Core Java

آموزش ساده و آسان جاوا

به نام خدا

تقدیم به هموطنان عزیزم

جاوا را با لذت یاد بگیرید!

Core Java

آموزش ساده و آسان جاوا

آموزش زبان برنامه نویسی جاوا

نمونه مثال از کلاس String

متد split()

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!

Core Java

آموزش ساده و آسان جاوا

سلام. دوست من امروز قصد داریم یکی دیگر از متدهای موجود در کلاس String را با هم بررسی کنیم.

متد `split()` برای تقسیم یک String به **زیر رشته های** آن بر اساس **جداکننده(delimiter)** داده شده استفاده می شود.

به بیان ساده تر وقتی قصد داریم یک String را بر اساس یک علامت خاص قطعه قطعه یا تجزیه کنیم از این متد استفاده می کنیم.

نکته: اون **علامت خاص** که بر اساس آن یک String را به زیر رشته های آن تقسیم می کنیم همون **جداکننده(delimiter)** ما می باشد.

- با مثال متوجه خواهید شد.

برای مثال فرض کنید یک String به صورت زیر داشته باشیم:

```
String str="r 2 5";
```

حالا قصد داریم رشته str را بر اساس فاصله بین زیر رشته های آن، تجزیه کنیم.

نکته: منظور از فاصله بین زیررشته های یک رشته همون **space** یا " " می باشد.

در زیر فاصله بین زیررشته های رشته "r 2 5" را با رنگ سبز نشان داده ایم:

"r 2 5"

همان طور که گفتیم قصد داریم رشته str را بر اساس علامت فاصله (" ") به زیر رشته های آن تجزیه یا جدا کنیم. اگر رشته str را با استفاده از متد split و بر اساس علامت فاصله (" ") جدا کنیم، نتیجه سه رشته مجزا به صورت زیر خواهد بود:

"r" "2" "5"

- همان طور که مشاهده کردید ما رشته "r 2 5" را با استفاده از متد split و بر اساس علامت " " به سه زیر رشته آن تقسیم(جدا) کردیم. جلوتر مثال کدنویسی میزنیم بهتر درک خواهید کرد.

Core Java

آموزش ساده و آسان جاوا

متد split دارای دو پارامتر مختلف می باشد:

```
String[] split(String regex):
```

- این متد یک رشته را بر اساس یک علامت خاص به زیر رشته های آن **جدا** می کند. به بیانی دیگر این متد زیر رشته های یک String را بر اساس یک علامت خاص تکه تکه می کند و به صورت آرایه ای از String برمی گرداند.
- پارامتر regex نقش علامتی که بر اساس آن String مورد نظر را قطعه قطعه می کنیم، برعهده دارد.

Example:

```
package javalike;

public class Test9 {

	public static void main(String[] args) {
		String str = "r 2 5";
		String array[] = str.split(" ");

		for (int i = 0; i < array.length; i++)
			System.out.println("Separated string " + i + ": " + array[i]);
	}

}
```

خروجی(output):

```
Separated string 0: r
Separated string 1: 2
Separated string 2: 5
```

توضیحات:

- در مثال بالا ما رشته str را با استفاده از متد split به سه زیر رشته آن ، بر اساس علامت فاصله یا " " تقسیم کرده و درون آرایه از نوع String ریخته ایم.

```
String str = "r 2 5";
```

- یک String تعریف کرده ایم که بین زیر رشته های آن علامت فاصله است.

Core Java

آموزش ساده و آسان جاوا

```
String array[] = str.split(" ");
```

- رشته str را بر اساس علامت " " یا فاصله به زیر رشته های آن جدا کرده و درون آرایه از نوع String ریخته ایم. در هر خانه از آرایه رشته جدا شده از رشته str قرار گرفته است.

```
    for (int i = 0; i < array.length; i++)
            System.out.println("Separated string " + i + ": " + array[i]);
    }
```

- حال رشته های جدا شده ای که درون آرایه ما قرار گرفته اند را با دستور بالا چاپ کرده ایم.

متد دوم.....

```
String[] split(String regex, int limit):
```

- این متد نیز مانند متد قبل برای جدا کردن زیر رشته های یک String بر اساس علامت خاص می باشد. تنها تفاوت این متد با متد قبل این است که می توانیم برای تعداد زیر رشته های استخراج شده از یک رشته محدودیت ایجاد کنیم.

