How to split a string in Java

Asked 9 years, 7 months ago Active 1 month ago Viewed 3.8m times

1608

I have a string, `"004-034556"`, that I want to split into two strings:

```
string1="004";
string2="034556";
```

388

That means the first string will contain the characters before `'-'`, and the second string will contain the characters after `'-'`. I also want to check if the string has `'-'` in it. If not, I will throw an exception. How can I do this?

java string split

edited Feb 29 at 22:58 — Marco, 410 3 10

asked Aug 14 '10 at 3:01 — riyana, 18k 10 29 30

34 Answers

Active | Oldest | Votes

1 2 Next

2872

Just use the appropriate method: `String#split()`.

```
String string = "004-034556";
String[] parts = string.split("-");
String part1 = parts[0]; // 004
String part2 = parts[1]; // 034556
```

+250

Note that this takes a regular expression, so remember to escape special characters if necessary.

> there are 12 characters with special meanings: the backslash `\`, the caret `^`, the dollar sign `$`, the period or dot `.`, the vertical bar or pipe symbol `|`, the question mark `?`, the asterisk or star `*`, the plus sign `+`, the opening parenthesis `(`, the closing parenthesis `)`, and the opening square bracket `[`, the opening curly brace `{`, These special characters are often called "metacharacters".

So, if you want to split on e.g. period/dot `.` which means "any character" in regex, use either backslash `\` to escape the individual special character like so `split("\\.")`, or use character class `[]` to represent literal character(s) like so `split("[.]")`, or use `Pattern#quote()` to escape the entire string like so `split(Pattern.quote("."))`.

```
String[] parts = string.split(Pattern.quote(".")); // Split on period.
```

To test beforehand if the string contains certain character(s), just use `String#contains()`.

```
if (string.contains("-")) {
    // Split it.
} else {
    throw new IllegalArgumentException("String " + string + " does not contain -
");
}
```

Note, this does not take a regular expression. For that, use `String#matches()` instead.

If you'd like to retain the split character in the resulting parts, then make use of positive lookaround. In case you want to have the split character to end up in left hand side, use positive lookbehind by prefixing `?<=` group on the pattern.

```
String string = "004-034556";
String[] parts = string.split("(?<=-)");
String part1 = parts[0]; // 004-
String part2 = parts[1]; // 034556
```

In case you want to have the split character to end up in right hand side, use positive lookahead by prefixing `?=` group on the pattern.

```
String string = "004-034556";
String[] parts = string.split("(?=-)");
String part1 = parts[0]; // 004
String part2 = parts[1]; // -034556
```

If you'd like to limit the number of resulting parts, then you can supply the desired number as 2nd argument of `split()` method.

```
String string = "004-034556-42";
String[] parts = string.split("-", 2);
String part1 = parts[0]; // 004
String part2 = parts[1]; // 034556-42
```

edited Aug 2 '17 at 12:34

shanethehat
14.9k 9 51 84

answered Aug 14 '10 at 3:05

BalusC
922k 323 3347 3356

27 Why do you use hash symbols to delimit String's methods? – Crowie Aug 1 '13 at 8:56

94 @Crowie: javadoc-style. – BalusC Aug 1 '13 at 12:04

8 Corner case: if it cannot find `reugalr expression` it returns one element array with whole string. – klimat May 23 '16 at 12:36

2 Cann't believe the most voted version is like this. 1) part2 is not what the poster want if the original string contains two "-" 2) No error handling as mentioned in the question. 3) Low efficienty. A single character search needs regular expression construction and matching. Extra array created, etc. – David Jan 17 '19 at 23:38

1 @Angela: the `|` is a special character in regex. As explained in the answer, you need to escape it if you want to interpret it literally. You can use backslash or character class or `Pattern#quote()` for this. So here are three examples which suit your specific case: `string.split("\\|")[2]`, `string.split("[|]")[2]` and `string.split(Pattern.quote("|"))[2]`. Just pick the one you think is best for you. – BalusC Oct 1 '19 at 16:34

76

An alternative to processing the string directly would be to use a regular expression with capturing groups. This has the advantage that it makes it straightforward to imply more sophisticated constraints on the input. For example, the following splits the string into two parts, and ensures that both consist only of digits:

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

class SplitExample
{
    private static Pattern twopart = Pattern.compile("(\\d+)-(\\d+)");

    public static void checkString(String s)
    {
        Matcher m = twopart.matcher(s);
        if (m.matches()) {
            System.out.println(s + " matches; first part is " + m.group(1) +
                               ", second part is " + m.group(2) + ".");
        } else {
            System.out.println(s + " does not match.");
        }
    }

    public static void main(String[] args) {
        checkString("123-4567");
        checkString("foo-bar");
        checkString("123-");
        checkString("-4567");
        checkString("123-4567-890");
    }
}
```

As the pattern is fixed in this instance, it can be compiled in advance and stored as a static member (initialised at class load time in the example). The regular expression is:

```
(\d+)-(\d+)
```

The parentheses denote the capturing groups; the string that matched that part of the regexp can be accessed by the Match.group() method, as shown. The \d matches and single decimal digit, and the + means "match one or more of the previous expression). The - has no special meaning, so just matches that character in the input. Note that you need to double-escape the backslashes when writing this as a Java string. Some other examples:

```
([A-Z]+)-([A-Z]+)          // Each part consists of only capital letters
([^-]+)-([^-]+)            // Each part consists of characters other than -
([A-Z]{2})-(\d+)           // The first part is exactly two capital letters,
                           // the second consists of digits
```

edited Jul 17 '17 at 16:09

answered Aug 14 '10 at 11:28

Rob Hague
1,229 8 12

1 Thanks. Looking at docs.oracle.com/javase/7/docs/api/java/util/regex/..., you're right — in line with most other regexp libraries, group 0 is the full match, and the captured groups start at 1. As you say, I suspect that this may have changed since I originally wrote the answer, but in any case I'll update it to reflect current behaviour. – Rob Hague Jul 17 '17 at 16:09

42

```
String[] result = yourString.split("-");
if (result.length != 2)
     throw new IllegalArgumentException("String not in correct format");
```

This will split your string into 2 parts. The first element in the array will be the part containing the stuff before the `-`, and the 2nd element in the array will contain the part of your string after the `-`.

If the array length is not 2, then the string was not in the format: `string-string`.

Check out the `split()` method in the `String` class.

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-int-

edited Jan 5 '16 at 7:36

rjdkolb
6,571 7 53 69

answered Aug 14 '10 at 3:06

jjnguy
122k 48 277 315

5 This will accept "-555" as input and returns [, 555]. The requirements aren't defined that clear, if it would be valid to accept this. I recommend writing some unit-tests to define the desired behaviour. – Michael Konietzka Aug 14 '10 at 6:36

29

```
String[] out = string.split("-");
```

should do thing you want. String class has many method to operate with string.

answered Aug 14 '10 at 3:06

secmask
5,627 3 28 49

28

```
// This leaves the regexes issue out of question
// But we must remember that each character in the Delimiter String is treated
// like a single delimiter

public static String[] SplitUsingTokenizer(String subject, String delimiters) {
   StringTokenizer strTkn = new StringTokenizer(subject, delimiters);
   ArrayList<String> arrLis = new ArrayList<String>(subject.length());

   while(strTkn.hasMoreTokens())
      arrLis.add(strTkn.nextToken());

   return arrLis.toArray(new String[0]);
}
```

edited Apr 23 '14 at 20:45

Nicolas
205 4 9

answered Nov 16 '12 at 6:30

Mnyikka
1,043 14 11

60 The JavaDoc clearly states: "*`StringTokenizer` is a legacy class that is retained for compatibility reasons although* ***its use is discouraged in new code****. It is recommended that anyone seeking this functionality use the `split` method of `String` or the `java.util.regex` package instead."* – bvdb Sep 9 '13 at 7:07

22

With Java 8:

```
List<String> stringList = Pattern.compile("-")
        .splitAsStream("004-034556")
        .collect(Collectors.toList());

stringList.forEach(s -> System.out.println(s));
```

edited May 21 '17 at 11:07

Peter Mortensen
25.3k 21 90 118

answered Dec 1 '16 at 9:32

Somaiah Kumbera
5,148 2 28 36

2 If you want to remove whitespace add `.map(String::trim)` after the `split` – Roland Mar 10 '17 at 15:11

18

The requirements left room for interpretation. I recommend writing a method,

```
public final static String[] mySplit(final String s)
```

which encapsulate this function. Of course you can use String.split(..) as mentioned in the other answers for the implementation.

You should write some unit-tests for input strings and the desired results and behaviour.

Good test candidates should include:

```
- "0022-3333"
- "-"
- "5555-"
- "-333"
- "3344-"
- "--"
- ""
- "553535"
- "333-333-33"
- "222--222"
- "222--"
- "--4555"
```

With defining the according test results, you can specify the behaviour.

For example, if `"-333"` should return in `[,333]` or if it is an error. Can `"333-333-33"` be separated in `[333,333-33] or [333-333,33]` or is it an error? And so on.

edited Mar 21 '15 at 8:49

Peter Mortensen
25.3k 21 90 118

answered Aug 14 '10 at 6:57

Michael Konietzka
4,923 2 21 29

4 Useful advice, but not actually an answer to the question. If you're supporting another answer's with detail a comment is preferred. – Chris Mountford Aug 24 '14 at 22:43

You can try like this also

16

```
String concatenated_String="hi^Hello";

String split_string_array[]=concatenated_String.split("\\^");
```

answered Jan 15 '13 at 9:58

SHUNMUGA RAJ PRABAKARAN

552 4 8

16

Assuming, that

- you don't really need regular expressions for your split
- you happen to already use apache commons lang in your app

The easiest way is to use StringUtils#split(java.lang.String, char). That's more convenient than the one provided by Java out of the box if you don't need regular expressions. Like its manual says, it works like this:

```
A null input String returns null.

StringUtils.split(null, *)         = null
StringUtils.split("", *)           = []
StringUtils.split("a.b.c", '.')    = ["a", "b", "c"]
StringUtils.split("a..b.c", '.')   = ["a", "b", "c"]
StringUtils.split("a:b:c", '.')    = ["a:b:c"]
StringUtils.split("a b c", ' ')    = ["a", "b", "c"]
```

I would recommend using commong-lang, since usually it contains a lot of stuff that's usable. However, if you don't need it for anything else than doing a split, then implementing yourself or escaping the regex is a better option.

answered Mar 25 '14 at 6:43

eis

40.3k 10 113 161

15

Use org.apache.commons.lang.StringUtils' split method which can split strings based on the character or string you want to split.

Method signature:

```
public static String[] split(String str, char separatorChar);
```

In your case, you want to split a string when there is a "-".

You can simply do as follows:

```
String str = "004-034556";

String split[] = StringUtils.split(str,"-");
```

Output:

```
004
034556
```

Assume that if `-` does not exists in your string, it returns the given string, and you will not get any exception.

edited Jun 25 '15 at 9:44

Lu55
15.3k 4 54 51

answered Jul 1 '14 at 4:35

sandeep vanama
551 7 8

14

To summarize: there are at least five ways to split a string in Java:

1. String.split():

```
String[] parts ="10,20".split(",");
```

2. Pattern.compile(regexp).splitAsStream(input):

```
List<String> strings = Pattern.compile("\\|")
      .splitAsStream("010|020202")
      .collect(Collectors.toList());
```

3. StringTokenizer (legacy class):

```
StringTokenizer strings = new StringTokenizer("Welcome to EXPLAINJAVA.COM!",
".");
while(strings.hasMoreTokens()){
    String substring = strings.nextToken();
    System.out.println(substring);
}
```

4. Google Guava Splitter:

```
Iterable<String> result = Splitter.on(",").split("1,2,3,4");
```

5. Apache Commons StringUtils:

```
String[] strings = StringUtils.split("1,2,3,4", ",");
```

So you can choose the best option for you depending on what you need, e.g. return type (array, list, or iterable).

Here is a big overview of these methods and the most common examples (how to split by dot, slash, question mark, etc.)

edited Feb 6 '18 at 15:10

Peter Mortensen
25.3k 21 90 118

answered Dec 13 '17 at 14:20

Dmytro Shvechikov
440 6 13

13

The fastest way, which also consumes the least resource could be:

```
String s = "abc-def";
int p = s.indexOf('-');
```

```
if (p >= 0) {
    String left = s.substring(0, p);
    String right = s.substring(p + 1);
} else {
  // s does not contain '-'
}
```

answered Mar 20 '14 at 4:37

David
429 5 7

6 The most scarce resource is often programmer's time and attention. This code consumes more of that resource than alternatives. – Chris Mountford Aug 24 '14 at 22:45

1 To do a simple split on a single character with error checking, this is no more complex than the regex version. – tekHedd Jan 16 '19 at 21:13

13

String Split with multiple characters using Regex

```
public class StringSplitTest {
     public static void main(String args[]) {
        String s = " ;String; String; String; String, String;
String;;String;String; String; String; ;String;String;String;String";
        //String[] strs = s.split("[,\\s\\;]");
        String[] strs = s.split("[,\\;]");
        System.out.println("Substrings length:"+strs.length);
        for (int i=0; i < strs.length; i++) {
            System.out.println("Str["+i+"]:"+strs[i]);
        }
     }
  }
```

Output:

```
Substrings length:17
Str[0]:
Str[1]:String
Str[2]: String
Str[3]: String
Str[4]: String
Str[5]: String
Str[6]: String
Str[7]:
Str[8]:String
Str[9]:String
Str[10]: String
Str[11]: String
Str[12]:
Str[13]:String
Str[14]:String
Str[15]:String
Str[16]:String
```

But do not expect the same output across all JDK versions. I have seen one bug which exists in some JDK versions where the first null string has been ignored. This bug is not present in the latest JDK version, but it exists in some versions between JDK 1.7 late versions and 1.8 early versions.

edited Jul 2 '16 at 18:58 answered Dec 2 '15 at 11:07


Peter Mortensen
25.3k 21 90 118


Ravindra babu
36.9k 6 192 163

13

For simple use cases `String.split()` should do the job. If you use guava, there is also a Splitter class which allows chaining of different string operations and supports CharMatcher:

```
Splitter.on('-')
        .trimResults()
        .omitEmptyStrings()
        .split(string);
```

edited Feb 15 '18 at 15:28


Iulian Popescu
2,184 4 17 26

answered May 13 '15 at 13:38


Vitalii Fedorenko
87k 21 139 110

10

```
public class SplitTest {

    public static String[] split(String text, String delimiter) {
        java.util.List<String> parts = new java.util.ArrayList<String>();

        text += delimiter;

        for (int i = text.indexOf(delimiter), j=0; i != -1;) {
            String temp = text.substring(j,i);
            if(temp.trim().length() != 0) {
                parts.add(temp);
            }
            j = i + delimiter.length();
            i = text.indexOf(delimiter,j);
        }

        return parts.toArray(new String[0]);
    }


    public static void main(String[] args) {
        String str = "004-034556";
        String delimiter = "-";
        String result[] = split(str, delimiter);
        for(String s:result)
            System.out.println(s);
    }
}
```

edited Aug 29 '17 at 9:29

answered Mar 15 '14 at 18:17


Akhilesh Dhar Dubey
1,961 2 23 35

9

You can split a string by a line break by using the following statement:

```
String textStr[] = yourString.split("\\r?\\n");
```

You can split a string by a hyphen/character by using the following statement:

```
String textStr[] = yourString.split("-");
```

edited Nov 29 '14 at 12:19

Peter Mortensen
25.3k 21 90 118

answered Sep 1 '14 at 13:39

RajeshVijayakumar
9,027 11 50 76

9

```
import java.io.*;

public class BreakString {

  public static void main(String args[]) {

    String string = "004-034556-1234-2341";
    String[] parts = string.split("-");

    for(int i=0;i<parts.length;i++)  {
      System.out.println(parts[i]);
    }
  }
}
```

edited Nov 17 '16 at 10:00

L Joey
115 2 7

answered Oct 2 '16 at 3:31

Ravi Pandey
415 4 5

4 if i may share advice, how your answer brings more value than the already accepted solution? stackoverflow.com/a/3481842/420096 on such situations you may cast vote on the existing solution, specially if this is a clear trivial case like that one. – Sombriks Oct 2 '16 at 3:51

8

You can use Split():

```
import java.io.*;

public class Splitting
{

    public static void main(String args[])
    {
        String Str = new String("004-034556");
        String[] SplittoArray = Str.split("-");
        String string1 = SplittoArray[0];
        String string2 = SplittoArray[1];
    }
}
```

Else, you can use StringTokenizer:

```
import java.util.*;
public class Splitting
{
    public static void main(String[] args)
    {
        StringTokenizer Str = new StringTokenizer("004-034556");
        String string1 = Str.nextToken("-");
        String string2 = Str.nextToken("-");
    }
}
```

edited Feb 6 '18 at 15:03

Peter Mortensen
25.3k 21 90 118

answered Jan 9 '17 at 17:28

Sarat Chandra
3,070 18 23

8

There are only two methods you really need to consider.

Use String.split for a one-character delimiter or you don't care about performance

If performance is not an issue, or if the delimiter is a single character that is not a regular expression special character (i.e., not one of `.$|()[{^?*+\`) then you can use `String.split` .

```
String[] results = input.split(",");
```

The split method has an optimization to avoid using a regular expression if the delimeter is a single character and not in the above list. Otherwise, it has to compile a regular expression, and this is not ideal.

Use Pattern.split and precompile the pattern if using a complex delimiter and you care about performance.

If performance is an issue, and your delimiter is not one of the above, you should pre-compile a regular expression pattern which you can then reuse.

```
// Save this somewhere
Pattern pattern = Pattern.compile("[,;:]");

/// ... later
String[] results = pattern.split(input);
```

This last option still creates a new `Matcher` object. You can also cache this object and reset it for each input for maximum performance, but that is somewhat more complicated and not thread-safe.

edited Sep 8 '19 at 16:58

Peter Mortensen
25.3k 21 90 118

answered Nov 20 '18 at 12:08

rghome
6,310 7 30 48

7

One way to do this is to run through the String in a for-each loop and use the required split character.

```
public class StringSplitTest {

    public static void main(String[] arg){
        String str = "004-034556";
        String split[] = str.split("-");
        System.out.println("The split parts of the String are");
        for(String s:split)
        System.out.println(s);
    }
}
```

Output:

```
The split parts of the String are:
004
034556
```

edited Oct 4 '15 at 18:49

Glorfindel
18.4k 11 56 81

answered Oct 4 '15 at 18:24

Keshav Pradeep Ramanath
1,124 3 18 27

7

Please don't use StringTokenizer class as it is a legacy class that is retained for compatibility reasons, and its use is discouraged in new code. And we can make use of the split method as suggested by others as well.

```
String[] sampleTokens = "004-034556".split("-");
System.out.println(Arrays.toString(sampleTokens));
```

And as expected it will print:

```
[004, 034556]
```

In this answer I also want to point out **one change that has taken place for `split` method in Java 8**. The String#split() method makes use of `Pattern.split`, and now it will remove empty strings at the start of the result array. Notice this change in documentation for Java 8:

> When there is a positive-width match at the beginning of the input sequence then an empty leading substring is included at the beginning of the resulting array. A zero-width match at the beginning however never produces such empty leading substring.

It means for the following example:

```
String[] sampleTokensAgain = "004".split("");
System.out.println(Arrays.toString(sampleTokensAgain));
```

we will get three strings: `[0, 0, 4]` and not four as was the case in Java 7 and before. Also check this similar question.

edited Jul 2 '16 at 18:59

Peter Mortensen
25.3k 21 90 118

answered May 18 '16 at 5:17

akhil_mittal
16.1k 7 72 69

7

Here are two ways two achieve it.

WAY 1: As you have to split two numbers by a special character you can use regex

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class TrialClass
{
    public static void main(String[] args)
    {
```

```
        Pattern p = Pattern.compile("[0-9]+");
        Matcher m = p.matcher("004-034556");

        while(m.find())
        {
            System.out.println(m.group());
        }
    }
}
```

WAY 2: Using the string split method

```
public class TrialClass
{
    public static void main(String[] args)
    {
        String temp = "004-034556";
        String [] arrString = temp.split("-");
        for(String splitString:arrString)
        {
            System.out.println(splitString);
        }
    }
}
```

edited Feb 6 '18 at 15:05

Peter Mortensen
25.3k 21 90 118

answered Mar 3 '17 at 9:39

Akshay Gaikwad
330 4 12

6

You can simply use StringTokenizer to split a string in two or more parts whether there are any type of delimiters:

```
StringTokenizer st = new StringTokenizer("004-034556", "-");
while(st.hasMoreTokens())
{
    System.out.println(st.nextToken());
}
```

edited Feb 6 '18 at 15:06

Peter Mortensen
25.3k 21 90 118

answered Apr 17 '17 at 3:53

Rohit-Pandey
1,359 12 19

4

Check out the `split()` method in the `String` class on javadoc.

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#split(java.lang.String)

```
String data = "004-034556-1212-232-232";
int cnt = 1;
for (String item : data.split("-")) {
        System.out.println("string "+cnt+" = "+item);
        cnt++;
}
```

Here many examples for split string but I little code optimized.

edited Jul 22 '16 at 10:04

RubioRic
2,264 4 20 31

answered May 9 '16 at 13:36

Divyesh Kanzariya
2,521 2 33 34

4

```
String str="004-034556"
String[] sTemp=str.split("-");// '-' is a delimiter

string1=004 // sTemp[0];
string2=034556//sTemp[1];
```

answered Nov 20 '16 at 4:43

Po.Bestie
1,070 14 21

2

I just wanted to write an algorithm instead of using Java built-in functions:

```
public static List<String> split(String str, char c){
    List<String> list = new ArrayList<>();
    StringBuilder sb = new StringBuilder();

    for (int i = 0; i < str.length(); i++){
        if(str.charAt(i) != c){
            sb.append(str.charAt(i));
        }
        else{
            if(sb.length() > 0){
                list.add(sb.toString());
                sb = new StringBuilder();
            }
        }
    }

    if(sb.length() >0){
        list.add(sb.toString());
    }
    return list;
}
```

edited Feb 6 '18 at 15:11

Peter Mortensen
25.3k 21 90 118

answered Jan 10 '18 at 6:28

None
1

1

You can use the method `split` :

```
public class Demo {
    public static void main(String args[]) {
        String str = "004-034556";

        if ((str.contains("-"))) {
            String[] temp = str.split("-");
            for (String part:temp) {
                System.out.println(part);
            }
        }
        else {
            System.out.println(str + " does not contain \"-\".");
        }
```

```
    }
}
```

edited Sep 8 '19 at 16:54

Peter Mortensen
25.3k 21 90 118

answered Dec 7 '18 at 8:48

Jamith
491 2 10 21

1

To split a string, uses String.split(regex). Review the following examples:

```
String data = "004-034556";
String[] output = data.split("-");
System.out.println(output[0]);
System.out.println(output[1]);
```

Output

```
004
034556
```

Note:

This split (regex) takes a regex as an argument. Remember to escape the regex special characters, like period/dot.

edited Sep 8 '19 at 16:59

Peter Mortensen
25.3k 21 90 118

answered Mar 8 '18 at 14:28

KIBOU Hassan
235 3 13

0

```
String s="004-034556";
for(int i=0;i<s.length();i++)
{
    if(s.charAt(i)=='-')
    {
        System.out.println(s.substring(0,i));
        System.out.println(s.substring(i+1));
    }
}
```

As mentioned by everyone, split() is the best option which may be used in your case. An alternative method can be using substring().

answered Feb 24 '17 at 10:12

SAM Jr
34 4

0

To split a string, use `String.split(regex)`:

```
String phone = "004-034556";
String[] output = phone.split("-");
System.out.println(output[0]);
System.out.println(output[1]);
```

Output:

```
004
034556
```

edited Feb 6 '18 at 15:06

Peter Mortensen
25.3k 21 90 118

answered Apr 7 '17 at 21:57

KIBOU Hassan
235 3 13

1 2 Next

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.