Apuntes de funciones para validad datos en un Formulario

Antes de utilizar en un programa los datos recibidos a través de un formulario, es necesario comprobar si el dato recibido corresponde a lo esperado. Algunas comprobaciones se pueden hacer con comparaciones (si no es vacío, si es un valor comprendido entre unos valores determinado, etc.), pero antes de hacer esas comparaciones hay que comprobar que es del tipo esperado (número entero o decimal, texto, etc.) para procesarlo sin error.

En esta lección se comentan una serie de familias de funciones que permiten comprobar la existencia, el tipo o el contenido de un dato:

- funciones is_
- funciones ctype_
- funciones filter_
- funciones _exists
- expresiones regulares

Pero antes se comenta un caso especial, la comprobación de números, para la que se aconseja utilizar las funciones is_numeric() y ctype_digit().

Comprobación de números con is_numeric() y ctype_digit()

Cuando se quiere comprobar que un dato recibido recibido a través de un formulario es un número, en el caso de las funciones is_ o ctype_, sólo tiene sentido utilizar dos de ellas.

1.- Comprobación de números con is_numeric()

Para comprobar si el dato que se ha recibido es un número (o se puede interpretar como número) se debe utilizar la función is_numeric($valor). Esta función lógica devuelve true si el argumento se puede interpretar como número y false si no lo es.

El motivo por el que se debe utilizar la función is_numeric($valor), es que lo que se recibe a través de un formulario se guarda siempre como cadena (aunque sea un número). La función is_numeric($valor) es la única que comprueba si el argumento se puede interpretar como número (aunque el argumento sea de tipo cadena), la función is_int($valor) o is_float($valor) hacen solamente comprobaciones sobre el tipo de los datos y devolverían siempre false.

2.- Comprobación de números enteros positivos con ctype_digit()

Para comprobar si el dato que se ha recibido es un número entero positivo (sin decimales) se puede utilizar la función ctype_digit($valor). Esta función lógica devuelve true si todos los caracteres del argumento son dígitos (0, 1, ..., 9) y false si no lo son.

3.- Funciones is_

Las funciones is_ son un conjunto de funciones booleanas que devuelven true si el argumento es de un tipo de datos determinado y false si no lo son.

	Función	Tipo de datos	alias (funciones equivalentes)
existencia	isset($valor)	devuelve si el dato está definido o no	
	is_null($valor)	null	
	is_bool($valor)	booleano	
números	is_numeric($valor)	número (puede tener signo, parte decimal y estar expresado en notación decimal, exponencial o hexadecimal).	
	is_int($valor)	entero	is_integer($valor), is_long($valor)
	is_float($valor)	float	is_double($valor), is_real($valor)
cadenas	is_string($valor)	cadena	
	is_scalar($valor)	escalar (entero, float, cadena o booleano)	
	is_array($valor)	matriz	
otros	is_callable($valor)	función	
	is_object($valor)	object	
	is_resource($valor)	recurso	

Si un dato es demasiado grande para el tipo de variable, las funciones devolverán false. Por ejemplo, si se usa como argumento un entero mayor que PHP_INT_MAX (2147483647 en Windows), la función is_int() devolverá false.

Estas funciones son en general de poca utilidad con datos provinientes de un formulario, ya que estas funciones lo que comprueban es el tipo de los datos y la información que llega de un formulario es siempre del tipo cadena. Las dos excepciones serían:

- la función is_numeric(), que evalúa si el argumento se puede interpretar como número (aunque el tipo sea cadena). Por tanto esta función se puede utilizar para comprobar si un dato recibido es un número.
- la función isset(), que evalúa si el argumento está o no definido, independientemente de su tipo.

4.- Funciones ctype_

Las funciones ctype_ son un conjunto de funciones booleanas que devuelven si todos los caracteres de una cadena son de un tipo determinado, de acuerdo con el juego de caracteres local. Estas funciones son las mismas que las que proporciona la biblioteca estándar de C ctype.h.

Función	Tipo de datos
ctype_alnum($valor)	alfanuméricos
ctype_alpha($valor)	alfabéticos (mayúsculas o minúsculas, con acentos, ñ, ç, etc)
ctype_cntrl($valor)	caracteres de control (salto de línea, tabulador, etc)
ctype_digit($valor)	digitos
ctype_graph($valor)	caracteres imprimibles (excepto espacios)
ctype_lower($valor)	minúsculas
ctype_print($valor)	caracteres imprimibles
ctype_punct($valor)	signos de puntuación (caracteres imprimibles que no son alfanuméricos ni espacios en blanco)
ctype_space($valor)	espacios en blanco (espacios, tabuladores, saltos de línea, etc)
ctype_upper($valor)	mayúsculas
ctype_xdigit($valor)	dígitos hexadecimales

Estas funciones se pueden aplicar a los datos recibidos de un formulario ya que hacen comprobaciones de todos los caracteres de una cadena. Quizás la más útil es:

- la función ctype_digit(), que evalúa si todos los caracteres son dígitos, por lo que permite identificar los números enteros positivos (sin punto decimal ni signo negativo)

5.- Funciones filter_

Las funciones filter se crearon como extensión PECL, pero se incluyeron en PHP 5.2 (2006).

La función filter más simple es la función filter_var($valor [, $filtro [, $opciones]]), que devuelve los datos filtrados o false si el filtro falla.

Los filtros predefinidos de validación son los siguientes:

Filtro	Tipo de datos
FILTER_VALIDATE_INT	entero
FILTER_VALIDATE_BOOLEAN	booleano
FILTER_VALIDATE_FLOAT	float
FILTER_VALIDATE_REGEXP	expresión regular
FILTER_VALIDATE_URL	URL
FILTER_VALIDATE_EMAIL	dirección de correo
FILTER_VALIDATE_IP	dirección IP

FILTER_VALIDATE_MAC	dirección MAC física

El problema de los filtros FILTER_VALIDATE_INT y FILTER_VALIDATE_FLOAT es que dan false si el argumento es 0, lo que complica su uso para detectar números porque el caso del 0 debe considerarse aparte al hacer la validación. Se supone que ese comportamiento se corrgió en PHP 5.4, pero **NO ESTA CLARO** que el problema esté resuelto en versiones posteriores.

6.- Expresiones REGEX y todas las funciones asociadas

(Ya visto anteriormente)