Examinación de strings en PHP

Principales funciones de cadenas de caracteres para buscar en strings y para proporcionar información sobre ellas

**Contenido modificable**

Si ves errores o quieres modificar/añadir contenidos, puedes [crear un pull request](https://github.com/diegotham/certificacion-php/blob/master/3-Strings-and-patterns/5-examinacion-de-strings-en-php.md). Gracias

**Indice de contenido**

|  |  |  |
| --- | --- | --- |
| **Contar strings** | **Buscar strings** | **Posición de strings** |
| 1. count\_chars | 5. strstr | 9. strpos |
| 2. str\_word\_count | 6. stristr | 10. stripos |
| 3. strlen | 7. strchr | 11. strrpos |
| 4. substr\_count | 8. strrchr | 12. strripos |

**Contar strings**

Funciones para obtener el número de caracteres o palabras dentro de un string.

**1. count\_chars**

mixed count\_chars (string $string [, int $mode = 0 ])

Devuelve información sobre los caracteres usados en un string *$string*. Cuenta el número de apariciones de cada byte-value (<255) en el *$string* y lo devuelve de diferentes formas dependiendo de *$mode*.

|  |  |
| --- | --- |
| **Mode** | **Devolución** |
| 0 | Array con el byte-value como clave y la frecuencia de cada uno como valor |
| 1 | Como el anterior pero mostrando sólo los byte-values con frecuencia superior a cero |
| 2 | Como el 0, pero listando los byte-values con frecuencia igual a 0 |
| 3 | Cadena con todos los caracteres únicos |
| 4 | Cadena con todos los caracteres no utilizados |

Ejemplo:

$cadena = "Hola";

$datos = count\_chars($cadena, 1);

foreach ($datos as $key => $value)

{

$key = chr($key);

echo "De este carácter: $key, hay este número: " . $value . "<br>";

}

/\*

\* Devuelve

De este carácter: H, hay este número: 1

De este carácter: a, hay este número: 1

De este carácter: l, hay este número: 1

De este carácter: o, hay este número: 1

\*/

**2. str\_word\_count**

mixed str\_word\_count (string $string [, int $format = 0 [, string $charlist ]] )

Cuenta el número de palabras de un string *$string*. El formato *$format* es opcional:

|  |  |
| --- | --- |
| **Formato** | **Devuelve** |
| 0 | Número de palabras encontradas |
| 1 | Array con las palabras encontradas |
| 2 | Array asociativo, donde la clave es la **posición numérica** (en caracteres) de la palabra dentro del *string* y el valor es la palabra en cuestión |

El *$charlist* es una lista de caracteres adicionales que se quiere que se consideren como palabra:

$str = "La semana esta dividida en 7 dias de 24 horas";

print\_r(str\_word\_count($str, 0, "724")); // Devuelve: 10

print\_r(str\_word\_count($str, 1, "724")); // Devuelve un array numérico

print\_r(str\_word\_count($str, 2, "724")); // Devuelve un array cuyas keys son el nº de carácter

**3. strlen**

int strlen (string $string)

Devuelve la **longitud de un string** *$string* (en número de bytes). Si está vacío devuelve 0.

Si se ejecuta sobre arrays devuelve **null** y emite un **E\_WARNING**.

$string = "Tomas";

echo strlen($string) . "<br>"; // Devuelve: 5

// Para caracteres multibyte: mb\_strlen

$string = "Tomás";

echo mb\_strlen($string) . "<br>"; // Devuelve: 5

**4. substr\_count**

int substr\_count (string $haystack, string $needle [, int $offset = 0 [, int $length ]] )

Cuenta el número de apariciones de un substring, el número de veces que *$needle* aparece en el *$haystack*. El needle es sensible a mayúsculas y minúsculas. No cuenta cadenas que se traslapan:

$texto = 'Esto es un texto';

echo strlen($texto); // 16

echo substr\_count($texto, 'to'); // 2

// el string es reducido a 'es un texto', así que muestra 1:

echo substr\_count($texto, 'to', 5);

// el texto es reducido a 'es un', así que muestra 0:

echo substr\_count($texto, 'to', 5, 4);

// genera una advertencia porque 8+10 > 16:

echo substr\_count($text, 'is', 8, 10);

// muestra sólo 1, no cuenta subcadenas traslapadas:

$texto2 = 'holaholahola';

echo substr\_count($texto2, 'holahola');

**Buscar strings**

Las siguientes funciones buscan elementos dentro de strings.

**5. strstr**

string strstr ( string $haystack, mixed $needle [, bool $before\_needle = false ])

Devuelve la parte del string de *$haystack* desde la primera aparición de *$needle* (incluyéndolo) hasta el final. **Sensible a mayúsculas y minúsculas**.

Si sólo se quiere saber si el *$needle* existe en el *$haystack*, es mejor utilizar *strpos()*, que es más rápida y requiere menos memoria.

