

Proposal for a UI for WET

Raghu Venkataramana
Qantom Software Pvt. Ltd.

Table of Contents

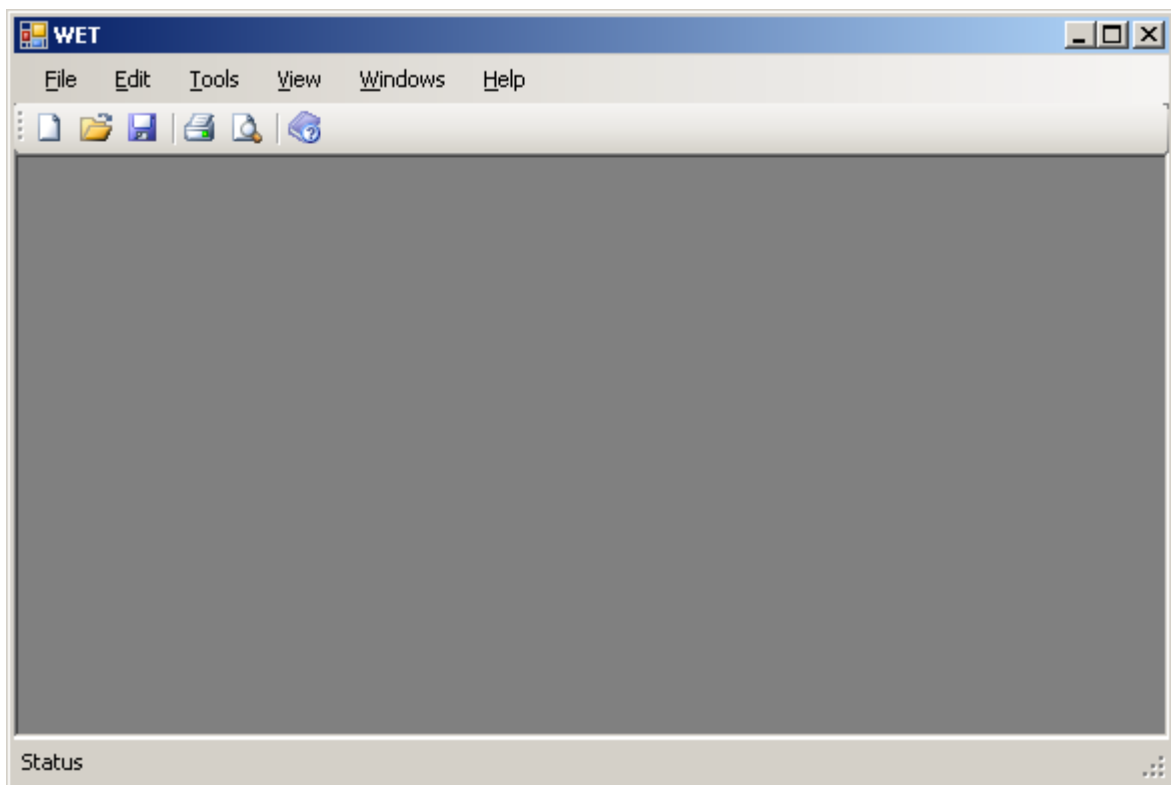
Introduction.....	2
Main WET UI.....	2
File Menu.....	3
Test Definition Window.....	4
Representation of Preconditions and Teardowns.....	7
Transactions.....	8
Browser Patrol.....	11
Object Repository.....	12
Object Identification.....	13
Configuration.....	14
Development IDE.....	14
A note about a recorder for WET.....	15
Conclusion.....	15

Introduction

WET is a framework for Web automation testing. WET has a lot of powerful features and has the potential to compete with most commercial tools. However the biggest impediment for WET is its usability and the amount of manual edits that have to be done in terms of creating and maintaining test definition files and creating XML repository files. In an effort to make the usability of WET a lot friendlier, it is proposed to have a UI for creating and managing WET Scripts.

