# Web Testing

## Selenium IDE

Custom Formaters: <http://koenwillemse.wordpress.com/2011/02/16/custom-c-formatter-for-selenium/>

Instalar previamente:

* Firebug: <https://addons.mozilla.org/es/firefox/addon/firebug/>
* FirePath: <https://addons.mozilla.org/en-US/firefox/addon/firepath/>
* Chrome Path: <https://chrome.google.com/webstore/detail/hgimnogjllphhhkhlmebbmlgjoejdpjl>
* Batch Export: https://addons.mozilla.org/en-US/firefox/addon/test-suite-batch-converter-sel/versions/

Antes de empezar a crear las pruebas vamos a explorar primero la página Stackoverflow. Como sabemos es una página donde se escriben preguntas sobre programación y otras personas las responden. Una de las funcionalidades de esta página es su buscador que nos permite buscar posts tanto por palabras clave como por “tags”.

TestSuite: SearchPostsByTag

TestCases:

* SearchResultHasTheCorrectTag
* Relatated Tags

Descargamos Selenium IDE: Abrir Firefox e ingresar a la siguiente dirección: <http://seleniumhq.org/download/>

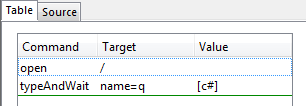
Abrimos Selenium: Ver- PanelLateral o Herramientas-SeleniumIDE.

Por defecto el plugin ya se encuentra en grabación. (Botón Rojo). Paramos la grabación por el momento.

Lo primero que tenemos que hacer para comenzar agravar algo es escribir la URL base de la página desde la cuál se van a comenzar a grabar las pruebas.

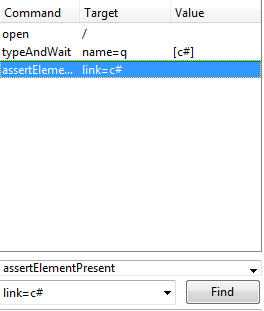
En la parte izquierda observamos la lista de tests cases que tenemos actualmente, podemos observar que no tiene aún un nombre, lo primero que vamos es dar un nombre que represente la prueba que queremos hacer. “SearchResultHasTheCorrectTag”.

* Escribimos [c#] en la barra del búsqueda.

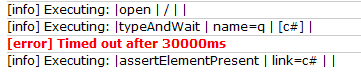


En la tabla podemos observar todas las acciones que están siendo grabadas, esta tabla tiene 3 columnas:

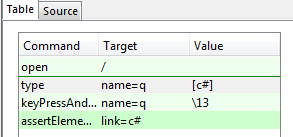
* + Command: El nombre del comando
  + Target: Representa el elemento sobre el elemento con el cuál estamos interactuando, este puede ser representando de muchas maneras como veremos luego.
  + Value: Son parámetros adicionales que son ingresados dependiendo del comando.
* Si hacemos click en alguno de estos comandos de la tabla podremos ver el panel inferior con los datos de este comando. A través de este panel podemos editar y crear nuevos elementos manualmente.
* En la parte inferior podemos observar documentación sobre el comando actual.
* Nos necesitamos asegurar que el resultado encontrado tiene el tag correcto. Es lo verificando que el link con el nombre “[c#]” se encuentra presente. Para esto utilizamos assertos. Para agregar un aserto solo tenemos que hacer click derecho en el elemento sobre el cuál queremos crear un aserto. Como sabemos que los tags están representados por links, el aserto más correcto sería constatar que existe un link con el texto “c#” en el primer resultado encontrado.



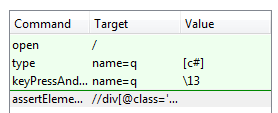
* Ahora q tenemos la prueba con el aserto, podemos ejecutarla para ver su funcionamiento. Vamos a observar que la prueba se detiene en el segundo comando y falla. De esto sacamos la primera lección al utilizar estas herramientas “No confiar en la grabación”, no son herramientas perfectas y muchas veces requieren que los comandos se modifiquen manualmente.
* En esta oportunidad la grabación no ha podido detectar que la búsqueda se ejecuta con el Enter y no con el click de un botón. Podemos observar el detalle del error en el log de la parte inferior.



