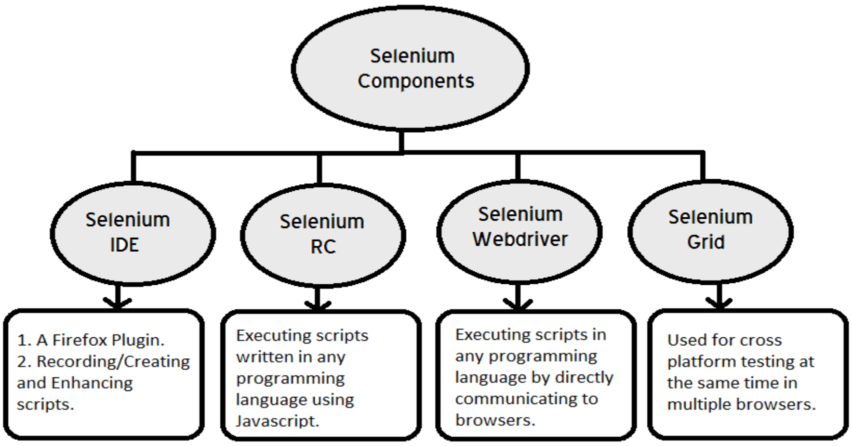
1. Study of any Web testing tool (e.g. Selenium)

SELENIUM

Introduction

- In 2004 invented by Jason R. Huggins and team.
- Selenium is open source software, released under the Apache 2.0 license and can be downloaded and used without charge.
- Selenium is a portable software testing framework for web applications.
- Selenium can be deployed on Windows, Linux, and Macintosh.
- Selenium consists of the following components :
    - Selenium IDE
    - Selenium RC
    - Selenium Webdriver
    - Selenium Grid



1. A Firefox Plugin.
2. Recording/Creating and Enhancing scripts.

Executing scripts written in any programming language using Javascript.

Executing scripts in any programming language by directly communicating to browsers.

Used for cross platform testing at the same time in multiple browsers.

Figure1: Selenium Components

Selenium IDE:

- Selenium IDE is a complete Integrated Development Environment (IDE) for creating Selenium tests.
- Selenium IDE previously known as Selenium Recorder.
- It is implemented as a Firefox extension, and allows us to record, edit, and replay the test in Firefox .

Features:

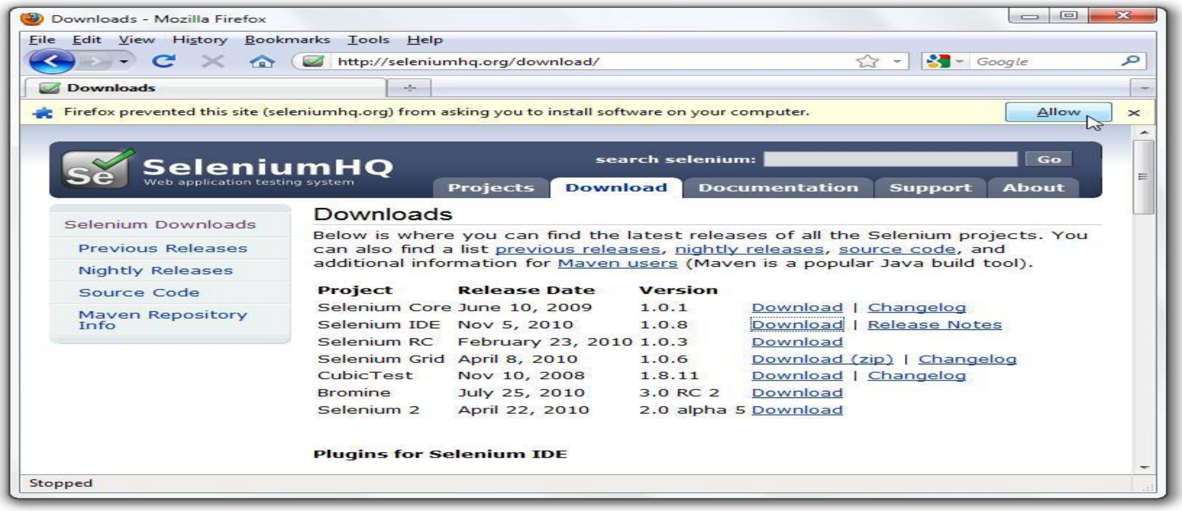
- It is simple and easy record and playback.
- Selenium IDE supports intellectual field selection options like ID's,XPath and Names.
- It allows setting breakpoints and debugging the scripts.

Limitations:

- Selenium IDE works only in Mozilla Firefox and it cannot be used with other browsers.

Installing the Selenium IDE:

- Using Firefox, first download the IDE from the www.seleniumhq.org website.
- Firefox will protect us from installing add-ons from unfamiliar locations, so we will need to click 'Allow' to proceed with the installation, as shown in the following screenshot.



Firefox prevented this site (seleniumhq.org) from asking you to install software on your computer.

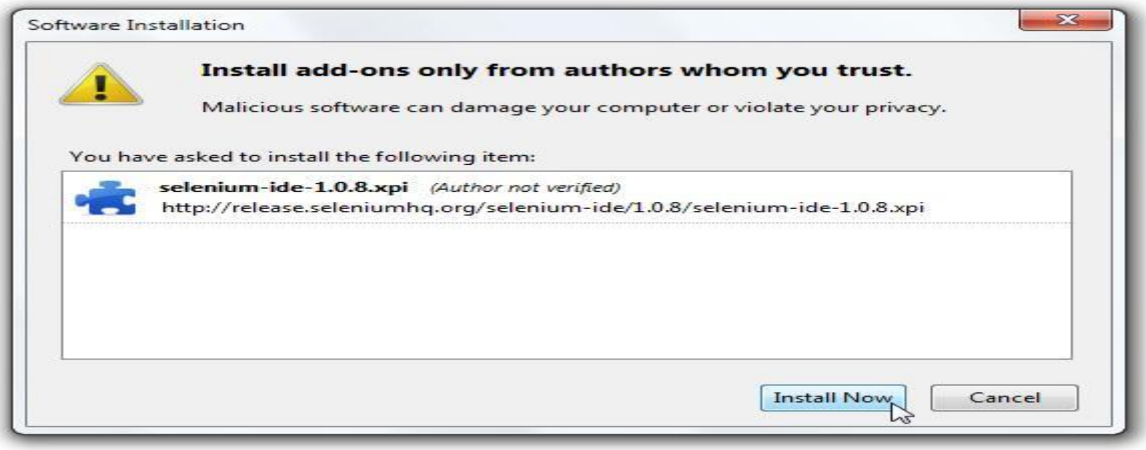
Downloads

Below is where you can find the latest releases of all the Selenium projects. You can also find a list previous releases, nightly releases, source code, and additional information for Maven users (Maven is a popular Java build tool).

Project	Release Date	Version	
Selenium Core	June 10, 2009	1.0.1	Download | Changelog
Selenium IDE	Nov 5, 2010	1.0.8	Download | Release Notes
Selenium RC	February 23, 2010	1.0.3	Download
Selenium Grid	April 8, 2010	1.0.6	Download (zip) | Changelog
CubicTest	Nov 10, 2008	1.8.11	Download | Changelog
Bromine	July 25, 2010	3.0 RC 2	Download
Selenium 2	April 22, 2010	2.0 alpha 5	Download

Plugins for Selenium IDE

- When downloading from Firefox, we'll be presented with the following window.



Software Installation

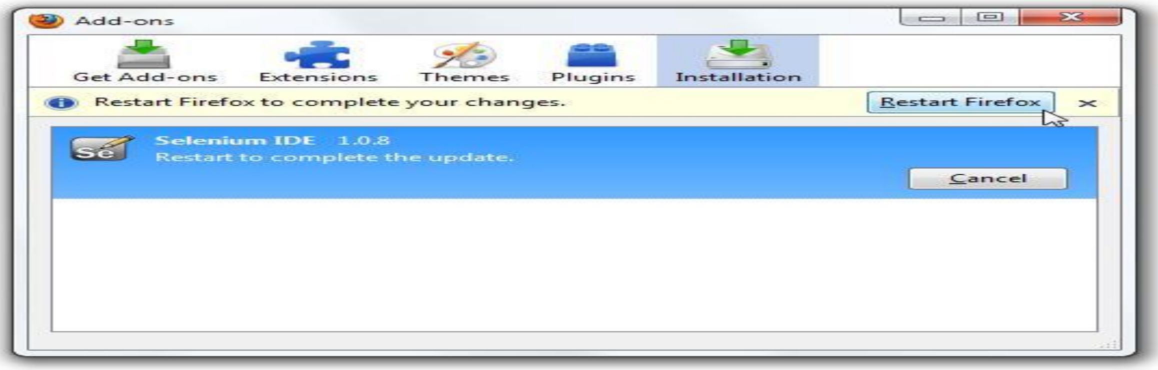
Install add-ons only from authors whom you trust.

Malicious software can damage your computer or violate your privacy.

You have asked to install the following item:

selenium-ide-1.0.8.xpi *(Author not verified)*
http://release.seleniumhq.org/selenium-ide/1.0.8/selenium-ide-1.0.8.xpi

- Select Install Now. The Firefox Add-ons window pops up, first showing a progress bar, and when the download is complete, displays the following.

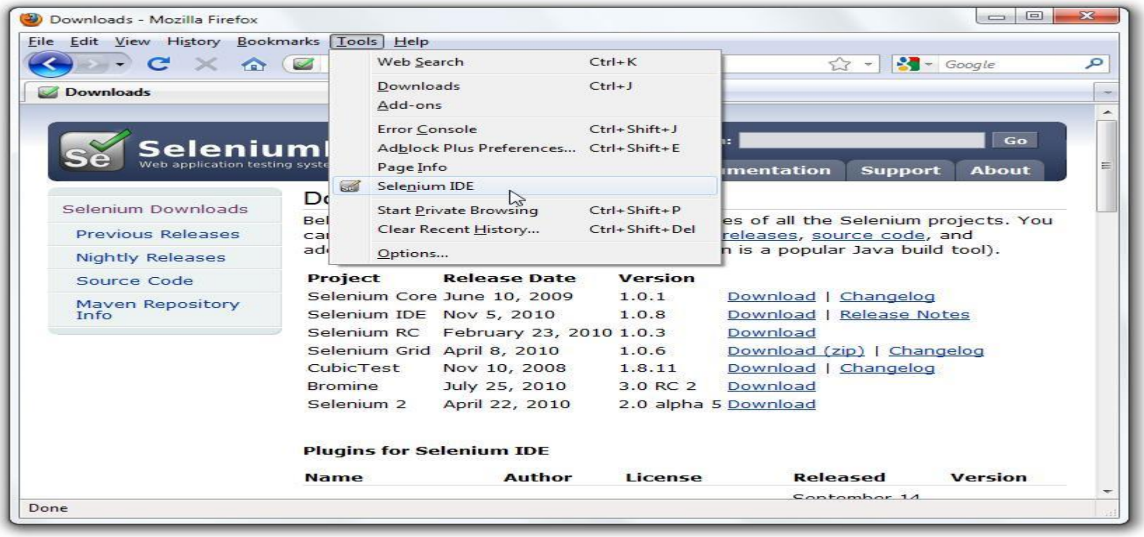


Add-ons

Restart Firefox to complete your changes.

Selenium IDE 1.0.8
Restart to complete the update.

- Restart Firefox. After Firefox reboots we will find the Selenium-IDE listed under the Firefox Tools menu.



Downloads - Mozilla Firefox

Project	Release Date	Version	
Selenium Core	June 10, 2009	1.0.1	Download | Changelog
Selenium IDE	Nov 5, 2010	1.0.8	Download | Release Notes
Selenium RC	February 23, 2010	1.0.3	Download
Selenium Grid	April 8, 2010	1.0.6	Download (zip) | Changelog
CubicTest	Nov 10, 2008	1.8.11	Download | Changelog
Bromine	July 25, 2010	3.0 RC 2	Download
Selenium 2	April 22, 2010	2.0 alpha 5	Download

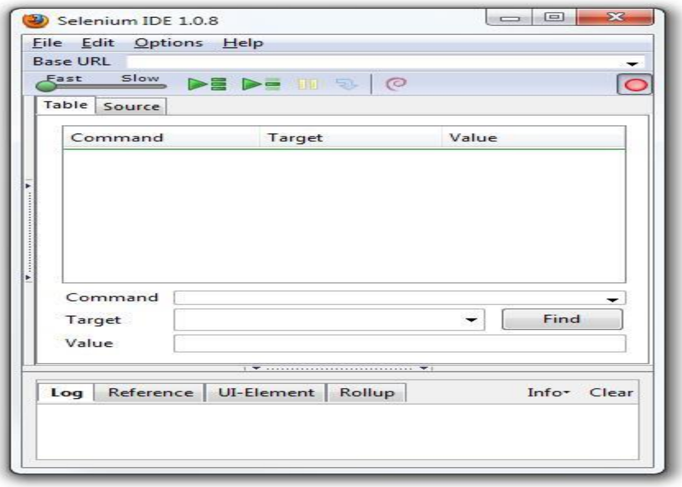
Plugins for Selenium IDE

Name	Author	License	Released	Version

File
Edit
View
History
Bookmarks
Tools
Help
Web Search Ctrl+K
Downloads Ctrl+J
Add-ons
Error Console Ctrl+Shift+J
Adblock Plus Preferences... Ctrl+Shift+E
Page Info
Selenium IDE
Start Private Browsing Ctrl+Shift+P
Clear Recent History... Ctrl+Shift+Del
Options...
Google
Downloads
Selenium
Web application testing syste
Go
mentation
Support
About
Selenium Downloads
Previous Releases
Nightly Releases
Source Code
Maven Repository Info
es of all the Selenium projects. You
releases, source code, and
n is a popular Java build tool).
Done

Opening the IDE:

- To run the Selenium-IDE, simply select it from the Firefox Tools menu. It opens as follows with an empty script-editing window and a menu for loading, or creating new test case.

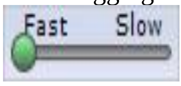


IDE Features

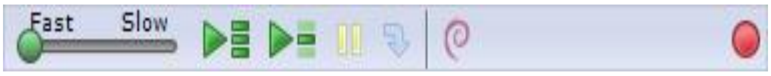
Menu Bar: The File menu has options for Test Case and Test Suite (suite of Test Cases). Using these we can add a new Test Case, open a Test Case, save a Test Case, export Test Case in a language of our choice. We can also open the recent Test Case. All these options are also available for Test Suite.

- The Edit menu allows copy, paste, delete, undo, and select all operations for editing the commands in our test case.
- The Options menu allows the changing of settings. we can set the timeout value for certain commands, add user-defined user extensions to the base set of Selenium commands, and specify the format (language) used when saving our test cases.
- The Help menu is the standard Firefox Help menu; only one item on this menu–UI-Element Documentation–pertains to Selenium-IDE

Toolbar: The toolbar contains buttons for controlling the execution of our test cases, including a step feature for debugging our test cases. The right-most button, the one with the red-dot, is the record button.



Speed Control: controls how fast our test case runs.



Run All: Runs the entire test suite when a test suite with multiple test cases is loaded.

Run: Runs the currently selected test. When only a single test is loaded this button and the Run All button have the same effect.

Pause/Resume: Allows stopping and re-starting of a running test case.

Step: Allows us to “step” through a test case by running it one command at a time. Use for debugging test cases.

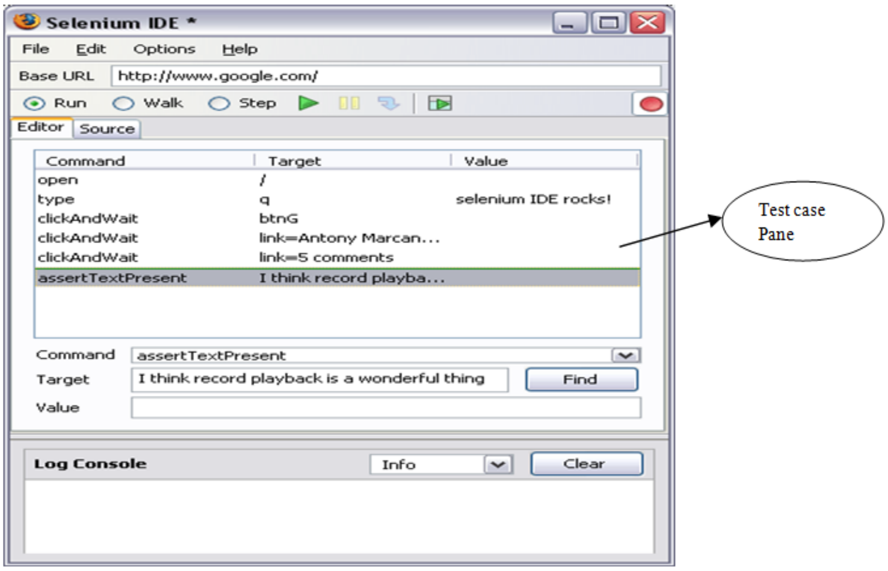
Test Runner Mode: Allows us to run the test case in a browser loaded with the Selenium-Core TestRunner. The Test Runner is not commonly used now and is likely to be deprecated. This button is for evaluating test cases for backwards compatibility with the TestRunner. Most users will probably not need this button.

Apply Rollup Rules: This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action. Detailed documentation on rollup rules can be found in the UI-Element Documentation on the Help menu.

Record: Records the user’s browser actions.

Test Case Pane

our script is displayed in the test case pane. It has two tabs, one for displaying the command and their parameters in a readable “table” format.



Base URL: http://www.google.com/

Command	Target	Value
open	/	
type	q	selenium IDE rocks!
clickAndWait	btnG	
clickAndWait	link=Antony Marcan...	
clickAndWait	link=5 comments	
assertTextPresent	I think record playba...	

Command: assertTextPresent
Target: I think record playback is a wonderful thing
Value:

Log Console: Info

- The other tab - Source displays the test case in the native format in which the file will be stored. By default, this is HTML although it can be changed to a programming language such as Java or C#, or a scripting language like Python.
- The Source view also allows one to edit the test case in its raw form, including copy, cut and paste operations.

Test Case

- Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

Test Suites

- A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch-job. When using Selenium-IDE, test suites also can be defined using a simple HTML file.

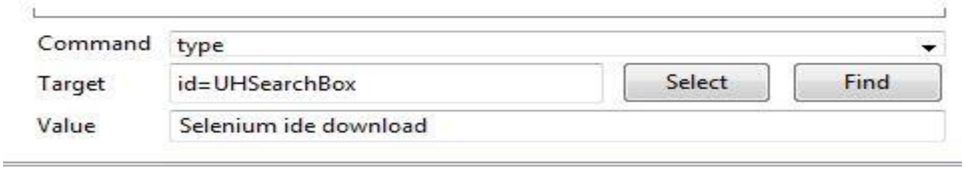
The Command, Target, and Value entry fields display the currently selected command along with its parameters. These are entry fields where we can modify the currently selected command.

Command: Command is the action that we wish to perform. This field having all commands list for selenium ide.

- If we start typing in the Command field, a drop-down list will be populated based on the first characters we type; we can then select our desired command from the drop-down.

Target: Target is used to identify web element on the webpage. There are different way to Locating Elements on the web page like Locating by Id, Locating by Name, Locating by XPath, Locating Hyperlinks by Link Text, Locating by CSS, Implicit Locators.

Value : The data or the input we would like to provide for the choosen target is known as value.



Command: type

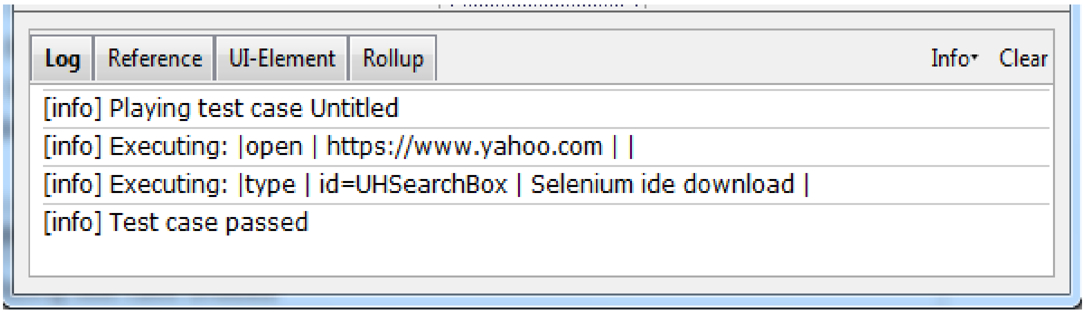
Target: id=UHSearchBox

Value: Selenium ide download

Log/Reference/UI-Element/Rollup Pane: The bottom pane is used for four different functions–Log, Reference, UI-Element, and Rollup– depending on which tab is selected.

Log

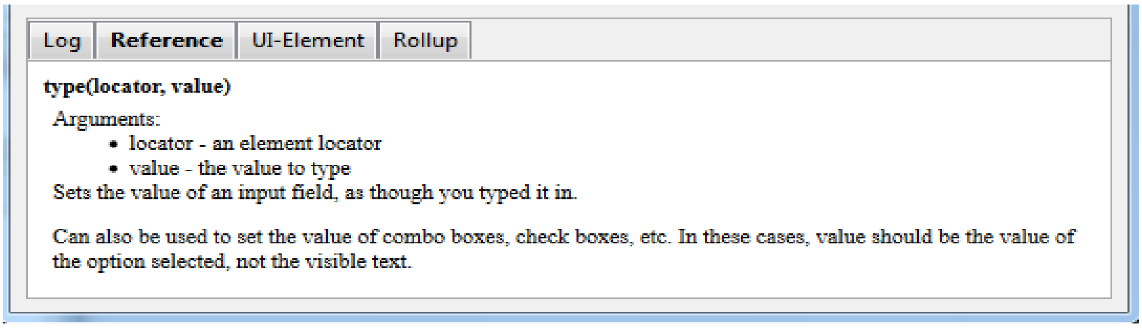
- When we run our test case, error messages and information messages showing the progress are displayed in this pane automatically, even if we do not first select the Log tab. These messages are often useful for test case debugging. Notice the Clear button for clearing the Log. Also notice the Info button is a drop-down allowing selection of different levels of information to log.



```
[info] Playing test case Untitled
[info] Executing: |open | https://www.yahoo.com | |
[info] Executing: |type | id=UHSearchBox | Selenium ide download |
[info] Test case passed
```

Reference

- The Reference tab is the default selection whenever we are entering or modifying Selenese commands and parameters in Table mode. In Table mode, the Reference pane will display documentation on the current command.
- When entering or modifying commands, whether from Table or Source mode, it is critically important to ensure that the parameters specified in the Target and Value fields match those specified in the parameter list in the Reference pane.
- The number of parameters provided must match the number specified, the order of parameters provided must match the order specified, and the type of parameters provided must match the type specified. If there is a mismatch in any of these three areas, the command will not run correctly.



type(locator, value)

Arguments:

- locator - an element locator
- value - the value to type

Sets the value of an input field, as though you typed it in.

Can also be used to set the value of combo boxes, check boxes, etc. In these cases, value should be the value of the option selected, not the visible text.

UI-Element and Rollup

Detailed information on these two panes (which cover advanced features) can be found in the UI-Element Documentation on the Help menu of Selenium-IDE.

Building Test Cases

There are three primary methods for developing test cases. Frequently, a test developer will require all three techniques.

Recording

When Selenium-IDE is first opened, the record button is ON by default. If we do not want Selenium-IDE to begin recording automatically we can turn this off by going under Options > Options... and deselecting "Start recording immediately on open."

During recording, Selenium-IDE will automatically insert commands into our test case based on our actions. Typically, this will include:

- clicking a link - click or clickAndWait commands
- entering values - type command
- The type command may require clicking on some other area of the web page for it to record.
- Following a link usually records a click command. we will often need to change this to clickAnd-Wait to ensure our test case pauses until the new page is completely loaded. Otherwise, our test case will continue running commands before the page has loaded all its UI elements. This will cause unexpected test case failures.

Adding Verifications and Asserts With the Context Menu

- Our test cases will also need to check the properties of a web-page. This requires assert and verify commands. Here we'll simply describe how to add them to our test case.
- With Selenium-IDE recording, go to the browser displaying our test application and right click anywhere on the page. We will see a context menu showing verify and/or assert commands.
- Open a web-page of our choosing and select a block of text on the page. Now right-click the selected text. The context menu should give us a verifyTextPresent command and the suggested parameter should be the text itself.
- Also, notice the Show All Available Commands menu option. This shows many, many more commands, again, along with suggested parameters, for testing our currently selected UI element.

Table View

- Select the point in our test case where we want to insert the command. To do this, in the Test Case Pane, left-click on the line where we want to insert a new command. Right-click and select Insert Command; the IDE will add a blank line just ahead of the line we selected. Now use the command editing text fields to enter our new command and its parameters.
- Simply select the line to be changed and edit it using the Command, Target, and Value fields.

