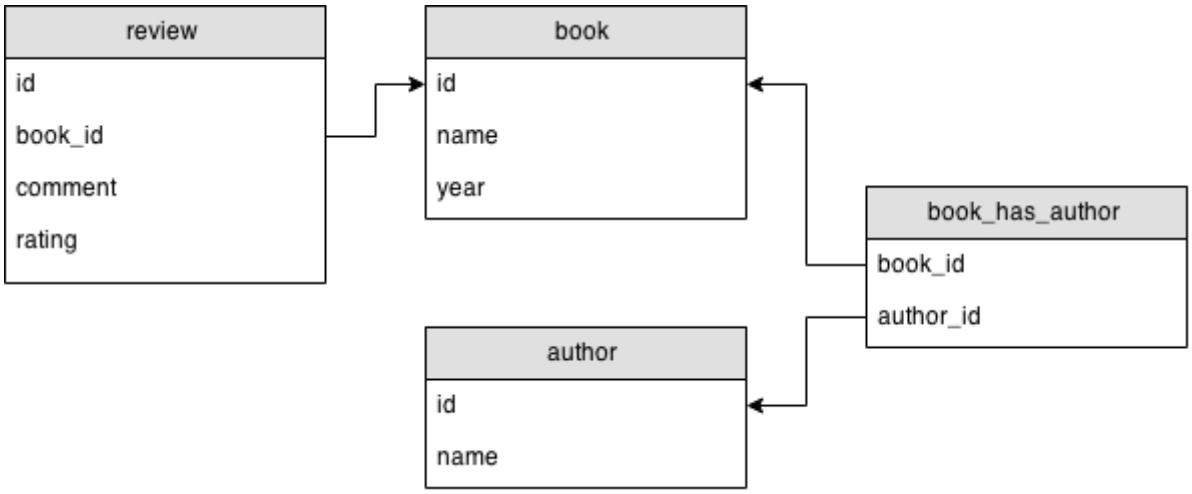
Модульное тестирование поведения Yii2 с помощью Codeception

Что тестируем?

Поведение, которое мы рассматриваем, позволяет при сохранении модели также сохранять ее связи с другими моделями. Для примера рассмотрим простую структуру данных, состоящую из книг (Book), авторов (Author) и отзывов на книги (Review). Книги и авторы связаны как N-N, то есть у книги может быть много авторов, а у автора — много книг. Книги и отзывы связаны как 1-N, то есть у книги может быть много отзывов, но каждый отзыв может относиться только к одной книге.



Во время тестирования мы будем сохранять одну книгу со всеми ее связями. При сохранении модели нужно рассмотреть несколько возможных вариантов входных данных:

1. Непустой массив идентификаторов связанной модели. При этом должны быть удалены старые связи и созданы новые.
2. Пустой массив, в результате чего должны быть удалены старые связи.
3. Полное отсутствие данных, соответствующее, например, форме редактирования книги, в которой нет полей, относящихся к авторам. В этом случае поведение не должно делать ничего, то есть существующие связи не должны измениться.

Необходимо проверить работоспособность поведения во всех трех случаях.

Особенности, связанные с Yii2

Так как поведение предназначено для совместной работы с Yii2, тестировать его без остального фреймворка нет смысла. Для тестирования мы фактически создадим консольное приложение Yii2, а в нем будем оперировать моделями. Считаем модель из базы данных, передадим ей нужные параметры, сохраним, снова считаем из базы, и проверим, правильно ли она сохранилась.

Разумеется, для тестирования нам понадобится база данных. Сами же тестовые данные будут храниться в виде дампа, который загружается перед каждым тестом.

Настройка Codeception

Для начала с помощью composer выполним глобальную установку codeception:

```
composer global require codeception/codeception
```

Теперь подготовим все необходимое для тестирования нашего поведения. В директории с поведением уже есть файл composer.json, в котором дано описание поведения и его зависимостей. Добавим к нему библиотеку yii2-codeception:

```
composer require --dev yiisoft/yii2-codeception
```

Затем выполним инициализацию окружения codeception в директории поведения:

```
codecept bootstrap --customize
```

Имя актора (actor) можно оставить по умолчанию (Tester), а набор тестов (suite) нам понадобится только один — unit.

