**Yii2**

Этот модуль обеспечивает интеграцию с yii framework (2.0). Он инициализирует yii framework в тестовой среде и предоставляет действия для функционального тестирования.

[**Yii2** 1](#_Toc503870619)

[Действия 5](#_Toc503870620)

[Проверки 5](#_Toc503870621)

[see 5](#_Toc503870622)

[dontSee 5](#_Toc503870623)

[seeCheckboxIsChecked 6](#_Toc503870624)

[dontSeeCheckboxIsChecked 6](#_Toc503870625)

[seeCookie 7](#_Toc503870626)

[dontSeeCookie 7](#_Toc503870627)

[seeCurrentUrlEquals 7](#_Toc503870628)

[dontSeeCurrentUrlEquals 7](#_Toc503870629)

[seeCurrentUrlMatches 7](#_Toc503870630)

[dontSeeCurrentUrlMatches 8](#_Toc503870631)

[seeElement 8](#_Toc503870632)

[dontSeeElement 8](#_Toc503870633)

[seeEmailIsSent 9](#_Toc503870634)

[dontSeeEmailIsSent 9](#_Toc503870635)

[seeInCurrentUrl 9](#_Toc503870636)

[dontSeeInCurrentUrl 9](#_Toc503870637)

[seeInField 10](#_Toc503870638)

[dontSeeInField 10](#_Toc503870639)

[seeInFormFields 10](#_Toc503870640)

[dontSeeInFormFields 11](#_Toc503870641)

[seeInSource 12](#_Toc503870642)

[dontSeeInSource 12](#_Toc503870643)

[seeInTitle 12](#_Toc503870644)

[dontSeeInTitle 13](#_Toc503870645)

[seeLink 13](#_Toc503870646)

[dontSeeLink 13](#_Toc503870647)

[seeNumberOfElements 13](#_Toc503870648)

[seeOptionIsSelected 13](#_Toc503870649)

[dontSeeOptionIsSelected 14](#_Toc503870650)

[seePageNotFound 14](#_Toc503870651)

[seeResponseCodeIs 14](#_Toc503870652)

[dontSeeResponseCodeIs 14](#_Toc503870653)

[dontSeeRecord 15](#_Toc503870654)

[Взять элементы 15](#_Toc503870655)

[getInternalDomains 15](#_Toc503870656)

[grabAttributeFrom 15](#_Toc503870657)

[grabComponent 15](#_Toc503870658)

[grabCookie 16](#_Toc503870659)

[grabFixture 16](#_Toc503870660)

[grabFixtures 16](#_Toc503870661)

[grabFromCurrentUrl 16](#_Toc503870662)

[grabLastSentEmail 16](#_Toc503870663)

[grabMultiple 17](#_Toc503870664)

[grabPageSource 17](#_Toc503870665)

[grabRecord 17](#_Toc503870666)

[grabSentEmails 17](#_Toc503870667)

[grabTextFrom 18](#_Toc503870668)

[grabValueFrom 18](#_Toc503870669)

[Работа с Элементами на форме 18](#_Toc503870670)

[attachFile 18](#_Toc503870671)

[checkOption 18](#_Toc503870672)

[uncheckOption 19](#_Toc503870673)

[selectOption 19](#_Toc503870674)

[setCookie 19](#_Toc503870675)

[submitForm 20](#_Toc503870676)

[fillField 22](#_Toc503870677)

[Запросы 23](#_Toc503870678)

[amOnPage 23](#_Toc503870679)

[amOnRoute 23](#_Toc503870680)

[sendAjaxGetRequest 23](#_Toc503870681)

[sendAjaxPostRequest 23](#_Toc503870682)

[sendAjaxRequest 23](#_Toc503870683)

[Авторизация 24](#_Toc503870684)

[amHttpAuthenticated 24](#_Toc503870685)

[amLoggedInAs 24](#_Toc503870686)

[Разное 24](#_Toc503870687)

[switchToIframe 24](#_Toc503870688)

[\_findElements 25](#_Toc503870689)

[\_getResponseContent 25](#_Toc503870690)

[\_loadPage 26](#_Toc503870691)

[\_request 26](#_Toc503870692)

[\_savePageSource 26](#_Toc503870693)

[haveFixtures 27](#_Toc503870694)

[haveHttpHeader 27](#_Toc503870695)

[haveRecord 28](#_Toc503870696)

[moveBack 28](#_Toc503870697)

[resetCookie 28](#_Toc503870698)

[click 28](#_Toc503870699)

[deleteHeader 29](#_Toc503870700)

**Конфигурация**

* configFile required - путь к файлу конфигурации приложения. Файл должен быть настроен для тестовой среды и возвращаемого массива конфигурации.
* entryUrl - начальный URL-адрес приложения (по умолчанию: http://localhost/index-test.РНР).
* entryScript - название скрипта front-end (например: index-test.РНР.) Если не установлено-берется из entryUrl.
* transaction - (по умолчанию: true) оберните все подключение к базе данных внутри транзакции и откатите его после теста. Должен быть отключен для приемочного тестирования..
* cleanup - (по умолчанию: true) очистка заготовленных даных после теста

Вы можете использовать этот модуль, установив параметрыв вашем functional.suite.yml

actor: FunctionalTester  
modules:  
 enabled:  
 - Yii2:  
 configFile: '/path/to/config.php'

**Parts**

По умолчанию загружаются все доступные методы, но можно указать детали, чтобы выбрать только необходимые действия и избежать конфликтов.

