Organización de permisos

Víctor José Rodríguez Martín

2015-12-07

Teoría

Supongamos que hay tres clases de usuarios que llamaremos A, B y C. Y dos procesos: ver y editar.

Cada usuario puede pertenecer a una clase de usuarios, a varias o a ninguna. Para indicarlo al programa tendremos en la base de datos un campo en la tabla USUARIOS que llamaré clase_usuario aquí se guarda un número en binario que indica la clase a la que pertenece el usuario:

Si el usuario no pertenece a ninguna clase (usuario desactivado) → clase_usuario = '000'

- Si pertenece a la clase A → clase_usuario = '001'
- Si pertenece a la clase B → clase_usuario = '010'
- Si pertenece a la clase C → clase_usuario = '100'.
- Si pertenece a varias se coloca un 1 en cada una de ellas. Por ejemplo: si pertenece a las clases A y B → clase_usuario = '011'

Cada documento tiene dos clases de permisos: permisos_ver y permisos_editar, que se indican también con un número binario de tres cifras correspondientes a los permisos a las clases A, B y C empezando por la derecha. Por ejemplo si queremos otorgar permiso de ver

- a los usuarios de clase A pero no a los demás → permisos_ver = '001',
- a todas las clases → permisos_ver = '111', a las clases A y B permisos_ver = '011'. Lo mismo con los permisos para editar.

Cuando un usuario intenta ver un documento se compara clase_usuario (del usuario) con permisos_ver (del documento) haciendo un AND bit a bit. Si es resultado no es cero se le dejará ver el documento pero si es cero no.

Un ejemplo:

5 usuarios

Usuario	clase_usuario	Comentario
Alipio	'001'	es de clase A
Argimiro	'010'	es de clase B
Adolfa	'100'	es de clase C
Crisóstomo	'011'	es de las clases A y B
Leucofrina	'111'	es de las clases A, B y C

3 documentos

Documento	permisos_ver	permisos_editar
Balance 2012	'111'	'011'
Balance 2013	'110'	'100'
Balance 2014	'100'	'000'

Supongamos que Alipio quiere ver el Balance 2012. Hacemos:

```
clase_usuario[de Alipio] and permisos_ver[de Balance 2012]
      001
      111  and
      001 ≠ 000 ⇒ puede verlo
```

Argimiro quiere ver Balance 2013:

```
clase_usuario[de Argimiro] and permisos_ver[de Balance 2013]
      010
      110  and
      010 ≠ 000 ⇒ puede verlo
```

Leucofrina quiere editar Balance 2014:

```
clase_usuario[de Leucofrina] and permisos_editar[de Balance 2014]
      111
      000  and
      000 = 000 ⇒ no puede editarlo
```

Adolfa quiere editar Balance 2012

```
clase_usuario[de Adolfa] and permisos_editar[de Balance 2012]
      100
      011  and
      000 = 000 ⇒ no puede editarlo
```

Con estos códigos damos los siguientes permisos:

Documento	Damos permiso de ver a las clases	Damos permiso de editar a las clases	Tienen permiso de ver a los usuarios	Tienen permiso de editar a los usuarios
Balance 2012	A,B,C	A,B	Alipio Argimiro Adolfa Crisóstomo Leucofrina	Alipio Argimiro Crisóstomo Leucofrina
Balance 2013	B,C	C	 Argimiro Adolfa Crisóstomo Leucofrina	 Adolfa Leucofrina
Alance 2014	C	ninguna	 Adolfa Leucofrina	nadie

Ejecución

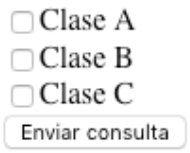
En la base de datos

En la tabla usuario, para cada uno de ellos guardaremos en un campo (clase_usuario, por ejemplo) la/las clase/s de usuario a la que pertenece. Puede guardarse en binario, pero es más práctico como un número decimal.

Entrada en la base de datos

Cuando se crea el usuario habrá un formulario en el que se indicará a qué clases pertenece. Por ejemplo mediante casillas de verificación.

```
echo "<input type='checkbox' id='clase_A' name='clase_A' value='1'>Clase A</input><br/>";
echo "<input type='checkbox' id='clase_A' name='clase_B' value='1'>Clase B</input><br/>";
echo "<input type='checkbox' id='clase_A' name='clase_C' value='1'>Clase C</input><br/>";
```



Si estas se llaman clase_A, clase_B y clase_C para obtener el número binario correspondiente a su clase de usuario se utilizará:

```
$clase_A = $_POST["clase_A"];
$clase_B = $_POST["clase_B"];
$clase_C = $_POST["clase_C"];

$clase_usuario_binario = $clase_A + $clase_B * 10 + $clase_C *100;
```

Para guardarlo en la base de datos como decimal usaremos la función

```
$clase_usuario_decimal = bindec($clase_usuario_binario);
```

Lo mismo haremos con los permisos de los documentos.

