

一、多线程相关

1、java事件机制包括哪三个部分？ 分别介绍。

java事件机制包括三个部分：事件、事件监听器、事件源。


- 1、事件。一般继承自java.util.EventObject类，封装了事件源对象及跟事件相关的信息。
- 2、事件监听器。实现java.util.EventListener接口,注册在事件源上,当事件源的属性或状态改变时,取得相应的监听器调用其内部的回调方法。
- 3、事件源。事件发生的地方，由于事件源的某项属性或状态发生了改变(比如BUTTON被单击、TEXTBOX的值发生改变等等)导致某项事件发生。换句话说就是生成了相应的事件对象。因为事件监听器要注册在事件源上,所以事件源类中应该要有盛装监听器的容器(List,Set等等)。

2、为什么要使用线程池？

3、线程池有什么作用？

4、说说几种常见的线程池及使用场景。

Java 中有哪几种线程池，分别用在什么场景？

 https://m.toutiao.com/i6779072117875933709/?traffic_source=CS1114&in_ogs=1&utm_source=HW&source=search_tab&utm_medium=wap_search&prevent_activate=1&original_source=1&in_tfs=HW&channel=

newCachedThreadPool

- 重用之前的线程
- 适合执行许多短期异步任务的程序。
- 调用 execute() 将重用以前构造的线程
- 如果没有可用的线程，则创建一个新线程并添加到池中
- 默认为60s未使用就被终止和移除
- 长期闲置的池将会不消耗任何资源

newFixedThreadPool

- 创建重用固定数量线程的线程池，不能随时新建线程
- 当所有线程都处于活动状态时，如果提交了其他任务，他们将在队列中等待一个线程可用
- 线程会一直存在，直到调用shutdown

newWorkStealingPool

- 获取当前可用的线程数量进行创建作为并行级别
- 使用ForkJoinPool
- 使用一个无限队列来保存需要执行的任务，可以传入线程的数量，不传入，则默认使用当前计算机中可用的cpu数量，使用分治法来解决问题，使用fork()和join()来进行调用

newSingleThreadExecutor

- 在任何情况下都不会有超过一个任务处于活动状态
 - 与newFixedThreadPool(1)不同是不能重新配置加入线程，使用FinalizableDelegatedExecutorService进行包装
 - 能保证执行顺序，先提交的先执行。
 - 当线程执行中出现异常，去创建一个新的线程替换之
- 源码：

newScheduledThreadPool

- 设定延迟时间，定期执行
- 空闲线程会进行保留

通过源码可以看出底层调用了ThreadPoolExecutor，维护了一个延迟队列，可以传入线程数量，传入延时的时间等参数，下面给出一个demo

```
1 public static void main(String[] args) {
2     ScheduledExecutorService pool = Executors.newScheduledThreadPool(5);
3     for (int i = 0; i < 15; i = i + 5) {
4         pool.schedule(() -> System.out.println("我被执行了，当前时间" + new
5             Date()), i, TimeUnit.SECONDS);
6     }
7     pool.shutdown();
8 }
```

执行结果

```
1 我被执行了，当前时间Fri Jan 12 11:20:41 CST 2018
2 我被执行了，当前时间Fri Jan 12 11:20:46 CST 2018
3 我被执行了，当前时间Fri Jan 12 11:20:51 CST 2018
```

五种线程池的适应场景

1. newCachedThreadPool：用来创建一个可以无限扩大的线程池，适用于服务器负载较轻，执行很多短期异步任务。
2. newFixedThreadPool：创建一个固定大小的线程池，因为采用无界的阻塞队列，所以实际线程数量永远不会变化，适用于可以预测线程数量的业务中，或者服务器负载较重，对当前线程数量进行限制。
3. newSingleThreadExecutor：创建一个单线程的线程池，适用于需要保证顺序执行各个任务，并且在任意时间点，不会有多个线程是活动的场景。
4. newScheduledThreadPool：可以延时启动，定时启动的线程池，适用于需要多个后台线程执行周期任务的场景。
5. newWorkStealingPool：创建一个拥有多个任务队列的线程池，可以减少连接数，创建当前可用cpu数量的线程来并行执行，适用于大耗时的操作，可以并行来执行

5、线程池都有哪几种工作队列？

6、怎么理解无界队列和有界队列？

7、线程池中的几种重要的参数及流程说明。

二、网络编程相关

11、你怎么理解http协议？

12、说说http协议的工作流程。

13、http有哪些请求提交方式？

14、http中的200,302,403,404,500,503都代表什么状态？

http的各种请求状态码

![img]

(file:///C:/Users/zy/AppData/Roaming/Tencent/QQTempSys/%W@GJ\$ACOF{TYDYECOKVDYB.png)<https://www.cnblogs.com/imxiu/p/3541951.html>

15、http get和post有什么区别？

GET方式主要用于获取网络资源，POST方式主要用于表单提交。由于GET方式的参数是在地址栏中的，所以总是可见的，不是很安全，而且长度也有限制。而POST方式的参数是封装成实体之后发送给服务器的，是不可见的，相对比较安全，用户的敏感信息一般采用POST方式提交。

