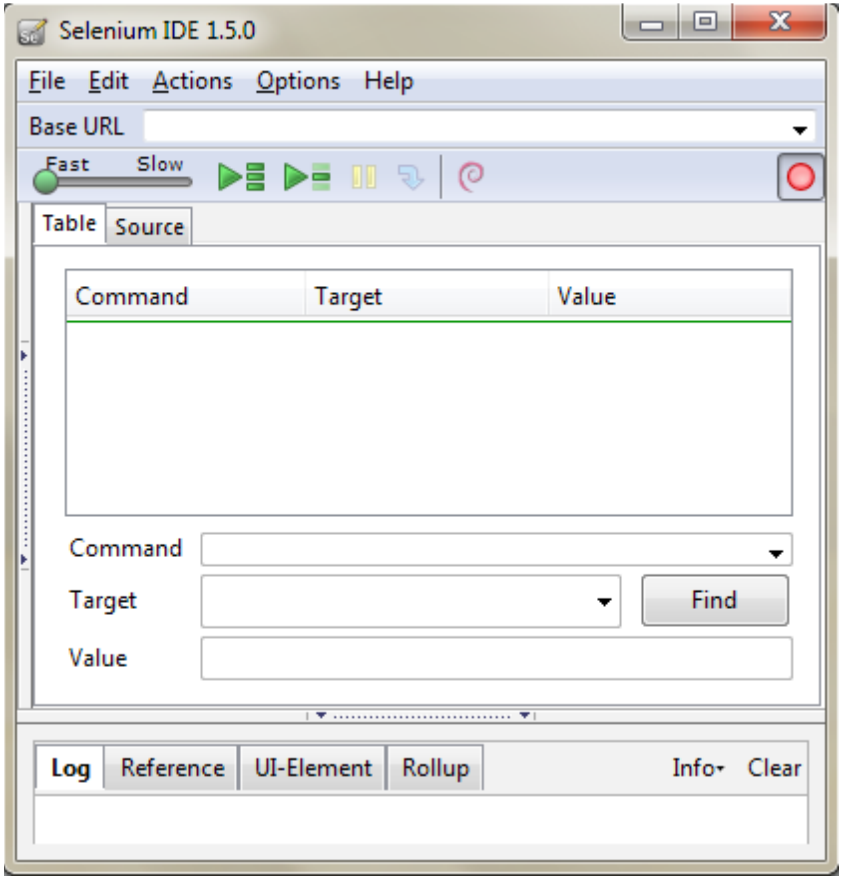
- 1. Go to http://seleniumhq.org/download/.
- ***2.*** Click on the download link for Selenium IDE. You may see a message appear saying **Firefox prevented this site (seleniumhq.org) from asking you to install software on your computer**. If you do, click the **Allow** button.



Installing Selenium IDE

- Once the install process is complete it will ask you to restart Firefox.
- Click the **Restart Now** button. Firefox will close and then re-open.
- If you have anything open in another browser it might be worth saving your work, as Firefox will try to go back to its original state but this cannot be guaranteed.

Selenium IDE



Selenium IDE

- Base URL: This is the URL that the test will start at. All open commands will be relative to the Base URL unless a full path is inserted in the open command.
- Speed Slider: This is the slider under the **Fast** and **Slow** labels on the screen.
- Run all the tests in the IDE.
- Run a single test in the IDE.
- Pause a test that is currently running.
- Step through the test once it has paused.
- This is the record button. This will be engaged when the test is recording.
- The **Command** selectbox has a list of all the commands that are needed to create a test. You can type into it to use the auto complete functionality or use it as a dropdown.

Selenium IDE

- The **Target** textbox allows you to input the location of the element that you want to work against.
- The **Find** button, once the target box is populated, can be clicked to highlight the element on the page.
- The **Value** textbox is where you place the value that needs to change. For example, if you want your test to type in an input box on the web page, you would put what you want it to type in the value box.
- The **Test** table will keep track of all your commands, targets, and values. It has been structured this way because the original version of Selenium was styled on FIT tests. FIT was created by Ward Cunningham and means Framework for Integrated Testing. The tests were originally designed to be run from HTML files and the IDE keeps this idea for its tests.
- If you click the **Source** tab you will be able to see the HTML that will store the test. Each of the rows will look like:

```
<tr>
  <td>open</td>
  <td>/chapter1</td>
  <td></td>
</tr>
```

Selenium IDE

- The area below the **Value** textbox will show the Selenium log while the tests are running. If an item fails, then it will have an `[error]` entry.
  This area will also show help on Selenium Commands when you are working in the Command selectbox. This can be extremely useful when typing commands into Selenium IDE instead of using the record feature.
- The **Log** tab will show a log of what is happening during the test. The **Reference** tab gives you documentation on the command that you have highlighted.

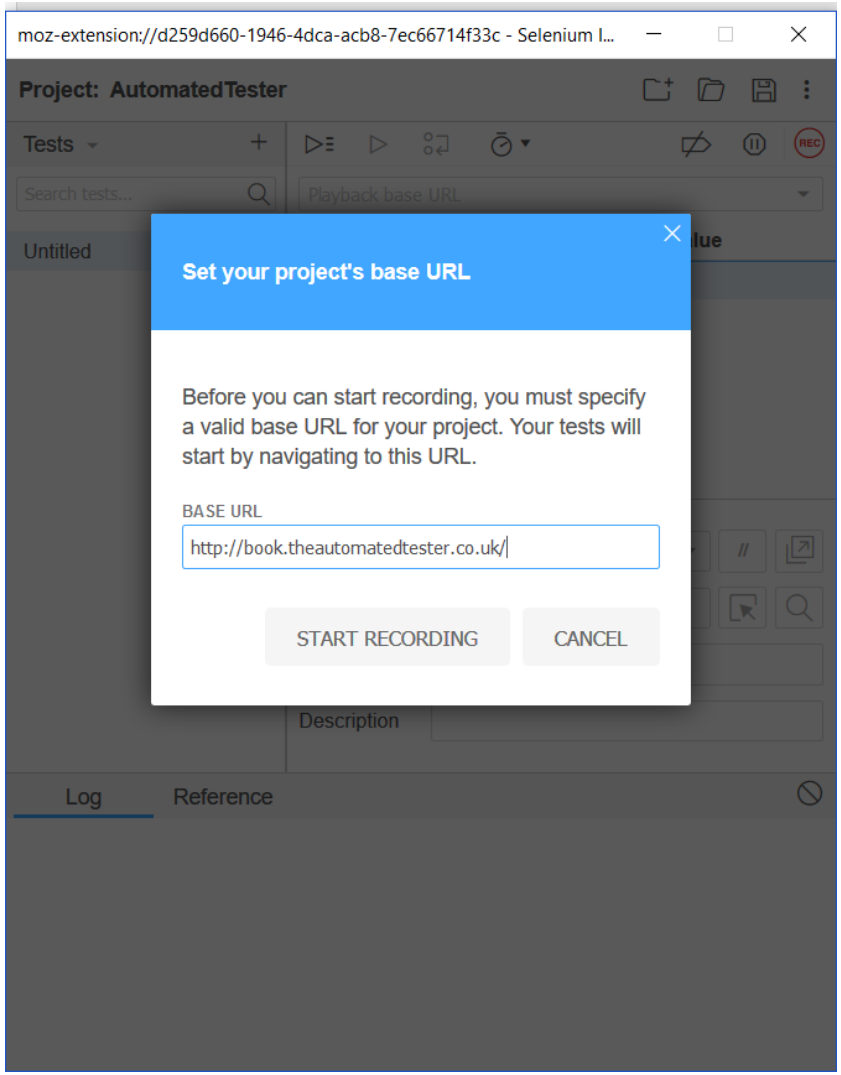
Rules for automation

- Tests should always have a known starting point.
- In the context of Selenium, this could mean opening a certain page to start a workflow.
- ‹Tests should not have to rely on any other tests to run. If a test is going to add something, do not have a separate test to delete it.
- This is to ensure that if something goes wrong in one test, it will not mean you have a lot of unnecessary failures to check.
- ৰTests should only test one thing at a time.
- ৰTests should clean up after themselves.

Recording your first test with Selenium IDE

- To start recording the tests we will need to start Mozilla Firefox.
- Once it has been loaded, you will need to start Selenium IDE.
- You will find it under the Tools dropdown menu in Mozilla Firefox or in the Web Developer dropdown menu.
- Once loaded it will look like the next screenshot. Note that the record button is engaged when you first load the IDE.

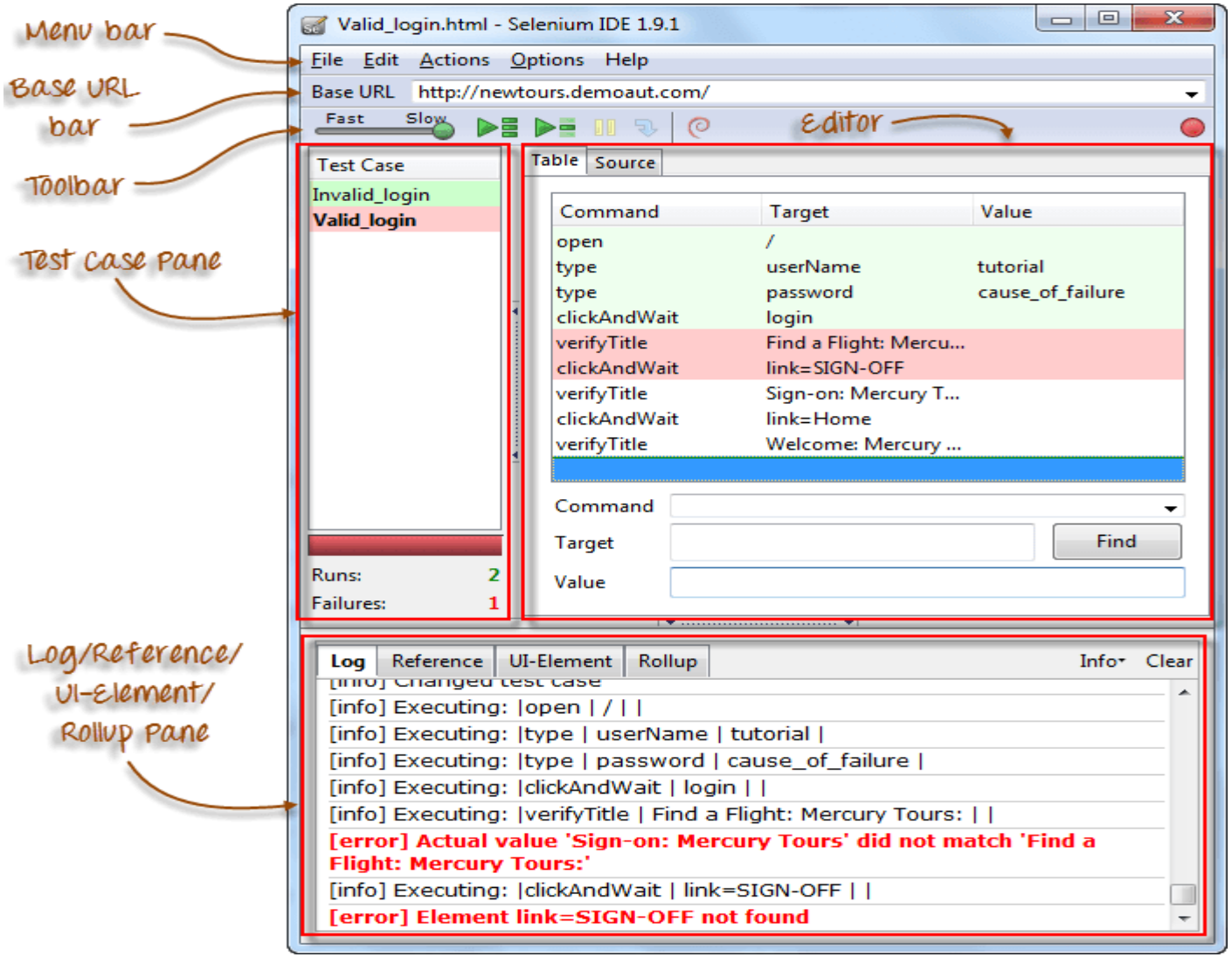
Recording your first test with Selenium IDE



