

## Capítulo 9. Creación de módulos

### 9.1. Estructura de módulos

Los módulos de PHP-Nuke están escritos en aplicaciones PHP que permiten manejar la parte central del espacio web. Por ejemplo, "News", "Forum", "Members List" etc. son módulos. Cada módulo, dependiendo de su propia complejidad, se estructura usando una parte para los usuarios y otra para el administrador, en este lugar están los contenidos que nosotros podemos modificar o configurar. Todo se ejecuta desde el archivo `modules.php`, el cual únicamente, lleva a cabo el trabajo de autenticación y manejo de los derechos de acceso al módulo. El archivo `modules.php` comprueba y verifica si el módulo está activado o no, y los derechos de acceso. Este nos reduce mucho trabajo porque no es necesario insertar estos controles en cada módulo que creamos.

Te invitamos a visitar la Sección 6.3 para informarte más exhaustivamente.

Por ejemplo, en el módulo AvantGo, para cargar el archivo `index.php`, es suficiente con pasar el nombre del módulo a la cadena de parámetros. El archivo que será buscado será el `index.php: modules.php?name=AvantGo`

Si en su lugar, queremos llamar a una página diferente del `index.php` por defecto, (por ejemplo, `print.php`), la cadena que tenemos que pasar es : `modules.php?name=AvantGo & file=print`

Esta es la variable archivo con el valor (print) que corresponde al nombre del archivo que nosotros queremos cargar, sin la extensión `.php`.

Dentro de la carpeta `modules/nombre_del_modulo` hay también una subcarpeta llamada "language". De esta forma, gestionamos de forma fácil y rápida la funcionalidad multiidioma dentro de los módulos.

Las filas de `modules.php` trabajan de la siguiente forma:

- Se incluye `mainfile.php`
- Se verifica que el módulo esté activo
- Se verifica si la cadena lleva un nombre de archivo diferente de `index.php`
- Se verifican los permisos del módulo (si todo el mundo puede verlo, o sólo los usuarios registrados, o sólo el administrador).

### 9.2. Creando módulos totalmente compatibles: reglas a seguir

Para la gente que tiene una base de conocimiento sobre el lenguaje php es muy simple construir un módulo. Generalmente, crear un módulo en PHP-Nuke significa:

- Crear archivos PHP para los usuarios, en la parte pública de la página
- Crear una interfaz de administración
- Verificar que todo lo que hemos creado cumple con las reglas de desarrollo de PHP-Nuke.
- 

¿Pero qué sabemos sobre esas reglas de desarrollo?

Será una buena idea detenernos en este punto antes de continuar con la parte de programación.

1. Los módulos deben estar incluidos en la carpeta `modules/nombre_modulo` en la parte pública y en la carpeta `admin/modules` en la parte de administración.
2. El archivo principal del módulo, incluido en `modules/nombre_modulo` se llamará `index.php`.
3. Las tablas en la sintaxis de php son identificadas por 'prefix'. Por ejemplo, las páginas de Nuke serán identificadas con `"$prefix."_pages`, donde `$prefix` cogerá el valor que el archivo `config.php` marque por defecto.
4. La localización de las imágenes o enlaces debe empezar desde la raíz de nuestra carpeta `html` y no desde la carpeta `modules/nombre_modulo` porque los archivos incluidos en ella, están a su vez incluidos en un archivo ubicado en la raíz de la carpeta `html`, llamado `modules.php`
5. Para manejar de forma óptima la función multiidoma debemos crear algunos textos que insertaremos en archivos dentro de la carpeta "language" de cada módulo. Todo su contenido será automáticamente traducido. Por ejemplo, si necesitamos crear un módulo con el nombre Topolino (nombre italiano de Mickey Mouse) deberemos dar la posibilidad a los que usen la interfaz en italiano de leer "Topolino" y a los que la usen en inglés de leer "Mickey Mouse" ; -).

¿Qué debemos hacer?

