Web Static Analysis

Introduction

While functional testing finds problems by simulating how click paths would operate in a browser, static analysis finds problems by inspecting pages' source code. Static analysis is just like a code review or code inspection. It reads and analyzes your source files, then it lets you know if it finds any coding errors that could cause functionality or presentation problems. It also pinpoints broken links and other navigational problems it finds. In addition, it can alert you to code that might not work correctly when people with disabilities try to use your site with adaptive devices, such as machines that read screen content out loud or convert content into Braille. Furthermore, static analysis can verify that custom design and content requirements, such as corporate branding, are met.

SOAtest facilitates static analysis by automating complex analyses that would otherwise take days. Static analysis can be customized to involve tools that help expose and prevent problems such as:

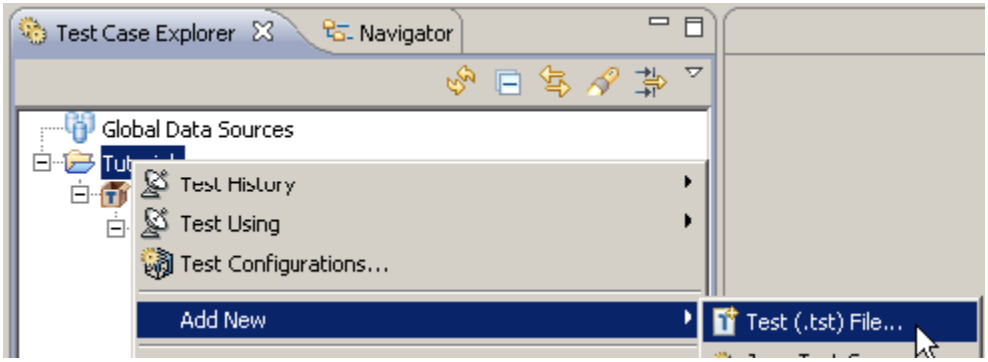
- Coding constructs that make code more error-prone and difficult to update.
- Navigational problems such as broken links, actions that invoke designated error pages, anchor problems, non-clickable links, and so forth.
- HTML, CSS, JavaScript, and VBScript coding problems that affect presentation, execution, dynamic content, performance, transformation, display in non-traditional browsers, etc.
- XML problems that affect transformations and data retrieval.
- Code and content that violates Web accessibility guidelines.
- Content that contains misspellings and typos.
- Code that violates application-specific requirements and business rules.
- Code that violates project-specific, organization-specific branding, design, or content guidelines.

The following exercises will demonstrate how to use SOAtest's Static Analysis feature to create a project, load an initial site into that project, and then quickly and easily analyze that project's files for any coding errors. We will not use the projects created from the previous exercises.

