

Eclipse and LiquidTest UI Introduction

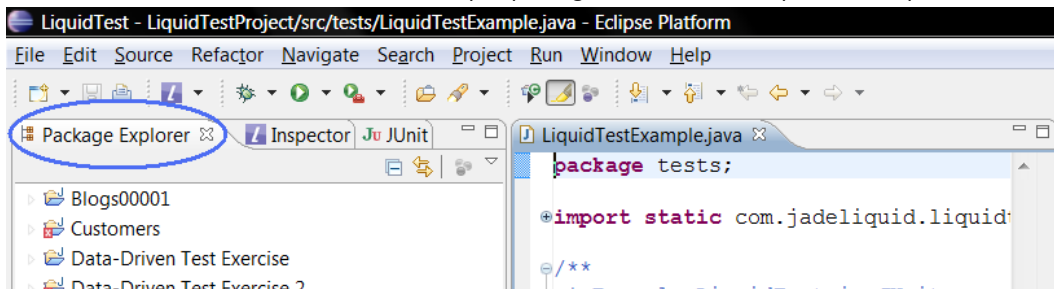
LiquidTest is available as a plug-in for the two most popular Integrated Development Environments (IDE), Eclipse and Visual Studio. This discussion will focus on the Eclipse version and will provide Eclipse-specific tips for LiquidTest users.

Testers and/or Developers will benefit from some of the features that are available in the IDE (Eclipse) itself, that are not part of the LiquidTest plug-in. Additionally, this discussion will cover some features and functions that are available in the Eclipse UI due to the LiquidTest plug-in.

Eclipse-Specific Features:

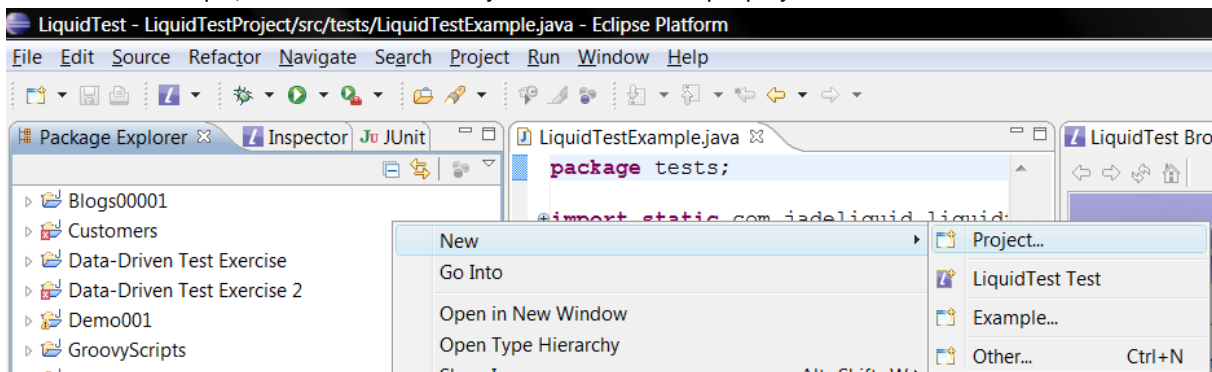
Eclipse Package Explorer tab

Eclipse uses the Package Explorer tab to organize projects, JRE system library files, the LiquidTest plug-in jar files, the JUnit files and the source folder and the Eclipse packages that hold the LiquidTest scripts.



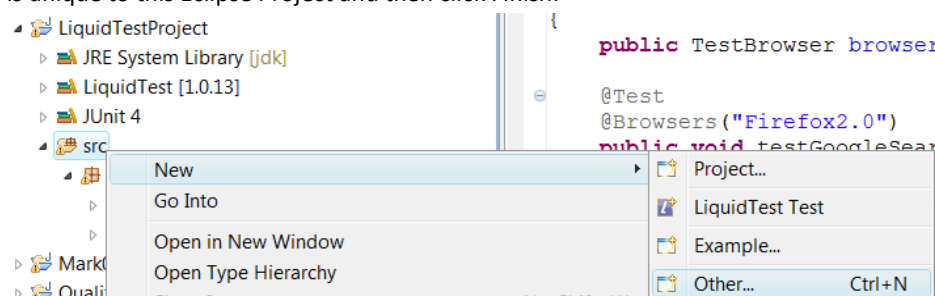
Eclipse Project creation

A user may want to create a new Eclipse project to hold LiquidTest scripts. To do this, select the Eclipse Package Explorer tab and either select File->New->Project or right-click in the Eclipse Package Explorer tab and select New->Project. For a project that will store Java/JUnit/Groovy (LiquidTest Script) test scripts, select Java->"Java Project". For a project that will store .NET test scripts, select .NET->".NET Project". Provide a unique project name and click Finish.



Eclipse Package creation

Within an Eclipse project, you can further refine the location that LiquidTest uses to store scripts. It is recommended that you create a new package to store scripts. To do this, in the Eclipse Package Explorer tab, right-click the "src" folder in the Eclipse Project that you have just created and select New->Other. Next, select Package, provide a package name that is unique to this Eclipse Project and then click Finish.



Eclipse Perspective button

Eclipse Perspectives are used to customize the actions and views that are available to the user. Various tabs, buttons, windows and areas are visible when a particular Perspective is selected via the “Perspective” button in the upper right-hand corner of the Eclipse UI.

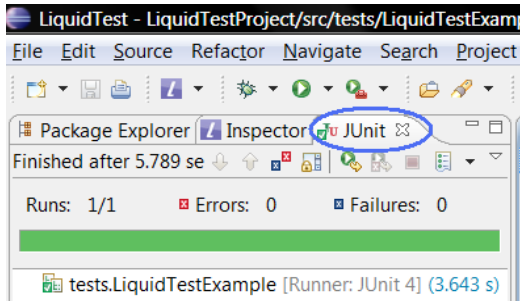


JUnit tab

The JUnit tab is used to track the status of a currently running LiquidTest test case, including iterations, errors and failures.

