Right now we can retrieve the component instance in two ways:

]

],

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

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

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

And we can use this object in our own controllers, widgets, and other places.

When we call any component such as cart:

Yii::$app->cart

We call the virtual property of the Application class instance in the Yii: : $app static variable. But the yii\base\Application class extends the yii\base\Module class, which extends the

yii\di\ServiceLocator class with the get magic method. This magic method just calls the get()

method of the yii\di\ServiceLocator class:

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);

}

}

// ...

}

As a result it is an alternative to directly calling the service via the get method:

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

When we get a component from the get method of service locator, the locator finds needed definition in its \_definitions list and if successful it creates a new object by the definition on the fly, registers it in its own list of complete instances \_components and returns the object.

If we get some component, multiplying the locator will always return the previous saved instance again and again:

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

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

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

It allows us to use the shared single cart instance Yii: :$app->cart or single database connection Yii: :$app->db instead of creating one large set from scratch again and again.