

QEngine Best Practices(Web Functional Testing)

This document describes the best practices to be followed while automating using QEngine

1. QEngine Best Practices(Web Functional Testing)	1
2. Best Practices For Functional Testing	2
• Installation and Setup	2
• Hardware Requirements: (Web Functional & Web Services Functional)	0
• Mail Server Settings	0
• Proxy/Firewall Settings:	0
3. Organizing Test Scripts and Testcases	3
4. Scripting Best Practices	3
5. Recording Usage	7
• Default Recording	7
• Analog Mode Recording	8
• Expert Mode Recording	8
6. Script Maintanace	9
7. Real Time Use-case	9
8. General Best Practices	10
9. Report Customization	10
10. Regression Testing	10
11. User Guidelines	0
12. Support,Forums & Blogs	11

Best Practices For Functional Testing

Installation and Setup

Hardware Requirements: (Web Functional & Web Services Functional)

Below table list the minimum hardware requirements for QEngine Web functional and Web Services Functional Testing

Operating Platform	Processor Type	Processor Speed	Memory	Hard Drive Space (For Installation)	Hard Drive Free Disk Space After Installation (To Run)
Windows	Pentium III or Above	500 MHz or Above	256 MB Ram or higher	120 MB	1 GB
Linux	Pentium III or Above	500 MHz or Above	256 MB Ram or higher	120 MB	1 GB

Mail Server Settings

Configure Mail Server settings under "Admin" tab at first time you connect as it will be useful for sending the reports to you mail id.

To Configure Mail Server Details,

- Click on "Admin" Tab or link available in the page and choose "Mail Configurations" Tab
- Here you can configure Mail Server Name and Mail Server Port.
- Also you can optionally configure Authentication details for your mail server.

Proxy/Firewall Settings:

Configure Proxy Settings based on your Network Environment settings.

To Configure Proxy Settings,

- Click on "Admin" Tab or link available in the page and choose "Proxy Settings" Tab.
- Here you can configure "Proxy / Firewall" information of your network for HTTP and HTTPS protocol.
- You can also configure "Non Proxy Host" which need not to be accessed via proxy.

Organizing Test Scripts and Testcases

1. The test scripts needs to be created under Suite. Suite is a container which contains scripts and testcases created.
2. Scripts for testing various modules in the application can be grouped under a module name for easy maintenance.

For Example: Say your application has modules such as Sales, Purchase, Leads etc.,

1. The test scripts for Sales module can be created in the suite as follows, sales/script1
(While creating new script specify the scriptname in the text field as sales/script1)
2. Similarly, for purchase and leads, modules can also be created.

Scripting Best Practices

1. Do not use (') single quotes inside commented lines in the script editor.
2. Before starting playback always compile to see any script error.
3. Do not edit the QEngine generated ID, instead edit through map file

Example: setText("name","QEngine",2)

- Here "name" is the QEngine generated ID, it should not be edited.
- If you edit the ID then it will result in playback failure stating "Element not found under current window".

- In case if you want to change the QEngine generated ID for better understanding, edit the ID in the GUI map file of a script,

Do the same following the below steps,

- Right click the script name and select "Edit Script Map".
- In the screen opened, select the element whose ID needs to be changed
- Choose the "Edit" icon available next to the "Element ID" field and edit the ID of the element.
- Always remember to make the appropriate changes in the script with respect to the changed ID after editing the GUI map file.

4. Identify frequent / regular operation in the script such as login / logout etc. and record it in a separate script and call it in all the scripts using callscript. This will be helpful for you to control the script maintainability.
5. While using callscript make sure that you mention the script path and script name correctly.

Example:

- To call the script "login" which is not formed under any module callScript("login","login")
- To call the script which is present inside any of the modules such as "Home/browse", then call script should be used as below callScript("browse","Home/browse")

6. Scripts grouped under module can be played back specifically to get the report for particular module.
7. To run a script against a different host than the recorded one, use "Host Port Configuration editor" to change the Host and Port details. You should not edit the scripts manually to change the host against which you want to play.
8. Do not copy the content of one script into another script. This may fail as the map details are not copied. QEngine requires Map file for its execution.
9. Don't remove the useLocalMapfile() from the script unless the map file is merged to global, this will affect the script playback
10. Design scripts as self sufficient and carryout clean up actions at the end of the script and bring the web application state to fresh. If this is not done this may affect subsequent script playback.
Example Scenario : If you are logged in to your web application initially means you should logout of your web application when the script finishes. Otherwise if you added "login" in your next script, that will fail as it is already logged in.
11. Always use Insert Built in Functions to insert a function in to the script.