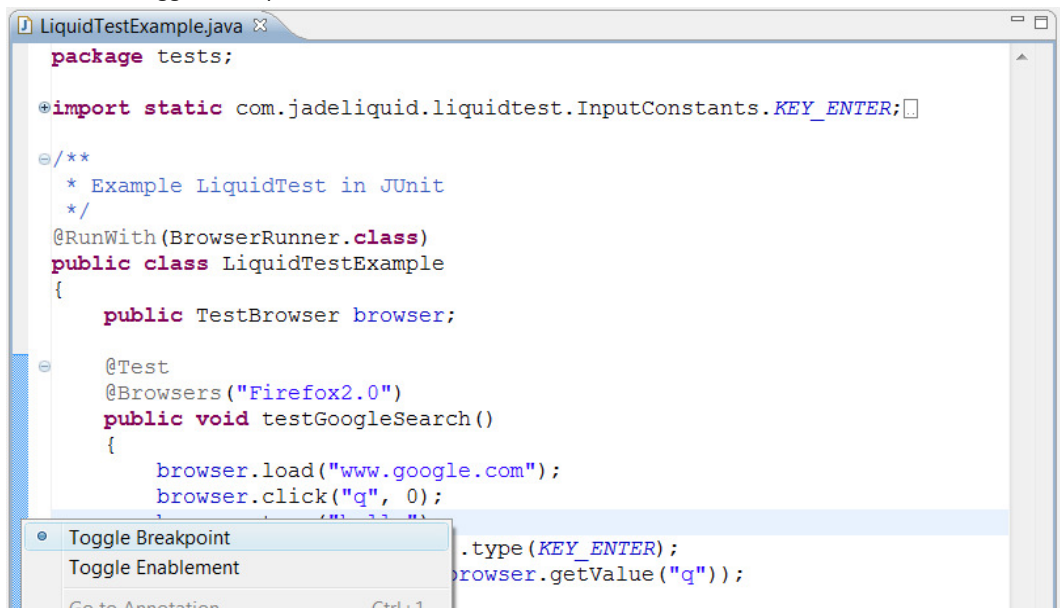
- Failure Trace area

The Failure Trace area allows you to view the details of any failures that were detected in the test case run.

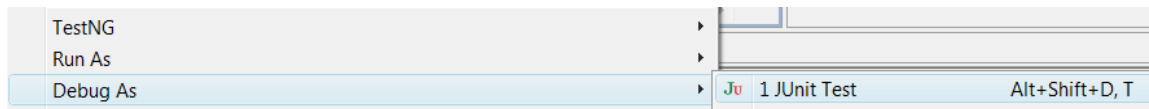


Break-points (for Java JUnit Test Scripts only)

Most LiquidTest users will not need to use break-points but if you need to (for example to pause the test execution to validate something), create your test script as usual, then in the LiquidTest Test code/script tab for the test of interest, select the line you want to set a breakpoint on, then in the area near the left-hand border of the test script tab, right-click and select “Toggle Breakpoint”.

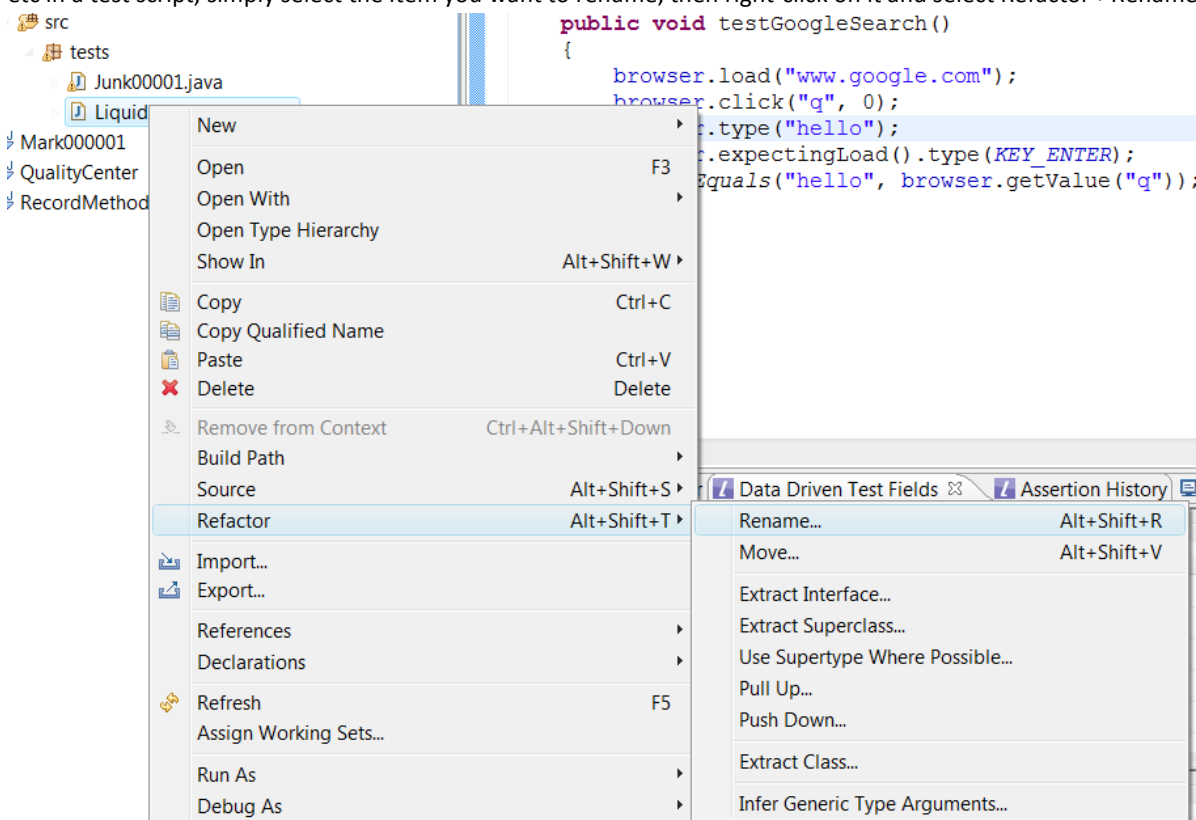


To run the test case and use the breakpoint, you must use “Debug As” by right-clicking on the script and selecting “Debug As” and then selecting the appropriate test type (in this case, JUnit). When the playback gets to the breakpoint, execution will pause. To continue, use the Run->Resume menu item (or F8).



