live search...

see
seeCheckboxIsChecked
seeCookie
seeCurrentUrlEquals
seeCurrentUrlMatches
seeElement
seeEmailIsSent
seeInCurrentUrl
seeInField
seeInFormFields
seeInSource
seeInTitle
seeLink
seeNumberOfElements
seeOptionIsSelected
seePageNotFound
seeRecord
seeResponseCodeIs
selectOption
sendAjaxGetRequest
sendAjaxPostRequest
sendAjaxRequest
setCookie
submitForm
switchToIframe
uncheckOption

Y source (https://github.com/Codeception/Codeception/blob/2.3/src/Codeception/Module/Yii2.php)
i master (https://github.com/Codeception/Codeception/blob/master/docs/modules/Yii2.md)
i 2.2 (https://github.com/Codeception/Codeception/blob/2.2/docs/modules/Yii2.md)
2.1 (https://github.com/Codeception/Codeception/blob/2.1/docs/modules/Yii2.md)
2.0 (https://github.com/Codeception/Codeception/blob/2.0/docs/modules/Yii2.md)
1.8 (https://github.com/Codeception/Codeception/blob/1.8/docs/modules/Yii2.md)

2

This module provides integration with Yii framework (http://www.yiiframework.com/) (2.0). It initializes Yii framework in test environment and provides actions for functional testing.

Config

- `configFile` *required* - the path to the application config file. File should be configured for test environment and return configuration array.
- `entryUrl` - initial application url (default: http://localhost/index-test.php).
- `entryScript` - front script title (like: index-test.php). If not set - taken from entryUrl.
- `transaction` - (default: true) wrap all database connection inside a transaction and roll it back after the test. Should be disabled for acceptance testing..
- `cleanup` - (default: true) cleanup fixtures after the test

You can use this module by setting params in your functional.suite.yml:

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

Parts

By default all available methods are loaded, but you can specify parts to select only needed actions and avoid conflicts.

- `init` - use module only for initialization (for acceptance tests).
- `orm` - include only `haveRecord/grabRecord/seeRecord/dontSeeRecord` actions.
- `fixtures` - use fixtures inside tests with `haveFixtures/grabFixture/grabFixtures` actions.
- `email` - include email actions `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

This module allows to use fixtures (http://www.yiiframework.com/doc-2.0/guide-test-fixtures.html) inside a test. There are two options for that. Fixtures can be loaded using haveFixtures method inside a test:

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

or, if you need to load fixtures before the test, you can specify fixtures with `_fixtures` method of a testcase:

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

URL

This module provide to use native URL formats of Yii2 for all codeception commands that use url for work. This commands allows input like:

```
<?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**

Actions

_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

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

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']);
```

attachFile

Attaches a file relative to the Codeception `_data` directory to the given file upload field.

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

- `param` $field
- `param` $filename

checkOption

Ticks a checkbox. For radio buttons, use the `selectOption` method instead.

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

- `param` $option

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.

dontSee

Checks that the current page doesn't contain the text specified (case insensitive). Give a locator as the second parameter to match a specific region.

```
<?php
$I->dontSee('Login');                         // I can suppose user is already logged in
$I->dontSee('Sign Up','h1');                  // I can suppose it's not a signup page
$I->dontSee('Sign Up','//body/h1');           // with XPath
$I->dontSee('Sign Up', ['css' => 'body h1']); // with strict CSS locator
```

Note that the search is done after stripping all HTML tags from the body, so `$I->dontSee('strong')` will fail on strings like:

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

But will ignore strings like:

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

For checking the raw source code, use `seeInSource()`.

- `param string` $text
- `param string` $selector optional

dontSeeCheckboxIsChecked

Check that the specified checkbox is unchecked.

```
<?php
$I->dontSeeCheckboxIsChecked('#agree'); // I suppose user didn't agree to terms
$I->seeCheckboxIsChecked('#signup_form input[type=checkbox]'); // I suppose user didn't check the first checkbox in form.
?>
```

- `param` $checkbox

dontSeeCookie

Checks that there isn't a cookie with the given name. You can set additional cookie params like `domain`, `path` as array passed in last argument.

