Theory: Shorthands

13 minutes 0 / 5 problems solved

Skip this topic Start practicing

3242 users solved this topic. Latest completion was about 1 hour ago.

There are some sets that are used more often than the other ones: sets for digits, or alphanumeric characters, or whitespace characters (there are quite a lot of whitespace characters, we must say). To make the usage of such sets easier and quicker, there are special shorthands, which are equivalent to certain sets, but have shorter "names".

§1. The list of shorthands

There are several pre-defined shorthands for the commonly used character sets:

- `\d` is any digit, short for `[0-9]`;
- `\s` is a whitespace character (including tab and newline), short for `[ \t\n\x0B\f\r]`;
- `\w` is an alphanumeric character (word), short for `[a-zA-Z_0-9]`;
- `\b` is a word boundary. This one is a bit trickier: it doesn't match any specific character, it rather matches a boundary between an alphanumeric character and a non-alphanumeric character (for example, a whitespace character) or a boundary of the string (the end or the start of it). This way, `"\ba"` matches all words (sequences of alphanumeric characters) starting with "a", `"a\b"` matches all words ending with "a", and `"\ba\b"` matches all separate "a" preceded and followed by non-alphanumeric characters.

There are also counterparts of these shorthands that are equivalent to the restrictive sets, and match everything except for the characters mentioned above:

- `\D` is a non-digit, short for `[^0-9]`;
- `\S` is a non-whitespace character, short for `[^ \t\n\x0B\f\r]`;
- `\W` is a non-alphanumeric character, short for `[^a-zA-Z_0-9]`.
- `\B` is a non-word boundary. It matches the situation opposite to that one of the `\b` shorthand: it finds its match every time whenever there is no "gap" between alphanumeric characters. For example, `"a\B"` matches all words that start with "a".

These shorthands make writing common patterns easier.

Each shorthand has the same first letter as its representation (digit, space, word, boundary). The uppercase characters are used to designate the restrictive shorthands.

§2. Example

Let's consider an example with the listed shorthand. Remember, that in Java we use additional backslash `\` character for escaping.

```
String regex = "\\s\\w\\d\\s";

" A5 ".matches(regex); // true
" 33 ".matches(regex); // true
"\tA4\t".matches(regex); // true, because tabs are whitespace as well

"q18q".matches(regex); // false, 'q' is not a space
" AB ".matches(regex); // false, 'B' is not a digit
" -1 ".matches(regex); // false, '-
' is not an alphanumeric character, but '1' is OK.
```

Here's how boundary shorthand will look in Java code:

Current topic:

Shorthands ···

Topic depends on:

✕ Sets, ranges, alternations ···

Topic is required for:

Regexes in programs ···

Table of contents:

```
String startRegex = "\\bcat"; // matches the part of the word that starts with "ca
t"
String endRegex = "cat\\b"; // matches the part of the word that ends with "cat"
String wholeRegex = "\\bcat\\b"; // matches the whole word "cat"
```

So far, we are not applying them in practice, because we only deal with `matches` method that requires a full string to match the regexp.

If you do not want to use shorthands, we can write the same regex as below:

```
String regex = "[ \\t\\n\\x0B\\f\\r][a-zA-Z_0-9][0-9]
[ \\t\\n\\x0B\\f\\r]";
```

This regex, however, is long and not nearly as readable as the previous one. It also has a lot of character repetitions. You can use the predefined shorthands instead of commonly used sets and ranges to simplify your regexes and make them more readable.

§3. Conclusions

In this lesson, we've learned:

- in the regex language, there are shorthands that are equal to some of the most commonly used sets
- there are shorthands equivalent to non-restrictive sets. Such shorthands are designated by a double backslash and a lowercase letter.
- there are also shorthands with the opposite meaning: they ban the characters that their counterparts match. Such shorthands have an uppercase letter instead of a lowercase.

Report a typo

293 users liked this theory. **6** didn't like it. **What about you?**

Start practicing

Comments (13) Hints (0) Useful links (2) Show discussion