{

public function construct(Session $session, $key) { ... }

}

Right now we can create an instance of the cart manually like this:

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

$cart = new ShoppingCart($storage)

Это позволяет нам создавать множество различных реализаций, таких как SessionStorage, CookieStorage или DbStorage. И мы можем повторно использовать независимый от фреймворка класс ShoppingCart с StorageInterface в разных проектах и разных фреймворках. Мы должны только реализовать класс хранения с методами интерфейса для необходимой платформы.

Но вместо того, чтобы вручную создавать экземпляр со всеми зависимостями, мы можем использовать контейнер внедрения зависимостей.

По умолчанию контейнер анализирует конструкторы всех классов и рекурсивно создает все необходимые экземпляры. Например, если у нас четыре класса:

class A

{

public function \_\_construct(B $b, C $c) { ... }

}

class B

{

...}

class C

{

public function \_\_construct(D $d) { ... }

}

class D

{

...

}

Мы можем получить экземпляр класса двумя способами:

$a = Yii::$container->get('app\services\A')

// или

$a = Yii::createObject('app\services\A')

Контейнер автоматически создает экземпляры классов B, D, C и A и внедряет их друг в друга.

В нашем случае мы отмечаем корзину как синглтон: Yii::$container->setSingleton('app\cart\ShoppingCart');

Это означает, что контейнер будет возвращать один экземпляр для каждого повторного вызова вместо создания корзины снова и снова.

Кроме того, наш ShoppingCart имеет Тип StorageInterface в своем собственном конструкторе, и контейнер знает, какой класс он должен создать для этого типа. Мы должны вручную привязать класс к интерфейсу, как это: