EBook Gratuito

APPRENDIMENTO zend-framework2

Free unaffiliated eBook created from
Stack Overflow contributors.

#zend-framework2

Sommario

Di

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capitolo 1: Iniziare con zend-framework2

Osservazioni

Zend Framework 2 (ZF2) è un framework PHP moderno e flessibile che aiuta gli sviluppatori di siti web a creare applicazioni web di diversa complessità. Lo sponsor principale della società Zend Framework è Zend Technologies , che lo rende molto forte e stabile. Ci sono due importanti miglioramenti di questa seconda versione rispetto a ZF1. Innanzitutto, un'architettura basata su moduli è stata adottata di default senza modifiche. Questo è utile quando si sviluppa un'applicazione web di grandi dimensioni che richiede una decomposizione ai moduli. In secondo luogo, ZF2 implementa tutte le funzionalità PHP5.3 + in grado di offrire in particolare gli spazi dei nomi. Nelle versioni precedenti, una classe controller è denominata come segue:

```
class IndexController extends Zend_Controller_Action
{

}
```

Questa stessa classe viene riscritta in ZF2 come segue:

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

class IndexController extends AbstractActionController
{

}
```

Le seguenti sono alcune altre interessanti funzionalità di ZF2:

- Iniezione di dipendenza
- Manager di eventi
- Responsabile del servizio

Examples

Installazione o configurazione

Istruzioni dettagliate su come installare o installare Zend Framework 2. Esistono vari modi per installare il framework. Di seguito sono alcuni di loro:

Uso di Composer - Modo consigliato

Supponendo che il `composer` sia installato nella casella di destinazione.

Per installare un'applicazione skc MVC, esegui nel tuo terminale per creare un nuovo progetto

zend framework 2 nella posizione specificata:

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

per installare manualmente una ZF2 **minima** (Zend MVC + la sua manciata di dipendenze), esegui nella riga di comando:

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

o per un **full-fledge** d ZF2 (+64 moduli):

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

Si noti che la prima opzione esegue un programma di installazione che fornirà un'applicazione completamente funzionante con la consueta struttura delle directory dell'applicazione. Altre opzioni ti permetteranno di costruire l'intera applicazione da zero, in quanto fornisce semplicemente moduli ZF2 su cui costruire.

Utilizzo dei sottomoduli Git

Esegui il comando seguente per clonare zf2 e le sue dipendenze in modo ricorsivo da Github:

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

Configurazione del server HTTP

Una tipica applicazione Web richiede l'esecuzione di un servizio HTTP che ascolta una porta dedicata (in genere: 80) per passare richieste in entrata all'applicazione, elaborare e servire l'output (risposta).

> Nota: è anche possibile scrivere applicazioni compatibili con la console in Zend Framework 2 senza necessità di un server HTTP.

OPZIONE 1 - Server CLI PHP

Il modo più semplice per iniziare se si utilizza PHP 5.4 o successivo è avviare il cli server interno PHP nella directory root.

Vai alla directory del progetto ed esegui:

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

Questo avvierà il cli-server integrato sulla porta `8080` e lo collegherà a tutte le interfacce di rete.

OPZIONE 2: un server HTTP personalizzato

Configura un virtualhost su Apache *o* Microsoft IIS Server *o* Nginx e passa le richieste HTTP in arrivo all'applicazione.

Un semplice Hello World

Nella riga di comando, inserire la directory in cui si desidera creare il progetto, quindi digitare: `composer create-project zendframework/skeleton-application helloWorldTest` . Durante l'installazione, ti verrà chiesto se desideri un'installazione minima: diciamo di sì per il momento, stiamo solo testando.

Per motivi di semplicità, utilizzeremo il server CLI PHP integrato. Dalla riga di comando, mettiti nella directory principale del tuo progetto (`helloWorldTest`), quindi esegui: `php -S 0.0.0.0:8080 -t public/ public/index.php` . Ora apri il browser web e vai a http: // localhost / , dovresti vedere la pagina di benvenuto dell'applicazione ZF2 Skeleton.

Se lo fai, ora installeremo una nuova pagina. In `module/Application/config/module.config.php` puoi vedere che un percorso dinamico è già configurato per la sottocartella dell'applicazione:

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

Imposta una nuova azione " `helloWorldAction()` " in `module/Applicaiton/src/Controller/IndexController.php` :

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

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

