# Sitio Web auto administrable- Introducción al Framework

septiembre 11

2016

Integrante: María Fernanda Segovia Chacón

# Tabla de contenido

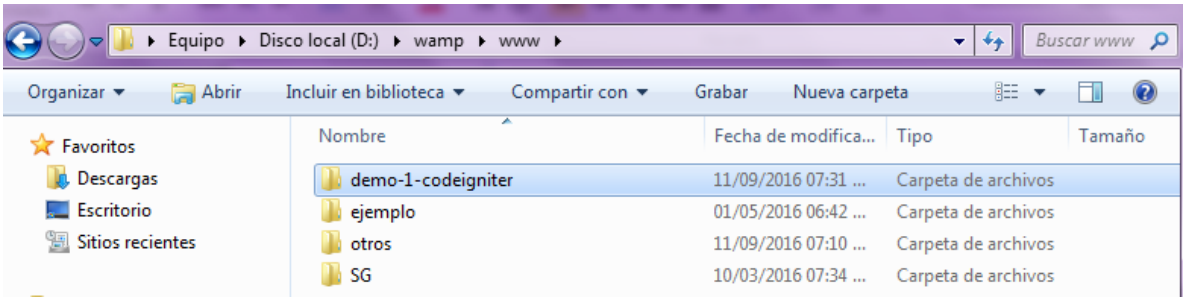
## 1. Requerimientos:

- Contar con un servidor WAMP o XAMP.
- Haber descargado el proyecto de "demo-1-codeigniter"

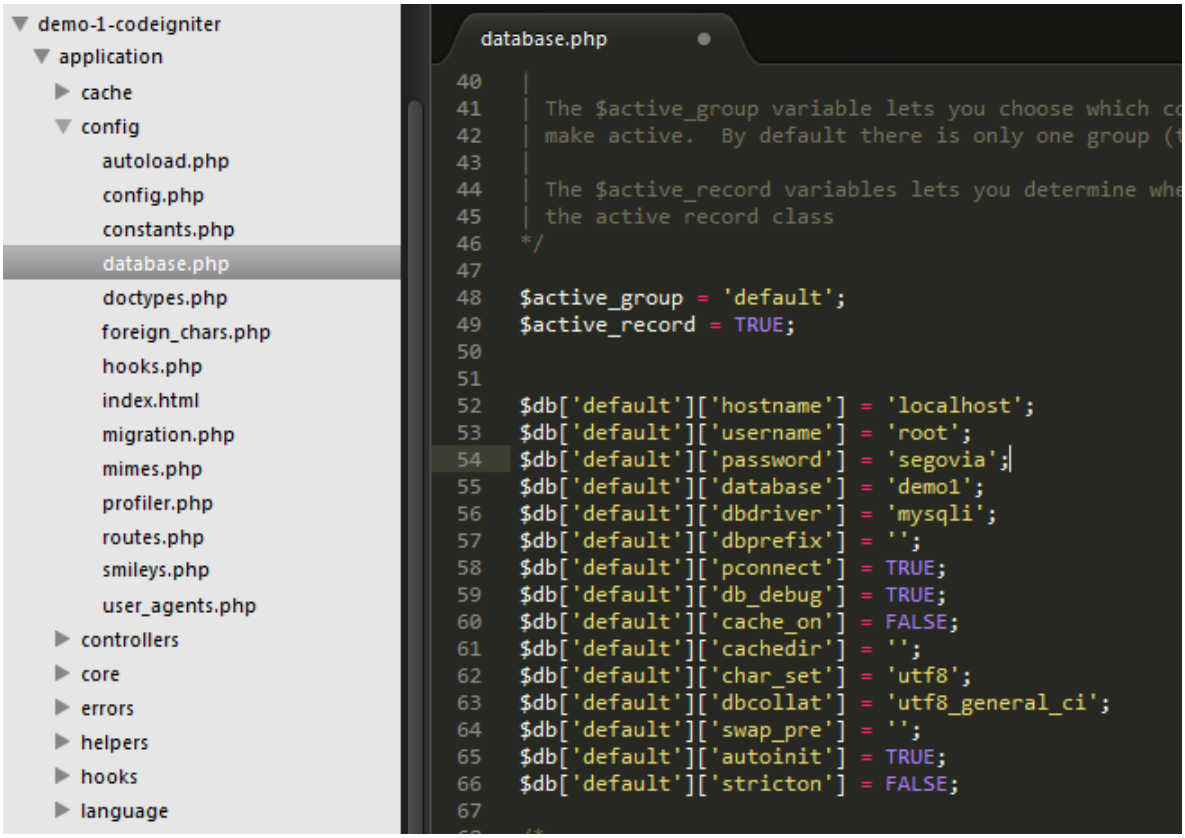
## 2. Desplegar el Proyecto:

1. En los Archivos descargados de "demo-1-codeigniter" existe un archivo llamado demo1.sql, el cual deberán importar a su base de datos Mysql en este caso estoy utilizando phpMyAdmin.

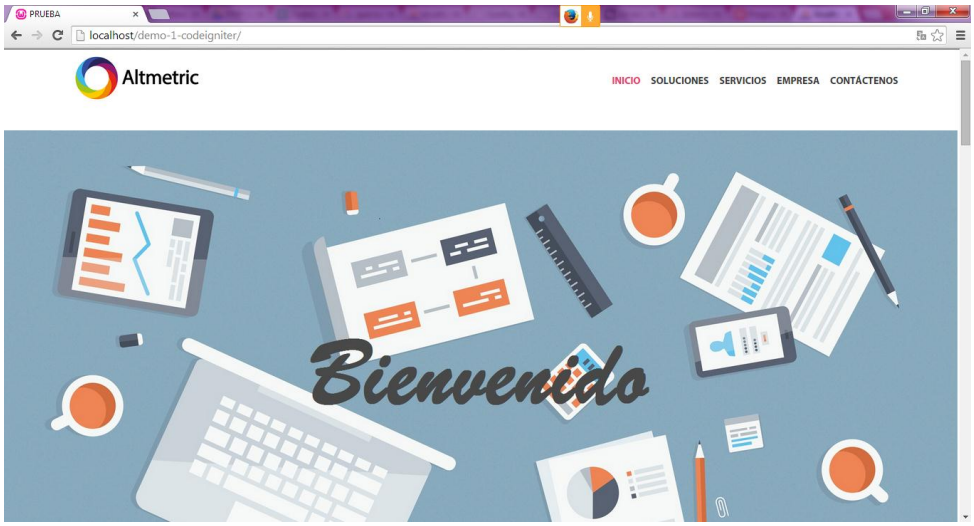

2. El proyecto que se ha descargado se tiene que poner en la cartepa www de wamp server, en mi caso es D:\wamp\www, en xampp la dirección seria c:\xampp\htdocs\ .

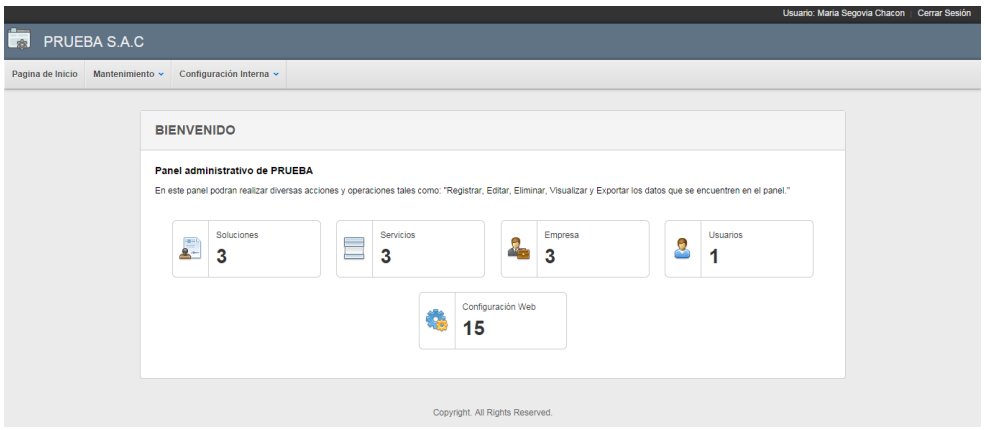



3. Luego de haber incluido la carpeta podemos ingresar al proyecto por medio de sublimetext para cambiar la conexión a la base de datos. Ingresamos a "demo-1-codeigniter/application/config/database.php" y aquí cambiaremos los parámetros de $db['default']['username'] = 'root', $db['default']['password'] = '', $db['default']['database'] = ' ' por el usuario, contraseña y nombre de nuestra base de datos. En mi caso mi conexión a mi base de datos tiene contraseña, de no tenerla dejarla vacía.




