**Giới thiệu**

**Regular Expression** hay còn gọi là biểu thức chính quy được dùng để xử lý chuỗi nâng cao thông qua biểu thức riêng của nó, những biểu thức này sẽ có những nguyên tắc riêng và bạn phải tuân theo nguyên tắc đó thì biểu thức của bạn mới hoạt động được. Ngoài cái tên gọi Regular Expression ra thì nó còn có thể viết tắt thành **Regex** hay **RegExr**.

Nguyên tắc hoạt động của biểu thức Regex là so khớp dựa vào khuôn mẫu (pattern), khuôn mẫu được xây dựng từ các quy tắc căn bản của biểu thức Regex.

Regex thường được sử dụng để xử lý chuỗi, xử lý văn bản với các tác vụ cụ thể như: tìm và thay thế chuỗi, kiểm tra tính hợp lệ của dữ liệu, trích xuất chuỗi con từ một chuỗi … Hầu hết các ngôn ngữ lập trình đều hỗ trợ Regex như Java, C#, PHP, JS, … Ngoài ra, Regex còn rất phổ biến trong các ứng dụng, công cụ khác nhau như: rewrite URL mod\_rewrite, tìm kiếm và thay thế trong các IDE.

**Các ký tự thường dùng trong Regex**

|  |  |
| --- | --- |
| **Biểu thức chính quy** | **Mô tả** |
| . | Khớp với **bất kỳ ký tự nào**. |
| ^regex | Khớp tại điểm **bắt đầu**. |
| regex$ | Khớp tại điểm **kết thúc**. |
| [abc] | Có thể khớp với **bất kỳ ký tự nào trong cặp dấu []**. Có thể khớp với a hoặc b hoặc c. |
| [abc][xy] | Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c và theo sau là x hay y. |
| [^abc] | Khi dấu ^ xuất hiện như là nhân vật đầu tiên trong dấu ngoặc vuông, nó **phủ định** lại các pattern còn lại. Điều này có thể khớp với bất kỳ ký tự nào ngoại trừ a hoặc b hoặc c. |
| () | Xác định 1 **group** (biểu thức con) xem như nó là một yếu tố đơn lẻ trong pattern .ví dụ ((a(b))c) sẽ khớp với b, ab, abc. |
| [a-d1-7] | **Phạm vi**: phù hợp với một chuỗi giữa a – d và các số từ 1 đến 7. |
| X|Z | Tìm X **hoặc** Z. |
| XZ | Tìm X và theo sau là Z. |
| \d | **Số bất kỳ**, viết ngắn gọn cho **[0-9]** |
| \D | Ký tự **không phải là số**, viết ngắn gon cho **[^0-9]** |
| \s | Ký tự **khoảng trắng**, viết ngắn gọn cho [ \t\n\x0b\r\f] |
| \S | Ký tự **không phải khoản trắng**, viết ngắn gọn cho **[^\s]** |
| \w | Ký tự **chữ**, viết ngắn gọn cho **[a-zA-Z\_0-9]** |
| \W | Ký tự **không phải chữ**, viết ngắn gọn cho **[^\w]** |
| \S+ | Một số ký tự không phải khoảng trắng (một hoặc nhiều). |
| \b | Ký tự thuộc a-z hoặc A-Z hoặc 0-9 hoặc \_, viết ngắn gọn cho **[a-zA-Z0-9\_]**. |
| \* | Xuất hiện 0 hoặc nhiều lần, viết ngắn gọn cho **{0,}**. Ví dụ A\*B khớp với B, AB, AAB |
| + | Xuất hiện 1 hoặc nhiều lần, viết ngắn gọn cho **{1,}**. Ví dụ A+B khớp với AB, AAB |
| ? | Xuất hiện 0 hoặc 1 lần, ? viết ngắn gọn cho **{0,1}**. Ví dụ A?B sẽ khớp với B hay AB. |
| {X} | Xuất hiện **X lần**.  Ví dụ A{2}) khớp đúng với 2 chữ A |
| {X, } | Xuất hiện **X lần trở lên**. Ví dụ A{2,} khớp vói AA, AAA, AAAA … |
| {X,Y} | Xuất hiện trong **khoảng X tới Y lần**. Ví dụ A{2,4} khớp vói AA,AAA,AAAA |
| \*? | \* có nghĩa là xuất hiện **0 hoặc nhiều lần**, thêm ? phía sau nghĩa là **tìm kiếm khớp nhỏ nhất**. |

Một số ký tự đặc biệt trong **Java Regex**: \.[{(\*+?^$|

Những ký tự liệt kê ở trên là các ký tự đặc biệt. Trong Java Regex bạn muốn nó hiểu các ký tự đó theo cách thông thường bạn cần thêm dấu **\** ở phía trước.

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Mẫu regex mô tả một ký tự bất kỳ.  String regex = ".";    // Mẫu regex mô tả ký tự dấu chấm.  String regex = "\\."; |

**Sử dụng Regex**

**1 Sử dụng String.matches(string\_regex )**

Sử dụng **String.matches(string\_regex)** để kiểm tra đối tượng toàn bộ String có khớp với regex hay không.

