How to Develop Selenium Tests

Introduction

There are many ways to do any job. The following methodology and approach to Selenium testing is just one approach. There are other approaches and you should use an approach that works for you. The purpose of this article is to outline one integrated approach where someone new to ERSelenium and Selenium testing wants some direction on how to get effective robust Selenium testing implemented:

Pre-requisites and Assumptions

You have configured your application for "EntityDevelopmentDefaults" which saves you a lot of typing during general development of WebObjects pages requiring data entry and allows you tom quickly develop higher level Selenium tests that involve mostly clicking on page controls while the majority of data entry geta automatically taken care of by the "EntityDevelopmentDefaults" feature.

It is assumed you are using FireFox and the Selenium IDE and FireBug plugins

General Approach to Developing a Test

Using the Selenium IDE, you can record a test to get a feel for the syntax of Selenium commands, or to quickly see the syntax for a specific type of user interface interaction, however, recorded tests are generally very fragile and will fall apart on dynamic pages where the layout and content are not static. So before we discuss building robust tests, let's first itemize our objectives or criteria under which our tests should operate robustly:

- Localization - if user interface text in submit buttons, links, etc. can change depending on the selected language, then we don't want to depend on those for identifying ui elements in our Selenium tests
- Data entry - rather than tediously program data entry for new enterprise objects, it is worth considering adding a Auto Development Defaults feature that can be enabled in Properties. Such a feature would assign default values (random, fixed, serial or otherwise) to attributes in EOEnterpriseObject's awakeFromInsertion method. This is a much more powerful way to generate input values serverside in java that in Selenium. Consequently, your Selenium tests will become much shorter and simpler, resulting in them consisting mostly of user click interactions.

Developing a Test

- First fire up your WebObjects application and open the Selenium IDE
- In the Selenium IDE, insert commands for each user action.
- The most common commands you will use are clickAndWait, addSelection, open.
- Specifying targets can be really easy as long as you are willing to sprinkle passive test identifiers into your WOComponents.
    - The easiest approach is to assign css classes to the elements you want to click on and then use that to identify the HTML element that is being clicked. So while your WO app is running, you can add css identifiers to the static or dynamic elements in your WOComponent, then save and refresh the page. Using FireBug's Inspect feature will allow you to easily verify the presence of your locator identifier.
    - Other approaches:
        - Assign ids to html elements if you are sure the element will only ever appear once on the page. With the css class identifiers, Selenium will act on the first one it locates in the page.
        - Wrap an element in a span tag and assign a css class or id to the span tag. Then you can make a Selenium locator expression that can click on some element type inside the span container.

Probably the best way to understand Selenium command, target, value combinations is by looking at the Selenium Command Examples.

Using ERSelenium's meta tag features for advanced test development

Using @include meta tag

The @include tag is ideal for those common setup and teardown tasks. It is common for a test to go to the main application entry page and to login as its first step. Here is how we could use @include meta tag to make this a reusable.

Using @repeat, @values, @done meta tags

These allow you to specify portions of a test to be repeated.

The @repeat tag denotes the beginning of the portion to be repeated and the @done denotes the end of the portion to be repeated. These (can/cannot?) be nested?

The first pass thru the repeated segment, the actual values in the test are used and on consecutive repetitions the items from the @values list are substituted into the value part (the 3rd column) of the step immediately following the @values declaration

See also: Selenium

WIKITODO
* Flesh this out more with screenshots and examples for meta tags usage