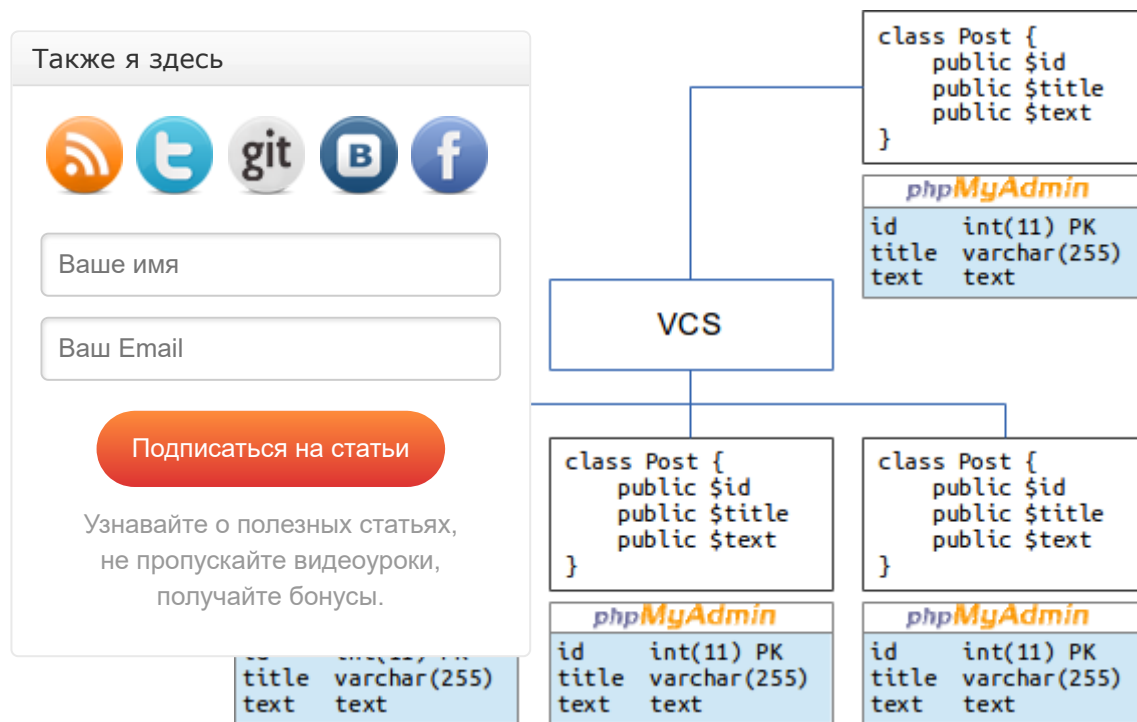


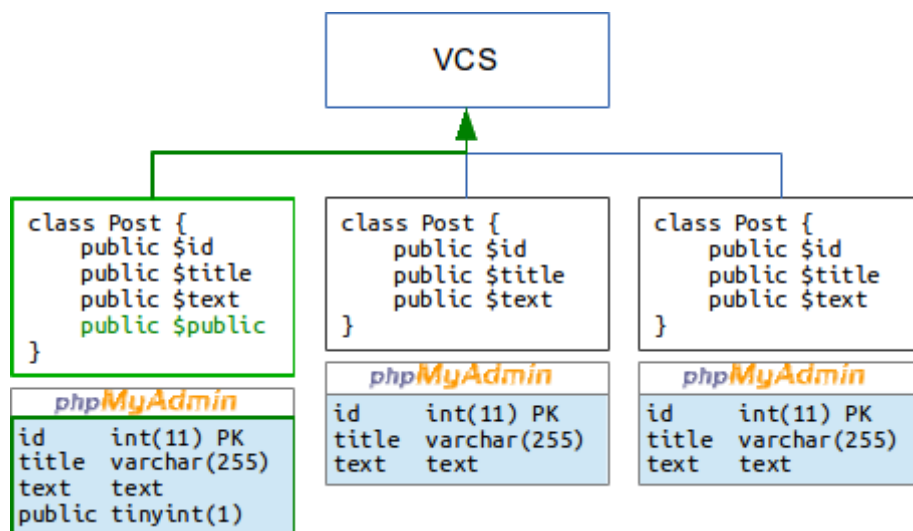
Три программиста пишут код проекта. И с кодом у них нет проблем. Есть три рабочих машины, центральный репозиторий и главный рабочий сервер, берущий файлы из этого же репозитория. Работа кипит. у каждого на своём компьютере установлен PhpMyAdmin, что им позволяет время от времени вносить изменения в свою базу данных.

Программный код спокойно держится в VCS. Любой из троих разработчиков делает pull и сливает к себе из центрального репозитория последнюю версию файлов, что-то дописывает, коммитит и отправляет обратно через push. Теперь остальные два разработчика сливают эти изменения к себе, дорабатывают свои дела и отправляют назад. Такими темпами движется разработка. В репозитории видна история изменения каждого файла и общая цепь правок, оформленных в коммиты. И всё хорошо.

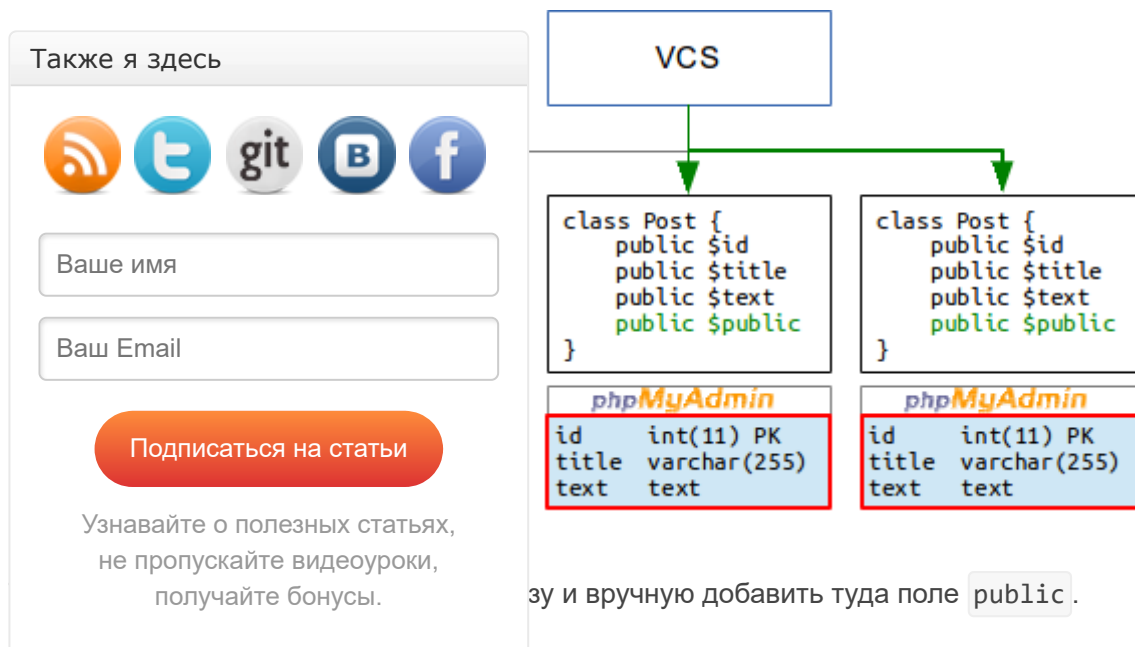


Там же среди файлов лежит SQL файл с дампом исходной базы. Любой новый программист свободно может клонировать проект на свой компьютер и присоединиться к разработке. Актуальную базу данных он может попросить и кого-то ещё или сдмпить с тестового сервера. Ну или если после каждого изменения вручную правится и коммитится дамп со схемой, то взять его последнюю версию.

И вот новый программист для выполнения своей задачи добавил в таблицу базы постов блога новое поле `public` и изменил выборку постов на сайте так, чтобы отображались только открытые записи. Изменение кода модели `Post` он сохранил в новую ревизию и отправил в центральный репозиторий:



Теперь ему надо открыть скайп, зайти в конференцию и попросить всех добавить в свои базы это поле. Иначе после загрузки изменений у них просто-напросто не обнаружится поле `public` и блог работать не будет:



Ну а старший программист при очередной выкладке изменений на рабочий сервер откроет историю переписки в скайпе и сделает то же самое на рабочем сайте.

Хранение изменений в системе контроля версий

Но вдруг менеджер проекта говорит, что в скайпе искать изменения неинтересно, что вручную переименовывать десятки колонок и переносить в них значения слишком муторно, и лучше записывать правки базы SQL-запросами где-то внутри проекта.

Тогда все договариваются либо записывать SQL-запросы с изменениями прямо в сообщения коммитов:

```
git commit -m 'Added public field into post
ALTER TABLE tbl_post ADD COLUMN public tinyint(1) NOT NULL;'
```

либо создать текстовый файл и записывать все правки в него.

Потом кто-то ещё захочет переименовать пару полей в какой-то таблице и снова запишет там что он сделал. В конце недели старший разработчик через diff в системе контроля версий посмотрит, какие запросы добавились и одним махом выполнит их на рабочем сервере. Это же теперь будут делать и сами разработчики при приёме свежей версии репозитория.

В Виллабаджио всё ещё пишут SQL-запросы в коммитах, а в Вилларибо уже замучились бегать по спискам и придумали сделать в проекте папку changes и просто добавлять туда отдельные файлы с SQL-запросами. Да не просто по одному файлику, а по два! У них уже несколько раз был случай, когда нужно было на тестовом сервере откатить изменения на предыдущее состояние, и в связи с этим возвращать базу данных назад вручную. Теперь они решили для любой операции создавать по два симметричных SQL-файла: для применения изменения (up) и для отката его назад (down).

Переименование поля таблицы у них выглядит так:

00015_up.sql:

```
ALTER TABLE tbl_post CHANGE name title text NOT NULL;
```

00015_down.sql

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

name text NOT NULL;

любом сервере нужно всего лишь выполнить нужный запрос:

015_up.sql

015_down.sql

вежую версию базы в репозитории. Теперь любому новому
ходную схему и применить все изменения:

```
mysql -uuser -ppassword base < data/base.sql
mysql -uuser -ppassword base < changes/00001_up.sql
mysql -uuser -ppassword base < changes/00002_up.sql
...
mysql -uuser -ppassword base < changes/00014_up.sql
mysql -uuser -ppassword base < changes/00015_up.sql
```

и база данных у него станет актуальной версии.

А если нечаянно появилась ошибка, то для её поиска кто-то сможет откатить пару последних изменений базы и вернуть состояние кода на вчерашнюю версию, а когда найдёт ошибку применить изменения и вернуть всё назад.

Все изменения базы данных хранятся в файлах вместе с кодом в репозитории. Внутри SQL файлов теперь можно записывать запросы любой сложности. Теперь ошибки при ручном внесении изменений фактически исключены.

Мы называем файлы по порядковому номеру как 00015_up.sql . Следующая миграция будет под номером 00016 . Но если два разработчика будут делать одновременно две задачи и добавят свои новые с номером 00016 , то возникнет конфликт. Поэтому удобнее называть миграции на основе даты и времени по UTC с точностью до секунды, например m130627_124312 для изменения от 27-го июня 2013 года в 12:43:12.

Теперь всё работало как надо, базу можно было откатывать на любое число шагов назад и возвращать вперёд. Стоило лишь запоминать, сколько так называемых «миграций» было выполнено и в каком состоянии база сейчас находится.

Определение текущего состояния БД

Если в любой момент посмотреть на отдельно взятую от кода базу, то довольно сложно оценить, свежая она или нет, какие изменения произведены и какие пропущены. Сейчас этого не видно.

Каждому разработчику нужно каждый раз смотреть на историю добавленных файлов и следить, чтобы

Также я здесь

RSS Twitter git В Facebook

Ваше имя

Ваш Email

Подписаться на статьи

Узнавайте о полезных статьях, не пропускайте видеоуроки, получайте бонусы.

чтобы ни одну не пропустить и чтобы не применить случайно

чтобы состояние любой попавшей к нам базы не глядя в код список применённых миграций в самой базе!

у migration:

```
migration (
    KEY
    tf8;
```

и будем добавлять её идентификатор в данную таблицу. А при тизации этого процесса достаточно добавить по одному

соответствующему запросу в каждый файл:

m130627_124312_up.sql:

```
ALTER TABLE tbl_post CHANGE name title varchar(255) NOT NULL;
INSERT INTO tbl_migration (id) VALUES ('130627_124312');
```

m130627_124312_down.sql:

```
ALTER TABLE tbl_post CHANGE title name varchar(255) NOT NULL;
DELETE FROM tbl_migration WHERE id = '130627_124312';
```

Теперь можно открыть базу и посмотреть последнюю строку в таблице migration.

Автоматизация процесса

Вот тимлид вернулся из отпуска и начал смотреть репозиторий, чтобы узнать, какие файлики изменений добавились. После этого начал их применять, импортируя в консоли одну за одной. А потом пришёл домой, прочёл о [консольных командах](#) и сказал: «Чёрт его побери! Мне же можно написать консольный скрипт, который делал бы всё сам!»

Итак, посмотрим, что тут можно сделать. Попробуем сделать скрипт миграций для PHP проектов.

Система миграций для PHP проектов

Первым делом заменим нативные SQL-файлы на продвинутые PHP-классы. Создадим базовый шаблон с методами `up` и `down`:

```
abstract class Migration
{
    public function up() {

    }

    public function down() {
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

трациями от него.

о может выглядеть так:

ation

```
tbl_post CHANGE name title varchar(255)');
```

```
tbl_post CHANGE title name varchar(255)');
```

Использование классов вместо SQL-файлов даёт преимущество, так как в них мы можем выполнять любые преобразования и организовывать любую логику с помощью PHP.

Проекты могут быть разными: на разных базах и с разными префиксами таблиц.

Всё изменчивое и касающееся различий баз лучше инкапсулировать в что-то более независимое. Можно взять тот же PDO или любой другой объектный инструмент для доступа к базе данных.