verify and assert methods

- ‹verifyElementPresent
- ‹assertElementPresent
- ‹verifyElementNotPresent
- ‹assertElementNotPresent
- ‹verifyText
- ‹assertText
- ‹verifyAttribute
- ‹assertAttribute
- ‹verifyChecked
- ‹assertChecked
- ‹verifyAlert
- ‹assertAlert
- ‹verifyTitle
- ‹assertTitle

Selenium IDE



Introduction to Selenium Commands - Selenese

- **3 Types of Commands**

Actions	These are commands that directly interact with page elements. Example: the "click" command is an action because you directly interact with the element you are clicking at. The "type" command is also an action because you are putting values into a text box, and the text box shows them to you in return. There is a two-way interaction between you and the text box.

Introduction to Selenium Commands - Selenese

- **3 Types of Commands**

Accessors	They are commands that allow you to store values to a variable. Example: the "storeTitle" command is an accessor because it only "reads" the page title and saves it in a variable. It does not interact with any element on the page.

Introduction to Selenium Commands - Selenese

- **3 Types of Commands**

Assertions	They are commands that verify if a certain condition is met. **3 Types of Assertions** •**Assert**. When an "assert" command fails, the test is stopped immediately. •**Verify**. When a "verify" command fails, Selenium IDE logs this failure and continues with the test execution. •**WaitFor**. Before proceeding to the next command, "waitFor" commands will first wait for a certain condition to become true. • If the condition becomes true within the waiting period, the step passes. • If the condition does not become true, the step fails. Failure is logged, and test execution proceeds to the next command. • By default, the timeout value is set to 30 seconds. You can change this in the Selenium IDE Options dialog under the General tab.

ASSERT

test execution was halted in this part

Command	Target	Value
open		
assertTitle	Welcome: Venus Tours	
type	name=userName	invalidUN
type	name=password	invalidPW
clickAndW...	name=login	

Command

Target

Find

Value

Log | Reference | UI-Element | Rollup | Info | Clear

[info] Executing: |open | | |

[info] Executing: |assertTitle | Welcome: Venus Tours | |

[error] Actual value 'Welcome: Mercury Tours' did not match 'Welcome: Venus Tours'

no further logs were displayed after this error message, meaning that execution indeed stopped

VERIFY

Execution continued despite the error

Commands after the Failed verify command were still executed

Command	Target	Value
open		
verifyTitle	Welcome: Venus Tours	
type	name=userName	invalidUN
type	name=password	invalidPW
clickAndW...	name=login	

Command

Target

Find

Value

Log | Reference | UI-Element | Rollup | Info | Clear

[info] Executing: |verifyTitle | Welcome: Venus Tours | |

[error] Actual value 'Welcome: Mercury Tours' did not match 'Welcome: Venus Tours'

[info] Executing: |type | name=userName | invalidUN |

[info] Executing: |type | name=password | invalidPW |

Command	Number of Parameters	Description
open	0 - 2	Opens a page using a URL.
click/clickAndWait	1	Clicks on a specified element.
type/typeKeys	2	Types a sequence of characters.
verifyTitle/assertTitle	1	Compares the actual page title with an expected value.
verifyTextPresent	1	Checks if a certain text is found within the page.
verifyElementPresent	1	Checks the presence of a certain element.
verifyTable	2	Compares the contents of a table with expected values.
waitForPageToLoad	1	Pauses execution until the page is loaded completely.
waitForElementPresent	1	Pauses execution until the specified element becomes present.

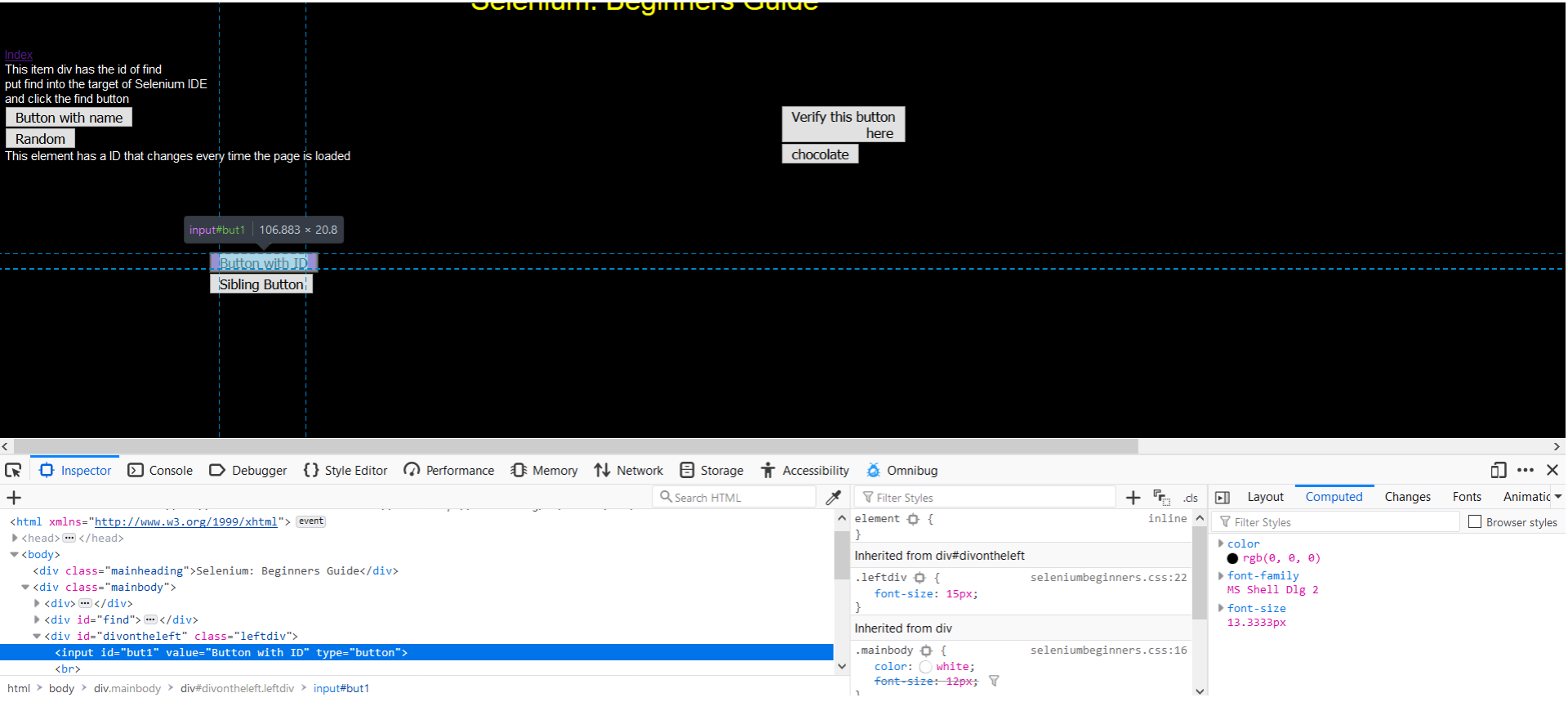
Locators

- *Locators allow us to find elements on a page that can be used in our tests.*
- Locate elements by ID
- ‹Locate elements by Name
- ‹Locate elements by Link
- ‹Locate elements by XPath
- ‹Locate elements by CSS
- ‹Locate elements by DOM

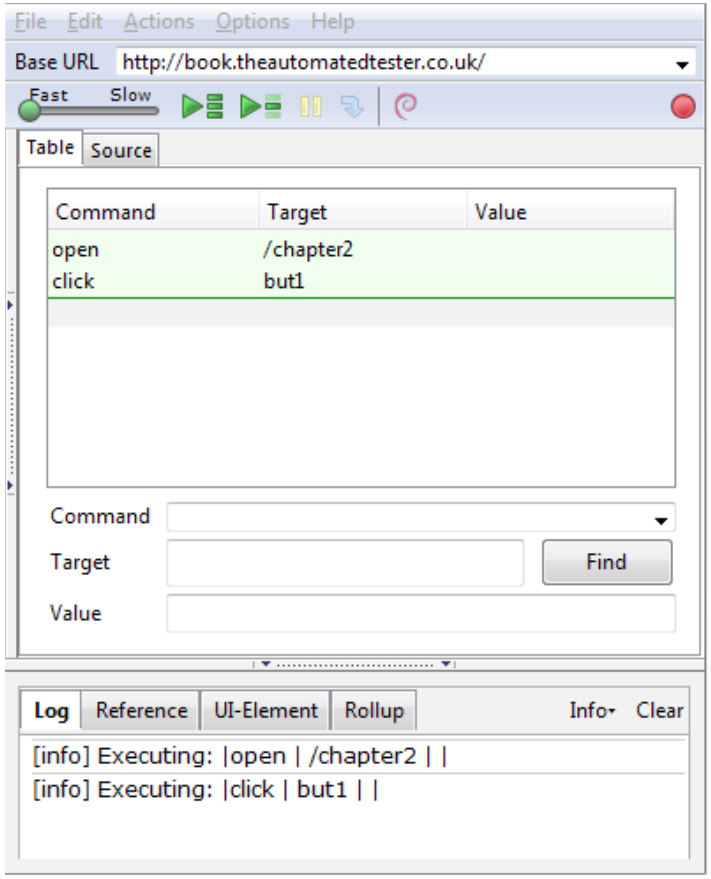
Locating elements by ID

- On web applications today, elements should have an ID attribute for all their controls on the page.
- A control would be an element that we can interact with and is not static text.
- This allows Selenium to find the unique item, since IDs should be unique, and then complete the action that it needs to do against that element.

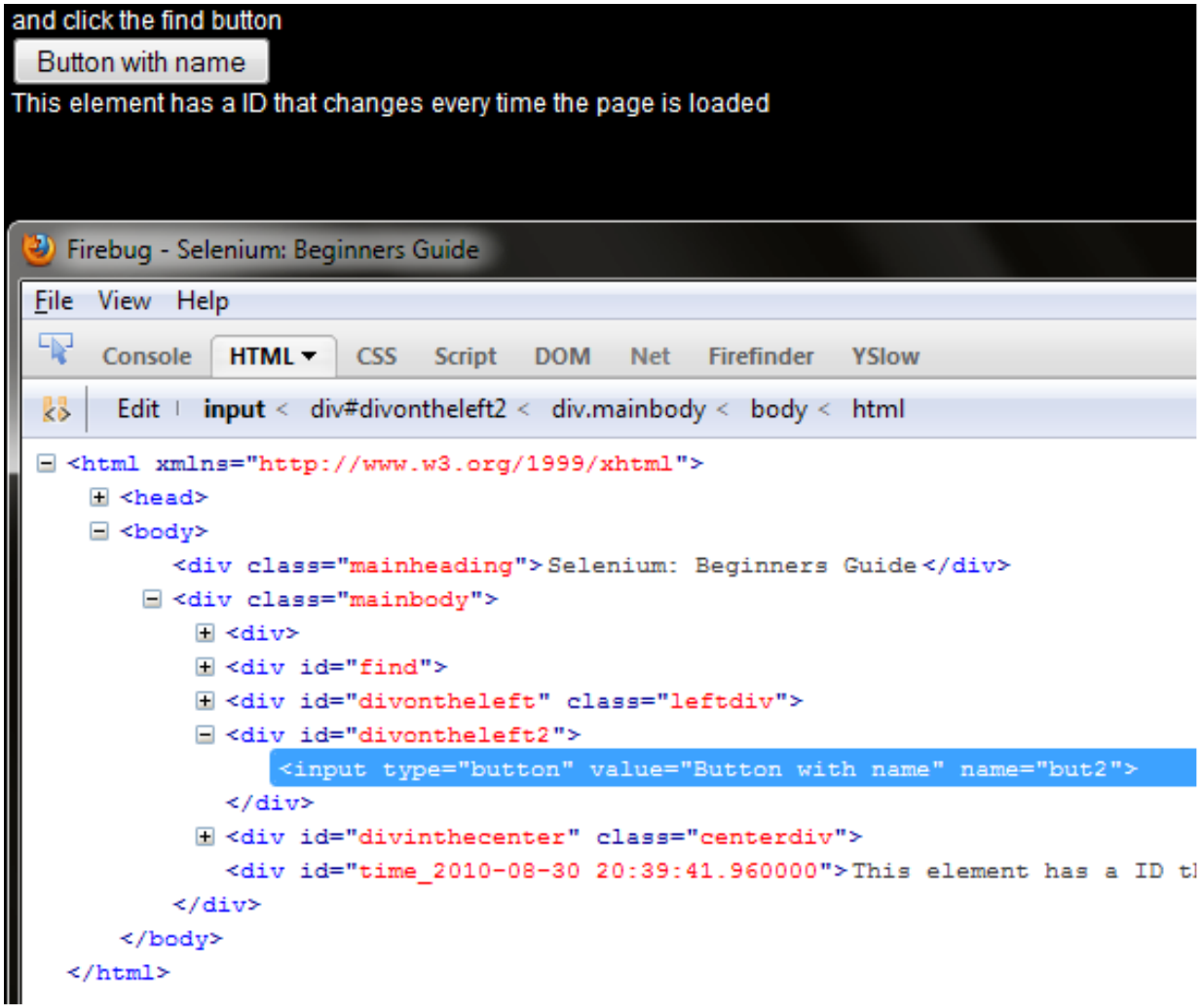
Locating elements by ID



Locating elements by ID

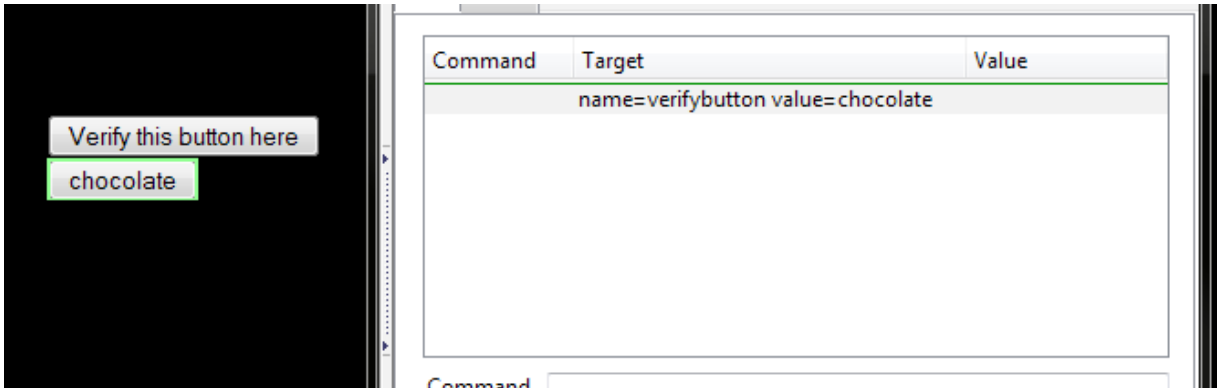


finding elements by name



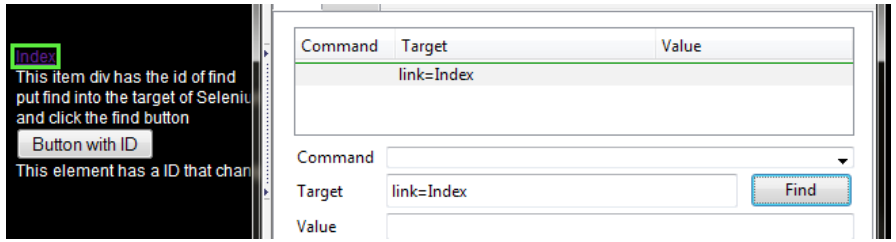
Adding filters to the name

- There are times when there may be elements on the page that have the same name but a different attribute.
- When this happens we can apply filters to the locator so that Selenium IDE can find the element that we are after.



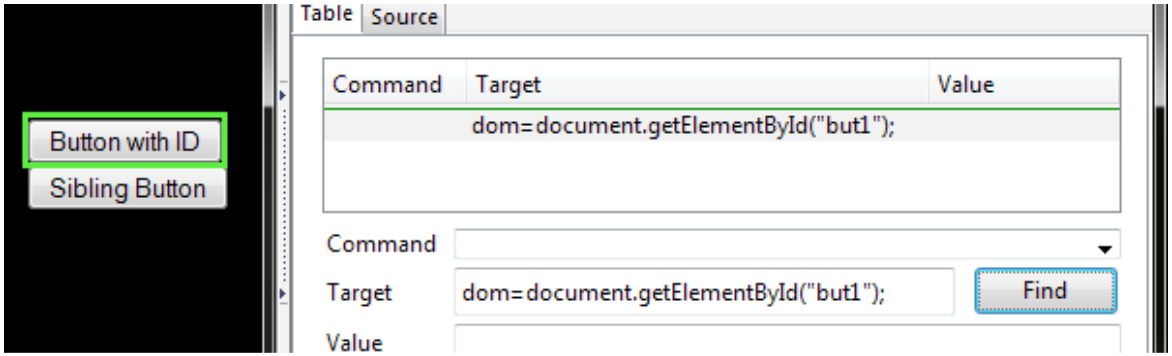
finding elements by link text

- Probably the most common element on a page is a link.
- Links allow pages to be joined together so end users can navigate your site with confidence.
- You can see a screenshot of the element being found in Selenium IDE.



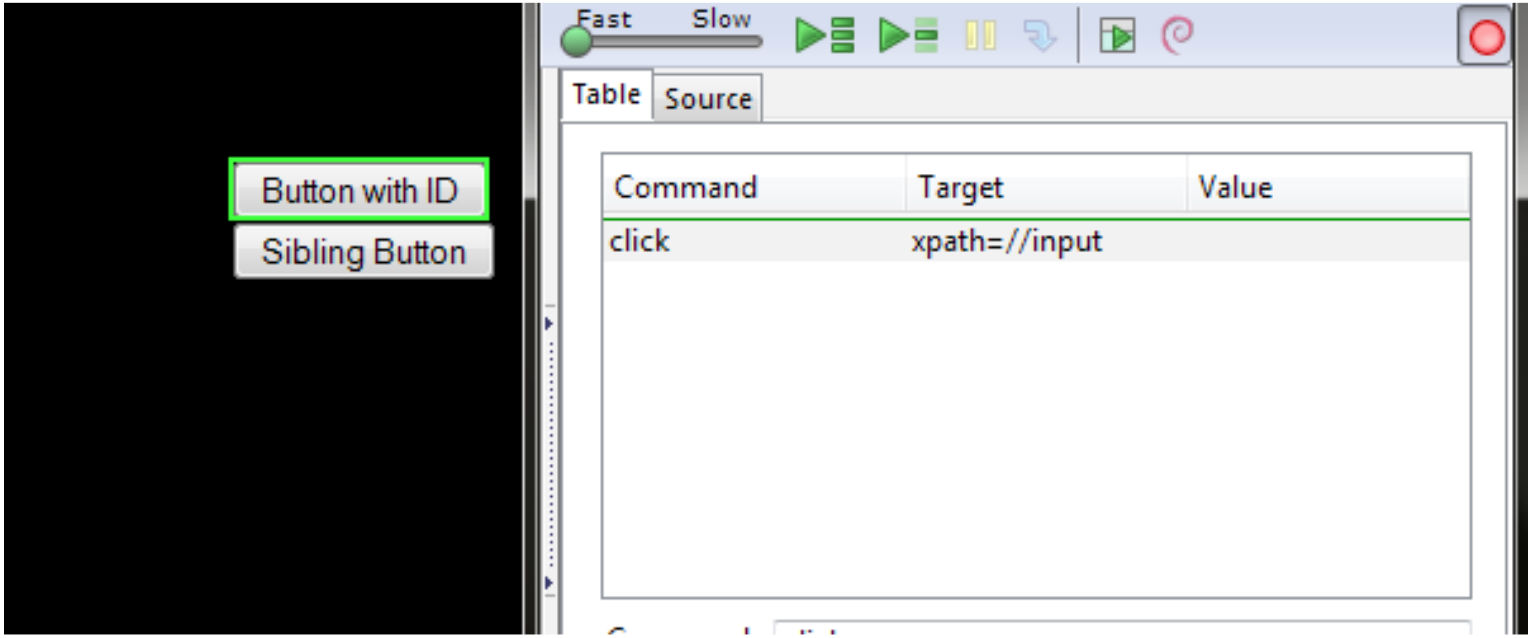
finding elements by accessing the DOM via JavaScript

- There are times where the DOM will be updated via AJAX and this means that our locator needed for the test will need some form of JavaScript to see if it is there.
- In JavaScript, calling the DOM to find the first link on the page would look like document.links[0];. document represents the HTML document and links is an array on that object.

