Cómo usar expresiones regulares de PHP para la coincidencia de patrones y la validación de datos

olivia j · Seguir

8 minutos de lectura · Hace 4 días

Las expresiones regulares son herramientas poderosas para encontrar y manipular texto basado en ciertos patrones. Se pueden usar para diversas tareas, como validar la entrada del usuario, extraer datos de páginas web, reemplazar texto en archivos y más.

En esta publicación, aprenderemos a usar expresiones regulares en PHP, un popular lenguaje de secuencias de comandos del lado del servidor. Cubriremos los conceptos básicos de la sintaxis de expresiones regulares, cómo crear y usar objetos de expresiones regulares y algunos ejemplos comunes del uso de expresiones regulares en PHP.

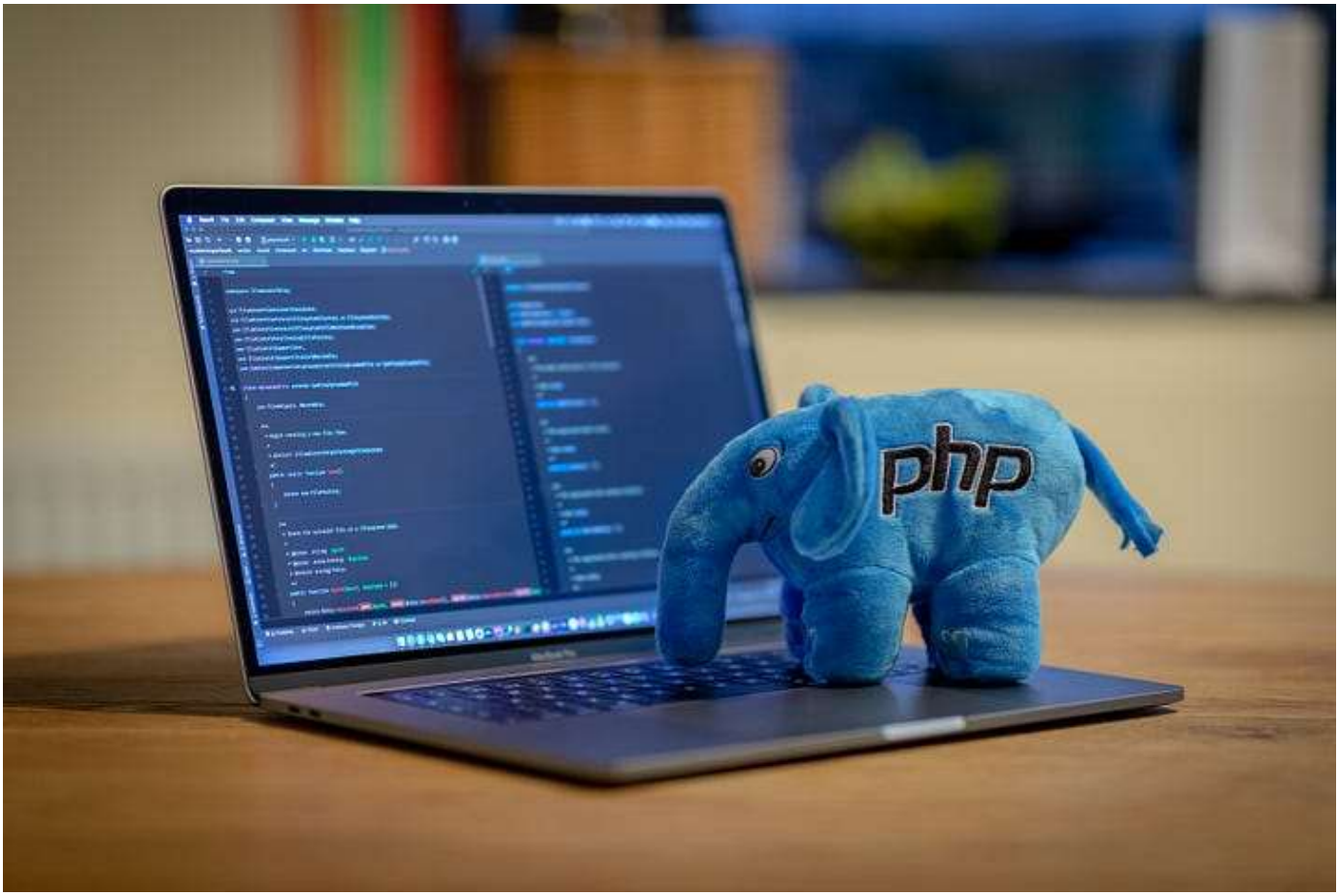

Foto de Ben Griffiths en Unsplash

¿Qué son las expresiones regulares?