```
abstract class Migration
{
    /**
     * @var DB
     */
    protected $db;

    public function __construct(DB $db) {
        $this->db = $db;
    }

    public function up() {

    }

    public function down() {

    }
}
```

```
class m130627_124312 extends Migration
{
    public function up() {
        $this->db->execute('ALTER TABLE `'. $this->db->prefix . 'post` CHANGE name tit

    }

    public function down() {
```

```
$this->db->execute('ALTER TABLE `'. $this->db->prefix . 'post` CHANGE title na
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

лосипедим» свой класс на основе MySQL:

```
{
    private $_id;
    public $prefix;

    public function __construct($host, $username, $password, $database, $prefix='', $na
        $this->connect($host, $username, $password, $database);
        $this->execute('SET NAMES "' . $this->escape($names) . '"');
        $this->prefix = $prefix;
    }

    public function execute($sql) {
        mysql_query($this->handleSql($sql), $this->_id);
        $this->checkErrors();
    }

    public function query($sql) {
        $result = mysql_query($this->handleSql($sql), $this->_id);
        $this->checkErrors();
        return new QueryResult($result);
    }

    public function escape($value) {
        return mysql_real_escape_string($value);
    }




    public function affected() {
        return mysql_affected_rows($this->_id);
    }

    public function insertId() {
        return mysql_insert_id($this->_id);
    }

    private function handleSql($sql) {
        return preg_replace_callback('#\{\{(.+)\}\}#', function($matches) {
            return $this->prefix . $matches[1];
        }, $sql);
    }

    private function checkErrors() {
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
if (/mysql_error($this->_id)) {  
    mysql_error();  
  
    $this->connect($host, $username, $password) {  
        connect($host, $username, $password))  
        Unable to connect to DB');  
        $this->connect($host, $username, $password)  
        Unable to choose database');  
    }  
}
```

получений при ошибках в запросах. Теперь если что-то пойдёт не
бку.

Обратим также внимание на то, что мы добавили метод для автоподстановки префиксов к именам таблиц в поступающих SQL запросов:

```
private function handleSql($sql) {
    return preg_replace_callback('#\\{\\{(.+?)\\}\\}\\#s', function($matches) {
        return $this->prefix . $matches[1];
    }, $sql);
}
```

Это позволит нам вместо ручного указания префикса:

```
$this->db->execute('ALTER TABLE ` ` . $this->db->prefix . 'post` CHANGE name title varchar(255)');

```

переложить эту обязанность на систему и использовать условную запись имён таблиц в фигурных скобках:

```
$this->db->execute('ALTER TABLE {{post}} CHANGE name title varchar(255)');
```

Это декоративный, но всё же приятный момент.

Результат выполнения `mysql_query` обрамляется в объект класса `QueryResult`, который реализует интерфейс итератора:



```
class QueryResult implements Iterator
{
    private $_result;
    private $_key = 0;

    public function __construct($result) {
        $this->_result = $result;
    }

    public function fetch() {
        return mysql_fetch_assoc($this->_result);
    }
}
```


Также я здесь



Ваше имя

Ваш Email

Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```

        }

        public function valid() {
            return $this->_key < $this->count();
        }

        public function rewind() {
            if ($this->count()) {
                mysql_data_seek($this->_result, 0);
            }
            $this->_key = 0;
        }
    }

```

Это позволяет помимо обычного подхода с условным циклом `while`:

```

$items = $this->db->query('SELECT title FROM {{post}}');

while ($item = $items->fetch()) {
    echo $item['title'];
}

```

использовать обход с помощью цикла `foreach`:

```

$items = $this->db->query('SELECT title FROM {{post}}');

foreach ($items as $item) {
    echo $item['title'];
}

```

В любом случае, внутри соответствующих методов этого класса скрывается метод `mysql_fetch_assoc`.

Итак, теперь у нас есть классы `Migration`, `DB`, `MySQL_DB`, `QueryResult` и первая миграция `migrations/m130627_124312.php`:






```

class m130627_124312 extends Migration
{
    public function up() {

```

```
$this->db->execute('ALTER TABLE {{post}} CHANGE name title varchar(255)');
```

Также я здесь

Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
TABLE {{post}} CHANGE title name varchar(255)');
```

дет запускать миграции и вести их историю.

откатывать» произвольное число миграций. То есть иметь
Причём метод `up` по умолчанию (если не передано количество
щихся, а `down` – отменять только одну последнюю. Это
удобным образом.

`create`, который бы на основе текущего времени создавал
пустой класс для новой миграции.

Итак, реализация может быть такой:

```
class MigrationManager
{
    public $table = '{{migration}}';
    public $path = 'migrations';

    /**
     * @var DB
     */
    protected $db;

    public function __construct($db) {
        $this->db = $db;
        $this->checkEnvironment();
    }

    public function up($count = 0) {
        $new = $this->getNewMigrations();
        $new = array_slice($new, 0, $count ? (int)$count : count($new));
        if ($new) {
            foreach ($new as $version) {
                echo '    ' . $version . PHP_EOL;
            }
            if ($this->confirm('Apply the above migrations?')) {
                foreach ($new as $version) {
                    echo 'Apply migration ' . $version . PHP_EOL;
                    if ($this->migrateUp($version) === false) {
                        echo 'Failed!' . PHP_EOL;
                        return 1;
                    }
                }
                echo 'Success!' . PHP_EOL;
            }
        } else {
            echo 'All migrations are ready' . PHP_EOL;
        }
    }
}
```

return 0;

Также я здесь



Ваше имя

Ваш Email

Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
0) {  
    tMigrations();  
    cent, 0, $count ? (int)$count : 1);
```

```
version) {  
    rsion . PHP_EOL;
```

```
event the above migrations?')) {  
    as $version) {  
        migration ' . $version . PHP_EOL;  
        grateDown($version) === false) {  
            led! ' . PHP_EOL;
```

```
echo 'Success!' . PHP_EOL;
```

```
    }  
    } else {  
        echo 'No migrations to revert' . PHP_EOL;  
    }  
    return 0;  
}
```

```
public function create() {  
    $version = gmdate('ymd_His');  
    echo 'Create migration ' . $version . PHP_EOL;  
    $file = $this->createFullFileName($version);  
    $class = $this->createClassName($version);  
    $content =
```

<<<END

<?php

```
class {$class} extends Migration  
{
```

```
    public function up() {  
  
    }
```

```
    public function down() {  
  
    }
```

```
}
```

END;

```
    file_put_contents($file, $content);  
}
```

```
public function help() {  
    echo <<<END
```

Usage:

```
    php migrate.php <action>
```

Actions:

```
    up [<count>]  
    down [<count>]  
    create
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
sage) {
) [yes]: ' ';
));
t, 'y' , 1);

ersion) {
igration($version);
false) {
on);
```

```
private function migrateDown($version) {
    $migration = $this->loadMigration($version);
    if ($migration->down() !== false) {
        $this->writeDown($version);
        return true;
    }
    return false;
}

private function loadMigration($version) {
    require_once($this->createFullFileName($version));
    $class = $this->createClassName($version);
    return new $class($this->db);
}

private function createClassName($version) {
    return 'm' . $version;
}

private function createFullFileName($version) {
    return $this->path . '/' . $this->createFileName($version);
}

private function createFileName($version) {
    return 'm' . $version . '.php';
}






private function getNewMigrations() {
    ...
}

private function getRecentMigrations() {
    ...
}

private function getAppliedMigrations() {
    ...
}

private function getExistingMigrations() {
```

Также я здесь

Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```

sion) {
T INTO `` . $this->table . `` (id) VALUES (" . $this-

ersion) {
E FROM `` . $this->table . `` WHERE id = " . $this->d

ment() {
ath)) {

E TABLE IF NOT EXISTS ' . $this->table . ' (`id` varch

```

В конструкторе данный класс принимает объект доступа к базе данных и вызовом `checkEnvironment` сам создаёт папку и таблицу миграций, если они отсутствуют.