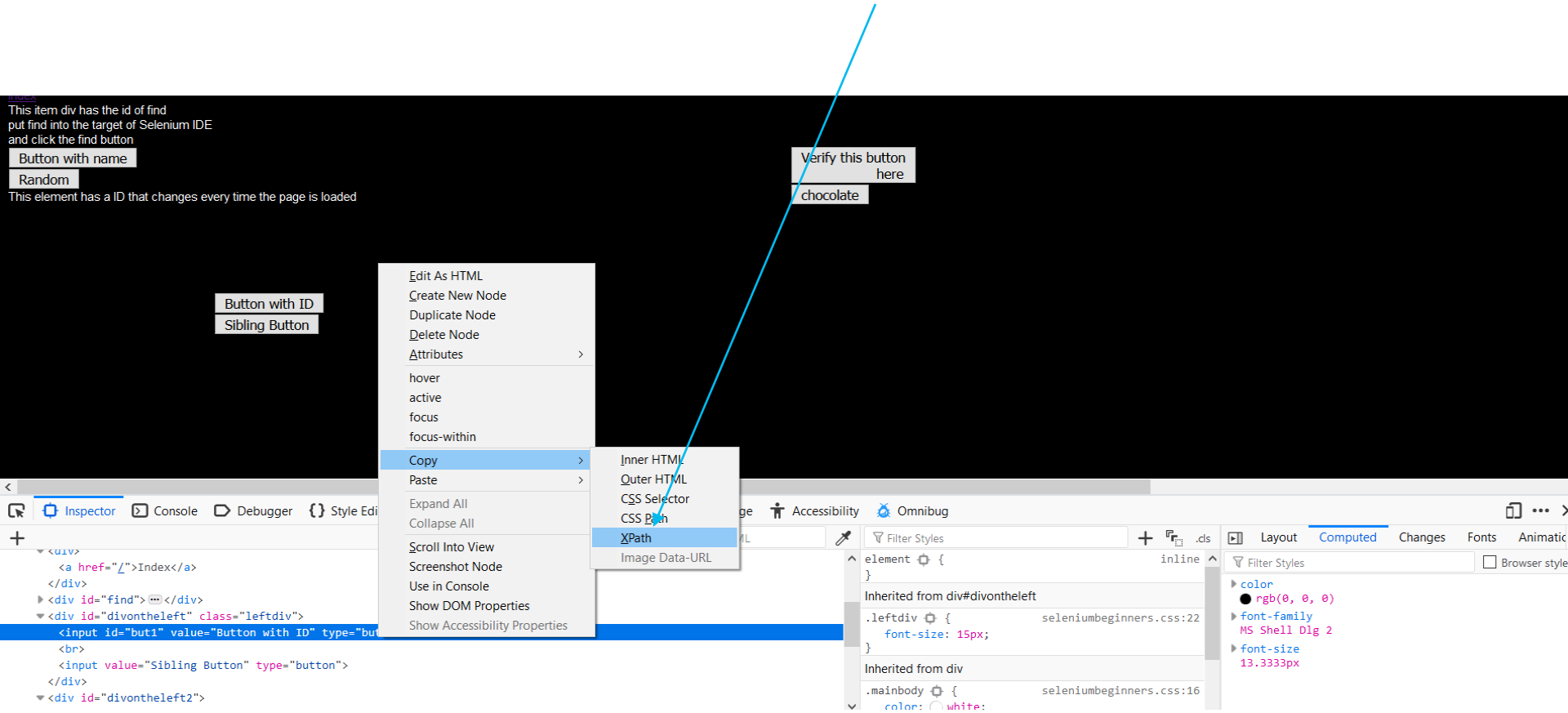


finding elements by XPath

- XPath allows us to query the DOM as though it were an XML document.
- With XPath we can do some rather complex queries to find elements on the page that may not have been accessible otherwise.



finding elements by XPath



Finding an element by the text it contains

- Finding elements by the text they contain can be quite useful when working with web pages that have been created dynamically.
- The elements could be from using a web based WYSIWYG editor or you might just like to find a paragraph on the page with specific text to then do further queries on.
- To do this your query will need to have the text() method call in the query.
- It will match the entire contents of the node if it has the format //element[text()='inner text'].

Finding an element by the text it contains

Table | Source

Comma...	Target	V...
	//div[contains(text(),'element has a ID')]	
	//div[text()='This element has a ID that changes every time...	

Button with ID

Sibling Button

Com...	Target	Value
	//input[@value='Button with ID']/following-sibling::input[@value='Sibling...	

Command

Target //input[@value='Button with ID']/following-sibling::input[@value='Sibling Button'] Find

Value

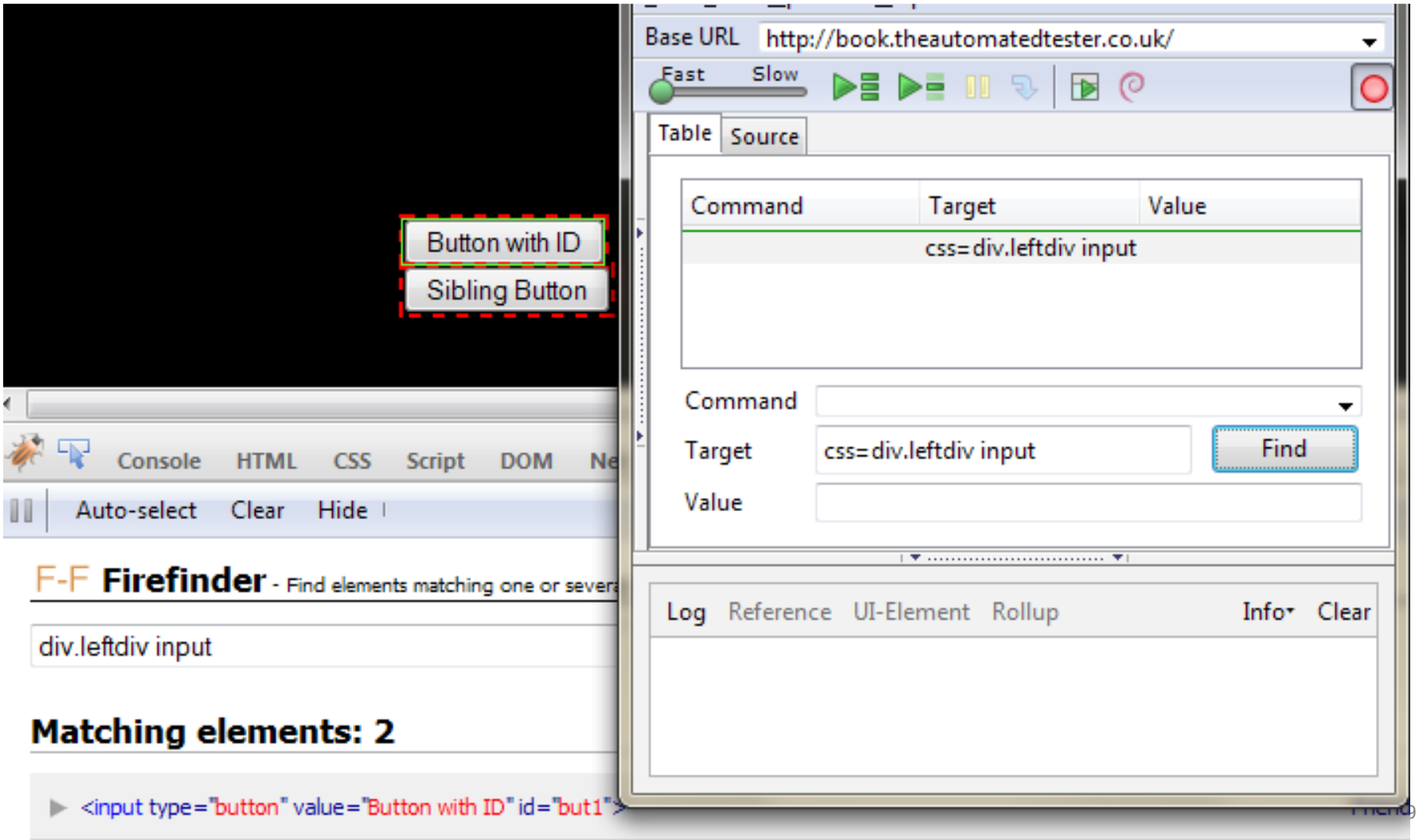
Finding an element by the text it contains

Axis name	Result
ancestor	Selects all the ancestors (parent, grandparent, and so on) of the element
descendant	Selects all the descendants (children, grandchildren, and so on) of the element
following	Selects all elements that follow the closing tab of the current element
following-sibling	Selects all the siblings after the current element
parent	Selects the parent of the current element
preceding	Selects all elements that are before the current element
preceding-Sibling	Selects all of the siblings before the current element

CSS selectors

- So, finding elements by XPath can be an extremely costly exercise.
- A way around this is to use CSS selectors to find the objects that you need. Selenium is compatible with CSS 1.0, CSS 2.0, and CSS 3.0 selectors.
- There are a number of items that are supported like namespace in CSS 3.0 and some pseudo classes and pseudo elements.

