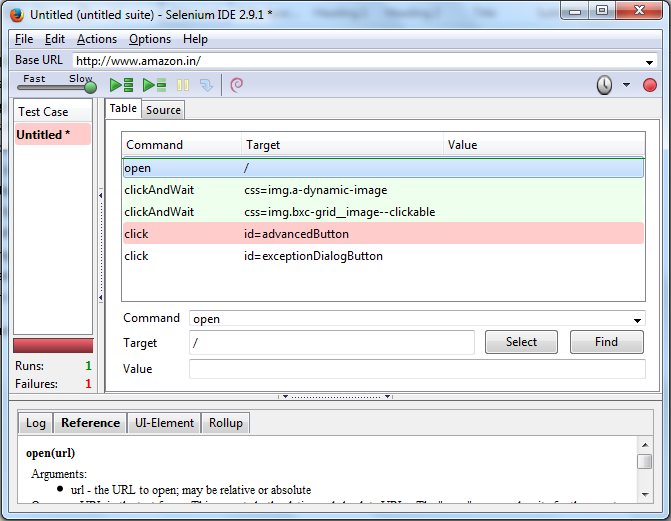
**Selenium IDE**

It works only on Firefox browser and used as firefox plug in. Due to integration in Firefox browser only it is slowly getting outdated. It is a prototyping tool for test as it doesn’t support any programming language. It only deals with element locator and various commands to perform. Various features of selenium IDE include:

1. It is a record and playback tool: record and playbacks the test cases.
2. Edit test cases: We can edit recorded test cases.
3. Execute test suites
4. One can type test cases/script using the element locator and selenese/Selenium IDE commands.
5. Debugging test cases through step by step execution.
6. Export test cases to other programming language whichever we want.
7. Selenium IDE default test case format is .html

Sample Screen of Selenium IDE 2.9.1



It can installed as Firefox add-ons from selenium website. It is installed in Firefox Browser as add on and can be executed from tool->Selenium IDE.

We can test our web project’s every element through element locator and Operation performed on that element. When we open selenium IDE in firebox browser then by default it start recording every user events and prepares the test case and executes by using element locator and operation performed on it.

Base URL: is url of website that we open.

Selenium executes various commands called selenese command like click, clickandwait, open etc and also locates the related elements and also checks for the element’s value if present.

Selenese commands

Open(url): Opens the url typed in the address bar and executes the test case.

ClickandWait(locator): Clicks on links, button, radio button, check box by determining its id on web application and waits for some time which is default set as 30s whether the page opens or not. If it opens then test case is passed else is fails.

Click(Locator): Clicks the link, button, checkbox by determining its id and waits for page to load.

Selenium IDE stores all the events by as user in table which has three fields namely

1. Commands
2. Target
3. Value

**Commands** columns has the various command executes by the selenium IDE while user action’s happened and records the same.

**Target** stores the related things performed on specific elements click events

**Value** stores the text entered in elements like textbox etc.

**Source** tab contains the recorded test case in .html format

**A sample test case in .html format**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head profile="http://selenium-ide.openqa.org/profiles/test-case">

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link rel="selenium.base" href="http://www.amazon.in/" />

<title>New Test</title>

</head>

<body>

<table cellpadding="1" cellspacing="1" border="1">

<thead>

<tr><td rowspan="1" colspan="3">New Test</td></tr>

</thead><tbody>

<tr>

<td>open</td>

<td>/</td>

<td></td>

</tr>

<tr>

<td>clickAndWait</td>

<td>css=img.a-dynamic-image</td>

<td></td>

</tr>

<tr>

<td>clickAndWait</td>

<td>css=img.bxc-grid\_\_image--clickable</td>

<td></td>

</tr>

<tr>

<td>click</td>

<td>id=advancedButton</td>

<td></td>

</tr>

<tr>

<td>click</td>

<td>id=exceptionDialogButton</td>

<td></td>

</tr>

</tbody></table>

</body>

</html>

We can generate the equivalent java code for the same test case in Selenium IDE. To generate the same go to File->Export Test Case As->Java/Junit/WebDriver

**Sample test case in Java format** will be generated as follows:

package com.example.tests;

import java.util.regex.Pattern;

import java.util.concurrent.TimeUnit;

import org.junit.\*;

import static org.junit.Assert.\*;

import static org.hamcrest.CoreMatchers.\*;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.Select;

public class Test2 {

private WebDriver driver;

private String baseUrl;

private boolean acceptNextAlert = true;

private StringBuffer verificationErrors = new StringBuffer();

@Before

public void setUp() throws Exception {

driver = new FirefoxDriver();

baseUrl = "http://www.amazon.in/";

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

}

@Test

public void test2() throws Exception {

driver.get(baseUrl + "/");

driver.findElement(By.cssSelector("img.a-dynamic-image")).click();

driver.findElement(By.cssSelector("img.bxc-grid\_\_image--clickable")).click();

driver.findElement(By.id("advancedButton")).click();

driver.findElement(By.id("exceptionDialogButton")).click();

}

@After

public void tearDown() throws Exception {

driver.quit();

String verificationErrorString = verificationErrors.toString();

if (!"".equals(verificationErrorString)) {

fail(verificationErrorString);

}

}

private boolean isElementPresent(By by) {

try {

driver.findElement(by);

return true;

} catch (NoSuchElementException e) {

return false;

}

}

private boolean isAlertPresent() {

try {

driver.switchTo().alert();

return true;

} catch (NoAlertPresentException e) {

return false;

}

}

private String closeAlertAndGetItsText() {

try {

Alert alert = driver.switchTo().alert();

String alertText = alert.getText();

if (acceptNextAlert) {

alert.accept();

} else {

alert.dismiss();

}

return alertText;

} finally {

acceptNextAlert = true;

}

}

}