12. Dataset for the scripts should be kept individually as modifying to one script should not affect another script which also shares the same dataset.
13. On using "database" for Dataset, make sure needed jar files are added to the classpath. Otherwise the database connection cannot be established.

Following are the Jar files which are needed for various databases,

- **MYSQL** - There is no need to keep any jar files for this as this is supported by default
- **ORACLE** - classes12.zip
- **MSSQL 2000** - msbase.jar, mssqlserver.jar, msutil.jar
- **MSSQL 2005** - sqljdbc.jar

The above jar files have to be placed under QEngine/jars folder. The jar files need to be added to the classpath,

- In the "Database Configuration Screen" choose appropriate database in the list and click on the "Set Classpath" link available next to Driver name text field.
- In the Classpath screen, configure the classpath as ".\jars\classes12.zip" Similarly add the needed classpath and click "OK" to save the details.

Following are the Drivers for various databases,

- **MYSQL** - org.gjt.mm.mysql.Driver
- **ORACLE** - oracle.jdbc.driver.OracleDriver
- **MSSQL 2000** - com.microsoft.jdbc.sqlserver.SQLServerDriver
- **MSSQL 2005** - com.microsoft.sqlserver.

The database URL for various databases is given below,

- **MYSQL** - jdbc:mysql://<HOST_NAME>:<PORT>/<DATABASE_NAME>
- **ORACLE** - jdbc:oracle:thin:@<HOST_NAME>:<PORT>:<DATABASE_NAME>
- **MSSQL 2000** -
jdbc:microsoft:sqlserver://<HOST_NAME>:<PORT>;DatabaseName=<DATABASE_NAME>
- **MSSQL 2005** -
jdbc:sqlserver://<HOST_NAME>:<PORT>;DatabaseName=<DATABASE_NAME>

14. You can use QEngine as API by recording the actions in to group of functions and in the main script, you can just call the function to perform the operation

Example: Following is the scenario what the script is going to perform (For this example assume the application is an intranet people search system)

- login (Login to the application)

- Search for an employee name
- View Employee details
- Logoff

To use QEngine as an API , record the above actions in a separate script. i.e., login action in script named "login". Similarly all other actions should be recorded in individual scripts.

- Merge the map file of all the scripts in to global map file.
- Now under QEngine/jars create a new file named custom.py
- Edit custom.py and insert following line

```
from Framework import *
```

And then below that line define a function as below

```
def login(username,password):
```

Above is the syntax to define a function in python language.

Then Copy the contents of the login script created inside the function as below,

(After function definition, the script line below it should be single indented)

```
def login(username,password):
    launchApplication("<application_URL>")
    setWindow("Intranet Application",2)
    setText("username",username,1)
    pw=getEncryptedValue(password)
    setText("password",pw,2)
    clickButton("login",2)
```

Similarly all the other operations can be brought inside the custom.py.

The main script can call the above defined function to complete the operation.

The main script is as below

```
from custom import *
login("qengine","qengine")
searchEmployee("victor")
```

```
viewEmpDetails()  
logout()
```

The main script just calls the functions created in the custom.py to accomplish the task.

Recording Best Practices

- QEngine needs to be opened in the first tab in the browser, otherwise QEngine toolbar buttons will not be enabled. You may not be able to record / playback the test scripts.
- Disable Javascript Error alert during recording / playback.

To disable Java Script Error notification follow the steps below:

- Open Tools->Internet Options in the IE Browser.
- Go to "Advanced" tab in the Internet options window
- Under "Browser" category "Un check" the option "Display a notification about every script error"
- Click "OK" to save the configuration and close the Internet Options window.
- Close all the other Internet Explorer browser opened before start the recording
- While recording the action of typing a text in the "Text Field" always type a value, even though there is some default value present. As if you do not record the setText here, playing back in different system which doesn't populate the default value may lead to failure of the script.

Recording Usage

• Default Recording

By default QEngine will do object based recording. (i.e) If any other element in the page changes or position of an element in the page changes that won't affect the playback as play will find the element in the page based on the element properties and not based on the page co-ordinates.

