Regular Expressions

(regex)

Définition de base

une regex est une chaîne de caractères qui décrit un motif (*pattern*) de texte, pour l'identifier dans des chaînes de caractères et ainsi valider des entrées utilisateur ou effectuer des opérations de recherche et de remplacement de texte

Caractères spéciaux et métacaractères

Caractères d'ancrage ^ et $

Les caractères de début et de fin de chaîne, respectivement ^ et $, représentent, comme leur nom l'indique, le début et la fin de la chaîne.

```
^Bonjour
```

```
revoir$
```

En MySQL, la recherche:

```
SELECT * FROM films WHERE label REGEXP '^[0-9]';
```

Caractères de conjonction

Imaginons maintenant que nous voulons rechercher, dans le texte, les mots Bonjour et revoir, c'est à dire le mot Bonjour OU le mot revoir : nous devrons alors utiliser la barre verticale |.

Ainsi la regex suivante sélectionnera toutes les occurrences de *Bonjour* et *revoir*

```
Bonjour|revoir
```

Il est aussi possible de sélectionner les occurrences du mot *Bonjour* se trouvant au début du texte et du mot *revoir* se trouvant à la fin, ce qui revient à fusionner, en les séparant par une barre verticale, les 2 regex vues dans la sous partie précédente. Voici donc la regex correspondant :

```
^Bonjour|revoir$
```

Classes de caractères: […]

Pour chercher dans notre texte, les mots *mots, mats* et *mits*. Nous pourrions très bien utiliser cette regex:

```
mots|mats|mits
```

Mais il y a plus simple et ce grâce aux ensembles de caractères qui font office, en quelques sortes, de OU en plus courts et plus puissants.

Un ensemble de caractère est délimité par des crochets dans lesquels se trouvent les caractères faisant parti du OU. Ainsi, la regex suivante:

```
m[oai]ts
```

est beaucoup plus succincte que la précédente et sélectionne les mêmes mots. Cette regex peut être explicitée par la phrase suivante : "Sélectionne les parties du texte où il y a un m, suivi d'un o ou d'un a ou d'un i, suivi d'un t, suivi d'un s."

Les ensembles de caractères permettent aussi d'exclure des caractères grâce à l'accent circonflexe ^. La regex suivante:

```
m[^oai]ts
```

sélectionnera, cette fois-ci, seulement le mots *mets* et peut être explicitée par la phrase suivante : "Sélectionne les parties du texte où il y a un m, suivi d'une lettre qui n'est ni o, ni a, ni i, suivi d'un t, suivi d'un s."

Intervalles

Imaginons que nous voulons sélectionner tous les mots commençant par un m, suivi de n'importe quelle lettre, suivi d'un t, suivi d'un s. La regex qui nous viendrait à l'esprit serait une regex de ce type :

```
m[abcdefghijklmnopqrstuvwxyz]ts
```

La regex serait donc longue et fastidieuse à écrire, surtout que pour celle-ci, seules les minuscules ont été sélectionnées ! Heureusement, un moyen plus simple existe pour écrire de telles regex : les intervalles

Intervalle	Equivalent	Traduction
[a-z]	[abcdefghijklmonpqrstuvwxyz]	Lettres minuscules de a à z
[A-Z]	[ABCDEFGHIJKLMNOPQRSTUVWXYZ]	Lettres majuscules de A à Z
[0-9]	[0123456789]	Chiffres de 0 à 9
[a-z0-9]	[abcdefghijklmonpqrstuvwxyz0123456789]	Lettres minuscules de a à z ou chiffres de 0 à 9

Reprenons donc notre regex précédente avec les intervalles:

```
m[a-z]ts
```

Ensembles

Ensemble	Equivalent
.	Absolument n'importe quel charactère
\w	[a-zA-Z0-9_]
\d	[0-9]
\n	Un retour à la ligne
\t	Une tabulation

```
m\wts
```

équivaut à

```
m[a-zA-Z0-9_]ts
```

Quantificateurs: +, *, ?, etc.