Primero crearemos la carpeta "language" dentro del directorio `modules/topolino`. Insertaremos en esta carpeta dos archivos php que llamaremos `lang-italian.php` y `lang-english.php`. Crearemos un texto para topolino, en `lang-italian.php` que será:

```
define("_TOPO", "Topolino");
```

Y en inglés será:

```
define("_TOPO", "Mickey Mouse");
```

De esta forma, al insertar `"_TOPO"` será automáticamente reemplazado por Topolino en la interfaz italiana y por Mickey Mouse en la interfaz inglesa.

### 9.3. Creación de un módulo, parte pública

Continuaremos usando el ejemplo de Topolino de la sección 9.2 y crearemos un módulo muy simple que presentará un GIF de Topolino con una lista de 3 nombres predefinidos, que serán editables por los usuarios. Este es un módulo absurdo, pero suficientemente fácil para que sea entendido por cualquiera. La base de datos que usaremos será MySQL, pero el ejemplo, cambiando algún detalle, puede hacerse con todas las bases de datos. Primero veamos cuál es el esqueleto que debemos construir en cada módulo:

```
<?php
if (!regi("modules.php", $PHP_SELF)) {
die ("You can't access this file directly...");
}
$index = 1;
require_once("mainfile.php");
$module_name = basename(dirname(__FILE__));
get_lang($module_name);
include("header.php");
include("footer.php");
?>
```

Antes de hacer nada es necesario crear una carpeta llamada "modules/Topolino", el archivo que estamos creando (con otros contenidos) se llamará `index.php` y deberá estar en esta carpeta.

Creamos una tabla en la base de datos llamada `nuke_topolino` con la siguiente estructura:

- idperson: Esta es una celda que contiene el id de persona (int 11, primary)
- nameperson: Esta es una celda que contiene los nombres de la gente (varchar 60)

Y manualmente insertamos (usando PHPMyAdmin - ver sección 11.3 – o un programa equivalente) los nombres de las 3 personas que nos interesan, ver figura 9.1 (el módulo, para simplificar, no permite añadir o cancelar personas pero se pueden editar las que ya existen).

- Id 1: Topolino
- Id 2: Minnie
- Id 3: Pluto

		idpersonaggi	nomepersonaggi
Edit	Delete	1	Topolino
Edit	Delete	2	Minnie
Edit	Delete	3	Pluto

Figura 9-1. PHPMyAdmin: insertando valores

Es posible incluir el 'footer' al final de cada función. Es una solución un poco más complicada porque debemos escribir más líneas, pero tengo que destacar que muchos módulos lo hacen así.

Una vez que la tabla de la base de datos esta lista, podemos empezar a crear el código que nos devolverá nuestra respuesta. Esta respuesta será una simple consulta con un círculo que devolverá los valores insertados en la base de datos (la cosa más simple del mundo).

¡Atención!

Para mantener los textos de las bases de datos, que podemos trabajar en diferentes tipos de bases, no podemos usar la sintaxis clásica de PHP que se utiliza normalmente en MySQL; , así que debemos utilizar la sintaxis disponible en el archivo includes/sql\_layer.php

La consulta que compilaremos se estructurará de la siguiente forma:

```
$resultpersons = sql_query("SELECT idperson, nameperson FROM
"$prefix."_topolino", $dbi);
for ($m=0; $m < sql_num_rows($resultpersons, $dbi); $m++)
{
list($idperson, $nameperson) = sql_fetch_row($resultpersons, $dbi);
echo "$idperson - $nameperson < br >";
}
```

Muy simple, ¿no os parece? Antes de pasar a la parte de administración de este módulo, empezaremos a modificarlo con la intención de darle un mínimo de estilo.

- insertar los resultados en una tabla
- ponerle un título y una descripción del módulo

Para hacer que todo el módulo sea compatible con el sistema multiidioma de PHP-Nuke, podemos definir los textos para las dos frases que necesitamos, en el archivo lang-english.php:

```
<?php
define("_BENVETOPOMOD", "¡Bienvenido al módulo Topolino!");
```

```
define("_DESCRITOPOMOD", "Este es un módulo de ejemplo que sirve para ilustrar
como se crea un módulo en PhpNuke.");
>
```

