**WebDriver**

Модуль Selenium WebDriver нового поколения.

[**WebDriver** 1](#_Toc503864952)

[Локальное Тестирование 3](#_Toc503864953)

[Selenium 3](#_Toc503864954)

[ChromeDriver 3](#_Toc503864955)

[PhantomJS 4](#_Toc503864956)

[Автономный Selenium in Docker 4](#_Toc503864957)

[Облачное Тестирование 4](#_Toc503864958)

[Конфигурация 5](#_Toc503864959)

[Действия 7](#_Toc503864960)

[Отправка запросов 7](#_Toc503864961)

[amOnPage 7](#_Toc503864962)

[amOnSubdomain 8](#_Toc503864963)

[amOnUrl 8](#_Toc503864964)

[submitForm 8](#_Toc503864965)

[Манипуляция DOM элементами 11](#_Toc503864966)

[fillField 11](#_Toc503864967)

[appendField 11](#_Toc503864968)

[checkOption 11](#_Toc503864969)

[uncheckOption 11](#_Toc503864970)

[selectOption 12](#_Toc503864971)

[unselectOption 12](#_Toc503864972)

[attachFile 13](#_Toc503864973)

[События 13](#_Toc503864974)

[click 13](#_Toc503864975)

[doubleClick 13](#_Toc503864976)

[clickWithRightButton 13](#_Toc503864977)

[clickWithLeftButton 14](#_Toc503864978)

[moveMouseOver 14](#_Toc503864979)

[dragAndDrop 15](#_Toc503864980)

[pressKey 15](#_Toc503864981)

[scrollTo 15](#_Toc503864982)

[switchToIFrame 15](#_Toc503864983)

[switchToNextTab 16](#_Toc503864984)

[switchToPreviousTab 16](#_Toc503864985)

[switchToWindow 16](#_Toc503864986)

[typeInPopup 17](#_Toc503864987)

[maximizeWindow 17](#_Toc503864988)

[resizeWindow 17](#_Toc503864989)

[moveBack 17](#_Toc503864990)

[moveForward 17](#_Toc503864991)

[openNewTab 18](#_Toc503864992)

[Методы проверки 18](#_Toc503864993)

[see 18](#_Toc503864994)

[dontSee 18](#_Toc503864995)

[seeCheckboxIsChecked 19](#_Toc503864996)

[dontSeeCheckboxIsChecked 19](#_Toc503864997)

[seeCookie 20](#_Toc503864998)

[dontSeeCookie 20](#_Toc503864999)

[seeCurrentUrlEquals 20](#_Toc503865000)

[dontSeeCurrentUrlEquals 20](#_Toc503865001)

[seeCurrentUrlMatches 20](#_Toc503865002)

[dontSeeCurrentUrlMatches 21](#_Toc503865003)

[seeElement 21](#_Toc503865004)

[dontSeeElement 21](#_Toc503865005)

[seeElementInDOM 22](#_Toc503865006)

[dontSeeElementInDOM 22](#_Toc503865007)

[seeInCurrentUrl 22](#_Toc503865008)

[dontSeeInCurrentUrl 22](#_Toc503865009)

[seeInField 23](#_Toc503865010)

[dontSeeInField 23](#_Toc503865011)

[seeInFormFields 23](#_Toc503865012)

[dontSeeInFormFields 24](#_Toc503865013)

[seeInPageSource 25](#_Toc503865014)

[dontSeeInSource 25](#_Toc503865015)

[seeInPopup 25](#_Toc503865016)

[dontSeeInPopup 26](#_Toc503865017)

[seeInSource 26](#_Toc503865018)

[dontSeeInPageSource 26](#_Toc503865019)

[seeInTitle 26](#_Toc503865020)

[dontSeeInTitle 26](#_Toc503865021)

[seeLink 26](#_Toc503865022)

[dontSeeLink 27](#_Toc503865023)

[seeNumberOfElements 27](#_Toc503865024)

[seeNumberOfElementsInDOM 27](#_Toc503865025)

[seeOptionIsSelected 27](#_Toc503865026)

[dontSeeOptionIsSelected 27](#_Toc503865027)

[Паузы в выполнении 28](#_Toc503865028)

[wait 28](#_Toc503865029)

[waitForElement 28](#_Toc503865030)

[waitForElementChange 28](#_Toc503865031)

[waitForElementNotVisible 28](#_Toc503865032)

[waitForElementVisible 29](#_Toc503865033)

[waitForJS 29](#_Toc503865034)

[waitForText 29](#_Toc503865035)

[pauseExecution 29](#_Toc503865036)

[Взять значения 30](#_Toc503865037)

[grabTextFrom 30](#_Toc503865038)

[grabAttributeFrom 30](#_Toc503865039)

[grabCookie 30](#_Toc503865040)

[grabFromCurrentUrl 30](#_Toc503865041)

[grabMultiple 31](#_Toc503865042)

[grabPageSource 31](#_Toc503865043)

[grabValueFrom 31](#_Toc503865044)

[Разное 32](#_Toc503865045)

[acceptPopup 32](#_Toc503865046)

