PegaRULES Process Commander Development

UnitTest RuleSet Documentation and Manual

An xUnit testing-framework for PegaRULES Process Commander

Paul Evans

07/13/2007

Revision Chart

The following chart lists the revisions made to this document. Use this to describe the edits each time this document is re-published (both in draft and final forms). The description should include as many details of the edits as possible, as well as identify the reviewers who requested the edits.

Date	Author	Description
02/01/2006	Paul Evans	Initial draft
07/22/2006	Paul Evans	Minor updates
12/12/2006	Paul Evans	Updates to account for UnitTest-Portal rule sets
06/09/2007	Paul Evans	Updates to account for batch-mode and show-step updates
06/23/2007	Paul Evans	Updates to account for 01-01-06 release
07/06/2007	Paul Evans	Updates to account for 01-01-07 release
07/13/2007	Paul Evans	Added LGPL license text
08/19/2007	Paul Evans	Updates to account for new standard for naming a test class

Contents

License

License Preamble

Full License Text

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
  - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
  - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not

6. Revised Versions of the GNU Lesser General Public License.

Introduction

Overview

In an effort to improve the quality of applications developed using PegaRULES Process Commander, I have developed an open source framework released under the LGPL for the explicit purpose of unit testing. The framework contains the basic class structure and rules that system architects, process architects and administrators can use to create repeatable test activities to unit test various rule types within PRPC. This document describes the motivations behind the development of this framework, how to install and use the framework with screenshots included for clarity. The last section of this document describes possible enhancements that can be made to this framework.

Motivations

The motivation for creating this unit testing framework is many-fold:

Repeatability

PRPC provides out-of-the-box functionality for unit testing various rule types. However such unit tests are not persisted to the rule base and therefore can not easily be repeated on demand. In addition, unit tests can only be created and executed by developers intimately familiar with the rules being tested. This greatly deters the ability to regression test code units within a PRPC-based application. Since the out-of-the-box pattern for running unit tests are not repeatable (they typically consist of hitting the "play" button which launches a run-dialog with the ability to set up clipboard pages necessary for the rule in question to execute properly), it cannot easily be executed in a repeatable fashion. Unit tests created using the UnitTest rule set are implemented as activities, and therefore persisted to the rule base. At any time a user, with the appropriate access group, can execute a test activity and know immediately if the test succeeded or failed. In addition a developer unfamiliar with the rules (perhaps a new addition to the development team for the project) can check to see if a rule passes its unit tests by simply running the corresponding unit test activity. Even non-technical members of a team can check if a rule is working properly by running the unit test activities. Unit test activities developed using this framework can simply be executed; no clipboard pages need to be manually created and initialized with data; this task is done by the unit test activity.

Immediate Feedback of Test Results

Unit tests developed using this framework provide the user running the test immediate feedback if the test failed or succeeded along with a reason-message in the event of a failure. In contrast, the out-of-the-box mechanism for running unit tests forces the user to manually analyze values displayed on the run-dialog or on the clipboard that result from running the unit test. This is time-consuming and error prone; especially for users running the tests who are not the original author of the rule being tested. In some cases, given the complexity of the rule being tested, it may be exceptionally difficult for new developers to successfully unit test a rule using the out-of-the-box mechanism.

Test-Driven Development (TDD)

Unit testing is a building block of test-driven development (TDD). This is important since experience has shown many PRPC projects typically live in rapid-development or iterative-development environments. It has also been reported Pegasystems' sales team touts the ability of PRPC to be utilized in an iterative, or agile environment. Test-driven development is a technique heavily emphasized in agile development circles in order to improve the quality of the code as well as aid in the actual design of the application. Restated, since unit testing (more specifically, unit tests that are repeatable) is central to an agile development approach, this package greatly enhances the probability for success. For more information on TDD, please refer to the resources section at the bottom of this document.

Regression Testing

In the event of code refactoring, which is commonplace in an agile development model, it is of great value to be able to re-run the unit tests to ensure the refactoring has not "broken" some other part of the system. This unit testing framework ships with a gadget that is included in the standard developer portals (`SysArchDD`, `SysAdminDD`, etc) on both the "Dashboard" and "Manage Rules" tabs that contains a button when clicked will search the rule base for all unit test activities accessible by the current operator, and runs them. A report is then presented showing the user which unit test activities failed or succeeded providing the developer with immediate feedback about whether edits to a rule have broken some other rule within the system.

Industry Standards-based (xUnit)

The design and usage of the rules defined within this framework are based on the testing framework developed by Kent Beck. The design of the UnitTest rule set follows the pattern of the xUnit family of testing frameworks. For more information about xUnit, please refer to the resources-section at the bottom of this document. The following present similarities between this framework and JUnit (http://www.junit.org):

- Developers create a "Test" class for each production-class to be tested (in JUnit, this class extends junit.framework.TestCase; in this framework, the Test class should direct-inherit from the class it is testing)
- Developers create a "Test" function/activity for each corresponding production function/activity (or any other unit-testable-rule type) to test
- The "Test" function/activity contains "Assert" calls verifying expected values versus actual values (in JUnit, the "Assert" methods are inherited from the TestCase class; in this framework, activity-calls are explicitly made to the appropriate UnitTest-Test.AssertXXX activity)
- Both frameworks have the ability to batch-run all of the found "Test" functions/activities providing a report of all the failures and successes; JUnit does this using a TestSuite; in this framework this is accomplished by using the "Run Unit Tests" portal-gadget

Given the myriad similarities between this framework and JUnit, PegaRULES developers with a Java background with experience with JUnit should experience a relatively small learning curve.

Documentation

In addition to providing the foundation for creating robust and reliable rules, unit test activities written using this framework double as a source of documentation for the developer on how the rule being tested is meant to be used. Because the unit test activity simulates the runtime-state of the clipboard as it would be in the production system, it provides a level of documentation and clarity about the rule being tested that is extremely valuable.

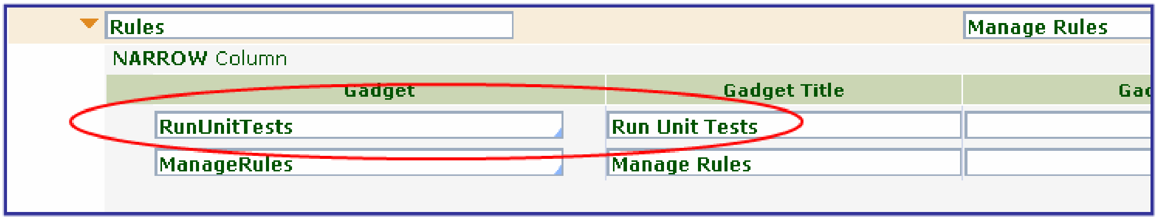
Installation Instructions

To install PRUnit:

- Download from http://prunit.sourceforge.net the latest release of the PRUnit Zip file
- Log in to PRPC with an account with administrator-access
- Upload the `PRUnit-PRPCvXX_rsvNN-NN-NN.zip` file you downloaded from SourceForge to the application server (the "XX" will either be "4.2" or "5" depending if you're using a 4.2 or 5-based PRPC system) (NN-NN-NN will be the release number)
- Modify the appropriate access groups adding the `UnitTest` and `UnitTest-Portal` rule sets to the list of available rule sets

V4.2-specific Instructions

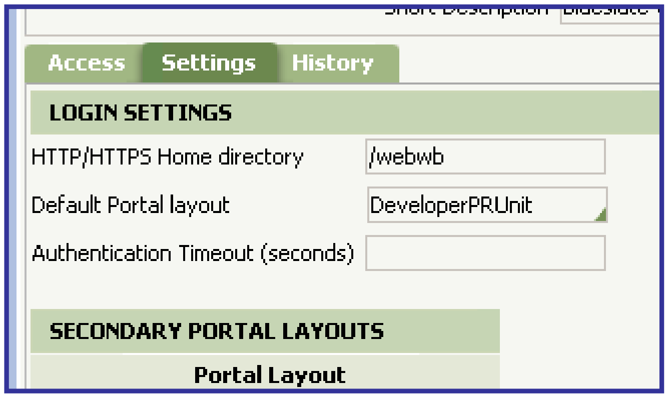
One the ZIP has been and imported and the access groups modified to include the UnitTest and UnitTest-Portal rule sets, if you are using custom portals, you will need to modify your portal(s) to include the "RunUnitTests" gadget:



Gadget	Gadget Title	Gad
RunUnitTests	Run Unit Tests	
ManageRules	Manage Rules	

V5-specific Instructions

One the ZIP has been and imported and the access groups modified to include the UnitTest and UnitTest-Portal rule sets, you need to update the developer/admin access groups to use the "DeveloperPRUnit" portal:



LOGIN SETTINGS

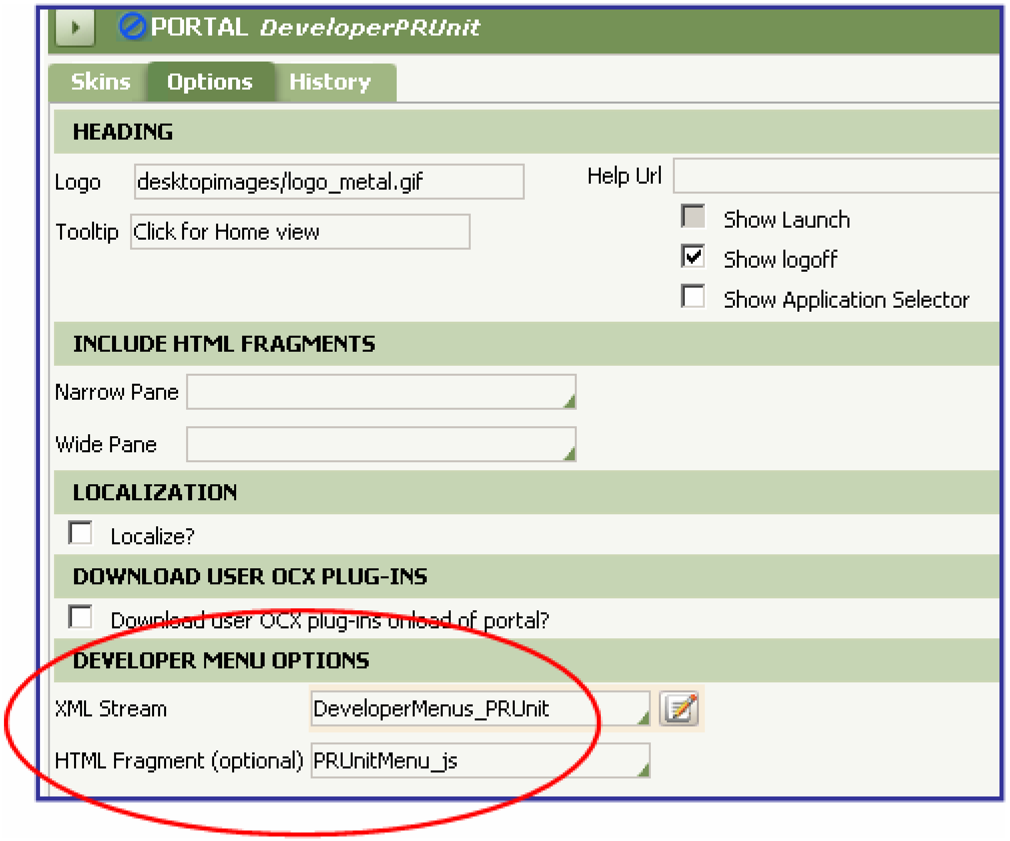
HTTP/HTTPS Home directory	/webwb
Default Portal layout	DeveloperPRUnit
Authentication Timeout (seconds)	

SECONDARY PORTAL LAYOUTS

Portal Layout

If you are using a custom portal and cannot leverage the DeveloperPRUnit portal, you can modify your custom portal to manually add the "Run All Unit Tests" menu item.

The custom portal in that case would need to leverage the "DeveloperMenus_PRUnit" XML rule and the "PRUnitMenu_js" HTML fragment:


PORTAL *DeveloperPRUnit*

Skins | Options | History

HEADING

Logo: desktopimages/logo_metal.gif

Tooltip: Click for Home view

Help Url:

☐ Show Launch
☑ Show logoff
☐ Show Application Selector

INCLUDE HTML FRAGMENTS

Narrow Pane:

Wide Pane:

LOCALIZATION

☐ Localize?

DOWNLOAD USER OCX PLUG-INS

☐ Download user OCX plug-ins onload of portal?

DEVELOPER MENU OPTIONS

XML Stream: DeveloperMenus_PRUnit

HTML Fragment (optional): PRUnitMenu_js

Description of Class Structure

UnitTest-
Base class of the UnitTest framework.

UnitTest-Assert
Contains the assert-activities used in test activities.

UnitTest-TestActivity
Models a test activity. Contains an activity and connect SQL rule for looking up all of the test activities in the rule base. Each test activity found is stored in an instance of this class within the Code-Pega-List created from the RDB-List method.

UnitTest-TestRunMetadata
Contains properties that store metadata about the full test run of all the test activities.

UnitTest-Utilities
Location of utility-related activities.

Rule-Utility-Function-Test
This test class comes with the UnitTest rule set to be used as a container for test activities for testing utility-function rules.

Description of Supporting Rules

Activities

UnitTest-.RunAllUnitTests

Searches the rule base for test activities defined in test classes defined in test rule sets visible to the current operator. Each test activity found is executed and the result of each test run is tabulated and reported to the user in a popup screen. The report contains the following information:

- Number of test activities found and executed
- Number of failures
- Number of successes
- Duration of entire test-run
- The test status (success or failure), test-activity name, test class, ruleset and message of each executed test activity

UnitTest-Assert.AssertTrue

Throws a `java.lang.RuntimeException` if the inputted named when rule evaluates to “False.”

UnitTest-Assert.AssertFalse

Throws a `java.lang.RuntimeException` if the inputted named when rule evaluates to “True.”

UnitTest-Assert.AssertTrue_prop

Throws a `java.lang.RuntimeException` if the inputted Boolean parameter is false.

UnitTest-Assert.AssertFalse_prop

Throws a `java.lang.RuntimeException` if the inputted Boolean parameter is true.

UnitTest-Assert.AssertEquals_str

Throws a `java.lang.RuntimeException` if the two inputted strings (`param.aExpected`, `param.aActual`) are not equal.

UnitTest-Assert.AssertEquals_int

Throws a `java.lang.RuntimeException` if the two inputted integers (`param.aExpected`, `param.aActual`) are not equal.

UnitTest-Assert.AssertEquals_dbl

Throws a `java.lang.RuntimeException` if the two inputted doubles (`param.aExpected`, `param.aActual`) are not equal.

UnitTest-Assert.AssertEquals_DecisionTree

Throws a `java.lang.RuntimeException` if the return value from the inputted named `Rule-Declare-DecisionTree` does not match `param.aExpected`.

UnitTest-Assert.AssertEquals_DecisionTable

Throws a `java.lang.RuntimeException` if the return value from the inputted named `Rule-Declare-DecisionTable` does not match `param.aExpected`.

UnitTest-Assert.AssertEquals_MapValue

Throws a `java.lang.RuntimeException` if the return value from the inputted named `Rule-Obj-MapValue` does not match `param.aExpected`.

UnitTest-Assert.Fail

Throws a `java.lang.RuntimeException` with the inputted message (`param.aMessage`). This is useful if an activity is meant to throw an exception given a set of inputs at a certain step; the step after this would invoke this activity causing a failure; the reason being, the UnitTest-Assert.Fail invocation shouldn't be reached if the preceding step threw the exception it was supposed to.

UnitTest-Assert.AssertGreaterThan_dbl

Throws a `java.lang.RuntimeException` if the inputted double value, `param.aValue` is greater than `param.aGreaterThanValue`.

UnitTest-Assert.AssertGreaterThan_int

Throws a `java.lang.RuntimeException` if the inputted integer value, `param.aValue` is greater than `param.aGreaterThanValue`.

UnitTest-Assert.AssertGreaterThanEqualTo_dbl

Throws a `java.lang.RuntimeException` if the inputted double value, `param.aValue` is greater than or equal to `param.aGreaterThanValue`.

UnitTest-Assert.AssertGreaterThanEqualTo_int

Throws a `java.lang.RuntimeException` if the inputted integer value, `param.aValue` is greater than or equal to `param.aGreaterThanValue`.

UnitTest-Assert.AssertLessThan_dbl

Throws a `java.lang.RuntimeException` if the inputted double value, `param.aValue` is less than `param.aLessThanValue`.

UnitTest-Assert.AssertLessThan_int

Throws a `java.lang.RuntimeException` if the inputted integer value, `param.aValue` is less than `param.aLessThanValue`.

UnitTest-Assert.AssertLessThanEqualTo_dbl

Throws a `java.lang.RuntimeException` if the inputted double value, `param.aValue` is less than or equal to `param.aLessThanValue`.

UnitTest-Assert.AssertLessThanEqualTo_int

Throws a `java.lang.RuntimeException` if the inputted integer value, `param.aValue` is less than or equal to `param.aLessThanValue`.

UnitTest-Assert.AssertPageExists

Throws a `java.lang.RuntimeException` if inputted named page does not exist.

UnitTest-Assert.AssertPageNotExists

Throws a `java.lang.RuntimeException` if inputted named page does exist.

UnitTest-Assert.AssertHasMessages

Throws a `java.lang.RuntimeException` if the primary page has no messages on it. Useful for testing validation rules.

UnitTest-Assert.AssertHasNoMessages

Throws a `java.lang.RuntimeException` if the primary page has messages on it. Useful for testing validation rules.

UnitTest-Assert.AssertFalse_prop

Throws a `java.lang.RuntimeException` if the inputted Boolean parameter is true.

UnitTest-Assert.AssertTrue_prop

Throws a `java.lang.RuntimeException` if the inputted Boolean parameter is false.

UnitTest-Assert.AssertNull_str

Throws a `java.lang.RuntimeException` if the inputted string parameter is not null.

UnitTest-Assert.AssertNotNull_str

Throws a `java.lang.RuntimeException` if the inputted string parameter is null.

UnitTest-Assert.AssertNull_PgProp

Throws a `java.lang.RuntimeException` if the named embedded page property is not null.

UnitTest-Assert.AssertNotNull_PgProp

Throws a `java.lang.RuntimeException` if the named embedded page property is null.

Connect SQL

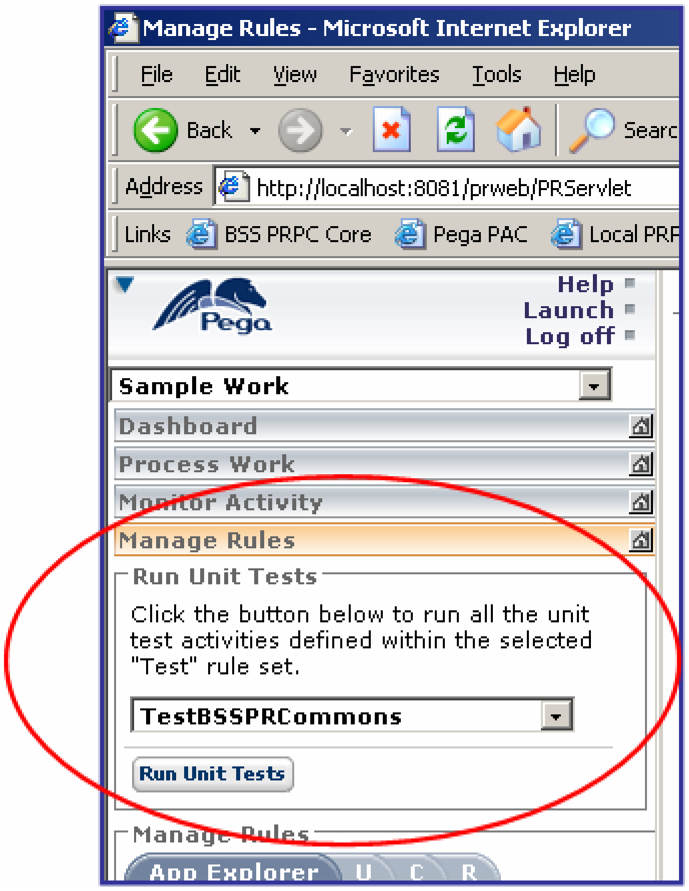
UnitTest-TestActivity.GetTestActivities

The "`Browse`" tab is populated with a query to retrieve all the activities stored in the rule base that begin with the string: "UT_" in a rule set starting with the string: "`Test`." The decision to use a Rule-Connect-SQL instead of an Obj-List invocation was due to the need of tightly controlling the query issued to the rule base in order to pull back the test activities. It is not possible using an Obj-List to specify that a "like" operand be used in the generated SQL.

