

Selenium IDE



Steve Kwon, Raphael Huang, Amad Hussain, Mubasil Shamim

Introduction

- **Selenium** is a portable software-testing framework for web applications
- Selenium IDE is a complete integrated development environment
- Implemented as a Firefox extension
- Selenium IDE includes the entire Selenium Core, can allow
 - recording
 - Editing
 - and debugging tests
- You can choose to use its recording capability,
- or you may edit your scripts by hand

Benefits

- Selenium IDE is very easy to use
- Convert the test to different programming languages
- User is not required to need any programming knowledge
- User can debug and set breakpoints

Disadvantages

- Selenium IDE is Firefox plugin, thus its support is limited to Firefox only
- Selenium IDE doesn't support error handling
- Selenium IDE doesn't support test script grouping
- Selenium IDE doesn't support Database testing

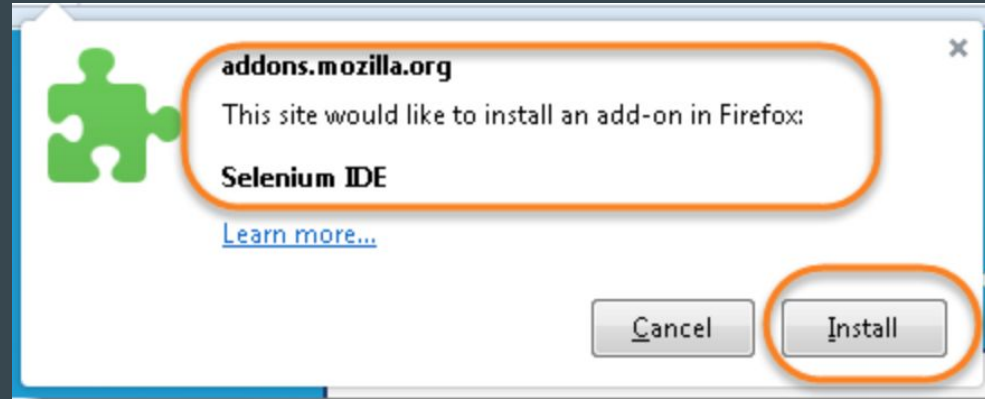
Download and Install Selenium IDE

- Launch Mozilla Firefox Browser.
- Type URL: <https://addons.mozilla.org/en-us/firefox/addon/selenium-ide/>



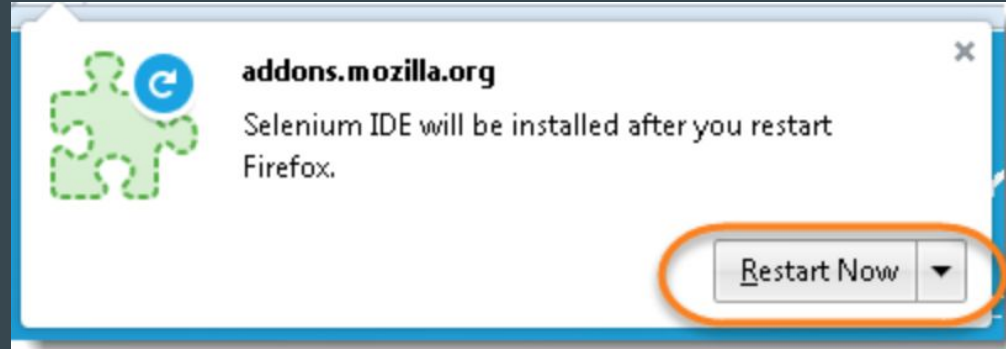
Download and Install Selenium IDE

- Firefox will show do you want to allow Mozilla Firefox to install Selenium IDE Add-ons or not.
- Click on Install button as Shown this image.



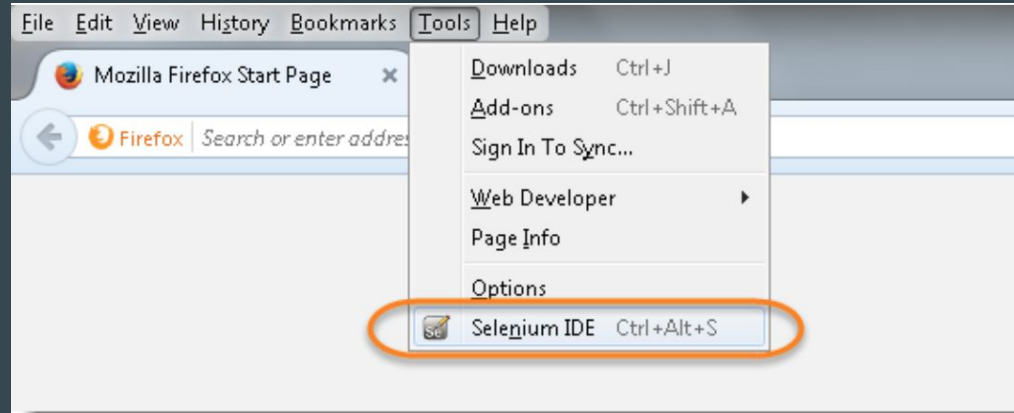
Download and Install Selenium IDE (continued)

- Firefox will automatically install Selenium IDE software.
- After the installation is completed, restart the Firefox.
- Click on the “Restart Now” button to reflect the Selenium IDE installation.



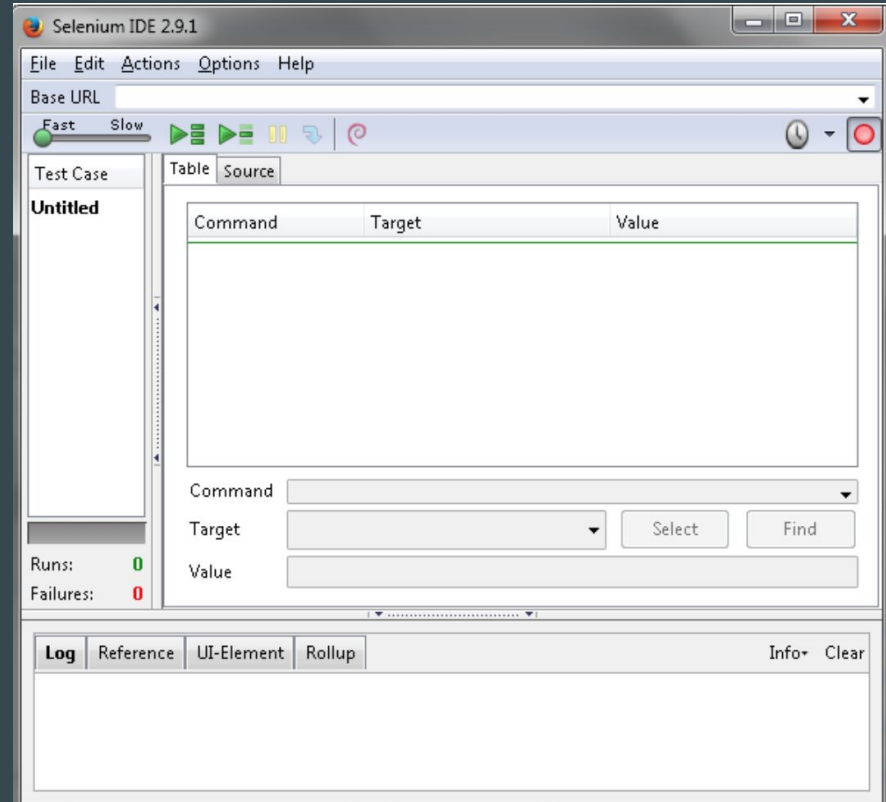
Download and Install Selenium IDE (continued)

- Once the Firefox is booted and started again, we can see selenium IDE under the tools menu list.
- Click on Tools menu list, Selenium IDE will be displayed in the list.



