

Codeception for



Yii2 Module (/docs/modules/Yii2)

Maintainer



Alex Makarov

GitHub (<https://github.com/samdark>)

samdark

Yii Framework provides basic and advanced application templates. Both include sample Codeception tests, thus to start with Codeception you need to start new Yii project from a one of those templates.

Yii Basic

Once you created a project from a basic template (<https://github.com/yiisoft/yii2-app-basic/>) you should see folder `tests` and `codeception.yml` configuration file. From start there are functional and unit test suites.

Run them by executing in terminal:

```
./vendor/bin/codecept run
```

Follow [@codeception](#)

```

Functional Tests (10) -----
✓ ContactFormCest: Open contact page (0.29s)
✓ ContactFormCest: Submit empty form (0.05s)
✓ ContactFormCest: Submit form with incorrect email (0.02s)
✓ ContactFormCest: Submit form successfully (0.11s)
✓ LoginFormCest: Open login page (0.01s)
✓ LoginFormCest: Internal login by id (0.01s)
✓ LoginFormCest: Internal login by instance (0.03s)
✓ LoginFormCest: Login with empty credentials (0.02s)
✓ LoginFormCest: Login with wrong credentials (0.01s)
✓ LoginFormCest: Login successfully (0.03s)
-----

Unit Tests (8) -----
✓ ContactFormTest: Email is sent on contact (0.03s)
✓ LoginFormTest: Login no user (0.00s)
✓ LoginFormTest: Login wrong password (0.00s)
✓ LoginFormTest: Login correct (0.00s)
✓ UserTest: Find user by id (0.00s)
✓ UserTest: Find user by access token (0.00s)
✓ UserTest: Find user by username (0.00s)
✓ UserTest: Validate user (0.00s)
-----

Time: 904 ms, Memory: 24.00MB

OK (18 tests, 52 assertions)

```

Unit Tests

Unit tests are located in `tests/unit` directory and are supposed to contain all kind of unit and integration testing.

Each test case extends `Codeception\Test\Unit` class, which is standard Codeception format for unit testing. It is pretty hard to develop completely isolated unit tests in Yii, so an application is bootstrapped before each test case. Tests are configured in `tests/unit.suite.yml` file with Yii2 module enabled:

```

modules:
    enabled:
        - Yii2:
            part: [orm, email]

```

This module starts Yii application for a test case and provides additional helper methods to simplify testing. It has only `orm` and `email` parts in order to exclude methods needed for functional testing only.

Follow @codeception

By accessing `$this->tester` class inside a test case you can use methods of Yii2 module (<http://codeception.com/docs/modules/Yii2>). So if you have orm and email parts enabled so you can call methods belonging from these parts:

```
<?php
// insert records in database
$this->tester->haveRecord('app/model/User', ['username' => 'davert']);
// check records in database
$this->tester->seeRecord('app/model/User', ['username' => 'davert']);
// test email was sent
$this->tester->seeEmailIsSent();
// get a last sent emails
$this->tester->grabLastSentEmail();
```

If you enable `fixtures` part you will also get methods to load and use fixtures in your tests:

```
<?php
// load fixtures
$this->tester->haveFixtures([
    'user' => [
        'class' => UserFixture::className(),
        // fixture data located in tests/_data/user.php
        'dataFile' => codecept_data_dir() . 'user.php'
    ]
]);
// get first user from fixtures
$this->tester->grabFixture('user', 0);
```

If Yii2 module is enabled you can safely call `Yii::$app` inside a test, as application is initialized and cleaned up after a test. If you want to add your helper methods or custom assertions for your test case you should not extend `Codeception\Test\Unit` but write your own separate Helper class (<http://codeception.com/docs/06-ModulesAndHelpers#Helpers>).

i Continue to Unit Testing Guide » (<http://codeception.com/docs/05-UnitTests>)

Functional Tests

When it comes to test *real* features of web applications you can't go with unit testing only. You want to test how application handles the requests, what responses it provides, what data is saved to database and so on. To test application in near user environment but without launching real webserver or a browser you can use functional tests. They are far more

Follow @codeception

simpler than unit tests in a way they are written. They describe interaction scenario in a simple DSL so you don't need to deal with application directly but describe actions from a user's perspective:

```
<?php
$I->amOnPage(['site/contact']);
$I->submitForm('#contact-form', []);
$I->expectTo('see validations errors');
$I->see('Contact', 'h1');
$I->see('Name cannot be blank');
$I->see('Email cannot be blank');
$I->see('Subject cannot be blank');
$I->see('Body cannot be blank');
```

This way you not only test the ContactForm on a site but actual output of application that user sees. Codeception provides standard set of actions like `amOnPage`, `submitForm`, `see` for testing. Yii2 module provides special methods, like `amLoggedInAs` (for fast authentication), `haveRecord`, `seeRecord`, `seeEmailIsSent` and others. They all are listed in module reference (<http://codeception.com/docs/modules/Yii2>).