Una expresión regular (o regex) es una secuencia de caracteres que define un patrón de búsqueda. Por ejemplo, la expresión regular `/[a-z]+/` coincide con una o más letras minúsculas, mientras que la expresión regular `/[0-9]{3}-[0-9]{3}-[0-9]{4}/` coincide con un número de teléfono en el formato `xxx-xxx-xxxx`.

Una expresión regular puede constar de caracteres literales (como `a`, `b`, `c`, etc.), caracteres especiales (como `^`, `$`, `.`, `*`, etc.) y clases de caracteres (como `[a-z]`, `[0-9]`, `\w`, `\d`, etc.). Cada carácter o grupo de caracteres tiene un significado y una función específicos en la expresión regular.

Cómo crear y usar objetos de expresiones regulares en PHP

En PHP, hay dos formas de crear y usar objetos de expresión regular: usando las `preg_` funciones o usando la `PCRE` clase.

Las `preg_` funciones son un conjunto de funciones integradas que le permiten realizar varias operaciones con expresiones regulares, como hacer coincidir, reemplazar, dividir y filtrar. `preg_` Las funciones más utilizadas son:

- `preg_match($pattern, $subject, $matches)` - Esta función intenta hacer coincidir una expresión regular `$pattern` con una cadena `$subject` . Si se encuentra una coincidencia, devuelve `true` y almacena los subpatrones coincidentes en una matriz `$matches` . De lo contrario, vuelve `false` .
- `preg_replace($pattern, $replacement, $subject)` - Esta función reemplaza todas las apariciones de una expresión regular `$pattern` en una cadena `$subject` con una cadena `$replacement` . Devuelve la cadena modificada o `null` si se produce un error.
- `preg_split($pattern, $subject)` - Esta función divide una cadena `$subject` en una matriz de subcadenas utilizando una expresión regular `$pattern` como delimitador. Devuelve la matriz de subcadenas o `false` si se produce un error.
- `preg_grep($pattern, $array)` - Esta función filtra una matriz `$array` de cadenas devolviendo solo aquellas que coinciden con una expresión regular `$pattern` . Devuelve la matriz filtrada o `false` si se produce un error.

Para usar las `preg_` funciones, debe encerrar su expresión regular en delimitadores (como `/` , `#` o `~`) y, opcionalmente, agregar modificadores (como `i` , `m` o `g`) después del delimitador de cierre. Por ejemplo:

```
// Coincide con cualquier palabra que comience con "cat"
$pattern = "/\bcat\w*/i" ; // El modificador "i" hace que la coincidencia no dist
$subject = "Me gustan los gatos y las orugas, pero no las catacumbas". ;
if ( preg_match ( $patrón , $sujeto , $coincidencias )) {
  echo  "Coincidencia encontrada:" . $coincidencias [ 0 ] . "\n" ; // Salida: Co
 } else {
  echo  "No se encontró ninguna coincidencia.\n" ;
}

// Reemplazar todas las apariciones de "perro" con "cachorro"
$patrón = "/perro/" ;
$reemplazo = "cachorro" ;
$subject = "El perro persiguió al gato y el perro le ladró al cartero". ;
```

```
$nuevo_sujeto = preg_replace ( $patrón , $reemplazo , $sujeto );
echo  $nuevo_sujeto . "\n" ; // Salida: El cachorro persiguió al gato y el cacho

// Dividir una cadena por comas o espacios
$patrón = "/[,\s]+/" ;
"rojo, verde azul, amarillo" ;
$matriz = preg_split ( $patrón , $sujeto );
imprimir_r ( $arreglo ); // Salida: matriz ([0] => rojo [1] => verde [2] => azul

una matriz de direcciones de correo electrónico por nombre de dominio
$patrón = "/@gmail\ .com$/" ;
$matriz = [ "alice@gmail.com" , "bob@yahoo.com" , "charlie@hotmail.com" , "david(
$nueva_matriz = preg_grep ( $patrón ,
imprimir_r ( $nueva_matriz ); // Salida: Matriz ([0] => alice@gmail.com [3] => da
```

La `PCRE` clase es un contenedor orientado a objetos para las `preg_` funciones. Le permite crear y manipular objetos de expresiones regulares utilizando métodos y propiedades. Los métodos y propiedades más utilizados son:

- `__construct($pattern)` - Este método crea un nuevo objeto de expresión regular con una expresión regular dada `$pattern` .
- `match($subject)` - Este método intenta hacer coincidir la expresión regular con una cadena `$subject` . Si se encuentra una coincidencia, devuelve una matriz de subpatrones coincidentes. De lo contrario, devuelve una matriz vacía.
- `replace($replacement, $subject)` - Este método reemplaza todas las apariciones de la expresión regular en una cadena `$subject` con una cadena `$replacement` . Devuelve la cadena modificada.
- `split($subject)` - Este método divide una cadena `$subject` en una matriz de subcadenas utilizando la expresión regular como delimitador. Devuelve la matriz de subcadenas.
- `grep($array)` - Este método filtra una matriz `$array` de cadenas y devuelve solo aquellas que coinciden con la expresión regular. Devuelve la matriz filtrada.
- `$delimiter` - Esta propiedad contiene el delimitador utilizado para la expresión regular. Se puede cambiar llamando al `setDelimiter($delimiter)` método.

