Using standalone actions

In Yii, you can define controller actions as separate classes and then connect them to your controllers.

This will help you to reuse some common functionality.

For example, you can move the backend for autocomplete fields to an action and save some time by not having to write it over and over again.

Another example is that we can create all CRUD operations as separate standalone actions. We will write, create, view, and delete operations of the model and view the list operation of models.

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>.
2. Let’s create post table. Create migration for this using the following command:

./yii migrate/create create\_post\_table

1. Update the just-created migration’s methods and list of imported classes as follows:

<?php

use yii\db\Schema; use yii\db\Migration;

class m150719\_152435\_create\_post\_table extends Migration {

const TABLE\_NAME = '{{%post}}';

public function up()

{

$tableOptions = null;

if ($this->db->driverName === 'mysql') {

$tableOptions = 'CHARACTER SET utf8 COLLATE utf8\_general\_ci ENGINE=InnoDB';

}

$this->createTable(self::TABLE\_NAME, [

'id' => Schema::TYPE\_PK,

'title' => Schema::TYPE\_STRING.'(255) NOT NULL',

'content' => Schema::TYPE\_TEXT.' NOT NULL',

], $tableOptions);

for ($i = 1; $i < 7; $i++) {

$this->insert(self::TABLE\_NAME, [

'title' => 'Test article #'.$i,

'content' => 'Lorem ipsum dolor sit amet, consectetur adipiscing

elit. '

.'Sed sit amet mauris est. Sed at dignissim dui. '

.'Phasellus arcu massa, facilisis a fringilla sit amet, '

. ' rhoncus ut enim.',

]);

}

}

public function down()

{

$this->dropTable(self::TABLE\_NAME);

}

}

1. Install all migrations using the following command:

./yii migrate up

1. Create the Post model using Gii.

How to do it...

1. Create the standalone action @app/actions/CreateAction. php as follows:

<?php

namespace app\actions; use Yii;

use yii\base\Action;

class CreateAction extends Action {

public $modelClass;

public function run()

{

$model = new $this->modelClass();

if ($model->load(Yii::$app->request->post()) && $model->save()) {

$this->controller->redirect(['view', 'id' => $model->getPrimaryKey()]); } else {

return $this->controller->render('//crud/create', [

'model' => $model

]);

}

}

}

1. Create the standalone action @app/actions/DeleteAction. php as follows:

<?php

namespace app\actions;

use yii\base\Action;

use yii\web\NotFoundHttpException;

class DeleteAction extends Action {

public $modelClass;

public function run($id)

{

$class = $this->modelClass;

if (($model = $class::findOne($id)) === null) {

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

}

$model->delete();

return $this->controller->redirect(['index']);

}

}

1. Create the standalone action @app/actions/IndexAction. php as follows:

<?php

namespace app\actions;

use yii\base\Action; use yii\data\Pagination;

class IndexAction extends Action {

public $modelClass; public $pageSize = 3;

public function run()

{

$class = $this->modelClass;

$query = $class::find();

$countQuery = clone $query;

$pages = new Pagination([

'totalCount' => $countQuery->count(),

]);

$pages->setPageSize($this->pageSize);

$models = $query->offset($pages->offset)

->limit($pages->limit)

->all();

return $this->controller->render('//crud/index', [

'pages' => $pages,

'models' => $models

]);

}

}

1. Create the standalone action @app/actions/ViewAction. php as follows:

<?php

namespace app\actions;

use yii\base\Action;

use yii\web\NotFoundHttpException;

class ViewAction extends Action {

public $modelClass;

public function run($id)

{

$class = $this->modelClass;

if (($model = $class::findOne($id)) === null) {

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

}

return $this->controller->render('//crud/view', [

'model' => $model

]);

}

}

1. Create the view file @app/views/crud/create. php as follows:

<?php

use yii\helpers\Html; use yii\widgets\ActiveForm;

/\*

* @var yii\web\View $this \*/

?>

<h1><?= Yii::t('app', 'Create post'); ?></h1>

<?php $form = ActiveForm::begin();?>

<?php $form->errorSummary($model); ?>

<?= $form->field($model, 'title')->textInput() ?>

<?= $form->field($model, 'content')->textarea() ?>

<?= Html::submitButton(Yii::t('app', 'Create'), ['class' => 'btn btn-primary']) ?> <?php ActiveForm::end(); ?>

1. Create the view file @app/views/crud/index.php as follows:

<?php

use yii\widgets\LinkPager; use yii\helpers\Html; use yii\helpers\Url;

/\*

* @var yii\web\View $this
* @var yii\data\Pagination $pages
* @var array $models \*/

?>

<h1>Posts</h1>

<?= Html::a('+ Create a post', Url::toRoute('post/create')); ?>

<?php foreach ($models as $model):?>

<h3><?= Html::encode($model->title);?></h3>

<p><?= Html::encode($model->content);?></p>

<p>

<?= Html::a('view', Url::toRoute(['post/view', 'id' => $model->id]));?> | <?= Html::a('delete', Url::toRoute(['post/delete', 'id' => $model->id]));?>