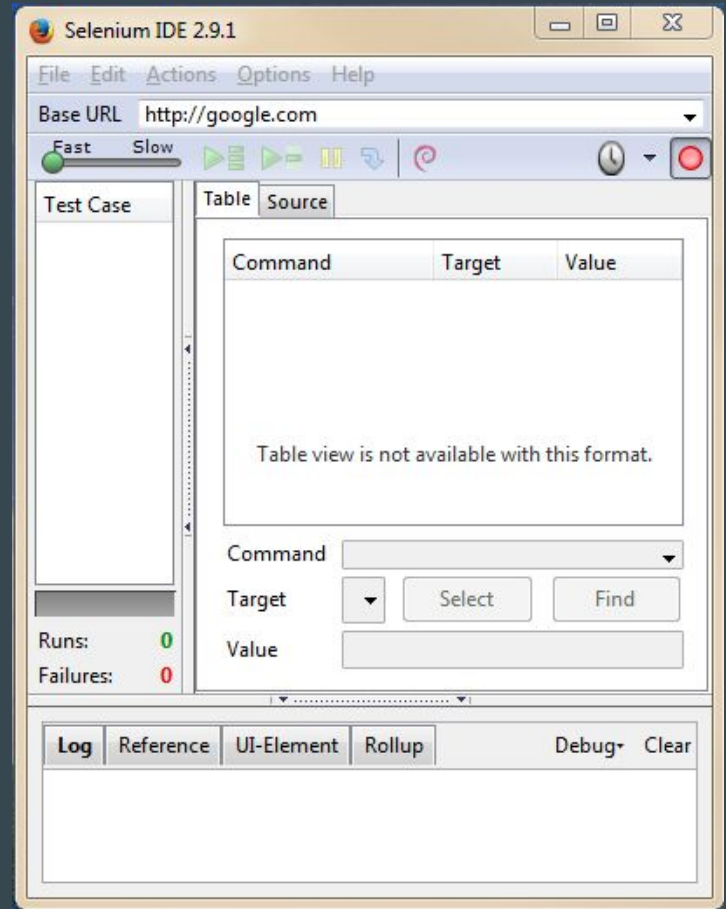
Download and Install Selenium IDE (continued)

- Click on Selenium IDE, it will launch Selenium IDE

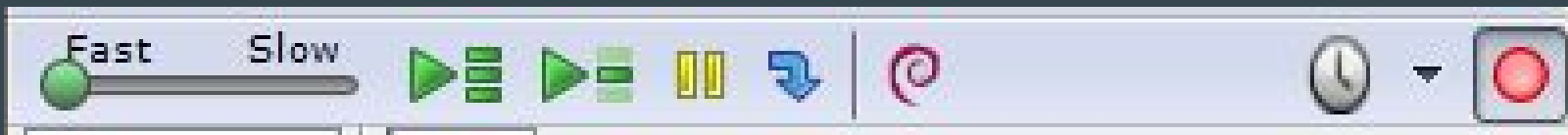


Opening the IDE

- To run the Selenium IDE, simply select it from the Firefox Tool menu.
- It opens with an empty script-editing window and a menu for loading, or creating new test cases.

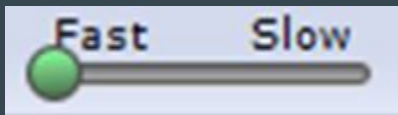


Toolbar



- The toolbar contains buttons for controlling the execution of your test cases, including a step features for debugging your test cases.
- The right-most button, the one with the red dot, is the record button.

Toolbar Elements



- 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.

Toolbar Elements (continued)



- Step: Allows you to “step” through a test case by running it one command at a time. It is useful 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.

Toolbar Element in Selenium IDE v2.9 (the most recent ver.)



- The Selenium IDE Scheduler works with jobs. A job contains information about what the scheduler should do. It contains a title, it contains the test suite that should be automatically run or played and it contains the schedule information, which is the time when the test suite should be run.