4. Luego podemos acceder a la página desde nuestro buscador y podremos ver la página web mediante el link “http://localhost/demo-1-codeigniter/”. Para ingresar al panel administrable ingresaremos a “http://localhost/demo-1-codeigniter/panel” con el usuario: admin y contraseña 123456.





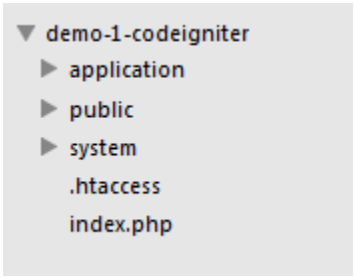

En este proyecto estamos utilizando URL's amigable por lo que puede pasar que les dé un 500 Internal Server Error. Para solucionarlo tendrán que editar el archivo httpd.conf del servidor, en este caso de apache. En WampServer lo encuentras en la carpetaC:\wamp\bin\apache\Apache2.4.4\conf o también clicando sobre el icono del wampserver en la barra de tareas, ir a apache y luego a httpd.conf. Con nuestro editor de texto Buscamos la siguiente linea #LoadModule rewrite_module modules/mod_rewrite.so. Quitamos el comentario "#". Guardamos los cambios. Así es como debería quedar:

LoadModule rewrite_module modules/mod_rewrite.so

Finalmente Reiniciamos el servidor apache.

## 3. MVC CodeIgniter

En este proyecto se creó una parte administrador para la vista principal de una página web para que cada vez que se quiera cambiar alguna imagen o texto no se tenga que modificar alguna parte del código. De esta manera la persona que administra la página no tiene que estar recurriendo al programador cada vez que necesite realizar algún cambio. En el proyecto tenemos la siguiente estructura:



En public se ordenaran todos los archivos de css, js, Font e imágenes que requiera la página. En system se encuentran todas las librerías del Framework. Si se piensa agregar alguna adicional se deberán agregar en application. En Application Se encuentran todos los archivos de nuestro proyecto y es donde trabajaremos. Aquí podemos encontrar las 3 carpetas principales:

- **Models:** Gestiona el acceso o manipulación de la base de datos.
- **Views**: Es la parte visual.
- **Controllers**: Se encarga de hacer de intermediario entre el modelo y la vista. Controla todo lo que pasa en la web, acceso de usuarios, errores
- **Helpers:** Se encuentran aquellas funciones que nos facilitarán la escritura de código. Codeigniter proporciona algunos, como facilitarnos la creación de formularios. Nosotros crearemos otros más adelante para comprenderlos mejor.
- **Hooks:** En esta carpeta alteraríamos el comportamiento que tiene normalmente el framework sin tocar en los archivos los contenidos de la carpeta system. Nosotros lo usaremos para cambiar la seguridad y controlar la entrada de usuarios.
- **Libreries:** La usaremos para la creación de funciones complicadas, que involucren varias tablas o que sencillamente no creamos conveniente alojarla en el modelo. La intención de esto es aislar la lógica de la aplicación con el acceso a la base de datos.
- **Third_party:** Serian las librerías que nosotros nos descargamos para ampliar funcionalidades de PHP. Como podría ser leer PDF o generarlos, envió de correos.
- **Config:** Se encuentras los archivos de configuración.

## 4. Configuración

En la dirección del proyecto “demo-1-codeigniter/application/config” existen varios archivos de configuración que se deben de cambiar cada ves que se inicia un nuevo proyecto.

