# 凯讯科技

## session的工作原理，有什么好处？

在java WEB开发中，session既是重点又是难点，很多初学者都没有建立一个清晰的概念，希望通过下面这个示例对大家有所帮助。

在一个JavaWeb应用程序中新建2个jsp页面，名称分别为first.jsp和second.jsp。都是如下代码：  
< %@ page language="java" pageEncoding="gb2312"%>  
< html>  
 < head>    
   < title>First.jsp page</title>  
  </head>  
  <body>  
  <%  
    out.println(session.getId()+”<br>”);  
    out.println(session.isNew());  
  %>  
  </body>  
< /html>  
然后我们启动服务器在同一个浏览器页面中先访问first.jsp然后再访问second.jsp，结果如下：  
first.jsp  
60FF62D46CCB9088DA4BA6125E612ED4  
true  
second.jsp  
60FF62D46CCB9088DA4BA6125E612ED4  
false  
可以看到Id号码是一样的，当我们第一次访问一个Jsp页面时session对象就会创建，而以后的其他页面都是引用相同的session对象。  
打开两个浏览器页面，分别访问first.jsp和second.jsp我们可以看到这时两个页面中的id号是不一样的，而且都是true，我们可以很容易得出结论， 在相同的浏览器页面中是共享同一个session对象的。在大多数书籍中都是这样介绍session作用域的。但是情况并非全部如此，多个浏览器页面其实是完全可以共享相同的session对象的。下面我们慢慢解开谜底。  
当我们打开浏览器页面第一次访问first.jsp 页面时，容器会生成一个session对象（设置了<%@ page session=false %>指令除外），并将这个session对象的Id号码通过cookie的形式传送到客户端（但是这个cookie不会写入到客户端硬盘上，通常是存在浏览器的缓存当中，当这个页面关闭后它就随之消失，在IE中这个cookie的设置叫做会话cookie，可以尝试一下将这个会话cookie禁用掉，结果会如何），这个cookie的名称一般称为jsessionid，值当然是sesion的id号码。当客户通过此浏览器页面再次访问该web应该程序的其他页面时，就会将这个cookie传送给容器，容器就会通过这个jsessionid号码找到已经生成session对象.如果当发送的请求没有与之相关联的jsessionid时，容器才会生成一个新的session对象。如果客户端禁用掉了会话cookie那么容器就不会根据请求找到任何已经建立的session对象，所以每次都会生成新的session对象，这样会对容器的性能大打折扣，所以在这种情况下可以设置session指令，告诉容器不必为每一个页面都生成一个新的session对象。但是这种方式有一定的局限性，除非每个页面都不使用session对象，但是session的功能早已经深入人心，这种情况我们可以使用重写URL的方式，手工的给其他的页面传递jsessionid号码，这样容器就能够通过传递的id号码找到以前的 session对象，而不会再生成新的session对象了。这种方式完全可以打破同一个浏览器页面共享一个session对象的说法，多个浏览器页面也完全可以。

## 虚拟机的工作流程和原理？

## pageContext，request，session，application区别？

pageContext,request,session,application  
application和session比较简单，这里主要说明request和page的作用范围。  
application：全局作用范围，整个应用程序共享，就是在部署文件中的同一个webApp共享，生命周期为：应用程序启动到停止。  
session：会话作用域，当用户首次访问时，产生一个新的会话，以后服务器就可以记住这个会话状态。生命周期：会话超时，或者服务器端强制使会话失效。  
request：请求作用域，就是客户端的一次请求。

## 一座大桥四个人过桥的时间分别是a:1分钟 b：2分钟 c、5分钟 d：10分钟。17分钟之内必须过桥，一次只能两个人过桥。他们只有一个手电，过桥必须要手电照明。

解：先是ab过桥，a回去，然后让cd过桥，让b回去，最后a和b一起过桥。总共花去17分钟时间。

## system.gc().

1.垃圾收集算法的核心思想

　　Java语言建立了垃圾收集机制，用以跟踪正在使用的对象和发现并回收不再使用(引用)的对象。该机制可以有效防范动态内存分配中可能发生的两个危险：因内存垃圾过多而引发的内存耗尽，以及不恰当的内存释放所造成的内存非法引用。

　2.触发主GC(Garbage Collector)的条件

　　JVM进行次GC的频率很高,但因为这种GC占用时间极短,所以对系统产生的影响不大。更值得关注的是主GC的触发条件,因为它对系统影响很明显。总的来说,有两个条件会触发主GC:

　　①当应用程序空闲时,即没有应用线程在运行时,GC会被调用。因为GC在优先级最低的线程中进行,所以当应用忙时,GC线程就不会被调用,但以下条件除外。

　　②Java堆内存不足时,GC会被调用。当应用线程在运行,并在运行过程中创建新对象,若这时内存空间不足,JVM就会强制地调用GC线程,以便回收内存用于新的分配。若GC一次之后仍不能满足内存分配的要求,JVM会再进行两次GC作进一步的尝试,若仍无法满足要求,则 JVM将报“out of memory”的错误,Java应用将停止。

1. 减少GC开销的措施

　(1)不要显式调用System.gc()

