**Commandes en Java - RegEx**

Remarque : définition d'une regex en Java :

String maRegEx = "\\w+"; les éléments autres que \b, \t, \n, \f, \r, \", \' et \\ doivent obligatoirement être échappés deux fois !

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbole | Signification | Code | Utilisation | |
| Classes |  |  |  | |
| Pattern | compile une RegEx afin de pouvoir l'utiliser |  |  |  |
| .matches( , ) | méthode permettant d'effectuer une recherche de manière simple et directe | booelan result = Pattern.matches(maRegex, monTexte); | boolean result = Pattern("a.\*", monTexte); |  |
| .compile( ) | prend la RegEx en paramètre pour définir et pouvoir réutiliser le pattern plusieurs fois | Pattern monPattern = Pattern.compile(maRegex);  Matcher m = monPattern.matcher(monTexte);  boolean result = m.matches() |  |  |
| .matcher( ) | effectue la recherche sur la string passée en paramètre |  |  |
| .split( )  .split( , ) | découpe la chaine en plusieurs sous-chaines selon la RegEx et retourne un tableau (limité au nombre d'éléments si le second paramètre est indiqué) | String[] results = monPattern.split(monTexte);  String[4] results = monPattern.split(monTexte, 4); |  | ATTENTION : une fois le nombre limite de sous-chaine atteint, la recherche s'arrête et le tableau est retourné. La dernière string peut donc contenir un ou plusieurs motifs de la RegEx sans avoir été découpée. |
| .flags() | retourne les flags du pattern |  |  |  |
| .pattern() | retourne le pattern |  |  |  |
| .toString() | retourne la RegEx |  |  |  |
| .quote() | retourne la string sur laquelle a été faite la recherche |  |  |  |
|  |  |  |  |  |
| Matcher | permet d'effectuer des traitements plus complexes avec les RegEx |  |  |  |
|  |  | Matcher monMatcher = monPattern.matcher(monTexte); |  |  |
|  | **Sélection de région à tester/analyser** |  |  |  |
| .region( , ) | définit les index entre lesquels le traitement doit être effectué et ignore ce qui précédé et succède | monMatcher.region(55, 340); |  | Rq: comme d'habitude, index de début inclus, index de fin exclus. |
| .regionStart() | retourne l'index de la limite de début du traitement |  |  |  |
| .regionEnd() | retourne l'index de la limite de fin du traitement |  |  |  |
|  |  |  |  |  |
|  | **Méthodes de recherche** |  |  |  |
| .matches() | teste si l'intégralité de la string respecte la RegEx | boolean result = monMatcher.matches(); |  |  |
| .lookingAt() | teste si la string commence par la RegEx | boolean result = monMatcher.lookingAt(); |  |  |
| .find() | recherche les occurrences de la RegEx dans la string  Rq: la recherche s'interrompt chaque fois que le motif est trouvé | while(monMatcher.find()) {  System.out.println("La portion de chaine : " + monMatcher.group() + " correspondant au pattern : " + m.pattern() + " a été trouvé entre les index " + monMatcher.start() + " et " + monMatcher.end());  } | |  |
| .group() | retourne la dernière chaine ayant été trouvée par la méthode .find() |  |
| .start() | retourne l'index du 1er ca. correspondant au motif trouvé lors de la dernière recherche par .find() |  |
| .end() | retourne l'index suivant le dernier ca. correspondant au motif trouvé lors de la dernière recherche par .find() |  |
|  |  |  |  |  |
|  | **Méthodes de remplacement** |  |  |  |
| .replaceAll( ) | remplace toutes les portions correspondantes à la RegEx par la string passée en paramètre | String monTexteModifie = monMatcher.replaceAll(maChaine); | *exemple :*  String txt = "1 - 2 - 3 - 4";  Pattern p = Pattern.compile("-");  Matcher m = p.matcher(txt);  String txtModifie = m.replaceAll(";"); |  |
| .replaceFirst( ) | idem mais seulement pour la première occurrence |  |  |  |
| .appendReplacement( , ) | allié à un StringBuffer, elle permet de remplacer les occurrences correspondantes à la RegEx par des strings variables | **StringBuffer** **sb** = new StringBuffer() ;  int i = 1 ;  while (m.find()) {  monMatcher.appendReplacement(**sb**, "[" + i++ + "]");  }  sb = monMatcher.appendTail(**sb**) ;  String monTexteModifie = sb.toString(); |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Syntaxe | voir JavaDoc - Pattern | RegEx | interprétation | exemple | retour |  |
