## 面试注意事项

### 介绍项目

开门见山，直接将自己的项目用最精炼的语言表达出来。

java面试的时候，面试官看你的项目主要是看你有没有这种开发的经验，但是不是你写的你做的项目越多越好。

介绍项目的时候，最先肯定是介绍你所使用的技术，以及完成的功能。

然后你就可以说出你觉得你项目中的闪光点，这个是是很重要的。

比如说你在项目开发中，使用了断点续传的功能，你就可以好好的介绍你的这个功能，但是有的时候面试官也会问你具体的功能，比如说你的购物车的功能是怎么实现的。

所以说你要对你的项目很了解。还有就是你所使用的技术，你要有一定的认识，比如你使用了hibernate的技术，你就要说的出它的工作基本原理，还有就是它的好处等等。

还有最重要的是，你可以引导面试关想你问问题，不要被动的被他问。

# 自我介绍

面试官，您好，很高兴见到您。我叫曾金龙，来自江西省赣州市，大学就读于位于江西南昌的江西科技师范大学，毕业于2018年。在大四的时候面试上了深圳的一家互联网公司实习工作至今大致有一年半的时间，主要从事的也是java后台开发方向，做过一些项目有一定的项目经验。平时的话比较喜欢看看一些技术博客，逛逛知乎什么的学习下新技术，毕竟it行业的话技术更新是比较快的，自己也一直保持着不断学习的兴趣。我的自我介绍大致就这些，谢谢。

好的，这个商城项目的话采用分布式架构，采用的是ssm框架，该购物平台是一个综合性的B2C平台，类似京东商城、天猫商城。会员可以在商城浏览商品、下订单，以及参加各种活动；管理员、运营可以在平台后台管理系统中管理商品、订单、会员等；客服可以在后台管理系统中处理用户的询问以及投诉。其大致分为

后台管理系统：实现管理商品、订单、类目、商品规格属性、用户管理以及内容发布等功能。

前台系统：实现用户可以在前台系统中进行注册、登录、浏览商品、首页、下单等操作。

会员系统：用户可以在该系统中查询已下的订单、收藏的商品、我的优惠券、团购等信息。

订单系统：提供下单、查询订单、修改订单状态、定时处理订单。

搜索系统：提供商品的搜索功能。

单点登录系统：为多个系统之间提供用户登录凭证以及查询登录用户的信息。

我主要负责的是前台管理系统的实现，例如用户的注册，用户的登录，用户信息的修改。对于每个业务的实现大致就是做一个从dao层开始对数据库的增删改查，然后在业务层实现对于逻辑代码的编写，在控制器层现在对于数据的控制与转发，最后在前端界面中实现页面的展示。比如说关于用户的注册加密技术的话我们用的是dh5实现加密以及随机盐，然后我们将盐值存储到数据库中，到了用户使用账号密码登录的时候从数据中读取就可以了。可能一开始还对于ajax不是很熟练，造成业务没有实现没有具体的提示，返回的json对象没有进入function方法。后来的话多练习几次就好了。

### socket出现丢包现象如何解决

解决方案：1、简单粗暴的就是在取数据之前让线程休息一会儿，保证操作系统把完整的包都发过来了；虽然这个能解决问题，但是不太合适；2、在取数据之前判断一下流里的数据长度够内容长度再取，或者用datainputstream的readfully方法，就是读的时候保证有参数byte[]长度的数据才取数据，不然就阻塞，也就是休息会儿，只有长度够了，就开始取数据了。

---------------------

Mysql序列是什么？怎么用？

Sql 子查询（例子）

8. 查询平均工资最高的部门信息（史诗级难度！！）

-得到每个部门的平均工资

select deptno,avg(sal) from emp group by deptno;

-得到最高的平均工资

select avg(sal) a from emp group by deptno order by a desc limit 0,1;

-通过最高的平均工资得到对应的部门编号

select deptno from emp group by deptno having avg(sal)=(select avg(sal) a from emp group by deptno order by a desc limit 0,1);

-通过部门编号得到部门信息

select \* from dept where deptno in(select deptno from emp group by deptno having avg(sal)=(select avg(sal) a from emp group by deptno order by a desc limit 0,1));

- 子查询可以写在什么位置

1. 写在where或having后面 当做查询条件的值

2. 写在创建表的时候 把查询结果保存成一张新的表

create table emp\_20 as (select \* from emp where deptno=20);

3. 写在from后面 当成一个虚拟表 \*\*必须有别名\*\*

select \* from emp where deptno=20;

select ename,sal from (select \* from emp where deptno=20) newtable;

1. 分组查询 group by 字段,字段

2. having 和group by结合使用，用于添加聚合函数的条件

3. 子查询(嵌套查询) 可以嵌套n层，1. 写在where和having后面 2.from后面当虚拟表 3. 写在创建表的时候

4. 关联查询：

- 等值连接:select \* from A,B where A.x=B.x and A.age=18;

- 内连接:select \* from A join B on A.x=B.x where A.age=18;

- 外连接:select \* from A left/right join B on A.x=B.x where A.age=18;

1. 项目中用到几张表，表间怎么关联，项目哪些是自己做的，怎么实现的？
2. 单点登录原理？
3. Mysql如何使用索引？
4. 你在公司负责什么？
5. Mysql怎么查时间？
6. 怎么理解事物的，事物的一致性是什么？
7. 说一下redis,为什么要用redis?
8. 适配器的类是哪个类？
9. 页面的传值两种方式分别有什么？
10. Get和post有什么区别？
11. 线程的实现方式？
12. 线程run和start的区别？
13. 线程一共有多少种状态？
14. 如果一个线程阻塞了怎么办？
15. 休眠的话有几种方式把他唤醒？
16. Mysql数据库怎么取系统时间？
17. 说说dubbo的原理
18. 如何解决高并发问题？
19. Shiro登陆验证的流程
20. Java的容器类有哪些？
21. Mybatis的优缺点
22. Spring的接口有哪些？
23. gc处理的是内存里的什么？
24. JVM里面有什么？
25. 常用的设计模式有哪些？怎么实现的？
26. 数据库优化
27. Mybatis如何实现批量插入？
28. Tomcat的配置有几种？
29. Tomcat的溢出异常你是如何处理的？
30. Redis集群如何搭建？
31. 什么是反向代理？什么是正向代理？原理都是什么？谈谈你的理解？
32. Hashset和hashtable有什么区别？底层如何实现？
33. 可变长度的数组是通过什么方式实现的？
34. 你为什么从上一家公司辞职？
35. 你的技术是如何获取的？是自学，还是参加过培训？
36. 如何查看Linux系统中的日志文件？
37. 什么是Json？有什么用？
38. 工厂模式用在什么地方？
39. Redis的数据结构有哪些？你常用的有哪几个？
40. 死锁的形成条件和解决方法？
41. Nginx如何配置？
42. 什么是镜像？谈谈镜像
43. 购物车如何实现？
44. 商品管理系统如何实现的？表关系是怎样的？
45. 前端数据如何展示？
46. 你们之前开发这个系统的时候，前后端开发如何协调解决的？
47. 用户权限如何分离出来？
48. Shiro的权限是挂载给谁的？
49. 说一下你们缓存的使用情况？
50. 你理解的session是什么？说一下session和cookie的区别和联系？
51. IOC中如何去调用一个service的实例？
52. 如何把windows的web项目部署到linux的tomcat上运行？
53. 如果你有异常了，如何排除？
54. 数据库的层面如何抗击高并发？
55. 请求的层面如何抗击高并发？
56. 索引有哪几种类型
57. 使用过存储过程吗
58. 数据库游标怎么写
59. Spring组件有哪些
60. 你用过的中间件有哪些
61. MySQL中操作大量数据怎么保证安全，比如插入1000条数据，你是不是用Java语句直接写进去
62. Spring的工作原理（或工作流程）
63. servlet的生命周期
64. 项目问你们有没有用高并发，怎样通过负载均衡来完成高并发的数据读写
65. 对分布式有什么了解？
66. 如何保证缓存和数据库之间数据的一致性
67. 工作中用到的设计模式有哪些
68. 你了解过HTTP协议吗
69. 你用Nginx做了静态资源服务器，这些静态资源是如何保存的
70. 项目中有没有使用多线程？如何使用？
71. 项目中使用了什么类型的线程池？线程池常用的有哪些策略类型？
72. 请讲一下left join 和 right join的区别，和join有什么关系？
73. 数据库优化了解多少？

1.redis缓存

使用redis做缓存工具

实现系统高可用，redis需要做主备，使用redis做分片集群，redis集群的搭建

向业务逻辑添加缓存

1 数据结构

常用结构：

1数组：把具有相同类型的若干变量按照有序的形式组织起来，这些按序排列的同类数据元素的集合称为数组

2栈：是只能在某一端插入和删除的特殊的线性表，先进后出的存储数据，最后的数据在栈顶，最后一个数据被第一个读出来。入栈 push 出栈pop

3队列：queue

特殊的线性表，只能从线形表达一端offer添加元素，从另一端取出poll元素（先进先出）区别与栈，jdk提供了queue接口，linkedlist实现了该接口，lineklist是实现接口是因为queue经常要进行添加和删除操作，而linkedlist在这方面效率较高

一般队尾添加offer 队首删除poll peek返回队首元素并不删除

2.Deque实现了queue接口，定义了所谓的”双端队列”linkedlist实现了该接口 即是：

从队列的两端都可以出列poll和入列offer

4链表：他既可以表示线性结构，也可以表示非线性结构，数据元素的逻辑顺序是通过链表中的指针连接次序是实现的

5.树 是包含n(n>0)个结点的有穷集合， 且在K中定义了一个关系N，N满足一下条件： 1 有且有一个根结点 root 2 除了根结点之外，每个结点有且仅有一个前驱，3.K中各结点，对关系N来说可以有m个后继(m>0)

6.图

7.堆 是一种特殊的树形数据结构，每个结点都有值。通常我们所说的堆的数据结构是指二叉堆。堆的特点是根结点的值最小（或最大），且根结点的两个子树也是一个堆

8.散列表

若结构中存在关键字和K相等的记录，则必定在f(K)的存储位置上。由此，不需比较便可以直接取得所查记录。称这个对应关系f为散列函数，按照这个思想建立的表为散列表。

## ###索引

- 什么是索引： 索引是数据库中提高查询效率的技术，类似于字典的目录

- 为什么使用索引：如果不使用索引数据会零散的保存在每一个磁盘块当中，查询数据时需要挨个的遍历每一个磁盘块查找数据，如果数据量超级大，遍历每一个磁盘块是件非常耗时的事情，添加索引后，会将磁盘块以树桩结构进行保存，查询数据时会有目的性的访问部分磁盘块，因为访问的磁盘块数量降低所以能起到提高查询效率的作用

- 索引是越多越好吗？

不是，因为索引会占磁盘空间，通过某个字段创建的索引可能永远用不上，则这个索引完全没有存在的意义，只需要对查询时频繁使用的字段创建索引

- 有索引就一定好吗？

不一定，如果数据量小使用索引反而会降低查询效率

- 索引的分类(了解)

1. 聚集索引(聚簇索引)： 通过主键创建的索引为聚集索引，添加了主键约束的表会自动添加聚集索引，聚集索引的树桩结构中保存了数据

2. 非聚集索引：通过非主键字段创建的索引叫做非聚集索引，树桩结构中只保存了数据所在磁盘块的地址并没有数据。

- 导入数据：

- windows电脑 把文件放到D盘下

source d:/item\_backup.sql;

- Linux系统 把文件放到桌面

source /home/soft01/桌面/item\_backup.sql;

- 测试：

1. show tables; 看是否有item2这张表

2. select count(\*) from item2; 看是否有172万条数据

3. select \* from item2 where title='100'; //看一下耗时 1.1秒

- 创建索引的格式：

create index 索引名 on 表名(字段名[(字符长度)]);

create index i\_item\_title on item2(title);

-创建完后继续执行

select \* from item2 where title='100'; //看一下耗时 0.02秒

- 查看索引：

show index from item2;

- 删除索引

drop index 索引名 on 表名;

drop index i\_item\_title on item2;

- 复合索引

通过多个字段创建的索引称为复合索引

-格式：create index 索引名 on 表名(字段1,字段2);

频繁使用多个字段进行数据查询时为了提高查询效率可以创建复合索引

select \* from item2 where title='100' and price<100;

create index i\_item2\_title\_price on item2(title,price);

- 总结：

1. 索引是用于提高查询效率的技术，类似目录

2. 索引会占用磁盘空间不是越多越好

3. 如果数据量小的话 添加索引会降低查询效率

4. 尽量不要在频繁改动的表上添加索引

####事务

- 数据库中执行SQL语句的最小工作单元，保证事务中的多条SQL全部成功或全部失败

- 事务的ACID特性：

1. Atomicity： 原子性， 最小不可拆分 保证全部成功或全部失败

2. Consistency： 一致性，从一个一致状态到另外一个一致状态

3. Isolation：隔离性，多个事务之间互相隔离互不影响

4. Durability:持久性， 事务提交后 数据持久保存到数据库文件中

- 事务相关指令：

- 查看自动提交状态： show variables like '%autocommit%'

- 修改自动提交: set autocommit=0/1;

- 提交： commit;

- 回滚： rollback;

- 保存回滚点： savepoint s1;

- 回滚到指定回滚点： rollback to s1;

### 什么是事务，什么时候使用事物，请举例说明死锁的情况

事务（Transaction）是数据库运行中的一个逻辑工作单位，是用户定义的一个逻辑操作

数据库中执行SQL语句的最小工作单元，保证事务中的多条SQL全部成功或全部失败

对数据库的数据进行批量或连表操作时，为了保证数据的一致性和正确性，我们需要添加事务管理机制进行管理。当对数据库的数据进行操作失败时，事务管理可以很好保证所有的数据回滚到原来的数据，如果操作成功，则保证所有需要更新的数据持久化。

两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程都陷入了无限的等待中

### 输入年月日，输出这是今年的第几天

public class asgasaf {

/\*\*

\* 输入某年某月某日，判断这一天是这一年的第几天？

\*

\*/

public static void main(String[] args) {

int year;

int mouth;

int day=0;

int days;

//累计天数

int d=0;

int e = 0;

Scanner scanner = new Scanner(System.in);

do {

System.out.println("输入年:");

year = scanner.nextInt();

System.out.println("输入月:");

mouth = scanner.nextInt();

System.out.println("输入日:");

days = scanner.nextInt();

if (mouth < 0 || mouth > 12 || days < 0 || days > 31) {

System.out.println("input error!");

e = 1;

}

} while (e == 1);

for (int i = 1; i <mouth; i++) {

switch (i) {

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12: {

day = 31;

break;

}

case 4:

case 6:

case 9:

case 11: {

day = 30;

break;

}

case 2: {

/\*\*

\* 闰年:①：非整百年数除以4，无余为闰，有余为平；②整百年数除以400，无余为闰有余平

\* 二月：平年28天、闰年29天

\*/

if ((year % 100 !=0 &&year % 4 == 0) || (year % 100 == 0 && year%400==0)) {

day = 29;

} else {

day = 28;

}

}

default:

break;

}

d+=day;

}

System.out.println("这是"+year+"年的"+(d+days)+"天");

}

}

---------------------

## 实现打印杨辉三角

public static void main(String[] args) {

// 定义数组的长度

int length = 10;

int arr[][] = new int[length][];

// 遍历二维数组

for (int i = 0; i < arr.length; i++) {

// 确定二维数组元素的大小

arr[i] = new int[i+1];

// 打印空格

for (int j = 0; j < arr.length-i-1; j++) {

System.out.print(" ");

}

// 遍历一维数组

for (int j = 0; j < arr[i].length; j++) {

if(j == 0 || arr[i].length-1 == j) {

// 三角形两腰全为 1

arr[i][j] = 1;

}else {

// 其他元素等于两肩数字之和

arr[i][j] = arr[i-1][j] + arr[i-1][j-1];

}

System.out.printf("%4d",arr[i][j]);

}

System.out.println();

}

}

---------------------

冒泡排序

public static void main(String[] args) {

int[] arr = {1,2,3,6,5,4,8,9,8};

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length-i-1; j++) {

if(arr[j] > arr[j+1]) {

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

System.out.println(Arrays.toString(arr));

}

---------------------

## 输入一组数组和一个元素实现返回元素下标（二分查找）

// 二分查找普通循环实现

**public** **static** **int** binSearch(**int** srcArray[], **int** key) {

**int** mid = srcArray.length / 2;

**if** (key == srcArray[mid]) {

**return** mid;

}

**int** start = 0;

**int** end = srcArray.length - 1;

**while** (start <= end) {

mid = (end - start) / 2 + start;

