**[JAVA的StringBuffer类](http://www.cnblogs.com/springcsc/archive/2009/12/03/1616330.html)**

**Posted on 2009-12-03 16:42**[**火之光**](http://www.cnblogs.com/springcsc/)**阅读(12539) 评论(1)**[**编辑**](http://www.cnblogs.com/springcsc/admin/EditPosts.aspx?postid=1616330)[**收藏**](http://www.cnblogs.com/springcsc/archive/2009/12/03/1616330.html)**http://www.cnblogs.com/springcsc/aggbug/1616330.html?type=1&webview=1**

**StringBuffer类和String一样，也用来代表字符串，只是由于StringBuffer的内部实现方式和String不同，所以StringBuffer在进行字符串处理时，不生成新的对象，在内存使用上要优于String类。**

**所以在实际使用时，如果经常需要对一个字符串进行修改，例如插入、删除等操作，使用StringBuffer要更加适合一些。**

**在StringBuffer类中存在很多和String类一样的方法，这些方法在功能上和String类中的功能是完全一样的。**

**但是有一个最显著的区别在于，对于StringBuffer对象的每次修改都会改变对象自身，这点是和String类最大的区别。**

**另外由于StringBuffer是线程安全的，关于线程的概念后续有专门的章节进行介绍，所以在多线程程序中也可以很方便的进行使用，但是程序的执行效率相对来说就要稍微慢一些。**

**1、StringBuffer对象的初始化**

**StringBuffer对象的初始化不像String类的初始化一样，Java提供的有特殊的语法，而通常情况下一般使用构造方法进行初始化。**

**例如：**

**StringBuffer s = new StringBuffer();**

**这样初始化出的StringBuffer对象是一个空的对象。**

**如果需要创建带有内容的StringBuffer对象，则可以使用：**

**StringBuffer s = new StringBuffer(“abc”);**

**这样初始化出的StringBuffer对象的内容就是字符串”abc”。**

**需要注意的是，StringBuffer和String属于不同的类型，也不能直接进行强制类型转换，下面的代码都是错误的：**

**StringBuffer s = “abc”;               //赋值类型不匹配**

**StringBuffer s = (StringBuffer)”abc”;    //不存在继承关系，无法进行强转**

**StringBuffer对象和String对象之间的互转的代码如下：**

**String s = “abc”;**

**StringBuffer sb1 = new StringBuffer(“123”);**

**StringBuffer sb2 = new StringBuffer(s);   //String转换为StringBuffer**

**String s1 = sb1.toString();              //StringBuffer转换为String**

**2、StringBuffer的常用方法**

**StringBuffer类中的方法主要偏重于对于字符串的变化，例如追加、插入和删除等，这个也是StringBuffer和String类的主要区别。**

**a、append方法**

**public StringBuffer append(boolean b)**

**该方法的作用是追加内容到当前StringBuffer对象的末尾，类似于字符串的连接。调用该方法以后，StringBuffer对象的内容也发生改变，例如：**

**StringBuffer sb = new StringBuffer(“abc”);**

**sb.append(true);**

**则对象sb的值将变成”abctrue”。**

**使用该方法进行字符串的连接，将比String更加节约内容，例如应用于数据库SQL语句的连接，例如：**

**StringBuffer sb = new StringBuffer();**

**String user = “test”;**

**String pwd = “123”;**

**sb.append(“select \* from userInfo where username=“)**

**.append(user)**

**.append(“ and pwd=”)**

**.append(pwd);**

**这样对象sb的值就是字符串“select \* from userInfo where username=test and pwd=123”。**

**b、deleteCharAt方法**

**public StringBuffer deleteCharAt(int index)**

**该方法的作用是删除指定位置的字符，然后将剩余的内容形成新的字符串。例如：**

**StringBuffer sb = new StringBuffer(“Test”);**

**sb. deleteCharAt(1);**

**该代码的作用删除字符串对象sb中索引值为1的字符，也就是删除第二个字符，剩余的内容组成一个新的字符串。所以对象sb的值变为”Tst”。**

**还存在一个功能类似的delete方法：**

**public StringBuffer delete(int start,int end)**

**该方法的作用是删除指定区间以内的所有字符，包含start，不包含end索引值的区间。例如：**

**StringBuffer sb = new StringBuffer(“TestString”);**

**sb. delete (1,4);**

**该代码的作用是删除索引值1(包括)到索引值4(不包括)之间的所有字符，剩余的字符形成新的字符串。则对象sb的值是”TString”。**

**c、insert方法**

**public StringBuffer insert(int offset, boolean b)**

**该方法的作用是在StringBuffer对象中插入内容，然后形成新的字符串。例如：**

**StringBuffer sb = new StringBuffer(“TestString”);**

**sb.insert(4,false);**

**该示例代码的作用是在对象sb的索引值4的位置插入false值，形成新的字符串，则执行以后对象sb的值是”TestfalseString”。**

**d、reverse方法**

**public StringBuffer reverse()**

**该方法的作用是将StringBuffer对象中的内容反转，然后形成新的字符串。例如：**

**StringBuffer sb = new StringBuffer(“abc”);**

**sb.reverse();**

**经过反转以后，对象sb中的内容将变为”cba”。**

**e、setCharAt方法**

**public void setCharAt(int index, char ch)**

**该方法的作用是修改对象中索引值为index位置的字符为新的字符ch。例如：**

**StringBuffer sb = new StringBuffer(“abc”);**

**sb.setCharAt(1,’D’);**

**则对象sb的值将变成”aDc”。**

**f、trimToSize方法**

**public void trimToSize()**

**该方法的作用是将StringBuffer对象的中存储空间缩小到和字符串长度一样的长度，减少空间的浪费。**

**总之，在实际使用时，String和StringBuffer各有优势和不足，可以根据具体的使用环境，选择对应的类型进行使用。**