¡Recuerda que debes insertar un archivo index.htm en tu carpeta de idioma! Necesitamos hacerlo para proteger el interior de la carpeta de visitas indeseadas. Cerca del final de la parte inicial, insertaremos la parte del código que personaliza el estilo del módulo que hemos creado. Tenemos dos partes construidas previamente (la inicial y la creada por nosotros) y le insertaremos la modificación del estilo:

```
<?php if (!ereg("modules.php", $PHP_SELF))
{ die ("You can't access this rows directly..."); }
$index = 1; require_once("mainfile.php");
$module_name = basename(dirname(__FILE__));
get_lang($module_name);
include("header.php");
echo "<br>";
echo""._BENVETOPOMOD."";
echo "<br><br>";
opentable();
echo "<br>";
echo""._DESCRITOPOMOD."";
echo "<br><br>";
$resultpersons = sql_query("SELECT idperson, nameperson FROM
".$prefix."_topolino", $dbi);
for ($m=0; $m < sql_num_rows($resultpersons, $dbi);
$m++) { list($idperson, $nameperson) = sql_fetch_row($resultpersons, $dbi); echo
"$idperson - $nameperson <br>";
}
closetable();
include("footer.php");
?>
```

Sólo tenemos texto añadido, algunos 'breaks' para los 'headers' y un "Opentable();" y "Closetable();" para incluir el texto. El resultado puede verse en la figura 9-2:

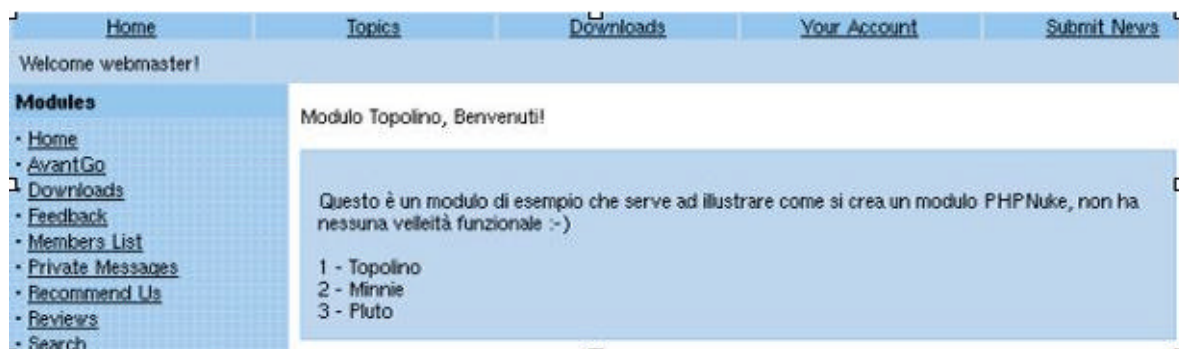


Figura 9-2. Módulo de ejemplo

#### 9.4. Creación de un módulo, parte de administración

Es hora de crear la parte de administración del módulo. En este módulo tan simple la única función que trabajará sobre la base de datos será la única en la que podemos modificar el texto de uno de los tres idiomas que hemos creado. Primero, tenemos que crear los archivos para insertar en las carpetas:

- admin/case

- admin/links
- admin/modules

Es importante recordar lo dicho en la sección 6.1:

Administración:

Contiene 4 subdirectorios (links, language, case, modules) que gestionan los diferentes módulos de administración. El directorio que recoge los archivos es modules/admin/.

La carpeta admin/links se encarga de decir qué módulo de administración será utilizado y la posición de un idioma en la administración, para un módulo determinado.

Ejemplo (Administración para el módulo FAQ):

```
if (($adminsuser==1) OR ($adminfaq==1)) {
adminmenu("admin.php?op=FaqAdmin ", "" _FAQ." ", "faq.gif");
}
```

Este módulo:

- Verifica los derechos de administración (puede establecer si puede ser visto por el superadministrador o un administrador).
- Envía un "CASE" (op=FaqAdmin) que indica al archivo admin.php (que incluye todos los módulos de administración) el módulo a llamar, asocia un valor para traducir la palabra "FAQ" y asocia una imagen para la administración gráfica (faq.gif).