* Para simular el enter en el teclado tenemos que agregar manualmente un comando y editar el comando type.



* Si observamos en los detalles del tag agregado podremos ver que su target es “link=c#”, este target podría no ser el más adecuado ya que es muy general (cambiar por otro texto y probar con el botón “find”) y tenemos muchos otros links con el mismo texto. Las forma más recomendada es utilizar el ID, pero en este caso nuestro elemento no tiene un id. Para esto usamos algo que se llama “XPATH”.
* Copiamos uno de los XPATH sugeridos y utilizamos el FirePath del Firebug para probar lo que significa.  
  //div[@id='question-summary-12138603']/div/div/a[contains(text(),'c#')]
* Con esto no quiero decir que siempre deban usar XPATH bastante precisos ya que hace que el test se vuelva más frágil a cambio. Cada vez que estén ingresando un locator deberán balancear la precisión de este con la fragilidad del test. Recordar que siempre es preferible usar el id en la medida de lo posible.
* Ejecutamos el test y grabamos.
* Creamos un nuevo test **ReleatedTags**.
* Grabamos ambos tests como parte del mismo Test Suite “SearchPostsByTag”.



//div[@class='module']/a[contains(@href, '/questions/tagged/c%23+.net')]

//h4[@id='h-related-tags']/following-sibling::a[contains(@href, '/questions/tagged/c%23+.net')]

* Ejecutamos y grabamos el test.
* Guardamos el testsuite “SearchPostByTag.testsuite.html”.
* Exportamos ambos testcases a C#. Podemos observar que el código generado pertenece a NUNIT y que tampoco ha podido exportar un elemento. Nueva lección: Selenium IDE no tiene una correspondencia directa con WebDriver y no es la forma correcta de crear scripts con WebDriver.

# Scripting

* Instalamos Selenium.Webdriver y Selenium Support
  + IE Problems: <http://www.jimmycollins.org/blog/?p=583>
  + Drivers: <http://code.google.com/p/selenium/downloads/list>
* Crear BD Bakery2 y cambiar el app.config y web.config con esa cadena.

## StackOverflow

[TestMethod]  
public void GettingStarted()  
{  
    //ARRANGE  
    var driver = new FirefoxDriver();  
    driver.Url = "http://www.google.com";  
  
    //ACT  
    driver.Navigate();  
  
    //ASSERT  
    Assert.AreEqual("Google", driver.Title);  
  
    //TEARDOWN  
    driver.Quit();  
}

[TestClass]  
public class StackoverflowTests  
{  
    private IWebDriver driver;  
    [TestInitialize]  
    public void Setup()  
    {  
        //driver = new InternetExplorerDriver(@"C:\Users\Snahider\Documents\Visual Studio 2010\Projects\ATes  
        //driver = new ChromeDriver(@"C:\Users\Snahider\Documents\Visual Studio 2010\Projects\WebTesting\Dri  
        driver = new FirefoxDriver();  
        driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(10));  
    }  
  
    [TestMethod]  
    public void ReleatedTags()  
    {  
        driver.Navigate().GoToUrl("http://stackoverflow.com/");  
  
        var textsearch = driver.FindElement(By.Name("q"));  
        textsearch.SendKeys("[c#]");  
        textsearch.Submit();  
  
        var relatedTags = driver.FindElements(By.XPath("//h4[@id='h-related-tags']/following-sibling::a"));  
        var exists = relatedTags.Any(x => x.Text == ".net");  
        Assert.IsTrue(exists);  
    }  
  
    [TestMethod]  
    public void ReleatedTags\_PageObject()  
    {  
        var stackoverflowPage = new StackoverflowPage(driver);  
  
        stackoverflowPage.Open();  
        stackoverflowPage.SearchFor("[c#]");  
  
        var relatedTags = stackoverflowPage.RelatedTags;  
        var exists = relatedTags.Any(x => x == ".net");  
        Assert.IsTrue(exists);  
    }  
  