**if** (key < srcArray[mid]) {

end = mid - 1;

} **else** **if** (key > srcArray[mid]) {

start = mid + 1;

} **else** {

**return** mid;

}

}

**return** -1;

}

苍老师讲过的在store 的test 以及 finally1807 排序 二分查找

二叉树(treeset) 双向链表（增，删除，toString）

利用二分查找找出前100大的数

**public** **class** demo04 {

**public** **static** **void** main(String[] args) {

/\*\*

\* 利用二分查找找出前100最大值

\*/

//创建随机

Random r=**new** Random();

//定义一个100的数组

**long** []arr=**new** **long**[100];

//遍历100万的数字

**for**(**int** i=0;i<1000000;i++) {

//生成一个随机数

**long** n=r.nextLong();

//找出插入点的下标

**int** index=Arrays.*binarySearch*(arr, n);

//如果要插入的数字不存在则其下标结果为-index-1;相反的则为-(index+1)

//如果为正，则正常

index=index<0?-(index+1):index;

//如果下标为0，则跳过当前循环进入下一次

**if**(index==0) **continue**;

//挪位置

**for**(**int** j=1;j<index;j++) {

arr[j-1]=arr[j];

}

//将n插入到下标index之前

arr[index-1]=n;

}

//输出100位数组的数

**for**(**long** l:arr) {

System.***out***.println(l);

}

}

}

利用集合api找出前一百大的数

**public** **class** CaseTest1 {

**public** **static** **void** main(String[] args) {

/\*\*

\* 从100万个数据中找出前100大的数据,

\* 运用的是集合的排序API效率较为的慢(利用二分查找将会更加的快)

\*/

/\*Random r=new Random();

long[]arr=new long[11];

for(int i=0;i<10000000;i++) {

long n=r.nextLong();

arr[0]=n;

Arrays.sort(arr);

}

for(int i=1;i<arr.length;i++) {

System.out.println(arr[i]);

}\*/

Random r=**new** Random();

**long** arr[]=**new** **long**[10];

**for**(**long** i=0;i<1000000;i++) {

**long** n=r.nextLong();

arr[0]=n;

Arrays.*sort*(arr);

}

**for**(**int** i=0;i<arr.length;i++) {

System.***out***.println(arr[i]);

}

}

查出每个字母在字符串中的多少

String str="dadfasffsdzfcsdfshgfjhythdvfsdq";

**int** count[]=**new** **int**[26];

**for**(**int** i=0;i<str.length();i++) {

count[str.charAt(i)-'a']++;

}

System.***out***.println(Arrays.*toString*(count));

}

查出每个数字在字符串中的多少

String str="1234012358786056789456312346";

**int**[]count=**new** **int**[10];

**for**(**int** i=0;i<str.length();i++) {

count[str.charAt(i)-'0']++;

}

System.***out***.println(Arrays.*toString*(count));

}

二分查找

**public** **static** **void** main(String[] args) {

/\*\*

\* 2分查找

\*/

**int** []arr= {8,12,22,35,45,56,57,89,91,98};

//找到返回位置

**int** index=Arrays.*binarySearch*(arr,57);

System.***out***.println(index);

index=Arrays.*binarySearch*(arr,55);

System.***out***.println(index);

}

倒叙输出

@Test

**public** **void** bid() {

String str="i like beijing.";

System.***out***.println("-->"+**new** StringBuilder(str).reverse());}

写一个进制算法

**public** **class** Testjinzhi {

//3进制算法（其他进制以此类推）

**public** **static** **void** main(String[] args) {

String str="123";

**int** base=1;

**int** sum=*parseInt*(str, base);

System.***out***.println(sum);

System.***out***.println(Integer.*parseInt*("123",3));

}

**public** **static** **int** parseInt(String num,**int** base) {

**int** weigth=1;

**int** sum=0;

**for**(**int** i=num.length()-1;i>=0;i--) {

sum+=(num.charAt(i)-'0')\*weigth;

weigth\*=base;

}

**return** sum;

}

双向链表的删除，添加，toString

**package** day03;

/\*\*

\* 双向链表(添加，删除，toString)

\* **@author** soft01

\*

\* **@param** <E>

\*/

**public** **class** LinkedList<E> {

**private** Node head;

**private** **int** size;

**private** **class** Node{

E data;

Node next;

Node prev;

Node(E e){

data=e;

}

}

**public** **boolean** add(E e) {

**if**(head==**null**) {

head=**new** Node(e);

head.prev=head;

head.next=head;

}**else** {

Node node=**new** Node(e);

Node last=head.prev;

head.prev=node;

node.prev=last;

last.next=node;

node.next=head;

}

size++;

**return** **true**;

}

//根据坐标插入

**public** **boolean** add(**int** index,E e) {

//下标越界一种情况

**if**(index<0|| index>size) {

//下标越界异常

**throw** **new** IndexOutOfBoundsException("越界"+index);

}

//如果下标等于size就是添加一个新的数据

**if**(index==size) {

add(e);//追加

}

Node next = findNode(index);

Node prev =next.prev;

//插入节点

Node node=**new** Node(e);

prev.next=node;

node.next=next;

next.prev=node;

node.prev=prev;

size++;

**return** **true**;

}

//找到下标位置

**private** Node findNode(**int** index) {

//找到插入位置

Node next;

**if**(index<size/2) {//正向查找

next=head;

**for**(**int** i=0;i<index;i++) {

next=next.next;

}

}**else** {//反向查找

next=head.prev;

**for**(**int** i=size-1;i>index;i--) {

next=next.prev;

}

}

**return** next;

}

/\*\*

\* 遍历

\*/

**public** String toString() {

**if**(head==**null**) {

**return** "[]";

}

StringBuilder buf=**new** StringBuilder("[").append(head.data);

Node node=head.next;

**while**(node!=head) {//链表的遍历

buf.append(",").append(node.data);

node=node.next;

}

**return** buf.append("]").toString();

}

**public** E remove(**int** index) {

**if**(index<0||index>=size) {

**throw** **new** IndexOutOfBoundsException("越界"+index);

}

**if**(size==1) {

Node n=head;

head=**null**;

size=0;

**return** n.data;

}

Node node=findNode(index);

Node next=node.next;

Node prev=node.prev;

prev.next=next;

next.prev=prev;

**if**(index==0) {

head=next;

}

size--;

**return** node.data;

}

}

二叉树

**package** day03;

/\*\*

\* 二叉树

\* **@author** soft01

\*

\* **@param** <E>

\*/

**public** **class** Tree<E **extends** Comparable<E>> {//comprable可比大小

**private** Node root;

**private** **class** Node{

E data;

Node parent;

Node left;

Node right;

**public** Node(E e) {

data=e;

}

**boolean** add(E e) {

//1.比较数据

**int** val =e.compareTo(data);

**if**(val==0) {

//2.如果一样返回false

**return** **false**;

}**else** **if**(val<0) {

//3.如果小则向left添加

**if**(left==**null**) {

left=**new** Node(e);

left.parent=**this**;

**return** **true**;

}**else** {

**return** left.add(e);

}

}**else** {

//4.如果大则向right添加

**if**(right==**null**) {

right=**new** Node(e);

right.parent=**this**;

**return** **true**;

}**else** {

**return** right.add(e);

}

}

}

//Node 中序遍历

**public** String toString() {

//先输出左，再输出当前data，最后输出右

StringBuilder buf=**new** StringBuilder();

**if**(left!=**null**) {

buf.append(left.toString());

}

buf.append(data).append(", ");

**if**(right!=**null**) {

buf.append(right.toString());

}

**return** buf.toString();

}

}

**public** **boolean** add(E e) {

**if**(root==**null**) {

root=**new** Node(e);

**return** **true**;

}

**return** root.add(e);

}

@Override

**public** String toString() {

**if**(root==**null**) {

**return** "[]";

}

String str=root.toString();

**return** "["+str.substring(0,str.length()-2)+"]";

}

}

2 排序算法

In-place sort （不占用额外的内存和占用常数的内存）

插入排序，选择排序，冒泡排序，堆排序，快速排序

Out-place sort

归并排序，计数排序，基数排序，桶排序

当数据量大的时候，In-place sort就显示出优点，因为只要占用常数的内存

//选择排序（从小到大排序，找出最小值，将最小值放到数组的第一个位置）

**public** **class** SelectionSort {

**public** **static** **void** main(String[] args) {

**int** [] arr= {12,13,45,89,45,1,56,111,5959,0};

*SelectionSort*(arr,arr.length);

System.***out***.println(Arrays.*toString*(arr));

}

**private** **static** **void** SelectionSort(**int** arr[],**int** length) {

//遍历数组

**for**(**int** i=0;i<length;i++) {

**int** index=i;

**for**(**int** j=i+1;j<length;j++) {

//如果arr[j]小于第一个元素，则他们位置互换

**if**(arr[j]<arr[index]) {

index=j;

}

}

//如果相等，则跳出循环

**if**(index==i) **continue**;

//其中index=i表示arr[index]是这次循环的第一位，arr[i]是不断变化的哪个数值

//如果比较不相等，互换

**else** {

**int** temp;

temp=arr[index];

arr[index]=arr[i];

arr[i]=temp;

}

}

}

//插入排序

**public** **class** InsertSort {

**public** **static** **void** main(String[] args) {

**int** arr[]= {12,56,89,45,0,23,11,10,23,12};

//将无序序列插入到有序序列中，遍历无序序列从第二个数字开始

**for**(**int** i=1;i<arr.length;i++) {

**int** j;

//如果插入的数字比有序的最后一个小，则需要进行排序

**if**(arr[i]<arr[i-1]) {

//定义一个有序序列的最大值temp

**int** temp=arr[i];

//因为有新的数字插入进来，进行有序序列的遍历排序

//从i-1开始，条件是j>=0并且arr[i]<arr[i-1]即是插入的数值必须小于

//有序数列的最大值

**for**(j=i-1;j>=0&&temp<arr[j];j--) {

//并且进行互换

arr[j+1]=arr[j];

}

//成功将有序 的最大值赋值给移动后的最后一位

arr[j+1]=temp;

}

}

System.***out***.println(Arrays.*toString*(arr));

}

}

3 spring注解

4 spring的执行流程

5 spring事务

6 事务的特性

7 抽象类与接口的区别

8 集合 hashtable与cuncurenthashmap以及和hashmap的区别

9 数列

10 数组差额插入一个数会移动位置吗

11 string 有哪些方法

12 object有哪些方法

13 多态的特征

一：集合

**11、什么是迭代器(Iterator)？**

Iterator接口提供了很多对集合元素进行迭代的方法。每一个集合类都包含了可以返回迭代器实例的迭代方法。迭代器可以在迭代的过程中删除底层集合的元素,但是不可以直接调用集合的remove(Object Obj)删除，可以通过迭代器的remove()方法删除。

### 1 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，在只有一个线程访问的情况下，效率要高于Hashtable。

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

HashMap 把Hashtable 的contains 方法去掉了，改成containsvalue 和containsKey。因为contains方法容易让人引起误解。

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

最大的不同是，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 就必须为之提供外同步。

Hashtable 和HashMap 采用的hash/rehash 算法都大概一样，所以性能不会有很大的差异。

就HashMap 与HashTable 主要从三方面来说。

一.历史原因:Hashtable 是基于陈旧的Dictionary 类的，HashMap 是Java 1.2 引进的Map 接口的一个实现

二.同步性:Hashtable 是线程安全的，也就是说是同步的，而HashMap 是线程序不安全的，不是同步的

三.值：只有HashMap 可以让你将空值作为一个表的条目的key 或value

### 2.LinkedList和ArrayList的区别

LinkedeList和ArrayList都实现了List接口，但是它们的工作原理却不一样。它们之间最主要的区别在于ArrayList是可改变大小的数组，而LinkedList是双向链接串列(doubly LinkedList)。ArrayList更受欢迎，很多场景下ArrayList比LinkedList更为适用。这篇文章中我们将会看看LinkedeList和ArrayList的不同，而且我们试图来看看什么场景下更适宜使用LinkedList，而不用ArrayList。

**LinkedList和ArrayList的区别**

LinkedList和ArrayList的差别主要来自于Array和LinkedList数据结构的不同。如果你很熟悉Array和LinkedList，你很容易得出下面的结论：

1) 因为Array是基于索引(index)的数据结构，它使用索引在数组中搜索和读取数据是很快的。Array获取数据的时间复杂度是O(1),但是要删除数据却是开销很大的，因为这需要重排数组中的所有数据。

2) 相对于ArrayList，LinkedList插入是更快的。因为LinkedList不像ArrayList一样，不需要改变数组的大小，也不需要在数组装满的时候要将所有的数据重新装入一个新的数组，这是ArrayList最坏的一种情况，时间复杂度是O(n)，而LinkedList中插入或删除的时间复杂度仅为O(1)。ArrayList在插入数据时还需要更新索引（除了插入数组的尾部）。

3) 类似于插入数据，删除数据时，LinkedList也优于ArrayList。

4) LinkedList需要更多的内存，因为ArrayList的每个索引的位置是实际的数据，而LinkedList中的每个节点中存储的是实际的数据和前后节点的位置。

**什么场景下更适宜使用LinkedList，而不用ArrayList**

我前面已经提到，很多场景下ArrayList更受欢迎，但是还有些情况下LinkedList更为合适。譬如：

1) 你的应用不会随机访问数据。因为如果你需要LinkedList中的第n个元素的时候，你需要从第一个元素顺序数到第n个数据，然后读取数据。

2) 你的应用更多的插入和删除元素，更少的读取数据。因为插入和删除元素不涉及重排数据，所以它要比ArrayList要快。

以上就是关于ArrayList和LinkedList的差别。你需要一个不同步的基于索引的数据访问时，请尽量使用ArrayList。ArrayList很快，也很容易使用。但是要记得要给定一个合适的初始大小，尽可能的减少更改数组的大小。

## 3遍历一个List有哪些不同的方式？使用哪种方式更加线程安全？

1）for-each(增强for循环)

2）迭代器遍历（线程更加安全）

## 22. Array和ArrayList有何区别？什么时候更适合用Array？

Array可以容纳基本类型和对象，而ArrayList只能容纳对象。

Array是指定大小的，而ArrayList大小是固定的。

Array没有提供ArrayList那么多功能，比如addAll、removeAll和iterator等。尽管ArrayList明显是更好的选择，但也有些时候Array比较好用。

（1）如果列表的大小已经指定，大部分情况下是存储和遍历它们。

（2）对于遍历基本数据类型，尽管Collections使用自动装箱来减轻编码任务，在指定大小的基本类型的列表上工作也会变得很慢。

（3）如果你要使用多维数组，使用[][]比List<List<>>更容易。

**73、Collection 和 Collections的区别。**Collections是个[java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBmWcYnHF-PvnknW--mvcz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWnLn1bvnHfY" \t "_blank)下的类，它包含有各种有关集合操作的静态方法。   
Collection是个[java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBmWcYnHF-PvnknW--mvcz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWnLn1bvnHfY" \t "_blank)下的接口，它是各种集合结构的父接口。

**74、List、Map、Set三个接口，存取元素时，各有什么特点？List特点：**元素有放入顺序，元素可重复   
**Set特点：**元素无放入顺序，元素不可重复（注意：元素虽然无放入顺序，但是元素在set中的位置是有该元素的HashCode决定的，其位置其实是固定的）   
**Map特点：**元素按键值对存储，无放入顺序 (应该知道什么是键值对吧！)  
List接口有三个实现类：LinkedList，ArrayList，Vector   
LinkedList：底层基于链表实现，链表内存是散乱的，每一个元素存储本身内存地址的同时还存储下一个元素的地址。链表增删快，查找慢   
ArrayList和Vector的区别：ArrayList是非线程安全的，效率高；Vector是基于线程安全的，效率低   
Set接口有两个实现类：HashSet(底层由HashMap实现)，LinkedHashSet   
SortedSet接口有一个实现类：TreeSet（底层由平衡二叉树实现）   
Query接口有一个实现类：LinkList   
Map接口有三个实现类：HashMap，HashTable，LinkeHashMap   
HashMap非线程安全，高效，支持null；HashTable线程安全，低效，不支持null   
SortedMap有一个实现类：TreeMap   
其实最主要的是，list是用来处理序列的，而set是用来处理集的。Map是知道的，存储的是键值对   
set 一般无序不重复.map kv 结构 list 有序

**20、HashMap 和 Hashtable 的区别。**

都属于 Map 接口的类，实现了将惟一键映射到特定的值上。

HashMap 类没有分类或者排序。它允许一个 null 键和多个 null 值。

Hashtable 类似于 HashMap，但是不允许 null 键和 null 值。它也比 HashMap 慢，因为它是同步的。

**21、Collection 和 Collections 的区别。**

Collections 是个 java.util 下的类，它包含有各种有关集合操作的静态方法。

Collection 是个 java.util 下的接口，它是各种集合结构的父接口。

**23、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是 equals()? 它们有何区别?**

Set 里的元素是不能重复的，那么用 iterator()方法来区分重复与否。equals()是判读两个 Set 是否相等。

equals()和==方法决定引用值是否指向同一对象 equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

**33、说出 ArrayList,Vector, LinkedList 的存储性能和特性**

ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了synchronized方法（线程安全），通常性能上较 ArrayList 差，而LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

**14、List、Set、Map的区别**

List：1.可以允许重复的对象。

2.可以插入多个null元素。

3.是一个有序容器，保持了每个元素的插入顺序，输出的顺序就是插入的顺序。

4.常用的实现类有ArrayList、LinkedList和Vector。ArrayList最为流行，它提供了使用索引的随意访问，而LinkedList 则对于经常需要从List中添加或删除元素的场合更为合适。

Set：1.不允许重复对象

2. 无序容器，你无法保证每个元素的存储顺序，TreeSet通过 Comparator或者Comparable维护了一个排序顺序。

3. 只允许一个 null 元素

4.Set 接口最流行的几个实现类是HashSet、LinkedHashSet以及TreeSet。最流行的是基于HashMap实现的HashSet；TreeSet还实现了SortedSet接口，因此TreeSet是一个根据其compare()和 compareTo()的定义进行排序的有序容器。

**15、HashMap和HashTable**

HashMap是线程不安全的,HashMap 是一个接口,是 Map的一个子接口,是将键映射到值得对象,不允许键值重复,允许空键和空值;由于非线程安全, HashMap的效率要较 HashTable 的效率高一些.

