Welcome to the Silk Test Workbench Script tutorial. In this tutorial, you will learn the basic steps required to create a script, play back the script, and then analyze the results of the playback. Additionally, you will learn how to use a number of features that allow you to quickly update and enhance a recorded script.

This tutorial assumes some basic knowledge of Microsoft Visual Basic and the Microsoft .NET framework. If you are unfamiliar with the .NET framework, refer to the Microsoft web site for additional help.

This tutorial uses the Silk Test sample Web application, http://demo.borland.com/InsuranceWebExtJS/, to create a real world scenario in which you practice using Silk Test Workbench to create repeatable tests.

**Note:** The sample application used in this tutorial is designed and optimized to run on Internet Explorer. To ensure a user experience consistent with the lessons in the tutorial, Micro Focus does not recommend running the tutorial sample application on one of the other supported browsers instead of Internet Explorer.

**Note:** Before you record or playback web applications, disable all browser add-ons that are installed in your system. To disable add-ons in Internet Explorer, click **Tools > Internet Options**, click the **Programs** tab, click **Manage add-ons**, select an add-on and then click **Disable**.

The lessons in this tutorial are designed to be completed in sequence as each lesson is based on the output of previous lessons.

### Recording a Script: Introduction

As you perform actions to create an insurance quote request in the sample Web application, Silk Test Workbench records the actions. When you have completed recording the actions needed for a script, you can see the recorded script in the Code window.

### Starting the Sample Web Application

For this tutorial, use the Silk Test sample Web application. This Web application is provided for demonstration purposes.

Use the Silk Test sample Web application with Internet Explorer. To ensure a user experience consistent with the lessons in the tutorial, we do not recommend running the sample Web application with one of the other supported browsers instead of Internet Explorer.

1. To record DOM functions to make your test faster and more reliable, perform the following steps:
   a) Click **Tools > Options**.
   b) Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
   c) Click **xBrowser**.
   d) From the **Record native user input** list box, select **No**.
   e) Click **OK**.

   **Note:** Typically, when you test Web applications, you use native user input rather than DOM functions. Native user input supports plug-ins, such as Flash and Java applets, and applications that use AJAX, while high-level API recordings do not.

2. Before you record or playback Web applications, you must disable all browser add-ons. To ensure that all browser add-ons are disabled, perform the following steps:
   a) In Internet Explorer choose **Tools > Internet Options**. The **Internet Options** dialog box opens.
b) Click the Programs tab and then click Manage add-ons. The Manage Add-ons dialog box opens.
c) In the list of add-ons, review the Status column and ensure that the status for each add-on is Disabled.
   If the Status column shows Enabled, select the add-on and then click Disable.
d) Click Close and then click OK.

3. To access the sample application remotely, click http://demo.borland.com/InsuranceWebExtJS. The sample application Web page opens.

Recording a Script for the Sample Web Application

During recording, Silk Test Workbench records all interactions with the test application (except interaction with Silk Test Workbench itself) until recording is stopped. After you have finished recording, you can modify the script you have generated to add and remove steps.

2. Select .NET Script from the asset types list, and then type a name for the script in the Asset name text box.
   For example, type AutoQuote as the title.
3. Check the Begin Recording check box to start recording immediately.
4. Click OK to save the script as an asset and begin recording. The Select Application dialog box opens.
5. Select the Web tab.
6. Select Internet Explorer from the list.
7. In the Enter URL to navigate text box, type demo.borland.com/InsuranceWebExtJS/.
8. Click OK. Silk Test Workbench minimizes and the Recording window opens.
9. In the Insurance Company Web site, perform the following steps:
   a) From the Select a Service or login list box, select Auto Quote. The Automobile Instant Quote page opens.
   b) Type a zip code and email address in the appropriate text boxes, click an automobile type, and then click Next.
      For example, type 92121 as the zip code, jsmith@gmail.com as the email address and specify Car as the automobile type.
   c) Specify an age, click a gender and driving record type, and then click Next.
      For example, type 42 as the age, specify the gender as Male and Good as the driving record type.
   d) Specify a year, make, and model, click the financial info type, and then click Next.
      For example, type 2010 as the year, specify Lexus and RX400 as the make and model, and Lease as the financial info type.
      A summary of the information you specified appears.
   e) Click Purchase.
      The Purchase A Quote page opens.
   f) Click Home near the top of the page to return to the home page where recording started.
10. Stop recording by pressing Alt+F10, clicking Stop in the Recording window, or clicking the Silk Test Workbench taskbar icon. The Recording Complete dialog box opens. If the Do not show this message again check box is checked in the Recording Complete dialog box, this dialog box does not appear after recording is stopped. In this case, the script displays.
11. Click Go to .NET Script. The script displays in the Code window.
12. Click Save.

