5. Now we will get the following structure:

book 1 cart

— src

— tests

— .gitignore

— composer.j son

— phpunit.xml.dist

— vendor

**Writing the extension code**

Copy the Cart, Storageinterface, and Sessionstorage classes from the Unit testing with PHPUnit recipe.

Finally, we must get the following structure:

book 1 cart

— src

storage

I— SessionStorage.php I— Storageinterface.php Cart.php

— tests

— .gitignore

— composer.j son

— vendor

**Writing the extension tests**

1. Add the book/cart/tests/bootstrap.php entry script:

<?php

defined('YII\_DEBUG') or define('YII\_DEBUG', true); defined('YII\_ENV') or define('YII\_ENV', 'test');

require( DIR . '/../vendor/autoload.php');

require( DIR . '/../vendor/yiisoft/yii2/Yii.php');

2. Create a test base class by initializing the Yii application before each test and by destroying the application after ones:

<?php

namespace book\cart\tests;

use yii\di\Container;

use yii\console\Application;

use mageekguy\atoum\test;

abstract class TestCase extends test {

public function beforeTestMethod($method)

{

parent::beforeTestMethod($method);

$this->mockApplication();

} public function afterTestMethod($method)