





Accueil > Cours > Concevez votre site web avec PHP et MySQL > Mémento des expressions régulières

# Concevez votre site web avec PHP et MySQL

70 heures Moyenne

Mis à jour le 15/12/2020



# Mémento des expressions régulières

Cette annexe va être utile à ceux qui ont lu les deux chapitres sur les expressions régulières. Il s'agit d'un mémento, c'est-à-dire d'un résumé qui vous sera utile lorsque vous écrirez vos propres regex.

Référez-vous à cette annexe dès que vous vous apprêtez à écrire une expression régulière. Elle vous servira de support pour vous rappeler toutes les possibilités des regex.

Cette annexe n'est PAS faite pour apprendre à se servir des regex. Si vous voulez apprendre, allez voir les chapitres correspondants dans ce cours.

Ici, les explications sont succinctes, car le but est de synthétiser au maximum tout ce qu'il y a à savoir sur les regex.

## Structure d'une regex



Une regex est entourée de symboles appelés *délimiteurs*. On peut choisir ce qu'on veut ; nous, nous utilisons le dièse.

Une regex a la forme suivante :

#Regex#Options

Pour tester une chaîne à partir d'une regex, on utilise preg\_match :

1 <?php preg\_match("regex","chaine"); ?>

php

Le tableau suivant présente une utilisation basique des regex.

regex	Explication
#guitare#	Cherche le mot « guitare » dans la chaîne.
#guitare piano#	Cherche le mot « guitare » OU « piano ».
#^guitare#	La chaîne doit commencer par « guitare ».
#guitare\$#	La chaîne doit se terminer par « guitare ».
#^guitare\$#	La chaîne doit contenir uniquement « guitare ».

#### Classes de caractères



Le tableau qui suit présente le mode d'emploi des classes de caractères.

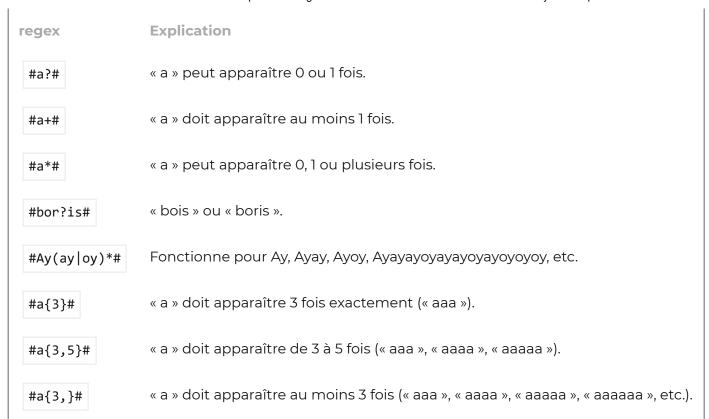
regex	Explication
#gr[ioa]s#	Chaîne qui contient « gris », ou « gros », ou « gras ».
[a-z]	Caractères minuscules de a à z.
[0-9]	Chiffres de 0 à 9.
[a-e0-9]	Lettres de « a » à « e » ou chiffres de 0 à 9.
[0-57A-Za- z]	Chiffres de 0 à 5, ou 7, ou lettres majuscules, ou lettres minuscules, ou un point, ou un tiret.
#[^0-9]#	Chaîne ne contenant PAS de chiffres.
#^[^0-9]#	Chaîne ne commençant PAS par un chiffre.

# **Quantificateurs**



Le tableau ci-après présente les différents quantificateurs qui existent.

regex Explication



## Métacaractères



Les métacaractères sont :

#!^\$()[]{}|?+\*.

Pour utiliser un métacaractère dans une recherche, il faut l'échapper avec un antislash : \ \ .

regex Explication

#Hein?# Cherche « Hei » ou « Hein ».

#Hein\?# Cherche « Hein? ».

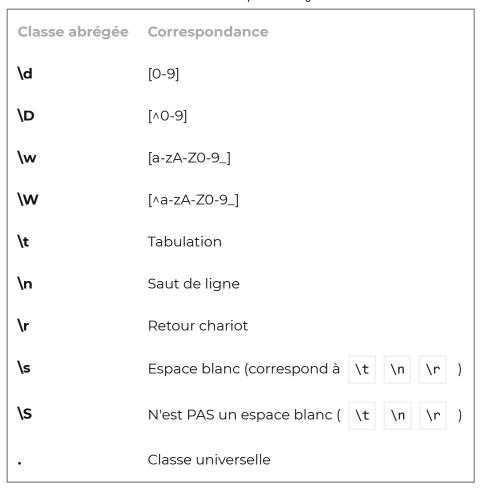
Les métacaractères n'ont pas besoin d'être échappés dans une classe, sauf pour « # » (symbole de fin de la regex), « ] » (symbole de la fin de la classe) et « \ » (si votre classe recherche un antislash), que l'on doit faire précéder d'un antislash.

Si on veut rechercher un tiret dans une classe de caractères, il faut le placer au début ou à la fin de la classe : [a-zA-Z0-9-] .

# Classes abrégées



Les classes abrégées sont supportées uniquement par les regex PCRE.



Le point est la classe universelle : il signifie « n'importe quel caractère ».

# Capture et remplacement



En utilisant la fonction **preg\_replace** , on peut automatiquement faire des remplacements à l'aide de regex.

```
1 <?php
2 $texte = preg_replace('#\[b\](.+)\[/b\]#i', '<strong>$1</strong>', $texte);
3 ?>
```

- Les parenthèses servent à entourer un bout de la regex pour créer des variables \$1 , \$2 ,
   \$3 , etc. qui seront utiles pour faire le remplacement.
- Il peut y avoir jusqu'à 99 parenthèses capturantes, donc jusqu'à \$99 .
- (?:texte) est une parenthèse non capturante : elle ne crée pas de variable.
- Une variable \$0 est toujours créée et correspond à l'ensemble de la regex.

Ainsi, la regex suivante...

```
#(anti)co(?:nsti)(tu(tion)nelle)ment#
```

... crée les variables suivantes :

- **\$0**: anticonstitutionnellement;
- **\$1**: anti;
- \$2: tutionnelle;
- **\$3**: tion.

# **Options**



Il existe de nombreuses options que l'on peut utiliser avec les regex PCRE.

Parmi les trois que nous sommes le plus souvent amenés à utiliser, il y a :

- i: la regex ne fera plus la différence entre majuscules et minuscules ;
- ${f s}$ : le point (classe universelle) fonctionnera aussi pour les retours à la ligne (  $\n$  );
- U : mode « ungreedy » (pas gourmand). Utilisé pour que la regex s'arrête le plus tôt possible.
   Pratique par exemple pour le BBCode [b][/b] : la regex s'arrêtera à la première occurrence de [/b] .

J'AI TERMINÉ CE CHAPITRE

<

PROTÉGEZ UN DOSSIER AVEC UN
.HTACCESS

### Le professeur

#### **Mathieu Nebra**

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

#### Découvrez aussi ce cours en...





Livre

**PDF**