

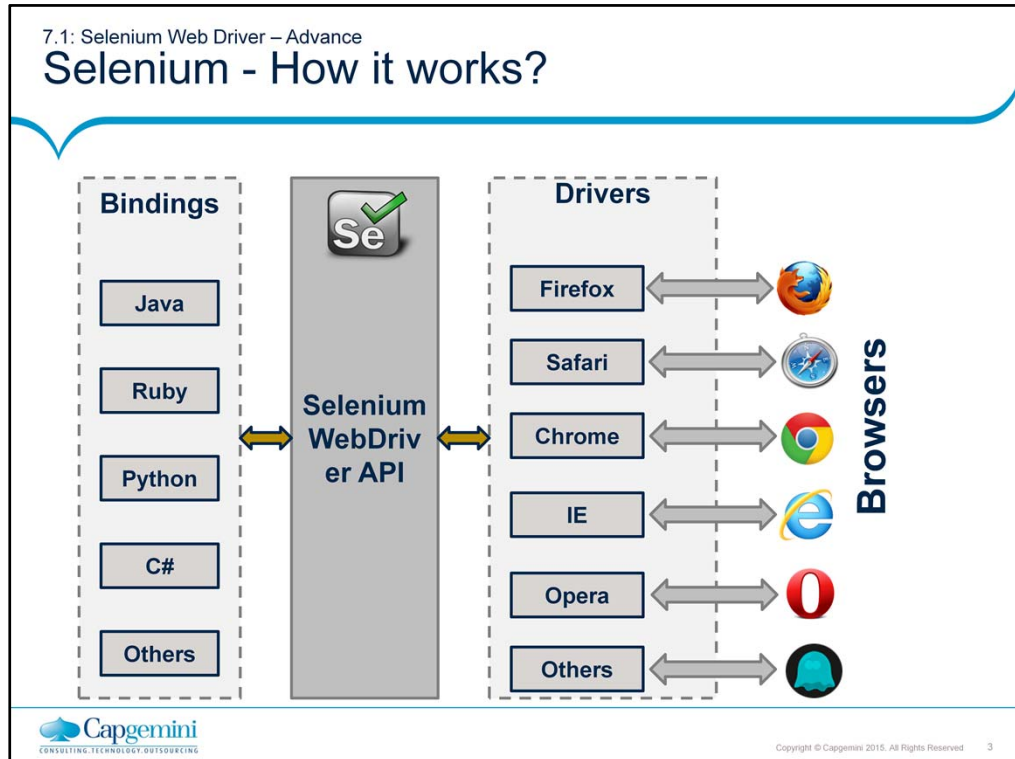
Test Automation & Advanced Selenium

Lesson 7: Selenium Web
Driver – Advance

Lesson Objectives

- Selenium: How it works
- Different drivers
 - Chrome
 - Firefox
 - Internet Explorer
 - Headless Browser
 - Ghost Driver and Phantom JS
 - Mobile Browsers
 - Selendroid & Appium
 - Remote Web Driver
 - Capabilities
 - Profile setting
 - Selenium Grid





7.1: Selenium Web Driver – Advance

Different Drivers

Firefox:

```
WebDriver driver = new FirefoxDriver();
```

IE:

```
WebDriver driver = new InternetExplorerDriver();
```

Chrome:

```
WebDriver driver = new ChromeDriver();
```

Safari:

```
WebDriver driver = new SafariDriver();
```

Opera:

```
WebDriver driver = new OperaDriver();
```

GhostDriver and PhantomJs:

```
WebDriver driver = new PhantomJSDriver();
```



Copyright © Capgemini 2015. All Rights Reserved

4

1. Architecture

WebDriver's architecture is simpler than Selenium RC's.

It controls the browser from the OS level

All you need are your programming language's IDE (which contains your Selenium commands) and a browser.

You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing

The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser

When you begin testing, Selenium RC Server "injects" a Javascript program called Selenium Core into the browser.

Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.

When the instructions are received, Selenium Core will execute them as Javascript commands.

