Contents

Readme for Java Compatibility & Change logs

1.8 The code base uses Java 1.8 minimum

From August 5th 2016 the code base now uses Java 1.8 as a minimum. Java 1.8 is the lowest supported version of Java bu Oracle and is used in version 3 of WebDriver.

This is primarily because of changes in version 3 of WebDriver.

Other than that the code only uses version 1.6 features.

switch on String not allowed in Java 1.6

In the videos you will see one Java 1.7 feature.

I use a switch statement that takes a string as an argument.

```
switch (defaultBrowser){
    case "FIREFOX":
        useThisDriver = BrowserName.FIREFOX;
        break;
```

Because a common FAQ has been related to this e.g. Why do I get the error ""“java: strings in switch are not supported in -source 1.6 (use -source 7 or higher to enable strings in switch)”

Which I describe on my Java For Testers blog.

http://javafortesters.blogspot.co.uk/2014/03/how-to-fix-java-language-level-issues.html

And because some people have tried to run the code using Java 1.6 (no reason why they shouldn't, it should mostly work).

I've decided to make the code base 'mostly' 1.6 compatible.

I have amended the code which uses the switch statement, so this is no longer in use. I now have a set of 'if' statements instead. But have left in the switch statements as comments so that those people interested can see the old code.

Selenium 3.141.59 compatibility

The course currently runs against version 3.141.59 of Selenium

Selenium 3.141.59 made some breaking changes - I think as the Selenium team were trying to get ready for v4.

Also we now use options instead of capabilities to instantiate drivers e.g.

e.g. instead of

```
Proxy proxy = new Proxy().setHttpProxy(Driver.PROXY); // e.g. "localhost:8888"
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability(CapabilityType.PROXY, proxy);
WebDriver driver = new EdgeDriver(capabilities);
```

We use

```
Proxy proxy = new Proxy().setHttpProxy(Driver.PROXY); // e.g. "localhost:8888"
WebDriver driver = new EdgeDriver(new EdgeOptions().setProxy(proxy));
```

This currently shows deprecation warnings in 3.141.59 but will change in v4 of Selenium.

Change Logs

18/February/2020

- updated the alerts lectures and code to use the new test pages
- https://testpages.herokuapp.com/styled/alerts/alert-test.html
- and created a pdf for the documentation of alerts

15/November/2019

- frames, iframes, windows and managing windows are now new sections with new code
- removed the old code for iframes, frames and windows from the code base
- have added the above code as a separate downloadable zip if people are interested in the downloads lecture

14/November/2019

I have recoded and re-recorded the frames, iframes, windows and window management sections.

The new examples use the https://testpages.herokuapp.com site and are generally better and simpler.

12/November/2019

- deprecated the Marionette Driver videos and information in the code
- added more information in the code for the refactoring exercise JavaScript example
- created a new video explaining the refactoring Page Objects javascript exercise through the GUI

Extra information about Marionette

- all deprecated code was moved to gists
- showing use of Marionette Driver
- https://gist.github.com/eviltester/7ce862f053529e6d4875b3ffa936f176
- showing portable firefox
- https://gist.github.com/eviltester/96cc5dfd33ddae4ea1e6bd880e39dc3a

Firefox Driver went through a difficult patch when the driver was moved out of Selenium WebDriver and into a separate driver executable.

The driver executable was renamed from wires.exe to geckodriver.exe

And there were different configuration options required for old Firefox and new Firefox.

Now, the geckodriver.exe is used and `new FirefoxDriver()` is enough to create browser. Legacy support is enabled when you use the options

```
options.setLegacy(true);
```

PortableFirefox was useful during this time to have a stable version of Firefox to work against but I no longer support that.

