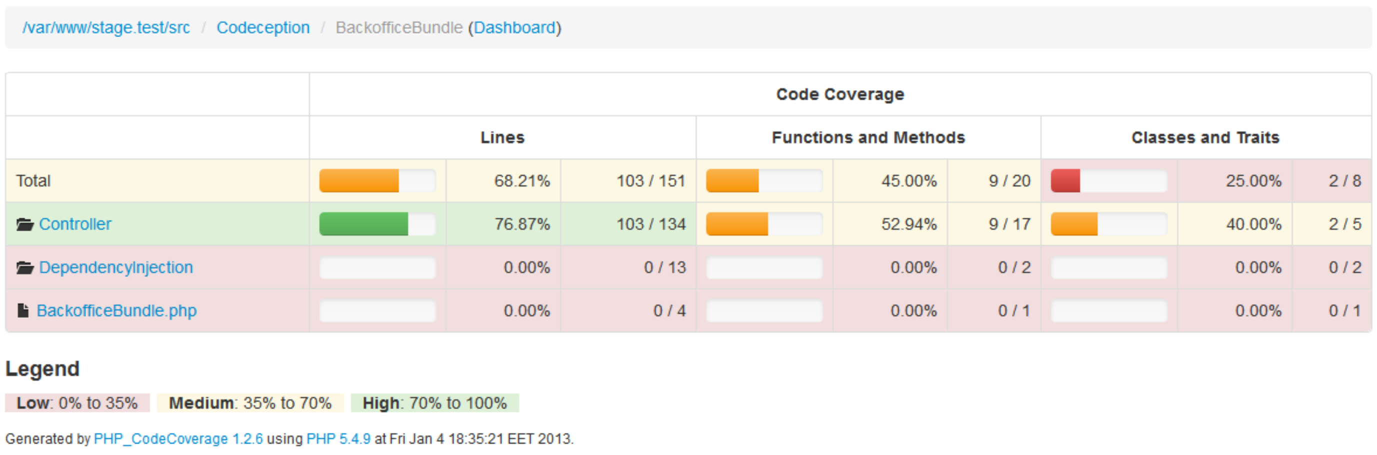
live search...

Code Coverage

At some point you want to review which parts of your application are tested well and which are not. Just for this case the CodeCoverage (http://en.wikipedia.org/wiki/Code_coverage) is used. When you execute your tests to collect coverage report, you will receive statistics of all classes, methods, and lines triggered by these tests. The ratio between all lines in script and all touched lines is a main coverage criterion. In the ideal world you should get 100% code coverage, but in reality 80% is really enough. Because even 100% code coverage rate doesn't save you from fatal errors and crashes.

*To collect coverage information `xdebug` is required**.



/var/www/stage.test/src / Codeception / BackofficeBundle (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		68.21%	103 / 151		45.00%	9 / 20		25.00%	2 / 8
Controller		76.87%	103 / 134		52.94%	9 / 17		40.00%	2 / 5
DependencyInjection		0.00%	0 / 13		0.00%	0 / 2		0.00%	0 / 2
BackofficeBundle.php		0.00%	0 / 4		0.00%	0 / 1		0.00%	0 / 1

Legend
Low: 0% to 35% **Medium**: 35% to 70% **High**: 70% to 100%

Generated by PHP_CodeCoverage 1.2.6 using PHP 5.4.9 at Fri Jan 4 18:35:21 EET 2013.

Coverage data can be collected manually for both local and remote tests. Remote tests may be executed on different nodes, or locally but running through web server. It may look hard to collect code coverage for Selenium tests or PhpBrowser tests. But Codeception supports remote codecoverage as well as local.

Configuration

Follow @codeception

To enable code coverage put these lines in the global configuration file `codeception.yml`:

```
coverage:
    enabled: true
```

That’s ok for now. But what files should be present in final coverage report? Pass an array of files or directory to include/exclude sections. The path ending with ‘*’ matches the directory. Also you can use ‘*’ mask in a file name, i.e. `app/models/*Model.php` to match all models.

There is a shortcut if you don’t need that complex filters:

```
coverage:
    enabled: true
    include:
        - app/*
    exclude:
        - app/cache/*
```

Include and exclude options can be redefined for each suite in corresponding config files.