Появится директория tests и файл codeception.yml, в котором мы зададим нужные нам параметры. Параметры по умолчанию нас вполне устраивают, за исключением подключения к базе данных.

```
actor: Tester
paths:
    tests: tests
    log: tests/_output
    data: tests/_data
    helpers: tests/_support
settings:
    bootstrap: _bootstrap.php
    colors: false
    memory_limit: 1024M
modules:
    config:
        Db:
            dsn: 'sqlite:tests/_output/temp.db'
            user: ''
            password: ''
            dump: tests/_data/dump.sql
```

Теперь нужно настроить набор тестов unit в файле tests/unit.suite.yml:

```
class_name: UnitTester
modules:
    enabled: [Asserts, Db]
```

Модуль UnitHelper, который был включен по умолчанию, нам не понадобится, зато мы добавили Asserts и Db. Теперь построим окружение с учетом выбранных модулей:

```
codecept build
```

Наконец, нужно настроить автозагрузчик Yii2 в файле tests/_bootstrap.php:

```php
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');
require_once __DIR__ . implode(DIRECTORY_SEPARATOR, ['', '..', 'vendor', 'autoload.php']);
require_once __DIR__ . implode(DIRECTORY_SEPARATOR, ['', '..', 'vendor', 'yiisoft', 'yii2', 'Yi
i.php']);
Yii::setAlias('@tests', __DIR__);
Yii::setAlias('@data', __DIR__ . DIRECTORY_SEPARATOR . '_data');
```

Перед тем, как писать тесты, нужно подготовить дамп базы данных и создать классы моделей.

Создаем таблицы book, author, review и book_has_author, заполняем их тестовыми данными. Затем делаем дамп и сохраняем его в tests/_data/dump.sql.

```sql
BEGIN TRANSACTION;
CREATE TABLE "review" (
    `id`    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    `book_id`   INTEGER,
    `comment`   VARCHAR(150) NOT NULL,
    `rating`    INTEGER NOT NULL
);
INSERT INTO `review` VALUES (1,3,'Старая книга, не потерявшая актуальность.',5);
INSERT INTO `review` VALUES (2,3,'Одобряю!',5);
INSERT INTO `review` VALUES (3,3,'Неплохо.',4);
INSERT INTO `review` VALUES (4,5,'Хлам!',2);
CREATE TABLE "book_has_author" (
    `book_id`   INTEGER NOT NULL,
    `author_id` INTEGER NOT NULL
);
INSERT INTO `book_has_author` VALUES (1,1);
INSERT INTO `book_has_author` VALUES (1,2);
INSERT INTO `book_has_author` VALUES (2,1);
INSERT INTO `book_has_author` VALUES (2,3);
INSERT INTO `book_has_author` VALUES (3,4);
INSERT INTO `book_has_author` VALUES (4,5);
INSERT INTO `book_has_author` VALUES (4,6);
INSERT INTO `book_has_author` VALUES (5,9);
CREATE TABLE "book" (
    `id`    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    `name`  VARCHAR(150) NOT NULL,
    `year`  INTEGER NOT NULL
);
INSERT INTO `book` VALUES (1,'Основы агрономии и ботаники.',2004);
INSERT INTO `book` VALUES (2,'Ботаника: учеб для с/вузов.',2005);
INSERT INTO `book` VALUES (3,'Краткий словарь ботанических терминов.',1964);
INSERT INTO `book` VALUES (4,'Ботаника с основами геоботаники.',1979);
INSERT INTO `book` VALUES (5,'Ботаника. Систематика высших или наземных растений.',2004);
CREATE TABLE "author" (
    `id`    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    `name`  VARCHAR(150) NOT NULL
);
INSERT INTO `author` VALUES (1,'Андреев Н.Г.');
INSERT INTO `author` VALUES (2,'Андреев Л.Н.');
INSERT INTO `author` VALUES (3,'Родман Л.С.');
INSERT INTO `author` VALUES (4,'Викторов Д.П.');
INSERT INTO `author` VALUES (5,'Суворов В.В.');
INSERT INTO `author` VALUES (6,'Воронов И.Н.');
INSERT INTO `author` VALUES (7,'Еленевский А.Г.');
INSERT INTO `author` VALUES (8,'Соловьева М.П.');
INSERT INTO `author` VALUES (9,'Тихомиров В.Н.');
COMMIT;
```

Конфигурирование приложения

Так как наше поведение будет тестироваться в рамках консольного приложения, нужно подготовить для него конфигурацию. Создаем файл tests/unit/_config.php:

```php
<?php
return [
    'id' => 'app-console',
    'class' => 'yii\console\Application',
    'basePath' => \Yii::getAlias('@tests'),
    'runtimePath' => \Yii::getAlias('@tests/_output'),
    'bootstrap' => [],
    'components' => [
        'db' => [
            'class' => '\yii\db\Connection',
            'dsn' => 'sqlite:'.\Yii::getAlias('@tests/_output/temp.db'),
            'username' => '',
            'password' => '',
        ]
    ]
];
```

Создание моделей

Файлы классов моделей создаем в директории tests/_data, и задаем им namespace data. Чтобы не делать это вручную, в другой директории развернём шаблон приложения basic, подключим его к базе данных и создадим классы с помощью gii.

