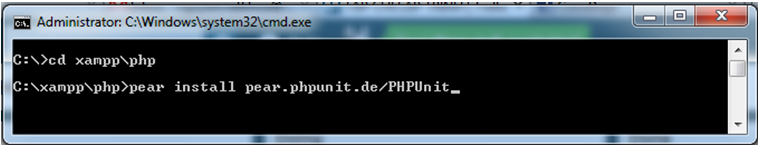
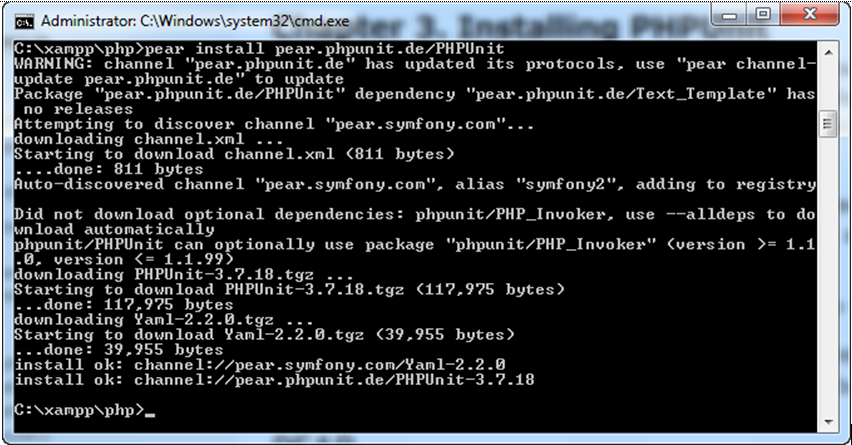
**How to Use PHPUnit**

1. **Installing PHPUnit**

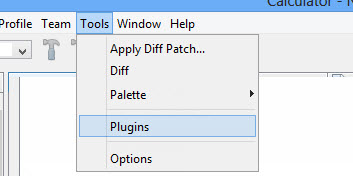
* Install XAMPP, WAMP, or any PHPWEBSERVER
* Open Command Prompt (⊞ Win+R opens the Run dialog box. Type CMD)
* Change to PHP Directory (eg: C:/Xampp/php)
* Run Command to install PHPUnit with Pear
* **pear config-set auto\_discover 1**
* **pear install pear.phpunit.de/PHPUnit**



* Install PHPUnit\_Selenium
* **pear install phpunit/PHPUnit\_Selenium**
* Install PHPUnit\_SkeletonGenerator
* **pear install phpunit/PHPUnit\_SkeletonGenerator**
* 

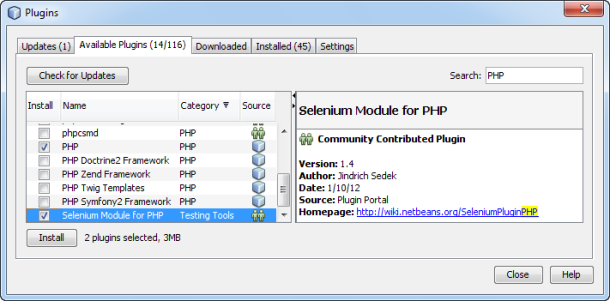
1. **PHPUnit with NetBeans**
   1. **Installing PHPUnit for NetBeans**

Open Plugins in Netbeans



**Install plugins:**

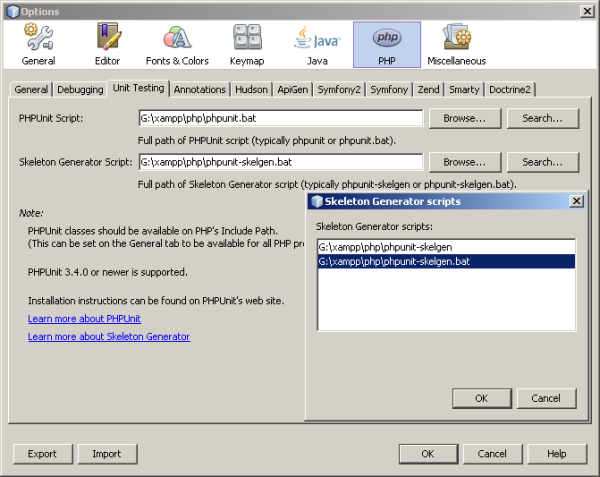
* + - * PHP
      * Selenium Module for PHP
      * PHP Documentor Tag Help
      * PHP Documentor



* 1. **Config Netbeans Options**

Netbeans > Tools > Options > tab: PHP

Open the Unit Testing tab. The paths to your PHPUnit and Skeleton Generator scripts should appear.