[cancelPopup 32](#_Toc503865047)

[closeTab 32](#_Toc503865048)

[debugWebDriverLogs 32](#_Toc503865049)

[executeInSelenium 32](#_Toc503865050)

[executeJS 32](#_Toc503865051)

[loadSessionSnapshot 33](#_Toc503865052)

[makeScreenshot 33](#_Toc503865053)

[performOn 33](#_Toc503865054)

[reloadPage 34](#_Toc503865055)

[saveSessionSnapshot 34](#_Toc503865056)

[setCookie 34](#_Toc503865057)

[resetCookie 34](#_Toc503865058)

[\_backupSession 34](#_Toc503865059)

[\_capabilities 35](#_Toc503865060)

[\_closeSession 35](#_Toc503865061)

# Локальное Тестирование

## Selenium

Для запуска сервера Selenium Вам потребуется Java, а также браузер Chrome или Firefox..

1. Скачайте [Selenium Standalone Server](http://docs.seleniumhq.org/download/)
2. При использовании Chrome, установите [ChromeDriver](https://sites.google.com/a/chromium.org/chromedriver/getting-started). При испльзовании Firefox, установите [GeckoDriver](https://github.com/mozilla/geckodriver).
3. Запустите Selenium Server: java -jar selenium-server-standalone-3.xx.xxx.jar. Для использования Chromedriver используйе опцию -Dwebdriver.chrome.driver=./chromedriver
4. Для Geckodriver используйте опцию -Dwebdriver.gecko.driver=./geckodriver.
5. Конфигурирование модуля (в acceptance.suite.yml) установите опции url и browser:

modules:  
 enabled:  
 - WebDriver:  
 url: 'http://localhost/'  
 browser: chrome # 'chrome' or 'firefox'

Запускайте Selenium Server перед запуском тестов.

java -jar "/path/to/selenium-server-standalone-xxx.jar"

## ChromeDriver

Для запуска тестов в браузере Chrome вы можете подключиться к ChromeDriver напрямую, без использования сервера Selenium.

1. Установите [ChromeDriver](https://sites.google.com/a/chromium.org/chromedriver/getting-started).
2. Запустите ChromeDriver: chromedriver --url-base=/wd/hub
3. Конфигурируйте этот модуль указав используемый ChromeDriver порт:

modules:  
 enabled:  
 - WebDriver:  
 url: 'http://localhost/'  
 window\_size: false # disabled in ChromeDriver  
 port: 9515  
 browser: chrome  
 capabilities:  
 chromeOptions: # additional chrome options

Дополнительные параметры [Chrome](https://sites.google.com/a/chromium.org/chromedriver/capabilities) могут быть установлены в возможностях chromeOptions.

## PhantomJS

PhantomJS is a [headless browser](https://en.wikipedia.org/wiki/Headless_browser) alternative to Selenium Server that implements [the WebDriver protocol](https://code.google.com/p/selenium/wiki/JsonWireProtocol). It allows you to run Selenium tests on a server without a GUI installed.

1. Скачайте [PhantomJS](http://phantomjs.org/download.html)
2. Run PhantomJS in WebDriver mode: phantomjs --webdriver=4444
3. Configure this module (in acceptance.suite.yml) by setting url and phantomjs as browser:

modules:  
 enabled:  
 - WebDriver:  
 url: 'http://localhost/'  
 browser: phantomjs

Since PhantomJS doesn’t give you any visual feedback, it’s probably a good idea to install [Codeception\Extension\Recorder](http://codeception.com/extensions#CodeceptionExtensionRecorder) which gives you screenshots of how PhantomJS “sees” your pages.

## Автономный Selenium in Docker

Docker can ship Selenium Server with all its dependencies and browsers inside a single container. Running tests inside Docker is as easy as pulling [official selenium image](https://github.com/SeleniumHQ/docker-selenium) and starting a container with Chrome:

docker run --net=host selenium/standalone-chrome

By using --net=host we allow selenium to access local websites.

### Облачное Тестирование

Cloud Testing services can run your WebDriver tests in the cloud. In case you want to test a local site or site behind a firewall you should use a tunnel application provided by a service.

#### SauceLabs

1. Create an account at [SauceLabs.com](http://saucelabs.com/) to get your username and access key
2. In the module configuration use the format username:access\_key@ondemand.saucelabs.com’ for host
3. Configure platform under capabilities to define the [Operating System](https://docs.saucelabs.com/reference/platforms-configurator/#/)
4. run a tunnel app if your site can’t be accessed from Internet

modules:  
 enabled:  
 - WebDriver:  
 url: http://mysite.com  
 host: '<username>:<access key>@ondemand.saucelabs.com'  
 port: 80  
 browser: chrome  
 capabilities:  
 platform: 'Windows 10'

#### BrowserStack

1. Create an account at [BrowserStack](https://www.browserstack.com/) to get your username and access key
2. In the module configuration use the format username:access\_key@hub.browserstack.com’ for host
3. Configure os and os\_version under capabilities to define the operating System
4. If your site is available only locally or via VPN you should use a tunnel app. In this case add browserstack.local capability and set it to true.

