ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Software Testing

Chapter 6
Test Automation Practice

http://vku.udn.vn/

Contents

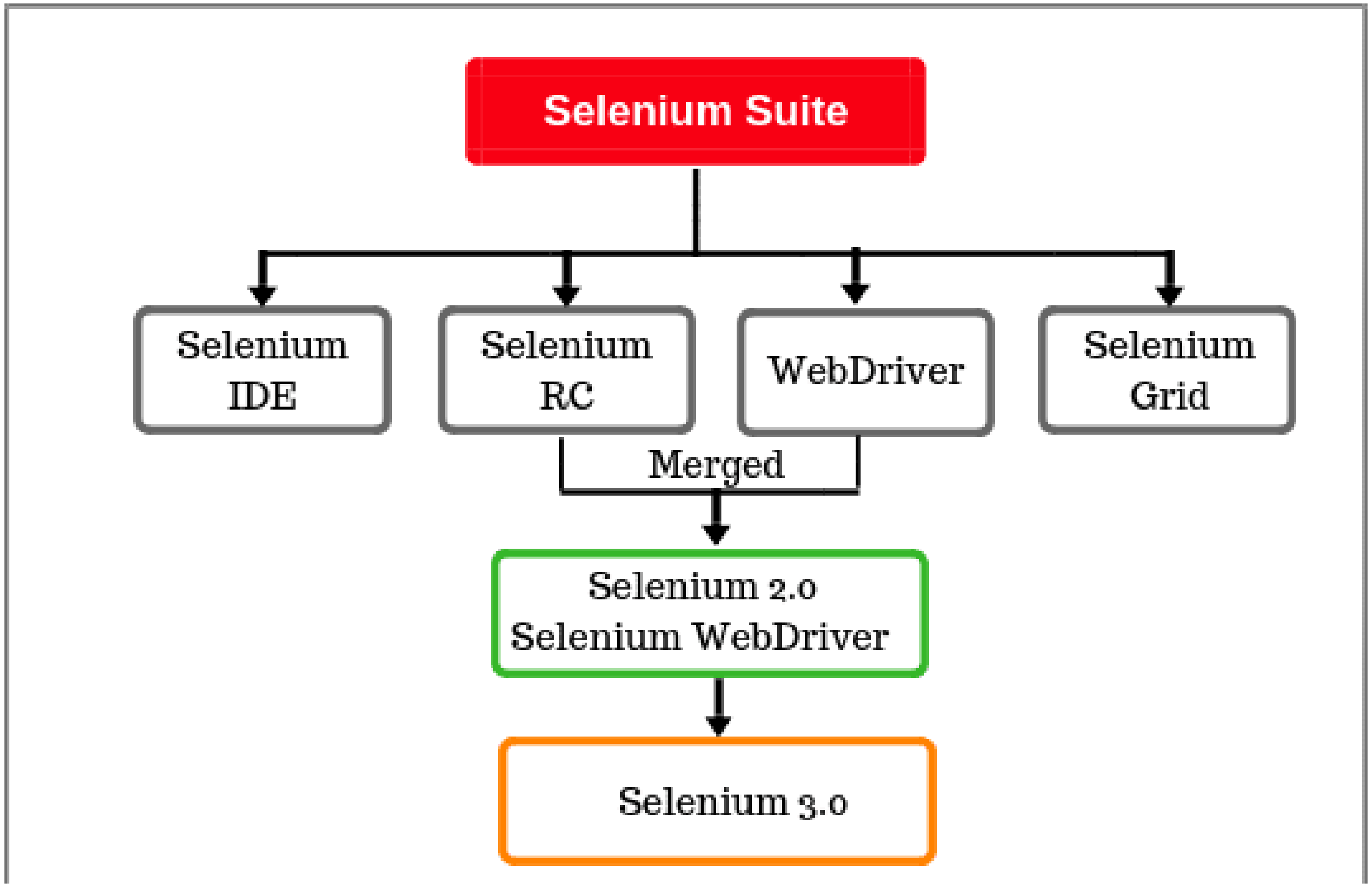
Overview

❖Selenium is a free and open-source functional automation testing tool that is used to test the functionality of the web-based application.

