Les expressions régulières (regex)

C. BENSARI

Introduction

- Les expressions régulières est un système permettant de vérifier, rechercher ou remplacer des termes dans des chaines de caractères
- Il existe deux types d'expressions régulières :
  - PCRE : issue du langage Perl
  - POSIX : issue de PHP, plus simple que ceux du PCRE mais elles sont plus lentes
- Sous PHP, ils existent plusieurs fonctions qui utilisent la puissance des expressions régulières : preg_match, preg_split, preg_replace, …

Utilisation des Regex

- Une regex doit être écrite entourée de caractères spéciaux appelés délimiteurs (# par exemple, "#ma_regex#")
- Utilisation avec la fonction **preg_match :**

```php
$chaine = "Symfony is a powerfull framework";
echo preg_match("#framework#", $chaine); // affiche 1
echo preg_match("#web#", $chaine); // affiche 0
```

```php
$chaine = "Symfony is a powerfull framework";
// Affichera : le terme 'framework' est bien présent
if (preg_match("#framework#", $chaine)) {
    echo "le terme 'framework' est bien présent";
} else {
    echo "le terme 'framework' n'est pas présent";
}
```

Utilisation des Regex

- Pour ignorer la sensibilité à la casse, utiliser l'option « i » qui doit se trouver après le deuxième « # »

```php
$chaine = "Symfony is a powerfull FrameWorK";
// Affichera : le terme 'framework' est bien présent
if (preg_match("#framework#i", $chaine)) {
    echo "le terme 'framework' est bien présent";
} else {
    echo "le terme 'framework' n'est pas présent";
}
```

- Etendre les possibilités avec l'opérateur logique « | » (ou)

```php
$chaine = "Symfony is a powerfull FrameWorK";
// Affichera : le terme 'framework' est bien présent
if (preg_match("#framework|symfony#i", $chaine)) {
    echo "les termes 'framework' ou 'symfony' sont bien présents";
} else {
    echo "le terme 'framework' n'est pas présent";
}
```

Utilisation des Regex

- Commence par, utilisation du symbole « ^ » en début de regex :

```php
$chaine = "Symfony is a powerfull FrameWorK";
// Affichera : la phrase commence bien par le terme 'symfony'
if (preg_match("#^symfony#i", $chaine)) {
    echo "la phrase commence bien par le terme 'symfony'";
} else {
    echo "la phrase ne commence pas par le terme 'symfony'";
}
```

- Se termine par , utilisation du symbole « $ » en fin de regex :

```php
$chaine = "Symfony is a powerfull FrameWorK";
// Affichera : La phrase ne se termine pas par le mot symfony
if (preg_match("#symfony$#i", $chaine)) {
    echo "la phrase se termine bien par le terme 'symfony'";
} else {
    echo "la phrase ne se termine pas par le terme 'symfony'";
}
```

Utilisation des Regex

Les classes de caractères

- **Classes simples :** utilisation entre crochets à l'intérieur de la regex. Equivalent à **'un caractère parmi les caractères suivants'**

```php
$chaine = "Symfony is a powerfull FrameworK";
// Affichera : PRESENT
if (preg_match("#framewor[kl]$#i", $chaine)) {
    echo "PRESENT";
} else {
    echo "ABSENT";
}
```

$chaine	pattern	affichage
Symfony is a powerfull Framework	#^[abkjf]#	ABSENT
Symfony is a powerfull Framework	#fra[mkl]ework#i	PRESENT
Symfony is a powerfull Framework	#[abfglm]$#	ABSENT

Utilisation des Regex

Les classes de caractères

- **Les intervalles de classe :** même utilisation que les classes simples avec l'introduction du symbole « - » pour représenter un intervalle de caractères

```php
$chaine = "Symfony is a powerfull FrameWorK";
// Affichera : PRESENT
if (preg_match("#framewor[a-m]$#i", $chaine)) {
    echo "PRESENT";
} else {
    echo "ABSENT";
}
```

$chaine	regex	affichage
Symfony is a powerfull Framework	#^[a-g]#	ABSENT
Symfony is a powerfull Framework	#fra[g-o]ework#i	PRESENT
Symfony 4 is a powerfull Framework	#^symfony[2-5]#i	ABSENT

Utilisation des Regex

Les classes de caractères

- **La négation:** pour utiliser la négation (ne contient pas) il suffit d'utiliser le symbôle « ^ », à nouveau ! Mais cette fois-ci, c'est entre les crochets

```php
$chaine = "Symfony 4 is a powerfull FrameWorK";
// Affichera : PRESENT
if (preg_match("#^symfony [^1-3]#i", $chaine)) {
    echo "VRAI";
} else {
    echo "FAUX";
}
```

