SCALA - REGULAR EXPRESSIONS

http://www.tutorialspoint.com/scala/scala_regular_expressions.htm

This chapter explains how Scala supports regular expressions through **Regex** class available in the scala.util.matching package.

Try the following example program where we will try to find out word **Scala** from a statement.

Example

```
import scala.util.matching.Regex

object Demo {
   def main(args: Array[String]) {
      val pattern = "Scala".r
      val str = "Scala is Scalable and cool"

      println(pattern findFirstIn str)
   }
}
```

Save the above program in **Demo.scala**. The following commands are used to compile and execute this program.

Command

```
\>scalac Demo.scala
\>scala Demo
```

Output

```
Some(Scala)
```

We create a String and call the **r** method on it. Scala implicitly converts the String to a RichString and invokes that method to get an instance of Regex. To find a first match of the regular expression, simply call the **findFirstIn** method. If instead of finding only the first occurrence we would like to find all occurrences of the matching word, we can use the **findAllIn** method and in case there are multiple Scala words available in the target string, this will return a collection of all matching words.

You can make use of the mkString method to concatenate the resulting list and you can use a pipe | to search small and capital case of Scala and you can use **Regex** constructor instead or **r** method to create a pattern.

Try the following example program.

Example

```
import scala.util.matching.Regex

object Demo {
   def main(args: Array[String]) {
      val pattern = new Regex("(S|s)cala")
      val str = "Scala is scalable and cool"
      
      println((pattern findAllIn str).mkString(","))
   }
}
```

Save the above program in **Demo.scala**. The following commands are used to compile and execute this program.

Command

```
\>scalac Demo.scala
\>scala Demo
```

Output

```
Scala,scala
```

If you would like to replace matching text, we can use **replaceFirstIn** to replace the first match or **replaceAllIn** to replace all occurrences.

Example

```
object Demo {
   def main(args: Array[String]) {
      val pattern = "(S|s)cala".r
      val str = "Scala is scalable and cool"
      
      println(pattern replaceFirstIn(str, "Java"))
   }
}
```

Save the above program in **Demo.scala**. The following commands are used to compile and execute this program.

Command

```
\>scalac Demo.scala
\>scala Demo
```

Output

```
Java is scalable and cool
```

Forming Regular Expressions

Scala inherits its regular expression syntax from Java, which in turn inherits most of the features of Perl. Here are just some examples that should be enough as refreshers –

Following is the table listing down all the regular expression Meta character syntax available in Java.

Subexpression	Matches
^	Matches beginning of line.
$	Matches end of line.
.	Matches any single character except newline. Using m option allows it to match newline as well.
[...]	Matches any single character in brackets.
[^...]	Matches any single character not in brackets
\\A	Beginning of entire string
\\z	End of entire string
\\Z	End of entire string except allowable final line terminator.
re*	Matches 0 or more occurrences of preceding expression.
re+	Matches 1 or more of the previous thing
re?	Matches 0 or 1 occurrence of preceding expression.
re{ n}	Matches exactly n number of occurrences of preceding expression.
re{ n,}	Matches n or more occurrences of preceding expression.
re{ n, m}	Matches at least n and at most m occurrences of preceding expression.
a|b	Matches either a or b.
re	Groups regular expressions and remembers matched text.
$? : re$	Groups regular expressions without remembering matched text.
$? > re$	Matches independent pattern without backtracking.

\\w	Matches word characters.
\\W	Matches nonword characters.
\\s	Matches whitespace. Equivalent to [\t\n\r\f].
\\S	Matches nonwhitespace.
\\d	Matches digits. Equivalent to [0-9].
\\D	Matches nondigits.
\\A	Matches beginning of string.
\\Z	Matches end of string. If a newline exists, it matches just before newline.
\\z	Matches end of string.
\\G	Matches point where last match finished.
\\n	Back-reference to capture group number "n"
\\b	Matches word boundaries when outside brackets. Matches backspace $0x08$ when inside brackets.
\\B	Matches nonword boundaries.
\\n, \\t, etc.	Matches newlines, carriage returns, tabs, etc.
\\Q	Escape $quote$ all characters up to \\E
\\E	Ends quoting begun with \\Q

Regular-Expression Examples

Example	Description
.	Match any character except newline
[Rr]uby	Match "Ruby" or "ruby"
rub[ye]	Match "ruby" or "rube"
[aeiou]	Match any one lowercase vowel
[0-9]	Match any digit; same as [0123456789]

[a-z]	Match any lowercase ASCII letter
[A-Z]	Match any uppercase ASCII letter
[a-zA-Z0-9]	Match any of the above
[^aeiou]	Match anything other than a lowercase vowel
[^0-9]	Match anything other than a digit
\\d	Match a digit: [0-9]
\\D	Match a nondigit: [^0-9]
\\s	Match a whitespace character: [\t\r\n\f]
\\S	Match nonwhitespace: [^ \t\r\n\f]
\\w	Match a single word character: [A-Za-z0-9_]
\\W	Match a nonword character: [^A-Za-z0-9_]
ruby?	Match "rub" or "ruby": the y is optional
ruby*	Match "rub" plus 0 or more ys
ruby+	Match "rub" plus 1 or more ys
\\d{3}	Match exactly 3 digits
\\d{3,}	Match 3 or more digits
\\d{3,5}	Match 3, 4, or 5 digits
\\D\\d+	No group: + repeats \\d
D+/ d	Grouped: + repeats \\D\d pair
$[Rr]uby($,?)+	Match "Ruby", "Ruby, ruby, ruby", etc.

Note – that every backslash appears twice in the string above. This is because in Java and Scala a single backslash is an escape character in a string literal, not a regular character that shows up in the string. So instead of '\', you need to write '\\' to get a single backslash in the string.

Try the following example program.

Example

```
import scala.util.matching.Regex

object Demo {
   def main(args: Array[String]) {
      val pattern = new Regex("abl[ae]\\d+")
      val str = "ablaw is able1 and cool"

      println((pattern findAllIn str).mkString(","))
   }
}
```

Save the above program in **Demo.scala**. The following commands are used to compile and execute this program.

Command

```
\>scalac Demo.scala
\>scala Demo
```

Output

```
able1
```