La carpeta admin/case sirve para definir el módulo que será usado en cada caso específico. Esto es importante cuando, usando el mismo archivo de administración, necesitamos interpretar más operaciones usando una declaración CASE:

```
case1 = insert
case2 = cancel
etc...
```

De hecho, dice qué módulo será utilizado para verificar la condición del CASE. Por ejemplo, en el módulo FAQ hay muchos 'cases', aquí tenemos los dos últimos:

```
case "FaqAdmin":
case "FaqCatGo":
include ("admin/modules/adminfaq.php");
break;
```

Ambas declaraciones CASE llaman al archivo adminfaq.php, pero se utilizan para diferentes operaciones. La primera llama al archivo con la escena por defecto, la segunda, da el visto bueno para insertar una nueva categoría..

Esto sucede a través de una cadena como esta: "admin.php?op=FaqAdmin" en el primer 'case' y "admin.php?op=FaqCatGo" en el segundo.

Ahora crearemos los archivos:

- admin/modules/topolino.php
- admin/case/case.topolino.php
- admin/links/links.topolino.php

Para crear el archivo dentro de la carpeta modules (modules/topolino.php) debemos tener una estructura de este tipo:

- Parte inicial del módulo (parecida en todos los módulos)
- Definición de las funciones necesarias.
- Definición de los 'cases' en orden para llamar a varias funciones en el módulo de administración.
- Parte final del archivo.

La sintaxis de la parte inicial del archivo es la siguiente:

```
<?php
if (!ereg("admin.php ", { die ("Access Denied"); }
$result = sql_query("select radminsuper, admlanguage from "$prefix."_authors to
where aid=' $aid '", $dbi);
list($radminsuper, $admlanguage) = sql_fetch_row($result, $dbi);
if ($radminsuper==1) {
Esta parte del archivo controla los derechos de administración para cualquiera que
acceda, tiene un control sobre el idioma que se utiliza (no en este módulo) y un control
sobre los derechos del administrador. Un administrador podría tener sólo derechos
parciales de administración en módulos que puedan ser manejados únicamente por el
superadministrador. En nuestro caso específico, el módulo podrá ser manejado sólo
por el superadmin porque el control es sólo para:
if ($radminsuper==1)
En el caso de que haya algunos derechos específicos (por ejemplo, en el módulo
reviews) el control de derechos debería haber sido:
if (($radminreviews==1) OR ($radminsuper==1))
```

Activando los derechos sobre dos niveles en los nuevos módulos no es difícil, debes especificar en la tabla nuke\_authors un nuevo campo que designa los derechos, después modificar admin/modules/authors.php añadiendo el 'checkbox' para los derechos del nuevo módulo y modificar las correspondientes consultas UPDATE.

Volvamos a nuestro módulo, la parte inicial de la sintaxis es obligatoriamente:

```
<?php
if (!ereg("admin.php ", { die ("Access Denied"); }
$result = sql_query("select radminsuper from "$prefix."_authors to where aid=' $aid '",
$dbi);
list($radminsuper) = sql_fetch_row($result, $dbi);
if ($radminsuper==1) {
and that end is instead:
} else {
echo "Access Denied";
}
}
```

Toda la parte central, son las funciones de gestión y los casos que deben ser comprobados, y que vamos a construir ahora. Hay un par de reglas a seguir en la construcción de funciones de administración:

Debemos incluir un 'header' al principio de la función y un 'footer' al final:

```
include("header.php");
include("footer.php");
```

Tenemos que incluir la función GraphicAdmin(); inmediatamente después del 'header'. Así desplegará la barra de navegación que encabezará a todos los otros enlaces de administración. Las funciones que crearemos ahora son:

- Exponer los registros de la base de datos
- Selección del registro e inserción en un campo de texto modificable
- Modificación del registro a través de la inserción en la base de datos del valor modificado en el campo de texto.
- 

Aquí está el código de la función que realiza la selección del registro:

```
function mousedisplay() {
```

```

global $admin, $bgcolor2, $prefix, $dbi, $multilingual;
include ("header.php");
GraphicAdmin();
Opentable();
$resultpersons = sql_query("SELECT idperson, nameperson FROM
".$prefix."_topolino", $dbi);
for ($m=0; $m < sql_num_rows($resultpersons, $dbi); $m++)
{
list($idperson, $nameperson) = sql_fetch_row($resultpersons, $dbi);
echo "$idperson - $nameperson < to href=\"admin.php=mouseselect &
idtopo=$idperson \" > Select mouse </to > < br > ";
}
closetable();
include("footer.php");
}

