Introduction

The idea behind testing is not new. You can't sleep well if you are not confident that your last commit didn't take down the whole application. Having your application covered with tests gives you more trust in the stability of your application. That's all.

In most cases tests don't guarantee that the application works 100% as it is supposed to. You can't predict all possible scenarios and exceptional situations for complex apps, but with tests you can cover the most important parts of your app and at least be sure they work as predicted.

There are plenty of ways to test your application. The most popular paradigm is Unit Testing (http://en.wikipedia.org/wiki/Unit_testing). For web applications, testing just the controller and/or the model doesn't prove that your application is working. To test the behavior of your application as a whole, you should write functional or acceptance tests.

Codeception supports all three testing types. Out of the box you have tools for writing unit, functional, and acceptance tests in a unified framework.

	Codeception Unit Tests	**Codeception Functional Tests**	**Codeception Acceptance Tests**
Scope of the test	Single PHP class	PHP Framework (Routing, Controllers, etc.)	Page in browser (Chrome, Firefox, or PhpBrowser)
Testing computer needs access to project's PHP files	Yes	Yes	No
Webserver required	No	No	Yes
JavaScript	No	No	Yes
Additional software required	None	None	- For WebDriver: Selenium Server or PhantomJS (deprecated) - For PhpBrowser: None
Test execution speed	High	High	Low
Configuration file	`unit.suite.yml`	`functional.suite.yml`	`acceptance.suite.yml`

One of the main advantages of Codeception is that you don't have to decide on just *one* type of testing. You can have all three! And chances are, that you will (sooner or later) need all three. That's why Codeception consists of three so-called "suites": A "unit suite" for all unit tests, a "functional suite" for all functional tests, and an "acceptance sui

acceptance tests.

Let's review those three testing types in reverse order.

Acceptance Tests

How does your client, manager, tester, or any other non-technical person know your website is working? By opening the browser, accessing the site, clicking on links, filling in the forms, and actually seeing the content on a web page. They have no idea of the programming language, framework, database, web-server, or why the application did (or did not) behave as expected.

This is what acceptance tests are doing. They cover scenarios from a user's perspective. With acceptance tests, you can be confident that users, following all the defined scenarios, won't get errors.

Any website can be covered with acceptance tests, even if you use a very exotic CMS or framework.

Sample acceptance test

```
<?php
$I = new AcceptanceTester($scenario);
$I->amOnPage('/');
$I->click('Sign Up');
$I->submitForm('#signup', ['username' => 'MilesDavis', 'email' => 'miles@davis.com']);
$I->see('Thank you for Signing Up!');
```

Functional Tests

What if we could check our application without running it on a server? That way we could see detailed exceptions on errors, have our tests run faster, and check the database against predictable and expected results. That's what functional tests are for.

For functional tests, you emulate a web request (`$_GET` and `$_POST` variables) which returns the HTML response. Inside a test, you can make assertions about the response, and you can check if the data was successfully stored in the database.

For functional tests, your application needs to be structured in order to run in a test environment. Codeception provides connectors to several popular PHP frameworks. You can also write your own.

Sample functional test

```
<?php
$I = new FunctionalTester($scenario);
$I->amOnPage('/');
$I->click('Sign Up');
$I->submitForm('#signup', ['username' => 'MilesDavis', 'email' => 'miles@davis.com']);
$I->see('Thank you for Signing Up!');
$I->seeEmailSent('miles@davis.com', 'Thank you for registration');
$I->seeInDatabase('users', ['email' => 'miles@davis.com']);
```

Unit Tests

Testing pieces of code before coupling them together is highly important as well. This way, you can be sure that some deeply hidden feature still works, even if it was not covered by functional or acceptance tests. This also shows care in producing stable and testable code.

Codeception is created on top of PHPUnit (http://www.phpunit.de/). If you have experience writing unit tests with PHPUnit you can continue doing so. Codeception has no problem executing standard PHPUnit tests, but, additionally, Codeception provides some well-built tools to make your unit tests simpler and cleaner.

Requirements and code can change rapidly, and unit tests should be updated every time to fit the requirements. The better you understand the testing scenario, the faster you can update it for new behavior.

Sample integration test

```
<?php
function testSavingUser()
{
    $user = new User();
    $user->setName('Miles');
    $user->setSurname('Davis');
    $user->save();
    $this->assertEquals('Miles Davis', $user->getFullName());
    $this->unitTester->seeInDatabase('users', ['name' => 'Miles', 'surname' => 'Davis']);
}
```

Conclusion

Despite the wide popularity of *TDD* (Test Driven Development), some PHP developers never write automated tests for their applications mostly because they think it's hard, slow or boring. The Codeception framework was developed to actually make testing fun. It allows writing unit, functional, integration, and acceptance tests in a single, coherent style.

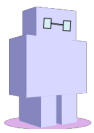
It can be called a *BDD* (Behavior Driven Development) framework. All Codeception tests are written in a descriptive manner. Just by looking at the test body, you can clearly understand what is being tested and how it is performed. Even complex tests with many assertions are written in a simple PHP Domain-Specific Language (*DSL*).

- **Next Chapter: GettingStarted > (/docs/02-GettingStarted)**

Looking for commercial support? Request official consulting service (http://sdclabs.com/codeception?utm_source=codeception.com&utm_medium=docs_bottom&utm_term=link&utm_campaign=reference)

- License (https://github.com/Codeception/Codeception/blob/master/LICENSE)

Codeception Family



Robo

(http://robo.li)(http://robo.li)
Modern PHP **Task Runner**. Allows to declare tasks with zero configuration in pure PHP.

CodeceptJS

(http://codecept.io)(http://codecept.io)
Codeception for **NodeJS**. Write acceptance tests in ES6 and execute in webdriverio, Selenium WebDriver, and Protractor.