09/November/2019

Created a new video for the SlowLoadableComponent, because that was the only class that was preventing upgrading to Selenium WebDriver 3.14.159. The version of HtmlUnitDriver in the example code has also been updated to the current version of 2.36.0

Previously Selenium WebDriver used an internal clock class for the SlowLoadableComponent constructor:

```
    public ProcessedFormPage(WebDriver aDriver) {
        super(new SystemClock(), 10);
        driver = aDriver;
    }
```

from:

```
import org.openqa.selenium.support.ui.SystemClock;
```

Now we use the `java.time.clock` class

e.g.

```
    public ProcessedFormPage(WebDriver aDriver) {
        super(Clock.systemDefaultZone(), 10);
        driver = aDriver;
    }
```

This change was introduced in version 3.14.0

HtmlUnitDriver 2.36.0

https://github.com/SeleniumHQ/htmlunit-driver

08/November/2019

The screenshot code has been updated and the videos have been updated to match. There is also a pdf of the screenshot material in the last lecture of screenshots. I'm going to try and create more pdf material for the course to have a summary and notes for the material.

Screenshot changes were a result of the HasCapabilities in WebDriver no longer reporting on the capability to take screenshots, which was preventing screenshots from being taken.

I took the opportunity to update these lectures to use the test pages:

https://testpages.heroku.com

And make them cross platform so they no longer use the "C:" drive in the tests.

- Taking Screenshots
  - all videos and code have been refreshed

23/October/2019

I patched some of the videos to remove out of date references:

- Getting Started - About the Tools
- Getting Started - About the Test Applications (added new video)
- Junit - JUnit 4 vs JUnit 5 (added new video)
- Getting Started - Final Points
  - Finding Elements in Firefox Dev Tools (added new video)
- Manipulation Exercises
  - patched video for Exercise 5
- Begin Coding (all new videos in section)

Details:

- Patched the Getting Started, About the Tools video because this still mentioned FireBug and Firepath
  - these were old plugins for Firefox when the Dev Tools were not very good. Sadly FirefPath was better than the current dev tools for CSS Selectors and XPath selectors which we describe in the Interrogation section. Also I now use Chrome as my default WebDriver browser rather than Firefox.
  - Section: Getting Started - Lecture: About the tools
- Created a video "Finding Elements in Firefox Dev Tools"
  - this shows how to use Firefox dev tools to find with CSS query selectors and a plugin called Try Xpath to replace Firepath. FirePath was better, but Try XPath is the best replacement plugin I have found.

I prefer to use the Chrome dev tools. A few versions of Firefox dev tools ago, we were able to search for XPath using the dev tools but that functionality has been removed.
    * Section: Getting Started Final Points - Lecture: Finding Elements in Firefox Dev Tools

- Deprecated the older "Install Firefox Plugins" this had text explaining do not install them, but I hope the new Firefox and Chrome videos explain what to do.
- Created a video on Junit 4 vs JUnit 5 to explain why we use JUnit 4 even though the most up to date version is JUnit 5
    - Section: JUnit - Lecture: JUnit 4 vs JUnit 5
- Changed the title of the Logistics Section to mention the Source Code
- Patched the Video for Manipulation Exercises Exercise 5 because the final answer I created no longer works.
    - The test has been ignored in the code so it won't run, but I left it in the code because it used to demonstrate the concept of refactoring to methods to avoid duplicated code.
    - The issue with the test was that sendkeys can no longer control drop down combo boxes the way it used to in earlier versions of WebDriver
    - Section: Manipulation Exercises - Lecture: Manipulation Exercises sample answers question five
- Deprecated the import project video as the most reliable way to import source projects into IntelliJ now is to "Open" the project
    - Section: Opening and Importing Maven Projects with IntelliJ

11/June/2018

Upgraded code to version 3.12.0 of WebDriver

```
<webdriver.version>3.12.0</webdriver.version>
<htmlunitdriver.version>2.28.1</htmlunitdriver.version>
```

Things to note:

Changed Driver.java to have a SAFARI driver

- To use Safari from Webdriver you must manually start safari and in the Develop menu select 'Allow Remote Automation' and then close safari to save the settings
- If you haven't enabled this then you will see the following error message when you try to use Safari from WebDriver

```
org.openqa.selenium.SessionNotCreatedException: Could not create a session: You must enable
Command duration or timeout: 53 milliseconds
Build info: version: '3.12.0', revision: '7c6e0b3', time: '2018-05-08T14:04:26.12Z'
System info: host: 'MacBook-Pro-2', ip: 'fe80:0:0:0:1455:d30b:10c2:fa34%en0', os.name: 'Mac
Driver info: driver.version: Driver
```

DesiredCapabilities is becoming Options The Options classes `ChromeOptions` and `FirefoxOptions` are easier to use because they have named methods, rather than trying to remember the. capabilities and String values.

