Creating reusable controller actions

Common actions such as deleting the AR model by the primary key or getting data for AJAX  
autocomplete could be moved into reusable controller actions and later attached to controllers as needed.

In this recipe, we will create a reusable delete action that will delete the specified AR model by its primary  
key.

Getting ready

1. Create a new yii2-app-basic application using the composer as described in the official guide at  
   [http://www.viiframework.com/doc-2.0/guide-start-installation.html](http://www.yiiframework.com/doc-2.0/guide-start-installation.html).
2. Create a new database and configure it.
3. Create and apply the following migration:

<?php

use yii\db\Migration;

class m160308\_093233\_create\_post\_table extends Migration  
{

public function up()

{

$this->createTable('{{%post}}', [

'id' => $this->primaryKey(),

'title' => $this->string()->notNull(),

'text' => $this->text()->notNull(),

]);

}

public function down()

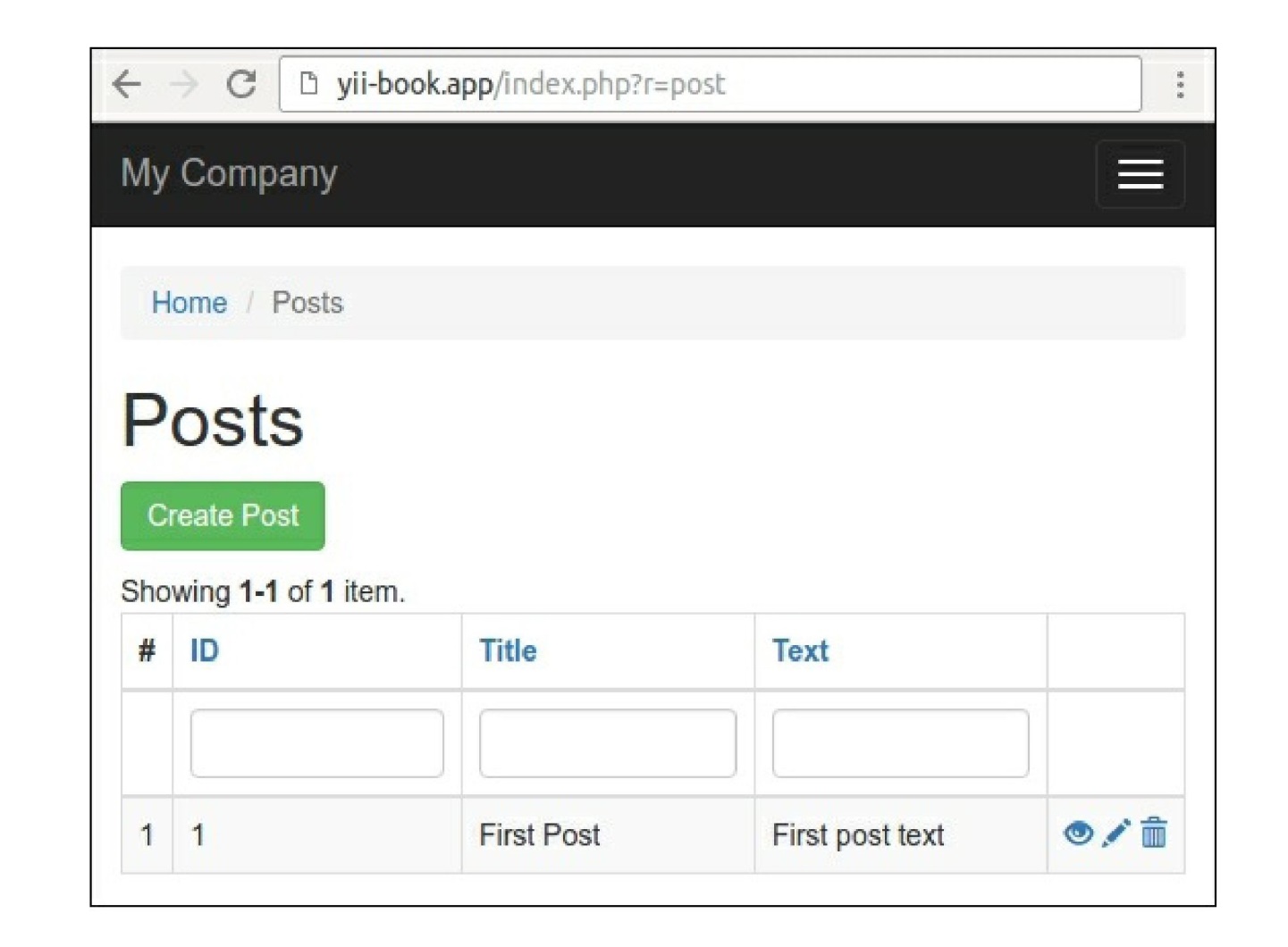
{

$this->dropTable('{{%post}}');

}

}

1. Generate models for posts and comments using Gii.
2. Generate the standard CRUD controller app\controllers\PostController in Gii.
3. Ensure that CRUD properly works:



7. In a success case, add a set of example posts.

How to do it...

Carry out the following steps:

1. Create the actions directory and add the DeleteAction standalone action:

<?php

namespace app\actions;

use yii\base\Action;  
use yii\base\InvalidConfigException;  
use yii\web\MethodNotAllowedHttpException;  
use yii\web\NotFoundHttpException;

class DeleteAction extends Action  
{

public $modelClass;

public $redirectTo = ['index'];

public function init()

{

if (empty($this->modelClass)) {

throw new InvalidConfigException('Empty model class.');

}

parent::init();

}

public function run($id)

{

if (!\Yii::$app->getRequest()->getIsPost()) {

throw new MethodNotAllowedHttpException('Method not allowed.');

}

$model = $this->findModel($id);

$model->delete();

return $this->controller->redirect($this->redirectTo);

}

/\*\*

* @param $id
* @return \yii\db\ActiveRecord
* @throws NotFoundHttpException  
  \*/

private function findModel($id)

{

$class = $this->modelClass;

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

