GUÍA DEL DESARROLLADOR

Introducción

Esta guía pretende ser una ayuda , necesaria y suficiente, para todo aquel programador, inexperto o experimentado, que desee colaborar o continuar con el desarrollo de *CleverFigures for MediaWiki.* En ella se explican todos los puntos a tener en cuenta antes de empezar a programar. Por favor, léala atentamente y siga los consejos que en ella se dan para que su colaboración con *CleverFigures* sea aceptada y disfrutada por todos los usuarios.

CleverFigures está desarrollado usando ***PHP*** en su framework ***CodeIgniter***. Así mismo, se ha usado ***HTML*** básico para el diseño de las vistas, y ***Javascript*** para dar dinamismo en el lado del cliente. Para la interacción con la base de datos se ha usado el lenguage ***SQL***.

Subversion y el respositorio de CleverFigures

CleverFigures ha sido desarrollado usando la forja de software de *RedIris*, y usando la aplicación *Subversion* que esta tiene disponible para el control de versiones. *Subversion* es un software que permite programar de manera colaborativa. Lleva un control de todas las modificaciones que se hacen sobre los archivos de la aplicación y permite recuperar versiones antiguas en caso de que se haya subido algún cambio erróneo al servidor.

Para empezar a colaborar con *CleverFigures*, primero debe disponer de *Subversion* en su sistema. Si está en un sistema *UNIX* basado en *Debian*, puede instalarlo con esta orden:

```
sudo apt-get install subversion
```

Si está en un sistema *UNIX* basado en *Red Hat*, puede hacerlo con la siguiente orden:

```
sudo yum install subversion
```

Para otros sistemas *UNIX*, la orden dependerá de su gestor de paquetes.

Si se encuentra en un sistema *Windows*, debe descargar el ejecutable de la web oficial, donde también podrá encontrar paquetes binarios para la mayoría de las distribuciones *Linux* (http://subversion.apache.org/packages.html).

Si está familiarizado con el uso de *Subversion*, solo cabe decir que con *CleverFigures* todo será igual que en cualquier otra aplicación que haya podido desarrollar usando dicho software. Si es usted nuevo en el uso de *Subversion*, por favor, lea la siguiente guía antes de continuar: http://svnbook.red-bean.com/en/1.2/svn-book.html

La base de datos de CleverFigures

CleverFigures usa *MySQL* como gestor de bases de datos. En el momento en el que se escribe esta guía, *CleverFigures* cuenta con las siguientes tablas en su base de datos:

- **User**: tabla donde se almacenan los usuarios de la aplicación (nombre, nombre real, último acceso, idioma, contraseña, email, ¿es admin?, ¿usa alto contraste?).
- **Wiki**: tabla donde se almacena toda la información relacionada con las wikis resgistradas en la aplicación (id, nombre, id de conexión, url base).
- **Color**: tabla donde se almacena toda la información relacionada con las fuentes cualitativas resgistradas en la aplicación (id, nombre, id de conexión).
- **Connection**: tabla que almacena los credenciales de las conexiones pertenecientes a cada

wiki o fuente cualitativa (id, nombre, servidor, usuario, contraseña).

- **Groups**: tabla que almacena los grupos formados con los estudiantes de las wikis (nombre, wiki).
- **Member**: tabla que almacena los miembros de los grupos de la tabla groups (nombre, grupo).
- **Analisis**: tabla que almacena la información relacionada con los análisi realizados (id, fecha, wiki, fuente cualitativa).
- **Student-Analysis**: tabla que almacena las relaciones entre estudiantes y análisis, para la compartición de estos (nombre del alumno, fecha del análisis).
- **User-Analisis**: tabla que almacena las relaciones entre usuarios y análisis, es decir, qué profesores han hecho qué análisis (nombre de usuario, fecha del análisis).
- **User-Color**: tabla que almacena las relaciones entre usuarios y fuentes cualitativas (nombre de usuario, nombre de la fuente).
- **User-Wiki**: tabla que almacena las relaciones entre usuarios y wikis (nombre de usuario, nombre de la wiki).

