

StringTokenizer Class

문자열이 **특정 구분자(delimiter)**로 연결되어 있을 때, 구분자를 기준으로 하여 문자열을 분리하기 위해서 `split()` 메소드 또는 `StringTokenizer` 클래스를 이용할 수 있습니다.

`split()`은 정규 표현식(Regular Expression)으로 구분하고, `StringTokenizer`는 문자로 구분한다는 차이점이 있습니다.

split() 메소드

`String` 클래스의 `split()` 메소드는 다음과 같이 호출되는데, 정규 표현식을 구분자로 해서 문자열을 분리한 후, 배열에 저장하고 리턴합니다.

```
1 | String[] result = "문자열".split("정규표현식"); cs
```

다음은 기호로 구분된 이름을 분리하여 출력하는 예제입니다.

* `StringSplitExam.java`

```
1 | package api;
2 |
3 |
4 | public class StringSplitExam {
5 |
6 |     public static void main(String[] args) {
7 |         String text = "Jack&Nick,Ciel-Tom,Jolie";
8 |
9 |         String[] names = text.split("&|,|-");
10 |
11 |         for (String name : names) {
12 |             System.out.println(name);
13 |         }
14 |     }
15 |
16 | }
17 |
18 |
19 |
```

Colored by Color Scripter cs

```
Problems Javadoc Declaration Console
<terminated> StringSplitExam [Java Application] /Library/Java/JavaVirtualM
Jack
Nick
Ciel
Tom
Jolie
http://palpit.tistory.com
```

StringTokenizer 클래스

문자열이 **한 종류의 구분자**로 연결되어 있을 경우, StringTokenizer 클래스를 사용하면 손 쉽게 문자열(토큰: token)을 분리해 낼 수 있습니다. StringTokenizer 객체를 생성할 때 첫 번째 파라미터로 전체 문자열을 주고, 두 번째 파라미터로 구분자를 주면 됩니다. 만약 구분자가 생략되면 구분자는 기본적으로 공백(space)가 됩니다.

```
1 StringTokenizer st = new StringTokenizer("문자열", "구분자");
2 Colored by Color Scripter cs
```

StringTokenizer 객체가 생성되면 부분 문자열을 분리해 낼 수 있는데, 다음 메소드들을 이용해서 전체 토큰 수, 남아 있는 토큰 여부를 확인한 다음, 토큰을 읽으면 됩니다.

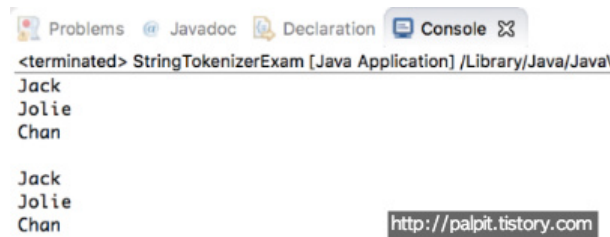
메소드	설명
int	countTokes() 꺼내지 않고 남아있는 토큰의 수
boolean	hasMoreTokens() 남아 있는 토큰이 있는지 여부
String	nextToken() 토큰을 하나씩 꺼내옴

다음은 두 가지 방법으로 토큰을 추출하는 방법을 보여줍니다.

* StringTokenizerExam.java

```
1 package api;
2
3
4 import java.util.StringTokenizer;
5
6 public class StringTokenizerExam {
7
8     public static void main(String[] args) {
9         String text = "Jack/Jolie/Chan";
10
11         // Way 1
12         StringTokenizer st = new StringTokenizer(text, "/");
13         int countTokens = st.countTokens();
14
15         for (int i = 0; i < countTokens; i++) {
16             String token = st.nextToken();
17             System.out.println(token);
18         }
19
20         System.out.println();
21
22         // Way 2
23         st = new StringTokenizer(text, "/");
24
25         while (st.hasMoreTokens()) {
26             String token = st.nextToken();
27             System.out.println(token);
28         }
29     }
30 }
31
32
33
34
```

Colored by Color Scripter cs



```
<terminated> StringTokenizerExam [Java Application] /Library/Java/Java
Jack
Jolie
Chan

Jack
Jolie
Chan
http://palpit.tistory.com
```

StringBuffer, StringBuilder Class

String 클래스는 replace 등을 이용하여 수정 할 때, 수정되는 것이 아니라, 수정된 문자열을 새롭게 만들어 가리킨다.

그러므로, 수정등을 자주 사용하게 된다면 메모리 부하가 걸리게 될 것이다.

반면, StringBuffer와 StringBuilder 는 내부 버퍼가 있어서, 문자열을 수정할 때, 버퍼안에 있는 문자열이 직접적으로 수정된다.

두 클래스의 차이는 StringBuffer는 멀티스레드에서 사용 할 수 있도록 동기화가 적용 됐지만 Builder는 단일 스레드 환경에서만 사용되
게 동기화가 적용되지 않았다.

StringBuffer, Builder의 사용법은 동일하다.

StringBuilder sb = new StringBuilder(초기크기); 생성자로 사용가능 하며, 초기크기는 자동으로 커지기 때문에 신경 안써도 되고 안넣
어도 된다.

메소드	설명	http://palpit.tistory.com
append(...)	문자열 끝에 주어진 파라미터 추가	
insert(int offset, ...)	문자열 중간에 주어진 파라미터 추가	
delete(int start, int end)	문자열의 일부분을 삭제	
deleteCharAt(int index)	문자열에서 주어진 index의 문자를 삭제	
replace(int start, int end, String str)	문자열의 일부분을 다른 문자열로 대체	
StringBuilder reverse()	문자열의 순서를 뒤바꿈	
setCharAt(int index, char ch)	문자열에서 주어진 index의 문자를 다른 문자로 대체	

append()와 insert() 메소드는 매개 변수가 다양한 타입으로 오버로딩되어 있기 때문에 대부분의 값을 문자로 추가 또는 삽입 할 수 있습니다.

* StringBuilderExam.java

```
1 package api;
2
3
4 public class StringBuilderExam {
5
6     public static void main(String[] args) {
7         StringBuilder sb = new StringBuilder();
8
9         sb.append("Java ");
10        sb.append("Project Meeting");
11        System.out.println(sb.toString());
12
13        sb.insert(4, "8");
14        System.out.println(sb.toString());
15
16        sb.setCharAt(4, '9');
17        System.out.println(sb.toString());
18
19        sb.replace(14, 21, "Design");
20        System.out.println(sb.toString());
21
22        sb.delete(4, 5);
23        System.out.println(sb.toString());
24
25        int length = sb.length();
26        System.out.println("총 문자수 : " + length);
27    }
28 }
29
30
31
32
```

Colored by Color Scripter cs

