Основи проектування та розробки автоматизованих тестів для підвищення якості інтерфейсу.

**Мета:** Засвоїти методику та виробити практичні навички у проектуванні та створенні автоматизованих тестів за допомогою Selenium.

**Теоретичні відомості.**

Огляд Selenium

Selenium 2.0 (він же Selenium WebDriver) - це інструмент автоматизації функціонального тестування, який включає в себе два незалежних проекту - Selenium Remote Control і WebDriver. Selenium WebDriver являє собою набір бібліотек для керування браузерами. У Selenium 2.0 можна використовувати всі принади як Selenium WebDriver, так і Selenium RC (в режимі сумісності з WebDriver). Розробники інструменту рекомендують використовувати Selenium WebDriver в тих місцях, де не справляється Selenium RC. При роботі над Selenium 2.0 основна робота йшла над Selenium WebDriver і режимом сумісності з Selenium RC.

Головна відмінність, яка поділяє WebDriver і Selenium RC, полягає в способі взаємодії з браузером. Selenium RC посилає команди браузеру за допомогою спеціального JavaScript ядра Selenium Core. Даний підхід дозволяє забезпечувати кросбраузерність. WebDriver, на відміну від Selenium RC "спілкується" з браузером через незмінний інтерфейс. Для кожного браузера свій незмінний інтерфейс, накладає певні складнощі з підтримкою різних браузерів в WebDriver. Зате, надає ряд переваг, таких як швидкість роботи, дії користувача емулюється максимально точно.

Архітектуру Selenium RC можна представити таким чином: Авто тести -> Selenium RC Server -> Браузер -> тестований додаток.



В архітектурі Selenium WebDriver відсутній "прошарок" Selenium RC Server. Зате додається шар Driver, який і відповідає за взаємодію з браузером.



Головне достоїнство Selenium WebDriver полягає в тому, що він використовує драйвери, адаптовані під конкретний браузер, тобто Selenium WebDriver працює з кожним браузером за "індивідуальною програмою". Це підвищує стабільність роботи тестів (так як вони заточуються під конкретний браузер), тести стає простіше писати і підтримувати, збільшується швидкість їх роботи. Selenium WebDriver використовує незмінні команди (намагається повністю емітувати дії користувача), що є важливою перевагою перед Selenium RC.

На даний момент існують наступні драйвери:

FirefoxDriver;

InternetExplorerDriver;

ChromeDriver;

OperaDriver - в даний момент ще не доступний для використання;

SafariDriver - в даний момент ще не доступний для використання;;

HtmlUnitDriver - багатоплатформовий драйвер на java, не вимагає установки когось браузера;

AndriodDriver - мобільний браузер. Можливість тестування веб-додатків під телефони - ще одна фішка в Selenium 2.0;

IphoneDriver - мобільний браузер.

Список підтримуваних мов у Selenium WebDriver: C #, Java, Ruby, Phyton.

WebDriver пропонує досить гнучку роботу з очікуваннями подій на сторінці. Це пояснюється тим, що WebDriver спрямований в першу чергу на роботу з складним, перевантаженими AJAX запитами веб-інтерфейсами. Розробники пропонують два механізми очікування (Explicit Waits, Implicit Waits), Explicit Waits (експліцитне очікування), з використанням класу WebDriverWait, представляє сабою фактично циклічне очікування події. Implicit Waits (імпліцитне очікування) - всі команди пошуку елементів автоматично чекають. До речі, Selenium WebDriver вміє працювати з HTML 5! У Selenium WebDriver додалися нові типи локаторів: partionalLinkText, tagName, name. Повний список локаторів в Selenium 2.0:

**By.id -** в якості локатора використовується атрибут id (унікальний ідентифікатор) елемента сторінки;

Код сторінки:

<div id="coolestWidgetEvah">...</div>

Код тесту для сторінки:

IWebElement element = driver.FindElement(By.Id("coolestWidgetEvah"));

**By.name** - в якості локатора використовується атрибут name елемента сторінки;

Код сторінки:

<input name="cheese" type="text"/>

Код тесту для сторінки:

IWebElement cheese = driver.FindElement(By.Name("cheese"));

**By.xpath** - використовується для пошуку елемента по XPath виразу;

Код сторінки:

<input type="text" name="example" />

<INPUT type="text" name="other" />

Код тесту для сторінки:

IList<IWebElement> inputs = driver.FindElements(By.XPath("//input"));

**By.tagName** - пошук по імені HTML тега;

Код сторінки:

<iframe src="..."></iframe>

Код тесту для сторінки:

IWebElement frame = driver.FindElement(By.TagName("iframe"));

**By.className** - пошук по CSS класу елемента;

Код сторінки:

<div class="cheese"><span>Cheddar</span></div><div class="cheese"><span>Gouda</span></div>

Код тесту для сторінки:

IList<IWebElement> cheeses = driver.FindElements(By.ClassName("cheese"));

**By.cssSelector** - даний тип локаторів заснований на описах таблиць стилів (CSS);

Код сторінки:

<div id="food"><span class="dairy">milk</span><span class="dairy aged">cheese</span></div>

Код тесту для сторінки:

IWebElement cheese = driver.FindElement(By.CssSelector("#food span.dairy.aged"));

**By.linkText** - пошук посилання із зазначеним текстом;

Код сторінки:

<a href="http://www.google.com/search?q=cheese">cheese</a>>

Код тесту для сторінки:

IWebElement cheese = driver.FindElement(By.LinkText("cheese"));

**By.partionalLinkText** - пошук по частині посилання із зазначеним текстом.

Код сторінки:

<a href="http://www.google.com/search?q=cheese">search for cheese</a>>

Код тесту для сторінки:

IWebElement cheese = driver.FindElement(By.PartialLinkText("cheese"));

Для того, щоб почати розробляти авто тести на Selenium 2.0 під. Net потрібно завантажити набір бібліотек для. Net (*https://code.google.com/p/selenium/downloads/list*). У архіві повинні знаходиться наступні бібліотеки: Castle.Core.dll, Selenium.WebDriverBackedSelenium.dll, ThoughtWorks.Selenium.Core.dll, WebDriver.dll, WebDriver. Support.dll. Дані бібліотеки потрібно підключити в проект. Для запуску тестів, потрібно встановити NUnit. NUnit - це допоміжний framework для створення тестів. Завантажити NUnit можна за (http://www.nunit.org/index.php?p=download). Бібліотеки NUnit: nmock.dll, nunit.core.dll, nunit. framework.dll також потрібно підключити в проект.

Створення тесту

Відкриваємо Visual Studio і створюємо новий проект типу Class Library (Библиотека классов), додаємо згадані вище бібліотеки за допомогою команди **Проект-Добавить ссылку..**

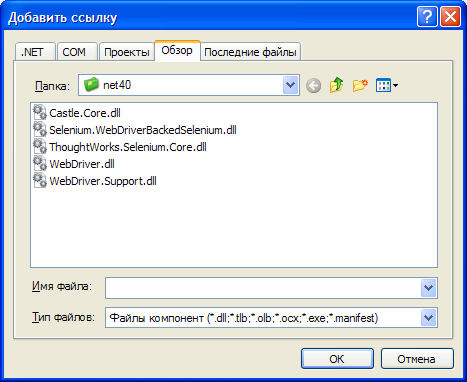
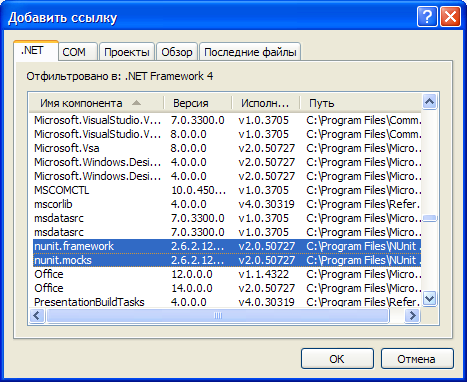
 

Рис. 1. Додавання бібліотек Selenium та NUnit

Після цього можна приступати до написання першого тесту. Кроки тесту будуть наступними:

1. Відкрити головну сторінку <http://bugscatcher.net>, перевірити title сторінки;
2. У полі "Пошук" внести значення "Selenium WebDriver";
3. Натиснути на кнопку "Search Submit" (лупа);
4. Перевірити результати пошуку.

