# JAVA面试题总结

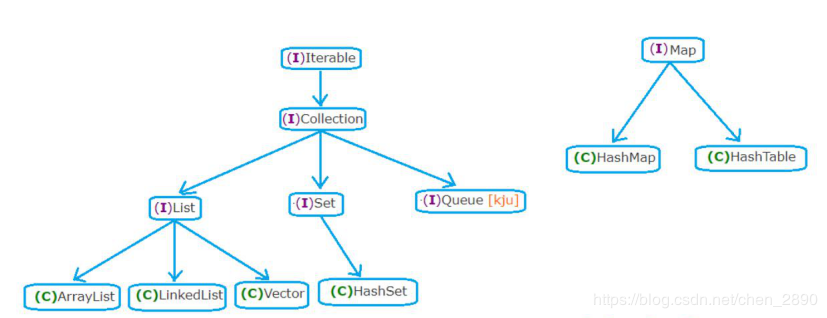
1. 基本数据类型和包装类型

数值型:byte short int long float double

字符型:char 布尔型:boolean

包装类型是对基本类型的补充,基本类型是值传递,包装类型是引用传递,同时提供了很多数据类型的转换方法,JAVA1.5以后可以自动装箱和拆箱

1. 集合



List：有序、可重复。可以通过索引快速查找，但进行增删操作时后续的数据需要移动，所以增删速度慢。

Set：无序、不可重复。

Map：键值对、键唯一、值不唯一。Map 集合中存储的是键值对，键不能重复，值可以重复。根据键得到值，对 map 集合遍历时先得到键的 set 集合，对 set 集合进行遍历，得到相应的值。

2.1 ArrayList和LinkedList的区别

ArrayList 是基于动态数组的数据结构，LinkedList 是基于链表的数据结构；对于随机访问 get 和 set，ArrayList 较优，因为 LinkedList 要移动指针；对于新增和删除操作 add 和 remove，LinedList 较优，因为ArrayList 要移动数据。

2.2HashMap和Hashtable的区别

HashMap允许空键值,Hashtable不允许;

HashMap继承自AbstractMap,Hashtable继承自Dictionary,两者实现了Map接口

HashMap的方法不是同步的,Hashtable的方法时同步的

Jdk1.5又引入了ConcurrencyHashMap，其弥补了HashMap线程不安全、HashTable性能低的缺失，是目前高性能的线程安全的HashMap类。

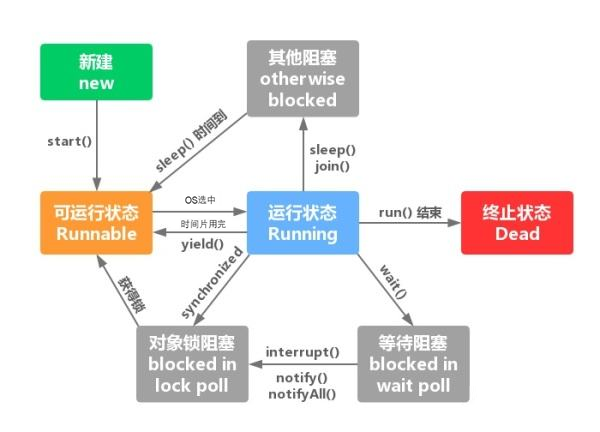
2.3 Hash冲突也称为Hash碰撞,当产生的两个不同的元素通过Hash函数得到的实际存储地址相同.在插入时发现位置已被占用,采用链地址法(数组+链表方法)解决

