Создание команд CLI

Yii имеет хорошую поддержку командной строки и позволяет создавать повторно используемые консольные команды. Консольные команды создаются быстрее, чем веб-GUI. Если вам нужно создать какую-то утилиту для вашего приложения, которая будет использоваться разработчиками или администраторами, консольные команды являются правильным инструментом.

Чтобы показать, как создать консольную команду, мы создадим простую команду, которая будет очищать различные вещи, такие как активы и временные каталоги.

Подготовка

Создайте новое приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу  
<http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.   
По русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation>.

Как это сделать...

Выполните следующие шаги для создания команд CLI:

1. Создайте commands/cleanController.php файл со следующим кодом:

<?php

namespace app\commands;

use yii\console\Controller;  
use yii\helpers\FileHelper;

/\*\*

\* Removes content of assets and runtime directories.

\*/

class CleanController extends Controller  
{

public $assetPaths = ['@app/web/assets'];  
public $runtimePaths = ['©runtime'];

/\*\*

* Removes temporary assets.

\*/

public function actionAssets()

{

foreach ((array)$this->assetPaths as $path) {

$this->cleanDir($path);

}

$this->stdout('Done' . PHP\_EOL);

}

/\*\*

* Removes runtime content.

\*/

public function actionRuntime()

{

foreach ((array)$this->runtimePaths as $path) {

$this->cleanDir($path);

}

$this->stdout('Done' . PHP\_EOL);

}

private function cleanDir($dir)

{

$iterator = new \DirectoryIterator(\Yii::getAlias($dir));  
foreach($iterator as $sub) {

if(!$sub->isDot() && $sub->isDir()) {

$this->stdout('Removed ' . $sub->getPathname() . PHP\_EOL);

FileHelper::removeDirectory($sub->getPathname());

}

}

}

}

1. Теперь мы можем использовать наш собственный контроллер консоли с настройками по умолчанию. Просто запустите сценарий оболочки yii:

. /yii

1. Ищите собственные чистые команды:

This is Yii version 2.0.7.

The following commands are available:

* asset Allows you to combine...

asset/compress Combines and compresses the asset.

asset/template Creates template of configuration file.

* clean Removes content of assets and runtime directories.

clean/assets Removes temporary assets.

clean/runtime Removes runtime content.

* fixture Manages fixture data loading and unloading.

fixture/load (default) Loads the specified fixture data.  
fixture/unload Unloads the specified fixtures.

1. Прямо сейчас запустите asset cleaning:

.yii clean/assets

1. Смотрите отчет о процессе:

Removed /yii-book.app/web/assets/25f82b8a  
Removed /yii-book.app/web/assets/9b3b2888  
Removed /yii-book.app/web/assets/f4307424  
Done

1. Если вы хотите использовать этот контроллер в приложении yii2-app-advanced, просто укажите пользовательские рабочие пути:

return [

'id' => 'app-console',

'basePath' => dirname( DIR ),

'bootstrap' => ['log'],

'controllerNamespace' => 'console\controllers',

'controllerMap' => [

'clean' => [

'class' => 'console\controllers\CleanController',

'assetPaths' => [

'@backend/web/assets',

'@frontend/web/assets',

],

'runtimePaths' => [

'@backend/runtime',

'@frontend/runtime',

'@console/runtime',

],

],

],

// ...

];

Как это работает...

Все консольные команды должны быть расширены из класса yii\console\controller. Поскольку все консольные команды выполняются в yii\console\Application вместо yii\web\Application, у нас нет способа определить значение псевдонима @webroot. Кроме того, в шаблоне yii2-app-advanced по умолчанию есть подкаталоги backend, frontend и console. Для этого мы создаем настраиваемые общедоступные свойства assetPaths и runtimePaths.

Сама структура консольных команд похожа на типичный контроллер. Мы определяем несколько действий, которые мы можем выполнить через yii < консольная команда> / <командное действие>.

Как вы можете видеть, представления не используются, поэтому мы можем сосредоточиться на задачах программирования, а не на дизайне, разметке и так далее. Тем не менее, вам нужно предоставить некоторые полезные результаты, чтобы пользователи знали, что происходит. Это делается с помощью простых операторов PHP echo.

Если ваша команда относительно сложна, например message или migrate в комплекте с Yii, рекомендуется предоставить дополнительное описание доступных опций и действий. Это можно сделать, переопределив метод getHelp:

public function getHelp()

{

$out = "Clean command allows you to clean up various temporary data Yii and an application are generating.\n\n";

return $out . parent::getHelp();

}

Выполните следующую команду:

./yii help clean

Вы можете увидеть полный вывод следующим образом:

DESCRIPTION

Clean command allows you to clean up various temporary data Yii and an application are generating.

Removes content of assets and runtime directories.

SUB-COMMANDS

* clean/assets Removes temporary assets.
* clean/runtime Removes runtime content.

По умолчанию при выполнении команды оболочки:

. /yii

Мы видели упрощенное описание всех команд в списке вывода:

- clean Removes content of assets and runtime directories.

clean/assets Removes temporary assets.

clean/runtime Removes runtime content.

Это описание будет взято из комментариев перед классом и действия:

/\*\*

\* Removes content of assets and runtime directories.

\*/

class CleanController extends Controller  
{

/\*\*

* Removes temporary assets.

\*/

public function actionAssets() { ... }

* Removes runtime content.

\*/

public function actionRuntime() { ... }

}

Он является обязательным для добавления описаний для ваших классов. Вы не должны делать это для своих собственных команд CLI.

Смотрите так же

* Рецепт создания многоразовых контроллеров в этой главе
* Изготовление расширений дистрибутив - готовый рецепт в этой главе