

# Class `java.util.StringTokenizer`

```
java.lang.Object
|
+----java.util.StringTokenizer
```

---

public class **StringTokenizer**  
extends [Object](#)  
implements [Enumeration](#)

`StringTokenizer` is a class that controls simple linear tokenization of a `String`. The set of delimiters, which defaults to common whitespace characters, may be specified at creation time or on a per-token basis.

Example usage:

```
String s = "this is a test";
StringTokenizer st = new StringTokenizer(s);
while (st.hasMoreTokens()) {
    println(st.nextToken());
}
```

Prints the following on the console:

```
this
is
a
test
```

**Version:**  
1.13, 08/10/95

---

## Constructor Index

- o **[StringTokenizer](#)**(`String`, `String`, `boolean`)  
Constructs a `StringTokenizer` on the specified `String`, using the specified delimiter set.
- o **[StringTokenizer](#)**(`String`, `String`)  
Constructs a `StringTokenizer` on the specified `String`, using the specified delimiter set.

- o **StringTokenizer**(String)

Constructs a StringTokenizer on the specified String, using the default delimiter set (which is " \t\n\r").

## Method Index

- o **countTokens**()

Returns the next number of tokens in the String using the current delimiter set.

- o **hasMoreElements**()

Returns true if the Enumeration has more elements.

- o **hasMoreTokens**()

Returns true if more tokens exist.

- o **nextElement**()

Returns the next element in the Enumeration.

- o **nextToken**()

Returns the next token of the String.

- o **nextToken**(String)

Returns the next token, after switching to the new delimiter set.

## Constructors

- o **StringTokenizer**

```
public StringTokenizer(String str,  
                     String delim,  
                     boolean returnTokens)
```

Constructs a StringTokenizer on the specified String, using the specified delimiter set.

**Parameters:**

str – the input String

delim – the delimiter String

returnTokens – returns delimiters as tokens or skip them

- o **StringTokenizer**

```
public StringTokenizer(String str,  
                     String delim)
```

Constructs a StringTokenizer on the specified String, using the specified delimiter set.

**Parameters:**

str – the input String

delim – the delimiter String

- o **StringTokenizer**

```
public StringTokenizer(String str)
```

Constructs a StringTokenizer on the specified String, using the default delimiter set (which is " \t\n\r").

**Parameters:**

str – the String

## Methods

### o hasMoreTokens

```
public boolean hasMoreTokens()
```

Returns true if more tokens exist.

### o nextToken

```
public String nextToken()
```

Returns the next token of the String.

**Throws:** NoSuchElementException

If there are no more tokens in the String.

### o nextToken

```
public String nextToken(String delim)
```

Returns the next token, after switching to the new delimiter set. The new delimiter set remains the default after this call.

**Parameters:**

delim – the new delimiters

### o hasMoreElements

```
public boolean hasMoreElements()
```

Returns true if the Enumeration has more elements.

### o nextElement

```
public Object nextElement()
```

Returns the next element in the Enumeration.

**Throws:** NoSuchElementException

If there are no more elements in the enumeration.

### o countTokens

```
public int countTokens()
```

Returns the next number of tokens in the String using the current delimiter set. This is the number of times `nextToken()` can return before it will generate an exception. Use of this routine to count the number of tokens is faster than repeatedly calling `nextToken()` because the substrings are not constructed and returned for each token.

---

[All Packages](#)[This Package](#)[Previous](#)[Next](#)