Commands (Selenese)

- Test Script: Sequence of commands used to test an application
- Can test:
 - Existence of UI elements based on HTML tags
 - Specific content
 - Broken Links
 - Input Fields
 - Selection list options
 - Submitting forms
 - Table Data
 - Etc...
- 3 Command Types:
 - Actions
 - Accessor
 - Assertion

Command	<input type="text"/>		
Target	<input type="text"/>	Select	Find
Value	<input type="text"/>		

Actions

- Manipulate the state of the application
 - Type this box / Click this link / Select option
- Test is stopped on failure
- Used with the “AndWait” suffix
 - clickAndWait / typeAndWait
 - Notifies the browser that the action calls the server and Selenium should wait for the response

Accessors

- Examine the state of the application and store the result in variables
 - `storeTitle`

- Also used to automatically generate assertions

Assertions

- Verify that the state of the application matches expectation
- Types
 - Assert: Aborted on failure
 - Verify: Continues on failure and logs the error
 - WaitFor: Succeed when a condition becomes true or immediately if already true. Fail and halt if the condition does not become true within the timeout setting

Common Commands

- `type` - Sets value of an input field as if typed in
- `open` - Opens a page using a url
- `click` - Clicks a link, button checkbox, or radio button
- `select` - Selects an option from a drop-down using an option locator
- `selectFrame` - Selects a frame within a current window

Common Commands

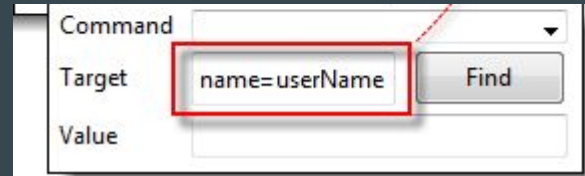
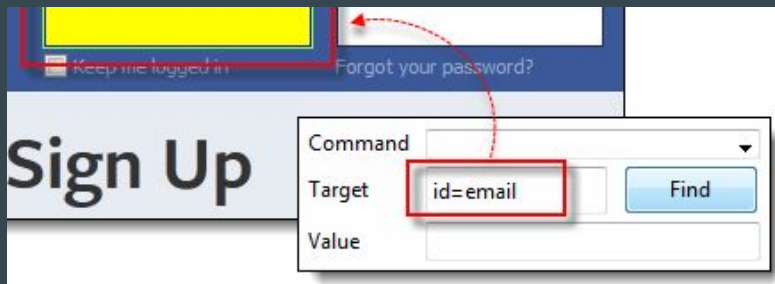
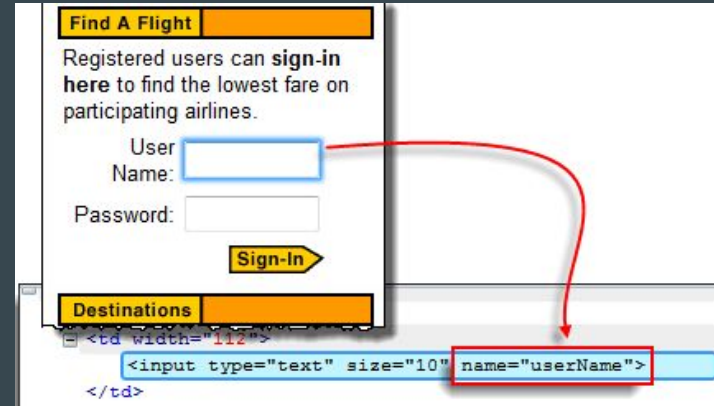
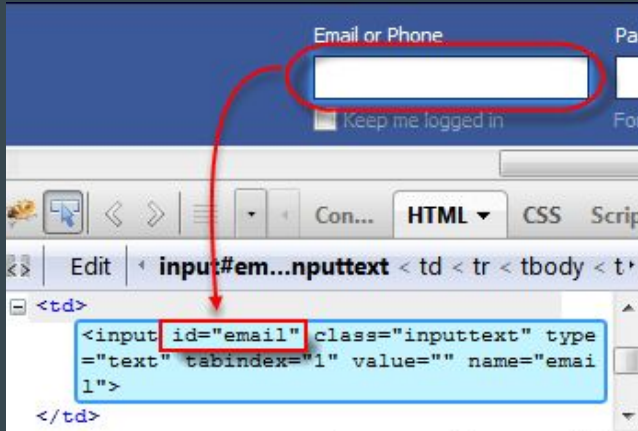
- `verifyTitle/assertTitle` - Verifies an expected page title
- `verifyTextPresent` - Verifies that a specified string appears somewhere on the rendered page
- `waitForPageToLoad` - Waits for new page to load
 - Use instead of the “andWait” suffix like with “clickAndWait” or “typeAndWait”
- `pause` - Wait for specified amount of time