* 1. **Create PHPUnit**

Create a PHP project named Calculator. In this project, create a file named calculator.php. In this file, type or paste the Calculator class

<?php

class Calculator {

public function add($a, $b) {

return $a + $b;

}

}

?>

Add a comment block with the @assert annotation and some sample input and output. Note that one incorrect assertion is included in this example.

[…]

class Calculator

{

/\*\*

\* @assert (0, 0) == 0

\* @assert (0, 1) == 1

\* @assert (1, 0) == 1

\* @assert (1, 1) == 2

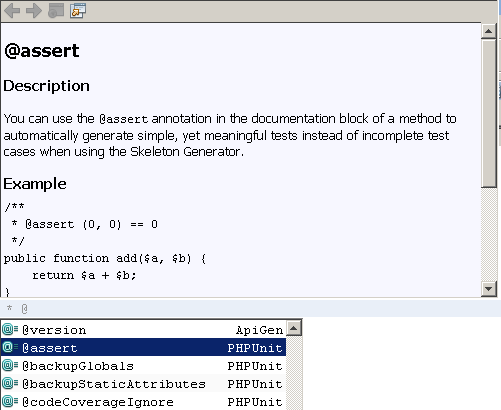
\* @assert (1, 2) == 4

\*/

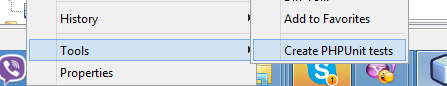
**public function add($a, $b)**

[…]

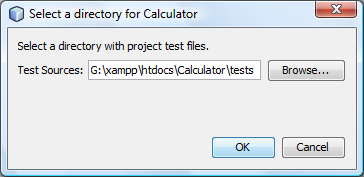
**Note:** You can use annotation code completion to add @assert annotations. Navigate between parameters with the Tab key, or click Enter after filling in a parameter value.



* In the Projects window
* Right-click the Calculator.php node
* select Tools > Create PHPUnit Tests.



* The first time you create tests, a dialog opens asking you for the directory in which you want to store test files. In this example, the Browse function was used to create a tests directory.

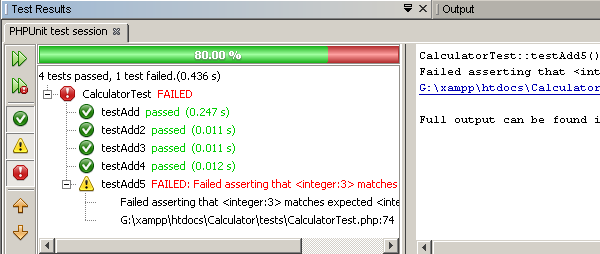


* The IDE generates a skeleton test class in a file called CalculatorTest.php
* Which appears in your Projects window and opens in the editor.

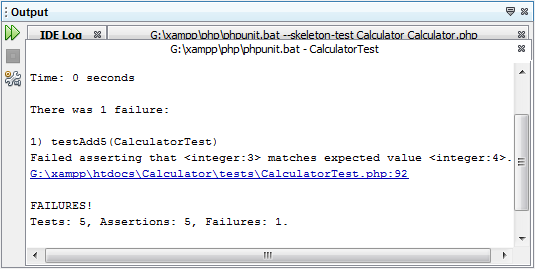
|  |  |
| --- | --- |
| Projects window showing new test class | /\*\*  \* Generated from @assert (1, 1) == 2.  \*/  public function testAdd4()  {  $this->assertEquals(  2,  $this->object->add(1, 1)  );  } |

* 1. **Run PHPUnit**

To test the project, right-click the project's parent node and select Test, or press Alt-F6. To test the Calculator.php file, right-clict the file's node and select Test, or press Ctrl-F6.

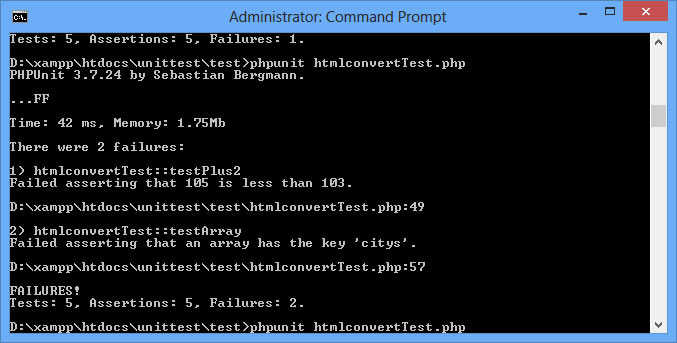
The IDE runs the tests and displays the results in the Test Results window.   


