**Les expressions régulières**

# Définition

## Généralités

Les expressions régulières, ou expressions rationnelles, permettent de définir un pattern de chaîne de caractères.

Elles ont diverses utilités, selon le contexte : recherche d’un pattern au sein d’un texte, contrôle de format, etc.

Bien que ces patterns puissent paraître imbitables, leur compréhension n’est pas hors de portée d’un débutant. Nous verrons à travers ce cours comment analyser et même écrire une expression régulière.

## L’utilisation des expressions régulières

De nos jours, de nombreux langages intègrent les expressions régulières : javascript, php, java, etc. Certaines commandes système, telles que *egrep* ou *awk* s’appuient également sur ce concept.

Certains éditeurs de texte évolués et EDI proposent eux aussi l’utilisation d’expressions régulières, au sein de leur fonction intégrée « rechercher / remplacer ». C’est notamment le cas de Sublime Text ou de Vim.

# L**a syntaxe d’une expression régulière**

Si l’on n’en connaît pas la syntaxe, une expression régulière est indéchiffrable. Par exemple : **[0-9A-F]{2}** correspondant à un octet hexadécimal, ou encore **0\d(\s\d{2}){4}** correspondant à un numéro de téléphone Français.

Mais rassurez-vous, d’ici la fin de ce chapitre, ces expressions vous sembleront beaucoup moins complexes.

## **Pattern d’un caractère**

Une expression régulière définit une séquence de caractères. Ce pattern est défini caractère par caractère.

Par exemple, **\s** représente un caractère de séparation (espace ou tabulation).

Pour définir le pattern d’un caractère, il est possible de lister les caractères utilisables entre crochets. Par exemple, l’expression **[ABC]** fait référence à un caractère pouvant être soit « A », soit « B », soit « C ». On appelle le résultat de cette notation, **classe de caractères**.

Entre ces crochets, il est également possible de définir un ensemble de possibilités plus simplement, en définissant une fourchette. Par exemple **[A-Z]** fait référence à une lettre majuscule, **[a-z]** à une lettre minuscule et **[0-9]** à un chiffre.

Par conséquent, l’expression **[0-9A-F]** fait référence à un caractère qui est un chiffre OU une lettre majuscule comprise entre A et F, incluses (donc, à un digit hexadécimal, n’utilisant que des majuscules).

Il est également possible d’utiliser un pattern d’exclusion, en utilisant le caractère « ^ ». Par exemple, **[^0-9]** fait référence à un caractère non numérique.

Mais il existe un certain nombre de raccourcis, pour les patterns les plus utilisés, dont voici les principaux :

**\w** → **[A-Za-z0-9\_]** : caractère alphanumérique ou « \_ »

**\W** → **[^A-Za-z0-9\_]** : tout caractère ne correspondant pas à \w

**\a** → **[A-Za-z]** : caractère alphabétique

**\s → [\t\v\f]** : caractère d’espacement (en Java, peut correspondre à un retour à la ligne)

**\S** → **[^\t\v\f]** : caractère n’étant pas un caractère d’espacement (en Java, exclut également les retours à la ligne)

**\d** → **[0-9]** : chiffre décimal

**\D** → **[^0-9]** : caractère n’étant pas un chiffre décimal

**.**: un et un seul caractère, quel qu’il soit

Vous pouvez également imposer un caractère précis, au sein du pattern. Pour ce faire, il vous suffit d’inscrire ce caractère. Par exemple au sein d’un pattern, **F** correspond tout simplement au caractère « F ».

## Classe d’équivalence

Il est possible d’utiliser une classe d’équivalence, afin de prendre en compte les caractères accentués, en encadrant le caractère de base de signes **=**. Par exemple, **[=e=]** est équivalent à **[eéèêë]**.

## **Les opérateurs**

Nous avons d’ores et déjà vu les opérateurs **[**, **[^** et **]**, permettant de définir une classe de caractères.

Les opérateurs **(** et **)** permettent de regrouper plusieurs patterns de caractères en un pattern plus long, ce qui peut être très pratique dans le cadre de l’emploi de quantificateurs (voir ci-dessous).

L’opérateur **+** après une expression, permet de la quantifier. Cela signifie que l’expression doit apparaître **au moins une fois**. Par exemple, **(cou)+** fait référence aux chaînes « cou », « coucou », « coucoucou », etc.

L’opérateur **\*** après une expression, permet lui aussi de la quantifier. Cela signifie que l’expression peut apparaître **de 0 à n fois**. Par exemple **oui\*** fait référence aux chaînes « ou », « oui », « ouiiiii », etc.

L’opérateur **?** après une expression, permet lui aussi de la quantifier. Cela signifie que l’expression peut apparaître **de 0 à 1 fois**. Par exemple **oui?** fait référence aux chaînes « ou » et « oui ».

Les opérateurs **{** et **}** permettent de définir un quantificateur précis pour l’expression qui en précède l’usage. L’accolade contient **la quantité attendue**, ou les quantités minimales et maximales séparées par une virgule. Par exemple, **\d{3}** signifie que l’on attend 3 caractères décimaux, **\w{2,5}** signifie que l’on attend de 2 à 5 caractères alphanumériques et **\d{2,}** que l’on attend au moins 2 caractères décimaux.

L’opérateur **|** placé entre deux expressions permet de définir un choix entre deux alternatives. Par exemple **oui|non** fait référence aux chaînes « oui » et « non ».

Enfin, les opérateurs **^** et **$** permettent de préciser que l’on recherche un pattern en **début** ou en **fin** de chaîne de caractères, **^** symbolisant le **début** de la chaîne de caractères et **$** la **fin**. Par exemple, **^\d** désigne une chaîne de caractères **commençant par** un chiffre décimal et **\w$** désigne une chaîne de caractères se **terminant par** un caractère alphanumérique.

En combinant ces deux derniers opérateurs, vous pouvez définir le pattern d’une chaîne de caractères complète. Par exemple, **^0\d(\s\d{2}){4}$** désigne une numéro de téléphone Français, dont les chiffres groupés par deux sont séparés par des espaces ou des tabulations, comme « 06 27 33 49 85 » ou encore « 09 12 38 15 42 ».

# Exercices

Pour l’exercice, nous