Refactoring (for Java JUnit Test Scripts only)

It is common for software projects to be refactored frequently to reorganize their form, without changing the external functionality. This involves activities such as renaming files, classes, methods, variables, etc. If this had to be done manually, it would be very difficult to know all of the places where this update must be made. Fortunately, Eclipse has a facility to do this automatically. This can be done in more than one way. In the Eclipse Package Explorer tab, you can rename a test script file (or other types as well) by right-clicking on that file and selecting Refactor->Rename. There are also other types of refactoring that are available on this menu as well. If you want to rename a class, method, variable, etc in a test script, simply select the item you want to rename, then right-click on it and select Refactor->Rename.

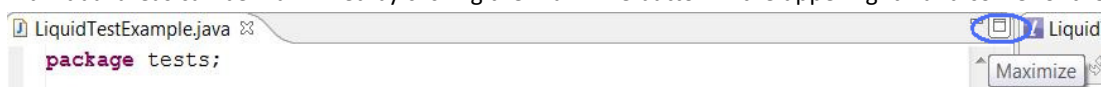


How to manage the areas of the Eclipse UI

The Eclipse UI is fairly flexible in how it presents information to the user.

- Maximizing /Minimizing/Restoring areas

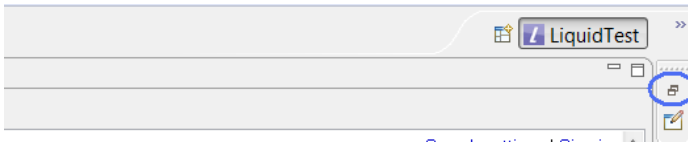
Individual areas can be maximized by clicking the Maximize button in the upper right-hand corner of the area.



Areas can be minimized by clicking the Minimize button in the upper right-hand corner of the area.

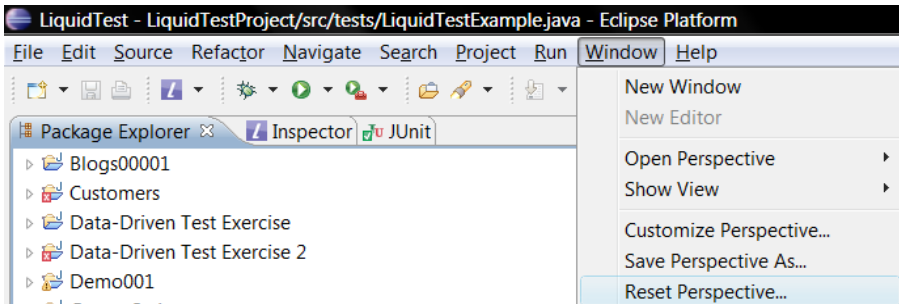


Areas can be restored by clicking on the Restore button for that area (which will be located to the far left and/or right sides of the screen).



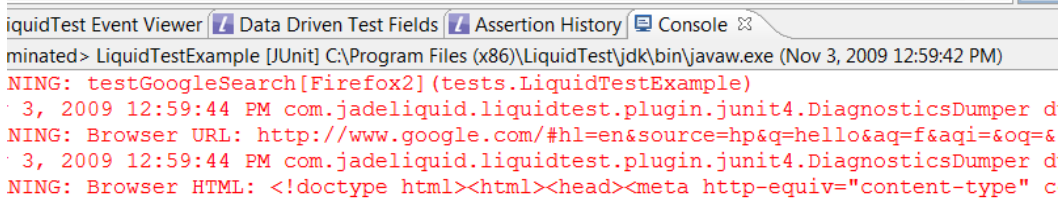
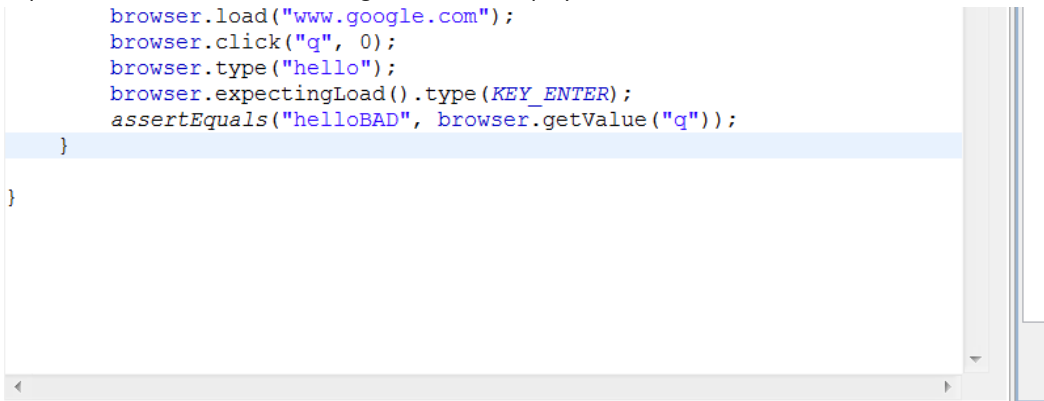
- Reset Perspective

As the user works with LiquidTest, they can make changes to the Eclipse Perspective as desired to customize the way information is displayed. To restore the default settings for a Perspective, make sure the Perspective is active (by using the Eclipse Perspective button) and then select Window->“Reset Perspective”.



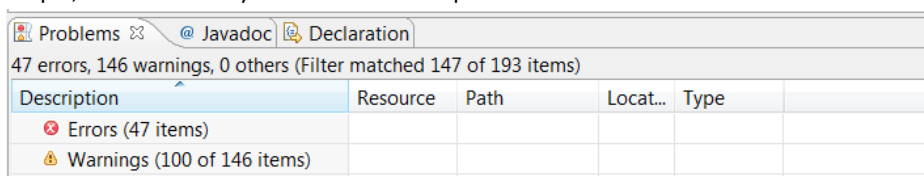
Console tab

The Console tab provides detailed information about warnings, errors and any other unexpected events that are experienced when the test is being recorded or replayed.



Problems tab

This tab is only available in the Java perspective and provides detailed warnings about the test script code, such as code errors; methods, identifiers and asserts that are never used; type warnings, imported objects not used; etc. For this example, I have manually edited the test script to insert intentional errors.