Portal-HTML (v4.2 systems)

Data-Gadget.RunUnitTests

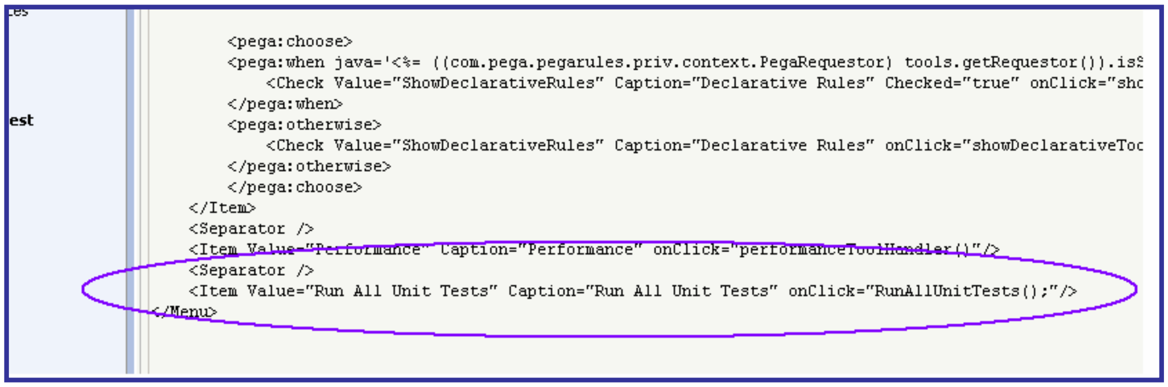
Portal interface for executing all of the unit test activities in the selected test rule set. The drop-down is populated with all of the "Test" rule sets accessible to the current operator (stored in UnitTest-Portal – 4.2 systems only).


Manage Rules - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Searc
Address http://localhost:8081/prweb/PRServlet
Links BSS PRPC Core Pega PAC Local PRP
Pega
Help
Launch
Log off
Sample Work
Dashboard
Process Work
Monitor Activity
Manage Rules
Run Unit Tests

Click the button below to run all the unit test activities defined within the selected "Test" rule set.

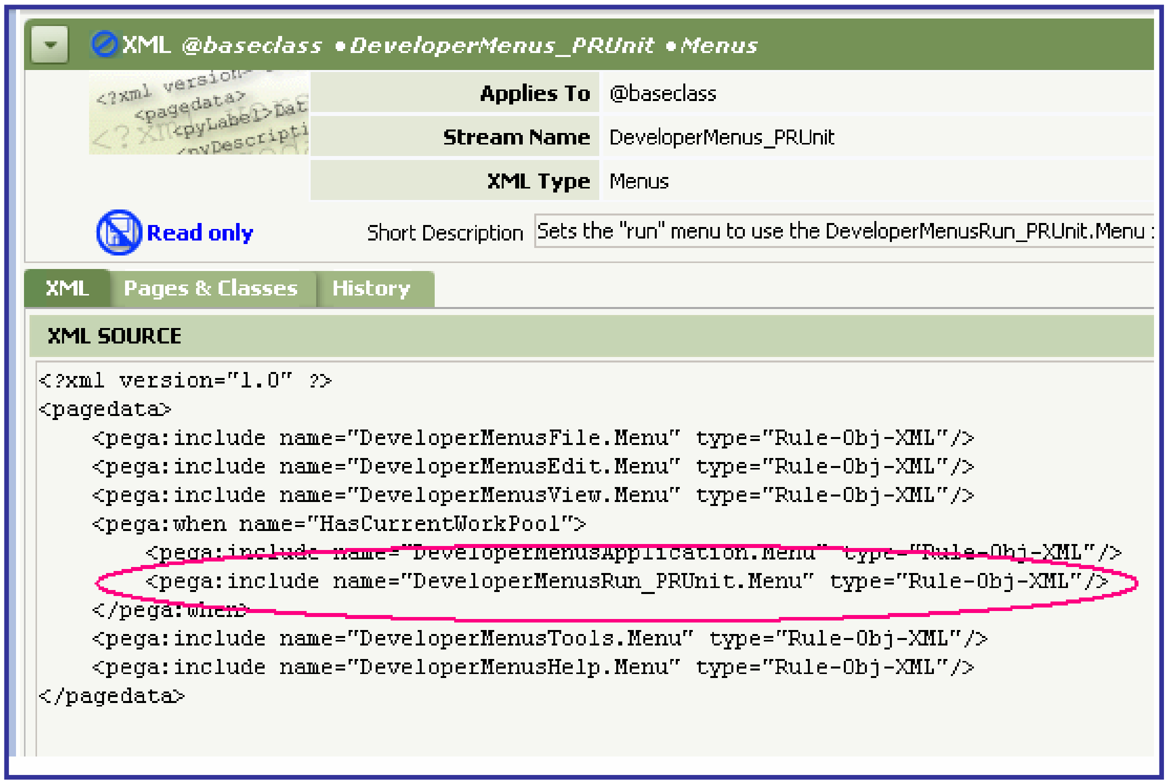
Menu (v5 systems)

@baseclasss.DeveloperMenusRun_PRUnit (Rule-Obj-XML)



```
        <pega:choose>
        <pega:when java='<%= ((com.pega.pegarules.priv.context.PegaRequestor) tools.getRequestor()).is
            <Check Value="ShowDeclarativeRules" Caption="Declarative Rules" Checked="true" onClick="sho
        </pega:when>
        <pega:otherwise>
            <Check Value="ShowDeclarativeRules" Caption="Declarative Rules" onClick="showDeclarativeToo
        </pega:otherwise>
        </pega:choose>
    </Item>
    <Separator />
    <Item Value="Performance" Caption="Performance" onClick="performanceToolHandler()"/>
    <Separator />
    <Item Value="Run All Unit Tests" Caption="Run All Unit Tests" onClick="RunAllUnitTests();"/>
</Menu>
```

@baseclass.DeveloperMenus_PRUnit (Rule-Obj-XML)



XML @baseclass • DeveloperMenus_PRUnit • Menus

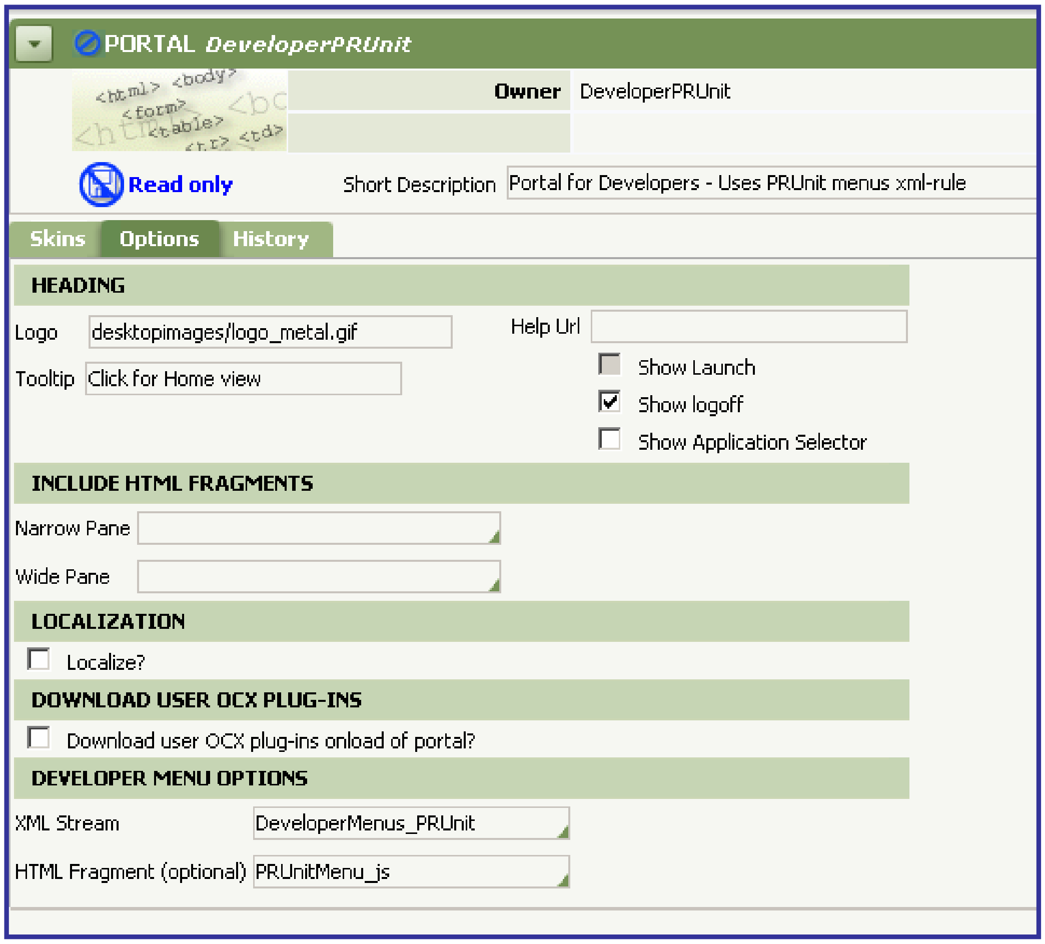
Applies To	@baseclass
Stream Name	DeveloperMenus_PRUnit
XML Type	Menus

Read only

Short Description: Sets the "run" menu to use the DeveloperMenusRun_PRUnit.Menu

```
<?xml version="1.0" ?>
<pagedata>
    <pega:include name="DeveloperMenusFile.Menu" type="Rule-Obj-XML"/>
    <pega:include name="DeveloperMenusEdit.Menu" type="Rule-Obj-XML"/>
    <pega:include name="DeveloperMenusView.Menu" type="Rule-Obj-XML"/>
    <pega:when name="HasCurrentWorkPool">
        <pega:include name="DeveloperMenusApplication.Menu" type="Rule-Obj-XML"/>
        <pega:include name="DeveloperMenusRun_PRUnit.Menu" type="Rule-Obj-XML"/>
    </pega:when>
    <pega:include name="DeveloperMenusTools.Menu" type="Rule-Obj-XML"/>
    <pega:include name="DeveloperMenusHelp.Menu" type="Rule-Obj-XML"/>
</pagedata>
```

Here is a snapshot of the “Options” tab of the DeveloperPRUnit portal rule that comes with the framework on V5 systems:


PORTAL DeveloperPRUnit

Owner DeveloperPRUnit

Read only

Short Description Portal for Developers - Uses PRUnit menus xml-rule

Skins Options History

HEADING

Logo desktopimages/logo_metal.gif

Help Url

Tooltip Click for Home view

- [ ] Show Launch
- [x] Show logoff
- [ ] Show Application Selector

INCLUDE HTML FRAGMENTS

Narrow Pane

Wide Pane

LOCALIZATION

- [ ] Localize?