نکته: برای هر زیر رشته استخراج شده از یک رشته از اصطلاح **token** استفاده می کنیم.
مثلا اگر رشته "r 2 5" را بر اساس علامت " " تکه تکه کنیم، به هر زیر رشته آن یک **token** می گوییم. مثلا رشته"r" یک **token** رشته "2" یک token و رشته "5" یک token خواهد بود.

- حالا کاربرد متد String[] split(String regex, int limit) چیست؟ با این متد می توانیم تعداد token یا زیر رشته های جدا شده از یک رشته را مشخص کنیم.
- پس متد split بر اساس regex ، به تعداد limit ، زیر رشته های یک String را تکه تکه کرده و به صورت آرایه از نوع String برمی گرداند. با مثال بهتر متوجه می شوید.

Core Java

آموزش ساده و آسان جاوا

Example:

```
package javalike;

public class Test9 {

	public static void main(String[] args) {
		String str = "Welcome-to-javapro.ir";
		String array1[] = str.split("-");
		System.out.println("");
		System.out.println("Return Value :");
		for (int i = 0; i < array1.length; i++)
			System.out.println("Separated string " + i + ": " + array1[i]);

		System.out.println("");
		System.out.println("Return Value :");
		String array2[] = str.split("-", 2);

		for (int i = 0; i < array2.length; i++)
			System.out.println("Separated string " + i + ": " + array2[i]);
	}

}
```

خروجی(output):

```
Return Value :
Separated string 0: Welcome
Separated string 1: to
Separated string 2: javapro.ir

Return Value :
Separated string 0: Welcome
Separated string 1: to-javapro.ir
```

توضیحات:

در مثال بالا ما زیر رشته های رشته str را با دو متد مختلف split تکه تکه کردیم.

Core Java

آموزش ساده و آسان جاوا

```
String array1[] = str.split("-");
```

- در این دستور با استفاده از متد split و بر اساس علامت " - " تمام زیر رشته های (token های) رشته str را درون آرایه ای از نوع String ریخته ایم. در اینجا محدودیتی برای تعداد زیر رشته های جدا شده رشته str ایجاد نکردیم. و خروجی زیر رشته های استخراج شده رشته str به صورت زیر می باشد:

```
Separated string 0: Welcome
Separated string 1: to
Separated string 2: javapro.ir
```

```
String array2[] = str.split("-", 2);
```

- در دستور بالا ما گفتیم تنها حق داری دو زیر رشته یا token از رشته str را بر اساس علامت "-" جدا کنی. خروجی زیر رشته های جدا شده از رشته str بر اساس دستور بالا به صورت زیر خواهد بود:

```
Separated string 0: Welcome
Separated string 1: to-javapro.ir
```

- همان طور که می بینید تنها دو زیر رشته از رشته str بر اساس علامت "-" جدا شده است.

Example:

```
package javalike;

public class SplitExample {
	public static void main(String args[]) {
		String str = new String("28/12/2013");
		System.out.println("split(String regex):");
		String array1[] = str.split("/");
		for (String temp : array1) {
			System.out.println(temp);
		}
		System.out.println("split(String regex, int limit) with limit=2:");
		String array2[] = str.split("/", 2);
		for (String temp : array2) {
			System.out.println(temp);
		}
```

```
    }
}
```

خروجی(output):

```
split(String regex):
28
12
2013
split(String regex, int limit) with limit=2:
28
12/2013
```

نکته: اگر مقدار limit پارامتر موجود در متد split(String regex, int limit) منفی یا صفر بود چه اتفاقی می افتد؟ با مثال به این مسئله می پردازیم:

Example:

در مثال زیر از متد split(String regex) استفاده کردیم.یعنی از متد split ای که محدودیتی در جدا کردن زیر رشته های یک رشته ندارد ، استفاده کرده ایم.

```
package javalike;

public class Test110 {

	public static void main(String[] args) {
		String str = "_a_b_c_d_e_f_";
		String array2[] = str.split("_");

		for (int i = 0; i < array2.length; i++)
			System.out.println("Separated string " + i + ": " + array2[i]);

	}

}
```

Core Java

آموزش ساده و آسان جاوا

خروجی(output):

```
Separated string 0:
Separated string 1: a
Separated string 2: b
Separated string 3: c
Separated string 4: d
Separated string 5: e
Separated string 6: f
```

