eBook Gratuit

APPRENEZ zend-framework2

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#zend-framework2

Table des matières

À propos 1

Chapitre 1: Démarrer avec zend-framework2 2

Remarques 2

Examples 2

Installation ou configuration 2

Utilisation de Composer - Méthode recommandée 2

Utilisation de sous-modules Git 3

Configuration du serveur HTTP 3

OPTION 1 - Serveur PHP CLI 3

OPTION 2 - Un serveur HTTP personnalisé 4

Un simple Hello World 4

Comment créer une usine 5

Chapitre 2: Installation 8

Examples 8

installation via compositeur (github) 8

Chapitre 3: Introduction à Zend Expressive 9

Examples 9

Un simple Hello World 9

Crédits 12

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: zend-framework2

It is an unofficial and free zend-framework2 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official zend-framework2.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec zend-framework2

Remarques

Zend Framework 2 (ZF2) est un framework PHP moderne et flexible qui aide les développeurs Web à créer des applications Web de différentes complexités. Le principal sponsor de la société Zend Framework est Zend Technologies , ce qui le rend très solide et stable. Il y a deux améliorations majeures de cette deuxième version sur ZF1. Tout d'abord, une architecture basée sur des modules a été adoptée par défaut sans aucune modification. Cela s'avère pratique lors du développement d'une application Web de grande taille nécessitant une décomposition en modules. Deuxièmement, ZF2 implémente toutes les fonctionnalités que PHP5.3 + peut offrir, en particulier les espaces de noms. Dans les versions précédentes, une classe de contrôleur est nommée comme suit:

```
class IndexController extends Zend_Controller_Action
{

}
```

Cette même classe est réécrite dans ZF2 comme suit:

```
namespace Application\Controller;
use Zend\Mvc\Controller\AbstractActionController;

class IndexController extends AbstractActionController
{

}
```

Voici quelques autres fonctionnalités intéressantes de ZF2:

- Injection de dépendance
- Responsable de l'événement
- Gestionnaire de services

Examples

Installation ou configuration

Instructions détaillées sur la configuration ou l'installation de Zend Framework 2. Il existe différentes manières d'installer le framework. En voici quelques unes:

Utilisation de Composer - Méthode recommandée

En supposant que `composer` est installé sur la boîte cible.

Pour installer une application squelette MVC, exécutez-la dans votre terminal pour créer un nouveau projet zend framework 2 à l'emplacement spécifié:

```
php composer.phar create-project -sdev \
    --repository-url="https://packages.zendframework.com" \
    zendframework/skeleton-application path/to/install
```

pour installer manuellement un ZF2 **minimal** (Zend MVC + sa poignée de dépendances), exécutez-le dans votre ligne de commande:

```
composer require zendframework/zend-mvc
```

ou pour une ZF2 d **full-envol** (+64 modules):

```
composer require zendframework/zendframework`
```

Veuillez noter que la première option exécute un programme d'installation qui vous fournira une application entièrement fonctionnelle avec la structure de répertoires d'application habituelle. D'autres options vous permettront de construire l'application entière à partir de zéro car elle fournit simplement des modules ZF2 sur lesquels s'appuyer.

Utilisation de sous-modules Git

Exécutez la commande ci-dessous pour cloner zf2 et ses dépendances récursivement à partir de Github:

```
git clone git://github.com/zendframework/ZendSkeletonApplication.git --recursive
```

Configuration du serveur HTTP

Une application Web type nécessite un service HTTP qui écoute un port dédié (généralement: 80) pour transmettre les demandes entrantes à l'application, traiter et restituer la sortie (réponse).

> Remarque: Vous pouvez également écrire des applications compatibles avec la console dans Zend Framework 2 sans recourir à un serveur HTTP.

OPTION 1 - Serveur PHP CLI

Le moyen le plus simple de commencer si vous utilisez PHP 5.4 ou supérieur est de démarrer le serveur PHP interne cli dans le répertoire racine.

Allez dans le répertoire du projet et lancez:

```
php -S 0.0.0.0:8080 -t public/ public/index.php`.
```

Cela démarrera le serveur CLI intégré sur le port `8080` et le liera à toutes les interfaces réseau.

OPTION 2 - Un serveur HTTP personnalisé

Configurez un hôte virtuel sur Apache *ou* Microsoft IIS Server *ou* Nginx et transmettez les requêtes HTTP entrantes à l'application.