The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.

The RC Server will receive the response of the browser and then display the results to you.

RC Server will fetch the next instruction from your test script to repeat the whole cycle.

2. Speed

WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

- Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.

3. Real-life Interaction

WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

- Selenium Core, just like other Javascript codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests. Differences in API

4. API

Selenium RC's API is more matured but contains redundancies and often confusing commands.

- For example, most of the time, testers are confused whether to use type or typeKeys; or whether to use click, mouseDown, or mouseDownAt. Worse, different browsers interpret each of these commands in different ways too
- WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.

5. Browser Support

WebDriver can support the headless HtmlUnit browser

- HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.
- It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.
- Since it is invisible to the user, it can only be controlled through automated means.
- Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

■ Firefox Driver:

```
1 // Create a new instance of the Firefox driver
2 WebDriver driver=new FirefoxDriver();
3 // opening Google
4 driver.get("https://www.google.com/");
5 // Closing driver
6 driver.close();
```

■ IE Driver:

```
1 //Setting system property for IE driver
2 System.setProperty("webdriver.ie.driver", "/path/to/IEDriver");
3 // Create a new instance of the IE driver
4 WebDriver driver=new InternetExplorerDriver();
5 // opening Google
6 driver.get("https://www.google.com/");
7 // Closing driver
8 driver.close();
```

■ Chrome Driver:

```
1 //Setting system property for Chrome driver
2 System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
3 // Create a new instance of the Chrome driver
4 WebDriver driver=new ChromeDriver();
5 // opening Google
6 driver.get("https://www.google.com/");
7 // Closing driver
8 driver.close();
```



Copyright © Capgemini 2015. All Rights Reserved 6

1. Architecture

WebDriver's architecture is simpler than Selenium RC's.

It controls the browser from the OS level

All you need are your programming language's IDE (which contains your Selenium commands) and a browser.

You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing

The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser

When you begin testing, Selenium RC Server "injects" a Javascript program called Selenium Core into the browser.

Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.

When the instructions are received, Selenium Core will execute them as Javascript commands.

The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.

The RC Server will receive the response of the browser and then display the results to you.

RC Server will fetch the next instruction from your test script to repeat the whole cycle.

2. Speed

WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

- Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.

3. Real-life Interaction

WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

- Selenium Core, just like other Javascript codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests. Differences in API

4. API

Selenium RC's API is more matured but contains redundancies and often confusing commands.

- For example, most of the time, testers are confused whether to use `type` or `sendKeys`; or whether to use `click`, `mouseDown`, or `mouseDownAt`. Worse, different browsers interpret each of these commands in different ways too
- WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.

5. Browser Support

WebDriver can support the headless HtmlUnit browser

- HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.
- It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.
- Since it is invisible to the user, it can only be controlled through automated means.
- Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

■ Headless Browser

- Web browser without a graphical user interface
- Normally, interaction with a website are done with mouse and keyboard using a browser with a GUI
- While most headless browser provides an API to manipulate the page/DOM, download resources etc.
- So instead of, for example, actually clicking an element with the mouse, a headless browser allows you to click an element by code
- Headers, Local storage and Cookies work the same way
- List of Headless Browsers
 - PhantomJS
 - HtmlUnit
 - TrifleJS
 - Splash



Copyright © Capgemini 2015. All Rights Reserved 8

Headless browsers are typically used in following situations:

1. You have a central build tool which does not have any browser installed on it. So to do the basic level of sanity tests after every build you may use the headless browser to run your tests.
2. You want to write a crawler program that goes through different pages and collects data, headless browser will be your choice. Because you really don't care about opening a browser. All you need is to access the webpages.
3. You would like to simulate multiple browser versions on the same machine. In that case you would want to use a headless browser, because most of them support simulation of different versions of browsers. We will come to this point soon.