Functional tests should be written inside Cest files (<http://codeception.com/docs/07-AdvancedUsage#Cest-Classes>), which is a scenario-driven test format of Codeception. You can easily create a new test by running:

```
./vendor/bin/codecept g:cest functional MyNewScenarioCest
```

i Functional tests are really powerful and simple. They are highly **recommended to use** for any Yii application.

Continue to Functional Testing Guide » (<http://codeception.com/docs/04-FunctionalTests>)

API Tests

API tests are not included in any Yii templates so you need to set up them manually if you developing a web service. API testing is done at functional testing level but instead of testing HTML responses on user actions, they test requests and responses via protocols like REST or SOAP. To start writing api tests you should create a suite for them

```
./vendor/bin/codecept g:suite api
```

Follow @codeception

You will need to enable REST, Yii2 module in `tests/api.suite.yml`:

```
class_name: ApiTester
modules:
  enabled:
    - REST:
        url: /api/v1
        depends: Yii2
    - \ApiBundle\Helper\Api
  config:
    - Yii2
```

Yii2 module actions like `amOnPage` or `see` should not be available for testing API. This is why Yii2 module is not enabled but declared with `depends` for REST module.

i Continue to REST API Testing Guide » (<http://codeception.com/docs/10-WebServices#REST>).

Acceptance Tests

From a test perspective acceptance tests do the same as functional tests. They test the user interaction with application but in this case using *real* browser and web server. They are much slower and much more fragile. They should not duplicate functional tests in matter of testing functionality but should be used for testing the UI of your application. If you are unsure which tests should be acceptance and which are functional, write acceptance tests for JavaScript-rich applications, where UI highly depends on a browser processing. You can also use acceptance tests for happy-path scenarios, just to ensure that a real user using a real browser achieve the same results you expect in functional tests.

By default in basic application acceptance tests are disabled (as they require web server, Selenium Server and browser to be running). You can easily enable them by renaming `acceptance.suite.yml.example` to `acceptance.suite.yml`

```
mv tests/acceptance.suite.yml.example tests/acceptance.suite.yml
```

Basic template uses `codeception/base` package which doesn't contain `facebook/webdriver` library required to run acceptance tests. Please change `codeception/base` to `codeception/codeception` in `composer.json` and run the update command.

Then you will need to launch application server in test mode:

Follow [@codeception](#)

```
./tests/bin/yii serve
```

and start a Selenium Server or PhantomJS

(<http://codeception.com/docs/modules/WebDriver#Local-Testing>). For acceptance WebDriver module is used. Please check its reference to learn how to work with it. Unlike Yii2 module it does know nothing about your application, so if you want to use features of Yii like fixtures for acceptance testing, you should check that enable Yii2 module is enabled as well:

```
# config at tests/acceptance.yml
modules:
  enabled:
    - WebDriver:
        url: http://127.0.0.1:8080/
        browser: firefox
    - Yii2:
        part: [orm, fixtures] # allow to use AR methods
        cleanup: false # don't wrap test in transaction
        entryScript: index-test.php
```

As it was said, functional and acceptance tests are similar, so in order to avoid conflicts with these modules you should load only a part of Yii2 module which you really need. You must also set `cleanup: false` so Yii2 changes to database to be saved and used by application running on web server. Use `entryScript` and `entryUrl` values to change the default host and script configuration for your app.

Similar as for functional tests it is recommended to use Cest format for acceptance testing:

```
./vendor/bin/codecept g:cest acceptance MyNewScenarioCest
```

i Continue to Acceptance Testing Guide » (<http://codeception.com/docs/03-AcceptanceTests>)

Yii Advanced

Advanced template also has Codeception preinstalled with examples of unit, functional, and acceptance tests. However, you won't find `tests` folder in a root of application. This because every application `frontend`, `backend`, and their `common` has their own tests. This is done to have tests and source to be placed in a same location. To run all tests from all application at once, you should execute `codecept run` from a project root. Global `codeception.yml` config was written to include tests from all applications.

Follow @codeception

```

Common\tests.unit Tests (3) -----
✓ LoginFormTest: Login no user (0.10s)
✓ LoginFormTest: Login wrong password (0.62s)
✓ LoginFormTest: Login correct (0.01s)

[frontend\tests]: tests from /home/davert/demos/yii2-app-advanced/frontend

Frontend\tests.functional Tests (12) -----
✓ AboutCest: Check about (0.2s)
✓ ContactCest: Check contact (0.1s)
✓ ContactCest: Check contact submit no data (0.2s)
✓ ContactCest: Check contact submit not correct email (0.1s)
✓ ContactCest: Check contact submit correct data (0.6s)
✓ HomeCest: Check open (0.0s)
✓ LoginCest: Check empty (0.0s)
✓ LoginCest: Check wrong password (0.6s)
✓ LoginCest: Check valid login (0.63s)
✓ SignupCest: Signup with empty fields (0.1s)
✓ SignupCest: Signup with wrong email (0.2s)
✓ SignupCest: Signup successfully (0.58s)

Frontend\tests.unit Tests (9) -----
✓ ContactFormTest: Send email (0.0s)
✓ PasswordResetRequestFormTest: Send message with wrong email address (0.8s)
✓ PasswordResetRequestFormTest: Not send emails to inactive user (0.7s)
✓ PasswordResetRequestFormTest: Send email successfully (0.9s)
✓ ResetPasswordFormTest: Reset wrong token (0.9s)
✓ ResetPasswordFormTest: Reset correct token (0.63s)
✓ SignupFormTest: Correct signup (0.117s)
✓ SignupFormTest: Not correct signup (0.9s)

[backend\tests]: tests from /home/davert/demos/yii2-app-advanced/backend

Backend\tests.functional Tests (1) -----
✓ LoginCest: Login user (0.01s)

Backend\tests.unit Tests (0) -----

Time: 6.05 seconds, Memory: 30.00MB
OK (24 tests, 67 assertions)
→ yii2-app-advanced git:(master) X

```

Tests also include namespaces for testcase classes and testers to avoid conflicts between tests from different applications.

i Learn more about testing multiple applications » (<http://codeception.com/docs/08-Customization#One-Runner-for-Multiple-Applications>)

Manual Setup & Configuration

To start you need to install Codeception via Composer

```
composer require "codeception/codeception" --dev
```

Create basic test suites

```
./vendor/bin/codecept bootstrap
```

Enable module Yii2 for functional tests inside `functional.suite.yml`:

```
# functional.suite.yml
modules:
  enabled:
    - Yii2:
        configFile: #insert path to config file
```

Follow @codeception

The only required parameter for Yii2 module is `configFile`. This file with configuration for test configuration of Yii application. It should merge original application config overriding `id` value and provide different database for testing:

```
<?php
// config/test.php
$config = yii\helpers\ArrayHelper::merge(
    require(__DIR__ . '/main.php'),
    require(__DIR__ . '/main-local.php'),
    [
        'id' => 'app-tests',
        'components' => [
            'db' => [
                'dsn' => 'mysql:host=localhost;dbname=yii_app_test',
            ]
        ]
    ]
);
return $config;
```

Test config is recommended to store in `config` folder of application. You should provide path to test config relatively to `codeception.yml` file.

Please also make sure that `YII_ENV` constant is set to `test` as it is done in `tests/_bootstrap.php` file of basic and advanced app templates.

Once you configured functional tests it should be easy to create setup for unit and acceptance tests, as it is described in this guide.

In basic and advanced application templates `configFile` is defined in global configuration file:

```
# inside codeception.yml
modules:
    config:
        Yii2:
            configFile: 'config/test.php'
```

This way we don't need to provide test config for each defined suite.

Codeception Tests

- ✓ **Recommended** by Yii core team
- ✓ Pre-installed for basic and advanced application templates
- ✓ Official Yii framework integration

Reference

- ✓ Yii2 Module (</docs/modules/Yii2>)
- ✓ Yii Basic Template (<https://github.com/yiisoft/yii2-app-basic>)
- ✓ Yii Advanced Template (<https://github.com/yiisoft/yii2-app-advanced>)

stable 2.3.5 (<http://codeception.com/thanks.html>) downloads 5.84 M



Community Forum (<http://phptest.club/c/codeception>)



GitHub Repository (<https://github.com/codeception/codeception>)

Yii Basic

Yii Advanced

Manual Setup && Configuration

Codeception is a BDD-styled PHP testing framework, brought to you by Codeception Team (<http://codeception.com/credits>). Logo by Mr. Adnan (<https://twitter.com/adnanblog>). OpenSource **MIT Licensed**.

Thanks to



(<https://www.jetbrains.com/phpstorm/>)



(<https://www.rebilly.com/>)



(<https://github.com/codeception/codeception>)



(<https://twitter.com/codeception>)

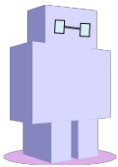
Follow @codeception



(<http://www.facebook.com/pages/Codeception/288959711204412>)

- Installation (/install)
- Credits (/credits)
- Releases (/changelog)
- Commercial Services (http://sdclabs.com/codeception?utm_source=codeception.com&utm_medium=bottom_menu&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.

© 2011–2017

Follow @codeception