- QEngine identify elements in the page by id/name/innertext of the element. if any of this attribute is dynamic then try to use dynamic functions for better results.

- It is better to assign "id" to all the elements in your web application which QEngine uses to identify the element in the web page.
- The script created can be played back without any re-record , even though the web page is changed after recording.
- **Checkpoint** -> With checkpoint you can verify the property of an element in the web page matches the expected value.
- **Testcase** -> With testcase you can verify the properties of an element in the web page meets the expected value. Also you can insert multiple checkpoint in a single testcase.
- **Script Editor** -> Script Editor shows the script for creation and editing.
- **Data Configuration** -> Data Configuration is to create data driven testcase. You can get data for the playback from CSV,databases etc.,

• Analog Mode Recording

Flash objects can be recorded using Analog mode options.

Here the recording will be based on the co-ordinates in the web page.

During playback QEngine will perform action based on the X, Y co-ordinate got recorded.

- If you see any of the object is not recognized in Default mode recording, then switch to Analog mode and record the actions.
- You cannot create checkpoint in Analog mode.

• Expert Mode Recording

In Expert mode recording, the map files are not maintained as a result, you can copy the scripts from one script to another. In case of any property is changed in the application it can be easily edited directly from the recorded script

- You can enable the following options by going to settings--->Recorder settings--->Enable expert mode
(or)
- Click new script and in the new script dialog box, enable the expert mode option, then Click on ok to start the recording in expert mode
(or)

- Click on the down arrow which is present next to start record button inside the UI and select "Record in Expert Mode"

Script Maintenance

For Maintaining the scripts in the repository, you have to check-in following files to the repository,

(All the below files will present under QEngine\projects folder)

- <suite_name>.proj file
- <suite_name> folder
- <suite_name>/conf/* (all the files present in this folder)
- <suite_name>/webscripts/* (all the files present in this folder)
- <suite_name>/dataset/* (all the files present in this folder)

The directory structure should be maintained while overwriting the files.

- If QEngine is used as an API as said in the scripting topic, then the custom.py file can also be checked in.
- You can also Export a suite and check-in the single file. When you export a suite that will form <suite_name>.qed file under QEngine folder. This is a zip file which can be checked in.
- This can be imported back in to QEngine by using "Import Suite" option.

Real Time Use-case

- While recording file chooser window, browse to the root folders (Such as C:) through combo on the top and then browse to needed location. Also make sure the file you are selecting is not to be scrolled and viewed in the file chooser.
- While inserting checkpoints to check a value of a table cell, go to insert checkpoint and select the function "checkCellValueAt" and then proceed to select the cell in the webpage.
- To retrieve formatted date string use \$date(dd/mm/yyyy). This will return 04/01/2008.

General Best Practices

1. Use Wait For Element enabled in settings for rich AJAX applications.
2. In the Automation suite development stage frequently take backup for the test suite created using "Export Suite" option, which you can import to restore the whole suite by using "Import Suite" option.
 - To Export Suite, select the "Suite" from the left tree.
 - Now click on the "Export Suite" option available in the right side of the screen.
 - This will Export the suite to "<SuiteName>.qed and save it in the given location.
 - The exported suite can be imported again for use by using "Import Suite" option

Report Customization

QEngine report format is customizable. You can customize the reports by writing a java class.

- Help on report customization is available at <http://qengine.wiki.zoho.com/Customizing-Web-Reports.html>
- The Javadocs for the customization API available at <http://www.manageengine.com/products/qengine/javadocs/index.html>
- Bug Report mail is customizable in QEngine. You can customize the reports by writing a java class.

Regression Testing

- In a given time in a client only one schedule can run.
- Always close the browser window after scheduling the test including the client window.

Use Guidelines

You can get the user guidelines from this link:

<http://qengine.wiki.zoho.com/-Installation-Guide.html>

- **Web Functional testing Datasheet**
<http://www.manageengine.com/products/qengine/datasheet-web-functional.html>
- **Help Documentation**
<http://qengine.wiki.zoho.com/Web-Functional-Testing.html>

Support, Forums & Blogs

Forums Link: <http://forums.manageengine.com/test-automation/qengine>

FAQ link : <http://www.manageengine.com/products/qengine/faq.html>

Blogs: <http://blogs.qengine.com>