Табличный ввод

В этом разделе мы покажем вам, как использовать модель, чтобы сохранить и проверить связанные модели. Иногда вам нужно будет обрабатывать несколько моделей одного и того же вида в одной форме.

Например, у нас есть конкурсы и призы для конкурсов. Любой конкурс может содержать неограниченное количество призов. Таким образом, нам нужна возможность создать конкурс с призами, проверить их, отобразить все ошибки и сохранить основную модель (модель конкурса) и все связанные модели (модели призов) в базу данных.

Подготовка

1. Создайте новое приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу  
   <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

1. Создавайте миграции для конкурсных и призовых таблиц с помощью следующих команд:

./yii migrate/create create\_table\_contest\_and\_prize\_table

Обновите только что созданные методы миграции up () и down () следующим кодом

public function up()

{

$tableOptions = null;

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

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

}

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

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

'name' => Schema::TYPE\_STRING . ' NOT NULL',

], $tableOptions);

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

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

'name' => Schema::TYPE\_STRING,

'amount' => Schema::TYPE\_INTEGER,

], $tableOptions);

$this->createTable('{{%contest\_prize\_assn}}', [

'contest\_id' => Schema::TYPE\_INTEGER,

'prize\_id' => Schema::TYPE\_INTEGER,

], $tableOptions);

$this-addForeignKey('fk\_contest\_prize\_assn\_contest\_id',  
'{{%contest\_prize\_assn}}', 'contest\_id', {{%contest}}', 'id');

$this->addForeignKey('fk\_contest\_prize\_assn\_prize\_id',

'{{%contest\_prize\_assn}}', 'prize\_id', '{{%prize}}', 'id');

}

public function down()

{

$this-dropForeignKey('fk\_contest\_prize\_assn\_contest\_id',

'{{%contest\_prize\_assn}}');

$this->dropForeignKey('fk\_contest\_prize\_assn\_prize\_id',

'{{%contest\_prize\_assn}}');

$this->dropTable('{{%contest\_prize\_assn}}');

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

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

}

1. Затем установите миграцию с помощью следующей команды:

./yii migrate/up

1. С помощью GII, создайте модели Contest, Priz, и ContestPrizeAssn.

How to do it...

1. Давайте создадим @app/controllers/ContestController.php со следующим кодом:  
   <?php

namespace app\controllers;  
use app\models\Contest;  
use app\models\ContestPrizeAssn;  
use app\models\Prize;  
use Yii;

use yii\base\Model;

use yii\helpers\VarDumper;

use yii\web\Controller;

class ContestController extends Controller  
{

public function actionCreate()

{

$contestName = 'Happy New Year';

$firstPrize = new Prize();

$firstPrize->name = 'Iphone 6s';

$firstPrize->amount = 4;

$secondPrize = new Prize();

$secondPrize->name = 'Sony Playstation 4';

$secondPrize->amount = 2;

$contest = new Contest();

$contest->name = $contestName;

$prizes = [$firstPrize, $secondPrize];

if ($contest->validate() && Model::validateMultiple($prizes)) {  
$contest->save(false);  
foreach ($prizes as $prize) {

$prize->save(false);

$contestPrizeAssn = new ContestPrizeAssn();  
$contestPrizeAssn->prize\_id = $prize->id;  
$contestPrizeAssn->contest\_id = $contest>id;  
$contestPrizeAssn->save(false);

}

return $this->renderContent(

'All prizes have been successfully saved!'

);

} else {

return $this->renderContent(

VarDumper::dumpAsString($contest->getErrors())

);

}

}

public function actionUpdate()

{

$prizes = Prize::find()->all();

if (Model::loadMultiple($prizes, Yii::$app->request->post()) &&Model::validateMultiple($prizes))

{

foreach ($prizes as $prize) {

$prize->save(false);

}

return $this->renderContent(

'All prizes have been successfully saved!'

);

}

return $this->render('update', ['prizes' => $prizes]);

}

}

1. Create @app/views/contest/update.php and place the following code inside it:

<?php

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

$form = ActiveForm::begin();  
foreach ($prizes as $i => $prize) {

echo $form->field($prize, "[$i]amount")->label($prize->name);

}

echo Html::submitButton('submit' , ['class' => 'btn btn-success']);

ActiveForm::end();

How it works...