</p>

<?php endforeach; ?>

<?= LinkPager::widget([

'pagination' => $pages,

]); ?>

1. Create the view file @app/views/crud/view.php as follows:

<?php

use yii\helpers\Html; use yii\helpers\Url;

/\*

* @var yii\web\View $this
* @var app\models\Post $model \*/

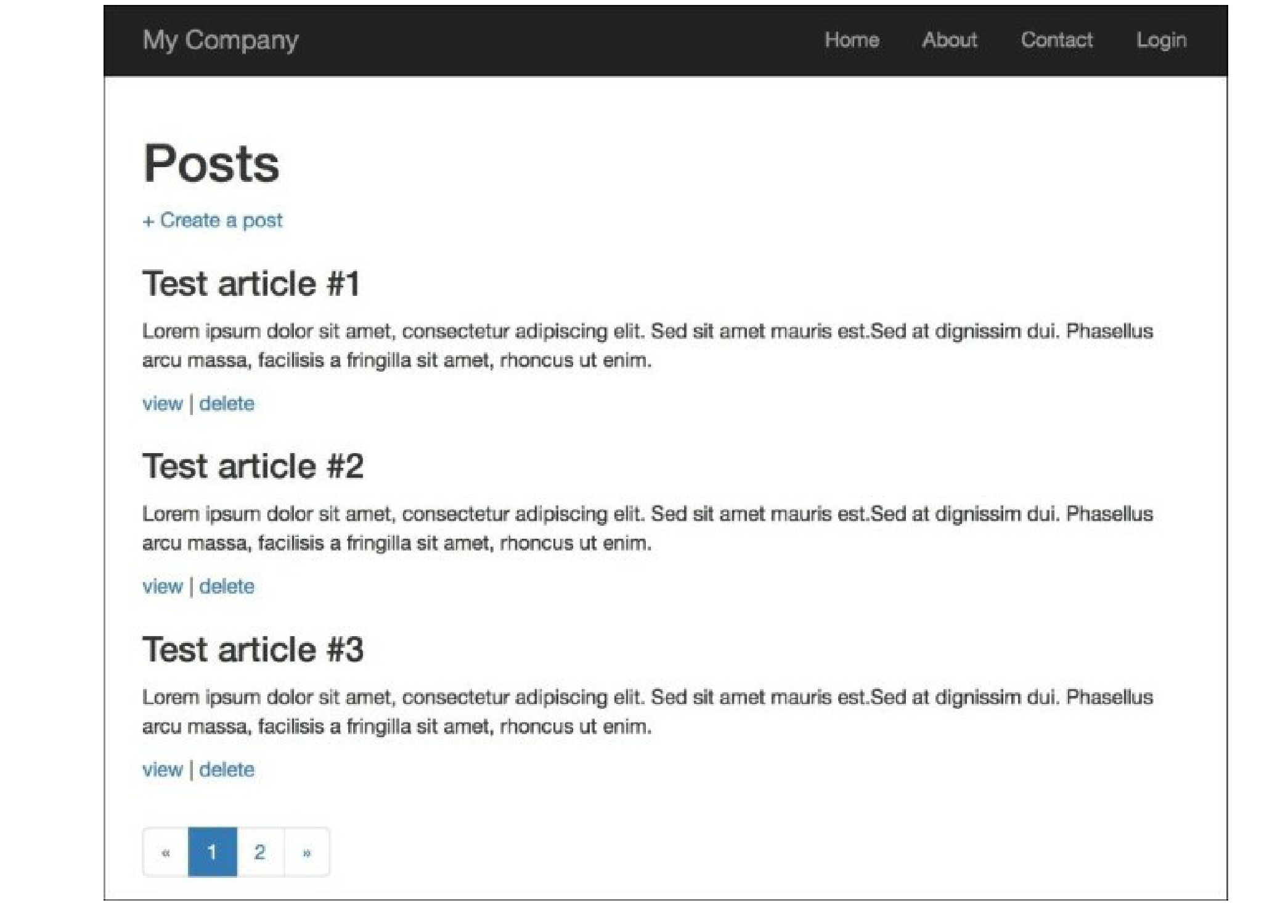
?>

<p><?= Html::a('< back to posts', Url::toRoute('post/index')); ?></p>

<h2><?= Html::encode($model->title);?></h2>

<p><?= Html::encode($model->content);?></p>

To use standalone actions, we declared it in the action map by overriding the actions method. 8. Run post/index:



How it works...

Every controller can be built from standalone actions, like a puzzle from pieces. The difference is that you can make standalone actions very flexible and reuse them in many places.

In our actions, we defined the modelciass public property, which helps to set up a specific Model class in the acti ons method of PostController.

See also

[For further information, refer to http://www.vijframework.com/doc-2.0/guide-structure-](http://www.yiiframework.com/doc-2.0/guide-structure-controllers.html%23standalone-actions) controllers.html#standalone-actions.