A more verbose textual version of the results is displayed in the Output window.



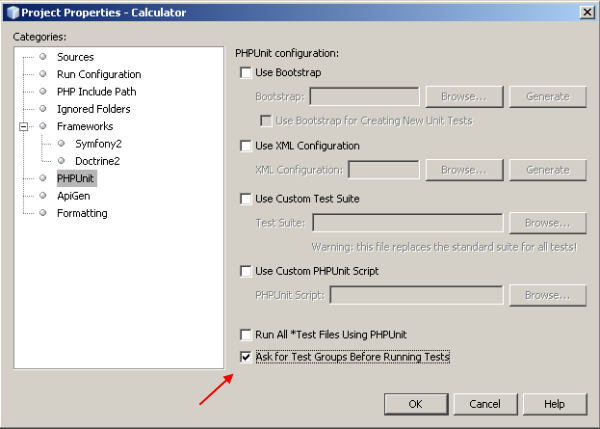
**Run PHPUnit with Command-Line Test Runner**

* Open command line and type
* C:\> phpunit UnitTest path/file.php



* 1. **Using Test Groups**

To create and run test groups:  
In the Projects window, right-click the Project node and select Properties.  
Select the PhpUnit category. Select Ask for Test Groups Before Running Tests. Click OK.

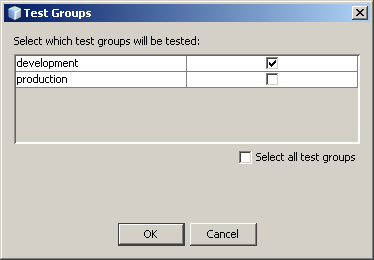


* Open CalculatorTest.php in the editor.
* For the methods **testAdd**, **testAdd3** and **testAdd5**, add the annotation @group production.
* For the methods testAdd2 and testAdd4, add the annotations @group production and @group development

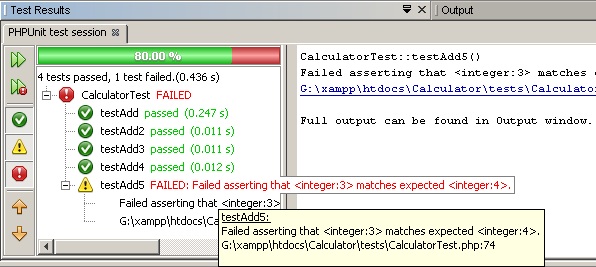
Code showing test group annotation

|  |  |
| --- | --- |
| Code showing test group annotation | Code showing test group annotations |

* Right-click the Calculator.php node and select Test. A dialog opens, asking you which test groups to run. Select "development" and click OK. The IDE only runs the tests that are annotated with @group development.

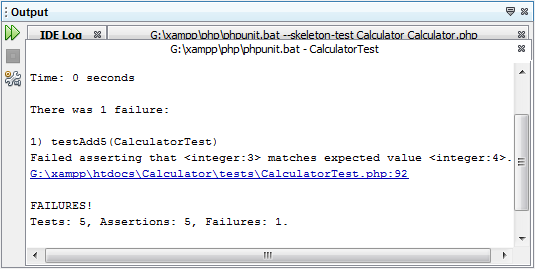


* 1. **Test Results and IDE Output**
     + - * In the Test Results window, you get information about failed tests from these locations:
         * Messages in the UI pane attached to the tree entry for the failed test
         * Text in the right-side pane, including links to the lines of test code that failed
         * Tooltip text that appears when you hover the cursor over a failed test in the UI pane

****

The Test Results window includes the following buttons on the left side:

* Rerun the test rerun button
* Show failed tests Show failed tests button
* Show passed tests Show passed tests button
* Show tests that passed but with errors Show tests with errors button
* Navigate between showing the next test result next test buttonor the previous test result previous test button



* 1. **Code Coverage**

Open Calculator.php and add a duplicate add function, called add2. The Calculator class now looks like the following:

[…]

public function add($a, $b) {

return $a + $b;

