Complex forms with multiple models

When dealing with some complex data, it is possible that you may need to use multiple different models to  
collect the user input. For example, you have an order form with user information such as first name, last  
name, and phone number; you also have a delivery address and some kind of product.

You would like to save all this data in one form. With Yii models and support forms, you can easily do this.  
Assuming that the user info will be stored in the user table and in the order form, we will save product  
information and the user\_id of the user who has ordered a product. We also have a product table with  
some information in it.

Getting ready

1. Create a new application by using the Composer package manger, as described in the official guide  
   at <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>**.**
2. Create migrations for contest and prize tables with the following commands:

./yii migrate/create create\_order\_tables

1. Update the newly-created migration’s up() and down () methods with the following code:

<?php

use yii\db\Schema;  
use yii\db\Migration;  
use app\models\Product;

class m150813\_161817\_create\_order\_form\_tables extends Migration  
{

public function up()

{

$tableOptions = null;

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

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

}

$this->createTable('user', [

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

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

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

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

], $tableOptions);

$this->createTable('product', [

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

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

'price' => Schema::TYPE\_FLOAT . '(6,2) ',

], $tableOptions);

$this->createTable('order', [

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

'user\_id' => Schema::TYPE\_INTEGER . ' NULL',

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

'product\_id' => Schema::TYPE\_INTEGER . ' NOT NULL',

], $tableOptions);

$product1 = new Product();

$product1->title = 'Iphone 6';

$product1->price = 400.5;

$product1->save();

$product3 = new Product();

$product3->title = 'Samsung Galaxy Note 5';

$product3->price = 900;

$product3->save();

$this->addForeignKey('fk\_order\_product\_id', 'order', 'product\_id',

' product', 'id');

}

public function down()

{

$this->dropTable('order');

$this->dropTable('user');

$this->dropTable('product');

}

}

1. Then, install migration with the following command:

./yii migrate/up

1. With Gii, generate user, order, and product models.

How to do it...

1. Create @app/controllers/TestController with the following code:

<?php

namespace app\controllers;  
use app\models\Order;  
use app\models\User;  
use Yii;

use yii\web\Controller;  
class TestController extends Controller  
{

public function actionOrder()

{

$user = new User();

$order = new Order();

if ($user->load(Yii::$app->request->post()) && $order->load(Yii::$app-  
>request->post())) {

if ($user->validate() && $order->validate()) {

$user->save(false);

$order->user\_id = $user->id;

$order->save(false);

$this->redirect(['/test/result', 'id' => $order->id]);

}

}

return $this->render('order', ['user' => $user, 'order' => $order]);

}

public function actionResult($id)

{

$order = Order::find($id)->with('product', 'user')->one();  
return $this->renderContent(

'Product: ' . $order->product->title . '</br>' .

'Price: ' . $order->product->price . '</br>' .

'Customer: ' . $order->user->first\_name . ' ' . $order->user->last\_name

. '</br>' .

'Address: ' . $order->address

);

}

}

1. Then create a view file, @app/views/test/order. php, and add the following code:

<?php

use yii\helpers\Html;

use yii\widgets\ActiveForm;  
use app\models\Product;  
use yii\helpers\ArrayHelper;