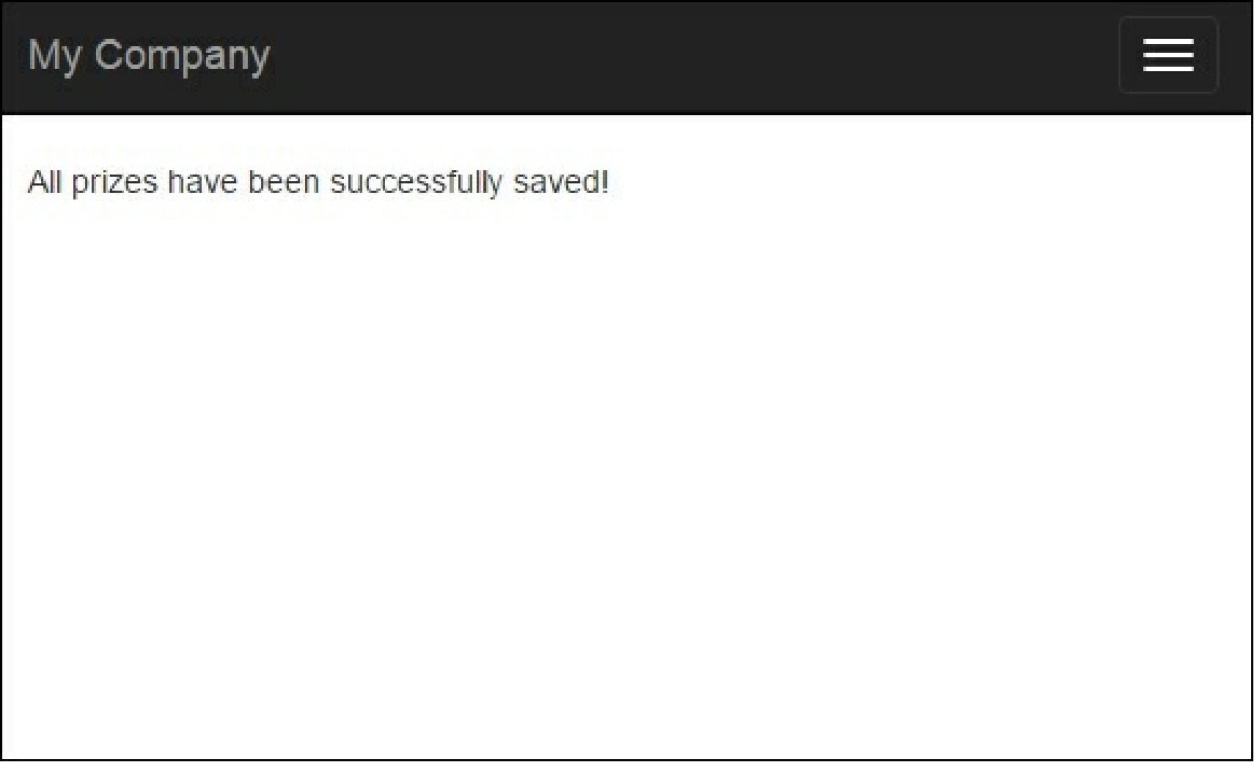
The following information shows how to implement tabular input with Yii.

In the contest/update action, we will be able to display all prizes with their amounts and edit them all at  
once. We’ve used two special Yii methods:

* Model: : loadMultiple (): This method populates a set of models with data from the end user
* Model: : validateMultiple(): This methods takes a set of models and validates them all at once