- `param` $cookie
- `param array` $params

dontSeeCurrentUrlEquals

Checks that the current URL doesn't equal the given string. Unlike `dontSeeInCurrentUrl`, this only matches the full URL.

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

- `param string` $uri

dontSeeCurrentUrlMatches

Checks that current url doesn't match the given regular expression.

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

- `param string` $uri

dontSeeElement

Checks that the given element is invisible or not present on the page. You can also specify expected attributes of this element

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

- `param` $selector
- `param array` $attributes

dontSeeEmailIsSent

Checks that no email was sent

- `[Part]` email

dontSeeInCurrentUrl

Checks that the current URI doesn't contain the given string.

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

- `param string` $uri

dontSeeInField

Checks that an input field or textarea doesn't contain the given value. For fuzzy locators, the field is matched by label text, CSS and 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

dontSeeInFormFields

Checks if the array of form parameters (name => value) are not set on the form matched with the passed selector.

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

To check that an element hasn't been assigned any one of many values, an array can be passed as the value:

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

Additionally, checkbox values can be checked with a boolean.

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

- `param` $formSelector
- `param` $params

dontSeeInSource

Checks that the current page contains the given string in its raw source code.

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

- `param` $raw

dontSeeInTitle

Checks that the page title does not contain the given string.

- `param` $title

dontSeeLink

Checks that the page doesn't contain a link with the given string. If the second parameter is given, only links with a matching "href" attribute will be checked.

```
<?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

dontSeeOptionIsSelected

Checks that the given option is not selected.

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

- `param` $selector
- `param` $optionText

dontSeeRecord

Checks that record does not exist in database.

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

- `param` $model
- `param array` $attributes
- `[Part]` orm

dontSeeResponseCodeIs

Checks that response code is equal to value provided.

```
<?php
$I->dontSeeResponseCodeIs(200);

// recommended \Codeception\Util\HttpCode
$I->dontSeeResponseCodeIs(\Codeception\Util\HttpCode::OK);
```

- `param` $code

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

getInternalDomains

Returns a list of regex patterns for recognized domain names

- `return` array

grabAttributeFrom

Grabs the value of the given attribute value from the given element. Fails if element is not found.

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

- `param` $cssOrXpath
- `param` $attribute

grabComponent

Gets a component from Yii container. Throws exception if component is not available

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

- `param` $component @throws ModuleException

grabCookie

Grabs a cookie value. You can set additional cookie params like `domain`, `path` in array passed as last argument.

- `param` $cookie
- `param array` $params

grabFixture

Gets a fixture by name. Returns a Fixture instance. If a fixture is an instance of `\yii\test\BaseActiveFixture` a second parameter can be used to return a specific model:

```
<?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

Returns all loaded fixtures. Array of fixture instances

- `[Part]` fixtures
- `return` array

grabFromCurrentUrl

Executes the given regular expression against the current URI and returns the first match. If no parameters are provided, the full URI is returned

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

- `param string` $uri optional

grabLastSentEmail

Returns last sent email:

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

- `[Part]` email

grabMultiple

Grabs either the text content, or attribute values, of nodes matched by $cssOrXpath and returns them as an array.

```
<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

Grabs current page source code.

@throws ModuleException if no page was opened.

- `return` string Current page source code.

grabRecord

Retrieves record from database

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

- `param` $model
- `param array` $attributes
- `[Part]` orm

grabSentEmails

Returns array of all sent email messages. Each message implements `yii\mail\Message` interface. Useful to perform additional checks using `Asserts` module:

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

- `[Part]` email
- `return` array @throws ModuleException

grabTextFrom

Finds and returns the text contents of the given element. If a fuzzy locator is used, the element is found using CSS, XPath, and by matching the full page source by regular expression.

```
<?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

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');
?>
```

- `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

see

Checks that the current page contains the given string (case insensitive).

You can specify a specific HTML element (via CSS or XPath) as the second parameter to only search within that element.