Common Commands

- `highlight` - Briefly changes the `backgroundColor` of the specified element.
 - Useful for debugging
- `store/storeExpression` - sets a variable with some value to be used with other commands
- `echo` - prints specified message into the log area
 - Useful for debugging
- `refresh` - simulates user clicking the “refresh” button

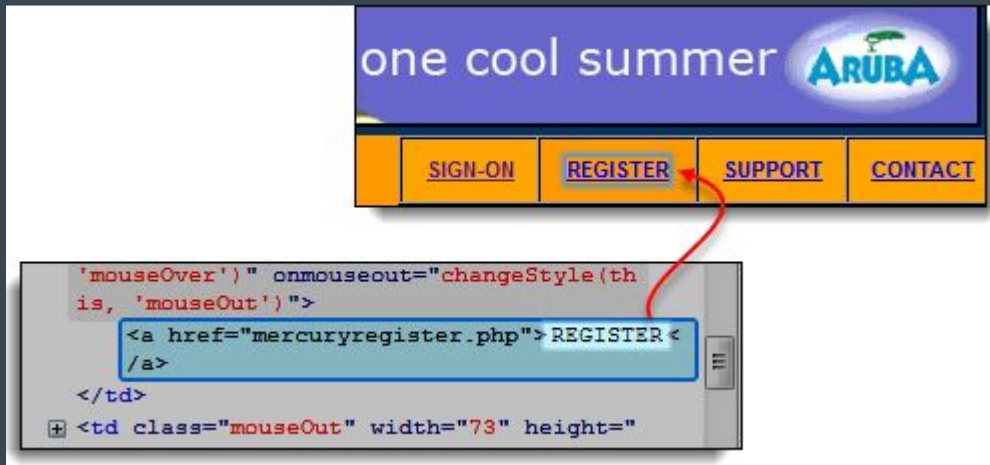
Locators - ID/Name

- Uses ID or Name html property to locate elements
- Defaults to first instance in the document if multiple matches



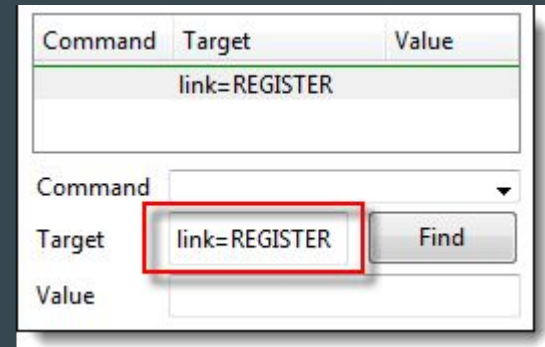
Locators - Link Text

- Finds the first instance of a hyperlink with specified text



The image shows a web page header with the text "one cool summer" and the "ARUBA" logo. Below the header is a navigation menu with four buttons: "SIGN-ON", "REGISTER", "SUPPORT", and "CONTACT". The "REGISTER" button is highlighted with a red arrow pointing to its source code in a code editor below. The code for the "REGISTER" button is:

```
'mouseover')" onmouseout="changeStyle(th  
is, 'mouseout')">  
<a href="mercuryregister.php">REGISTER <  
/a>  
</td>  
+ <td class="mouseout" width="73" height="
```



The image shows a Selenium IDE interface with a table of commands and their targets and values. The table has three columns: Command, Target, and Value. The first row contains the command "link=REGISTER". Below the table, there is a "Command" dropdown menu, a "Target" input field containing "link=REGISTER" (highlighted with a red box), a "Find" button, and a "Value" input field.

