# 什么是 Spring 框架？Spring 框架有哪些主要模块？

# 使用 Spring 框架能带来哪些好处？

# 什么是控制反转(IOC) ？什么是依赖注入？

# 请解释下 Spring 框架中的 IoC ？

# BeanFactory 和 和 ApplicationContext 有什么区别？

6 、Spring 有几种配置方式？

7 、如何用基于 XML 配置的方式配置 Spring ？

8 、如何用基于 Java 配置的方式配置 Spring ？

9 、怎样用注解的方式配置 Spring ？

10 、请解释 Spring Bean 的生命周期？

11 、Spring Bean 的作用域之间有什么区别？

12 、什么是 Spring inner beans ？

13 、Spring 框架中的单例 Beans 是线程安全的么？

14 、请举例说明如何在 Spring 中注入一个 Java Collection ？

15 、如何向 Spring Bean 中注入一个 Java.util.Properties ？

16 、请解释 Spring Bean 的自动装配？

17 、请解释自动装配模式的区别？

18 、构造方法注入和设值注入有什么区别？

19 、Spring 框架中有哪些不同类型的事件？

20 、Spring 框架中都用到了哪些设计模式？

1、spring工作机制?

2、为什么用Spring的工作机制?

3、Jdo是什么?

4、简述你对IoC（Inversion of Control）的理解，

5、简单描述Spring framework与Struts的不同之处，整合Spring与Struts有哪些方法，哪种最好，为什么?

6、Spring是什么?

7、为什么要使用Spring?

8、 简述spring的组成?9、 简述.IoC的类型?

10、简述Spring容器的功能?

11、什么是AOP?

12、简述AOP核心?

13、说出Spring的通知类型?

14、 简述ProxyFactoryBean的作用?

15、 叙述Spring对持久层支持采用的策略?

16、 叙述AOP事务的含义?

17、 如何简化事务配置?

18、 简述Spring的事务机制?

# 深入 Java 源码剖析之字符串常量

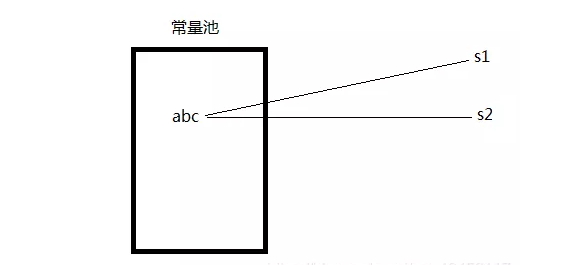
字符串在内存中的保存方式:

在内存中有一个区域叫做常量池，而当我们以这样的方式去创建字符串：

这个字符串就一定会被保存到常量池中。而Java虚拟机如果发现常量池中已经存在需要创建的字符串中，它就不会重复创建，而是指向那个字符串即可。

String s1 = **"abc"**;  
String s2 = **"abc"**;  
System.***out***.println(s1 == s2);

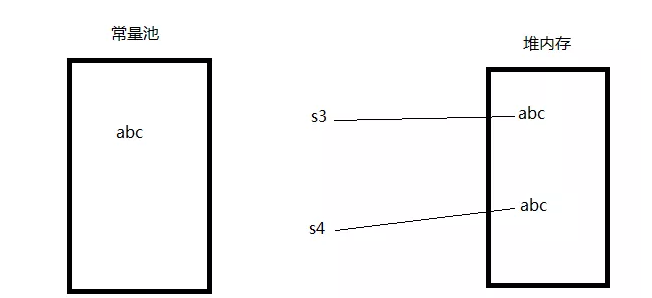
所以上述代码段的执行结果一定是true。



但是如果使用new关键字区创建字符串，过程就不太一样了。

String s3 = **new** String(**"abc"**);  
String s4 = **new** String(**"abc"**);  
System.***out***.println(s3 == s4);

过程是这样的：首先将abc保存在常量池中，此时并没有引用，然后new关键字会去创建一个字符串对象，就会在堆内存中创建abc，然后s3变量指向abc。当执行第二句声明时，因为常量池中已经存在abc，所以不会重复创建，而new关键字又会去堆内存开辟空间存放abc，然后s4变量指向abc。



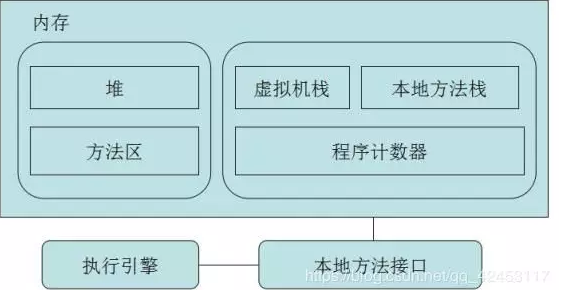
当相同的字符串常量被多次创建时，注意是使用双引号(" ")显式声明时，字符串常量对象会被保存在常量池中，且只会创建一个对象，这就是字符串驻留，这个名词的产生就是为了提升性能。简单提一下，字符串中有一个方法叫做intern()；那么这个方法有什么作用呢？该方法会去常量池中寻找当前调用该方法的字符串常量，若找到，则直接返回该字符串对象，若没有，则将当前字符串放入常量池并返回，总之该方法一定会返回字符串。

System.***out***.println(**"s3.intern() == s4.intern() "**+(s3.intern() == s4.intern()));

