Manual de ZWF

Documentación de ZWF

- Introducción
- Instalación
- Configuración
- Actualización
- Estructura

Introducción

ZERFREX™ Web Framework, en adelante ZWF, es un framework muy ligero escrito en PHP. Proporciona los servicios básicos de enrutamiento y vistas y una estructura de aplicación.

Existe un número de frameworks masivos y universales con una ingente cantidad de funcionalidad, opciones de configuración, clases y servicios. Dichos proyectos son maduros, estables y libres. ZWF no pretende formar parte de dicho grupo ni ser reemplazo de ninguno de ellos.

El objetivo de ZWF es proporcionar:

- Una base MVC muy reducida.
- Soporte completo y automático de Unicode.
- Acceso a PostgreSQL y MySQL básico y sencillo.
- Gestión automática de idiomas y URLs.
- Facilidad de integración con otros proyectos.

Instalación

Requisitos

Los siguientes requisitos definen el entorno donde el funcionamiento de ZWF está probado.

- Sistema operativo CentOS 6, Debian 7, Debian 8, Ubuntu LTS 12.04, Fedora 19/20.
- Servidor Apache 2 (`apache2` en caso de Debian/Ubuntu; `httpd` en caso de CentOS/Fedora).
- PHP 5.3 instalado como módulo de Apache.
- Extensión de PHP: `pgsql` si se va usar las funciones de ZWF de acceso a BD con este motor.
- Extensión de PHP: `mysqli` si se va a usar las funciones de ZWF de acceso a BD con este motor.
- Extensión de PHP: `mbstring`.
- Opcional: base de datos PostgreSQL 8.4 o superior o bien MySQL 5.1 o superior.

Incompatibilidades

En algunos casos el módulo `mod_negotiation`, en concreto la funcionalidad `MultiViews`, puede provocar problemas si ZWF va a coexistir con otros ficheros en el mismo directorio y esos ficheros tienen el mismo nombre que algún controlador de ZWF. En este caso puede ser necesario añadir

```
Options -MultiViews
```

al fichero de configuración de Apache en la sección `<VirtualHost>` o análoga correspondiente donde vaya a estar alojado ZWF.

Instalación

Descomprimir y copiar en su caso **el contenido del directorio src** del paquete ZWF en el directorio raíz del sitio web donde se va a ejecutar la aplicación. Este directorio suele ser el establecido por `DocumentRoot` en la configuración de Apache, pero también puede ser un subdirectorio si es que la aplicación se alojará allí.

ZWF no se ejecuta desde ninguna subcarpeta. Si al descomprimirlo se creó alguna carpeta (muchos descompresores actúan así), es el contenido de dicha carpeta el que debe ser copiado al

directorio raíz del sitio web.

El resultado final será que tanto `index.php` como `.htaccess` quedarán en el directorio raíz y, a ese mismo nivel, el resto de directorios.

El fichero `.htaccess` es un fichero oculto. A la hora de copiarlo al directorio raíz del sitio web, podría ser omitido por el programa que hace la copia (clientes FTP, SFTP, exploradores de archivos, etc). Hay que asegurarse de que se copia correctamente.

Configuración de Apache

PHP

El módulo PHP para Apache es necesario que esté instalado y activo. Se requiere la versión 5.3 ó superior. El módulo debe estar compilado, o bien el paquete instalado, con la extensión **mbstring** (Multi-Byte String) y, si se va a usar PostgreSQL usando los módulos de acceso a datos de ZWF, la extensión **pgsql**. Análogamente, en caso de usar MySQL, instalar la extensión **mysqli**.