DOWNLOAD USER OCX PLUG-INS

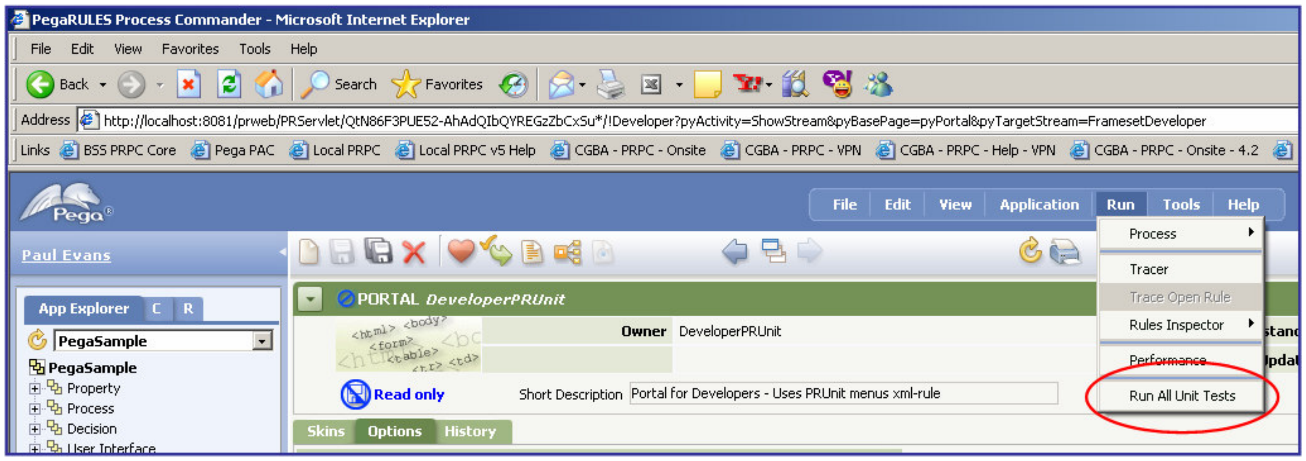
- [ ] Download user OCX plug-ins onload of portal?

DEVELOPER MENU OPTIONS

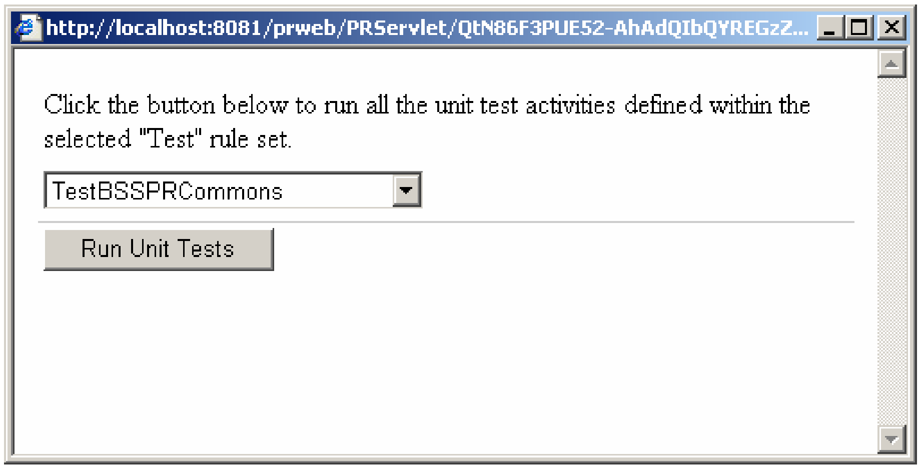
XML Stream DeveloperMenus_PRUnit

HTML Fragment (optional) PRUnitMenu_js

And here is the “Run All Unit Tests” menu item:


PegaRULES Process Commander - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/prweb/PRServlet/QtN86F3PUE52-AhAdQIbQYREGzZbCxSu*/!Developer?pyActivity=ShowStream&pyBasePage=pyPortal&pyTargetStream=FramesetDeveloper
Links BSS PRPC Core Pega PAC Local PRPC Local PRPC v5 Help CGBA - PRPC - Onsite CGBA - PRPC - VPN CGBA - PRPC - Help - VPN CGBA - PRPC - Onsite - 4.2
File Edit View Application Run Tools Help
Paul Evans
App Explorer C R
PegaSample
Property
Process
Decision
User Interface
Process
Tracer
Trace Open Rule
Rules Inspector
Performance
Run All Unit Tests

When this menu item is selected, the following popup appears to allow you to select the test rule set (of “All”) and run the tests:


http://localhost:8081/prweb/PRServlet/QtN86F3PUE52-AhAdQIbQYREGzZ...

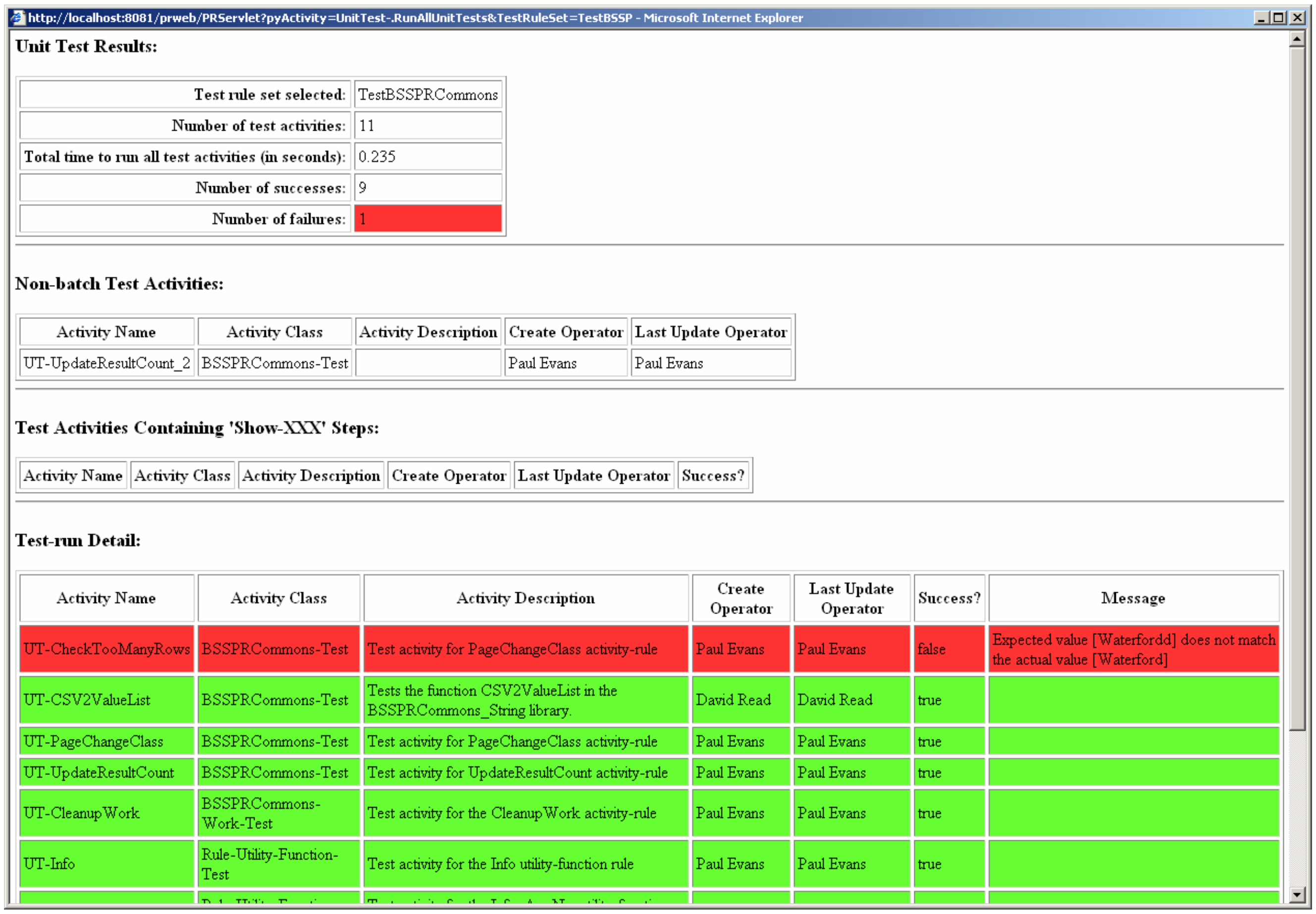
Click the button below to run all the unit test activities defined within the selected "Test" rule set.