modules:  
 enabled:  
 - WebDriver:  
 url: http://mysite.com  
 host: '<username>:<access key>@hub.browserstack.com'  
 port: 80  
 browser: chrome  
 capabilities:  
 os: Windows  
 os\_version: 10  
 browserstack.local: true # for local testing

#### TestingBot

1. Create an account at [TestingBot](https://testingbot.com/) to get your key and secret
2. In the module configuration use the format key:secret@hub.testingbot.com’ for host
3. Configure platform under capabilities to define the [Operating System](https://testingbot.com/support/getting-started/browsers.html)
4. Run [TestingBot Tunnel](https://testingbot.com/support/other/tunnel) if your site can’t be accessed from Internet

modules:  
 enabled:  
 - WebDriver:  
 url: http://mysite.com  
 host: '<key>:<secret>@hub.testingbot.com'  
 port: 80  
 browser: chrome  
 capabilities:  
 platform: Windows 10

## Конфигурация

* url требуется - Стартовый URL для вашего приложения.
* browser требуется – Браузер для запуска.
* host - хост сервера Selenium (по умолчанию 127.0.0.1).
* port - порт сервера Selenium (по умолчанию 4444).
* restart - установите в «false» (по умолчанию) для использования одного и того же окна браузера для всех тестов или установите «true» для создания нового окна для каждого теста. В любом случае, когда все тесты завершены, окно браузера закрывается..
* start - автозапуск браузера для тестирования. Может быть отключен, если сеанс браузера начинается с «\_initializeSession» внутри .Helper.
* window\_size - - Начальный размер окна. Установите значение «maximize» или пропишите в формате «640x480»..
* clear\_cookies - Установите значение «false», чтобы сохранить файлы cookie, или установите значение «true» (по умолчанию), чтобы удалить все куки между тестами.
* wait (по умолчанию: 0 секунд) - всякий раз, когда элемент требуется и не находится на странице, подождите несколько секунд, чтобы найти его перед сбоем.
* capabilities - Настраивает Selenium [желаемые возможности] (https://github.com/SeleniumHQ/selenium/wiki/DesiredCapabilities). Должен быть массив значений ключа.
* connection\_timeout - тайм-аут для открытия соединения с удаленным сервером (по умолчанию 30 секунд).
* request\_timeout - тайм-аут для запроса вернуть что-то из удаленного сервера selenium (по умолчанию 30 секунд).
* pageload\_timeout - время ожидания загрузки страницы до того, как будет выдана ошибка (по умолчанию 0 секунд)..
* http\_proxy - - устанавливает URL-адрес прокси-сервера HTTP для тестирования удаленного сервера.
* http\_proxy\_port - - устанавливает порт HTTP-прокси-сервера
* debug\_log\_entries - сколько записей Selenium для печати с «debugWebDriverLogs» или при сбое (по умолчанию 15).
* log\_js\_errors установите значение «true», чтобы включить возможный отчет JavaScript в HTML, или установите значение «false» (по умолчанию) для деактивации.

Подробнее можно посмотреть в блоке ## Configuration файла vendor\codeception\codeception\src\Codeception\Module\WebDriver.php

Example (acceptance.suite.yml)

modules:  
 enabled:  
 - WebDriver:  
 url: 'http://localhost/'  
 browser: firefox  
 window\_size: 1024x768  
 capabilities:  
 unexpectedAlertBehaviour: 'accept'  
 firefox\_profile: '~/firefox-profiles/codeception-profile.zip.b64'

**Использование**

**Размещение Элементов**

Most methods in this module that operate on a DOM element (e.g. click) accept a locator as the first argument, which can be either a string or an array.

If the locator is an array, it should have a single element, with the key signifying the locator type (id, name, css, xpath, link, or class) and the value being the locator itself. This is called a “strict” locator. Examples:

* ['id' => 'foo'] matches <div id="foo">
* ['name' => 'foo'] matches <div name="foo">
* ['css' => 'input[type=input][value=foo]'] matches <input type="input" value="foo">
* ['xpath' => "//input[@type='submit'][contains(@value, 'foo')]"] matches <input type="submit" value="foobar">
* ['link' => 'Click here'] matches <a href="google.com">Click here</a>
* ['class' => 'foo'] matches <div class="foo">

Writing good locators can be tricky. The Mozilla team has written an excellent guide titled [Writing reliable locators for Selenium and WebDriver tests](https://blog.mozilla.org/webqa/2013/09/26/writing-reliable-locators-for-selenium-and-webdriver-tests/).

If you prefer, you may also pass a string for the locator. This is called a “fuzzy” locator. In this case, Codeception uses a a variety of heuristics (depending on the exact method called) to determine what element you’re referring to. For example, here’s the heuristic used for the submitForm method:

1. Does the locator look like an ID selector (e.g. “#foo”)? If so, try to find a form matching that ID.
2. If nothing found, check if locator looks like a CSS selector. If so, run it.
3. If nothing found, check if locator looks like an XPath expression. If so, run it.
4. Throw an ElementNotFound exception.