Reviewing the Recorded Script

Silk Test Workbench records all actions in all applications other than itself. If you followed the instructions carefully, Silk Test Workbench captured only the actions performed on the sample application Web site. Silk Test Workbench repeats these actions during playback.
Your script should look similar to the following sample.

```vbnet
Imports SilkTest.Ntf.XBrowser
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop
    Public Sub Main()
        _desktop.Control("control1").TypeKeys("9")
        With _desktop.BrowserApplication("webBrowser")
            With .BrowserWindow("browserWindow")
                .DomTextField("autoquoteZipcode").SetText("92121")
                .DomTextField("autoquoteEMail").SetText("jsmith@gmail.com")
                .DomRadioButton("autoquoteVehicle0").Select()
                .DomButton("autoquoteNext").Select()
                .DomTextField("autoquoteAge").SetText("42")
                .DomRadioButton("autoquoteGender0").Select()
                .DomRadioButton("autoquoteType1").Select()
                .DomButton("autoquoteNext").Select()
                .DomTextField("autoquoteYear").SetText("2010")
                .DomElement("img").DomClick(MouseButton.Left, New Point(8, 9))
                .DomElement("lexus").DomClick(MouseButton.Left, New Point(87, 7))
                .DomElement("img3").DomClick(MouseButton.Left, New Point(11, 10))
                .DomElement("rX400").DomClick(MouseButton.Left, New Point(96, 11))
                .DomRadioButton("autoquoteFinInfo2").Select()
                .DomButton("autoquoteNext").Select()
                .DomLink("home").Select()
            End With
        End With
    End Sub
End Module
```

Your script may not exactly match the preceding example. Different users interact with applications differently. For example, when filling out a form, some users click from field to field and others use the Tab key. Silk Test Workbench records these actions differently, though they achieve the same results. Your script should play back correctly regardless of these differences.

## Playing Back the Recorded Script

Once you have recorded and saved your script, you can play it back to verify that the script works.

1. Perform one of the following steps:
   - Choose **Actions > Playback**.
   - Click **Playback** on the toolbar.

   The Playback dialog box opens. This dialog box lets you determine how the result is saved.

2. In the **Result description** text box, type *Initial test results for the recorded test*.

3. Click **OK**.

4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   Each result is identified with a unique test run number.

   Silk Test Workbench minimizes and the script plays back. During playback, the actions you performed while recording the script are played back on the screen against the sample application. When playback completes successfully, the **Playback Complete** dialog box opens.

5. Click **Go to Result**. The **Result** window opens.
Analyzing Results: Introduction

After playing back a script, Silk Test Workbench generates a test result. A test result contains information about the playback of the script. Information such as the name of the script, the run number, the date and time each step executed, the pass or fail status of each step, and other important information.

Enhancing the Script: Introduction

Enhancing a test includes making updates to the existing test to ensure that it works with newer versions of the test application. For example, to handle and verify varying conditions in the test application you can insert test logic. Additionally, to increase the readability of a test or to remind yourself or others about important aspects of the test, you can insert a message box.

These are just some of the ways in which you can use Silk Test Workbench to enhance existing tests to create more powerful, robust, and flexible tests.

Inserting a Verification

A verification is test logic that evaluates a user-defined condition, and then sends a pass/fail message to the playback result.

In this lesson, you will insert a verification to ensure that the quote uses the correct vehicle model.

1. Select the following text that defines the model type for the auto quote.
   `.DomElement("RX400").DomClick(MouseButton.Left, New Point(96, 11))`

2. Navigate to the page in the instant quote wizard where the Model type is specified and note a different model type. For example, GS430 is a model type for the Lexus make.

3. In Silk Test Workbench, change the model type.
   Change the following textContents code from:
   `.DomElement("RX400").DomClick(MouseButton.Left, New Point(96, 11))`
   to:
   `.DomElement("GS430").DomClick(MouseButton.Left, New Point(96, 11))`

4. To compare the expected value with the actual value and add a comment, type:
   `Workbench.Verify ("GS430", .DomTextField("modelCombo").Text, "The model type is correct")`

You have successfully enhanced the recorded script by inserting test logic that verifies the value of a property in the sample application.

Creating and Storing Application Data in a Local Variable

Variables enhance tests by providing the ability to store data values for use in other parts of the script. Data can also be output to other types of files.

The sample application displays a unique email address on the Get Instant Auto Quote page. The text on this page containing the email address is the property value of a control on the page.