Nous venons de voir qu'un ensemble de caractères permet de définir de manière très simple les valeurs possible d'un caractère. Mais qu'en est-il si l'on définir les mêmes valeurs possibles pour plusieurs caractères ? Par exemple, si l'on veut sélectionner les parties du texte où il y a un m, suivi d'un a, suivi de 3 fois n'importe quelle lettre minuscule, est-on obligé d'utiliser une regex de ce type :

```
ma[a-z][a-z][a-z]
```

Il existe une méthode plus simple qui consiste à utiliser les quantificateurs : ce sont des caractères qui indiquent le nombre de répétition du caractère ou de la suite de caractère qui les précèdent. Le quantificateur, dans sa forme explicite, peut s'écrire de 4 façons :

- {min,max} : le nombre de répétition varie entre la valeur minimale et maximale, incluses
- {min,} : le nombre de répétition varie entre la valeur minimale incluse et l'infini
- {,max} : le nombre de répétition varie entre 0 et la valeur maximale incluse
- {nombre} : le nombre de répétition correspond au nombre marqué entre les accolades

```
[a-zA-Z]{6}

[0-9]{2,4}
```

Quantificateur	Traduction	Équivalent
*	0 ou plusieurs répétitions	{0,}
+	1 ou plusieurs répétitions	{1,}
?	0 ou 1 répétition	{,1}

Echappement

Imaginons que l'on veuille y sélectionner les noms de domaine des adresse email *contact@johndoe.fr* et *contact@johndoe.com*. La regex qui nous viendrait donc à la tête serait la suivante :

johndoe.[a-z]{2,3}

johndoe\.[a-z]{2,3}

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

En javascript, l’identification:

```
const str = "La VIE est Belle";
const regex = /[A-Z]+/g;
const matches = str.match(regex);
console.log(matches);
```

En javascript, le remplacement :

```
const str = "Nina est mon chat";
const str = "Ce chien punk s’appelle Capsule";
const str = "Mon papa est un hamster";

const regex = /(chat|chien|hamster)/g;
const resultat = str.replace(regex, "animal domestique");
console.log(resultat);
```

En PHP, la validation:

```
function valider_email($email) {

  $email = trim($email);
  $regex = '/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/';

  if (preg_match($regex, $email)) {
    return true;
  } else {
    return false;
  }
}
```

En MySQL, la substitution:

```
SELECT
  CAST(REGEXP_SUBSTR(`id`, '[0-9]+$', 1) as UNSIGNED) as `id`,
  id as `legacy_id`,
  STR_TO_DATE(datestamp,'%Y-%m-%d %H:%i:%s') as `created_on`,
  urlparms as `slug`,
  tri as `rank`,
  field01 as `label`,
  field03 as `content`,
  field02 as `authors`,
  field04 as `price`,
  field05 as `legacy_lien`,
  field06 as `legacy_photo_illu`

FROM `content_item`
WHERE area = 'ventedvd' AND category = 'livre'
```

Exercices sur les regex simples :

Ecrire des regexes qui correspondent à

1. tous les nombres entiers positifs.
2. tous les mots qui commencent par la lettre "a".
3. toutes les adresses e-mail qui se terminent par ".com".
4. tous les noms de fichiers qui ont une extension de trois lettres.

Exercices sur les regex avec intervalles :

Ecrire des regexes qui correspondent à

1. tous les nombres entre 10 et 99.
2. tous les codes postaux de cinq chiffres.
3. toutes les adresses IP valides.
4. tous les mots qui ont entre 5 et 10 caractères.

Exercices sur les regex avec quantificateurs :

Ecrire des regexes qui correspondent à

1. toutes les séquences de lettres "a" qui ont au moins 3 "a".
2. tous les nombres décimaux qui ont au moins une décimale.
3. tous les mots qui ont une longueur impaire.
4. toutes les chaînes de caractères qui ont au moins un chiffre et au moins une lettre.

Exercices validation:

Ecrire des regexes pour valider

1. les adresses e-mail valides, avec un nom d'utilisateur (commence et termine par une lettre, peut contenir des chiffres et _ et – et .) puis un "@" séparant du nom de domaine une extension de domaine de 2 à 6 caractères.

2. le contenu des mots de passe:
    1. au moins une lettre majuscule et une lettre minuscule
    2. au moins un caractère spécial tel que @, #, $, %, etc.
    3. au moins un chiffre

3. les numéros de téléphone européens, avec
    1. un code de pays optionnel,
    2. un code régional suivi d’un slash optionnel
    3. un numéro de téléphone séparé par des tirets ou des espaces.

4. les noms de fichiers qui ont une extension de fichier images