Features

- ❖Selenium tests can be run on multiple browsers
- ❖Allow scripting in several language like Java, C#, PHP, Python
- ❖Assertion statements provide an efficient way of comparing expected and actual results
- ❖Inbuilt reporting mechanism

Selenium Components



Selenium Web driver

- ❖Selenium WebDriver is a web framework that permits you to execute cross-browser tests.
- ❖This tool is used for automating web-based application testing to verify that it performs expectedly.

Selenium Web driver

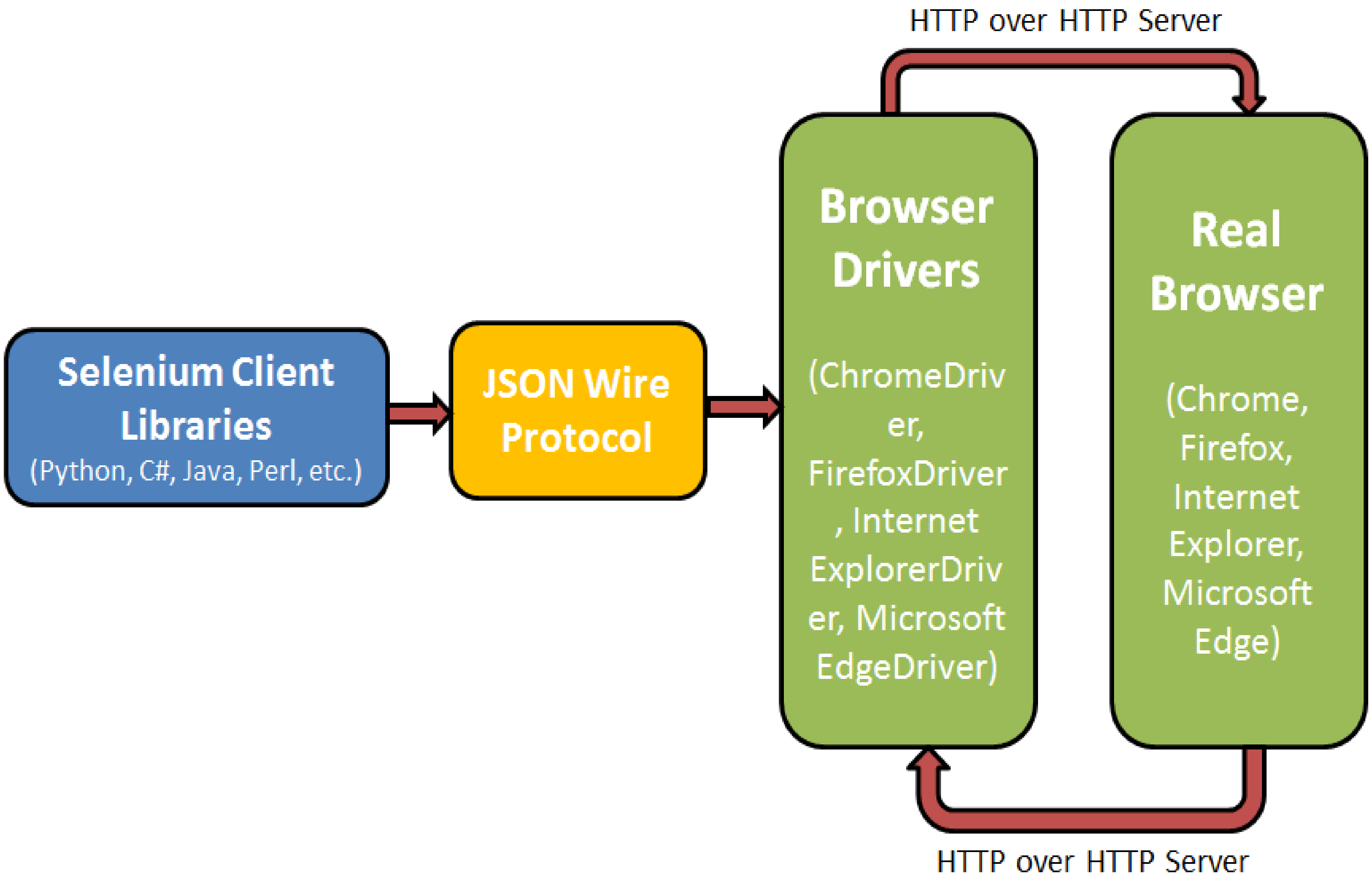
- ❖WebDriver is a tool for automating testing we applications. WebDriver interacts directly with the browser without any intermediary.
- ❖Multi-browser testing including improved functionality for browsers
- ❖Handling multiple frames, multiple browser windows, popups and alerts.
- ❖Complex page navigation
- ❖Advanced user navigation such as drag and drop
- ❖AJAX-based UI elements