```
<?php
$I->see('Logout');                      // I can suppose user is logged in
$I->see('Sign Up', 'h1');               // I can suppose it's a signup page
$I->see('Sign Up', '//body/h1');        // with XPath
$I->see('Sign Up', ['css' => 'body h1']); // with strict CSS locator
```

Note that the search is done after stripping all HTML tags from the body, so `$I->see('strong')` will return true for strings like:

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

But will *not* be true for strings like:

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

For checking the raw source code, use `seeInSource()`.

- `param string` $text
- `param string` $selector optional

seeCheckboxIsChecked

Checks that the specified checkbox is checked.

```
<?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 fo
$I->seeCheckboxIsChecked('//form/input[@type=checkbox and @name=agree]');
?>
```

- `param` $checkbox

seeCookie

Checks that a cookie with the given name is set. You can set additional cookie params like `domain, path` as array passed in last argument.

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

- `param` $cookie

- `param array` $params

seeCurrentUrlEquals

Checks that the current URL is equal to the given string. Unlike `seeInCurrentUrl`, this only matches the full URL.

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

- `param string` $uri

seeCurrentUrlMatches

Checks that the current URL matches the given regular expression.

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

- `param string` $uri

seeElement

Checks that the given element exists on the page and is visible. You can also specify expected attributes of this element.

```
<?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

seeEmailIsSent

Checks that email is sent.

```
<?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

seeInCurrentUrl

Checks that current URI contains the given string.

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

- `param string` $uri

seeInField

Checks that the given input field or textarea *equals* (i.e. not just contains) the given value. Fields are matched by label text, the “name” attribute, CSS, or 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

seeInFormFields

Checks if the array of form parameters (name => value) are set on the form matched with the passed selector.

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

For multi-select elements, or to check values of multiple elements with the same name, an array may be passed:

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

Additionally, checkbox values can be checked with a boolean.

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

Pair this with submitForm for quick testing magic.

```
<?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

seeInSource

Checks that the current page contains the given string in its raw source code.

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

- `param` $raw

seeInTitle

Checks that the page title contains the given string.

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

- `param` $title

seeLink

Checks that there's a link with the specified text. Give a full URL as the second parameter to match links with that exact URL.

```
<?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

seeNumberOfElements

Checks that there are a certain number of elements matched by the given locator on the page.

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

- `param` $selector
- `param mixed` $expected int or int[]

seeOptionIsSelected

Checks that the given option is selected.

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

- `param` $selector
- `param` $optionText

seePageNotFound

Asserts that current page has 404 response status code.

seeRecord

Checks that record exists in database.

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

- `param` $model
- `param array` $attributes
- `[Part]` orm

seeResponseCodeIs

Checks that response code is equal to value provided.

```
<?php
$I->seeResponseCodeIs(200);

// recommended \Codeception\Util\HttpCode
$I->seeResponseCodeIs(\Codeception\Util\HttpCode::OK);
```

- `param` $code

selectOption

Selects an option in a select tag or in radio button group.

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

Provide an array for the second argument to select multiple options:

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

Or provide an associative array for the second argument to specifically define which selection method should be used:

```
<?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

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

setCookie

Sets a cookie with the given name and value. You can set additional cookie params like `domain`, `path`, `expires`, `secure` in array passed as last argument.

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

- `param` $name
- `param` $val
- `param array` $params

submitForm

Submits the given form on the page, with the given form values. Pass the form field's values as an array in the second parameter.

Although this function can be used as a short-hand version of `fillField()`, `selectOption()`, `click()` etc. it has some important differences:

- 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

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

uncheckOption

Unticks a checkbox.

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

- `param` $option

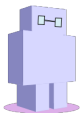
Module reference is taken from the source code. Help us to improve documentation. Edit module reference (https://github.com/Codeception/Codeception/tree/2.3/src/Codeception/Module/Yii2.php)


(https://www.jetbrains.com/phpstorm/) Rebilly (https://www.rebilly.com/)


(https://github.com/codeception/codeception) 
(https://twitter.com/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.