}

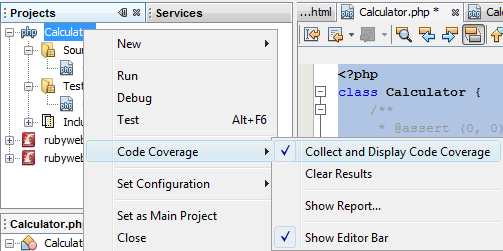
public function add2($a, $b) {

return $a + $b;

}

[…]

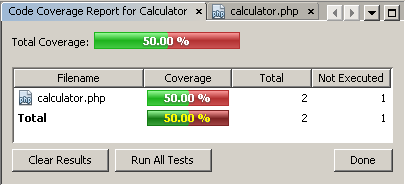
* Right-click the **project node**. From the context menu, select **Code Coverage > Collect and Display Code Coverage**. By default, Show Editor Bar is also selected.



* Click Test to test the open file or All Tests to run all tests for the project.
* Covered code is highlighted in green and uncovered code is highlighted in red



* To open Code Coverage report In the Editor Bar, click on Report.
* Code Coverage report showing the results of all tests run on your project:

****

* 1. **DRFGDFH**

1. **PHPUNIT Assertions**  
   *This section lists the various assertion methods that are available.*

* **assertArrayHasKey**(mixed $key, array $array[, string $message = ''])

Reports an error identified by $message if $array does not have the $key.

* **assertClassHasAttribute**($attributeName, string $className[, string $message = ''])

Reports an error identified by $message if $className::attributeName does not exist.

* **assertContains**(mixed $needle, Iterator|array $haystack[, string $message = ''])

Reports an error identified by $message if $needle is not an element of $haystack.

* **assertEquals**(mixed $expected, mixed $actual[, string $message = ''])

Reports an error identified by $message if the two variables $expected and $actual are not equal.

* assertSame

(mixed $expected, mixed $actual[, string $message = ''])

Reports an error identified by $message if the two variables $expected and $actual do not have the same type and value.

* assertGreaterThan

(mixed $expected, mixed $actual[, string $message = ''])

Reports an error identified by $message if the value of $actual is not greater than the value of $expected.

* assertClassHasStaticAttribute()
* [assertContainsOnly()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertContainsOnly)
* [assertContainsOnlyInstancesOf()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertContainsOnlyInstancesOf)
* [assertCount()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertCount)
* [assertEmpty()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertEmpty)
* [assertEqualXMLStructure()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertEqualXMLStructure)
* [assertFalse()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertFalse)
* [assertFileEquals()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertFileEquals)
* [assertFileExists()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertFileExists)
* [assertGreaterThan()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertGreaterThan)
* [assertGreaterThanOrEqual()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertGreaterThanOrEqual)
* [assertInstanceOf()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertInstanceOf)
* [assertInternalType()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertInternalType)
* [assertJsonFileEqualsJsonFile()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertJsonFileEqualsJsonFile)
* [assertJsonStringEqualsJsonFile()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertJsonStringEqualsJsonFile)
* [assertJsonStringEqualsJsonString()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertJsonStringEqualsJsonString)
* [assertLessThan()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertLessThan)
* [assertLessThanOrEqual()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertLessThanOrEqual)
* [assertNull()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertNull)
* [assertObjectHasAttribute()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertObjectHasAttribute)
* [assertRegExp()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertRegExp)
* [assertStringMatchesFormat()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertStringMatchesFormat)
* [assertStringMatchesFormatFile()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertStringMatchesFormatFile)
* [assertSame()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertSame)
* [assertSelectCount()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertSelectCount)
* [assertSelectEquals()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertSelectEquals)
* [assertSelectRegExp()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertSelectRegExp)
* [assertStringEndsWith()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertStringEndsWith)
* [assertStringEqualsFile()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertStringEqualsFile)
* [assertStringStartsWith()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertStringStartsWith)
* [assertTag()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertTag)
* [assertThat()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertThat)
* [assertTrue()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertTrue)
* [assertXmlFileEqualsXmlFile()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertXmlFileEqualsXmlFile)
* [assertXmlStringEqualsXmlFile()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertXmlStringEqualsXmlFile)
* [assertXmlStringEqualsXmlString()](http://phpunit.de/manual/3.7/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions.assertXmlStringEqualsXmlString)

For more information of Assertions visit: [http://phpunit.de/manual/current/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions](http://phpunit.de/manual/current/en/writing-tests-for-phpunit.html)