Source View

- Select the point in our test case where we want to insert the command. To do this, in the Test Case Pane, left-click between the commands where we want to insert a new command, and enter the HTML tags needed to create a 3-column row containing the Command, first parameter (if one is required by the Command), and second parameter (again, if one is required). Be sure to save our test before switching back to Table view.
- Since Source view provides the equivalent of a WYSIWYG (What You See is What You Get) editor, simply modify which line we wish–command, parameter, or comment.

Insert Comment

- Comments may be added to make our test case more readable. These comments are ignored when the test case is run.
- Comments may also be used to add vertical white space (one or more blank lines) in our tests; just create empty comments. An empty command will cause an error during execution.

Opening and Saving a Test Case

- Like most programs, there are Save and Open commands under the File menu. However, Selenium distinguishes between test cases and test suites.
- To save our Selenium-IDE tests for later use we can either save the individual test cases, or save the

test suite. If the test cases of our test suite have not been saved, we'll be prompted to save them before saving the test suite.When we open an existing test case or suite, Selenium-IDE displays its Selenium commands in the Test Case Pane.

Running Test Cases

- Run a Test Case Click the Run button to run the currently displayed test case.
- Run a Test Suite Click the Run All button to run all the test cases in the currently loaded test suite.
- Stop and Start The Pause button can be used to stop the test case while it is running. The icon of this button then changes to indicate the Resume button. To continue click Resume.
- Stop in the Middle we can set a breakpoint in the test case to cause it to stop on a particular command. This is useful for debugging our test case. To set a breakpoint, select a command, right-click, and from the context menu select Toggle Breakpoint.
- Run Any Single Command Double-click any single command to run it by itself. This is useful when writing a single command. It lets us immediately test a command we are constructing, when we are not sure if it is correct. we can double-click it to see if it runs correctly.

Selenium Commands – "Selenese"

- Selenium commands, often called selenese, are the set of commands that run our tests. A sequence of these commands is a test script. Here we explain those commands in detail.
- Selenium provides a rich set of commands for fully testing our web-app in virtually any way we can imagine. These commands essentially create a testing language.
- In selenese, one can test the existence of UI elements based on their HTML tags, test for specific content, test for broken links, input fields, selection list options, submitting forms, and table data among other things. In addition Selenium commands support testing of window size, mouse position, alerts, Ajax functionality, pop up windows, event handling, and many other web-application features.
- A command is what tells Selenium what to do. Selenium commands come in three "flavors": Actions, Accessors, and Assertions.
- Actions are commands that generally manipulate the state of the application. They do things like "click this link" and "select that option". If an Action fails, or has an error, the execution of the current test is stopped.
- Many Actions can be called with the "AndWait" suffix, e.g. "clickAndWait". This suffix tells Selenium that the action will cause the browser to make a call to the server, and that Selenium should wait for a new page to load.
- Accessors examine the state of the application and store the results in variables, e.g. "storeTitle". They are also used to automatically generate Assertions.
- Assertions are like Accessors, but they verify that the state of the application conforms to what is expected. Examples include "make sure the page title is X" and "verify that this checkbox is checked".
- All Selenium Assertions can be used in 3 modes: "assert", "verify", and " waitFor". For example, we can "assertText", "verifyText" and "waitForText". When an "assert" fails, the test is aborted. When a "verify" fails, the test will continue execution, logging the failure. This allows a single "assert" to ensure that the application is on the correct page, followed by a bunch of "verify" assertions to test form field values, labels, etc.
- "waitFor" commands wait for some condition to become true (which can be useful for testing Ajax applications). They will succeed immediately if the condition is already true. However, they will fail and halt the test if the condition does not become true within the current timeout setting .

Script Syntax

Selenium commands are simple, they consist of the command and two parameters. For example:

The parameters are not always required; it depends on the command. In some cases both are required(eg.type), in others one parameter is required(eg.verifyTextPresent), and in still others the command may take no parameters at all.

Selenium scripts that will be run from Selenium-IDE will be be stored in an HTML text file format. This consists of an HTML table with three columns. The first column identifies the Selenium command, the second is a target, and the final column contains a value. The second and third columns may not require values depending on the chosen Selenium command, but they should be present. Each table row represents a new Selenium command. Here is an example of a test that opens a page, asserts the page title and then verifies some content on the page:

```
<table> <tr><td>open</td><td>/download/</td><td></td></tr>
        <tr><td>assertTitle</td><td></td><td>Downloads</td></tr>
        <tr><td>verifyText</td><td>//h2</td><td>Downloads</td></tr>
</table>
```

Rendered as a table in a browser this would look like the following:

Open	/download/	
assertTitle		Downloads
verifyText	//h2	Downloads

The Selenese HTML syntax can be used to write and run tests without requiring knowledge of a programming language. With a basic knowledge of selenese and Selenium-IDE we can quickly produce and run testcases.

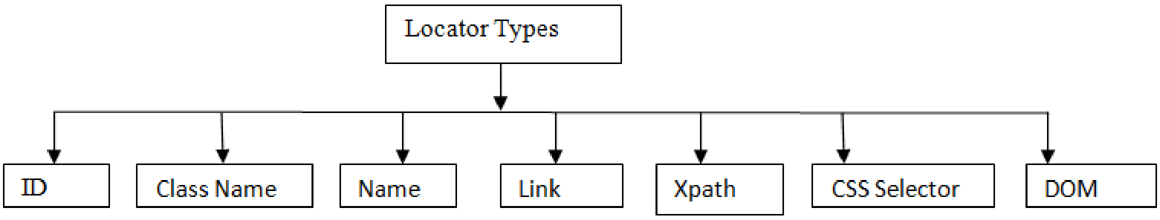
Commonly Used Selenium Commands

- The following are probably the most commonly used commands for building tests.

open : Opens a page using a URL.
type : command is useful for typing keyboard key values into text box of software web application.
echo: command is used to print the value in to the selenium IDS log.
click/clickAndWait : Performs a click operation, and optionally waits for a new page to load.
verifyTitle/assertTitle : Verifies an expected page title.
verifyTextPresent : Verifies expected text is somewhere on the page.
verifyElementPresent : Verifies an expected UI element, as defined by its HTML tag, is present on the page.
verifyText: Verifies expected text and its corresponding HTML tag are present on the page.
verifyTable : Verifies a table's expected contents.
waitForPageToLoad : Pauses execution until an expected new page loads. Called automatically when clickAndWait is used.
waitForElementPresent : Pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.

Locating Strategies

For many Selenium commands, a target is required. This target identifies an element in the content of the web application, and consists of the location strategy followed by the location in the format locatorType=location. The various locator types are explained below with examples for each.



Types of Locators in Selenium

Locator: - These are used by Selenium to find and match the elements of our AUT(Application Under Test) with which it needs to perform some action(like clicking, typing, selecting, verifying).It is used in Target column of Selenium IDE.
In simple words it tells Selenium which HTML element of our application a command has to perform the action.

Locating Principle:-
locatorType=location

Locator Type:-

Every object(control) visible on a webpage is a “WebElement”.
Selenium has different ways of locating controls (web elements).

- Id
- Class Name
- Name
- Link
- XPath
- CSS Selector
- DOM

ID:-
This approach is considered superior compared to other locatorsUnfortunately there are many cases when an element does not have an id (or the id is somehow dynamically generated and unpredictable). In these cases we will need to use an alternative locator strategy.
Target Format: id=id of the element
Example:- id = "userName"

Class Name : There may be multiple elements with the same name, if we just use find Element By Class Name.
Target Format: class=class name of the element
Example*:-* class=”D(ib) Py(0) Zoom Va(t) uhBtn Ff(ss)! Fw(40) Bxz(bb) Td(n) D(ib) Zoom Va(m) Ta(c) Bgr(rx) Bdrs(3px) Bdw(1px) M(0)! C(#fff) uh-ignore-rapid Cur(p)”

Name:-
Locating elements by name are very similar to locating by ID, except that we use the *"name=" prefix* instead.
Note:-If multiple elements have the same value for a name attribute, then we can use filters to further refine our location strategy.
Target Format: name=name of the element
Example :- name=”p”

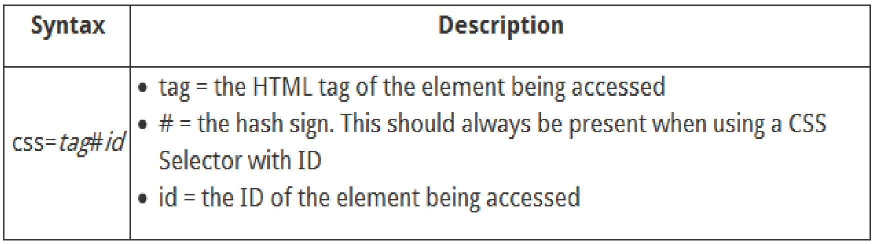
Link:- This approach uses a hyperlink in the web page to locate the element by using the text of the link.we can find elements of “**a**” tags(**Link**) with the link names. Use this when we know link text used within an anchor tag. If two links with the same text are present, then the first match will be used.

Target Format: link=link_text
Example*:-* link=Sign in

XPath:-
XPath is the language used when locating XML (Extensible Markup Language) nodes. While DOM is the recognized standard for navigation through an HTML element tree, XPath is the standard navigation tool for XML; and an HTML document is also an XML document (xHTML).
It can access almost any element, even those without class, name, or id attributes.

Target : xpath=//tagname[@attribute=”value”]
Example : xpath=//input[@id=”username”]
xpath=//form[@name=”loginForm”]
xpath=//*[@name=”loginForm”]

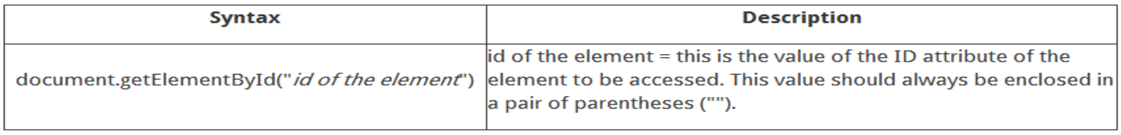
CSS:-
CSS is a language which is used for beautification of HTML controls like button should have green color.
CSS uses Selectors for binding style properties to elements in the document. These Selectors can be used by Selenium as another locating strategy.
Locating by CSS Selector is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.(Like X-Path)



Syntax	Description
css=*tag#id*	• tag = the HTML tag of the element being accessed • # = the hash sign. This should always be present when using a CSS Selector with ID • id = the ID of the element being accessed

Example: Using CSS ID,   css=form#loginForm
CSS Using Class:css=tagname.classname
Example: css= input.passfield
CSS Using name:css=tagname[name=’value’]
Example: css=input[name=’user name’]

DOM:-
The Document Object Model (DOM), in simple terms, is the way by which HTML elements are structured. Selenium is able to use the DOM structure in accessing page elements.



Syntax	Description
document.getElementById("*id of the element*")	id of the element = this is the value of the ID attribute of the element to be accessed. This value should always be enclosed in a pair of parentheses ("").

There are four basic ways to locate an element through DOM:

- getElementById
- getElementsByName
- dom:name (applies only to elements within a named form)
- dom:index

- Since selenium core is able to interpret the dom as document so we generally remove the locator type and directly use the location value.(Thus we can omit DOM as locator Type and use directly the location value)
  ***Example*:-**
  We can locate password object by its id, using getElementById
  we can either use dom=document.getElementById("password") or direct value.

2. Study of Any Open Source Testing Tool (e.g. Test Link)

TEST LINK

Test link is an open source test management tool. It enables creation and organization of test cases and helps manage into test plan. It allows execution of test cases from test link itself. One can easily track test results dynamically, generate reports, generate test metrics, prioritize test cases and assign unfinished tasks.

It's a web based tool with GUI, which provides an ease to develop test cases, organize test cases into test plans, execute these test cases and generate reports.

We can create test project and software test case document for the same. Test link tool is managed by admin. Admin can create different rolls according to work and assign them different task for manage whole project.

Test link exposes API, written in PHP, can help generate quality assurance dashboards. The functions like Add Test Case To Test Plan, Assign Requirements, Create Test Case etc. helps create and organize test cases per test plan. Functions like Get Test Cases For Test Plan, Get Last Execution Result allows one to create quality assurance dashboard.

Test Link enables easily to create and manage Test cases as well as organize them into Test plans. These Test plans allow team members to execute Test cases and track test results dynamically, generate reports, trace software requirements, prioritize and assign tasks.

Advantages of Test Link

- It supports multiple projects
- Easy export and import of test cases
- Easy to integrate with many defect management tools
- Automated test cases execution through XML-RPC
- Easy filtration of test cases with version, keywords, test case ID and version
- Easy to assign test cases to multiple users
- Easy to generate test plan and test reports in various formats
- Provide credentials to multiple users and assign roles to them

The following are the steps for test link

Login to Test Link

Step 1 : Open the Test link home-page and enter the login details

1. Enter the user ID – admin

2. Enter the password

3. Click on the login tab

Creating a Test Project

Step 1: In the main window click on Test Project Management, it will open another window.

Step 2: Click on tab "create" to create a new project.

Step 3: Enter all the required fields in the window like category for test project, name of the project, prefix, description, etc. After filling all necessary details, click on tab "Create" at the end of the window. This will create our project successfully.

Creating a Test Plan

Test plan holds the complete information like scope of testing, milestone, test suites and test cases. Once we have created a Test Project, next step is to create Test plan.
Step 1: From the home-page, click on Test Plan Management from home-page .

Step 2: It will open another page, at the bottom of the page click on a tab "Create".

Step 3: Fill out all the necessary information like name, description, create from existing test plan, etc. in the open window, and click on "Create tab"

Step 4: Yahoo Login Test Plan is created successfully.

Build Creation

Build is a specific release of software.

Step 1: Click on Build/Release under test plan from the homepage.

Step 2: In the next window, fill all necessary details for software release and click on create to save our release.

1. Enter the Title name.

2. Enter the description about the software release.

3. Mark the check box for status-Active.

4. Mark the check box for status-Public.

5. Choose the date of release

6. Click in Create button.

Creating Test Suite

Test suite is a collection of test cases which may be testing or validating the same component. Following are the steps for creating test suite.
Step 1: Click on Test specification option from the home page.

Step 2: On the right hand side of panel , Click on the setting icon ⚙ it will display series of test operations.
Step 3: Click on the Create tab for the test suite.
Step 4: Fill- up all the details of test suite and click on save tab.
1. Enter the test suit name
2. Enter the details about our test suite.
3. Click on save button to save the details of test suite.
We can see the test suite for yahoo login created.
Our Test suite is appears in the left hand side of panel, under folder structure tree.

Creating a Test case

Test case holds a sequence of test steps to test a specific scenario with expected result. Below steps will explain how to create a test-case along with test steps.
Step 1: Click on the test suite folder on the left side of the panel under folder tree structure.
Step 2: Click on the setting icon in the right side panel. List of test case operations will be displayed on the right side panel.
Step 3: New window will open, to create test cases click on create button in test-case operations .
Step 4: Enter the details in the test case specification page.
Step 5: After entering the details, click on "create" button to save the details. The test-case for yahoo login is created successfully.
Step 6: Click on test-case from the folder as shown above, it will open a window. Click on "create steps" button in test case. It will open a test case step editor .

Step 7: It will open another window on the same page, in that window we have to enter the following details
1. Enter the step-action for our test case.
2. Enter the details about the step action.
3. Click save it and add another step action OR click save and exit tab if there is no more test step to add
Step 8: Once we save and exit the test step, it will appear in the window.

Assigning test case to test plan

For test case to get execute, it should be assign to test plan. Here we will see how we can assign a test-case to test plan.

Step 1: Click on the setting icon ⚙ on the test panel. It will show the list of operations.
Step 2: Click on "Add to Test plans".
Step 3: New window will open ,search our project yahoo login.plan.
1. Mark the check box against our test plan.
2. Click on Add button.
This will add our test case to test plan.

Creating Users and assigning Roles in Test link

Test link provides user management and authorization features. Below list is default roles in test link and their rights.

Role	**Test Cases**	**Test Metrics**
Guest	View	View
Tester	Execute	View
Senior Tester	Edit & Execute	View
Leader & Admin	Edit & Execute	Edit & Execute

Step 1: From the test link home page ,click on users/roles icon from the navigator bar.
Step 2: Click on create.
Step 3: Fill out all the users details and click on save button.
Here is the list we can see the users have been created.
Step 4: Allotting test project role to the user.
1. Click on "Assigning Test Project Roles" tab.
2. Choose the project name.
3. Select the user Roles from the drop down.

Writing Requirements

Step 1: From the navigation bar click on the Requirements link it will open requirements page.
Step 2: From the requirements page ,on the right side of panel click on create button.
Step 3: A new window will open, enter all the details like
1. Document Id
2. Title name

3. Requirement Description
4.Click on “save” button
For the type we can choose the option from drop down ,here we choose “user-Requirement specifications”.
Step 4: It should create Requirements specification ,and displayed on left side panel under project yahoo login.
Step 5: Select the setting button from requirements specification home page.It will open another window.
Step 6: Click “create “tab under Requirement Operations.
Step 7: Fill out all the specified details and click on save button.
1. Enter the Document ID
2. Enter the Title name
3. Enter the description
4. Enter the Status
5. Enter the type
6. Enter the number of test cases needed
7. Click on “Save “ button at the end

Assigning Requirements to test cases

Step 1: From the test specification section, open any single test case and click on requirement icon.
Step 2: To assign Requirements specifications to test case we have to follow the below steps
1.Scroll down the drop down box to select the requirements specification
2. Mark the requirement check box
3. Click on “Assign” tab
After clicking on Assign tab a window will appear stating “Assigned Requirement”.

Executing A Test Case

In Test link we can run a test case and change execution status of a test case. Status of a test case can be set to Passed, Blocked or Failed. Initially it will be in Not Run status but once we have updated it, it can’t be altered to not run status again.
Step 1: From the Navigation bar Click on “Test Execution “ link. It will direct us to the test execution panel.
Step 2: Pick the test cases we want to run from the left side panel.
Step 3: Once we have selected a test case it will open a new window.
Step 4: Perform the following steps**.**
1. Enter the notes related to test case execution.
2. Select its status
Step 5: On the same page we have to fill the similar details about the execution of test case, Fill the details and select the status and click on “Save Execution”.

Generating Test Reports

Test link supports various test link formats like MS-Word, HTML,MS- Excel ,Open office Writer etc.
Step 1: From the navigation bar ,Click on Test Report option.
Step 2: From the left hand side panel, Click on “Test Report” link.
Step 3: To generate a report follow the following steps
1. Mark and unmark the option we want to highlight in our test report
2. Click on our project folder
The test report will appear in window.

Export Test case/ Test Suite

Test link provides the features to export test projects/test suites in our Test link and then we can import them into another Test link project on different server or system. In order to do that we have to follow the following step
Step 1: Choose the test case we want to export in the Test specification page.

Step 2: Now on the right-hand side of the panel click on the ⚙ setting icon, it will display all the operations that can be performed on the test case.

Step 3: Click the **"Export"** button

Step 4: It will open another window, mark the option as per requirement and click on the export tab.

Importing Test case/ Test suite

Step 1: Select the Test suite folder inside which we want to import the test case.

Step 2: Click on the setting icon ⚙ on the right hand-side of the panel, it will display all the operations that can be executed on the test suite/test case .

Step 3: Click on the import button in the test case operations list.

Step 4: Browse and attach the xml test case file that we have exported from test link and click on upload button.

1.Use the browse option to attach the XML test case file that we have exported from test link

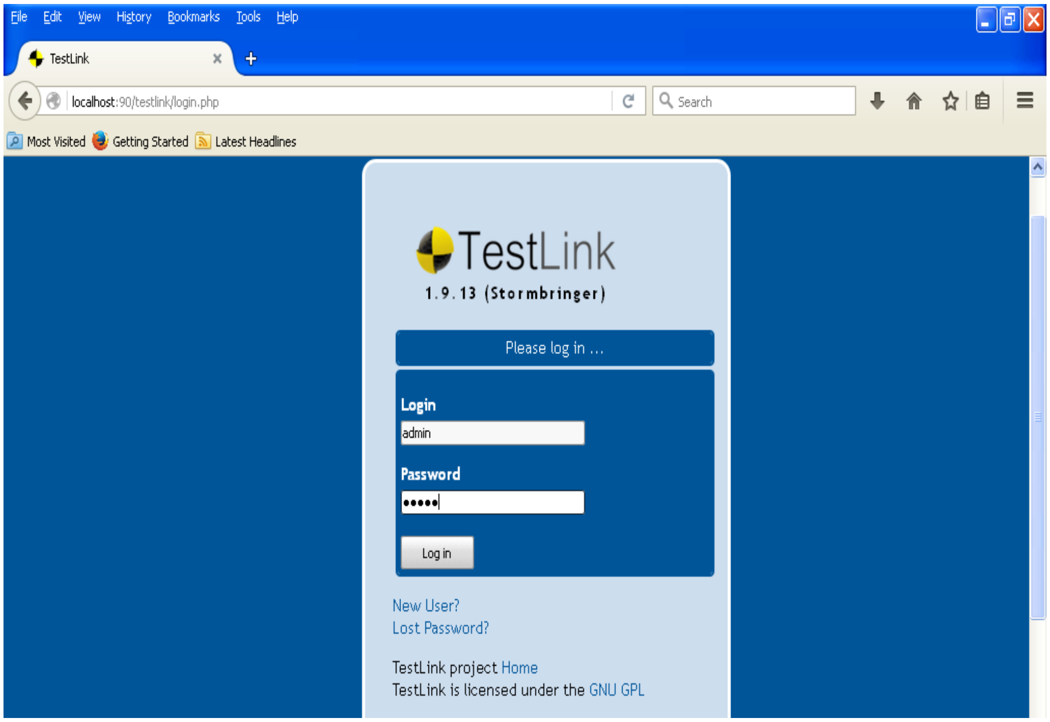
2.Click on upload file

When we upload a file, it will open window stating import test cases.

Step 5: Test case will be uploaded and displayed on the right-hand side of the panel

The following are the screenshots for Test link:

Login to Test Link



1.9.13 (Stormbringer)

Please log in ...

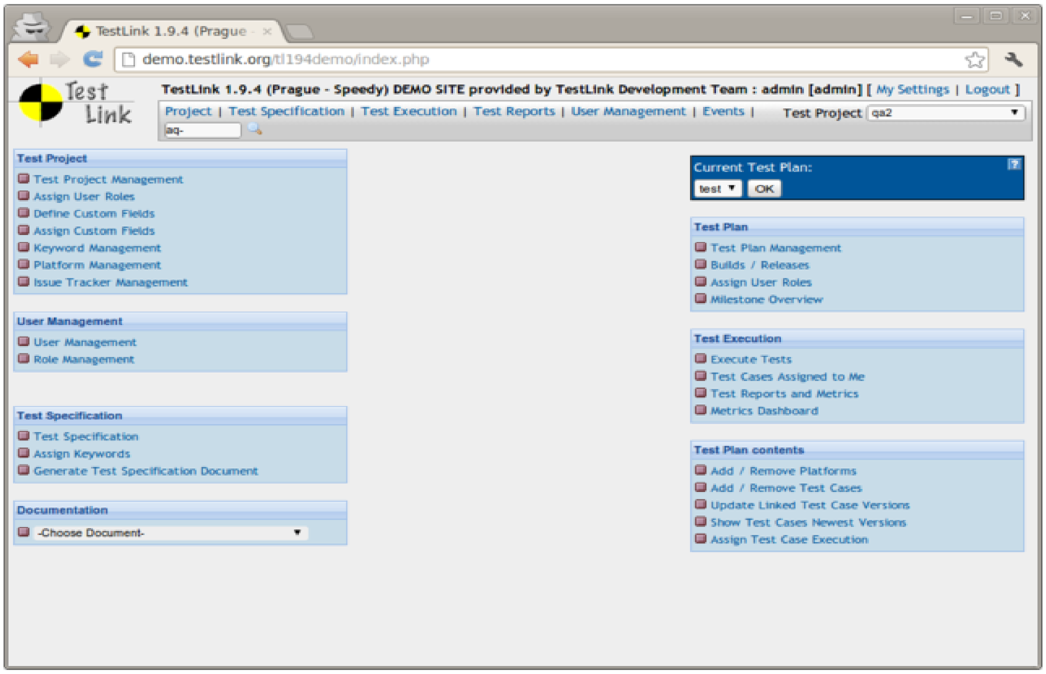
Login
admin

Password
•••••

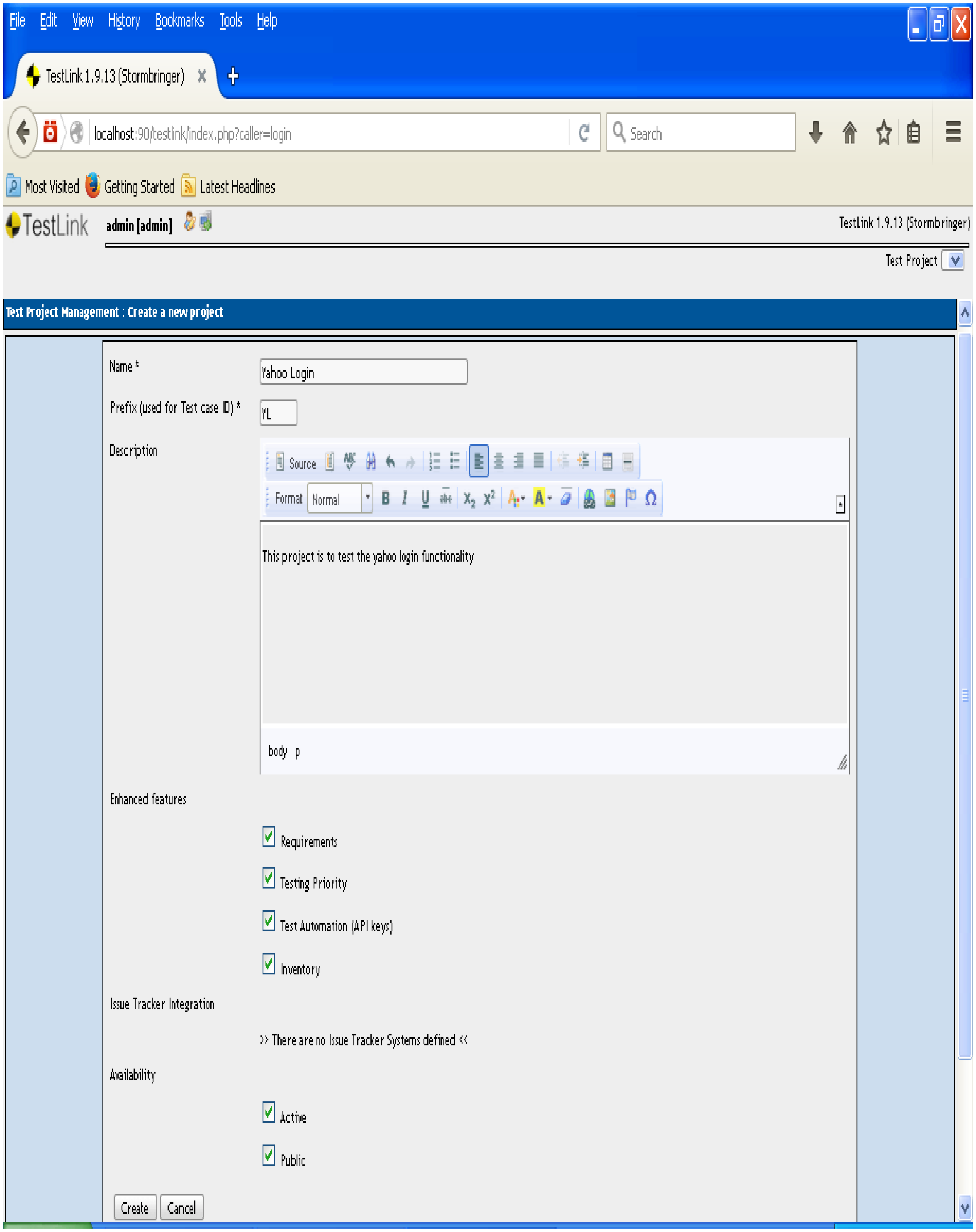
New User?
Lost Password?

