Selenium Test Scripts Writing Guide

1. Prerequisites

- Firefox is installed (we only use Firefox WebDriver at the moment)
- A development environment has been correctly set up.
    - NET Framework
        - Visual Studio Express 2012 for Desktop
    - JVM (also requires Maven 3)
        - Eclipse Kepler
        - IntelliJ version 12

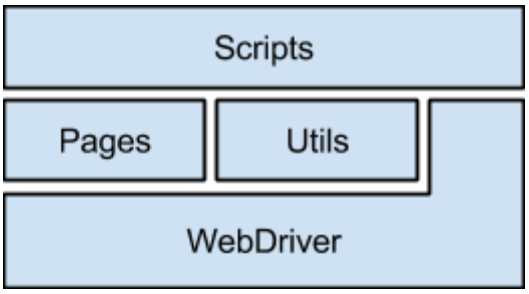
2. File Structure

There are three folders in the package.

- IDETest: contains the tests recorded and modified by Selenium IDE.
- WebDriverTests: contains test scripts written by using Selenium WebDriver and Page Object pattern. There are three template projects in the folder testing the same functions, but are implemented by Eclipse (Java, JUnit), IntelliJ (Java, JUnit) and VS Express for Desktop (C#, VS Unit Testing Framework).
- テスト用HTML: contains the local html pages to test.

3. Test Framework Structure

There are three components in the framework: Pages (Page Objects), Utils (or Helpers) and Scripts (real tests in unit test format).



4. Customization

These are template (sample) projects, therefore in order to add your own test scripts you need to take the following steps:

a. To add your own page objects

- to extend your page from PageTemplate class. Then you have to implement *init* method in which you normally do some preparation work such as waiting for the page is completed loaded.

- to add your own page url by changing this private static variable *url*.

```
private static final String url = "..."
```

- to define the web elements on your page. Because we are using PageFactory, the following declaration style is recommended.

```
@FindBy(how = How.XPATH, using = "//input[@type='text' and @name='PRICE01']")
private WebElement textPrice1;
```

We suggest that all the web elements are declared as private, but all the actions are public, therefore scripts cannot call web elements directly instead use those public actions to run test cases. Get methods (or Properties in C#) are also set public to retrieve the statuses of web elements, so that various verifications can be performed.

b. To write your own test scripts to test the pages you added

- to initialize your test. In the test setup method, you need to create an instance of your page object, and then initialize all the predefined web elements.

```
homePage = (HomePage) new HomePage(driver).get();
homePage.initializeWebElement();
```

- to write your test methods based on your test cases.

c. To add helper classes if necessary

You may also add some Helper or Utility classes to bring convenience to your test scripts. If necessary please feel free to add your owns.

The above coding examples are only in Java. C#'s syntax is quite similar to Java, so you can study from our sample code, or refer to a C# tutorial book.

5. **Test Execution**

   The Java source files of Eclipse project and IntelliJ project are exactly the same, but separated only for using in different IDEs. So you can use the test related functions in those IDEs to run unit tests and then check the results.
   In case of Java, you also can use the Maven command to run the tests without any IDEs: *mvn install* (or *mvn test*).

6. **To Do**

   As some members are using Mac OSX, so testing the Java projects under Mac OSX is necessary.

Any questions please feel free to let me know by either talking to me directly or emailing me at lei.wang@shiftinc.jp.