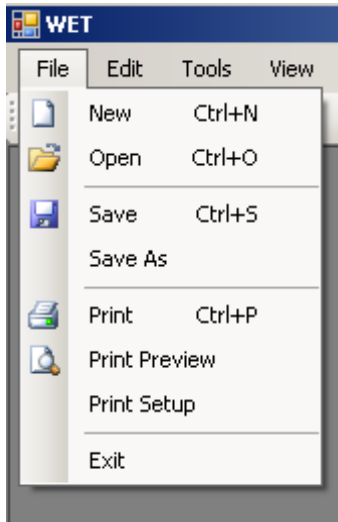
This document captures these proposals and form the base for arriving at a the Requirements for the WET UI. Please add all of your comments, suggestions and feedbacks to wet-users@lists.sourceforge.net.

Main WET UI



The above screen-shot illustrates how the WET UI is expected to look. It has the look and feel typical to most windows applications. It has both drop down menu items and tool bar menu items. The WET UI displays its windows in a MDI Window style. However at this time, it is not proposed that the WET application itself is MDI. That is, it will not be possible to open multiple WET tests simultaneously. The MDI Windowing technique will be used to display various windows related to WET Tests within the same UI. For example, Test Definition Window, WET Object Repository Window, Object Identification Window, etc.,

File Menu



The File menu will allow access to File Functions like New, Open, Save, Print and Exit.

File->New will start a new WET Test (Test definition)

File -> Open will open an existing WET Test

File -> Save / Save As can be used to save test definitions

File -> Exit will simply close the WET Application.

Note : Print has been added to the file menu. As of now, I am unclear as to what format the WET test should be printed out.

Test Definition Window

The Test definition is like the 'core' of a WET test. This is the place where you can control how the flow of actions / expected results in a test would look like. With a proper use of the test definition, you can achieve a very flexible and powerful way to control tests. However, partly due to a lack of proper documentation and partly due to the effort involved in creating test definitions with complex navigations (like preconditions and tear-downs) by hand. By having a UI to assist in this, will help even a Novice to easily get started on WET, while the more experienced hands can spend less time in constructing test definitions, object repositories, et. al. and utilize their effort in building libraries, frameworks, etc.,



Test Definition UI

The Test definition UI has two main components:

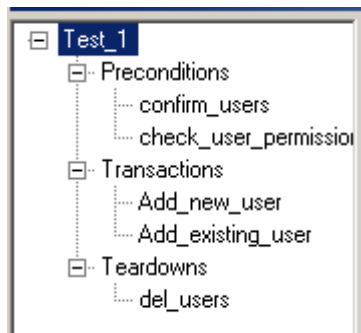
Left Pane has a treeview displaying the test's preconditions, tear-downs and transactions.

Right Pane has a panel which allows the automater to edit details about the test.

Details about the testdefinition view:

The left treeview displays a tests preconditions, transactions and tear-downs in a treeview. The name of the test appears as the root node. Under the root node, there are three treenodes – 1) Precondition nodes, 2) Transactions node and 3) Tear-downs node. Each of these nodes can be further expanded.

The Preconditions node has all the defined preconditions appearing as sub-nodes. The Transactions node has all the defined transactions appearing as sub-nodes, while the tear-downs node has all the defined teardowns appearing as sub-nodes.



Editing the name of the test : To edit the test's name, you may do so by directly editing the root node's text.

Edit test description: The Test's description can be edited by entering text into the text area displayed on the right hand side.

Setting the results path: To set the results path, use the 'open file dialog' option on the right hand side with the caption 'results path'

Setting the repository path: To set the object repository file, use the 'open file dialog' option on the right hand side with the caption 'repository'

Datatable : WET allows you to have a data driven approach by letting you select data from 'datatables'. These datatables can either be XL Spreadsheet or XML Files following the rules specified by WET Datatable XML syntax. These datatables are created separately and the test definition then points to this datatable by using the parameter `Common.testdata.path` parameter. With the UI, you no longer have to do this. Instead you can directly enter data in the Dataview component seen on the right hand side panel of the test definition UI.