- http://chromedriver.chromium.org/capabilities

This changes the constructor for the driver e.g. when we want to configure a profile for Firefox

Instead of:

```
FirefoxProfile profile = new FirefoxProfile();
profile.setPreference("extensions.firebug.currentVersion", "1.10.5");
profile.addExtension(new File(extensionPath));

WebDriver firefox = new FirefoxDriver(profile);
```

We now do:

```
FirefoxProfile profile = new FirefoxProfile();
profile.setPreference("extensions.firebug.currentVersion", "1.10.5");
profile.addExtension(new File(extensionPath));

FirefoxOptions options = new FirefoxOptions();
options.setProfile(profile);

WebDriver firefox = new FirefoxDriver(options);
```

Instead of:

```
WebDriver firefox = new FirefoxDriver(
        new FirefoxBinary(
                new File("c:/webdrivers/FirefoxPortable/v47_0_1/FirefoxPortable.exe")),
        new FirefoxProfile(),
        portableCapabilities);
```

We now do:

```
FirefoxOptions options = new FirefoxOptions();
options.setBinary(new FirefoxBinary(
        new File("c:/webdrivers/FirefoxPortable/v47_0_1/FirefoxPortable.exe")));
options.setProfile(new FirefoxProfile());
options.setLegacy(true);
```

```
WebDriver firefox = new FirefoxDriver(options);
```

Tests that no longer work `submitFormWithDropDownFiveUsingKeyboardSpecialKeys` - this was an 'edge' case example of using keys to select a drop down using cursor keys

I amended the `WindowManageExercise` and `WindowsManageExample` test as both of these had very small window sizes that caused an issue on larger monitors - this is an edge case and changing window sizes like this is rarely done

ElementNotInteractableException WebDriver now throws an `ElementNotInteractableException` instead of an `ElementNotVisibleException`

Mac install Easiest way to install ChromeDriver and GeckoDriver for firefox on mac is to use HomeBrew

To install firefox driver on Mac

http://brewformulas.org/geckodriver

```
brew install geckodriver
```

http://brewformulas.org/Chromedriver

```
brew install chromedriver
```

You will know if it has installed correclty because you can type `chromedriver` and `geckodriver` at the command line and you will see them run.

If you do this then you can run them without setting the path properties and hardcoding a path for the driver.

22/September/2017

- I have upgraded the code from version 3.3.1 of Selenium to 3.5.3
  - I seem to have missed that out in some release or release notes before
- The `WaitingExercisesUsingPredicateTest` was failing because WebDriver now uses a Function rather than a Predictate
  - this also required a change to `FluentWaitForWebElementExampleTest` where it is easy to see the difference between using Predicate and Function
  - I've documented the changes I maded to the test in the code, I haven't yet updated the videos, this is a rather 'niche' usage
- I've updated HtmlUnitDriver to 2.27 - which also meant re-organizing the order of the pom.xml to avoid HTMLUnitDriver dependencies overriding the Selenium 3.5.3 dependencies

- AllBrowsers was tested against webdriver 3.5.3 Chromedriver 2.32 and Chrome 61.0.3163.100 for a clean run
- HTMLUnit Suite runs clean
- I added a minor check in the Driver code on maximise to display any exceptions to console
- FirefoxDriver has some deprecated config so I'll have to revisit the course for firefox soon
- I have decided to bump the version of the pom.xml to match the latest version of Selenium WebDriver that the code has been executed against
  - so it moved from 1.0 to 3.5.3
- because my test app moved from http to https, I have amended some of the assertions to be scheme independent and ignore the http or https in the url assertions

19/October/2016

Selenium WebDriver 3 has been released.

I have amended the code to run against 3.0.1.

I recommend ChromeDriver as the default Driver. Rather than FirefoxDriver.

If, you follow the instructions to install ChromeDriver, and anywhere in the videos you see 'FirefoxDriver' you use 'ChromeDriver' instead, then you should be fine for the course.

