## Анализ покрытия кода тестами

Этот анализ позволяет посмотеть, какие участки кода отработали в ходе тестирования. В Ubuntu и Debian устанавливаем стандартно:

aptitude install php5-xdebug

В Windows нужно в php.ini добавить

[xdebug]

zend\_extension\_ts=C:/php/ext/php\_xdebug.dll

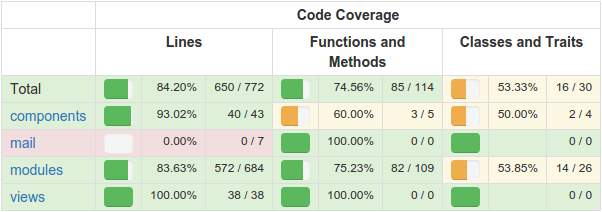
Но если в ваших тестах используется $this->specify(...), то будет постоянно вываливаться ошибка из-за ограничений уровня вложенности. Для решения проблемы нужно открыть php.ini или /etc/php/conf.d/xdebug.ini и дописать:

xdebug.max\_nesting\_level = 1000

С установкой всё. Попробуем запустить всё снова, но с анализом покрытия:

codecept run unit,functional --coverage-html

И после завершения посмотрим \_output/coverage/index.html:

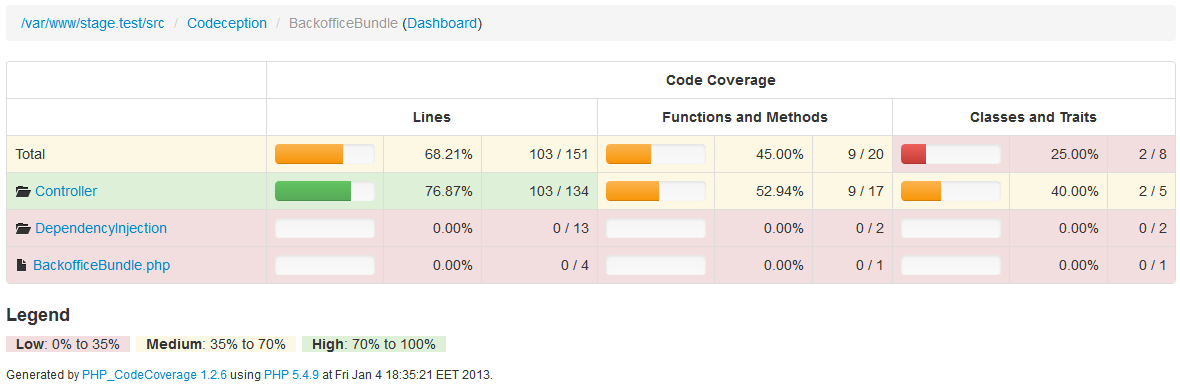


Это удобно тем, что можно посмотреть, какие фрагменты кода не выполнялись и дописать проверки для них. Теперь нам не страшно будет вносить правки в исходный код. В любой момент можно проверить правильность работы проекта.

## Покрытие кода тестами (Code Coverage)

В какой то момент вам захочется узнать, какие части приложения покрыты тестами, а какие нет.  
Именно для этого и используется [CodeCoverage](http://en.wikipedia.org/wiki/Code_coverage). Во время выполнения тестов для сбора данных о покрытии, вы получите статистику по всем классам, методам и строкам кода, которые затронуты вашими тестами.  
Разница между количеством строк кода и количеством затронутых при тестировании строк является главным критерием покрытия кода тестами. В идеальном мире покрытие кода должно составлять 100%, однако в реальности достаточно и 80%. Однако, даже 100% покрытие кода не защитит вас от ошибок и падения приложения.

**Codeception включает инструменты CodeCoverage начиная с версии 1.5. Для сбора информации о покрытии тестами необходимxdebug**.



Данные о покрытии тестами могут быть собраны вручную для локальных или удаленных тестов. Удаленные тесты могут выполняться на другом узле,  
или локально, однако с использованием веб сервера. Может показаться сложным получение покрытия кода для тестов Selenium или PhpBrowser. Однако Codeception  
поддерживает удаленное покрытие точно так же как и локальное.

### Конфигурация

Чтобы включить утилиту покрытия тестами, добавьте следущие строки в глобальный конфигурационный файл codeception.yml:

coverage:

enabled: true

выглядит неплохо. Но какие файлы должны попасть в отчет о покрытии? Вы можете фильтровать файлы используя black-листы и white-листы.

coverage:

enabled: true

whitelist:

include:

- app/\*

exclude:

- app/cache/\*

blacklist:

include:

- app/controllers/\*

exclude:

- app/cache/CacheProvider.php

Что такое black-листы и white-листы?

* Список **whitelist** это список файлов которые должны быть включены в отчет, даже если они не затронуты.
* Список **blacklist** это список файлов которые должны быть исключены из отчета, даже если они затронуты.

