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.