Kostenloses eBook

LERNEN zend-framework2

Free unaffiliated eBook created from
Stack Overflow contributors.

#zend-framework2

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit zend-framework2

Bemerkungen

Zend Framework 2 (ZF2) ist ein modernes und flexibles PHP-Framework, das Webentwicklern dabei hilft, Webanwendungen unterschiedlicher Komplexität zu erstellen. Der Hauptsponsor des Unternehmens Zend Framework ist Zend Technologies , was es sehr stark und stabil macht. Es gibt zwei wichtige Verbesserungen dieser zweiten Version gegenüber ZF1. Erstens wurde eine modulbasierte Architektur standardmäßig ohne Änderung angenommen. Dies ist praktisch, wenn Sie eine große Webanwendung entwickeln, die eine Zerlegung in Module erfordert. Zweitens implementiert ZF2 alle Funktionen, die PHP5.3 + insbesondere für die Namespaces bieten kann. In den vorherigen Versionen wurde eine Controller-Klasse wie folgt benannt:

```
class IndexController extends Zend_Controller_Action
{

}
```

Dieselbe Klasse wird in ZF2 wie folgt umgeschrieben:

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

class IndexController extends AbstractActionController
{

}
```

Nachfolgend einige weitere aufregende Funktionen von ZF2:

- Abhängigkeitsspritze
- Event Manager
- Service Manager

Examples

Installation oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von Zend Framework 2. Es gibt verschiedene Möglichkeiten, das Framework zu installieren. Nachfolgend einige davon:

Composer verwenden - Empfohlene Vorgehensweise

Vorausgesetzt, `composer` ist auf der Zielbox installiert .

Um eine MVC-Skelettanwendung zu installieren, führen Sie in Ihrem Terminal ein neues zend

Framework 2-Projekt an einem bestimmten Ort aus:

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

Um einen **minimalen** ZF2 (Zend MVC + seine Handvoll Abhängigkeiten) manuell zu installieren, führen Sie die Befehlszeile aus:

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

oder für einen **Voll fledge** d ZF2 (+64 Module):

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

Bitte beachten Sie, dass die erste Option ein Installationsprogramm ausführt, das Ihnen eine voll funktionsfähige Anwendung zusammen mit der üblichen Struktur der Anwendungsverzeichnisse zur Verfügung stellt. Mit anderen Optionen können Sie die gesamte Anwendung von Grund auf neu erstellen, da ZF2-Module einfach darauf aufbauen.

Git-Submodule verwenden

Führen Sie den folgenden Befehl aus, um zf2 und seine Abhängigkeiten rekursiv von Github zu klonen:

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

HTTP-Server-Setup

Eine typische Webanwendung erfordert, dass ein HTTP-Dienst ausgeführt wird, der einen dedizierten Port überwacht (normalerweise: 80), um eingehende Anforderungen an die Anwendung zu übergeben, zu verarbeiten und die Ausgabe (Antwort) zurückzugeben.

> Hinweis: In Zend Framework 2 können Sie auch konsolenabhängige Anwendungen schreiben, ohne dass ein HTTP-Server erforderlich ist.

Option 1 - PHP-CLI-Server

Der einfachste Einstieg, wenn Sie PHP 5.4 oder höher verwenden, ist das Starten des internen PHP-Cli-Servers im Stammverzeichnis.

Gehen Sie zum Projektverzeichnis und führen Sie Folgendes aus:

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

Dadurch wird der integrierte CLI-Server an Port `8080` gestartet und an alle Netzwerkschnittstellen gebunden.

OPTION 2 - Ein benutzerdefinierter HTTP-Server

Konfigurieren Sie einen virtuellen Host auf Apache *oder* Microsoft IIS Server *oder* Nginx und leiten Sie eingehende HTTP-Anforderungen an die Anwendung weiter.

Eine einfache Hallo Welt

`composer create-project zendframework/skeleton-application helloWorldTest` Sie in der Befehlszeile das Verzeichnis auf, in dem Sie das Projekt erstellen möchten, und geben Sie dann `composer create-project zendframework/skeleton-application helloWorldTest` : `composer create-project zendframework/skeleton-application helloWorldTest` . Während der Installation werden Sie gefragt, ob Sie eine minimale Installation wünschen: Sagen wir "Ja", wir testen gerade.

Der Einfachheit halber verwenden wir den integrierten PHP-CLI-Server. `helloWorldTest` Sie über die Befehlszeile in das Stammverzeichnis Ihres Projekts (`helloWorldTest`) und führen Sie dann `helloWorldTest` aus: `php -S 0.0.0.0:8080 -t public/ public/index.php` . Öffnen Sie jetzt Ihren Webbrowser und rufen Sie http: // localhost / auf . Die Begrüßungsseite der ZF2-Skeleton-Anwendung sollte angezeigt werden.

Wenn Sie dies tun, richten wir jetzt eine neue Seite ein. In `module/Application/config/module.config.php` Sie sehen, dass bereits eine dynamische Route für den Anwendungsunterordner eingerichtet ist:

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

Setzen Sie eine neue Aktion " `helloWorldAction()` " in `module/Applicaiton/src/Controller/IndexController.php` :

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

```
    {
        ...
    }

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

Erstellen Sie `module/Application/view/application/index/hello-world.phtml` das `module/Application/view/application/index/hello-world.phtml` mit folgendem Inhalt:

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

Gehen Sie jetzt zu http: // localhost / application / hello-world und sagen Sie Hallo zu ZF2!