- 3.0.1 is compatible with HTMLUnit 2.23
- I have amended Driver
  - to use Chrome by default
  - FIREFOX will now default to geckodriver using FirefoxDriver
  - FIREFOXMARIONETTE still uses the MarionetteDriver and now defaults to look for geckodriver.exe instead of wires.exe
  - FIREFOXPORTABLE uses FirefoxDriver in legacy mode to use the internal firefox driver
- the DriverSanity tests have not yet been updated. I need to fix these and amend the lectures that use them.
- I added some additional synchronisation for GeckoDriver in the tests
  - `WebElement languageWeUsed = new WebDriverWait(driver,10).until(elementToBeClickable By.id("_valuelanguage_id")));`
  - after submitting a form, GeckoDriver was not fully waiting for the page to load,so I have added some synchronisation in some of the tests to suppoort this.
- EdgeSuite has changed to include more tests because more tests are passing on Edge now
- some of the frames tests I now `driver.switchTo().defaultContent();` after clicking on links between frames, because this helps GeckoDriver synchronise on the load

16/September/2016

- Added code for new lecture "Adding a Browser Driver to the Path" in the "Different Browsers" section.
  - `package com.seleniumsimplified.webdriver.drivers.onpath;`
  - `ChromeDriverPathTest`

5/August/2016 - Tidy Code and use Java 1.8

- Added PortableFirefox example test
- Trimmed the pom.xml to remove a lot of comments for older versions of JUnit and Hamcrest
- added comment in pom.xml showing the testng exclusion required to use 3.0.0-beta2 with JUnit
- removed some GoogleChrome and IE workarounds in the code that are no longer required
- amended screenshot test to use Java 1.8 base 64 instead of Selenium code which is removed in version 3
- updated pom to enforce java 1.8 in readiness for Selenium WebDriver 3
- CI Suite on github has been amended to remove some tests that failed on Edge and Marionette, so it should now run clean on all browsers.
- readme text was re-order to read from top to bottom as most recent to last
- updated Firefox47 document

Code was run against:

Selenium WebDriver 2.53.1 - FirefoxDriver using FirefoxPortable 47.1 - AllBrowserSuite - Firefox v 48 with Marionette v 0.9.0 - MarionetteSuite - Chrome v 52.0.2743.82 m with ChromeDriver v v2_22 - AllBrowserSuite - Edge v 14.14366 with Edge Driver (Insider) - EdgeSuite - IE - 1 failing test in AllBrowserSuite - HTMLUnit v2.21 - with HTMLUnitSuite

28/July/2016 Continuous Integration and Other Browsers

I completely changed the Continuous Integration section.

The old section was based on SVN, and used the main source code base. To access it you had to create an account on xp-dev and we had a whole manual hassle and forms to give you access. Plus the main code wasn't designed for CI and it meant I kept adding fixes.

Now. The code that is designed for CI is in Git. Much more modern. And much more in demand in the 'real world'.

I created a public Git repository that has a very small selection of example test code from the course, and which has been amended to run cleanly on HTMLUnit and Firefox, and probably Chrome and IE as well.

This means that:

- I have deprecated most of the existing CI lectures - you can find them at the very bottom of the syllabus if you do want to watch them
- You don't need to create any accounts to follow the CI lectures
- The code for CI is much more stable
- You can use the zip file as the main code base for the course

This should make everything easier and the CI section is now a lot easier and cleaner.

I have left some of the old CI lectures untouched because they provide overview of changes made for cross browser testing in CI, which is still useful information.

As part of this exercise I also re-ran the code against other browsers, so I have added two new lectures on EDGE and have added EDGE into the Driver class. I made a few small tidy up code changes into the Driver class that won't impact anything.

I've also renamed all the suites ro remove 'Test' in the name, this means that to run them from commandline you have to specify the name as a property e.g. -Dtest=AllBrowserSuite

I added a few more suites for Marionette, HTMLUnit and Edge to show which tests pass and which tests fail on those drivers.

I have also cut down the number of profiles because Firefox and ChromeDriver have pretty much reached parity, so they both run the same tests.

