{

$this->destroyApplication(); parent::afterTestMethod($method);

}

protected function mockApplication()

{

new Application([

'id' => 'testapp',

'basePath' => DIR ,

'vendorPath' => dirname( DIR ) . '/vendor',

'components' => [

'session' => [

'class' => 'yii\web\Session',

],

]

]);

}

protected function destroyApplication()

{

\Yii::$app = null;

\Yii::$container = new Container();

}

}

1. Add a memory-based clean fake class that implements the StorageInterface interface:

<?php

namespace book\cart\tests;

use book\cart\storage\StorageInterface;

class FakeStorage implements StorageInterface {

private $items = [];

public function load()

{

return $this->items;

}

public function save(array $items)

{

$this->items = $items;

}

}

This will store items into a private variable instead of working with the real session. It allows us to run tests independently (without real storage driver) and also improves testing performance.

1. Add the Cart test class:

<?php

namespace book\cart\tests\units;

use book\cart\tests\FakeStorage; use book\cart\Cart as TestedCart; use book\cart\tests\TestCase;

class Cart extends TestCase {

/\*\*

@var TestedCart