LiquidTest Plug-in Features:

LiquidTest Perspective

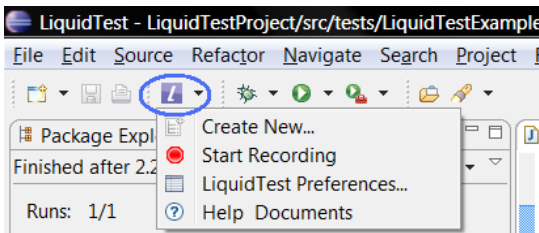
LiquidTest takes advantage of Eclipse perspectives to customize the actions and views that are available to the user. Various LiquidTest-specific tabs, buttons, windows and areas are visible when the LiquidTest Perspective is selected via the “Eclipse Perspective” button in the upper right-hand corner of the Eclipse UI. You can verify that Eclipse is in the LiquidTest perspective by checking that LiquidTest is displayed in the upper right-hand corner.



Like other Eclipse Perspectives, if the LiquidTest Perspective has been modified, the user can restore the LiquidTest Perspective to its default configuration by first selecting the LiquidTest Perspective, then selecting Window->”Reset Perspective”. For more information on Eclipse perspectives, see: <http://www.eclipse.org/articles/using-perspectives/PerspectiveArticle.html>

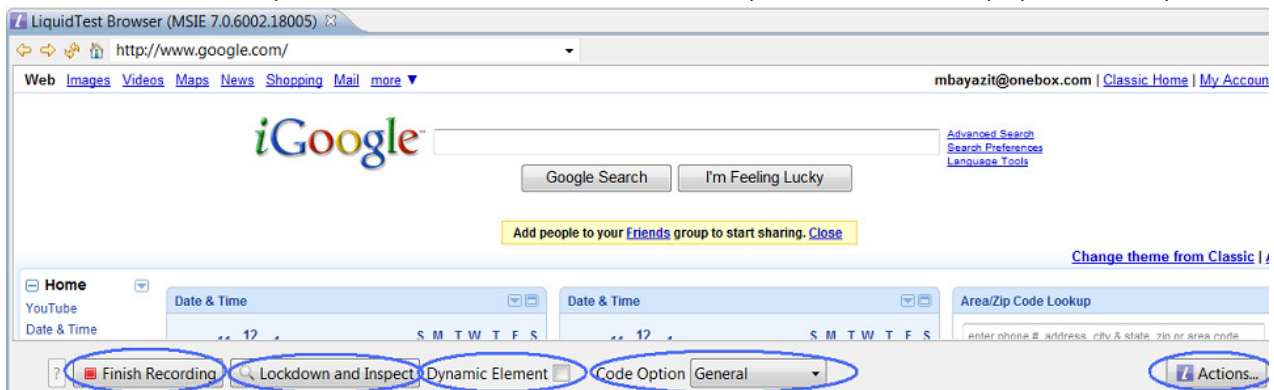
LiquidTest button

The LiquidTest button is located directly below the Eclipse menus (File, Edit, Source, etc.) and allows the user to create a new LiquidTest test script, Start Recording, Set LiquidTest Preferences and access online LiquidTest documentation.



LiquidTest Browser window

This is the window the LiquidTest user interacts with to create test scripts and observe the replay of these scripts.



- “Finish Recording”
This button is used to finish a recording session.
- “Lockdown and Inspect”/ “Unlock and Continue”
This button is used to allow several test creation actions to be accomplished, such as setting up data-driven form fields, using the LiquidTest Inspector and creating assertions to validate content. While the user is recording a test script, click “Lockdown and Inspect”. Then use the Inspector, data-driven and input field and/or create an assertion, then click “Unlock and Continue”.
- Dynamic Element
Allows a user to specify that this object’s identification will change on different iterations of the test, so simply identify and locate the object by the full XPath.
- Code Options

Pick which Code Option to use from a list of established Code Options. Code options allow the user to define how HTML objects are identified and are particularly useful for new web frameworks. For more information on code options, see: http://www.jadeliquid.com/liquidtest/docs/doku.php?id=general:recording_options

- LiquidTest Actions

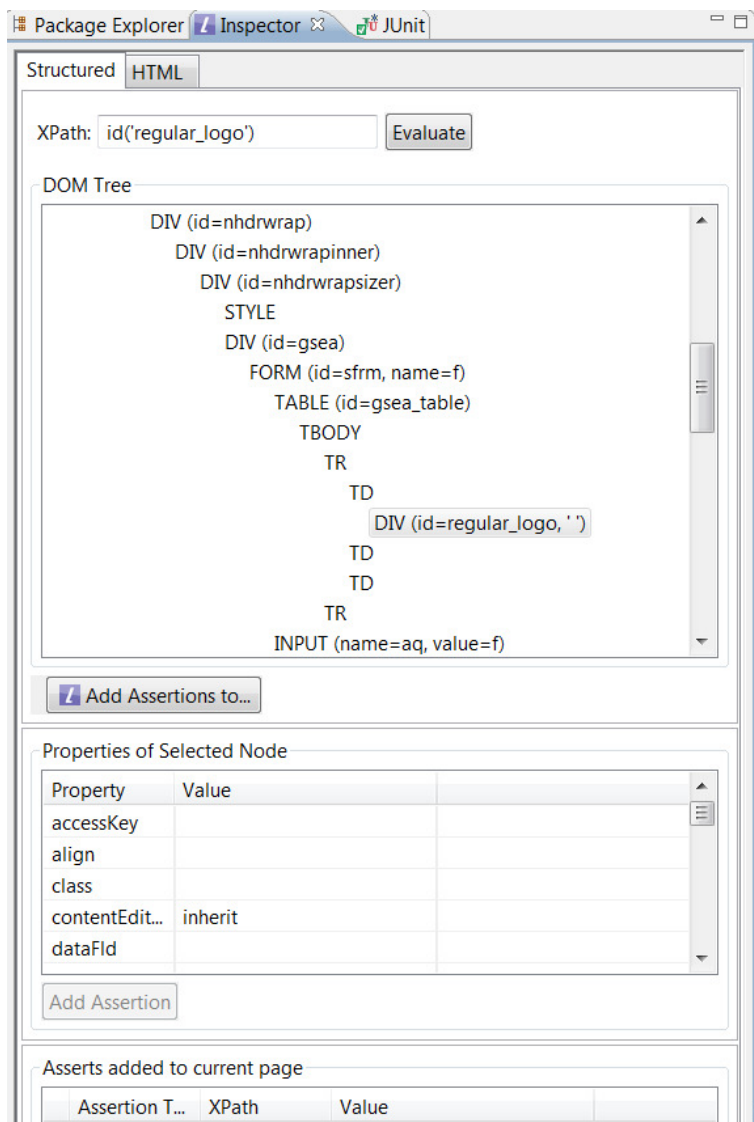
Allows the user to attach an Excel spreadsheet for parameterization data and to insert or run methods from other files.