TestLink project Home
TestLink is licensed under the GNU GPL

Test Link Homepage



Creating a Test Project



Test Project Management : Create a new project

Name * Yahoo Login

Prefix (used for Test case ID) * YL

Description

This project is to test the yahoo login functionality

Enhanced features

☑ Requirements

☑ Testing Priority

☑ Test Automation (API keys)

☑ Inventory

Issue Tracker Integration

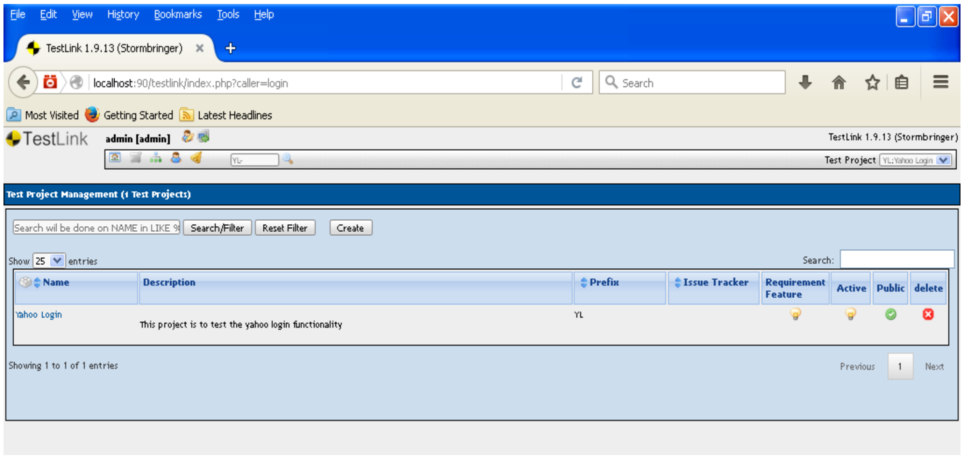
>> There are no Issue Tracker Systems defined <<

Availability

☑ Active

☑ Public

Yahoo Login Project created successfully



File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php?caller=login
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login

Test Project Management (1 Test Projects)

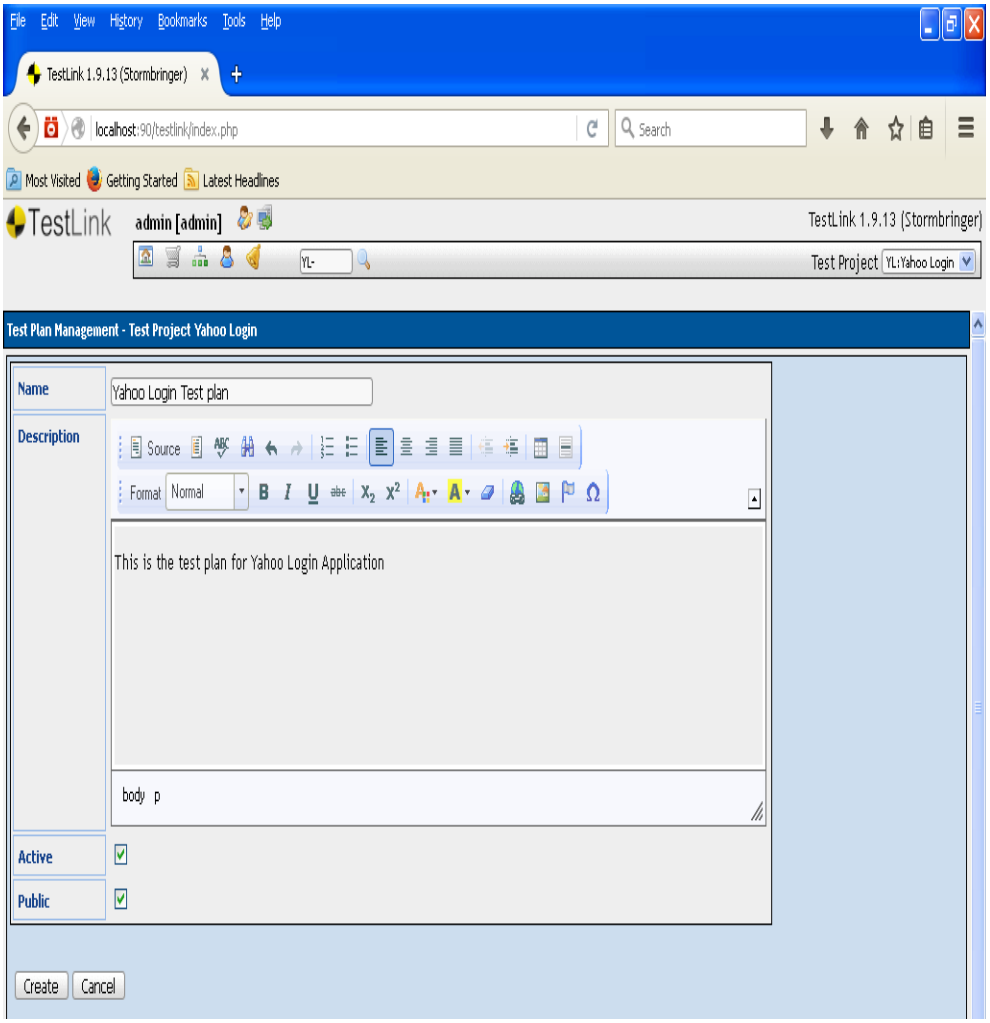
Search wil be done on NAME in LIKE % | Search/Filter | Reset Filter | Create

Show 25 entries Search:

Name	Description	Prefix	Issue Tracker	Requirement Feature	Active	Public	delete
Yahoo Login	This project is to test the yahoo login functionality	YL				✓	☒

Showing 1 to 1 of 1 entries

Creating a Test Plan



File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login

Test Plan Management - Test Project Yahoo Login

Name: Yahoo Login Test plan

Description:

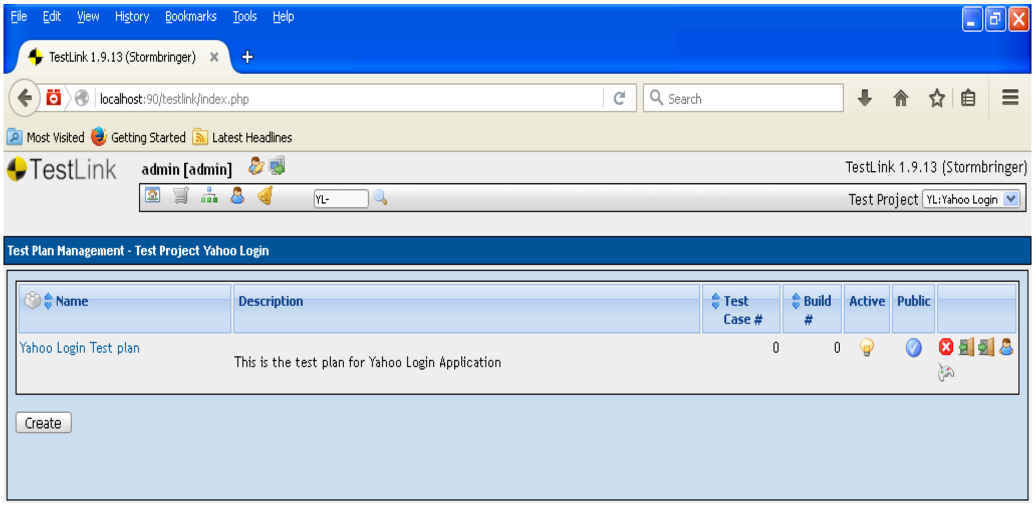
Source
Format Normal
B I U abc
body p

This is the test plan for Yahoo Login Application

Active ☑

Public ☑

Test Plan Created Successfully

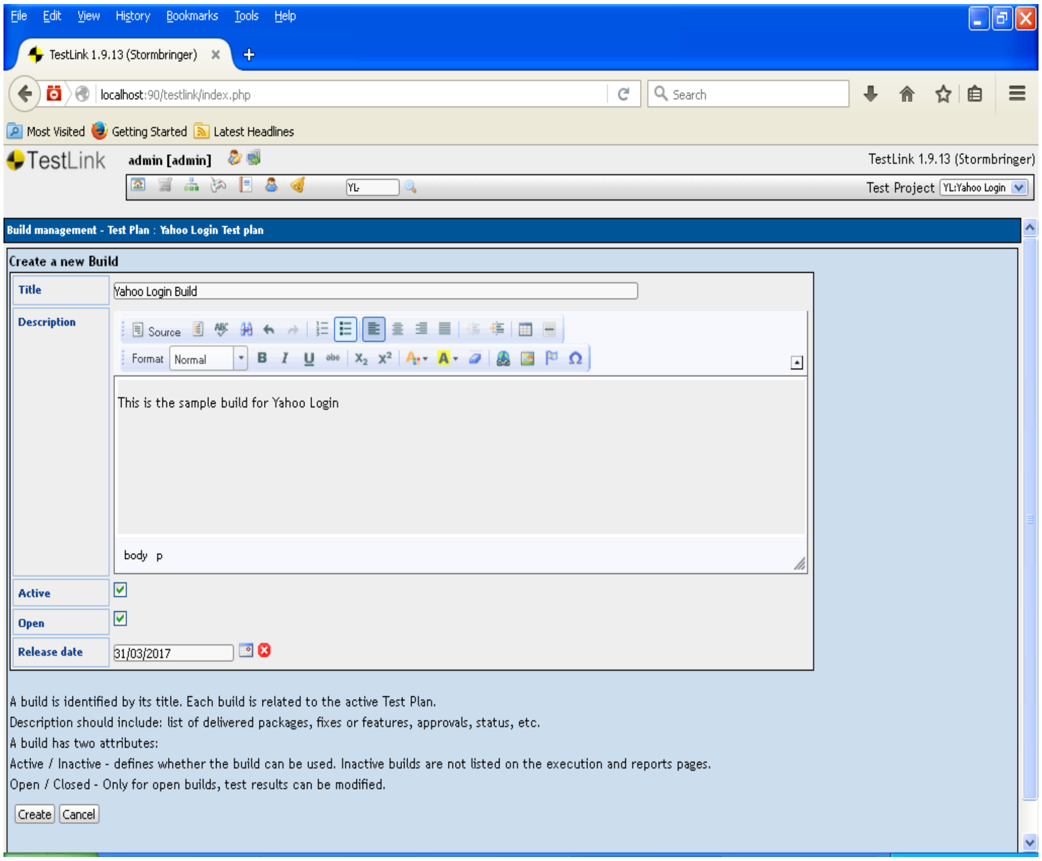


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login

Test Plan Management - Test Project Yahoo Login

Name	Description	Test Case #	Build #	Active	Public	
Yahoo Login Test plan	This is the test plan for Yahoo Login Application	0	0			

Build Creation



File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login
Source
Format Normal
B I U X₂ X²
body p

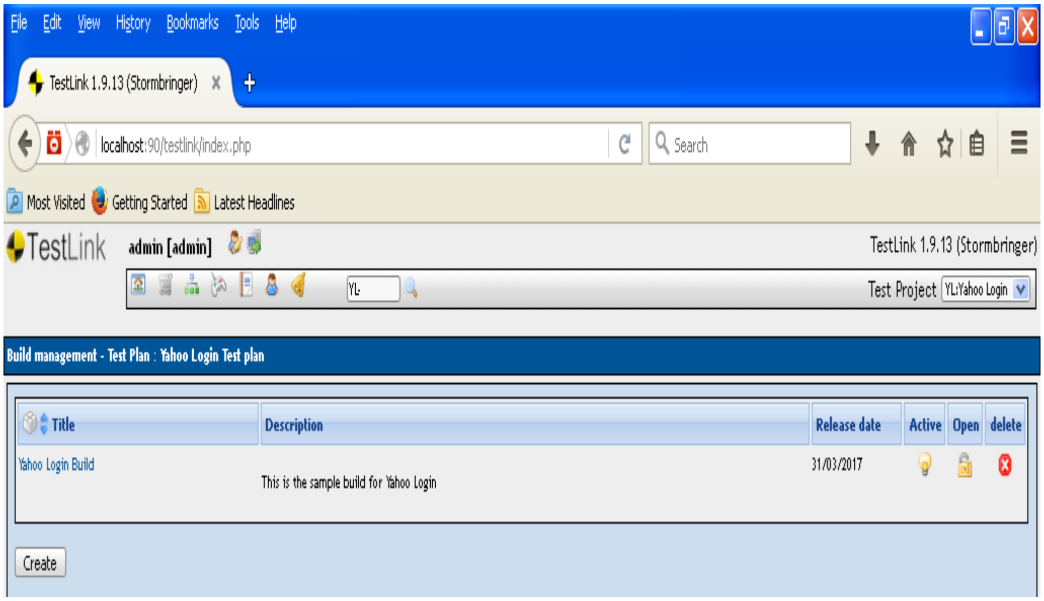
Build management - Test Plan : Yahoo Login Test plan

Create a new Build

Title	Yahoo Login Build
Description	This is the sample build for Yahoo Login
Active	☑
Open	☑
Release date	31/03/2017

A build is identified by its title. Each build is related to the active Test Plan.
Description should include: list of delivered packages, fixes or features, approvals, status, etc.
A build has two attributes:
Active / Inactive - defines whether the build can be used. Inactive builds are not listed on the execution and reports pages.
Open / Closed - Only for open builds, test results can be modified.

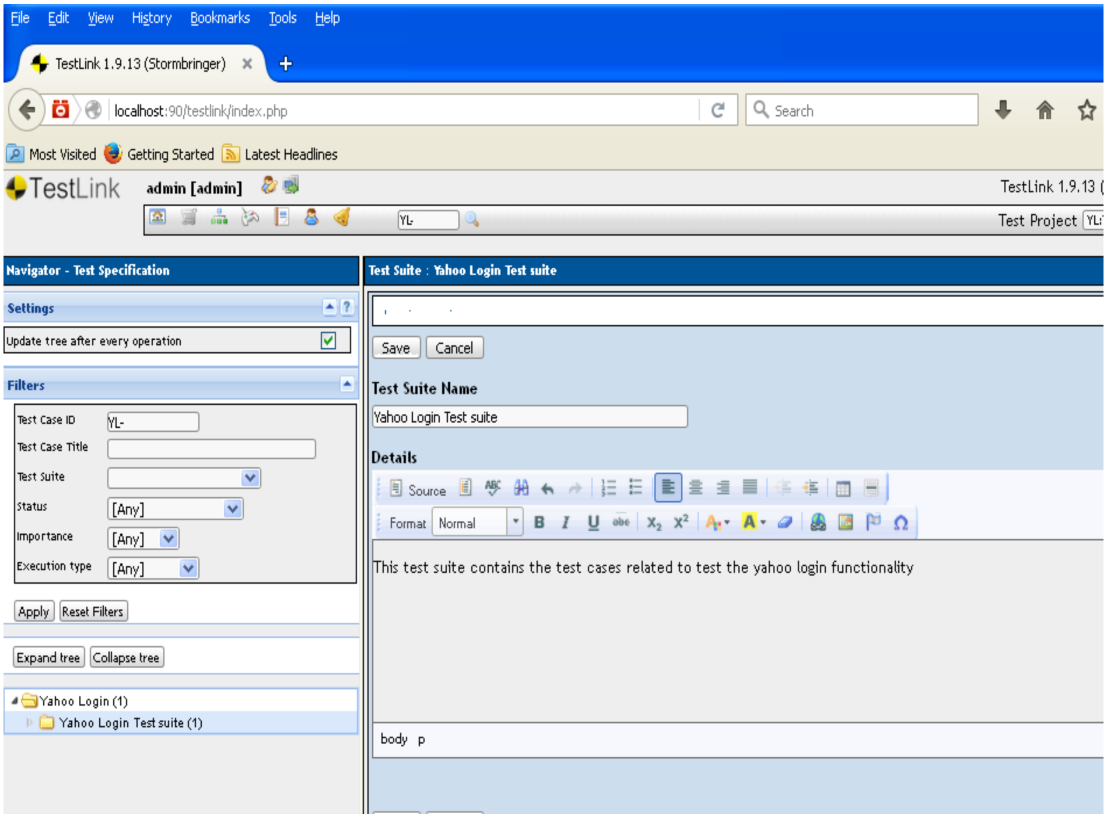
Build Created Successfully


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login

Build management - Test Plan : Yahoo Login Test plan

Title	Description	Release date	Active	Open	delete
Yahoo Login Build	This is the sample build for Yahoo Login	31/03/2017	💡	🔓	☒

Creating Test Suite


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (
YL-
Test Project YL:
Navigator - Test Specification
Settings
Filters
Apply
Reset Filters
Expand tree
Collapse tree
Save
Cancel
Source
Format
Normal
body p

Update tree after every operation ☑

Test Case ID: YL-

Test Case Title:

Test Suite:

Status: [Any]

Importance: [Any]

Execution type: [Any]

Yahoo Login (1)

Yahoo Login Test suite (1)

Test Suite : Yahoo Login Test suite

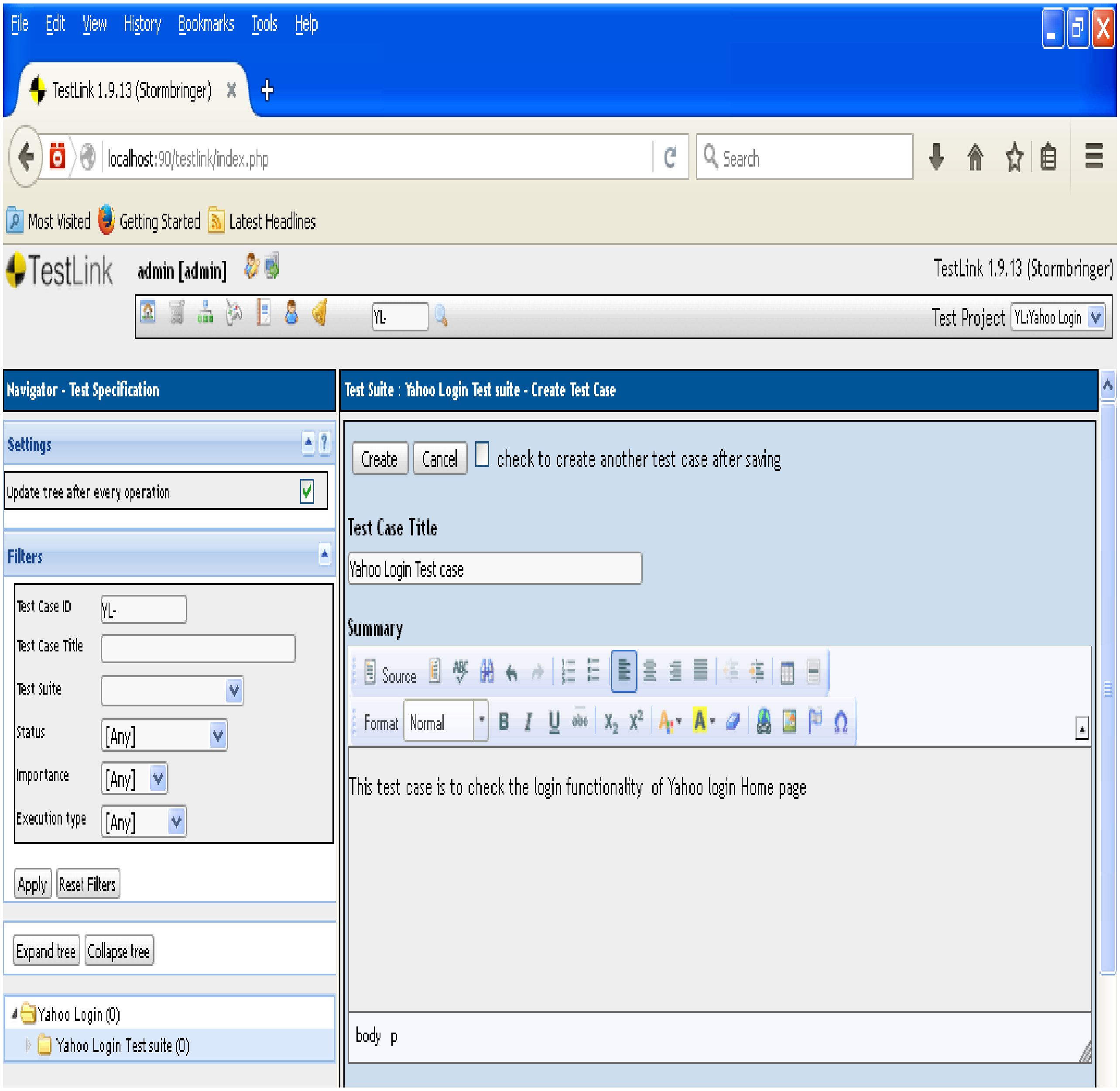
Test Suite Name

Yahoo Login Test suite

Details

This test suite contains the test cases related to test the yahoo login functionality