Things to pay attention to before using headless browser

Headless browsers are simulation programs, they are not your real browsers. Most of these headless browsers have evolved enough to simulate, to a pretty close approximation, like a real browser. Still you would not want to run all your tests in a headless browser. JavaScript is one area where you would want to be really careful before using a Headless browser. JavaScript are implemented differently by different browsers. Although JavaScript is a standard but each browser has its own little differences in the way that they have implemented JavaScript. This is also true in case of headless browsers also. For example HtmlUnit headless browser uses the Rihno JavaScript engine which not being used by any other browser.

Selenium support for headless browser

Selenium supports headless testing using its class called HtmlUnitDriver. This class internally uses HtmlUnit headless browser. HtmlUnit is a pure Java implementation. HtmlUnitWebDriver can be created as mentioned below:

```
HtmlUnitDriver unitDriver = new HtmlUnitDriver();
```


7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

- PhantomJS (Headless Browser)

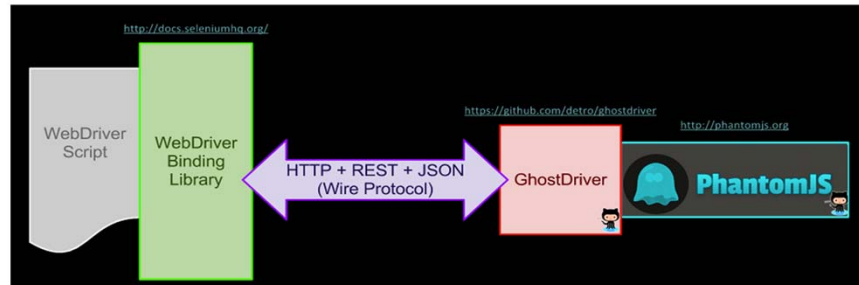
- HEADLESS WEBSITE TESTING
 - Headless Web Kit with JavaScript API
- SCREEN CAPTURE
 - Programmatically capture web contents, including SVG and Canvas
- PAGE AUTOMATION
 - Access and manipulate webpages with the standard DOM API, or with usual libraries like jQuery
- Example of interacting with a page using PhantomJS:

```
page.evaluate(function() {  
    //Fill in form on page  
    document.getElementById('Name').value = 'John Doe';  
    document.getElementById('Email').value = 'john.doe@john.doe';  
    //Submit  
    $('#SubmitButton').click();  
});
```

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

- GhostDriver(Headless Browser)
 - Pure JavaScript implementation of the WebDriver Wire Protocol for PhantomJS Remote
 - WebDriver that uses PhantomJS as back-end



GhostDriver is the project that provides the code that exposes the WebDriver API for use within PhantomJS.

PhantomJS bakes the GhostDriver code into itself, and ships it as part of its downloadable executable.

Thus, to use "GhostDriver" with PhantomJS, only PhantomJS is needed.

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

■ Mobile Browsers

- Mobile web browsers differ greatly in terms of features offered and operating systems supported
- Best can display most websites and offer page zoom and keyboard shortcuts, while others can only display websites optimized for mobile devices
- Appium and Selendroid are the two cross browser mobile automation tools

■ Appium(Mobile Browser)

- Open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms, which is handled by a Appium node.js server.
- Cross-platform which allows to write tests against multiple platforms (iOS, Android), using the same API
- Enables code reuse between iOS and Android test suites



Copyright © Capgemini 2015. All Rights Reserved 11

Native apps are those written using the iOS or Android SDKs.

Mobile web apps are web apps accessed using a mobile browser (Appium supports Safari on iOS and Chrome or the built-in 'Browser' app on Android). Hybrid apps have a wrapper around a "webview" -- a native control that enables interaction with web content. Projects like Phonegap, make it easy to build apps using web technologies that are then bundled into a native wrapper, creating a hybrid app.

Appium was designed to meet mobile automation needs according to a philosophy outlined by the following four tenets:

You shouldn't have to recompile your app or modify it in any way in order to automate it.

You shouldn't be locked into a specific language or framework to write and run your tests.

A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.

A mobile automation framework should be open source, in spirit and practice as well as in name.