16、你怎么理解cookie和session，有哪些不同点？

90%程序员都没有完全答对Cookie和Session的区别

✓https://m.toutiao.com/i6693986851193094664/?traffic_source=CS1114&in_ogs=1&utm_source=HW&source=search_tab&utm_medium=wap_search&prevent_activate=1&original_source=1&in_tfs=HW&channel=

Cookie 和 Session 有什么不同？

- 作用范围不同，Cookie 保存在客户端（浏览器），Session 保存在服务器端。
- 存取方式的不同，Cookie 只能保存 ASCII，Session 可以存任意数据类型，一般情况下我们可以在 Session 中保持一些常用变量信息，比如说 UserId 等。
- 有效期不同，Cookie 可设置为长时间保持，比如我们经常使用的默认登录功能，Session 一般失效时间较短，客户端关闭或者 Session 超时都会失效。
- 隐私策略不同，Cookie 存储在客户端，比较容易遭到不法获取，早期有人将用户的登录名和密码存储在 Cookie 中导致信息被窃取；Session 存储在服务端，安全性相对 Cookie 要好一些。
- 存储大小不同，单个 Cookie 保存的数据不能超过 4K，Session 可存储数据远高于 Cookie。

17、什么是web缓存？有什么优点？

18、什么是https，说说https的工作原理？

19、什么是http代理服务器，有什么用？

20、什么是虚拟主机及实现原理？

21、什么是Java虚拟机，为什么要使用？

22、说说Java虚拟机的生命周期及体系结构。

三、JVM虚拟机相关

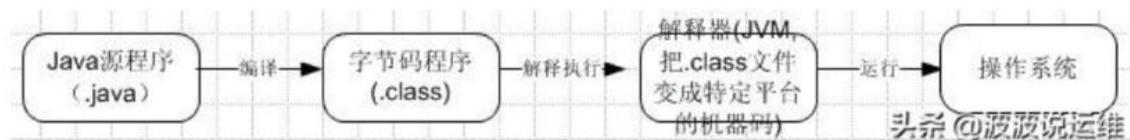
8、什么是反射机制？

9、说说反射机制的作用。

10、反射机制会不会有性能问题？

2. Java语言运行的过程

Java语言写的源程序通过Java编译器，编译成与平台无关的‘字节码程序’(.class文件，也就是0，1二进制程序)，然后在OS之上的Java解释器中解释执行。



23、说一说Java内存区域。

24、什么是分布式系统？

25、分布式系统你会考虑哪些方面？

35、Java虚拟机中，数据类型可以分为哪几类？

36、怎么理解栈、堆？堆中存什么？栈中存什么？

1) 堆中存的是对象。栈中存的是基本数据类型和堆中对象的引用。一个对象的大小是不可估计的，或者说是可以动态变化的，但是在栈中，一个对象只对应了一个4byte的引用。

37、为什么要把堆和栈区分出来呢？栈中不是也可以存储数据吗？

38、在Java中，什么是栈的起始点，同是也是程序的起始点？

程序要运行总是有一个起点的。同C语言一样，java中的Main就是那个起点。无论什么java程序，找到main就找到了程序执行的入口

39、为什么不把基本类型放堆中呢？

2) 为什么不把基本类型放堆中呢？因为其占用的空间一般是1~8个字节——需要空间比较少，而且因为是基本类型，所以不会出现动态增长的情况——长度固定，因此栈中存储就够了，如果把他存在堆中是没有什么意义的（还会浪费空间，后面说明）。可以这么说，基本类型和对象的引用都是存放在栈中，而且都是几个字节的一个数，因此在程序运行时，他们的处理方式是统一的。但是基本类型、对象引用和对象本身就有所区别了，因为一个是栈中的数据一个是堆中的数据。最常见的一个问题就是，Java中参数传递时的问题。

40、Java中的参数传递时传值呢？还是传引用？

3) Java中的参数传递时传值呢？还是传引用？程序运行永远都是在栈中进行的，因而参数传递时，只存在传递基本类型和对象引用的问题。不会直接传对象本身。

Java在方法调用传递参数时，因为没有指针，所

41、Java中有没有指针的概念？

42、Java中，栈的大小通过什么参数来设置？

43、一个空Object对象的占多大空间？

- 44、对象引用类型分为哪几类？
- 45、讲一讲垃圾回收算法。
- 46、如何解决内存碎片的问题？
- 47、如何解决同时存在的对象创建和对象回收问题？
- 48、讲一讲内存分代及生命周期。
- 49、什么情况下触发垃圾回收？
- 50、如何选择合适的垃圾收集算法？
- 51、JVM中最大堆大小有没有限制？
- 52、堆大小通过什么参数设置？
- 53、JVM有哪三种垃圾回收器？
- 54、吞吐量优先选择什么垃圾回收器？响应时间优先呢？
- 55、如何进行JVM调优？有哪些方法？
- 56、如何理解内存泄漏问题？有哪些情况会导致内存泄露？如何解决？