　　此函数建议JVM进行主GC,虽然只是建议而非一定,但很多情况下它会触发主GC,从而增加主GC的频率,也即增加了间歇性停顿的次数。

　　(2)尽量减少临时对象的使用

　　临时对象在跳出函数调用后,会成为垃圾,少用临时变量就相当于减少了垃圾的产生,从而延长了出现上述第二个触发条件出现的时间,减少了主GC的机会。

　　(3)对象不用时最好显式置为Null

　　一般而言,为Null的对象都会被作为垃圾处理,所以将不用的对象显式地设为Null,有利于GC收集器判定垃圾,从而提高了GC的效率。

　　(4)尽量使用StringBuffer,而不用String来累加字符串(详见blog另一篇文章JAVA中String与StringBuffer)

　　由于String是固定长的字符串对象,累加String对象时,并非在一个String对象中扩增,而是重新创建新的String对象,如 Str5=Str1+Str2+Str3+Str4,这条语句执行过程中会产生多个垃圾对象,因为对次作“+”操作时都必须创建新的String对象,但这些过渡对象对系统来说是没有实际意义的,只会增加更多的垃圾。避免这种情况可以改用StringBuffer来累加字符串,因StringBuffer 是可变长的,它在原有基础上进行扩增,不会产生中间对象。

　　(5)能用基本类型如Int,Long,就不用Integer,Long对象

　　基本类型变量占用的内存资源比相应对象占用的少得多,如果没有必要,最好使用基本变量。

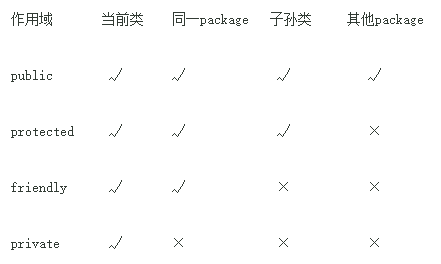
　　(6)尽量少用静态对象变量

　　静态变量属于全局变量,不会被GC回收,它们会一直占用内存。

　　(7)分散对象创建或删除的时间

　　集中在短时间内大量创建新对象,特别是大对象,会导致突然需要大量内存,JVM在面临这种情况时,只能进行主GC,以回收内存或整合内存碎片, 从而增加主GC的频率。集中删除对象,道理也是一样的。它使得突然出现了大量的垃圾对象,空闲空间必然减少,从而大大增加了下一次创建新对象时强制主GC 的机会。

## 六、public、protected、private、final、static有哪些作用,分别作用于class、variable、function的作用？



1. public：任何其他类、对象只要可以看到这个类的话，那么它就可以存取变量的数据，或使用方法。
2. protected变量和方法：

如果一个类中变量或方法有修饰字protected，同一类、同一包可以使用。不同包的类要使用，必须是该类的子类，可以存取变量或调用。

final：

final在方法之前,防止该方法被覆盖；

final在类之前,标是该类不能被继承；

final在变量之前,定义一个常量。

static：

在变量或方法之前,表明它们是属于类的；

静态变量在各实例间共享，如果是public静态变量,则其它类可以不通过实例化访问它们；

静态方法称为类的方法，因此不用实例化即可调用(面向过程)

一个对象的方法可以访问对象的数据成员,尽管不属于方法的局部变量；一个类的方法只能访问自己的局部变量。

# 易保科技

## 一、图是如何存储的？如从图中找到最短路径？

1、邻接矩阵方法：用两个数组存储。一个一维数组存储节点。一个二维数组存储节点之间的关系。用广度遍历和深度遍历进行查找最小路径、迪杰斯特拉算法处理无向图中最短路径

2、邻接表方法：

## 二、JVM执行流程？

## 三、JVM垃圾回收机制？

## 四、Java执行程序时，内存如果分配？

## 五、Mysql左连接和右连接？

## 六、Mysql中group by 和 having语句？

Group by是分组 having是进行条件过滤

## 七、编写代码：把一篇英语文章赋值给数组？输出？要求：指定每行最多max字符，最后一个单词不可以分开？

## 八、问到了我的具体事情，比如：详细介绍你是计算机副部长，怎么做了？

## 九、建模大赛如何做的？怎么建模的?现场演示？

# 面试总结

## 一、String是最基本的数据类型吗?

String不是基本的数据类型。

基本的数据类型只有8种，byte、short、int、long、char、float、double、boolean，

基本数据类型仅仅只有8种。到1.8以后有增加了一种void。

## String、StringBuffer、stringbuilder的区别？

String的变量不可改变

stringbuffer、和stringbuilder的变量是可以改变的

stringbuffer线程安全的

stringbuilder是线程非安全的

Stringbuilder效率最高

## 三、&和&&的区别？

&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与（and）。

## 四、HashMap和Hashtable的区别？

HashMap是Hashtable的轻量级实现（非线程安全的实现），他们都完成了Map接口，主要区别在于HashMap允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。

HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。

Hashtable继承自Dictionary类，而HashMap是Java1.2引进的Map?interface的一个实现。

不同是，Hashtable的方法是Synchronize的，而HashMap不是，在多个线程访问Hashtable时，不需要自己为它的方法实现同步，而HashMap?必须为之提供外同步。

## 五、sleep()?和?wait()?有什么区别??

sleep是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用sleep不会释放对象锁。

wait是Object类的方法，对此对象调用wait方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出notify方法（或notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

## 六、heap（堆）和stack（栈）有什么区别？

栈是一种线形集合，其添加和删除元素的操作应在同一段完成。栈按照后进先出的方式进行处理。堆是栈的一个组成元素

## 七、如何创建线程？

◆需要从Java.lang.Thread类派生一个新的线程类，重载它的run()方法；

◆实现Runnalbe接口，重载Runnalbe接口中的run()方法。