- `$modifiers` - Esta propiedad contiene los modificadores utilizados para la expresión regular. Se puede cambiar llamando al `setModifiers($modifiers)` método.

Para usar la clase PCRE, debe incluirla en su script usando la `require_once()` función. Por ejemplo:

```
// Incluir la clase PCRE
require_once ( "PCRE.php" );

// Crear un nuevo objeto de expresión regular
$regex = new  PCRE ( "/\bcat\w*/i" );

// Coincide con cualquier palabra que comience con "gato"
$subject = "Me gustan los gatos y las orugas, pero no las catacumbas". ;
$coincidencias = $regex -> coincidencia ( $asunto );
if (! vacío ( $coincidencias )) {
  echo  "Coincidencia encontrada: " . $coincidencias [ 0 ] . "\norte"// Salida: (
 } else {
  echo  "No se encontró ninguna coincidencia.\n" ;
}

// Reemplazar todas las apariciones de "perro" con "cachorro"
$regex -> setPattern ( "/perro/" );
$reemplazo = "cachorro" ;
$subject = "El perro persiguió al gato y el perro le ladró al cartero". ;
$nuevo_asunto = $regex -> reemplazar ( $reemplazo , $asunto );
echo  $nuevo_sujeto . "\n" ;// Salida: El cachorro persiguió al gato y el cachor

// Dividir una cadena por comas o espacios
$regex -> setPattern ( "/[,\s]+/" );
$sujeto = "rojo, verde, azul, amarillo" ;
$matriz = $regex -> split ( $asunto );
imprimir_r ( $arreglo ); // Salida: matriz ([0] => rojo [1] => verde [2] => azul

una matriz de direcciones de correo electrónico por nombre de dominio
$regex -> setPattern ( "/ @gmail\.com$/" );
$matriz = ["alice@gmail.com" , "bob@yahoo.com" , "charlie@hotmail.com" , "david@
$nueva_matriz = $regex -> grep ( $matriz );
imprimir_r ( $nueva_matriz ); // Salida: Matriz ([0] => alice@gmail.com [3] => da
```

Ejemplos de uso de expresiones regulares en PHP

Aquí hay algunos ejemplos de cómo puede usar expresiones regulares en PHP para varios propósitos:

Validación de la entrada del usuario

Un caso de uso común para las expresiones regulares es validar la entrada del usuario antes de procesarla. Por ejemplo, puede usar expresiones regulares para verificar si una dirección de correo electrónico es válida, si una contraseña cumple con ciertos criterios, si un número de teléfono tiene un formato correcto, etc.

Por ejemplo:

```
// Validar una
función de dirección de correo electrónico  validate_email ( $email ) {
  // Definir una expresión regular para el formato de correo electrónico
  $regex = "/^[a-zA-Z0-9._%+-]+@[a-zA-Z0- 9.-]+\.[a-zA-Z]{2,}$/" ;
  // Devuelve verdadero si el correo electrónico coincide con la expresión regula
  devuelve  preg_match ( $regex , $email );
}

// Validar una
función de contraseña  validate_password ( $password ) {
  // Definir una expresión regular para los criterios de contraseña
  // Al menos 8 caracteres
  // Al menos una letra mayúscula
  // Al menos una letra minúscula
  // Al menos un dígito
  // Al menos un carácter especial
  $regex = "/^(?=.*[AZ])(?=.*[az])(?=.*\d)(?=.* [!@#$%^&*]).{8,}$/" ;
  // Devuelve verdadero si la contraseña coincide con la expresión regular, falso
  devuelve  preg_match ( $regex , $contraseña );
} // Validar una función

 de número de teléfono validar_teléfono ( $teléfono ) { // Definir una expresión




  $regex = "/^(\+|00\d{1,3})?[- ]?\d{3}[- ]?\d{7}$/" ;
```

```
  // Devuelve verdadero si el teléfono coincide con la expresión regular, de lo c
  devuelve  preg_match ( $regex , $teléfono );
}
```

Extracción de datos de páginas web

Otro caso de uso común para las expresiones regulares es extraer datos de páginas web u otras fuentes de texto. Por ejemplo, puede usar expresiones regulares para extraer información de etiquetas HTML, analizar datos JSON, buscar URL o direcciones de correo electrónico en texto, etc.