	Item1	Item2	Item3
▶ C			
C			
*			

The UI also has an option to set whether the test has to be run on all rows of the datatable or only one(The first one). To do this you need to check or uncheck the 'Run on all iterations' checkbox



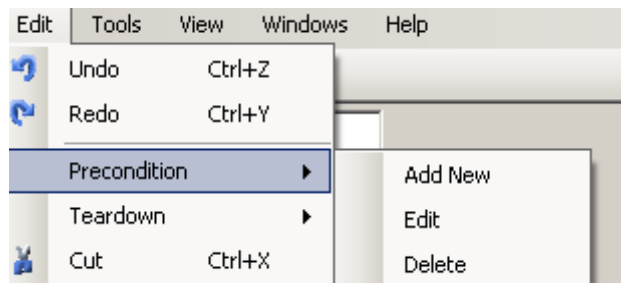
Now for how the Transactions, Preconditions and Tear-downs themselves are represented in the UI.

Representation of Preconditions and Teardowns

A precondition or teardown in WET is simply a call to another WET test residing else where. The use of preconditions and teardowns allows you to script your tests in such a fashion that they closely emulate real test cases. In terms of how WET handles Preconditions and Teardowns, both are exactly identical. So I will use "Preconditions" as the feature to propose about both.

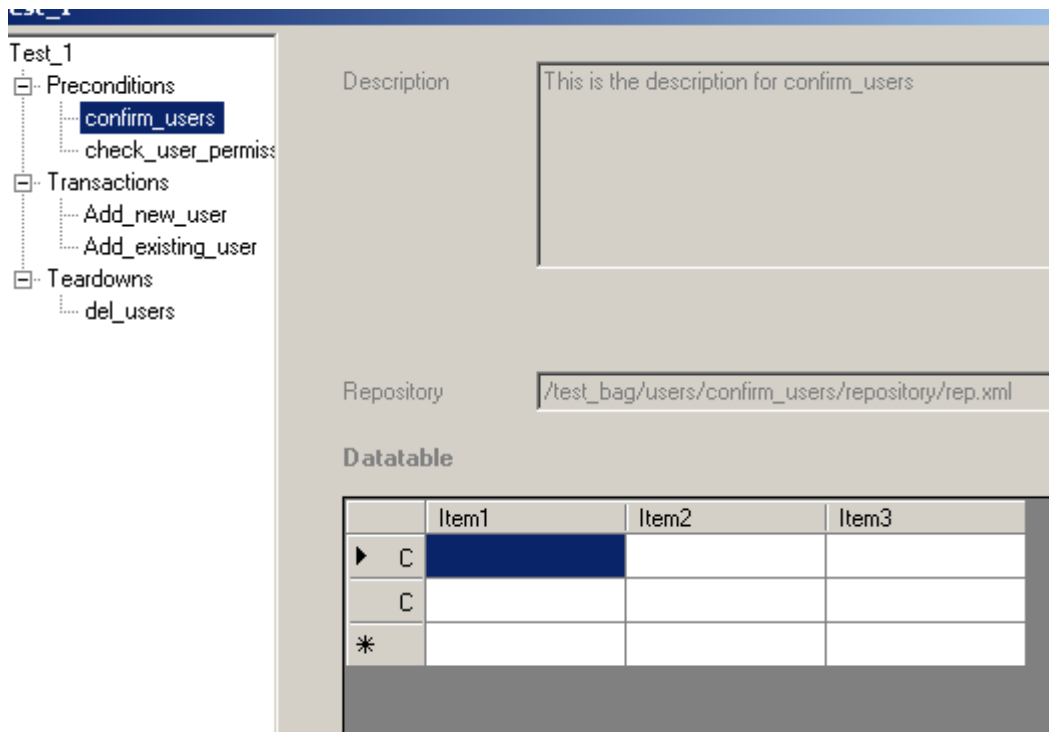
Adding a precondition

To add a new precondition, use the Edit -> Precondition option and say 'Add New'



This will present a dialog where you can select the test to be added. After you add the precondition, it appears as one of the **preconditions** to this test as seen by the left side tree view of the test definition. On the right hand side, a view similar to the main *Test Definition* right panel is seen. However in this case the right hand side view is not editable.

Editing a precondition



Precondition View

To edit an existing precondition, use the Edit -> Precondition option and say 'Edit'. This will open the test corresponding to the precondition which can now be edited.

This feature may be put on hold for now. The reason is that the first version will not allow multiple tests to be opened in the same MDI window

Removing a precondition

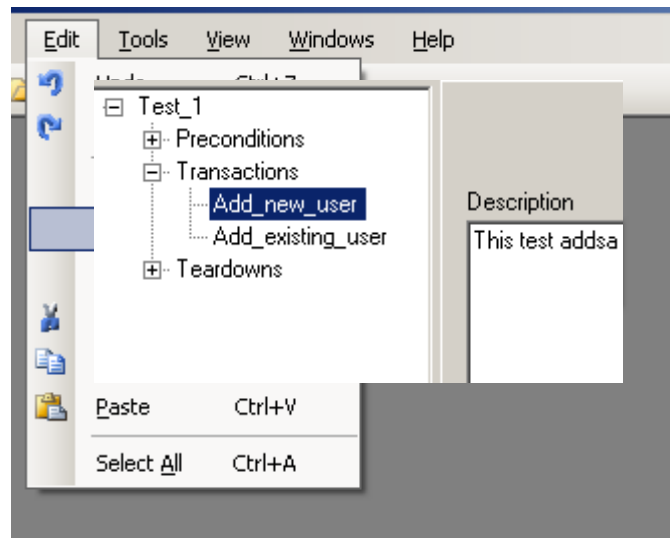
To add a new precondition, use the Edit -> Precondition option and say 'Delete'. This will remove the precondition from the test.

Removing the precondition only removes the call to the precondition from this test. It does not delete the precondition test itself.

Transactions

The main objective of a WET script is realized by means of transactions. Using the WET UI it should be possible to easily create test transactions. To create a new transaction, say

Edit->Transactions->Add New



After you do this, a new transaction is added as seen by left hand side Test Definition treeview.

