这一两天在学习string的一些用法，记得在来北京面试的时候，别人现场的考的试题里面就包含对string方法的使用，由于当时对string方法不熟悉，就模棱两可的回答别人，现在学习了后，想想都觉得当时汗颜 啊，嘿嘿，不过还好，现在基本用法都已经基本掌握了，做下这些笔记希望对自己或阅读的朋友有一定的帮助。

string类适用于描述字符串事物。  
那么它就提供了多个方法对字符串进行操作

以下是string的七种用法，注意哦，记得要时常去查看**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/fetch001/article/details/_blank)**的API文档，那个里面也有很详细的介绍

1，获取  
 1.1：字符串中包含的字符数，也就是字符串的长度。  
  int length():获取长度  
 1.2：根据位置获取位置上某个字符。  
  char charAt(int index)  
 1.3:根据字符获取该字符在字符串中的位置。  
  int indexOf(int ch):返回的是ch在字符串中第一次出现的位置。  
  int indexOf(int ch,int fromIndex):从fromIndex指定位置开始，获取ch在字符串中出现的位置。  
    
  int indexOf(String str):返回的是str在字符串中第一次出现的位置。  
  int indexOf(String str，int fromIndex):从fromIndex指定位置开始，获取str在字符串中出现的位置。  
 1.4：int lastIndexOf(String str):反向索引。

2，判断  
 2.1：字符串中是否包含某一个子串。  
      boolean contains(str);  
   特殊之处：indexOf(str):可以索引str第一次出现为止，如果返回-1，表示该str不在字符串中存在。  
             所以，也可以用于对指定判断是否包含。  
       if(str.indexOf("a")!=1)

       而且该方法既可以判断，也可以获取出现的位置。

 2.2：字符串中是否有内容。   
       boolean isEmpty():原理就是判断长度是否为0。  
 2.3：字符串是否以指定内容开头。  
   boolean startsWith(str);  
 2.4：字符串是否以指定内容结尾。  
   boolean endsWith(str);  
 2.5：判断字符内容是否相同，复写了object类中的equals方法。  
   boolean equals(str);  
 2.6:判断内容是否相同，并忽略大小写。  
  boolean.equalsIgnorecase();

3.转换。  
 3.1：将字符数组转成字符串。  
  构造函数：String(char[])  
      String(char[],offset,count):将字符数组中的一部分转成字符串  
   
  静态方法：  
     static String copyValueOf(char[]);  
     static String copyValueOf(char[] data,int offset,int count);

     static String valueOf(char[]);  
   
 3.2：将字符串转成字符组  
  char[] tocharArray();

 3.3：将字节数组转成字符串。  
      String(byte[])  
      String(byte[],offset,count):将字节数组中的一部分转成字符串  
   
 3.4：将字符串转成字节数组。  
  byte[] getBytes()

 3.5:将基本数据类型转成字符串，  
 static String valueOf(int)  
 static String valueOf(double)

 // 3+"" 与 String.valueOf(3)的值是一样的  
   
 特殊：字符串和字节数组在转换过程中，是可以指定编码的。

4.替换  
 String replace(oldchar,newchar);

5.切割  
 String[] split(regex);

6.子串。获取字符串中的而一部分  
 String subString(begin);  
 String subString(begin,end);

7.转换，去除空格，比较。  
 7.1：将字符串转成大写或小写  
   String toUpperCsae() 大转小  
   String toLowerCsae() 小转大

 7.2：将字符串两端的多个空格去除  
   String trim();

 7.3：对两个字符串进行自然顺序的比较  
   int compareTo(string);

 请看如下代码，下面的代码都是针对上面string七种用法而进行一一举例说明：

class StringMethodDemo   
{  
 public static void method\_Zhuanhuan\_Qukong\_Bijiao()  
 {  
  String s = "     hello Java    ";  
    
  //打印结果是：（hello和java前后门都有空格）hello java  
  sop(s.toUpperCase());

  //打印结果是：（HELLO和JAVA前后门都有空格）HELLO JAVA  
  sop(s.toLowerCase());

  //打印及结果是：不带空格的“hello java”  
  sop(s.trim());

  //比较数的大写，打印结果是：1，因为b对应ascii值是98，  
  //a对应是97，所以b-a=1  
  String s1 = "abc";  
  String s2 = "aaa";  
  sop(s1.compareTo(s2));

 }

 public static void method\_sub()  
 {  
  String s = "abcdef";

  //打印结果是：cdef，从指定位置开始到结尾。如果角标不存在，会出现字符串角标越界。  
  sop(s.substring(2));

  //打印结果是：cd，包含头，不包含尾。  
  sop(s.substring(2,4));  
 }  
 public static void method\_split()  
 {  
  String s = "zhangsan,lisi,wangwu";

  String[] arr = s.split(",");

  for(int x=0; x<arr.length; x++)  
  {  
   sop(arr[x]);  
  }  
 }

 public static void method\_replace()  
 {  
  String s = "hello java";

  //String s1 = s.replace('a','n');  
  //String s1 = s.replace('w','n');  如果要替换的字符不存在，返回的还是原串  
    
  String s1 = s.replace("java","world");//打印结果是：hello world

  sop("s="+s); //打印结果是：hello java因为字符串一旦被初始化，值就不可被改变  
  sop("s1="+s1);//打印结果是：hello jnvn

 }

 public static void method\_trans()  
 {  
  char[] arr = {'a','b','c','d','e','f'};

  String s = new  String(arr,1,3);

  sop("s="+s);//打印结果是：bcd

  String s1 = "zxcvbnm";  
  char[] chs = s1.toCharArray();

  for(int x=0; x<chs.length; x++)  
  {  
   sop("ch="+chs[x]);//打印结果是：ch=z，x,c,v,b,n,m  
  }  
 }

 public static void method\_is()  
 {  
  String str = "ArrayDemo.java";

 //判断文件名称是否是Array单词开头  
  sop(str.startsWith("Array"));  
   
 //判断文件名称是否是.java的文件  
  sop(str.endsWith(".java"));  
   
 //判断文件中是否包含Demo  
  sop(str.contains("Demo"));  
 }

 public static void method\_get()  
 {  
  String str = "abcdeakpf";

  //长度  
  sop(str.length());

  //根据索引获取字符  
  sop(str.charAt(4));  
  //sop(str.charAt(40));当访问到字符串中不存在的角标时会发生StringIndexOutOfBoundsException（字符串角标越界异常）

  //根据字符获取索引  
  //sop(str.indexOf('a'));  
  sop(str.indexOf('a',3));//打印的是5，因为角标3是d，  
        //所以从d后面开始找a，第5个角标是a  
  //sop(str.indexOf('t',3))打印：-1，如果没有找到角标，返回-1

  //反向索引一个字符出现的位置(从右往左查找，但是角标还是从左开始)  
  sop(str.lastIndexOf("a"));  
 }

 public static void main(String[] args)   
 {  
   method\_Zhuanhuan\_Qukong\_Bijiao();  
  //method\_sub();  
  //method\_split();  
  //method\_replace();    
  //method\_trans();   
  //method\_is();  
  //method\_get();

  /\*  
  String s1 = "abc";  
  String s2 = new String("abc");

  String s3 = "abc";  
  System.out.println(s1==s2);  
  System.out.println(s1==s3);  
  \*/  
 }

 public static void sop(Object obj)  
 {  
  System.out.println(obj);  
 }  
}

String性能分析：

字符串的拼接比较多（千万级别）时，stringbuffer.append()方法效率最高，string.concat()第二，+最低，但是，较少时，+效率最高，String中charAt（）方法效率高，比如判断某个字符串是否是该字符的开头时而不要startwith，能用该方法能解决问题，不要用其他方法解决。charAt方法的效率比startswith的效率高一赔多，字符分割尽量用stringtokens，string对象一旦确定就不能改变，而stringbuffer可以改变，当字符的拼接比较多，string中插入string，或者删除string时用stringbuffer，而且stringbuffer是线程安全的，stringbuffer和stringbuilder都继承abstractStringBuilder，但是stringbuilder是线程不安全的。