HashTable 是线程安全的一个集合,不允许 null 值作为一个 key 值或者Value值;

HashTable是sychronize(同步化),多个线程访问时不需要自己为它的方法实现同步,而HashMap在被多个线程访问的时候需要自己为它的方法实现同步。

**20、HashMap 和 Hashtable 的区别。**

都属于 Map 接口的类，实现了将惟一键映射到特定的值上。

HashMap 类没有分类或者排序。它允许一个 null 键和多个 null 值。

Hashtable 类似于 HashMap，但是不允许 null 键和 null 值。它也比 HashMap 慢，因为它是同步的。

**21、Collection 和 Collections 的区别。**

Collections 是个 java.util 下的类，它包含有各种有关集合操作的静态方法。

Collection 是个 java.util 下的接口，它是各种集合结构的父接口。

**23、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是 equals()? 它们有何区别?**

Set 里的元素是不能重复的，那么用 iterator()方法来区分重复与否。equals()是判读两个 Set 是否相等。

equals()和==方法决定引用值是否指向同一对象 equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

**33、说出 ArrayList,Vector, LinkedList 的存储性能和特性**

ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了synchronized方法（线程安全），通常性能上较 ArrayList 差，而LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

**线程**

##### **75、线程的生命周期？**

**1.线程的生命周期**线程是一个动态执行的过程，它也有一个从产生到死亡的过程。

(1)生命周期的五种状态

**新建（new Thread）**当创建Thread类的一个实例（对象）时，此线程进入新建状态（未被启动）。  
例如：Thread  t1=new Thread();

**就绪（runnable）**线程已经被启动，正在等待被分配给CPU时间片，也就是说此时线程正在就绪队列中排队等候得到CPU资源。例如：**t1.start();**

**运行（running）**线程获得CPU资源正在执行任务（run()方法），此时除非此线程自动放弃CPU资源或者有优先级更高的线程进入，线程将一直运行到结束。

**死亡（dead）**  
当线程执行完毕或被其它线程杀死，线程就进入死亡状态，这时线程不可能再进入就绪状态等待执行。

自然终止：正常运行run()方法后终止

异常终止：调用**stop()**方法让一个线程终止运行

**堵塞（blocked）**  
由于某种原因导致正在运行的线程让出CPU并暂停自己的执行，即进入堵塞状态。

正在睡眠：用sleep(long t) 方法可使线程进入睡眠方式。一个睡眠着的线程在指定的时间过去可进入就绪状态。

正在等待：调用wait()方法。（调用motify()方法回到就绪状态）

被另一个线程所阻塞：调用suspend()方法。（调用resume()方法恢复）

**2.常用方法**

void run()   创建该类的子类时必须实现的方法

void start() 开启线程的方法

static void sleep(long t) 释放CPU的执行权，不释放锁

static void sleep(long millis,int nanos)

final void wait()释放CPU的执行权，释放锁

final void notify()

static void yied()可以对当前线程进行临时暂停（让线程将资源释放出来）

**3.（1）结束线程原理：**就是让run方法结束。而run方法中通常会定义循环结构，所以只要控制住循环即可

(2)方法----可以boolean标记的形式完成，只要在某一情况下将标记改变，让循环停止即可让线程结束

（3）public final void join()//让线程加入执行，执行某一线程join方法的线程会被冻结，等待某一线程执行结束，该线程才会恢复到可运行状态

4.**临界资源：多个线程间共享的数据称为临界资源**

（1）**互斥锁**

a.每个对象都对应于一个可称为“互斥锁”的标记，这个标记用来保证在任一时刻，只能有一个线程访问该对象。

b.Java对象默认是可以被多个线程共用的，只是在需要时才启动“互斥锁”机制，成为专用对象。

c.关键字synchronized用来与对象的互斥锁联系

d.当某个对象用synchronized修饰时，表明该对象已启动“互斥锁”机制，在任一时刻只能由一个线程访问，即使该线程出现堵塞，该对象的被锁定状态也不会解除，其他线程任不能访问该对象

##### **77、启动一个线程是用run()还是start()?**

  启动线程肯定要用start()方法。当用start()开始一个线程后，线程就进入就绪状态，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。当cpu分配给它时间时，才开始执行run()方法(如果有的话)。START()是方法,它调用RUN()方法.而RUN()方法是你必须重写的. run()方法中包含的是线程的主体

**78、当一个线程进入一个对象的一个synchronized方法后，其它线程是否可进入此对象的其它方法?**

可以进入其他非synchronized的方法，synchronized的方法不可以的！

Java中的每个对象都有一个锁（lock）或者叫做监视器（monitor），当访问某个对象的synchronized方法时，表示的将该对象上锁，此时其他任何线程都无法再去访问该synchronized方法了，直到之前的那个线程执行方法完毕后（或者是抛出了异常），才将该对象的锁释放掉，其他线程才有可能再去访问该synchronized方法。

如果一个对象有多个synchronized方法，某一时刻某个线程已经进入到了某个synchronized方法，那么在该方法没有执行完毕前，其他线程是无法访问该对象的任何synchronized方法的。

**23. 创建线程有几种不同的方式？你喜欢哪一种？为什么？**

有三种方式可以用来创建线程：

继承Thread类

实现Runnable接口

应用程序可以使用Executor框架来创建线程池

实现Runnable接口这种方式更受欢迎，因为这不需要继承Thread类。在应用设计中已经继承了别的对象的情况下，这需要多继承（而Java不支持多继承），只能实现接口。同时，线程池也是非常高效的，很容易实现和使用。

**24. 什么是死锁(deadlock)？如何避免deadlock？**

两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程都陷入了无限的等待中。

使用多线程的时候，一种非常简单的避免死锁的方式就是：指定获取锁的顺序，并强制线程按照指定的顺序获取锁。因此，如果所有的线程都是以同样的顺序加锁和释放锁，就不会出现死锁了。

**1、创建线程的几种方式**

一般有四种方法，Thread,Runnable,Callable,使用线程池。

1、通过实现Runnable接口来创建Thread线程。

2、通过实现Callable接口来创建Thread线程。

3、通过继承Thread类来创建一个线程。

4、使用Executor框架来创建线程池。

**2、请说出你所知道的线程同步的方法。**

wait():使一个线程处于等待状态，并且释放所持有的对象的 lock。

sleep(): 使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉InterruptedException 异常。

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。

Allnotity():唤醒所有处入等待状态的线程，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。

**3、线程**

一个进程是一个独立的运行环境，可以看做是一个程序，而线程可以看做是进程的一个任务。在多线程程序中，多线程并发可以提高程序的效率，cpu不会因为某个线程等待资源而进入空闲状态，它会把资源让给其他的线程。用户线程就是我们开发程序是创建的线程，而守护线程为系统线程，如JVM虚拟中的GC线程的优先级别：每一个线程都有优先级别，有限级别高的可以先获取CPU资源使该线程从就绪状态转为运行状态。也可以自定义线程的有限级别。

死锁：至少两个以上线程争取两个以上cpu资源，避免死锁就避免使用嵌套锁，只需要在他们需要同步的地方加锁和避免无限等待。

**异常**

**1、运行时异常与一般异常有何异同？**

异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

**2、Java中的异常处理机制的简单原理和应用**

当JAVA程序违反了JAVA的语义规则时，JAVA虚拟机就会将发生的错误表示为一个异常。违反语义规则包括2种情况。一种是JAVA类库内置的语义检查。例如数组下标越界,会引发IndexOutOfBoundsException;访问null的对象时会引发NullPointerException。另一种情况就是JAVA允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择在何时用throw关键字引发异常。

所有的异常都是java.lang.Thowable的子类

**3、Error与Exception有什么区别？**

Error 表示系统级的错误和程序不必处理的异常，

Exception 表示需要捕捉或者需要程序进行处理的异常。

error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。

exception 表示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。

**4、垃圾回收的优点和原理。并考虑2种回收机制。**

Java语言中一个显著的特点就是引入了垃圾回收机制，使c++程序员最头疼的内存管理的问题迎刃而解，它使得Java程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java中的对象不再有“作用域”的概念，只有对象的引用才有“作用域”。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

**前端**

##### **84、CSS选择器有那些，分别说一下?**

Id选择器

类选择器

选择器组

派生选择器

##### **85、锚点的作用是什么？如何创建锚点？**

锚点是[网页制作](https://www.baidu.com/s?wd=%E7%BD%91%E9%A1%B5%E5%88%B6%E4%BD%9C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLrA7buAf3PHm4uhPhn1D10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHfzP10dPHnv" \t "_blank)中超级链接的一种，又叫[命名锚记](https://www.baidu.com/s?wd=%E5%91%BD%E5%90%8D%E9%94%9A%E8%AE%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLrA7buAf3PHm4uhPhn1D10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHfzP10dPHnv" \t "_blank)。[命名锚记](https://www.baidu.com/s?wd=%E5%91%BD%E5%90%8D%E9%94%9A%E8%AE%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLrA7buAf3PHm4uhPhn1D10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHfzP10dPHnv" \t "_blank)像一个迅速定位器一样是一种页面内的超级链接，运用相当普遍。

**86、什么是javaScript？** JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在[HTML](http://baike.baidu.com/view/692.htm" \t "_blank)（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。

**87、javaScript嵌入HTML页面有几种方式？**1：事件定义式：在元素上定义事件是直接写JS

2.：嵌入式：载script标签里写JS，该标签可以放在head里可以放在 body里

3.：文件调用式：在JS文件中写JS，并引入到页面中使用

1. **简要描述 JavaScript 的数据类型！**字符串、数字、布尔、数组、对象、Null、Undefined

##### **88、举几个 JavaScript 中常用的全局函数，并描述其作用?**

1、decodeURI()  参数:string

     功能描述:对 encodeURI() 函数编码过的 URI 进行解码。

     实例:

       可把 http://www.cnblogs.com/My%20first/ 解码为 http://www.cnblogs.com/My first/

2、decodeURIComponent() 参数:string

功能描述:函数可对 encodeURIComponent() 函数编码的 URI 进行解码。

3、encodeURI() 参数:string

功能描述:可把字符串作为 URI 进行编码。

提示：如果 URI 组件中含有分隔符，比如 ? 和 #，则应当使用 encodeURIComponent() 方法分别对各组件进行编码。

4、encodeURIComponent()

功能描述:可把字符串作为 URI 组件进行编码。

请注意 encodeURIComponent() 函数 与 encodeURI() 函数的区别之处，前者假定它的参数是 URI 的一部分（比如协议、主机名、路径或查询字符串）。

因此 encodeURIComponent() 函数将转义用于分隔 URI 各个部分的标点符号。

5、escape() 参数:string

功能描述:可对字符串进行编码，这样就可以在所有的计算机上读取该字符串。该方法不会对 ASCII 字母和数字进行编码，

也不会对下面这些 ASCII 标点符号进行编码： - \_ . ! ~ \* ' ( ) 。其他所有的字符都会被转义序列替换。

温馨提示：ECMAScript v3 反对使用该方法，应用使用 decodeURI() 和 decodeURIComponent() 替代它。

6、unescape() 参数:string

功能描述:可对通过 escape() 编码的字符串进行解码。该函数的工作原理是这样的：通过找到形式为 %xx 和 %uxxxx 的字符序列（x 表示十六进制的数字），

用 Unicode 字符 \u00xx 和 \uxxxx 替换这样的字符序列进行解码。

温馨提示:ECMAScript v3 已从标准中删除了 unescape() 函数，并反对使用它，因此应该用 decodeURI() 和 decodeURIComponent() 取而代之。

7、eval() 参数:string

功能描述:可计算某个字符串，并执行其中的的 JavaScript 代码。

实例：

document.write(eval("12+2")) 将输出 14

注意：参数必需是string类型的,否则该方法将不作任何改变地返回.

8、isFinite() 参数: number

功能描述:用于检查其参数是否是有穷大的。如果 number 是有限数字（或可转换为有限数字），

那么返回 true。否则，如果 number 是 NaN（非数字），或者是正、负无穷大的数，则返回 false。

实例:

isFinite(-125)和isFinite(1.2)返回true,

而isFinite('易水寒')和isFinite('2011-3-11')返回false.

9、isNaN() 参数:无限制

功能描述:函数用于检查其参数是否是非数字值。

实例:

isNaN(123) 和isNaN(0)返回false

isNaN("易水寒")和isNaN("100") 返回true.