Creating a Test case



Test Suite : Yahoo Login Test suite - Create Test Case

□ check to create another test case after saving

Test Case Title

Yahoo Login Test case

Summary

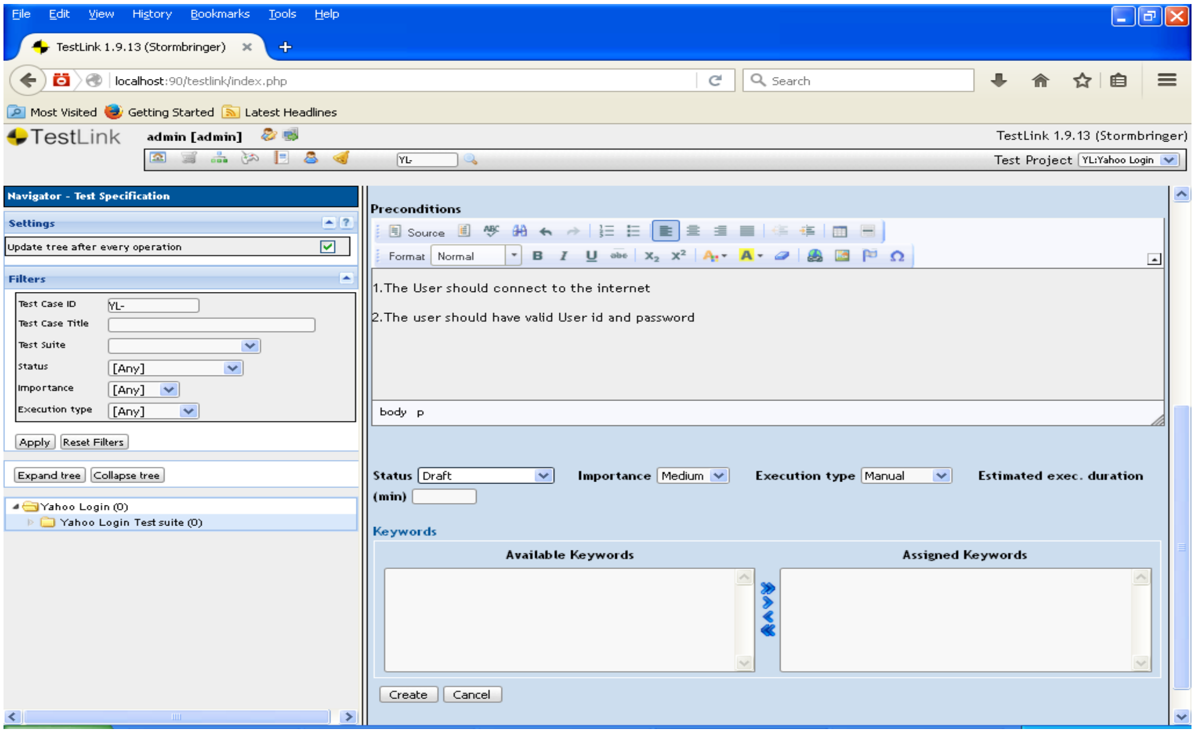
This test case is to check the login functionality of Yahoo login Home page

Update tree after every operation ☑

Test Case ID	YL-
Test Case Title	
Test Suite	
Status	[Any]
Importance	[Any]
Execution type	[Any]

- Yahoo Login (0)
  - Yahoo Login Test suite (0)

Creating a Test case



localhost:90/testlink/index.php

TestLink 1.9.13 (Stormbringer)

admin [admin]

Test Project YL:Yahoo Login

Navigator - Test Specification

Update tree after every operation ☑

Test Case ID: YL-

Status: [Any]

Importance: [Any]

Execution type: [Any]

Yahoo Login (0)
Yahoo Login Test suite (0)

Preconditions

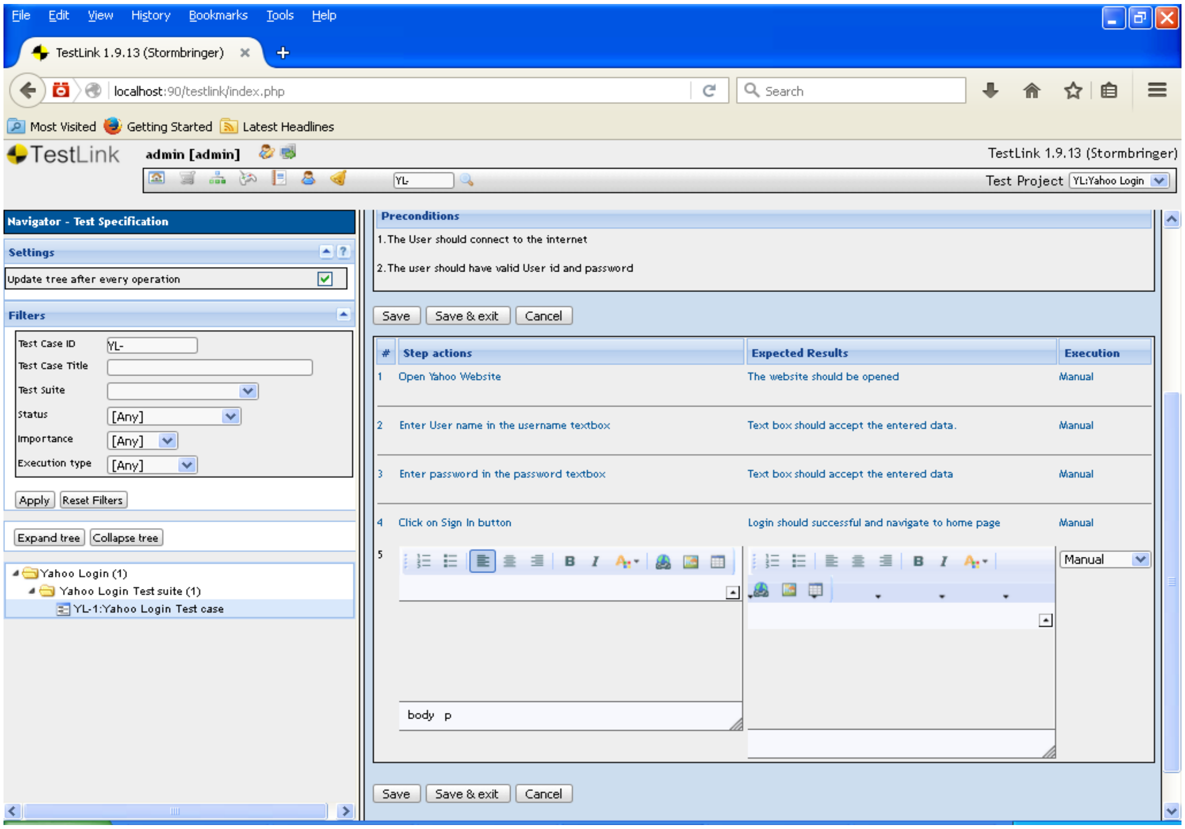
1.The User should connect to the internet

2.The user should have valid User id and password

Status: Draft | Importance: Medium | Execution type: Manual | Estimated exec. duration (min):

Keywords

Available Keywords | Assigned Keywords



localhost:90/testlink/index.php

TestLink 1.9.13 (Stormbringer)

admin [admin]

Test Project YL:Yahoo Login

Navigator - Test Specification

Update tree after every operation ☑

Test Case ID: YL-

Status: [Any]

Importance: [Any]

Execution type: [Any]

Yahoo Login (1)
Yahoo Login Test suite (1)
YL-1:Yahoo Login Test case

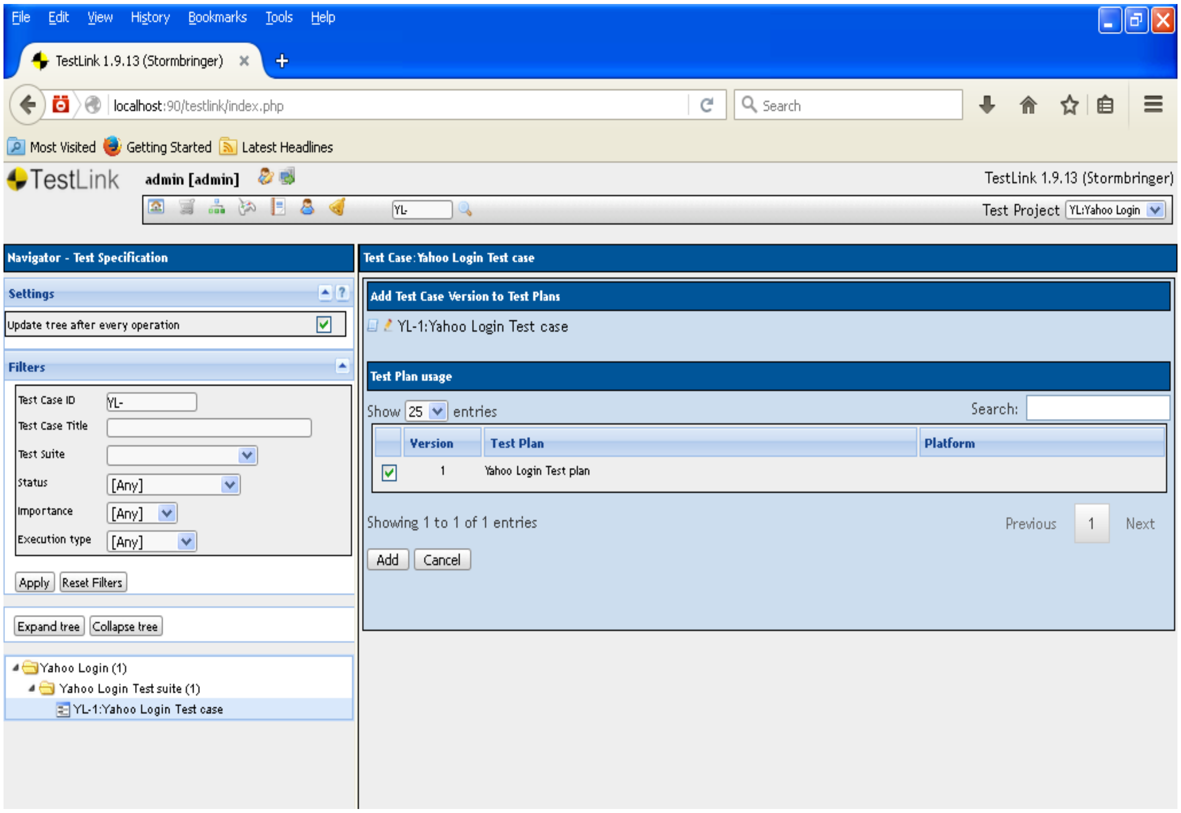
Preconditions

1.The User should connect to the internet

2.The user should have valid User id and password

#	Step actions	Expected Results	Execution
1	Open Yahoo Website	The website should be opened	Manual
2	Enter User name in the username textbox	Text box should accept the entered data.	Manual
3	Enter password in the password textbox	Text box should accept the entered data	Manual
4	Click on Sign In button	Login should successful and navigate to home page	Manual
5			Manual

Assigning test case to test plan


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
Apply
Reset Filters
Expand tree
Collapse tree
Previous
1
Next
Add
Cancel

Test Project: YL:Yahoo Login

Navigator - Test Specification

Settings

Update tree after every operation ☑

Filters

Test Case ID	YL-
Test Case Title	
Test Suite	
Status	[Any]
Importance	[Any]
Execution type	[Any]

- Yahoo Login (1)
  - Yahoo Login Test suite (1)
    - YL-1:Yahoo Login Test case

Test Case : Yahoo Login Test case

Add Test Case Version to Test Plans

YL-1:Yahoo Login Test case

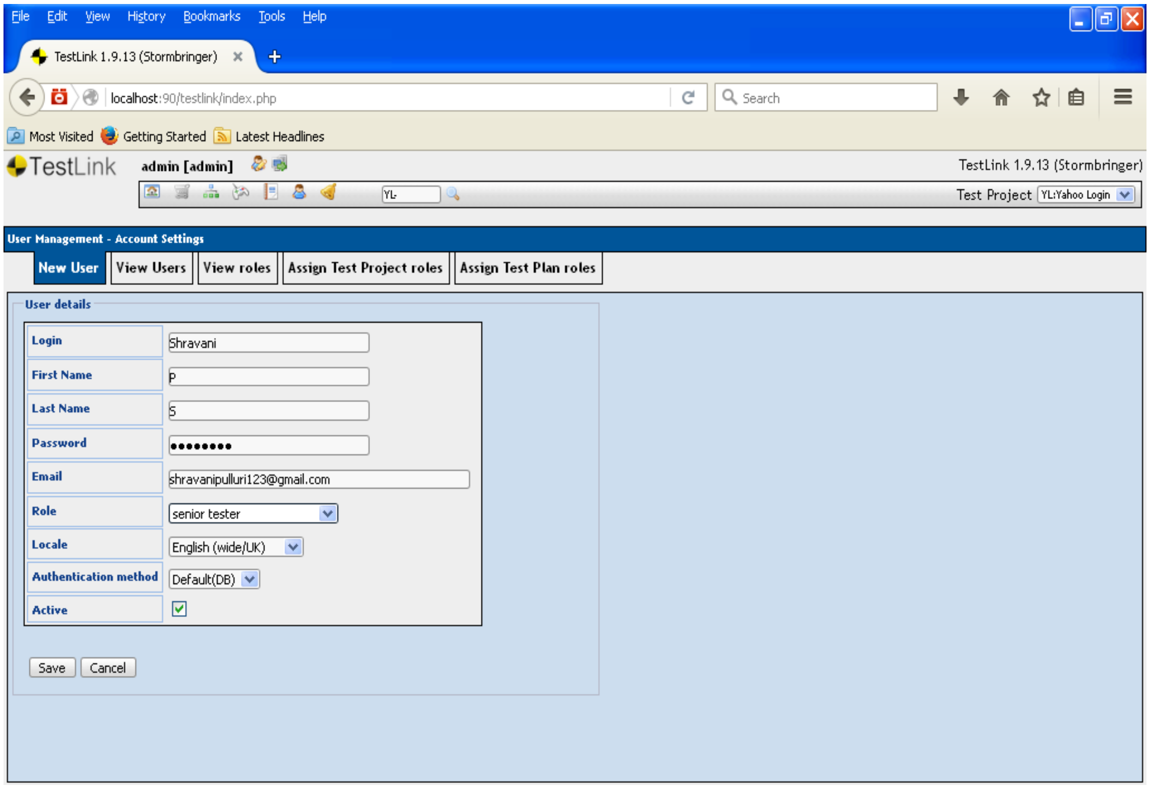
Test Plan usage

Show 25 entries Search:

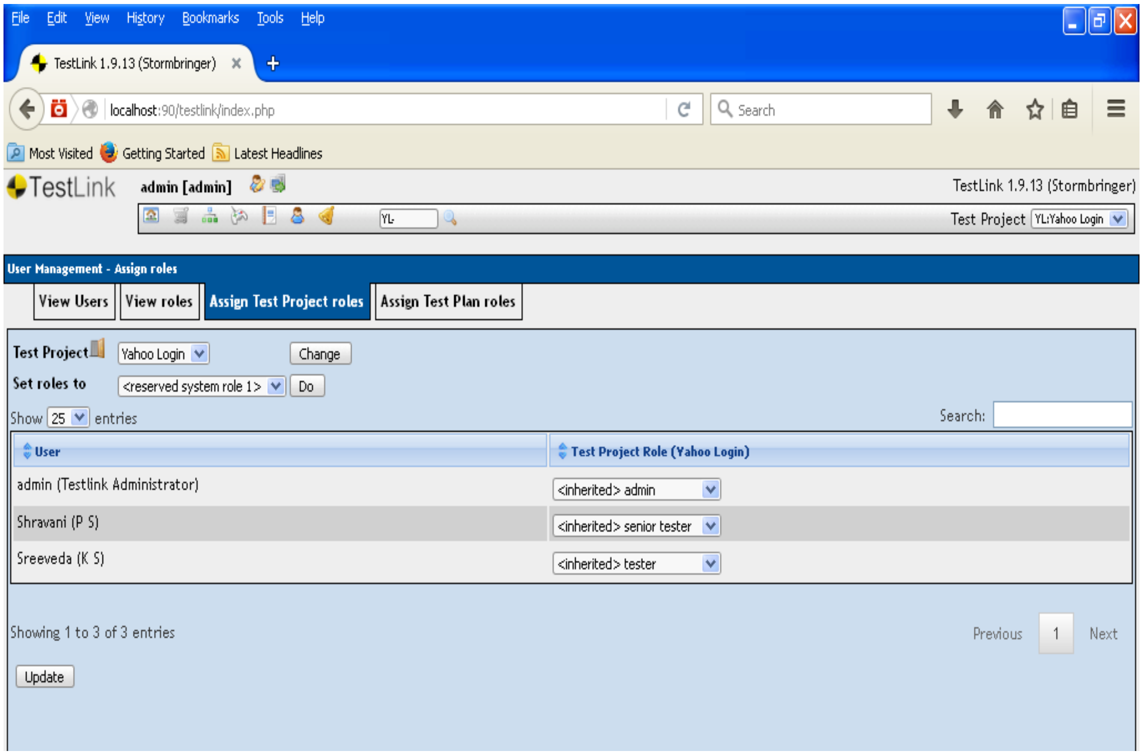
	Version	Test Plan	Platform
☑	1	Yahoo Login Test plan	

Showing 1 to 1 of 1 entries

Creating Users and assigning Roles in Test link


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
Test Project YL:Yahoo Login
User Management - Account Settings
New User
View Users
View roles
Assign Test Project roles
Assign Test Plan roles
User details
Save
Cancel

Login	Shravani
First Name	P
Last Name	S
Password	••••••••
Email	shravanipulluri123@gmail.com
Role	senior tester
Locale	English (wide/UK)
Authentication method	Default(DB)
Active	☑



User Management - Assign roles

Test Project Yahoo Login

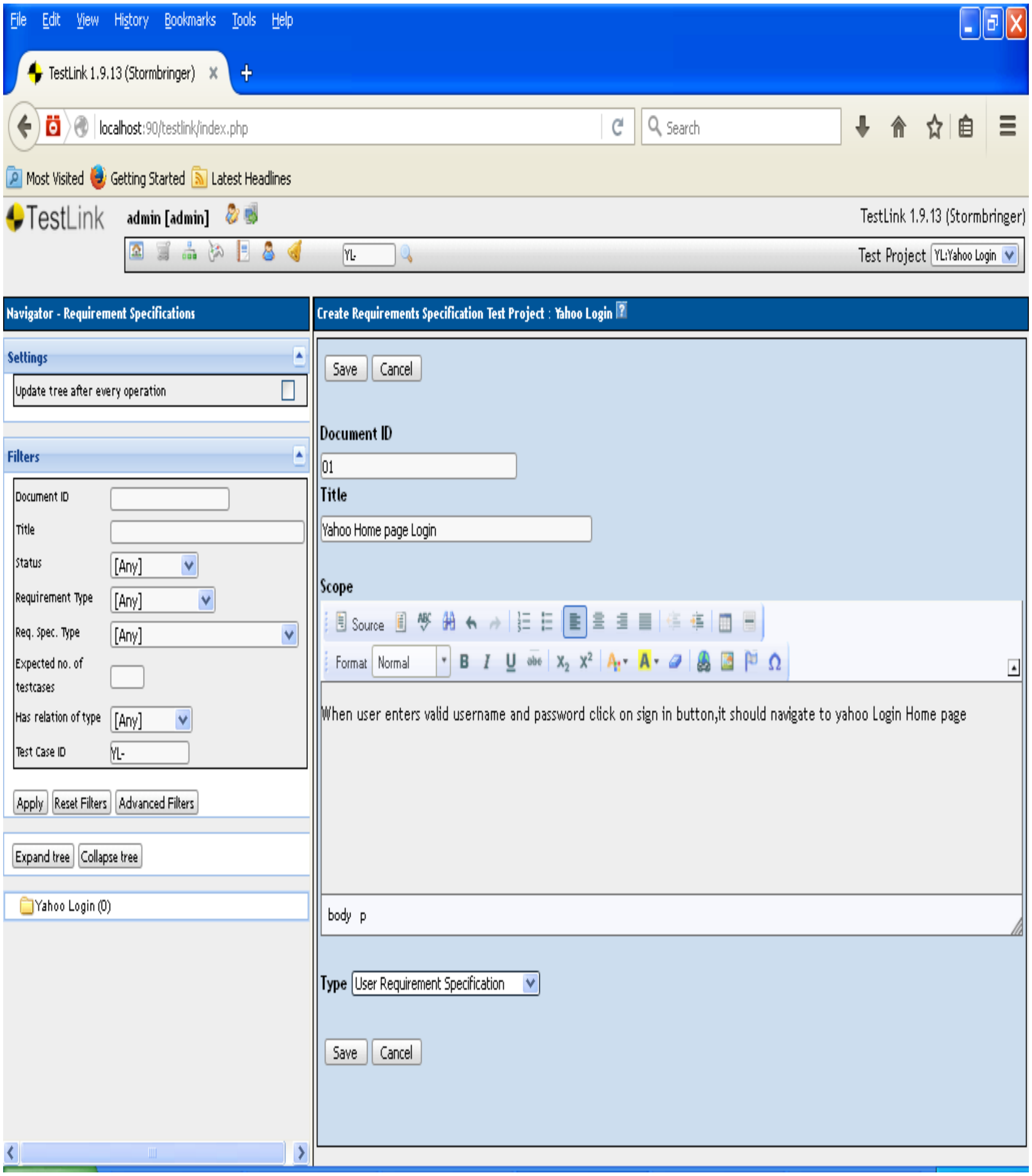
Set roles to <reserved system role 1>

Show 25 entries

User	Test Project Role (Yahoo Login)
admin (Testlink Administrator)	<inherited> admin
Shravani (P S)	<inherited> senior tester
Sreeveda (K S)	<inherited> tester

Showing 1 to 3 of 3 entries

Writing Requirements


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login
Navigator - Requirement Specifications
Settings
Update tree after every operation
Filters
Document ID
Title
Status [Any]
Requirement Type [Any]
Req. Spec. Type [Any]
Expected no. of testcases
Has relation of type [Any]
Test Case ID YL-
Apply
Reset Filters
Advanced Filters
Expand tree
Collapse tree
Yahoo Login (0)
Save
Cancel
Source
Format Normal
body p

Create Requirements Specification Test Project : Yahoo Login

Document ID

01

Title

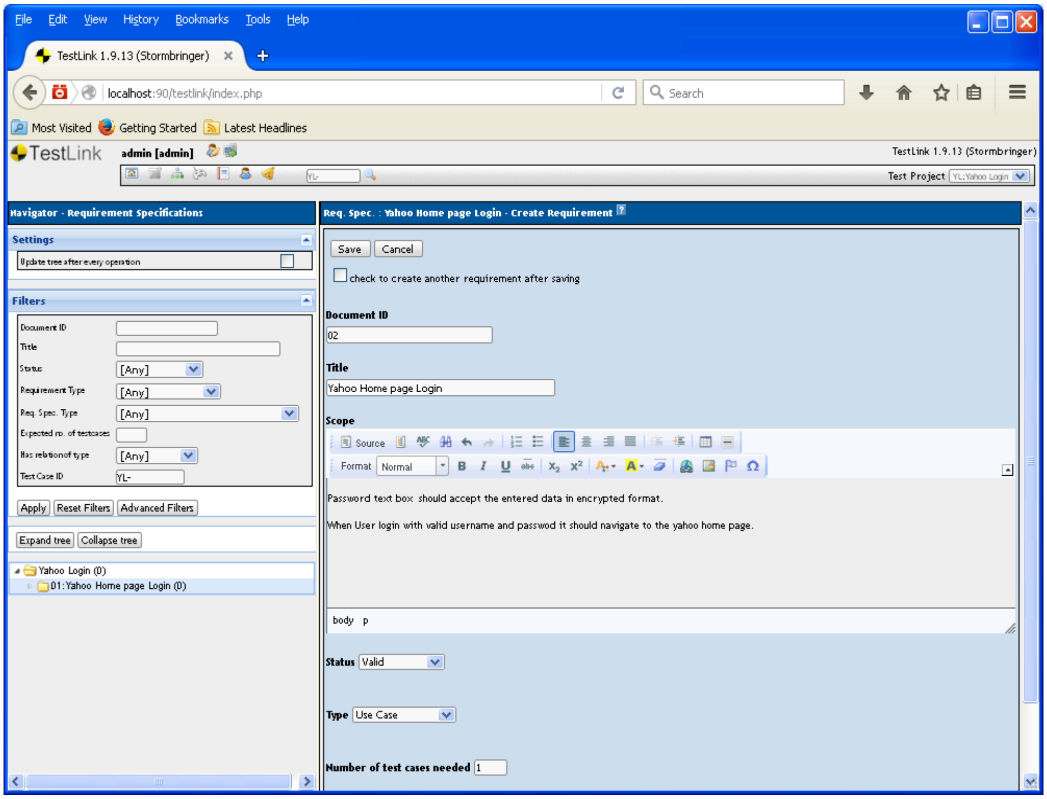
Yahoo Home page Login

Scope

When user enters valid username and password click on sign in button,it should navigate to yahoo Login Home page

Type User Requirement Specification

Writing Requirement Operations


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
Most Visited Getting Started Latest Headlines
Search
Source Format Normal
Save Cancel
Apply Reset Filters Advanced Filters
Expand tree Collapse tree

localhost:90/testlink/index.php

TestLink admin [admin] TestLink 1.9.13 (Stormbringer)

Test Project YL:Yahoo Login

Navigator - Requirement Specifications

Settings

Update tree after every operation □

Filters

Document ID	
Title	
Status	[Any]
Requirement Type	[Any]
Req. Spec. Type	[Any]
Expected no. of testcases	
Has relation of type	[Any]
Test Case ID	YL-

Yahoo Login (0)
01:Yahoo Home page Login (0)

Req. Spec. : Yahoo Home page Login - Create Requirement

□ check to create another requirement after saving

Document ID
02

Title
Yahoo Home page Login

Scope

Password text box should accept the entered data in encrypted format.

When User login with valid username and passwod it should navigate to the yahoo home page.

