Service locator

Вместо того чтобы вручную создавать экземпляры различных общих служб (компонентов приложения), мы можем получить их из специального глобального объекта, который содержит конфигурации и экземпляры всех компонентов.

Локатор служб-это глобальный объект, который содержит список компонентов или определений, однозначно идентифицируемых по идентификатору, и позволяет получить любой необходимый экземпляр по его идентификатору. Локатор создает один экземпляр компонента "на лету" при первом вызове и возвращает предыдущий экземпляр при последующих вызовах.

В этом рецепте мы создадим компонент корзины и напишем контроллер корзины для работы с ним.

Подготовка

Создайте новое приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу [http://www.viiframework.com/doc-2.0/guide-start-installation.html](http://www.yiiframework.com/doc-2.0/guide-start-installation.html).

(от переводчика на русском http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation)

Как это сделать...

Для создания компонента корзины выполните следующие действия:

1. Создайте компонент корзины покупок. Он будет хранить выбранные элементы в сеансе пользователя:

<?php

namespace app\components;

use Yii;

use yii\base\Component;

class ShoppingCart extends Component

{

public $sessionKey = 'cart';

private $\_items = [];

public function add($id, $amount)

{

$this->loadItems();

if (array\_key\_exists($id, $this->\_items)) {

$this->\_items[$id]['amount'] += $amount;

} else {

$this->\_items[$id] = [

'id' => $id,

'amount' => $amount,

];

}

$this->saveItems();

}

public function remove($id)

{

$this->loadItems();

$this->\_items = array\_diff\_key($this->\_items, [$id => []]);

$this->saveItems();

}

public function clear()

{

$this->\_items = [];

$this->saveItems();

}

public function getItems()

{

$this->loadItems();

return $this->\_items;

}

private function loadItems()

{

$this->\_items = Yii::$app->session->get($this->sessionKey, []);

}

private function saveItems()

{

Yii::$app->session->set($this->sessionKey, $this->\_items);

}

}

1. Создайте контроллер корзины:

<?php

namespace app\controllers;

use app\models\CartAddForm;

use Yii;

use yii\data\ArrayDataProvider;

use yii\filters\VerbFilter;

use yii\web\Controller;

class CartController extends Controller

{

public function behaviors()

{

return [

'verbs' => [

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

'actions' => [

'delete' => ['post'],

],

],

];

}

public function actionIndex()

{

$dataProvider = new ArrayDataProvider([

'allModels' => Yii::$app->cart->getItems(),

]);

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

'dataProvider' => $dataProvider,

]);

}

public function actionAdd()

{

$form = new CartAddForm();

if ($form->load(Yii::$app->request->post()) && $form->validate()) {

Yii::$app->cart->add($form->productId, $form->amount);

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

}

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

'model' => $form,

]);

}

public function actionDelete($id)

{

Yii::$app->cart->remove($id);

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

}

}

1. Создаем форму:

<?php

namespace app\models;

use yii\base\Model;

class CartAddForm extends Model {

public $productId; public $amount;

public function rules()

{

return [

[['productId', 'amount'], 'required'],

[['amount'], 'integer', 'min' => 1],

];

}

}

1. Создаем представление views/cart/index.php :

<?php

use yii\grid\ActionColumn; use yii\grid\GridView; use yii\grid\SerialColumn; use yii\helpers\Html;

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

/\* @var $dataProvider yii\data\ArrayDataProvider \*/

$this->title = 'Cart';

$this->params['breadcrumbs'][] = $this->title;

?>

<div class="site-contact">

<h1><?= Html::encode($this->title) ?></h1>

<p><?= Html::a('Add Item', ['add'], ['class' => 'btn btn-success']) ?></p>

<?= GridView::widget([

'dataProvider' => $dataProvider,

'columns' => [

['class' => SerialColumn::className()],

'id:text:Product ID',

'amount:text:Amount',

[

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

'template' => '{delete}',

]

],

]) ?>

</div>

1. Создаем представление views/cart/add. php:

<?php

use yii\helpers\Html;

use yii\bootstrap\ActiveForm;

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

/\* @var $form yii\bootstrap\ActiveForm \*/

/\* @var $model app\models\CartAddForm \*/

$this->title = 'Add item';

$this->params['breadcrumbs'][] = ['label' => 'Cart', 'url' => ['index']]; $this->params['breadcrumbs'][] = $this->title;

?>

<div class="site-contact">

<h1><?= Html::encode($this->title) ?></h1>

<?php $form = ActiveForm::begin(['id' => 'contact-form']); ?>

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

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

<div class="form-group">

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

<?php ActiveForm::end(); ?>

</div>

1. Добавляем пункты в меню:

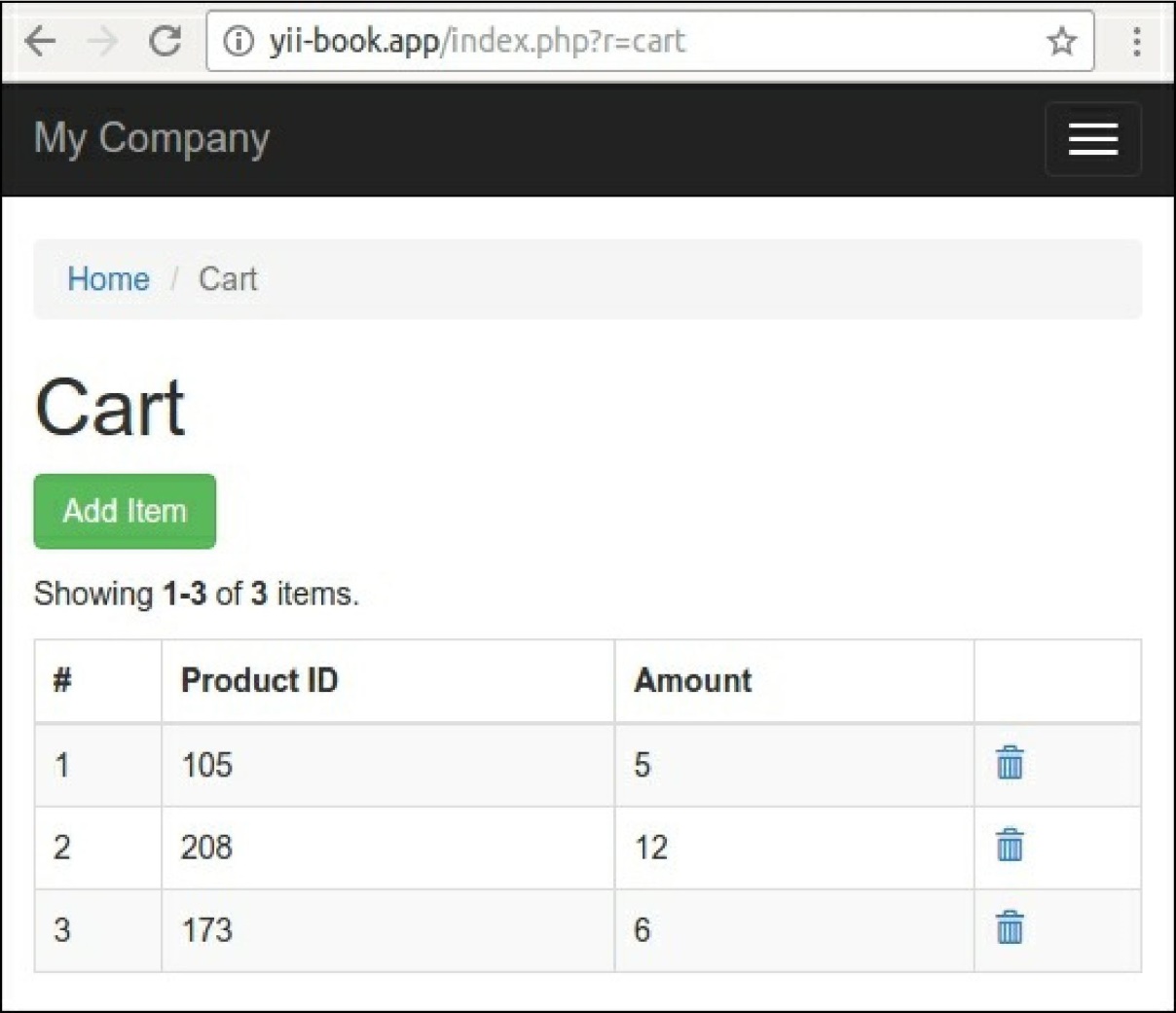
['label' => 'Home', 'url' => ['/site/index']],

['label' => 'Cart', 'url' => ['/cart/index']],

['label' => 'About', 'url' => ['/site/about']],

// ...

1. Откройте страницу корзина и попробуйте добавить строки:



Как это работает

Прежде всего, мы создали собственный класс с опцией public sessionKey:

<?php

namespace app\components;

use yii\base\Component;

class ShoppingCart extends Component {

public $sessionKey = 'cart';

// ...

}

Во-вторых, мы добавили определение компонента в раздел components файла конфигурации: 'components' => [

'cart => [

Сейчас мы можем получить экземпляр компонента двумя способами:

]

],

'class' => 'app\components\ShoppingCart', 'sessionKey' => 'primary-cart',

$cart = Yii::$app->cart;

$cart = Yii::$app->get('cart');

И мы можем использовать этот объект в наших собственных контроллерах, виджетах и других местах.

Когда мы вызываем любой компонент как корзина:

Yii::$app->cart

Мы вызываем виртуальное свойство экземпляра класса Application в статической переменной Yii:: $app. Класс yii\base\Application расширяет в yii\base\Module, который расширяет

yii\di\ServiceLocator с волшебным методом \_\_get. Этот волшебный метод просто вызывает метод get()

из класса yii\di\ServiceLocator :

namespace yii\di;

class ServiceLocator extends Component {

private $\_components = []; private $\_definitions = [];

public function \_\_get($name)

{

if ($this->has($name)) {

return $this->get($name);

} else {

return parent:: get($name);

}

}

// ...

}

В результате это альтернатива непосредственному вызову сервиса через get метод:

Yii::$app->get('cart);

Когда мы получаем компонент из метода Get сервис локатора, локатор находит необходимое определение в своем списке \_definitions и в случае успеха создает новый объект по определению на лету, регистрирует его в своем собственном списке экземпляров \_ Components и возвращает объект.

Если мы получим какой-то компонент, множественный запуск локатора всегда будет возвращать предыдущий сохраненный экземпляр снова и снова:

$cart1 = Yii::$app->cart;

$cart2 = Yii::$app->cart;

var\_dump($cart1 === $cart2); // bool(true)

Это позволяет нам использовать общий экземпляр одной корзины Yii: :$app->cart или одно соединение с базой Yii: :$app->db вместо того чтобы создавать один большой набор с нуля снова и снова.

Смотреть также

* Дополнительные сведения о локаторе служб и основных компонентах платформы см. в разделе [http://www.yiiframework.eom/doc-2.0/guide-concept-service-locator.html](http://www.yiiframework.com/doc-2.0/guide-concept-service-locator.html)

(на русском <http://yiiframework.domain-na.me/doc/guide/2.0/ru/concept-service-locator>)

* Рецепт Конфигурация компонентов
* Рецепт Создание компонентов в главе 8, Extending Yii