

Web Functional Testing

The following exercises demonstrate how to use SOAtest to perform functional testing on the Web interface. It covers:

- [Introduction](#)
- [Cross-browser Testing and Validation Video Overview](#)
- [Recording in a Browser](#)
- [Adding a Data Source](#)
- [Parameterizing a Form Input](#)
- [Configuring Validation on a Page Element](#)
- [Configuring a Browser Stub](#)
- [Playing a Recorded Scenario in a Browser](#)
- [Performing Static Analysis During Web Scenario Execution](#)

Introduction

Web interface testing is difficult to automate. Teams often abandon automated testing in favor of manual testing due to too many false positives or too much effort required to maintain the test suites.

SOAtest facilitates the creation of automated test suites that are reliable and dependable. Its ability to isolate testing to specific elements of the Web interface eliminates noise and provides accurate results.

SOAtest isolates and tests individual application components for correct functionality across multiple browsers without requiring scripts. Dynamic data can be stubbed out with constant data to reduce test case noise. Validations can be performed at the page object level as well as the HTTP message level. SOAtest also verifies the client-side JavaScript engine under expected and unexpected conditions through asynchronous HTTP message stubbing.

Cross-browser Testing and Validation Video Overview

Recording in a Browser

Before performing the following exercises, ensure that you have started the ParaBank Server (as described in [Setting Up ParaBank](#)) and that you can navigate successfully to the default address (localhost:8080/parabank). Close the browser window after you verify this.

To record in a browser: Right-click the project from the previous exercises, then choose **Add New** > **Test (.tst) File** > **File** from the shortcut menu.

Enter a name for the file, then click **Next** > **Select Web** > **Record web scenario** and click **Next**.

In the first Record Web Scenario wizard page, ensure that **Record new web scenario** is selected, then click **Next**.

Complete the next Record Web Scenario wizard page as follows:

- Enter `localhost:8080/parabank` in the **Start Recording From** field.
- Ensure that the following options are checked, and the others are not:
 - **Generate Functional Test**
 - **Generate Asynchronous Request Tests**
- Click the **Finish** button. The test will begin, and a browser window will open.
- Within the browser window that opens, perform the following actions:
 - Type `john` in the **Username** field.
 - Type `demo` in the **Password** field.
 - Click **Log In**.
- Click the first account number listed in the Accounts Overview page.

Validate... text now contains only the dollar amount. The Left-hand text that you added removes the dollar sign from the validation, allowing you to focus on validating the actual amount.

Ensure that **Validate** is selected, and that the expected text is displayed on the rendered page.

A Browser Validation Tool will be added to this test. As the red highlight indicates, it is set up to check that the element you selected remains the same as the application evolves.

To configure a browser stub:

- Expand the **Scenario: Web Functional Testing** branch.
- Right-click **Scenario: Form login**; Test 3: Click 'Log In' and choose **Add Output** from the shortcut menu.
- In the Add Output wizard that opens, choose **HTTP Traffic**, then click **Next**.
- In the Add Output wizard that opens, choose **HTTP Traffic**, then click **Next**.
- Select **http://localhost:8080/parabank/overview.htm** and click **Next**.
- In the left panel, select **Both**; Stub **Request/Response**.
- In the right panel, select any **Browser Stub**.
- Click **Finish**.

In the Response Body section of the Browser Stub configuration view (opened on the right side of the GUI), change the dollar amount in the table from its existing value of -\$2300 to \$1000.00. The easiest way to do this is to copy the entire text in the Response Body into a text editor (e.g. Notepad), use its search capability to find the previous dollar amount, modify the value, and then copy all the text from the editor and replace the current contents in the Response Body. This response body will be provided in place of the actual server response when the recorded scenario is run later in this example.

Click **Save** to save the changes.

To playback the recorded scenario in a browser:

- In the Test Case Explorer, select the **Scenario: Web Functional Testing** node.
- Click the **Test** toolbar button.

The recorded scenario will now be played back in your browser, once for each search value that we parameterized in a previous section of this example. Please wait for each action to be played out in your browser.

Note if you have been following the complete tutorial, errors will be reported due to the Browser Stub tool that was previously added. This is expected.

To configure SOAtest to perform static analysis on the Web pages that the browser downloads as web scenarios execute:

- In the Test Case Explorer, select the **Scenario: Web Functional Testing** node.
- Open the **Test** toolbar button's pull-down menu, then choose one of the available static analysis configurations.

When SOAtest is finished performing static analysis, static analysis violations are shown in the Quality Tasks view. To facilitate review of these results:

- Open the pull-down menu on the top right of the Quality Tasks view.
- Choose **Show**; SOAtest Static Analysis for Functional Tests Layout.