Be warned that fuzzy locators can be significantly slower than strict locators. Especially if you use Selenium WebDriver with wait (aka implicit wait) option. In the example above if you set wait to 5 seconds and use XPath string as fuzzy locator, submitForm method will wait for 5 seconds at each step. That means 5 seconds finding the form by ID, another 5 seconds finding by CSS until it finally tries to find the form by XPath). If speed is a concern, it’s recommended you stick with explicitly specifying the locator type via the array syntax.

**Публичные свойства**

* webDriver - instance of \Facebook\WebDriver\Remote\RemoteWebDriver. Can be accessed from Helper classes for complex WebDriver interactions.

// inside Helper class  
$this->getModule('WebDriver')->webDriver->getKeyboard()->sendKeys('hello, webdriver');

# Действия

## Отправка запросов

### amOnPage

Открывает страницу для данного относительного URI.

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

* param string $page

### amOnSubdomain

Изменяет поддомен для параметра конфигурации 'url'. Не открывает страницу; используйте для этого amOnPage.

<?php  
// If config is: 'http://mysite.com'  
// or config is: 'http://www.mysite.com'  
// or config is: 'http://company.mysite.com'  
  
$I->amOnSubdomain('user');  
$I->amOnPage('/');  
// moves to http://user.mysite.com/  
?>

* param $subdomain

### amOnUrl

Открывает веб-страницы, в абсолютном URL-адресе и устанавливает имя хоста в качестве базового узла.

<?php  
$I->amOnUrl('http://codeception.com');  
$I->amOnPage('/quickstart'); // moves to http://codeception.com/quickstart  
?>

### submitForm

Отправляет данную форму на странице с заданными значениями формы. Задайте значения полей формы в виде массива. Обратите внимание, что доступ к скрытым полям невозможен.

Отсутствующие поля будут заполнены значениями из страницы. После этого вам не нужно нажимать кнопку "Отправить". Эта команда сама инициирует запрос к действию формы.

При необходимости можно указать значение кнопки для включения в запрос с последним параметром в качестве альтернативы явному заданию его значения во втором параметре, поскольку значения кнопок в противном случае не включаются в запрос.

Пример:

<?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>

Вы можете написать следующее, чтобы отправить его:

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

Обратите внимание, что” 2 “будет представленное значение для поля” план", так как это выбранная опция.

Также обратите внимание, что это отличается от PhpBrowser, в том, что

'user' => [ 'login' => 'Davert' ]

на данный момент не поддерживается. Именованные ключи массива должны быть включены в имя, как указано выше.

Соедините это с seeInFormFields для быстрой магии испытания.

<?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);  
?>

Значения параметров должны быть установлены в массивы для нескольких полей ввода с одним и тем же именем или в поля со списком с несколькими выборками. Для флажков можно использовать либо строковое значение, либо логические значения, которые будут заменены значением флажка в 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'  
 ]  
]);  
?>

Смешивание строковых и логических значений для значения флажка не поддерживается и может привести к неожиданным результатам.

Имена полей, заканчивающиеся на " []", должны передаваться без символов квадратной скобки и содержать массив для их значения. Это позволяет передавать несколько значений с одинаковым именем

$I->submitForm('#my-form', [  
 'field[]' => 'value',  
 'field[]' => 'another value', // 'field[]' is already a defined key  
]);

Решение заключается в передаче значения массива:

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

Параметр $buttom может быть либо строка, массив или Facebook\WebDriver\WebDriverBy. Когда это строка, ищется «имя» атрибута. Если $button является массивом, то он будет рассматриваться как строгий селектор и WebDriverBy будет использоваться дословно.

Например, даны следующие HTML:

<input type="submit" name="submitButton" value="Submit" />

$button может быть любым из следующих:

* ‘submitButton’
* [‘name’ => ‘submitButton’]
* WebDriverBy::name(‘submitButton’)
* param $selector
* param $params
* param $button

## Манипуляция DOM элементами

### fillField

Заполняет текстовое поле или текстовое поле заданной строкой.

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

* param $field
* param $value

### appendField

Добавить текст в поле выбора. Также можно добавить выделение в поле выбора.

<?php  
$I->appendField('#mySelectbox', 'SelectValue');  
$I->appendField('#myTextField', 'appended');  
?>

* param string $field
* param string $value @throws \Codeception\Exception\ElementNotFound

### checkOption

Выбрать флажок. Для переключателей используйте вместо этого метод selectOption.

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

* param $option

### uncheckOption

Снимите флажок.

<?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

### unselectOption

Снимите флажок в данном поле выбора.

* param $select
* param $option

### 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

## События

### click

Выполните щелчок по ссылке или кнопке, заданной локатором. Если задан нечеткий локатор, страница будет искать кнопку, ссылку или изображение, соответствующее строке локатора. Для кнопок выполняется поиск атрибута ”value“, атрибута ”name" и внутреннего текста. Для ссылок выполняется поиск текста ссылки. Для изображений выполняется поиск атрибута “alt” и внутреннего текста родительских ссылок

Второй параметр-это контекст (CSS или xpath locator) для сужения поиска.

