Работа с данными

Тесты не должны ломать другие тесты. Таковы правила. Тесты могут изменять данные при работе с БД. Это может привести к рассогласованию данных. Тест может попытаться вставить запись которая уже существует, или восстановить уже удаленную запись. Для предотвращения провала тестов по данным причинам, база данных должна быть возвращена в исходное состояние. Codeception использует несколько разных методов и подходов для реализации такой возможности.

Данный раздел объединяет всю информацию об работе с данными из предыдущих разделов и предлагает лучшие решения для управления базами данных в ваших тестах.

Когда мы собираемся очистить базу данных, мы должны сделать это как можно быстрее. Тесты всегда должны выполняться быстро. Восстановление базы с нуля не лучший путь, хотя иногда единственный. В любом случае, вы должны использовать отдельную базу данных для тестирования. Не в коем случае не используйте development или production базы при тестировании!

Ручная очистка

Ни что не мешает вам создавать записи перед выполнением теста и удалять их после. Это довольно хорошее решение, если вы не имеете совместно используемых несколькими тестам данных.  
Однако не следует добавлять подобный код в файл теста. Все потому, что тесты выполняются дважды: с целью анализа и для непосредственно выполнения, таким образом это может привести к неожиданным результатам. Таким образом вы должны указать место, где ваш код будет выполнен. Хорошая идея, добавить данные перед анализом и удалить после того как тест выполнен. Используйте специальные методы объекта $scenario: running() и preload() для определения текущего состояния.

preload()) {

$user = new User();

$user->name('davert');

$user->save();

}

$I = new WebGuy($scenario);

$I->amOnPage('/admin/users');

$I->see('User: '. $user->name);

if ($scenario->running()) {

$user->delete();

}

?>

Таким образом вы можете вставить любой код в ваши тесты. Однако не забывайте указывать шаг на котором ваш код должен быть включен, иначе он будет выполнен дважды!

Автоматическая очистка

Codeception имеет Db модуль, позволяющий производить с базой данных большинство необходимых действий. По умолчанию он будет пытаться заполнить базу, данными из дампа, и очистить ее после выполнения каждого теста. Данный модуль использует дампы баз данных в SQL формате. Он включен в файл конфигурации codeception.yml и готов для настройки.

modules:

config:

Db:

dsn: 'PDO DSN HERE'

user: 'root'

password:

dump: tests/\_data/your-dump-name.sql

После включения данного модуля он автоматически будет заполнять базу из дампа и перезаполнять ее после запуска каждого теста. Данные настройки могут быть изменены с помощью опций *populate* and *cleanup*, которые могут быть отключены.

Модуль Db довольно полезный инструмент. Он работает с любыми базами данных пооддерживающими PDO. Он может быть использован с любыми тестами, если они не являются довольно медленными. Загрузка дампа может занять продолжительное время, поэтому для данной цели могут быть использованы и другие подходы. В случае, если ваши тесты используют одно подключение, как это может быть при функциональном и модульном тестировании, лучший способ ускорить их, использование транзакций или базы SQLite. Когда для работы с базами данных используются несколько подключений, как это бывает в приемочных тестах, лучшим решением является использование базы данных SQLite, с последующие ее заменой после каждого теста, вместо заполнения и очистки.

Независимые подключения

В приемочных тестах, ваши тесты взаимодействуют с приложением через веб-сервер. В данном случае нет возможности получить экземпляр подключения к базе данных от сервера. Это значит, что тесты и приложение будут работать с одной базой данных но использую разные подключения. Указав в модуле Db такие же настройки подключения к базе данных что и у приложения появляется возможность проверять данные в базе (seeInDatabase действия) и использовать автоматическую очистку базы данных.

Ускорение с помощью SQLite

Для ускорения ваших тестов мы рекомендуем вам попробовать использовать SQLite. Для конвертации вашего дампа в формат SQLite. Используйте одну из [описанных утилит](http://www.sqlite.org/cvstrac/wiki?p=ConverterTools).

Помните, что поля в SQLite чувствительны к регистру, поэтому необходимо установить COLLATE NOCASE для каждого text и varchar поля.