Si el *$needle* no es un string se convierte como número entero y se aplica el valor ordinal del carácter.

Si se define \_$before*needle*, lo que devuelve es lo contrario, es decir, la parte del haystack antes de needle, sin incluir el needle.

Si no encuentra el *$needle* devuelve **false**.

$email = "minombre@ejemplo.com";

$dominio = strstr($email, "@");

echo $dominio; // Muestra: @ejemplo.com

$usuario = strstr($email, "@", true);

echo $usuario; // Muestra: minombre

**6. stristr**

string stristr ( string $haystack, mixed $needle [, bool $before\_needle = false])

Es igual que la anterior, pero **no** es sensible a mayúsculas y minúsculas:

$email = "MINOMBRE@GMAIL.COM";

if(stristr($email, '@gmail.com')) {

echo "Tu email es de google";

}

**7. strchr**

\*\*strchr ()\*\*

Alias de *strstr()*.

**8. strrchr**

string strrchr ( string $haystack, mixed $needle)

Encuentra la última aparición de un carácter *$needle* en un string *$haystack*. Devuelve la parte del *$haystack* que comienza en la última aparición de *$needle* y acaba hasta el final de *$haystack*.

Si *$needle* contiene más de un carácter, sólo se utiliza el primero. Si no es un *string*, se convierte en un entero y se aplica como valor ordinal de un carácter.

$directorio = "User/usuario/sitios/miweb";

$carpeta\_web = strrchr($directorio, "/");

echo $carpeta\_web; // Devuelve: /miweb

**Posición de strings**

Funciones que examinan la posición de ciertos caracteres o strings dentro de otros.

**9. strpos**

mixed strpos ( string $haystack, mixed $needle [, int $offset = 0])

Encuentra la posición numérica de la **primera ocurrencia** del *$needle* en el string *$haystack*.

Si el *$needle* no es un string, se convierte en integer y se interpreta como valor ordinal de un carácter.

Si se proporciona un *$offset*, la búsqueda comenzará desde este número de caracteres contados desde el principio del string. No puede ser negativo.

Devuelve la posición donde se encuentra el needle (sin contar el *$offset*). Si no se encuentra, devuelve **false**.

$haystack = "emailusuario@ejemplo.com";

$needle = "@";

$position = strpos($haystack, $needle);

if ($position !== false){

echo "El nombre del email de usuario es: " . substr($haystack, 0, $position) . "<br>";

echo "El nombre de dominio del email es: " . substr($haystack, $position + 1);

} else {

echo "No se ha encontrado el email";

}

En el ejemplo anterior se utiliza !==, no !=, ya que las posiciones empiezan en 0 y puede dar problemas, ya que (0 != false) genera false.

Ejemplo con *$offset*:

$frase = "estaremos hasta esta tarde";

$buscar = "es";

$position = strpos($frase, $buscar, 1);

echo $position; // Devuelve: 16

**10. stripos**

mixed stripos (string $haystack, string $needle [, int $offset = 0 ])

Es igual que *strpos()* pero esta función **no** tiene en cuenta las mayúsculas y minúsculas:

$algo = "No importa usar MAYUSCUlas o minusculas";

$aguja = "mayu";

$position = stripos($algo, $aguja);

var\_dump($position); // Devuelve 16

// Si usáramos strpos() no lo encontraría y devolvería false

**11. strrpos**

int strrpos (string $haystack, string $needle [, int $offset = 0] )

Encuentra la posición numérica de la **última aparición** del *$needle* en el string *$haystack*.

Es similar a *strpos()*: si el *$needle* no es un string, se convierte en integer y se aplica como el valor ordinal de un carácter. Pero el \_$offset*puede ser****negativo****, lo que significa que se contará desde la posición marcada por el offset contando desde el final hasta el principio del*$haystack\_ (como si fuera una búsqueda hacia atrás).

El valor devuelto es el mismo, encuentra la posición del *$needle* que se busca (independientemente del *$offset*).

$abecedario = "abcdefghijklmnñopqrstuvwxyzabcdefghijklmnñopqrstuvwxyz";

$aguja = "g";

var\_dump(strrpos($abecedario, $aguja, 50)); // False

var\_dump(strrpos($abecedario, $aguja, -10)); // Devuelve: 34

$aguja = "t";

var\_dump(strrpos($abecedario, $aguja, 30)); // Devuelve 49

var\_dump(strrpos($abecedario, $aguja, -10)); // Devuelve 21

**12. strripos**

int strripos (string $haystack, string $needle [, int $offset = 0 ])

Es igual a *strrpos()* pero insensible a mayúsculas y minúsculas.

$abecedario = "abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ";

$aguja = "t";

var\_dump(strripos($abecedario, $aguja)); // Devuelve 49

var\_dump(strripos($abecedario, $aguja, -15)); // Devuelve 21

$abecedario = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ";

$aguja = "jk";

var\_dump(strripos($abecedario, $aguja)); // Devuelve 9