Example:

در مثال زیر از متد `split(String regex, int limit)` استفاده کردیم. و مقدار پارامتر `limit` را صفر قرار داده ایم:

```
package javalike;

public class Test110 {

	public static void main(String[] args) {
		String str = "_a_b_c_d_e_f_";
		String array2[] = str.split("_", 0);

		for (int i = 0; i < array2.length; i++)
			System.out.println("Separated string " + i + ": " + array2[i]);

	}

}
```

خروجی(output):

```
Separated string 0:
Separated string 1: a
Separated string 2: b
Separated string 3: c
Separated string 4: d
Separated string 5: e
```

```
Separated string 6: f
```

- همان طور که در خروجی دو مثال قبل مشاهده می کنید ، هیچ تفاوتی بین خروجی ها وجود ندارد. به عبارت دیگر وقتی مقدار limit پارامتر موجود در متد `split(String regex, int limit)` برابر **صفر** قرار می دهیم هیچ تفاوتی با دستور `split(String regex)` نمی کند.
- پس وقتی مقدار پارامتر limit متد split برابر صفر باشد همه زیر رشته های یک رشته بدون محدودیت بر اساس علامت مورد نظر تکه تکه می شوند.

Example:

در مثال زیر از متد `split(String regex, int limit)` استفاده کردیم. و مقدار پارامتر `limit` را یک **عدد منفی** قرار داده ایم:

```
package javalike;

public class Test110 {

	public static void main(String[] args) {
		String str = "_a_b_c_d_e_f_";
		String array2[] = str.split("_", -1);

		for (int i = 0; i < array2.length; i++)
			System.out.println("Separated string " + i + ": " + array2[i]);

	}

}
```

خروجی(output):

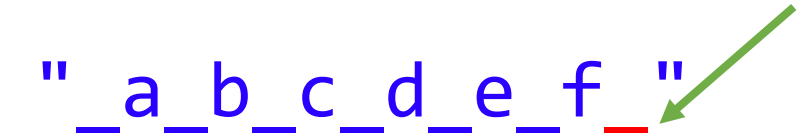
```
Separated string 0:
Separated string 1: a
Separated string 2: b
Separated string 3: c
Separated string 4: d
Separated string 5: e
Separated string 6: f
```

```
Separated string 7:
```

- وقتی مقدار پارامتر limit متد split(String regex, int limit) برابر یک عدد منفی باشد (فرق نمیکنه چه عدد منفی باشد) ،سمت راست آخرین فضای خالی بعد علامتی که بر اساس آن یک رشته را تجزیه می کنیم نیز به عنوان یک زیر رشته یا token از رشته اصلی جدا می شود. اما در حالتی که limit برابر با صفر یا یک عدد مثبت بود آخرین فضای خالی بعد علامت جدا کننده نادیده گرفته میشد.

نتیجه گیری کلی: در متد split(String regex, int limit):

۱. وقتی limit برابر با صفر یا عدد مثبت باشد(limit>=0) آخرین فضای خالی بعد از علامت جدا کننده نادیده گرفته می شود:شکل(۱)



شکل(۱)

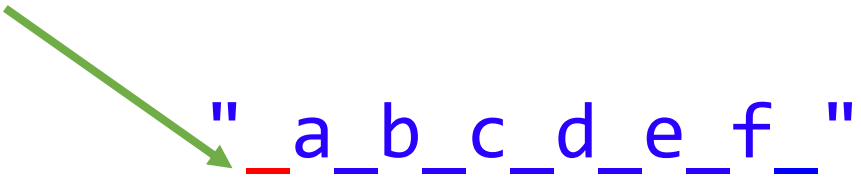
- منظور از فضای خالی بعد از آخرین علامت جداکننده بخشی است که با فلش بهش اشاره شده است. خب بله حق دارید فضای خالی بعد از آخرین علامت "_" دیده نمی شود اما دیگه متد split با مقدر limit منفی اون فضا را در نظر می گیرد.

۲. وقتی limit برابر یک عدد منفی باشد (limit<0) فضای سفید بعد از آخرین علامت جداکننده به عنوان یک زیر رشته یا token در نظر گرفته می شود.