1. 多线程

一个操作系统中可以有多个进程，一个进程中可以有多个线程，每个进程有自己独立的内存，每个线程共享一个进程中的内存，每个线程又有自己独立的内存。

线程生命周期，总共有五种状态：

1. 新建状态（New）：当线程对象对创建后，即进入了新建状态，如：Thread t = new MyThread()；
2. 就绪状态（Runnable）：当调用线程对象的start()方法（t.start();），线程即进入就绪状态。处于就绪状态的线程，只是说明此线程已经做好了准备，随时等待CPU调度执行，并不是说执行了t.start()此线程立即就会执行；
3. 运行状态（Running）：当CPU开始调度处于就绪状态的线程时，此时线程才得以真正执行，即进入到运行状态。注：就绪状态是进入到运行状态的唯一入口，也就是说，线程要想进入运行状态执行，首先必须处于就绪状态中；
4. 阻塞状态（Blocked）：处于运行状态中的线程由于某种原因，暂时放弃对CPU的使用权，停止执行，此时进入阻塞状态，直到其进入到就绪状态，才有机会再次被CPU调用以进入到运行状态；
5. 根据阻塞产生的原因不同，阻塞状态又可以分为三种：
   1. 等待阻塞：运行状态中的线程执行wait()方法，使本线程进入到等待阻塞状态；
   2. 同步阻塞：线程在获取synchronized同步锁失败(因为锁被其它线程所占用)，它会进入同步阻塞状态；
   3. 其他阻塞：通过调用线程的sleep()或join()或发出了I/O请求时，线程会进入到阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。
6. 死亡状态（Dead）：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。



* 启动线程不要自己调用run()方法，那样就成普通方法了，只能等待OS操作系统自己去调用
* 如果start()一个线程后想其立即执行，技巧：Thread.sleep(1)
* 一个多处理器的机器上，将会有多个线程并行（Parallel）执行，当线程数大于处理器数时，才会出现多个线程并发（Concurrent），在一个CPU上轮换执行
* 并行和并发的差异，一个是同时执行，一个是交替执行
* 线程在运行中，一般会被中断，目的是给其他线程执行机会，雨露均沾
* 线程调用sleep()方法主动放弃所占用的CPU资源
* 调用yield()方法可以让运行状态的线程转入就绪状态
* join()方法是一个线程等待另外一个线程
  1. wait()与sleep()的区别

sleep()来自 Thread 类，wait()来自 Object 类；调用 sleep()方法，线程不会释放对象锁。而调用 wait 方法线程会释放对象锁；sleep()睡眠后不出让系统资源，wait 让其他线程可以占用 CPU；sleep(milliseconds)需要指定一个睡眠时间，时间一到会自动唤醒。而 wait()需要配合 notify()或者 notifyAll()使用。

1. 什么是GC?为什么要有GC?

GC（Garbage Collection）是垃圾收集的意思，负责清除对象并释放内存。Java 提供的 GC 功能可以自动检测对象是否超过作用域从而达到自动回收内存的目的，从而防止内存泄漏。

1. heap和stack有什么区别

栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。

堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用new创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用final修饰后，放在堆中，而不是栈中。

1. final finally和finalized的区别

（1）final：被final修饰的类，不被能继承；被final修饰的方法，不能被重写；被fianl修饰的量，为常量，只能被赋值一次；

（2）finally：异常处理，和try、catch结合使用，可加可不加，用于执行一些必须执行的代码，如释放资源等；

（3）finalized：Object类中的方法，其中定义了对象要被垃圾回收器回收之前，要做的一些清理工作。

1. GC线程是否为守护线程

GC线程是守护线程。线程分为守护线程和非守护线程（即用户线程）。只要当前JVM实例中尚存在任何一个非守护线程没有结束，守护线程就全部工作；只有当最后一个非守护线程结束时，守护线程随着JVM一同结束工作。

1. ReenTrantLock可重入锁和synchronized的区别

可重入性：

从名字上理解，ReenTrantLock的字面意思就是再进入的锁，其实synchronized关键字所使用的锁也是可重入的，两者关于这个的区别不大。两者都是同一个线程没进入一次，锁的计数器都自增1，所以要等到锁的计数器下降为0时才能释放锁。

锁的实现：

Synchronized是依赖于JVM实现的，而ReenTrantLock是JDK实现的，有什么区别，说白了就类似于操作系统来控制实现和用户自己敲代码实现的区别。前者的实现是比较难见到的，后者有直接的源码可供阅读。

性能的区别：

在Synchronized优化以前，synchronized的性能是比ReenTrantLock差很多的，但是自从Synchronized引入了偏向锁，轻量级锁（自旋锁）后，两者的性能就差不多了，在两种方法都可用的情况下，官方甚至建议使用synchronized，其实synchronized的优化我感觉就借鉴了ReenTrantLock中的CAS技术。都是试图在用户态就把加锁问题解决，避免进入内核态的线程阻塞。