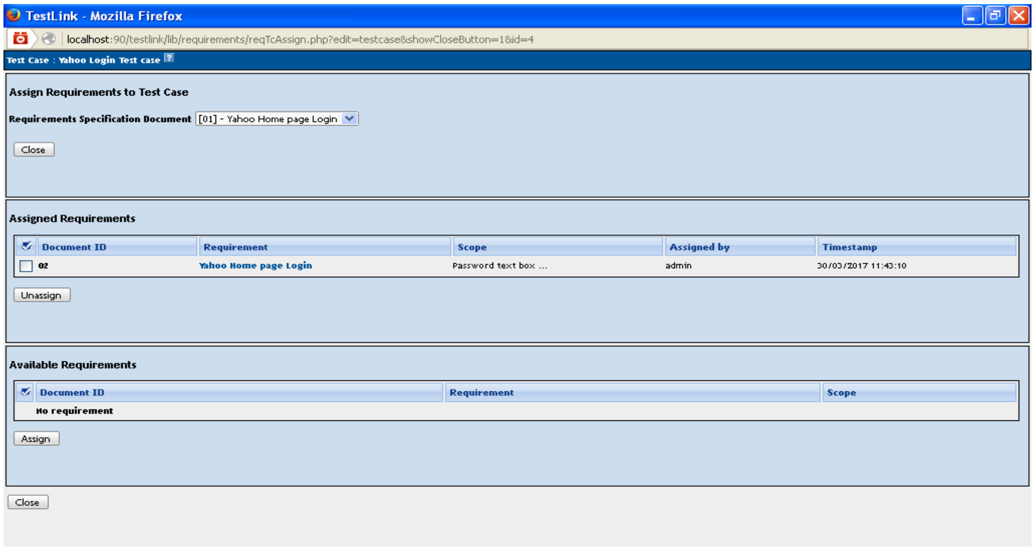
body p

Status Valid

Type Use Case

Number of test cases needed 1

Assigning Requirements to Test case



localhost:90/testlink/lib/requirements/reqTcAssign.php?edit=testcase&showCloseButton=1&id=4

Test Case : Yahoo Login Test case

Assign Requirements to Test Case

Requirements Specification Document [01] - Yahoo Home page Login

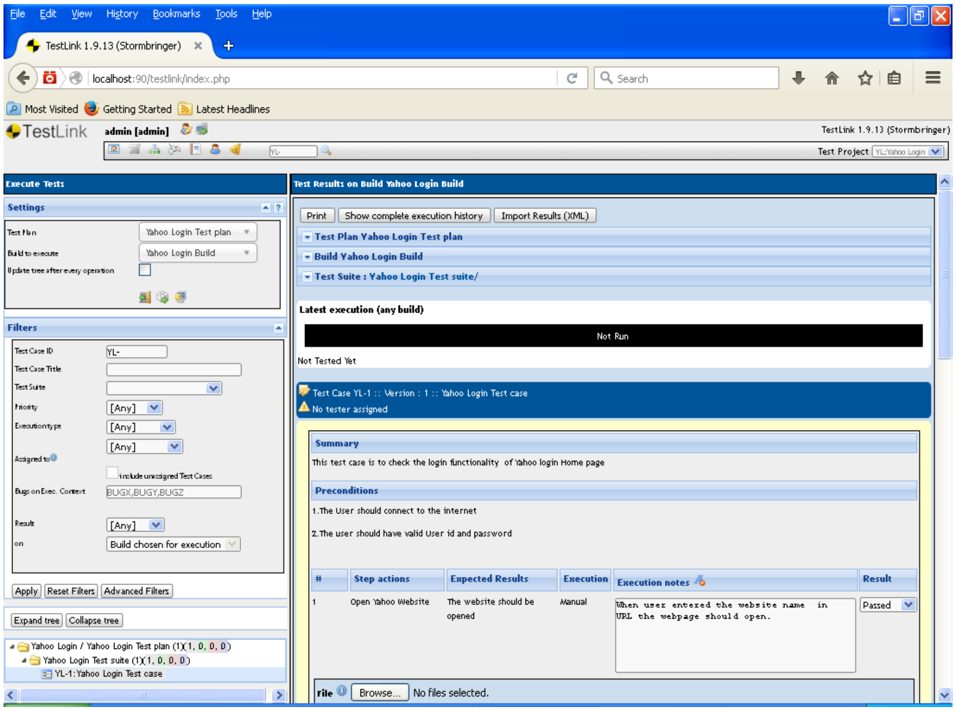
Assigned Requirements

	Document ID	Requirement	Scope	Assigned by	Timestamp
□	02	Yahoo Home page Login	Password text box ...	admin	30/03/2017 11:43:10

Available Requirements

	Document ID	Requirement	Scope
	No requirement		

Test Execution


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
Test Project YL:Yahoo Login
Execute Tests
Settings
Filters
Apply
Reset Filters
Advanced Filters
Expand tree
Collapse tree
Print
Show complete execution history
Import Results (XML)
Browse...

Test Plan: Yahoo Login Test plan

Build to execute: Yahoo Login Build

Update tree after every operation ☐

Test Case ID: YL-

Priority: [Any]

Execution type: [Any]

Assigned to: [Any]

☐ include unassigned Test Cases

Bugs on Exec. Context: BUGX,BUGY,BUGZ

Result: [Any]

on: Build chosen for execution

Yahoo Login / Yahoo Login Test plan (1)(1, 0, 0, 0)
Yahoo Login Test suite (1)(1, 0, 0, 0)
YL-1:Yahoo Login Test case

Test Results on Build Yahoo Login Build

Test Plan Yahoo Login Test plan

Build Yahoo Login Build

Test Suite : Yahoo Login Test suite/

Latest execution (any build)

Not Run

Not Tested Yet

Test Case YL-1 :: Version : 1 :: Yahoo Login Test case

No tester assigned

Summary

This test case is to check the login functionality of Yahoo login Home page

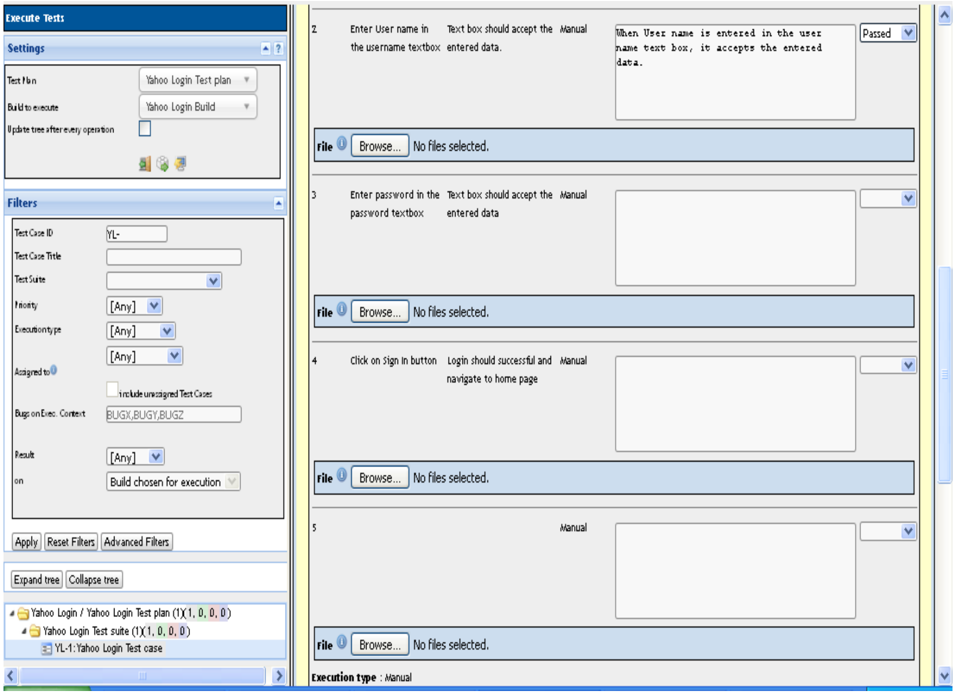
Preconditions

1.The User should connect to the internet

2.The user should have valid User id and password

#	Step actions	Expected Results	Execution	Execution notes	Result
1	Open Yahoo Website	The website should be opened	Manual	When user entered the website name in URL the webpage should open.	Passed

File: No files selected.



Test Plan: Yahoo Login Test plan

Build to execute: Yahoo Login Build

Update tree after every operation ☐

Test Case ID: YL-

Priority: [Any]

Execution type: [Any]

Assigned to: [Any]

☐ include unassigned Test Cases

Bugs on Exec. Context: BUGX,BUGY,BUGZ

Result: [Any]

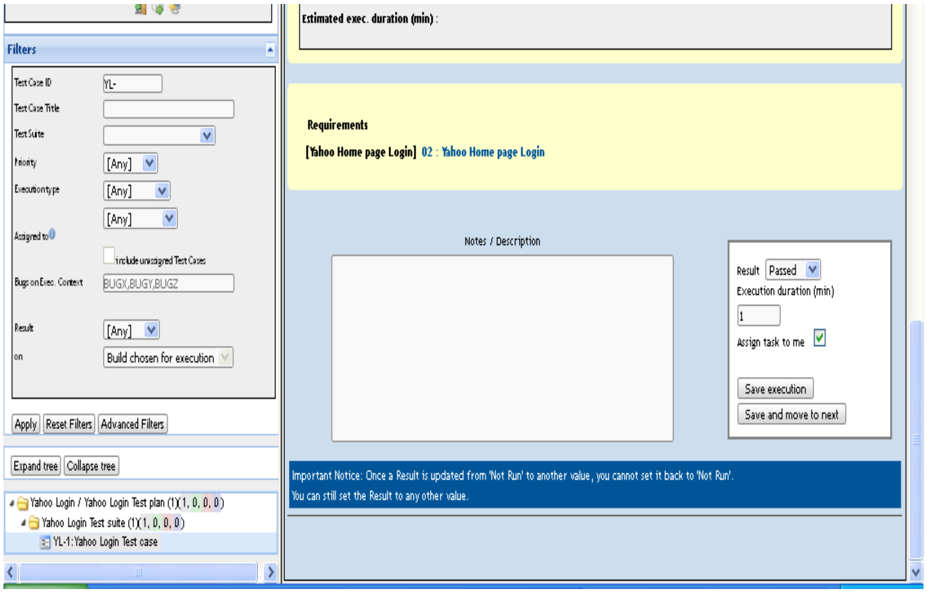
on: Build chosen for execution

Yahoo Login / Yahoo Login Test plan (1)(1, 0, 0, 0)
Yahoo Login Test suite (1)(1, 0, 0, 0)
YL-1:Yahoo Login Test case

#	Step actions	Expected Results	Execution	Execution notes	Result
2	Enter User name in the username textbox	Text box should accept the entered data.	Manual	When User name is entered in the user name text box, it accepts the entered data.	Passed
3	Enter password in the password textbox	Text box should accept the entered data	Manual		
4	Click on Sign in button	Login should successful and navigate to home page	Manual		
5			Manual		

File: No files selected.

Execution type : Manual



Test Case ID: YL-
Test Case Title:
Test Suite:
Priority: [Any]
Execution type: [Any]
Assigned to: [Any]
□ include unassigned Test Cases
Bugs on Exec. Context: BUGX,BUGY,BUGZ
Result: [Any]
on: Build chosen for execution

Yahoo Login / Yahoo Login Test plan (1)(1, 0, 0, 0)
Yahoo Login Test suite (1)(1, 0, 0, 0)
YL-1:Yahoo Login Test case

Estimated exec. duration (min) :

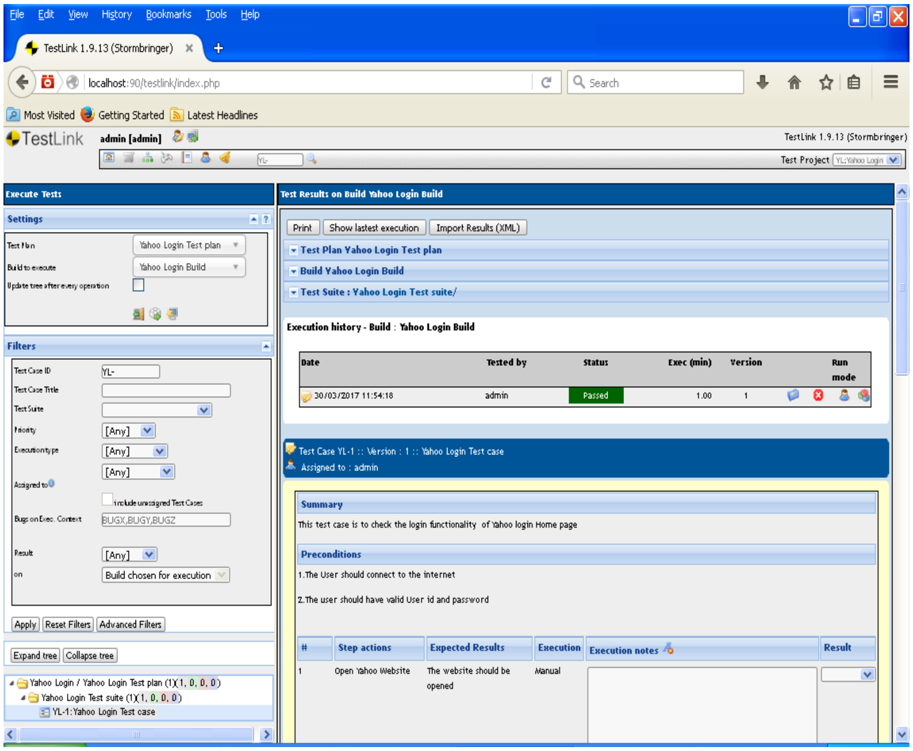
Requirements

[Yahoo Home page Login] 02 : Yahoo Home page Login

Notes / Description

Result: Passed
Execution duration (min): 1
Assign task to me ☑

Important Notice: Once a Result is updated from 'Not Run' to another value, you cannot set it back to 'Not Run'.
You can still set the Result to any other value.



localhost:90/testlink/index.php

admin [admin]
TestLink 1.9.13 (Stormbringer)
Test Project: YL:Yahoo Login

Test Plan: Yahoo Login Test plan
Build to execute: Yahoo Login Build
Update tree after every operation □

Test Case ID: YL-
Test Case Title:
Test Suite:
Priority: [Any]
Execution type: [Any]
Assigned to: [Any]
□ include unassigned Test Cases
Bugs on Exec. Context: BUGX,BUGY,BUGZ
Result: [Any]
on: Build chosen for execution

Yahoo Login / Yahoo Login Test plan (1)(1, 0, 0, 0)
Yahoo Login Test suite (1)(1, 0, 0, 0)
YL-1:Yahoo Login Test case

Test Results on Build Yahoo Login Build

Test Plan Yahoo Login Test plan

Build Yahoo Login Build

Test Suite : Yahoo Login Test suite/

Execution history - Build : Yahoo Login Build

Date	Tested by	Status	Exec (min)	Version	Run mode
30/03/2017 11:54:18	admin	Passed	1.00	1	

Test Case YL-1 :: Version : 1 :: Yahoo Login Test case
Assigned to : admin

Summary

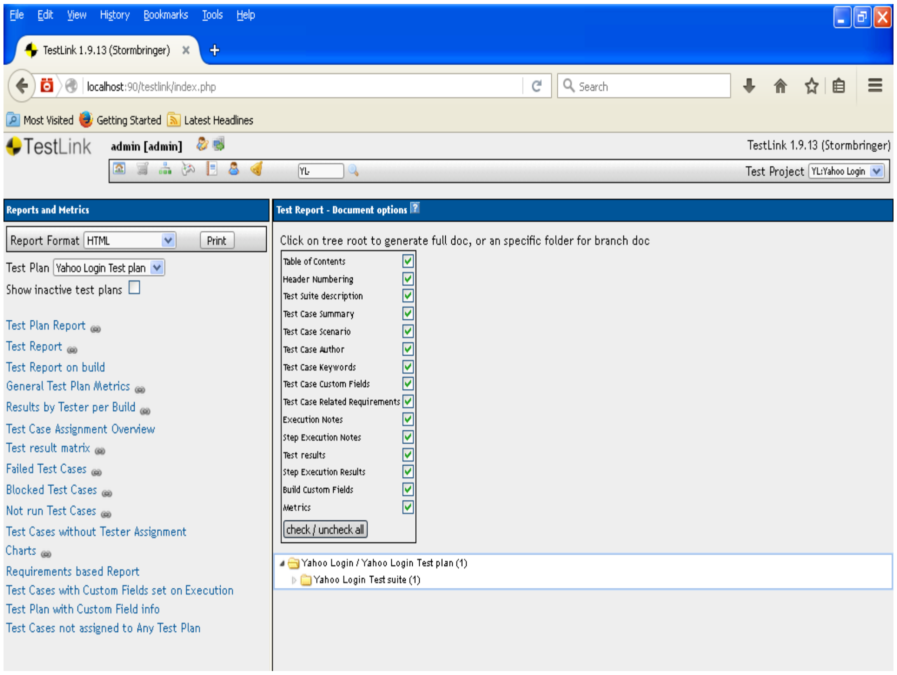
This test case is to check the login functionality of Yahoo login Home page

Preconditions

1.The User should connect to the internet

2.The user should have valid User id and password

#	Step actions	Expected Results	Execution	Execution notes	Result
1	Open Yahoo Website	The website should be opened	Manual		


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login
Reports and Metrics
Print
Test Plan Report
Test Report
Test Report on build
General Test Plan Metrics
Results by Tester per Build
Test Case Assignment Overview
Test result matrix
Failed Test Cases
Blocked Test Cases
Not run Test Cases
Test Cases without Tester Assignment
Charts
Requirements based Report
Test Cases with Custom Fields set on Execution
Test Plan with Custom Field info
Test Cases not assigned to Any Test Plan
Test Report - Document options
check / uncheck all

Report Format HTML

Test Plan Yahoo Login Test plan

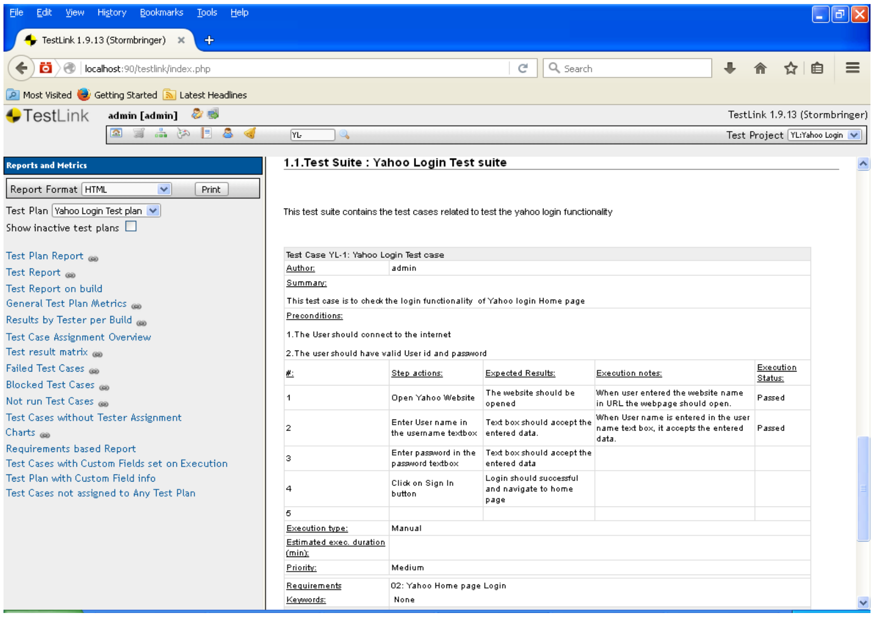
Show inactive test plans □

Click on tree root to generate full doc, or an specific folder for branch doc

Table of Contents	☑
Header Numbering	☑
Test Suite description	☑
Test Case Summary	☑
Test Case Scenario	☑
Test Case Author	☑
Test Case Keywords	☑
Test Case Custom Fields	☑
Test Case Related Requirements	☑
Execution Notes	☑
Step Execution Notes	☑
Test results	☑
Step Execution Results	☑
Build Custom Fields	☑
Metrics	☑

Yahoo Login / Yahoo Login Test plan (1)
Yahoo Login Test suite (1)

Generating Test Reports


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited Getting Started Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login
Reports and Metrics
Print
Test Plan Report
Test Report
Test Report on build
General Test Plan Metrics
Results by Tester per Build
Test Case Assignment Overview
Test result matrix
Failed Test Cases
Blocked Test Cases
Not run Test Cases
Test Cases without Tester Assignment
Charts
Requirements based Report
Test Cases with Custom Fields set on Execution
Test Plan with Custom Field info
Test Cases not assigned to Any Test Plan

Report Format HTML

Test Plan Yahoo Login Test plan

Show inactive test plans □

1.1.Test Suite : Yahoo Login Test suite

This test suite contains the test cases related to test the yahoo login functionality

Test Case YL-1: Yahoo Login Test case

Author: admin

Summary:

This test case is to check the login functionality of Yahoo login Home page

Preconditions:

1.The User should connect to the internet

2.The user should have valid User id and password

#:	Step actions:	Expected Results:	Execution notes:	Execution Status:
1	Open Yahoo Website	The website should be opened	When user entered the website name in URL the webpage should open.	Passed
2	Enter User name in the username textbox	Text box should accept the entered data.	When User name is entered in the user name text box, it accepts the entered data.	Passed
3	Enter password in the password textbox	Text box should accept the entered data		
4	Click on Sign In button	Login should successful and navigate to home page		
5				

Execution type: Manual

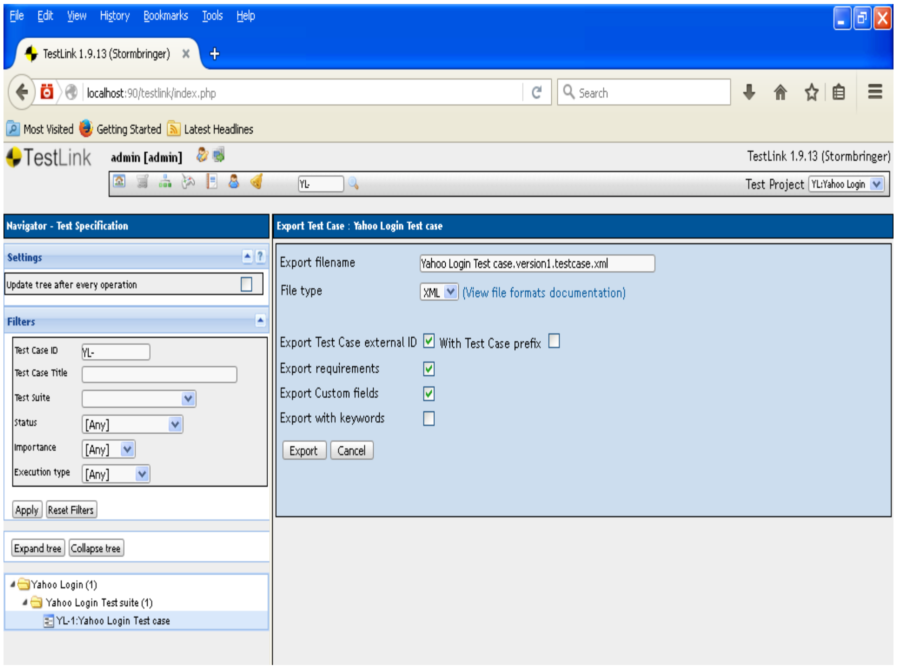
Estimated exec. duration (min):

Priority: Medium

Requirements 02: Yahoo Home page Login

Keywords: None

Exporting Test case


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login
Navigator - Test Specification
Settings
Update tree after every operation □
Filters
Apply
Reset Filters
Expand tree
Collapse tree
Yahoo Login (1)
Yahoo Login Test suite (1)
YL-1:Yahoo Login Test case

Test Case ID: YL-
Test Case Title:
Test Suite:
Status: [Any]
Importance: [Any]
Execution type: [Any]

Export Test Case : Yahoo Login Test case

Export filename: Yahoo Login Test case.version1.testcase.xml

File type: XML (View file formats documentation)

Export Test Case external ID ☑ With Test Case prefix □

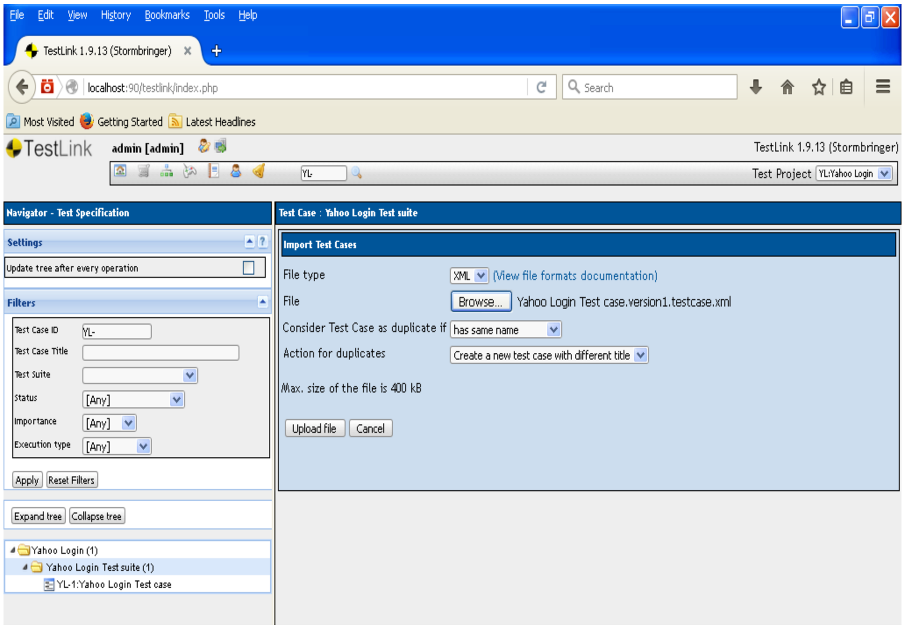
Export requirements ☑

Export Custom fields ☑

Export with keywords □

Export | Cancel

Importing Test case


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
YL-
Test Project YL:Yahoo Login
Navigator - Test Specification
Settings
Update tree after every operation □
Filters
Apply
Reset Filters
Expand tree
Collapse tree
Yahoo Login (1)
Yahoo Login Test suite (1)
YL-1:Yahoo Login Test case