* init - используйте модуль только для инициализации (для приемочных испытаний).
* orm - подключается только haveRecord/grabRecord/seeRecord/dontSeeRecord действий.
* fixtures - используйте заготовленные данные внутри испытаний с haveFixtures/grabFixture/grabFixtures действиями.
* email – подключайте только seeEmailsIsSent/grabLastSentEmail/...

**Example (functional.suite.yml)**

actor: FunctionalTester  
modules:  
 enabled:  
 - Yii2:  
 configFile: 'config/test.php'

**Example (unit.suite.yml)**

actor: UnitTester  
modules:  
 enabled:  
 - Asserts  
 - Yii2:  
 configFile: 'config/test.php'  
 part: init

**Example (acceptance.suite.yml)**

actor: AcceptanceTester  
modules:  
 enabled:  
 - WebDriver:  
 url: http://127.0.0.1:8080/  
 browser: firefox  
 - Yii2:  
 configFile: 'config/test.php'  
 part: ORM # allow to use AR methods  
 transaction: false # don't wrap test in transaction  
 cleanup: false # don't cleanup the fixtures  
 entryScript: index-test.php

**Fixtures**

Этот модуль позволяет использовать приспособления внутри испытания. Для этого есть два варианта. Заготовленные данные могут быть загружены с помощью метода haveFixtures внутри теста:

<?php  
$I->haveFixtures(['posts' => PostsFixture::className()]);

или, если вам нужно загрузить заготовленные данные до испытания, можно указать с методом \_fixtures из теста:

<?php  
// inside Cest file or Codeception\TestCase\Unit  
public function \_fixtures()  
{  
 return ['posts' => PostsFixture::className()]  
}

**URL**

Этот модуль обеспечивает использование собственных форматов URL-адрес в коде с yii2 для всех команд codeception, которые используют URL-адреса для работы. Эта команда позволяет вводить как:

<?php  
$I->amOnPage(['site/view','page'=>'about']);  
$I->amOnPage('index-test.php?site/index');  
$I->amOnPage('http://localhost/index-test.php?site/index');  
$I->sendAjaxPostRequest(['/user/update', 'id' => 1], ['UserForm[name]' => 'G.Hopper');

**Status**

Maintainer: **samdark** Stability: **stable**

## Действия

## Проверки

### see

Проверяет, содержит ли текущая страница заданную строку (без учета регистра).

Можно указать определенный элемент HTML (через CSS или XPath) в качестве второго параметра для поиска только внутри этого элемента.

<?php  
$I->see('Logout'); // Я предполагаю, что пользователь вошел в систему  
$I->see('Sign Up', 'h1'); // Я могу предположить, что это страница регистрации  
$I->see('Sign Up', '//body/h1'); // с XPath  
$I->see('Sign Up', ['css' => 'body h1']); // со строгим CSS локатором

Обратите внимание, что поиск выполняется после удаления всех HTML-тегов из тела, поэтому $I - > see ('strong') вернет true для таких строк, как:

* <p>I am Stronger than thou</p>
* <script>document.createElement('strong');</script>

Но не будет верно для таких строк, как:

* <strong>Home</strong>
* <div class="strong">Home</strong>
* <!-- strong -->

Для проверки исходного кода используйте seeInSource ().

* param string $text
* param string $selector optional

### dontSee

Проверяет, что текущая страница не содержит указанного текста (без учета регистра). Задайте локатор в качестве второго параметра для соответствия определенной области.

<?php  
$I->dontSee('Login'); // Я предполагаю, что пользователь уже вошел в систему  
$I->dontSee('Sign Up','h1'); // Я могу предположить, что это не страница регистрации  
$I->dontSee('Sign Up','//body/h1'); // с XPath  
$I->dontSee('Sign Up', ['css' => 'body h1']); // со строгим CSS локатором

Обратите внимание, что поиск производится после зачистки все HTML-теги из организма, поэтому $я->dontSee ("strong") будет выполнена в строк, как:

* <p>I am Stronger than thou</p>
* <script>document.createElement('strong');</script>

Но будет игнорировать строки, как:

* <strong>Home</strong>
* <div class="strong">Home</strong>
* <!-- strong -->

Для проверки исходного кода используйте seeInSource ().

