Yii::$container->set('app\cart\storage\StorageInterface',

'app\cart\storage\CustomStorage', );

Но наш класс SessionStorage имеет нестандартный конструктор:

class SessionStorage implements StorageInterface {

public function \_\_construct(Session $session, $key) { ... }

}

Поэтому мы используем анонимную функцию для создания экземпляра вручную:

Yii::$container->set('app\cart\storage\StorageInterface', function() {

return new SessionStorage(Yii::$app->session, 'primary-cart');

});

И в конце концов мы можем получить объект cart из контейнера вручную в наших собственных контроллерах, виджетах и других местах:

$cart = Yii::createObject('app\cart\ShoppingCart')

Каждый контроллер и другие объекты будут созданы с помощью метода createObject внутри фреймворка. И мы можем использовать инжекцию через конструктор контроллера:

class CartController extends Controller {

private $cart;

public function \_\_construct($id, $module, ShoppingCart $cart, $config = [])

{

$this->cart = $cart;

parent:: construct($id, $module, $config);

}

// ...

}

Используйте этот впрыснутый объект :

public function actionDelete($id)

{

$this->cart->remove($id);

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

}

Смотреть также

• Дополнительные сведения о DIP см [https://en.wikipedia.org/wiki/Dependencv inversion principle](https://en.wikipedia.org/wiki/Dependency_inversion_principle)

• Чтобы узнать больше о контейнере внедрения зависимостей, обратитесь к [http://www.yiiframework.com/doc-2.0/gurde-concept-di-container.html](http://www.yiiframework.com/doc-2.0/guide-concept-di-container.html)

(от переводчика: на русском

<http://yiiframework.domain-na.me/doc/guide/2.0/ru/concept-di-container> )