Gestion des chaîne de caractères

Expression régulières

- Langage de recherche et remplacement dans une chaîne de caractère
- Syntaxe barbare mais puissante
- Disponible avec la plupart des langages de programmation et avec la console Linux

Syntaxe

Premier exemple

```
$str = "PHP c'est énorme";

//retourne 1
var_dump(preg_match('/PHP/', $str));

//retourne 0
var_dump(preg_match('/php/', $str));
```

Le schéma est une chaîne encadrée
par des caractères /
(on peut également utiliser le #)

Gestion de la casse

```
$str = "PHP c'est énorme";

//retourne 1
var_dump(preg_match('/php/i', $str));
```

Le paramètre i indique que la recherche ne sera pas sensible à la casse

Recherche sur un mot complet

```
$str = "Je ne fais pas de webdesign";

//retourne 1
var_dump(preg_match('/web/', $str));

//retourne 0
var_dump(preg_match('/web\b/', $str));
```

Le paramètre \b force la recherche sur un mot complet

ou

```
$str = "Java c’est énorme";

//retourne 1
var_dump(preg_match('/php|java/i', $str));
```

Le | permet de tester plusieurs schémas dans la même expression

Début de chaîne

```
$str = "Hello world";

//retourne 1
var_dump(preg_match('/^hello/i', $str));
```

Le ^ indique
un début de chaîne

Fin de chaîne

```
$str = "say hello";

//retourne 1
var_dump(preg_match('/hello$/i', $str));
```

Le $ indique
une fin de chaîne

Classe de caractère

```
$str = "say toto";

//Équivalent à (a|i|o)
$pattern = '/t[aio]t[aio]/i';

//retourne 1
var_dump(preg_match($pattern, $str));
```

Intervalle de classe

[a-z]	Toutes les lettres minuscules non accentuées
[A-Z]	Toutes les lettres majuscules non accentuées
[0-9]	Tous les chiffres
[^0-9]	Tout sauf un chiffre (le caractère ^ devant un crochet est une négation)

Intervalle de classe

```
$str = "P08";

//Une lettre en majuscule
//suivie de deux chiffres
$pattern = '/[A-Z][0-9][0-9]/';

//retourne 1
var_dump(preg_match($pattern, $str));
```

Cardinalités

?	zéro ou une fois
+	une ou plusieurs fois
*	zéro, une ou plusieurs fois
{n}	exactement n fois
{n,}	au moins n fois
{n,m}	au moins n fois au plus m fois

Cardinalités

```
$str = "say toto";

//Équivalent à (a|i|o)
$pattern = '/t[aio]{2}/i';

//retourne 1
var_dump(preg_match($pattern, $str));
```

Regroupements

```
$str = "say toto";

//Équivalent à (a|i|o)
$pattern = '/say (t[aio]){2}/i';

//retourne 1
var_dump(preg_match($pattern, $str));
```

portée de la cardinalité

Échappement des caractères spéciaux

Les caractères suivants :

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

doivent être précédés d'un caractère \

sauf dans une classe de caractères
où seuls les caractères
] # -
sont concernés

Classes abrégées

\d	un chiffre, équivaut à [0-9]
\D	pas un chiffre, équivaut à [^0-9]
\w	un caractères alphanumérique ou un tiret bas,
\W	pas un caractère alphanumérique, équivaut à
\t	une tabulation
\n	une nouvelle ligne
\r	un retour chariot
\s	un espace blanc (espace, tabulation, saut de ligne ou retour chariot)
.	n'importe quel caractère

Exercices

Modèle	String
/^[a-z]$/	abcdef
/^[a-z]*$/	abcdef
/^[a-z]?$/	abcdef
/^[a-z]+$/	abcdef
/[a-z]/	abcdef1
/[a-z]*/	abcdef1
/^[a-z]*$/	aBcdef
/^[a-z]*$/i	aBcdef
/^[a-zA-Z]*$/	aBcdef
/^[a-zA-Z]*$/	aBcdef1
/^[a-zA-Z0-9]*$/	aBcdef
/.com/	abc.com
/[.com]/	abc.com
/[.moc]/	abc.com
/.moc/	abc.com
/cool/	cool
/cool/	coool
/c(o)l/	cool
/c(o)*l/	cool
/c(o)?l/	cool
/c(o){1}l/	cool
/c(o){2}l/	cool
/c(o){3}l/	cool
/c(o){0,2}l/	cool
/(w){3}\.([a-z]+)\.(fr|com)/	www.google.fr
/(w){3}\.([a-z]+)\.(fr|com)/	www.google.com
/(w){3}\.([a-z]+)\.(fr|com)/	www.g1oogle.fr
/(w){3}\.([a-z]+)\.(fr|com)/	www..fr
/(w){3}\.([a-z]+)\.(fr|com)/	google.fr
/(w){3}\.([a-z]+)\.(fr|com)/	ww.google.fr

