



QEngine White Paper

Automating Dynamic Web Application with QEngine

Table of Contents

<u>Abstract</u>	<u>3</u>
<u>Introduction</u>	<u>4</u>
<u>Prerequisites</u>	<u>4</u>
<u>Trouble Spots in Automation</u>	<u>5</u>
<u>Handling Applications with Varying Response Speeds</u>	<u>5</u>
<u>Handling Application Window Title / Size / Content Changes.....</u>	<u>6</u>
<u>Handling Window Element Changes</u>	<u>7</u>
<u>Handling Applications with Dynamic Windows & Element.....</u>	<u>12</u>
<u>Handling UnExpected Application Popup during playback.....</u>	<u>14</u>
<u>Handling Application Server Host and Port Changes.....</u>	<u>15</u>
<u>Conclusion.....</u>	<u>16</u>

Abstract

Web sites and applications are becoming crucial to the success of businesses and hence need to be tested thoroughly and frequently. However, these applications change constantly and the challenge of test automation projects is in dealing with these changes.

Successful automation is possible only if the automation tool can handle the changes and allow reuse of the automation project. In this paper, we present an overview of how QEngine allows you to reuse the test automation project even after the application has changed.

Introduction

The purpose of this paper is to outline the provisions of QEngine product that allows users to reuse the test scripts when the application has changes. Reuse of scripts offers tremendous productivity gains in terms of time and effort.

For further information about QEngine, visit the URL:

<http://www.manageengine.com/products/qengine/index.html>

Prerequisites

- Working knowledge of the web application to be tested.
- Basic working knowledge of using ManageEngine QEngine Web Functional Testing Tool.
- Understanding of testing concepts.

Audience

- Quality Assurance Engineers.
- Users of QEngine who need to automate their Web Application functionality testing.

Trouble Spots in Automation

Applications under test may change and they change constantly and the challenge of test automation projects is dealing with these changes. Successful automation is possible only if the automation tool can handle all of the scenarios described below:

1. [Handling Application with varying response speeds based on the system / network in which it runs.](#)
2. [Handling Application Window changes in terms of size, title content based on the user input.](#)
3. [Handle Application Window Element changes based on user input or for a new release.](#)
4. [Handling Application Windows and Elements that vary dynamically during playback.](#)
5. [Handling Applications with dynamic Windows & Elements.](#)
6. [Handling Unexpected popup windows at runtime for unattended test execution.](#)
7. [Handling Application location changes - server host and port changes during play.](#)

Handling Applications with varying Response Speeds

An application's response speed is dependent on the system / network in which it runs. The expected window might appear immediately on a faster system / network or might appear after a significant delay when running on slower systems / network.

QEngine has an option to wait for the configured timeout period(in seconds) for the window to appear. By default, QEngine will wait for 60 seconds for the particular window to appear. If the window appears before the configured timeout period, it will immediately proceed to the next script actions. If the needed window does not appear within the configured timeout period, the particular setWindow action will be reported as "Failed" and the script will proceed to execute the remaining part of the script. The timeout period can be modified as required, as explained below in "**Adjusting Playback Speed**".

Adjusting Playback Speed

To synchronize QEngine playback with the speed with which the application window appears, configure the settings as given below:

- Click on the "**Settings**" link available in the Suite tree in the Web Functional Home Page UI.
- In the Settings page, increase or decrease the value for "**Maximum wait time for document ready**" variable.

This will affect the overall playback speed of QEngine as it applies for all the setWindow() calls in the script.

- The timeout can also be configured during runtime using the script function

setDownloadTimeLimit(seconds) in the script.

Handling Application Window Title / Size / Content Changes

Application window title and size are other aspects of application that might undergo change from one version of application to another.

Application window size changes will not affect QEngine playback as it does not depend on screen coordinates for locating Window elements. Similarly, inserting additional elements or removing elements into or from the application after recording will also not affect playback. QEngine will identify the elements on the web page based on the element properties such as name / id / innertext, etc.,

However, if the window title changes, then the change has to be incorporated in the QEngine script. QEngine has an effective way to handle the changes with the support of regular expressions for a successful playback.