WebDriver: Architecture

❖WebDriver Architecture is made up of four major components:

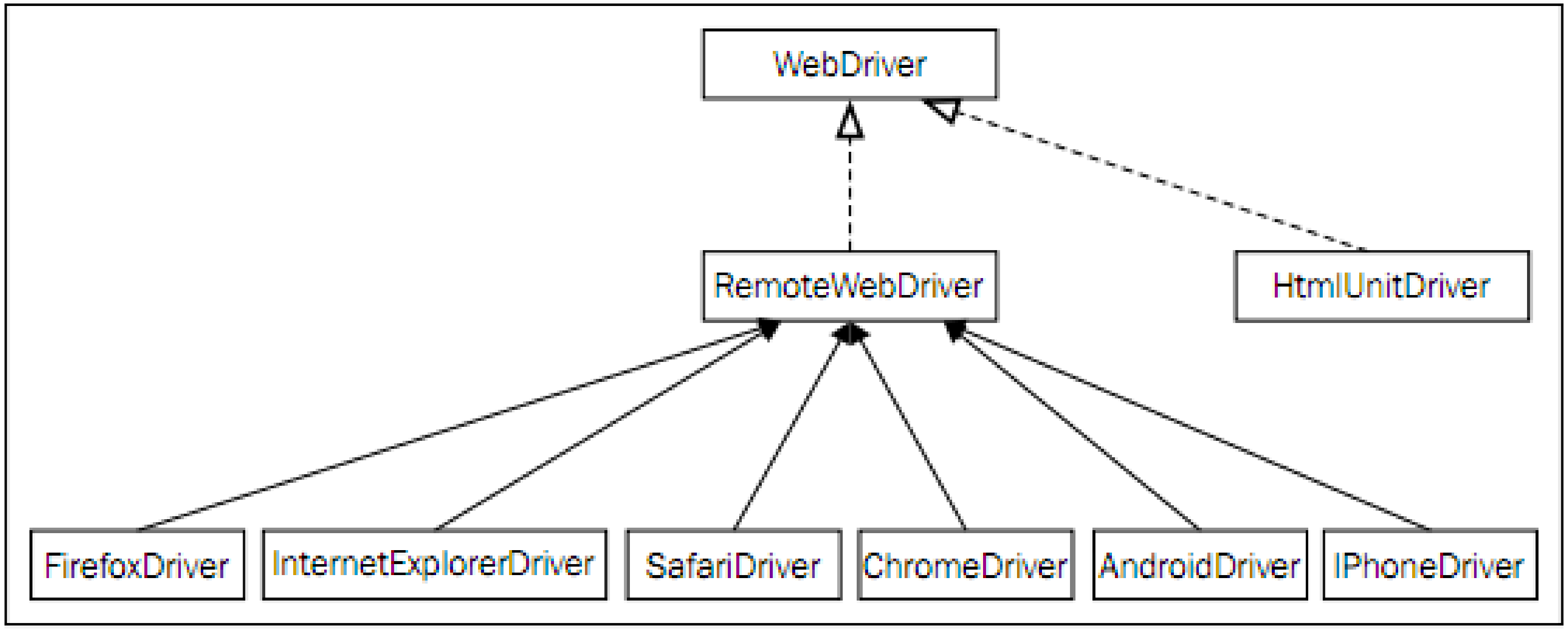
- Selenium Client library
- JSON wire
  protocol over HTTP
- Browser Drivers
- Browsers

WebDriver: Architecture


HTTP over HTTP Server
Selenium Client Libraries
(Python, C#, Java, Perl, etc.)
JSON Wire Protocol
Browser Drivers
(ChromeDriver, FirefoxDriver, Internet ExplorerDriver, Microsoft EdgeDriver)
Real Browser
(Chrome, Firefox, Internet Explorer, Microsoft Edge)
HTTP over HTTP Server

WebDriver interface

❖WebDriver is an interface whose concrete implementation is done in two classes: RemoteWebDriver and HtmlUnitDriver



Selenium API

❖WebDriver – to control the browser

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

❖void get(string url) – open page

❖void quit() – close browser

Selenium API

❖WebElement – to identify web element (s) on the page

```
WebElement findElement(By by);
List<WebElement> findElements(By by);
```

- Throws a NoSuchElementException if element is not found

❖The findElement command returns an object of type WebElement. It can use various locator strategies such as ID, Name, ClassName, TagName, LinkText, PartialLinkText, Xpath, CSS Selector.

Selenium API

❖The findElement method returns an object of type WebElement. It can use various **locator** strategies such as ID, Name, ClassName, TagName, LinkText, PartialLinkText, Xpath, CSS Selector.

❖Ex:

```
WebElement loginLink =driver.findElement(By.linkText("Login"));
```

❖The findElements method returns a list of web elements and returns an empty list if there are no elements found using the given locator strategy and locator value

```
List<WebElement> listOfElements =
                    driver.findElements(By.xpath("//div"));
```

Selenium API: basic operations on elements

- void click()
- void submit()
- String getValue()
- void sendKeys(keysToSend)
- void clear()
- string getElementName()
- string getAttribute(string name)

Selenium API: Waiting for Web Elements to load

❖ Implicit Wait time

- The implicit wait will tell the WebDriver to wait a certain amount of time before it throws a "No Such Element Exception.”

```
driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);
```

❖ Explicit Wait time

- Explicit waits are a concept from the dynamic wait, which waits dynamically for specific conditions.
- It can be implemented by the WebDriverWait class

```
WebDriverWait wait=new WebDriverWait(WebDriveReference,TimeOut);
```

Selenium API: Locating target windows and iFrames

- ❖ Working with browser windows:
  - driver.getWindowHandles()
  - driver.switchTo().window(windowName)
- ❖ Working with frames
  - Driver.switchTo().frame(frameName)

Selenium API: Handling alerts

- ❖WebDriver provides an API to handle alert dialogs:
  - Alert alert()
- ❖The Alert interface contains a number of APIs to execute different actions:
  - void accept()
  - void dismiss()
  - String getText()
  - void sendKeys(keysToSend)

Selenium API: Navigate

- ❖Navigate is one such feature of WebDriver that allows the test script developer to work with the browser's Back, Forward, and Refresh controls

```
WebDriver.Navigation navigate()
```

- ❖Some methods:

```
driver.navigate().to(String url)
```

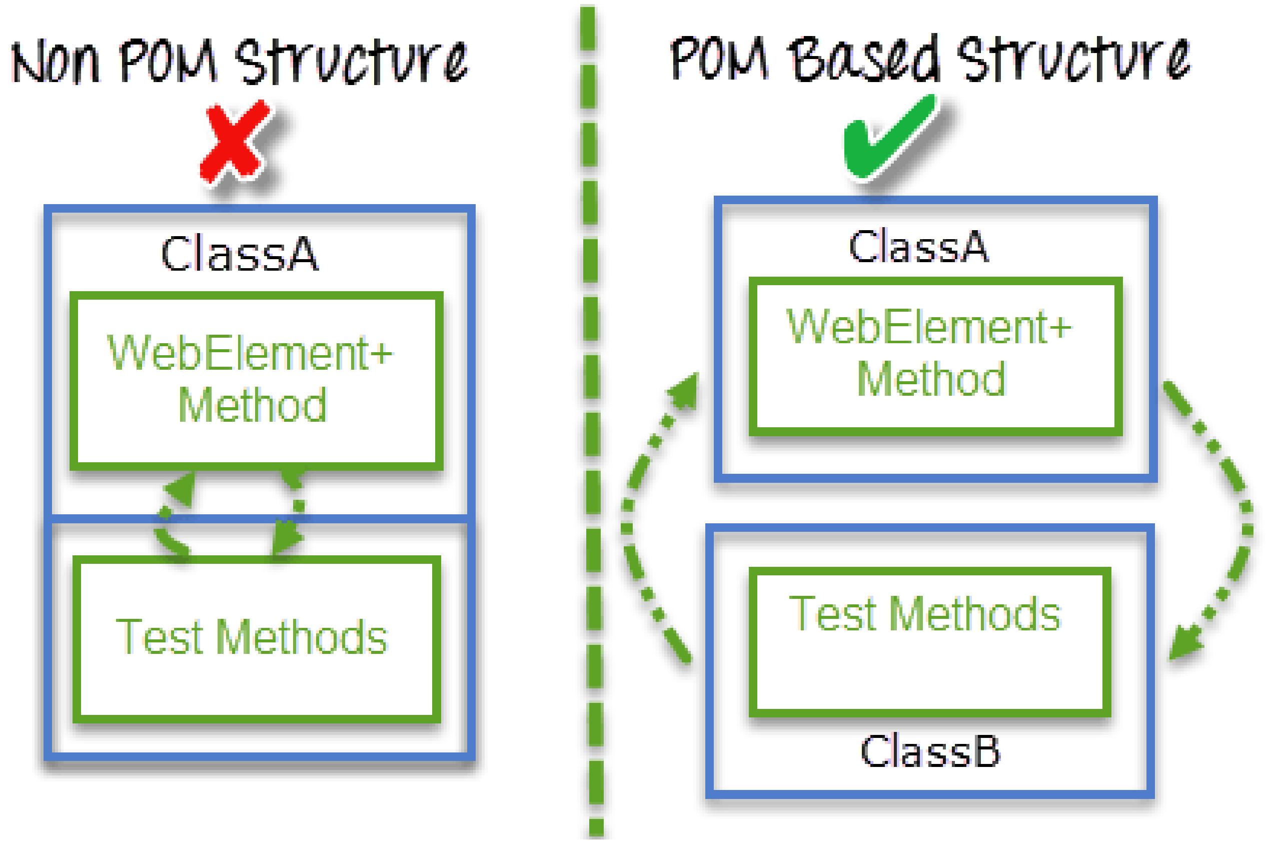
```
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
```

Basic Steps in a Selenium WebDriver Script

- ❖Create a WebDriver instance.
- ❖Navigate to a webpage.
- ❖Locate a web element on the webpage via locators in selenium.
- ❖Perform one or more user actions on the element.
- ❖Preload the expected output/browser response to the action.
- ❖Run test.
- ❖Record results and compare results from them to the expected output.

Understanding PageObject Pattern

❖Page Object Model (POM)



Understanding PageObject Pattern

- ❖ Page Object Model (POM) is a design pattern, popularly used in test automation that creates Object Repository for web UI elements.
- ❖ The advantage of the model is that it reduces code duplication and improves test maintenance.

Understanding PageObject Pattern

❖**Page Factory in Selenium** is an inbuilt Page Object Model framework concept for Selenium WebDriver but it is very optimized.

❖It is used for initialization of Page objects or to instantiate the Page object itself. It is also used to initialize Page class elements without using "FindElement/s“

❖**AjaxElementLocatorFactory** is a lazy load concept in Page Factory - page object design pattern to identify WebElements only when they are used in any operation.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Thank You !

http://vku.udn.vn/