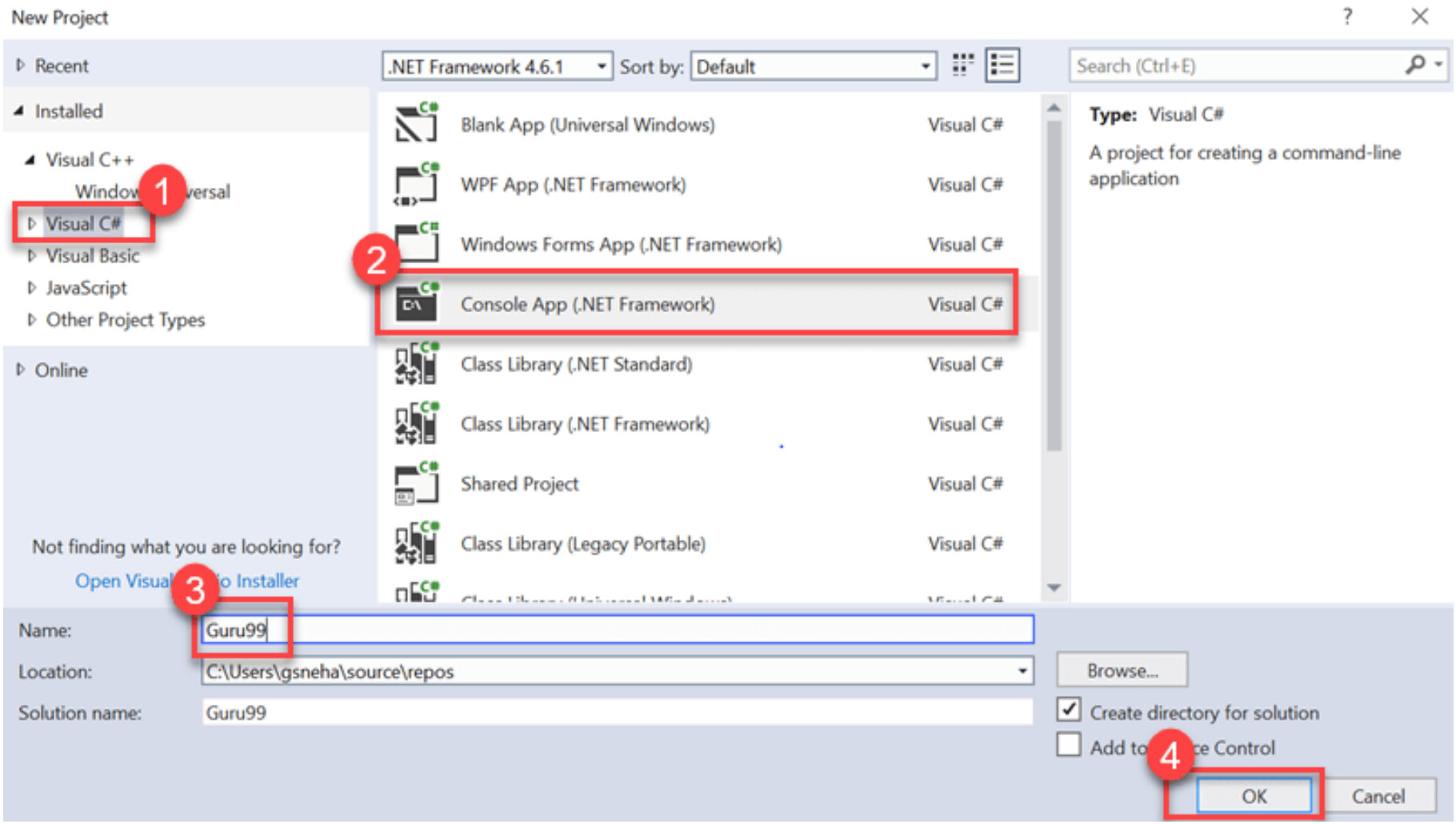
CSS selectors



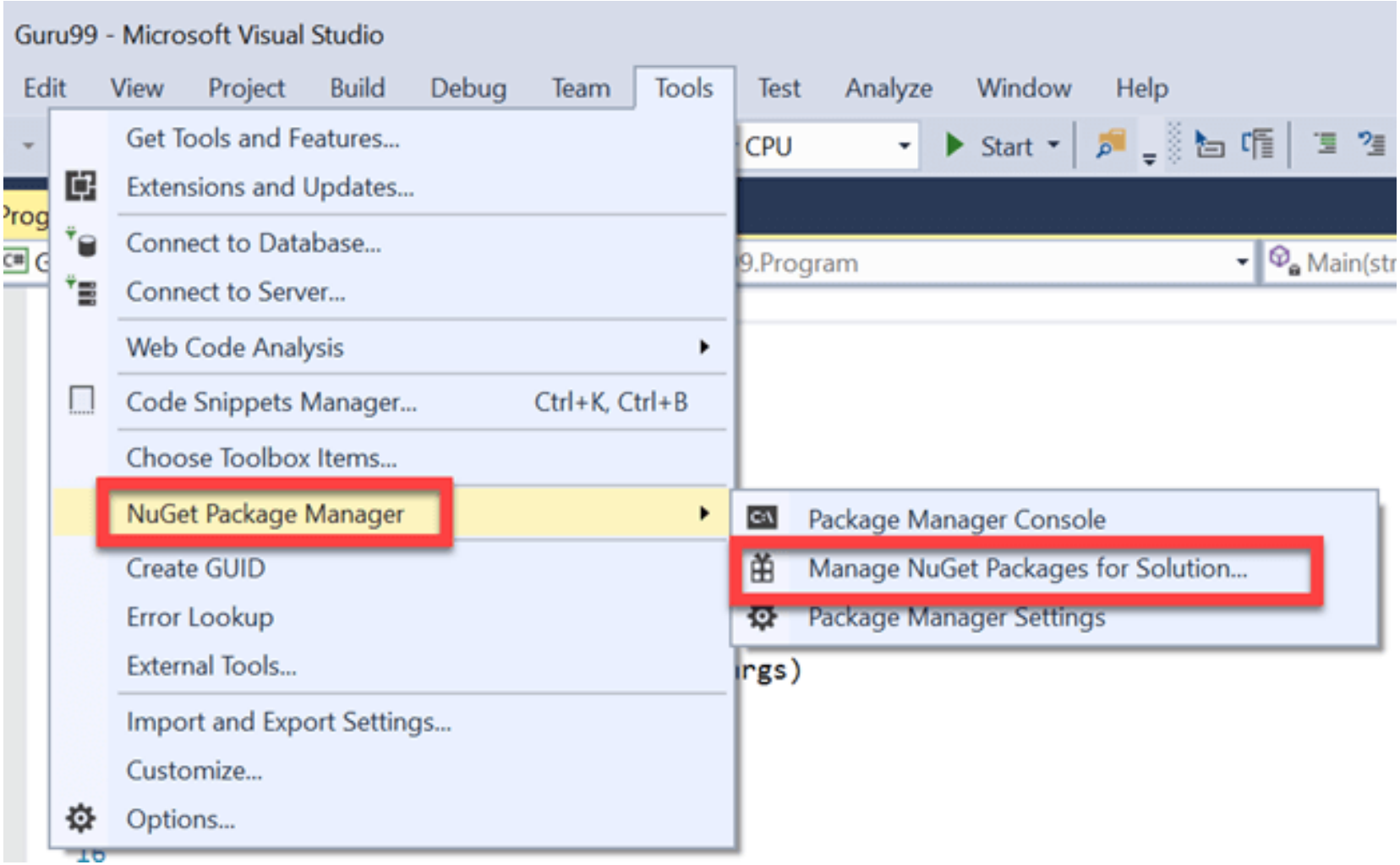
Selenium WebDriver

- WebDriver on the other hand tries to control the browser from outside the browser. It uses accessibility API to drive the browser.
- The accessibility API is used by a number of applications for accessing and controlling applications when they are used by disabled users and is common to web browsers.
- WebDriver uses the most appropriate way to access the accessibility API. If we look at Firefox, it uses JavaScript to access the API.
- If we look at Internet Explorer, it uses C++. This approach means we can control browsers in the best possible way but has the downside that new browsers entering the market will not be supported straight away like we can with Selenium RC.