Command	Target	Value
	link=REGISTER	

Command

Target Find

Value

Locators - CSS

- Generalized way to select using HTML tags
- Uses the following levels of granularity for syntax:

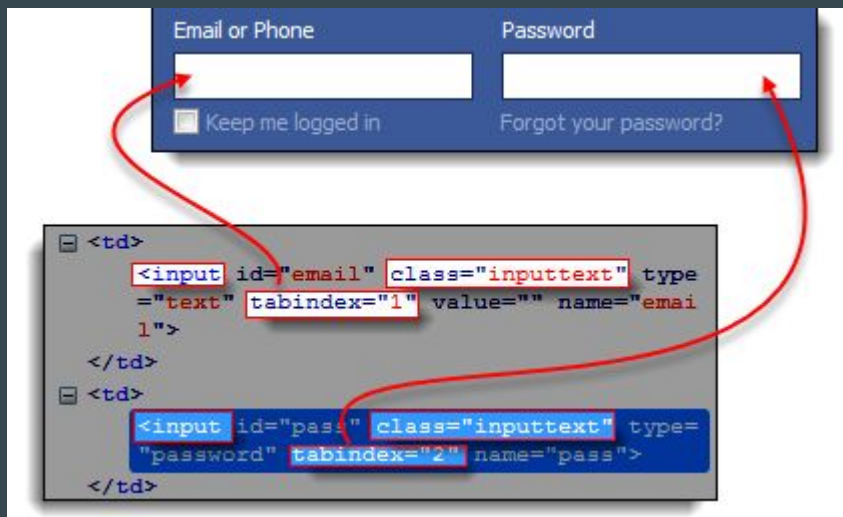
Find By	Target Syntax	Example
Tag and ID	css=tag#id	css=input#email
Tag and Class	css=tag.class	css=input.inputtext
Tag and Attribute	css=tag[attribute=value]	css=input[name=lastName]
Tag, Class, and Attribute	css=tag.class[attribute=value]	css=input.inputtext[tabindex=1]
Inner Text	css=tag:contains("inner text")	css=span:contains("Help")

Locators - CSS

Tag, Class, and Attribute

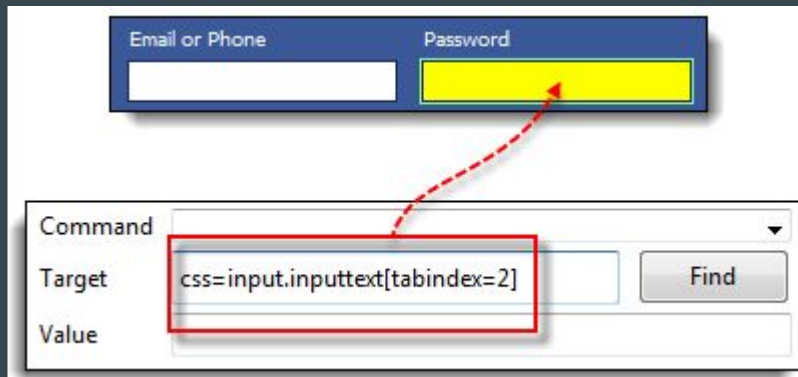
css=tag.class[attribute=value]

css=input.inputtext[tabindex=1]



The image shows a login form with two input fields: "Email or Phone" and "Password". Below the form is the HTML code for these fields. Red arrows point from the code to the form elements.

```
<td>  
<input id="email" class="inputtext" type="text" tabindex="1" value="" name="email" />  
</td>  
<td>  
<input id="password" class="inputtext" type="password" tabindex="2" name="password" />  
</td>
```



The image shows a search tool interface. The 'Target' field contains the CSS locator `css=input.inputtext[tabindex=2]`. A red dashed arrow points from the 'Password' field in the form above to the 'Target' field.

