{

$this->\_items = $this->storage->load();

}

private function saveItems()

{

$this->storage->save($this->\_items);

}

}

2. Он будет работать только с собственными предметами. Вместо встроенного хранения элементов в сеансе он делегирует эту ответственность любому внешнему классу хранения, который будет реализовывать Интерфейс Storageinterface.

3. Класс cart просто получает объект хранения в собственный конструктор, сохраняет его экземпляр в закрытый $storage поле и вызывает load() и save() методы.

4. Определите общий интерфейс хранения корзины с необходимыми методами:

<?php

namespace app\cart\storage;

interface StorageInterface {

/\*\*

\*@return array of cart items

\*/

public function load();

/\*\*

\*@param array $items from cart

\*/

public function save(array $items);

}

5. Создайте простую реализацию хранилища. Он будет хранить выбранные элементы в сеансе сервера:

<?php

namespace app\cart\storage;

use yii\web\Session;

class SessionStorage implements StorageInterface {

private $session;

private $key;

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

{

$this->key = $key;

$this->session = $session;

}

public function load()

{

return $this->session->get($this->key, []);

}

public function save(array $items)

{

$this->session->set($this->key, $items);

}

}

6. Хранилище получает любой экземпляр framework session в constructor и использует его позже для получения