Setting up a slug automatically

On the web, slug is a short text used in a URL to identify and describe a resource. A slug is the part of a URL which identifies a page using human-readable keywords. Sluggable behavior is the Yii2 model behavior that allows us to generate unique slugs.

In this section, we will be guiding you through modifying Yii’s default view URL routes for model objects to be more user-friendly and search engine-friendly. Yii provides built-in support for this via its sluggable behaviors.

Getting ready

1. Create a new application using the Composer package manager, as described in the official guide at

<http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.

1. Set up the database connection and create a table named blog\_post, as follows:

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,

'slug' VARCHAR(255) NOT NULL,

'text' TEXT NOT NULL,

PRIMARY KEY ('id')

);

1. Use Gii to create a model for the post table.

How to do it...

1. Add the following behaviors method to models/BlogPost. php:

<?php

namespace app\models; use Yii;

use yii\db\BaseActiveRecord;

class BlogPost extends \yii\db\ActiveRecord {

// ..

public function behaviors()

{

return [

[

'class' => 'yii\behaviors\SluggableBehavior', 'attribute' => 'title',

'slugAttribute' => 'slug',

'immutable'=> false,

'ensureUnique' => true

]

];

}

// ..

}

1. Create controllers/TestController.php as follows:

<?php

namespace app\controllers;

use app\models\BlogPost;

use Yii;

use yii\helpers\Html;

use yii\helpers\VarDumper;

use yii\web\Controller;

/\*\*

* Class TestController
* @package app\controllers \*/

class TestController extends Controller {

public function actionIndex()

{

$blogPostA = new BlogPost();

$blogPostA->title = 'Super Quote title 1';

$blogPostA->text = 'The price of success is hard work, dedication to the job at hand';

$blogPostA->save();

$blogPostB = new BlogPost();

$blogPostB->title = 'Super Quote title 2';

$blogPostB->text = 'Happiness lies in the joy of achievement...'; $blogPostB->save();

return $this->renderContent(

'<pre>' .

VarDumper::dumpAsString(

$blogPostA->attributes

).

VarDumper::dumpAsString(

$blogPostB->attributes

) .

'</pre>'

);

}

}

1. The result will be as follows:



How it works...

* Yii offers some nice enhancements to sluggableBehavior for useful scenarios.
* For example, once a search engine records a slug, you probably don’t want the page URL to change.
* The immutable attribute tells Yii to keep the slug the same after it’s first created—even if the title will be updated.
* If users enter messages that overlap in content, the ensureunique property will automatically append a unique suffix to duplicates. This makes certain that each message has a unique URL, even if the message is identical.
* If you go ahead and create another post with the exact same title, you’ll see that its slug is incremented to hot-update-for-ios-devices-2.

Note

Note: If you get an error related to the immutable property, it may be that you need to run a Composer update to get the latest version of Yii.

There’s more.

1. Use Gii to generate CRUD for the model class app\models\Post and the controller class app\controllers\BlogPostController.
2. Add the following action to controllers/BlogPostController . php:

/\*\*

* @param $slug

\*

* @return string
* @throws NotFoundHttpException

\*/

public function actionSlug($slug)

{

$model = BlogPost::findOne(['slug'=>$slug]); if ($model === null) {

throw new NotFoundHttpException('The requested page does not exist.');

}

return $this->render('view', [

'model' => $model,

]);

}

3. That it is. If you run blogpost/slug with the slug value as sluggablebehavior-test, you will get the following result:

1. It’s suggested that the previous slug recipe be successfully completed with a created instance of Post model.
2. To beautify the URL, add the following urlManager component in config\web.php:

//..

'urlManager' => [

'enablePrettyUrl' => true,

'rules' => [

'blog-post' => 'blog-post/index',

'blog-post/index' => 'blog-post/index',

'blog-post/create' => 'blog-post/create',

'blog-post/view/<id:\d+>' => 'blog-post/view',

'blog-post/update/<id:\d+>' => 'blog-post/update', 'blog-post/delete/<id:\d+>' => 'blog-post/delete',

'blog-post/<slug>' => 'blog-post/slug',

'defaultRoute' => '/site/index',

],

]

//..

1. It’s important that the 'blog-post/<slug>' => 'blog-post/slug' rule is the last in the post URL

rule list.

7. Now, if you go to the page using your slug URL, such as index. php/blog-post/super-quote- title-i/, you will get a result like similar to that in step 3:



See also

For further information, refer to:

* [http://www.yiiframework.com/doc-2.0/vii-behaviors-sluggablebehavior.hrml](http://www.yiiframework.com/doc-2.0/yii-behaviors-sluggablebehavior.html)
* [http://www.viiframework.eom/doc-2.0/guide-runtime-routing.html#url-rules](http://www.yiiframework.com/doc-2.0/guide-runtime-routing.html%23url-rules)