You can configure the Regular Expression for Window Title as explained in the “**Using Regular Expressions**” topic below:

Using Regular Expressions for Window Title

Assume that the application window title is changing dynamically such that only the starting or ending word of the title remains the same across pages. To playback the scripts that includes such pages, you can specify regular expressions for the window or parent title in the GUI Map file. The steps to achieve it are as follows:

1. From the tree view of the Web Functional Home Page UI, select a script and right click it.
2. A popup menu will be displayed. Choose **Edit Script Map** menu item from the popup or choose the **Edit GUI Map File** option in the script editor tool bar. This will open **GUI Map Editor** screen.
3. The **GUI Map File Editor** will show all the window and element object properties of a recorded web page in a tree-view. In the left pane, you can view the parent node being the window node and the child nodes being the objects belonging to that window node.
4. Click on the required window node from the left pane. You can view the property name and values displayed in a table in the right-pane.
5. In the right pane, from the **Condition** column for the required window title or parent title, choose the specific condition such as **equals, starts with or ends with** to use regular expressions. In the Value column, enter the starting or ending word of the title that will not change. Update and Save the configured values.

6. **Environment variables** also can be used to configure values. The environment variables should be prefixed with the “\$” symbol and should be given in the value field. During test execution after retrieving the window properties for playback, the variables will be substituted with appropriate value of the variable. The environment variable can be configured in script runtime with the **setEnvironmentVariable(“variable”, “value”)** function call inside the script.
7. This will playback the script searching for the specified starting or ending word in the window or parent title. This eliminates the need to re-record the script.

Handling Application Window Element Changes

A good automated testing solution should ensure that tests can run unattended regardless of application behavior during test execution.

During a play session, the elements in the web page will be identified based on the element properties stored in the GUI Map file. The GUI Map file will contain the properties that were got during the recording session.

Following are the element properties that governs the element identification during playback:

1. ID – id of the element present in the web pages
2. name
3. innertext
4. src – source of the image element
5. firstcellvalue – value present in the table cell at 0,0 position

In the above properties, “**ID**” will take precedence, followed by other properties appropriate to the element type. If any of the main identification property value used to identify the element changes after the recording, the test may fail during playback session.

Following are the various categories of property changes that need to be handled by the automation software.

1. [Element Property Changes after Recording](#)
2. [Element Location Changes](#)
3. [Element Type Changes](#)
4. [Element Property Changes Dynamically](#)

Handling of the above property changes are discussed further here.

Category 1 : Element Property Changes after Recording

When the property values of the objects in your application has changed to another value after recording, QEngine allows you to modify the corresponding test object property values in the GUI map (or Object repository), so that you can continue to use your existing test scripts.

For Example : The **ID** attribute of an element changes as required in the web page after the automation script is recorded. This should be changed appropriately in the GUI Map file in order to ensure successful execution of script.

You can configure the property changes of the element in the map file as explained in the “**Configuring Element Property Changes in Map File**” topic below:

Configuring Element Property Changes in Map File

To configure the element property changes in the Map file, follow the steps given below:

1. From the tree view of the Web Functional Home Page UI, select a script and right click it.
2. A popup menu is displayed. Choose **Edit Script Map** menu item from the popup or choose the **Edit GUI Map File** option in the script editor toolbar. This will open the **GUI Map Editor** screen.
3. The **GUI Map File Editor** will show all the window and element object properties of a recorded web page in a tree-view. In the left pane, you can view the parent node being the window node and the child nodes being the objects belonging to that window node.
4. Click the required element node from the left pane. You can view the property name and values displayed in a table in the right pane.
5. Edit the property values for the selected node and click the “**Update**” button. Similarly, edit all the properties you wish to modify and save by clicking on the “**Save**” button.
6. **Environment variables** also can be used to configure values. The environment variables should be prefixed with the “**\$**” symbol and should be given in the value field. During test execution after retrieving the element properties for playback, the variables will be substituted with the appropriate value of the variable. The environment variable can be configured dynamically with the **setEnvironmentVariable(“variable”, “value”)** function inside the script.
7. This will successfully play back the script. During playback, QEngine reads the changed object properties(window and element properties) in the GUI Map file and then looks for an HTML object with the same properties in the web application being tested.