Se debe intentar no modificar, en la medida de lo posible, la estructura de la base de datos para mantener la compatibilidad con versiones anteriores. De otra forma, cualquier migración supondrá modificar la base de datos ya existente en el servidor. En caso de que se modifique al liberar una nueva versión, **se deberá generar un script SQL** que corrija y/o añada las tablas o campos necesarios sin dañar los datos ya guardados.

Metodología MVC (Modelo – Vista – Controlador)

CleverFigures ha sido diseñado siguiendo la metodología MVC. Esta consiste en dividir las aplicaciones en tres capas:

- **Modelos**: es la **capa más baja**, la más lejana al usuario final y la más cercana a la base de datos. En ella deben programarse todas las funciones que interactúen con la base de datos, ficheros, etc; en general, **todo lo que deba ser abstraído del usuario**.
- **Vistas**: es la **capa más alta**, la más cercana al usuario. Normalmente se maqueta en *HTML*, *Javascript*, o cualquier otro lenguaje de diseño de interfaces web. **No deben ejecutarse funciones en ellas**; cuando se requiera insertar dinámicamente un dato, se pondrá una variable a la que se le pasará el valor necesario. En la medida de lo posible, se debe evitar escribir código de ejecución en el servidor en las vistas, a excepción de los bucles y condicionales necesarios para la creación de tablas o tareas similares.
- **Controladores**: son la capa intermedia, que **se situa entre los modelos y las vistas**. Son la lógica de la aplicación. En ellas crearemos funciones que, a su vez, usarán las funciones de los modelos para obtener los datos necesarios para pasarlos a las vistas.

CodeIgniter y su implementación de MVC

CodeIgniter implementa la *metodología MVC* mediante una **distrubución determinada de los archivos** de la aplicación. Dejando de lado todos los demás directorios (que incluyen los archivos necesarios para que puedas usar el lenguaje propio del framework), el que más nos interesa, es el directorio ***Application***, donde se gesta realmente la estructura *MVC*. En nuestro caso, como

CleverFigures usa la estructura estándar de los desarrollos con *Subversion*, este se encuentra dentro de la carpeta *trunk.*

El directorio Application

En *Application* están prácticamente todos los ficheros que se han incluído a *CodeIgniter* para formar *CleverFigures*, es el alma de la aplicación. Los ficheros están distribuídos en directorios, que explicamos a continuación.

Creación de vistas

Las vistas que se quieran añadir a *CleverFigures*, deben ir en el directorio *views*. Aquí encontrará **dos carpetas: *content* y *templates***. En la primera, se guardan todas las vistas que sirven como interfaz para cualquier funcionalidad específica de la aplicación, mientras que en la segunda, se guardan vistas comunes a todas las funcionalidades, como pueden ser la cabecera y el pie de página.

Aclarar que las vistas del directorio content **no necesitan empezar y terminar por las etiquetas iniciales de *HTML***. Solo se debe diseñar en ellas la parte correspondiente a la funcionalidad a implementar, es decir, las tablas, divs, scripts, etc, sin incluir etiquetas como <*BODY*> o <*HTML*>. De esta última tarea ya se encargan las vistas *header* y *footer* del directorio templates.

Por comodidad y convenio, he utilizado el sufijo *_views* para los nombres de cada fichero de vista.

Creación de modelos

Los modelos (recordamos: ficheros con funciones que funcionan al más bajo nivel) deben alojarse en el directorio *models*. Estos ficheros **deben seguir la estructura determinada por *CodeIgniter***, cuya explicación queda fuera del objetivo de esta guía. De nuevo, por comodidad y convenio, he utilizado el sufijo *_model* para cada fichero de modelo.