متد split با همه پارامترهای مختلف و مقدار limit های مثبت یا منفی یا صفر، اولین فضای سفید قبل از علامت جداکننده را درنظر می گیرند:شکل(۲)

Core Java

آموزش ساده و آسان جاوا



شکل(۲)

نکته مهم: اگر در یک رشته در کنار هم علامت جدا کننده تکراری داشته باشیم چه اتفاقی می افتد؟ به مثال زیر توجه کنید:

Example:

در مثال زیر از متد split(String regex) استفاده کردیم. به گونه ای که در رشته str دو علامت جداکننده تکراری "-" کنار هم قرار گرفته اند.

```
package javalike;

public class Test110 {

	public static void main(String[] args) {
		String str = "--a--b--c--d--e--f--";
		String array2[] = str.split("-");

		for (int i = 0; i < array2.length; i++)
			System.out.println("Separated string " + i + ": " + array2[i]);

	}

}
```

خروجی(output):

```
Separated string 0:
Separated string 1:
Separated string 2: a
```

```
Separated string 3:
Separated string 4: b
Separated string 5:
Separated string 6: c
Separated string 7:
Separated string 8: d
Separated string 9:
Separated string 10: e
Separated string 11:
Separated string 12: f
```

- تنها تفاوت این حالت ،این است که یکی از علامت های "-" به عنوان علامت جدا کننده و فضای سفید جایگزین علامت "-" دوم میشود و به عنوان یک زیر رشته یا token استخراج می شود.
- برای سایر پارامترهای متد split و limit های متنوع هم همین قانون صادق است.

Example:

```
package javalike;

public class Test110 {
	public static void main(String[] args) {
		String str = "--a--b--c--d--e--f--";
		String array1[] = str.split("-", -1);

		System.out.println("limi<0");
		for (int i = 0; i < array1.length; i++)
			System.out.println("Separated string " + i + ": " + array1[i]);
		String array2[] = str.split("-", 0);
		System.out.println("");
		System.out.println("limi=0");
		for (int i = 0; i < array2.length; i++)
			System.out.println("Separated string " + i + ": " + array2[i]);
		System.out.println("limi=15");
		System.out.println("");
		String array3[] = str.split("-", 15);
		for (int i = 0; i < array3.length; i++)
			System.out.println("Separated string " + i + ": " + array3[i]);
	}

}
```

Core Java

آموزش ساده و آسان جاوا

خروجی(output):

```
limi<0
Separated string 0:
Separated string 1:
Separated string 2: a
Separated string 3:
Separated string 4: b
Separated string 5:
Separated string 6: c
Separated string 7:
Separated string 8: d
Separated string 9:
Separated string 10: e
Separated string 11:
Separated string 12: f
Separated string 13:
Separated string 14:

limi=0
Separated string 0:
Separated string 1:
Separated string 2: a
Separated string 3:
Separated string 4: b
Separated string 5:
Separated string 6: c
Separated string 7:
Separated string 8: d
Separated string 9:
Separated string 10: e
Separated string 11:
Separated string 12: f
limi=15

Separated string 0:
Separated string 1:
Separated string 2: a
Separated string 3:
Separated string 4: b
Separated string 5:
```

```
Separated string 6: c
Separated string 7:
Separated string 8: d
Separated string 9:
Separated string 10: e
Separated string 11:
Separated string 12: f
Separated string 13:
Separated string 14:
```

تمام تلاشم کردن به ساده ترین شکل ممکن این مبحث رو آموزش بدم ، اگر باز در توضیحات ابهامی دارید نگران نباشید، این کاملا طبیعی است!!! چندین و چندین بار به مثال ها نگاه کنید و برای خودتون در IDE مثل ایکلیپس و.. run و اجرا کنید و تغییر بدید ورودی ها رو مطمئنا مفهومش دستتون میاد . ☺☺☺

پیروز و موفق باشید

Core Java

آموزش ساده و آسان جاوا

سایت آموزش زبان جاوا به زبان ساده،آسان و شیرین!!!

www.JAVAPro.ir

آموزش جاوا SE را با تجربه شخصی و به زبان خودمونی یاد بگیرید!!!!

بازدید از کانال

بازدید از سایت

هر روز مفاهیم و مثال های جدید به سایت اضافه می شود برای اطلاع از مطالب جدید روی سایت عضو کانال شوید.