    [TestMethod]  
    public void SearchResultHasTheCorrectTag()  
    {  
        driver.Navigate().GoToUrl("http://stackoverflow.com/");  
  
        var textsearch = driver.FindElement(By.Name("q"));  
        textsearch.SendKeys("[c#]");  
        textsearch.Submit();  
        //driver.FindElement(By.LinkText("save username but shows name in c#")).Click();  
        driver.FindElement(By.XPath("//div[@id='questions']/div[1]/div/h3/a")).Click();  
  
        var relatedTags = driver.FindElements(By.XPath("//div[@class='post-taglist']/a"));  
        var exists = relatedTags.Any(x => x.Text == "c#");  
        Assert.IsTrue(exists);  
    }  
  
    [TestCleanup]  
    public void Teardown()  
    {  
        driver.Quit();  
    }  
}

public class StackoverflowPage  
{  
    private readonly IWebDriver driver;  
  
    [FindsBy(How = How.Name, Using = "q")]  
    private IWebElement searchInput;  
  
    public StackoverflowPage(IWebDriver driver)  
    {  
        this.driver = driver;  
        PageFactory.InitElements(driver, this);  
    }  
  
    public void Open()  
    {  
        driver.Url = "http://stackoverflow.com/";  
    }  
  
    public void SearchFor(string query)  
    {  
        searchInput.SendKeys(query);  
        searchInput.Submit();  
    }  
  
  
    public IEnumerable<string> RelatedTags  
    {  
        get  
        {  
            return driver  
                .FindElements(By.XPath("//h4[@id='h-related-tags']/following-sibling::a"))  
                .Select(x => x.Text);  
        }  
    }  
}

## Bakery

### Organización

Separa los proyectos UnitTests de Integration y UI.

Representar todo en términos del usuario final.

Organizar los test por pantalla o por funcionalidad.

[TestClass]  
public class PurchasingCakes  
{  
    [TestMethod]  
    public void SeeProductDetailsWhenPlacingOrder()  
    {  
          
    }  
  
    [TestMethod]  
    public void PurchaseOneCake()  
    {  
  
    }  
}

**Primera Prueba (**

[TestMethod]  
public void SeeProductDetailsWhenPlacingOrder()  
{  
    //ARRANGE  
    //Load Data (Insert product with PRODUCTNAME y PRODUCTPRICE)

var driver = new ChromeDriver();  
  
    //ACT  
    //Navegar a la pagina  
    driver.Navigate().GoToUrl(ROOT\_URL + RELATIVE\_URL);  
    //Realizar click en un producto (nos muestra una nueva pantalla)  
    driver.FindElement("Find link by PRODUCTNAME").Click();  
  
    //ASSERT  
    //Verificar que se muestra el nombre del producto seleccionado  
    var productName = driver.FindElement("Find productname element").Text;  
    Assert.AreEqual("PRODUCTNAME", productName);  
    //Verificar que se muestra el precio del producto seleccionado  
    var price = driver.FindElement("Find price element").Text;  
    Assert.AreEqual("PRODUCTPRICE", price);  
  
    //TEARDOWN  
    //Cleanup Data

driver.quit();  
}

Completando los locators.

public void SeeProductDetailsWhenPlacingOrder()  
{  
    //ARRANGE  
    //Load Data (Insert product with PRODUCTNAME y PRODUCTPRICE)  
    //Configurar Driver  
    var driver = new ChromeDriver(@"C:\Users\Snahider\Documents\Visual Studio 2010\Projects\A  
  
    //ACT  
    //Navegar a la pagina  
    driver.Navigate().GoToUrl("http://localhost:6709/");  
    //Realizar click en un producto (nos muestra una nueva pantalla)  
    driver.FindElement(By.XPath("//h3[contains(text(),'Carrot Cake')]/../.././/a")).Click();  
  
    //ASSERT  
    //Verificar que se muestra el nombre del producto seleccionado  
    var productName = driver.FindElement(By.TagName("h1")).Text;  
    Assert.IsTrue(productName.Contains("Carrot Cake"));  
    //Verificar que se muestra el precio del producto seleccionado  
    var price = driver.FindElement(By.Id("orderPrice")).Text;  
    Assert.IsTrue(price.Contains("8.99"));  
  
    //TEARDOWN  
    //Cleanup Data  
    //Cerrar Driver  
    driver.Quit();  
}

Completando la inicialización de los datos y removiendo problemas.