By default, if coverage is reported to be < 35% it is marked as low, and >70% is high coverage. You can also define high and low boundaries with `low_limit` and `high_limit` config options:

```
coverage:
    enabled: true
    low_limit: 30
    high_limit: 60
```

Local CodeCoverage

The basic codecoverage can be collected for functional and unit tests. If you performed configuration steps from above, you are ready to go. All you need is to execute codeception with `--coverage` option.

To generate a clover xml report or a tasty html report append also `--coverage-xml` and `--coverage-html` options.

```
codecept run --coverage --coverage-xml --coverage-html
```

Follow @codeception

XML and HTML reports are stored to the `_output` directory. The best way to review report is to open `index.html` from `tests/_output/coverage` in your browser. XML clover reports are used by IDEs (like PHPStorm) or Continuous Integration servers (like Jenkins).

Remote CodeCoverage

Local Server

If you run your application via Webserver (Apache, Nginx, PHP WebServer) you don't have a direct access to tested code, so collecting coverage becomes a non-trivial task. The same goes for scripts that are tested on different nodes. To get access to this code you need `xdebug` installed with `remote_enable` option turned on. Codeception also requires a little spy to interact with your application. As your application runs standalone, without even knowing it is being tested, a small file should be included in order to collect coverage info.

This file is called `c3.php` and is available on GitHub (https://github.com/Codeception/c3). `c3.php` should be downloaded and included in your application at the very first line from controller. By sending special headers Codeception will command your application when to start codecoverage collection and when to stop it. After the suite is finished, a report will be stored and Codeception will grab it from your application.

Please, follow installation instructions described in a readme file (https://github.com/Codeception/c3).

To connect to `c3` Codeception uses url config from PhpBrowser or WebDriver module. But URL of index with `c3.php` included can be specified explicitly with `c3_url` parameter defined:

```
coverage:
    # url of file which includes c3 router.
    c3_url: 'http://127.0.0.1:8000/index-test.php/'
```

After the `c3.php` file is included in application you can start gather coverage. In case you execute your application locally there is nothing to be changed in config. All codecoverage reports will be collected as usual and merged afterwards. Think of it: Codeception runs remote coverage in the same way as local.

Remote Server

Follow @codeception

But if you run tests on different server (or your webserver doesn't use code from current directory) a single option `remote` should be added to config. For example, let's turn on remote coverage for acceptance suite in `acceptance.suite.yml`:

```
coverage:
    remote: true
```

In this case remote Code Coverage results won't be merged with local ones, if this option is enabled. Merging is possible only in case a remote and local files have the same path. But in case of running tests on a remote server we are not sure of it.

CodeCoverage results from remote server will be saved to `tests/_output` directory. Please note that remote codecoverage results won't be displayed in console by the reason mentioned above: local and remote results can't be merged, and console displays results for local codecoverage.

Remote Context Options

HTML report generation can at times take a little more time than the default 30 second timeout. Or maybe you want to alter SSL settings (verify_peer, for example) To alter the way c3 sends it's service requests to your webserver (be it a local or a remote one), you can use the `remote_context_options` key in `coverage` settings.

```
coverage:
    remote_context_options:
        http:
            timeout: 60
        ssl:
            verify_peer: false
```

Context stream options are well documented at php.net (http://php.net/manual/en/context.php)

Conclusion

It's never been easier to setup local and remote code coverage. Just one config and one additional file to include! **With Codeception you can easily generate CodeCoverage reports for your Selenium tests** (or other acceptance or api tests). Mixing reports for `acceptance`, `functional`, and `unit` suites provides you with the most complete information on which parts of your applications are tested and which are not.

- **Next Chapter: ContinuousIntegration > (/docs/12-ContinuousIntegrat**

Follow @codeception

- **Previous Chapter: < WebServices (/docs/10-WebServices)**

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

Thanks to

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

(http://robo.li)

Robo

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

Follow @codeception

(http://codecept.io)

CodeceptJS

(http://codecept.io)

Codeception for **NodeJS**. Write acceptance tests in ES6 and execute in webdriverio, Selenium WebDriver, and Protractor.

Follow @codeception