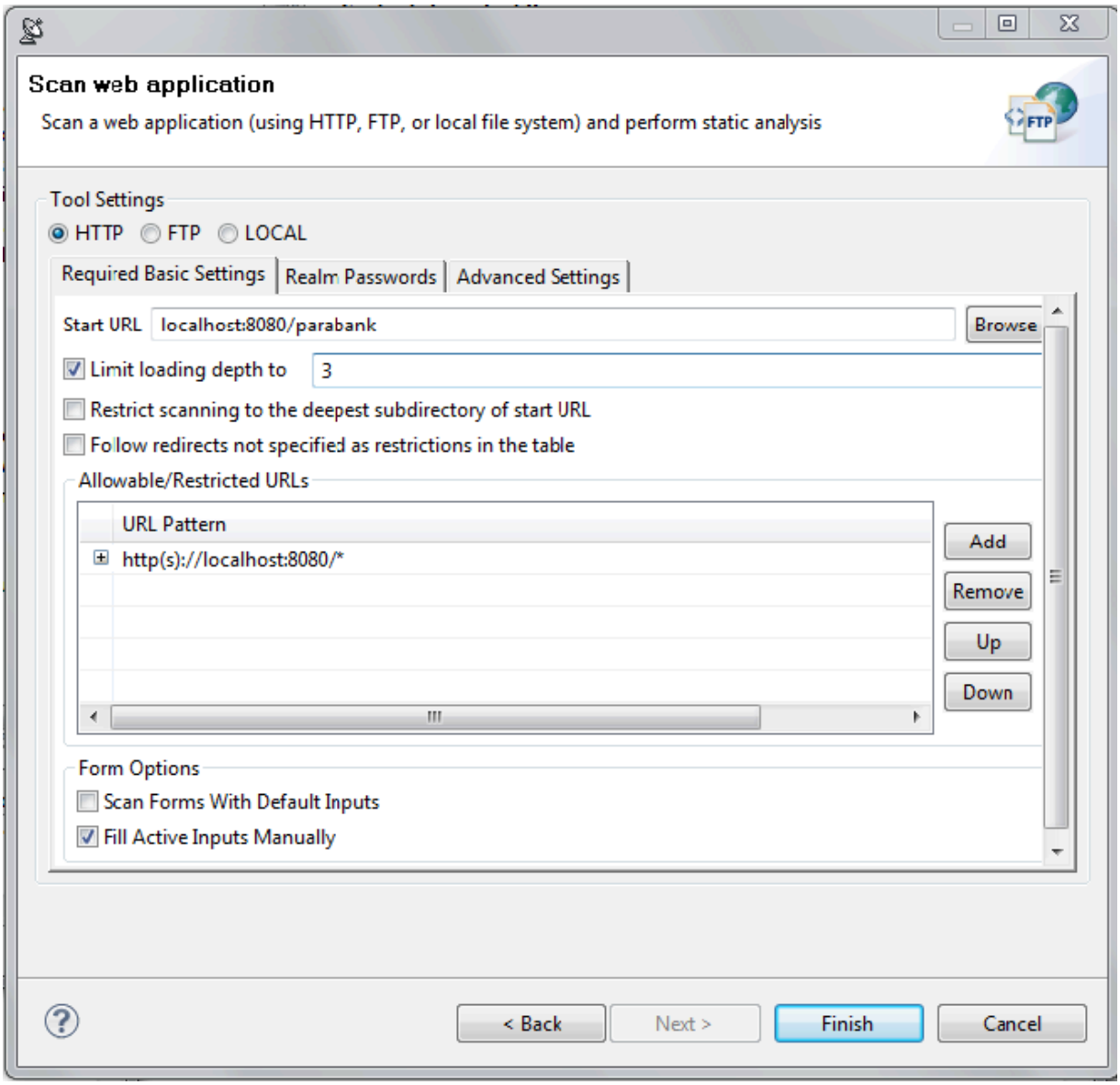
Performing Static Analysis

To perform static analysis, complete the following:

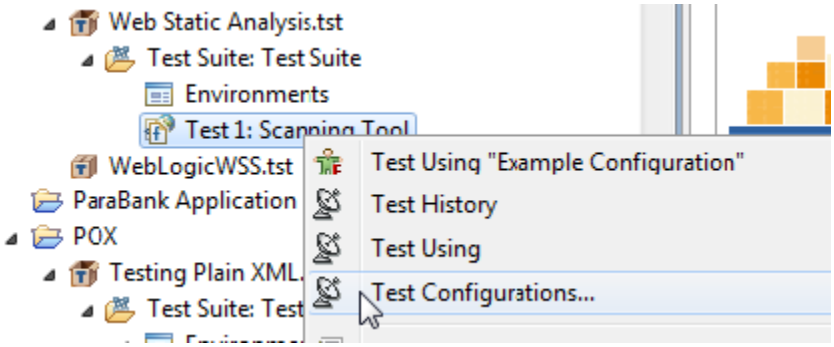
1. Right-click the project from the previous exercises, then choose **Add New> Test (.tst) File** from the shortcut menu.



2. Enter a name for the file (i.e. `Web Static Analysis`), then click **Next**.
3. Select **Web> Scanweb application** and click **Next**.
4. Complete the Scan HTTP/FTP/Local Resources page as follows:
    1. Enter `localhost:8080/parabank` in the **Start URL** field. SOAtest will add this URL to the Allowable/Restricted URLs table.
    2. Check the **Limit loading depth** to check box and enter 3 in the field to the immediate right. This depth setting determines the number of clicks (links) deep to load the site. For example, a loading depth of 3 tells SOAtest to load only pages that can be reached in two clicks from the start URL (Note: redirect is also considered to be 1 depth).
    3. In the Form Options section at the bottom of the panel, set the options as follows:
        - **Scan Forms With Default Inputs:** Unchecked.
        - **Fill Active Inputs Manually:** Checked.