功能区别：

便利性：很明显Synchronized的使用比较方便简洁，并且由编译器去保证锁的加锁和释放，而ReenTrantLock需要手工声明来加锁和释放锁，为了避免忘记手工释放锁造成死锁，所以最好在finally中声明释放锁。

锁的细粒度和灵活度：很明显ReenTrantLock优于Synchronized

ReenTrantLock独有的能力：

1.      ReenTrantLock可以指定是公平锁还是非公平锁。而synchronized只能是非公平锁。所谓的公平锁就是先等待的线程先获得锁。

2.      ReenTrantLock提供了一个Condition（条件）类，用来实现分组唤醒需要唤醒的线程们，而不是像synchronized要么随机唤醒一个线程要么唤醒全部线程。

3.      ReenTrantLock提供了一种能够中断等待锁的线程的机制，通过lock.lockInterruptibly()来实现这个机制。

ReenTrantLock实现的原理：

在网上看到相关的源码分析，本来这块应该是本文的核心，但是感觉比较复杂就不一一详解了，简单来说，ReenTrantLock的实现是一种自旋锁，通过循环调用CAS操作来实现加锁。它的性能比较好也是因为避免了使线程进入内核态的阻塞状态。想尽办法避免线程进入内核的阻塞状态是我们去分析和理解锁设计的关键钥匙。

什么情况下使用ReenTrantLock：

答案是，如果你需要实现ReenTrantLock的三个独有功能时。

1. Collection和Collections的区别

Collection是一个接口，它是Set、List等容器的父接口；Collections是个一个工具类，提供了一系列的静态方法来辅助容器操作，这些方法包括对容器的搜索、排序、线程安全化等

1. Error和Exception的区别

Error表示系统级的错误和程序不必处理的异常，是恢复不是不可能但很困难的情况下的一种严重问题；比如内存溢出，不可能指望程序能处理这样的情况；

Exception表示需要捕捉或者需要程序进行处理的异常，是一种设计或实现问题；也就是说，它表示如果程序运行正常，从不会发生的情况。

1. Iterator和ListIterator的区别

Iterator可用来遍历Set和List集合，但是ListIterator只能用来遍历List。

Iterator对集合只能是前向遍历，ListIterator既可以前向也可以后向。

ListIterator实现了Iterator接口，并包含其他的功能，比如：增加元素，替换元素，获取前一个和后一个元素的索引，等等。

1. (高级)j2ee常用的设计模式?说明工厂模式

Java 中的23 种设计模式：  
Factory（工厂模式）， Builder（建造模式）， Factory Method（工厂方法模式），  
Prototype（原始模型模式），Singleton（单例模式）， Facade（门面模式），  
Adapter（适配器模式）， Bridge（桥梁模式）， Composite（合成模式），  
Decorator（装饰模式）， Flyweight（享元模式）， Proxy（代理模式），  
Command（命令模式）， Interpreter（解释器模式）， Visitor（访问者模式），  
Iterator（迭代子模式）， Mediator（调停者模式）， Memento（备忘录模式），  
Observer（观察者模式）， State（状态模式）， Strategy（策略模式），  
Template Method（模板方法模式）， Chain Of Responsibleity（责任链模式）  
工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例，通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

1. Jar包的作用是什么

JAR（Java Archive，Java 归档文件）是与平台无关的文件格式，它允许将许多文件组合成一个压缩文件。为 J2EE 应用程序创建的 JAR 文件是 EAR 文件（企业 JAR 文件）。  
1) 用于发布和使用类库  
2) 作为应

用程序和扩展的构建单元  
3) 作为组件、applet 或者插件程序的部署单位  
4) 用于打包与组件相关联的辅助资源

1. Java.lang.ThreadLocal的作用和原理?列举在哪些程序中见过ThreadLocal使用

类java.lang.ThreadLocal的作用

要编写一个多线程安全(Thread-safe)的程序是困难的，为了让线程共享资源，必须小心地对共享资源进行同步，同步带来一定的效能延迟，而另一方面，在处理同步的时候，又要注意对象的锁定与释放，避免产生死结，种种因素都使得编写多线程程序变得困难。

尝试从另一个角度来思考多线程共享资源的问题，既然共享资源这么困难，那么就干脆不要共享，何不为每个线程创造一个资源的复本。将每一个线程存取数据的行为加以隔离，实现的方法就是给予每个线程一个特定空间来保管该线程所独享的资源。

