**String类剖析**

**一、两种实例化方式：**

String str = “abc”；

String str = new String("abc");

一个字符串就是String的匿名对象。

"hello".equals(str)    一个字符串能够调用一个函数，可以看出，一个字符串是String的匿名对象。

**二、比较**

**（1）直接赋值**

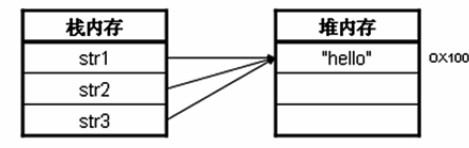
String str = “abc”；表示一个堆内存指向给了栈内存。而且abc存在于字符串常量池中

直接赋值可以节省内存

举例：

**[java]** [view plain](http://blog.csdn.net/xiazdong/article/details/6723101) [copy](http://blog.csdn.net/xiazdong/article/details/6723101)

1. **public** **class** StringDemo01{
2. **public** **static** **void** main(String args[]){
3. //享元模式
4. //在堆内存中有一个"hello"的String对象，而str1和str2和str3指向这个匿名对象
5. String str1 = "hello";
6. String str2 = "hello";
7. String str3 = "hello";
8. System.out.println("str1 == str2  --> "+(str1 == str2));
9. System.out.println("str1 == str3  --> "+(str1 == str3));
10. System.out.println("str2 == str3  --> "+(str2 == str3));
11. }
12. }

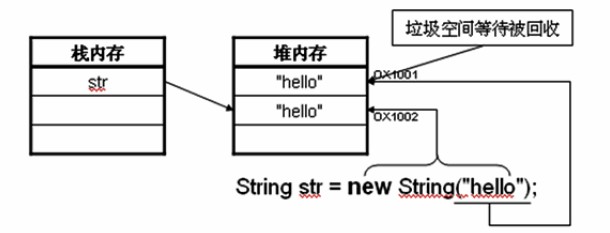


**（2）new**

举例：

**[java]** [view plain](http://blog.csdn.net/xiazdong/article/details/6723101) [copy](http://blog.csdn.net/xiazdong/article/details/6723101)

1. **public** **class** StringDemo01{
2. **public** **static** **void** main(String args[]){
4. String str1 = **new** String("hello");
5. String str2 = **new** String("hello");
6. System.out.println("str1==str2 -->"+(str1==str2));
7. }
8. }



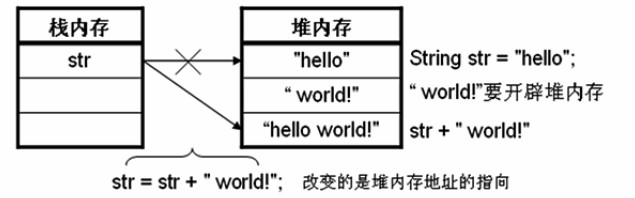
1.创建了"hello"的匿名String对象。

2.创建一个str的对象，并开辟新的堆空间。

3.原来的“hello”的匿名对象变成垃圾。

因此开发中最好使用直接赋值。

**三、字符串的不可改变性**



为什么str+="world"这种连接操作效率低，就是因为这个。

所以才会有StringBuilder或者StringBuffer。

**四、String和StringBuilder的性能比较**

**[java]** [view plain](http://blog.csdn.net/xiazdong/article/details/6723101) [copy](http://blog.csdn.net/xiazdong/article/details/6723101)

1. **public** **class** StringDemo02{
2. **public** **static** **void** main(String args[]){
3. **long** begin1 = System.currentTimeMillis();
4. String str = "";
5. **for**(**int** i=0;i<10000;i++){
6. str+=i;
7. }
8. **long** end1 = System.currentTimeMillis();
9. System.out.println("String用时： "+(end1-begin1)+" ms");    //用时2000ms

12. **long** begin2 = System.currentTimeMillis();
13. StringBuilder sbuilder = **new** StringBuilder();
14. **for**(**int** i=0;i<10000;i++){
15. sbuilder.append(i);
16. }
17. **long** end2 = System.currentTimeMillis();
18. System.out.println("StringBuilder用时： "+(end2-begin2)+" ms");//用时0ms，所以快了很多
19. }
20. }