En la medida de lo posible, **se debe evitar el solapamiento de modelos**, es decir, el uso de funciones de un modelo dentro de las funciones de otro. Esto es necesario para no violar el principio de la *metodología MVC*. En casos en los que sea estrictamente necesario, se puede incluir un modelo en otro de la siguiente forma (no proporcionada por *CodeIgniter*):

```
$varname =& get_instance();
$varname->load->model('example_model');
```

Hecho esto, ya se pueden usar las funciones del modelo incluído en el modelo actual, tal que así:

```
$this->example_model->example_function($params,...);
```

Creación de controladores

Para los controladores tenemos el directorio *controllers*. Para manejar cada funcionalidad extra que queramos añadir a *CleverFigures*, será necesario crear uno o más controladores. No se ha adoptado ningún convenio con respecto al nombre de los ficheros que los contienen, debido a que es necesario que el nombre del fichero sea igual que el nombre de la clase controladora que contiene. Como *CodeIgniter* hace que el nombre de la clase sea parte de la url, era antiestético añadir sufijos o similares, y se optó por darles, tanto a los archivos como a sus clases, un nombre adecuado

teniendo en cuenta la funcionalidad que llevan a cabo. Por ejemplo: *login, close_session, analyze*,...

Como ocurre con los modelos, siempre que sea posible se debe evitar la horizontalidad en el flujo de ejecución, pero es más común (comparado con la horizontalidad en los modelos) que un controlador llame a otro, directamente o a una función específica. Un ejemplo de esto puede ser:

```
//...
//Código de ejemplo
if($is_admin)
     redirect('adminpanel');
else
     redirect('commonpanel');
```

A parte de por otro controlador, **un controlador puede ser llamado desde las vistas, mediante los formularios**. En este caso **el nombre del controlador debe coincidir con el del formulario**, y este será llamado automáticamente cuando el formulario sea enviado. Desde el controlador podrá acceder a los datos del formulario mediante

```
$this->input->post('form_item_id');
```

Configuración

El directorio *configuration* contiene todos los archivos a modificar para que *CodeIgniter* se comporte como deseamos. Los archivos más importantes son ***config.php***, que guarda parámetros como el **idioma por defecto, el charset, o la configuración de las cookies**; ***autoload.php***, que permite **definir los modelos, helpers, etc**. que queramos que se carguen automaticamente; y ***database.php***, donde deberemos introducir los **credenciales de nuestra base de datos** principal.

Estos archivos deberían, si no se ha modificado nada, venir configurados (a excepción del idioma por defecto, que deberemos escogerlo según nuestras preferencias). Normalmente solo deberemos seguir el asistente de instalación inicial y estos ficheros se configurararán por sí mismos.

Idiomas

El directorio *languages* nos permite crear traducciones de *CleverFigures*. Existe una carpeta para cada idioma, que contiene un fichero ***voc_lang.php***. Este fichero contiene todas las palabras y frases usadas en *CleverFigures*, y debe ser traducido al idioma correspondiente a la carpeta donde se aloja. Posteriormente, deberemos modificar la vista de configuración para que permita elegir el nuevo idioma.

La función encargada de tomar las líneas de estos ficheros será ***lang('code')***, la cual debemos escribir en cada lugar en el que necesitemos introducir algún término o frase dependiente del idioma. Para saber más sobre esto, lea la documentación del helper *language* de *CodeIgniter*.

El directorio trunk/analisis

Este directorio estará inicialmente vacío y **servirá para guardar los ficheros de datos una vez empecemos a hacer análisis**. Cada fichero pertenecerá a un análisis y tendrá como nombre el ***datetime*** en el que se realizó dicho análisis. Como *CleverFigures* necesitará escribir en este directorio, debemos asegurarnos de que **tiene los permisos correctos**.

El directorio css

Aquí se alojan los ficheros de hojas de estilos. Los más importantes son *styles.css*, que contiene las definiciones de estilos generales, y *acstyles.css*, que define unos estilos de **alto contraste** para personas con deficiencias visuales (opción accesible desde el menú de configuración una vez ejecutada la aplicación).