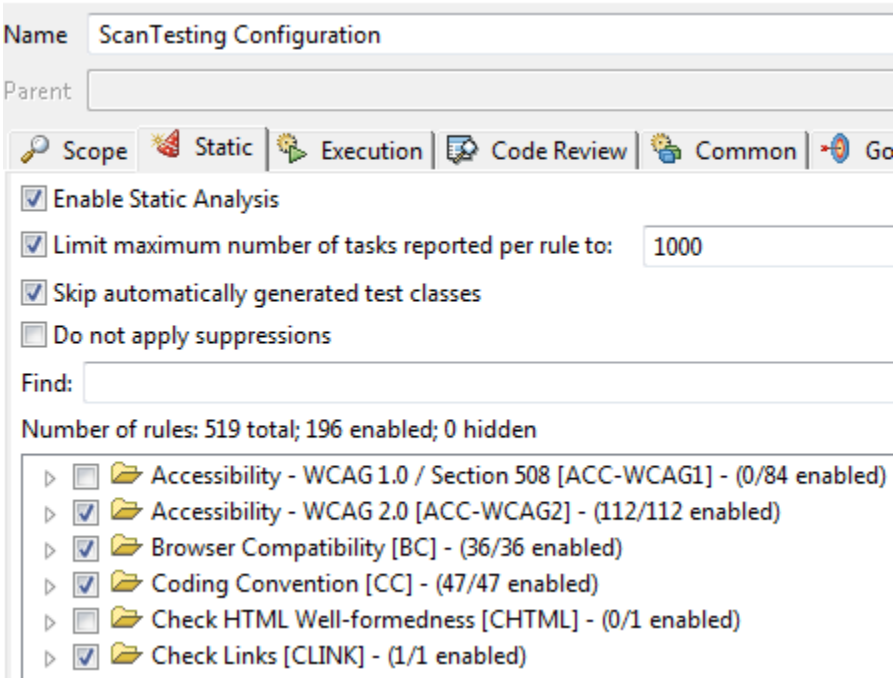


4. Click **Finish**.
   A new .tst file, and a test suite with one test, will be added to the Test Case Explorer.
5. Right-click the **Test 1: Scanning Tool** node and select **Test Configurations** from the shortcut menu.

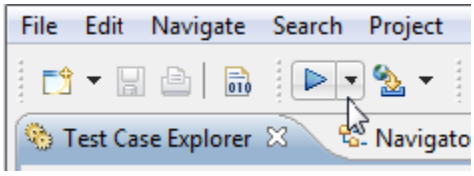


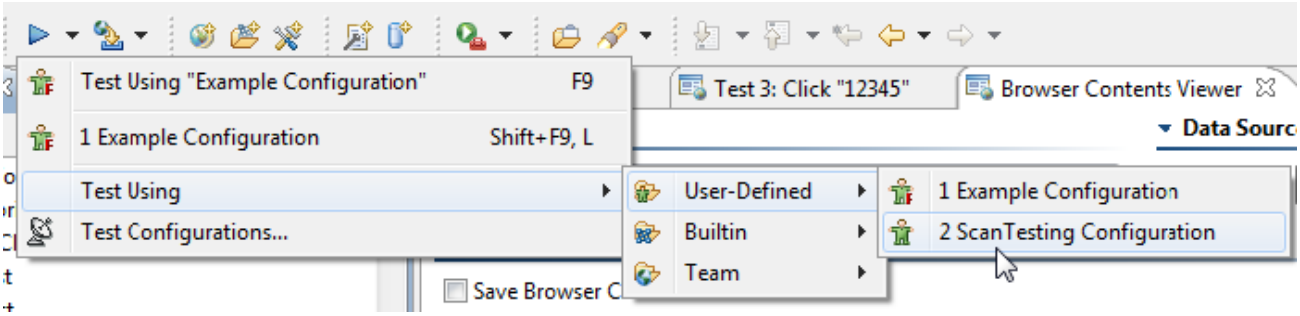
6. In the Test Configuration dialog that appears, select **User-defined** in the left panel and click the **New** button under the panel.
7. In the new configuration panel that appears to the right, Change the **Name** field to `ScanTesting Configuration`.
8. Open the **Static** tab, and check the **Enable Static Analysis** check box.
9. In the list of rules below in the same tab, check the following options:

- **Accessibility - WCAG 2.0 [ACC-WCAG2]**
- **Browser Compatibility [BC]**
- **Coding Convention [CC]**
- **Check Links [CLINK]**