比如：在Hibernate中的Session就有使用。

ThreadLocal的原理

ThreadLocal是如何做到为每一个线程维护变量的副本的呢？其实实现的思路很简单，在ThreadLocal类中有一个Map，用于存储每一个线程的变量的副本。

1. Java.util.Map的实现类有哪些?HashMap,Hashtable,LinkedHashMap,TreeMap
2. Java.sql.Date和Java.util.Date的联系和区别

1） java.sql.Date是java.util.Date的子类，是一个包装了毫秒值的瘦包装器，允许 JDBC 将毫秒值标识为 SQL DATE 值。毫秒值表示自 1970 年 1 月 1 日 00:00:00 GMT 以来经过的毫秒数。 为了与 SQL DATE 的定义一致，由 java.sql.Date 实例包装的毫秒值必须通过将时间、分钟、秒和毫秒设置为与该实例相关的特定时区中的零来“规范化”。 说白了，java.sql.Date就是与数据库Date相对应的一个类型，而java.util.Date是纯java的Date。

2）JAVA里提供的日期和时间类，java.sql.Date和java.sql.Time,只会从数据库里读取某部分值，这有时会导致丢失数据。例如一个包含2002/05/22 5:00:57 PM的字段，读取日期时得到的是2002/05/22,而读取时间时得到的是5:00:57 PM. 你需要了解数据库里存储时间的精度。有些数据库，比如MySQL,精度为毫秒，然而另一些数据库，包括Oracle,存储SQL DATE类型数据时，毫秒部分的数据是不保存的。以下操作中容易出现不易被发现的BUG：获得一个JAVA里的日期对象。 从数据库里读取日期 试图比较两个日期对象是否相等。如果毫秒部分丢失，本来认为相等的两个日期对象用Equals方法可能返回false。.sql.Timestamp类比java.util.Date类精确度要高。这个类包了一个getTime()方法，但是它不会返回额外精度部分的数据，因此必须使用...

总之，java.util.Date 就是Java的日期对象，而java.sql.Date 是针对SQL语句使用的，只包含日期而没有时间部分。

1. Java和JavaScript的比较

JavaScript 与Java是两个公司开发的不同的两个产品。Java 是原Sun 公司推出的面向对象的程序设计语言，特别适合于互联网应用程序开发；而JavaScript是Netscape公司的产品，为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言，它的前身是LiveScript；而Java 的前身是Oak语言。

下面对两种语言间的异同作如下比较：

1）基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object-Based）和事件驱动（Event-Driven）的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用；

2）解释和编译：Java 的源代码在执行之前，必须经过编译；JavaScript 是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行；

3）强类型变量和类型弱变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量声明，采用其弱类型。即变量在使用前不需作声明，而是解释器在运行时检查其数据类型；

4）代码格式不一样。

补充：上面列出的四点是原来所谓的标准答案中给出的。其实Java和JavaScript最重要的区别是一个是静态语言，一个是动态语言。目前的编程语言的发展趋势是函数式语言和动态语言。在Java中类（class）是一等公民，而JavaScript中函数（function）是一等公民。对于这种问题，在面试时还是用自己的语言回答会更加靠谱。

1. Java中final和abstract关键字的作用

final和abstract关键字的作用

final和abstract是功能相反的两个关键字，可以对比记忆

abstract可以用来修饰类和方法，不能用来修饰属性和构造方法；使用abstract修饰的类是抽象类，需要被继承，使用abstract修饰的方法是抽象方法，需要子类被重写。

final可以用来修饰类、方法和属性，不能修饰构造方法。使用final修饰的类不能被继承，使用final修饰的方法不能被重写，使用final修饰的变量的值不能被修改，所以就成了常量。

特别注意：final修饰基本类型变量，其值不能改变，由原来的变量变为常量；但是final修饰引用类型变量，栈内存中的引用不能改变，但是所指向的堆内存中的对象的属性值仍旧可以改变。例如

class Test {  
public static void main(String[] args) {  
final Dog dog = new Dog("欧欧");  
dog.name = "美美";//正确  
dog = new Dog("亚亚");//错误  
} }