Test Case ID: YL-
Test Case Title:
Test Suite:
Status: [Any]
Importance: [Any]
Execution type: [Any]

Test Case : Yahoo Login Test suite

Import Test Cases

File type: XML (View file formats documentation)

File: Browse... Yahoo Login Test case.version1.testcase.xml

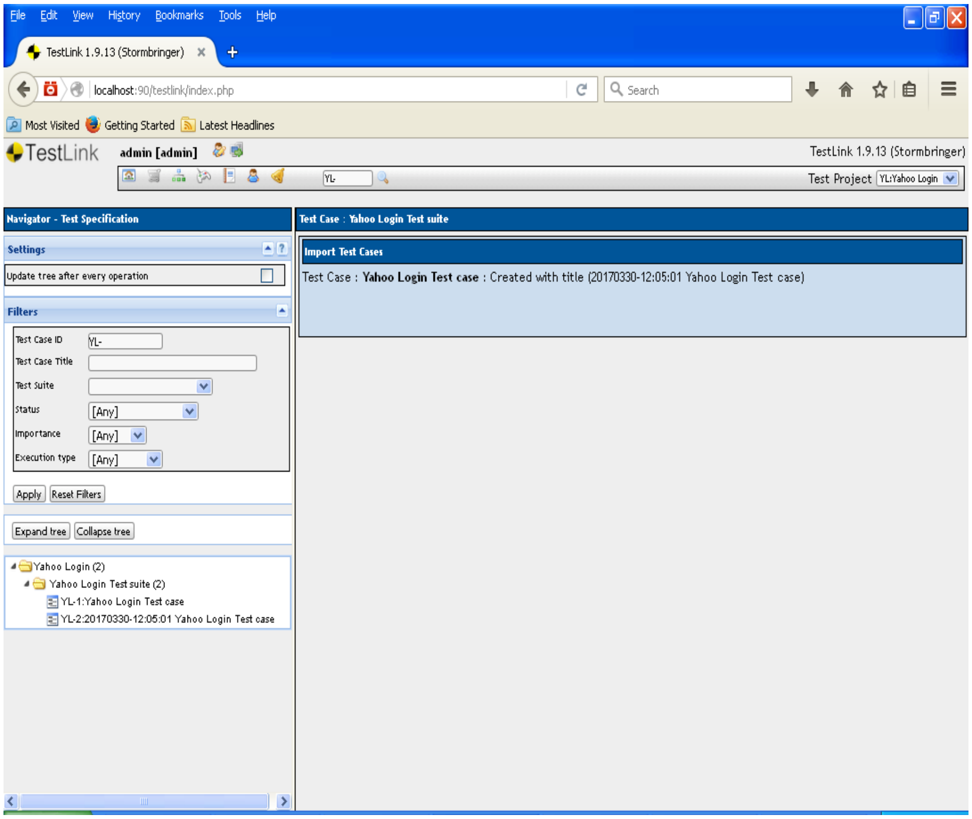
Consider Test Case as duplicate if: has same name

Action for duplicates: Create a new test case with different title

Max. size of the file is 400 kB

Upload file | Cancel

Test case Imported Successfully


File Edit View History Bookmarks Tools Help
TestLink 1.9.13 (Stormbringer)
localhost:90/testlink/index.php
Search
Most Visited
Getting Started
Latest Headlines
TestLink
admin [admin]
TestLink 1.9.13 (Stormbringer)
Test Project
Navigator - Test Specification
Test Case : Yahoo Login Test suite
Settings
Filters
Apply
Reset Filters
Expand tree
Collapse tree

Test Project: YL:Yahoo Login

Update tree after every operation □

Test Case ID	YL-
Test Case Title	
Test Suite	
Status	[Any]
Importance	[Any]
Execution type	[Any]

Yahoo Login (2)
Yahoo Login Test suite (2)
YL-1:Yahoo Login Test case
YL-2:20170330-12:05:01 Yahoo Login Test case

Import Test Cases

Test Case : **Yahoo Login Test case** : Created with title (20170330-12:05:01 Yahoo Login Test case)

3. Study of Any Bug Tracking Tool (e.g.Bugzilla)

BUGZILLA

Bugzilla is a “Bug Tracking System” that can efficiently keep track of outstanding bugs in a product. Multiple users can access this database and query, add and manage these bugs. Bugzilla essentially comes to the rescue of a group of people working together on a product as it enables them to view current bugs and make contributions to resolve issues.

Its basic repository nature works out better than the mailing list concept and an organized database is always easier to work with.

Advantage of Using Bugzilla:

1. Bugzilla is very adaptable to various situations. Known uses currently include IT support queues, Systems Administration deployment management, chip design and development problem tracking (both pre-and-post fabrication), and software and hardware bug tracking for luminaries such as Red hat, NASA, Linux-Mandrake, and VA Systems. Combined with systems such as CVS, Bugzilla provides a powerful, easy-to-use solution to configuration management and replication problems.

2. Bugzilla can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. Ultimately, Bugzilla puts the power in user’s hands to improve value to business while providing a usable framework for natural attention to detail and knowledge store to flourish.

The bugzilla utility basically allows to do the following:

- Add a bug into the database
- Review existing bug reports
- Manage the content

Bugzilla is organized in the form of bug reports that give all the information needed about a particular bug. A bug report would consist of the following fields.

- Product–>Component
- Assigned to
- Status (New, Assigned, Fixed etc)
- Summary
- Bug priority
- Bug severity (blocker, trivial etc)
- Bug reporter

Using Bugzilla:

Bugzilla usage involves the following activities

- Setting Parameters and Default Preferences
- Creating a New User
- Impersonating a User
- Adding Products
- Adding Product Components
- Modifying Default Field Values
- Creating a New Bug
- Viewing Bug Reports

Setting Parameters and Default Preferences:

When we start using Bugzilla, we’ll need to set a small number of parameters and preferences. At a minimum, we should change the following items, to suit our particular need:

▪ Set the *maintainer*
▪ Set the *mail_delivery_method*
▪ Set *bug change policies*
▪ Set the display order of bug reports

To set parameters and default preferences:

1. Click *Parameters* at the bottom of the page.
2. Under *Required Settings*, add an email address in the *maintainer* field.
3. Click *Save Changes*.
4. In the left side *Index* list, click *Email*.
5. Select from the list of mail transports to match the transport we’re using. If evaluating a click2try application, select *Test*. If using SMTP, set any of the other SMTP options for your environment. Click *Save Changes*.

6. In the left side *Index* list, click *Bug Change Policies*.
7. Select On for *comment on create*, which will force anyone who enters a new bug to enter a comment, to describe the bug. Click *Save Changes*.
8. Click *Default Preferences* at the bottom of the page.
9. Select the display order from the drop-down list next to the *When viewing a bug, show comments in this order* field. Click *Submit Changes*.

Creating a New User

Before entering bugs, make sure we add some new users. We can enter users very easily, with a minimum of information. Bugzilla uses the email address as the user ID, because users are frequently notified when a bug is entered, either because they entered the bug, because the bug is assigned to them, or because they've chosen to track bugs in a certain project.

To create a new user:

1. Click **Users**.
2. Click add a new user.
3. Enter the **Login name**, in the form of an email address.
4. Enter the **Real name**, a password, and then click **Add**.
5. Select the **Group access options**. we'll probably want to enable the following options in the row titled User is a member of these groups:
   - *canconfirm*
   - *editbugs*
   - *editcomponents*
6. Click **Update** when done with setting options.

Impersonating a User

Impersonating a user is possible, though rare, that we may need to file or manage a bug in an area that is the responsibility of another user when that user is not available. Perhaps the user is on vacation, or is temporarily assigned to another project. We can impersonate the user to create or manage bugs that belong to that user.

Adding Products

We'll add a product in Bugzilla for every product we are developing. To start with, when we first login to Bugzilla, we'll find a test product called **TestProduct**. We should delete this and create a new product.

To add a product:

1. At the bottom of the page, click **Products**.
2. In the **TestProduct** listing, click **Delete**.
3. Click **Yes, Delete**.
4. Now click **Add a product**.
5. Enter a product name, such as "Widget Design Kit."
6. Enter a description.
7. Click **Add**. A message appears that you'll need to add at least one component.

Adding Product Components

Products are comprised of components. Software products, in particular, are typically made up of many functional components, which in turn are made up of program elements, like classes and functions. It's not unusual in a software development team environment for different individuals to be responsible for the bugs that are reported against a given component. Even if there are other programmers working on that component, it's not uncommon for one person, either a project lead or manager, to be the gatekeeper for bugs. Often, they will review the bugs as they are reported, in order to redirect them to the appropriate developer or even another team, to review the priority and severity supplied by the reporter, and sometimes to reject bugs as duplicates or enhancement requests, for example.

To add a component:

1. Click the link **add at least one component** in the message that appears after creating a new product.
2. Enter the **Component** name.
3. Enter a **Description**.
4. Enter a **default assignee**. Use one of the users we've created. Remember to enter the assignee in the form of an email address.
5. Click **Add**.
6. To add more components, click the name of product in the message that reads edit other components of product <**product name**>.

Modifying Default Field Values

Once we begin to enter new bugs, we'll see a number of drop-down lists containing default values. Some of these may work just fine for our product. Others may not. We can modify the values of these fields, adding new values and deleting old ones. Let's take a look at the OS category.

To modify default field values:

1. At the bottom of the page, in the **Edit** section, click **Field Values**.
2. Click the link, in this case **OS**, for the field we want to edit. The OS field contains a list of operating system names. We are going to add browsers to this list. In reality, we might create a custom field instead, but for the sake of this example, just add them to the OS list.
3. Click **Add a value**. In the **Value** field, enter "IE7." Click **Add**.
4. Click **Add a value** again.
5. In the **Value** field, enter "Firefox 3."
6. Click **Add**.
7. Where it reads **Add other values for the op_sys field**, click **op_sys**.
8. This redisplays the table. We should now see the two new entries at the top of the table. These values will also appear in the OS drop-down list when we create a new bug.

Creating a New Bug : Creating bugs is a big part of what Bugzilla does best.

To create a new bug:

1. In the top menu, click **New**.
2. If we've defined more than one component, choose the component from the component list.
3. Select a **Severity** and a **Priority**. **Severity** is self-explanatory, but **Priority** is generally assumed to be the lower the number, the higher the priority. So, a **P1** is the highest priority bug, a *showstopper*.
4. Click the **OS** drop-down list to see the options, including the new browser names we entered.
5. Select one of the options.
6. Enter a summary and a description. We can add any other information of choice, but it is not required by the system, although we may determine that our bug reporting policy requires certain information.
7. Click **Commit**. Bugzilla adds our bug report to the database and displays the detail page for that bug.

Viewing Bug Reports

Eventually, we'll end up with thousands of bugs listed in the system. There are several ways to view the bugs. The easiest is to click the My Bugs link at the bottom of the page. Because we've only got one bug reported, we'll use the standard Search function.

To find a bug:

1. Click **Reports**.
2. Click the **Search** link on the page, not the one in the top menu. This opens a page titled "Find a Specific Bug."
3. Select the **Status**.
4. Select the **Product**.
5. Enter a word that might be in the title of the bug.
6. Click **Search**. If any bugs meet the criteria that we have entered, Bugzilla displays them in a list summary.
7. Click the **ID** number link to view the full bug report.

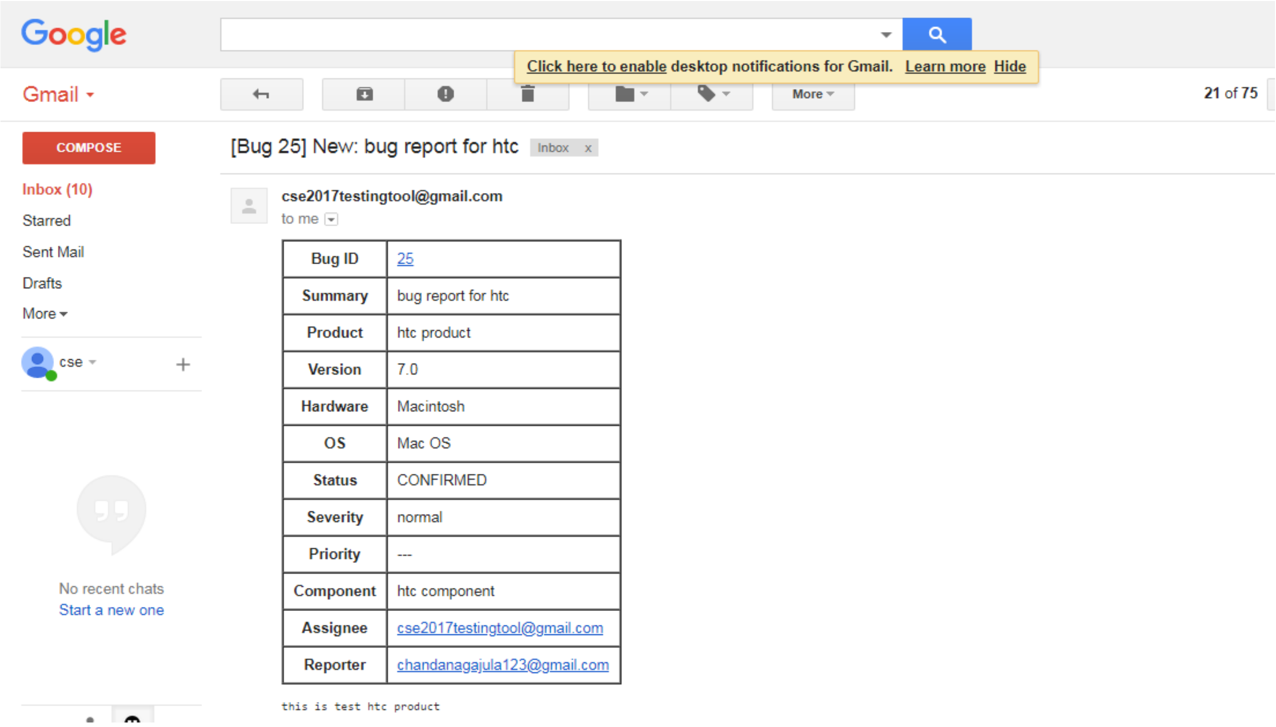
Modifying Bug Reports

Suppose we want to change the status of the bug. We've reviewed it and have determined that it belongs to one of the users we have created earlier

To modify a bug report:

1. Scroll down the full bug description and enter a comment in the **Additional Comments** field.
2. Select "Reassign bug to" and replace the default user ID with one of the other user IDs you created. It must be in the format of an email address.

Screen shot for sample bug report



[Bug 25] New: bug report for htc

cse2017testingtool@gmail.com
to me

Bug ID	25
Summary	bug report for htc
Product	htc product
Version	7.0
Hardware	Macintosh
OS	Mac OS
Status	CONFIRMED
Severity	normal
Priority	---
Component	htc component
Assignee	cse2017testingtool@gmail.com
Reporter	chandanagajula123@gmail.com

```
this is test htc product
```

4.Study of any testing tool (e.g. WinRunner)

WinRunner

Introduction and Features:

- Win runner is the most automated software testing tool.
- It is developed by Mercury Interactive Enterprise.
- Win runner is mainly designed for testing GUI(Graphical User Interface) based applications.
- It is a functionality testing tool for Microsoft windows based applications.
- To automate our manual tests win runner uses TSL Test Script Language.
- It is used to quickly create and run sophisticated automated test on our application.
- This win runner has the ability to record various interactions and transform those into tsl scripts

Understanding the Testing Process

The WinRunner testing process consists of 6 main phases:
1.Learning the objects in our application by Winrunner.

2.Creating additional test scripts that test our application's functionality
WinRunner writes scripts automatically when we record actions in our application, or we can program directly in Mercury Interactive's Test Script Language (TSL).

3.**Debugging the tests :**We debug the tests to check that they operate smoothly and without interruption.

4.Running the tests on a new version of the application : We run the tests on a new version of the application in order to check the application's behavior.

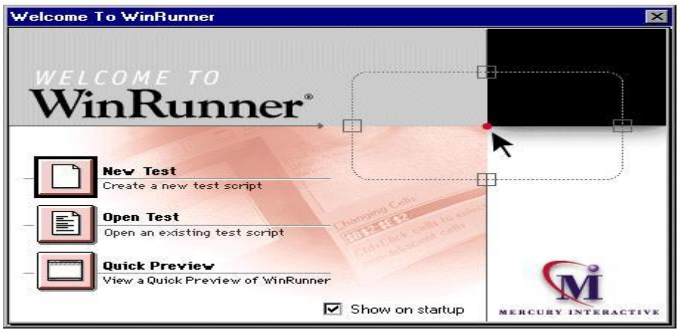
5.Examining the test results : We examine the test results to pinpoint defects in the application.

6.Reporting defects : If we have the TestDirector 7.0i, the Web Defect Manager (TestDirector 6.0), or the Remote Defect Reporter (TestDirector 6.0), we can report any defects to a database. The Web Defect Manager and the Remote Defect Reporter are included in TestDirector, Mercury Interactive's software test management tool.

Exploring the WinRunner Window

Before We begin creating tests, we should familiarize ourself with the WinRunner main window.

To start WinRunner: Choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu. The first time we start WinRunner, the Welcome to WinRunner window opens. From the welcome window we can create a new test, open an existing test, or view an overview of WinRunner in our default browser.



☑ Show on startup

The first time we select one of these options, the WinRunner main screen opens with the “What’s New in WinRunner” section of the help file on top. If we do not want the welcome window to appear the next time we start WinRunner, clear the **Show on startup** check box. Each test we create or run is displayed by WinRunner in a test window. we can open many tests at one time.

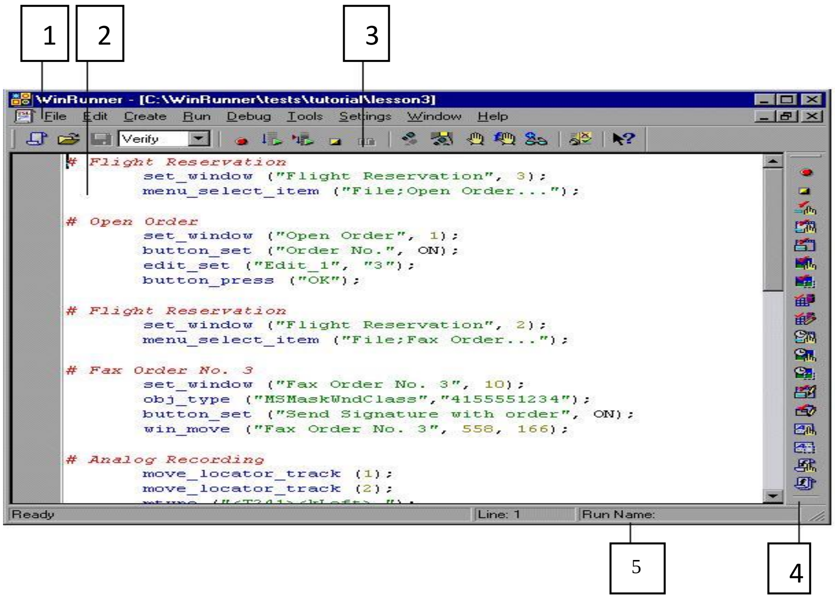
1 *The WinRunner window displays all open tests.*

2 *Each test appears in its own test window. We use this window to record, program, and edit test scripts.*

3 *Buttons on the Standard toolbar help we quickly open, run, and save tests.*

4 *The User toolbar provides easy access to test creation tools.*

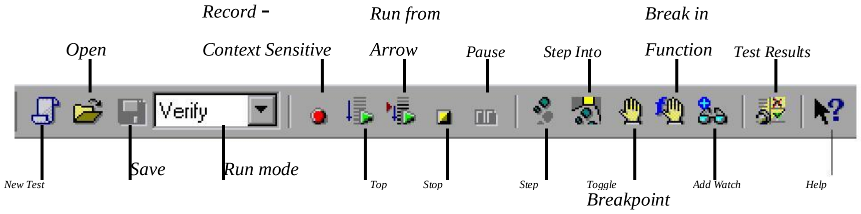
5 *The status bar displays information about selected commands and the current test run.*



```
# Flight Reservation
        set_window ("Flight Reservation", 3);
        menu_select_item ("File;Open Order...");

# Open Order
        set_window ("Open Order", 1);
        button_set ("Order No.", ON);
        edit_set ("Edit_1", "3");
        button_press ("OK");

# Flight Reservation
        set_window ("Flight Reservation", 2);
        menu_select_item ("File;Fax Order...");

# Fax Order No. 3
        set_window ("Fax Order No. 3", 10);
        obj_type ("MSMaskWndClass","4155551234");
        button_set ("Send Signature with order", ON);
        win_move ("Fax Order No. 3", 558, 166);

# Analog Recording
        move_locator_track (1);
        move_locator_track (2);
```

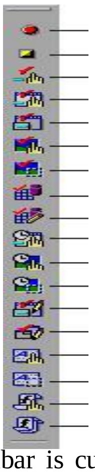
1
2
3
WinRunner - [C:\WinRunner\tests\tutorial\lesson3]
File Edit Create Run Debug Tools Settings Window Help
Verify
Ready
Line: 1
Run Name:
5
4

- ➢ The *Standard toolbar* provides easy access to frequently performed tasks, such as opening, executing, and saving tests, and viewing test results.



- ➢ The *User toolbar* displays the tools we frequently use to create test scripts. By default, the User toolbar is hidden.

- ➢ To display the User toolbar choose **Window** > **User Toolbar**. When we create tests, we can minimize the WinRunner window and work exclusively from the toolbar.



Record - Context Sensitive
Stop
GUI Checkpoint for Single Property
GUI Checkpoint for Object/Window
GUI Checkpoint for Multiple Objects
Bitmap Checkpoint for Object/Window
Bitmap Checkpoint for Screen Area
Default Database Checkpoint
Custom Database Checkpoint
Synchronization Point for Object/Window PropertSynchronization Point for Object/Window Bitmap
Synchronization Point for Screen Area Bitmap
Edit GUI Checklist
Edit Database Checklist
Get Text from Object/Window
Get Text from Screen Area
Insert Function for Object/Window
Insert Function from Function Generator

- ➢ The User toolbar is customizable. We choose to add or remove buttons using the **Settings** > **Customize User Toolbar** menu option. When we re-open WinRunner, the User toolbar appears as it was when we last closed it.

Experiment 1. Setting Up the GUI Map

GUI applications are made up of GUI objects such as windows, buttons, lists, and menus. When WinRunner learns the description of a GUI object, it looks at the object's physical *properties*. Each GUI object has many physical properties such as "class," "label," "width," "height", "handle," and "enabled" to name a few. WinRunner, however, only learns the properties that uniquely distinguish an object from all other objects in the application. For example, when WinRunner looks at an OK button, it might recognize that the button is located in an Open window, belongs to the pushbutton object class, and has the text label "OK."

Spying on GUI Objects

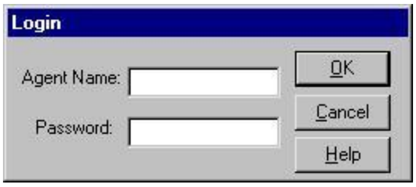
To help us understand how WinRunner identifies GUI objects, examine the objects in the sample Flight Reservation application.

1. Start the Flight Reservation application.

Flight 1A

Choose **Programs** > **WinRunner** > **Sample Applications** > **Flight 1A** on the **Start** menu.

The Login window opens.

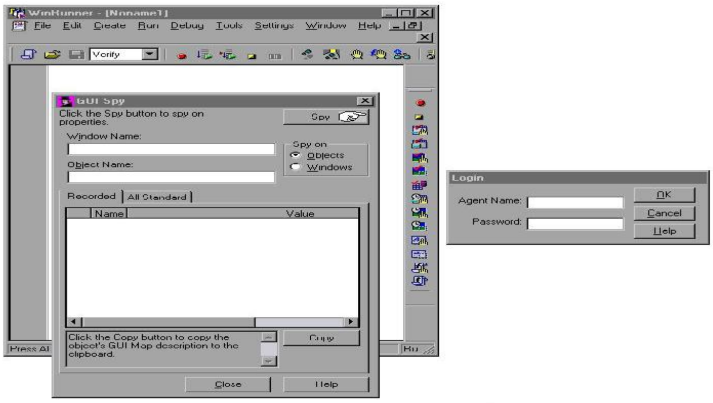


2. Start WinRunner.

Choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu. In the Welcome window, click the **New Test** button. If the Welcome window does not open, choose **File** > **New**.