LiquidTest Inspector tab

This tab is used when recording scripts, using the “Lockdown and Inspect” feature and for selecting the desired object to create an assertion to validate that content is appearing correctly or to parameterize data. To use this feature, start recording a script, navigate to the point where you want to use the Inspector and click “Lockdown and Inspect” and then click on the object of interest.

- Structured tab

The Structured tab shows a lot of useful information about the HTML elements that appear on the page.



*XPath

Full XPath of the HTML element.

*DOM tree

An orderly display of the HTML elements, including some information such as the name, id and value of the element.

*Properties of Selected Node

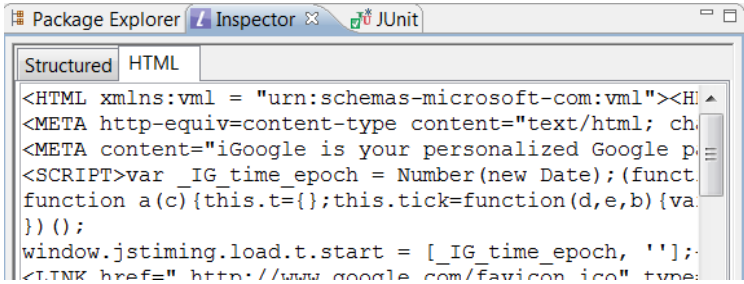
This area allows you to observe and select all the available properties of the selected element and also allows you to assert on any selected property.

*Asserts added to current page

Shows a listing of all the assertions that have been created for this page.

*HTML tab

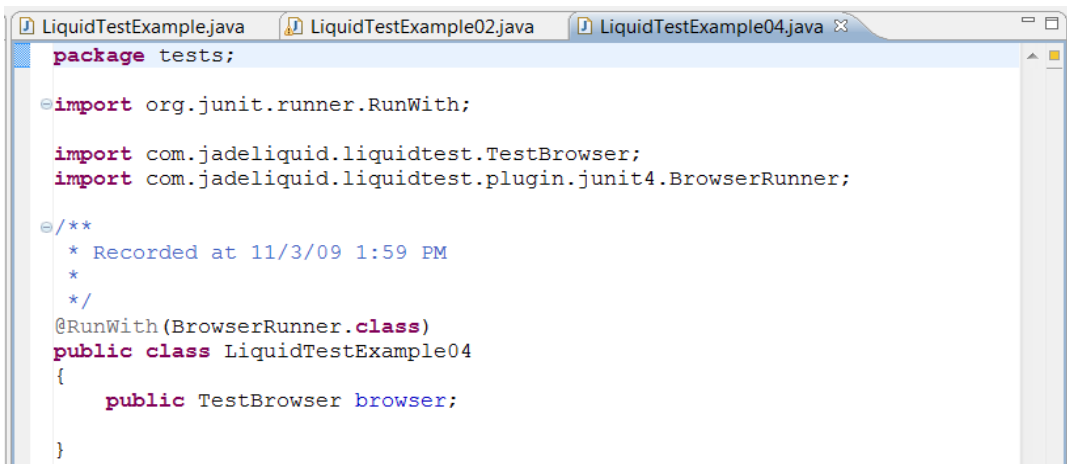
Shows the HTML code that has been downloaded to the browser.



LiquidTest Test code/script tabs

LiquidTest uses one area to group one or more LiquidTest Test cases which are selectable by clicking their individual tabs.

Within these tabs, the user can observe the code, modify and save changes, run the test and more.



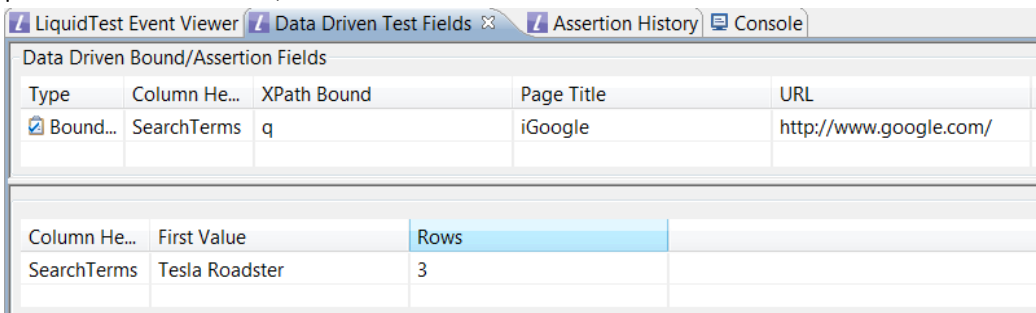
LiquidTest Event Viewer tab

This tab is located near the bottom of the Eclipse UI and allows the user to see what HTML events have occurred at any point during the recording process. This is useful to ensure that an activity has been completed before the test case recording engine proceeds to the next step.



LiquidTest Data Driven Fields tab

This tab is located near the bottom of the Eclipse UI and allows the user to see the properties of all the data-driven parameterization of fields, as well as the data-driven asserts that have been created for this test script.



The following information is displayed:

- Type – Either “Bound Field” or “Assertion”;
- Column Header – Which column is being used in the spreadsheet for this item;
- XPath Bound – What XPath or identifier is being used to identify the element being data-driven;
- Page Title – What is the title of the HTML page;
- URL –The URL of the HTML page.

LiquidTest Assertion History tab

While the LiquidTest script is being recorded, this tab shows which assertions have been created and information about: Assertion Type – What type of assertion is being performed (text content, attribute, etc.);

- Verifying Value – The value that is being asserted;
- Verifying XPath – The XPath or other identifier being used to identify the object;
- Page Title – Title of the HTML page.
- Page URL – Full URL of the page being tested.

