Cómo generar códigos QR utilizando PHP

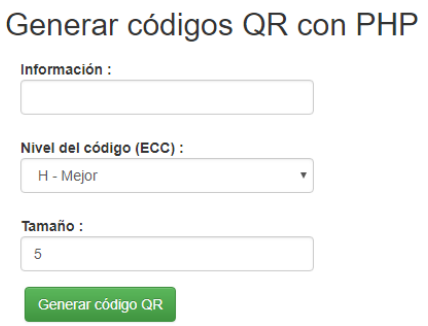
Damos por hecho que todo el mundo conoce o ha oído hablar de los códigos QR (Quick Response Code), esos códigos en forma de cuadrado y que acompañan a multitud de productos, cartelería o páginas web, entre otros elementos. Se puede definir como la evolución de los códigos de barras y en ellos es posible almacenar información que puede ser de gran importancia al usuario. A la hora de generarlos podemos encontrarnos multitud de herramientas que nos ofrecen este servicio, pero hoy veremos cómo poder implementar nuestro propio generador de códigos QR utilizando PHP y Ajax.

Para generar este código, nos apoyaremos en el uso de la librería PHP QR Code que nos proporciona todas las funcionalidades necesarias para su creación. Además de esta librería, nuestro proyecto estará formado por tres archivos:

- **index.php:** Será donde se pinte el formulario con las opciones que podemos elegir para generar nuestro código.
- **generate_code.php:** se encargará de mostrar el código generado.
- **ajax_generate_code.js:** Aquí estará el código encargado de realizar la llamada al archivo generate_code.php utilizando Ajax y mostrando en pantalla el resultado final.

Veamos con más detalle cada uno de estos archivos que utilizaremos para generar nuestros códigos QR.

Archivo index.php



Como hemos dicho anteriormente, ente archivo contendrá todo el código HTML que se encargará de mostrar al usuario el formulario con las diferentes opciones para generar el código QR. En la imagen superior se puede ver su apariencia final.

Este formulario estará dentro de un contenedor creado mediante una etiqueta "**div**" que a su vez se dividirá en dos columnas. En la izquierda de muestra el formulario mientras que en la derecha será donde se pintará el código QR. De esta forma, el código de la columna izquierda sería el siguiente.

HTML

```
<div class="col-md-3">
     <form class="form-horizontal" method="post" id="codeForm" onsubmit="return false">
        <div class="form-group">
            <label class="control-label">Información : </label>
              <input class="form-control col-xs-1" id="content" type="text"
required="required">
          </div>
          <div class="form-group">
              <label class="control-label">Nivel del código (ECC) : </label>
              <select class="form-control col-xs-10" id="ecc">
                 <option value="H">H - Mejor</option>
                 <option value="M">M</option>
                 <option value="Q">Q</option>
                 <option value="L">L - Peor</option>
              </select>
          </div>
          <div class="form-group">
             <label class="control-label">Tamaño : </label>
             <input type="number" min="1" max="10" step="1" class="form-control col-xs-
10" id="size" value="5">
          </div>
          <div class="form-group">
             <label class="control-label"></label>
             <input type="submit" name="submit" id="submit" class="btn btn-success"
value="Generar código QR">
          </div>
     </form>
</div>
```

Como se puede ver, se trata de un código muy sencillo con el que se genera el formulario visto anteriormente. Lo más destacable es el uso del evento "**onsubmit**" dentro de la etiqueta del formulario.

HTML

```
<form class="form-horizontal" method="post" id="codeForm" onsubmit="return false">
```

Este evento devolverá el valor "**false**" cuando se haya pulsado el botón de "**Generar código QR**". Con esto se evita que se envíe el formulario y que se recargue la página que estamos viendo. Esto lo hacemos ya que la imagen con el código generado se mostrará mediante Ajax.

Por otro lado, el código de la columna derecha será mucho más sencillo, tal y como se muestra a continuación.

HTML

```
<div class="col-md-6">
    <div class="showQRCode"></div>
</div>
```

Si os fijáis en este código, el primero de los "**div**" es el encargado de indicar el tamaño de la columna. El segundo de los "**div**", que tiene asignada la clase "**showQRCode**", es utilizado para pintar en su interior la imagen generada con el código.

También es importante comentar que antes del cierre de la etiqueta "**head**", hemos añadido la llamada a nuestro archivo "**ajax_generate_code.js**", que se encargará de todo el proceso Ajax.

HTML
```
<script src="script/ajax_generate_code.js"></script>
```

El código complete de este archivo index.php es el siguiente.

HTML
```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>

        <title>Generar códigos QR con PHP</title>
        <link rel="stylesheet" href="css/style.css">
        <script src="script/ajax_generate_code.js"></script>
    </head>

    <body class="">
        <div class="container" style="min-height:500px;">
            <div class="container">
                <div class="row">
                    <h2>Generar códigos QR con PHP</h2>
                </div>

                <div class="col-md-3">
                    <form class="form-horizontal" method="post" id="codeForm"
onsubmit="return false">
                        <div class="form-group">
                            <label class="control-label">Información : </label>
                            <input class="form-control col-xs-1" id="content" type="text"
required="required">
                        </div>
                        <div class="form-group">
                            <label class="control-label">Nivel del código (ECC) :
</label>
                            <select class="form-control col-xs-10" id="ecc">
                                <option value="H">H - Mejor</option>
                                <option value="M">M</option>
                                <option value="Q">Q</option>
                                <option value="L">L - Peor</option>
                            </select>
                        </div>
                        <div class="form-group">
                            <label class="control-label">Tamaño : </label>
                            <input type="number" min="1" max="10" step="1" class="form-
control col-xs-10" id="size" value="5">
                        </div>
                        <div class="form-group">
                            <label class="control-label"></label>
                            <input type="submit" name="submit" id="submit" class="btn
btn-success" value="Generar código QR">
                        </div>
                    </form>
                </div>

                <div class="col-md-6">
                    <div class="showQRCode"></div>
                </div>
            </div>
        </div>
        </div>
```

```
        <div class="insert-post-ads1" style="margin-top:20px;">

        </div>
        </div>
    </body>
</html>
```

