

PHP et Expressions Régulières

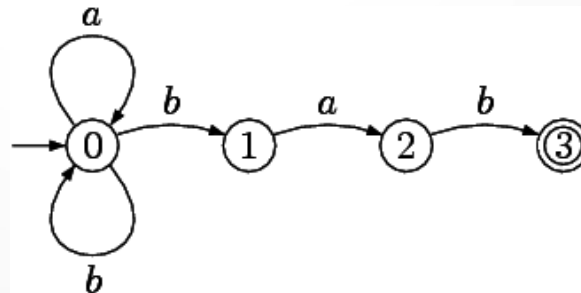


CommitStrip.com

Une gestion des chaînes de caractères rapide, standard, complète et efficace

PHP et Regex: Introduction

- Les expressions régulières (regex) sont représentées par un langage de script
- Ne nécessite aucune installation spéciale car déjà inclus dans PHP et même MYSQL
- Ce langage dérivé d'anciens outils UNIX (sed, grep, awk, ...) est hautement spécialisé dans la manipulation des chaines de caractères d'un texte donné



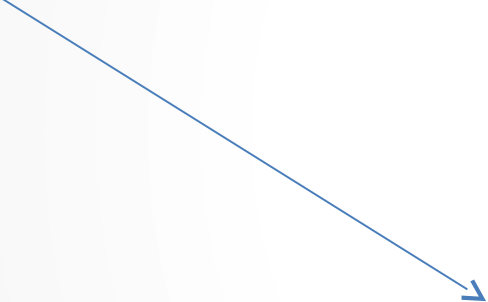
PHP et Regex : Introduction

- Trouver un mot dans une chaîne en PHP pur :
 - \$chaine1 = « guitare »
 - \$chaine2 = « J'aime jouer de la Guitare »
 - if (string stristr (string \$chaine2 , mixed \$chaine1 [, bool \$before_chaine1 = false]))
 - {
 - echo 'VRAI' ;
 - }
 - else
 - {
 - echo 'FAUX' ;
 - }

- Trouver un mot dans une chaîne en PHP/Regex :
 - if (preg_match("#guitare#i", "J'aime jouer de la Guitare."))
 - {
 - echo 'VRAI';
 - }
 - else
 - {
 - echo 'FAUX';
 - }

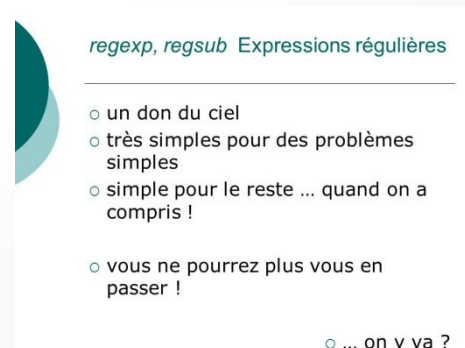
PHP et Regex: A quoi ça sert?

- Il y a deux types de langages Regex:
 - POSIX
 - PCRE

- 
- preg_grep
 - preg_split
 - preg_quote
 - preg_match
 - preg_match
 - . . .

PHP et Regex: A quoi ça sert?

- Un exemple concret de Regex:
 - `preg_match(#guitare#, "J'aime jouer de la guitare")`
- Dans cette fonction:
 - `preg_match()` est la fonction en elle-même prenant 2 arguments
 - `#guitare#` est la regex, avec "guitare" la chaîne recherchée
 - "J'aime..." est la chaîne où on recherche la regex



PHP et Regex: Recherches simples

- `preg_match(#guitare#, « J'aime jouer de la guitare »)`
- On peut noter que « guitare » est entouré de #
- Il s'agit de DELIMITEURS : symboles indiquant précisément la chaîne cible
- Ils peuvent être d'autres symboles :
 - `/guitare/`
 - `#guitare#`
 - `~guitare~`

PHP et Regex: Recherches simples

- Ces recherches sont encore très limitées (sensibles à la casse, pas d'indication de position. . .)
- Voilà pourquoi existent les **OPTIONS**: elles modifient les conditions de recherche de la chaîne ciblée.
- Sont toujours des lettres
- Un exemple d'option:
 - `i` : rend la chaîne insensible à la casse (min/maj)
 - `preg_match(#guitare#i, « J'aime jouer de la . . . »)` sera VRAI avec:
 - « GUITARE »
 - « Guitare »
 - « guitare »
 - . . .

PHP et Regex: Symboles

- Le Regex devient vraiment un langage de script grâce aux SYMBOLES.
- Ce sont des stand-in pour des connecteurs logiques comme OU, NON, ET...
- Sont toujours des symboles comme | , \$, ^ . . .
- Un exemple pour bien comprendre: OU
- `preg_match(#guitare|banjo#, . . .)`
 - « J'aime jouer de la guitare et du banjo. » = VRAI
 - « J'aime jouer de la guitare et du piano. » = VRAI
 - « J'aime jouer du piano. » = FAUX

PHP et Regex: Symboles

- Les symboles les plus importants :
 - \$ = Fin de chaine
 - Recherche si la chaine-cible est au début de la chaine-objet. Exemple:
 - preg_match("#guitare\$i, . . .) :
 - « Guitare, banjo, piano, tamtam » = VRAI
 - « J'aime jouer de la guitare » = VRAI
 - ^ = Début de chaine
 - Recherche si la chaine-cible est au début de la chaine objet. Exemple:
 - preg_match("#^guitare#i, . . .) :
 - « Guitare, banjo, piano, tamtam » = VRAI
 - « J'aime jouer de la guitare » = FAUX