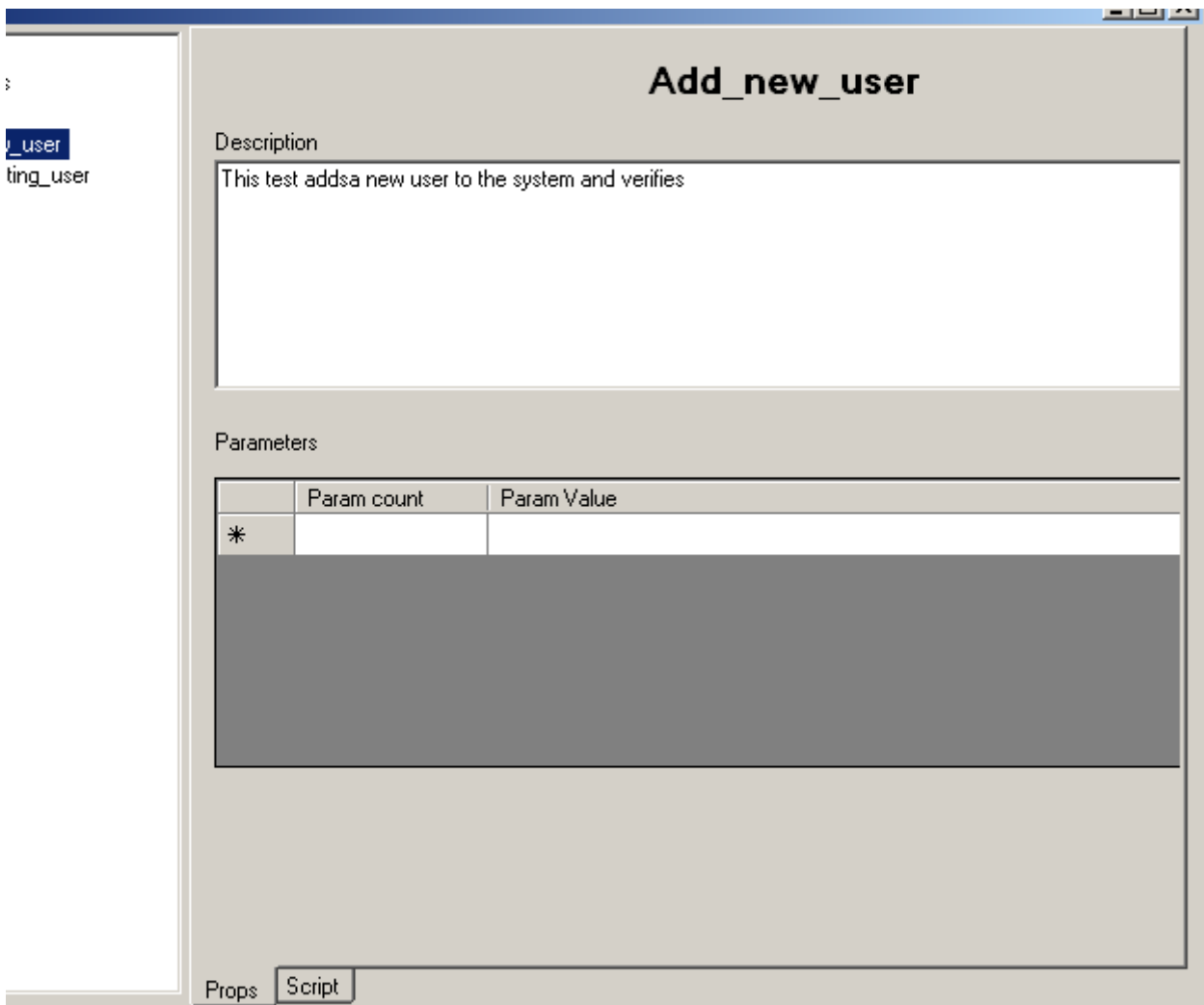
On the right hand side, a tabbed panel representing the transaction is displayed. It has two tabs. The first tab allows you to enter the properties of the transaction – description and parameters.



The second tab is for editing the actual script tied to the transaction.

Properties View:

The properties view (seen by clicking on the *Props* tab) displays the properties associated with this transaction. Currently, the properties of a transaction are a Name, Description and Parameters. To edit the name of the transaction, you need to edit the treenode's label. The description is a textarea and the parameters are displayed in a datatable view.