Category 2: Element Location Changes

The location of Window Elements usually will not change, but could change between one release of your application and another, basically to improve the aesthetics of the screen. QEngine playback does not require any configuration as QEngine identifies the element on the web page based on the properties of the element, which is not dependent on the order of the elements on the screen.

Category 3: Element Type Changes

Another possibility of a web application change is the element type itself changing from one to another such as link to a button / button to image, etc. after recording. In this case, the element properties are required to be added to the GUI Map file using learn mode and the script action for this element should be inserted into the script from the learn mode view.

Configure the element properties using learn mode, as explained in **Configuring Element Type Changes using Learn Mode** in topic given below.

Configuring Element Type Changes using Learn Mode

Let us consider an image in the application is replaced with a button in the web application that got recorded. The above change should be incorporated in the recorded script. Otherwise, the script will fail stating "ELEMENT_NOT_FOUND" for the particular image element, which does not exist now. To reflect the change of element in the script, we have to learn the particular button element properties and should use the appropriate action over the element for the successful execution of the script.

To learn the properties of the particular element to the GUI Map file and to insert the relevant function or user action for the element type using learn mode, follow the steps given below:

1. From the tree view of the Web Functional Home Page UI, select the script and right click it.
2. From the QEngine Toolbar, click "**Launch Browser**" option. In the launched browser, type the application URL and browse to the web page where the change has happened.
3. In the QEngine window, right click over the script and choose "**Orchestrate**" menu item from the popup menu.
4. Orchestration page will open, listing all the elements stored in the GUI Element Repository in a tree.
5. Click on the "**Start Record**" button from the toolbar to start the recording.
6. Click the "**Add**" button below the element tree.

7. Click on the appropriate element from the web page whose element properties needed to be learned. This will learn the selected element object properties and store the same in the GUI map file.
8. Click “**Stop Record**“ button from the toolbar to stop the recording.
9. Now, choose the newly learned element id from the tree.
10. This will show the element properties in the right side pane.
11. From the “**Script Text**“ text box, copy the script line to use inside the script for the particular element learned. Replace the old element action with the above-said script text in the script.
12. Save the script and click on the “**Home**“ tab to go back to the Web Functional Home Page UI.

You can now successfully playback the script with the changes in the element incorporated into the script.

Category 4 : Element Property Changes Dynamically

The element properties such as **ID, name or innertext** of a certain element might change dynamically to meet the goal of the application. In such a scenario, the recorded element properties will not be helpful in identifying the element in the web page. If the changed property value is known, we can override the element property value stored in the GUI Map file with the above value during runtime using overriding functions.

Let us consider an example where auto generated ids are used in the elements in the page as ID or innertext. Each time it is played, the element identification property needs to be overwritten. QEngine provides property-overriding functions to override the particular property of the element.

Configure the overriding property as explained in the “**Configuring Overriding Property**“ topic below:

Configuring Overriding Property

Assume an online ticket booking system where a user fills in a form to book a ticket. Once a ticket is booked, assume a message is displayed in the HTML page as “**New Ticket ID: T1001 has been successfully generated**” and a dynamic link is created to view the booked ticket details. Now, to click on this dynamic link whose id and innertext properties include the Ticket ID, you need to fetch the TicketID from the message that is displayed using the webGetText() and pass the fetched TicketID value as the fourth argument in the clickLink() as shown below:

Example script which uses the webGetText() and clickLink() with four arguments for a link whose innertext and id properties are dynamically generated during playback:

```
useLocalMapFile()
launchApplication("http://localhost:4444/booktickets/jsp/ticketbooking.jsp")

ticketIDVal =webGetText("Ticket ID: ", "has")
displayMessage(ticketIDVal)
clickLink("BookedTicketDetails",1,"innertext",ticketIDVal) # here innertext or linktext property is
overridden with the value obtained from webGetText function.
```

Here, the webGetText() will search and fetch the value(TicketID) placed in between the given prefix and suffix text and stores it in the ticketIDVal variable. This variable is then used as the fourth argument for the clickLink() to click on the dynamically generated link.

