Java中的StringBuffer和StringBuilder

1. **StringBuffer** ：字符串缓冲区，就是用于存储字符串的**容器**。
2. StringBuffer和StringBuilder的关系：StringBuffer 在JDK1.0就出现了，而StringBuilder是在JDK 1.5出现的，功能和StringBuffer()一模一样，它们的构造方法和一般方法都一样，唯一的差别尽在于在于一个同步一个不同步：StringBuffer是同步的，而StringBuilder是不同步的。

**两者的不同点** ：StringBuffer是线程**同步**的，通常用于多线程（JDK1.0便有了）；StringBuilder是线程**不同步**的，通常用于单线程，是为了提高效率而出来的，JDK1.5出来的。由于StringBuilder不需要同步，所以执行效率更快。

1. StringBuffer和StringBuilder的最主要的方法是**append和insert**，这是String类中没有的方法。还有**reverse方法**等特有的。这两个类中有些方法功能与String重复，有些是特有的，但是有些String有而这两个没有，如spilt只是String类有。
2. **注意：利用append（）方法的好处：添加数据后，还是这个容器，而利用**

**str = str + “XXX”；相当于创建了一个新的字符串。**

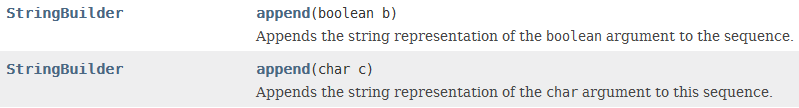
1. StringBuffer的特点：
   1. 长度是可以变化的（超过了就会自动复制到新的更大的缓冲区来实现自动变化）；
   2. 可以存储不同类型的数据，利用**insert方法**，最终转换成**字符串**使用。
2. **StringBuffer有4个构造方法（StringBuilder的构造方法与StringBuffer的一样）**：
3. StringBuffer() ：默认初始容量是16个字符，此时capacity为16，length为0.
4. StringBuffer(CharSequence seq)
5. StringBuffer(int capacity)：指定初始容量值，capacity为设定值，length为0。
6. **StringBuffer(String str)**：容器初始为设定的字符串，然而capacity为16 + 字符串长度，如：StringBuffer sb = new StringBuffer("zhaohong"); capacity为24，length为8.
7. **至于可变长度理解：**是自动把原来的数据复制到一个更大的容器中（将原来的复制一倍等方法进行扩大），数组无论如何都是不可变长度的。
8. **StringBuffer** 既然是一个存储容器，应该具备的功能（**成员方法介绍**）：
9. **添加：**
   1. **方法 append（data）；方法**

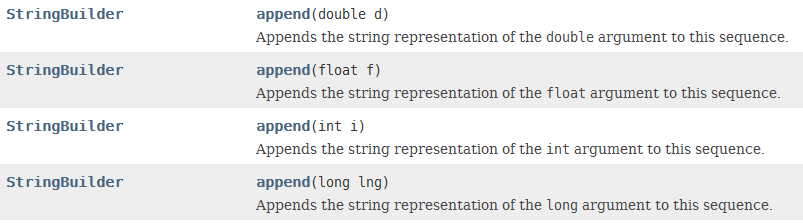
**data**可以是除了short、byte之外的任何类型，但是short、byte类型可以利用int进行接收。此方法返回的是**当前的StringBuffer对象**。而且返回的仍然是同一个StringBuffer对象，此方法可以以**方法链的方式**调用：

如 StringBuffer sb = new StringBuffer("abc");

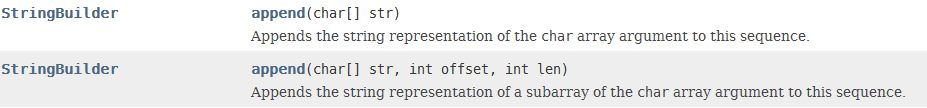
StringBuffer sb2 = sb.append("def").append(true).append(344);

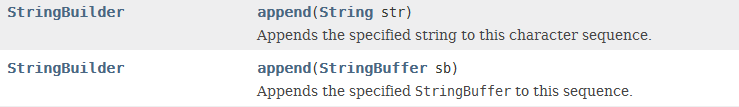
System.out.println(sb == sb2);//返回结果为true