注意:可以用 isNaN() 函数来检测算数错误，比如用 0 作除数的情况。

10、Number() 参数:无限制

功能描述:把对象的值转换为数字。

如果参数是 Date 对象，Number() 返回从 1970 年 1 月 1 日至今的毫秒数。

如果对象的值无法转换为数字，那么 Number() 函数返回 NaN。

**29. JavaScript如何定义数组？**

答：var arrTest = new Array();

**30.JavaScript能否操作cookie和session？**

答：JavaScript可以操作cookie，但是不能操作session。

**31.请写出JavaScript中常用的三种事件？**

答：onclick，onblur，onChange

**32.JS中的三种弹出式消息提醒的命令是什么？**

答：alert，confirm prompt

**6、xml 有哪些解析技术?区别是什么?**

答:有 DOM,SAX,STAX 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由DOM的树结构所造成的，这种结构占用的内存较多，而且DOM必须在解析文件之前把整个文档装入内存,适合对XML的随机访问

SAX:不现于DOM,SAX是事件驱动型的XML解析方式。它顺序读取XML文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理XML文件，适合对XML的顺序访问。

**7、你在项目中用到了 xml 技术的哪些方面?如何实现的?**

用到了数据存贮，信息配置两方面。在做数据交换平台时，将不能数据源的数据组装成 XML文件，然后将XML文件压缩打包加密后通过网络传送给接收者，接收解密与解压缩后再同 XML 文件中还原相关信息进行处理。在做软件配置时，利用 XML 可以很方便的进行，软件的各种配置参数都存贮在XML 文件中。

**44. 说说XML的理解？说明web应用中的web.xml文件的作用？**

答：XML即可扩展标记语言，它与HTML一样，都是SGML标准通用标记语言；

XML是Internet环境中跨平台的，依赖于内容的技术，是当前处理结构化

文档信息的有力工具。扩展标记语言XML是一种简单的数据存储语言，使用

一系列简单的标记描述数据，而这些标记可以用方便的方式建立，虽然XML

占用的空间比二进制数据要占用更多的空间，但XML极其简单易于掌握和使用。

Web.xml的作用是配置欢迎页，servlet，filter，listener等的。

sevlet和jsp

### 1jsp 有哪些内置对象?作用分别是什么? 分别有什么方法？

答:JSP 共有以下9 个内置的对象：

request 用户端请求，此请求会包含来自GET/POST 请求的参数

response 网页传回用户端的回应

pageContext 网页的属性是在这里管理

session 与请求有关的会话期

application servlet 正在执行的内容

out 用来传送回应的输出

config servlet 的构架部件

page JSP 网页本身

exception 针对错误网页，未捕捉的例外

### 2. JSP 和Servlet 有哪些相同点和不同点，他们之间的联系是什么？

JSP 是Servlet 技术的扩展，本质上是Servlet 的简易方式，更强调应用的外表表达。JSP 编译后是"类servlet"。Servlet 和JSP 最主要的不同点在于，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。

### 3. get与post之间的区别：

1>请求参数的存放位置：

get: 在url后面用?拼接

post: 参数放在请求数据包请求实体部分。

2>参数的数据量大小：

get: 传递数据量大小有限制

post: 理论上没有限制, 存放在实体部分

3>安全性：

get: 相对不安全

post: 相对安全

4>编码相关：

get: 不适合传递中文参数

post: 适合传递中文参数

**43. Servlet的生命周期？**

答：1：实例化：容器收到请求时，会创建一个serlvet实例。

2：初始化：容器在创建好servlet对象之后，会接着调用servlet对象的init()方法。注意：该方法只会执行一次。作用是，获取资源。

3：就绪：调用servlet对象的service()方法

4：销毁：容器会依据自身的算法，删除servlet对象。在删除之前，会先调用destroy()方法。

### 4. 同步和异步有何异同，在什么情况下分别使用他们？举例说明。

如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

### 5. Servlet当中如何获取到jsp的9个隐含对象！

JspFactory jf=JspFactory.getDefaultFactory();

// PageContext pageContext=jf.getPageContext(

// this, request, response,null,false,

// JspWriter.DEFAULT\_BUFFER, true);

### 6. 问题：Servlet是如何工作的？Servlet 如何实例化、共享变量、并进行多线程处理？

假设我有一个运行了大量 Servlet 的 web 服务器。通过 Servlet 之间传输信息得到 Servlet 上下文，并设置 session 变量。

现在，如果有两名或更多使用者向这个服务发送请求，接下来 session 变量会发生什么变化？究竟是所有用户都是用共同的变量？还是不同的用户使用的变量都不一样？如果是后者，服务器如何区分不同用户？

另一个相似的问题，如果有 \*n\* 名用户访问一个特定的 Servlet，那么该 Servlet 是仅在第一个用户首次访问的时候实例化，还是分别为每个用户实例化？

回答（BalusC）：

**ServletContext**

当 Servlet 容器（比如 Apache Tomcat）启动后，会部署和加载所有 web 应用。当web 应用被加载，Servlet 容器会创建一次 ServletContext，然后将其保存在服务器的内存中。web 应用的 web.xml 被解析，找到其中所有 servlet、filter 和 Listener 或 @WebServlet、@WebFilter 和@WebListener 注解的内容，创建一次并保存到服务器的内存中。对于所有过滤器会立即调用 init()。当 Servlet 容器停止，将卸载所有 web 应用，调用所有初始化的 Servlet 和过滤器的 destroy() 方法，最后回收 ServletContext 和所有 Servlet、Filter 与 Listener 实例。

当问题中的 Servlet 配置的 load-on-startup 或者 @WebServlet(loadOnStartup) 设置了一个大于 0 的值，则同样会在启动的时候立即调用 init() 方法。“load-on-startup”中的值表示那些 Servlet 会以相同顺序初始化。如果配置的值相同，会遵循 web.xml 中指定的顺序或 @WebServlet 类加载的顺序。另外，如果不设置 “load-on-startup” 值，init() 方法只在第一次 HTTP 请求命中问题中的 Servlet 时才被调用。

**HttpServletRequest 与 HttpServletResponse**

Servlet 容器附加在一个 web 服务上，这个 web 服务会在某个端口号上监听 HTTP 请求，在开发环境中这个端口通常为 8080，生产环境中通常为 80。当客户端（web 浏览器）发送了一个 HTTP 请求，Servlet 容器会创建新的 [HttpServletRequest](http://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletRequest.html" \t "_blank) 和 [HttpServletResponse](http://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletResponse.html" \t "_blank) 对象，传递给已创建好并且请求的 URL 匹配 url-pattern 的 Filter 和 Servlet 实例中的方法，所有工作都在同一个线程中处理。

request 对象可以访问所有该 HTTP 请求中的信息，例如 request header 和 request body。response 对象为你提供需要的控制和发送 HTTP 响应方法，例如设置 header 和 body（通常会带有 JSP 文件中的 HTML 内容）。提交并完成HTTP 响应后，将回收 request 和 response 对象。

**HttpSession**

当用户第一次访问该 web 应用时，会通过 request.getSession() 第一次获得 HttpSession。之后 Servlet 容器将会创建 HttpSession，生成一个唯一的 ID（可以通过 session.getId() 获取）并储存在服务器内存中。然后 Servlet 容器在该次 HTTP 响应的 Set-Cookie 头部设置一个 Cookie，以 JSESSIONID 作为 Cookie 名字，那个唯一的 session ID 作为 Cookie 的值。

按照 HTTP cookie 规则（正常 web 浏览器和 web 服务端必须遵循的标准），当 cookie 有效时，要求客户端（浏览器）在后续请求的 Cookie 头中返回这个 cookie。使用浏览器内置的 HTTP 流量监控器，你可以查看它们（在 Chrome、Firefox23+、IE9+ 中按 F12，然后查看 Net/Network 标签）。Servlet 容器将会确定每个进入的 HTTP 请求的 Cookie 头中是否存在名为JSESSIONID 的 cookie，然后用它的值（session ID）从服务端内存中找到关联的 HttpSession。

你可以在 web.xml 中设置 session-timeout ，默认值为 30 分钟。超时到达之前 HttpSession 会一直存活。所以当客户端不再访问该 web 应用超过 30 分钟后，Servlet 容器就会回收这个 session。后续每个请求，即使指定 cookie 名称也不能再访问到相同的 session。Servlet 容器会创建一个新的 Cookie。

另一方面，客户端上的 session cookie 有一个默认存活时间，该事件和该浏览器实例运行时间一样长。所以，当客户端关闭该浏览器实例（所有标签和窗口）后，这个 session 就会被客户端回收。新浏览器实例不再发送与该 session 关联的 cookie。一个新的 request.getSession() 将会返回新的 HttpSession 并设置一个拥有新 session ID 的 cookie。

**概述**

ServletContext 与 web 应用存活时间一样长。它被所有 session 中的所有请求共享。

只要客户端一直与相同浏览器实例的web应用交互并且没有超时，HttpSession就会存在。

HttpServletRequest 和 HttpServletResponse 的存活时间为客户端发送完成到完整的响应（web 页面）到达的这段时间。不会被其他地方共享。

所有 Servlet、Filter 和 Listener 对象在 web 应用运行时都是活跃的。它们被所有 session 中的请求共享。

你设置在 HttpServletRequest、HttpServletResponse 和 HttpSession 中的所有属性在问题中的对象存活时都会一直保持存活。

**35. JSP页面之间传递参数的方式有哪些？**

答：request、session、application、提交表单、超链接

##### **89、get/post请求的区别?**

Get 请求 传输少量数据 Post请求 传输大量数据

属于不安全 数据安全

一般用来获取数据 用来发送数据

##### **90、JSP中四种对象作用域有哪些区别?**

page：当前页面，也就是只要跳到别的页面就失效了  
request：一次会话，简单的理解就是一次请求范围内有效  
session：浏览器进程，只要当前页面没有被关闭（没有被程序强制清除），不管怎么跳转都是有效的  
application：服务器，只要服务器没有重启（没有被程序强制清除），数据就有效

**1、servlet执行流程**

客户端发出http请求，web服务器将请求转发到servlet容器，servlet容器解析url并根据web.xml找到相对应的servlet，并将request、response对象传递给找到的servlet，servlet根据request就可以知道是谁发出的请求，请求信息及其他信息，当servlet处理完业务逻辑后会将信息放入到response并响应到客户端。

**2、说出Servlet的生命周期**

Servlet被服务器实例化后，容器运行其init方法，请求到达时运行其service方法，service方法自动派遣运行与请求对应的doXXX方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其destroy方法。

**14、描述 Servlet 生命周期**

Servlet 在第一次被访问时创建 Servlet 实例,创建之后服务器会立即调用 init 方法进行初始化的操作,创建之后,Servlet 实例会一直驻留在服务器的内存中,为后续的请求服务. 只要有请求来访问这个 Servlet,服务器就会调用 service 方法来处理请求,直到服务器关闭,或者 WEB 应用被移出容器,随着 WEB 应用的销毁,Servlet 实例也会跟着销毁,在销毁之前服务器会调用 destroy 方法进行善后的处理。

**3、java web的进化历程：**

开始：servlet

因为标签解耦：jsp

因为页面数据解耦：jsp+javabean

因为控制层解耦：jsp+servlet+javabean

因为控制层共用：jsp+struts

因为解耦数据库：jsp+struts+hibernate

因为解耦控制层和数据层：jsp+struts+spring+hibernate

因为界面和控制层解耦：jsp+struts2+spring+hibernate

因为struts2安全问题：jsp+springmvc+spring+hibernate

因为mybatis更能适应业务复杂性

jsp+springmvc+spring+mybatis

........

整个过程前期主要推动因素解耦，后期两次是因为安全和业务复杂性要求。

hibernate其实也可以用于复杂业务复杂sql，性能也可以比较高，但是这样的话对技能要求就比较高。估计是很多人掌握不了其内部的逻辑，瞬时态，持久态，托管态，延时加载，事务边界等。

我们用ssm框架主要是可以解决我们90%的问题，学习比较简单，但是不好的地方就是用spring后性能不好，特别是写数据库多时跟jsp+servlet+javabean比差不多有10倍性能差距，内存消耗也特别大，在高性能高并发的情况下就有点难了，可以了解一下Spring boot+Spring cloud。

**4、JAVA SERVLET API 中 forward() 与 redirect()的区别？**

前者仅是容器中控制权的转向，在客户端浏览器地址栏中不会显示出转向后的地址；后者则是完全的跳转，浏览器将会得到跳转的地址，并重新发送请求链接。这样，从浏览器的地址栏中可以看到跳转后的链接地址。所以，前者更加高效，在前者可以满足需要时，尽量使用 forward()方法，并且，这样也有助于隐藏实际的链接。在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用sendRedirect()方法。

**8、描述 HTTP 协议的作用及具体内容(请求和响应的结构)**

(1)Http协议的作用:用来规定浏览器客户端和服务器之间通信的格式。规定了浏览器在发送请求时该符合什么的格式,在做出响应时该符合什么的格式。

(2)HTTP协议的内容:HTTP 请求(请求行、若干请求头、请求实体内容)、HTTP响应(状态行、若干响应头、响应实体内容)

**9、描述四大域对象的特征**

1、PageContext 域

(1)生命周期:开始访问 JSP 页面时创建 PageContext 对象,访问 JSP 页面结束时销毁 PageContext 对象

(2)作用范围:在整个 JSP 页面中

(3)主要功能:在整个 JSP 页面中实现数据的共享。

2、request 域

(1)生命周期:一次请求开始时创建 request 对象,一次请求结束时销毁 request对象。

(2)作用范围:在整个请求链中

(3)主要功能:在整个请求链中实现数据的共享

3、session 域

(1)生命周期:

创建: 当第一次调用 request.getSession()方法时将会创建 session 对象.

销毁:session 分为三种情况:

a)超时销毁: 如果超过 30 分钟不操作 session, session 将会超时销毁.

b)自杀:当调用session.invalidate()方法时会立即销毁session

c)意外身亡 : 当服务器非正常关闭时, session会销毁! 当服务器正常关闭时,session 将会以文件的形式保存在 tomcat 服务器 work 目录下

(2)作用范围:在整个会话范围内

(3)主要功能:在整个会话范围内实现数据的共享

4、ServletContext 域

(1)生命周期:服务器启动 WEB 应用被加载之后立即创建 ServletContext 对象,服务器关闭或 WEB 应用被移出容器,销毁 ServletContext 对象。

(2)作用范围:在整个 WEB 应用中(3)主要功能:在整个 WEB 应用范围内实现数据的共享

**10、servlet上下文**

什么是servlet上下文?

容器在启动之后，会为每一个web应用创建唯一的一个符合ServletContext接口要求的对象，该对象会一直存在，除非应用被卸载或者容器关闭。

该对象有两个特点:

a.唯一性：一个web应用对应一个上下文。

b.持久性：上下文会一直存在，除非应用被卸载或者容器关闭。

**4、jsp 有哪些动作?作用分别是什么?**

JSP 共有以下6种基本动作

jsp:include：在页面被请求的时候引入一个文件。

jsp:useBean：寻找或者实例化一个 JavaBean。

jsp:setProperty：设置 JavaBean 的属性。

jsp:getProperty：输出某个 JavaBean 的属性。

jsp:forward：把请求转到一个新的页面。

jsp:plugin：根据浏览器类型为 Java 插件生成 objecs 或 EMBED 标记。

**5、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？**

动态 INCLUDE 用 jsp:include 动作实现，它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数。

静态 INCLUDE 用 include 伪码实现,定不会检查所含文件的变化，适用于包含静态页面。

**6、JSP的内置对象及方法。**

request表示HttpServletRequest对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取cookie, header, 和session数据的有用的方法。

response表示HttpServletResponse对象，并提供了几个用于设置送回 浏览器的响应的方法（如cookies,头信息等）。

out对象是javax.jsp.JspWriter的一个实例，并提供了几个方法使你能用于向浏览器回送输出结果。

pageContext表示一个javax.servlet.jsp.PageContext对象。它是用于方便存取各种范围的名字空间、servlet相关的对象的API，并且包装了通用的servlet相关功能的方法。

session表示一个请求的javax.servlet.http.HttpSession对象。Session可以存贮用户的状态信息。

applicaton 表示一个javax.servle.ServletContext对象。这有助于查找有关servlet引擎和servlet环境的信息。

config表示一个javax.servlet.ServletConfig对象。该对象用于存取servlet实例的初始化参数。

page表示从该页面产生的一个servlet实例

**数据库**

**36. Oracle 对象有哪些？并分别说明下用途？**

答：试图，序列，存储函数，同义词，索引，表。

约束条件：保证数据完整性

试图：虚表，命名的查询语句

索引：加速查询的速度

序列：一串连续递增或递减的数字，步长相同，(代理键)。

同义词：一个对象的另外一个叫法(对象的别名)。

存储过程：用于操作。

函数：用作复杂运算的。用于计算。

触发器：由事件触发的存储过程。

**38. 说说数据库触发器的好处？**

答：比如上网发日志，事先会自动通知好友；其实就是在增加发日志时做

了一个后触发，先告知好友，再向通知表中写入信息；效率高。

**39. 说一些关于数据库优化方面的经验？**

答：有外键约束会影响（DML）插入和删除性能，如果程序能够保证数据的完整性，

那么就在创建数据库时去掉外键约束。

扩展：实际开发有一个种叫分区优化技术。

主要的优化方法有如下几点：  
1、表的设计要规范，即要符合[数据库设计](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWPyf3n1DkPAnvujRkuHuW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWc1P1RvnHckPHTLn1Tvnj6Y" \t "_blank)三范式。  
2、适当建立索引，在频繁作为检索条件，更新较少的字段上建立索引，以提高查询速度。  
3、分表查询，有水平分割、垂直分割。  
4、读写分离，读(read)、写(create、update、delete)。  
5、建立存储过程  
6、oracle服务器的升级

**40. 处理数据库大数据量下的分页解决方法？**

答：最好的办法是利用sql语句进行分页，这样每次查询出的结果集中就只包含某页数据内容，再sql语句无法实现分页的情况下，可以考虑对大的结果集通过游标定位方式来获取某页的数据。

