VALIDACIÓN DE DATOS CON "PREG_MATCH"

La función **preg_match** la usaremos para saber simplemente, si en una cadena de texto se encuentra lo que buscamos.

```
<?
    preg_match($patron, $string)
?>
```

Usaremos un ejemplo para explicar su funcionamiento:

```
<?
    $cadena = '135hola842';
    echo preg_match('/hola/', $cadena);
?>
```

Nos devuelve 1, porque lo ha encontrado. Si no lo hubiera encontrado nos habría devuelto 0.

El patrón que va como primer parámetro de la función preg_match deberá ir entre comillas simples o dobles, a no ser que esté dentro de una variable.

Cuando se usan patrones, se requiere que el patrón esté encerrado entre **delimitadores**. Los delimitadores que se usan a menudo son barras oblicuas (/), signos de número (#) y tildes (~).

Los elementos de un patrón pueden ser los siguientes:

- **Modificadores de patrón**: Modifican el funcionamiento del patrón. Se colocan detrás del delimitador final del patrón. Los más usados son.
  - ✓ i➔ Las letras en el patrón coincidirán tanto en letras mayúsculas como minúsculas.
  - ✓ x➔Se ignoran los espacios en blanco del patrón excepto cuando sean escapados.

    Ejemplo: $patron='/hola/i';
- **Meta-caracteres**: Sirven para formar el patrón:
  - ✓ ^algo ➔ Se usa para encontrar algo al inicio: "algo asd" -> Correcto
  - ✓ algo**$** ➔ Se usa para encontrar algo al final: "asd algo" -> Correcto
  - ✓ "\" ➔se usa para escapar caracteres.
  - ✓ "**.**" ➔representa cualquier carácter, salvo el de nueva línea.
  - ✓ ***** ➔ Se usa cuando puede aparecer 0 o más veces: abc(d*)ef => abcdef, abcef, abcddddef -> Correcto

- ✓ **?** ➔ Se usa cuando puede aparecer 0 o 1 vez: abc(d?)ef => abcdef, abcef -> Correcto
- ✓ **+** ➔ Se usa cuando puede aparecer 1 o más veces: abc(d+)ef => abcdef, abcdddddef -> Correcto
- ✓ **( )** ➔Los paréntesis se usan cuando se quiere buscar algo literal. (abc) Para que sea verdadero tiene que tener "abc".
- ✓ **|** ➔Las barras verticales son sinónimo del operador lógico OR. Es verdadero si una de las dos aparece: (a|b) Para que sea verdadero tiene que ser, o 'a', o 'b'.
- ✓ **\d{x}** ➔ Se usa cuando queremos que aparezca un número X veces. **\d{x,y}**➔ Se usa cuando queremos que aparezca un número X o Y veces.
- ✓ Los paréntesis ( ) también se usan se usan para delimitar un subpatrón:
- ✓ x(y{a,b})z Se usa cuando puede aparecer un mínimo de 'a' veces y un máximo de 'b' veces.
  Ejemplo: x(y{1,3})z => xyz, xyyz, xyyyz -> Correcto
- ✓ x(y{n})z Se usa cuando puede aparecer n veces
  Ejemplo: x(y{2})z => xyyz -> Correcto
  Otros ejemplos:
  x(y{2, })z -> Se usa cuando puede aparecer dos o más veces.
  x(y{0,3})z > Se usa cuando puede aparecer tres o menos veces.
- ✓ Los corchetes [ ] nos sirven para definir rangos (de caracteres ascii). Ejemplo: ab[c-f]g => abcg, abdg, abeg, abfg -> Correcto
- ✓ Se usa ^ dentro de los corchetes para negar. [^cd] Se usa cuando no puede aparecer ni c ni d.

  Ejemplo: ab[^cd]ef => abjef -> Correcto, abcef -> Incorrecto

- **Tipos de caracteres genéricos**:

- ✓ \s equivale a un espacio en blanco.
- ✓ \S equivale a cualquier cosa que no sea un espacio en blanco.
- ✓ \d equivale a cualquier dígito.
- ✓ \D equivale a cualquier cosa que no sea un dígito.
- ✓ \w equivale a cualquier dígito, letra o guión bajo.
- ✓ \W equivale a cualquier cosa que no sea dígito, letra o guión bajo.

  Ejemplo: $patron='/[a-z\d]*/i'

EJEMPLO: Validar un nombre de usuario

Esta regla es para permitir usuarios de 4 hasta 28 caracteres de longitud, alfanuméricos y permitir guiones bajos.

```
$string = "userNaME4234432_";
if (preg_match('/^[a-z\d_]{4,28}$/i', $string))
{
   echo "CORRECTO";
}
```

ALMACENAR DATOS DE UN SELECT MÚLTIPLE

La opción **SELECT MULTIPLE** dentro de un **form** típico de HTML permite elegir **ninguno**, **uno** o **varios** de los elementos de la lista.

Para recoger los valores de esta opción se define dentro de la etiqueta SELECT un **name** tipo **array**. **<SELECT MULTIPLE name=***var***[ ] SIZE=6>**

El *truco* está en los **values** de cada **option** dentro de ese **select**

<option value=**1**>Castellano <option value=**2**>Francés

<option value=**4**>Inglés <option value=**8**>Alemán

<option value=**16**>Búlgaro <option value=**32**>Chino

- ✓ Se ha mantenido el mismo *orden* en que fueron definidos en el campo **SET** de la tabla. Los valores **1**, **2**, **4**, **8**, **16** y **32** son *las potencias de 2:* 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, y 2^6
- ✓ Al ir seleccionando valores, van añadiéndose al *array*. Por ejemplo. Si se selecciona **Francés** y **Búlgaro** el array sería : **var[0]=2**, **var[1]=16**
- ✓ Si se suman esos valores el resultado sería **18**, y al convertir a **binario** este valor: **010010** que es como decirle a MySQL (mirando la cadena de derecha a izquierda, que incluya los valores *segundo* (Francés) y *quinto* (Búlgaro) del SELECT MULTIPLE. Hay que asignarle la suma en el INSERT.

Cuando se extraen los datos en un select, se devuelve como un **string** donde los campos seleccionados están separados por comas.

Para almacenar estos valores, en la BD, el campo debe estar definido como de tipo **SET**.