|  | **bases** |  |  |  |  |  |
| a | représente le caractère a | a | exact. 1 ca. = "a" | "a"  "a "  "abc"  "aa" | true  false  false  false |  |
| . | représente n'importe quel caractère (y compris l'espace) | .  a. | exact. 1 ca.  exact. 2 ca. dont le premier est a |  |  |  |
| ^ | en début de ligne  ATTENTION : signification ≠ quand utilisé dans une classe (voir plus bas) | ^a.\* | commençant par "a" | ""  "a"  "a "  " a"  "abc"  "bac" | false  true  true  false  true  true |  |
| $ | une fin de ligne |  |  |  |  |  |
| | | ou | a|c|j | exact. 1 ca. = "a" ou "c" ou "j" |  |  |  |
|  |  |  |  |  |  |  |
|  | **quantifieurs** |  |  |  |  |  |
| ? ?? | 0 ou une fois | ab?c | 2 ou 3 ca. = "ac" ou "abc" |  |  | Rq: les quantifieurs définissent le mode de recherche :  première forme = **mode greedy**  2e forme (avec le ?) = **mode reluctant** |
| \* \*? | 0 à n fois | .\*  a\* | toute chaine de ca. (vide compris)  de 0 à n ca. avec uniquement "a" | ""  "a"  "aaa"  "aba" | true  true  true  false |
| + +? | au moins 1 fois | a+.\* | commençant par au moins un "a" |  |  |
| {n} {n}? | exactement n fois | .\*a{2}.\* | contenant "aa" (incluant des séries de "a" plus longues) | "abc"  "baac"  "aaaabc" | false  true  true |
| {n,} {n,}? | au moins n fois |  |  |  |  |
| {n, m} {n, m}? | au moins n fois et au plus m fois | .\*a{2, 3}.\* | contenant "aa" ou "aaa" mais excluant des série de "a" plus longues | "abc"  "baac"  "aaaabc" | false  true  false |
|  |  |  |  |  |  |  |
|  | **classes de ca.** |  |  |  |  |  |
| [ ] | représente un seul ca. pouvant prendre plusieurs valeurs | [abc] | exact. 1 ca. = "a" | "b" | "c" |  |  |  |
| ^ | négation | [^abc] | exact. 1 ca. ≠ "a" | "b" | "c" |  |  |  |
| - | inclus tous les ca. entre les 2 bornes | [a-zA-Z]  [a-gmn] | exact. 1 ca. de type lettre en min ou maj  exact. 1 ca. = "entre a et g" | "m" | "n" |  |  |  |
| && | intersection entre deux classes | [a-g&&c-k] | exact. 1 ca. = "entre c et g" |  |  |  |
|  |  |  |  |  |  |  |
|  | **classes prédéfinies** |  |  |  |  |  |
| \d | chiffre | [0-9] |  |  |  |  |
| \D | ca. qui n'est pas un chiffre | [^0-9] |  |  |  |  |
| \s | ca. blanc (espace, tab, retour chariot, …) | [\t\n\x0B\f\r] |  |  |  | Rq:   |  |  | | --- | --- | | \t | tabulation | | \n | retour à la ligne | | \r | retour chariot (= \n\t) | | \f | saut de page | | \x0B | tabulation verticale | |
| \S | ca. qui n'est pas un ca. blanc | [^\s] |  |  |  |  |
| \w | ca. utilisable dans un mot (word) (cad lettres min et maj, chiffres, et \_) | [a-zA-Z\_0-9] |  |  |  | ATTENTION : exclu les caractères accentués |
| \W | ca. de séparation | [^\w] |  |  |  |  |
| \p{javaLowerCase} | ca. minuscule **incluant les ca. accentués** |  |  |  |  | Rq: ces classes utilisent des propriétés de la classe Character de Java.  Rq: toutes propriétés de Character retournant un boolean peut être utilisée de la même manière sous la forme :  isProperty(char) => \p{javaProperty} |
| \p{javaUpperCase} | ca. majuscules **incluant les ca. accentués** |  |  |  |  |
| \p{javaLetter} | ca. de type lettre **incluant les ca. accentués** |  |  |  |  |
| \p{javaWhitespace} | les espaces |  |  |  |  |
| \p{javaMirrored} | ca. écrit en miroir au sens Unicode |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **caractères spéciaux** |  |  |  |  |  |
| \b | le début ou la fin d'un mot | \bun\b  ≠  un  \bune?\b | exact. le mot "un"  ≠  exact. 2 ca. = "un"  exact. les mots "un" ou "une" | **un** livre  **un**e histoire  **un**e t**un**ique | |  |  |  | | --- | --- | --- | | true  false  false | ≠ | true  true  true x2 | |  |
| \B | le début ou la fin d'un élément qui n'est pas un mot |  |  |  |  |  |
| \G | la fin de la chaine qui a été trouvée précédemment |  |  |  |  |  |
| \A | le début d'une entrée |  |  |  |  |  |
| \z | la fin d'une entrée, y compris la fin du texte |  |  |  |  |  |
| \Z | la fin d'une entrée, sauf s'il s'agit de la fin du texte |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |