* WebDriver is a tool for testing web applications **across different browsers** using different programming languages.
* You are now able to make powerful tests because WebDriver **allows you to use a programming language** of your choice in designing your tests.
* WebDriver is **faster than Selenium RC** because of its simpler architecture.
* WebDriver **directly talks to the browser** while Selenium RC needs the help of the RC Server in order to do so.
* WebDriver's API is more**concise** than Selenium RC's.
* WebDriver **can support HtmlUnit** while Selenium RC cannot.
* The only drawbacks of WebDriver are:
* It **cannot readily support new browsers**, but Selenium RC can.
* It **does not have a built-in command** for automatic generation of test results.
* Aside from a browser, you will need the following to start using WebDriver
* Java Development Kit (JDK). <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
* Eclipse IDE - <http://www.eclipse.org/downloads/>
* Java Client Driver - <http://seleniumhq.org/download/>
* When starting a WebDriver project in Eclipse, do not forget to import the Java Client Driver files onto your project. These files will constitute your Selenium Library.
* With new version of Selenium, there is no browser that you can automate without the use of a Driver Server.

……………..

* To start using the WebDriver API, you must import at least these two packages.
* org.openqa.selenium.\*
* org.openqa.selenium.firefox.FirefoxDriver
* The get() method is the equivalent of Selenium IDE's "open" command.
* Locating elements in WebDriver is done by using the findElement() method.
* The following are the available options for locating elements in WebDriver:
* By.className
* By.cssSelector
* By.id
* By.linkText
* By.name
* By.partialLinkText
* By.tagName
* By.xpath
* The By.cssSelector() does not support the "contains" feature.
* You can instantiate an element using the WebElement class.
* Clicking on an element is done by using the click() method.
* WebDriver provides these useful get commands:
* get()
* getTitle()
* getPageSource()
* getCurrentUrl()
* getText()
* WebDriver provides these useful navigation commands
* navigate().forward()
* navigate().back()
* navigate().to()
* navigate().refresh()
* The close() and quit() methods are used to close browser windows. Close() is used to close a single window; while quit() is used to close all windows associated to the parent window that the WebDriver object was controlling.
* The switchTo().frame() and switchTo().alert() methods are used to direct WebDriver's focus onto a frame or alert, respectively.
* Implicit waits are used to set the waiting time throughout the program, while explicit waits are used only on specific portions.
* You can use the isEnabled(), isDisplayed(),isSelected(), and a combination of WebDriverWait and ExpectedConditions methods when verifying the state of an element. However, they do not verify if the element exists.
* When isEnabled(), isDisplayed(),or isSelected() was called while the element was not existing, WebDriver will throw a NoSuchElementException.
* When WebDriverWait and ExpectedConditions methods were called while the element was not existing, WebDriver would throw a TimeoutException.
* Note:
* driver.get() : It's used to go to the particular website , But it doesn't maintain the browser History and cookies so , we can't use forward and backward button , if we click on that , page will not get schedule
* driver.navigate() : it's used to go to the particular website , but it maintains the browser history and cookies, so we can use forward and backward button to navigate between the pages during the coding of Testcase

…………….