Cú pháp:

|  |  |
| --- | --- |
| 1 | public boolean matches(String regex) |

Ví dụ

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | String str = "gpcoder";    str.matches("."); // = false: vì khớp với ký tự bất kỳ nhưng không chỉ có một ký tự    str.matches(".\*"); // = true: vì khớp với bất kỳ ký tự nào 0 hoặc nhiều lần    str.matches("^g"); // = false: vì khớp với ký tự bắt đầu là g nhưng không chỉ có một ký tự    str.matches("^g.+"); // = true: vì khớp với ký tự bắt đầu là g và sau đó là ký tự bất kỳ, xuất hiện 1 hoặc nhiều lần.    str.matches("r$"); // = false: vì khớp với ký tự kết thúc là r nhưng không chỉ có một ký tự    str.matches(".{1,6}r$"); // = true: Ký tự bất kỳ xuất hiện 6 lần và kết thúc là r. |

**2 Sử dụng String.split(string\_regex)**

Sử dụng **String.split(string\_regex)** để tách chuỗi thành mảng dựa vào phần so khớp với regex.

Cú pháp:

|  |  |
| --- | --- |
| 1 | public String[] split(String regex) |

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | String str = "Welcome   to gpcoder ";    // Khoảng trắng xuất hiện 1 hoặc nhiều lần.  // Các ký tự khoảng trắng: \t\n\x0b\r\f  // Kết hợp quy tắc: \s và +  String[] arr = str.split("\\s+");  System.out.println(Arrays.toString(arr)); // [Welcome, to, gpcoder]    // Thay thế tất cả các khoảng trắng với ký tự tab.  String newString = str.replaceAll("\\s+", "-");  System.out.println(newString); // Welcome-to-gpcoder- |

**3 Sử dụng java.util.regex.Pattern và java.util.regex.Matcher**

**Pattern** là một đối tượng mẫu, một phiên bản đã được biên dịch của một biểu thức chính quy. Nó không có cấu tử (constructor) public, và chúng ta sẽ sử dụng method tĩnh compile(String) để tạo đối tượng, với tham số là biểu thức chính quy.

**Matcher** là một phương tiện để so khớp chuỗi dữ liệu đầu vào với đối tượng Pattern đã được tạo ra ở trên. Class này không có cấu tử public, và chúng ta lấy đối tượng này thông qua method matcher(String) của đối tượng Pattern. Với tham số đầu vào String là văn bản cần kiểm tra.

**PatternSyntaxException** sẽ bị ném ra một ngoại lệ (excepton) nếu biểu thức chính quy có ngữ pháp không chính xác.

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | String str = "Welcome   to gpcoder";    // Ký tự bất kỳ xuất hiện nhiều lần lần và kết thúc là r  String regex = ".\*r$";    // Tạo đối tượng Pattern thông qua method tĩnh.  Pattern pattern = Pattern.compile(regex);    // Lấy ra đối tượng Matcher  Matcher matcher = pattern.matcher(str);    // Kiểm tra có khơp với regex không  boolean match = matcher.matches();  System.out.println("Match = " + match); // Match = true |

**4 Một số pattern thông dụng**

* Check phone number của Việt Nam:

|  |  |
| --- | --- |
| 1 | (84|0[3|5|7|8|9])+([0-9]{8}) |

* Check UUID:

|  |  |
| --- | --- |
| 1 | [0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12} |

* Check date format (yyyy-mm-dd):

|  |  |
| --- | --- |
| 1 | ([12]\d{3}-(0[1-9]|1[0-2])-(0[1-9]|[12]\d|3[01])) |

* Check chỉ bao gồm chữ cái và số:

|  |  |
| --- | --- |
| 1 | [^A-Za-z0-9]+ |

* Check valid URL:

|  |  |
| --- | --- |
| 1 | ^(http:\/\/www.|https:\/\/www.|http:\/\/|https:\/\/)?[a-z0-9]+([-.]{1}[a-z0-9]+).[a-z]{2,5}(:[0-9]{1,5})?(\/.)?$ |

* Check IP4:

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

* Check IP6:

|  |  |
| --- | --- |
| 1 | (([0-9a-fA-F]{1,4}:){7,7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:((:[0-9a-fA-F]{1,4}){1,6})|:((:[0-9a-fA-F]{1,4}){1,7}|:)|fe80:(:[0-9a-fA-F]{0,4}){0,4}%[0-9a-zA-Z]{1,}|::(ffff(:0{1,4}){0,1}:){0,1}((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9]).){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])|([0-9a-fA-F]{1,4}:){1,4}:((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9]).){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])) |

* Check email:

|  |  |
| --- | --- |
| 1 | (?:[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\.[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+)\*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])\*")@(?:(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]\*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\]) |

Một số online tool các bạn có thể tạo pattern và test:

* <https://regexr.com/>
* <https://www.regextester.com/>

Trên đây là những hướng dẫn cơ bản về biểu thức chính quy (Regular Expression) trong Java. Regex là một trong những phần rất hay và phức tạp của Java, tùy vào yêu cầu thực tế mà bạn vận dụng.