Приклад тесту на Selenium 1.0 WebDriver з використанням C #:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using NUnit.Framework;

using OpenQA.Selenium;

using OpenQA.Selenium.Firefox;

using OpenQA.Selenium.IE;

using System.Threading;

namespace SeleniumTest

{

public class Class1

{

private IWebDriver driver;

private StringBuilder verificationErrors;

private string baseURL;

[SetUp]

public void SetupTest()

{

// Ви можете використовувати будь-яку реалізацію WebDriver.

//Firefox використовується тут в якості прикладу

driver = new FirefoxDriver();

baseURL = "http://bugscatcher.net";//посилання на веб-сайт

verificationErrors = new StringBuilder();

}

[TearDown]

public void TeardownTest()

{

try

{

driver.Quit();

}

catch (Exception)

{

//Ігнорувати помилки, якщо не в змозі закрити браузер

}

Assert.AreEqual("", verificationErrors.ToString());

}

[Test]

public void TheUntitledTest()

{

// open | /qa |

driver.Navigate().GoToUrl(baseURL);

// assertTitle | Bugs Catcher |

Assert.AreEqual("Bugs Catcher", driver.Title);

// type | name=s | Selenium WebDriver

driver.FindElement(By.Name("s")).SendKeys("Selenium WebDriver");

// click | id=searchsubmit |

driver.FindElement(By.Id("searchsubmit")).Click();

// assertTitle | Search Results Selenium WebDriver -Bugs Catcher |

Assert.AreEqual("Search Results Selenium WebDriver -Bugs Catcher", driver.Title);

//assertText present

Assert.IsTrue(driver.PageSource.Contains("Результаты поиска"));

//assertLink present

driver.FindElement(By.LinkText("ButerbroD"));

}

}

}

Перед запуском проекту потрібно сбілдити проект в Visual Studio (F6) і перевіртити, що немає помилок. Якщо все добре то:

Запускаємо раніше встановлений NUnit. У NUnit вибираємо "File -> Open Project ..." та в діалоговому вікні вказуємо шлях до. Dll нашого тесту (... bin\Debug\SeleniumTest.dll). Завантажуємо проект і натискає кнопку "Run" (її буде важко не помітити).

В результаті повинен запуститися написаний нами раніше авто-тест на Selenium. На рис. 2 зображено вигляд вікна NUnit-а з завантаженим тестом.

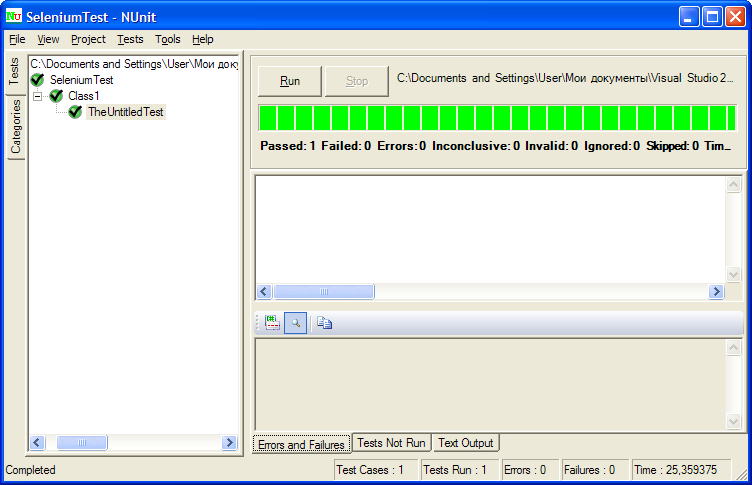


Рис. 2. Вигляд вікна NUnit-а з завантаженим тестом.

У даному прикладі використовувалися наступні команди Selenium WebDriver:

void GoToUrl (string url) - перейти за вказаною в "baseURL" адресою;

string Title {get;} - повертає Title активної сторінки;

FindElement (OpenQA.Selenium.By by) - пошук елемента сторінки за вказаною типом локатора. Наприклад, driver.FindElement (By.Name ("s")) або driver.FindElement (By.LinkText ("Випуск Selenium 2.0")). Список підтримуваних типів локаторів надано вище;

void SendKeys (string text) - ввести значення в текстове поле;

void Click () - натискання на елемент сторінки;

public bool Contains (string value) - повертає значення, що вказують, чи є введене значення частиною об'єкта.