总结：简单就是用sql分页语句进行分页；再大的数据量可以考虑对大的结果集

通过游标定位方式来获取某页的数据。

扩展：(说说你对数据库大数据量怎么处理)？

答：可以用分区功能：

范围分区：范围分区就是对数据表中的某个值的范围进行分区，根据某个值的范围，决定将该数据存储在哪个分区上。如根据序号分区，根据时间等来进行分区。

Hash分区(散列分区)：散列分区为通过指定分区编号来均匀分布数据的一种分区类型，因为通过在I/O设备上进行散列分区，使得这些分区大小一致。

也就是只命名分区名称，这样均匀进行数据分布。

复合分区：有时候我们需要根据范围分区后，每个分区内的数据再散列地分布在几个表空间中，这样我们就要使用复合分区。复合分区是先使用范围分区， 然后在每个分区内再使用散列分区的一种分区方法。

**42. JDBC中的perparedStatement比Statement的好处？**

答：一个sql命令发给服务器区执行的步骤是：语法检查，语言分析，

编译成内部指令，缓存指令，执行指令等过程；所以perparedStatement比Statement可以防止sql注入。

##### **79、什么是数据库的事务？**

(1):事务（Transaction）是数据库运行中的一个逻辑工作单位，是用户定义的一个逻辑操作。这些操作要么不成功，要么都成功，是一个不可分割的工作单位。通过事务，SQL Server能将逻辑相关的一组操作绑定在一起，以便服务器保持数据的完整性。   
(2):事务通常是以begin transaction开始，以COMMIT或ROLLBACK结束。   
COMMIT表示提交，即提交事务的所有操作。具体地说就是将事务中所有对数据库的更新写回到磁盘上的物理数据库中去，事务正常结束。   
ROLLBACK表示回滚，即在事务运行的过程中发生了某种故障，事务不能继续进行，系统将事务中对数据库的所有以完成的操作全部撤消，滚回到事务开始的状态。   
(3):事务运行的三种模式:   
A:自动提交事务   
每条单独的语句都是一个事务。每个语句后都隐含一个COMMIT。   
B:显式事务   
以BEGIN TRANSACTION显式开始，以COMMIT或ROLLBACK显式结束。   
C:隐性事务   
在前一个事务完成时，新事务隐式启动，但每个事务仍以COMMIT或ROLLBACK显式结束。   
(4):事务的特性(ACID特性)   
A:原子性(Atomicity)   
事务是数据库的逻辑工作单位，事务中包括的诸操作要么全做，要么全不做。   
B:一致性(Consistency)   
事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。   
C:隔离性(Isolation)   
一个事务的执行不能被其他事务干扰。   
D:持续性/永久性(Durability)   
一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。   
注:事务是恢复和并发控制的基本单位。

##### **80、在数据库中条件查询速度很慢的时候,如何优化?**

1.建索引（a 提高查询检索的性能b 创建唯一索引c 创建主键d

归类）  
2.减少表之间的关联  
3.优化sql，尽量让sql很快定位数据，不要让sql做全表查询，

应该走索引,把数据量大的表排在前面  
4.简化查询字段，没用的字段不要，已经对返回结果的控制，尽

量返回少量数据

##### **81、数据库存储过程的优点？**

1.存储过程可以使得程序执行效率更高、安全性更好，因为过程建立之后 已经编译并且储存到[数据库](http://www.2cto.com/database/" \t "_blank)，直接写sql就需要先分析再执行因此过程效率更高，直接写sql语句会带来安全性问题，如：sql注入

2.建立过程不会很耗[系统](http://www.2cto.com/os/" \t "_blank)资源，因为过程只是在调用才执行。

3.存储过程可以用于降低网络流量，存储过程代码直接存储于数据库中，所以不会产生大量T-sql语句的代码流量。

4.使用存储过程使您能够增强对执行计划的重复使用，由此可以通过使用远程过程调用 (RPC) 处理服务器上的存储过程而提高性能。RPC 封装参数和调用服务器端过程的方式使引擎能够轻松地找到匹配的执行计划，并只需插入更新的参数值。

5.可维护性高，更新存储过程通常比更改、测试以及重新部署程序集需要较少的时间和精力。

6.代码精简一致，一个存储过程可以用于应用程序代码的不同位置。

7.增强安全性：

a、通过向用户授予对存储过程（而不是基于表）的访问权限，它们可以提供对特定数据的访问；

b、提高代码安全，防止 SQL注入（但未彻底解决，例如，将数据操作语言－－DML，附加到输入参数）；

c、SqlParameter 类指定存储过程参数的数据类型，作为深层次防御性策略的一部分，可以验证用户提供的值类型（但也不是万无一失，还是应该传递至数据库前得到附加验证）。

##### **82、JDBC连接数据库的生命周期是什么？**

1、加载JDBC驱动程序：   
 在连接数据库之前，首先要加载想要连接的数据库的驱动到JVM（Java虚拟机），   
 3、创建数据库的连接   
 要连接数据库，需要向java.sql.DriverManager请求并获得Connection对象， 该对象就代表一个数据库的连接。

4、创建一个Statement

•要执行[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yvny7hryn4njR4uHD1mHuB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH03PWRYPHb3" \t "_blank)，必须获得java.sql.Statement实例，Statement实例分为以下3种类型：   
1、执行静态[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yvny7hryn4njR4uHD1mHuB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH03PWRYPHb3" \t "_blank)。通常通过Statement实例实现。   
2、执行动态[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yvny7hryn4njR4uHD1mHuB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH03PWRYPHb3" \t "_blank)。通常通过PreparedStatement实例实现。   
3、执行数据库存储过程。通常通过CallableStatement实例实现。   
5、执行SQL语句   
Statement接口提供了三种执行SQL语句的方法：executeQuery 、executeUpdate 和execute   
6、处理结果

两种情况：   
1、执行更新返回的是本次操作影响到的记录数。   
2、执行查询返回的结果是一个ResultSet对象。   
7、关闭JDBC对象   
操作完成以后要把所有使用的JDBC对象全都关闭，以释放JDBC资源，关闭顺序和声明

##### **83、数据连接池的工作机制是什么？**

该模式正是为解决资源频繁分配、释放所造成的问题。数据库连接池的基本思想就是为数据库连接建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需要从缓冲池中取出一个了，使用完毕后再放回去。我们可以通过设定连接池最大数来防止系统无尽的与数据库连接。更为重要的是我们可以通过连接池的管理机制监视数据库连接使用数量，使用情况，为系统开发，测试以及性能调整提供依据。

连接池的工作原理：

连接池的核心思想是连接的复用，通过建立一个数据库连接池以及一套连接使用、分配和管理策略，使得该连接池中的连接可以得到高效，安全的复用，避免了数据库连接频繁建立和关闭的开销。

连接池的工作原理主要由三部分组成，分别为连接池的建立，连接池中连接的使用管理，连接池的关闭。

1. **连接池的建立。**

一般在系统初始化时，连接池会根据系统配置建立，并在池中建立几个连接对象，以便使用时能从连接池中获取，连接池中的连接不能随意创建和关闭，这样避免了连接随意建立和关闭造成的系统开销。java中提供了很多容器类，可以方便的构建连接池，例如Vector,stack等。

1. **连接池的管理**。

连接池管理策略是连接池机制的核心，连接池内连接的分配和释放对系统的性能有很大的影响。其策略是：

当客户请求数据库连接时，首先查看连接池中是否有空闲连接，如果存在空闲连接，则将连接分配给客户使用；如果没有控线连接，则查看当前所开的连接数是否已经达到最大连接数，例如如果没有达到就重新创建一个请求的客户；如果达到，就按设定的最大等待时间进行等待，如果超出最大等待时间，则抛出异常给客户。

当客户释放数据库连接时，先判断该连接的引用次数是否超过了规定值，如果超过了就从连接池中删除该连接，否则就保留为其他客户服务。该策略保证了数据库连接的有效复用，避免了频繁建立释放连接所带来的系统资源的开销。

1. **连接池的关闭**。

当应用程序退出时，关闭连接池中所有的链接，释放连接池相关资源，该过程正好与创建相反。比如公交车一样

**7、数据库优化**

选择合适的字段，比如邮箱字段可以设为char（6），尽量把字段设置为notnull，这样查询的时候数据库就不需要比较null值

使用关联查询（ left join on）查询代替子查询，开启事物，当数据库执行多条语句出现错误时，事物会回滚，可以维护数据库的完整性，使用外键，事物可以维护数据的完整性但是它却不能保证数据的关联性，使用外键可以保证数据的关联性，使用索引，索引是提高数据库性能的常用方法，它可以令数据库服务器以比没有索引快的多的速度检索特定的行，特别是对于max，min，order by查询时，效果更明显，优化的查询语句，绝大多数情况下，使用索引可以提高查询的速度，但如果sql语句使用不恰当的话，索引无法发挥它的特性。

**8、事物的理解**

事物具有原子性，一致性，持久性，隔离性

原子性：是指在一个事物中，要么全部执行成功，要么全部失败回滚。

一致性：事物执行之前和执行之后都处于一致性状态

持久性：事物多数据的操作是永久性

隔离性：当一个事物正在对数据进行操作时，另一个事物不可以对数据进行操作，也就是多个并发事物之间相互隔离。

**9、session和cookie**

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的cookie并进行cookie欺骗，考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用cookie。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、可以考虑将登陆信息等重要信息存放为session，其他信息如果需要保留，可以放在cookie中。

cookie是Web服务器发送给浏览器的一块信息。浏览器会在本地文件中给每一个Web服务器存储cookie。以后浏览器在给特定的Web服务器发请求的时候,同时会发送所有为该服务器存储的cookie。

下面列出了session和cookie的区别:

无论客户端浏览器做怎么样的设置,session都应该能正常工作。客户端可以选择禁用cookie,但是,session仍然是能够工作的,因为客户端无法禁用服务端的session。在存储的数据量方面session和cookies也是不一样的。session能够存储任意的Java对象,cookie只能存储String类型的对象。

**10、JDBC操作数据库的基本步骤：**

1）加载（注册）数据库驱动（到JVM）。

2）建立（获取）数据库连接。

3）创建（获取）数据库操作对象。

4）定义操作的SQL语句。

5）执行数据库操作。

6）获取并操作结果集。

7）关闭对象，回收数据库资源（关闭结果集-->关闭数据库操作对象-->关闭连接）。

**12、内联接,外联接区别?**

内连接：只查询满足连接条件的数据,连接条件以外的数据则被过滤掉。

外链接：不满足连接条件的数据,连接条件以外的数据也查询出来,外链接分为左外连接、右外连接和全外连接。

外连接:

1)左连接(左外连接)以左表为基准进行查询,左表数据会全部显示出来,右表如果和左表匹配的数据则显示相应字段的数据,如果不匹配,则显示为 NULL;

2)右连接(右外连接)以右表为基准进行查询,右表数据会全部显示出来,右表如果和左表匹配的数据则显示相应字段的数据,如果不匹配,则显示为 NULL;

3)全连接就是先以左表进行左外连接,然后以右表进行右外连接。

内连接:显示表之间有连接匹配的所有行。

**13、五种约束是什么,分别起什么作用**

主键 primary key：一行数据的唯一标识,不重复且不能为空,自动创建索引

外键 foreign key：只能引用指定的主键字段中存在的值,可以取 null 值,自动创建索引

唯一 unique：限制字段中的取值不能重复,可以取 null 值,自动创建索引

非空 not null：限制字段取值不能取 null 值

检查 check：检查字段中的取回是否符合指定的条件,比如年龄范围,性别范围等。（MySql 不支持检查约束）

**14、索引的作用和缺点**

索引可以加速字段的检索效率,类似于书的目录,可以快速定位想要的数据;

缺点:对表中数据进行增删改操作时,需要同时修改索引数据,降低效率,另外,索引本身会占用大量存储空间。

**15、什么是事务**

事务将一组 SQL 语句作为一个操作整体,所有 SQL 语句操作成功,事务整体才成功;任何一个语句操作失败那么整个操作失败,回滚到操作前状态。事务具有四个特性:ACID,即原子性,一致性,隔离性和持久性。

**16、什么是存储过程?它有什么优点?**

存储过程是一组予编译的 SQL 语句,它的优点有:允许模块化程序设计,就是说只需要创建一次过程,以后在程序中就可以调用该过程任意次。允许更快执行,如果某操作需要执行大量 SQL 语句或重复执行,存储过程比 SQL 语句执行的要快。减少网络流量,例如一个需要数百行的 SQL 代码的操作有一条执行语句完成,不需要在网络中发送数百行代码。更好的安全机制,对于没有权限执行存储过程的用户,也可授权他们执行存储过程。

**框架**

**45.Http协议的一些技术特点是什么？**

1.支持客户/服务器模式。

2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。

**4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。**

**5.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。**

##### **45、什么是 HTTP 协议?**

HTTP 协议定义:[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvnAn1PAfLm1mYuHmvPWcv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWRzrHDYP16d" \t "_blank)和客户端之间[文件传输](https://www.baidu.com/s?wd=%E6%96%87%E4%BB%B6%E4%BC%A0%E8%BE%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvnAn1PAfLm1mYuHmvPWcv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWRzrHDYP16d" \t "_blank)的沟通方式

最基本过程:   
1、客户端连接一个主机  
2、服务器接收连接  
3、客户端请求一个文件   
4、服务器发送一个应答

**46. 什么是web容器？**

答：容器就是一种服务程序，在服务器一个端口就有一个提供相应服务的程序，

而这个程序就是处理从客户端发出的请求，如JAVA 仲的Tomcat容器，ASP

的IIS 或 PWS 都是這样的容器。

**47. Tomcat服务器的默认端口是多少？怎样修改tomcat的端口？**

答：默认端口为8080，可以通过service.xml的connector元素的port属性来修改

端口。

**48. 基于<bean>（xml配置）的Spring MVC控制流程：**

客户端发送请求经过DispatcherServlet控制器处理之后，通过HandlerMapping组件找到

相应的Controller控制器，Controller会调用业务层Model处理请求并且返回ModelAndView，

然后DispatcherServlet会通过ViewResolver返回相应的jsp视图，在客户端进行页面呈现；

**49. 接收请求参数值的3种方式：**

-使用HttpServletRequest获取（getParameter()方法）

\*优点：直接获取数据，获取到的数据比较完整（表单，ip，url，cookies等）；

缺点：需要自己处理数据类型转换；

-Spring会自动将表单name属性参数注入到方法参数中（参数名一致）

使用@RequestParam注解映射参数名不一致的属性

\*优点：数据类型自动转换；

缺点：有可能出现数据类型转换异常；

-使用自动机制封装成Bean属性

定义FromBean实体类，属性名与表单组件的name名一致

\*优点：表单属性量大，解耦，简化代码量；

**50.说说Tomcat有几种部署方式？**

答：（1）利用Tomcat自动部署

（2）利用控制台进行部署

3）增加自定义的Web部署文件(%Tomcat\_Home%\conf\Catalina\localhost\AppName.xml)

（4）手动修改%Tomcat\_Home%\conf\server.xml文件来部署web应用

**51.Tomcat的优化经验？**

答：去掉对web.xml的监视，把jsp提前编辑成Servlet。

有富余物理内存的情况，加大tomcat使用的jvm的内存。

**52.说说您对Ajax的理解，还有为什么要用Ajax？有什么好处？**

答：Ajax就是异步的JavaScript跟xml同时访问数据库。

**53.Hibernate是什么及优缺点？**

答：hibernate是基于ORM对象关系映射（完成对象数据到关系数据映射的机制）实现的，做数据持久化的工具。

Hibernate的缺点:

1.性能比较差，尤其是批处理方面，如果在大数据量开发的时候，最好使用JDBC。

**54.Hibernate的核心接口有哪些？**

答：Session、SessionFactory、Transaction、Query、Criteria、Configuration

**55.Hibernate 中，不看数据库，不看XML文件，不看查询语句，怎么样能知道表结构？**

答：可以看与XML文件对应的域模型。

### 134、锁机制有什么用？简述Hibernate的悲观锁和乐观锁机制。

答：有些业务逻辑在执行过程中要求对数据进行排他性的访问，于是需要通过一些机制保证在此过程中数据被锁住不会被外界修改，这就是所谓的锁机制。

Hibernate支持悲观锁和乐观锁两种锁机制。悲观锁，顾名思义悲观的认为在数据处理过程中极有可能存在修改数据的并发事务（包括本系统的其他事务或来自外部系统的事务），于是将处理的数据设置为锁定状态。悲观锁必须依赖数据库本身的锁机制才能真正保证数据访问的排他性，关于数据库的锁机制和事务隔离级别在[《Java面试题大全（上）》](http://blog.csdn.net/jackfrued/article/details/44921941" \t "_blank)中已经讨论过了。乐观锁，顾名思义，对并发事务持乐观态度（认为对数据的并发操作不会经常性的发生），通过更加宽松的锁机制来解决由于悲观锁排他性的数据访问对系统性能造成的严重影响。最常见的乐观锁是通过数据版本标识来实现的，读取数据时获得数据的版本号，更新数据时将此版本号加1，然后和数据库表对应记录的当前版本号进行比较，如果提交的数据版本号大于数据库中此记录的当前版本号则更新数据，否则认为是过期数据无法更新。Hibernate中通过Session的get()和load()方法从数据库中加载对象时可以通过参数指定使用悲观锁；而乐观锁可以通过给实体类加整型的版本字段再通过XML或@Version注解进行配置。

