#### 1:String,StringBuilder,StringBuffer三者的区别:

这三个类的区别主要在两个方面，即运行速度和线程安全方面。

:首先说运行速度，或者说执行速度，在执行速度快慢方面的顺序为:

StringBuilder > StringBuffer > String

String 为字符串常量，而StringBuffer和StringBuilder均为字符串变量，即String 对象创建后，该对象是不可更改的，但是后两者对象是变量。是可以修改的。

代码示例:

|  |
| --- |
| String str = "abc"; System.out.println("控制台输出:" + str); str = str + "def"; System.out.println("控制台输出:" + str); |

如果运行这段代码，会先输出“abc”,然后又输出“adcef”,好像是str这个对象被更改了，其实，只是一种假摆设，jvm对应这几行代码是这样处理的，首先创建一个String对象 str ,并把“abc”赋值给str 。然后在第三行中，其实jvm又创建了一个新的对象也名为str, 然后再把原来的str的值和“def”加起来赋值给新的str,原来的str会被jvm的垃圾回收机制（GC）给回收掉。所以，str实际上并没有被更改，也就是“String对象一旦创建，就不可被更改”。

固: 只创建String对象速度还是很快的，但java中对String对象的操作，实际是一个不断创建新的对象并且将旧的对象回收的一个过程，所以执行速度很慢。

而StringBuilder和StirngBuffer的对象是变量，对变量进行的操作就是直接对该对象进行更改，而不需要进行创建和回收操作，所以速度比String快很多。

:线程安全，StringBuilder是线程不安全的，StringBuffer是线程安全的

:总结:

String :适用于少量字符串操作的情况

StringBuilder : 适用于单线程下在字符串缓冲区进行大量操作的情况

StringBuffer : 使用于多线程下在字符串缓冲区进行大量操作的情况

#### 2:HashMap、HashTable、HashSet之间的区别

1. :实现的接口不同:

HashMap和HashTable是Map接口的实现，而HashSet是Set接口的实现，Set接口又是继承Collection接口。

1. :线程安全性、同步

HashTable中的方法加了同步锁（synchronized）,所以对象是线程安全的，而HasMap是异步的，所以存放的对象并不是线程安全的，HashSet底层是用HasMap实现的，所以也不是线程安全的。

1. :执行效率:

HashTable是同步的，而hashMap是异步的，所以HashMap的效率比HashTable要高。三者效率比较是:HashMap 　> 　HashSet 　>　HashTable 。

1. :key ,value 能都存放空值:

HashMap的key,value可以为null,

HashTable是不能存放null ,

HashSet存放的不是键值对，而是对象，也可以为null。

1. :添加元素的方法不同:

HashMap是通过put(), 来添加元素。

HashSet是通过add(),来添加元素。

#### 3:如何理解数据库事务:

1. 定义:

数据库事务是指:作为单个逻辑工作单元执行的-系列的数据库操作。

1. 事务的特性ACID

Atomicity-原子性:

把整个事务执行看做一个不可再分的原子，即整个事务的操作要么全部执行，要么全部回滚-ROLLBACK

.Consistency-一致性:事务开始前和结束后，数据库的物理约束和逻辑约束保持一致（物理约束-外键、非空、唯一等；逻辑约束-根据业务逻辑定、由开发人员保证）

.Isolation-隔离性:

在并发环境下，各个事务执行过程应该是相互隔离的。比如:两个事务同时操作一条数据，事务1看到的数据要么是事务2修改之前，要么是事务2修改之后的，绝不会看到事务2修改中间状态的数据。

Durability-持久性:

一旦事务提交，事务的所有操作就都在数据库里面生效了，数据持久存在（事务执行中，临时数据会保存到事务日志文件中，以便提交或回滚）

#### 4:ajax跨越问题（五种方案）:

1. 响应头添加Header允许访问
2. Jsonp 只支持get不支持post
3. HttpClient内部转发
4. 使用nginx
5. springCloud zuul

#### 5: io阻塞和nio非阻塞

阻塞:做某件事情，直到完成，除非超时，如果没有完成，继续等待。

非阻塞:做一件事情，尝试着做，如果做不完，就不做了直接返回，能做就做。

#### 6:反射（reflect）

定义什么是反射:

反射简单来说，就是动态加载对象，并对对象进行剖析。在Java中的反射机制是指在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法;对于任意一个对象，都能够调用它的任意一个方法，这种动态获取信息以及动态调用对象方法的功能称为Java语言的反射机制

简述反射在Java中的反射使用:

可以通过配置文件来动态配置和加载类，以实现软件工程理论里所提及的类与类，模块与模块之间的解耦。反射最经典的应用是spring框架。

动态性质：

运行时生成对象实例;

运行期间调用方法；

运行时更改属性；

Java反射机制能实现的功能：

在运行时判断任意一个对象所属的类

在运行时构造任意一个类的对象

在运行时判断任意一个类所具有的方法和属性

在运行时调用任意一个对象的方法

生成动态代理

Java反射应用场合：

在Java程序中许多对象在运行时都会出现两种类型：编译时类型和运行时类型

编译时的类型由声明该对象时使用的类型决定，运行时的类型由实际赋给对象的类型决定

如：Person p =new Student();

编译时类型为Person，而运行时为Student

除此之外，程序在运行时还可能接收到外部传入的一个对象，该对象的编译时类型为Object，但程序又需要调用该对象运行时类型的方法。为了这些问题程序需要在运行时发现对象和类的真实信息。然而，如果编译时根本无法预知该对象和类可能属于哪些类，程序只依靠运行时信息来发现该对象和类的真实信息，此时就必须使用反射

JAVA反射API

反射API用来生成在当前JAVA虚拟机中的类、接口或者对象的信息。

Class类:反射的核心类，可以获取类的属性，方法等内容信息。

Field类:Java.lang.reflect.表示类的属性，可以获取和设置类的中属性值。

Method类:Java.lang.reflect。表示类的方法，它可以用来获取类中方法的信息或者执行

反射的实现：

1.通过类的全路径名来实现：Class cla = Class.forName(classpsth);

2.通过对象的getClass()方式类实现：

public ReflectDemo2(Person p){

Class cla = p.getClass();

3.直接使用.class属性：Class cla=Person.class;

#### 7:JDK动态代理

1. 什么是jdk动态代理模式：

动态代理中的代理类是由工具类或者工厂类动态生成的，代理关系是在程序运行过程中确立的。

只学一个方法：

方法的作用：在运行时，动创建现一组指定接口的实现类对象（运行时，创建实现了指定的接口的对象）

动态代理作用

动态代理最终是面向切面AOP与装饰者模式相似，但更牛