/\*\*

* @var $user \app\models\User
* @var $order \app\models\Order  
  \*/

$form = ActiveForm::begin([

'id' => 'order-form',

'options' => ['class' => 'form-horizontal'],

]) ?>

<?= $form->field($user, 'first\_name')->textInput(); ?>

<?= $form->field($user, 'last\_name')->textInput(); ?>

<?= $form->field($user, 'phone')->textInput(); ?>

<?= $form->field($order, 'product\_id')-

>dropDownList(ArrayHelper::map(Product::find()->all(), 'id', 'title')); ?>

<?= $form->field($order, 'address')->textInput(); ?>

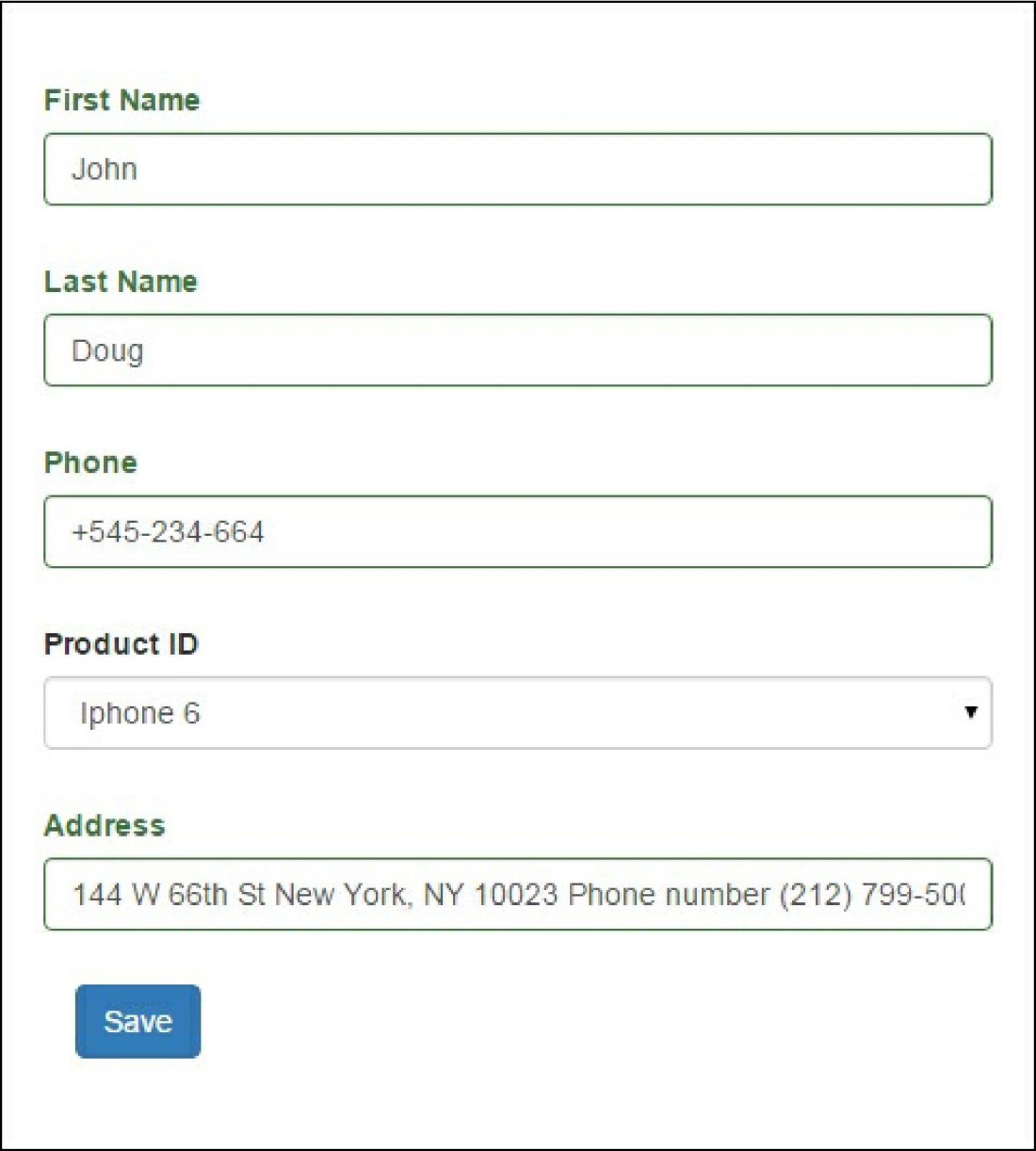
<?= Html::submitButton('Save', ['class' => 'btn btn-primary']) ?>

<?php ActiveForm::end() ?>

How it works...

You can see the form at http: //yii-book. app/index. php?r=test/order. Our form collects information  
from the user and order models.

Let’s fill out our form:



**After saving, you will see the following result:**



In the controller, we validate and store it. Of course, this example is very simple. In real projects, you may  
have more than one model and you will be able to use this approach for them. This approach is very useful  
when you want to create or update more than one instance in the same form.

See also

For further information, refer to [http://www.yiiframework.com/doc-2.0/guide-inpnt-mnltiple-models.html](http://www.yiiframework.com/doc-2.0/guide-input-multiple-models.html)