Код вспомогательных методов здесь не приведён для сокращения сути.

Полный комплект файлов: [migrations.zip](#).

В методе `create` создаётся новый файл миграции с пустым классом. В качестве имени задаётся текущая дата и время по UTC. Так что даже если разработчики находятся в разных часовых поясах их миграции будут всегда следовать в правильном алфавитном порядке.

Мы реализовали только минимальный набор методов `up`, `down` и `create`. Также полезно было бы добавить `status` для простого показа новых миграций без применения, `to` для преобразования базы к указанному конкретному состоянию и ещё модифицировать `up` и `down` для работы только с одной указанной миграцией.

Рассмотрим подробнее метод `up`:






```

public function up($count = 0) {
    $new = $this->getNewMigrations();
    $new = array_slice($new, 0, $count ? (int)$count : count($new));
    if ($new) {
        foreach ($new as $version) {
            echo '    ' . $version . PHP_EOL;
        }
        if ($this->confirm('Apply the above migrations?')) {
            foreach ($new as $version) {
                echo 'Apply migration ' . $version . PHP_EOL;
                if ($this->migrateUp($version) === false) {
                    echo 'Failed!' . PHP_EOL;
                    return 1;
                }
            }
        }
    }
}

```

echo 'Success!'; PHP_EOL;

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

ready' . PHP_EOL;

в, которые нужно применить. Внутренним методом неприменённых миграций (посредством сравнения списка базы), отсортированный по дате. Затем с помощью несколько первых значений (если указано число \$count),

ыведется сообщение об их отсутствии. Иначе их список подтверждения действия. Если пользователь ответит утвердительно, то для каждого элемента запустится метод migrateUp :

```
private function migrateUp($version) {  
    $migration = $this->loadMigration($version);  
    if ($migration->up() !== false) {  
        $this->writeUp($version);  
        return true;  
    }  
    return false;  
}
```

Он подключит нужный файл, создаст экземпляр класса и запустит его метод `up`. Если это не вернёт значение `false`, то запустится `writeUp` и добавит запись об успешно применённой версии в таблицу `migration`.

В методах `up` и, что чаще, в `down` миграции можно прервать выполнение очереди посредством `return false`. Но данную возможность следует использовать только если это крайне необходимо.

Метод `down` работает аналогично, но только по умолчанию откатывает только одну последнюю применённую миграцию и удаляет пометку о её версии из базы.

Теперь осталось только написать консольный скрипт-контроллер, который бы принимал пользовательские параметры, конфигурировал все компоненты и запускал нужное действие.

Скрипт `migrate.php`:

```
<?php  
<?php  
  
require_once(dirname(__FILE__) . '/inc/DB.php');  
require_once(dirname(__FILE__) . '/inc/Migration.php');  
require_once(dirname(__FILE__) . '/inc/MigrationManager.php');  
  
$db = new MySQL_DB('localhost', 'root', 'root', 'test', 'tbl_');
```

```
$command = new MigrationManager($db);
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
) . '/migrations';
```

```
rst($argv[1]) : 'help';
```

```
on)) {
```

```
c_array(array($command, $action), $params)) {
```

зу с таблицей `tbl_post`:

```
CREATE TABLE IF NOT EXISTS `tbl_post` (  
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  `name` varchar(255) NOT NULL,  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

зайти в консоли в нашу папку и попробовать «погонять» миграции туда-сюда:

```
php migrate.php create  
php migrate.php up  
php migrate.php down  
php migrate.php create  
php migrate.php up  
php migrate.php down 2
```

Теперь любой разработчик может создать новую миграцию, выполнив:

```
php migrate.php create
```

записать в созданный класс свой функционал и попробовать применить и откатить пару раз у себя. Теперь нужно добавить этот файл в репозиторий проекта и отправить на центральный сервер.

Другие разработчики после скачивания изменений должны запустить у себя:

```
php migrate.php up
```

и все новые изменения произведутся в их базах. Сразу после применения можно продолжать работу.

Таким образом, мы теперь можем написать вполне самостоятельный компонент для поддержки миграций для любой самописной или готовой CMS.

А теперь посмотрим устройство команды `migrate` в Yii, а также рассмотрим некоторые примеры применения и дадим несколько общих советов.

Миграции в Yii Framework

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

анды в Yii и создавали [минимизатор файлов стилей](#). Так вот, в

то иное как встроенная в Yii возможность работать с
й командой [MigrateCommand](#). Рассмотрим её подробнее.

кционала с добавлением времени:

```
ation {...}
```

для миграции в Yii нужно указывать любое текстовое имя, например:

```
class m130627_124312_rename_post_title extends CDbMigration {...}
```

Его нужно передавать при создании миграции:

```
php yiic.php migrate create rename_post title
```

При поиске новых миграций, применениях, откатах и других операциях это имя не учитывается, поэтому при необходимости (при опечатках или не очень подходящем названии) миграции можно переименовывать, сохраняя неизменным номер версии.

Также в Yii имеются несколько не только разобранных нами, но и ещё несколько дополнительных действий. Общий их список таков:

- up – применить все (или заданное количество) свежие миграции
- down – откатить одну (или заданное количество)
- redo – откатить и заново применить одну (или заданное количество)
- to – перевести базу в указанное состояние по идентификатору нужной миграции
- mark – вручную пометить указанную миграцию только что применённой
- history – вывести список применённых миграций
- new – вывести список свежих ещё не применённых
- create – создать новую с указанным именем

Поддержка транзакций

Предположим, что в одной миграции выполняется несколько последовательных запросов переносом данных. На случай, если в течение выполнения миграции что-то пойдёт не так, лучше бы было иметь возможность отменять изменения.

Если ваша база данных поддерживает транзакции (например, таблицы в MySQL вместо MyISAM созданы с движком InnoDB), то для защиты от не полностью завершённой серии запросов достаточно просто переименовать методы `up()` и `down()` в `safeUp()` и `safeDown()` соответственно:

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
n extends CDbMigration
```

он, то увидим, что они оборачиваются в транзакцию

транзакции работают только с данными (изменения данных). Для операций добавления, переименования и удаления TABLE, BEGIN, CREATE INDEX, DROP DATABASE, DROP : транзакции не отменяются. Применение методов `safeUp` с ки бесполезно.

Вспомогательные методы

В миграциях можно использовать свои модели `ActiveRecord` или работать напрямую с базой через DAO. Вместо `Yii::app()->db` эффективнее обращаться к соединению вызвав метод `getDbConnection` самой транзакции, так как в конфигурационном файле `config/console.php` может быть настроено какое-угодно соединение.