Salida de la base de datos

Cuando se consulte a la base para obtener el campo clase_usuario obtendremos un número en decimal que convertimos a binario:

```
$clase_usuario_binario = decbin($clase_usuario_decimal);
```

También de la base de datos obtenemos información de los permisos de los documentos que estarán guardados en decimal por lo que hay que convertirlos a binaro de la misma manera.

Comparación de permisos

Queremos comparar las clases a las que pertenece un usuario con los permisos para ver de un documento. Tendremos:

```
$clase_usuario_binario → clases a las que pertenece el usuario

$permisos_ver_binario → permisos para ver del documento
```

No hay ninguna función de PHP que permita hacer esto, así que he creado una:

```php
/**
 * f_and()
 *
 * Operación and de dos números binarios. No se hacen comprobaciones, por lo que si los números
 * no son binarios el resultado es impredecible
 *
 * @param              $binA, $binB dos números binarios
 * @param              $cifras número de cifras de los números
 * @return             string con el resultado $binA and $binB
 *
 * @author             Víctor Rodríguez Martín
 * @version            1.0 VRM 2015-12-07
 */
function f_and($binA,$binB,$cifras){
	/**
	 * Convierte cada operando a string y le añade tantos ceros como sea necesario para
	 *  completar las cifras.
	 *       Ej.: si $binA = 11 y $cifras = 3 quedará $binA = '011'
	 */
	if ( $binA != 0 ) {// Si es diferente de cero se obtiene las cifras usando logaritmo decimal
		$num_cifras_binA = floor(log10($binA)) + 1;// Obtiene el número de cifras de $binA
	} else {// Si es cero el número de cifras es cero log10(0) -> error
		$num_cifras_binA = 1;
	}
	$ceros_add_A = $cifras - $num_cifras_binA;// Número de ceros a añadir por la izquierda a $binA
	$ceros = "";// Cadena de ceros
	for ($i=0;$i<$ceros_add_A;$i++) {
		$ceros .= "0";
	}
	$binA = $ceros.$binA;

	if ( $binB != 0 ) {// Si es diferente de cero se obtiene las cifras usando logaritmo decimal
		$num_cifras_binB = floor(log10($binB)) + 1;// Obtiene el número de cifras de $binB
	} else {// Si es cero el número de cifras es cero log10(0) -> error
		$num_cifras_binB = 1;
	}
	$ceros_add_B = $cifras - $num_cifras_binB;// Número de ceros a añadir por la izquierda a $binB
	$ceros = "";// Cadena de ceros
	for ($i=0;$i<$ceros_add_B;$i++) {
		$ceros .= "0";
	}
	$binB = $ceros.$binB;

	/**
	 * Se comparan los dígitos uno a uno comenzando por la izquierda
	 */
	$resultado = "";
	for ($i=0;$i<$cifras;$i++) {
		$resultado .= substr($binA,$i,1) * substr($binB,$i,1);
	}
	return $resultado;
}
```

Básicamente lo que hace es convertir cada uno de los operandos en un string de ceros y unos teniendo en cuenta de añadir los ceros necesarios para que cada string tenga el número de caracteres que indica el parámetro $cifras. Luego multiplica cada cifra de $binA con la equivalente de $binB y se coloca el resultado en $resultado.

Nótese que la función no comprueba que los parámetros de entrada sean correctos, como que $binA y $binB son binarios. Lo que quiere decir que si se introducen parámetros erroneos el resultado es impredecible. Por ejemplo si se quiere compara la clase_usuario = 2 (en binario 010) con permisos_ver = 1 (en binario 001) nos da 2 que se interpretaría como “con permiso” pero si se pasan los parámetros en binario en resultado es 0 que se interpretaría como “sin permiso”.

Para comparar se puede usar este código:

```
if ( f_and($clase_usuario_binario,$permisos_ver_binario,3) > 0 ) {
  // Se puede ver el documento
  echo "Documento ...";
} else {
  ?><script>alert("No tiene permisos para esta operación")</script><?php
}
```

Como la función f_and() devuelve un string no un número, utilizar esta comparación no funcionará.

```
f_and($clase_usuario_binario,$permisos_ver_binario,3) == true
```