Expresiones Regulares en PHP

November 25, 2025

Contents

1 Introducción a las Expresiones Regulares en PHP **3**

2 Funciones comunes de expresiones regulares en PHP **3**

3 Sintaxis básica y ejemplos comunes **4**

3.1 Metacaracteres esenciales 4

3.2 Cuantificadores 5

3.3 Ejemplos prácticos con `preg_match()` 5

4 Construcción de expresiones regulares personalizadas **6**

4.1 Ejemplo: Validar una fecha en formato DD/MM/AAAA 7

5 Ejemplos adicionales con `preg_replace()` y `preg_split()` **7**

5.1 Reemplazar espacios múltiples por uno solo 7

5.2 Separar palabras por espacios en un array 8

6 Consejos para dominar las expresiones regulares en PHP **8**

7 Conclusión **9**

1 Introducción a las Expresiones Regulares en PHP

Las expresiones regulares (o *regex*) son herramientas poderosas para buscar, validar y manipular texto mediante patrones definidos. En PHP, estas expresiones permiten procesar cadenas para detectar coincidencias, reemplazar fragmentos o extraer información específica. Su utilidad abarca desde la validación de formularios hasta la depuración de datos complejos, siendo un recurso indispensable para cualquier desarrollador que busque precisión y eficiencia en el manejo de textos.

PHP implementa dos principales familias de funciones para trabajar con expresiones regulares: las basadas en la extensión PCRE (Perl Compatible Regular Expressions), que utilizan la sintaxis similar a Perl y son las más usadas actualmente, y las funciones POSIX, que están obsoletas y no se recomiendan.

Esta guía aborda la sintaxis fundamental de las expresiones regulares en PHP, ejemplos prácticos con las funciones más comunes, y consejos para crear tus propios patrones adaptados a tus necesidades específicas.

2 Funciones comunes de expresiones regulares en PHP

PHP ofrece varias funciones para trabajar con expresiones regulares, pero las más comunes y efectivas provienen de PCRE, cuyo prefijo habitual es `preg_`. A continuación, describimos las principales:

Función	Descripción
`preg_match()`	Busca una coincidencia en una cadena. Devuelve 1 si hay coincidencia, 0 si no, o false en caso de error. Útil para validar formatos.
`preg_match_all()`	Busca todas las coincidencias en una cadena y las almacena en un array.
`preg_replace()`	Reemplaza las coincidencias por un texto dado.
`preg_split()`	Divide una cadena usando una expresión regular como delimitador.

3 Sintaxis básica y ejemplos comunes

Las expresiones regulares en PHP se delimitan con caracteres especiales, comúnmente barras diagonales `/`, aunque pueden usarse otros delimitadores. Entre estos delimitadores se escribe el patrón a buscar, que puede incluir caracteres literales, metacaracteres y cuantificadores.

3.1 Metacaracteres esenciales

- `.` : Cualquier carácter excepto salto de línea.
- `\d` : Dígito (equivalente a [0-9]).
- `\D` : No dígito.
- `\w` : Caracter alfanumérico (_ incluido).
- `\W` : No alfanumérico.
- `\s` : Espacio en blanco (espacio, tab, salto línea).
- `\S` : No espacio en blanco.

- `[abc]` : Cualquier carácter entre corchetes (en este caso a, b o c).
- $[^abc]$: $Caracteresexceptolosindicados.$
- $:Iniciodelnea.$
- `$ : Fin de línea.`

3.2 Cuantificadores

- `*` : Cero o más veces.
- `+` : Una o más veces.
- `?` : Cero o una vez.
- `{n}` : Exactamente n veces.
- `{n,m}` : Entre n y m veces.

