MozMill Functional Test Automation Extension June 4, 2008

Purpose

To provide a tool that allows the community to efficiently build and execute functional tests against Mozilla applications in order to maximize QA efficiency and improve product release cycles.

Test Recording

Using event listeners attached to all application windows we can collect and analyze the event triggered and it's target. Using DOM id, name, JavaScript, XPath and RegEx we can build a reasonable way to access this element again in the future to "play back" the event that was fired. If the user is blocking the capturing phase with event.stopPropagation as well as event.preventDefault the test editing/writing GUI can be used to build this "action" manually. Since we are working in trusted mode the possibility of detecting that this event was fired even though it was not allowed to propagate to our listeners may now be a possibility as well.

Test Editing/Writing

Recording a user session in the GUI is very useful, but there are things that the recorder can't pick up at this time including timing, page loading issues and thorough assertions. This GUI will allow the user to manually, but still quickly build actions to execute against the application.

- The first tool is a JavaScript Shell with built in hooks for interacting with the controller to simulate and navigate the the DOM from the command line.

- The second is a DOM explorer which features quick integrated action building based on point and click against visible elements.

- The third is an Assertion Tool allowing point and click on the fly introspection of DOM Elements in order to generate assertions based on their state and contents. This will allow test builders to insert test actions that can verify nearly every aspect of the applications state at a certain point in time. This is a very important functional test feature that has shown very effective in catching regressions in the Open Source Chandler Calender Server AJAX UI.

Test Execution

Using the JavaScript Test Execution Framework originally written for the Windmill Testing Framework we can leverage the flexibility of functional tests written purely in JavaScript. This uses the Element Resolution Component to access the provided DOM

element and the Controller Component to replicate the events specified or recorded by the user.

This gives the test writer full control of their test and choice in whether they would like to write a test method as a JavaScript function in order to execute more advanced logic that can't be accomplished with the provided Controller API.

The JavaScript Test Framework is tied to the Results and Reporting components allowing MozMill to communicate it's testing state to many possible external sources for notification and analysis.

Compatibility

One of the concerns of functional testing on the web is the ability to compare functional cases against other browsers. Since the Windmill Testing Framework is cross browser and cross platform and uses similar JavaScript Test Framework, Controller and Element Resolution the MozMill JavaScript tests that don't involve Chrome components and target the window.document can be executed using Windmill against the other browsers as well.