Identificadores únicos en PHP

En innumerables ocasiones necesitamos, en nuestros scripts PHP, generar identificadores únicos. Un caso muy típico de esta necesidad es cuando tenemos una aplicación web que permite al usuario enviar archivos que deberán alojarse en el servidor. Los ficheros enviados por los usuarios no pueden almacenarse con el nombre original, ya que pueden darse varios tipos de conflictos (nombres repetidos, con caracteres no aptos, etc). Por lo tanto, aunque almacenemos el nombre original, o una referencia al mismos, en una base de datos, o lo que sea, el fichero en sí mismo debe almacenarse con un nombre único que, además, sea apto para el almacenamiento de archivos en disco.
Tradicionalmente, esto podría ser un pequeño quebradero de cabeza, dado que, de algún modo, debemos "generar" una cadena aleatoria, y asegurarnos de que no tengamos la "mala pata" de que dicha cadena se repita. Afortunadamente, PHP tiene un modo de solucionar esto.

LAS MARCAS DE TIEMPO
Las marcas de tiempo son unos contadores que implementan todos los sistemas operativos y que, por lo tanto, están disponibles desde cualquier lenguaje. Cuentan el tiempo, en microsegundos, transcurrido desde lo que se ha dado en llamar el momento cero de la era Unix, que corresponde a las 00:00 horas del día 1 de enero de 1970. Por lo tanto, es una cifra (enorme) que está en constante cambio, y que nunca se repite. Si tienes curiosidad por ver esta cifra, incluye en un script lo siguiente:

```
echo "MicroTime: ".microtime();
```

El resultado en pantalla será algo similar a esto:

```
MicroTime: 0.91145600 1509122709
```

Como ves, el resultado aparece en dos partes separadas por un espacio en blanco: la segunda parte es el número de segundos transcurridos desde el comienzo de la era Unix y la primera representa los microsegundos transcurridos del segundo actual.
Puedes obtener un resultado similar en un formato más legible (bueno, relativamente más legible) si a la función le pasas el argumento `true`, así:

```
echo "MicroTime: ".microtime(true);
```

En ese caso obtienes un número flotante, de forma similar a la que ves a continuación:

```
MicroTime: 1509123085.3219
```

EL GENERADOR DE IDENTIFICADORES ÚNICOS]
PHP cuenta con la función `uniqid()`, que emplea las marcas de tiempo del sistema operativo para generar un identificador único, mediante un algoritmo del propio lenguaje. En su forma más simple, podemos emplearla así:

```
$id = uniqid();
 echo $id;
```

Esto nos mostrará por pantalla algo similar a lo siguiente:

```
59f3640d4e94a
```

En principio, esta cadena puede ser perfectamente válida para emplearla como nombre de un archivo para almacenarlo en disco, pero veamos como podemos mejorarlo. Imagina que tu aplicación puede recibir ficheros desde distintos sitios, y que tienes tantos visitantes que corres el riesgo de que a dos ficheros se les asigne un nombre en el mismo microsegundo (que ya sería mala pata, pero para eso existen las leyes de Murphy). Lo que puedes hacer es añadir, como argumento de la función, un prefijo que indique, digamos, el script que está generando el identificador. Algo como esto:

```
$id = uniqid($_SERVER['PHP_SELF');
```

```
echo $id;
```

Lo que obtendrás en pantalla es algo parecido a lo siguiente:

```
/unico/index.php59f3676520e18
```

De este modo, aunque la función se ejecute exactamente al mismo tiempo en dos puntos de tu aplicación, los resultados cambiarán por el prefijo relativo al punto de la aplicación en que se ejecuta.

Pero aún podemos mejorar esto más. La función uniqid() admite un segundo argumento que, por defecto, es false. Puedes ponerlo a true, así:

```
$id = uniqid($_SERVER['PHP_SELF', true);
echo $id;
```

El resultado será un uso interno más complejo del algoritmo, generando una secuencia más larga. Algo así:

```
/unico/index.php59f3676520e1c7.19279972
```

LOS NOMBRES DE LOS FICHEROS]

Por supuesto, el ejemplo que acabamos de ver, tal como devuelve el resultado, no nos sirve como nombre de fichero. Al incluir barras inclinadas, PHP las interpretaría como separadores de ruta y no nos almacenaría nunca el fichero en la ubicación que esperamos. Lo que podemos hacer es usar el algoritmo de encriptación **md5**. Este algoritmo fue concebido, inicialmente, como sistema de seguridad para encriptación de contraseñas. Hoy día está en desuso para tal finalidad, ya que ha sido "roto" innumerables veces en todo el mundo, y no es fiable. Sin embargo, para darle un formato adecuado a una cadena que usaremos como nombre para almacenar un archivo es perfectamente válido. Supongamos que, sobre el último resultado que has visto, haces algo como lo siguiente:

```
echo md5($id);
```

Lo que obtienes ahora tiene un aspecto muy similar a esto:

```
658d2f03f0189469816bfc1524f682bd
```

LA EXTENSIÓN]

Desde luego, ya tenemos un identificador completamente único, que estamos seguros al cien por cien de que no se va a repetir nunca jamás, y que tiene el formato adecuado para grabar el fichero en el disco. Sin embargo, a la hora de "construir" el nombre con el que, finalmente, se grabará el fichero, esto no es suficiente. A la última cadena vista debemos añadirle la extensión relativa al tipo de fichero. Si se trata de un archivo de texto plano, puede ser .txt; si se trata de una imagen, puede ser .jpg o .png, por ejemplo; si es un archivo de datos, puede ser .csv, .xml, etc. Las posibilidades pueden ser muchas. La extensión la vamos a sacar del nombre original del archivo. Cuando recibimos un archivo en PHP, en el correspondiente elemento de la matriz $_FILES tenemos, entre otras, la propiedad name, que contiene el nombre original con el que el usuario ha enviado el archivo. Suponte que te han mandado Foto de mi gato.jpg. Haremos lo siguiente:

```
$info = new SplFileInfo('Foto de mi gato.jpg');
$extension = '.'.$info->getExtension();
```

Esto nos devuelve, en la variable $extension, el valor .jpg. Ahora, simplemente, lo añadimos al identificador único que teníamos, y ya sí tenemos construido un nombre adecuado para grabar el fichero en la ubicación que deseemos en nuestro servidor.

La clase SplFileInfo nos puede permitir obtener muchísima información adicional sobre un fichero. Si quieres conocerla más a fondo, puedes revisar la documentación oficial en este enlace.