3.3 Ejemplos prácticos con `preg_match()`

- **Validar un número de teléfono (formato simple):**

```
$pattern = ”/^\d{3}-\d{3}-\d{4}$/”;
$phone = ”555-123-4567”;
if (preg_match($pattern, $phone)) {
    echo ”Teléfono válido”;
}
```

- **Detectar correos electrónicos básicos:**

```
$pattern = ”/^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,6}$/”;
$email = ”usuario.ejemplo@mail.com”;
if (preg_match($pattern, $email)) {
    echo ”Correo válido”;
}
```

- **Comprobar que una cadena contiene solo letras:**

```
$pattern = "/^[a-zA-Z]+$/";
$cadena = "SoloLetras";
if (preg_match($pattern, $cadena)) {
    echo "Cadena válida";
}
```

4 Construcción de expresiones regulares personalizadas

Crear tu propia expresión regular requiere entender el patrón de texto que deseas capturar o validar. Este proceso puede dividirse en pasos lógicos:

1. **Definir el objetivo:** ¿Qué texto quieres detectar o manipular? Por ejemplo, validar un código postal o extraer fechas.
2. **Descomponer el patrón:** Analiza la estructura del texto. Por ejemplo, una fecha puede tener formato `dd/mm/aaaa` donde `dd` es día, `mm` mes y `aaaa` año.
3. **Seleccionar metacaracteres adecuados:** Usa clases de caracteres y cuantificadores para representar cada parte. Ejemplo para día: `\d{2}`.
4. **Agregar límites:** Para evitar coincidencias parciales, incluye anclas y`$`$para indicar inicio y fin exacto.$
5. **Probar y ajustar:** Usa funciones como `preg_match()` para verificar y depurar el patrón.

4.1 Ejemplo: Validar una fecha en formato DD/MM/AAAA

Construcción paso a paso:

- Día: dos dígitos, posible del 01 al 31 (simplificado como `\d{2}`).
- Separador: barra `/`.
- Mes: dos dígitos, 01 a 12 (simplificado).
- Separador: barra `/`.
- Año: cuatro dígitos.

Expresión regular:

```
$pattern = "/^\d{2}\/\d{2}\/\d{4}$/";
```

Código PHP:

```
$fecha = "25/12/2023";
if (preg_match($pattern, $fecha)) {
    echo "Fecha válida";
} else {
    echo "Formato incorrecto";
}
```

5 Ejemplos adicionales con `preg_replace()` y `preg_spli`

5.1 Reemplazar espacios múltiples por uno solo

```
$texto = "PHP    es   genial";
$patron = "/\s+/";
$textoLimpio = preg_replace($patron, " ", $texto);
echo $textoLimpio; // "PHP es genial"
```

5.2 Separar palabras por espacios en un array

```
$frase = ”Aprender PHP es divertido”;
$patron = ”/\s+/”;
$palabras = preg_split($patron, $frase);
print_r($palabras);
```

Resultado:

```
Array
(
    [0] => Aprender
    [1] => PHP
    [2] => es
    [3] => divertido
)
```

6 Consejos para dominar las expresiones regulares en PHP

- Siempre delimita correctamente los patrones con `/` o delimitadores alternativos si el patrón contiene barras.
- Escapa caracteres especiales si deseas que se interpreten literalmente (ejemplo: punto `.` como `\.`).
- Usa paréntesis `()` para capturar subcadenas específicas.
- Familiarízate con cuantificadores para controlar repeticiones.
- Emplea herramientas online de prueba para validar tus expresiones antes de usarlas en código.

7 Conclusión

Las expresiones regulares en PHP son una herramienta esencial para cualquier desarrollador que trabaje con textos o datos en cadena. Su dominio permite realizar validaciones robustas, extracciones precisas y manipulaciones eficientes. Aunque al principio la sintaxis puede parecer compleja, la práctica y la estructuración lógica de los patrones facilitan su comprensión y aplicación.

Crear tus propias expresiones regulares requiere paciencia y análisis detallado del texto objetivo, pero con los ejemplos y conceptos aquí expuestos, el camino para diseñar patrones personalizados y efectivos será más claro y seguro.

Referencias

- Manual PHP - Extensión PCRE
- Regular-Expressions.info - Expresiones regulares en PHP
- Regex101 - Herramienta interactiva para pruebas