So erstellen Sie eine Fabrik

Wenn eine Klasse mit harten Abhängigkeiten versehen werden muss, empfiehlt es sich, ein Konstruktor-Einfügungsmuster zu verwenden, bei dem diese Abhängigkeiten mithilfe einer Factory eingefügt werden.

Nehmen wir an, `MyClass` ist stark von einem Wert `$dependency` , der aus der Anwendungskonfiguration gelöst werden muss.

```
<?php
namespace Application\Folder;

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

class MyClass
{
    protected $dependency;

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

Um diese Abhängigkeit einzufügen, wird eine Factory-Klasse erstellt. Diese Factory löst die Abhängigkeit von der Konfiguration auf, fügt den Konfigurationswert bei der Erstellung der Klasse ein und gibt das Ergebnis zurück:

```
<?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;
    }
}
```

Nachdem die Factory-Klasse erstellt wurde, muss sie in der Service Manager-Konfiguration in der Modul-Konfigurationsdatei `module.config.php` unter den Schlüsselfactorys `module.config.php` . Es ist empfehlenswert, sowohl für die Klasse als auch für die Factory dieselben Namen zu verwenden, um sie im Projektordnerbaum leicht zu finden:

```
<?php

namespace Application;

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

Alternativ können die Klassennamenkonstanten verwendet werden, um sie zu registrieren:

```
<?php

namespace Application;

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

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

Jetzt kann die Klasse beim Service Manager mit dem Schlüssel abgeholt werden, den wir bei der Registrierung der Factory für diese Klasse verwendet haben:

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

oder

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

Der Service Manager sucht, sammelt und führt die Factory aus und gibt dann Ihre Klasseninstanz mit der eingefügten Abhängigkeit zurück.

Erste Schritte mit zend-framework2 online lesen: https://riptutorial.com/de/zend-framework2/topic/1304/erste-schritte-mit-zend-framework2

Kapitel 2: Einführung in Zend Expressive

Examples

Eine einfache Hallo Welt

Führen Sie mit composer den folgenden Befehl in dem Verzeichnis aus, in dem die Anwendung installiert wird: `composer create-project zendframework/zend-expressive-skeleton expressive-skeleton` .

Während des Installationsvorgangs werden Sie aufgefordert, verschiedene Entscheidungen zu treffen.

1. Sagen Sie für die Standardinstallationsfrage no (`n`);
2. Als Router verwenden wir den Aura Router (# `1`).
3. Als Container verwenden wir Zend ServiceManager (# `3`).
4. Als Vorlage verwenden wir Zend View (# `3`).
5. Zum Schluss verwenden wir für den Fehlerbehandler Whoops (# `1`).

Nach der Installation gelangen Sie in das Stammverzeichnis, `expressive-skeleton` , und starten Sie den integrierten PHP-CLI-Server: `php -S 0.0.0.0:8080 -t public public/index.php` . Gehen Sie mit Ihrem Browser zu http: // localhost: 8080 / , Ihre Anwendung sollte jetzt betriebsbereit sein.

Lassen Sie uns einen neuen Pfad zu einer neuen Middleware konfigurieren. Öffnen Sie zuerst die Konfigurationsdatei des Routers in `config/autoload/routes.global.php` und fügen Sie die Zeilen wie folgt hinzu:

```
<?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'],
        ],
    ],
];
```

Fügen Sie den folgenden Inhalt 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);
    }
}
```

Dann dieser Inhalt 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'));
    }
}
```

`templates/app/hello-world.phtml` einfach folgendes in `templates/app/hello-world.phtml` :

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

Wir sind fertig ! Navigieren Sie zu http: // localhost: 8080 / Hallo-Welt und sagen Sie "Hallo" zu Zend Expressive!

Einführung in Zend Expressive online lesen: https://riptutorial.com/de/zend-framework2/topic/6109/einfuhrung-in-zend-expressive

Kapitel 3: Installation

Examples

Installation via Composer (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
```

Installation online lesen: https://riptutorial.com/de/zend-framework2/topic/6458/installation

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit zend-framework2	Community, edigu, Hassan, Sanjeev kumar, Shirraz, Wilt
2	Einführung in Zend Expressive	Shirraz
3	Installation	edigu, Waqar Haider