A Transaction as seen using its properties View

Script View

This is simply a large notepad like panel in which the transaction's script can be viewed and edited.

```

tions
ions
new_user
.existing_user
ns

Browser.new().goto("http://wet.qantom.org/handson.html")
Browser("title:=Submit a bug").check_text(/^Bug Submission Form

Browser("title:=Submit a bug").TextField("name:=Name").set "Big
Browser("title:=Submit a bug").TextField("name:=Email").set
"tester@example.org"
Browser("title:=Submit a bug").TextField("name:=BugSummary").set
existing employee hangs the system"

Browser("title:=Submit a bug").List("name:=ProductType").select
management"
Browser("title:=Submit a bug").List("name:=DefectType").select "
Crashed"

Browser("title:=Submit a bug").Textarea("id:=ml1", "index:=1").s
trying to add an existing employee as a new employee, the applic
crashed"
Browser("title:=Submit a bug").Textarea("name:=Notes").set "Win
Browser("title:=Submit a bug").Textarea("id:=ml1", "index:=2").s
not want to reveal that"

Browser("title:=Submit a bug").Button("value:=Submit Query").cli

table = Browser("title:=Form Output").Table("text:=/^Bug Details
table.Row("index:=2").check_property("text", /Description *When
add an existing employee as a new employee, the application cras
table.Row("index:=3").check_property("text", /BugSummary *Adding

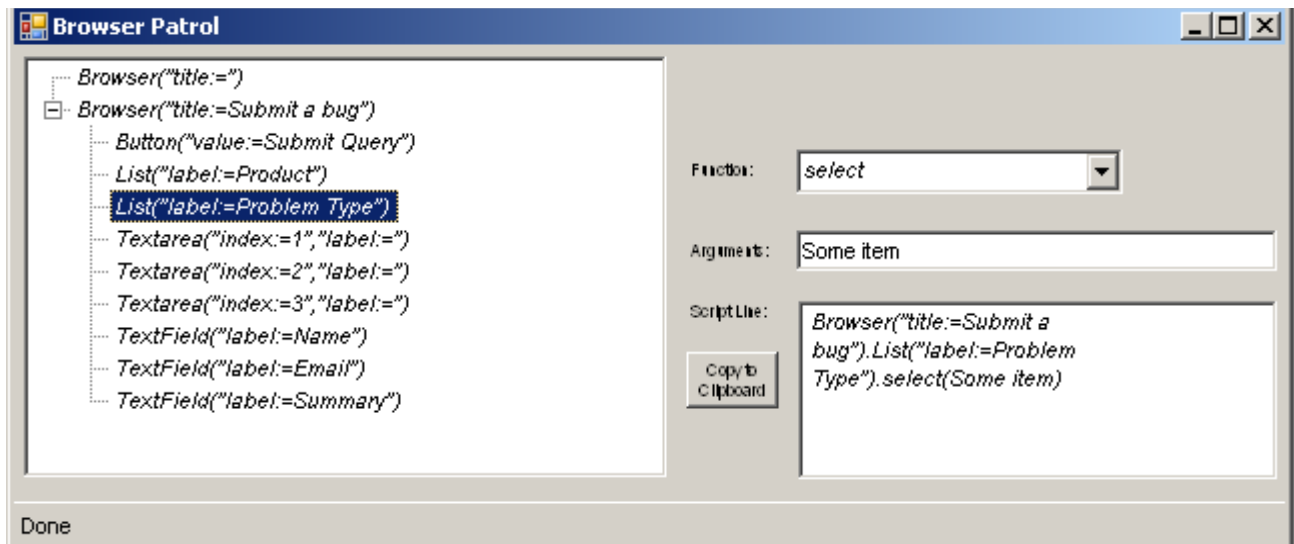
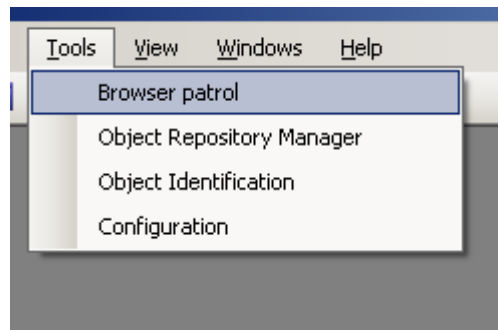
```

A Transaction as seen using its script View

Browser Patrol

The browser patrol is a slick utility that monitors open IE browsers and represents the DOM object in a treeview that is integratable with WET. This utility was written by a co-user, *Venugopal Shenoy*. The Browser Patrol also has the ability to directly produce Script lines that can be copy-pasted into WET scripts. Another powerful feature of the Browser Patrol is its ability to convert this view directly to a XML Object Repository format. To see more about the Browser Patrol and its associated utilities, check the zip file at http://openqa.org/wet/downloads/WETScriptAssistant_0.1.zip. It has a user document to get you started.