Similarly, there are a variety of other four argument functions such as clickButton, clickImage, selectItem, clickElement, setText, clickList, etc., available to handle specific HTML elements such as Button, Image, Select, Table, Textfield, Div, etc. whose property value changes dynamically.

For details, refer to the help documentation in <http://www.manageengine.com/products/qengine/help.html>

If multiple identification properties of the same element change in the web page, we can use the substitution of **Environment Variable** for the element properties in the GUI Map File.

Configure the **Environment Variable** for the property value in the GUI Map file as explained in the **Configuring GUI Map Element Properties with Environment Variable** topic below.

Configuring GUI Map Element Properties with Environment Variable

Let us assume the identification properties such as **id** and **name** of an element in a web page change dynamically for each session. In such a scenario, the element identification will fail with the recorded properties during playback session as the property value changes dynamically. For successful playback, both **id** and **name** properties of the element should be substituted properly for the playback session. With the overriding element property value discussed above, we can override a single property of an element. We cannot override multiple properties of the same element.

To dynamically substitute multiple properties for the same element, we have to use the "**Environment Variables**" for substitution.

Follow the below steps to configure “**Environment Variables**” for substituting element properties:

1. From the tree view of the Web Functional Home Page UI, select the script and right click it.
2. A popup menu is displayed. Choose **Edit Script Map** menu item from the popup or choose the **Edit GUI Map File** option in the script editor toolbar. This will open the **GUI Map Editor** screen.
3. The **GUI Map File Editor** will show all the window and element object properties of a recorded web page in a tree-view. In the left pane, you can view the parent node being the window node and the child nodes being the objects belonging to that window node.
4. Select the element from the left tree that will show the element properties and values in the right pane.
5. Now configure values for **id** and **name** properties of the element as **\$eid** and **\$ename** respectively. The “\$” is the indication for using Environment Variable.
6. Save the changes and close the GUI Map editor.
7. In the script, just before invoking the action over the particular element, set the value for the above-said Environment Variables as shown below:

```
setEnvironmentVariable("eid", "<Value_for_id>")  
setEnvironmentVariable("ename", "<Value_for_name>")
```

8. During playback session, the element properties taken from GUI Map file will be substituted with the Environment Variable value.

Similarly, we can configure the substitution for dynamically changing GUI Map element property values.

Handling Applications with Dynamic Windows & Element

Applications can sometimes be designed to have varying window elements that cannot be predicted at the time of playback.

Following are the scenarios to be handled,

1. Some new elements can be added to the application dynamically.
2. Some new window may be launched instead of the one launched in recording.

QEngine has built-in functions to fire event on such dynamic elements & windows. You need not record all the element actions and the window that comes after the click. You can handle all the possibilities with QEngine's built-in functions itself.

Utilize the built-in functions of QEngine as discussed in **Using Dynamic Functions** of QEngine topic given below.

Using Dynamic Functions of QEngine

Handling dynamically added elements

Let us assume a scenario where a new text field is added to the web page based on the user input. To input to the above text field that got added dynamically, we can use the `fireEventOnElement` function available with QEngine.

`fireEventOnElement()` - To fire the specified event over the given HTML element.

Syntax: `fireEventOnElement("tagName", "propertyName", "propertyValue", occurrence of the element, "actionName", "actionValue", "useRegExp")`

Example : `fireEventOnElement("input", "type", "text", 1, "setText", "dynamic value", "false")`

The above function will set text over the dynamically added text field in the web page. Similarly, we can fire event over any HTML Element. QEngine supports following actions over the HTML Element via `fireEventOnElement` function,

1. `setText`
2. `click`
3. `selectItem`
4. `doubleClick`
5. `keyPress`

Handling dynamically launched window

To dynamically identify the window that appears based on dynamic element click, use the following functions:

`setDynamicWindow` - To set browser window that got launched during playback which is not part of the recording. These windows may not have any GUI Map File entries.