Rewrite

ZWF usa directivas del módulo `mod_rewrite` de Apache. Están definidas en el fichero `.htaccess`. Por lo tanto es necesario que `mod_rewrite` esté instalado y activo en Apache. En la sección `<VirtualHost>` o análoga de Apache es necesario al menos la siguiente directiva:

```
AllowOverride FileInfo
```

También funcionará:

```
AllowOverride All
```

Esto es importante, porque a veces se tiene lo siguiente:

```
AllowOverride None
```

y en tal caso las directivas de `mod_rewrite` que hay en `.htaccess` no funcionarán y, por lo tanto, ZWF tampoco.

Permisos de los ficheros

No se requieren permisos especiales. Como toda aplicación web, puede ser conveniente:

- Crear un directorio con permisos de lectura y escritura para guardar logs, ficheros subidos, etc.
- Los ficheros de configuración pueden tener contraseñas. En el caso de ZWF, estos ficheros se encuentran en el directorio `cfg/`. Estos ficheros deberían tener permisos únicamente de lectura y ser accesibles solo por el usuario (y grupo, dependiendo de la configuración que usemos) de Apache.

Problemas habituales

Si se encuentra un error *file not found* o *index not found*, asegúrese de que el fichero `.htaccess` existe y es el que venía con el paquete ZWF.

Si el error es *access denied*, compruebe los permisos y propietarios de los ficheros. La instalación de Apache en Linux suele contener algún fichero de muestra `index.html` con los permisos y propietarios correctos.

El módulo de Apache `mod_rewrite` debe estar activo. Normalmente al activarlo es necesario reiniciar el servidor (`apache2` o `httpd`).

Configuración

La configuración se realiza creando ficheros de código PHP en el directorio `cfg/` con código similar a:

```
$cfg['opcion1'] = 'valor1';
$cfg['opcion2'] = 'valor2';
$cfg['opcion3'] = 'valor3';
// ...
$cfg['opcionN'] = 'valorN';
```

Siendo `opcion1`, `opcion2`, ... claves de configuración y `valor1`, `valor2`, ... sus valores. El nombre del array `cfg` es necesario.

Mediante este sistema se puede cambiar la configuración predeterminada (como se verá más adelante) o crear configuración propia según las necesidades de nuestro proyecto.

Por ejemplo: definiremos, para una supuesta aplicación, la clave de configuración `defaultEmailAddr` :

```
$cfg['defaultEmailAddr'] = 'webmaster@mydomain.com';
```

En cualquier punto de nuestra aplicación podemos recuperar esta clave usando la función `\zfx\Config::get()`. Ejemplo:

```
function contactFormSent($text)
{
    //...
    $to = \zfx\Config::get('defaultEmailAddr');
    writeMail($to, $text);
    // ...
}
```

Al ser los ficheros de configuración ficheros de código PHP se puede elaborar esquemas de configuración sofisticados. Por ejemplo:

```
$cfg['max_usuarios'] = 100;
$cfg['max_grupos'] = $cfg['max_usuarios'] / 2;
if ($cfg['max_grupos'] > 25) $cfg['max_grupos'] = 25;
```

Configuración predeterminada

Cada **módulo** de ZWF puede tener un fichero de configuración. Es lo que se llama **la configuración predeterminada del módulo**. Estos ficheros no están en `cfg/`, sino en el directorio de cada módulo.

El fichero de configuración del módulo **core** es `base/zfx/core/core-config.php`.

El fichero de configuración de **cualquier otro módulo siempre se llama** `module-config.php` y *está en su directorio correspondiente*. Todos los ficheros de configuración predeterminados de los módulos considerados activos son cargados al inicio.

Cambiar la configuración. Orden de carga

Para configurar la aplicación, sobreescribiendo los valores predeterminados, es necesario colocar en el directorio `cfg/` ficheros de configuración con los nuevos valores. Sin embargo, es necesario conocer el orden de carga y los nombres de fichero a utilizar para tener un control preciso del sistema.