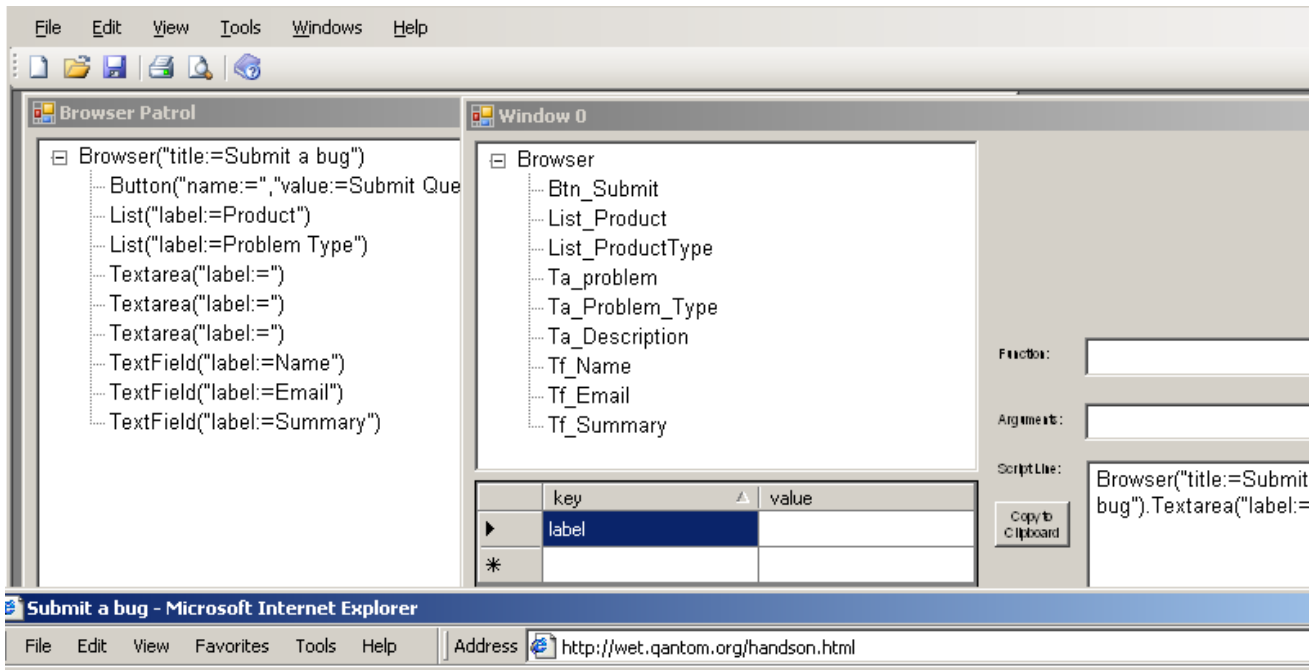
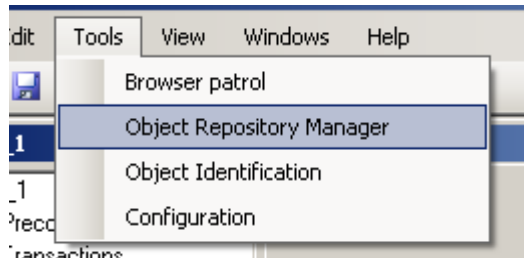
The Complete WET UI will integrate the browser patrol as an integral part of the UI. You should be able to access the Browser Patrol by clicking on Tools -> Browser Patrol



Browser Patrol – does a lot more than Patrolling!

Object Repository

The Object Repository of WET (in an XML format) is one of the most powerful features of WET. It lets you organize your scripts in a highly maintainable way. However, the biggest inhibition towards using the object repository is the difficulty involved in creating a repository. By providing a UI for creating the object repository, this difficulty can be removed and it would be a lot easier for scripters to use the object repository feature. The Script Assistant utility contributed by Venugopal Shenoy is already a great step in that direction. I propose to integrate his utility into the UI that is to be built for WET.