1. Static关键字的作用

static可以修饰变量、方法、代码块和内部类

static属性属于这个类所有，即由该类创建的所有对象共享同一个static属性。可以对象创建后通过对象名.属性名和类名.属性名两种方式来访问。也可以在没有创建任何对象之前通过类名.属性名的方式来访问。

static变量和非static变量的区别(都是成员变量，不是局部变量)

1.在内存中份数不同

不管有多少个对象，static变量只有1份。对于每个对象，实例变量都会有单独的一份

static变量是属于整个类的，也称为类变量。而非静态变量是属于对象的，也称为实例变量

2.在内存中存放的位置不同

2.在内存中存放的位置不同

3.访问的方式不同

实例变量： 对象名.变量名 stu1.name="小明明";

静态变量：对象名.变量名 stu1.schoolName="西二旗小学"; 不推荐如此使用

类名.变量名 Student.schoolName="东三旗小学"; 推荐使用

4.在内存中分配空间的时间不同

Student.schoolName="东三旗小学";或者Student stu1 = new Student("小明","男",20,98);

static方法也可以通过对象名.方法名和类名.方法名两种方式来访问

static代码块。当类被第一次使用时（可能是调用static属性和方法，或者创建其对象）执行静态代码块，且只被执行一次，主要作用是实现static属性的初始化。

static内部类：属于整个外部类，而不是属于外部类的每个对象。不能访问外部类的非静态成员（变量或者方法），.可以访问外部类的静态成员

1. Java中this和super关键字的作用

this是对象内部指代自身的引用,同时也是解决成员变量和局部变量同名问题；this可以调用成员变量，不能调用局部变量；this也可以调用成员方法，但是在普通方法中可以省略this，在构造方法中不允许省略，必须是构造方法的第一条语句。，而且在静态方法当中不允许出现this关键字。

super代表对当前对象的直接父类对象的引用，super可以调用直接父类的成员变量（注意权限修饰符的影响，比如不能访问private成员）

super可以调用直接父类的成员方法（注意权限修饰符的影响，比如不能访问private成员）；super可以调用直接父类的构造方法，只限构造方法中使用，且必须是第一条语句。

1. Java中会存在内存泄露吗,请简单说明?理论上Java因为有垃圾回收机制（GC）不会存在内存泄露问题（这也是Java被广泛使用于服务器端编程的一个重要原因）；然而在实际开发中，可能会存在无用但可达的对象，这些对象不能被GC回收，因此也会导致内存泄露的发生。例如Hibernate的Session（一级缓存）中的对象属于持久态，垃圾回收器是不会回收这些对象的，然而这些对象中可能存在无用的垃圾对象，如果不及时关闭（close）或清空（flush）一级缓存就可能导致内存泄露。
2. Java中如何实现序列化,有什么意义?

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决对象流读写操作时可能引发的问题（如果不进行序列化可能会存在数据乱序的问题）。

要实现序列化，需要让一个类实现Serializable接口，该接口是一个标识性接口，标注该类对象是可被序列化的，然后使用一个输出流来构造一个对象输出流并通过writeObject(Object)方法就可以将实现对象写出（即保存其状态）；如果需要反序列化则可以用一个输入流建立对象输入流，然后通过readObject方法从流中读取对象。

例如，在web开发中，如果对象被保存在了Session中，tomcat在重启时要把Session对象序列化到硬盘，这个对象就必须实现Serializable接口。如果对象要经过分布式系统进行网络传输或通过rmi等远程调用，这就需要在网络上传输对象，被传输的对象就必须实现Serializable接口。

1. Java中实现多态的机制是什么?

靠的是父类或接口定义的引用变量可以指向子类或具体实现类的实例对象，而程序调用的方法在运行期才动态绑定，就是引用变量所指向的具体实例对象的方法，也就是内存里正在运行的那个对象的方法，而不是引用变量的类型中定义的方法。