TestBSSPRCommons

UnitTest-.TestResults

Rule-Obj-HTML rule displaying the results of the executed test activities. Each row of the test results is click-able; clicking a row opens the test activity. There's also a section that displays those unit test activities that contain any active (non commented-out) steps that perform a "Show-***" method. For example, if a test activity contains a Show-HTML, Show-Page, Show-Harness, etc, method, it can interrupt the processing of the batch unit test execution. Those test activity activities that contains a Show-*** step will not be included in the batch-execution and will display as a separate section in the report so that they can easily be identified and corrected.

Example with 1 error in the unit test results:



http://localhost:8081/prweb/PRServlet?pyActivity=UnitTest-.RunAllUnitTests&TestRuleSet=TestBSSP - Microsoft Internet Explorer

Unit Test Results:

Test rule set selected:	TestBSSPRCommons
Number of test activities:	11
Total time to run all test activities (in seconds):	0.235
Number of successes:	9
Number of failures:	1

Non-batch Test Activities:

Activity Name	Activity Class	Activity Description	Create Operator	Last Update Operator
UT-UpdateResultCount_2	BSSPRCommons-Test		Paul Evans	Paul Evans

Test Activities Containing 'Show-XXX' Steps:

Activity Name	Activity Class	Activity Description	Create Operator	Last Update Operator	Success?

Test-run Detail:

Activity Name	Activity Class	Activity Description	Create Operator	Last Update Operator	Success?	Message
UT-CheckTooManyRows	BSSPRCommons-Test	Test activity for PageChangeClass activity-rule	Paul Evans	Paul Evans	false	Expected value [Waterfordd] does not match the actual value [Waterford]
UT-CSV2ValueList	BSSPRCommons-Test	Tests the function CSV2ValueList in the BSSPRCommons_String library.	David Read	David Read	true	
UT-PageChangeClass	BSSPRCommons-Test	Test activity for PageChangeClass activity-rule	Paul Evans	Paul Evans	true	
UT-UpdateResultCount	BSSPRCommons-Test	Test activity for UpdateResultCount activity-rule	Paul Evans	Paul Evans	true	
UT-CleanupWork	BSSPRCommons-Work-Test	Test activity for the CleanupWork activity-rule	Paul Evans	Paul Evans	true	
UT-Info	Rule-Utility-Function-Test	Test activity for the Info utility-function rule	Paul Evans	Paul Evans	true	

The report consists of the following 4 sections:

Unit Test Results

Displays some metadata about the test run including: the name of the "Test" rule set selected, the number test activities found, the total duration of the test run, the number of successes and failures.

Non-batch Test Activities

Displays the number test activities found marked as "[non-batch]." These activities were not executed in the test run, but are displayed in the section for informational purposes. Each row is clickable and will open the unit test activity.

Test Activities Containing 'Show-XXX' Steps

This section displays the test activities found but that contain "Show" step(s). Test activities found with steps that invoke a show method (e.g. Show-Page, Show-HTML, Show-Harness, etc) are considered failed tests. It is "illegal" for unit test activities to invoke a "Show" method. Each row is clickable and will open the unit test activity.

Test-run Detail

This section displays the result of executing each unit test activity. Each row is clickable and will open the unit test activity.

Usage Overview

"Test" Rule sets

For each application rule set, create a corresponding "`Test`" rule set. For example, for an application rule set: "`Acme`," create a test rule set: "`TestAcme`." The "Test" rule set should have as its prerequisite the target rule set along with the "`UnitTest`" rule set. For example, the "TestAcme" rule set should list as its prerequisite, "Acme:01-01." Once created, edit the TestAcme rule set version adding to its required-list the UnitTest:01-01 rule set version. This is required so that test activities defined within the TestAcme rule set can invoke the UnitTest-Assert.AssertXXX activities defined in the UnitTest rule set. Be sure to list all "Test" rule sets in the appropriate access groups.

By storing all test-related rules in "Test" rule sets, they can easily be omitted from builds bound for QA and production environments. In addition access groups can be created if desired that do not include the "Test" rule sets in order to give a "clean" view of the rule base without also viewing the unit test class structures.

"Test" Classes

For each class that contains unit-testable rule(s), create a corresponding "`-Test`" sub-class to house the unit test activities. For example, given a class: `Acme-Data-Vehicle`, create a test class: `Acme-Data-Vehicle-Test` in the "`TestAcme`" rule set with pattern-inheritance disabled and the directed-parent set to `@baseclass`. Creating test classes to store the test activities is not required to use the framework, but it is recommended.

"UT_" Activities

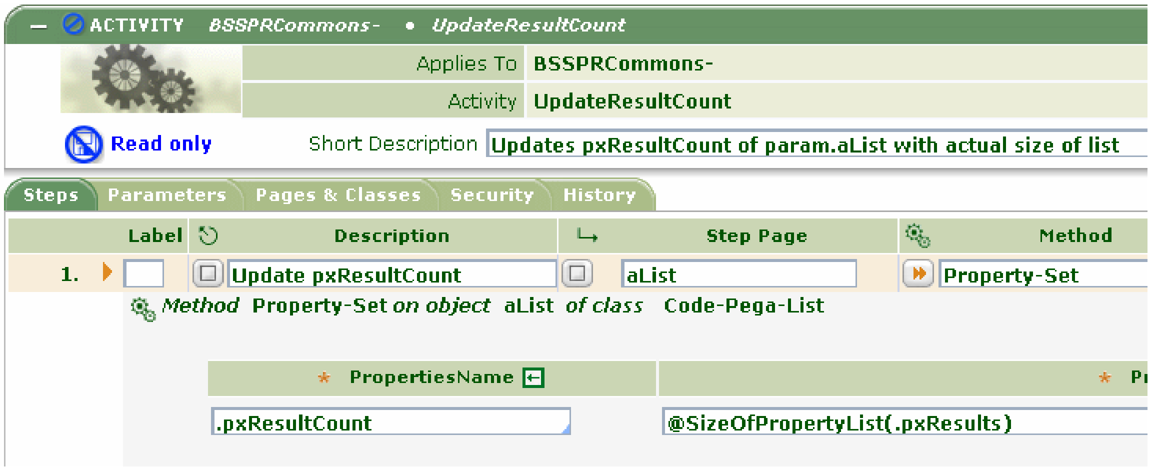
For each unit-testable rule, create a test activity of the form: "`UT_`*`RULETYPE_RULETESTED`*" where ***RULETYPE*** is the type of rule being test and *`RULETESTED`* is the name of the rule being tested (the prefix UT- is also allowed but is deprecated). For example, for a `Rule-Obj-When` rule named: `IsApproved` defined in the `Acme-Work-CalculatePremium` class, you would create a unit test activity to exercise the paths through the when rules' boolean expression:

```
Acme-Work-CalculatePremium-Test.UT_When_IsApproved
```

This activity should be saved in the "`TestAcme`" rule set.

Examples:

Below is a very simple activity that takes in a `Code-Pega-List` as input and updates its `pxResultCount` property to reflect the actual length of its `pxResults` pagelist. Many times an activity needs to manually filter the `pxResults` pagelist of a `Code-Pega-List` page using a `Java` step or `Page-Remove` methods; the `pxResultCount` is not automatically updated as elements are added or removed to the `pxResults` pagelist property. This activity ensures the `param.aList.pxResultCount` property reflects the actual size of `param.aList.pxResults`:



– ACTIVITY *BSSPRCommons- • UpdateResultCount*

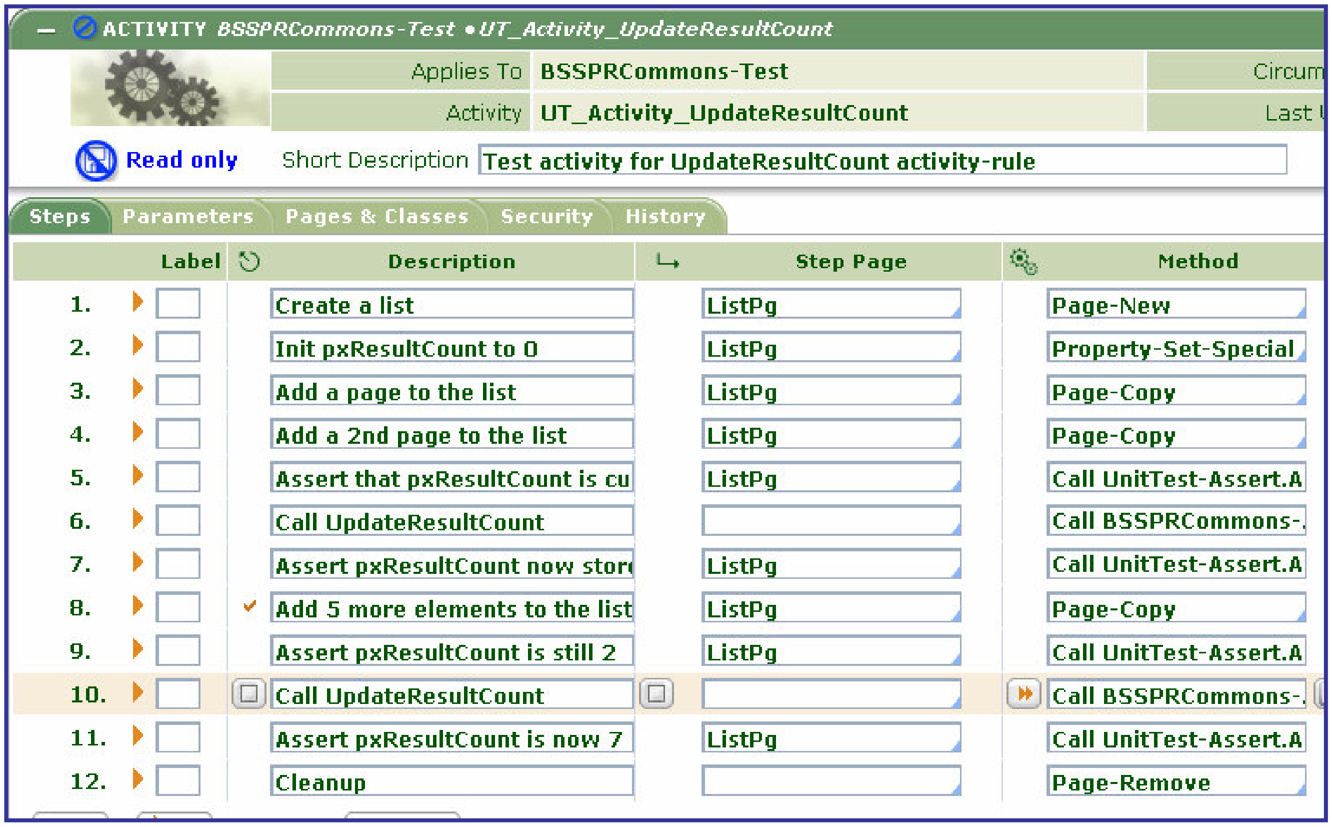
Applies To BSSPRCommons-
Activity UpdateResultCount
Read only
Short Description Updates pxResultCount of param.aList with actual size of list

Label	Description	Step Page	Method
1.	Update pxResultCount	aList	Property-Set

Method Property-Set *on object* aList *of class* Code-Pega-List

* PropertiesName	* Pr
.pxResultCount	@SizeOfPropertyList(.pxResults)

The following is a unit test activity to test the above activity using the `UnitTest` framework. Note that the above activity being tested is defined in the `BSSPRCommons-` class; although the ruleset is not displayed, it is saved in the `BSSPRCommons` rule set. The following unit test activity is defined in the `BSSPRCommons-Test` class within the `TestBSSPRCommons` rule set. Both the `BSSPRCommons-`Test class and unit test activity are saved in the `TestBSSPRCommons` rule set.



ACTIVITY BSSPRCommons-Test • UT_Activity_UpdateResultCount

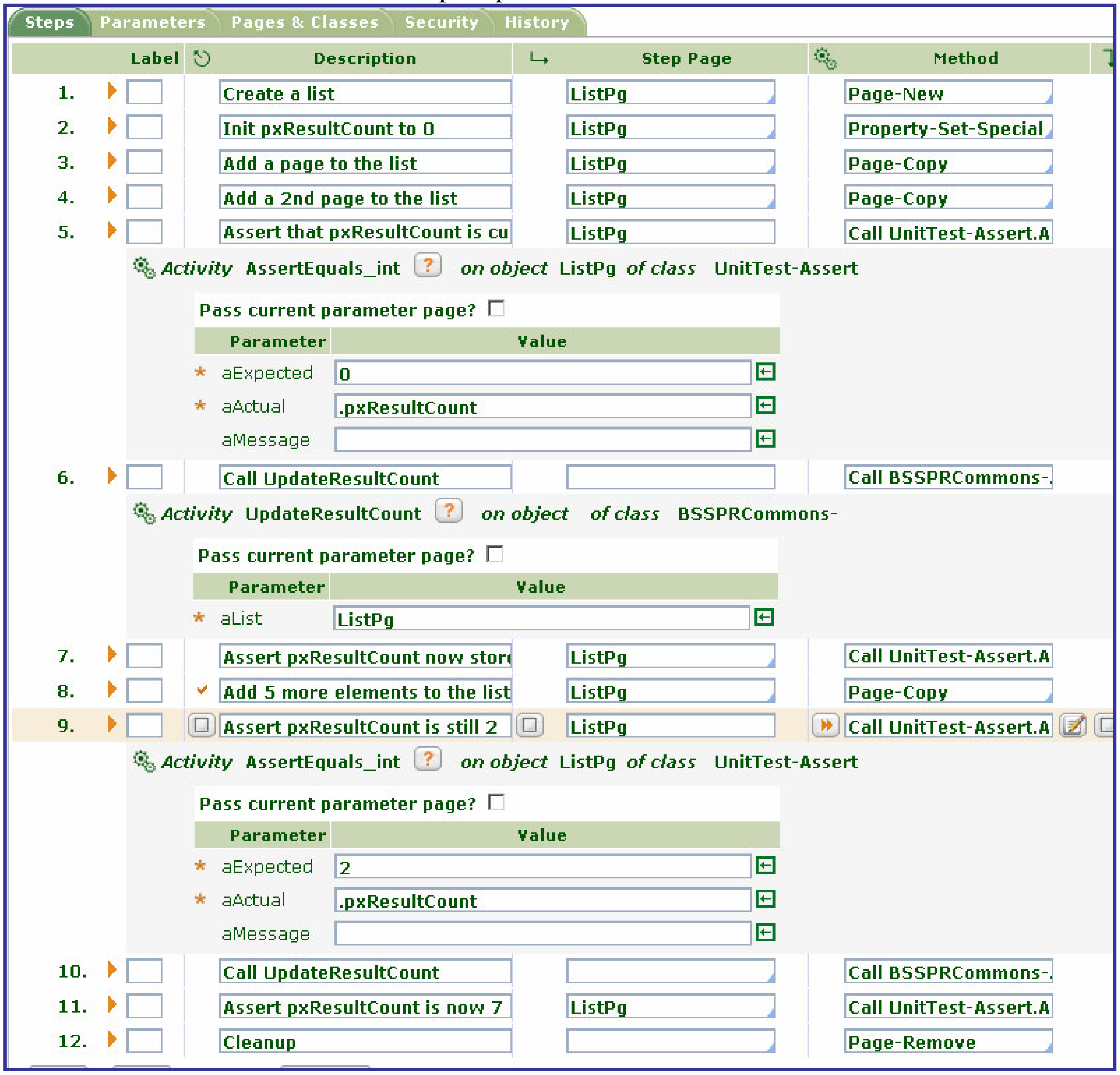
Applies To	BSSPRCommons-Test	Circum
Activity	UT_Activity_UpdateResultCount	Last

Read only

Short Description: Test activity for UpdateResultCount activity-rule

	Label	Description	Step Page	Method
1.		Create a list	ListPg	Page-New
2.		Init pxResultCount to 0	ListPg	Property-Set-Special
3.		Add a page to the list	ListPg	Page-Copy
4.		Add a 2nd page to the list	ListPg	Page-Copy
5.		Assert that pxResultCount is cu	ListPg	Call UnitTest-Assert.A
6.		Call UpdateResultCount		Call BSSPRCommons-.
7.		Assert pxResultCount now stor	ListPg	Call UnitTest-Assert.A
8.		✓ Add 5 more elements to the list	ListPg	Page-Copy
9.		Assert pxResultCount is still 2	ListPg	Call UnitTest-Assert.A
10.		Call UpdateResultCount		Call BSSPRCommons-.
11.		Assert pxResultCount is now 7	ListPg	Call UnitTest-Assert.A
12.		Cleanup		Page-Remove

Another screenshot with a few of the steps expanded:



	Label	Description	Step Page	Method
1.		Create a list	ListPg	Page-New
2.		Init pxResultCount to 0	ListPg	Property-Set-Special
3.		Add a page to the list	ListPg	Page-Copy
4.		Add a 2nd page to the list	ListPg	Page-Copy
5.		Assert that pxResultCount is cu	ListPg	Call UnitTest-Assert.A

Activity AssertEquals_int *on object* ListPg *of class* UnitTest-Assert

Pass current parameter page? ☐

Parameter	Value
* aExpected	0
* aActual	.pxResultCount
aMessage	

	Label	Description	Step Page	Method
6.		Call UpdateResultCount		Call BSSPRCommons-.

Activity UpdateResultCount *on object* *of class* BSSPRCommons-

Pass current parameter page? ☐

Parameter	Value
* aList	ListPg

	Label	Description	Step Page	Method
7.		Assert pxResultCount now stor	ListPg	Call UnitTest-Assert.A
8.		Add 5 more elements to the list	ListPg	Page-Copy
9.		Assert pxResultCount is still 2	ListPg	Call UnitTest-Assert.A

Activity AssertEquals_int *on object* ListPg *of class* UnitTest-Assert

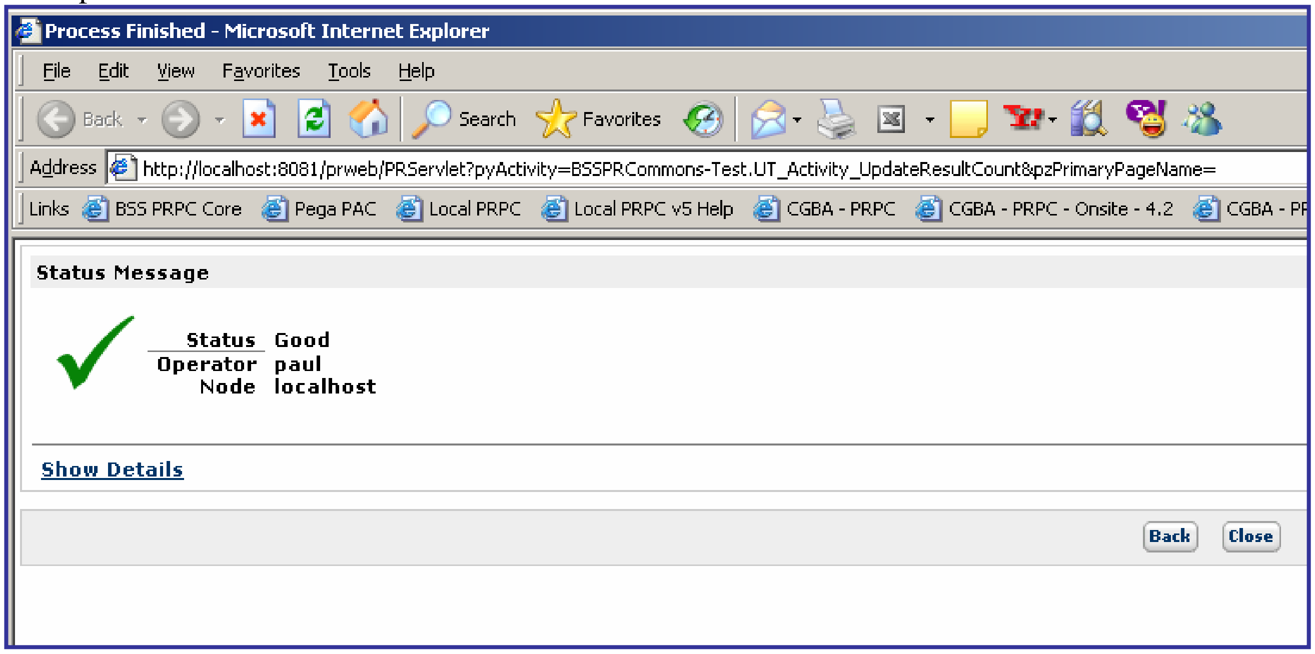
Pass current parameter page? ☐

Parameter	Value
* aExpected	2
* aActual	.pxResultCount
aMessage	

	Label	Description	Step Page	Method
10.		Call UpdateResultCount		Call BSSPRCommons-.
11.		Assert pxResultCount is now 7	ListPg	Call UnitTest-Assert.A
12.		Cleanup		Page-Remove