Por ejemplo:

```
// Extrae el título de una
función de página web  extract_title ( $html ) {
  // Define una expresión regular para la etiqueta del título
  $regex = "/<título>(.*?)<\/título>/" ;
  // Intenta hacer coincidir la expresión regular con la cadena HTML
  if ( preg_match ( $regex , $html , $coincidencias )) {
    // Devuelve el primer subpatrón (el texto del título)
    return  $coincidencias [ 1 ];
  } else {
    // Devuelve una cadena vacía si no se encuentra ninguna coincidencia
    return  "" ;
  }
}

// Extraer los valores de una
función de objeto JSON  extract_json_values ( $json ) {
  // Definir una expresión regular para los pares clave-valor en formato JSON
  $regex = "/\"(\w+)\":\s*(\" .*?\"|\d+|verdadero|falso|nulo)/" ;
  // Intenta hacer coincidir la expresión regular con la cadena JSON
  if ( preg_match_all ( $regex , $json , $coincidencias )) {
    // Devuelve una matriz asociativa de claves y valores
    return  array_combine ( $coincidencias [ 1 ], $coincidencias [ 2 ] );
  }else {
    // Devuelve una matriz vacía si no se encuentra ninguna coincidencia
    return [];
  }
}

// Extraer todas las URL de una
función de texto  extract_urls ( $ texto ) {
  // Definir una expresión regular para el formato de URL
```

```
  $regex = "/https?:\/\/[\w\-\.]+(:\ d+)?(\/[\w\-\.\/\?=&%#]*)?/" ;
  // Intenta hacer coincidir la expresión regular con la cadena de texto
  if ( preg_match_all ( $regex , $texto , $coincidencias )) {
    // Devuelve una matriz de URL
    devuelve  $coincidencias [ 0 ];
  }else {
    // Devuelve una matriz vacía si no se encuentra ninguna coincidencia
    return [];
  }
}
```

Sustitución de texto en archivos

Un tercer caso de uso común para las expresiones regulares es reemplazar texto en archivos o cadenas. Por ejemplo, puede usar expresiones regulares para realizar operaciones de búsqueda y reemplazo, dar formato al texto de acuerdo con ciertas reglas, generar texto nuevo a partir de plantillas, etc.

Por ejemplo:

```
// Reemplazar todas las apariciones de "foo" con "bar" en una
función de archivo  replace_foo_with_bar ( $filename ) {
  // Leer el contenido del archivo en una variable de cadena
  $content = file_get_contents ( $filename );
  // Definir una expresión regular para la palabra "foo"
  $regex = "/\bfoo\b/" ;
  // Reemplazar todas las coincidencias con "barra"
  $nuevo_contenido = preg_replace ( $regex , "barra" , $contenido );

  $nuevo_contenido );
}

// Dar formato a una cadena de fecha en formato AAAA-MM-DD
function  format_date ( $fecha ) {
  // Definir una expresión regular para el formato de fecha
  // MM/DD/AAAA o M/D/AAAA
  $regex = "/(\ d{1,2})\/(\d{1,2})\/(\d{4})/" ;
  // Reemplazar las coincidencias con el formato AAAA-MM-DD
  $nueva_fecha = preg_replace ( $regex , " $3 - $1 - $2 " , $fecha );
  // Devuelve la fecha formateada
  return  $new_date ;
}

// Generar un mensaje de saludo a partir de una
```

```
función de plantilla  generar_saludo ( $nombre ) {
  // Definir una expresión regular para el marcador de posición {nombre}
  $regex = "/\{nombre\}/" ;
  // Definir una plantilla para el mensaje de saludo
  $plantilla = "Hola {nombre}, ¡bienvenido a nuestro sitio!" ;
  // Reemplace el marcador de posición con el parámetro de nombre
  $mensaje = preg_replace ( $regex , $name , $template );
  // Devolver el mensaje generado
  return  $mensaje ;
}
```

En esta publicación, hemos aprendido a usar expresiones regulares en PHP para la comparación de patrones y la validación de datos. Hemos cubierto los conceptos básicos de la sintaxis de expresiones regulares, cómo crear y usar objetos de expresiones regulares y algunos ejemplos comunes del uso de expresiones regulares en PHP.

Las expresiones regulares son herramientas muy útiles y versátiles para trabajar con datos de texto. Pueden ayudarlo a realizar varias tareas, como validar la entrada del usuario, extraer datos de páginas web, reemplazar texto en archivos y más.

Sin embargo, las expresiones regulares también pueden ser complejas y difíciles de escribir y depurar. Por lo tanto, es importante probar sus expresiones regulares antes de usarlas en su código. Puede usar herramientas en línea como Regex101 o RegExr para probar y visualizar sus expresiones regulares.

Espero que hayas disfrutado este post y hayas aprendido algo nuevo. Si tiene alguna pregunta o comentario, por favor deje un comentario a continuación. ¡Gracias por leer!