Modèle	String	Réponse
/^[a-z]$/	abcdef	NON
/^[a-z]*$/	abcdef	OUI
/^[a-z]?$/	abcdef	NON
/^[a-z]+$/	abcdef	OUI
/[a-z]/	abcdef1	OUI
/[a-z]*/	abcdef1	OUI
/^[a-z]*$/	aBcdef	NON
/^[a-z]*$/i	aBcdef	OUI
/^[a-zA-Z]*$/	aBcdef	OUI
/^[a-zA-Z]*$/	aBcdef1	NON
/^[a-zA-Z0-9]*$/	aBcdef	OUI
/.com/	abc.com	OUI
/[.com]/	abc.com	OUI
/[.moc]/	abc.com	OUI
/.moc/	abc.com	NON
/cool/	cool	OUI
/cool/	coool	NON
/c(o)l/	cool	NON
/c(o)*l/	cool	OUI
/c(o)?l/	cool	NON
/c(o){1}l/	cool	NON
/c(o){2}l/	cool	OUI
/c(o){3}l/	cool	NON
/c(o){0,2}l/	cool	OUI
/(w){3}\.([a-z]+)\.(fr|com)/	www.google.fr	OUI
/(w){3}\.([a-z]+)\.(fr|com)/	www.google.com	OUI
/(w){3}\.([a-z]+)\.(fr|com)/	www.g1oogle.fr	NON
/(w){3}\.([a-z]+)\.(fr|com)/	www..fr	NON
/(w){3}\.([a-z]+)\.(fr|com)/	google.fr	NON
/(w){3}\.([a-z]+)\.(fr|com)/	ww.google.fr	NON

Une date mysql

- année, mois et jour séparés par un tiret
- 1 ou 2 suivi de 3 chiffres pour l'année
- Le mois [0-9] ou 10 ou 11 ou 12
- Le jour 0 ou 1 ou 2 suivi d'un chiffre ou 3 suivi de 0 ou 1

année

mois

jour

```
/^(1|2)\d{3}-(0?[0-9]|1[0-2])-([0-2]?[0-9]|3[0-1])/
```

Un numéro de téléphone

- un caractère 0 suivi de [1-9]
- un tiret, un espace, un point ou rien entre chaque groupe de numéro
- 4 fois un groupe de deux chiffres

```
/^0[1-9]([-. ]?[0-9]{2}){4}$/
```

Une adresse IP

- 4 nombres de 0 à 255 séparés par un point

3 fois un nombre de 0 à 255 suivi d'un point

```
/^((([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$/
```

1 fois un nombre de 0 à 255

Adresse email

- N'importe quels caractères alphanumériques ou tiret ou point
- Un caractère @
- N'importe quels caractères alphanumériques ou tiret ou point
- Se termine par un point suivi de 2 à 4 caractères alpha

```
^[a-z0-9._-]+@[a-z0-9._-]{2,}\.[a-z]{2,4}$
```

Pattern avancés

Back reference

- référence un groupe précédemment déclaré

Back reference

```
$str = "a=a";

$pattern = '/(\w+)=\1/i';
```

groupe 1

référence au groupe 1

Balise html

```
$str = "<h1>…</h1>";

$pattern = '#<(h[0-9])>.+</\1>#i';
```

balise <hn>

fermeture de balise

Balise html

```
$str = "<div> <em>test</em></div>";
$pattern = '#<([A-Z][A-Z0-9]*)\b[^>]*>.*</\1>#i';
```

balise <hn>

fermeture de balise

Répétition

```
$str = "mot mot";
$pattern = '#(\w+\b)\s\1#i';
```

Lookaround

- assertion positive ou négative
- ne capture aucun caractère

lookahead positif

```
$str = "<div> <em>test</em></div>";
$pattern = '#^
(?=.*\d)
(?=.*[a-z])
(?=.*[A-Z])
.{4,8}$#';
```

Remplacements

Syntaxe

```
preg_replace($pattern, $replacement, $subject);
```

Remplacement simple

```
$subject = "Il fait chaud";
$pattern = "#\bchaud\b#";
$replacement = "beau";

echo preg_replace(
  $pattern,
  $replacement,
  $subject
);
```

Remplacement avec tableau

```
$subject = "toto";
$pattern = ["t","o"];
$replacement = ["l", "u"];


echo preg_replace(
   $pattern,
   $replacement,
   $subject
);
```

Permutations

```
$subject = "14/08/2015";
$pattern = "#(\d{2})/(\d{2})/(\d{4})#";
$replacement = "$3-$2-$1";
```