Обратите внимание, что если локатор соответствует кнопке типа submit, форма будет отправлена.

<?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

### doubleClick

Выполняет двойной щелчок на элементе, совпавшие с помощью CSS или xpath.

* param $cssOrXPath @throws \Codeception\Exception\ElementNotFound

### clickWithRightButton

Выполняет контекстный щелчок правой кнопкой мыши на элементе. Если первый параметр null, то смещение относительно фактического положения мыши. Если заданы второй и третий параметры, то мышь перемещается в смещение левого верхнего угла элемента. В противном случае мышь перемещается в центр элемента.

<?php  
$I->clickWithRightButton(['css' => '.checkout']);  
$I->clickWithRightButton(null, 20, 50);  
$I->clickWithRightButton(['css' => '.checkout'], 20, 50);  
?>

* param string $cssOrXPath css or xpath of the web element (body по умолчанию).
* param int $offsetX
* param int $offsetY

@throws \Codeception\Exception\ElementNotFound

### clickWithLeftButton

Выполняет щелчок левой кнопкой мыши на элементе. Если первый параметр null, то смещение относительно фактического положения мыши. Если заданы второй и третий параметры, то мышь перемещается в смещение левого верхнего угла элемента. В противном случае мышь перемещается в центр элемента.

<?php  
$I->clickWithLeftButton(['css' => '.checkout']);  
$I->clickWithLeftButton(null, 20, 50);  
$I->clickWithLeftButton(['css' => '.checkout'], 20, 50);  
?>

* param string $cssOrXPath css or xpath of the web element (body по умолчанию).
* param int $offsetX
* param int $offsetY

@throws \Codeception\Exception\ElementNotFound

### moveMouseOver

Наведите указатель мыши на первый элемент, соответствующий данному локатору. Если первый параметр null, то используется страница. Если заданы второй и третий параметры, то мышь перемещается в смещение левого верхнего угла элемента. В противном случае мышь перемещается в центр элемента.

<?php  
$I->moveMouseOver(['css' => '.checkout']);  
$I->moveMouseOver(null, 20, 50);  
$I->moveMouseOver(['css' => '.checkout'], 20, 50);  
?>

* param string $cssOrXPath css or xpath of the web element
* param int $offsetX
* param int $offsetY

@throws \Codeception\Exception\ElementNotFound

### dragAndDrop

Выполняет простую операцию перетаскивания мышью.

<?php  
$I->dragAndDrop('#drag', '#drop');  
?>

* param string $source (CSS ID or XPath)
* param string $target (CSS ID or XPath)

### pressKey

Нажимает на данный элемент или заданную клавишу. Чтобы указать символ и модификатор (например, ctrl, alt, shift, meta), передайте массив для $char с модификатором в качестве первого элемента и символ в качестве второго. Для специальных ключей используйте ключевые константы из класса WebDriverKeys.

<?php  
// <input id="page" value="old" />  
$I->pressKey('#page','a'); // => olda  
$I->pressKey('#page',array('ctrl','a'),'new'); //=> new  
$I->pressKey('#page',array('shift','111'),'1','x'); //=> old!!!1x  
$I->pressKey('descendant-or-self::\*[@id='page']','u'); //=> oldu  
$I->pressKey('#name', array('ctrl', 'a'), \Facebook\WebDriver\WebDriverKeys::DELETE); //=>''  
?>

* param $element
* param $char string|array Can be char or array with modifier. You can provide several chars. @throws \Codeception\Exception\ElementNotFound

### scrollTo

Переместитесь в середину заданного элемента, соответствующего заданному локатору. Дополнительный сдвиг, вычисленный из верхнего левого угла элемента, можно установить путем передачи параметров $ offsetX и $offsetY.

<?php  
$I->scrollTo(['css' => '.checkout'], 20, 50);  
?>

* param $selector
* param int $offsetX
* param int $offsetY

### switchToIFrame

Switch to another frame on the page.

Example:

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

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

* param string|null $name

### switchToNextTab

Switches to next browser tab. An offset can be specified.

<?php  
// switch to next tab  
$I->switchToNextTab();  
// switch to 2nd next tab  
$I->switchToNextTab(2);

Can’t be used with PhantomJS

* param int $offset 1

### switchToPreviousTab

Switches to previous browser tab. An offset can be specified.

<?php  
// switch to previous tab  
$I->switchToPreviousTab();  
// switch to 2nd previous tab  
$I->switchToPreviousTab(2);

Can’t be used with PhantomJS

* param int $offset 1

### switchToWindow

Переключитесь в другое окно, идентифицированное по имени.

Окно можно идентифицировать только по имени. Если параметр $name пуст, будет использоваться родительское окно.

Пример:

<input type="button" value="Open window" onclick="window.open('http://example.com', 'another\_window')">

<?php  
$I->click("Open window");  
# switch to another window  
$I->switchToWindow("another\_window");  
# switch to parent window  
$I->switchToWindow();  
?>

Если у окна Нет имени, сопоставьте его, переключившись на следующую активную вкладку с помощью метода switchToNextTab.