1. Java中实现线程通信的三个方法的作用是什么?

Java提供了3个方法解决线程之间的通信问题，均是java.lang.Object类的方法，都只能在同步方法或者同步代码块中使用，否则会抛出异常。

|  |  |
| --- | --- |
| 方法名 | 作 用 |
| final void wait() | 表示线程一直等待，直到其它线程通知 |
| void wait(long timeout) | 线程等待指定毫秒参数的时间 |
| final void wait(long timeout,int nanos) | 线程等待指定毫秒、微妙的时间 |
| final void notify() | 唤醒一个处于等待状态的线程。注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且不是按优先级。 |
| final void notifyAll() | 唤醒同一个对象上所有调用wait()方法的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争 |

1. Java接口的作用

通过接口可以实现不相关类的相同行为，而不需要了解对象所对应的类。

通过接口可以指明多个类需要实现的方法。

通过接口可以了解对象的交互界面，而不需了解对象所对应的类。

另：Java是单继承，接口可以使其实现多继承的功能。

1. Java中有几种类型的流?

1）字节流 InputStream/OutputStream

 FileInputStream/FileOutputStream：文件字节流，用于文件的读写操作

 BufferedInputStream/BufferedFileOutputStream：加缓冲区的字节流，用于提高效率

2）字符流 Reader/Writer

 FileReader/FileWriter：文件字符流，用于文本文件的读写操作

 BufferedReader/BufferedWriter：加缓冲区的字符流，用于提高效率

1. 转换流 InputStreamReader/OutputStreamWriter
2. Java中用到的线程调度算法是什么?

抢占式。一个线程用完CPU之后，操作系统会根据线程优先级、线程饥饿情况等数据算出一个总的优先级并分配下一个时间片给某个线程执行。

1. Java的两种异常类型是什么?有什么区别?Exception和Error有什么区别?

Java中有两种异常：受检查的(checked)异常和不受检查的(unchecked)异常。不受检查的异常不需要在方法或者是构造函数上声明，就算方法或者是构造函数的执行可能会抛出这样的异常。而且不受检查的异常可以传播到方法或者是构造函数的外面。相反，受检查的异常必须要用throws语句在方法或者是构造函数上声明。  
Exception和Error都是Throwable的子类。Exception用于用户程序可以捕获的异常情况。Error定义了不期望被用户程序捕获的异常。

1. 说说http,https协议

HTTP：

是互联网上应用最为广泛的一种网络协议，是一个客户端和服务器端请求和应答的标准（TCP），用于从WWW服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

HTTPS：

是以安全为目标的HTTP通道，简单讲是HTTP的安全版，即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。

区别：

1)https协议需要到ca申请证书，一般免费证书较少，因而需要一定费用。

2)http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

3)http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

4)http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

1. Spring中的自动装配有哪些限制?

①如果使用了构造器注入或者setter注入，那么将覆盖自动装箱的依赖关系。

②基本数据类型的值、字符串字面量、类字面量无法使用自动装箱来注入。

③有先考虑使用显示的装配来进行更精确的依赖注入而不是使用自动装配

1. “==”和equals方法区别

①==比较的是值，equals比较的是地址值

②一般我们会重写equals让它按照我们的意愿去比较相关属性，而不是单纯的比较地址值。

1. &和&&区别

①&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为true时，整个运算结果才为true，否则，只要有一方为false，则结果为false。

②&&还具有短路的功能，即如果第一个表达式为false，则不再计算第二个表达式，例如，对于if(str != null && !str.equals(“ ”))表达式，当str为null时，后面的表达式不会执行，所以不会出现NullPointerException如果将&&改为&，则会抛出NullPointerException异常。If(x==33 & ++y>0) y会增长，If(x==33 && ++y>0)不会增长

③&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作，我们通常使用0x0f来与一个整数进行&运算，来获取该整数的最低4个bit位，例如，0x31 & 0x0f的结果为0x01。

1. .super.getClass()方法调用

下面程序的输出结果是多少？

import java.util.Date;

public class Test extends Date{

public static void main(String[] args) { new Test().test(); }

public void test(){ System.out.println( super.getClass().getName()); }

}

答：结果是Test。

原因：在test方法中，直接调用getClass().getName()方法，返回的是Test类名

由于getClass()在Object类中定义成了final，子类不能覆盖该方法，所以，在test方法中调用getClass().getName()方法，其实就是在调用从父类继承的getClass()方法，等效于调用super.getClass().getName()方法，所以，super.getClass().getName()方法返回的也应该是Test。

如果想得到父类的名称，调用：getClass().getSuperClass().getName();