Notice that in this test activity, the UpdateResultCount activity is fairly well tested. If any of the steps that calls an Assert-activity (step #s: 3, 5, 8, 9, 11, 13) throws an exception, the test fails. Note the UnitTest-Assert.AssertXXX activities follow the same pattern as the JUnit assert-methods; each Assert-activity receives as input an expected-value along with an actual value (or some derivative thereof); if the 2 inputs don't match, a runtime exception is thrown with an appropriate message effectively failing the test. Individual test activities such as this one can be executed directly by clicking the ">" (play) button on the activity rule form, or indirectly by clicking the "Run All Tests" button from the "Run Unit Tests" portal-gadget. The following screenshots illustrate this activity being invoked directly; showing a success-run and a failure-run:

Example of the succeeded unit test:



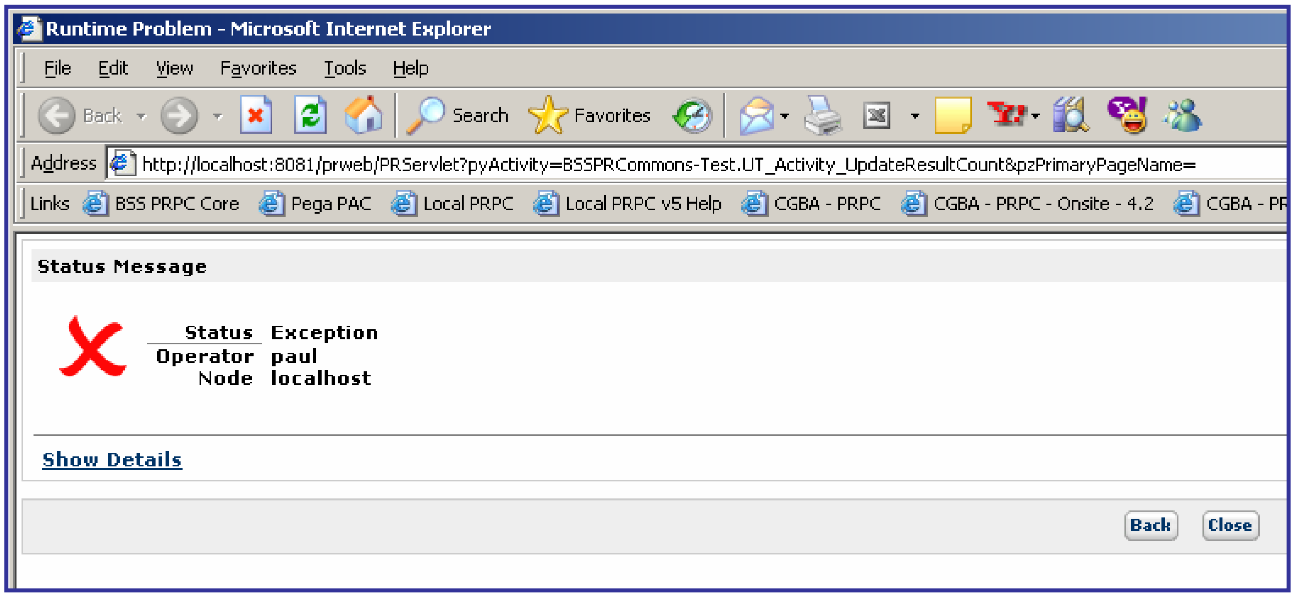
Process Finished - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/prweb/PRServlet?pyActivity=BSSPRCommons-Test.UT_Activity_UpdateResultCount&pzPrimaryPageName=
Links BSS PRPC Core Pega PAC Local PRPC Local PRPC v5 Help CGBA - PRPC CGBA - PRPC - Onsite - 4.2 CGBA - PR

Status Message

Status	Good
Operator	paul
Node	localhost

Show Details

Example of the failed unit test after intentionally introducing a bug in the activity being tested (details collapsed):



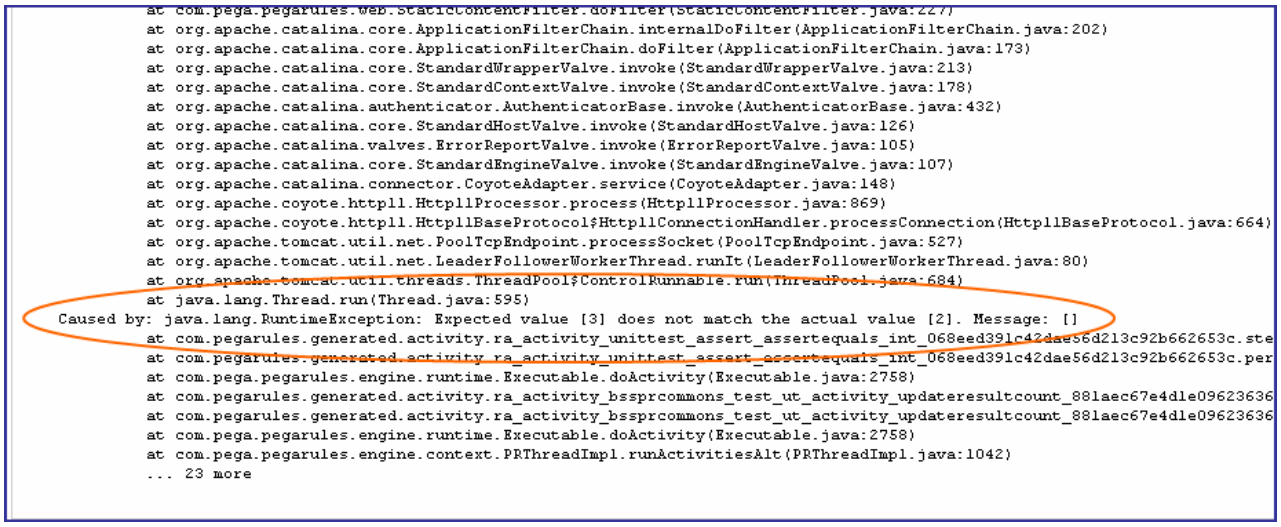
Runtime Problem - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/prweb/PRServlet?pyActivity=BSSPRCommons-Test.UT_Activity_UpdateResultCount&pzPrimaryPageName=
Links BSS PRPC Core Pega PAC Local PRPC Local PRPC v5 Help CGBA - PRPC CGBA - PRPC - Onsite - 4.2 CGBA - PR

Status Message

Status	Exception
Operator	paul
Node	localhost

Show Details

Failure screen with the detail expanded – notice that the cause of the failure is clearly visible in the "Caused by" section:



```
            at com.pega.pegarules.web.StaticContentFilter.doFilter(StaticContentFilter.java:227)
            at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:202)
            at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:173)
            at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:213)
            at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:178)
            at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:432)
            at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:126)
            at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:105)
            at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:107)
            at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:148)
            at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:869)
            at org.apache.coyote.http11.Http11BaseProtocol$Http11ConnectionHandler.processConnection(Http11BaseProtocol.java:664)
            at org.apache.tomcat.util.net.PoolTcpEndpoint.processSocket(PoolTcpEndpoint.java:527)
            at org.apache.tomcat.util.net.LeaderFollowerWorkerThread.runIt(LeaderFollowerWorkerThread.java:80)
            at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:684)
            at java.lang.Thread.run(Thread.java:595)
   Caused by: java.lang.RuntimeException: Expected value [3] does not match the actual value [2]. Message: []
            at com.pegarules.generated.activity.ra_activity_unittest_assert_assertequals_int_068eed391c42dae56d213c92b662653c.ste
            at com.pegarules.generated.activity.ra_activity_unittest_assert_assertequals_int_068eed391c42dae56d213c92b662653c.per
            at com.pega.pegarules.engine.runtime.Executable.doActivity(Executable.java:2758)
            at com.pegarules.generated.activity.ra_activity_bssprcommons_test_ut_activity_updateresultcount_881aec67e4d1e09623636
            at com.pegarules.generated.activity.ra_activity_bssprcommons_test_ut_activity_updateresultcount_881aec67e4d1e09623636
            at com.pega.pegarules.engine.runtime.Executable.doActivity(Executable.java:2758)
            at com.pega.pegarules.engine.context.PRThreadImpl.runActivitiesAlt(PRThreadImpl.java:1042)
            ... 23 more
```

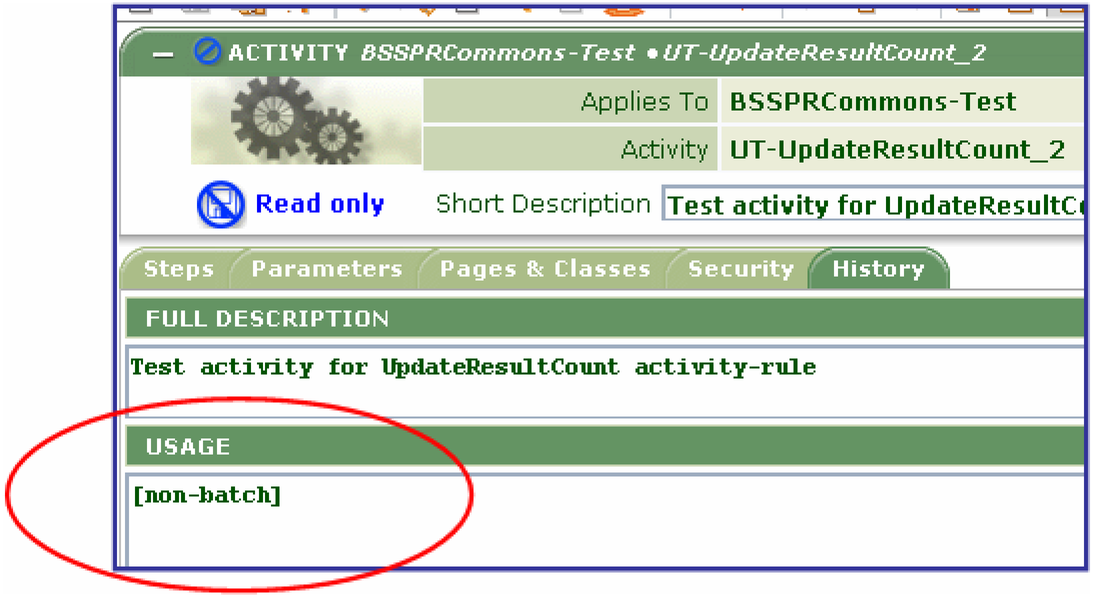
You'll from these screenshots that it is immediately known to the user running the test if the test succeeded or failed. In the event of a failure, the reason for the failure is easily determined making it as "easy" as possible to track down the source of the bug.