3. Open the GUI Spy. This tool lets us to "spy" on the properties of GUI objects.

Choose **Tools** > **GUI Spy**. The GUI Spy opens. Position the GUI Spy on the desktop so that both the Login window and the GUI Spy are clearly visible.

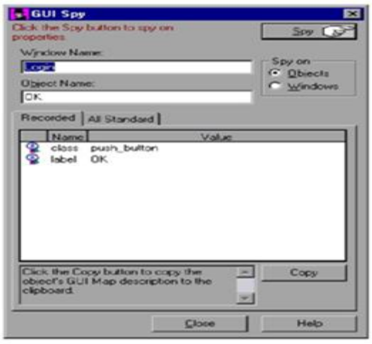


Click the Spy button to spy on properties.

Click the Copy button to copy the object's GUI Map description to the clipboard.

4. View the properties that provide a unique description of the OK button.

In the GUI Spy, click the **Spy** button. Move the pointer over objects in the Login window. Notice that each object flashes as we move the pointer over it, and the GUI Spy displays its properties. Place the pointer over the **OK** button and press **Left Ctrl** + **F3**. This freezes the OK button's description in the GUI Spy.



Click the Spy button to spy on properties

Window Name: Login

Object Name: OK

Name	Value
class	push_button
label	OK

Click the Copy button to copy the object's GUI Map description to the clipboard.

5. Examine the properties of the OK button.

At the top of the dialog box, the GUI Spy displays the name of the window in which the object is located and the object's logical name. In the Recorded tab, the property names and values that would be recorded are listed. For example, "label OK" indicates that the button has the text label "OK", and "class push_button" indicates that the button belongs to the push button object class.

As we can see, WinRunner needs only a few properties to uniquely identify the object.

6. Take a few minutes to view the properties of other GUI objects in the Login window.

Click the **Spy** button and move the pointer over other GUI objects in the Login window. If we would like to view an expanded list of properties for each object, press **Left Ctrl** + **F3** to stop the current Spy, and then click the **All Standard** tab.

7. Exit the GUI Spy.

Click **Close**.

Experiment 2a.Choosing a GUI Map Mode

Before we start teaching WinRunner the GUI of an application, we should consider whether we want to organize our GUI map files in the *GUI Map File per Test* mode or the *Global GUI Map File* mode.

The GUI Map File per Test Mode

In the *GUI Map File per Test* mode, WinRunner automatically creates a new GUI map file for every new test we create. WinRunner automatically saves and opens the GUI map file that corresponds to our test.If we are new to WinRunner or to testing, we may want to consider working in the *GUI Map File per Test* mode. In this mode, a GUI map file is created automatically every time we create a new test. The GUI map file that corresponds to our test is automatically saved whenever we save our test and automatically loaded whenever we open our test. This is the simplest mode for inexperienced testers and for ensuring that updated GUI Map files are saved and loaded.

The Global GUI Map File Mode

In the Global GUI Map File mode, we can use a single GUI map for a group of tests. When we work in the Global *GUI Map* File mode, we need to save the information that WinRunner learns about the properties into a GUI map file. When we run a test, we must load the appropriate GUI map file.

If we are familiar with WinRunner or with testing, it is probably most efficient to work in the Global *GUI* Map File mode.

Setting Our Preferred GUI Map File Mode

By default, WinRunner is set to the Global *GUI* Map File mode. To change the mode to the *GUI* Map File per Test mode choose **Settings > General Options,** click the **Environment** tab, and select **GUI Map File per Test.** Click **OK** to close the dialog box.

Note: If we change the GUI Map File mode, we must restart WinRunner for the changes to take effect.

Getting Started

The remaining sections can be performed only in the Global *GUI* Map File mode. If we choose to work in the *GUI* Map File per Test mode, change the mode setting as described above.

If we choose to work in the Globa*l GUI* Map File mode, proceed to the section below on Using the RapidTest Script Wizard.

Experiment 2b.Using the RapidTest Script Wizard

If we choose the *Global GUI Map File* mode, the RapidTest Script wizard is usually the easiest and quickest way to start the testing process.

Note: The RapidTest Script wizard is not available when we work in *GUI MapFile per Test* mode.

The RapidTest Script wizard systematically opens the windows in our application and learns the description of every GUI object. The wizard stores this information in a GUI map file. To observe WinRunner's learning process, use the RapidTest Script wizard on the Flight Reservation application.

Note: The RapidTest Script wizard is not available when the Terminal Emulator, the WebTest or the Java add-in is loaded..

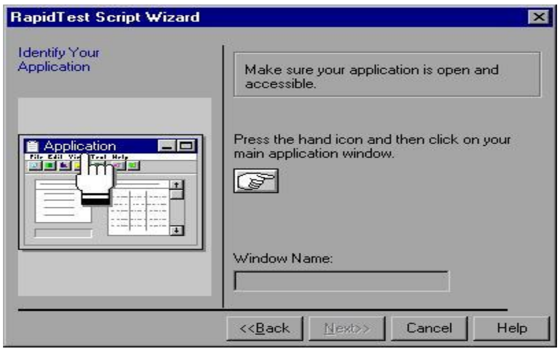
1. Log in to the Flight Reservation application : If the Login window is open, type our name in the **Agent Name** field, and mercury in the **Password** field and click **OK**. The name we type must be at least four characters long. If the Login window is not already open on our desktop, choose **Programs WinRunner** > **Sample Applications** > **Flight 1A** on the **Start** menu and then log in, as described in the previous paragraph.

2. Start WinRunner : If WinRunner is not already open, choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu.

3. Open a new test. If the Welcome window is open, click the **New Test** button. Otherwise, choose **File - New**. A new test window opens in WinRunner

.

4. Start the RapidTest Script wizard.
Choose **Create** > **RapidTest Script Wizard**. Click **Next** in the wizard's welcome screen to advance to the next screen.



RapidTest Script Wizard

Identify Your Application

Make sure your application is open and accessible.

Press the hand icon and then click on your main application window.

Window Name:

5. Point to the application we want to test.
Click the ☞ button and then click anywhere in the **Flight Reservation** application. The application's window name appears in the wizard's Window Name box. Click **Next**.

6. Make sure that all the check boxes are cleared.
For the purposes of this exercise, confirm that all the check boxes are cleared. We will use the wizard only to learn the GUI of the Flight Reservation application. Click **Next**.
Note: A regression test is performed when the tester wishes to see the progress of the testing process by performing identical tests before and after a bug has been fixed. A regression test allows the tester to compare expected test results with the actual results.

7. Accept the default navigation controls.
Navigation controls tell WinRunner which GUI objects are used to open windows. The Flight Reservation application uses the default navigation controls (... and > >) so we do not need to define additional controls. Click **Next**.

8. Set the learning flow to "Express."

The learning flow determines how WinRunner walks through our application. Two modes are available: *Express* and *Comprehensive*. Comprehensive mode lets We customize how the wizard learns GUI object descriptions. First-time WinRunner users should use Express mode.

Click the **Learn** button. The wizard begins walking through the application, pulling down menus, opening windows, and learning object descriptions. This process takes a few minutes.

If a pop-up message notifies us that an interface element is disabled, click the **Continue** button in the message box.

If the wizard cannot close a window, it will ask us to show it how to close the window. Follow the directions on the screen.

9. Accept "No" in the Start Application screen.

We can choose to have WinRunner automatically open the Flight Reservation application each time We start WinRunner. Accept the default "**No**." Click **Next**.

10. Save the GUI information and a startup script.

The wizard saves the GUI information in a *GUI map file*.

The wizard also creates a startup script. This script runs automatically each time we start WinRunner. It contains a command which loads the GUI map file so that WinRunner will be ready to test our application.

Accept the default paths and file names or define different ones. Make sure that we have write permission for the selected folders. Click **Next**.

11. Click OK in the Congratulations screen : The information WinRunner learned about the application is stored in a GUI map file.

Experiment 3.Recording Tests

Choosing a Record Mode

By recording, we can quickly create automated test scripts. We work with our application as usual, clicking objects with the mouse and entering keyboard input.WinRunner records our operations and generates statements in TSL, Mercury Interactive's Test Script Language. These statements appear as a script in a WinRunner test window. Before we begin recording a test, we should plan the main stages of the test and select the appropriate record mode. Two record modes are available: Context Sensitive and Analog.

Context Sensitive

Context Sensitive mode records our operations in terms of the GUI objects in our application. WinRunner identifies each object we click (such as a window, menu, list, or button), and the type of operation we perform (such as press, enable, move, or select). For example, if we record a mouse click on the **OK** button in the Flight Reservation Login window, WinRunner records the following TSL statement in our test script: button_press ("OK");

When we run the script, WinRunner reads the command, looks for the **OK** button, and presses it.

Analog

In Analog mode, WinRunner records the exact coordinates traveled by the mouse, as well as mouse clicks and keyboard input. For example, if we click the **OK** button in the Login window, WinRunner records statements that look like this:

When this statement is recorded... .... it really means:

move_locator_track (1);	*mouse track*
mtype ("<T110><kLeft>-");	*left mouse button press*
mtype ("<kLeft>+");	*left mouse button release*

When we run the test, WinRunner retraces the recorded movements using absolute screen coordinates. If our application is located in a different position on the desktop, or the user interface has changed, WinRunner is not able to execute the test correctly.

Note: We should record in Analog mode only when exact mouse movements are an important part of our test, for example, when recreating a drawing.

When choosing a record mode, consider the following points:

Choose Context Sensitive if...	Choose Analog if...
The application contains GUI objects.	The application contains bitmap areas (such as a drawing area).
Exact mouse movements are not required.	Exact mouse movements are required.
we plan to reuse the test in different versions of the application.	

If we are testing an application that contains both GUI objects and bitmap areas, we can switch between modes as we record.

Recording a Context Sensitive Test

In this exercise we will create a script that tests the process of opening an order in the Flight Reservation application. We will create the script by recording in Context Sensitive mode.

1.Start WinRunner.

If WinRunner is not already open, choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu.

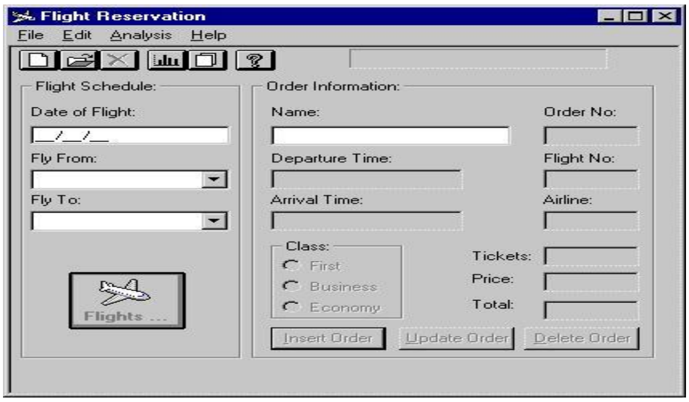
2. Open a new test.

If the Welcome window is open, click the **New Test** button. Otherwise, choose **File** > **New**. A new test window opens in WinRunner.

Flight 1A

3. Start the Flight Reservation application and log in.

Choose **Programs** > **WinRunner** > **Sample Applications** > **Flight 1A** on the **Start** menu. In the Login window, type our name and the password mercury, and click **OK**. The name we type must be at least four characters long. Position the Flight Reservation application and WinRunner so that they are both clearly visible on our desktop.



4. Start recording in Context Sensitive mode: In WinRunner, choose **Create** > **Record— Context Sensitive** or click the **Record** button on the toolbar. From this point on, WinRunner records all mouse clicks and keyboard input. Note that the text, "Rec" appears in blue above the recording button. This indicates that we are recording in Context Sensitive mode. The status bar also informs us of our current recording mode.

5. **Open order #3:** In the Flight Reservation application, choose **File** > **Open Order**. In the Open Order dialog box, select the **Order No.** check box. Type 3 in the adjacent box, and click **OK**.

6. Stop recording: In WinRunner, choose **Create** > **Stop Recording** or click the **Stop** button on the toolbar.

7. Save the test : Choose **File** > **Save** or click the **Save** button on the toolbar. Click **Save** to close the Save Test dialog box.Note that WinRunner saves the test in the file system as a folder, and not as an individual file. This folder contains the test script and the results that are generated when we run the test.

Note: If we are working in the *GUI Map File per Test* mode, GUI map is saved automatically with our test. If we are working in the *Global GUI Map File* mode, we must save the GUI map before we close WinRunner.

Understanding the Test Script

In the previous exercise, we recorded the process of opening a flight order in the Flight Reservation application. As we worked, WinRunner generated a test script similiar to the following:

```
# Flight Reservation
set_window ("Flight Reservation", 3);
menu_select_item ("File;Open Order...");
# Open Order
set_window ("Open Order", 1);
button_set ("Order No.", ON);
edit_set ("Edit_1", "3");
button_press ("OK");
```

As we can see, the recorded TSL statements describe the objects we selected and the actions we performed. For example, when we selected a menu item, WinRunner generated a **menu_select_item** statement.

The following points will help usto understand our test script:

- When we click an object, WinRunner assigns the object a *logical name*, which is usually the object's text label. The logical name makes it easy for us to read the test script. For example, when we selected the Order No. check box, WinRunner recorded the following statement:
  button_set ("Order No.", ON);
  "Order No." is the object's logical name.
- By default, WinRunner automatically adds a comment line each time we begin working in a new

window so that our script is easier to read. For example, when we clicked on the Flight Reservation window, WinRunner generated the following comment line: *# Flight Reservation*

- WinRunner generates a **set_window** statement each time we begin working in a new window. The statements following a **set_window** statement perform operations on objects within that window. For example, when we opened the Open Order dialog box, WinRunner generated the following statement: set_window ("Open Order", 10);
- When we enter keyboard input, WinRunner generates a **type**, an **obj_type**, or an **edit_set** statement in the test script. For example, when we typed 3 in the Order Number box, WinRunner generated the following statement:

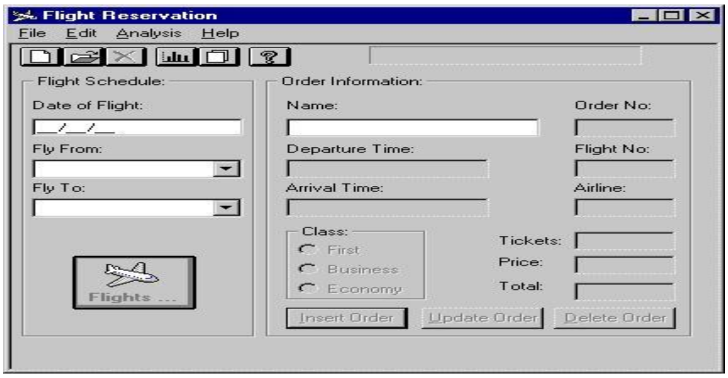
  edit_set ("Edit", "3");
  For more information about the different ways in which WinRunner records keyboard input,choose **Help** > **TSL Online Reference**.

Recording in Analog Mode

In this we are going to record the application in analog mode which contains the same steps as in context sensitive but the recording modes are changes and the generated tsl scripts also changed in analog mode

1.Start WinRunner.
If WinRunner is not already open, choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu.
2.Open a new test.
If the Welcome window is open, click the **New Test** button. Otherwise, choose **File** > **New**. A new test window opens in WinRunner.
3.Start the Flight Reservation application and log in.

Choose **Programs** > **WinRunner** > **Sample Applications** > **Flight 1A** on the **Start** menu.
In the Login window, type our name and the password mercury, and click **OK**. The name we type must be at least four characters long. Position the Flight Reservation application and WinRunner so that they are both clearly visible on our desktop.



4 Start recording in Analog mode: In WinRunner, choose **Create** > **Record— Analog mode** . From this point on, WinRunner records all mouse clicks and mouse movements and keyboard inputs. The status bar informs us of our current recording mode.

5. Open order #3: In the Flight Reservation application, choose **File** > **Open Order**. In the Open Order dialog box, select the **Order No.** check box. Type 3 in the adjacent box, and click **OK**.

6. Stop recording: In WinRunner, choose **Create** > **Stop Recording** or click the **Stop** button on the toolbar.

7 Save the test : Choose **File** > **Save** or click the **Save** button on the toolbar. Click **Save** to close the Save Test dialog box.Note that WinRunner saves the test in the file system as a folder, and not as an individual file. This folder contains the test script and the results that are generated when we run the test.

Note: If we are working in the *GUI Map File per Test* mode, GUI map is saved a automatically with our test. If we are working in the *Global GUI Map File* mode, we must save the GUI map before we close WinRunner.

Running the Test

We are now ready to run our recorded test script and to analyze the test results. WinRunner provides three modes for running tests. We select a mode from the toolbar.

- Use *Verify mode* when running a test to check the behavior of our application, and when we want to save the test results.
- Use *Debug mode* when we want to check that the test script runs smoothly without errors in syntax.
- Use *Update mode* when we want to create new expected results for a GUI checkpoint or bitmap checkpoint.

To run the test:

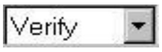
1. Check that WinRunner and the main window of the Flight Reservation application are open on our desktop.

2. Make sure that the test window is active in WinRunner.

Click the title bar of the test window. If the test is not already open, choose **File > Open** and select the test.

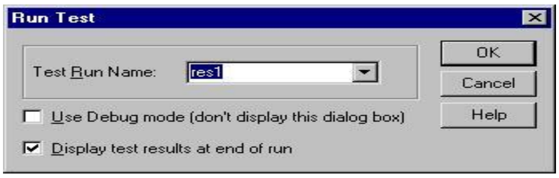
3. Make sure the main window of the Flight Reservation application is active.

If any dialog boxes are open, close them.



4. Make sure that Verify mode is selected in the toolbar.

5. Choose Run from Top: Choose **Run** > **Run from Top** or click the **Run from Top** button. The **Run Test** dialog box opens.


Run Test

Test Run Name: res1

□ Use Debug mode (don't display this dialog box)

☑ Display test results at end of run

6. Choose a Test Run name.

Define the name of the folder in which WinRunner will store the results of the test. Accept the default folder name “res1.” The results folder will be stored within the test’s folder.

Note:The **Display Test Results at end of run** check box at the bottom of the dialog box. When this check box is selected, WinRunner automatically displays the test results when the test run is completed. Make sure that this check box is selected.

7.Run the test. Click **OK** in the Run Test dialog box. WinRunner immediately begins running the test.

8.Review the test results.

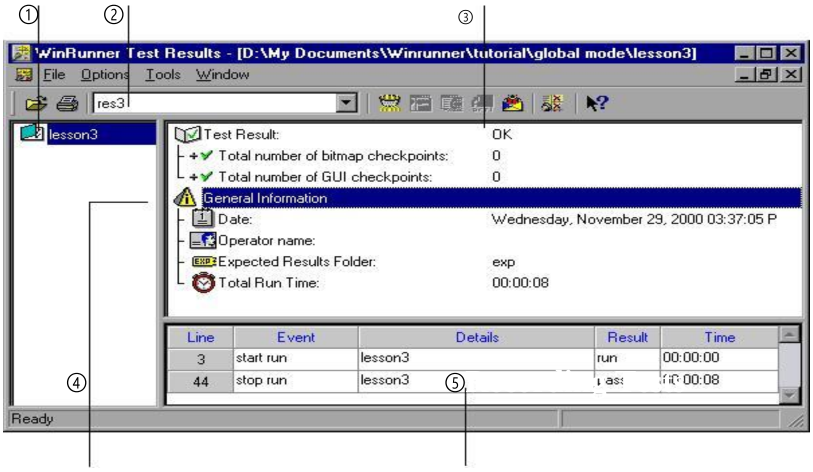
When the test run is completed, the test results automatically appear in the WinRunner Test Results window. See the next section to learn how to analyze the test results.

Analyzing Test Results

Once a test run is completed, we can immediately review the test results in the WinRunner Test Results window. WinRunner color-codes results (green indicates passed and red indicates failed) so that we can quickly draw conclusions about the success or failure of the test.

1. Make sure that the WinRunner Test Results window is open and displays the test results.

If the WinRunner Test Results window is not currently open, first click the test window to activate it, and then choose **Tools** > **Test Results** or click the **Test Results** button.


①
②
③
④
⑤
WinRunner Test Results - [D:\My Documents\Winrunner\tutorial\global mode\lesson3]
File
Options
Tools
Window
Ready

res3

lesson3

Test Result: OK
Total number of bitmap checkpoints: 0
Total number of GUI checkpoints: 0
General Information
Date: Wednesday, November 29, 2000 03:37:05 P
Operator name:
Expected Results Folder: exp
Total Run Time: 00:00:08

Line	Event	Details	Result	Time
3	start run	lesson3	run	00:00:00
44	stop run	lesson3	[illegible]	00:00:08

1 Displays the name of the current test.

2 Shows the current results directory name.

3 Shows whether a test run passed or failed.

4 Includes general information about the test run such as date, operator name, and total run time. To view these details, double click the Information icon.

5 The test log section lists the major events that occurred during the test run. It also lists the test script line at which each event occurred.

2. Review the results.

3. Close the Test Results window.

Choose **File** > **Exit** in the WinRunner Test Results window.

4. Close the test. Choose **File** > **lose**.

5. Close the Flight Reservation application. Choose **File** > **Exit**.

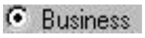
Experiment 4 Checking GUI Objects

When working with an application, we can determine whether it is functioning properly according to the behavior of its GUI objects. If a GUI object does not respond to input as expected, a defect probably exists somewhere in the application's code.

We check GUI objects by creating *GUI checkpoints*. A GUI checkpoint examines the behavior of an object's properties. For example, we can check:

Justine

the content of a field


○ Business

whether a radio button is on or off

Insert Order

whether a pushbutton is enabled or disabled

a. GUI Checkpoints For Single Property

In this exercise we will check the properties of single object in the Calculator application.

1.Start WinRunner and open a new test.

If WinRunner is not already open, choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu. If the Welcome window is open, click the **New Test** button. Otherwise, choose **File** > **New**. A new test window opens.

2. Start the Calculator application.

Choose**All Programs** > **Accessories** > **Calculator** on the **Start** menu. Now Calculator and WinRunner both are clearly visible on our desktop.

3. Start recording in Context Sensitive mode.

Choose **Create** > **Record—Context Sensitive** or click the **Record** button on the toolbar.Now click on the Calculator application window anywhere to activate the window

4. Create a GUI checkpoint for the single object i.e, button CE in Calculator application.

Choose **Create** > **GUI Checkpoint** > **For Single Object**, or click the **GUI Checkpoint for Single Object** button on the User toolbar.The checkpoint appears as an **obj_check_gui** statement.

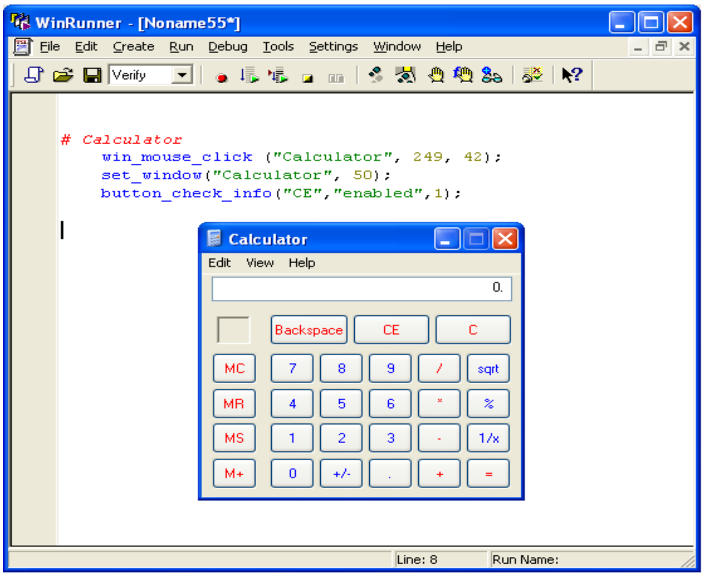
5. Place the ☝ on CE button and click on that button in the calculator application.

This check captures the current properties of the CE button and stores it as expected results..If we want to check the other properties then we can select the property which we want to check, from the gui property window dropdown and click on paste to copy the content to our tsl script window.