```
}
```

Infine, crea il `module/Application/view/application/index/hello-world.phtml` visualizzazione file `module/Application/view/application/index/hello-world.phtml` con il seguente contenuto:

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

Ora vai a http: // localhost / application / hello-world e saluta ZF2!

Come creare una fabbrica

Quando una classe ha bisogno di essere fornita con le dipendenze pesanti, la migliore pratica è quella di utilizzare un modello di iniezione del costruttore in cui tali dipendenze vengono iniettate usando una fabbrica.

Supponiamo che `MyClass` sia molto dipendente da un valore `$dependency` che deve essere risolto dalla configurazione dell'applicazione.

```
<?php
namespace Application\Folder;

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

class MyClass
{
    protected $dependency;

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

Per iniettare questa dipendenza viene creata una classe factory. Questo factory risolverà la dipendenza dalla configurazione e inserirà il valore di configurazione sulla costruzione della classe e restituirà il risultato:

```
<?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;
    }
```

```
}
```

Ora che la classe factory è stata creata, deve essere registrata all'interno della configurazione del gestore servizi nel file di configurazione del modulo `module.config.php` sotto le fabbriche principali. È buona norma usare gli stessi nomi sia per la classe che per la fabbrica, quindi è facile trovarli nell'albero delle cartelle del progetto:

```
<?php

namespace Application;

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

In alternativa, è possibile utilizzare le costanti del nome della classe per registrarle:

```
<?php

namespace Application;

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

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

Ora la classe può essere ritirata presso il gestore del servizio utilizzando la chiave che abbiamo utilizzato al momento della registrazione della factory per quella classe:

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

o

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

Il gestore servizi troverà, raccoglierà ed eseguirà la fabbrica e quindi restituirà l'istanza della classe con la dipendenza iniettata.

Leggi Iniziare con zend-framework2 online: https://riptutorial.com/it/zend-

framework2/topic/1304/iniziare-con-zend-framework2

Capitolo 2: Installazione

Examples

installando tramite compositore (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
```

Leggi Installazione online: https://riptutorial.com/it/zend-framework2/topic/6458/installazione

Capitolo 3: Introduzione a Zend Expressive

Examples

Un semplice Hello World

Usando il compositore, eseguire il seguente comando nella directory in cui verrà installata l'applicazione: `composer create-project zendframework/zend-expressive-skeleton expressive-skeleton` .

Durante il processo di installazione, ti verrà chiesto di prendere varie decisioni.

1. Per la domanda di installazione predefinita, dì no (`n`);
2. Per il router, usiamo Aura Router (# `1`);
3. Per il contenitore, usiamo Zend ServiceManager (# `3`);
4. Per il modello, usiamo Zend View (# `3`);
5. Infine, per il gestore degli errori, usiamo Whoops (# `1`).

Una volta installato, mettiti nella directory root, `expressive-skeleton` , avvia il server PHP CLI integrato: `php -S 0.0.0.0:8080 -t public public/index.php` . Vai a http: // localhost: 8080 / con il tuo browser, la tua applicazione ora dovrebbe essere attiva e funzionante.

Configuriamo un nuovo percorso per un nuovo middleware. Innanzitutto, apri il file di `config/autoload/routes.global.php` del router in `config/autoload/routes.global.php` e aggiungi le linee come segue:

```
<?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'],
```

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

Inserisci il seguente contenuto in `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);
    }
}
```

Quindi, questo contenuto in `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'));
    }
}
```

Quindi, alla fine, metti semplicemente quanto segue in `templates/app/hello-world.phtml` :

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

Abbiamo chiuso ! Vai a http: // localhost: 8080 / hello-world e pronuncia "ciao" a Zend Expressive!

Leggi Introduzione a Zend Expressive online: https://riptutorial.com/it/zend-framework2/topic/6109/introduzione-a-zend-expressive

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con zend-framework2	Community, edigu, Hassan, Sanjeev kumar, Shirraz, Wilt
2	Installazione	edigu, Waqar Haider
3	Introduzione a Zend Expressive	Shirraz