### A. Config.php

En éste se deberemos configurar la URL base, el nombre con el que por defecto funcionará nuestro sitio sin hacer ningún cambio, pero si no queremos estar cambiando cada rato esta dirección cada que pasemos nuestro proyecto a otro lado hay que sustituir el $config['base_url'] por lo siguiente:

```
$root  = "http://".$_SERVER['HTTP_HOST'];
$root .= str_replace(basename($_SERVER['SCRIPT_NAME']),"",$_SERVER['SCRIPT_NAME']);

$config['base_url'] = $root;
```

Aquí también podemos cambiar nuestra llave de encriptación:

```
$config['encryption_key'] = 's0tf4r3f4act0ry';
```

Y los nombre de nuestras cookies y tokens:

```
$config['csrf_protection'] = TRUE;
$config['csrf_token_name'] = 'csrf_token_softwarefactory';
$config['csrf_cookie_name'] = 'csrf_cookie_token_softwarefactory';
$config['csrf_expire'] = 7200;
```

## B. Routes.php

Lo que ofrece este archivo, sencillamente, es poder seleccionar nuestro controlador por defecto y una página de error por si la carga de la web falla dentro del servidor. Por defecto el controlador que viene es welcome, nosotros lo vamos a cambiar y le ponemos home. En este proyecto además agregamos una ruta para el panel de administración:

```
$route['default_controller'] = "home";
$route['panel']              = 'panel/login';
$route['404_override']       = '';
```

## C. Databases.php

El siguiente archivo es el datebase anteriormente modificado. Aquí modificaremos todo lo referente a la base de datos: tanto el acceso como el tipo que usemos. Aquí ponemos el hostname, el username de nuestra db como la contraseña. Además, se ingresa el nombre de la base de datos a utilizar.

```
$active_group = 'default';
$active_record = TRUE;


$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = 'segovia';
$db['default']['database'] = 'demo1';
$db['default']['dbdriver'] = 'mysqli';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

## D. Helpers.php

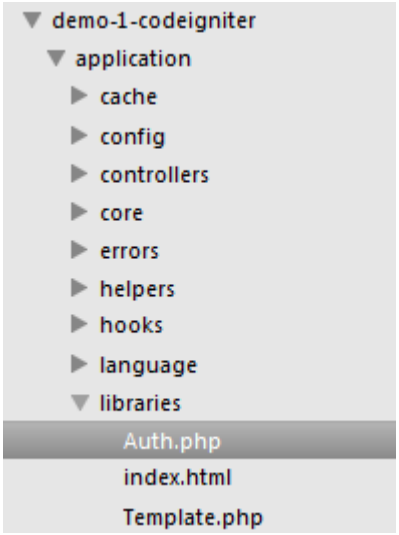
Aquí se pueden agregar helper adicionales para usar en el proyecto. En este proyecto agregamos un helper para la encriptación de las contraseñas.

util_helper.php

```php
<?php
function encrypt_decrypt($action, $string)
{
    $codigo = false;

    $encrypt_method = "AES-256-CBC";
    $secret_key     = '23647586978AFDBGEJRKKDKSK';
    $secret_iv      = 'sjskekedkdkdkdkdkdkdkdkdk';

    $key = hash('sha256', $secret_key);
    $iv  = substr(hash('sha256', $secret_iv), 0, 16);

    if( $action == 'encrypt' ) {
        $codigo = openssl_encrypt($string, $encrypt_method, $key, 0, $iv);
        $codigo = base64_encode($codigo);
    }else if( $action == 'decrypt' ){
        $codigo = openssl_decrypt(base64_decode($string), $encrypt_method, $key, 0, $iv);
    }