En cualquier petición, la configuración es cargada en este orden:

1. Se lee la configuración predeterminada del módulo core, o sea, `base/zfx/core/core-config.php`.
2. Se lee la configuración personalizada por el usuario para el módulo core, si es que existe, o sea: `cfg/core-config.php`
   En dicho fichero tenemos la oportunidad de activar/desactivar módulos y también de especificar *una lista personalizada de ficheros de configuración.*
3. Se lee secuencialmente la configuración de los módulos activos. Por defecto todos los módulos (dev, data-access) están activos, así que se leerían los siguientes ficheros:

   ```
   base/zfx/dev/module-config.php
   base/zfx/data-access/module-config.php
   ```

4. Si se especificó la lista personalizada de ficheros de configuración en el paso 2, entonces se cargarán secuencialmente. Podemos aprovechar en este punto para sobreescribir la configuración predeterminada de los módulos que se cargó en el paso 3.

Esquema de configuración habitual

El esquema más sencillo pero completamente funcional es establecer dos ficheros de configuración: `cfg/core-config.php`, ya que siempre se intenta cargar este fichero, y otro (que llamaremos `cfg/my-config.php`) para configurar el resto de módulos (por ejemplo el acceso a la base de datos) y cualquier necesidad de nuestra aplicación.

El fichero `cfg/core-config.php` debería tener al menos el siguiente contenido:

```
$cfg['rootUrl']        = 'http://www.miaplicacion.com/';
$cfg['showErrors']     = false;
$cfg['autoLoadConfig'] = array('my-config');
```

La clave `rootUrl`

ZWF necesita saber su propia URL. Se especifica en la clave `rootUrl`.

Por ejemplo, supongamos que hemos instalado ZWF en el directorio `/var/www/test` de nuestro servidor y es accesible en la URL `http://www.example.com/test`.

Entonces en el fichero `cfg/core-config.php` introduciremos lo siguiente:

```
$cfg['rootUrl'] = 'http://www.example.com/test/';
```

Nótese la barra al final de la URL. **Por convenio, todas las URLs que se configuren en ZWF terminan con la barra `/`.**

También se puede especificar una ruta relativa; los navegadores web la suelen interpretar correctamente:

```
$cfg['rootUrl'] = '/test/';
```

Supongamos que tenemos una intranet y queremos acceder por una determinada IP y puerto:

```
$cfg['rootUrl'] = 'http://192.168.1.23:8080/testapp/';
```

Nótese la barra siempre al final de la URL.

La clave `showErrors`

Indica si se deben mostrar, o no, los errores de PHP. A menudo se combina con la detección automática del directorio o dirección del servidor para saber si estamos en producción o no. Ejemplo:

```
if ($_SERVER['DOCUMENT_ROOT'] == '/var/www/html')
{
    // Estamos en producción
    $cfg['rootUrl']    = 'http://www.miaplicacion.com/';
    $cfg['showErrors'] = false;
}
else if ($_SERVER['DOCUMENT_ROOT'] == '/var/www/test')
{
    // Estamos en desarrollo
    $cfg['rootUrl']    = 'http://dev.miempresa.com/';
    $cfg['showErrors'] = true;
}
```

La clave `autoLoadConfig`

Esta clave permite la carga de ficheros adicionales de configuración. En nuestro ejemplo contiene un único elemento cuyo valor es `my-config`.

Actualización

Si se quiere actualizar a una versión superior de ZWF, solo hay que descomprimir y copiar el contenido del directorio src del paquete de distribución de ZWF encima de nuestra instalación la nueva versión, sobreescribiendo los ficheros.

La distribución de ZWF no interfiere con una instalación existente salvo que, obviamente, se hayan modificado los directorios y ubicaciones predeterminadas.

