Creating a Regular Expression

Visual Studio 2010

A *regular expression* describes one or more strings to match when you search a body of text. The regular expression serves as character pattern to compare with the text being searched.

Regular expressions are constructed much like arithmetic expressions are created. Small expressions are combined by using a variety of metacharacters and operators to create larger expressions.

The components of a regular expression can be individual characters, sets of characters, ranges of characters, or choices between characters. Components can also be any combination of these components.

Constructing a Regular Expression

You construct a regular expression by putting the various components of the expression between a pair of delimiters. In JScript, the delimiters are a pair of forward slash (/) characters, as shown in the following example.

```
/expression/
```

Examples of Regular Expressions

The following table contains examples of typical regular expressions.

Regular expression	**Description**
`/^\s*$/`	Matches a blank line.
`/\d{2}-\d{5}/`	Matches an ID number that consists of two digits, a hyphen, and an additional five digits.
`/<\s*(\S+)(\s[^>]*)?>[\s\S]*<\s*\/\1\s*>/`	Matches an HTML tag.

Ordinary Characters

The simplest form of a regular expression is a single, ordinary character that is compared with a searched string. For example, the single-character regular expression `A` matches the letter A wherever it appears in the searched string.

The following are some examples of JScript single-character regular expressions.

```
/a/
/7/
/M/
```

You can combine several single characters to form a longer expression. For example, the expression `/the/` matches "the" in the following searched strings: "the", "there", "other", and "over the lazy dog".

No concatenation operator is needed. All that is required is that you put one character after another.

Metacharacters

In addition to ordinary characters, a regular expression can contain *metacharacters*. An example of a metacharacter is `\d`, which matches a digit character.

See Regular Expression Syntax for more information.

Ordinary characters consist of all printable and non-printable characters that are not explicitly designated as metacharacters. This includes all uppercase and lowercase alphabetical characters, all digits, all punctuation marks, and some symbols.

Matching Any Character

The period character (.) matches any single printing or non-printing character in a string, except the newline character (\n). The `/a.c/` regular expression matches "aac", "abc", "acc", "adc", "a1c", "a2c", "a-c", and "a#c".

To match a period (.) that is contained in the searched string, you can precede the period in the expression with a backslash (\) character. The `/filename\.ext/` expression matches "filename.ext".

Lists of Matching Characters

You can create a list of matching characters by enclosing one or more individual characters in brackets [ ].

Any character enclosed in a bracket expression matches only a single character in the position in the regular expression where the bracket expression appears. For example, the `/Chapter [12345]/` expression matches "Chapter 1", "Chapter 2", "Chapter 3", "Chapter 4", and "Chapter 5".

To express the matching characters by using a range instead of the characters themselves, you can use the hyphen (-) character. The `/Chapter [1-5]/` expression is equivalent to `/Chapter [12345]/`.

You can find all the characters that are not in the list or range by including a caret (^) character at the start of the list. For example, the `/[^aAeEiIoOuU]/` expression matches any non-vowel character.

For more information, see Lists of Matching Characters.

Quantifiers

You can use *quantifiers* to specify a regular expression in which a character or set of characters is repeated a specific number of times.

A quantifier refers to the expression immediately previous (to the left of) the quantifier.

Quantifiers are enclosed in braces {}, and include number values for the lower and upper occurrence limits. For example, `c{1,2}` matches 1 or 2 occurrences of the letter c.

When only one number is specified, it is used as the upper bound unless it is followed by a comma. For example, `c{3}` matches exactly 3 characters of the letter c, and `c{5,}` matches 5 or more occurrences of the letter c.

Single-character quantifiers are also available, as shown in the following table.

Quantifier	**Explicit quantifier**	**Meaning**
`*`	`{0,}`	Matches the previous element zero or more times.
`+`	`{1,}`	Matches the previous element one or more times.
`?`	`{0,1}`	Matches the previous element zero or one time.

The following are some sample expressions together with search strings that they match.

Regular expression	**Meaning of quantifier**	**Matches**
`/Chapter [1-9][0-9]{0,}/` or `/Chapter [1-9][0-9]*/`	Matches `[0-9]` zero or more times.	"Chapter 1", "Chapter 25", "Chapter 401320"
`/Chapter [0-9]{1,2}/`	Matches `[0-9]` one or two times.	"Chapter 0", "Chapter 03", "Chapter 1", "Chapter 25", "Chapter 40"
`/Chapter [1-9][0-9]{0,1}/` or `/Chapter [1-9][0-9]?/`	Matches `[0-9]` zero or one time.	"Chapter 1", "Chapter 25", "Chapter 40"

For more information, see Quantifiers in JScript.

Line and Word Boundaries

Anchors enable you to specify that a regular expression must appear either at the start or end of the searched string, or the start or end of a line or word in the searched string, to be a match. For more information, see Anchors.

Specifying Alternatives

The "|" character specifies that two or more alternatives represent a match. For example, the JScript regular expression `/(Chapter|Section) [1-9]/` matches the following: "Chapter 1", "Chapter 9", and "Section 2". For more information, see Alternation and Subexpressions.

Using Submatches

Parentheses are used in a regular expression to create a subexpression. The resulting submatch can be retrieved by the program. For more information, see Alternation and Subexpressions.

You can refer to a subexpression from within a regular expression, and from within a replacement string. For more information, see Backreferences in JScript.

See Also

Concepts
Regular Expression Programming
Regular Expression Syntax

Community Additions