

# Selenium

Ramtin Bagheri

Amirhossein Kargaran

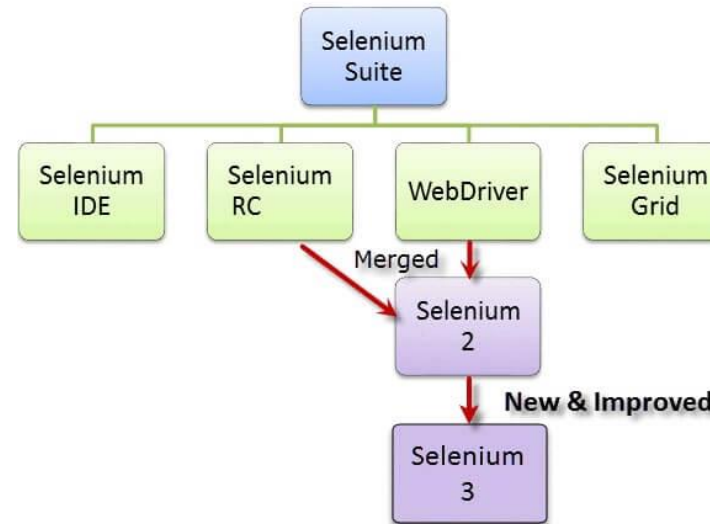
# What is selenium?



- ▶ Free (open-source) automated testing framework used to validate web applications across different browsers and platforms.
- ▶ Supports multiple programming languages like Java, C#, Python etc. to create Selenium Test Scripts.
- ▶ Selenium Software is not just a single tool.

# List of tools

- ▶ Selenium Remote Control (RC)
- ▶ Selenium Grid
- ▶ Selenium IDE
- ▶ WebDriver



- ▶ At the moment, Selenium RC and WebDriver are merged into a single framework to form **Selenium 2** and Improved to **Selenium3**. Selenium 1, by the way, refers to Selenium RC.

# Who developed Selenium?



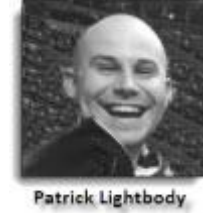
- ▶ Since Selenium is a collection of different tools, it had different developers as well.
- ▶ Primarily, Selenium was created by Jason Huggins in 2004.
- ▶ Having realized that the repetitious Manual Testing of their application was becoming more and more inefficient, he created a JavaScript program that would automatically control the browser's actions. He named this program as the "**JavaScriptTestRunner**."
- ▶ Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as **Selenium Core**.

# Selenium Remote Control (RC)



- ▶ Problem: Testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the **same origin policy**.
- ▶ Same origin policy: prohibits JavaScript code from accessing elements from a domain that is different from where it was launched and cant have a access to different domains.
- ▶ **Paul Hammant**, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the **Selenium Remote Control** or **Selenium 1**.

# Selenium Grid



- ▶ Problem: We need to minimize test execution times as much as possible.
- ▶ **Patrick Lightbody** address this problem by making a system called "**Hosted QA.**" It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously.**

# Selenium Integrated Development Environment (IDE)



- ▶ Problem: We need a Lightweight browser extension can automate the browser through a record-and-playback feature to increase the speed in creating test cases.
- ▶ **Shinya Kasatani** created **Selenium IDE**, a Firefox extension and donated Selenium IDE to the Selenium Project in **2006**.

# WebDriver



- ▶ Problem: We need a cross-platform testing framework that could control the browser from the OS level.
- ▶ **Simon Stewart** created WebDriver circa **2006** when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core.
- ▶ In **2008**, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called **Selenium 2**, with **WebDriver being the core**.

# How to Choose the Right Selenium Tool for Your Need

Tool	Why Choose
Selenium 3 (Web driver and RC)	<ul style="list-style-type: none"><li>• To use programming language in designing your test case.</li><li>• To test your application against any browser.</li></ul>
Selenium IDE	<ul style="list-style-type: none"><li>• To learn about concepts on automated testing and Selenium</li><li>• To create simple test cases and test suites that you can export later to RC or WebDriver.</li><li>• Just FireFox and Chrome</li></ul>
Selenium Grid	<ul style="list-style-type: none"><li>• To run your Selenium RC scripts in multiple browsers and operating systems simultaneously.</li><li>• To run a huge test suite, that needs to complete in the soonest time possible.</li></ul>

# Installing Selenium and Driver

- ▶ Download Driver: e.g. ChromeDriver
- ▶ Move Driver somewhere that Python and Selenium will be able to find it (a.k.a. in your PATH).
- ▶ `pip install selenium`
- ▶ Installing Browser : e.g. Chrome
- ▶ You also need to make sure Python can find your browser.
- ▶ Test with examples in next slides

# Validating a Test with Assert and Verify

- ▶ They are commands that verify if a certain condition is met:
  - ▶ **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.

