El Uso de StringTokenizer

La clase java.util.StringTokenizer le permite dividir una cadena en tokens.
Nos proporciona la posibilidad de diferenciar números, cadenas entre comillas, identificadores, etc., como la clase StreamTokenizer.
Hay 3 tipos de constructores definidos en la clase StringTokenizer.

- StringTokenizer(String str): Genera un StringTokenizer con una cadena específica.
- StringTokenizer(String str, String delim): Genera un StringTokenizer con una cadena y un delimitador específico.
- StringTokenizer(String str, String delim, boolean returnValue): Genera un StringTokenizer con la cadena especificada, delimitador y returnValue. Si el valor de retorno es verdadero (True), los caracteres delimitadores se consideran tokens. Si es falso (False), los caracteres delimitadores sirven para separar tokens.

Métodos de la clase StringTokenizer:

Los 6 métodos útiles de la clase StringTokenizer son los siguientes:

boolean hasMoreTokens(): Comprueba si hay más tokens disponibles.
String nextToken(): Devuelve el siguiente token del objeto StringTokenizer.
String nextToken(String delim): Devuelve el siguiente token según el delimitador.
boolean hasMoreElements(): Hace lo mismo que el método hasMoreTokens().
Object nextElement(): Igual que nextToken() pero su tipo de retorno es Object.
int countTokens(): Devuelve el número total de tokens.

Ejemplos de cómo usar los métodos de StringTokenizer:

Tokenizar una cadena "Esto es una prueba" sobre la base de espacios en blanco.

```
String prueba="Esto es una prueba";

StringTokenizer st = new StringTokenizer(prueba);
while (st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}
```

```
Esto
es
una
prueba
```

y te sale tokenizado por cada espacio que esté en la frase.
Otro ejemplo que es similar, pero esta vez con las comas,(aunque con los espacios se hacen igual)

```
StringTokenizer st2 =  new  StringTokenizer ( "Esto,es,una,prueba" , "," );
while  (st2.hasMoreTokens ()) {
System.out.println (st2.nextToken ());  }
```

```
Esto
es
una
prueba
```

El resultado es el mismo, y te suprime las comas

El último ejemplo sería en localizar el primer token (aunque también puedes ponerlo en cualquier sitio) que esté en una frase.

```
StringTokenizer st3 =  new  StringTokenizer ( "Esto,es,una,prueba" );

// imprimiendo el siguiente token
System.out.println ( "El siguiente token es: "  + st3.nextToken ( "," ));
```

y el programa te regresará el token que esté en la frase.

```
El siguiente token es: Esto
```

Actualmente la clase StringTokenizer está obsoleta y se recomienda utilizar el método split() de la clase String o regex (Regular Expression o Expresión Regular).

Split()

Mientras programamos, podemos necesitar romper una cadena basada en algunos atributos. En su mayoría, este atributo será un separador o un elemento común con el que desea dividir o dividir la cadena.

Método de cadena Split() divide una cadena en una matriz de subcadenas con un delimitador específico.

Parámetros:

Regex: la expresión regular se aplica al texto / cadena
Límite (limit): un límite es un número máximo de valores de la matriz. Si se omite o es cero,

Resultado:
Devolverá todas las cadenas que coincidan con una expresión regular.

Emite:
PatternSyntaxException: Si la sintaxis de la expresión regular proporcionada no es válida.

Ejemplos del Uso del Split

Queremos separar esta frase unidos por comas:
Lunes,Martes,Miércoles,Jueves,Viernes,Sábado,Domingo.
para eso usamos esta fórmula:

```
String strMain = "Lunes,Martes,Miércoles,Jueves,Viernes,Sábado,Domingo";
String[] arrSplit = strMain.split(",");

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

y el resultado sería el siguiente

```
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

1. Se guarda la frase en un string.

```
String strMain = "Lunes,Martes,Miércoles,Jueves,Viernes,Sábado,Domingo";
```

2. Luego metemos en un array de string, pero con la condición de separarlo en comas, se usará la función split().

```
String[] arrSplit = strMain.split(",");
```

3. Luego se recorre el array con un For, y se imprime por pantalla.

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

Otro método Java string split () con expresiones regulares y longitud.

Considere una situación en la que solo necesita los primeros elementos “X” después de la operación de división, pero quiere que el resto de la cadena permanezca como está.
Para qué el resultado sea como este:

- Lunes
- Martes
- Miércoles
- Jueves
- Viernes
- Sábado,Domingo.

Esto se puede lograr pasando otro argumento junto con la operación de división, y ese será el límite de cadenas requeridas.

```
String strMain = "Lunes,Martes,Miércoles,Jueves,Viernes,Sábado,Domingo";

String[] arrSplit_2 = strMain.split(",", 6);
for (int i = 0; i < arrSplit_2.length; i++) {
    System.out.println(arrSplit_2[i]);
}
```

Resultado:

```
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado,Domingo
```

El manejo es parecido pero la unica diferencia seria que al meter el estrin en el array ponemos el limitador.

```
String[] arrSplit_2 = strMain.split(",", 6);
```