Сохраните SQLite базу данных в каталог **tests/\_data**, и укажите ее в модуле Db:

modules:

config:

Db:

dsn: 'sqlite:tests/\_data/application.db'

user: 'root'

password:

dump: tests/\_data/sqlite\_dump.sql

Перед запуском тестов база данных SQLite будет создана и скопирована. После выполнения каждого теста, копия заменит базу данных.

Совместные подключения

Когда ваше приложение или его часть запущены внутри Codeception, вы можете использовать подключение созданное вашим приложением внутри тестов.  
Если вы может получить доступ к подключению, значит все операции могут быть вынесены в одну глобальную транзакцию и могут быть отменены после выполнения тестов. Это значительно увеличит быстродействие. Ни чего не будет записано в базу данных по окончанию тестирования, таким образом перезаполнение базы данных больше не нужно.

ORM модули

Если ваше приложение использует ORM, такие как Doctrine или Doctrine2, подключите соответствующие модули Codeception к вашим тестам. По умолчанию они будут оборачивать все в транзакции. Если вы используете несколько подключений или транзакции не поддерживаются используемой ORM, данный модуль будет для вас бесполезен.

ORM модуль может быть подключен используя Db модуль, но по умолчанию оба будут выполнять очистку. Поэтому вы должны конкретно указать, какой модуль использовать:

В **tests/functional.suite.yml**:

modules:

enabled: [Db, Doctrine1, TestHelper]

config:

Db:

cleanup: false

В данном случае модуль Db все еще будет заполнять базу перед выполнением каждого теста. Используйте populate: false чтобы отключить эту возможность.

Dbh модуль

Если вы используете PostgreSQL, или любую другую базу данных поддерживающую вложенные транзакции вы можете использовать модуль Dbh. Он получает PDO объект из вашего приложения, запускает транзакции в начале теста, и откатывает их по его окончанию.  
PDO подключение может быть установлено в bootstrap файле. Кроме того, этот модуль перезаписывает действия seeInDatabase иdontSeeInDatabase модуля Db.

Для использования Db модуля для заполнения данных и Dbh для очистки, используйте следующий конфигурационный файл:

modules:

enabled: [Db, Dbh, TestHelper]

config:

Db:

cleanup: false

Запомните, модуль Dbh должен быть указан после модуля Db. Это позволит модулю Dbh перезаписывать действия.

Фикстуры (Fixtures)

Фикстуры - это наборы данных используемых тестами. Эти данные могут быть сгенерированы или получены из базы данных. Фикстуры должны быть заданы в bootstrap файле. В зависимости от типов тестов suite, объявление фикстур может отличаться.

Фикстуры для Acceptance и Functional тестов

Использование фикстур в acceptance и functional тестах могут быть довольно просто определены и использованы.

Объявление фикстуры в \_\_bootstrap.php\_

findOneBy('name', 'davert');

?>

Использование фикстуры в acceptance или functional тестах.

<?php

$I = new TestGuy($scenario);

$I->amLoggedAs($davert);

$I->see('Welcome, Davert');

?>

Все переменные из bootstrap передаются *Cept* файлам в тестового набора.

Вы можете использовать библиотеку [Faker](https://github.com/fzaninotto/Faker) для создания тестовых данных в bootstrap файле.

Фикстуры в Unit тестах.

Передача фикстур *Cest* файлам немного сложнее, потому как вы не можете передать переменную внутрь класса. Поэтому для данных целей случит специальный класс Fixtures.

Объявление фикстур в файле \_\_bootstrap.php\_ для unit тестов.

<?php

use \Codeception\Util\Fixtures as Fixtures;

Fixtures::add('davert', Doctrine::getTable('User')->findOneBy('name', 'davert'));

?>

Использование в простом unit тесте:

<?php

$I->execute(function () {

$user = Fixtures::get('davert');

return $user->getName();

});

$I->seeResultEquals('davert');

?>

Классы фикстур хранят и получают значение любой переменной по ее ключу. Вы можете расширить это при помощи добавления пространства имен.

В файле \_\_bootstrap.yml\_

findOneBy('name', 'davert'));

?>

Внутри теста:

<?php

$I->execute(function () {

$user = Fixtures::get('user','davert');

// or

$user = Fixtures::getUser('davert');

return $user->getName();

});

$I->seeResultEquals('davert');

?>

Использование фикстур упрощает ваш тестовый код. Именуйте фикстуры со смыслом и ваши тесты станут значительно читабельней.