    return $codigo;
}
?>
```

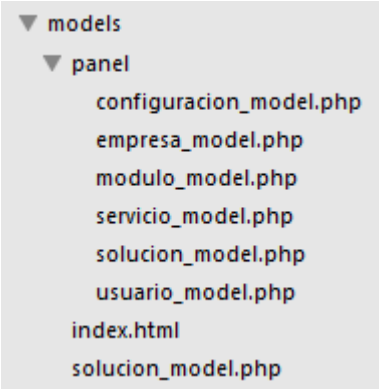
## E. Libraries.php

Aquí se pueden agregar librerias adicionales para usar en el proyecto. En este proyecto agregamos dos clases, una para la autentificación y otra para usar un template.



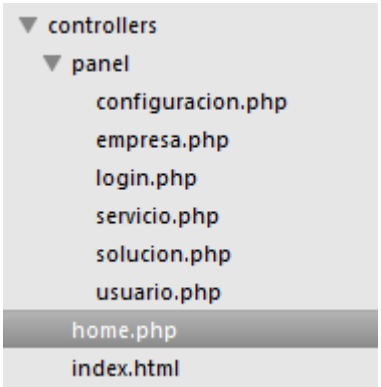
## 5. Modelos

En la carpeta modelos de application creamos una carpeta panel en el cual pondremos todos los modelos que le pertenecen a la vista administrador y fuera de la carpeta estarán los modelos que le pertenecen a la vista principal de la página web.




## 6. Controladores

En la carpeta controllers de application creamos una carpeta panel en el cual pondremos todos los controlladores que le pertenecen a la vista administrador y fuera de la carpeta estarán los controlladores que le pertenecen a la vista principal de la página web.



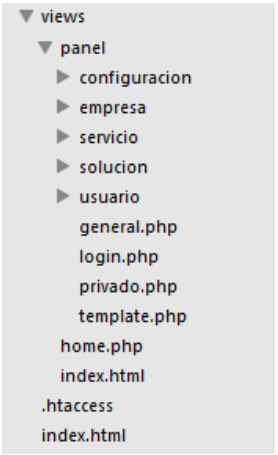

Aquí por ejemplo en el controlador de configuración, en el constructor se llama a los modelos que utilizaras para mandar información a las vistas. Aquí estamos utilizando el modelo de modulo, el cual utilizamos para recibir los módulos en los que podemos ingresar, y el modelo de configuración, en donde se guardan los valores de las variables globales que mostramos en nuestras vista principal. En nuestra función listado solo verifica que un usuario este logueado, utiliza las consultas que necesite de los modelos, se los

asigna a una variable y lo redirección a la vista del listado de configuración utilizando nuestra librería template.

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Configuracion extends CI_Controller
{

    public function __construct()
    {
        parent::__construct();


        $this->load->model('panel/modulo_model','modulo');
        $this->load->model('panel/configuracion_model','configuracion');

        $this->load->library('pagination');

        $this->output->set_title();

    }


    public function listado()
    {

        $auth = new Auth();
        if(!$auth->check(FALSE)){
            $this->load->view('panel/login');
        }else{

            $data['modulos']       =  $this->modulo->get_modulos();
            $data['configuracion'] =  $this->configuracion->get_listado();

            $this->template->display('panel/configuracion/listado', $data);
        }
    }
```

## 7. Vistas

En la carpeta views de application creamos una carpeta panel en el cual pondremos todas las vistas que le pertenecen a administrador, aquí creamos carpetas para cada controlador y justamos las vistas que le pertenece, y fuera de la carpeta estarán las que le pertenecen a la página principal.



En la vista del listado de configuracion podemos ver que solo tenemos un html en que recibimos las variables que nos manda el controlador, aquí recibimos las variables $configuracion y $modulos. En este proyecto las mostramos de la siguiente manera.

```php
<?php if($this->session->flashdata('message')){ echo $this->session->flashdata('message'); } ?>
<section id="content">
    <div class="head_box">
        <h2>Listado De Configuraciones</h2>
        <ul>
            <li>
                <a href="<?php echo base_url() ?>panel/configuracion/nuevo" class="">Nueva Configuración</a>
            </li>
        </ul>
    </div>
    <div class="body_box">
        <table class="tabla">
            <thead>
                <tr>
                    <th>Titulo</th>
                    <th>Valor</th>
                    <th class="opciones_grande">Opciones</th>
                </tr>
            </thead>
            <tbody id="lista_noticia">
                <?php
                    if(is_array($configuracion) && count($configuracion)>0){
                        foreach ($configuracion as $value) {
                ?>
                <tr id="exp_<?php echo $value['id_configuracion'] ?>">
                    <td><?php echo $value['contitulo'] ?></td>
                    <td><?php echo $value['convalor'] ?></td>
                    <td class="center">
                        <a href="<?php echo base_url() ?>panel/configuracion/editar/<?php echo $value['id_configuracion'] ?>" title="Editar Registro" class="editar tooltip"></a>
                    </td>
                </tr>
                <?php }}else{ ?>
                <tr>
                    <td colspan="2" class="mensaje_sin_data">-- No se encontraron registros --</td>
                </tr>
                <?php } ?>
            </tbody>
        </table>
        <ul id="pagination-digg">
            <?php echo $this->pagination->create_links(); ?>
        </ul>
    </div>
</section>
```