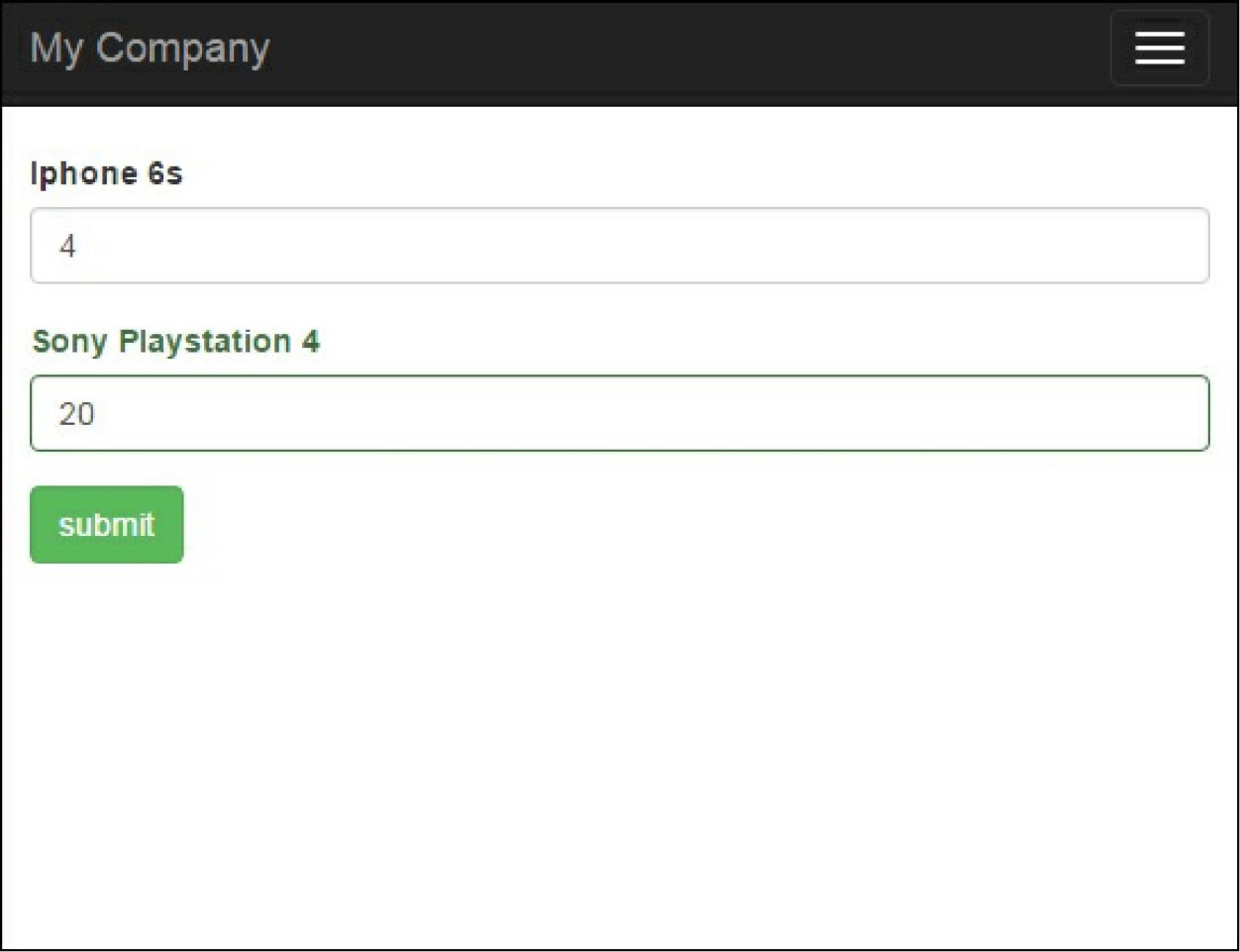
Because we’ve validated our models before with validateMultiple(), we’re passing false as a  
parameter to save() to avoid running validation twice.

First, visit /index. php?r=contest/create page. After visiting, you will see the page that will validate and  
create ' Happy New Year' with two prizes, and will pass the prizes to the current contest model. You  
should note that we will only save the contest model and prizes to the database if they are valid:



It is provided by following condition:

if ($contest->validate() && Model::validateMultiple($prizes)) { ...}  
Go to the /index. php?r=contest/update page and you will see this form:



In the @app/views/contest/update. php for each prize, we render a name and an input with an amount.  
We must add an index to each input name so that Model: : loadMultiple() may identify which model to  
fill with which values.

In conclusion, this approach is used for collecting tabular input data when you process all your attributes  
from a view form and populate parent and related models from the form.

See also

For further information, refer to the following URL:

[http://www.yiiframework.com/doc-2.0/guide-input-tabular-input.html#co]lecting-tabular-input](http://www.yiiframework.com/doc-2.0/guide-input-tabular-input.html%23collecting-tabular-input)