Или используйте собственные функции Selenium, чтобы получить доступ ко всем открытым окнам:

<?php  
$I->executeInSelenium(function (\Facebook\WebDriver\Remote\RemoteWebDriver $webdriver) {  
 $handles=$webdriver->getWindowHandles();  
 $last\_window = end($handles);  
 $webdriver->switchTo()->window($last\_window);  
});  
?>

* param string|null $name

### typeInPopup

Вводит текст в собственное всплывающее окно приглашения JavaScript, созданное window.prompt.

* param $keys

@throws \Codeception\Exception\ModuleException

### maximizeWindow

Maximizes the current window.

### resizeWindow

Resize the current window.

<?php  
$I->resizeWindow(800, 600);

* param int $width
* param int $height

### moveBack

Moves back in history.

### moveForward

Движется вперед в истории.

### openNewTab

Открывает новую вкладку браузера (где это возможно) и переключается на нее.

<?php  
$I->openNewTab();

Вкладка открывается с помощью window.open javascript в браузере. Обратите внимание, что Adblock может ограничить создание таких закладок.

Не смогите быть использовано с PhantomJS

## Методы 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'); // 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

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'); // 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

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

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

* param $checkbox

### dontSeeCheckboxIsChecked

Убедитесь, что указанный флажок снят.

<?php  
$I->dontSeeCheckboxIsChecked('#agree'); // Я полагаю, что пользователь не согласился с условиями  
$I->seeCheckboxIsChecked('#signup\_form input[type=checkbox]'); // Я полагаю, что пользователь не проверил первый флажок в форме.  
?>

* param $checkbox

### seeCookie

Проверяет, что cookie с заданным именем установлен. Вы можете установить дополнительные параметры cookie domain, path они как массив передаются в последний аргумент.

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

* param $cookie
* param array $params

### dontSeeCookie

Проверяет, нет ли файлов cookie с указанным именем. Вы можете установить дополнительные файлы cookie params, как домен, путь как массив передается в последний аргумент.

* param $cookie
* param array $params

### seeCurrentUrlEquals

Проверяет, что текущий URL-адрес равен заданной строке. В отличие от seeInCurrentUrl, это проверяет соотвествие только полный URL-адрес.

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

* param string $uri

### dontSeeCurrentUrlEquals

Проверяет, что текущий URL-адрес не равен заданной строке. В отличие от seeInCurrentUrl, это проверяет соотвествие только полный URL-адрес.

<?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 его 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

### seeElementInDOM

Проверяет что данный элемент существует на странице, даже если он невидим.

<?php  
$I->seeElementInDOM('//form/input[type=hidden]');  
?>

* param $selector
* param array $attributes

### dontSeeElementInDOM

Противоположность seeElementInDOM.

* param $selector
* param array $attributes

### 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) заданное значение. Поля соответствуют надписи, атрибутe “name” , CSS или xpath-выражения.

<?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, не содержит ли поле ввода или textarea заданного значения. Для нечетких локаторов поле соответствует тексту метки, CSS и 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) устанавливаются по форме соответствовать прошел селектор.

<?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 выражения.

<?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 параметров формы (имя => значение) не устанавливаются на форме соответствовать прошел селектор.

<?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 выражения.

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

* param $formSelector
* param $params

### seeInPageSource

Проверяет, что источник страницы содержит заданную строку.

<?php  
$I->seeInPageSource('<link rel="apple-touch-icon"');

* param $text

### dontSeeInSource

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

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

* param $raw

### seeInPopup

Проверяет, что активное всплывающее окно JavaScript, созданное window.alert|window.confirm|window.prompt, содержит заданную строку.

* param $text

@throws \Codeception\Exception\ModuleException

### dontSeeInPopup

Проверяет, что активное всплывающее окно JavaScript, созданное window.alert|window.confirm|window.prompt, НЕ содержит заданную строку.

* param $text

@throws \Codeception\Exception\ModuleException

### seeInSource

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

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

* param $raw

### dontSeeInPageSource

Проверяет, что источник страницы не содержит заданной строки.

* param $text

### 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. Укажите 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, не содержит ли страница ссылки на данную строку. Если задан второй параметр, будут проверяться только ссылки с соответствующим атрибутом "href".

<?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[]

### seeNumberOfElementsInDOM

**not documented**

### 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

## Паузы в выполнении

### wait

Wait for $timeout seconds.

* param int|float $timeout secs @throws \Codeception\Exception\TestRuntimeException

### waitForElement

Waits up to $timeout seconds for an element to appear on the page. If the element doesn’t appear, a timeout exception is thrown.

<?php  
$I->waitForElement('#agree\_button', 30); // secs  
$I->click('#agree\_button');  
?>

* param $element
* param int $timeout seconds @throws \Exception

### waitForElementChange

Waits up to $timeout seconds for the given element to change. Element “change” is determined by a callback function which is called repeatedly until the return value evaluates to true.

<?php  
use \Facebook\WebDriver\WebDriverElement  
$I->waitForElementChange('#menu', function(WebDriverElement $el) {  
 return $el->isDisplayed();  
}, 100);  
?>

* param $element
* param \Closure $callback
* param int $timeout seconds @throws \Codeception\Exception\ElementNotFound

### waitForElementNotVisible

Waits up to $timeout seconds for the given element to become invisible. If element stays visible, a timeout exception is thrown.