Un simple Hello World

Dans votre ligne de commande, entrez le répertoire dans `composer create-project zendframework/skeleton-application helloWorldTest` vous voulez créer le projet, puis tapez: `composer create-project zendframework/skeleton-application helloWorldTest` . Lors de l'installation, il vous sera demandé si vous souhaitez une installation minimale: disons oui pour le moment, nous ne faisons que tester.

Pour simplifier, nous utiliserons le serveur PHP CLI intégré. Depuis la ligne de commande, placez-vous dans le répertoire racine de votre projet (`helloWorldTest`), puis exécutez: `php -S 0.0.0.0:8080 -t public/ public/index.php` . Maintenant, ouvrez votre navigateur Web et accédez à http: // localhost / , vous devriez voir la page d'accueil de l'application ZF2 Skeleton.

Si vous le faites, nous allons maintenant configurer une nouvelle page. Dans `module/Application/config/module.config.php` vous pouvez voir qu'une route dynamique est déjà configurée pour le sous-dossier de l'application:

```
return [
    'router' => [
        'routes' => [
            'home' => [
                ...
            ],
            'application' => [
                'type'    => Segment::class,
                'options' => [
                    'route'    => '/application[/:action]',
                    'defaults' => [
                        'controller'    => Controller\IndexController::class,
                        'action'        => 'index',
                    ],
                ],
            ],
        ],
    ],
```

Définissez une nouvelle action " `helloWorldAction()` " dans le `module/Applicaiton/src/Controller/IndexController.php` :

```
class IndexController extends AbstractActionController
{
    public function indexAction()
    {
        ...
    }
```

```
    public function helloWorldAction()
    {
        return new ViewModel();
    }
}
```

Enfin, créez le `module/Application/view/application/index/hello-world.phtml` fichier de vue `module/Application/view/application/index/hello-world.phtml` avec le contenu suivant:

```
<?php
echo "Hello World !";
```

Maintenant, allez à http: // localhost / application / hello-world , et dites bonjour à ZF2!

Comment créer une usine

Lorsqu'une classe doit être dotée de dépendances matérielles, la meilleure pratique consiste à utiliser un modèle d'injection de constructeur où ces dépendances sont injectées à l'aide d'une fabrique.

Supposons que `MyClass` dépend difficilement d'une dépendance de valeur `$dependency` qui doit être résolue à partir de la configuration de l'application.

```
<?php
namespace Application\Folder;

use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class MyClass
{
    protected $dependency;

    public function __construct($dependency)
    {
        $this->dependency = $dependency;
    }
}
```

Pour injecter cette dépendance, une classe de fabrique est créée. Cette fabrique va résoudre la dépendance de la configuration et injecter la valeur de configuration lors de la construction de la classe et renvoyer le résultat:

```
<?php
namespace Application\Factory;

use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class MyClassFactory implements FactoryInterface
{
    public function createService(ServiceLocatorInterface $serviceLocator)
    {
```

```
        $config = $servicelocator->get('Config');
        $dependency = $config['dependency'];
        $myClass = new MyClass($dependency);
        return $myClass;
    }
}
```

Maintenant que la classe de fabrique a été créée, elle doit être enregistrée dans la configuration du gestionnaire de services dans le fichier de configuration du module `module.config.php` sous les fabriques de clés. Il est recommandé d'utiliser les mêmes noms pour la classe et l'usine, il est donc facile de les trouver dans l'arborescence du dossier du projet:

```
<?php

namespace Application;

return array(
    //...
    'service_manager' => [
        'factories' => [
            'Application\Folder\MyClass' => 'Application\Factory\MyClassFactory'
        ]
    ],
    //...
);
```

Les constantes du nom de classe peuvent également être utilisées pour les enregistrer:

```
<?php

namespace Application;

use Application\Folder\MyClass;
use Application\Factory\MyClassFactory;

return array(
    //...
    'service_manager' => [
        'factories' => [
            MyClass::class => MyClassFactory::class'
        ]
    ],
    //...
);
```

Maintenant, la classe peut être collectée chez le gestionnaire de services à l'aide de la clé utilisée lors de l'enregistrement de l'usine pour cette classe:

```
 $serviceManager->get('Application\Folder\MyClass');
```

ou

```
 $serviceManager->get(MyClass::class);
```

Le gestionnaire de services trouvera, collectera et exécutera la fabrique, puis retournera votre instance de classe avec la dépendance injectée.