* param string $text
* param string $selector optional

### seeCheckboxIsChecked

Проверяет, установлен ли указанный флажок.

<?php  
$I->seeCheckboxIsChecked('#agree'); // I suppose user agreed to terms  
$I->seeCheckboxIsChecked('#signup\_form input[type=checkbox]'); // I suppose user agreed to terms, If there is only one checkbox in form.  
$I->seeCheckboxIsChecked('//form/input[@type=checkbox and @name=agree]');  
?>

* param $checkbox

### dontSeeCheckboxIsChecked

Проверяет, что указанный флажок не установлен.

<?php  
$I->dontSeeCheckboxIsChecked('#agree'); // Я полагаю, что пользователь не согласился с условиями  
$I->seeCheckboxIsChecked('#signup\_form input[type=checkbox]'); // I suppose user didn't check the first checkbox in form.  
?>

* param $checkbox

### seeCookie

Проверяет, что cookie с заданным именем установлен. Вы можете установить дополнительные параметры как domain, path как массив который передается в последнем аргументе.

<?php  
$I->seeCookie('PHPSESSID');  
?>

* param $cookie
* param array $params

### dontSeeCookie

Проверяет, что cookie с заданным именем НЕ установлен. Вы можете установить дополнительные параметры как domain, path как массив который передается в последнем аргументе.

* param $cookie
* param array $params

### seeCurrentUrlEquals

Проверяет, что текущий URL-адрес равен заданной строке. В отличие от seeInCurrentUrl, это только соответствествие полному URL-адресу.

<?php  
// to match root url  
$I->seeCurrentUrlEquals('/');  
?>

* param string $uri

### dontSeeCurrentUrlEquals

Проверяет, что текущий URL-адрес не равен заданной строке. В отличие от dontSeeInCurrentUrl, это только соотвествие полному URL-адресу.

<?php  
// current url is not root  
$I->dontSeeCurrentUrlEquals('/');  
?>

* param string $uri

### seeCurrentUrlMatches

Проверяет соответствие текущего URL-адреса заданному регулярному выражению.

<?php  
// to match root url  
$I->seeCurrentUrlMatches('~$/users/(\d+)~');  
?>

* param string $uri

### dontSeeCurrentUrlMatches

Проверяет, что текущий url-адрес не соответствует данному регулярному выражению.

<?php  
// to match root url  
$I->dontSeeCurrentUrlMatches('~$/users/(\d+)~');  
?>

* param string $uri

### seeElement

Проверяет, существует ли данный элемент на странице и виден. Можно также указать ожидаемые атрибуты этого элемента.

<?php  
$I->seeElement('.error');  
$I->seeElement('//form/input[1]');  
$I->seeElement('input', ['name' => 'login']);  
$I->seeElement('input', ['value' => '123456']);  
  
// strict locator in first arg, attributes in second  
$I->seeElement(['css' => 'form input'], ['name' => 'login']);  
?>

* param $selector
* param array $attributes @return

### dontSeeElement

Проверяет, является ли данный элемент невидимым или нет на странице. Можно также указать ожидаемые атрибуты этого элемента.

<?php  
$I->dontSeeElement('.error');  
$I->dontSeeElement('//form/input[1]');  
$I->dontSeeElement('input', ['name' => 'login']);  
$I->dontSeeElement('input', ['value' => '123456']);  
?>

* param $selector
* param array $attributes

### seeEmailIsSent

Проверяет, что письмо отправлено.

<?php  
// check that at least 1 email was sent  
$I->seeEmailIsSent();  
  
// check that only 3 emails were sent  
$I->seeEmailIsSent(3);

* param int $num @throws ModuleException
* [Part] email

### dontSeeEmailIsSent

Проверяет, что письмо НЕ отправлено

* [Part] email

### seeInCurrentUrl

Проверяет, что текущий URI содержит заданную строку.

<?php  
// to match: /home/dashboard  
$I->seeInCurrentUrl('home');  
// to match: /users/1  
$I->seeInCurrentUrl('/users/');  
?>

* param string $uri

### dontSeeInCurrentUrl

Проверяет, что текущий URI не содержит заданную строку.

<?php  
$I->dontSeeInCurrentUrl('/users/');  
?>

* param string $uri

### seeInField

Проверяет, что данное поле ввода или textarea равно (т. е. не просто содержит) заданному значению. Поля сопоставляются текстом метки, атрибутом”name", CSS или XPath.

