

JAVA 面试、笔试题 (2016 版)

面试宝典

欲想成功，必须用功！

目录

一、HTML&CSS 部分.....	10
1、HTML 中定义表格的宽度用 80px 和 80%的区别是什么？	10
2、CSS 样式定义优先级顺序是？	10
3、div 和 span 的区别？	10
4、CSS 选择器包括？	11
5、用 css3 语法中，如何实现一个矩形框的圆角效果和 50%红色透明效果？，请写出关键脚本.....	11
6、Div 与 Table 的区别.....	11
7、行级标签转块级标签，块级标签转行级标签.....	12
二、Java 基础部分.....	12
1、java 中有哪些基本类型？	12
2、java 为什么能够跨平台运行？	12
3、String 是基本数据类型吗？我可不可以写个类继承于 String？	12
4、谈谈&和&&的区别？	13
5、Switch 语句里面的条件可不可以是 byte、long、String？使用时还应注意什么？	13
6、short s1=1;s1=s1+1;有什么错？ short s1 = 1;s1+=1 有什么错？	13
7、char 为什么能存贮一个汉字？	13
8、用最效率的办法算出 2 乘以 8 等于几？	13
9、final 关键字的用法？	14
10、静态变量和实例变量的区别？	14
11、面向对象的基本特征是什么？	14
12、作用域 public,private,protected,以及不写时的区别？	14
13、Overload 和 Override 的区别。	15
14、构造器可不可以被重载或重写？	15
15、Java 中有没有多继承？	15
16、抽象类和接口的区别？	15
17、java 中实现多态的机制是什么？	16
18、int 和 integer 的区别？	16

19、String 和 StringBuffer 的区别？StringBuffer 和 StringBuilder 区别？	16
20、String s=new String("xyz");创建了几个 String Object?.....	17
21、数组中有没有 length()方法，String 中有没有 length（）方法？	17
22、try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后?.....	17
23、final, finally, finalize 的区别。	17
24、‘==’ 和 equals 的区别？	18
25、error 和 exception 有什么区别？	18
26、heap 和 stack 有什么区别。	18
27、GC 是什么？为什么要有 GC?.....	19
28、什么是内部类？分为哪几种？	19
29、为什么需要内部类？	19
30、内部类可以引用它的包含类的成员吗？有没有什么限制？	19
31、Anonymous Inner Class (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)?.....	20
32、使用 java 命令查看 java 虚拟机版本.....	20
33、数字转字符有多少种方式，分别是什么.....	21
34、Java 创建对象有几种方式.....	21
35、写出验证 Email 的正则表达式.....	21
36、空指针异常是怎样的？怎么解决.....	21
37、Asp 与 Java 有什么不一样？	22
38、面向对象与面向过程的区别.....	22
39、说出十种常见的异常.....	23
40、一个静态方法，里面可不可以用 this 和 super 关键字.....	24
三、JavaScript/JQuery/Ajax 部分.....	24
1、用 js 和 jQuery 怎么进行表单验证.....	24
2、Java 和 Javascript 区别在哪?.....	24
3、列举 javascript 的 3 种主要数据类型，2 种复合数据类型和 2 种特殊数据类型。	25
4、谈谈你的 JS 的理解？	26
5、ajax 的优点？	27

6、简述一下 ajax 调试代码查找错误的方法？	27
7、简述 ajax 中 Js 脚本缓存问题该如何解决？	27
8、Ajax 应用和传统的 web 应用有何不同？	27
9、javascript 的作用？	28
10、为什么要有 jquery？	28
11、jQuery 选择器有多少种？	28
12、jquery 选择器有哪些优势？	28
13、你是如何使用 jquery 中的 ajax 的？	29
14、jquery 中的\$.get 和\$.post 请求区别？	29
15、jquery 中如何操作样式的？	29
16、如何设置和获取 HTML 和文本的值？	29
17、Jquery 能做些什么？	29
18、在 ajax 中 data 主要有哪几种？	30
19、jQuery 中 ajax 由几部分组成？	30
20、js 和 jQuery 获取 value 值得区别.....	30
四、jsp/servlet 部分.....	31
1、Tomcat 的优化经验.....	31
2、Tomcat 根目录下有哪些文件.....	31
3、什么是 TOMCAT,怎样启动停止，配置文件，日志文件的存储。.....	31
4、解释一下什么是 servlet;什么是 servlet 容器;.....	32
5、说一说 Servlet 的生命周期，执行过程?.....	32
6、实例化 servlet 有几种方式.....	32
7、HTTP 请求的 GET 与 POST 方式的区别.....	33
8、请写一个 Servlet 的基本架构。	33
9、forward 和 redirect 的区别?.....	34
10、servlet 中怎么定义 forward 和 redirect.....	34
11、过滤器有哪些作用？	34
12、JSP 的常用指令？	34
13、JSP 和 Servlet 中的请求转发分别如何实现？	35
14、JSP 乱码如何解决？	35