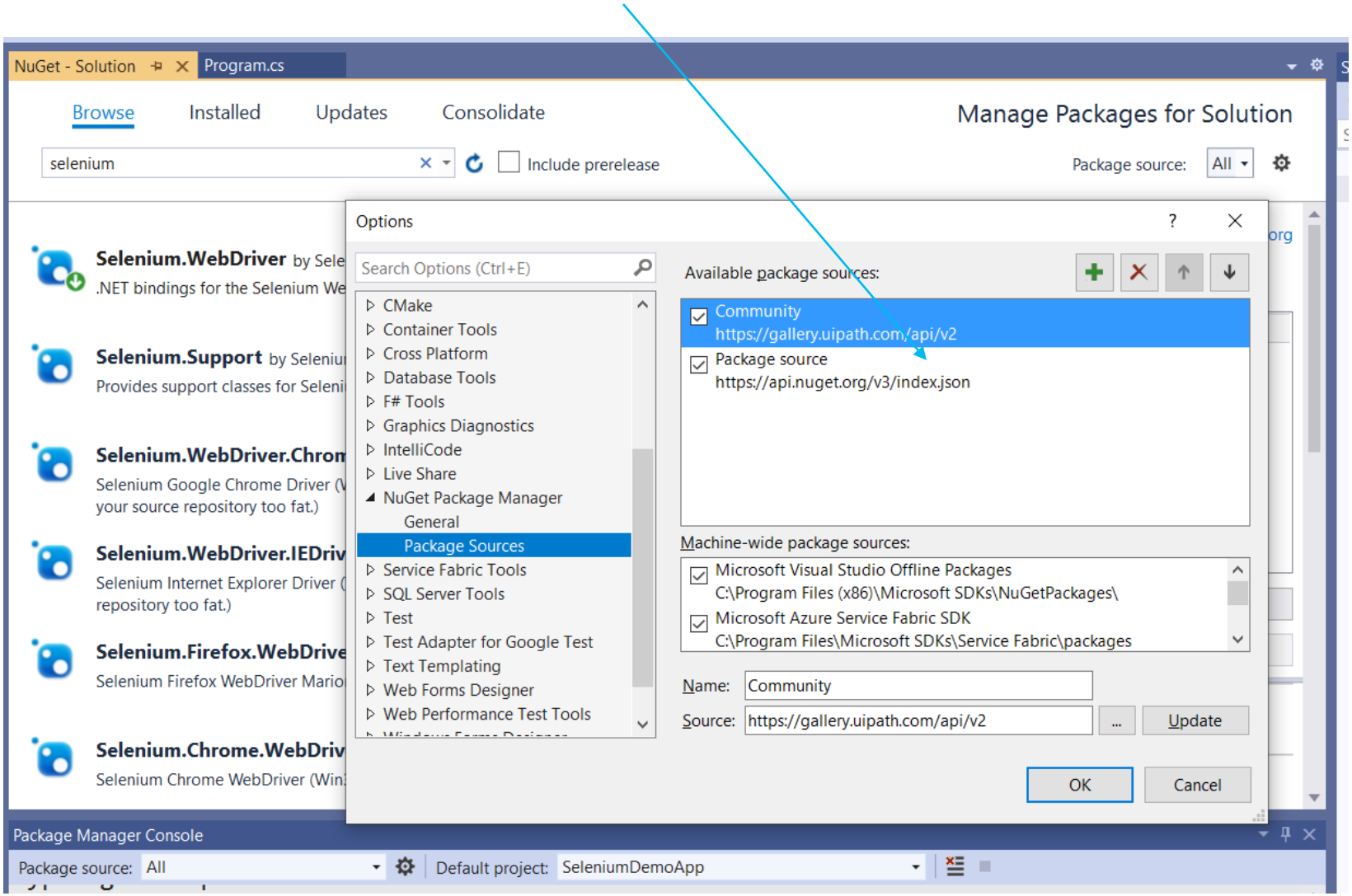
Visual Studio .NET



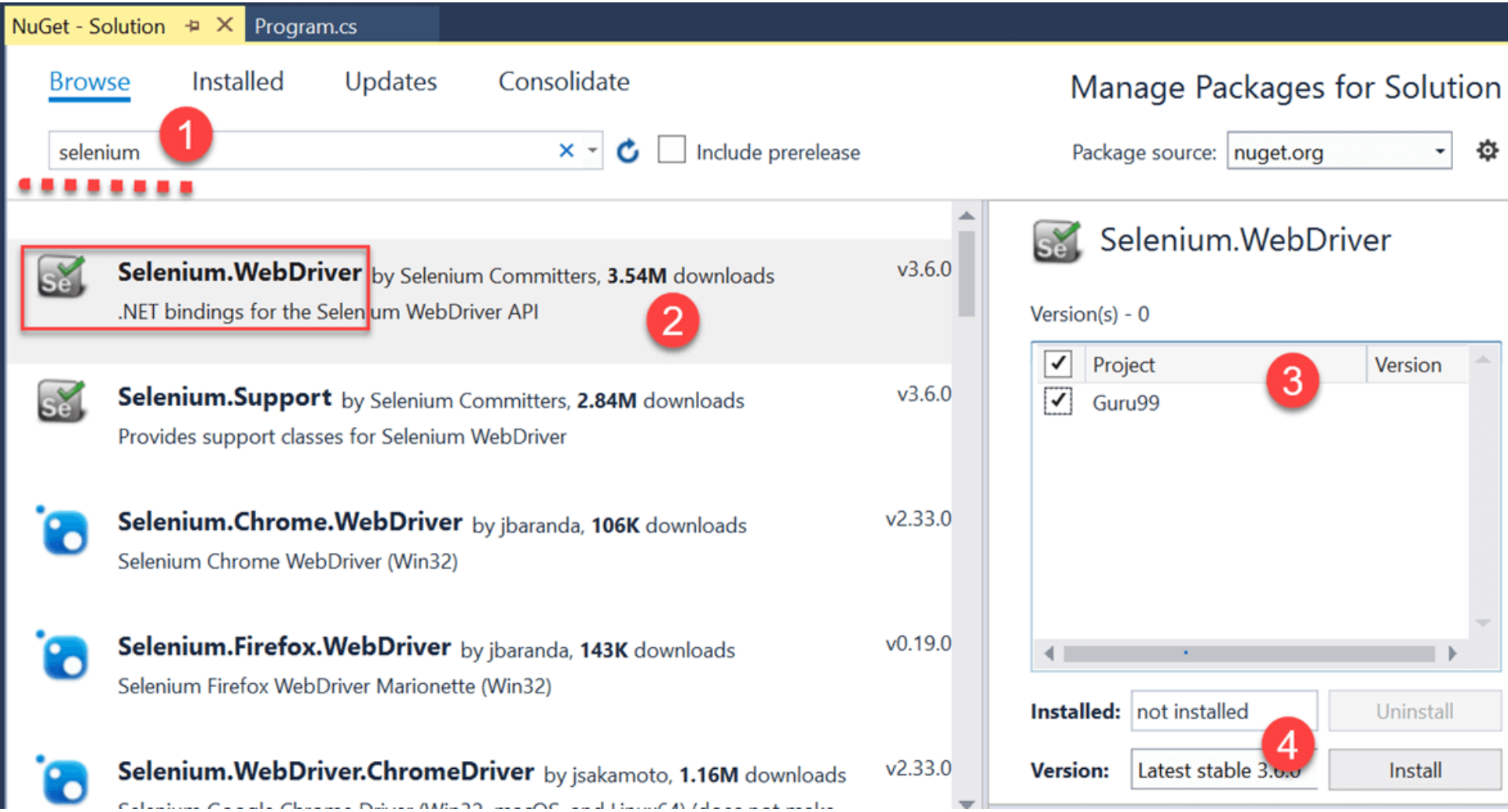
Visual Studio .NET



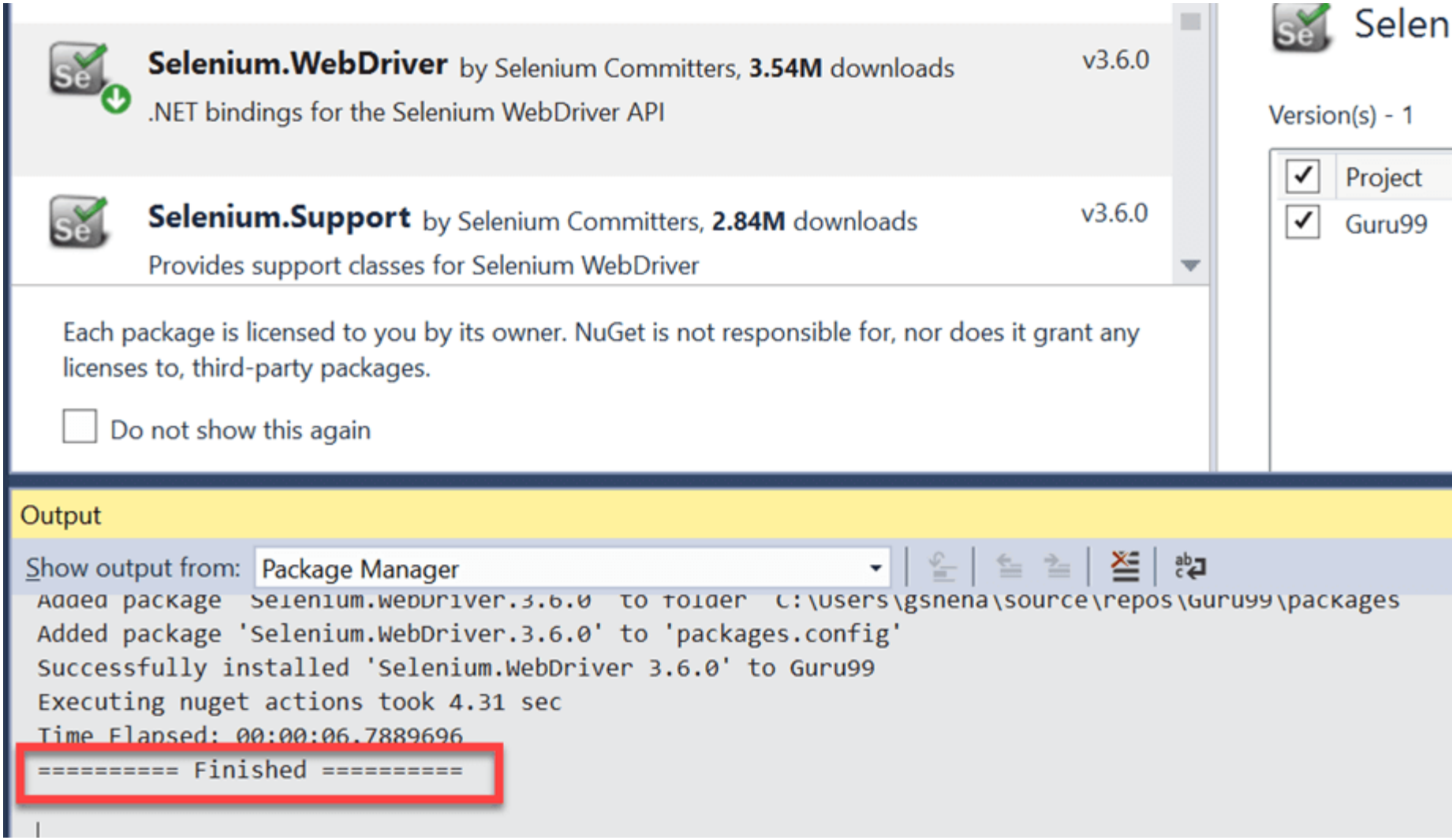
Visual Studio .NET



Visual Studio .NET



Visual Studio .NET



Visual Studio .NET

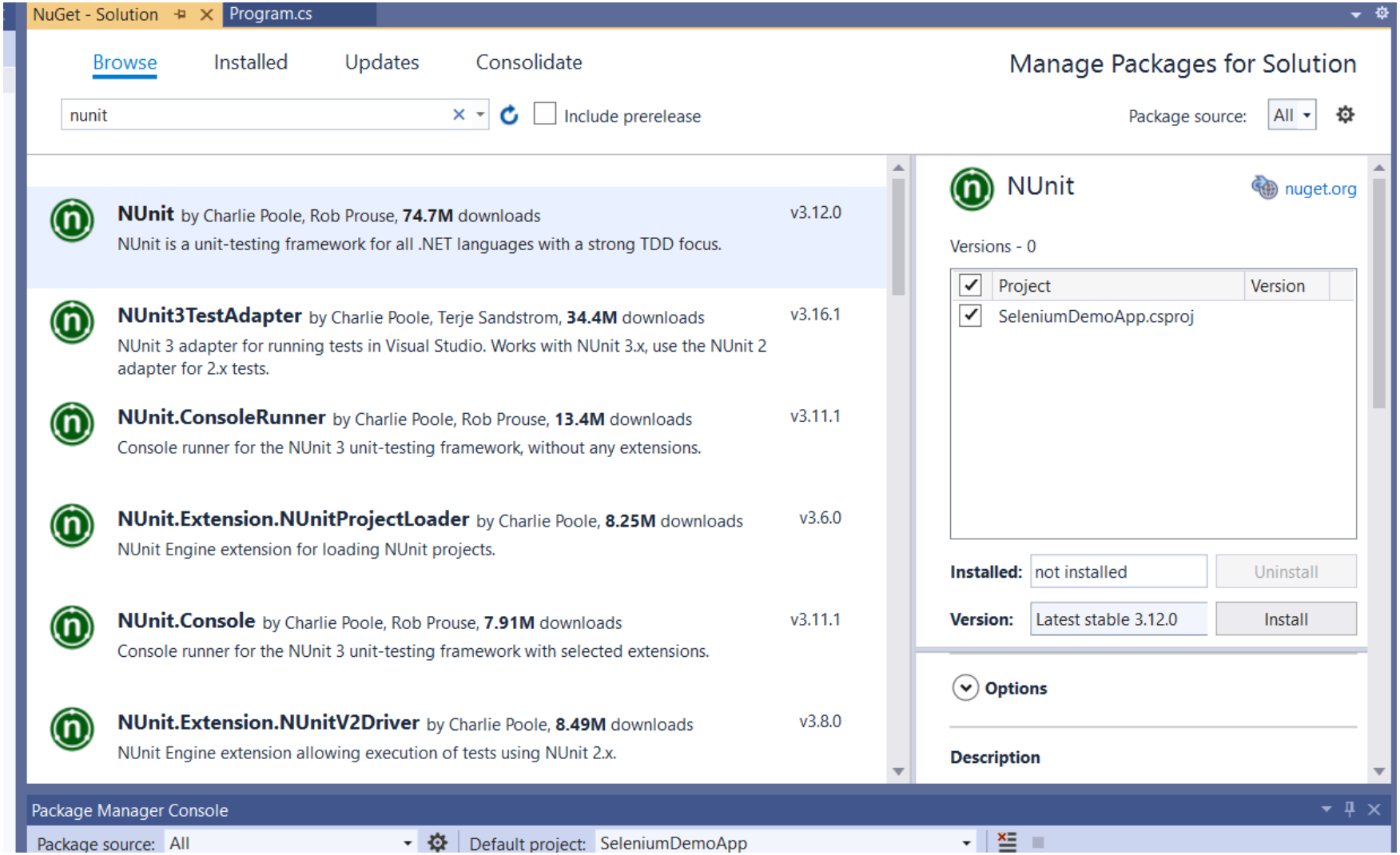
Package Manager Console

Package source: All | Default project: SeleniumDemoApp

```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it
grant any licenses to, third-party packages. Some packages may include dependencies which
 are governed by additional licenses. Follow the package source (feed) URL to determine
any dependencies.

Package Manager Console Host Version 5.5.0.6473

Type 'get-help NuGet' to see all available NuGet commands.

PM> Install-Package Selenium.WebDriver -Version 3.141.0
```