<?php  
$I->waitForElementNotVisible('#agree\_button', 30); // secs  
?>

* param $element
* param int $timeout seconds @throws \Exception

### waitForElementVisible

Waits up to $timeout seconds for the given element to be visible on the page. If element doesn’t appear, a timeout exception is thrown.

<?php  
$I->waitForElementVisible('#agree\_button', 30); // secs  
$I->click('#agree\_button');  
?>

* param $element
* param int $timeout seconds @throws \Exception

### waitForJS

Executes JavaScript and waits up to $timeout seconds for it to return true.

In this example we will wait up to 60 seconds for all jQuery AJAX requests to finish.

<?php  
$I->waitForJS("return $.active == 0;", 60);  
?>

* param string $script
* param int $timeout seconds

### waitForText

Waits up to $timeout seconds for the given string to appear on the page.

Can also be passed a selector to search in, be as specific as possible when using selectors. waitForText() will only watch the first instance of the matching selector / text provided. If the given text doesn’t appear, a timeout exception is thrown.

<?php  
$I->waitForText('foo', 30); // secs  
$I->waitForText('foo', 30, '.title'); // secs  
?>

* param string $text
* param int $timeout seconds
* param string $selector optional @throws \Exception

### pauseExecution

Приостанавливает выполнение теста в режиме отладки. Для продолжения теста нажмите "ENTER" в консоли.

Этот метод полезен при написании тестов, так как позволяет просматривать текущую страницу в середине тестового набора.

## Взять значения

### grabTextFrom

Находит и возвращает текстовое содержимое данного элемента. Если используется нечеткий локатор, элемент найден с помощью CSS, XPath или 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

### grabAttributeFrom

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

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

* param $cssOrXpath
* param $attribute

### grabCookie

Захватывает значение cookie. Вы можете установить дополнительные параметры, как domain,path в массиве который передается в качестве последнего аргумента.

* param $cookie
* param array $params

### 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

### grabMultiple

Захватывает текстовое содержимое или значения атрибутов узлов, совпадающих с $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.

### grabValueFrom

Находит значение для данного поля формы. При использовании нечетких локатор, в поля по имени поля, CSS и xpath-выражения.

<?php  
$name = $I->grabValueFrom('Name');  
$name = $I->grabValueFrom('input[name=username]');  
$name = $I->grabValueFrom('descendant-or-self::form/descendant::input[@name = 'username']');  
$name = $I->grabValueFrom(['name' => 'username']);  
?>

* param $field

## Разное

### acceptPopup

Accepts the active JavaScript native popup window, as created by window.alert|window.confirm|window.prompt. Don’t confuse popups with modal windows, as created by [various libraries](http://jster.net/category/windows-modals-popups).

### cancelPopup

Отклоняет всплывающее окно JavaScript, созданное window.alert, window.confirm, или window.prompt.

### closeTab

Closes current browser tab and switches to previous active tab.

<?php  
$I->closeTab();

Can’t be used with PhantomJS

### debugWebDriverLogs

Print out latest Selenium Logs in debug mode

* param TestInterface $test

### executeInSelenium

Low-level API method. If Codeception commands are not enough, this allows you to use Selenium WebDriver methods directly:

$I->executeInSelenium(function(\Facebook\WebDriver\Remote\RemoteWebDriver $webdriver) {  
 $webdriver->get('http://google.com');  
});

This runs in the context of the [RemoteWebDriver class](https://github.com/facebook/php-webdriver/blob/master/lib/remote/RemoteWebDriver.php). Try not to use this command on a regular basis. If Codeception lacks a feature you need, please implement it and submit a patch.

* param callable $function

### executeJS

Executes custom JavaScript.

This example uses jQuery to get a value and assigns that value to a PHP variable:

<?php  
$myVar = $I->executeJS('return $("#myField").val()');  
?>

* param $script

### loadSessionSnapshot

* param string $name
* return bool

### makeScreenshot

Takes a screenshot of the current window and saves it to tests/\_output/debug.

<?php  
$I->amOnPage('/user/edit');  
$I->makeScreenshot('edit\_page');  
// saved to: tests/\_output/debug/edit\_page.png  
$I->makeScreenshot();  
// saved to: tests/\_output/debug/2017-05-26\_14-24-11\_4b3403665fea6.png

* param $name

### performOn

Ожидает элемента и выполняет последовательность действий внутри его контекста. Действия могут быть определены с помощью массива, обратного вызова, или Codeception\Util\ActionSequence instance.

Действия как массив рекомендуется для простого сочетания "waitForElement" с утверждениями;waitForElement($el) and see('text', $el) может быть упрощен до:

<?php  
$I->performOn($el, ['see' => 'text']);

Список действий можно прагматично строить с помощью Codeception\Util\ActionSequence:

<?php  
$I->performOn('.model', ActionSequence::build()  
 ->see('Warning')  
 ->see('Are you sure you want to delete this?')  
 ->click('Yes')  
);

Действия, выполняемые из массива или ActionSequence, будут печатать Выходные данные отладки для действий и добавят имя действия к исключению при сбое.