Получив текущее соединение, мы можем добавить колонку прямо его методом:

```
public function up()
{
    $this->getDbConnection()->createCommand()->addColumn('{{post}}', 'public, 'tinyint(
```

Но давайте заглянем внутрь класса `CDbMigration`. Там мы найдём одноимённый метод:

```
abstract class CDbMigration extends CComponent
{
    ...
    public function addColumn($table, $column, $type)
    {
        echo " > add column $column $type to table $table ...";
        $time=microtime(true);
        $this->getDbConnection()->createCommand()->addColumn($table, $column, $type);
        echo " done (time: ".sprintf('%.3f', microtime(true)-$time).")\n";
    }
    ...
}
```

Это своеобразный декоратор над методом `CDbCommand::addColumn`, дополненный выводом сообщения о текущей операции и времени на экран. Для удобства и информирования целесообразно применять именно его:

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
'public, 'tinyint(1) NOT NULL');
```

угих методов. Их удобно использовать вместо простых SQL-

з SQL-файлов является возможность программировать в них

миграции, добавляющей поддержку нескольких категорий для
(гим) вместо одной (один-ко многим) с сохранением всех

```
class m130401_103942_added_snop_multicategory extends EDbMigration
{
    public function up()
    {
        // Создаём таблицу связи многие-ко-многим
        $this->createTable('{{shop_product_category}}', array(
            'id' => 'pk',
            'product_id' => 'int(11) NOT NULL',
            'category_id' => 'int(11) NOT NULL',
        ), 'ENGINE=InnoDB DEFAULT CHARSET=utf8');

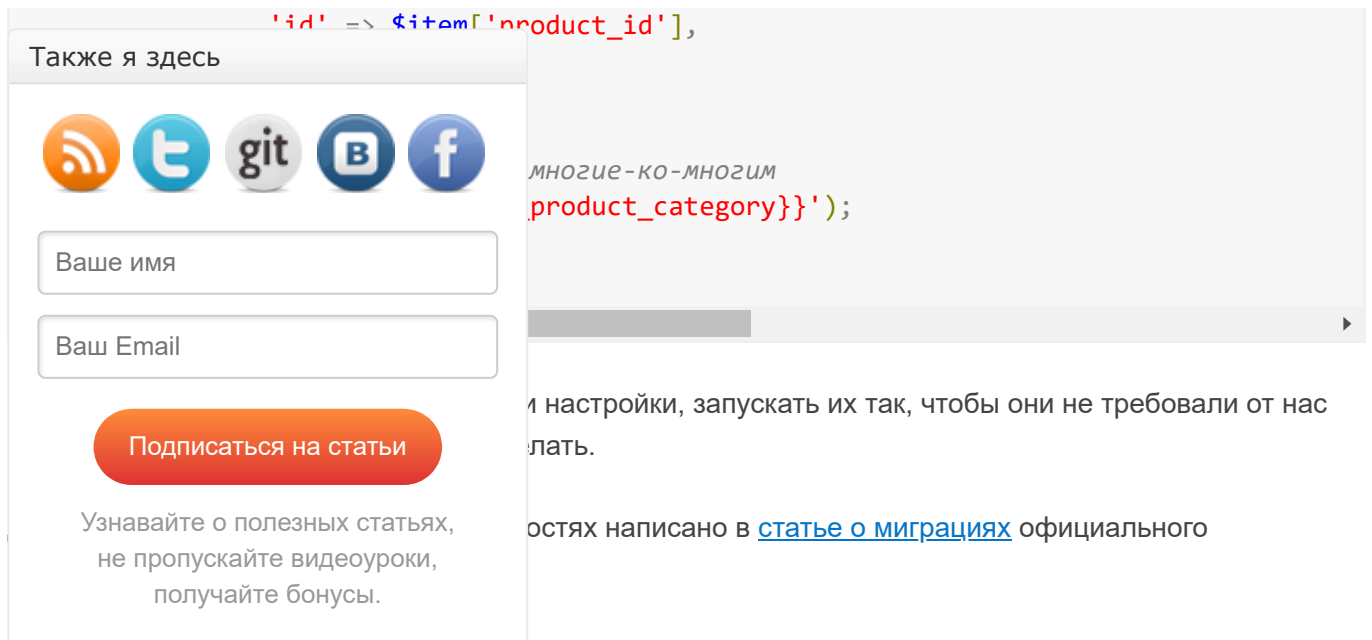
        $this->createIndex('product_id', '{{shop_product_category}}', 'product_id');
        $this->createIndex('category_id', '{{shop_product_category}}', 'category_id');

        // Заносим туда уже имеющиеся связи:
        $products = $this->getDbConnection()->createCommand('SELECT id, category_id FROM');
        foreach ($products as $item)
        {
            $this->insert('{{shop_product_category}}', array(
                'product_id' => $item['id'],
                'category_id' => $item['category_id'],
            ));
        }

        // Удаляем теперь лишнюю колонку старой связи
        $this->dropColumn('{{shop_product}}', 'category_id');
    }

    public function down()
    {
        // Возвращаем колонку связи один-ко-многим
        $this->addColumn('{{shop_product}}', 'category_id', 'int(11) NOT NULL');
        $this->createIndex('category_id', '{{shop_product}}', 'category_id');

        // Возвращаем старые связи
        $relations = $this->getDbConnection()->createCommand('SELECT product_id, category_id FROM');
        foreach ($relations as $item)
        {
            $this->getDbConnection()->createCommand('UPDATE {{shop_product}} SET category_id = ' . $item['category_id'] . ' WHERE product_id = ' . $item['product_id'] . ');
```



Рекомендации к использованию

Теперь мы знаем, что миграции служат не только для обмена изменениями, но и для двустороннего перевода базы данных в состояние на любую дату. То есть если нам надо откатиться на вчерашнюю версию нашего сайта, то мы отменяем сегодняшние миграции и для возврата вчерашнего состояния файлов переключаемся на любую вчерашнюю резервную точку системы контроля версий. При этом должна быть возможность беспрепятственно передвигаться по истории в любую сторону на любое число шагов.

В связи с этим желательно соблюдать главные рекомендации:

- **все изменения структуры базы производить в миграциях;**
- **сохранять только изменения схемы, а не данных;**
- **не исправлять уже добавленные миграции.**

По первому пункту: Можете, конечно, во время разработки потренироваться вручную в PhpMyAdmin, но всё равно потом верните всё «как было» и запишите все необходимые изменения в программный код.

По второму: На боевом сервере свои данные, на тестовом вторые, на рабочем – третьи. На большом сайте новые комментарии добавляются каждую секунду. Не обращайте внимания на пользовательские данные.

Если в базе данных хранятся системные настройки модулей сайта, то можете включить их изменение в миграцию. Остальные же данные (пользователи, комментарии или записи блога) не фиксируйте.






И по третьему: Создавайте всегда новую миграцию для исправления недочётов, а не исправляйте старую. За исключением случаев, когда старая содержит очень вредную ошибку. Миграции всегда должны уметь выполняться последовательно.

Пример усложнения жизни

В понедельник вы добавили миграцию переименования таблицы `tbl_posts` в `tbl_post`, во вторник передумали и переименовали `tbl_post` в `tbl_entry`. Сейчас среда. Если при этом для двух переименований созданы две миграции, то можно легко переключаться на среду/вторник/понедельник/вторник/среду всего лишь запуская по одному разу `down` и `up`.

А теперь представим, что в поле ввода мы

Также я здесь



[Подписаться на статьи](#)

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

Как мы добавили первую миграцию `tbl_posts→tbl_post`, а во второй сделали эту же в `tbl_posts→tbl_entry` и применили заново. То есть создали экземпляра одной миграции с разным функционалом. Теперь чтобы попасть из среды во вторник нужно откатить среду на понедельник) → переключить файлы на состояние среды, но уже с кодом вторника. То есть надо как-то все сделать так чтобы попасть только двумя шагами через понедельник. Такую задачу нужно запомнить.

Через месяц при разгребании завалов он понял, что там осталась старая таблица оказалась не нужна. Но вместо создания новой он добавил строку и закомментировал строку в старой. Никто об этом не узнал.

ищет себе репозиторий, применяет миграции и получает

свежую рабочую базу. Заказчик заново задумался внедрить систему рейтингов. Программист пишет новую миграцию:

```
public function up() {
    $this->addColumn('{{post}}', 'rating', 'int(3) NOT NULL');
}

public function down() {
    $this->dropColumn('{{post}}', 'rating');
}
```

тестирует её у себя и всё происходит нормально. На следующий день эту миграцию применяют на рабочем сервере и... она не применяется со словами «Колонка 'rating' уже существует». А наш разработчик даже не знал, что в его личную базу это поле не добавилось, так как в коде одной из миграций оно было заботливо закомментировано полгода назад...

Это просто неприятный безобидный пример. Но что было бы, если б кто-то забыл убрать или лишний раз убрал что-то более важное? Была бы остановка сервера на час с аварийным восстановлением из резервной копии.

Пользовательские настройки

Любую команду в Yii можно настроить в секции `commandMap` конфигурационного файла `console.php`. Достаточно открыть класс `MigrateCommand` и посмотреть, какие у него есть публичные свойства и сеттеры. Присвоить им любые значения можно как и другим компонентам:

```
return array(  
    ...  
    'components' => array(  
        'db' => array(...),  
    ),  
  
    'commandMap' => array(  
        'migrate' => array(  
            'migrationPath' => 'application.migrations',  
            'migrationTable' => '{{migration}}',  
        ),  
    ),  
);
```

```
'connectionID'=>'db'
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

d ищет и применяет миграции из папки
можем её конфигурировать как любой компонент, то мы можем
одменить используемый класс команды.

ботанный класс [EMigrateCommand](#), который позволяет
и каждого модуля любого проекта на Yii. В минимальном
модули и заменить им класс команды:

```
...  
  
'modules' => array(  
    ...  
)  
  
'components' => array(  
    'db' => array(...),  
) ,  
  
'commandMap' => array(  
    'migrate' => array(  
        'class' => 'ext.migrate-command.EMigrateCommand',  
        'migrationTable' => '{{migration}}',  
        'applicationModuleName' => 'core',  
        'migrationSubPath' => 'migrations',  
        'connectionID' => 'db',  
    ),  
) ,  
) ;
```

Теперь вместо:

```
yiic migrate create create_post_table
```

для создания миграции в папке `migrations` модуля блога следует вызывать:

```
yiic migrate create blog create_post_table
```

То есть в командах просто указывать имя модуля. Это изменение касается и некоторых других команд.

При этом можно и дальше пользоваться командами:

```
yiic migrate up  
yiic migrate down
```

комментарие. Не забудьте также создать папки для миграций в модулях.

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

ния используется кэширование. Например, в настройках
ы базы 3600 секунд опцией `schemaCachingDuration` или как-
менения желательно очищать кэш. Для этого можно добавить

```
leCommand
```

```
{  
);  
d' . PHP_EOL;
```

и запускать:

```
yii cache clear
```

после применения миграций.

Но рассмотрим другую ситуацию. В один прекрасный момент мы применяем миграцию с добавлением новой колонки `alias` для перевода блога на ЧПУ. В модели мы добавили заполнение этого поля транслитом наименования в методе `beforeSave`:

```
class Post extends CActiveRecord  
{  
    ...  
  
    protected function beforeSave()  
    {  
        if (parent::beforeSave) {  
            if (!$this->alias) {  
                $this->alias = TextHelper::translit($this->title);  
            }  
            return true;  
        }  
        return false;  
    }  
}
```

Здесь могли бы быть другие преобразования, но это сейчас не важно.

Итак, теперь мы должны добавить саму миграцию и заполнить новое поле `alias`. Для этого проще всего пересохранить все модели:

```
public function up()  
{  
    $this->addColumn('post', 'alias', 'varchar(255) NOT NULL');
```

```
foreach (Post::model()->findAll() as $post) {
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

нении этой миграции после вызова другой операции с моделью (то ещё в приложении) или при включенном кэшировании поле `alias` не найдено в базе данных. Действительно, или таблицу с того момента, как было произведено или. Да и даже без кэширования схема после чтения из базы / поле `$post->alias` при втором обращении к той же модели

можно не только отключить кэширование для консольного `ration` и используя заглушку `CDummyCache`), но и в теле самой и очистку кэша (если он включен) и перезагрузить схему:

```
public function up()
{
    $this->addColumn('post', 'alias', 'varchar(255) NOT NULL');

    Yii::app()->cache->flush();
    Yii::app()->db->schema->getTables();
    Yii::app()->db->schema->refresh();

    foreach (Post::model()->findAll() as $post) {
        $post->save();
    }
}
```

Теперь какое бы кэширование ни было включено, проблем с изменёнными полями не возникнет. Можно коммитить миграцию с новой моделью `Post` и отправлять на центральный сервер.

Другие инструменты и возможности

Вот так мы ознакомились с назначением миграций базы данных, вместе написали свою систему и перешли к управлению изменениями базы в Yii Framework. Но мы не рассмотрели другие готовые инструменты.






Тот же компонент Doctrine Migration, используемый обычно как компонент Doctrine в Symfony Framework, немного отличается по своим доступным операциям для управления миграциями. Все модели в Doctrine (а именно название таблиц, имена и типы полей) дополнительно описываются в конфигурационных файлах, что дополнительно даёт нам возможность автоматически генерировать запросы в миграциях. Например, мы добавили новое поле в нашу модель (или сделали индексным существующее) и вызываем команду `diff`:

```
php app/console doctrine:migrations:diff
```

Данная консольная команда сравнит схему из конфигурационного файла каждой модели с текущими таблицами в самой базе и сгенерирует миграцию, в которой сама запишет запросы по изменению/добавлению/удалению поля или индекса. Это полезная возможность, но полагаться полностью на неё

не стоит, так как вместо переименования одной колонки может сгенерироваться два запроса: на то приведёт к потере значений.

Также я здесь



Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

инструменты, например `rake` в Ruby on Rails. Каждый имеет свой фреймворк, но общие принципы действия с ними остаются

вопросы или предложения, то напишите их в комментариях.

Бонусы и мастер-классы:

Поделиться



Другие статьи



[Элементы SEO для Yii Framework](#)

При построении любого сайта в какой-то момент разработчик сталкивается с требованиями поисковой оптимизации. Традиционно она включает в себя построение правильной адресной структуры, исключение из индексации служебных страниц и неинтересных для поискового робота фрагментов, добавление метаинформации для записей. Более расширенный вариант подразумевает специфическое распределение ссылочного веса и борьбу с дубликатами адресов.



[Предпраздничные новости декабря](#)

Вот и прошёл недавно первый год второй жизни блога. В то время, пока вы делитесь эмоциями в комментариях ко вчерашней статье о миграциях, пора мне выйти из тени и поделиться парой новостей. Итак, если у вас есть какие-нибудь предложения к развитию блога в будущем году, если у вас уже праздничное настроение, то прошу пожаловать под кат.



[Консольный минимизатор скриптов и стилей в Yii](#)

На прошлом уроке мы познакомились с консольным режимом в PHP и с консольными командами в Yii. Теперь пришла пора собрать вместе наши знания и перейти к практике. При разработке любых проектов удобно разделять CSS и JavaScript на отдельные файлы, но их обилие в секции HEAD заметно уменьшает скорость загрузки веб-страницы. Итак, поехали!



[Консольные команды в Yii](#)

Для решения специфических задач часто используются готовые консольные команды. Но намного интереснее не только использовать чужие, но и уметь создавать свои. Это поможет легко автоматизировать любую рутинную работу, на которую обычно тратится довольно много времени. Многие фреймворки имеют встроенные инструменты для написания не только самих веб-приложений, но и для создания инфраструктуры пользовательских консольных команд.






Комментарии

[Иван](#)

Спасибо, очень хорошая статья получилась.

11 декабря 2013 17:23 0

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

Если бы вы писали эти статьи сразу с англоязычной версией, то посещалка бы заметно возросла. У нас то намного ограниченной ареал читателей данной темы.


[Ответить](#)

...таки при миграциях нет смысла
...ия, достаточно только сбросить кэш
...ситуации и задачи.

...еобходимости.

...ьясните.

...ii, могли бы ОЧЕНЬ многому у вас поучиться.




Redee

18 декабря 2013 20:53 0

А счетчик внизу сайта в районе 500 посещений что уж так мало для личного блога ???
Думаю вполне нормальный показатель.

[Ответить](#)




Фарид

10 февраля 2014 12:34 0

Отличная статья, спасибо, подписался!

Опечаточка: "Актуальную базу данных от может попросить и кого-то"

[Ответить](#)




Дмитрий Елисеев

10 февраля 2014 15:31 0

Спасибо! Исправил.

[Ответить](#)




Дмитрий

15 мая 2014 13:19 0

Дмитрий подскажите где должна располагаться папка с migrate.php и зависимыми файлами ?
после выполнения: php migrate.php create ничего не происходит...я пока создал папку mirateBD в
которой лежат все файлы и положил в protected

[Ответить](#)



Дмитрий Елисеев

15 мая 2014 16:18 0

А зачем Вы используете этот migrate.php в Yii? Там уже всё встроенное есть. Создайте папку
protected/migrations и в protected запустите:

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

create my_first_migration

2 июля 2014 00:42 0 ♥

дписался.
e my_first_migration
st_migration
в папку protected.

2 июля 2014 15:34 0 ♥



Можно, но это только в Windows сразу заработает. В Unix сначала надо файл исполняемым сделать.

[Ответить](#)



Дмитрий

20 августа 2014 19:34 0 ♥

Дмитрий, можно рассказать как применять миграции на рабочем сервере?
Вот мы сделали на локальном сервере некоторые изменения, создали нужные миграции, зашли в консоль и применили их. Теперь изменения нужно сделать на рабочей машине.

[Ответить](#)



Дмитрий Елисеев

26 августа 2014 16:18 0 ♥

Также в консоли. Через SSH.

[Ответить](#)



Дмитрий Бухтяк

26 августа 2014 14:29 0 ♥

Я подписался и зарегистрировался, однако ссылку на полный архив с файлами так и не увидел.
В тексте продолжает висеть optin форма для подписки.

[Ответить](#)



Дмитрий Елисеев

26 августа 2014 15:11 0 ♥

Ссылка должна была прийти в приветственном письме.

[Ответить](#)



Дмитрий Бухтяк

26 августа 2014 15:23 0 ♥

Пришла лишь ссылка на подтверждение подписки. В тексте этого письма написано: "Копия этого текста вместе с бонусами уже отправлена Вам в приветственном письме и дойдёт с минуты на минуту." - однако ничего не пришло (даже в спам)

[Ответить](#)

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

[Ответить](#)

26 августа 2014 16:14 1 ♥

гическое письмо пока не дошло. Отправил скрипт Вам на почту

26 августа 2014 14:52 0 ♥

дить свой класс DB из устаревший либы mysql, почему не на базе

26 августа 2014 15:13 0 ♥



Тимофей

18 октября 2014 04:09 0 ♥

Дмитрий, планируются ли статьи о модульном и функциональном тестировании, TDD, selenium ?

[Ответить](#)



Дмитрий Елисеев

18 октября 2014 14:14 0 ♥

Может быть. Тема веьма спорная и интересная.

[Ответить](#)



Виталий Гончаров

27 февраля 2015 16:02 0 ♥

Дмитрий, насколько я понял, в миграциях нельзя использовать модели (например для заполнения только что созданных полей таблиц). Разработчики считают, что "... You should never use model in migrations since model code can be changed slightly while one should be able to run migrations from the very first (where model was different) to the current.". А иногда хочется... а как бы вы рекомендовали обновлять данные, если для этого нужно задействовать логику моделей? Спасибо.

[Ответить](#)



Дмитрий Елисеев

27 февраля 2015 21:56 0 ♥

Можно повторить нужную логику модели в миграции. Какая именно логика Вас интересует?

[Ответить](#)



Виталий Гончаров

28 февраля 2015 11:09 0 ♥

Ну допустим, есть модели "статья" и "комментарий" я хочу хранить некий рейтинг статьи, который является суммой лайков к статье + комментариям. Для этого в модель "статья" добавляю поле "рейтинг" и в метод "afterSave" статьи и комментария прописываю логику расчета рейтинга соотв. статьи.

Соотв. в миграции хочу пройти по всем статьям/комментариям и их пересохранить, чтобы рассчитались рейтинги уже существующих статей.

Повторять эту логику в теле миграции - не совсем удобно, она же в принципе может быть

Также я здесь

можно сделать "служебный" экшн где-нибудь, но это уже надо
рациями воспользоваться, и тоже не до конца подходит



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

28 февраля 2015 12:21 0 ♥

предположим, переделаете систему рейтингов и эта миграция
как модель поменялась. Новый программист клонирует проект
его поднять из миграций без необходимости откатываться на тот
ботало. Предостережение по неиспользованию моделей как раз
и введено.

тываете всё сами и пишете миграции только для себя, то можете
если делаете чужой проект, то уже пересчитывайте в самой
each.



slo_nik

12 мая 2015 23:55 0 ♥

Доброй ночи. Подскажите, а можно при выполнении миграции вставить сразу несколько строк в
таблицу? Таблица содержит название стран, структура "id" и "name". По одной строке без
проблем получается, а вот сразу несколько никак не пойму как делать. В документации есть
batchInsert(), но не пойму, как использовать.

[Ответить](#)



slo_nik

13 мая 2015 00:07 0 ♥

Поспешил с вопросом))) Решил задачу сам)

[Ответить](#)



nemesis

23 июля 2015 20:17 0 ♥

А как скачать? Я подписался, ссылка не появилась.

[Ответить](#)



Дмитрий Елисеев

24 июля 2015 10:39 0 ♥

В приветственном письме рассылки внизу будут бонусы для записей.

[Ответить](#)



nemesis

27 июля 2015 14:04 0 ♥

Спасибо за скрипт, очень сэкономил время...

[Ответить](#)



EVG

31 июля 2015 16:02 0 ♥

Дмитрий, не могу разобраться с миграциями в yii2, пытаюсь создать миграцию (php yii migrate
create create_user_table), получаю ошибку: Fatal error: Uncaught exception
'yii\base\InvalidConfigException' with message 'The "basePath" configuration for the Application is
required.' и еще куча сопутствующей выдачи. Базовый путь задан в конфигурации по умолчанию

('basePath' => dirname(__DIR__)), поэтому не понимаю что ему не нравится. Есть мысли в какую

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

31 июля 2015 17:42 0 ♥

e create_user_table

/console.php?

31 июля 2015 18:05 0 ♥



А вот и не указан:) Таки помогло. В вебе проверил, а в консольке проверить не догадался. Каюсь.
А пропал он оттуда по глупости, в посте "Установка и настройка приложения" в блоке по console.php у вас этот параметр выпилен, ну я маханул и забыл.

[Ответить](#)



Сергей

7 августа 2015 10:30 1 ♥

А поправить этот косяк в своей статье, чтобы те, кто копируют оттуда - на него не нарывались?
Удивительное, б****, пренебрежение - в комментариях люди пишут: здесь забыл, здесь выпилил кусок и без него не работает - а сами статьи оставляем без изменений, а чо, пусть пое****я.
Учитель х****, блин.

[Ответить](#)



Дмитрий Елисеев

8 августа 2015 08:19 0 ♥

Это не косяк.

[Ответить](#)



В Сергей Новиков

13 сентября 2015 21:42 0 ♥

Добрый день!
Дмитрий, а можно как-то вывести Raw SQL для миграций?

Работаю над проектом который хостится на Виртуальном хостинге, и из доступа только FTP и PhpMyAdmin. Хотелось бы у себя на локальной машине получать Sql для написанных миграций, и применять его через PhpMyAdmin. Пока не придумал ничего лучше чем логировать консольное приложение и смотреть логи sql запросов.

[Ответить](#)



Дмитрий Елисеев

16 сентября 2015 14:25 0 ♥

Да, по логам можно. А так можно сделать скрипт web/deploy.php:

```
<?php
```

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

```
if ($? (Test-Path $SERVER_IP_REMOTE_ADDR) == 'свой IP') {  
    php yii migrate/up --interactive=0 > log.txt');  
    set_contents(__DIR__ . '/../log.txt');
```

использовать любой WebShell (или даже [модуль](#) для Yii2).

29 июля 2016 18:13

0 ♥

сть ли для второго юи инструменты, что бы по готовой базе
орк, переписывать вручную десятки таблиц не хочется.



Дмитрий Елисеев

29 июля 2016 18:49

0 ♥

Вроде были. Погуглите по фразе "yii2 migration generator".

[Ответить](#)



Иван

14 сентября 2017 12:15

0 ♥

Здравствуйте, Дмитрий!

А можно ли как нибудь применить одну миграцию пропустив другую?

Я начал писать большую миграцию и отложил её, появилась необходимость быстро создать
другую миграцию и применить её, но недописанная еще миграция тоже рвется в исполнение :(
как её проигнорировать?

[Ответить](#)



Дмитрий Елисеев

25 сентября 2017 12:45

0 ♥

Переложить пока в другую папку, в ветку или в stash в git.

[Ответить](#)

[Оставить комментарий](#)

[Войти](#) | [Завести аккаунт](#) | Войти через

Ваше имя *

Ваш Email * (никто не увидит)

Ваш сайт

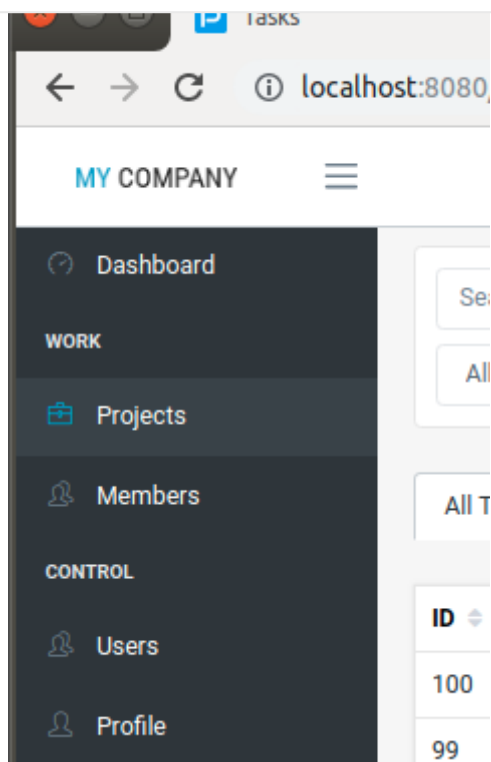
Комментарий *

Также я здесь



Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.



Разделы блога

[Программирование](#)

[Администрирование](#)

[UX-дизайн](#)

[Работа и бизнес](#)

[Саморазвитие](#)

[События и акции](#)

Метки

Также я здесь



Ваше имя

Профиль

Подписаться на статьи

Узнавайте о полезных статьях,
не пропускайте видеоуроки,
получайте бонусы.

[D](#) [Git](#) [JavaScript](#) [Laravel](#) [MySQL](#) [PHP](#) [PSR](#) [RabbitMQ](#)
[DD](#) [Yii](#) [Yii2](#) [Архитектура](#) [Вебинар](#) [Видео](#)
[и](#) [Ликбез](#) [Рефакторинг](#) [Саморазвитие](#) [События](#)
[Тестирование](#)

[Регистрация](#) | [Забыли?](#)

[ElisDN](#)

[Карта сайта](#)

[Блог](#)

[Продукты](#)

[Поблагодарить](#)

[Портфолио](#)

[Контакты](#)

© ИП Елисеев Дмитрий Николаевич, 2009-2019

ОГРН: 309574636200019; ИНН: 570600870325

[Политика конфиденциальности](#)

Email: mail@elisdn.ru

Права на все материалы, опубликованные на сайте, принадлежат автору.

Незаконное копирование материалов преследуется по закону.

Использование материалов возможно лишь при наличии
активной ссылки на источник. [Использование материалов](#)

927
452
305