$chaine	regex	affichage
Symfony is a powerfull Framework	#[^0-9]#	VRAI
Symfony is a powerfull Framework	#fra[^g-o]ework#i	FAUX
Symfony 4 is a powerfull Framework	#^symfony[^2-5]#i	VRAI

Utilisation des Regex

Les quantificateurs

- **Zéro ou une fois :** utiliser le symbole « ? », elle s'applique à la lettre se trouvant directement devant

```php
$chaine = "Symfony 4 is a powerfull FrameWorK";
// Affichera : VRAI
if (preg_match("#power?#", $chaine)) {
    echo "VRAI";
} else {
    echo "FAUX";
}
```

$chaine	regex	affichage
Les classes sont des types communs d'objets	#classes?#	VRAI
Symfony Framework	#<h1>?#	VRAI
Symony 4 is a powerfull Framework	#^symf?ony#i	VRAI

Utilisation des Regex

Les quantificateurs

- **Au moins une fois :** utiliser le symbole « + », elle s'applique à la lettre se trouvant directement devant

$chaine	regex	affichage
Les classes sont des types communs d'objets	#clas+es?#	VRAI
<h1 id='title'>Symfony Framework</h1>	#^<h1>+#	FAUX

- **Zéro ou plusieurs fois :** utiliser le symbole « * », elle s'applique à la lettre se trouvant directement devant

$chaine	regex	affichage
Les classes sont des types communs d'objets	#com*uns#	VRAI
Les interfaces ne sont pas instanciables	#u*#	VRAI

Utilisation des Regex

Les quantificateurs

- On peut donner plus de précisions pour le nombre de répétition. Pour cela, il faut utiliser les accolades :
  - **#a{3}#** fonctionne uniquement pour une chaine contenant un **aaa**
  - **#a{1,3}#** fonctionnera pour une phrase contenant **a**, **aa** ou **aaa**
  - **#a{2,}#** fonctionnera pour **aa**, **aaa**, **aaaa**, etc

$chaine	regex	affichage
GoooGoooGoooo	#(Gooo){1,2}#	VRAI
56482	#^[2-8]{5}$#	VRAI
ayeayeaye	#(aye){2,}#	VRAI

Echappement de caractères

- Pour rechercher les caractères réservés aux expressions régulières, il faut utiliser le symbole « \ »

$chaine	regex	affichage
C++ est un langage orienté objet	#^C\+\+#	VRAI
Une variable de classe ou d'instance ?	#\?$#	VRAI
Voir condition*	#*$#	VRAI
20$	#^[0-9]{2}\$$#	VRAI

- Liste des caractères à échapper : # ^ $ ( ) [ ] { } \ . ? * + ! |
- N.B: sur PHP, pour échapper le « $ » vous devez mettre votre regex entre apostrophes et non pas des doubles guillemets

Echappement de caractères

Cas particuliers

- Pour le cas des classes ([0-9a-z], ..) nous n'avons pas besoin d'échapper les caractères réservés, ainsi, on peut écrire des regex qui ressembles à celle-ci **#[a-z?+*{}]#** qui signifie que c'est accepté d'avoir une lettre, un point d'interrogation, un plus etc..
- Ils existent trois exceptions pour lesquelles on doit échapper avec le **« \ »**
  - Le **« # »** car il indiquera la fin de la regex => utiliser le **« \ »**
  - Le **«] »** car il indiquera la fin de la classe => utiliser le **« \ »**
  - Le « \ » car il indique un caractère d'échappement => utiliser « \\ »
  - Le **« - »** car il indique un intervalle => utiliser le « \ » ou mettre le **« - »** en début ou en fin de la classe : **#[A-Z0-9-]#**

Les classes abrégées

Signe	Signification
\d	Un chiffre
\D	Tous ce qui n'est pas un chiffre
\w	Alphanumérique plus l'underscore « _ »
\W	Ni alphanumérique ni underscore
\t	tabulation
\n	Fin de ligne
\r	Retour chariot
\s	Espace
\S	Tous ce qui n'est pas espace (\t \r \ n)
.	Autorise tout sauf retour à la ligne (ajouter un « s » après la fin de la regex

Exercices :

- Ecrire une REGEX vérifiant le format d'une référence Client

  Format de la référence :

  - Commence par la Lettre F
  - Contient 9 chiffres mais pas de 0
  - Exemple : F526348524, F157523581
- Ecrire une REGEX Permettant de vérifier un numéro de téléphone
- Ecrire une REGEX permettant de vérifier une adresse mail
- Ecrire une REGEX permettant de vérifier un numéro de sécurité sociale : https://www.service-public.fr/particuliers/vosdroits/F33078