Archivo ajax_generate_code.js

Dentro de este archivo, nos encontramos el código que se encargará de hacer la llamada al archivo "**generate_code.php**" que se encarga de crear el código QR. El código de este archivo es el siguiente:

JAVASCRIPT
```
$(document).ready(function() {
    $("#codeForm").submit(function(){
        $.ajax({
            url:'generate_code.php',
            type:'POST',
            data: {formData:$("#content").val(), ecc:$("#ecc").val(),
size:$("#size").val()},
            success: function(response) {
                $(".showQRCode").html(response);
            },
         });
    });
});
```

Lo aquí se hace es controlar cuando un usuario pulsa el botón de enviar el formulario. Esto se hace mediante la siguiente línea de código.

JAVASCRIPT
```
$("#codeForm").submit(function(){
```

Una vez pulsado, se hace la llamada Ajax utilizando la librería jQuery. A esta llamada se le pasa la url del archivo que se tiene que ejecutar, en nuestro caso "**generate_code.php**".

JAVASCRIPT
```
url:'generate_code.php',
```

El tipo método de envío que utilizará el formulario. En nuestro caso será mediante POST.

JAVASCRIPT
```
type:'POST',
```

Por último, también se les pasará los datos introducidos en el formulario. Estos datos serán los que se utilicen para la creación del código QR.

JAVASCRIPT
```
data:{formData:$("#content").val(), ecc:$("#ecc").val(), size:$("#size").val()},
```

Tras esta llamada Ajax, se recibe una respuesta. Si esta es satisfactoria, y no ha habido ningún tipo de error, lo que se hace es pintar en el "**div**" que hemos visto anteriormente y que contenía la clase "**showQRCode**" la imagen devuelta por el archivo "**generate_code.php**". Esto se hace con las siguientes líneas de código.

JAVASCRIPT
```
success: function(response) {
      $(".showQRCode").html(response);
},
```

Archivo generate_code.php

Por último nos queda ver el código del archivo "**generate_code.php**". Se puede decir que se trata del código principal de la aplicación ya que se encargar de generar el resultado final solicitado. El código completo es el siguiente.

PHP
```
<?php
if(isset($_POST) && !empty($_POST)) {
    include('library/phpqrcode/qrlib.php');
    $codesDir = "codes/";
    $codeFile = date('d-m-Y-h-i-s').'.png';
    QRcode::png($_POST['formData'], $codesDir.$codeFile, $_POST['ecc'], $_POST['size']);
    echo '<img class="img-thumbnail" src="'.$codesDir.$codeFile.'" />';
} else {
    header('location:./');
}
?>
```

Lo primero que se hace en este código es verificar que se ha enviado la petición mediante el método POST y que este no está vacío, es decir, incluye los datos para generar el código.

PHP
```
if(isset($_POST) && !empty($_POST)) {
```

Si esto ocurre, se hace la llamada a la librería que comentamos al principio de este White Paper y que nos ofrece todos los métodos para la creación del código QR.

PHP
```
include('library/phpqrcode/qrlib.php');
```

También definimos la carpeta donde se irán almacenando las imágenes que se vayan creando con cada solicitud.

PHP
```
$codesDir = "codes/";
```

Lo siguiente que hacemos es utilizar la fecha y hora actual para crear el nombre de la imagen que se vaya a generar. De esta forma nos aseguramos de que los nombres serán únicos y no habrá conflictos entre ellos.

PHP
```
$codeFile = date('d-m-Y-h-i-s').'.png';
```

Una vez que tenemos todo esto, lo que debemos hacer es realizar la llamada al método "**png**" de la librería que se encargará de crear el código.

PHP
```
QRcode::png($_POST['formData'], $codesDir.$codeFile, $_POST['ecc'], $_POST['size']);
```

A esta función hay que pasarle una serie de parámetros:

- **Información a codificar**: En nuestro caso la proporciona la variable "**$_POST['formdata']**". Recordemos que esta información la indicamos desde el formulario de recogida de datos.
- **Ubicación donde guardar el código generado**: En este ejemplo se guarda dentro de la carpeta "**codes**" con el nombre generado a partir de la fecha y hora del sistema a la hora de realizar la petición.
- **Nivel de código**: Al igual que la información a codificar, se obtiene de lo marcado en el formulario.
- **Tamaño**: Hace referencia al tamaño del código generado. Se puede poner un valor fijo para que todos sean iguales o bien, que el usuario pueda elegir el tamaño, tal y como ocurre en nuestro caso.

Por último, devolvemos el código generado a la llamada Ajax que hemos visto en el punto anterior y que se encargará de colocar este código en el contenedor que hay en la parte derecha del formulario.

PHP
```
echo '<img class="img-thumbnail" src="'.$codesDir.$codeFile.'" />';
```

El resultado final sería algo similar a lo que se muestra en la siguiente imagen.

Al tratarse de una imagen en formato PNG, los usuarios podrán descargársela y colocarla en cualquier sitio donde quieran utilizarla.