Передайте массив файлов/директорий в секции include/exclude. Путь заканчивающийся на '' относится к директории.Так же вы можете использовать маску '' в именах файлов, например app/models/\*Model.php для указания всех моделей.

Вот пример, если вам не нужны сложные фильтры:

coverage:

enabled: true

include:

- app/\*

exclude:

- app/cache/\*

Опции include и exclude здесь добавляют или удаляют файлы из white-листа.

Каждая из этих настроек может быть переопределена для каждого набора в сооветсвующих конфигурационых файлах.

## Локальное покрытие тестами

Базовый отчет о покрытии может быть получен для функциональных и модульных тестов.  
Если вы настроили конфигурацию как было показано выше, вы готовы к действиям.  
Все что вам нужно выполнить codeception с опцией --coverage.  
Для генерации xml отчета или красивого html отчета добавьте опции --xml и --html.

codecept run --coverage --xml --html

XML и HTML отчеты распологаются в директории \_logs. Лучший способ посмотреть отчет, открыть index.html находящийся в директорииtests/\_logs/coverage с помощью вашего браузера.  
XML отчеты используются в IDEs (такими как PHPStorm) или серверами Continuous Integration (Такими как Jenkins).

## Удаленное покрытие тестами

В случае, есои ваше приложение запущено с помощью веб-сервера (Apache, Nginx, PHP WebServer) у вас нет доступа к тестируемому коду,  
таким образом сбор данных о покрытии кода становится нетривиальной задачей. То же самое каается скриптов запущенных на другом узле.  
Для того чтобы получить доступ к коду, у вас должно быть установлено расширение xdebug с включенной опцией remote\_enable.  
Codeception так же требует наличия небольшого "шпиона" для взаимодействия с вашим приложением.  
Даже если ваше приложение работает автономно, даже не зная что оно было протестировано, данный файл должен быть подключен для того чтобы стало возможным собирать информацию о покрытии тестами.

Этот файл называется c3.php и он [доступен на GitHub](https://github.com/Codeception/c3).  
c3.php должен быть скачан и включен в ваше приложение в первой строке of it's from controller.  
Посылая специальные заголовки Codeception будет отдавать команды вашему приложению, позволяющие запускать сбор данных о покрытии кода и останавливать его.  
После выполнения набора тестов, отчет будет сохранен и Codeception считает его с вашего приложения.

Следуйте инструкциям по установке описанным в файле [readme](https://github.com/Codeception/c3).

После того как файл c3.php будет включен в ваше приложение, вы сможете начать сбор данных о покрытии кода тестами.  
В случае, если вы запускаете приложение локально, не нужно производить ни каких дополнитльных действий в конфигурации.  
Все данные будут собраны, после чего скомпонованы.  
Подумайте над этим: Codeception запускает удаленно утилиту сбора покрытия кода тестами, точно так же как и локально.

Никогда настройка удаленного формирования отчета о покрытии кода тестами не была такой простой. В любом другом фреймворке. Серьезно.

Однако, если вы запускаете тесты на удаленном сервере (или ваш веб-сервер не использует код из текущей директории), в конфигурационный файл необходимо добавить опцию remote.  
К примеру давайте включим удаленную утилиту сбора информации о покрытии кода тестами в набор acceptance.suite.yml

coverage:

enabled: true

remote: true

В данном случае результаты отчеты не будут слиты с локальными, если опция включена.  
Объединение отчетов возможно только в том случае, если файл запущеный удаленно и локально имеет один путь.  
Однако в случае запуска тестов на удаленно сервере нет уверенности, что это так.

**Настройка покрытия кода**

Для формирования HTML и XML отчётов включаем утилиту coverage в файле codeception.yml и указываем какие файлы должны попасть в отчет:

coverage:

enabled: true

whitelist:

include:

- app/\*

exclude:

- app/commands/\*

- app/config/\*

- app/migrations/\*

blacklist:

include:

- app/migrations/\*

exclude:

- app/services/Sendler.php

Где whitelist - это перечень файлов которые будут включены в отчет (даже если не затронуты), а blacklist - это перечень файлов которые не будут включены в отчет (даже если затронуты). Опции include и exclude здесь добавляют или удаляют файлы.

Запускаем тесты с формированием отчетов:

$ ./vendor/bin/codecept run --coverage --html --xml

После успешного выполнения тестов мы видим что-то подобное:

- HTML report generated in file:///home/user/projects/projectName/tests/\_output/report.html

- XML report generated in file:///home/user/projects/projectName/tests/\_output/report.xml

Это адреса файлов с отчетами тестирования. Можем перейти и посмотреть на них. Но если мы хотим посмотреть отчет о покрытии тестами файлов проекта, тогда запускаем тесты так:

$ ./vendor/bin/codecept run --coverage-html

А файл с отчетом смотрим по адресу projectName/tests/\_output/coverage/index.html.