I was surprised at how good IE_Server_32 has become, all but one of the tests ran on IE with the 2.53.1 32 bit version of IE driver when I tried it on my machine. HTMLUnit 2.21 has become pretty good as well, so I've increased the version of HTMLUnit in the pom.xml.

I also added a /tools folder as a peer to the code folder to demonstrate the layout I use for browser drivers locally, I haven't included the driver(.exe) files in there but I have added a few readmes and links to the drivers.

11/July/2016 Appium

I replaced the AndroidDriver video with a set of Appium videos. These describe installation and show Appium in action with Chrome on physical device, in an Emulator with AVD. Also description of the cross platform changes required to make the tests run effectively on Android with Appium.

Minor source code changes for Appium - described in the above videos.

I also added a new set of slides in the Appium section and a 'transcript' of sorts. (a pdf to cover the whole section).

07/July/2016 Marionette etc.

I have updated the code to use WebDriver 2.53.1

There was some confusion when Firefox 47 failed to work with Selenium Webdriver 2.53.0

Firefox 47 introduced a bug and was incompatible with WebDriver 2.53.0

WebDriver 2.53.1 is compatible with Firefox 47.1

Remember Firefox is moving to use the Marionette driver and the bug may have been caused by work related to that.

I have added `\docs\` folder containing `firefox47.pdf`

This contains explanation of what is going on with Firefox and various workarounds in case Selenium WebDriver becomes incompatible with Firefox again.

I have updated the `Driver.java` to allow use of FIREFOXPORTABLE as the driver name. This assumes that you have a firefox portable .exe in a tools directory as per the path that is coded into Driver.java. By default the Driver.java still uses Firefox.

I have also moved `readme.txt` into the `\docs\` folder and converted it into a `.pdf` using `pandoc`.

22/April/2016 Marionette

As from Firefox 46 the inbuilt Firefox Driver in Selenium 2 will not work to use it we need to stay on Firefox 45 (Extended Support Release) download from here https://www.mozilla.org/en-US/firefox/organizations/all/

More details about this on David Burn's Blog:

http://www.theautomatedtester.co.uk/blog/2016/selenium-webdriver-and-firefox-46.html

And on my blog:

http://seleniumsimplified.com/2016/04/how-to-use-the-firefox-marionette-driver/

More details about marionette here:

https://developer.mozilla.org/en-US/docs/Mozilla/QA/Marionette/WebDriver

To install:

- Download the .zip from the above page
- Unarchive to a folder
- Rename the .exe to wires.exe

- Either:
  - Add this wires.exe to your path
  - or set a property called webdriver.gecko.driver that points to your wires.exe

Then instead of using FirefoxDriver use MarionetteDriver

I have added a test `FirefoxMarionetteDriverTest.java` which demonstrates using the marionette driver.

I have updated `Driver.java` to have a FIREFOXMARIONETTE driver.

Currently Driver still defaults to Firefox as I recommend you stay on Firefox 45 at the moment.

I have added additional videos to the course to describe Marionette driver.

21/March/2016 upgrade to webdriver 2.53.0

WebDriver 2.53.0 has removed HTMLUnit Driver from the code, this is now in a new driver.

HTMLUnitDriver

```
<!-- from WebDriver 2.53.0 HTMLUnit is no longer distributed with Selenium WebDrive
<!-- https://github.com/SeleniumHQ/htmlunit-driver -->
<!-- need to bring in the htmlunit driver on its own -->
<!-- https://github.com/SeleniumHQ/selenium/blob/master/java/CHANGELOG -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>${htmlunitdriver.version}</version>
</dependency>
```

I updated my build machine to ChromeDriver 2.21.

I've amended two of the cookie tests because an issue with Chrome has been resolved:

- changeCookieVisitsCountUsingCookieBuilder(com.seleniumsimplified.webdriver.cookies.CookiesExercisesTestWorkWithExtraSync): Chrome cookie creation issue has be en resolved
- changeCookieVisitsCount(com.seleniumsimplified.webdriver.cookies.CookiesExercisesTestWorkWithExtraSync): Chrome cookie creation issue has been resolved

The issue was 7499

Test have been run in Chrome (49.0.2623.87 m), and Firefox (45.0.1).