所以上述代码段的执行结果一定是true，因为字符串驻留只允许常量池中一个相同字符串的存在。

刚才一直在说常量池，那么常量池具体在哪呢？这就要来研究一下JVM的内存结构。

JVM分为堆、栈、方法区，栈又分为本地方法栈和Java栈。



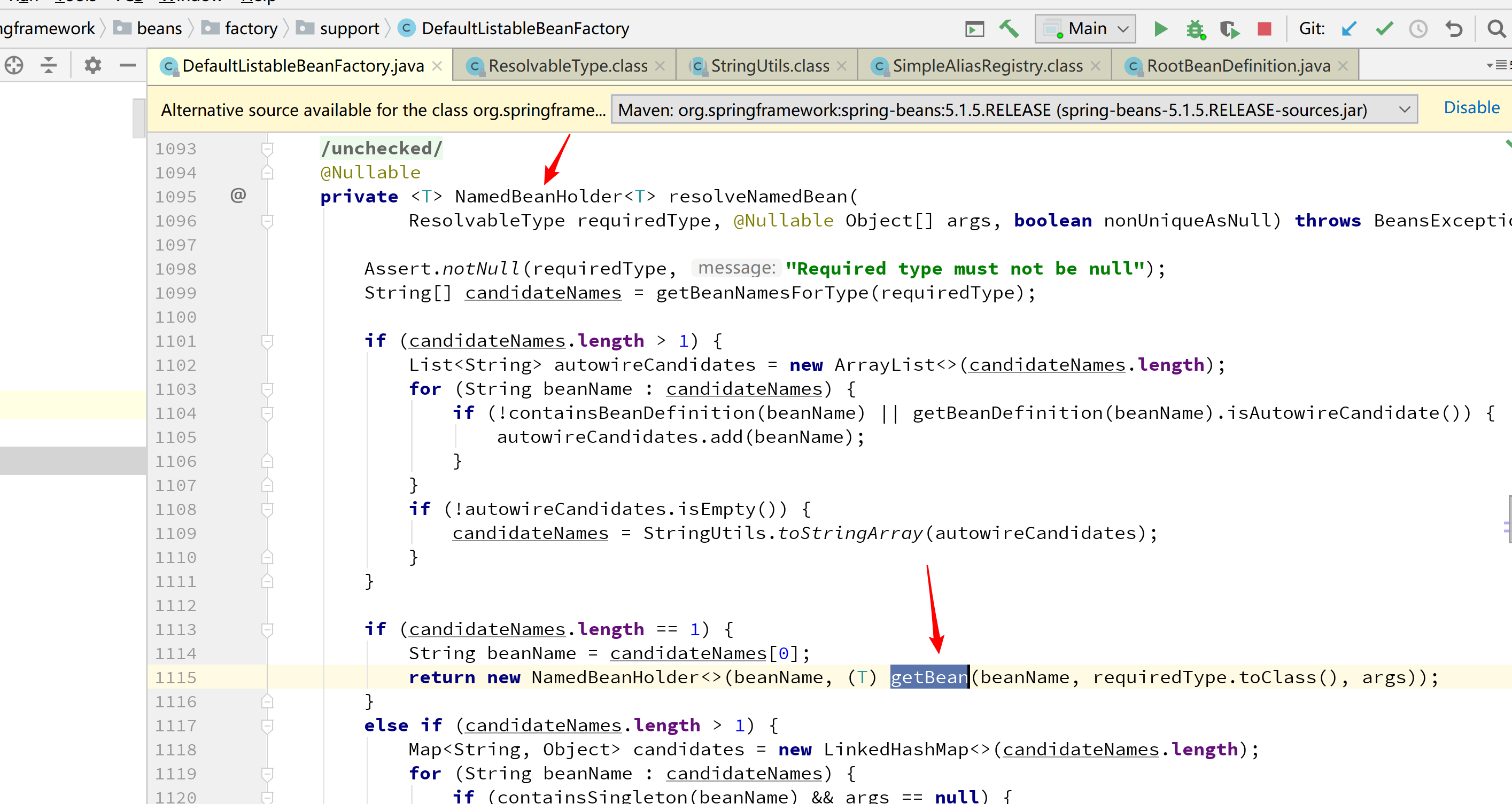
在Java7之前常量池就放在方法区里，而从Java7开始，常量池被移到了堆。这样说过于抽象，我们可以通过代码来感受这一过程。

String s5 = **new** String(**"hello"**) + **new** String(**"world"**);  
String s6 = **"helloworld"**;  
System.***out***.println(**"s5 == s6 "** + (s5 == s6));

上述程序段的执行结果一定是false。因为s5变量在堆中，而s6变量在常量池中，两者肯定不相同。

String s7 = **new** String(**"hello"**) + **new** String(**"world"**);  
System.***out***.println(**"s7.intern() == s7 "** + (s7.intern() == s7));

按照刚才的分析，intern()返回的一定是常量池里的字符串，而s7变量在堆中，它们肯定是不一样的，但运行结果竟然是true。那是不是就能解释常量池在堆中，所以它们指向的是同一个对象呢？其实还不完全是，我们可以继续看一段代码。



# Spring 为啥默认把 Bean 设计成单例的

熟悉Spring开发的朋友都知道Spring提供了5种scope分别是singleton、prototype、request、session、global session。

如下图是官方文档上的截图，感兴趣的朋友可以进去看看这五种分别有什么不同。今天要介绍的是这五种中的前两种，也是Spring最初提供的bean scope singleton 和 prototype。