This post is for those who've done PHP based projects and are having trouble carrying out unit testing due to lack of of proper tutorials or guides.  
  
I've compiled and written what I learned thees past few days as some basic steps to get you started with unit testing using Composer and PHPUnit. I also added all the files (drive link) at the end. I'm sharing this because our lectures and slides only included instructions on how to work with java. Also, since many people had and have back to back exams, unlike me, they cannot afford the extra time to look into it this much and a some have already asked for this. Hope this helps at least a few people.  
  
  
  
**Setting up Composer --**

1. Copy paste this link to download executable Composer installation file : [https://getcomposer.org/Composer-Setup.exe](https://getcomposer.org/Composer-Setup.exe?fbclid=IwAR300i8AhB0O4DUUddxdFFeSYjpqW97WrWmIEqdSNqfy_iyxCACpAo23ZlY) and run it.

\* The above installer is for Windows and for Linux, Unix or MacOS check the instructions here : [https://getcomposer.org/doc/00-intro.md...](https://getcomposer.org/doc/00-intro.md?fbclid=IwAR02yEu9wnwGCAoVru-JBG_s5KHes-uq6sSIRX3_wXA1jsWsJ0tOJB0n4n0#installation-linux-unix-macos)

**Setting up PHPUnit --**

1. Create a new folder for your project, I named mine "phpunit".

2. Navigate to the created folder in CMD (Command Prompt) (for example, if your project folder is in Desktop, simply write and run "cd desktop" and then "cd your folder name" this in my case is "cd phpunit").

3. Once in the directory within CMD write the following to install PHPUnit in that particular directory "composer require phpunit/phpunit".

4. Try running PHPUnit by simply writing "phpunit" in CMD.

**Setting up the json and xml files for PHPUnit --**

1. Inside the phpunit or project folder create two more folders named "tests" and "app".

2. There will be a json file in your project folder named "composer.json" open this file and replace the codes with the following :

{

"require": {

"phpunit/phpunit": "^5.7"

},

"autoload": {

"psr-4": {

"App\\": "app"

}

}

}

3. Now, go back to CMD and write the following "composer dump-autoload -o".

4. After this, create a new file called "phpunit.xml" in the project folder and add the following code to it :

<?xml version="1.0" encoding="UTF-8"?>

<phpunit bootstrap="vendor/autoload.php"

colors="true"

verbose="true"

stopOnFailure="false">

<testsuites>

<testsuite name="Test suite">

<directory>tests</directory>

</testsuite>

</testsuites>

</phpunit>

5. In CMD check if everything is working by writing ".\vendor\bin\phpunit" it should show "No tests executed!" if it doesn't you did something wrong in the previous steps.

**Checking to see if PHPUnit can run simple tests --**

1. Create a folder named "unit" inside the "tests" folder.

2. Create a "SampleTest.php" file and add the code below then run phpunit from CMD (it should say "No tests found in class "SampleTest"." with "0 test, 0 assertion")

<?php

class SampleTest extends \PHPUnit\Framework\TestCase

{

//

}

3. Now update the code to the following and run again (this time it should show "Tests: 1" but "Assertions: 0" as we haven't asserted anything yet)

<?php

class SampleTest extends \PHPUnit\Framework\TestCase

{

public function testTrueAssertsToTrue()

{

//

}

}

4. Finally, update the code with the following and re-run (this time it should show "Tests: 1, Assertions: 1" as we tested if true equals true)

<?php

class SampleTest extends \PHPUnit\Framework\TestCase

{

public function testTrueAssertsToTrue()

{

$this->assertTrue(true);

}

}

5. Also try this by passing "false" into the assertTrue() method for a quick check. It should now also show "Failures: 1"

**Finally running an actual tests with PHPUnit --**

1. Create a "Model" folder inside "app" and a "user.php" file inside "Model" and add the following code :

<?php

namespace App\Models;

class User

{

//

}

2. Create a "UserTest.php" file inside "tests" and the following code :

<?php

class UserTest extends \PHPUnit\Framework\TestCase

{

public function testThatWeCanGetTheFirstName()

{

$user = new \App\Models\User;

$user->setFirstName('Billy');

$this->assertEquals($user->getFirstName(), 'Billy');

}

}

3. Run the code from CMD and it should show "Call to undefined method setFirstName()" as the method is not declared inside user.

4. Add the following code and run again only this time you'd get the error "Call to undefined method getFirstName()" as this method isn't added yet either.

<?php

namespace App\Models;

class User

{

public $first\_name;

public function setFirstName($firstName)

{

$this->first\_name = $firstName;

}

}

5. Add the following code and run again, notice the "return 'Billy';" line is just mocking to check if everything else is working :

<?php

namespace App\Models;

class User

{

public $first\_name;

public function setFirstName($firstName)

{

$this->first\_name = $firstName;

}

public function getFirstName()

{

return 'Billy';

}

}

6. In the above code change "return 'Billy';" to "return $this->first\_name" and run again, your test should pass this time as well and this shows that everything is working as it should.

7. Now update both the "UserTest.php" and "User.php" file to add the test and methods for the last name just as we did for the first name.

8. Run your code again after adding blank spaces to the names, like "Billy " and you will see that the test fails and the output will show that the expected and actual do not match.

9. Fixing this is part of the unit test and you can go to your set method and use the "trim()" function on the "$firstName" variable like :

public function setFirstName($firstName)

{

$this->first\_name = trim($firstName);

}

10. Run the code again and it will pass.

**Understanding setUp() methods --**

1. For the "UserTest.php" file add this code block before any of the test methods written :

protected $user;

public function setUp()

{

//the line below checks whether setUp runs before each test block.

//var\_dump('1');

$this->user = new \App\Models\User;

//the line below shows actual use case of setUp() as repeating the line below for every test is unnecessary.

//$this->user = new \App\Models\User;(['email' => 'alex@codesource.com']);

}

2. Next, replace each instance of "$user" only in the test methods with "$this->user" as $user was initilized as a global variable in the above code snippet.

Till here are the absolute basics of PHP Unit Testing using Composer and PHPUnit, do read up different documentation present online to learn ways to use different types of assertion methods, collections, etc.  
  
[https://drive.google.com/.../1TIbHFO4gyPB1PZm9bMrwLk365JI...](https://drive.google.com/drive/folders/1TIbHFO4gyPB1PZm9bMrwLk365JIQB8aJ?usp=sharing&fbclid=IwAR1cLnMkGgIk7SHpEomcr7MDiCzPpdY4QD2Obqzv_oJAOI0Hofep_EJYhLk)