**提示：**使用乐观锁会增加了一个版本字段，很明显这需要额外的空间来存储这个版本字段，浪费了空间，但是乐观锁会让系统具有更好的并发性，这是对时间的节省。因此乐观锁也是典型的空间换时间的策略。

**56.Struts是什么？**

答：struts1是基于JSP和servlet的一个开源的Web应用框架，使用的是MVC的设计模式 struts2是基于webwork技术的框架，是sun和webwork公司联手开发的一个功能非常齐全的框架，struts2和struts1没有关系，是一个全新的框架。

**57. 说说 你对Struts的理解？**

答：struts是一个按MVC模式设计的Web层框架，其实它就是一个大大的servlet，

这个Servlet名为ActionServlet，或是ActionServlet的子类。我们可以在web.xml文件中 将符合某种特征的所有请求交给这个Servlet处理，这个Servlet再参照一个配置文件（通常为/WEB-INF/struts-config.xml）将各个请求分别分配给不同的action去处理。

总结：struts其实是以MVC设计模式的web层框架，也就是一个大大的servlet，

而这个servlet我们称之为ActionServlet，我们可以在web.xml配置文件中

将符合某种特征的请求交给对应的Servlet类处理，再通过struts.xml配置文

件对应的action处理。

**58.Spring是什么？**

答：spring是一个集成了许多第三方框架的大杂烩，其核心技术是IOC（控制反转，也称依赖注入）和AOP（面向切面编程）。

**59.什么是AOP？什么IOC 及好处什么？**

答：-- AOP简称为：面向方面编程或面向切面编程。

-- AOP关注点是共同处理，可以通过配置将其作用到某一个或多个目标对象上。

好处：是实现组件复利用，改善程序的结构，提高灵活性。将共通组件与目标对象解耦。

**60.什么是IOC？**

答： -- IOC主要是解决两个组件对象调用的问题，可以以低耦合方式建立使用的关系。

**61.依赖注入指的是什么？**

答：依赖注入指的是在运行期间，由外部容器(BeanFactory)动态体将依赖对象(DAO)注入到组件(Action)中。

**62.Bean注入方式与注入数据类型？**

答：注入方式分为

Setter注入法 默认 <bean> <property>

构造方法注入法 <bean> <constructor-arg>

注入数据类型分为

引用型注入 ref

基本数据类型注入 value

集合类型注入 <list><set><map><props>

**63.在项目中用过Spring的哪些方面？及用过哪些Ajax框架？**

答：在项目使用过SpringIOC,AOP,DAO,ORM，还有上下文环境。

在项目使用过Ext，Juery等Ajax框架。

**64. 从控制器向页面传值的4种方式（将业务处理结果在界面显示）：**

-使用HttpServletRequest，session，application的setAttribute()方法

-使用ModelAndView对象

Controller处理完业务之后通过散列表HashMap接收处理结果 ，之后将结果数据绑定到

ModelAndView对象中，通过绑定名向jsp页面传递数据；

Map<String,Object> map=

new HashMap<String,Object>();

map.put("绑定名",业务处理结果);

-使用ModelMap参数对象

在Controller处理业务的方法中追加一个ModelMap类型的参数，ModelMap会利用

HttpServletRequest的Attribute将数据传递到jsp页面；

model.addAttribute("绑定名",业务处理结果);

-使用@ModelAttribute注解

在Controller处理业务方法的参数部分或者Bean属性方法上使用@ModelAttribute注解

@ModelAttribute会利用HttpServletRequest的Attribute将数据传递到jsp页面；

##### **91、MVC的各个部分都有那些技术来实现?如何实现?**

视图层：jsp、html超文本标记语言

控制层： serverlet

数据层：mybatis

##### **92、什么是 Ajax，优点是什么？**

本质是浏览器的特殊对象，异步发出请求，服务器返回更新页面不用刷新页面

优点：减轻服务器的负担,按需取数据,最大程度的减少冗余请求  
局部刷新页面,减少用户心理和实际的等待时间,带来更好的用户体验  
基于xml标准化,并被广泛支持,不需安装插件等  
进一步促进页面和数据的分离  
缺点：[AJAX](https://www.baidu.com/s?wd=AJAX&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YduAwBmH0kPjRLnjNhrymk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc3PWRYPjRL" \t "_blank)大量的使用了javascript和[ajax](https://www.baidu.com/s?wd=ajax&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YduAwBmH0kPjRLnjNhrymk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc3PWRYPjRL" \t "_blank)引擎,这些取决于浏览器的支持.在编写的时候考虑对浏览器的兼容性.  
[AJAX](https://www.baidu.com/s?wd=AJAX&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YduAwBmH0kPjRLnjNhrymk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc3PWRYPjRL" \t "_blank)只是局部刷新,所以页面的后退按钮是没有用的.

##### **93、什么是同步请求，什么是异步请求**

同步：提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事   
异步: 请求通过事件触发->服务器处理（这是浏览器仍然可以作其他事情）->处理完毕

**1、AOP与IOC的概念（即spring的核心）**

IOC：Spring是开源框架，使用框架可以使我们减少工作量，提高工作效率并且它是分层结构，即相对应的层处理对应的业务逻辑，减少代码的耦合度。而spring的核心是IOC控制反转和AOP面向切面编程。IOC控制反转主要强调的是程序之间的关系是由容器控制的，容器控制对象，控制了对外部资源的获取。而反转即为，在传统的编程中都是由我们创建对象获取依赖对象，而在IOC中是容器帮我们创建对象并注入依赖对象，正是容器帮我们查找和注入对象，对象是被获取，所以叫反转。

AOP：面向切面编程，主要是管理系统层的业务，比如日志，权限，事物等。AOP是将封装好的对象剖开，找出其中对多个对象产生影响的公共行为，并将其封装为一个可重用的模块，这个模块被命名为切面（aspect），切面将那些与业务逻辑无关，却被业务模块共同调用的逻辑提取并封装起来，减少了系统中的重复代码，降低了模块间的耦合度，同时提高了系统的可维护性。

**2、描述 Spring IOC 机制及原理**

Spring中IOC的核心机制是

a)由Spring框架实现对象的创建及生命周期的管理,以更好利用系统资源。

b)由Spring框架管理对象之间依赖,更好实现对象之间的解耦。

Spring中IOC的实现原理

a)容器初始化时依据元数据的描述通过反射创建对象.

b)对象创建时或创建以后通过元数据的描述通过 set 或构造注入方式为属性赋值

c)外界需要对象时,从spring容器中根据 key查询并获取对象。

**3、描述 Spring AOP 机制及原理**

AOP 机制：基于OCP原则,实现扩展功能的动态切入,例如事务管理,权限检测等

AOP 原理：通过jdk或cglib为目标对象创建代理对象,然后通过代理对象实现功能扩展。

**3、Spring**

(1)Spring是什么?

是一个轻量级的、用来简化企业级应用开发的开发框架。Spring对常用的api做了简化，比如，使用Spring,jdbc来访问数据库，就不用再考虑如何获取连接、关闭连接、处理异常等等。Spring容器(Spring框架的一个模块)帮我们管理对象（包括对象的创建及对象之间的依赖关系），这样一来，对象之间的耦合度会大大降低，提高了系统的维护性。Spring可以将其它一些常用的框架集成进来，比如可以将Quartz,MyBatis等集成进来。

(2)Spring容器

Spring框架的一个核心模块，用于管理对象。

注：管理对象指的是，spring容器会用我们创建对象，并且建立对象之间的依赖关系。

**4、SpringMVC框架**

1. 作用

解决了V-C的交互问题，即视图与控制器的交互问题。在原生的Java EE技术中，使用Servlet作为项目中的控制器，用于接收用户的请求，并给予响应结果。这种做法最大的问题在于：在常规做法中，每个Servlet对应1个请求路径，例如LoginServlet处理login.do的请求，而RegisterServlet处理register.do的请求，所以，会导致Servlet数量太多，不便于管理（无论从源文件的数量，还是从配置文件的内容），且对象太多进而占用大量内存空间的问题！并且，在请求的处理过程中，还存在许多操作不简便的做法！

2. SpringMVC核心组件

DispatcherServlet:前端控制器，主要职责是接收所有请求（根据配置文件来决定），并将请求转发给对应的控制器，接收控制器的处理结果，确定最终由哪个视图完成响应！

HandlerMapping:处理请求路径与控制器的映射关系。

Controller:实际处理请求的组件，例如接收请求参数，决定最终是转发或重定向的方式来响应。

ModelAndView:控制器的处理结果，其中的Model表示转发的数据（如果是重定向，则Model没有意义），而View表示最终负责响应的视图组件的名称。

ViewResolver:根据视图组件的名称，确定具体使用的是哪个视图组件。

**5、MVC 的各个部分都有那些技术来实现?如何实现?**

MVC 是 Model－View－Controller 的简写。"Model" 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现）， "View" 是应用的表示面（由 JSP 页面产生），"Controller" 是提供应用的处理过程控制（一般是一个 Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

**6、J2EE 是什么？**

Je22是Sun公司提出的多层(multi-diered),分布式(distributed),基于组件(component-base)的企业级应用模型(enterpriese application model).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietn tier)组件,web层和组件,Business 层和组件,企业信息系统(EIS)层。

**7、两种跳转方式分别是什么?有什么区别?**

答：有两种，分别为： 转发，重定向

前者页面不会转向 include 所指的页面，只是显示该页的结果，主页面还是原来的页面。执行完后还会回来，相当于函数调用。并且可以带参数.后者完全转向新页面，不会再回来。

在转发过程中，客户端只发出过1次请求！在浏览器的地址栏中，也只会显示第1次请求的路径！转发是在服务器内部完成的，可以传递数据！

当服务器响应重定向时，客户端会发出第2次请求！最终，在浏览器的地址栏中，会显示第2次请求的路径！由于是2次不同的请求，基于Http协议是无状态协议，没有经过特殊处理（Session/Cookie/数据库存取……）的数据是无法在2次请求之间传递的！

**10、spring mvc 处理流程**

1)客户端向服务端发起请求

2)请求经过滤器到达 DispatcherServlet 前端控制器,并初始化 Spring MVC 配置。

3)前端控制器 DispatcherServlet 将请求资源 url 传递给 HandlerMapping 对象处理。

4)HandlerMapping 对象依据 URL 获取对象请求的执行链对象,然后先执行拦截器,再执行后端处理器对象

5)将后台控制器的处理结果例如视图和数据封装到 ModelAndView 对象,最后返回给spring mvc

6)Spring mvc 解析控制层方法的返回结果(例如视图解析器解析视图)再进一步处理。

7)请求处理结束,返回客户端相关信息

**11、spring mvc 应用的优点**

1)解决了请求及响应时数据的处理细节问题(例如参数类型转换及赋值等操作)

2)更好的实现对象解耦,简化了代码编写,提高程序可读性,以及可维护性。

**12、Nginx 主要实现负载均衡以及方向代理.**

负载均衡的是指根据特定均衡规则,向不同的服务器发出请求.负载均衡的主要方式有

1. 轮询方式:根据 upstream 中的配置项依次项服务端发出请求.

2. 权重方式:根据数据库的性能,可以通过权重的方式,选择服务器

3. IP\_hash: 根 据 Ip 地 址 绑 定 服 务 器 . 虽 然 能 够 实 现 Session 共 享 , 但 是Session 黏着问题比较严重.负载均衡中还有备用机机制,当主服务器忙时不能及时提供才会启动备用机.同时 Nginx 的启动和关闭都是秒级,所以使用 nginx 在部署时很方便.

**13、反向代理服务:**

可以通过 Nginx 中内部 conf.xml 的配置.实现具体的反向代理实现,反向代理主要是客户端发出请求后,由 nginx 经过内部处理,将请求再次转发.发往目标地址.同时当 Nginx 接收到数据后,再由 nginx 发给客户端,最终实现用户请求.

**15、什么是会话?**

将浏览器和服务器之间的多次请求和响应加在一起就可以看作浏览器和服务器之间的一次会话

**16、 描述什么是过滤器及过滤器的作用**

1、过滤器也叫做 Filter,是 JavaWeb 的三大组件之一。可以拦截用户对资源的访问。

2、所谓的拦截其实就是将代表请求的 request 对象和代表响应的 response 对象拦截下来。

3、通过过滤器可以实现在用户访问某个资源之前或者之后,对访问的请求和响应进行拦截,从而做一些相关的处理。

**17、拦截器与过滤器有什么区别**

拦截器是Interceptor，过滤器是Filter；

拦截器是SpringMVC中的组件，过滤器是Java EE中的组件；

拦截器是配置在Spring的配置文件中的，过滤器是配置在web.xml中的；

拦截器是运行在DispatcherServlet之后、Controller之前的，且在Controller执行完后还会调用2个方法，而过滤器是运行在所有的Servlet之前的；

拦截器的配置非常灵活，可以配置多项黑名单，也可以配置多项白名单，过滤器的配置非常单一，只能配置1项过滤路径；

拦截器与过滤器也有很多相似之处，例如：都可拒绝掉某些访问，也可以选择放行；都可以形成链。

相比之下，在一个使用SpringMVC框架的项目中，拦截器会比过滤器要好用一些，但是，由于执行时间节点的原因，它并不能完全取代过滤器！

**17、Spring 中的事务处理方式及优缺点**

Spring 中的事务处理有两种方式:编程式事务处理和声明式事务处理。

编程式事务处理:需手动编写事务处理代码,代码量比较大,侵入性比较高,现已经不推荐使用。

声明式事务处理:采用 xml 或注解方式进行低侵入性事务处理,代码量比较小,开发效率高。可维护性更好。

**18、Spring 的理解**

首先,spring 是一个开源框架,Spring 为简化企业级应用开发而生,使用 Spring 可以使简单的 JavaBean 实现以前只有 EJB 才能实现的功能。Spring 是一个 IOC 和 AOP 容器框架。

Spring 主要核心是:

(1)控制反转(IOC):传统的 java 开发模式中,当需要一个对象时,我们会自己使用 new 或者 getInstance 等直接或者间接调用构造方法创建一个对象,而在 Spring 开发模式中,Spring 容器使用了工厂模式为我们创建了所需要的对象,我们使用时不需要自己去创建,直接调用 Spring 为我们提供的对象即可,这就是控制反转的思想。实例化一个 java 对象有三种方式:使用类构造器,使用静态工厂方法,使用实例工厂方法,当使用 spring 时我们就不需要关心通过何种方式实例化一个对象,spring 通过控制反转机制自动为我们实例化一个对象。

(2)依赖注入(DI):Spring 使用 Java Bean 对象的 Set 方法或者带参数的构造方法为我们在创建所需对象时将其属性自动设置所需要的值的过程就是依赖注入的基本思想。

(3)面向切面编程(AOP):在面向对象编程(OOP)思想中,我们将事物纵向抽象成一个个的对象。而在面向切面编程中,我们将一个个对象某些类似的方面横向抽象成一个切面,对这个切面进行一些如权限验证,事物管理,记录日志等公用操作处理的过程就是面向切面编程的思想。

在Spring中,所有管理的对象都是 JavaBean 对象,而 BeanFactory 和 ApplicationContext就是 spring 框架的两个 IOC 容器,现在一般使用 ApplicationContext,其不但包含了BeanFactory 的作用,同时还进行更多的扩展。

**19、Spring Bean 生命周期**

1.Spring 容器 从 XML 文件中读取 Bean 的定义,并实例化 Bean。

2.Spring 根据 Bean 的定义填充所有的属性。

3.如果 Bean 实现了 BeanNameAware 接口,Spring 传递 bean 的 ID 到 setBeanName 方法。

4. 如 果 Bean 实 现 了 BeanFactoryAware 接 口 , Spring 传 递 beanfactory 给setBeanFactory 方法。

5. 如 果 有 任 何 与 bean 相 关 联 的 BeanPostProcessors , Spring 会 在postProcesserBeforeInitialization()方法内调用它们。

6.如果bean实现IntializingBean了,调用它的afterPropertySet 方法,如果 bean 声明了初始化方法,调用此初始化方法。

7. 如 果 有 BeanPostProcessors和 bean关 联 , 这 些 bean 的postProcessAfterInitialization() 方法将被调用。

8.如果 bean 实现了 DisposableBean,它将调用 destroy()方法。

**19、Spring 中的设计模式**

1.代理模式—Spring 中两种代理方式,若目标对象实现了若干接口,spring 使用 JDK 的java.lang.reflect.Proxy 类代理,若目标对象没有实现任何接口,spring 使用 CGLIB 库生成目标对象的子类。

2.单例模式—在 spring 配置文件中定义的 bean 默认为单例模式。

3.模板方法模式—用来解决代码重复的问题。

4.前端控制器模式—Srping 提供了 DispatcherServlet 来对请求进行分发。

5.视图帮助(View Helper )—Spring 提供了一系列的 JSP 标签,高效宏来辅助将分散的代码整合在视图里。

6.依赖注入—贯穿于 BeanFactory/ApplicationContext 接口的核心理念。

7.工厂模式—在工厂模式中,我们在创建对象时不会对客户端暴露创建逻辑,并且是通过使用一个共同的接口来指向新创建的对象。Spring 中使用 BeanFactory 用来创建对象的实例。

**20、Spring 注解**

Spring 在 2.5 版本以后开始支持用注解的方式来配置依赖注入。可以用注解的方式来替代XML 方式的 bean 描述,可以将 bean 描述转移到组件类的内部,只需要在相关类上、方法上或者字段声明上使用注解即可。注解注入将会被容器在 XML 注入之前被处理,所以后者会覆盖掉前者对于同一个属性的处理结果。注解装配在 Spring 中是默认关闭的。所以需要在 Spring 文件中配置一下才能使用基于注解的装配模式。

**22、SpringMVC 执行流程**描述:

1. 用户向服务器发送请求,请求被SpringMVC前端控制器 DispatcherServlet 捕获;

2. DispatcherServlet 对请求 URL 进行解析,得到请求资源标识符(URI)。然后根据该 URI,调用 HandlerMapping获得该 Handler 配置的所有相关的对象(包括 Handler 对象以及Handler 对象对应的拦截器),最后以HandlerExecutionChain 对象的形式返回;

3. DispatcherServlet 根据获得的 Handler ,选择一个合的 HandlerAdapter;(附注:如果成功获得HandlerAdapter 后,此时将开始执行拦截器的 preHandler 方法)

4. 提取 Request 中的模型数据,填充 Handler 入参,开始执行 Handler(Controller)。 在填充 Handler的入参过程中,根据你的配置,SpringMVC 将帮你做一些额外的工作:

HttpMessageConveter: 将请求消息(如 Json、xml 等数据)转换成一个对象,将对象转换为指定的响应信息。

数据转换:对请求消息进行数据转换。

数据根式化:对请求消息进行数据格式化。 如将字符串转换成格式化数字或格式化日期等。

数据验证: 验证数据的有效性(长度、格式等),验证结果存储到 BindingResult 或 Error 中。

5. Handler 执行完成后,向 DispatcherServlet 返回一个 ModelAndView 对象；

6. 根据返回的 ModelAndView,选择一个适合的ViewResolver(必须是已经注册到 SpringMVC 容器中的ViewResolver)返回给 DispatcherServlet;

7. ViewResolver 结合 Model 和 View,来渲染视图;

8. 将渲染结果返回给客户端。

**23、Spring Boot 的理解**

spring-boot 是由 Pivotal 团队提供的全新框架,其设计目的是用来简化 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置,从而使开发人员不再需要定义样板化的配置。通过这种方式,Boot 致力于在蓬勃发展的快速应用开发领域(rapidapplication development)成为领导者。

Spring Boot 的目标不在于为已解决的问题域提供新的解决方案,而是为平台带来另一种开发体验,从而简化对这些已有技术的使用。对于已经熟悉 Spring 生态系统的开发人员来说,Boot 是一个很理想的选择,不过对于采用 Spring 技术的新人来说,Boot 提供一种更简洁的方式来使用这些技术。

**1、STRUTS 的应用(如 STRUTS 架构)**

Struts 是采用 Java Servlet/JavaServer Pages 技术，开发 Web 应用程序的开放源码的 framework。 采用 Struts 能开发出基于 MVC(Model-View-Controller)设计模式的应用构架。 Struts 有如下的主要功能：

一.包含一个 controller servlet，能将用户的请求发送到相应的 Action 对象。

二.JSP 自由 tag 库，并且在 controller servlet 中提供关联支持，帮助开发员创建交互式表单应用。

三.提供了一系列实用对象：XML 处理、通过 Java reflection APIs 自动处理 JavaBeans 属性、国际化的提示和消息。

**2、描述Mybatis**

MyBatis是持久层框架，它是支持JDBC的！简化了持久层开发！使用MyBatis时，只需要通过接口指定数据操作的抽象方法，然后配置与之关联的SQL语句，即可完成！

持久化存储：在程序运行过程中，数据都是在内存（RAM，即内存条）中的，内存中的数据不是永久存储的，例如程序可以对这些数据进行销毁，或者由于断电也会导致内存中所有数据丢失！而把数据存储到硬盘中的某个文件中，会使得这些数据永久的存储下来，常见做法是存储到数据库中，当然，也可以使用其它技术把数据存储到文本文件、XML文件等其它文件中！

Mybatis：是一个半自动映射的框架。这里所谓的”半自动"是相对于Hibernate全表映射而言，Mybatis需要手动匹配提供POJO、SQL和映射关系，而Hibernate只需要POJO和映射关系即可。与Hibernate相比，虽然使用Mybatis手动编写SQL要比使用Hibernate的工作量大，但Mybatis可以配置动态SQL并优化SQL，可以通过配置决定SQL的映射规则，它还支持存储过程等。对于一些复杂的和需要优化性能的项目来说，显然使用Mybatis更加合适。

Mybatis是当前主流的Java持久层框架之一，它与Hibernate一样，也是一种ORM框架。因其性能优异，且具有高度的灵活性、可优化性和易于维护等特点，所以受到了广大互联网企业的青睐，是目前大型互联网项目的首选框架。

**3、Struts工作流程**

客户端发出一个请求到servlet容器，请求经过一些列过滤被filterdispatcher调用，filterdispatch通过actionMapper去找相对应的action。Actionmapper找到对应的action返回给filterdispatch，dispatch把处理权交给actionproxy，actionproxy通过配置文件找到对应的action类，Actionproxy创建一个actionIinvocation的实例处理业务逻辑，一旦action处理完毕，actioninvocation负责根据stuts.xml的配置找到对应的返回结果。返回结果通常是jsp页面。

**4、应用中#与$有什么异同点**

相同点:#和$都可以应用在 mybatis 映射文件的 sql 语句中获取 DAO 方法中参数的值

不同点:mybatis 映射文件中的 sql 语句中使用#号的位置会生成预编译的 sql,运行效率高,使用$符号的位置会直接使用字符串替换,存在 sql 注入风险。

**5、MyBatis 应用动态 SQL 解决了什么问题**

1、简化不同条件下 sql 语句的拼接操作

2、更好的处理 sql 语句中的空格和逗号问题

3、更加灵活的实现数据的持久化操作。

**6、Mybatis 的理解**

MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手工设置参数以及抽取结果集。MyBatis 使用简单的 XML 或注解来配置和映射基本体,将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java 对象)映射成数据库中的记录。

Mybatis 的优点:

1、简单易学。mybatis 本身就很小且简单。没有任何第三方依赖,最简单安装只要两个 jar加配置几个 sql 映射文件,易于学习,易于使用,通过文档和源代码,可以比较完全的掌握它的设计思路和实现;

2、灵活。mybatis 不会对应用程序或者数据库的现有设计强加任何影响。 sql 写在 xml 里,便于统一管理和优化。通过 sql 基本上可以实现我们不使用数据访问框架可以实现的所有功能,或许更多;

3、解除 sql 与程序代码的耦合。通过提供 DAO 层,将业务逻辑和数据访问逻辑分离,使系统的设计更清晰,更易维护,更易单元测试。sql 和代码的分离,提高了可维护性;

4、提供映射标签,支持对象与数据库的 orm 字段关系映射;

5、提供对象关系映射标签,支持对象关系组建维护;

**1、对ajax的理解**

Ajax为异步请求，即局部刷新技术，在传统的页面中，用户需要点击按钮或者事件触发请求，到刷新页面，而异步技术为不需要点击即可触发事件，这样使得用户体验感增强，比如商城购物车的异步加载，当你点击商品时无需请求后台而直接动态修改参数。

**2、ajax是什么？有什么用？**

ajax不是新的编程语言，而是一种使用现有标准的新方法。通过在后台与服务器进行少量数据交换。ajax可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。而传统的网页（不使用ajax）如果需要更新内容，必须重载整个网页面。

ajax的优点：

1、最大的一点是页面无刷新，用户的体验非常好。

2、使用异步方式与服务器通信，具有更加迅速的响应能力。。

3、可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，ajax的原则是“按需取数据”，可以最大程度的减少冗余请求，和响应对服务器造成的负担。

4、基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

5、ajax可使因特网应用程序更小、更快，更友好。

ajax的缺点：

1、ajax不支持浏览器back按钮。

2、安全问题 AJAX暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。

4、破坏了程序的异常机制。

5、不容易调试。

**3、AJAX适用范围**

页面上首次显示的内容尽量不要用 Ajax，显示更多（或换页）的时候再用，切换页面的时候不用要 Ajax，不要因为网页的 header 或者 side 部分没有变化而用 Ajax。因为这是不同的页面，换句话说，是有完全不同的内容或者完全不同的页面逻辑。他们的 URL 就应该不同。消息提醒的部分一般用 Ajax。

**java基础**

### 1.我们能不能声明main()方法为非静态？

不能，main()方法必须声明为静态的，这样JVM才可以调用main()方法而无需实例化它的类。

如果从main()方法去掉“static”这个声明，虽然编译依然可以成功，但在运行时会导致程序失败。

### 2.不用main方法如何定义一个类？

不行，没有main方法我们不能运行Java类。

在Java 7之前，你可以通过使用静态初始化运行Java类。但是，从Java 7开始就行不通了。

### 3. String类为什么是final的。

主要是为了“效率”和“安全性”的缘故。若 String允许被继承, 由于它的高度被使用率,可能会降低程序的性能，所以String被定义成final;

### 4.使用final 关键字修饰一个变量时，是引用不能变，还是引用的对象不能变?

使用final 关键字修饰一个变量时，是指引用变量不能变，引用变量所指向的对象中的内容还是可以改变的。

### 5. string、stringbuilder、stringbuffer区别

a.可变与不可变

String类中使用字符数组保存字符串，如下就是，因为有“final”修饰符，所以可以知道string对象是不可变的。

private final char value[];

StringBuilder与StringBuffer都继承自AbstractStringBuilder类，在AbstractStringBuilder中也是使用字符数组保存字符串，如下就是，可知这两种对象都是可变的。

char[] value;

b.是否多线程安全

String中的对象是不可变的，也就可以理解为常量，显然线程安全。

AbstractStringBuilder是StringBuilder与StringBuffer的公共父类，定义了一些字符串的基本操作，如expandCapacity、append、insert、indexOf等公共方法。

StringBuffer对方法加了同步锁或者对调用的方法加了同步锁，所以是线程安全的

StringBuilder并没有对方法进行加同步锁，所以是非线程安全的

c.StringBuilder与StringBuffer共同点

StringBuilder与StringBuffer有公共父类AbstractStringBuilder(抽象类)。

区别在于StringBuffer支持并发操作，线性安全的，适合多线程中使用。StringBuilder不支持并发操作，线性不安全的，不适合多线程中使用。新引入的StringBuilder类不是线程安全的，但其在单线程中的性能比StringBuffer高

抽象类与接口的其中一个区别是：抽象类中可以定义一些子类的公共方法，子类只需要增加新的功能，不需要重复写已经存在的方法；而接口中只是对方法的申明和常量的定义。

StringBuilder、StringBuffer的方法都会调用AbstractStringBuilder中的公共方法，如super.append(...)。只是StringBuffer会在方法上加synchronized关键字，进行同步。

最后，如果程序不是多线程的，那么使用StringBuilder效率高于StringBuffer。

### 6. 抽象类和接口的区别

1).抽象类可以有构造方法，接口中不能有构造方法。

2).抽象类中可以有普通成员变量，接口中没有普通成员变量

3).抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

4). 抽象类中的抽象方法的访问类型可以是public，protected 和（默认类型,虽然

eclipse 下不报错，但应该也不行），但接口中的抽象方法只能是public 类型的，并且默认即为

public abstract 类型。

5). 抽象类中可以包含静态方法，接口中不能包含静态方法

6). 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，

但接口中定义的变量只能是public static final 类型，并且默认即为public static final 类型。

7). 一个类可以实现多个接口，但只能继承一个抽象类。

### 7. Java 中应该使用什么数据类型来代表价格？

如果不是特别关心内存和性能的话，使用BigDecimal，否则使用预定义精度的 double 类型。

### 8. 静态变量和实例变量的区别？

在语法定义上的区别：

静态变量前要加static 关键字，而实例变量前则不加。

在程序运行时的区别：

实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。

### 9、final, finally, finalize 的区别。

final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

内部类要访问局部变量，局部变量必须定义成final 类型，例如，一段代码……

finally 是异常处理语句结构的一部分，表示总是执行。

finalize 是Object 类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。JVM 不保证此方法总被调用

### 10. sleep() 和wait() 有什么区别?

sleep 就是正在执行的线程主动让出cpu，cpu 去执行其他线程，在sleep 指定的时间过后，cpu才会回到这个线程上继续往下执行，如果当前线程进入了同步锁，sleep 方法并不会释放锁，即使当前线程使用sleep 方法让出了cpu，但其他被同步锁挡住了的线程也无法得到执行。

wait 是指在一个已经进入了同步锁的线程内，让自己暂时让出同步锁，以便其他正在等待此锁的线程可以得到同步锁并运行，只有其他线程调用了notify 方法（notify 并不释放锁，只是告诉调用过wait 方法的线程可以去参与获得锁的竞争了，但不是马上得到锁，因为锁还在别人手里，别人还没释放。如果notify 方法后面的代码还有很多，需要这些代码执行完后才会释放锁，可以在notfiy 方法后增加一个等待和一些代码，看看效果），调用wait 方法的线程就会解除wait 状态和程序可以再次得到锁后继续向下运行。

11.

12.

**线程安全**

即便如此，你最关心的可能是线程安全。你现在应该学习到 Servlet 和 filter 被所有请求共享。那是 Java 的一个优点，使得多个不同线程（读取 HTTP 请求）可以使用同一个实例。否则为每个请求重新创建线程的开销实在过于昂贵。

但你应该也意识到永远不要将任何 request 或 session 域中的数据赋值给 servlet 或 filter 的实例变量。它将会被所有其他 session 中的所有请求共享。那是非线程安全的！

19

**20. 为什么接口要规定成员变量必须是public static final的呢?**

首先接口是一种高度抽象的"模版"，,而接口中的属性也就是’模版’的成员，就应当是所有实现"模版"的实现类的共有特性，所以它是public static的 ,是所有实现类共有的 .假如可以是非static的话，因一个类可以继承多个接口，出现重名的变量，如何区分呢？

其次,接口中如果可能定义非final的变量的话，而方法又都是abstract的，这就自相矛盾了，有可变成员变量但对应的方法却无法操作这些变量，虽然可以直接修改这些静态成员变量的值，但所有实现类对应的值都被修改了，这跟抽象类有何区别? 又接口是一种更高层面的抽象，是一种规范、功能定义的声明，所有可变的东西都应该归属到实现类中，这样接口才能起到标准化、规范化的作用。所以接口中的属性必然是final的。

最后，接口只是对事物的属性和行为更高层次的抽象 。对修改关闭，对扩展（不同的实现implements）开放，接口是对开闭原则（[Open-Closed Principle](http://www.cnblogs.com/wxx/archive/2005/07/07/187582.html" \t "_blank) ）的一种体现。

**25. 在System.out.println()里面,System, out, println分别是什么?**

System是系统提供的预定义的final类，out是一个PrintStream对象，println是out对象里面一个重载的方法。

**26. JAVA源文件中是否可以包括多个类，有什么限制？**

答：一个java源文件中可以包含多个类，但每个源文件中只允许一个public类，

如果源文件中没有public类，则源文件用什么名字都可以。

**27. 类有哪三个基本特性？各特性的优点？**

答：类具有封装性、继承性和多态性。

封装性：类的封装性为类的成员提供公有、缺省、保护和私有等多级访问权限，目的

是隐藏类中的私有变量和类中方法的实现细节。

继承性：类的继承提供从已存在的类创建新类的机智，继承使一个新类自动拥有被

继承类（父类）的全部可继承的成员。

多态性：类的多态性提供类中方法执行的多样性，多态性有两种表示形式：重载和

覆盖。

**28. 列出你常用的JDK包？**

答：java.io：这里面是所有输入输出有关的类，都是基于这个包下；比如文件操作等

java.net：这里面是与网络有关的类，比如URL,URLConnection等。

java.util：这个是系统辅助类，特别是集合类Collection，List，Map等。

Java.sql：这个是数据库操作的类，Connection，Statemenmt，ResultSet等。

**33. 请说出5种常见的runtime exception？**

答：NullPointerException：当操作一个空引用时会出现此错误。

NumnerFormatException：数据格式转换出现问题时出现此异常。

ClassCastException：强制类型转换类型不匹配时出现此异常。

ArrayIndexOutOfBoundsException：数组下标越界，当使用一个不存在

的数组下标时出现此异常。

**34. 数组有没有length()这个方法？String有没有length()这个方法？**

答：数组没有length()这个方法,但有这个属性；String有length()方法。

**41. 运行异常与一般异常有何异同？**

答：异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机

的通常操作中可能遇到异常，是一种常见运行错误，java编译器要求方法

必须声明抛出可能发生的非运行时异常，但是并不要去必须声明抛出未被

捕获的运行时异常。

**65. 转发与重定向的区别：**

-转发 ：一次请求，地址不变，一个request（数据可以共享），不能访问外部资源，效率较高

-重定向：二次请求，地址会变，两个request（数据不可以共享），能访问外部资源，效率较低

1. **项目过程中遇到了什么问题？什么原因产生的？如何解决？**

**67.说说四个访问控制修饰成员的访问权限是什么？**

**1、私有权限（private）**  
　　private可以修饰数据成员，构造方法，方法成员，不能修饰类（此处指外部类，不考虑内部类）。被private修饰的成员，只能在定义它们的类中使用，在其他类中不能调用。  
　　**2、默认权限（default）**  
　　类，数据成员，构造方法，方法成员，都能够使用默认权限，即不写任何关键字。默认权限即同包权限，同包权限的元素只能在定义它们的类中，以及同包的类中被调用。  
　　**3、受保护权限（protected）**  
　　protected可以修饰数据成员，构造方法，方法成员，不能修饰类（此处指外部类，不考虑内部类）。被protected修饰的成员，能在定义它们的类中，同包的类中被调用。如果有不同包的类想调用它们，那么这个类必须是定义它们的类的子类。  
　**4、公共权限（public）**　　public可以修饰类，数据成员，构造方法，方法成员。被public修饰的成员，可以在任何一个类中被调用，不管同包或不同包，是权限最大的一个修饰符。

**68、Overload(重载)和Override(重写)的区别？**

**69、多态的表现形式有哪些?**重写和重载。向上造型

##### **70、说出一些常用的类，包，接口，请各举5个**

**类:**

1.java.lang.Object  
2.java.lang.String  
3.java.lang.System  
4.java.io.file  
5.java.io.FileInputStream  
**包:**

1.java.lang包  
2.java.io包  
3.java.swt包  
4.java.util包  
5.java.sql包  
**接口:**

1.java.util.List<E>  
2.java.util.Map<E>  
3.java.util.Iterator<E>  
4.java.sql.CallableStatement  
5.java.lang.Comparable<T>

##### **71、error和exception有什么区别?**

Error（错误）表示系统级的错误和程序不必处理的异常，是java运行环境中的内部错误或者硬件问题。比如：内存资源不足等。对于这种错误，程序基本无能为力，除了退出运行外[别无选择](https://www.baidu.com/s?wd=%E5%88%AB%E6%97%A0%E9%80%89%E6%8B%A9&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmhmLmhPWnju-nynzrARd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWbzrjRznjns" \t "_blank)，它是由Java虚拟机抛出的。  
 Exception（违例）表示需要捕捉或者需要程序进行处理的异常，它处理的是因为程序设计的瑕疵而引起的问题或者在外的输入等引起的一般性问题，是程序必须处理的。  
Exception又分为运行时异常，受检查异常。  
 运行时异常，表示无法让程序恢复的异常，导致的原因通常是因为执行了错误的操作，建议终止程序，因此，编译器不检查这些异常。  
 受检查异常，是表示程序可以处理的异常，也即表示程序可以修复（由程序自己接受异常并且做出处理）， 所以称之为受检查异常。

**72、构造器能不能重写，重载**能被重载，不能被重写

##### **72、String s = new String("xyz");创建了几个String Object?**

String s=new String("xyz")究竟对象个数分为两种情况：  
1.如果String中，已经创建"xyz"，则不会继续创建，此时只创建了一个对象new String("xyz")；  
2.如果String常理池中，没有创建"xyz"，则会创建两个对象，一个对象的值是"xyz"，一个对象new String("xyz")。

##### **76、  队列和栈是什么，列出它们的区别？**

队列（Queue）是限定只能在表的一端进行插入和在另一端进行删除操作的线性表。

栈（Stack）是限定只能在表的一端进行插入和删除操作的线性表。

1.队列先进先出，栈先进后出。

2. 对插入和删除操作的"限定"。 栈是限定只能在表的一端进行插入和删除操作的线性表。队列是限定只能在表的一端进行插入和在另一端进行删除操作的线性表。

3.遍历数据速度不同。队列遍历数据的速度要快得多。

**类，对象，数组**

**1、内部类与外部类的调用**

内部类可以直接调用外部类包括private的成员变量，使用外部类引用的this.关键字调用即可，而外部类调用内部类需要建立内部类对象.

**2、Arraylist与linkedlist的区别**

都是实现list接口的列表，arraylist是基于数组的数据结构，linkedlist是基于链表的数据结构，当获取特定元素时，ArrayList效率比较快，它通过数组下标即可获取，而linkedlist则需要移动指针。当存储元素与删除元素时linkedlist效率较快，只需要将指针移动指定位置增加或者删除即可，而arraylist需要移动数据。

**3、面向对象的特征有哪些方面**

**1.抽象**：抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象，二是数据抽象。

**2.继承：**继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。

**3.封装：**封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。

**4. 多态：**多态是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

实现多态主要有以下三种方式：

1. 接口实现

2. 继承父类重写方法

3. 同一类中进行方法重载

**4、Java中的String，StringBuilder，StringBuffer三者的区别**

1.首先说运行速度，或者说是执行速度，在这方面运行速度快慢为：StringBuilder > StringBuffer > String。

2. 再来说线程安全，在线程安全上，StringBuilder是线程不安全的，而StringBuffer是线程安全的。

3. 总结一下

String：适用于少量的字符串操作的情况。

StringBuilder：适用于单线程下在字符缓冲区进行大量操作的情况。

StringBuffer：适用多线程下在字符缓冲区进行大量操作的情况。

**5、Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型?**

方法的重写Overriding和重载Overloading是Java多态性的不同表现。

重写Overriding是父类与子类之间多态性的一种表现，重载Overloading是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。

子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded的方法是可以改变返回值的类型。

**6、接口是否可继承接口? 抽象类是否可实现(implements)接口? 抽象类是否可继承实体类(concrete class)?**

接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数。

**7、数组有没有length()这个方法? String有没有length()这个方法？**

数组没有length()这个方法，有length的属性。String有有length()这个方法。

**8、JDK、JRE、JVM关系是什么？**

JDK（Java Development Kit）即为Java开发工具包，包含编写Java程序所必须的编译、运行等开发工具以及JRE。

JRE（Java Runtime Environment）即为Java运行环境，提供了运行Java应用程序所必须的软件环境，包含有Java虚拟机（JVM）和丰富的系统类库。系统类库即为java提前封装好的功能类，只需拿来直接使用即可，可以大大的提高开发效率。

JVM（Java Virtual Machines）即为Java虚拟机，提供了字节码文件（.class）的运行环境支持。

简单说，就是JDK包含JRE包含JVM。

**9、接口和抽象类的区别是什么？**

从设计层面来说，抽象是对类的抽象，是一种模板设计，接口是行为的抽象，是一种行为的规范。

Java提供和支持创建抽象类和接口。它们的实现有共同点，不同点在于：

接口中所有的方法隐含的都是抽象的。而抽象类则可以同时包含抽象和非抽象的方法。 类可以实现很多个接口，但是只能继承一个抽象类 类可以不实现抽象类和接口声明的所有方法，当然，在这种情况下，类也必须得声明成是抽象的。抽象类可以在不提供接口方法实现的情况下实现接口。Java接口中声明的变量默认都是final的。抽象类可以包含非final的变量。Java接口中的成员函数默认是public的。抽象类的成员函数可以是private，protected或者是public。接口是绝对抽象的，不可以被实例化。抽象类也不可以被实例化，但是，如果它包含main方法的话是可以被调用的。

**10、谈谈final, finally, finalize 的区别。**

final—修饰符（关键字）如果一个类被声明为final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 abstract的，又被声明为final的。将变量或方法声明为final，可以保证它们在使用中不被改变。被声明为final的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改。被声明为final的方法也同样只能使用，不能重载。

finally—再异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。

finalize—方法名。Java 技术允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 objecs 类中定义的，因此所有的类都继承了它。子类覆盖 finalize() 方法以整理系统资源或者执行其他清理工作。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的。

**11、“==”和“equals”的区别**

==是判断两端对象的地址是否是一个，如果地址是一个，那么内容也是相同的

equals方法判断的是内容，两个对象的地址可以不相同，但前提必须重写过equals方法，如果没有重写过的equals方法是Object的方法，此方法判断的是地址。

**12、排序都有哪几种方法？**

排序的方法有：插入排序（直接插入排序、希尔排序），交换排序（冒泡排序、快速排序），选择排序（直接选择排序、堆排序），归并排序，分配排序（箱排序、基数排序），快速排序的伪代码。

**13、队列和栈是什么，列出它们的区别？**

队列（Queue）是限定只能在表的一端进行插入和在另一端进行删除操作的线性表。

栈（Stack）是限定只能在表的一端进行插入和删除操作的线性表。

1.队列先进先出，栈先进后出。

2.对插入和删除操作的“限定”。栈是限定只能在表的一端进行插入和删除操作的线性表。队列是限定只能在表的一端进行插入和在另一端进行删除操作的线性表。

3.遍历数据速度不同。队列遍历数据的速度要快得多。

**17、 Stack(栈)和Heap(堆)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

**18、(匿名内部类) 是否可以 extends(继承)其它类，是否可以implements(实现)interface(接口)?**

匿名的内部类是没有名字的内部类。不能 extends(继承) 其它类，但一个内部类可以作为一个接口，由另一个内部类实现。

**22、sleep() 和 wait() 有什么区别?**

sleep()方法是使线程停止一段时间的方法。在 sleep 时间间隔期满后，线程不一定立即恢复执行。这是因为在那个时刻，其它线程可能正在运行而且没有被调度为放弃执行，除非(a)“醒来”的线程具有更高的优先级 (b)正在运行的线程因为其它原因而阻塞。

wait()是线程交互时，如果线程对一个同步对象 x 发出一个 wait()调用，该线程会暂停执行，被调对象进入等待状态，直到被唤醒或等待时间到。

**24、启动一个线程是用 run()还是 start()?**

启动一个线程是调用 start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 JVM 调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。

**25、构造器 Constructor 是否可被 override?**

构造器 Constructor 不能被继承，因此不能重写 Overriding，但可以被重载 Overloading。

**30、当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?**

是值传递。Java 编程语言只由值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

**31、swtich是否能作用在byte上，是否能作用在long上，是否能作用在String上?**

switch（expr1）中，expr1是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、short、char或byte。long不能作用于 swtich。(JDK1.8之后可以使用String)

**32、char 型变量中能不能存贮一个中文汉字?为什么?**

是能够定义成为一个中文的，因为 java 中以 unicode 编码，一个 char 占16个字节，所以放一个中文是没问题的

**34、静态变量和实例变量的区别?**

在语法定义上的区别:

静态变量前要加 static 关键字,而实例变量前则不加。

在程序运行时的区别:

实例变量属于某个对象的属性,必须创建了实例对象,其中的实例变量才会被分配空间,才能使用这个实例变量。

静态变量不属于某个实例对象,而是属于类,所以也称为类变量,只要程序加载了类的字节码,不用创建任何实例对象,静态变量就会被分配空间,静态变量就可以被使用了。总之,实例变量必须创建对象后才可以通过这个对象来使用,静态变量则可以直接使用类名来引用。

**35、什么是序列化,如何实现序列化?**

如果要将一个对象持久化到磁盘,或对象用于网络传输,此时就需要该对象是可序列化的。序列化是指将对象转换为字节序列的过程。

实现序列化:对象所属的类实现Serializable接口。

**36、是否可以从一个 static方法内部不创建对象而直接发出对非**

**static方法的调用?为什么不可以？**

非static方法是实例方法,必须通过实例对象才可以调用,而 static方法内部如果没有创建该实例对象,static方法内部也没有 this关键字,此时是不能直接发起对一个非 static方法的调用的。

**37、类的加载过程:**

当使用 java 命令运行 java 程序时,此时 JVM 启动,并去方法区下找 java 命令后面跟的类是否存在,如果不存在,则把类加载到方法区下在类加载到方法区时,会分为两部分:先加载非静态内容到方法区下的非静态区域内,再加载静态内容到方法区下的静态区域内。当非静态内容载完成之后,就会加载所有的静态内容到方法区下的静态区域内

1、先把所有的静态内容加载到静态区域下

2、所有静态内容加载完之后,对所有的静态成员变量进行默认初始化

3、当所有的静态成员变量默认初始化完成之后,再对所有的静态成员变量显式初始化

4、当所有的静态成员变量显式初始化完成之后,JVM 自动执行静态代码块(静态代码块在栈中执行)[如果有多个静态代码,执行的顺序是按照代码书写的先后顺序执行]

5、所有的静态代码块执行完成之后,此时类的加载完成

**38、对象的创建过程:**

当在 main 方法中创建对象时,JVM 会先去方法区下找有没有所创建对象的类存在,有就可以创建对象了,没有则把该类加载到方法区在创建类的对象时,首先会先去堆内存中开辟一块空间,开辟完后分配该空间(指定地址)，当空间分配完后,加载对象中所有的非静态成员变量到该空间下，所有的非静态成员变量加载完成之后,对所有的非静态成员进行默认初始化，所有的非静态成员默认初始化完成之后,调用相应的构造方法到栈中，在栈中执行构造函数时,先执行隐式,再执行构造方法中书写的代码。

1. 广盛易付

Mabits 如何实现批量查询

金蝶（问项目，问项目中的一些难点以及解决过程，数据库笔试比较大多）

1虚拟机的内存管理

2 类的实例化

3 数据库的几种应用模式，mysql的数据库类型

4 数据库的缓存，数据库知识，分页的sql语句，

5 spering 原理

6 mybits的好处

7 gc原理

8 jvm内存模式

9 线程

10 java特性

11 项目经历 java基础 数据库优化 学习主动性

12 http状态码

### comparator与comparable的比较

栗子（实现以名字排序，然后在按照其他条件排序）

@Data

@NoArgsConstructor

@AllArgsConstructor

/\*\*

\* @author hjb

\*/

public class Person implements Comparable {

private String name;

private Integer age;

private String mind;

@Override

public int compareTo(Object o) {

Person targetPerson = (Person)o;

int ret = this.name.compareTo(targetPerson.getName());

if(ret == 0){

return Integer.compare(this.age,targetPerson.getAge());

}

if(ret == 0){

return this.getMind().compareTo(targetPerson.getMind());

}

return ret;

}

}

### 类加载机制

前面是对类加载器的简单介绍，它的原理机制非常简单，就是下面几个步骤:

1.装载:查找和导入class文件;

2.连接:

      (1)检查:检查载入的class文件数据的正确性;

      (2)准备:为类的静态变量分配存储空间;

      (3)解析:将符号引用转换成直接引用(这一步是可选的)

3.初始化:初始化静态变量，静态代码块。

      这样的过程在程序调用类的静态成员的时候开始执行，所以静态方法main()才会成为一般程序的入口方法。类的构造器也会引发该动作。