**Documentación Cubic**

**404 page**

Se cambió la página 404 de CI por la de Bootstrap/HTML5Boilerplate.

Para ello se ha implementado lo siguiente:

**Archivos Modificados**

**error\_404.php en ./views/errors/html/error\_404.php**

Es la página que aparecerá en caso de no poder resolverse la URI ingresada por el usuario.

**Assets folder**

Una carpeta que contiene los archivos css, js, etc.

Para ello se ha implementado lo siguiente:

**Archivos Nuevos**

**MY\_url\_helper.php en ./application/helpers/MY\_url\_helper.php**

Contendrá el conjunto de funciones añadidas al url\_helper que viene en el núcleo de CI.

**Archivos Modificados**

**autoload.php en ./application/config/autoload.php**

Se agregó el ítem ‘url’ dentro del array $autoload['helper'].

**config.php en ./application/config/config.php**

Se agregó la variable:

$config['assets\_url'] = $config['base\_url'] . 'assets/';

**Controlador general**

Un controlador hecho a medida para no modificar el controlador del núcleo (core).

Para ello se ha implementado lo siguiente:

**Archivos Nuevos**

**MY\_Controller.php en ./application/core/MY\_Controller.php**

Contendrá el conjunto de funciones que todos los controladores usarán ya que lo heredarán de este.

**Google Analytics**

Para ello se ha implementado lo siguiente:

**Archivos Nuevos**

**front\_google\_analytics.php en ./application/views/front\_google\_analytics.php**

Contendrá el script de Google Analytics.

**Archivos Modificados**

**front\_index.php en ./application/views/front\_index.php**

Se imprimirá el contenido del view que contiene el script de Google Analytics usando el código:

<?php echo $google\_analytics; ?>

**site.php en ./application/controllers/site.php**

Ante de llamar a un view que vaya a contener el script de Google Analytics se tendrá que llamar al view “front\_google\_analytics.php” con el siguiente código modelo:

$this->data['google\_analytics'] = $this->load->view(

$this->get\_view\_name( $this->prefix, 'google\_analytics' )

, NULL

, TRUE

);

**Index.php (No usar)**

Modificaciones hechas para no usar “index.php” en la URI.

Para ello se ha implementado lo siguiente:

**Archivos Nuevos**

**.htaccess en ./ .htaccess**

Líneas de código que usan mod\_rewrite (se debe activar en caso de usarse Cubic en local) para evitar el uso de index.php en la URI.

**Archivos Modificados**

**config.php en ./application/config/config.php**

El valor de la variable “index\_page” será ‘’ (vacío).

**Lenguajes**

Cubic acepta más de un lenguaje. Para ello se ha implementado lo siguiente:

**Archivos Nuevos**

**MY\_Lang.php en ./application/core/MY\_Lang.php**

Al tener el prefijo “MY\_”, es cargado automáticamente por CI al momento de cargar el framework.

A la clase principal se le añadió la función “prefered\_language” la cual permite extraer el lenguaje que ha sido configurado como preferido en el navegador del usuario.

**MY\_language\_helper.php en ./application/helpers/MY\_language\_helper.php**

Helper que actualmente contiene solo una función (t) cuyo objetivo es imprimir (echo) la traducción de determinada oración.

El helper se carga desde la función load\_language que se encuentra en MY\_Controller.php

**Archivos Modificados**

**URI.php en ./system/core/URI.php**

Se agregó la función “\_reindex\_segments” ya que es usado en la función principal de “MY\_Lang.php”.

**config.php en ./application/config/config.php**

Se agregaron 3 nuevos elementos al array $config: language\_abbr, lang\_uri\_abbr y lang\_ignore.

Si el valor de lang\_ignore es

TRUE entonces se usará una cookie para almacenar el lenguaje del site.

FALSE entonces se analizará la URI del site (/es/, /en/, etc) para determinar el lenguaje del site.

Por defecto se dejará en TRUE

**routes.php en ./application/config/routes.php**

Se agregaron dos expresiones regulares a routear:

$route['(\w{2})/(.\*)'] = '$2';

Para ignorar el primer parámetro (/es/, /en/, etc) y ejecutar todo lo que está inmediatamente después.

$route['(\w{2})'] = $route['default\_controller'];

Para ejecutar el controlador por defecto en caso de que no haya nada inmediatamente después del primer parámetro (/es/, /en/, etc).

**Mail (envío de email)**

Cubic ha implementado la clase PhpMailer para el envío de email.

Para ello se ha implementado lo siguiente:

**Archivos Nuevos**

**Phpmailer.php en ./application/libraries/Phpmailer.php**

Implementación de Phpmailer.

**Smtp.php en ./application/libraries/Smtp.php**

Implementación de SMTP para usarlo con Phpmailer.

**Archivos Modificados**

**site.php en ./application/controllers/site.php**

Se agregó la función “mail” que configura y envía un email.