160 %

Visual Studio .NET

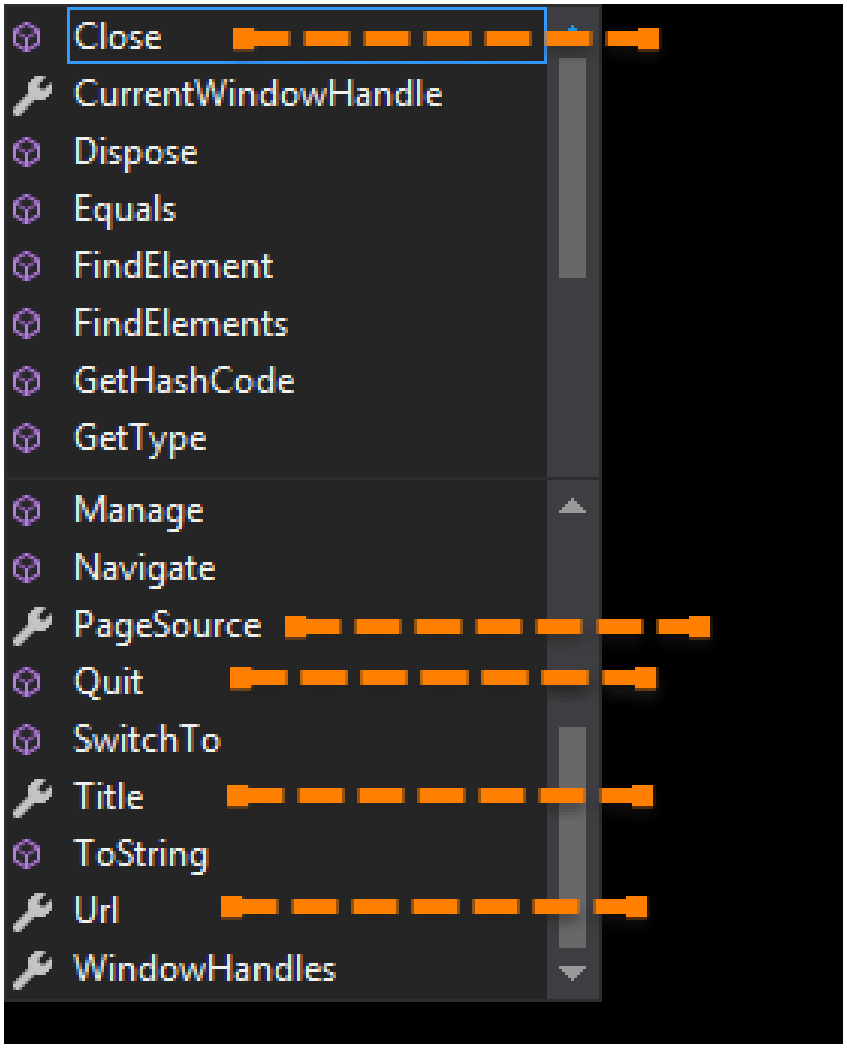


Getting Started with Selenium Web Driver

Selenium WebDriver Commands in C#:

- C# uses the interface 'IWebDriver' for browser interactions. The following are the category of commands available in C#.
- Browser commands
- Web Element commands
- Dropdown commands

IWebDriver Browser Commands in C#



IWebDriver Browser Commands in C#

Return Type

Method Name

```
void IWebDriver.Close()
Close the current window, quitting the browser if it is the last window currently open.
```

Method Description

Return Type is String

```
string IWebDriver.PageSource { get; }
Gets the source of the page last loaded by the browser.
```

This method returns & set value

```
string IWebDriver.Url { get; set; }
Gets or sets the URL the browser is currently displaying.
```

SELENIUM TESTING

```
30          webDriver.Url = test_url;
31      }
32      [Test]
        0 references
33      public void TestTitle()
34      {
35          String                                  ealth, Travel &amp; " +
36                                                  oyal Sundaram";
37          webDri
38          String
39          Assert                                  ;
40
41      }
42
43      [TearDown]
        0 references
44      public voi
```

- Quick Actions and Refactorings... Ctrl+.
- Rename... Ctrl+R, Ctrl+R
- Remove and Sort Usings Ctrl+R, Ctrl+G
- View Code F7
- Peek Definition Alt+F12
- Go To Definition F12
- Go To Base Alt+Home
- Go To Implementation Ctrl+F12
- Find All References Shift+F12
- View Call Hierarchy Ctrl+K, Ctrl+T
- Run Test(s) Ctrl+R, T
- Debug Test(s) Ctrl+R, Ctrl+T
- Create Unit Tests
- Breakpoint
- Run To Cursor Ctrl+F10
- Execute in Interactive Ctrl+E, Ctrl+E
- Snippet
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Annotation
- Outlining

```
The thread 0x51a8 has exited with code 0 (0x0).
The program '[2792] testhost.x86.exe' has exited with code
```

Selenium WebDriver Commands in C#
Browser commands

Browser commands:

Command Name	Description	Syntax
Url Command	This command is used to open a specified URL in the browser.	driver.Url = "https://www.demotour.com"
Title Command	This command is used to retrieve the page title of the web page that is currently open	String title = driver.Title
PageSource Command	This command is used to retrieve the source code of web page that is currently open.	String pageSource = driver.PageSource

Selenium WebDriver Commands in C#
Browser commands:

Close Command	This command is used to close the recently opened browser instance.	driver.Close();
Quit Command	This command is used to close all open browser instances	driver.Quit();
Back Command	This command is used to navigate to the previous page of browser history.	driver.Navigate().Back();

Selenium WebDriver Commands in C#: Browser commands:

Forward Command	This command is used to navigate to the next page of browser history.	driver.Navigate().Forward ()
Refresh Command	This command is used to perform browser refresh.	driver.Navigate().Refresh ()

Selenium Web driver commands in c#
Browser commands:

```
static void Main(string[] args)
    {
        IWebDriver driver = new
InternetExplorerDriver(@"C:\Users\abc\Desktop\Server");
        driver.Navigate().GoToUrl("https://demoqa.com");


        driver.FindElement(By.XPath(".//*[@id='menu-item-
374']/a")).Click();
        driver.Navigate().Back();
        driver.Navigate().Forward();
        driver.Navigate().Refresh();
        driver.Close();
    }
```

Selenium WebDriver Commands in C# Webelement Commands

Command Name	Description	Syntax
Click command	This command is used to click on a Webelement. For the element to be clickable, the element must be visible on the webpage. This command is used for checkbox and radio button operations as well.	IWebelement element = driver.FindElement(By.xpath("x path of Webelement")); element.Click();
Clear command	This command is specifically used for clearing the existing contents of textboxes.	IWebelement element = driver.FindElement(By.xpath("x path of Webelement")); element.Clear();

Selenium WebDriver Commands in C#
Webelement Commands

SendKeys command	This command is used to input a value onto text boxes. The value to be entered must be passed as a parameter to	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); element.SendKeys("guru99");
Displayed command	This command is used to identify if a specific element is displayed on the webpage. This command returns a Boolean value; true or false depending on the visibility of web element.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); Boolean status = element.Displayed;
Enabled command	This command is used to identify if a particular web element is enabled on the web page. This command returns a Boolean value; true or false as a result.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); Boolean status = element.Enabled;

Selenium WebDriver Commands in C#: Webelement Commands

Selected command	This command is used to identify if a particular web element is selected. This command is used for checkboxes,radio buttons, and select operations.	IWebelement element = driver.FindElement(By.xpath(" xpath of Webelement")); Boolean status = element.Selected;
Submit command:	This command is similar to click command, The difference lies in whether the HTML form has a button with the type Submit. While the click command clicks on any button, submit command clicks on the only the buttons with type submit.	IWebelement element = driver.FindElement(By.xpath(" xpath of Webelement")); element.submit();
Text command	This command returns the innertext of a Webelement. This command returns a string value as a result.	IWebelement element = driver.FindElement(By.xpath(" xpath of Webelement")); String text=element.Text;

Selenium WebDriver Commands in C# Webelement Commands

Selected command	This command is used to identify if a particular web element is selected. This command is used for checkboxes,radio buttons, and select operations.	IWebelement element = driver.FindElement(By.xpath(" xpath of Webelement")); Boolean status = element.Selected;
Submit command:	This command is similar to click command, The difference lies in whether the HTML form has a button with the type Submit. While the click command clicks on any button, submit command clicks on the only the buttons with type submit.	IWebelement element = driver.FindElement(By.xpath(" xpath of Webelement")); element.submit();
Text command	This command returns the innertext of a Webelement. This command returns a string value as a result.	IWebelement element = driver.FindElement(By.xpath(" xpath of Webelement")); String text=element.Text;

Web Element Example

```
WebDriver driver = new FirefoxDriver();
        // Launch the ToolsQA WebSite
        driver.Url = ("http://toolsqa.com/Automation-practice-form/");

        // Type Name in the FirstName text box
        driver.FindElement(By.Name("firstname")).SendKeys("Lakshay");

        //Type LastName in the LastName text box
        driver.FindElement(By.Name("lastname")).SendKeys("Sharma");

        // Click on the Submit button
        driver.FindElement(By.Id("submit")).Click();
```

Web Element Example

```
// Create a new instance of the FireFox driver
        IWebDriver driver = new FirefoxDriver();
        // Launch the Online Store WebSite
        driver.Url = ("http://toolsqa.com/Automation-practice-form/");
         // Click on "Partial Link Text" link
        driver.FindElement(By.PartialLinkText("Partial")).Click();
        Console.WriteLine("Partial Link Test Pass");
         // Convert element in to a string
        String sClass = driver.FindElements(By.TagName("button")).ToString();
        Console.WriteLine(sClass);
         // Click on "Link Text" link
        driver.FindElement(By.LinkText("Link Test")).Click();
        Console.WriteLine("Link Test Pass");
```