Assertion T...	Verifying Value	Verifying XPath	Page Title	Page URL
Text Content			iGoogle	http://www.google.com/
Text Content	Advanced Search	Advanced Search	iGoogle	http://www.google.com/
Text Content			iGoogle	http://www.google.com/
Attribute	regular_logo	id('regular_logo')	iGoogle	http://www.google.com/

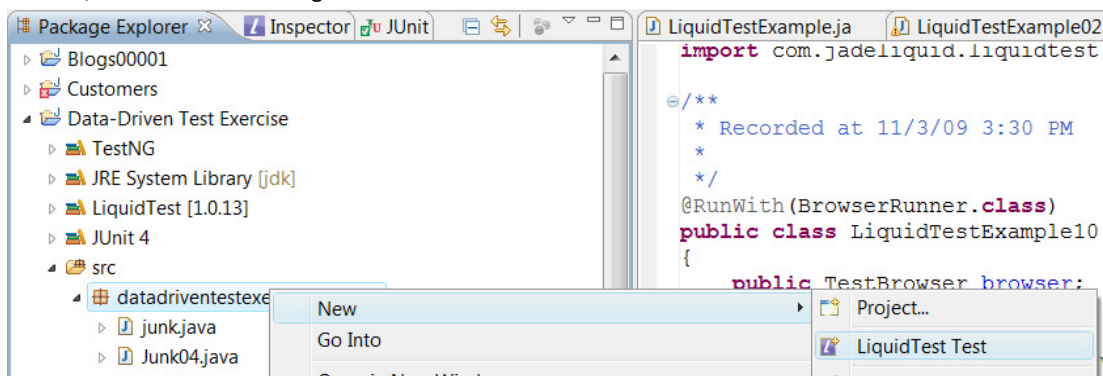
LiquidTest Script Runner tab

This tab only appears when you are running a LiquidTest Script (Groovy) test case and provides technical information about a LiquidTest Script as it is being run, such as warnings, diagnostic messages, pass/fail etc. This is useful for script debugging.

Test Name	Browser	Status	Problem Description	Problem Trace
testMethod	Firefox...	PASSED		

LiquidTest Script Creation

To create a new LiquidTest test case - In the Eclipse Package Explorer tab, open the project that you want to store the test script in, then right-click the specific package you want to store this test script in and select New->”LiquidTest Test”. Change the Test Case Writer selection if needed, provide a name for the test case, add a data-driven spreadsheet if desired, click “Start Recording Now” and then click Finish.



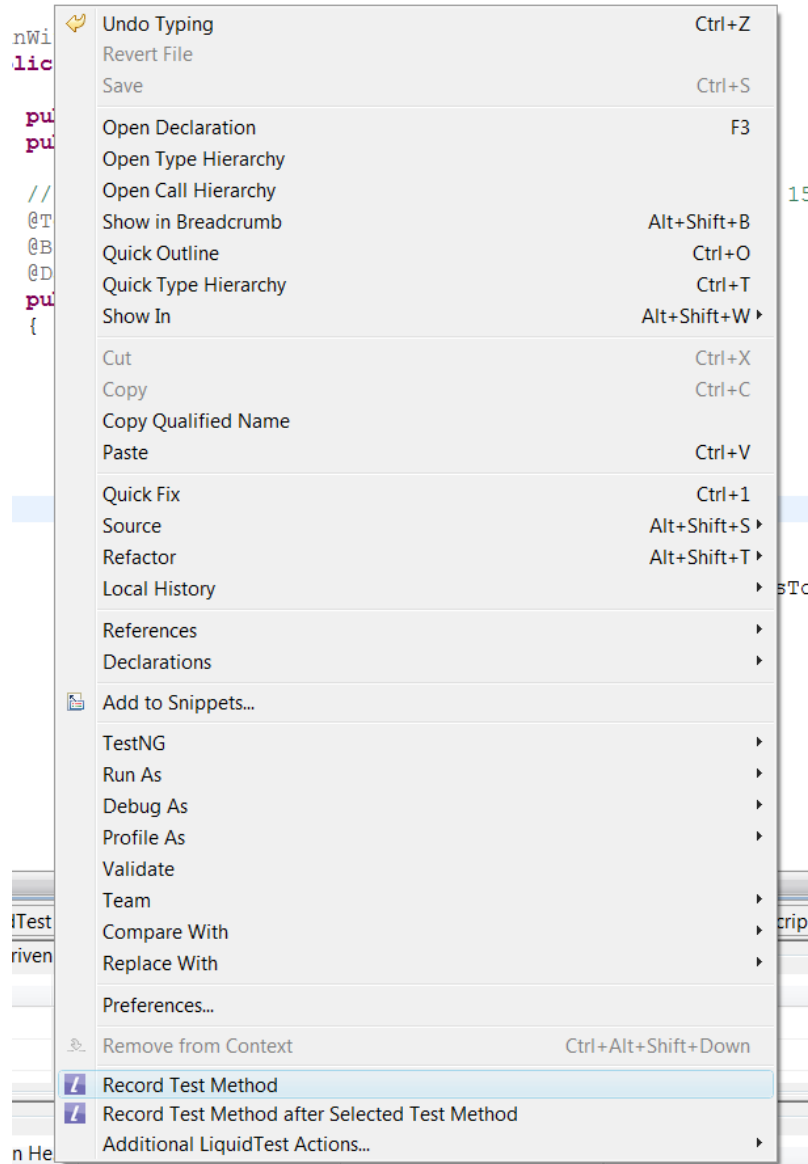
Right-Click on Test Script menu-options (Run, Record, etc.)