In this lesson, you will store this text to the local variable stremailAddr.
1. In the script, navigate to the email value. The code should look similar to the following:
   ```
   .DomTextField("autoquoteEMail").SetText("jsmith@gmail.com")
   ```
2. Insert the following code following the email value:
   ```
   Dim StremailAddr As String
   StremailAddr = .DomTextField("autoquoteEMail").Text
   ```
   This step creates a new local variable, StremailAddr, that stores the email address text from the Get Instant Auto Quote page to the local variable.
3. To include output that displays the variable text during playback, include a `Console.Write` command in your script.
   For example, include:
   ```
   Console.Write (StremailAddr)
   ```
4. To view the console output, choose View > Output. The Output window opens. When you playback a script, the Output window is populated.

To confirm that the test captures the property value and stores it properly, play back the test and review the result.

### Playing Back and Analyzing the Enhanced Script

Now that you have made several enhancements to the recorded script, play back the script and analyze the result.

1. Perform one of the following steps:
   - Choose Actions > Playback.
   - Click Playback on the toolbar.
   
   The Playback dialog box opens.
2. In the Result description text box, type Enhanced test results for the script.
3. Click OK.
4. If multiple browsers that are supported for replay are installed on the machine, the Select Browser dialog box opens. Select the browser and click Run. Silk Test Workbench plays back the enhanced script.
5. Click Go to Result on the Playback Complete dialog box.
   
   The Result window opens to the Summary tab by default.
   
   The Summary tab shows that the script passed, which means it played back successfully without any errors, or failed verifications.
6. Click the Passed (1) tab.
   
   The number in parentheses indicates the total number of verifications that passed. The Test Steps pane displays the verification step and the Result Detail column displays the pass text description of the verification.
7. Click the Details tab to display the result of every action.
8. In the Output window, scroll down to the line that shows the result of storing the email address to a variable.
   
   The text is similar to the following:
   ```
   jsmith@gmail.com
   ```
9. Click the Passed tab to see the results of the verification for the model type.
   
   The text is similar to the following:
   ```
   Main:Verify Passed: The model type is correct
   ```

The verification results also display in the Result and Result Detail columns.
Congratulations! You have successfully created a script that reliably tests the sample application. In the next lesson, you will learn about several advanced testing concepts and features such as how to quickly and easily execute a script within another script.

**Executing a Script Within a Script: Introduction**

In this tutorial, you created a single script that performs every action required to receive an auto insurance quote from the web application. A single script is useful when implementing a basic test case against a simple application. However, most software testing requires a more rigorous approach that involves testing every aspect of an application. An additional requirement is the ability to rapidly update existing scripts whenever the test application changes.

To provide an efficient means for solving these testing challenges, Silk Test Workbench supports modular testing, in which you can "chunk" common sets of actions of a particular testing solution into a single test, and then reuse the script in other scripts that require the same set of actions.

**Modular Testing**

Before creating visual tests, scripts, and other Silk Test Workbench assets to build application testing solutions, it is a good practice to plan a testing strategy.

It is not necessary to include all the parts of a specific test solution in a single visual test or script and is not usually beneficial to do so.

Typically, the most efficient testing approach is a modular approach. Think of your application testing in terms of distinct series of transaction units.

For example, testing an online ordering system might include the following distinct transaction units:

- Log on to the online system
- Create a customer profile
- Place orders
- Log off the online system

If one test is created to handle all of these distinct units and there are ten different scenarios that use this test, you would need to record ten different tests to handle the scenarios. If any change occurs to the application, for example if an extra field is added to the logon window, ten different tests would require a change to accommodate data input to the new field.

Rather than creating one visual test or script that tests all of these transaction units, and then recreating it ten times for each scenario, it may be more beneficial to create separate tests as test "modules" that handle each one of these transaction units. If a separate test is created for each of the separate transaction units and reused for each of the test scenarios, then only the test that handles the logon transaction unit would require change.

Now that you understand the basics of modular testing, you are ready to create a second test and add it to the test you created in the previous lessons.

**Recording the Second Script**

In this section of the lesson, you record a second script for the tutorial and learn an alternate way to create a script.

2. Select .NET Script from the asset types list, and then type a name for the script in the Asset name text box.
   For this tutorial, type AddAccount for the name.
3. Check the **Begin Recording** check box to start recording immediately.
4. Click **OK** to save the script as an asset and begin recording. The **Select Application** dialog box opens.
5. Select the **Web tab**.
6. Select **Internet Explorer** from the list.
7. From the **Home** page of the sample application, click **Sign Up** in the **Login** section. The **Create A New Account** page opens.
8. Provide the following information in the appropriate fields.
   Press the **Tab** key to move from one field to the next.
   