<?php  
$I->seeInField('Body','Type your comment here');  
$I->seeInField('form textarea[name=body]','Type your comment here');  
$I->seeInField('form input[type=hidden]','hidden\_value');  
$I->seeInField('#searchform input','Search');  
$I->seeInField('//form/\*[@name=search]','Search');  
$I->seeInField(['name' => 'search'], 'Search');  
?>

* param $field
* param $value

### dontSeeInField

Проверяет, что данное поле ввода или textarea НЕ равно заданному значению. Поля сопоставляются текстом метки, атрибутом”name", CSS или XPath

<?php  
$I->dontSeeInField('Body','Type your comment here');  
$I->dontSeeInField('form textarea[name=body]','Type your comment here');  
$I->dontSeeInField('form input[type=hidden]','hidden\_value');  
$I->dontSeeInField('#searchform input','Search');  
$I->dontSeeInField('//form/\*[@name=search]','Search');  
$I->dontSeeInField(['name' => 'search'], 'Search');  
?>

* param $field
* param $value

### seeInFormFields

Проверяет, является ли массив параметров формы (имя => значение) устанавливаемых на форме прошел валидацию.

<?php  
$I->seeInFormFields('form[name=myform]', [  
 'input1' => 'value',  
 'input2' => 'other value',  
]);  
?>

Для элементов с несколькими выборками или для проверки значений нескольких элементов с одинаковым именем может быть передан массив:

<?php  
$I->seeInFormFields('.form-class', [  
 'multiselect' => [  
 'value1',  
 'value2',  
 ],  
 'checkbox[]' => [  
 'a checked value',  
 'another checked value',  
 ],  
]);  
?>

Кроме того, значения флажков можно проверить с помощью логического.

<?php  
$I->seeInFormFields('#form-id', [  
 'checkbox1' => true, // passes if checked  
 'checkbox2' => false, // passes if unchecked  
]);  
?>

Соедините это с submitForm для быстрого тестирования магии.

<?php  
$form = [  
 'field1' => 'value',  
 'field2' => 'another value',  
 'checkbox1' => true,  
 // ...  
];  
$I->submitForm('//form[@id=my-form]', $form, 'submitButton');  
// $I->amOnPage('/path/to/form-page') may be needed  
$I->seeInFormFields('//form[@id=my-form]', $form);  
?>

* param $formSelector
* param $params

### dontSeeInFormFields

Проверяет, является ли массив параметры из формы (имя => значение) не установились на форме

<?php  
$I->dontSeeInFormFields('form[name=myform]', [  
 'input1' => 'non-existent value',  
 'input2' => 'other non-existent value',  
]);  
?>

Чтобы проверить, что элементу не присвоено ни одно из многих значений, в качестве значения может быть передан массив:

<?php  
$I->dontSeeInFormFields('.form-class', [  
 'fieldName' => [  
 'This value shouldn\'t be set',  
 'And this value shouldn\'t be set',  
 ],  
]);  
?>

Кроме того, значения флажков можно проверить с помощью логического.

<?php  
$I->dontSeeInFormFields('#form-id', [  
 'checkbox1' => true, // fails if checked  
 'checkbox2' => false, // fails if unchecked  
]);  
?>

* param $formSelector
* param $params

### seeInSource

Проверяет, содержит ли текущая страница заданную строку в исходном коде.

<?php  
$I->seeInSource('<h1>Green eggs &amp; ham</h1>');

* param $raw

### dontSeeInSource

Проверяет, текущая страница НЕ содержит заданную строку в исходном коде.

<?php  
$I->dontSeeInSource('<h1>Green eggs &amp; ham</h1>');

* param $raw

### seeInTitle

Проверяет, содержит ли Заголовок страницы заданную строку.

<?php  
$I->seeInTitle('Blog - Post #1');  
?>

* param $title

### dontSeeInTitle

Проверяет, не содержит ли Заголовок страницы заданной строки.

* param $title

### seeLink

Проверяет наличие ссылки на указанный текст. Предоставьте полный URL в качестве второго параметра для сопоставления ссылок с этим точным URL-адресом.

<?php  
$I->seeLink('Logout'); // matches <a href="#">Logout</a>  
$I->seeLink('Logout','/logout'); // matches <a href="/logout">Logout</a>  
?>

* param string $text
* param string $url optional

### dontSeeLink

Проверяет, что страница не содержит ссылки на данную строку. Если второй параметр задан, только ссылки с соответствующим атрибутом “href” будут проверены.

<?php  
$I->dontSeeLink('Logout'); // I suppose user is not logged in  
$I->dontSeeLink('Checkout now', '/store/cart.php');  
?>

* param string $text
* param string $url optional

### seeNumberOfElements

Проверяет наличие определенного количества элементов, совпадающих с заданным локатором на странице.

<?php  
$I->seeNumberOfElements('tr', 10);  
$I->seeNumberOfElements('tr', [0,10]); // between 0 and 10 elements  
?>

* param $selector
* param mixed $expected int or int[]

### seeOptionIsSelected

Проверяет, что выбран данный параметр.