PHP et Regex: Classes

- Les Regex, étant déjà des simplifications, ont aussi leur propres simplifications!
- Le symbole | est utile pour choisir entre plusieurs chaines longues mais si on veut faire une recherche plus précise, entre plus de critères, on peut utiliser une CLASSE.
- Une classe signifie que entre les crochets, chaque lettre peut être une option
- Exemple:
 - `preg_merge(#gras|gris|gros#, . . .)` OU `preg_merge(#gr[aio]s#, . . .)`
 - « Le chat est gris » = VRAI

PHP et Regex: Classes

- Les classes elles-mêmes peuvent être raccourcies!
- Un exemple:
 - Imaginons que l'on souhaite trouver si une phrase contient un chiffre
 - On peut soit utiliser . . .
 - `preg_match('#[0123456789]#', « J'habite dans le 91 »)`
 - Soit. . . :
 - `preg_match('#[0-9]#', « J'habite dans le 91 »)`
- Ce tiret indique une INTERVALLE de classe et indique que la recherche couvre toute les lettre/chiffres entre les limites. Ils peuvent aussi être combinés dans une classe:
 - `preg_match('#[a-zA-Z]#', . . .)`

PHP et Regex: Classes

- Ce tiret indique une INTERVALLE de classe et indique que la recherche couvre toute les lettre/chiffres entre les limites. Ils peuvent aussi être combinés dans une classe:
- Pour chercher si une chaine contient des lettres on peut utiliser:
 - `preg_match('#[a-zA-Z]#', ...)`
 - « 123456 » = FAUX
 - « 12345F » = VRAI
 - « Guitare » = VRAI
- Pour chercher si une chaine NE contient PAS de lettres on peut utiliser:
 - `preg_match('#[^a-zA-Z]#', ...)`

PHP et Regex: Quantificateurs

- Enfin, il est aussi nécessaire de savoir si une sous-chaine se répète plusieurs fois dans une chaine, et si oui combien de fois (par exemple, si on veut avoir que des adresses se terminant par .com que par .fr)
- Dans ce cas là, on utilise les
QUANTIFICATEURS.
- Il s'agit d'une catégorie de symboles qui se mettent derrière une lettre ou un groupe de lettre pour contrôler leur répétition.
 - `*+?`

PHP et Regex: Quantificateurs

- * = la lettre est FACULTATIVE = elle se répète 0 ou n fois
 - preg_match(#a*#, ...):
 - « bbb » = VRAI
 - « aaa » = VRAI
 - « a » = VRAI
- + = la lettre est OBLIGATOIRE = elle se répète 1 ou n fois:
 - preg_match(#a+#, ...):
 - « bbb » = FAUX
 - « aaa » = VRAI
 - « a » = VRAI
- ? = la lettre est FACULTATIVE = elle se répète 1 ou 0 fois
 - preg_match(#a?#, ...):
 - « bbb » = VRAI
 - « aaa » = FAUX
 - « a » = VRAI

PHP et Regex: Quantificateurs

- Enfin, on peut utiliser des accolades couplées aux quantificateurs pour PRECISER EXACTEMENT LE NOMBRE DE REPETITIONS que l'on souhaite contrôler dans notre chaîne-cible:
 - Des exemples:
 - `#a{3}#` = « a » doit être répété exactement trois fois
 - `#a{3, 5}#` = « a » doit être répété de trois à cinq fois
 - `#a{3, }#` = « a » doit être répété au minimum trois fois, sans limite supérieure

PHP et Regex : Conclusion

- On utilise aussi les regex pour des modifications :
 - `preg_replace` ('[#Expression régulière#]', 'valeur de remplacement', 'Endroit ou chercher l'expression')
- Cet ordre de remplacement permet de créer un langage et est utile pour
 - laisser l'utilisateur styliser ses commentaires
 - rendre les urls cliquables.



regexp, regsub Expressions régulières

- un don du ciel
- très simples pour des problèmes simples
- simple pour le reste ... quand on a compris !
- vous ne pourrez plus vous en passer !

○ ... on y va ?

PHP et Regex: Quiz

- Vrai ou Faux : ' Le regex est un langage de script qui permet une manipulation poussée des chaînes de caractères et de nombres.'
- Marier ces symboles avec leur signification : | , ^ , \$, ? , + , *
- Décrivez la chaîne que cible cette regex :
`#^Blip(bloup|blam)+$#`
- La regex `#[abcdefghijklmnopqrstuvwxyz]#` est-elle la seule qui permette de cibler toutes les lettres minuscules de l'alphabet ?
- Quelle est la fonction PCRE qui cherche une sous-chaine dans une autre chaîne ?

PHP et Regex : Exercices

- L'intégralité de ce cours se trouve sur le site d'OpenClassroom, intitulé « Les expressions régulières (1/2) ». N'hésitez pas à l'utiliser.
 - Chercher une sous-chaine dans une chaîne, qu'importe la syntaxe
 - Vérifier si une chaîne se termine, commence par un mot.
 - Vérifier si un numéro de téléphone est valide.
 - Trouver une url et la remplacer par un lien cliquable.