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Pat</td>
</tr>
<tr>
<td>Last Name</td>
<td>Smith</td>
</tr>
<tr>
<td>Birthday</td>
<td>February 12, 1990</td>
</tr>
<tr>
<td>E-Mail Address</td>
<td><a href="mailto:smith@test.com">smith@test.com</a></td>
</tr>
<tr>
<td>Mailing Address</td>
<td>1212 Test Way</td>
</tr>
<tr>
<td>City</td>
<td>San Diego</td>
</tr>
<tr>
<td>State</td>
<td>CA</td>
</tr>
<tr>
<td>Postal Code</td>
<td>92121</td>
</tr>
<tr>
<td>Password</td>
<td>test</td>
</tr>
</tbody>
</table>

   **Note:** Click the down arrow next to the month and year in the calendar control to change the month and year and then select 12 on the calendar.
9. Click **Sign Up**.
10. Click **Continue**. The contact information is displayed.
11. Click **Home** near the top of the page to return to the home page where recording started.
12. Click **Log Out**.
13. Press **Alt+F10** to complete recording. The **Recording Complete** dialog box opens.
14. Click **Save**. The script opens in the **Code** window.

## Inserting One Script Within Another

In this section of the lesson, you will learn how to insert the second script, which adds a user account, in the original script before the code that perform the request for an auto quote.

Executing scripts within scripts is a powerful method for efficiently testing the same basic actions in scripts.

**Tip:** When inserting a script within another script, it is important to ensure that any test applications are in the correct initial playback state.

1. Choose **File > Open**. The **Asset Browser** opens.
2. Select **.NET Script** in the left pane to display the list of scripts.
3. From the list, double-click **AutoQuote** to open it.
   
   **AutoQuote** is the first test that you created in this tutorial.
4. In the **Code** window, position the cursor after the **Public Sub Main()** code, press **Enter** to add a new line, and type:
   
   ```vbscript
   Workbench.RunScript ("AddAccount")
   ```

   where **AddAccount** is the name of the second script that you created.
Because we want to add the account information before we execute the quote steps, we added the Workbench.RunScript command before the With statement. To execute the AddAccount script after the quote steps, add the Workbench.RunScript command after the End With statement.

Responding to Playback Errors: Introduction

Errors encountered during playback can be caused by a variety of factors, such as changes in the test application and improper workflow. Quickly diagnosing and fixing these errors minimizes test maintenance and allows for a more efficient team testing effort.

First, begin this lesson by playing back the modular script you created in the previous lesson.

Playing Back the Modular Script

In the previous lesson, you created a modular script by inserting AddAccount into the AutoQuote script. In this section of the lesson, you will play back the modular script and encounter an error during playback.

1. With the AutoQuote script open, click Playback on the toolbar. The Playback dialog box opens.
2. In the Result description text box, type Responding to errors in a modular test.
3. Click OK.
4. If multiple browsers that are supported for replay are installed on the machine, the Select Browser dialog box opens. Select the browser and click Run.

   During playback, the test stops on the Create A New Account page and an error message opens.

   This error occurs because the database requires a unique email address for each customer record. Since you have already entered the email address during the recording of the AddAccount script, the email address already exists in the database and the test fails.

Reviewing the Result

Review the results of the script.

1. Click End to stop playback. The Playback Complete dialog box opens.
2. Click Go to Result. The AutoQuote result appears with the Summary tab displayed by default.

   The Summary tab displays the overall details of the test run. Note that the Visual tests or .NET Scripts (number of times each ran) field lists AutoQuote(1) and the inserted script, AddAccount(1).
3. Click the Details tab.
4. Scroll down to the steps in blue text.

   By reviewing the Result and Result Detail columns, you can quickly find information about any errors that occurred during playback.

   Note: The Failed tab does not display steps containing playback errors. It only displays failed verifications.

You have learned how to diagnose playback errors. For the purposes of this tutorial and to successfully replay the script once, you can manually modify the email in the script to avoid the error. To do so, change smith@test.com in the AddAccount script to psmith@test.com
Index

E
enhancing scripts
  adding variables 7
  adding verifications 7
  overview 7
  playing back 8

M
modular scripts
  errors 11
  inserting 10
  overview 9
modular testing
  overview 9
modular tests
  overview 9

P
playing back
  recorded scripts 6
playing back scripts
  enhanced scripts 8
  modular 11

R
recording
  scripts, introduction 4
recording scripts
  sample application 5

S
sample application
  recording script 5
  starting 4
scripts
  inserting 10
  modular 9
  playback errors 11
  playing back 6, 8
  recording for sample application 5
  recording, introduction 4
  reviewing 5
  sample application 4
  tutorial, recording second script 9

T
tests
  modular 9

V
variables
  adding 7
verifications
  adding to scripts 7
visual tests
  modular overview 9