EBook Gratis

APRENDIZAJE zend-framework2

Free unaffiliated eBook created from
Stack Overflow contributors.

#zend-framework2

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con zend-framework2

Observaciones

Zend Framework 2 (ZF2) es un marco de PHP moderno y flexible que ayuda a los desarrolladores web a crear aplicaciones web de diferentes complejidades. El principal patrocinador de la empresa Zend Framework es Zend Technologies , lo que lo hace muy fuerte y estable. Hay dos mejoras importantes de esta segunda versión sobre ZF1. Primero, una arquitectura basada en módulos ha sido adoptada por defecto sin ningún tipo de ajuste. Esto resulta útil cuando se desarrolla una aplicación web de gran tamaño que requiere una descomposición de los módulos. Segundo, ZF2 implementa todas las características que PHP5.3 + puede ofrecer particularmente los espacios de nombres. En las versiones anteriores, una clase de controlador se denomina como sigue:

```
class IndexController extends Zend_Controller_Action
{

}
```

Esta misma clase se reescribe en ZF2 de la siguiente manera:

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

class IndexController extends AbstractActionController
{

}
```

Las siguientes son algunas otras características interesantes de ZF2:

- Inyección de dependencia
- Administrador de evento
- Gerente de Servicio

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar Zend Framework 2. Hay varias formas de instalar el framework. A continuación hay algunos de ellos:

Usando Composer - forma recomendada

Suponiendo que el `composer` está instalado en el cuadro de destino.

Para instalar una aplicación MVC esquelética, ejecute en su terminal para crear un nuevo proyecto de zend framework 2 en una ubicación específica:

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

para instalar manualmente un ZF2 **mínimo** (Zend MVC + sus pocas dependencias), ejecute en su línea de comando:

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

o para un d zf2 **full-** fledge (+64 módulos):

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

Tenga en cuenta que la primera opción ejecuta un instalador que le proporcionará una aplicación totalmente funcional junto con la estructura de directorios de la aplicación habitual. Otras opciones le permitirán compilar toda la aplicación desde cero, ya que simplemente proporciona módulos ZF2 sobre los que construir.

Usando Submódulos Git

Ejecute el siguiente comando para clonar zf2 y sus dependencias recursivamente desde Github:

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

Configuración del servidor HTTP

Una aplicación web típica requiere que se ejecute un servicio HTTP que escuche un puerto dedicado (generalmente: 80) para pasar las solicitudes entrantes a la aplicación, procesar y devolver la salida (respuesta).

> Nota: también puede escribir aplicaciones compatibles con la consola en Zend Framework 2 sin necesidad de un servidor HTTP.

OPCIÓN 1 - PHP CLI Server

La forma más sencilla de comenzar si está utilizando PHP 5.4 o superior es iniciar el servidor interno de PHP en el directorio raíz.

Ir al directorio del proyecto y ejecutar:

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

Esto iniciará el cli-server incorporado en el puerto `8080` y lo vinculará a todas las interfaces de red.

OPCIÓN 2 - Un servidor HTTP personalizado

Configure un host virtual en Apache *o* Microsoft IIS Server *o* Nginx y pase las solicitudes HTTP entrantes a la aplicación.

Un simple hola mundo

En su línea de comandos, ingrese en el directorio en el que desea crear el proyecto, luego escriba: `composer create-project zendframework/skeleton-application helloWorldTest` . Durante la instalación, se le preguntará si desea una instalación mínima: Digamos que sí por el momento, solo estamos probando.

Para simplificar, utilizaremos el servidor CLI de PHP incorporado. Desde la línea de comandos, `helloWorldTest` directorio raíz de su proyecto (`helloWorldTest`), luego ejecute: `php -S 0.0.0.0:8080 -t public/ public/index.php` . Ahora, abra su navegador web y vaya a http: // localhost / , debería ver la página de bienvenida de la aplicación ZF2 Skeleton.

Si lo haces, ahora configuraremos una nueva página. En el `module/Application/config/module.config.php` puede ver que una ruta dinámica ya está configurada para la subcarpeta de la aplicación:

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

Establezca una nueva acción " `helloWorldAction()` " en el `module/Applicaiton/src/Controller/IndexController.php` :

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

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

Finalmente, cree el `module/Application/view/application/index/hello-world.phtml` archivo de vista `module/Application/view/application/index/hello-world.phtml` con el siguiente contenido:

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

Ahora, vaya a http: // localhost / application / hello-world , y salude a ZF2.

Cómo crear una fábrica

Cuando a una clase se le debe proporcionar una dependencia sólida, la mejor práctica es usar un patrón de inyección de constructor en el que esas dependencias se inyecten utilizando una fábrica.

Supongamos que `MyClass` es muy dependiente de una `$dependency` valor que debe resolverse desde la configuración de la aplicación.

