**Validate e Sanitize**

Os filtros dividem-se em dois tipos: validate e sanitize. Gosto de traduzi-los como validação e limpeza, respectivamente. O objetivo dos filtros de validação (validate) é verificar se determinado valor corresponde a um padrão previamente especificado. Já os filtros de limpeza (sanitize) irão retirar caracteres que não correspondam ao padrão permitido.

Cada filtro possui uma constante correspondente. No [capítulo de filtros do manual do PHP](http://php.net/filter.filters) é possível encontrar uma lista completa com todos os filtros disponíveis.

A função filter\_var() obedece ao seguinte protótipo:

1. mixed filter\_var ( mixed **$variable** [, int **$filter** = FILTER\_DEFAULT [, mixed **$options** ]] )

Sendo assim, o único parâmetro obrigatório é o primeiro, que indica o valor a ser filtrado. Porém, o filtro padrão é não aplicar filtro nenhum. Dessa forma, pra que ocorra de fato uma filtragem, é necessário especificar o segundo parâmetro.

Basicamente, para aplicarmos um filtro em uma variável precisamos apenas indicar **filter\_var**($valor, CONSTANTE\_DE\_FILTRO).

Os filtros estão organizados em constantes. Validações encontram-se nas constantes FILTER\_VALIDATE\_\* e os filtros de limpeza nas constantes FILTER\_SANITIZE\_\*.

Nos casos de validação, o retorno da função será o valor passado como entrada, em caso de sucesso, e *false* em caso contrário. Já os filtros de limpeza irão retornar o valor após retirados os caracteres impróprios.

**Validando variáveis**

Conforme já mencionei, no manual do PHP existe uma lista com todos os filtros disponíveis. Aqui, apresento uma lista com os filtros que julgo serem mais relevantes e o objetivo de cada um.

|  |  |
| --- | --- |
| **Filtro** | **Objetivo** |
| **FILTER\_VALIDATE\_EMAIL** | Validação de e-mails de acordo com o padrão RFC 822. |
| **FILTER\_VALIDATE\_INT** | Validação de inteiros. Caso seja passada uma string contendo um inteiro, o valor será convertido e validado. |
| **FILTER\_VALIDATE\_IP** | Validação de formato IP. Funciona tanto com IPv4 quanto IPv6. |
| **FILTER\_VALIDATE\_MAC** | Validação de MAC address, endereço físico de placas de rede. |
| **FILTER\_VALIDATE\_URL** | Validação de URLs de acordo com o padrão RFC 2396. |

Separei alguns exemplos de retorno com FILTER\_VALIDATE\_INT e  FILTER\_VALIDATE\_EMAIL.

|  |  |
| --- | --- |
| Entrada | FILTER\_VALIDATE\_INT |
| ‘7’ | int(7) |
| 7 | int(7) |
| 0 | int(0) |
| true | int(1) |
| false | bool(false) |
| ’10 teste’ | bool(false) |
| ‘palavra’ | bool(false) |
| -5 | int(-5) |
| 3.1 | bool(false) |
| +0 | bool(false) |
| -0 | bool(false) |

|  |  |
| --- | --- |
| **Entrada** | **FILTER\_VALIDATE\_EMAIL** |
| “fulano@site.com.br” | string(18) “fulano@site.com.br” |
| 5 | bool(false) |
| “site.com.br” | bool(false) |
| “1@2.3” | bool(false) |
| “1@2” | bool(false) |
| ‘”fulano beltrano”@site.com’ | bool(false) |

**Observação:** Esse último exemplo, embora alguns servidores aceitem, o PHP considera inválido para fins de validação.

**Limpando variáveis**

Quando utilizamos filtros do tipo sanitize, todos os caracteres que não pertecem ao padrão escolhido são retirados do valor passado na entrada. Da mesma forma que fiz com os filtros de validação, preparei uma listagem de alguns filtros sanitize.