Estructura

La función de los ficheros y directorios instalados con ZWF es la siguiente:

- `base/` Contiene las clases del framework. No es necesario modificarlo.
- `cfg/` Directorio donde colocar los ficheros de configuración.
- `controllers/` Directorio donde colocar los controladores de la aplicación.
- `lib/` Directorio de libre disposición.
- `models/` Directorio donde colocar el modelo de datos de la aplicación.
- `res/` Directorio donde colocar JS, CSS, imágenes y otros recursos públicos. Accesible desde fuera.
- `views/` Directorio donde colocar las vistas de la aplicación.
- `.htaccess` Directivas de Apache 2
- `index.php` Controlador frontal

Módulos y espacios de nombres

Las diferentes clases y ficheros de configuración de ZWF están agrupados por módulos. **Un módulo, pues, es un conjunto de clases opcionalmente unido a un fichero de configuración**.

ZWF en su instalación predeterminada tiene tres módulos:

- El módulo **core** contiene las clases básicas de ZWF.
- El módulo **dev** contiene las clases de ayuda al desarrollo.
- El módulo **data-access** contiene las clases de acceso a la base de datos.

Los módulos, a su vez, siempre se agrupan por **espacios de nombres**. El único espacio de nombres que se usa en ZWF es `zfx` y los tres módulos suministrados están bajo dicho espacio.

El directorio `base/` contiene tantos subdirectorios como espacios de nombres haya disponibles. En la instalación predeterminada solo encontraremos `base/zfx/`.

En el interior de cada directorio correspodiente a un espacio de nombres hay tantos subdirectorios como módulos haya definidos. En la instalación predeterminada de ZWF, dentro de `base/zfx/`, encontraremos los directorios correspondientes a los tres módulos disponibles:

```
base/zfx/core/
base/zfx/dev/
base/zfx/data-access/
```

Todas las clases de un módulo están definidas bajo el espacio de nombres al que pertenece dicho módulo. Cuando vayamos a invocar una clase de ZWF, en la mayoría de los casos, será necesario anteponer el espacio de nombres o usar `use`.

Es posible añadir nuevos espacios de nombres y módulos en función de las necesidades del proyecto.

Los módulos son fácimente desactivables. Un módulo desactivado es ignorado por el sistema y no se carga su configuración ni se incluye en el sistema de autocarga de clases. El módulo **core** no se puede desactivar y su carga es obligatoria.

CSS, JavaScript y otros recursos

El directorio `res/` está configurado en `.htaccess` para ser accesible públicamente. Se recomienda que el CSS, JavaScript, imágenes, vídeos, etc. de la aplicación se coloquen bajo dicho directorio.

Biblioteca de clases

A continuación se describen algunas clases suministradas con ZWF y su función. La funcionalidad se ha reducido al mínimo, siguiendo el principio de implementar sólo aquello que realmente se necesita.

La biblioteca de ZWF está documentada siguiendo el convenio PHPDocumentor y por lo tanto se puede extraer la documentación del API mediante este método.

Módulo core

- La clase `HtmlTools` contiene métodos estáticos que facilitan la creación de elementos HTML como tablas o `<select>`
- La clase `Paginator` gestiona el control y generado de código HTML para dibujar un paginador. Es altamente personalizable.
- La clase `StrFilter` contiene numersos métodos de ayuda para operar con cadenas UTF-8. **Es muy recomendable su uso.**
- La clase `StrValidator` contiene métodos de validación de tipos de datos comunes. Algunos son un stub, como el del email, que es elemental y necesitaría ser desarrollado en profundidad para cumplir con el RFC.

- La clases `Mat` y `Sys` son stubs con la intención de ser desarrolladas en un futuro. Actualmente contienen un método estático para el redondeo de euros a dos decimales y un método para obtener una ruta local respectivamente.

Módulo data-access

- La clase `DataTools` contiene herramientas útiles para el trabajo con la BD.

Biblioteca de terceros

Además de los controladores y del modelo de datos, la aplicaciones normalmente hacen uso de otras clases. Se recomienda usar el directorio `lib/` para alojar dichas clases.

Dicho directorio está bajo el sistema de carga automática de clases. Para que funcione la clase y el fichero deben tener el mismo nombre.