```

La siguiente función a implementar es la que corresponde a la selección de uno de los tres registros:

```

function mouseselect() {
global $admin, $bgcolor2, $prefix, $dbi, $multilingual, $idtopo;
include ("header.php");
GraphicAdmin();
Opentable();
$resultpersons = sql_query("SELECT idperson, nameperson FROM ".$prefix."_topolino
to where idperson=' $idtopo '", $dbi);
for ($m=0; $m < sql_num_rows($resultpersons, $dbi); $m++)
{
list($idperson, $nameperson) = sql_fetch_row($resultpersons, $dbi);
echo "< form action=\"admin.php\" method=\"post\">";
echo "< input type=\"text\" name=\"nameperson \" \"size=\"20\" maxlength=\"20 \"
\"value=\"\$personname \" >< br >< br > ";
echo "< input type=\"hidden\" name=\"idperson\" value=\"\$idtopo\" > ";
echo "< input type=\"hidden\" name=\"op \" \"value=\"mousemodify\" > ";
echo "< input type=\"submit\" value=\"\"._ADDDTOPO.\"\" > ";
echo "</form >";
}
closetable();
include("footer.php");
}

```

Es muy importante tomar nota de los siguientes aspectos:

1. Las variables que insertemos serán comprobadas, de hecho puedes ver que la variable "\$idtopo" fue insertada entre las variables usadas.
2. El valor chequeado en la afirmación CASE nos pasa a través de un campo oculto de formulario.  
(< input type="hidden" name="op" value="mousemodify">).

La última función que consideraremos es la que corresponde a la actualización de los valores en la base de datos (Aquí también añadiremos las dos variables en las que estábamos interesados (\$nameperson, \$idperson):

```

function mousemodify() {
global $admin, $bgcolor2, $prefix, $dbi, $multilingual, $idtopo, $nameperson,
$idperson;

```

```

include ("header.php");
GraphicAdmin();
Opentable();
sql_query("update "$prefix."_topolino set nameperson=' $nameperson' where
idperson=$idperson", $dbi);
echo"OK ";
die(mysql_error());
closetable();
include("footer.php");
}

```

Los dos últimos elementos para insertar son las definiciones para las afirmaciones CASE (esto es, qué función se llama de acuerdo con la variable pasada al módulo) y cerramos el archivo.

Definición de los 'cases':

```

switch($op) {
case "":
mousedisplay();
break;
case "topolino":
mousedisplay();
break;
case "mouseselect":
mouseselect();
break;
case "mousemodify":
mousemodification();
break;
}
Closing of the file:
} else {
echo "Access Denied";
}
}
>

```

La página de definición de los 'cases' es muy fácil de construir, recogemos los 'cases' incluidos en el archivo admin/modules/topolino.php y los ponemos dentro del archivo admin/case/case.topolino.php

Esta es la sintaxis:

```

<?php
if (!pregi("admin.php ", { die ("Access Denied"); }
switch($op) {
case "topolino":
include("admin/modules/topolino.php");
break;
case "mouseselect":
include("admin/modules/topolino.php");
break;
case "mousemodify":
include("admin/modules/topolino.php");
break;
}
}
>

```

Las dos últimas cosas que tenemos que hacer son la compilación del archivo admin/links/link.topolino.php y la creación del módulo de idioma.

Compilación del archivo link.topolino.php:

```
<?php
if ($radminsuper==1) {
adminmenu("admin.php?op=topolino ", "" _EDITTOPOLINO." ", "topolino.gif");
}
>
```

Donde: admin.php?op=topolino define que módulo será llamado, " \_EDITTOPOLINO" es el término a traducir (debe estar definido en admin/language).

Para la modificación del módulo de idioma, os remito al párrafo anterior con una simple anotación. El archivo de idioma de la sección de administración es común para todos (admin/language), los diferentes idiomas deben añadirse siempre que existan. Otra apreciación, la sintaxis de este ejemplo no es perfecta, más allá de la intención de hacer un trabajo perfecto, queremos ilustrar la operativa de un módulo.