```
<?php
namespace Application\Folder;

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

class MyClass
{
    protected $dependency;

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

Para inyectar esta dependencia se crea una clase de fábrica. Esta fábrica resolverá la dependencia de la configuración e inyectará el valor de configuración en la construcción de la clase y devolverá el resultado:

```
<?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;
    }
}
```

Ahora que se ha creado la clase de fábrica, debe registrarse dentro de la configuración del administrador de servicios en el archivo de configuración del módulo `module.config.php` en las fábricas clave. Es una buena práctica usar los mismos nombres tanto para la clase como para la fábrica, por lo que es fácil encontrarlos en el árbol de carpetas del proyecto:

```
<?php

namespace Application;

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

Alternativamente, las constantes de nombre de clase pueden usarse para registrarlas:

```
<?php

namespace Application;

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

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

Ahora la clase se puede recopilar en el administrador de servicios usando la clave que usamos al registrar la fábrica para esa clase:

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

o

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

El administrador de servicios encontrará, recopilará y ejecutará la fábrica y luego devolverá su

instancia de clase con la dependencia inyectada.

Lea Empezando con zend-framework2 en línea: https://riptutorial.com/es/zend-framework2/topic/1304/empezando-con-zend-framework2

Capítulo 2: Instalación

Examples

instalando vía compositor (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
```

Lea Instalación en línea: https://riptutorial.com/es/zend-framework2/topic/6458/instalacion

Capítulo 3: Introducción a Zend Expressive

Examples

Un simple hola mundo

Con Composer, ejecute el siguiente comando en el directorio en el que se instalará la aplicación: `composer create-project zendframework/zend-expressive-skeleton expressive-skeleton` .

Durante el proceso de instalación, se le pedirá que tome varias decisiones.

1. Para la pregunta de instalación predeterminada, diga no (`n`);
2. Para el enrutador, usemos el enrutador Aura (# `1`);
3. Para el contenedor, vamos a usar Zend ServiceManager (# `3`);
4. Para la plantilla, usemos Zend View (# `3`);
5. Finalmente, para el manejador de errores, usemos Whoops (# `1`).

Una vez instalado, ingrese al directorio raíz, `expressive-skeleton` , inicie el servidor CLI de PHP incorporado: `php -S 0.0.0.0:8080 -t public public/index.php` . Vaya a http: // localhost: 8080 / con su navegador, su aplicación ahora debería estar en funcionamiento.

Vamos a configurar una nueva ruta a un nuevo middleware. Primero, abra el archivo de configuración del enrutador en `config/autoload/routes.global.php` y agregue las líneas de la siguiente manera:

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

```
    ],
];
```

Ponga el siguiente contenido en `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);
    }
}
```

Luego, este contenido en `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'));
    }
}
```

Luego, finalmente, simplemente coloque lo siguiente en `templates/app/hello-world.phtml` :

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

Hemos terminado ! Vaya a http: // localhost: 8080 / hello-world , y diga "hola" a Zend Expressive!

Lea Introducción a Zend Expressive en línea: https://riptutorial.com/es/zend-framework2/topic/6109/introduccion-a-zend-expressive

Creditos

S. No	Capítulos	Contributors
1	Empezando con zend-framework2	Community, edigu, Hassan, Sanjeev kumar, Shirraz, Wilt
2	Instalación	edigu, Waqar Haider
3	Introducción a Zend Expressive	Shirraz