Работа с событиями

События Yii обеспечивают простую реализацию, которая позволяет прослушивать и подписываться на различные события, происходящие в вашем веб-приложении. Например, вы можете отправить уведомление о новой статье подписчикам при каждой публикации нового материала.

Подготовка

1. Создайте новое приложение с помощью composer, как описано в официальном руководстве по <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>. (от переводчика по русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation> )
2. Выполните следующий код SQL на сервере, чтобы создать таблицу статей:

CREATE TABLE 'article' (

'id' int(11) NOT NULL AUTO\_INCREMENT,

'name' varchar(255) DEFAULT NULL,

'description' text,

PRIMARY KEY ('id')

) ENGINE=InnoDB AUTO\_INCREMENT=29 DEFAULT CHARSET=utf8;

1. Создайте модель Article с помощью Gii.
2. Запустите ваш webserver командой ./yii serve.

Как это работает

1. Добавить тест действия в \controllers\SiteController:

public function actionTest()

{

$article = new Article();

$article->name = ‘День Святого Валентина приближается? Вот дерьмо! Я забыл снова завести девушку!';

$article->description = ‘Бендер злится на Фрая за то, что он встречается с роботом. Держись подальше от наших женщин. У тебя металлическая лихорадка, парень. Лихорадка металла';

// $event - объект класса yii\base\Event или дочернего класса

$article->on(ActiveRecord::EVENT\_AFTER\_INSERT, function($event) {

$followers = ['john2@teleworm.us', 'shivawhite@cuvox.de', 'kate@dayrep.com'

];

foreach($followers as $follower) {

Yii::$app->mailer->compose()

->setFrom('techblog@teleworm.us')

->setTo($follower)

->setSubject($event->sender ->name)

->setTextBody($event->sender->description)

->send();

}

echo 'Emails has been sent';

});

if (!$article->save()) {

echo VarDumper::dumpAsString($article->getErrors());

};

}

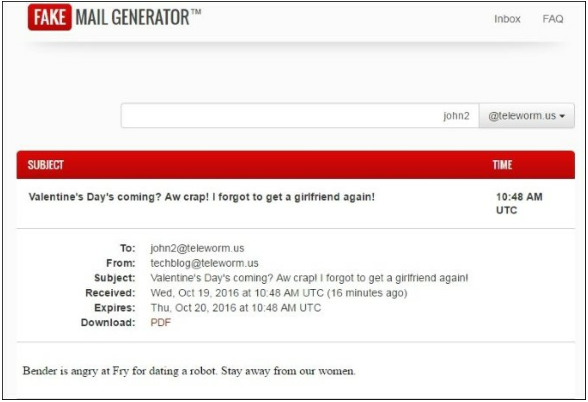
1. Обновите config/web.php, компонент mailer использует следующий код.

'mailer' => [

'class' => 'yii\swiftmailer\Mailer', 'useFileTransport' => false,

],

1. Запустите это URL в вашем браузере: http: //localhost: 8080/index . php?r=site/test.
2. Проверьте <http://www.fakemailgenerator.com/inbox/teleworm.us/john2/>.



Как это работает…

Мы создали модель статьи и добавили обработчик для события ActiveRecord: : EVENT\_AFTER\_INSERT в нашу модель статьи. Это означает, что каждый раз, когда мы сохраняем новую статью, событие инициируется, и наш прикрепленный обработчик будет вызван.

В реальном мире, мы хотели бы уведомить наших подписчиков блога каждый раз, когда мы публикуем новую статью. В реальном приложении у нас будет Таблица подписчиков или пользователей и с различными разделами блога, а не только один блог. В этом примере после сохранения нашей модели мы уведомляем наших последователей john2@teleworm. us, shivawhite@cuvox. de, and kate@dayrep. com. На последнем шаге мы просто доказываем, что пользователи получили наши уведомления, особенно john2. Вы можете создать собственное мероприятие с любым именем. В этом примере мы используем встроенное событие ActiveRecord:: EVENT\_AFTER\_INSERT, которое вызывается после каждой вставки в базу данных.

Например, мы можем создать ваше собственное событие. Просто добавьте новое событие TestNew со следующим кодом:

public function actionTestNew()

{

$article = new Article();

$article->name = 'Valentine\'s Day\'s coming? Aw crap! I forgot to get a girlfriend again!';

$article->description = 'Bender is angry at Fry for dating a robot. Stay away from our women.

You've got metal fever, boy. Metal fever';

// $event is an object of yii\base\Event or a child class

$article->on(Article::EVENT\_OUR\_CUSTOM\_EVENT, function($event) {

$followers = ['john2@teleworm.us',

'shivawhite@cuvox.de',

'kate@dayrep.com' ];

foreach($followers as $follower) {

Yii::$app->mailer->compose()

->setFrom('techblog@teleworm.us')

->setTo($follower)

->setSubject($event->sender->name)

->setTextBody($event->sender->description)

->send();

}

echo 'Emails have been sent';

});

if ($article->save()) {

$article->trigger(Article::EVENT\_OUR\_CUSTOM\_EVENT);

}

}

Также добавьте константу event\_our\_custom\_event в models/Article :

class Article extends \yii\db\ActiveRecord {

CONST EVENT\_OUR\_CUSTOM\_EVENT = 'eventOurCustomEvent';

}

Запустите http: //localhost:8080/index.php?r=site/test-new.

Вы должны увидеть тот же результат, и все уведомления последователям будут отправлены снова. Главное отличие в том, что мы использовали наше пользовательское имя события.

После сохранения мы запустили наше событие. События могут быть вызваны вызовом триггером yii\base\Component::trigger(). Метод требует имя события и при необходимости объект события, описывающий параметры, которые должны быть переданы обработчикам событий.

Смотрите также

Для получения дополнительной информации о событиях обратитесь к  [http://www.yiiframework.com/doc-2.0/guide-concept-](%20%20http://www.yiiframework.com/doc-2.0/guide-concept-) events.html

( по русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/concept-events>)