El directorio documentation

En este directorio se guardan, clasificados en sus respectivas carpetas, **documentación** sobre la base de datos de *MediaWiki*, la de *AssessMediaWiki*, la de *CodeIgniter* en sí mismo, y por supuesto las guías de usuario y desarrollador de *CleverFigures*.

El directorio images

Aquí se guardan todas las **imágenes** que se muestran en *CleverFigures,* clasificadas en carpetas según su función (imágenes de carga, logos, etc).

El directorio web

Este directorio contiene una **pequeña web** informativa de *CleverFigures* en inglés, también disponible para su visualización en http://cleverfigures.forja.rediris.es/.

Añadir una funcionalidad a CleverFigures

Por último, voy a explicar cuál sería el proceso completo a seguir para añadir una funcionalidad a *CleverFigures*.

1. Para empezar, creamos un modelo con las funciones de bajo nivel necesarias para llevar a cabo la función que deseamos implementar. En este caso, podremos como ejemplo que queremos crear una pantalla de login (que, obviamente, ya no es necesaria en *CleverFigures*). El modelo contendría un constructor y la función de *login* en sí, que comprobaría si el nombre de usuario y contraseña dados están en la base de datos:

```
<?php
class Login_model extends CI_Model{
```

```
        function User_model(){
                parent::__construct();
                $this->load->database();
        }

        function login($uname, $pass){
                $this -> db -> from('user')
                        -> where('user_username = ' . "'" . $uname . "'")
                        -> where('user_password = ' . "'" . MD5($pass) . "'")
                        -> limit(1);

                $query = $this -> db -> get();

                if($query->result()){
                        foreach($query->result() as $row)
                                $sess_array =
                                        array('username' => $row->user_username);
                                $this -> session -> set_userdata($sess_array);
                        return true;
                }
                else
                return false;
        }
}
```

2. Una vez creado el modelo, creamos el fichero que contendrá a la vista, que puede ser algo así, dependiedo del gusto del programador:

```
<table id = "maintable">
<tr>
        <td>
                <?
                        echo img('images/logo/logotrans.png');
                ?>
        </td>
        <td>
                <?= form_open('login_form') ?>
                <table id = "formins">
                        <tr>
                                <td><?=lang('voc.i18n_username')?></td>
                                <td><?= form_input('username') ?></td>
                        </tr>
```

```
                  <tr>
                        <td><?=lang('voc.i18n_password')?></td>
                        <td><?= form_password('password') ?></td>
                  </tr>
                  <tr>
                        <th colspan="2">
                              <?
=form_submit('submit',lang('voc.i18n_submit'))?>
                        </th>
                  </tr>
            </table>
            <?= form_close() ?>
      </td>
</tr>
</table>
```

3. Una vez finalizada la vista (observamos que solo incluímos la tabla a mostrar), nos fijamos en el nombre que le hemos dado al formulario, en este caso *login_form*. El controlador que creemos debe tener dicho nombre y puede ser algo tal que así:

```
class Login_form extends CI_Controller {
      function Login_form(){
                  parent::__construct();
                  $this->load->model('login_model');
                  $this->lang->load('voc',
                                    $this->session->userdata('language'));
      }

      function index(){
            if($this->user_model->login($this->input->post('username'),
                                        $this->input->post('password'))){
                  redirect('teacher');
            }
            else
                  redirect('loginerror');
      }
}
```

4. Vemos que ha sido necesario cargar el modelo *user_model*, para usar la función que creamos al principio. En caso de que el usuario no tenga un login satisfactorio, se llamará a otro controlador *loginerror*. Hecho esto, colocamos cada archivo en su carpeta correspondiente, comprobamos que todo funciona bien, y ya tenemos nuestra nueva funcionalidad añadida.

Agradecimientos

Quisiera terminar esta guía dándole las gracias por su interés en colaborar con CleverFigures y deseándole en el proceso la mejor suerte y disfrute. Cuanto menos, se llevará a cambio la satisfacción de haber creado parte de un proyecto que espero facilite la labor de muchos profesores y ayude a implementar el software libre en las aulas.