|  |  |
| --- | --- |
| **Filtro** | **Objetivo** |
| **FILTER\_SANITIZE\_EMAIL** | Remove todos os caracteres, exceto letras, números e os caracteres !#$%&’\*+-=?^\_`{|}~@.[]. |
| **FILTER\_SANITIZE\_NUMBER\_FLOAT** | Remove todos os caracteres, exceto números, +- e, opcionalmente, .,eE. |
| **FILTER\_SANITIZE\_NUMBER\_INT** | Remove todos os caracteres, exceto números e sinais de positivo e negativo. |
| **FILTER\_SANITIZE\_SPECIAL\_CHARS** | Escapa em formato de entidades HTML os caracteres ‘””<>& e todos os caracteres que possuam valor ASCII menor que 32. Opcionalmente é possível retirar ou converter outros caracteres especiais. |
| **FILTER\_SANITIZE\_STRING** | Retira tags. Opcionalmente é possível retirar ou converter outros caracteres especiais. |
| **FILTER\_SANITIZE\_URL** | Remove todos os caracteres, exceto letras, números e $-\_.+!\*'(),{}|\^~[]`<>#%””;/?:@&=. |

Preparei também alguns exemplos de retorno com FILTER\_SANITIZE\_URL e  FILTER\_SANITIZE\_STRING.

|  |  |
| --- | --- |
| **Entrada** | **FILTER\_SANITIZE\_URL** |
| “[http://phpit.com.br](http://phpit.com.br/)” | string(19) “[http://phpit.com.br](http://phpit.com.br/)” |
| “[http://phpit.com.br£](http://phpit.com.xn--br-cda/)” | string(19) “[http://phpit.com.br](http://phpit.com.br/)” |
| “phpitº.com.br” | string(12) “phpit.com.br” |
| “br¶” | string(2) “br” |
| “®” | string(0) “” |

|  |  |
| --- | --- |
| **Entrada** | **FILTER\_SANITIZE\_STRING** |
| “uma string” | string(10) “uma string” |
| “<tag>uma string \*” | string(12) “uma string \*” |
| “<tag>uma string&lt;tag&gt;” | string(21) “uma string&lt;tag&gt;” |

**Opções extras**

É importante ler a documentação do PHP pois existem algumas maneiras de modificar o comportamento desses filtros de validação. Os filtros de validação podem aceitar flags e opções, enquanto os de limpeza podem aceitar apenas flags. Todos esses atributos estão listados junto às definições de cada filtro no manual.

Você pode utilizar um array de opções e/ou flags, indicados pelos índices *options* e *flags*, respectivamente. Em funções que aceitam apenas flags, você pode optar por passá-las diretamente.

O trecho de código abaixo foi adaptado da página sobre a função **filter\_var()** no manual do PHP. Adicionei alguns comentários para tentar elucidar melhor a maneira de funcionamento das opções.

Das opções, a única que está presente em todas é *default*, que vai indicar o valor a ser retornado caso o dado não seja validado. Fora essa, você encontrará outras opções, como *min\_range*e *max\_range*, que indicam qual o valor mínimo e máximo para que o dado seja validado.

As flags são mais específicas do que as opções. Na validação de inteiros, por exemplo, você pode utilizar a flag  FILTER\_FLAG\_ALLOW\_OCTAL, que irá permitir valores do sistema octal.

1. *// você pode utilizar opções dessa maneira*
2. **$opcoes** = **array**(
3. *// as opções ficam neste índice*
4. 'options' => **array**(
5. *// valor a ser retornado se o filtro falhar*
6. 'default' => 3,
7. *// outras opções vão sendo inseridas aqui*
8. 'min\_range' => 0
9. ),
10. *// as flags ficam neste índice*
11. 'flags' => FILTER\_FLAG\_ALLOW\_OCTAL,
12. );
13. **$var** = filter\_var('0755', FILTER\_VALIDATE\_INT, **$opcoes**);
14. *// para filtros que aceitam apenas flags, você pode passá-las diretamente*
15. **$var** = filter\_var('algum valor', FILTER\_VALIDATE\_BOOLEAN, FILTER\_NULL\_ON\_FAILURE);
16. *// para filtros que aceitam apenas flags, você também pode passar como array*
17. **$var** = filter\_var('algum valor', FILTER\_VALIDATE\_BOOLEAN, **array**('flags' => FILTER\_NULL\_ON\_FAILURE));

Fonte:

<http://www.phpit.com.br/artigos/filtrando-e-validando-dados-no-php-com-filter_var.phpit>

<https://www.php.net/manual/en/filter.filters.sanitize.php>

<https://www.php.net/manual/en/filter.filters.validate.php>