# Locators

Locator	Description
1. find_element_by_id	The first element with the <b>id</b> attribute value matching the location will be returned.
2. find_element_by_name	The first element with the <b>name</b> attribute value matching the location will be returned.
3. find_element_by_xpath	The first element with the <b>xpath</b> syntax matching the location will be returned.
4. find_element_by_link_text	The first element with the <b>link text</b> value matching the location will be returned.
5. find_element_by_partial_link_text	The first element with the <b>partial link text</b> value matching the location will be returned.
6. find_element_by_tag_name	The first element with the given <b>tag name</b> will be returned.
7. find_element_by_class_name	the first element with the matching <b>class attribute name</b> will be returned.
8. find_element_by_css_selector	The first element with the matching <b>CSS selector</b> will be returned.

# 1. find\_element\_by\_id

For instance, consider this page source:

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

Selenium syntax:

```
login_form = driver.find_element_by_id('loginForm')
```

## 2. find\_element\_by\_name

For instance, consider this page source:

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

Selenium syntax:

```
login_form = driver.find_element_by_name('username')
```

### 3. find\_element\_by\_xpath

For instance, consider this page source:

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

Selenium syntax:

```
login_form = driver.find_element_by_xpath("/html/body/form[1]")
```

```
login_form = driver.find_element_by_xpath("//form[1]")
```

## 4. find\_element\_by\_link\_text

For instance, consider this page source:

```
<html>
<body>
  <p>Are you sure you want to do this?</p>
  <a href="continue.html">Continue</a>
  <a href="cancel.html">Cancel</a>
</body>
</html>
```

Selenium syntax:

```
login_form = driver.find_element_by_link_text('Continue')
```

## 5. find\_element\_by\_partial\_link\_text

For instance, consider this page source:

```
<html>
<body>
  <p>Are you sure you want to do this?</p>
  <a href="continue.html">Continue</a>
  <a href="cancel.html">Cancel</a>
</body>
</html>
```

Selenium syntax:

```
login_form = driver.find_element_by_partial_link_text('Conti')
```

## 6. find\_element\_by\_tag\_name

For instance, consider this page source:

```
<html>
<body>
  <h1>Welcome</h1>
  <p>Site content goes here.</p>
</body>
</html>
```

Selenium syntax:

```
login_form = driver.find_element_by_tag_name('h1')
```

## 7. find\_element\_by\_class\_name

For instance, consider this page source:

```
<html>  
<body>  
  <p class="content">Site content goes here.</p>  
</body>  
</html>
```

Selenium syntax:

```
content = driver.find_element_by_class_name('content')
```

## 8. find\_element\_by\_css\_selector

For instance, consider this page source:

```
<html>  
<body>  
  <p class="content">Site content goes here.</p>  
</body>  
</html>
```

Selenium syntax:

```
content = driver.find_element_by_css_selector('p.content')
```

# Practical Time

- ▶ We will see 4 practical examples:
  - ▶ [01\\_simple\\_usages.py](#)
  - ▶ [02\\_using\\_unittest.py](#)
  - ▶ [03\\_explicit\\_waiting.py](#)
  - ▶ [04\\_headless\\_example.py](#)
  - ▶ [05\\_action\\_chain.py](#)

# Quiz: Automate User Registration Process

- ▶ Please upload your answers at <https://forms.gle/MWpLKWGxTcST7Ujj9>
  
- ▶ Steps to Automate:
  1. Open a signup URL (without captcha)
  2. Enter your Personal Information
  3. Click on Register button.
  4. Validate that user is created.

# Source

- ▶ History: <https://www.guru99.com/introduction-to-selenium.html>
- ▶ Examples are accessible at: <https://github.com/R4MT1N/selenium-walkthrough>
- ▶ More: <https://www.geeksforgeeks.org/tag/selenium/>