Також розробниками Selenium 2.0 створений план по переходу від Selenium 1.0 до Selenium 2.0 в режимі WebDriver. План приблизно такий:

1. Використовуємо емулятор інтерфейсу Selenium 1.0 для Java і. Net. Для цього потрібно використовувати в ініціалізації WebDriverBackedSelenium замість DefaultSelenium.

IWebDriver driver = new FirefoxDriver();

Selenium selenium = new WebDriverBackedSelenium(driver, "http://www.yoursite.com");

1. Створення нових тестів і переписування старих під Selenium 2.0 в режимі WebDriver. Після того, як всі тести будуть готові працювати в режимі WebDriver в класі ініціалізації Selenium замість емулятора WebDriverBackedSelenium потрібно прописати команду запуску чистого WebDriver API:

IWebDriver driver = ((WrapsDriver) selenium).getWrappedDriver();

Як приклад візьмемо раніше розглянутий тест і і перетворимо його таким чином, щоб можна було використовувати команди як Selenium 1.0, так і Selenium 2.0. Після цього запустимо отриманий тест в змішаному "режимі".

Тест буде мати вигляд:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using NUnit.Framework;

using OpenQA.Selenium;

using OpenQA.Selenium.Firefox;

using OpenQA.Selenium.IE;

using System.Threading;

using Selenium;

namespace SeleniumTest

{

public class Class1

{

private IWebDriver driver;

private StringBuilder verificationErrors;

private string baseURL;

private ISelenium selenium;

[SetUp]

public void SetupTest()

{

// Ви можете використовувати будь-яку реалізацію WebDriver.

//Firefox використовується тут в якості прикладу

driver = new FirefoxDriver();

baseURL = "http://bugscatcher.net";//посилання на веб-сайт

selenium = new WebDriverBackedSelenium(driver, baseURL);

selenium.Start();

verificationErrors = new StringBuilder();

}

[TearDown]

public void TeardownTest()

{

try

{

//driver.Quit();

selenium.Stop();

}

catch (Exception)

{

//Ігнорувати помилки, якщо не в змозі закрити браузер

}

Assert.AreEqual("", verificationErrors.ToString());

}

[Test]

public void TheUntitledTest()

{

// open | /qa |

//driver.Navigate().GoToUrl(baseURL);

selenium.Open(baseURL);

// assertTitle | Bugs Catcher |

Assert.AreEqual("Bugs Catcher", driver.Title);

// type | name=s | Selenium WebDriver

driver.FindElement(By.Name("s")).SendKeys("Selenium WebDriver");

selenium.Click("searchsubmit");

selenium.WaitForPageToLoad("3000");

// assertTitle | Search Results Selenium WebDriver -Bugs Catcher |

Assert.AreEqual("Search Results Selenium WebDriver -Bugs Catcher", driver.Title);

//assertText present

Assert.IsTrue(driver.PageSource.Contains("Результаты поиска"));

//assertLink present

driver.FindElement(By.LinkText("ButerbroD"));

}

}

}

Також автоматизовані тести можна створювати в консольному проекті, тільки незабути під єднати dll бібліотеки Selenium.

Приклад тесту на Selenium 1.0 WebDriver з використанням консольного проекту:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using OpenQA.Selenium.Firefox;

using OpenQA.Selenium;

namespace ConsoleApplication25

{

class Program

{

static void Main(string[] args)

{

IWebDriver driver = new FirefoxDriver();

string baseURL = "http://bugscatcher.net";//посилання на веб-сайт

// open | /qa |

driver.Navigate().GoToUrl(baseURL);

// type | name=s | Selenium WebDriver

driver.FindElement(By.Name("s")).SendKeys("Selenium WebDriver");

driver.FindElement(By.Name("searchsubmit")).Click();

if (driver.PageSource.Contains("Результаты поиска"))

{

Console.WriteLine("існує");

}

else

{

Console.WriteLine("не існує");

}

Console.ReadKey();

}

}

}

Завдання

Виберіть довільний веб-сайт, спроектуйте та розробіть автоматизований тест, котрий би як мінімум проходив по всіх сторінках сайту, шукав довільний текст на сторінках а також працював з елементами керування на сайті.