15、session 和 application 的区别?	35
16、jsp 有哪些内置对象?作用分别是什么?.....	36
17、Jsp 有哪些动作?作用分别是什么?.....	36
18、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别?	36
19、JSP 和 Servlet 有哪些相同点和不同点, 他们之间的联系是什么?	37
20、页面传递对象的方法?	37
21、Cookied 和 session 区别?	37
22、Filter 的生命周期与执行过程.....	38
23、Tomcat 系统内存怎么配置.....	38
24、JSTL 标签库包含哪些?	39
五、数据库部分.....	39
1、触发器的作用?	39
2、什么是存储过程? 用什么来调用?	39
3、存储过程的优缺点?	39
4、存储过程与函数的区别.....	40
5、索引的作用? 和它的优点缺点是什么?	40
6、什么样的字段适合建索引.....	41
7、索引类型有哪些?	41
8、什么是事务? 什么是锁?	41
9、什么叫视图? 游标是什么?	42
10、视图的优缺点.....	42
11、列举几种表连接方式,有什么区别?	42
12、主键和外键的区别?	43
13、在数据库中查询语句速度很慢, 如何优化?	43
14、数据库三范式是什么?.....	43
15、union 和 union all 有什么不同?.....	44
16、char、varchar2、varchar 有什么区别?	44
17、Oracle 和 Mysql 的区别?	45
18、Oracle 语句有多少类型.....	45
19、oracle 分页语句.....	46

20、从数据库中随机取 50 条.....	46
21、order by 与 group by 的区别.....	46
22、commit 在哪里会运用.....	47
23、行转列、列换行怎么转.....	47
24、什么是 PL/SQL?	49
25、序列的作用.....	49
26、表和视图的关系.....	49
27、oracle 基本数据类型.....	49
28、drop、truncate、 delete 区别.....	50
29、oracle 获取系统时间.....	50
30、oracle 怎么去除去重.....	50
31、合并查询有哪些?	50
32、SQL 语句执行顺序.....	50
33、null 的含义.....	51
34、mysql 分页.....	51
35、MySQL、SqlServer、oracle 写出字符存储、字符串转时间.....	52
36、update 语句可以修改结果集中的数据吗?	52
六、Java 高级部分.....	52
1、java 中有几种方法可以实现一个线程? 用什么关键字修饰同步方法? stop()和 suspend()方法为何不推荐使用?	52
2、sleep() 和 wait() 有什么区别?.....	53
3、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?.....	53
4、线程的基本概念.....	53
5、什么是多线程.....	54
6、程序、进程、线程之间的关系.....	54
7、创建线程有几种方式, 分别是什么?	54
8、线程的生命周期.....	55
9、线程 currentThread()与 interrupt()方法的使用.....	55
10、线程状态.....	56
11、什么是 java 序列化, 如何实现 java 序列化?	56

12、编写一个程序，将 d:\java 目录下的所有.java 文件复制到 d:\jad 目录下，并将原来文件的扩展名从.java 改为.jad。	56
13、java 中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？	59
14、字节流与字符流的区别.....	59
15、怎么判断指定路径是否为目录.....	62
16、怎么获取指定路径下的全部文件.....	62
17、Java 怎么读取文件和写入文件.....	62
18、java 怎么复制文件.....	64
19、用 JDBC 如何调用存储过程.....	66
20、JDBC 中的 PreparedStatement 相比 Statement 的好处.....	68
21、写一个用 jdbc 连接实例。	68
22、ArrayList 和 Vector 的区别？	70
23、List、Set 和 Map 的区别？	71
24、Collection 和 Collections 的区别。	71
25、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？.....	71
26、HashMap 与 Hashtable 的区别.....	71
27、Java 中有多少种数据结构，分别是什么？	72
28、Arraylist 和 linklist 的区别.....	72
29、List 遍历方式有多少种.....	73
30、Map 怎么遍历.....	73
31、怎么获取 Map 所有的 key，所有的 value.....	73
32、获取 Class 的实例有几种方式.....	73
33、怎么获取类中所有的方法，所有属性.....	73
34、JDBC 常用接口有哪些？	74
35、Statement 中 execute、executeUpdate、executeQuery 这三者的区别.....	74
36、jdbc 中怎么做批量处理的？	77
37、什么是 json.....	79
38、json 与 xml 的区别.....	79
39、XML 和 HTML 的区别？	80