Всякий раз, когда вам нужно определить больше действий обратный вызов может быть использован. Модуль с webdriver передается в качестве аргумента:

<?php  
$I->performOn('.rememberMe', function (WebDriver $I) {  
 $I->see('Remember me next time');  
 $I->seeElement('#LoginForm\_rememberMe');  
 $I->dontSee('Login');  
});

In 3rd argument you can set number a seconds to wait for element to appear

* param $element
* param $actions
* param int $timeout

### reloadPage

Reloads the current page.

### saveSessionSnapshot

* param string $name

### 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

### 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

### \_backupSession

hidden API method, expected to be used from Helper classes

Возвращает текущий сеанс WebDriver для сохранения

* return RemoteWebDriver

### \_capabilities

hidden API method, expected to be used from Helper classes

Change capabilities of WebDriver. Should be executed before starting a new browser session. This method expects a function to be passed which returns array or [WebDriver Desired Capabilities](https://github.com/facebook/php-webdriver/blob/community/lib/Remote/DesiredCapabilities.php) object. Additional [Chrome options](https://github.com/facebook/php-webdriver/wiki/ChromeOptions) (like adding extensions) can be passed as well.

<?php // in helper  
public function \_before(TestInterface $test)  
{  
 $this->getModule('WebDriver')->\_capabilities(function($currentCapabilities) {  
 // or new \Facebook\WebDriver\Remote\DesiredCapabilities();  
 return \Facebook\WebDriver\Remote\DesiredCapabilities::firefox();  
 });  
}

to make this work load \Helper\Acceptance before WebDriver in acceptance.suite.yml:

modules:  
 enabled:  
 - \Helper\Acceptance  
 - WebDriver

For instance, [**BrowserStack** cloud service](https://www.browserstack.com/automate/capabilities) may require a test name to be set in capabilities. This is how it can be done via \_capabilities method from Helper\Acceptance:

<?php // inside Helper\Acceptance  
public function \_before(TestInterface $test)  
{  
 $name = $test->getMetadata()->getName();  
 $this->getModule('WebDriver')->\_capabilities(function($currentCapabilities) use ($name) {  
 $currentCapabilities['name'] = $name;  
 return $currentCapabilities;  
 });  
}

In this case, please ensure that \Helper\Acceptance is loaded before WebDriver so new capabilities could be applied.

* param \Closure $capabilityFunction

### \_closeSession

hidden API method, expected to be used from Helper classes

Manually closes current WebDriver session.

<?php  
$this->getModule('WebDriver')->\_closeSession();  
  
// close a specific session  
$webDriver = $this->getModule('WebDriver')->webDriver;  
$this->getModule('WebDriver')->\_closeSession($webDriver);

* param $webDriver (optional) a specific webdriver session instance

**\_findClickable**

hidden API method, expected to be used from Helper classes

Locates a clickable element.

Use it in Helpers or GroupObject or Extension classes:

<?php  
$module = $this->getModule('WebDriver');  
$page = $module->webDriver;  
  
// search a link or button on a page  
$el = $module->\_findClickable($page, 'Click Me');  
  
// search a link or button within an element  
$topBar = $module->\_findElements('.top-bar')[0];  
$el = $module->\_findClickable($topBar, 'Click Me');

* param $page WebDriver instance or an element to search within
* param $link a link text or locator to click
* return WebDriverElement

**\_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('WebDriver')->\_findElements('.items');  
$els = $this->getModule('WebDriver')->\_findElements(['name' => 'username']);  
  
$editLinks = $this->getModule('WebDriver')->\_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

**\_getCurrentUri**

hidden API method, expected to be used from Helper classes

Uri of currently opened page.

* return string @throws ModuleException

**\_getUrl**

hidden API method, expected to be used from Helper classes

Returns URL of a host.

@throws ModuleConfigException

**\_initializeSession**

hidden API method, expected to be used from Helper classes

Manually starts a new browser session.

<?php  
$this->getModule('WebDriver')->\_initializeSession();

**\_loadSession**

hidden API method, expected to be used from Helper classes

Loads current RemoteWebDriver instance as a session

* param RemoteWebDriver $session

**\_restart**

hidden API method, expected to be used from Helper classes

Перезапускает веб-браузер. Может использоваться с \_reconfigure для открытия браузера с другой конфигурацией

<?php  
// inside a Helper  
$this->getModule('WebDriver')->\_restart(); // just restart  
$this->getModule('WebDriver')->\_restart(['browser' => $browser]); // reconfigure + restart

* param array $config

**\_savePageSource**

hidden API method, expected to be used from Helper classes

Сохраняет исходный HTML-код страницы в файл

* param $filename

**\_saveScreenshot**

hidden API method, expected to be used from Helper classes

Сохранение скриншота текущей страницы в файл

$this->getModule('WebDriver')->\_saveScreenshot(codecept\_output\_dir().'screenshot\_1.png');

* param $filename