CheckBox & Radio Button Operations in selenium C#

```
IList <IWebElement> oCheckBox = driver.FindElements(By.Name("tool"));
  // This will tell you the number of checkboxes are present
 int Size = oCheckBox.Count;
  // Start the loop from first checkbox to last checkboxe
 for (int i = 0; i < Size; i++) {
 // Store the checkbox name to the string variable, using 'Value' attribute
 String Value = oCheckBox.ElementAt(i).GetAttribute("value");
  // Select the checkbox it the value of the checkbox is same what you are looking for
 if (Value.Equals("toolsqa")) {
 oCheckBox.ElementAt(i).Click();
 // This will take the execution out of for loop
 break;}}
```

Selenium WebDriver Commands in C# Dropdown Commands

Command Name	Description	Syntax
SelectByText Command	This command selects an option of a dropdown based on the text of the option.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.SelectByText("Guru99");
SelectByIndex Command	This command is used to select an option based on its index. Index of dropdown starts at 0.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.SelectByIndex("4");

Selenium WebDriver Commands in C# Dropdown Commands

Command Name	Description	Syntax
SelectByValue Command	This command is used to select an option based on its option value.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.SelectByValue("Guru99");

Selenium WebDriver Commands in C# Dropdown Commands

Options Command	This command is used to retrieve the list of options displayed in a dropdown.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); List<IWebelement> options = select. Options; int size = options.Count; for(int i=0;i<options.size();i++) { String value = size.elementAt(i).Text; Console.writeLine(value); }

Selenium WebDriver Commands in C# Dropdown Commands

IsMultiple command	This command is used to identify if a dropdown is a multi select dropdown; A multi select dropdown enables user to select more than one option in a dropdown at a time. This command returns a Boolean value.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); Boolean status = select.IsMultiple();
DeSelectAll command	This command is used in multi select dropdowns. It clears the options that have already been selected.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.DeSelectAll();
DeSelectByIndex command	This command deselects an already selected value using its index.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.DeSelectByIndex("4");

Selenium WebDriver Commands in C# Dropdown Commands

DeSelectByValue command	This command deselects an already selected value using its value.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.DeSelectByValue("Guru99");
DeSelectByText command	This command deselects an already selected value using its text.	IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); SelectElement select = new SelectElement(element); select.DeSelectByText("Guru99");

Drop down list

- // Create a new instance of the Firefox driver
- IWebDriver driver = new FirefoxDriver();
- // Put an Implicit wait, this means that any search for elements on the page could take the time the implicit wait is set for before throwing exception
- driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(10));
- // Launch the URL
- driver.Url= "http://toolsqa.com/automation-practice-form";
- // Step 3: Select 'Selenium Commands' Multiple select box (Use Name locator to identify the element)
- SelectElement oSelection = new SelectElement(driver.FindElement(By.Name("selenium_commands")));
- // Step 4: Select option 'Browser Commands' and then deselect it (Use selectByIndex and deselectByIndex)
- oSelection.SelectByIndex(0);
- Thread.Sleep(2000);
- oSelection.DeselectByIndex(0);
-

Drop down list

```
 // Step 5: Select option 'Navigation Commands'  and then deselect it (Use selectByVisibleText and deselectByVisibleText)
oSelection.SelectByText("Navigation Commands");
Thread.Sleep(2000);
oSelection.DeselectByText("Navigation Commands");
 // Step 6: Print and select all the options for the selected Multiple selection list.
IList <IWebElement> oSize = oSelection.Options;
int iListSize = oSize.Count;
 // Setting up the loop to print all the options
for (int i = 0; i < iListSize; i++)   {
    // Storing the value of the option
    String sValue = oSelection.Options.ElementAt(i).Text;
    // Printing the stored value
```

Drop down list

```
Console.WriteLine("Value of the Item is :" + sValue);
            // Selecting all the elements one by one
            oSelection.SelectByIndex(i);

            Thread.Sleep(2000);
        }

        // Step 7: Deselect all
        oSelection.DeselectAll();

        // Kill the browser
        driver.Close();
```

Explicit Wait

- An explicit wait in Selenium with a timeout of 10 seconds is set using the WebDriverWait class.
- WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
- IWebElement SearchResult = wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementExists(By.XPath(target_xpath)));

Fluent Wait

- DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);
- fluentWait.Timeout = TimeSpan.FromSeconds(5);
- fluentWait.PollingInterval = TimeSpan.FromMilliseconds(250);
- /* Ignore the exception - NoSuchElementException that indicates that the element is not present */
- fluentWait.IgnoreExceptionTypes(typeof(NoSuchElementException));
- fluentWait.Message = "Element to be searched not found";
-
- driver.Url = test_url;
- driver.FindElement(By.Name("q")).SendKeys("LambdaTest" + Keys.Enter);
-

Fluent Wait

- /* Explicit Wait */
- /*
- WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
- IWebElement SearchResult = wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementExists(By.XPath(target_xpath))); */
- IWebElement searchResult = fluentWait.Until(x => x.FindElement(By.XPath(target_xpath)));
- searchResult.Click();

WebDriver API

- The WebDriver API is the part of the system that you interact with all the time.
- Things have changed from the 140 line long API that the Selenium RC API had.
- This is now more manageable and can actually fit on a normal screen.
- This is made up of the WebDriver and the WebElement objects.
- driver.findElement(By.name("q"))
- and
- element.sendKeys("I love cheese");
- These commands are then translated to the SPI, which is stateless.

Common Exceptions in Selenium Web driver

Exception name	Description
ElementNotVisibleException	This type of Selenium exception occurs when an existing element in DOM has a feature set as hidden.
ElementNotSelectableException	This Selenium exception occurs when an element is presented in the DOM, but you can be able to select. Therefore, it is not possible to interact.
NoSuchElementException	This Exception occurs if an element could not be found.
NoSuchFrameException	This Exception occurs if the frame target to be switched to does not exist.
NoAlertPresentException	This Exception occurs when you switch to no presented alert.

Common Exceptions in Selenium Web driver

Exception name	Description
NoSuchWindowException	This Exception occurs if the window target to be switch does not exist.
StaleElementReferenceException	This Selenium exception occurs happens when the web element is detached from the current DOM.
SessionNotFoundException	The WebDriver is acting after you quit the browser.
TimeoutException	Thrown when there is not enough time for a command to be completed. For Example, the element searched wasn't found in the specified time.
WebDriverException	This Exception takes place when the WebDriver is acting right after you close the browser.

Common Exceptions in Selenium Web driver

Exception name	Description
ConnectionClosedException	This type of Exception takes place when there is a disconnection in the driver.
ElementClickInterceptedException	The command may not be completed as the element receiving the events is concealing the element which was requested clicked.
ElementNotInteractableException	This Selenium exception is thrown when any element is presented in the DOM. However, it is impossible to interact with such an element.
ErrorInResponseException	This happens while interacting with the Firefox extension or the remote driver server.
ErrorHandler.UnknownServerException	Exception is used as a placeholder in case if the server returns an error without a stack trace.

Common Exceptions in Selenium Web driver

Exception name	Description
InvalidArgumentException	It occurs when an argument does not belong to the expected type.
InvalidCookieDomainException	This happens when you try to add a cookie under a different domain instead of current URL.
InvalidCoordinatesException	This type of Exception matches an interacting operation that is not valid.
InvalidElementStateExceptio	It occurs when command can't be finished when the element is invalid.
InvalidSessionIdException	This Exception took place when the given session ID is not included in the list of active sessions. It means the session does not exist or is inactive either.

Common Exceptions in Selenium Web driver

Exception name	Description
NoSuchAttributeException	This kind of Exception occurs when the attribute of an element could not be found.
MoveTargetOutOfBounds Exception	It takes place if the target provided to the ActionChains move() methodology is not valid. For Example, out of the document.
NoSuchContextException	ContextAware does mobile device testing.
NoSuchCookieException	This Exception occurs when no cookie matching with the given pathname found for all the associated cookies of the currently browsing document.
NotFoundException	This Exception is a subclass of WebDriverException. This will occur when an element on the DOM does not exist.

Common Exceptions in Selenium Web driver

Exception name	Description
ScreenshotException	It is not possible to capture a screen.
SessionNotCreatedException	It happens when a new session could not be successfully created.
UnableToSetCookieException	This occurs if a driver is unable to set a cookie.
UnexpectedTagNameException	Happens if a support class did not get a web element as expected.
UnhandledAlertException	This expectation occurs when there is an alert, but WebDriver is not able to perform Alert operation.

Common Exceptions in Selenium Web driver

Exception name	Description
UnreachableBrowserException	This Exception occurs only when the browser is not able to be opened or crashed because of some reason.
UnsupportedCommandException	This occurs when remote WebDriver does n't send valid commands as expected.

Regular Expression

[abc]	A single character of: a, b, or c	.	Any single character
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character
[a-z]	Any single character in the range a-z	\S	Any non-whitespace character
[a-zA-Z]	Any single character in the range a-z or A-Z	\d	Any digit
^	Start of line	\D	Any non-digit
$	End of line	\w	Any word character (letter, number, underscore)
\A	Start of string	\W	Any non-word character
\z	End of string	\b	Any word boundary

(...)	Capture everything enclosed
(a|b)	a or b
a?	Zero or one of a
a*	Zero or more of a
a+	One or more of a
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{3,6}	Between 3 and 6 of a

Regular Expression

- Pattern Match
- . any single character
- [ ] character class: any single character that appears inside the brackets
- * quantifier: 0 or more of the preceding character (or group)
- + quantifier: 1 or more of the preceding character (or group)
- ? quantifier: 0 or 1 of the preceding character (or group)
- {1,5} quantifier: 1 through 5 of the preceding character (or group)
- | alternation: the character/group on the left or the character/group on the right
- ( ) grouping: often used with alternation and/or quantifier
- Regular expression patterns in Selenese need to be prefixed with either regexp: or regexpi:. The former is case-sensitive; the latter is case-insensitive.

Regular Expression

- Command Target Value
- click link=regexp:Film.*Television Department
- verifyTitle regexp:.*Film.*Television.*

Regular Expression

- Command Target Value
- open http://weather.yahoo.com/forecast/USAK0012.html
- verifyTextPresent regexp:Sunrise: *[0-9]{1,2}:[0-9]{2} [ap]m

Regular Expression

- Sunrise: * The string Sunrise: followed by 0 or more spaces
- [0-9]{1,2} 1 or 2 digits (for the hour of the day)
- : The character : (no special characters involved)
- [0-9]{2} 2 digits (for the minutes) followed by a space
- [ap]m “a” or “p” followed by “m” (am or pm)

Module Summary

- Lambdas
- Stream API
- Base 64
- Nashorn
- Reflection
- Date Time API
- Repeated Annotations
- JDBC Improvements