<?php  
$I->seeOptionIsSelected('#form input[name=payment]', 'Visa');  
?>

* param $selector
* param $optionText

### dontSeeOptionIsSelected

Проверяет, что данный параметр не выбран.

<?php  
$I->dontSeeOptionIsSelected('#form input[name=payment]', 'Visa');  
?>

* param $selector
* param $optionText

### seePageNotFound

Утверждает, что текущая страница имеет код состояния ответа 404.

### seeRecord

Проверяет, существует ли запись в базе данных.

$I->seeRecord('app\models\User', array('name' => 'davert'));

* param $model
* param array $attributes
* [Part] orm

### dontSeeRecord

Проверяет, что запись не существует в базе данных.

$I->dontSeeRecord('app\models\User', array('name' => 'davert'));

* param $model
* param array $attributes
* [Part] orm

### seeResponseCodeIs

Проверяет, что код ответа равен указанному значению.

<?php  
$I->seeResponseCodeIs(200);  
  
// recommended \Codeception\Util\HttpCode  
$I->seeResponseCodeIs(\Codeception\Util\HttpCode::OK);

* param $code

### dontSeeResponseCodeIs

Проверяет, что код ответа равен указанному значению.

<?php  
$I->dontSeeResponseCodeIs(200);  
  
// recommended \Codeception\Util\HttpCode  
$I->dontSeeResponseCodeIs(\Codeception\Util\HttpCode::OK);

* param $code

## Взять элементы

### getInternalDomains

Возвращает список шаблонов регулярных выражений для распознанных доменных имен

* return array

### grabAttributeFrom

Захватывает значение данного значения атрибута из данного элемента. Не удается, если элемент не найден.

<?php  
$I->grabAttributeFrom('#tooltip', 'title');  
?>

* param $cssOrXpath
* param $attribute

### grabComponent

Возвращает компонент из контейнера yii. Создает исключение, если компонент недоступен

<?php  
$mailer = $I->grabComponent('mailer');

* param $component @throws ModuleException

### grabCookie

Захватывает значение cookie. Вы можете установить дополнительные файлы cookie params, как домен, путь в массиве передается в качестве последнего аргумента.

* param $cookie
* param array $params

### grabFixture

Получает предустановленные данные по имени. Возвращает экземпляр предустановленные данные. Если предустановленные данные является экземпляром \ yii\test\BaseActiveFixture, можно использовать второй параметр для возврата определенной модели:

<?php  
$I->haveFixtures(['users' => UserFixture::className()]);  
  
$users = $I->grabFixture('users');  
  
// get first user by key, if a fixture is instance of ActiveFixture  
$user = $I->grabFixture('users', 'user1');

* param $name @throws ModuleException if a fixture is not found
* [Part] fixtures

### grabFixtures

Возвращает все загруженные данные. Массив экземпляров предустановленных данных

* [Part] fixtures
* return array

### grabFromCurrentUrl

Выполняет заданное регулярное выражение для текущего URI и возвращает первую группу записи. Если параметры не указаны, возвращается полный URI.

<?php  
$user\_id = $I->grabFromCurrentUrl('~$/user/(\d+)/~');  
$uri = $I->grabFromCurrentUrl();  
?>

* param string $uri optional

### grabLastSentEmail

Возвращает последнее отправленное письмо:

<?php  
$I->seeEmailIsSent();  
$message = $I->grabLastSentEmail();  
$I->assertEquals('admin@site,com', $message->getTo());

* [Part] email

### grabMultiple

Захватывает текстовое содержимое или значения атрибутов узлов, соответствующих $css или Xpath и возвращает их в виде массива.

<a href="#first">First</a>  
<a href="#second">Second</a>  
<a href="#third">Third</a>

<?php  
// would return ['First', 'Second', 'Third']  
$aLinkText = $I->grabMultiple('a');  
  
// would return ['#first', '#second', '#third']  
$aLinks = $I->grabMultiple('a', 'href');  
?>

* param $cssOrXpath
* param $attribute
* return string[]

### grabPageSource

Захватывает исходный код текущей страницы.

@throws ModuleException if no page was opened.

* return string Current page source code.

### grabRecord

Извлекает запись из базы данных

$category = $I->grabRecord('app\models\User', array('name' => 'davert'));

* param $model
* param array $attributes
* [Part] orm

### grabSentEmails

Возвращает массив всех отправленных сообщений электронной почты. Каждое сообщение реализует интерфейс сообщение\\yii на почте. Полезно выполнять дополнительные проверки с помощью модуля Asserts:

<?php  
$I->seeEmailIsSent();  
$messages = $I->grabSentEmails();  
$I->assertEquals('admin@site,com', $messages[0]->getTo());

* [Part] email
* return array @throws ModuleException

### grabTextFrom

Находит и возвращает текстовое содержимое данного элемента. Если используется нечеткий локатор, элемент найден с помощью CSS, XPath и путем сопоставления источника полной страницы регулярным выражением.

<?php  
$heading = $I->grabTextFrom('h1');  
$heading = $I->grabTextFrom('descendant-or-self::h1');  
$value = $I->grabTextFrom('~<input value=(.\*?)]~sgi'); // match with a regex  
?>

* param $cssOrXPathOrRegex

### grabValueFrom

* param $field

|  |  |  |  |
| --- | --- | --- | --- |
| return array | mixed | null | string |

## Работа с Элементами на форме

### attachFile

Прикрепляет файл относительно каталога Codeception \_data к данному полю загрузки файла.

<?php  
// file is stored in 'tests/\_data/prices.xls'  
$I->attachFile('input[@type="file"]', 'prices.xls');  
?>

* param $field
* param $filename

### checkOption

Галочки флажок. Для радио-кнопки, использовать вместо метода selectOption.

<?php  
$I->checkOption('#agree');  
?>

* param $option

### uncheckOption

Unticks a checkbox.

<?php  
$I->uncheckOption('#notify');  
?>

* param $option

### selectOption

Выбор параметра в теге select или в группе переключателей.

<?php  
$I->selectOption('form select[name=account]', 'Premium');  
$I->selectOption('form input[name=payment]', 'Monthly');  
$I->selectOption('//form/select[@name=account]', 'Monthly');  
?>

Представить массив в качестве второго аргумента, чтобы выбрать несколько вариантов:

<?php  
$I->selectOption('Which OS do you use?', array('Windows','Linux'));  
?>

Или предоставьте ассоциативный массив для второго аргумента, чтобы конкретно определить, какой метод выбора должен использоваться:

<?php  
$I->selectOption('Which OS do you use?', array('text' => 'Windows')); // Only search by text 'Windows'  
$I->selectOption('Which OS do you use?', array('value' => 'windows')); // Only search by value 'windows'  
?>

* param $select
* param $option

### setCookie

Устанавливает cookie с заданным именем и значением. Вы можете установить дополнительные файлы cookie, как domain, path, expires, secure в массиве передается как последний аргумент.

<?php  
$I->setCookie('PHPSESSID', 'el4ukv0kqbvoirg7nkp4dncpk3');  
?>

* param $name
* param $val
* param array $params

### submitForm

Представляет данную форму на странице с заданными значениями формы. Передать значения поля формы в виде массива во втором параметре.

Хотя эту функцию можно использовать как короткую версию fillField(), selectOption(), click() etc. имеет некоторые важные отличия:

* Only field names may be used, not CSS/XPath selectors nor field labels
* If a field is sent to this function that does not exist on the page, it will silently be added to the HTTP request. This is helpful for testing some types of forms, but be aware that you will not get an exception like you would if you called fillField() or selectOption() with a missing field.

Fields that are not provided will be filled by their values from the page, or from any previous calls to fillField(), selectOption() etc. You don’t need to click the ‘Submit’ button afterwards. This command itself triggers the request to form’s action.

You can optionally specify which button’s value to include in the request with the last parameter (as an alternative to explicitly setting its value in the second parameter), as button values are not otherwise included in the request.

Examples:

<?php  
$I->submitForm('#login', [  
 'login' => 'davert',  
 'password' => '123456'  
]);  
// or  
$I->submitForm('#login', [  
 'login' => 'davert',  
 'password' => '123456'  
], 'submitButtonName');

For example, given this sample “Sign Up” form:

<form action="/sign\_up">  
 Login:  
 <input type="text" name="user[login]" /><br/>  
 Password:  
 <input type="password" name="user[password]" /><br/>  
 Do you agree to our terms?  
 <input type="checkbox" name="user[agree]" /><br/>  
 Select pricing plan:  
 <select name="plan">  
 <option value="1">Free</option>  
 <option value="2" selected="selected">Paid</option>  
 </select>  
 <input type="submit" name="submitButton" value="Submit" />  
</form>

You could write the following to submit it:

<?php  
$I->submitForm(  
 '#userForm',  
 [  
 'user' => [  
 'login' => 'Davert',  
 'password' => '123456',  
 'agree' => true  
 ]  
 ],  
 'submitButton'  
);

Note that “2” will be the submitted value for the “plan” field, as it is the selected option.

You can also emulate a JavaScript submission by not specifying any buttons in the third parameter to submitForm.

<?php  
$I->submitForm(  
 '#userForm',  
 [  
 'user' => [  
 'login' => 'Davert',  
 'password' => '123456',  
 'agree' => true  
 ]  
 ]  
);

This function works well when paired with seeInFormFields() for quickly testing CRUD interfaces and form validation logic.