Al estar utilizando la librería template la otra parte de la vista esta en views/panel/template. Aquí podemos ver que utilizamos la variable $modulo y en la mitad tenemos la línea: <?php $this->load->view($content_template); ?> que lo que hace es insertar la otra parte del código para asi no estar reutilizando código html en vano.

```php
<?php $auth = new Auth(); ?>
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width , initial-scale=1 ,maximum-scale=1" />
        <title>MÓDULO DE ADMINISTRACIÓN - <?php echo TITULO_PAGINA ?></title>

        <link rel="shortcut icon" type="image/x-icon" href="<?php base_url() ?>public/images/favicon.ico" />
        <link rel="stylesheet" type="text/css" href="<?php echo base_url() ?>public/css/panel/reset.css" />
        <link rel="stylesheet" type="text/css" href="<?php echo base_url() ?>public/css/panel/style.css" />
        <script type="text/javascript" src="<?php echo base_url() ?>public/js/panel/jquery-1.7.1.min.js"></script>
        <script type="text/javascript" src="<?php echo base_url() ?>public/js/panel/ddsmoothmenu.js"></script>
        <script type="text/javascript" src="<?php echo base_url() ?>public/js/panel/panel.js" ></script>
</head>
    <body>
        <input type="hidden" name="base_url" id="base_url" value="<?php echo base_url() ?>" />
        <header>
            <div class="datos_usuarios">
                <a href="javascript:;">Usuario: <?php echo $auth->__get("_usuario"); ?></a> |
                <a href="<?php echo base_url() ?>panel/login/cerrar_sesion">Cerrar Sesi&oacute;n</a>
            </div>
            <div class="titulo_web">
                <h1><a href="<?php echo base_url() ?>panel"><?php echo RAZON_SOCIAL ?></a></h1>
            </div>
            <nav  id="smoothmenu1" class="ddsmoothmenu">
                <ul>
                    <li><a href="<?php echo base_url() ?>panel">Pagina de Inicio</a></li>
                    <?php if(is_array($modulos)){
                            foreach ($modulos as $value) {
                    ?>
                    <li><a href="javascript:;"><?php echo $value['modulos']['modnombre'] ?></a>
                        <ul>
                            <?php
                            $secciones = $value['secciones'];
                            if(is_array($secciones)){
                                foreach ($secciones as $sec) {
                            ?>
                            <li><a href="<?php echo base_url() ?><?php echo $sec['securl'] ?>"><?php echo $sec['secnombre'] ?></a></li>
                            <?php }} ?>
                        </ul>
                    </li>
                    <?php }} ?>
                </ul>
                <br style="clear: left" />
            </nav>
        </header>

        <?php $this->load->view($content_template); ?>

        <footer>
```

En la vista principal de la pagina views/home.php se podrán dar cuenta que existen varias variables predefinidas.

```php
<section id="content-contactenos">
    <img src="<?php echo base_url() ?>public/images/banner_contactenos.png" class="bg_contacto" />
    <div class="contactenos">
        <h2>Contáctenos</h2>
        <div id="mapa"></div>
        <div class="datos">

                <article class="detalle">
                    <h3>Información de Contacto</h3>
                    <div class="email"><?php echo EMAIL_CONTACTO ?></div>
                    <div class="telefono"><?php echo TELEFONOS ?></div>
                    <div class="direccion"><?php echo DIRECCION ?></div>
                </article>

        </div>
    </div>
```

Estas variables las creamos en una clase que se encuentra en la carpeta application/core/MY_Output.php. Aquí se crean todas las variables si es que se encuentran en la base de datos. Todo lo que se encuentre dentro de este archivo se ejecutara apenas se abra la página.

```php
<?php
class MY_Output extends CI_Output {

    function __construct()
    {
        parent::__construct();

    }

    function set_title(){

        $CI =& get_instance();

        $query = $CI->db->query("SELECT * FROM configuracion");

        foreach($query->result_array() as $row){
            define($row['contitulo'],$row['convalor']);
        }

    }
}
?>
```