Don't Create the Primary Page

Unit test activities should never create the primary page, nor should they create a named-clipboard class of the class they are defined in (i.e. never create a named clipboard page of type Acme-Data-Company-Test). Instead the first step of most every test activity should be a Page-New specifying a named page where the class type should be the class that is currently being tested. For example, an activity: UT_Activity_CalculatePremium defined in the CommonApps-Data-Policy-Test class should do a Page-New on PolicyPg where PolicyPg is defined on the "Pages & Classes" tab as type: CommonApps-Data-Policy. Test classes should be viewed ONLY has locations for unit test activities to be stored w/in (this is why it's recommended that pattern inheritance be turned off and directed inheritance set to @baseclass).

Marking Test Activities as Non-Batch

For those test activities that should not be executed in batch mode, include within the "Usage" field of the rule the string: "[non-batch]":



Applies To BSSPRCommons-Test

Activity UT-UpdateResultCount_2

Short Description Test activity for UpdateResultC

FULL DESCRIPTION

Test activity for UpdateResultCount activity-rule

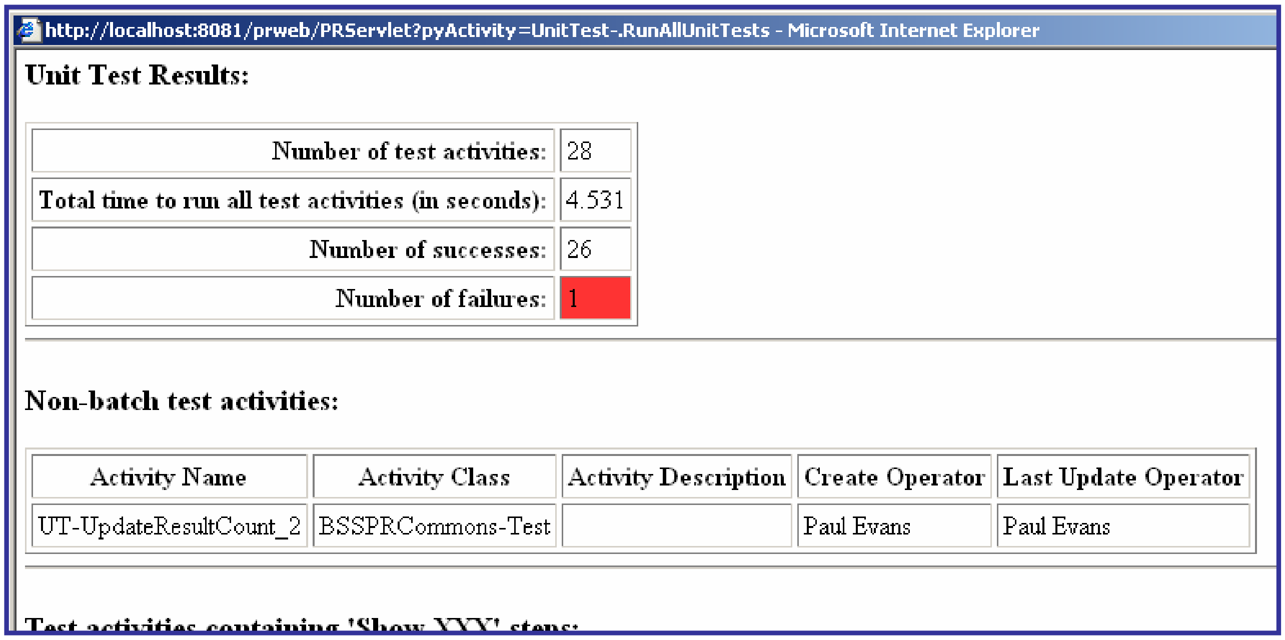
USAGE

[non-batch]

There are several reasons why you may want to mark a test activity in this way:

- The nature of the rule being tested is such that the test activity is very performance heavy and it being included in the batch run would overload the system.
- There's a bug in a test activity that is causing the instance to crash and marking it as '[non-batch]' omits it from the batch test run. This allows developers to focus on fixing it while leaving other developers the ability to continue safely executing the batch unit test run.

The batch unit test report will indicate the test activities that were omitted from the batch run (they are **not** counted as failures):



http://localhost:8081/prweb/PRServlet?pyActivity=UnitTest-.RunAllUnitTests - Microsoft Internet Explorer

Unit Test Results:

Number of test activities:	28
Total time to run all test activities (in seconds):	4.531
Number of successes:	26
Number of failures:	1

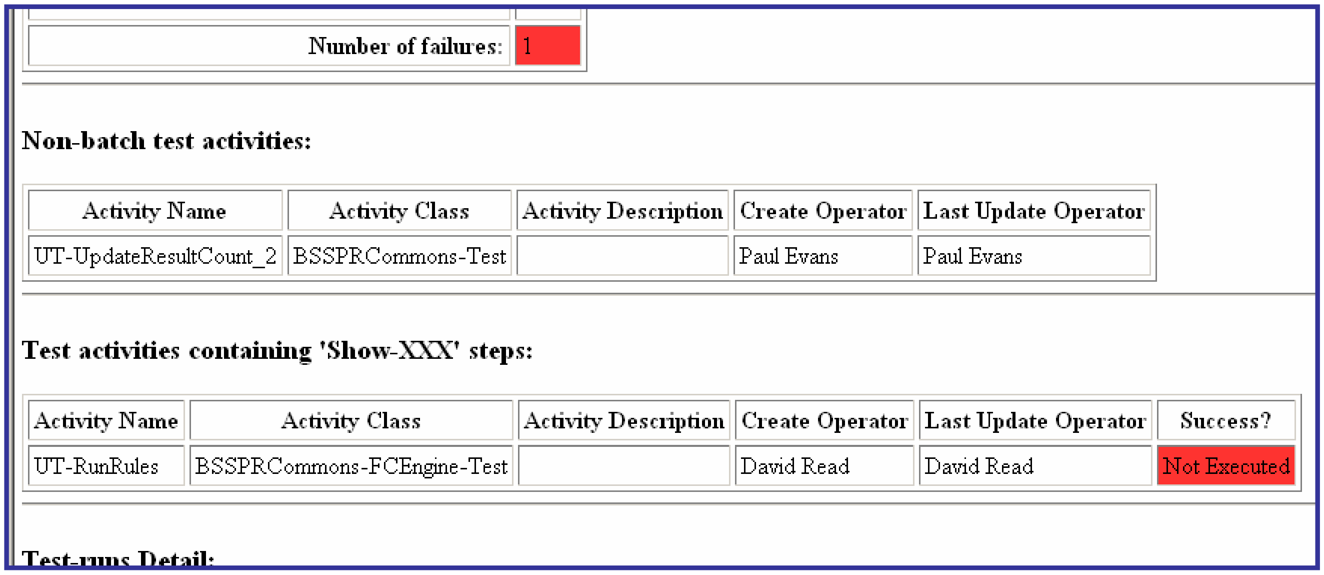
Non-batch test activities:

Activity Name	Activity Class	Activity Description	Create Operator	Last Update Operator
UT-UpdateResultCount_2	BSSPRCommons-Test		Paul Evans	Paul Evans

Test Activities Containing Show-XXX Steps

Test activities should not contain any show steps including: Show-Page, Show-HTML or Show-Harness, etc. The reason is that when a show-step is executed during a batch-run, it can interrupt the batch-run and prevent the report from displaying properly.

The batch unit test report will indicate the test activities that were omitted form the batch run due to containing "Show-XXX" step(s) (they **are** counted as failures):



Number of failures:	1

Non-batch test activities:

Activity Name	Activity Class	Activity Description	Create Operator	Last Update Operator
UT-UpdateResultCount_2	BSSPRCommons-Test		Paul Evans	Paul Evans

Test activities containing 'Show-XXX' steps:

Activity Name	Activity Class	Activity Description	Create Operator	Last Update Operator	Success?
UT-RunRules	BSSPRCommons-FCEngine-Test		David Read	David Read	Not Executed

Ideas for Enhancement

The following are ideas regarding how this framework might be enhanced:

Automated Test Run

It would be useful to have the unit tests executed in an automated fashion on a recurring cycle using an agent-queue rule. This would provide an element of continuing integration to the PRPC product-line; in addition workflow could be put in place to allow for the notification of the results of a test run to interested parties via email.

Initiate Test Run from External System

It may provide worthwhile to create service rules to allow the unit tests to be executed by some external system on demand and allow for the reporting of the test results.

Test-Code Generation

Similar to the accelerators, a rule-generation wizard that would generate the test classes and test activities given a target class. For each testable rule present in the target class, a corresponding UT-XXX activity would be created in the –Test class within the specified Test rule set.

Resources

PRUnit Homepage

http://prunit.sourceforge.net

Test-driven Development

http://www.testdriven.com/
Site dedicated to TDD

http://en.wikipedia.org/wiki/Test_driven_development
Wikipedia article about TDD

xUnit Testing Framework Pattern

http://en.wikipedia.org/wiki/XUnit
Wikipedia article about xUnit