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

Selendroid

Test automation framework which is used Android native & hybrid applications (apps) and mobile web

- Full compatibility with the JSON Wire Protocol
- No modification of app under test required in order to automate it
- Testing the mobile web using built in Android driver webview app
- UI elements can be found by different locator types
- Selendroid can interact with multiple Android devices (emulators or hardware devices) at the same time
- Existing emulators are started automatically
- Supports hot plugging of hardware devices
- Full integration as a node into Selenium Grid for scaling and parallel testing

7.1: Selenium Web Driver – Advance

Remote Web Driver

- Implementation class of the WebDriver interface that a test script developer can use to execute their test scripts via the Remote WebDriver server on a remote machine
- Needs to be configured so that it can run your tests on a separate machine
- If **driver is not Remote WebDriver**, communication to the web browser is local. So driver will be used as below:
Webdriver driver = new FirefoxDriver();
using driver will access Firefox on the local machine, directly
- If **Remote WebDriver**, needs location Selenium Server (Grid) web browser
- For example,

```
WebDriver driver = new RemoteWebDriver(new  
URL("http://localhost:4444/wd/hub"),  
DesiredCapabilities.firefox());
```



Copyright © Capgemini 2015. All Rights Reserved 13

This example assumes that Selenium Server is running on localhost with the default port of 4444. The nice thing about this is you can run Selenium Server on any machine, change the URL to point to the new machine and run the tests. For example, I run the test code from my Mac OS X computer but I run Selenium Server on a Window XP machine. This way I can launch Internet Explorer tests from my Mac OS X computer.

7.1: Selenium Web Driver – Advance

Capabilities and Profile Setting

- Capabilities describes a series of key/value pairs that encapsulate aspects of a browser
- DesiredCapabilities set properties for the WebDriver
- Profile used to create custom Firefox profile and use it with desired capabilities

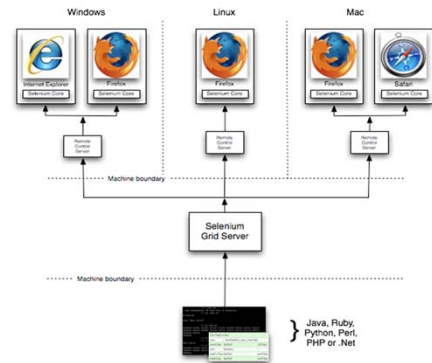
- Example to use Firefox profile with desired capabilities:

```
DesiredCapabilities dc=DesiredCapabilities.firefox();  
FirefoxProfile profile = new FirefoxProfile();  
dc.setCapability(FirefoxDriver.PROFILE, profile);  
Webdriver driver = new FirefoxDriver(dc);
```

7.1: Selenium Web Driver – Advance

Selenium Grid

Allows you run your tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems. Essentially, Selenium-Grid support distributed test execution. It allows for running your tests in a distributed test execution environment.



Summary

- In this lesson, you have learnt
- In this lesson, you have understood that how the selenium works and interacts with client server.
- Different drivers that are available for selenium- Chrome , IE ,Firefox ,headless browsers and mobile browsers.
- In remote webdriver the server will always run on the machine with the browser you want to test. And ,there are two ways to user the server: command line or configured in code.



Add the notes here.

Summary

- Capabilities gives facility to set the properties of browser. Such as to set BrowserName, Platform, Version of Browser.
- There are two reasons why you might want to use Selenium-Grid.
 - To run your tests against multiple browsers, multiple versions of browser, and browsers running on different operating systems.
 - To reduce the time it takes for the test suite to complete a test pass.



Add the notes here.

Review Question

- Question 1:
 - PhantomJS is
 - Chrome Driver
 - Mobile Browser
 - Headless Browser
 - Firefox Driver
- Question 2: True/False
 - Firefox saves your information such as cookies and browser history in a file called your profile.
- Question 3: Fill in the Blanks
 - Client driver will send the program that is written in eclipse IDE as _____ and send it to Selenium server