Some actions can be invoked easily by right-clicking on the test script of interest and selecting:

- “Run As”->”JUnit Test” (or “Ruby Application” or “TestNG Test”). This runs the test case;
- “Record a Test Method” to start recording a new test method;
- “Record Test Method After Selected Test Method” to replay the script to this point and start recording a new

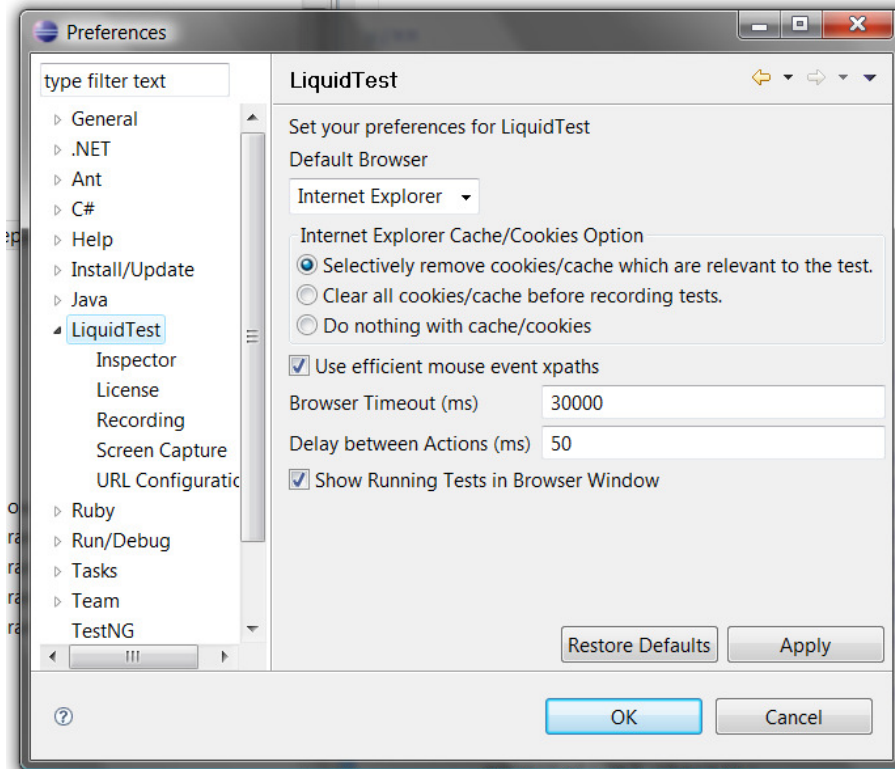
test method;

- “Additional LiquidTest Recording Actions”->”Record Test Method from Current Page” to just start recording on the page that is currently loaded into the LiquidTest Browser tab;
- “Additional LiquidTest Recording Actions”->”Record Test Method After Selected Methods from Files” to replay a test method from a different file and then start recording;
- “Additional LiquidTest Recording Actions”->”Record Test in Current Method” to replay the test method back and then start recording in the same test method.



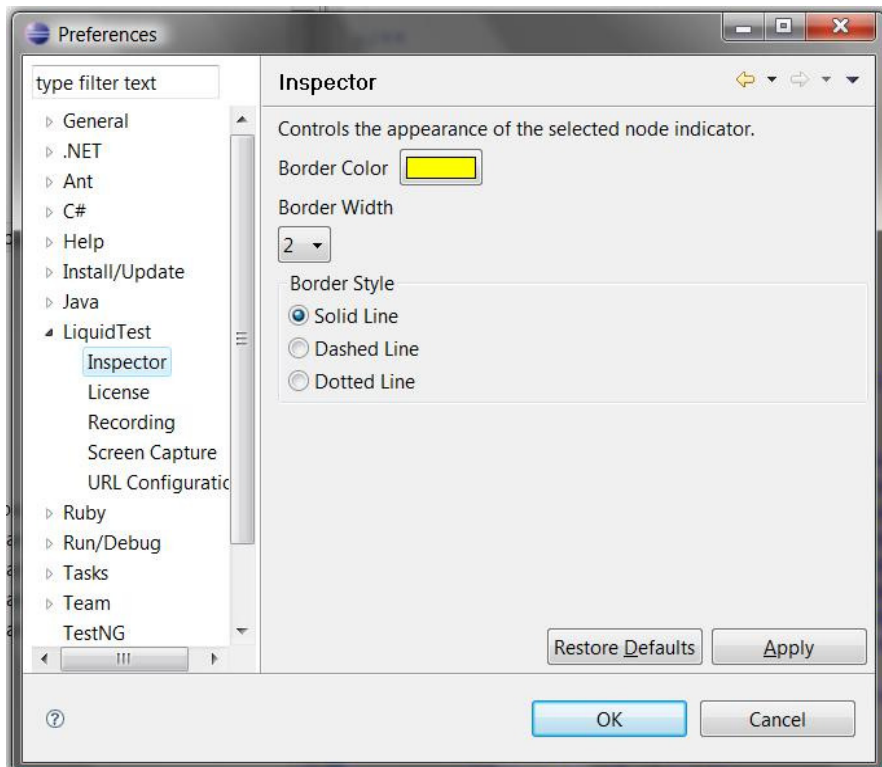
LiquidTest Preferences

This can be accessed via the LiquidTest button drop-down menu or the Eclipse menus via Window->Preferences->LiquidTest. Some basic preferences can be set at this level (such as default browser, cookie handling, mouse event handling, timeouts/delays and showing running tests).