Bug Submission Form

Please submit bugs that you encounter while using the Qantom.org products using the following form. Enter details about yourself: (Leave email field blank if you dont wish to be contacted)

Name Email

Please give a one line summary about the bug

Summary

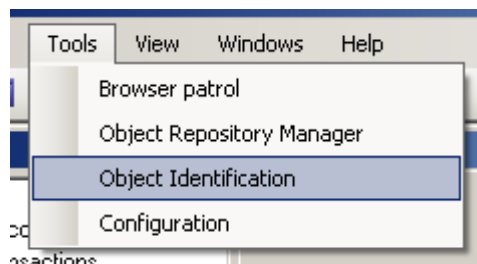
In which product did you see the bug? How serious do you think the problem is?

Product Problem Type

Please describe the problem in detail (Include the steps to reproduce in your description)

Object Repository

Object Identification

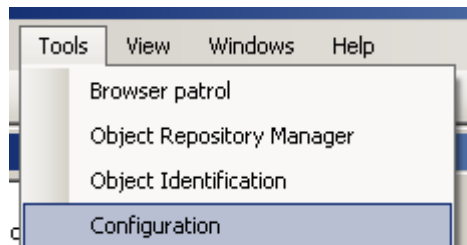


To configure the way objects are identified, click on Tools -> Object Identification. This will open up a window that lets you configure how the DOM objects are displayed. Currently WET uses the object_identification.xml to configure how show_objects displays the DOM tree view. This configuration should be used by the BrowserPatrol to display the object tree.

Note :- I haven't yet created a prototype for the Object Identification window. The Window for the object identification feature should have the ability to edit all the elements and attributes of the object identification xml file.

Configuration

Using the Tools->Configuration menu option, you will be able to configure the global-configs.txt



Note : The prototype for this has still not been created. The expected UI for this could be a tabbed dialog box in which each tab is used to edit one category of the global configuration file. This is the typical fashion in which configuration is edited using a UI.

Development IDE

For the development of the WET UI, the IDE to be used is Visual C# Express Edition which is available for download from Microsoft's website. Before arriving at MS VC# as the desired IDE, a few other options were considered were:

WX Ruby – WET being written in Ruby, this seemed to be the most obvious choice. However WX Ruby doesn't seem to have all the components that are required for a good UI. Most importantly, ruby doesn't handle threads too well (remember the click_no_blocking method – it has to spawn a click request on an entirely new process – this was due to the threading issues with Ruby). Without Threading, one cannot accomplish a great deal in UI development.

Java – Java was a very strong contender in the game and was riding strong. The only reason Java didn't make it was the fact that it was unclear as to how Java would have interacted with Ruby and moreover, since the WET code bases interacts a lot with WIN32 objects, java didnt win.

Mono Dot Net with GTK # - This is a cross-platform port of the DotNet Environment and of C#. This has a lot of potential and was almost was finalized. The only flip side to this was the difficulty in creating the UI using Mono.

Based on all of the above research Microsoft Visual C# Express Edition was chosen to be the IDE for the WET UI.

A note about a recorder for WET

There has always been an argument of whether or not script recorders should be an important part of any test automation tool, recorders will continue to play some role in the test automation space. Although I personally, am not an advocate of recorders, I have seen in my experience that recorders are very useful utilities in making a quick start in the automation project and for evaluation of whether or not, the tool is suitable for the application intended to be tested.

However mainly due to lack of time and due to the fact that the other features mentioned in the previous sections are far more important when compared to recorders, the recorder is not planned as a part of the first version of the WET UI. I hope that it will make it into WET on a later date.

Conclusion

WET, has a lot of potential in becoming a leading test automation tool for Web Application testing. However some of the most powerful features of WET like Object Repository, Preconditons & Teardowns, Library scripts, Datatables, etc., are not put to as much use as they could be. The two main reasons for this are a) Lack of proper documentation and b) Difficulty in creating the test definitions / configurations manually. The former will have to be fixed by means of thorough documentation, while the UI Proposal is a right step in the direction of addressing (2).

The above is just a proposal of how the WET UI should look & feel and is not a user document for an application that has been completed.

Please send all of your comments, suggestions and feedback on this proposal to wet-users@lists.sourceforge.net or post your comments in the Jira issue that has been written to track the UI Issue - <http://jira.openqa.org/browse/WET-23>