# StringTokenizer in Java

**1. What is StringTokenizer?**

* StringTokenizer is a **utility class** in java.util package.
* It is used to **break (tokenize)** a string into **smaller pieces (tokens)** based on **delimiters** (separators like space, comma, etc.).
* It is similar to split() method but older (introduced in **JDK 1.0**).

👉 Think of it like cutting a sentence into words.  
Example:  
"Java is Powerfull" → Tokens: ["Java", "is", "Powerfull"].

**2. When to Use StringTokenizer?**

* When you need to **process strings** piece by piece (like words, CSV values, sentences).
* When **memory is limited** (since it doesn’t create arrays like split() does).
* When you are working on **older Java versions** (before Java 1.4).
* When parsing **input data from files, command line, or user input**.

**3. How StringTokenizer Works?**

* It **iterates through a string** and **extracts tokens** separated by delimiters.
* Default delimiter = **whitespace (space, tab, newline)**.
* You can define your own delimiter (e.g., comma ,, semicolon ;, colon :).

**4. Constructors of StringTokenizer**

1. StringTokenizer(String str)  
   → Uses whitespace as delimiter.
2. StringTokenizer(String str, String delimiter)  
   → Uses given delimiter.
3. StringTokenizer(String str, String delimiter, boolean returnDelimiters)  
   → If true, delimiters are also returned as tokens.

**5. Important Methods**

| **Method** | **Description** |
| --- | --- |
| hasMoreTokens() | Returns true if more tokens are available. |
| nextToken() | Returns the next token. |
| countTokens() | Returns the number of remaining tokens. |
| hasMoreElements() | Same as hasMoreTokens() (because it implements Enumeration). |
| nextElement() | Same as nextToken(). |

**6. Code Examples**

import java.util.StringTokenizer;  
  
public class STokenizer {  
  
 *// Demonstration of Java StringTokenizer* public static void main(String[] args) {  
  
 *// Input string* String s = "Hello Astinill how are you";  
  
 *// Create a StringTokenizer object  
 // with space as the delimiter* StringTokenizer st = new StringTokenizer(s, " ");  
  
 *// Tokenize the string and print each token* while (st.hasMoreTokens()) {  
 System.*out*.println(st.nextToken());  
 }  
  
 *// Demonstration of String Tokenizer Constructors  
 // Example with Constructor 1* System.*out*.println("Using StringTokenizer Constructor 1: ");  
  
 *// Using StringTokenizer to split the string into  
 // tokens using space (" ") as the delimiter* StringTokenizer st1 = new StringTokenizer(  
 "Welcome Astinil Technologies", " ");  
  
 *// Iterate through tokens while  
 // there are more tokens available* while (st1.hasMoreTokens())  
  
 *// Getting and printing the next token* System.*out*.println(st1.nextToken());  
  
 *// Example with Constructor 2* System.*out*.println("Using StringTokenizer Constructor 2: ");  
  
 *// Using StringTokenizer to split the string  
 // using ":" as the delimiter* StringTokenizer st2 = new StringTokenizer(  
 "java : Code : String : Tokenizer", " :");  
  
 *// Iterate through tokens and print them* while (st2.hasMoreTokens())  
 System.*out*.println(st2.nextToken());  
  
 *// Example with Constructor 3* System.*out*.println("Using StringTokenizer Constructor 3: ");  
  
 *// Using StringTokenizer with returnDelims = true  
 // to include delimiters as tokens* StringTokenizer st3 = new StringTokenizer(  
 "java : Code", " :", true);  
  
 *// Iterate through tokens (including delimiters)  
 // and print them* while (st3.hasMoreTokens())  
 System.*out*.println(st3.nextToken());  
  
 *// Demonstration of StringTokenizer Methods  
 // Creating a StringTokenizer* StringTokenizer st4 = new StringTokenizer(  
 "Welcome to Astinil Technologies");  
  
 StringTokenizer st5 = new StringTokenizer("");  
  
 *// countTokens Method* int c = st4.countTokens();  
 System.*out*.println(c);  
  
 *// hasMoreTokens Methods* System.*out*.println("Welcome to Astinil Technology: "+ st4.hasMoreTokens());  
 System.*out*.println("(Empty String) : "+ st5.hasMoreTokens());  
  
 *// nextElement() Method* System.*out*.println("\nTraversing the String:");  
  
 while(st4.hasMoreTokens()){  
 System.*out*.println(st4.nextElement());  
 }  
  
  
 }  
}

Output:

Hello

Astinill

how

are

you

Using StringTokenizer Constructor 1:

Welcome

Astinil

Technologies

Using StringTokenizer Constructor 2:

java

Code

String

Tokenizer

Using StringTokenizer Constructor 3:

java

:

Code

4

Welcome to Astinil Technology: true

(Empty String) : false

Traversing the String:

Welcome

to

Astinil

Technologies

**7. Where is it Used in Real-Time?**

* **Banking Applications**: Parsing customer data like Name, AccountNo, Balance.
* **CSV Files**: Splitting rows into columns.
* **Log Analysis**: Splitting logs like "2025-09-23 10:22:33 ERROR Connection failed".
* **Chatbots**: Splitting user input into commands.
* **Compiler Design**: Splitting code into tokens (keywords, identifiers, operators)

8. **Block Diagram of StringTokenizer:**

nextToken()

nextToken()

"50000"

nextToken()

"Prashanth"

"101"

Delimiter: ","

StringTokenizer

**9. Difference: StringTokenizer vs split()**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Feature | StringTokenizer | split() | | Introduced | JDK 1.0 | JDK 1.4 | | Returns | Tokens (iterative) | Array of strings | | Delimiter | Single character set | Regex | | Performance | Faster (no regex) | Flexible (supports regex) | | Usage Today | Legacy (still works) | Preferred | |

## ****Summary****

* StringTokenizer is used to **split strings into tokens**.
* Best for **simple, fast parsing** (no regex needed).
* Used in **older codebases**, log parsing, banking systems, and text processing.
* Modern replacement: **String.split()** or **Scanner**.