包含但不仅限于以下题目，希望举一反三，不会哪里就连着那一片的知识点去整理。

# Java 基础

## 1.JDK 和 JRE 有什么区别？

JRE可以支撑Java程序的运行，包括JVM虚拟机和基本的类库。

JDK可以支持Java程序的开发，包括编译器，开发工具和更多的类库。

## 2.== 和 equals 的区别是什么？

== 如果作用于基本数据类型的变量，则直接比较其存储的 “值”是否相等。

equals方法不能作用于基本数据类型的变量，两边的操作数必须是同一类型的才能编译通过。

简单来说就是 == 比较数值，equals方法比较的是值和类型。

## 3.两个对象的 hashCode()相同，则 equals()也一定为 true，对吗？

equals相等，则hashcode一定相等，反之则不然。为何会有这样的要求？在集合中，比如HashSet中，要求放入的对象不能重复，怎么判定呢？首先会调用hashcode，如果hashcode相等，则继续调用equals，也相等，则认为重复。

## 4.final 在 Java 中有什么作用？

final修饰类，这样的类不能被继承。

final修饰方法，这样的方法不能被重写。

final修饰变量，这样的变量的值不能被修改，是常量。

## 5.Java 中的 Math.round(-1.5) 等于多少？

-1 这个函数就是四舍五入 小数不太一样

Math.round(-1.0)  -1  
Math.round(-1.4)  -1  
Math.round(-1.5) -1   
Math.round(-1.6) -2

## 6.String 属于基础的数据类型吗？

不是。String是一个类，是java等编程语言的字符串。

8种基本数据类型是：

boolean char byte short int long float double

1位 16位 8位 16位 32位 64位 32位 64位

8 位的范围是：-2^7 ~ 2^7-1

16位的范围是：-2^15 ~ 2^15-1

……

## 7.Java 中操作字符串都有哪些类？它们之间有什么区别？

String、StringBuffer、StringBuilder

区别：String是不可变的对象，对每次对String类型的改变时都会生成一个新的对象，StringBuffer和StringBuilder是可以改变对象的。

　对于操作效率：StringBuilder > StringBuffer > String

　对于线程安全：StringBuffer 是线程安全，可用于多线程；StringBuilder 是非线程安全，用于单线程

　不频繁的字符串操作使用 String。反之，StringBuffer 和 StringBuilder 都优于String

## 8.String str="i"与 String str=new String(“i”)一样吗？

不一样，因为他们不是同一个对象。

## 9.如何将字符串反转？

String s1 = "asdfghjkl";

System.out.println(new StringBuilder(s1).reverse().toString());

## 10.String 类的常用方法都有那些？

自己瞎掰吧，反正那么多。

## 11.抽象类必须要有抽象方法吗？

抽象类中不一定要包含抽象(abstrace)方法。也就是，抽象类中可以没有抽象方法。反之，类中含有抽象方法，那么类必须声明为抽象类。

## 12.普通类和抽象类有哪些区别？

1、抽象类不能被实例

2、抽象类不能有构造函数，抽象方法也不能被声明为静态

3、抽象类可以有抽象方法

4、抽象类的抽象方法必须被非抽象子类继承

## 13.抽象类能使用 final 修饰吗？

不能，抽象类中的抽象方法是未来继承之后重写方法，而用final修饰的类，无法被继承。

## 14.接口和抽象类有什么区别？

1、抽象类是被子类继承，接口是被类实现

2、接口只能做方法申明，抽象类中可以做方法申明，也可以做方法实现

3、接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量

4、接口是设计的结果 ，抽象类是重构的结果

## 15.Java 中 IO 流分为几种？

可以分4种。

字节输入流（InputStream）

字节输出流（OutputStream）

字符输入流（Reader）

字符输出流（Writer）

## 16.BIO、NIO、AIO 有什么区别？

BIO：线程发起IO请求，不管内核是否准备好IO操作，从发起请求起，线程一直阻塞，直到操作完成。

NIO(reactor模型)：线程发起IO请求，立即返回；内核在做好IO操作的准备之后，通过调用注册的回调函数通知线程做IO操作，线程开始阻塞，直到操作完成。

AIO(proactor模型)：线程发起IO请求，立即返回；内存做好IO操作的准备之后，做IO操作，直到操作完成或者失败，通过调用注册的回调函数通知线程做IO操作完成或者失败。

## 17.Files的常用方法都有哪些？

创建：

createNewFile()在指定位置创建一个空文件，成功就返回true，如果已存在就不创建，然后返回false。  
mkdir()  在指定位置创建一个单级文件夹。

删除：

delete()  删除文件或者一个空文件夹，不能删除非空文件夹，马上删除文件，返回一个布尔值。

判断：

exists()  文件或文件夹是否存在。  
isFile()  是否是一个文件，如果不存在，则始终为false。  
isDirectory()  是否是一个目录，如果不存在，则始终为false。

获取：

getName()  获取文件或文件夹的名称，不包含上级路径。  
getAbsolutePath()获取文件的绝对路径，与文件是否存在没关系

文件夹相关：

list()  返回目录下的文件或者目录名，包含隐藏文件。对于文件这样操作会返回null。

# 容器

## 18.Java 容器都有哪些？

答：List、Set、Map

## 19.Collection 和 Collections 有什么区别？

Collection是集合类的顶级接口，其派生了两个子接口 Set 和 List。

Collections则是集合类的一个工具类/帮助类，其中提供了一系列静态方法，用于对集合中元素进行排序、搜索以及线程安全等各种操作。

总的来说：Collection是一个接口，而Collections是个类。

## 20.List、Set、Map 之间的区别是什么？

List：可以允许重复对象，可以插入多个null元素，是一个有序容器

Set：不允许重复对象，只允许一个null元素，无序容器

Map:Map不是Collection的子接口或实现类。Map是一个接口，Map 的每个Entry都特有两个对象，也就是一个键一个值，Map可能会持有相同的值对象但键对象必须是唯一的，Map里可以拥有随意个niull值但最多只能有一个null键

## 21.HashMap 和 Hashtable 有什么区别？

HashMap线程安全，HashTable线程不安全。

## 22.如何决定使用 HashMap 还是 TreeMap？

HashMap和TreeMap都是Map的实现，HashMap是一个基于HashTable的实现存储的是键值对，无序。TreeMap是在红黑树的基础上实现的，它最大的好处是得到的数据是排好序的。选择谁取决于需不需要排序。

## 23.说一下 HashMap 的实现原理？

HashMap的主干是一个Entry数组。Entry是HashMap的基本组成单元，每一个Entry包含一个key-value键值对。HashMap由数组+链表组成的，数组是HashMap的主体，链表则是主要为了解决哈希冲突而存在的，如果定位到的数组位置不含链表，那么对于查找，添加等操作很快，仅需一次寻址即可；如果定位到的数组包含链表，对于添加操作，其时间复杂度为O(n)，首先遍历链表，存在即覆盖，否则新增；对于查找操作来讲，仍需遍历链表，然后通过key对象的equals方法逐一比对查找。所以，性能考虑，HashMap中的链表出现越少，性能才会越好。

## 24.说一下 HashSet 的实现原理？

HashSet中不允许有重复元素，这是因为HashSet是基于HashMap实现的，HashSet中的元素都存放在HashMap的key上面，而value中的值都是统一的一个final的 Object对象。HashSet跟HashMap一样，都是一个存放链表的数组。

## 25.ArrayList 和 LinkedList 的区别是什么？

ArrayList和LinkedList都是非线程安全的，由于LinkedList是基于链表的数据结构在做链表之间插入数据 或删除数据会比较方便，当读多的时候采取ArrayList,插入和删除多的话选取LinkedList。随机访问，循环的时候ArrayList性能更好，做中部插入，删除的时候LinkedList性能更好。

## 26.如何实现数组和 List 之间的转换？

List到数组 ：String[] array = (String[])list.**toArray**(new String[size]);

数组到List ：List<String> list=Arrays**.asList**(array);

## 27.ArrayList 和 Vector 的区别是什么？

Vector的方法都是同步的,是线程安全的，而ArrayList的方法不是，由于线程的同步必然要影响性能，因此,ArrayList的性能比Vector好。

当Vector或ArrayList中的元素超过它的初始大小时,Vector会将它的容量翻倍,而ArrayList只增加50%的大小，这样,ArrayList就有利于节约内存空间。

## 28.Array 和 ArrayList 有何区别？

很明显，Array是数组，而ArrayList是集合，Array可以包含基本类型和对象类型，ArrayList只能包含对象类型。

Array大小是固定的，ArrayList的大小是动态变化的。这是最基本的区别。

## 29.在 Queue 中 poll()和 remove()有什么区别？

1、add()和offer()区别:

add()和offer()都是向队列中添加一个元素。一些队列有大小限制，因此如果想在一个满的队列中加入一个新项，调用 add() 方法就会抛出一个 unchecked 异常，而调用 offer() 方法会返回 false。因此就可以在程序中进行有效的判断！

 2、poll()和remove()区别：

remove() 和 poll() 方法都是从队列中删除第一个元素。如果队列元素为空，调用remove() 的行为与 Collection 接口的版本相似会抛出异常，但是新的 poll() 方法在用空集合调用时只是返回 null。因此新的方法更适合容易出现异常条件的情况。

3、element() 和 peek() 区别：

element() 和 peek() 用于在队列的头部查询元素。与 remove() 方法类似，在队列为空时， element() 抛出一个异常，而 peek() 返回 null。

下面是Java中Queue的一些常用方法：

add         增加一个元索                      如果队列已满，则抛出一个IIIegaISlabEepeplian异常

remove    移除并返回队列头部的元素     如果队列为空，则抛出一个NoSuchElementException异常

element   返回队列头部的元素              如果队列为空，则抛出一个NoSuchElementException异常

offer        添加一个元素并返回true         如果队列已满，则返回false

poll          移除并返问队列头部的元素      如果队列为空，则返回null

peek        返回队列头部的元素              如果队列为空，则返回null

put          添加一个元素                       如果队列满，则阻塞

take        移除并返回队列头部的元素

## 30.哪些集合类是线程安全的？

线程安全(Thread-safe)就是当多线程访问时，采用了加锁的机制；即当一个线程访问该类的某个数据时，会对这个数据进行保护，其他线程不能对其访问，直到该线程读取完之后，其他线程才可以使用。防止出现数据不一致或者数据被污染的情况。Java中这样的集合对象有：Vector，HashTable，StringBuffer。

线程不安全就是不提供数据访问时的数据保护，多个线程能够同时操作某个数据，从而出现数据不一致或者数据污染的情况。Java中线程不安全的集合有：ArrayList 、LinkedList，HashMap、HashSet、TreeMap、TreeSet、StringBulider。

对于线程不安全的问题，一般会使用**synchronized**关键字加锁同步控制。

## 31.迭代器 Iterator 是什么？

对于Java数据容器，都会有很多操作上的共性，增删改查，Java采用了迭代器来为各种容器提供了公共的操作接口。这样使得对容器的遍历操作与其具体的底层实现相隔离，达到解耦的效果。在Iterator接口中定义了三个方法：hasnext(),next(),remove()。

为了实现对其数据的遍历，我们经常使用到了Iterator(迭代器)。使用迭代器，你不需要干涉其遍历的过程，只需要每次取出一个你想要的数据进行处理就可以了。

## 32.Iterator 怎么使用？有什么特点？

Iterator遍历集合元素的过程中不允许线程对集合元素进行修改，否则会抛出异常。

Iterator遍历集合元素的过程中可以通过remove方法来移除集合中的元素。

Iterator必须依附某个Collection对象而存在，Iterator本身不具有装载数据对象的功能。

Iterator.remove方法删除的是上一次Iterator.next()方法返回的对象。

next（）方法，该方法通过游标指向的形式返回Iterator下一个元素。

## 33.Iterator 和 ListIterator 有什么区别？

1. ListIterator有add()方法，可以向List中添加对象，而Iterator不能

2. ListIterator和Iterator都有hasNext()和next()方法，可以实现顺序向后遍历，但是ListIterator有hasPrevious()和previous()方法，可以实现逆向（顺序向前）遍历。Iterator就不可以。

3. ListIterator可以定位当前的索引位置，nextIndex()和previousIndex()可以实现。Iterator没有此功能。

4. 都可实现删除对象，但是ListIterator可以实现对象的修改，set()方法可以实现。Iierator仅能遍历，不能修改。

## 34.怎么确保一个集合不能被修改？

可以使用Collections.synchronizedList获取一个线程安全的集合，但是这无法保证集合不在循环的时候被其它线程所更改。它的目的就是使在多线程中去保证集合的安全。

# 多线程

35.并行和并发有什么区别？

36.线程和进程的区别？

37.守护线程是什么？

38.创建线程有哪几种方式？

39.说一下 runnable 和 callable 有什么区别？

40.线程有哪些状态？

41.sleep() 和 wait() 有什么区别？

42.notify()和 notifyAll()有什么区别？

43.线程的 run()和 start()有什么区别？

44.创建线程池有哪几种方式？

45.线程池都有哪些状态？

46.线程池中 submit()和 execute()方法有什么区别？

47.在 Java 程序中怎么保证多线程的运行安全？

48.多线程锁的升级原理是什么？

49.什么是死锁？

50.怎么防止死锁？

51.ThreadLocal 是什么？有哪些使用场景？

52.说一下 Synchronized 底层实现原理？

53.Synchronized 和 Volatile 的区别是什么？

54.Synchronized 和 Lock 有什么区别？

55.Synchronized 和 ReentrantLock 区别是什么？

56.说一下 Atomic 的原理？

# 反射

## 57.什么是反射？

个人观点：Java中的反射只是Java语言基于RTTI这个概念的一个落地实现。RTTI，（Run Time Type Identification）运行期类型识别。反射也就是运行期间类的信息。如果不知道一个类的确切信息，RTTI会帮助调查，JVM虚拟机首先就会检查那个类型的class对象是否已经载入，如果没有，JVM就会查找同名的.class文件，并且将其载入，一旦那个类型的class进入内存，就会用它去创建那一个类型的所有对象。对于这个类，都能够知道这个类的所有属性和方法；对于这个对象，都能够调用它的任意方法和属性；并且能改变它的属性。

## 58.什么是 Java 序列化？什么情况下需要序列化？

在《Thinking in Java》本书里面，对象序列化的解释是：那些实现了Serializable接口的对象，可以将他们转换成一系列字节数据，并且在以后可以完全恢复原来的样子，这个过程可以通过网络进行，利用它可以实现“有限的持久化”。“持久化”意味着对象的生存时间并不取决于程序是否在执行，而是存在于每一次程序的调用之间，通过序列化一个对象，以后再程序重新调用的时候就可以恢复那个对象，实现一种持久的效果。当Java语言里面增加了对象序列化的概念之后，就可以提供远程方法调用RMI (Remote Method Invocation)，使得本来存在于其他机器上的对象可以表现出好像就在本机上的行为。

将消息发送给远程对象的时候需要通过对象序列化来传输参数和返回值，另外，序列化也是JavaBean锁必须的。当然如果不希望某个字段在序列化的过程中被序列化，可以将这个字段声明为**transient**(临时)，就不会序列化这个字段。

## 59.动态代理是什么？有哪些应用？

动态代理(dynamic proxy) ：利用Java的反射技术在运行时**创建一个实现某些给定接口的新类**及其实例。

java提供了一个Proxy类和一个InvocationHandler接口，通过这个类和这个接口可以生成JDK动态代理类和动态代理对象。

1、创建一个InvocationHandler对象

2、使用Proxy类的getProxyClass静态方法生成一个动态代理类stuProxyClass

3、获得stuProxyClass 中一个带InvocationHandler参数的构造器constructor

4、通过构造器constructor来创建一个动态实例stuProxy

当然，上面四个步骤可以通过Proxy类的newProxyInstances方法来简化：

//创建一个与代理对象相关联的InvocationHandler

InvocationHandler stuHandler = new MyInvocationHandler<Person>(stu);

//创建一个代理对象stuProxy，代理对象的每个执行方法都会替换执行Invocation中的invoke方法

Person stuProxy= (Person) Proxy.newProxyInstance(Person.class.getClassLoader(), new Class<?>[]{Person.class}, stuHandler);

动态代理的应用：

1、解决特定问题：比如需求是这样，一个接口的实现在编译时无法知道，需要在运行时才能实现

2、实现某些设计模式：适配器(Adapter)或修饰器(Decorator)

3、面向切面编程：如AOP in Spring

不难看出，JDK提供的动态代理使用起来难度确实很大，所以AOP便是在JDK动态代理的基础上添加了一层抽象，使用几个简单的注解就可以完成相同的功能。

## 60.怎么实现动态代理？

JDK动态代理：

1、People接口：public void sayHello();

2、Chinese 类 ：implements People

实现了sayHello方法：

public void sayHello() {

System.out.println("Chinese say hello.");

}

3、PeopleInvocationHandler 类实现了：implements InvocationHandler

private Object peolple;

Intermediary(Object people){

this.people = people;

}

public Object invoke(Object proxy, Method method, Object[] args)

throws Throwable {

Object invoke = method.invoke(people, args);

System.out.println("-------- end ---------");

return invoke;

}

4、使用的时候：

People chinese = new People();

//实例化代理类

PeopleInvocationHandler invocationHandler = new PeopleInvocationHandler(chinese);

//生成代理

People proxy = (People) Proxy.newProxyInstance(chinese.getClass().getClassLoader(), chinese.getClass().getInterfaces(), invocationHandler);

//调用代理方法

proxy.sayHello();

第一步：从接口中实现需要代理的方法

第二步：编写接口的代理类，实现InvocationHandler接口，在invoke方法中编写逻辑代码。

第三步：通过Proxy的newProxyInstance方法来创建我们的代理对象，通过反射传入的参数有： 代理对象，真实对象， 和InvocationHandler 这个对象

第四步；通过代理对象去调用方法。

这里难度确实有点大，如果没有实际动手编码过的话，问到的时候回答InvocationHandler和Proxy应该问题不大。

# 对象拷贝

## 61.为什么要使用克隆？

clone()方法是java中顶层父类Object中的一个方法，clone是一个native方法，运行起来就是快啊，在底层实现的。克隆的对象可能包含一些已经修改过的属性,保留着你想克隆对象的值，而new出来的对象的属性全是一个新的对象，对应的属性没有值。即当需要一个新的对象来保存当前对象的“状态”就靠clone方法了。我把这个对象的临时属性一个一个的赋值给我新new的对象也是可以的。但是比较麻烦，使用clone就会比较快一点。

## 62.如何实现对象克隆？

1、对象的类实现Cloneable接口；

2、覆盖Object类的clone()方法 （覆盖clone()方法，访问修饰符设为public，默认是protected）；

3、在clone()方法中调用super.clone()；

## 63.深拷贝和浅拷贝区别是什么？

浅克隆是指拷贝对象时仅仅拷贝对象本身（包括对象中的基本变量），而不拷贝对象包含的引用指向的对象。

深克隆不仅拷贝对象本身，而且拷贝对象包含的引用指向的所有对象。

# Java Web

## 64.JSP 和 Servlet 有什么区别？

Jsp 本质上是servlet，用jsp实现的页面用servlet也能实现。jsp经编译后就变成了Servlet。

Jsp更擅长表现于页面显示,servlet更擅长于逻辑控制。

Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。  
而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。  
JSP侧重于视图，Servlet主要用于控制逻辑  
Servlet更多的是类似于一个Controller，用来做控制。

## 65.JSP 有哪些内置对象？作用分别是什么？

1、request对象

request 对象是 javax.servlet.httpServletRequest类型的对象。 该对象代表了客户端的请求信息，主要用于接受通过HTTP协议传送到服务器的数据。（包括头信息、系统信息、请求方式以及请求参数等）。request对象的作用域为一次请求。

2、response对象

response 代表的是对客户端的响应，主要是将JSP容器处理过的对象传回到客户端。response对象也具有作用域，它只在JSP页面内有效。

3、session对象

session 对象是由服务器自动创建的与用户请求相关的对象。服务器为每个用户都生成一个session对象，用于保存该用户的信息，跟踪用户的操作状态。session对象内部使用Map类来保存数据（K/V）。 session对象的value可以使复杂的对象类型，而不仅仅局限于字符串类型。

4、application对象

 application 对象可将信息保存在服务器中，直到服务器关闭，否则application对象中保存的信息会在整个应用中都有效。与session对象相比，application对象生命周期更长，类似于系统的“全局变量”。

5、out 对象

out 对象用于在Web浏览器内输出信息，并且管理应用服务器上的输出缓冲区。

6、pageContext 对象

获取JSP页面的out、request、response、session、application等对象，在JSP页面中可以直接使用 pageContext对象。

7、config 对象

config 对象的主要作用是取得服务器的配置信息。

8、page 对象

page 对象代表JSP本身，类似于Java编程中的 this 指针。

9、exception 对象

exception 对象的作用是显示异常信息。

## 66.说一下 JSP 的 4 种作用域？

1、page作用域

代表变量只能在当前页面上生效

2、request作用域

request表示一次客户端的请求。一次请求的生命周期从客户端发起到服务器接收并响应该请求，或者将该请求forward到另一个页面或者Servlet进行处理而结束

3、session作用域

当我们向服务器发送第一个请求开始，只要页面不关闭，或者会话未过期接下来的操作都属于同一次会话的范畴

4、application作用域

最广的，它代表着整个Web应用的全局变量，对每一个页面，每一个Servlet都是有效的

## 67.Session 和 Cookie 有什么区别？

1、Cookie以文本文件格式存储在浏览器中，而session存储在服务端

2、cookie的存储限制了数据量，只允许4KB，而session是无限量的

3、cookie值比session的值更加容易访问到

4、可以设置cookie生存时间，但session只能销毁

## 68.说一下 Session 的工作原理？

**session技术就是一种基于后端有别于数据库的临时存储数据的技术，**因为HTTP的无状态性，所以我们没有办法在HTTP发送请求的时候知道当前用户的状态，也就是比如说，当前是哪个用户的之类的这种信息，所以这个时候我们需要session来标识当前的状态。

整个流程大概分成这样的几步：

1. 第一步将本地的cookie中的session标识和用户名，密码带到后台中

2. 第二步后台检测有没有对应的session标识

3. 没有的话直接生成一个新的session。有的话，检测对应的文件是否存在并且有效

4. 失效的话，我们需要清除session然后生成新的session。不失效，使用当前的session

## 69.如果客户端禁止 Cookie 能实现 Session 还能用吗？

不能。因为Session是用Session ID来确定当前对话所对应的服务器Session，而Session ID是通过Cookie来传递的，禁用Cookie相当于失去了Session ID，也就得不到Session了。

## 70.Spring MVC 和 Struts 的区别是什么？

两者都是基于MVC的设计理念的经典落地实现。不同的是，SpringMVC将一个请求映射让一个方法去解决，而Struts是让一个Java类去处理。

Struts2是类级别的拦截，一个类对应一个request上下文，SpringMVC是方法级别的拦截。

开发性能上，SpringMVC优于Struts2，并且配置文件也比Struts2少得多。

在Spring3.0之后，SpringMVC已经全面超越了Struts，成为现在最优秀的MVC框架。

## 71.如何避免 SQL 注入？

SQL注入攻击指的是通过构建特殊的输入作为参数传入Web应用程序，而这些输入大都是SQL语法里的一些组合，通过执行SQL语句进而执行攻击者所要的操作，其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。

防范：

1.永远不要信任用户的输入。对用户的输入进行校验，可以通过[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)，或限制长度；对单引号和

双"-"进行转换等。

2.永远不要使用动态拼装sql，可以使用参数化的sql或者直接使用[存储过程](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B)进行数据查询存取。

3.永远不要使用[管理员](https://baike.baidu.com/item/%E7%AE%A1%E7%90%86%E5%91%98)权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

4.不要把机密信息直接存放，加密或者hash掉密码和敏感的信息。

5.应用的异常信息应该给出尽可能少的提示，最好使用自定义的[错误信息](https://baike.baidu.com/item/%E9%94%99%E8%AF%AF%E4%BF%A1%E6%81%AF)对原始错误信息进行包装

6.sql注入的检测方法一般采取辅助[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6)或网站平台来检测。

## 72.什么是 XSS 攻击，如何避免？

XSS又叫CSS  (Cross Site Script) ，跨站脚本攻击。它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意用户的特殊目的。

它与SQL注入攻击类似，SQL注入攻击中以SQL语句作为用户输入，从而达到查询/修改/删除数据的目的，而在xss攻击中，通过插入恶意脚本，实现对用户游览器的控制，获取用户的一些信息。

避免办法：

表单提交或者url参数传递前，对需要的参数进行过滤。

过滤用户输入的 检查用户输入的内容中是否有非法内容。

<>（尖括号）、”（引号）、 ‘（单引号）、%（百分比符号）、;（分号）、()（括号）、&（& 符号）、+（加号）

## 73.什么是 CSRF 攻击，如何避免？

CSRF(Cross site request forgery)，即跨站请求伪造。

一般而且存在XSS漏洞的网站，也极有可能存在CSRF漏洞。因为CSRF攻击中的那个“伪造的请求”的URL地址，一般是通过XSS攻击来注入到服务器中的。所以其实CSRF是以XSS为基础的，也可以看做是XSS攻击的一种。CSRF一般的攻击过程是，攻击者向目标网站注入一个恶意的CSRF攻击URL地址(跨站url)，当(登录)用户访问某特定网页时，如果用户点击了该URL，那么攻击就触发了，我们可以在该恶意的url对应的网页中，利用 <img src="" /> 来向目标网站发生一个get请求，该请求会携带cookie信息，所以也就借用了用户的身份，也就是伪造了一个请求，该请求可以是目标网站中的用户有权限访问的任意请求。也可以使用javascript构造一个提交表单的post请求。比如构造一个转账的post请求。所以CSRF的攻击分为了两步，首先要注入恶意URL地址，然后在该地址中写入攻击代码，利用<img> 等标签或者使用Javascript脚本。

CSRF防御

1、referer

因为伪造的请求一般是从第三方网站发起的，所以第一个防御方法就是判断 referer 头，如果不是来自本网站的请求，就判定为CSRF攻击。但是该方法只能防御跨站的csrf攻击，不能防御同站的csrf攻击(虽然同站的csrf更难)。

2、使用验证码

每一个重要的post提交页面，使用一个验证码，因为第三方网站是无法获得验证码的。还有使用手机验证码，比如转账是使用的手机验证码。

3、使用token

每一个网页包含一个web server产生的token, 提交时，也将该token提交到服务器，服务器进行判断，如果token不对，就判定位CSRF攻击。将敏感操作又get改为post,然后在表单中使用token. 尽量使用post也有利于防御CSRF攻击。

# 异常

## 74.throw 和 throws 的区别？

throw是语句抛出异常，需要解决，而throws只是方法可能抛出异常的一个申明。

throws抛出异常出现在 public void function() throws Exception{}; throw抛出异常是在方法内。throws是“抛弃”，一旦出现异常就将其抛到调用该方法的地方去，让其他人解决这个异常，throw是“产生”，一旦出现异常就处理它。

## 75.final、finally、finalize 有什么区别？

1、final

final关键字可以用于类，方法，变量前，用来表示该关键字修饰的类，方法，变量具有不可变的特性。

2、finally

当代码抛出一个异常时，就会终止方法中剩余代码的处理，并退出这个方法的执行。假如我们打开了一个文件，但在处理文件过程中发生异常，这时文件还没有被关闭，此时就会产生资源回收问题。对此，java提供了一种好的解决方案，那就是finally子句，finally子句中的语句是一定会被执行的，所以我们只要把前面说的文件关闭的语句放在finally子句中无论在读写文件中是否遇到异常退出，文件关闭语句都会执行，保证了资源的合理回收。

3、finalize

finalize方法来自于java.lang.Object，用于回收资源。

可以为任何一个类添加finalize方法。finalize方法将在垃圾回收器清除对象之前调用。

在实际应用中，不要依赖使用该方法回收任何短缺的资源，这是因为很难知道这个方法什么时候被调用。

## 76.try-catch-finally 中哪个部分可以省略？

可以的情况组合有：try-catch，try-finally，try-catch-finally。

## 77.try-catch-finally 中，如果 catch 中 return 了，finally 还会执行吗？

会执行。

1、不管有没有异常，finally中的代码都会执行

2、当try、catch中有return时，finally中的代码依然会继续执行

3、finally是在return后面的表达式运算之后执行的，此时并没有返回运算之后的值，而是把值保存起来，不管finally对该值做任何的改变，返回的值都不会改变，依然返回保存起来的值。也就是说方法的返回值是在finally运算之前就确定了的。

4、如果return的数据是引用数据类型，而在finally中对该引用数据类型的属性值的改变起作用，try中的return语句返回的就是在finally中改变后的该属性的值。

5、finally代码中最好不要包含return，程序会提前退出，也就是说返回的值不是try或catch中的值。

## 78.常见的异常类有哪些？

|  |  |
| --- | --- |
| NullPointerException | 空指针异常 |
| SQLException | 数据库访问错误 |
| IndexOutOfBoundsException | 数组下标越界 |
| IOException | 某种I/O异常 |
| IllegalArgumentException | 方法传递了一个不合法或不正确的参数 |
| ClassCastException | 强制转换为不是实例的子类时，抛出该异常 |
| RuntimeException | Java虚拟机正常运行期间抛出的异常 |

# 网络

## 79.HTTP 响应码 301 和 302 代表的是什么？有什么区别？

301和302状态码都表示重定向，就是说浏览器在拿到[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)返回的这个状态码后会自动跳转到一个新的URL地址

301 redirect: 301 代表永久性转移(Permanently Moved)

302 redirect: 302 代表暂时性转移(Temporarily Moved )

## 80.Forward 和 Redirect 的区别？

Forward服务器内部转发：是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

Redirect重新定向：就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。

## 81.简述 TCP 和 UDP 的区别？

TCP：可靠，稳定。在传递数据之前，会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完后，还会断开连接用来节约系统资源。 TCP的缺点： 慢，效率低，占用系统资源高，易被攻击。

UDP：快，比TCP稍安全。没有的握手、确认、窗口、重传、拥塞控制等机制，UDP是一个无状态的传输协议，所以它在传递数据时非常快。UDP的缺点： 不可靠，不稳定，会很容易丢包。

## 82.TCP 为什么要三次握手，两次不行吗？为什么？

3次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。

如果把三次握手改成仅需要两次握手，死锁是可能发生的。相互猜疑。一个在认为连接可能还未建立，忽略分组包。一个则又一直在发送分组包。

## 83.说一下 TCP 粘包是怎么产生的？

1、什么是粘包现象

TCP粘包是指发送方发送的若干包数据到接收方接收时粘成一包，从接收缓冲区看，后一包数据的头紧接着前一包数据的尾。

2、为什么出现粘包现象

（1）发送方原因

TCP默认会使用Nagle算法。而Nagle算法主要做两件事：

1）只有上一个分组得到确认，才会发送下一个分组；2）收集多个小分组，在一个确认到来时一起发送。

所以，正是Nagle算法造成了发送方有可能造成粘包现象。

（2）接收方原因

TCP接收到分组时，并不会立刻送至应用层处理，或者说，应用层并不一定会立即处理；实际上，TCP将收到的分组保存至接收缓存里，然后应用程序主动从缓存里读收到的分组。这样一来，如果TCP接收分组的速度大于应用程序读分组的速度，多个包就会被存至缓存，应用程序读时，就会读到多个首尾相接粘到一起的包。

## 84.OSI 的七层模型都有哪些？

1、物理层协议有：EIA/TIA-232， EIA/TIA-499，V.35， V.24，RJ45， Ethernet， 802.3

2、数据链路层协议有：Frame Relay，HDLC，PPP， IEEE 802.3/802.2

3、网络层协议有：IP，IPX，AppleTalk DDP

4、传输层协议有：TCP，UDP，SPX

5、会话层协议有：RPC，SQL，NFS，NetBIOS，names，AppleTalk

6、表示层协议有：TIFF，GIF，JPEG，PICT，ASCII，EBCDIC，encryption

7、应用层协议有：FTP，WWW，Telnet，NFS，SMTP，Gateway，SNMP

## 85.Get和 Post 请求有哪些区别？

Get请求有长度限制，参数会显示在URL地址栏。

Post请求没有长度显示，参数隐藏。

所以Get请求一般用于获取数据，Post请求一般用于提交数据。

## 86.如何实现跨域？

跨域，指的是浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器施加的安全限制。

使用ajax请求访问其他服务器的数据,此时,客户端会出现跨域问题。localhost和127.0.0.1虽然都指向本机，但也属于跨域。说到底就是客户端浏览器的一种保护策略，但是我们又需要获取到其他域名下的数据。

常用的跨域方法有JSONP，CORS，postmessage等。

## 87.说一下 JSONP 实现原理？

ajax请求受同源策略影响，不允许进行跨域请求，而script标签src属性中的链接却可以访问跨域的js脚本，利用这个特性，服务端不再返回JSON格式的数据，而是返回一段调用某个函数的js代码，在src中进行了调用，这样实现了跨域。

# 设计模式

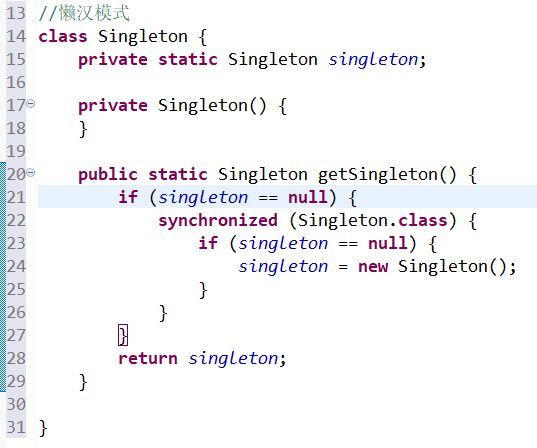
## 88.说一下你熟悉的设计模式？

设计模式三杰：单例、工厂、观察者。

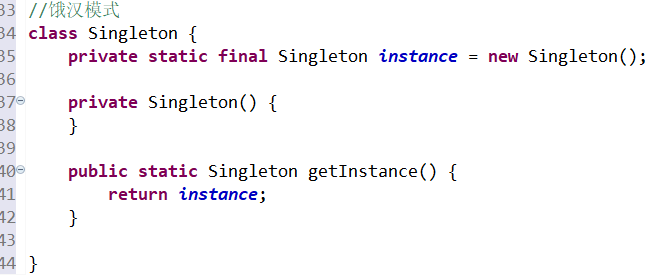
### 单例模式Singleton

它的意图是保证一个类只有一个实例，并且提供一个访问它的全局访问点。

将属性私有，构造方法私有，再提供一个全局的共有方法给使用者。这样一个最简单的懒汉式的单例模式就写完了。在暴露的方法上加上synchronized关键字，就变成了线程安全的。



直接将属性声明为static final 并且new一个空值，就能保证它是线程安全的了。



饿汉模式，加载慢，获取快，线程安全

懒汉模式，加载快，获取慢，线程不安全

### 观察者模式 Observer

它的意图是定义一种一对多的依赖关系，当一个对象的状态发生改变的时候，所有依赖于它的对象都得到通知并且自动被更新。简单来说，就是四个字“触发联动”。

举个例子，在Spring Cloud Config配置服务中，将配置文件提交到github上，新建了Config Server去获取在github上的配置文件，在这个module里面只需要修改bootstrap.yml这个系统级别的资源配置项，这样各个微服务只需要关注自身的业务逻辑，而无需再自己手动去配置各个繁琐的配置。在这例子中，Config Server就是一个观察的目标，而各个微服务项目都是观察者，当观察目标的配置文件修改了数据库用户，环境变更，域名等等这些信息变更的时候，各个微服务不用再一个个的去修改。

观察者模式也成为发布-订阅模式，它有推模型和拉模型。

### 责任链模式 Chain Of Responsibility

它的意图就是使得多个对象之间都会有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系。将这些对象连成一条链，并且沿着这条链传递该请求，直到有一个对象能够处理它为止。

最常见的例子就是Java语言中异常的掷出（抛出异常）try-cach-finally，还有对用户输入数据的层层校验，不合法的、带有攻击型的这类数据会先被程序执行判断，如果校验的足够严密，才会将正确的数据交给主要的业务逻辑代码。

## 89.简单工厂和抽象工厂有什么区别？

简单工厂、工厂方法、抽象工厂，头疼。没啥区别，那个类图看起来都挺头疼的。

<https://www.jianshu.com/p/4523dcb131c8>

# Spring/Spring MVC

## 90.为什么要使用 Spring？

拆分功能模块和非业务模块。使得各个模块耦合更松散，可以在业务逻辑之外进行增强代理，用起来更加的得心应手。在IOC和AOP的基础上可以整合各种企业应用的开源框架和优秀的第三方类库，达到一站式的开发便利程度。另外从2004年3月spring发布以来，它的发展业内有目共睹，在社区活跃程度、给开发带来的便捷程度来说，也必须使用啊。（感觉这问题问的有点措手不及，凭什么不用啊？）

## 91.解释一下什么是 AOP？

由实际问题而引入的面向切面的编程思想，代码混乱且分散。原来JDK的动态代理和Cglib的动态代理使用起来难度太大了，Spring添加了往上面抽象了一层。将编程对象转移为切面，切面模块化横切关注点，每个事务的业务逻辑位于一个位置，代码不分散，便于维护和升级，同时也使得业务模块更加的简洁，只去关注核心业务代码。

## 92.解释一下什么是 IOC？

反转资源获取的方向，传统的资源查找方式要求资源主动的向容器发起请求查找资源作为回应。而有力IOC之后，则会变成容器主动的将资源推送给它所管理的组件，组件需要做的事情仅仅是选择一种合适的方式来接受资源。所以IOC也更加确切的称之为DI依赖注入。

## 93.Spring 有哪些主要模块？

Spring有七大功能模块，分别是Spring Core，AOP，ORM，DAO，MVC，WEB，Context。

1，Spring Core

Core模块是Spring的核心类库，Spring的所有功能都依赖于该类库，Core主要实现IOC功能，Sprign的所有功能都是借助IOC实现的。

2，AOP

AOP模块是Spring的AOP库，提供了AOP（拦截器）机制，并提供常用的拦截器，供用户自定义和配置。

3，ORM

Spring 的ORM模块提供对常用的ORM框架的管理和辅助支持，Spring支持常用的Hibernate，ibtas，jdao等框架的支持，Spring本身并不对ORM进行实现，仅对常见的ORM框架进行封装，并对其进行管理

4，DAO模块

Spring 提供对JDBC的支持，对JDBC进行封装，允许JDBC使用Spring资源，并能统一管理JDBC事物，并不对JDBC进行实现。（执行sql语句）

5，WEB模块

WEB模块提供对常见框架如Struts1，WEBWORK（Struts 2），JSF的支持，Spring能够管理这些框架，将Spring的资源注入给框架，也能在这些框架的前后插入拦截器。

6，Context模块

Context模块提供框架式的Bean访问方式，其他程序可以通过Context访问Spring的Bean资源，相当于资源注入。

7，MVC模块

WEB MVC模块为Spring提供了一套轻量级的MVC实现，在Spring的开发中，我们既可以用Struts也可以用Spring自己的MVC框架，相对于Struts，Spring自己的MVC框架更加简洁和方便。

## 94.Spring 常用的注入方式有哪些？

## 95.Spring 中的 Bean 是线程安全的吗？

96.Spring 支持几种 Bean 的作用域？

97.Spring 自动装配 Bean 有哪些方式？

98.Spring 事务实现方式有哪些？

99.说一下 Spring 的事务隔离？

100.说一下 Spring MVC 运行流程？

101.Spring MVC 有哪些组件？

102.@RequestMapping 的作用是什么？

103.@Autowired 的作用是什么？

# Spring Boot/Spring Cloud

104.什么是 Spring Boot？

105.为什么要用 Spring Boot？

106.Spring Boot 核心配置文件是什么？

107.Spring Boot 配置文件有哪几种类型？它们有什么区别？

108.Spring Boot 有哪些方式可以实现热部署？

109.JPA 和 Hibernate 有什么区别？

110.什么是 Spring Cloud？

111.Spring Cloud 断路器的作用是什么？

112.Spring Cloud 的核心组件有哪些？

# Hibernate

113.为什么要使用 Hibernate？

114.什么是 ORM 框架？

115.Hibernate 中如何在控制台查看打印的 SQL 语句？

116.Hibernate 有几种查询方式？

117.Hibernate 实体类可以被定义为 final 吗？

118.在 Hibernate 中使用 Integer 和 int 做映射有什么区别？

119.Hibernate 是如何工作的？

120.get()和 load()的区别？

121.说一下 Hibernate 的缓存机制？

122.Hibernate 对象有哪些状态？

123.在 Hibernate 中 getCurrentSession 和 openSession 的区别是什么？

124.Hibernate 实体类必须要有无参构造函数吗？为什么？

# Mybatis

125.Mybatis 中 #{}和 ${}的区别是什么？

126.Mybatis 有几种分页方式？

127.RowBounds 是一次性查询全部结果吗？为什么？

128.Mybatis 逻辑分页和物理分页的区别是什么？

129.Mybatis 是否支持延迟加载？延迟加载的原理是什么？

130.说一下 Mybatis 的一级缓存和二级缓存？

131.Mybatis 和 Hibernate 的区别有哪些？

132.Mybatis 有哪些执行器（Executor）？

133.Mybatis 分页插件的实现原理是什么？

134.Mybatis 如何编写一个自定义插件？

# RabbitMQ

135.RabbitMQ 的使用场景有哪些？

136.RabbitMQ 有哪些重要的角色？

137.RabbitMQ 有哪些重要的组件？

138.RabbitMQ 中 VHost 的作用是什么？

139.RabbitMQ 的消息是怎么发送的？

140.RabbitMQ 怎么保证消息的稳定性？

141.RabbitMQ 怎么避免消息丢失？

142.要保证消息持久化成功的条件有哪些？

143.RabbitMQ 持久化有什么缺点？

144.RabbitMQ 有几种广播类型？

145.RabbitMQ 怎么实现延迟消息队列？

146.RabbitMQ 集群有什么用？

147.RabbitMQ 节点的类型有哪些？

148.RabbitMQ 集群搭建需要注意哪些问题？

149.RabbitMQ 每个节点是其他节点的完整拷贝吗？为什么？

150.RabbitMQ 集群中唯一一个磁盘节点崩溃了会发生什么情况？

151.RabbitMQ 对集群节点停止顺序有要求吗？

# Kafka

152.Kafka 可以脱离 ZooKeeper 单独使用吗？为什么？

153.Kafka 有几种数据保留的策略？

154.Kafka 同时设置了 7 天和 10G 清除数据，到第五天的时候消息达到了 10G，这个时候 Kafka 将如何处理？

155.什么情况会导致 Kafka 运行变慢？

156.使用 Kafka 集群需要注意什么？

# ZooKeeper

157.ZooKeeper 是什么？

158.ZooKeeper 都有哪些功能？

159.ZooKeeper 有几种部署模式？

160.ZooKeeper 怎么保证主从节点的状态同步？

161.集群中为什么要有主节点？

162.集群中有 3 台服务器，其中一个节点宕机，这个时候 ZooKeeper 还可以使用吗？

163.说一下 ZooKeeper 的通知机制？

# MySQL

164.数据库的三范式是什么？

165.一张自增表里面总共有 7 条数据，删除了最后 2 条数据，重启 MySQL 数据库，又插入了一条数据，此时 ID 是几？

166.如何获取当前数据库版本？

167.说一下 ACID 是什么？

168.Char 和 VarChar 的区别是什么？

169.Float 和 Double 的区别是什么？

170.MySQL 的内连接、左连接、右连接有什么区别？

171.MySQL索引是怎么实现的？

172.怎么验证 MySQL的索引是否满足需求？

173.说一下数据库的事务隔离？

174.说一下 MySQL常用的引擎？

175.说一下 MySQL的行锁和表锁？

176.说一下乐观锁和悲观锁？

177.MySQL问题排查都有哪些手段？

178.如何做 MySQL的性能优化？

# Redis

179.Redis 是什么？都有哪些使用场景？

180.Redis 有哪些功能？

181.Redis 和 MemeCache 有什么区别？

182.Redis 为什么是单线程的？

183.什么是缓存穿透？怎么解决？

184.Redis 支持的数据类型有哪些？

185.Redis 支持的 Java 客户端都有哪些？

186.Jedis 和 Redisson 有哪些区别？

187.怎么保证缓存和数据库数据的一致性？

188.Redis 持久化有几种方式？

189.Redis 怎么实现分布式锁？

190.Redis 分布式锁有什么缺陷？

191.Redis 如何做内存优化？

192.Redis 淘汰策略有哪些？

193.Redis 常见的性能问题有哪些？该如何解决？

# JVM

194.说一下 JVM 的主要组成部分？及其作用？

195.说一下 JVM 运行时数据区？

196.说一下堆栈的区别？

197.队列和栈是什么？有什么区别？

198.什么是双亲委派模型？

199.说一下类加载的执行过程？

200.怎么判断对象是否可以被回收？

201.Java 中都有哪些引用类型？

202.说一下 JVM 有哪些垃圾回收算法？

203.说一下 JVM 有哪些垃圾回收器？

204.详细介绍一下 CMS 垃圾回收器？

205.新生代垃圾回收器和老生代垃圾回收器都有哪些？有什么区别？

206.简述分代垃圾回收器是怎么工作的？

207.说一下 JVM 调优的工具？

208.常用的 JVM 调优的参数都有哪些？