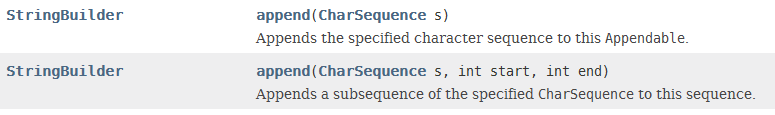


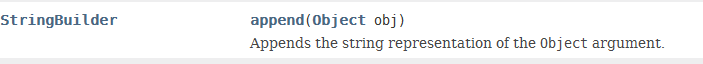






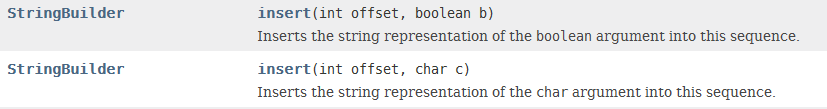


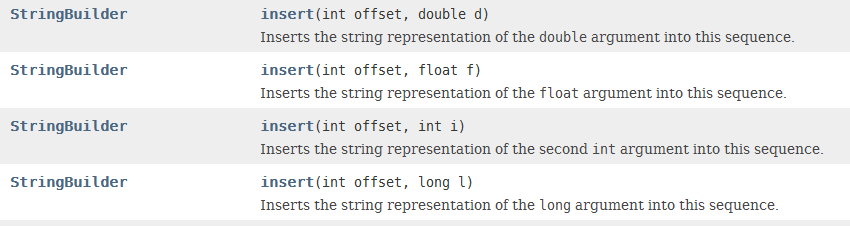


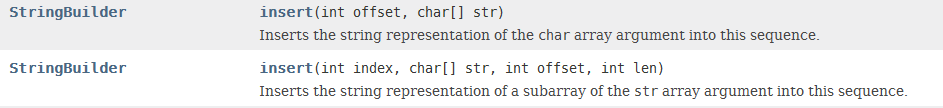


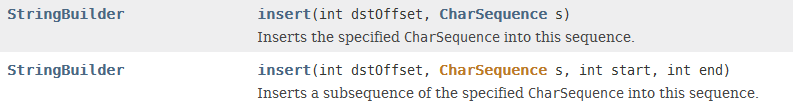
* 1. **insert（）方法： insert(int offset/index, data)；返回仍然是一个StringBuffer对象。**

**insert（）方法和append（）的方法大体相同，能append什么，就能够insert什么。**









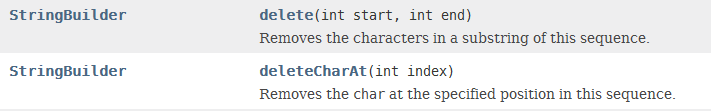




1. **删除方法：**
   1. **StringBuffer** [**delete**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/StringBuffer.html#delete-int-int-)**(int start, int end)**

如果清空一个容器，delete（0，sb .length()）

* 1. StringBuffer [deleteCharAt](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/StringBuffer.html#deleteCharAt-int-)(int index)



1. **查找：**
   1. **char** [**charAt**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/StringBuffer.html#charAt-int-)**(int index)方法**
   2. int indexOf(String str) **方法**
   3. int lastIndexOf(String str) **方法**
   4. int [lastIndexOf](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/StringBuffer.html#lastIndexOf-java.lang.String-int-)([String](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html) str, int fromIndex) **方法**
2. **修改：**
   1. StringBuffer replace(int start, int end, String str)
   2. void [setCharAt](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/StringBuffer.html#setCharAt-int-char-)(int index, char ch)
3. **方法void setLength(int newLength)** ：

如果大于容器的长度，则自动在后面以空字符添加，如果**设置的长度**小于容器长度，则自动舍弃后面的字符。

1. JDK的提高目的就是三点：**简化书写，提高效率，增加安全性。**
2. 可以向**StringBuffer**存放很多数据，但是从**StringBuffer** 从取出来的数据都是**字符串**。
3. StringBuffer 和StringBuilder的方法使用一模一样，它们与**String的方法**有些区别：
4. 前者不仅有**length方法**，还有**capacity方法**，返回都是int，后者只有length方法；
5. 前者没有**比较方法，没有实现Comparable接口**，String有**compareTo方法**，实现了接口Comparable；
6. 都有**查找方法**，indexOf，charAt等，可以找出某个索引处的字符或字符串，也可以根据字符或字符串找出索引，可以正序也可以反序；
7. 都可以从原来的字符串中**截取新**的字符串，有两个方法，

1.subString（int start） 2. subString（int start ， int end）；

1. 前者有**append方法**，可以添加除了short、byte外的六种基本数据类型，还有字符串、StringBuffer，任何类的引用Object等，字符数组，或字符数组的一部分。
2. 前者有**insert方法**，插入的东西和append的一样。
3. 前者有**删除操作**，两个方法，第一个delete（int start，int end），第二个是deleteCharAt（int index），返回都是一个StringBuffer或StringBuilder。而String中没有删除方法。（**注意：字符串缓冲区中的删除方法名字为delete，然而集合中的删除操作名称为clear，remove**）
4. 都有替换方法，**replace**。
5. 前者有设置字符操作：**setCharAt（int index，char ch）**
6. 前者有设置字符长度操作：**setLength（int newLength）**
7. String单独的功能还有很多，可以进行**字符串的连接**concat方法，包含关系的判断**contains方法**等。