[TestMethod]  
public void SeeProductDetailsWhenPlacingOrder()  
{  
    context.Products.Add(new Product  
        {  
        Name = "Apple Cake",  
        Description = "Default Descripcion",  
        Price = 8.99m,  
        ImageName = "carrot\_cake.jpg"  
    });  
    context.SaveChanges();  
  
    driver.Navigate().GoToUrl("http://localhost:6709/");  
    driver.FindElement(By.XPath("//h3[contains(text(),'Carrot Cake')]/../.././/a")).Click();  
  
    var productName = driver.FindElement(By.TagName("h1")).Text;  
    Assert.IsTrue(productName.Contains("Apple Cake"));  
    var price = driver.FindElement(By.Id("orderPrice")).Text;  
    Assert.IsTrue(price.Contains("8.99"));  
}

* Remover la inicialización de los datos en los test setup y teardown

        private static IWebDriver driver;  
   
        private AppDbContext context;  
   
        [AssemblyInitialize]  
        public static void SetupOnlyOnce(TestContext testContext)  
        {  
            driver = new ChromeDriver();  
        }  
   
        [TestInitialize]  
        public void Setup()  
        {  
            context=new AppDbContext();  
        }  
   
        [TestCleanup]  
        public void Teardown()  
        {  
            context.Database.ExecuteSqlCommand("Delete from Products");  
        }  
   
        [AssemblyCleanup]  
        public static void TearDownOnlyOnce()  
        {  
            driver.Quit();  
        }

* Page Objects

public class ChooseProductPage  
{  
    private readonly IWebDriver driver;  
  
    public ChooseProductPage(IWebDriver driver )  
    {  
        this.driver = driver;  
    }  
  
    public void Open()  
    {  
        driver.Navigate().GoToUrl("http://localhost:6709/");  
    }  
  
    public PlaceOrderPage GoToPlaceOrderForProduct(string name)  
    {  
        driver.FindElement(By.XPath("//h3[contains(text(),'"+name+"')]/../.././/a"))  
              .Click();  
        return new PlaceOrderPage(driver);  
    }  
}

public class PlaceOrderPage  
{  
    private readonly IWebDriver driver;  
  
    [FindsBy(How = How.TagName, Using = "h1")]  
    private IWebElement productName;  
  
    [FindsBy(How = How.Id, Using = "orderPrice")]  
    private IWebElement productPrice;  
  
    public PlaceOrderPage(IWebDriver driver)  
    {  
        this.driver = driver;  
        PageFactory.InitElements(driver, this);  
    }  
  
    public string ProductName  
    {  
        get  
        {  
            return productName.Text;  
        }  
    }  
  
    public string ProductPrice  
    {  
        get  
        {  
            return productPrice.Text;  
        }  
    }  
}

[TestMethod]  
public void SeeProductDetailsWhenPlacingOrder()  
{  
    DataFactory.Load(new Product  
        {  
            Name = "Apple Cake",  
            Description = "Default Descripcion",  
            Price = 8.99m,  
            ImageName = "carrot\_cake.jpg"  
        });  
  
    var chooseProductPage = new ChooseProductPage(driver);  
    chooseProductPage.Open();  
    var placeOrderPage = chooseProductPage.GoToPlaceOrderForProduct("Apple Cake");  
  
    Assert.IsTrue(placeOrderPage.ProductName.Contains("Apple Cake"));  
    Assert.IsTrue(placeOrderPage.ProductPrice.Contains("8.99"));  
}

**Segundo Test**