08/August/2015 upgraded to 2.47.1

Upgraded webdriver to 2.47.1 - tests were run locally and through Browserstack and Saucelabs (Win 7, Firefox).

Added properties in the pom.xml to make the upgrade easier in future - change version at top of file.

24/July/2015 allow checking against currentDriver and currentBrowser

Initially, the tests didn't really run on Grid, and I have code in the tests, to handle workarounds on different browsers.

But as I use grid more, my current approach of using currentDriver to code workarounds for specific browsers, doesn't work because I'll just get GRID when I want to know FIREFOX.

So I added a method called currentBrowser to Driver which returns the Browser in use.

I can still find out what currentDriver is with `Driver.currentDriver`. But if I want to know the browser, I should use `Driver.currentBrowser()`.

So I amended tests in places to use currentBrowser() e.g. in CookiesExercisesTest

23/July/2015 amended for gridconfig

Previously the grid config files were attached to a lecture.

I have created a /gridconfig folder as a peer of /src which contains the config I used in the early grid and later 'mac node on grid' lectures.

21/July/2015 amended for extra grid code to allow execution on BrowserStack

Increased fluent wait timeout on the FluentWaitExercisesTest because of grid latency impacting execution.

Amended Driver.java to add extra code to support defining any capability through environment vars or properties by creating an envrionment var or property prefixed with "WEBDRIVER_GRID_CAP_X_"

e.g. WEBDRIVER_GRID_CAP_X_os_version=XP would create a capability "os_version" with value "XP"

e.g. WEBDRIVER_GRID_CAP_X_browserstack.debug=true would create a capability "browserstack.debug" with value "true"

20/July/2015 amended for extra grid code

I amended Driver to make it easier to run on grid - by adding a config method which can pull config from properties, environment variables or just use defaults.

I amended some code for grid:

WindowManageExerciseTest.java would fail on screen size comparison because it was comparing remote window size to local screensize - check are only done when not running on Grid.

FluentWaitExercisesTest.java would fail sometimes because grid was slow and two calls were made to extract the elemennt text, at which point it had changed due to grid speed, so amended to avoid a 2nd call.

MyFailingWebDriverWaitTest.java usually wouldn't fail on grid because it was slower, so added a check that when Grid - throw exception to allow test to pass.

01/July/2015 upgrade to webdriver 2.46.0

- 2.46.0 means that Opera is completely gone. It no longer compiles against properly. (Opera, really last worked in v2.34.0 but still compiled against)
- I have removed all the Opera code from the codebase
- I have now uninstalled Opera 12.17 and will no longer be updating this code
- https://github.com/eviltester/operaWebDriverExample

I ran all the tests against: * WebDriver 2.46.0, * Firefox 38.0.5, * ChromeDriver 2.16, * Chrome v 43.0.2357.130 m * Mac OS X, Windows 8

Changes I had to make:

Cookies

- Because of ChromeDriver 2.16
  - ChromeDriver adds 2 cookies when you add a cookie ((https://code.google.com/p/selenium/issues/deta
  - It also prefixes a '.' on the domain value, so applications may create duplicate cookies
  - I ameded the waitForCookieWithValue to check for multiple cookies, rather than return the first
- Because of Firefox:
  - Firefox needs me to append a '/' on the cookie path
  - I had to add extra synchronisation into `switchToNewWindow` method because it no longer blocks on window opening before switching context back to original context.

Select

- In `SelectSupportTest` I had to trim the return value which was padded with spaces

Chrome

- Chrome and ChromeDriver 2.16 no longer have issues with Frames and Windows so I have no 'failing' Chrome tests and the same Chrome and Firefox tests execute now.

Mac

- One of the window management position tests failed because I set the Y value to 20, but Mac needs a minimum of 23 to accommodate the top menu bar, I changed the Y to 40

Intermittency

- Some of the JavaScript and window management tests seem intermittent in a CI build, but I haven't fixed this yet.

WebDriverBackedSelenium

- Since all the `WebDriverBackedSelenium` examples are deprecated by the main Selenium project, it is only a matter of time before they are removed. I have moved the `WebDriverBackedSelenium` example test to its own github repo
- https://github.com/eviltester/webDriverBackedExample