Lire Démarrer avec zend-framework2 en ligne: https://riptutorial.com/fr/zend-framework2/topic/1304/demarrer-avec-zend-framework2

Chapitre 2: Installation

Examples

installation via compositeur (github)

```
1 cd my/project/dir
2 git clone git://github.com/zendframework/ZendSkeletonApplication.git
3 cd ZendSkeletonApplication
4 php composer.phar self-update
5 php composer.phar install
```

Lire Installation en ligne: https://riptutorial.com/fr/zend-framework2/topic/6458/installation

Chapitre 3: Introduction à Zend Expressive

Examples

Un simple Hello World

En utilisant composer, exécutez la commande suivante dans le répertoire dans lequel l'application sera installée: `composer create-project zendframework/zend-expressive-skeleton expressive-skeleton` .

Au cours du processus d'installation, vous serez invité à prendre différentes décisions.

1. Pour la question d'installation par défaut, dites non (`n`);
2. Pour le routeur, utilisons Aura Router (# `1`);
3. Pour le conteneur, utilisons Zend ServiceManager (# `3`);
4. Pour le modèle, utilisons Zend View (# `3`);
5. Enfin, pour le gestionnaire d'erreurs, utilisons Whoops (# `1`).

Une fois installé, installez-vous dans le répertoire racine, `expressive-skeleton` , lancez le serveur PHP CLI intégré: `php -S 0.0.0.0:8080 -t public public/index.php` . Accédez à http: // localhost: 8080 / avec votre navigateur, votre application devrait maintenant être opérationnelle.

Configurons un nouveau chemin vers un nouveau middleware. Tout d'abord, ouvrez le fichier de `config/autoload/routes.global.php` du routeur dans `config/autoload/routes.global.php` et ajoutez les lignes suivantes:

```
<?php

return [
    'dependencies' => [
        ...
    ],

    'routes' => [
        [
            'dependencies' => [
                'invokables' => [
                    ...
                ],
                'factories' => [
                    ...
                    // Add the following line
                    App\Action\HelloWorldAction::class => App\Action\HelloWorldFactory::class,
                ],
            ],
        ],
        // Following lines should be added
        [
            'name' => 'hello-world',
            'path' => '/hello-world',
            'middleware' => App\Action\HelloWorldAction::class,
            'allowed_methods' => ['GET'],
```

```
        ],
    ],
];
```

Placez le contenu suivant dans `src/App/Action/HelloWorldFactory.php` :

```
<?php

namespace App\Action;

use Interop\Container\ContainerInterface;
use Zend\Expressive\Template\TemplateRendererInterface;

class HelloWorldFactory
{
    public function __invoke(ContainerInterface $container)
    {
        $template = ($container->has(TemplateRendererInterface::class))
            ? $container->get(TemplateRendererInterface::class)
            : null;

        return new HelloWorldAction($template);
    }
}
```

Ensuite, ce contenu dans `src/App/Action/HelloWorldAction.php` :

```
<?php

namespace App\Action;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;
use Zend\Diactoros\Response\HtmlResponse;
use Zend\Diactoros\Response\JsonResponse;
use Zend\Expressive\Template;
use Zend\Expressive\Plates\PlatesRenderer;
use Zend\Expressive\Twig\TwigRenderer;
use Zend\Expressive\ZendView\ZendViewRenderer;

class HelloWorldAction
{
    private $template;

    public function __construct(Template\TemplateRendererInterface $template = null)
    {
        $this->template = $template;
    }

    public function __invoke(ServerRequestInterface $request, ResponseInterface $response,
callable $next = null)
    {
        $data = [];

        return new HtmlResponse($this->template->render('app::hello-world'));
    }
}
```

Ensuite, placez simplement les éléments suivants dans `templates/app/hello-world.phtml` :

```
<?php echo 'Hello World'; ?>
```

Nous avons fini ! Accédez à http: // localhost: 8080 / hello-world et dites "salut" à Zend Expressive!

Lire Introduction à Zend Expressive en ligne: https://riptutorial.com/fr/zend-framework2/topic/6109/introduction-a-zend-expressive

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec zend-framework2	Community, edigu, Hassan, Sanjeev kumar, Shirraz, Wilt
2	Installation	edigu, Waqar Haider
3	Introduction à Zend Expressive	Shirraz