Автоматизация временных меток

Например, у нас есть простое приложение. Как и в любом блоге, в нем есть посты, комментарии и так далее. Мы хотели бы, чтобы заполнить время во время создания/обновления событий для сообщений. Предположим, что наша модель записи называется моделью Blogpost.

Подготовка

1. Создайте новое приложение с помощью composer, как описано в официальном руководстве <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.   
   (по русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation> )
2. Настройте подключение к базе данных и создайте таблицу с именем blog\_post следующим образом:

DROP TABLE IF EXISTS 'blog\_post';

CREATE TABLE IF NOT EXISTS 'blog\_post' (

'id' INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

'title' VARCHAR(255) NOT NULL,

'text' TEXT NOT NULL,

'created\_date' INTEGER,

' modified\_date'INTEGER,

PRIMARY KEY ('id')

);

1. Используйте Gii для создания модели для таблицы blog\_post.

Как это сделать...

1. Добавьте следующий метод в models/BlogPost.php:

/\*\*

* @return array
* \*/

public function behaviors()

{

return [

'timestamp'=> [

'class' => 'yii\behaviors\TimestampBehavior', 'createdAtAttribute' => 'creation\_date', 'updatedAtAttribute' => 'modified\_date'

]

];

}

1. Создание controllers/TestController.php следующим образом:

<?php

namespace app\controllers;

use app\models\BlogPost;

use yii\helpers\Html;

use yii\helpers\VarDumper;

use yii\web\Controller;

/\*\*

\* Class TestController.

\*/

class TestController extends Controller

{

public function actionIndex()

{

$blogPost = new BlogPost();

$blogPost->title = 'Gotcha!';

$blogPost->text = 'We need some laughter to ease the tension of holiday shopping.';

$blogPost->save();

return $this->renderContent(Html::tag('pre',

VarDumper::dumpAsString($blogPost->attributes)

));

}

}

3. Вот и все. Теперь запустите test/index. Вы должны получить следующий результат:

Как это работает...

По умолчанию поведение Метки времени заполняет created\_at (метка времени, указывающая на время создания модели) и updated\_at (время обновления модели). Это стандартная практика, чтобы назвать эти поля, но если мы хотели бы внести изменения, мы можем указать поля, которые будут обновлены, и события модели.

Есть еще.

Например, наши поля называются creation\_date и modified\_date.

Давайте настроим нашу модель поведения в соответствии с этими полями. Кроме того, мы должны добавить код нашего поведения в нашу почтовую модель:

<?php

namespace app\models;

use Yii;

use yii\db\BaseActiveRecord;

class Post extends \yii\db\ActiveRecord {

// . .

public function behaviors()

{

return [

[

'class' => 'yii\behaviors\TimestampBehavior',

'attributes' => [

BaseActiveRecord::EVENT\_BEFORE\_INSERT => 'creation\_date', BaseActiveRecord::EVENT\_BEFORE\_UPDATE => 'modified\_date',

]

]

];

}

// ..

}

В этом примере мы указали на атрибуты creation\_date и modified\_date перед созданием и обновлением нашей модели соответствующим образом с помощью специальных событий ActiveRecord: EVENT\_BEFORE\_INSERT и EVENT\_BEFORE\_UPDATE.

Кроме того...

Можно сохранить метку времени для пользовательских сценариев. Допустим, вы хотите обновить поле last\_login, например, для определенного действия контроллера. В этом случае можно вызвать обновление метки времени для конкретного атрибута с помощью следующих:

$model->touch('last\_login');

Имейте в виду, что touch() нельзя использовать для новых моделей. В этом случае Вы получите InvalidCallException:

$model = new Post();

$model->touch('creation\_date ' );

Метод touch() вызывает сохранение модели внутри себя, поэтому вам не нужно сохранять модель после ее вызова.

Смотрите так же

Для получения дополнительной информации обратитесь к  
<http://www.yiiframework.com/doc-2.0/guide-concept-behaviors.html#using-timestamp-behavior>

По русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/concept-behaviors#using-timestamp-behavior>