} else {

throw new NotFoundHttpException('Page does not exist.');

}

}

}

2. Now we need to attach it to the controllers/PostController.php controller. Remove the  
controller’s actionDelete and behaviors methods and attach your own action in the action  
method:

<?php

namespace app\controllers;

use app\actions\DeleteAction;  
use Yii;

use app\models\Post;

use app\models\PostSearch;

use yii\web\Controller;

use yii\web\NotFoundHttpException;

class PostController extends Controller  
{

public function actions()

{

return [

'delete' => [

'class' => DeleteAction::className(),

'modelClass' => Post::className(),

],

];

}

public function actionIndex() { ... }

public function actionView($id) { ... }

public function actionCreate() { ... }

public function actionUpdate($id) { ... }

protected function findModel($id)

{

if (($model = Post::findOne($id)) !== null) {  
return $model;

} else {

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

}

}

}

3. That is it. Ensure that the delete operation still works correctly, and after the deletion, you will be  
redirected to a corresponding index action.

How it works...

To create an external controller action, you need to extend your class from yii\base\Action. The only  
mandatory method to implement is run. In our case, it accepts the parameter named $id from $\_get using  
the automatic parameter binding feature of Yii and tries to delete a corresponding model.

To make it customizable, we’ve created two public properties configurable from the controller. These are  
modelName, which holds the name of the model we are working with, and redirectTo that specifies a  
route the user will be redirected to.

The configuration itself is done by implementing the actions method in your controller. There, you can  
attach the action once or multiple times and configure its public properties.

You can access the original controller object via the controller property if you need it to redirect to another  
action or render a specific view.

See also

* [To learn more about controllers and actions refer, to http://www.yiiframework.com/doc-2.0/guide-](http://www.yiiframework.com/doc-2.0/guide-structure-controllers.html)  
  structure-controllers.html
* The Creating reusable controllers recipe in this chapter