Command

Target `css=input.inputtext[tabindex=2]` Find

Value

Locators - DOM

- Use the Javascript document object to find elements/text on the page
- Useful for dynamically generated content

Preferences

Service Class: Economy class
 Business class
 First class

Airline: No Preference

```
<td>
  <font size="2">
    <input type="radio" value="Coach" name="servClass" checked="">
    <font face="Arial, Helvetica, sans-serif">
      Economy class
      <br>
      <input type="radio" value="Business" name="servClass">
      Business class
      <br>
      <input type="radio" value="First" name="servClass">
      First class
    </font>
  </font>
</td>
```

Command

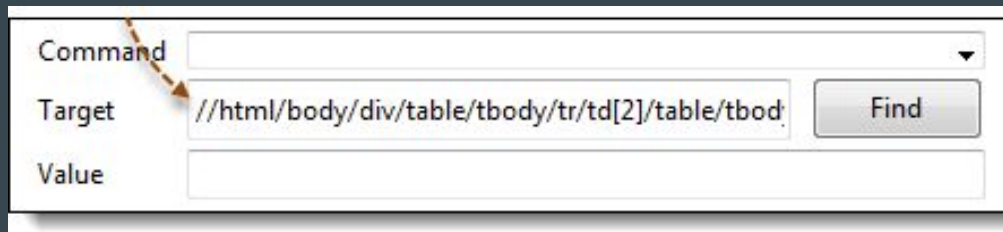
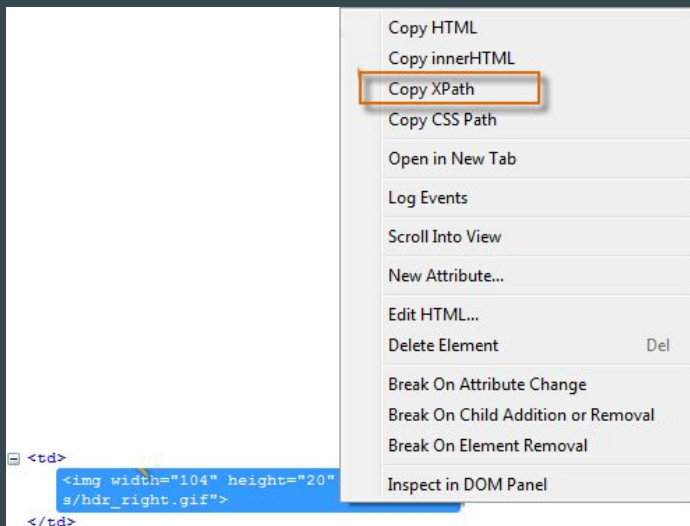
Target: `document.getElementsByName("servClass")[0]` Find

Value

Service Class: Economy class
 Business class
 First class

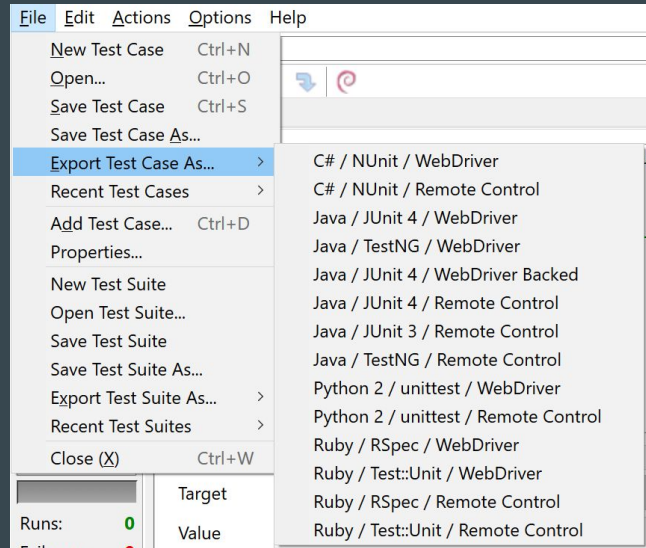
Locators - XPath

- If we represent the HTML as an XML Tree, XPath is the path to a specific element node.
- Easy to implement as long as your page structure does not change.
- Can use plugins such as Firebug (Firefox) or XPath Helper (Chrome) to generate



Converting to Other Programming Languages

- Test cases can be exported to Python, Java, etc, for use in integrated unit tests.
- Be sure to install the required Selenium dependencies using pip/maven
- WebDriver is generally more concise and better supported than Remote Control



Sources/More info

- <http://toolsqa.com/selenium-ide-tutorial/>
- <https://www.guru99.com/selenium-tutorial.html>
- <http://www.softwaretestinghelp.com/selenium-ide-script-selenium-tutorial-3/>