6. Stop recording. Choose **Create** > **Stop Recording** or click the **Stop** button.

7. Save the test.

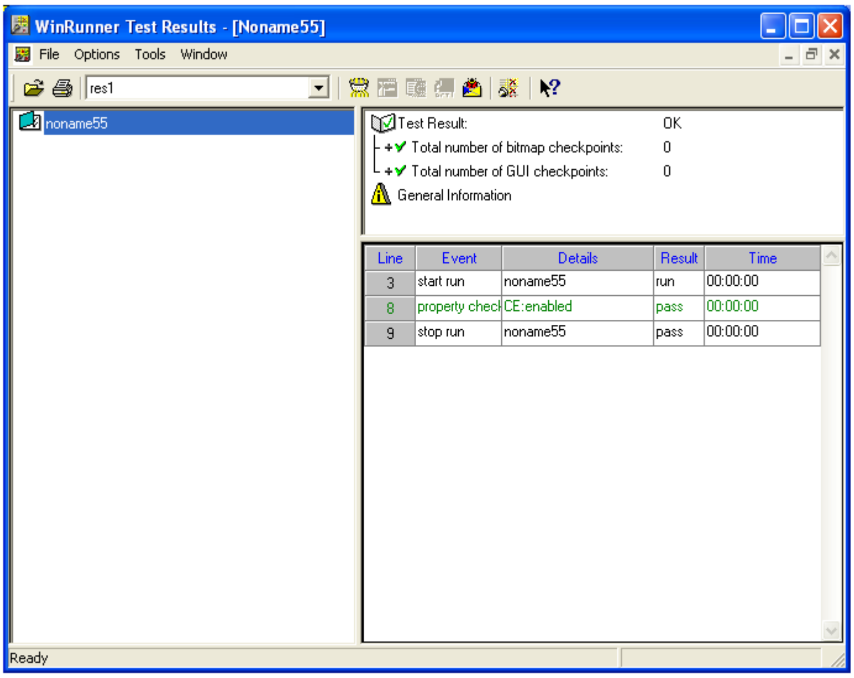
Choose **File** > **Save** or click the **Save** button. Save the test in a convenient location on our hard drive. Click **Save** to close the Save Test dialog box.



```
# Calculator
    win_mouse_click ("Calculator", 249, 42);
    set_window("Calculator", 50);
    button_check_info("CE","enabled",1);
```

Line: 8 Run Name:

Fig.GUI checkpoint for Single Property Test Script Window


WinRunner Test Results - [Noname55]
File Options Tools Window
res1
noname55
Ready

Test Result: OK
- Total number of bitmap checkpoints: 0
- Total number of GUI checkpoints: 0

General Information

Line	Event	Details	Result	Time
3	start run	noname55	run	00:00:00
8	property check	CE:enabled	pass	00:00:00
9	stop run	noname55	pass	00:00:00

Fig.GUI Check Point for Single Property Test Results Window

b. GUI Checkpoints For Multiple objects

In this exercise we will check the properties of multiple object in the Calculator application.

1. Start WinRunner and open a new test.

If WinRunner is not already open, choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu. If the Welcome window is open, click the **New Test** button. Otherwise, choose **File** > **New**. A new test window opens.

2. Start the Calculator application.

ChooseAl**l** **Programs** > **Accessories** > **Calculator** on the **Start** menu. Now Calculator and WinRunner both are clearly visible on our desktop.

3. Start recording in Context Sensitive mode.

Choose **Create** > **Record—Context Sensitive** or click the **Record** button on the toolbar.Now click on the Calculator application window anywhere to activate the window.

4. Create a GUI checkpoint for the multiple object i.e, button CE and button 7,8,9 in Calculator application.

Choose **Create** > **GUI Checkpoint** > **For Multiple Objects**, or click the **GUI Checkpoint for Multiple Object/Window** button on the User toolbar.

The checkpoint appears as an **obj_check_gui** statement.

5 Now Click on Add in the window then Place the ☝ on these CE,7,8,9 buttons and click on those buttons in the calculator application.

This check captures all the objects whatever we want, just we have to click on the required objects and shows all current properties of these buttons and stores expected results.If we want to check the specific properties then we can select the properties by clicking the checkboxes, from the gui property window .

6. Stop recording. Choose **Create** > **Stop Recording** or click the **Stop** button.

7. Save the test.

Choose **File** > **Save** or click the **Save** button. Save the test in a convenient location on our hard drive. Click **Save** to close the Saved Test .

Running the Test

1. Make sure that the Flight Reservation application is open on our desktop.
2. In WinRunner, check that Verify mode is selected in the Standard toolbar.
3. Choose Run from Top.

Choose **Run** > **Run from Top**, or click the **Run from Top** button. The **Run Test** dialog box opens. Accept the default test run name “res1.” Make sure that the

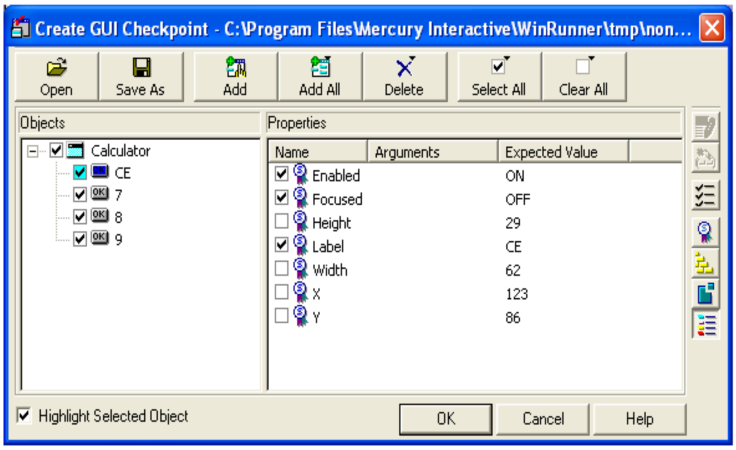
4. Display test results at end of run check box is selected.
5. Run the test. Click **OK** in the Run Test dialog box.
6. Review the results.

When the test run is completed, the test results appear in the WinRunner Test Results window. In the test log section all “end GUI checkpoint” events should appear in green (indicating success). Double-click an end GUI checkpoint event to view detailed results of that GUI checkpoint.

7. Close the test results.

Click **OK** to close the GUI Checkpoint Results dialog box. Then choose **File** > **Exit** to close the Test Results window.

8. Close the Calculator application. Choose **File** > **Exit**.

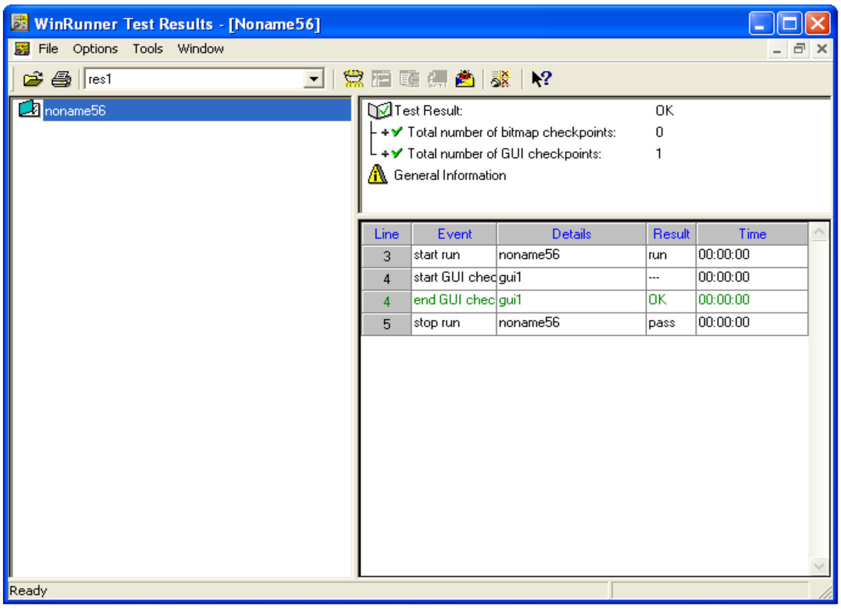

Create GUI Checkpoint - C:\Program Files\Mercury Interactive\WinRunner\tmp\non...
Open
Save As
Add
Add All
Delete
Select All
Clear All
Objects
Properties
OK
Cancel
Help

- ☑ Calculator
  - ☑ CE
  - ☑ 7
  - ☑ 8
  - ☑ 9

Name	Arguments	Expected Value
☑ Enabled		ON
☑ Focused		OFF
□ Height		29
☑ Label		CE
□ Width		62
□ X		123
□ Y		86

☑ Highlight Selected Object

Fig.GUI CheckPoint for Multiple Objects Properties Window


WinRunner Test Results - [Noname56]
File Options Tools Window
res1
noname56
Ready

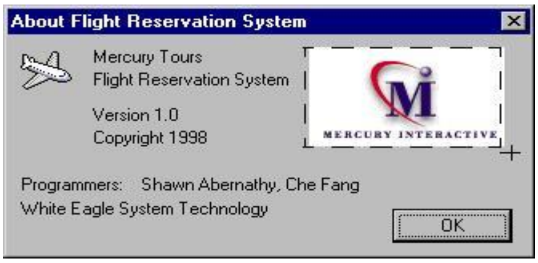
Test Result: OK
Total number of bitmap checkpoints: 0
Total number of GUI checkpoints: 1
General Information

Line	Event	Details	Result	Time
3	start run	noname56	run	00:00:00
4	start GUI chec	gui1	---	00:00:00
4	end GUI chec	gui1	OK	00:00:00
5	stop run	noname56	pass	00:00:00

Fig .GUI Check Point For Multiple Objects Test Result Window

Experiment5. Checking Bitmaps

If our application contains bitmap areas, such as drawings or graphs, we can check these areas using a *bitmap checkpoint*. A bitmap checkpoint compares captured bitmap images pixel by pixel.
To create a bitmap checkpoint, we indicate an area, window, or object that we want to check. For example:



About Flight Reservation System

Mercury Tours
Flight Reservation System

Version 1.0
Copyright 1998

MERCURY INTERACTIVE

Programmers: Shawn Abernathy, Che Fang
White Eagle System Technology

WinRunner captures a bitmap image and saves it as *expected* results. It then inserts an **obj_check_bitmap** statement into the test script if it captures an object, or a **win_check_bitmap** statement if it captures an area or window.

When we run the test on a new version of the application, WinRunner compares the expected bitmap with the actual bitmap in the application. If any differences are detected, we can view a picture of the differences from the Test Results window.

Adding Bitmap Checkpoints to a Test Script

In this exercise we will test the Agent Signature box in the Fax Order dialog box. We will use a bitmap checkpoint to check that we can sign our name in the box. Then we will use another bitmap checkpoint to check that the box clears when we click the Clear Signature button.

1.Start WinRunner and open a new test.

If WinRunner is not already open, choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu. If the Welcome window is open, click the **New Test** button. Otherwise, choose **File** > **New**. A new test window opens.

2. Start the Paint application.

Choose **All Programs** >**Accessories** >**Paint** on the **Start** menu.Now Paint application and winrunner window both are clearly visible on our desktop.

3. Now Draw any image in the paint window and save the image.

4. Start recording in Context Sensitive mode.

Choose **Create** > **Record—Context Sensitive** or click the **Record** button on the toolbar.

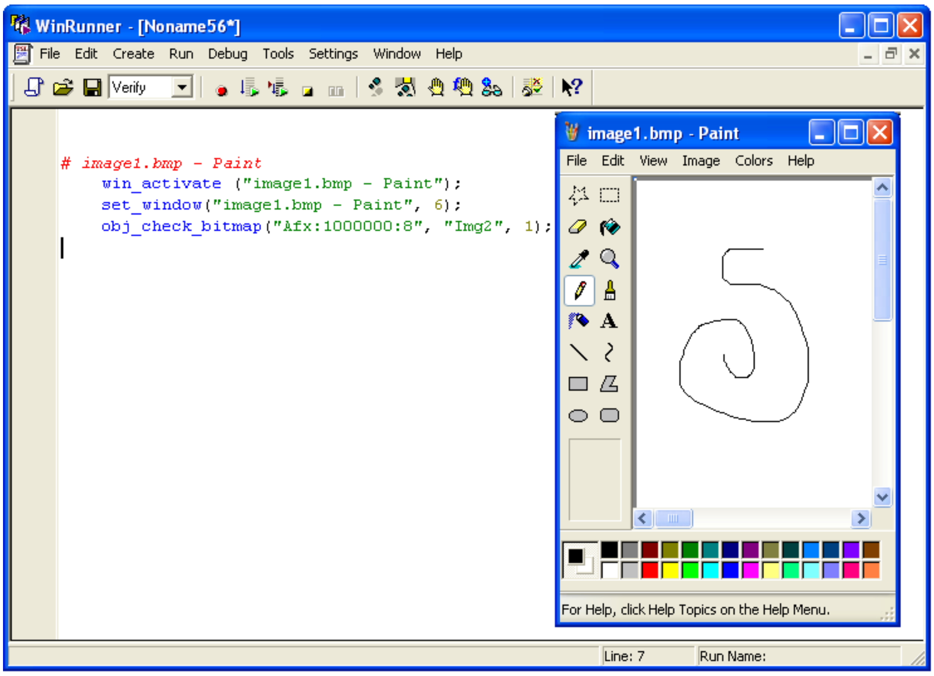
5. Click on the Paint window anywhere to make the window activate.

6. Go to winrunner window, Insert a bitmap checkpoint that checks our image.

Choose **Create** > **Bitmap Checkpoint** > **For Object/Window** or click the **Bitmap Checkpoint for Object/Window** button on the User toolbar. Use the ☝ pointer to click the image on the paint window. WinRunner captures the bitmap and inserts an **obj_check_bitmap** statement into the test script.

7. Stop recording. Choose **Create** > **Stop Recording** or click the **Stop** button.

8. Save the test. Choose **File** > **Save** or click the **Save** button. Save the test in a convenient location on our hard drive. Click **Save** to close the Save Test dialog box.


WinRunner - [Noname56*]
File Edit Create Run Debug Tools Settings Window Help
Verify
image1.bmp - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.
Line: 7
Run Name:

```
# image1.bmp - Paint
    win_activate ("image1.bmp - Paint");
    set_window("image1.bmp - Paint", 6);
    obj_check_bitmap("Afx:1000000:8", "Img2", 1);
```

Fig.Bit map check point for Object/Window

Viewing Expected Results

We can now view the expected results of the test.

1. **Open the WinRunner Test Results window:** Choose **Tools** > **Test Results** or click the **Test Results** button. The Test Results window opens.

2. **View the captured bitmaps.** In the test log section, double-click the first "capture bitmap" event, or select it and click the **Display** button

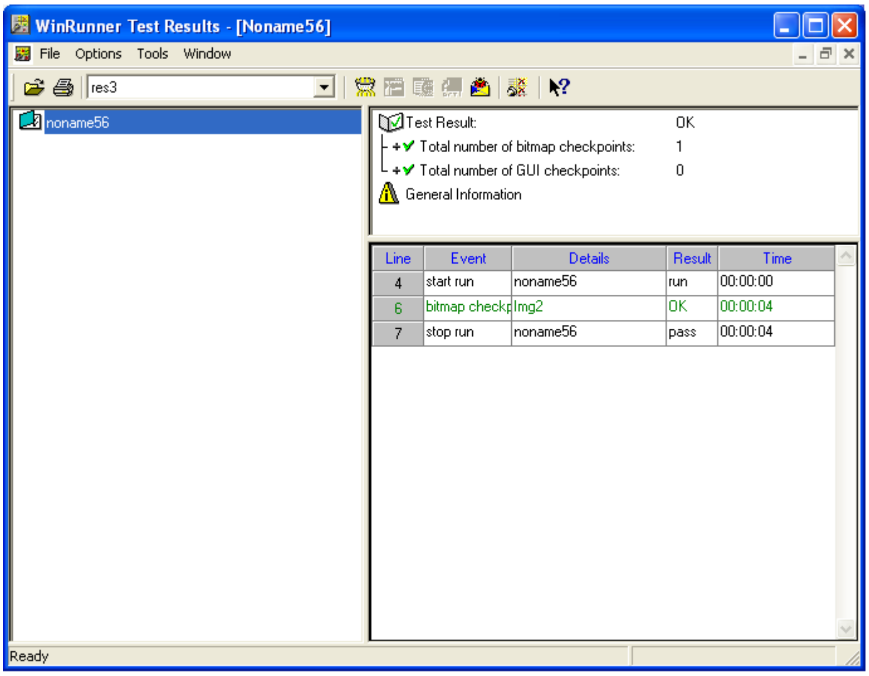
3. Close the Test Results window.

Close the bitmaps and choose **File** > **Exit** to close the Test Results window.

4. Close the Test Results window. Choose **File** > **Exit** to close the Test Results window.

5. Close the test.

Choose **File** > **Close**. Choose paintwindow exit.File>Exit



Test Result: OK

Total number of bitmap checkpoints: 1

Total number of GUI checkpoints: 0

General Information

Line	Event	Details	Result	Time
4	start run	noname56	run	00:00:00
6	bitmap checkp	Img2	OK	00:00:04
7	stop run	noname56	pass	00:00:04

Fig.Bitmap checkpoint Test Results Window

Experiment 6. Synchronization Test

When we run tests, our application may not always respond to input with the same speed. For example, it might take a few seconds:

· To retrieve information from a database

· For a window to pop up

· For a progress bar to reach 100%

· For a status message to appear

- WinRunner waits a set time interval for an application to respond to input. The default wait interval is up to 10 seconds. If the application responds slowly during a test run, WinRunner’s default wait time may not be sufficient, and the test run may unexpectedly fail.
- If we discover a synchronization problem between the test and our application,we can either:
- Increase the default time that WinRunner waits. To do so, we change the value of the Timeout for Checkpoints and CS Statements option in the Run tab of the General Options dialog box (Settings > General Options). This method affects all our tests and slows down many other Context Sensitive operations.
- Insert a synchronization point into the test script at the exact point where the problem occurs. A synchronization point tells WinRunner to pause the test run in order to wait for a specified response in the application. This is the recommended method for synchronizing a test with our application. In the following exercises we will:
- Create a test that opens a new order in the Flight Reservation application and inserts the order into the database
- Change the synchronization settings
- Identify a synchronization problem
- Synchronize the test
- Run the synchronized test

Creating a Test

In this first exercise we will create a test that opens a new order in the Flight Reservation application and inserts the order into a database.

1 Start WinRunner and open a new test.

If WinRunner is not already open, choose Programs > WinRunner > WinRunner on the Start menu. If the Welcome window is open, click the New Test button. Otherwise, choose File > New. A new test window opens.

2 Start the Flight Reservation application and log in.

Choose Programs > WinRunner > Sample Applications > Flight 1A on the Start menu. In the Login window, type our name and the password mercury, and click OK. Reposition the Flight Reservation application and WinRunner so that they are both clearly visible on our desktop.

3 Start recording in Context Sensitive mode.

Choose Create > Record—Context Sensitive or click the Record button on the toolbar.WinRunner will start recording the test.

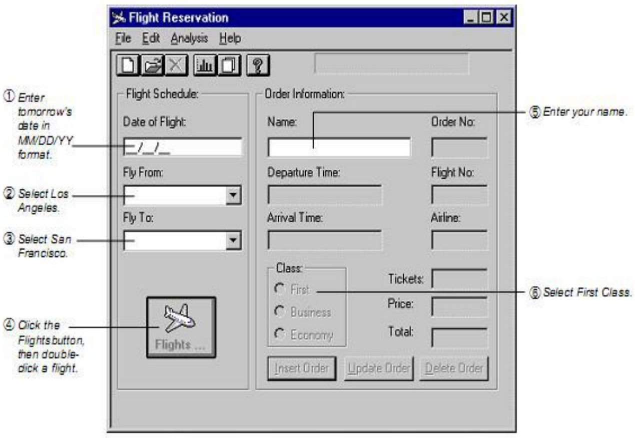
4 Create a new order.

Choose File > New Order in the Flight Reservation application

5 Fill in flight and passenger information.

6 Insert the order into the database.

Click the Insert Order button. When the insertion is complete, the “Insert Done” message appears in the status bar



① Enter tomorrow’s date in MM/DD/YY format.

② Select Los Angeles.

③ Select San Francisco.

④ Click the Flights button, then double-click a flight.

⑤ Enter your name.

⑥ Select First Class.

7 Delete the order.

Click the Delete Order button and click Yes in the message window to confirm the deletion.

8 Stop recording.

Choose Create > Stop Recording or click the Stop button.

9 Save the test.

Choose File > Save. Save the test in a convenient location on our hard drive. Click Save to close the Save Test dialog box.

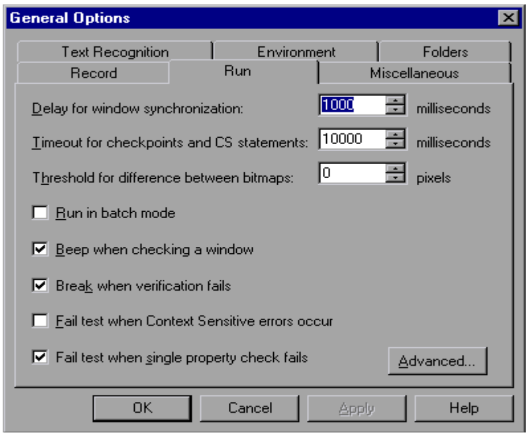
Changing the Synchronization Setting

The default interval that WinRunner waits for an application to respond to input is 10 seconds. In the next exercise we will identify a synchronization problem and add a synchronization point to solve it. To run the test we have just recorded with a synchronization problem, we need to change the default synchronization setting.

1 Open the General Options dialog box.

Choose **Settings** > **General Options**.

2Click the Run tab.



Delay for window synchronization: 1000 milliseconds

Timeout for checkpoints and CS statements: 10000 milliseconds

Threshold for difference between bitmaps: 0 pixels

☐ Run in batch mode

☑ Beep when checking a window

☑ Break when verification fails

☐ Fail test when Context Sensitive errors occur

☑ Fail test when single property check fails

3 Change the value to 1000 milliseconds (1 second).

In the Timeout for Checkpoints and CS statements box, change the value to “1000”.

4 Click OK to close the dialog box.

Identifying a Synchronization Problem

We are now ready to run the test. As the test runs, look for a synchronization problem.

1 Make sure that the test window is active in WinRunner.

Click the title bar of the test window.

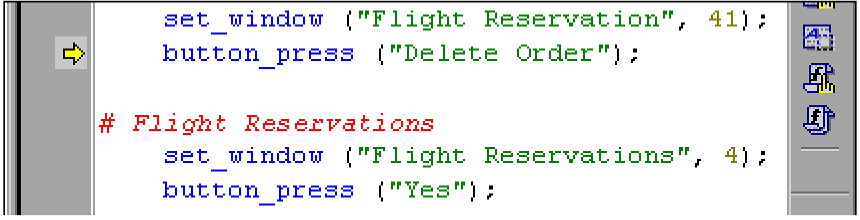
2 Choose Run from Top.

Choose **Run** > **Run from Top** or click the **Run from Top** button. The Run Test dialog box opens. Accept the default test run name “res1.” Make sure that the **Display test results at end of run** check box is selected.

3 Run the test.

Click **OK** in the Run Test dialog box. WinRunner starts running the test. Watch what happens when WinRunner attempts to click the Delete button.

4 Click Pause in the WinRunner message window.
WinRunner fails to click the Delete Order button because the button is still disabled. This erroror occurred because WinRunner did not wait until the Insert Order operation was completed. Note that the execution arrow has paused opposite the command to click the Delete Order button.



```
    set_window ("Flight Reservation", 41);
    button_press ("Delete Order");

# Flight Reservations
    set_window ("Flight Reservations", 4);
    button_press ("Yes");
```

Synchronizing the Test

In this exercise we will insert a synchronization point into the test script. The synchronization point will capture a bitmap image of the "Insert Done"message in the status bar. Later on when we run the test, WinRunner will wait for the "Insert Done" message to appear before it attempts to click the Delete Order button.

1 Make sure that the test window is active in WinRunner.
Click the title bar of the test window.

2 Place the cursor at the point where we want to synchronize the test.
Add a blank line below the button_press ("Insert Order"); statement. Place the cursor at the beginning of the blank line.

3 Synchronize the test so that it waits for the "Insert Done" message to appear in the status bar.
Choose **Create** > **Synchronization Point** > **For Object/Window** or click the **Synchronization Point for Object/Window** button on the User toolbar. Use the pointer to click the message "Insert Done" in the Flight Reservation window. WinRunner automatically inserts an **obj_wait_bitmap** synchronization point into the test script. This statement instructs WinRunner to wait 1 second for the "Insert Done" message to appear in the status bar.

```
obj_wait_bitmap("Insert Done...", "Img1", 10);
```

4 Save the test.
Choose **File** > **Save** or click the **Save** button.

Running the Synchronized Test

In this exercise we will run the synchronized test script and examine the test results.

1 Confirm that the test window is active in WinRunner.
Click the title bar of the test window.

2 Confirm that Verify mode is selected in the Standard toolbar.
Verify mode will stay in effect until we choose a different mode.

3 Choose Run from Top.
Choose **Run** > **Run from Top** or click the **Run from Top** button. The Run Test dialog box opens. Accept the default name "res2." Make sure that the **Display test results at end of run** check box is selected.

4 Run the test.
Click **OK** in the Run Test dialog box. WinRunner starts running the test from the first line in the script. Watch how WinRunner waits for the "Insert Done" message to appear in the status bar.

5 Review the results.

6 Close the Test Results window.
Choose **File** > **Exit**.

7 Close the test.
Choose **File** > **Close** in WinRunner.

8 Close the Flight Reservation application.
Choose **File** > **Exit**.

9 Change the timeout value back to 10000 milliseconds (10 seconds).
Choose **Settings > General Options** to open the General Options dialog box.Click the **Run** tab. In the **Timeout for Checkpoints and CS statements** box, change the current value to "10000". Click **OK** to close the dialog box.