<?php  
$form = [  
 'field1' => 'value',  
 'field2' => 'another value',  
 'checkbox1' => true,  
 // ...  
];  
$I->submitForm('#my-form', $form, 'submitButton');  
// $I->amOnPage('/path/to/form-page') may be needed  
$I->seeInFormFields('#my-form', $form);

Parameter values can be set to arrays for multiple input fields of the same name, or multi-select combo boxes. For checkboxes, you can use either the string value or boolean true/false which will be replaced by the checkbox’s value in the DOM.

<?php  
$I->submitForm('#my-form', [  
 'field1' => 'value',  
 'checkbox' => [  
 'value of first checkbox',  
 'value of second checkbox',  
 ],  
 'otherCheckboxes' => [  
 true,  
 false,  
 false  
 ],  
 'multiselect' => [  
 'first option value',  
 'second option value'  
 ]  
]);

Mixing string and boolean values for a checkbox’s value is not supported and may produce unexpected results.

Field names ending in [] must be passed without the trailing square bracket characters, and must contain an array for its value. This allows submitting multiple values with the same name, consider:

<?php  
// This will NOT work correctly  
$I->submitForm('#my-form', [  
 'field[]' => 'value',  
 'field[]' => 'another value', // 'field[]' is already a defined key  
]);

The solution is to pass an array value:

<?php  
// This way both values are submitted  
$I->submitForm('#my-form', [  
 'field' => [  
 'value',  
 'another value',  
 ]  
]);

* param $selector
* param $params
* param $button

### fillField

Fills a text field or textarea with the given string.

<?php  
$I->fillField("//input[@type='text']", "Hello World!");  
$I->fillField(['name' => 'email'], 'jon@mail.com');  
?>

* param $field
* param $value

## Запросы

### amOnPage

Opens the page for the given relative URI.

<?php  
// opens front page  
$I->amOnPage('/');  
// opens /register page  
$I->amOnPage('/register');

* param string $page

### amOnRoute

Similar to amOnPage but accepts route as first argument and params as second

$I->amOnRoute('site/view', ['page' => 'about']);

### sendAjaxGetRequest

If your page triggers an ajax request, you can perform it manually. This action sends a GET ajax request with specified params.

See ->sendAjaxPostRequest for examples.

* param $uri
* param $params

### sendAjaxPostRequest

If your page triggers an ajax request, you can perform it manually. This action sends a POST ajax request with specified params. Additional params can be passed as array.

Example:

Imagine that by clicking checkbox you trigger ajax request which updates user settings. We emulate that click by running this ajax request manually.

<?php  
$I->sendAjaxPostRequest('/updateSettings', array('notifications' => true)); // POST  
$I->sendAjaxGetRequest('/updateSettings', array('notifications' => true)); // GET

* param $uri
* param $params

### sendAjaxRequest

If your page triggers an ajax request, you can perform it manually. This action sends an ajax request with specified method and params.

Example:

You need to perform an ajax request specifying the HTTP method.

<?php  
$I->sendAjaxRequest('PUT', '/posts/7', array('title' => 'new title'));

* param $method
* param $uri
* param $params

## Авторизация

### amHttpAuthenticated

Authenticates user for HTTP\_AUTH

* param $username
* param $password

### amLoggedInAs

Authorizes user on a site without submitting login form. Use it for fast pragmatic authorization in functional tests.

<?php  
// User is found by id  
$I->amLoggedInAs(1);  
  
// User object is passed as parameter  
$admin = \app\models\User::findByUsername('admin');  
$I->amLoggedInAs($admin);

Requires user component to be enabled and configured.

* param $user @throws ModuleException

## Разное

### switchToIframe

Switch to iframe or frame on the page.

Example:

<iframe name="another\_frame" src="http://example.com">

<?php  
# switch to iframe  
$I->switchToIframe("another\_frame");

* param string $name

### \_findElements

hidden API method, expected to be used from Helper classes

Locates element using available Codeception locator types:

* XPath
* CSS
* Strict Locator

Use it in Helpers or GroupObject or Extension classes:

<?php  
$els = $this->getModule('Yii2')->\_findElements('.items');  
$els = $this->getModule('Yii2')->\_findElements(['name' => 'username']);  
  
$editLinks = $this->getModule('Yii2')->\_findElements(['link' => 'Edit']);  
// now you can iterate over $editLinks and check that all them have valid hrefs

WebDriver module returns Facebook\WebDriver\Remote\RemoteWebElement instances PhpBrowser and Framework modules return Symfony\Component\DomCrawler\Crawler instances

* param $locator
* return array of interactive elements

### \_getResponseContent

hidden API method, expected to be used from Helper classes

Returns content of the last response Use it in Helpers when you want to retrieve response of request performed by another module.

<?php  
// in Helper class  
public function seeResponseContains($text)  
{  
 $this->assertContains($text, $this->getModule('Yii2')->\_getResponseContent(), "response contains");  
}  
?>

* return string @throws ModuleException

### \_loadPage

hidden API method, expected to be used from Helper classes

Opens a page with arbitrary request parameters. Useful for testing multi-step forms on a specific step.

<?php  
// in Helper class  
public function openCheckoutFormStep2($orderId) {  
 $this->getModule('Yii2')->\_loadPage('POST', '/checkout/step2', ['order' => $orderId]);  
}  
?>

* param $method
* param $uri
* param array $parameters
* param array $files
* param array $server
* param null $content

### \_request

hidden API method, expected to be used from Helper classes

Send custom request to a backend using method, uri, parameters, etc. Use it in Helpers to create special request actions, like accessing API Returns a string with response body.

<?php  
// in Helper class  
public function createUserByApi($name) {  
 $userData = $this->getModule('Yii2')->\_request('POST', '/api/v1/users', ['name' => $name]);  
 $user = json\_decode($userData);  
 return $user->id;  
}  
?>

Does not load the response into the module so you can’t interact with response page (click, fill forms). To load arbitrary page for interaction, use \_loadPage method.

* param $method
* param $uri
* param array $parameters
* param array $files
* param array $server
* param null $content
* return mixed|Crawler @throws ExternalUrlException @see \_loadPage

### \_savePageSource

hidden API method, expected to be used from Helper classes

Saves page source of to a file

$this->getModule('Yii2')->\_savePageSource(codecept\_output\_dir().'page.html');

* param $filename

### haveFixtures

Creates and loads fixtures from a config. Signature is the same as for fixtures() method of yii\test\FixtureTrait

<?php  
$I->haveFixtures([  
 'posts' => PostsFixture::className(),  
 'user' => [  
 'class' => UserFixture::className(),  
 'dataFile' => '@tests/\_data/models/user.php',  
 ],  
]);

Note: if you need to load fixtures before the test (probably before the cleanup transaction is started; cleanup options is true by default), you can specify fixtures with \_fixtures method of a testcase

<?php  
// inside Cest file or Codeception\TestCase\Unit  
public function \_fixtures(){  
 return [  
 'user' => [  
 'class' => UserFixture::className(),  
 'dataFile' => codecept\_data\_dir() . 'user.php'  
 ]  
 ];  
}

instead of defining haveFixtures in Cest \_before

* param $fixtures
* [Part] fixtures

### haveHttpHeader

Sets the HTTP header to the passed value - which is used on subsequent HTTP requests through PhpBrowser.

Example:

<?php  
$I->haveHttpHeader('X-Requested-With', 'Codeception');  
$I->amOnPage('test-headers.php');  
?>

To use special chars in Header Key use HTML Character Entities: Example: Header with underscore - ‘Client\_Id’ should be represented as - ‘Client\_Id’ or ‘Client\_Id’

<?php  
$I->haveHttpHeader('Client&#95;Id', 'Codeception');  
?>

* param string $name the name of the request header
* param string $value the value to set it to for subsequent requests

### haveRecord

Inserts record into the database.

<?php  
$user\_id = $I->haveRecord('app\models\User', array('name' => 'Davert'));  
?>

* param $model
* param array $attributes
* [Part] orm

### moveBack

Moves back in history.

* param int $numberOfSteps (default value 1)

### resetCookie

Unsets cookie with the given name. You can set additional cookie params like domain, path in array passed as last argument.

* param $cookie
* param array $params

### click

Perform a click on a link or a button, given by a locator. If a fuzzy locator is given, the page will be searched for a button, link, or image matching the locator string. For buttons, the “value” attribute, “name” attribute, and inner text are searched. For links, the link text is searched. For images, the “alt” attribute and inner text of any parent links are searched.

The second parameter is a context (CSS or XPath locator) to narrow the search.

Note that if the locator matches a button of type submit, the form will be submitted.

<?php  
// simple link  
$I->click('Logout');  
// button of form  
$I->click('Submit');  
// CSS button  
$I->click('#form input[type=submit]');  
// XPath  
$I->click('//form/\*[@type=submit]');  
// link in context  
$I->click('Logout', '#nav');  
// using strict locator  
$I->click(['link' => 'Login']);  
?>

* param $link
* param $context

### deleteHeader

Deletes the header with the passed name. Subsequent requests will not have the deleted header in its request.

Example:

<?php  
$I->haveHttpHeader('X-Requested-With', 'Codeception');  
$I->amOnPage('test-headers.php');  
// ...  
$I->deleteHeader('X-Requested-With');  
$I->amOnPage('some-other-page.php');  
?>

* param string $name the name of the header to delete.