Важно, чтобы в модели Book были объявлены нужные отношения:

```php
public function getAuthors()
{
    return $this->hasMany(Author::className(), ['id' => 'book_id'])
                ->viaTable('book_has_author', ['author_id' => 'id']);
}

public function getReviews()
{
    return $this->hasMany(Review::className(), ['book_id' => 'id']);
}
```

Туда же добавляем и поведение:

```php
public function behaviors()
{
return
    [
        [
            'class' => \voskobovich\behaviors\ManyToManyBehavior::className(),
            'relations' => [
                'author_list' => ['authors'],
                'review_list' => ['reviews'],
            ]
        ]
    ];
}
```

Обязательно указываем валидатор для атрибутов, которые создаются поведением:

```php
public function rules()
{
    return [
        [['author_list', 'review_list'], 'safe'],
         ...
```

Создание тестов

В codeception тест-кейсы оформляются как классы. Чтобы работать с объектами Yii2 нужно создать класс, унаследованный от yii\codeception\TestCase. Имя класса и имя файла должны заканчиваться с Test.

В файле tests/unit/BehaviorTest.php создадим тест-кейс BehaviorTest, а в нем метод testSaveManyToMany, проверяющий, сохраняется ли корректный набор данных для связи N-N:

```php
class BehaviorTest extends \yii\codeception\TestCase
{
    public $appConfig = '@tests/unit/_config.php';

    public function testSaveManyToMany()
    {
        //load
        $book = Book::findOne(5);

        //simulate form input
        $post = [
            'Book' => [
                'author_list' => [7, 9, 8]
            ]
        ];

        $this->assertTrue($book->load($post), 'Load POST data');
        $this->assertTrue($book->save(), 'Save model');

        //reload
        $book = Book::findOne(5);

        //must have three authors
        $this->assertEquals(3, count($book->authors), 'Author count after save');

        //must have authors 7, 8, and 9
        $author_keys = array_keys($book->getAuthors()->indexBy('id')->all());
        $this->assertContains(7, $author_keys, 'Saved author exists');
        $this->assertContains(8, $author_keys, 'Saved author exists');
        $this->assertContains(9, $author_keys, 'Saved author exists');
    }

    ...
```

Мы выполняем действия, которые обычно связаны с сохранением формы. Определенные данные приходят из запроса (переменная $post). Метод load() используется для записи этих данных в атрибуты модели. Затем модель сохраняется с помощью метода save().

После наших манипуляций у книги должно появиться три автора с ключами 7, 8 и 9, что и проверяется.

Аналогично описываются и остальные тесты, например, сохранение пустого набора даных для связи 1-N:

```php
public function testResetOneToMany()
{
    //load
    $book = Book::findOne(3);

    //simulate form input
    $post = [
        'Book' => [
            'review_list' => []
        ]
    ];

    $this->assertTrue($book->load($post), 'Load POST data');
    $this->assertTrue($book->save(), 'Save model');

    //reload
    $book = Book::findOne(3);

    //must have zero reviews
    $this->assertEquals(0, count($book->reviews), 'Review count after save');
}
```

Если выполнить codecept run, система проведет все доступные тесты и отчитается об их результатах:

```
Codeception PHP Testing Framework v2.0.11
Powered by PHPUnit 4.5.0 by Sebastian Bergmann and contributors.

Unit Tests (2) ------------------------------------------------------------------------------
------
Test save many to many (BehaviorTest::testSaveManyToMany)
Ok
Test reset one to many (BehaviorTest::testResetOneToMany)
Ok
------------------------------------------------------------------------------------------
------


Time: 390 ms, Memory: 9.00Mb

OK (2 tests, 9 assertions)
```

Полезные ссылки:

1. Модульное тестирование поведения Yii2 с помощью Codeception. https://habr.com/ru/post/254509/
2. Написание тестов для PHPUnit. https://phpunit.de/manual/6.5/en/writing-tests-for-phpunit.html
3. Сервис на Yii2: Тестирование приложения с Codeception. https://elisdn.ru/blog/84/seo-service-on-yii2-application-testing