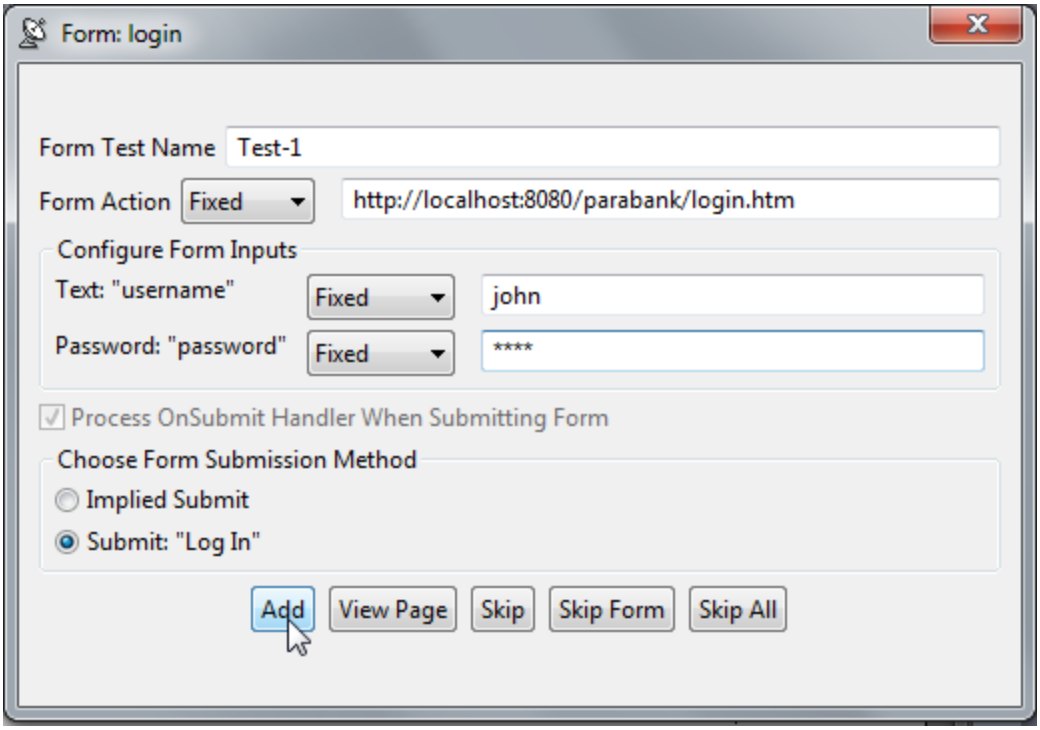
10. Click the **Apply** button to save the configuration, then click **Close**.
11. Select the scan testing test suite in the Test Case Explorer.
12. Open the pull-down menu for the **Test** toolbar button, then choose **Test Using> User-Defined> ScanTesting Configuration**.





While scanning the specified site, SOAtest will open a Form dialog box each time it loads a form that allows user input. In this example, a Form input dialog box will open for the form1 form.

13. Complete the Form:Form1 dialog as follows:
    1. Select **Fixed** from the **Text: "username"** drop-down menu, then enter `john` in the corresponding text field.
    2. Select **Fixed** from the **Password: "password"** drop-down menu, then enter `demo` in the corresponding text field. This tells SOAtest how to populate this particular form's input elements.
    3. Click the **Add** button.

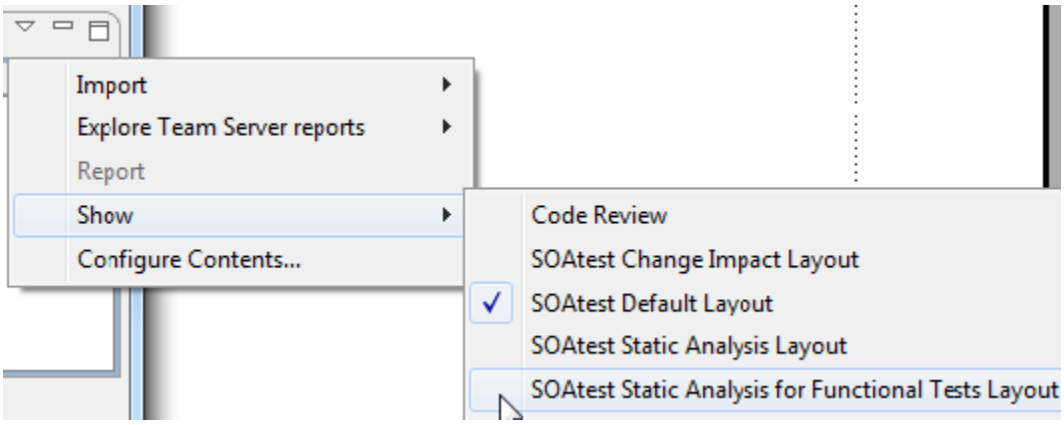


SOAtest will then reopen the same form dialog box so you can enter additional inputs if desired.

4. Click the **Skip All** button to indicate that you do not want to enter any more inputs for forms in this site.

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

1. Open the pull-down menu on the top right of the Quality Tasks view.



2. Choose **Show> SOAtest Static Analysis for Functional Tests Layout.**

Additional Options

You can customize static analysis tests by:

- Enabling or disabling static analysis rules.
- Customizing the parameters of the various rules applied during static analysis.
- Designing rules that verify compliance with unique team or project requirements, then configuring SOAtest to apply those rules during static analysis.
- Configuring SOAtest to suppress error messages that are not relevant to your project.