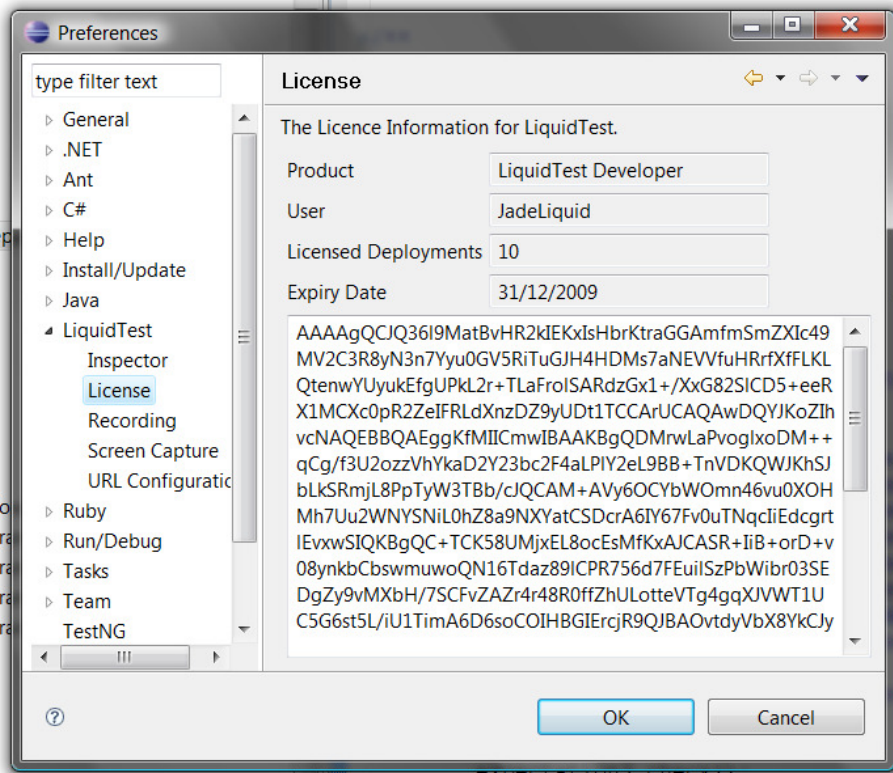


More preferences can be set by selecting the LiquidTest sub-preferences:

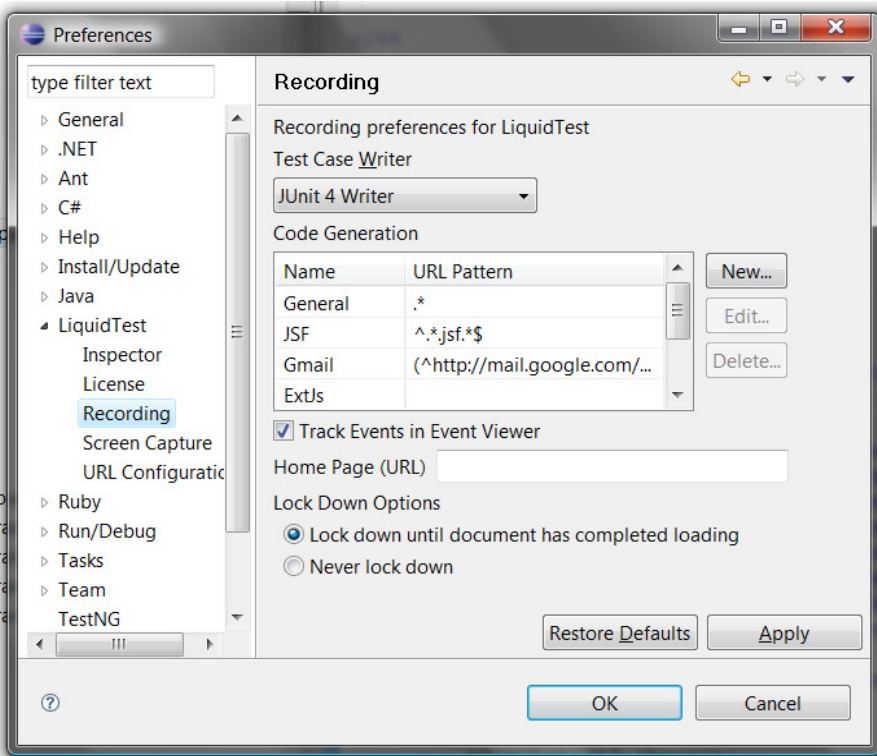
- Inspector – Set the border color, width and style for highlighting objects in the Inspector.



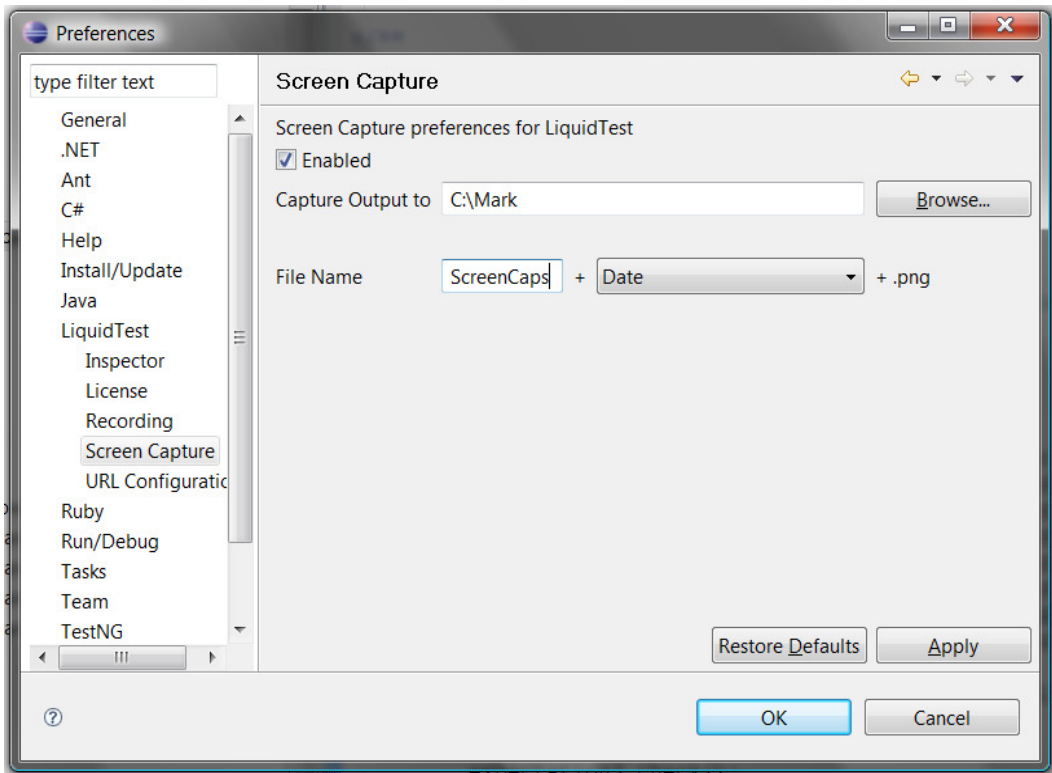
- License – View license details and update the license key.



- Recording – Set default test case type, code options, event tracking, default URL and lockdown options.



- Screen Capture – Set preferences for where to store screen captures when test cases fail.



- URL Configuration – Set URL replacement preferences to allow you to create scripts on one website and then play them back against another (for example, scripting against a test environment and running them against a production environment).