Syntax: `setDynamicWindow(parent_window_title, parent_window_index, frame_window_title or frame_window_name, value_of_frame_title or value_of_name, frame_window_index)`

Example: `setDynamicWindow("My_parent_window_title", 1, "frame_window_title", "My_frame_window_title", 1)`

This will dynamically identify the window with parent title as "My_parent_window_title" and frame window title as "My_frame_window_title" so that further actions on any element in this window can be achieved using `fireEventOnElement`.

You can refer the dynamic functions reference in the following link:

http://www.manageengine.com/products/qengine/help/built_in_functions/dynamic_functions.html

Handling UnExpected Application Popup during playback

One of the hardest things when developing automation scripts is handling the case where an unexpected popup, like an ALERT box, pops up. Let us take the case of an application that generates an error message if the user input data exceeds the allowed limit, and does not generate an error message if your input data is valid during the next test run.

The Exception Handling mechanism of QEngine allows you to handle unexpected popup that arises during playback.

Configure Exception handling as explained in the **Handling Exceptions in Playback** topic below:

Handling Exceptions in Playback

To automatically handle unexpected popups during playback, follow the steps given below:

1. Choose the **Settings** link from the Suites tree in the Web Functional Home Page UI.
2. In the Settings page, select **Exception Handling** Tab
3. To handle unexpected popup during playback, choose any or all of the check boxes under **Popup Exceptions** category in **Exception Handling** such as
 1. **Report & Continue Play** – This will report the title of the unexpected popup in testout0.txt
 2. **Stop Play** – This stop the playback
 3. **Capture Screen & Continue Play** – This will save the screenshot of the popup and continue to play.
 4. **Close & Continue Play** – This will close the popup and continue to play.
 5. **Call Script** - You can call any other script to handle the scenario.
4. Click the Apply button to save the Settings.

Handling Application Server Host and Port Changes

Another aspect of the application that will change after recording or anytime during playback is the host in which the application runs and the port it uses. QEngine allows configuring of modified host and port without changing the scripts or GUI map file. This also allows the tester to run the automation project against any number of servers by just a single point and click.

Configure Host Port changes as explained in **Handling Host Port Changes during Playback** topic below.

Handling Host Port Changes during Playback

To configure the server and port changes, follow the steps given below:

1. Click the **Settings** link from the suite tree in the Web Functional Home Page UI.
2. From the Settings page, select **Host-Port Settings** tab, select the Use Host-port Configuration check box, and then click the Host-Port Editor link. This will display the Host-Port Configuration UI.
3. In the Host-Port Configuration UI, click the New button placed next to the File Name combo. This will display a dialog that prompts to specify a property file name. Enter a file name in which the configured host and port details has to be stored and click the OK button. This will populate the newly created property file name in the combo.
4. Now the UI will show the host, port and protocol details parsed from the script present in the particular suite.
5. Configure the changed host, port and protocol details appropriately for all the listed details.
6. Now select the check box for “**Activate Selected Property File**” to use this configuration during playback session.
7. Click the Apply button to save the changes.

Conclusion

This concludes the paper on handling dynamic changes of application in your automation project with QEngine. We hope that you now have a better understanding of the features and capabilities of ManageEngine QEngine.

Visit our website www.qengine.com for more information.

For any clarification on this article mail srinivasar@zohocorp.com

ManageEngine Applications Manager (www.appmanager.com) provides out of the box performance monitoring for J2EE Application Servers, Databases and Servers.

ManageEngine is the leader in low-cost enterprise IT management software. The ManageEngine suite offers enterprise IT management solutions including Network Management, HelpDesk & ITIL, Bandwidth Monitoring, Application Management, Desktop Management, Security Management, Password Management, Active Directory reporting, and a Managed Services platform. ManageEngine products are easy to install, setup and use and offer extensive support, consultation, and training. More than 30,000 organizations from different verticals, industries, and sizes use ManageEngine to take care of their IT management needs cost effectively. ManageEngine is a division of ZOHOO Corporation. For more information, please visit www.manageengine.com.