40、XML 文档定义有几种形式？它们之间有何本质区别？	80
七、框架部分.....	81
1、谈谈你对 Struts2 的理解。	81
2、谈谈你对 Hibernate 的理解。	82
3、你对 Spring 的理解。	82
4、Struts2 优缺点.....	83
5、说说 struts1 与 struts2 的区别。	84
6、struts2 的核心组件有哪些？	85
7、Struts2 的执行过程.....	85
8、为什么要使用 struts2？	86
9、openSession 和 getCurrentSession.....	86
10、拦截器的作用？拦截器和过滤器的区别？	87
11、struts.xml 中 result 的 type 有哪些类型？	87
12、一般情况下，关系数据模型与对象模型之间有哪些匹配关系?.....	87
13、hibernate 数据的三个状态.....	87
14、Hibernate 中 load 和 get 的区别？	88
15、Hibernate 的工作原理?.....	88
16、hibernate 优缺点？	88
17、Hibernate 是如何延迟加载的？	89
18、如果优化 Hibernate？	90
19、什么是 ORM？	90
20、Hibernate 的主键生成策略？	90
21、Hibernate 的级联操作.....	91
22、Hibernate 有哪 5 个核心接口？	91
23、什么是重量级？什么是轻量级？	91
24、谈谈 Spring 的 IOC 和 DI.....	92
25、什么是 AOP？	93
26、Spring 的通知类型有哪些？	93
27、什么是 MVC？	94
28、hibernate 查询方式有多少种？	94

29、spring 中 Bean 的 scope.....	94
30、SSH 对应 MVC 的哪些层.....	95
31、spring 注入方式有几种.....	95
32、spring mvc 的工作原理.....	95
33、mybatis 的工作原理.....	96
34、spring mvc 与 Struts2 的区别.....	96
35、mybatis 与 hibernate 的区别.....	97
36、spring 和 springmvc 的区别.....	99
37、java 中的悲观锁和乐观锁.....	99
八、设计模式部分.....	100
请写出你所知道的设计模式？	100
九、算法部分.....	101
1、说明生活中遇到的二叉树，用 java 实现二叉树.....	101
2、第 1 个人 10，第 2 个比第 1 个人大 2 岁，依次递推，请用递归方式计算出第 8 个人多大？	109
3、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。	110
4、金额转换，阿拉伯数字的金额转换成中国传统的形式如：（¥1011）—>（一千零一拾一元整）输出。	111
5、从类似如下的文本文件中读取所有的姓名，并打印出重复的姓名和重复的次数，并按重复次数排序：	113
6、写一个 Singleton 出来。	118
7、古典问题：有一对兔子，从出生后第 3 个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少？	121
8、简单的说个递归.....	122
9、什么是平衡二叉树.....	122
10、怎么判断二叉树是否有环.....	122

一、HTML&CSS 部分

1、HTML 中定义表格的宽度用 80px 和 80% 的区别是什么？

PX 标识像素，%标识整个页面的宽度百分比

2、CSS 样式定义优先级顺序是？

内联样式最高优先权，然后是内部样式，然后才是外部样式

3、div 和 span 的区别？

DIV 和 SPAN 元素最大的特点是默认都没有对元素内的对象进行任何格式化渲染。主要用于应用样式表（共同点）。

两者最明显的区别在于 DIV 是块元素，而 SPAN 是行内元素(也译作内嵌元素)。

详解：1.所谓块元素，是以另起一行开始渲染的元素，行内元素则不需另起一行，测试一下下面的代码你会有更形象的理解：

测试紧跟前面的"测试"显示

这里会另起一行显示

4、CSS 选择器包括？

- 1) 类别选择器 用 “.” 来标识
- 2) 标签选择器 用 HTML 标签来标识
- 3) ID 选择器 用 “#” 号来标识
- 4) 通配符选择器 用 “*” 号来标识

5、用 css3 语法中，如何实现一个矩形框的圆角效果和 50% 红色透明效果？，请写出关键脚本

```
<style>
    div{
        width:200px;
        height:200px;
        border-radius: 30px;
        opacity: 0.5;/* 火狐 */
        /* filter:alpha(opacity=50); IE */
        background-color:red;
    }
</style>
```

6、Div 与 Table 的区别

- 1) div 大大缩减页面代码，提高页面浏览速度，table 代码量繁多，页面浏览效率慢。
- 2) div 结构清晰，可读性非常强，也容易被搜索引擎搜索到，优化了搜索引擎，Table 结构复杂，可读性差。
- 3) div 缩短改版时间。只要简单的修改几个 CSS 文件就可以改变很多页面。Table 要想改变的话，需要一个页面一个页面的去修改。
- 4) div 表现和内容相分离，非常符合 w3c 标准。

5) table 制作效率高于 div

6) table 浏览器兼容性高于 div, 我们常用的 IE6.0, IE7.0 火狐 Firefox 浏览器对 div css 设置上非常挑剔。

7、行级标签转块级标签，块级标签转行级标签

行级转块级: display:block

块级转行级: float:left

二、Java 基础部分

1、java 中有哪些基本类型？

byte、short、int、long、float、double、char、boolean

2、java 为什么能够跨平台运行？

因为 Java 程序编译之后的代码不是能被硬件系统直接运行的代码，而是一种“中间码”——字节码。然后不同的硬件平台上安装有不同的 Java 虚拟机(JVM)，由 JVM 来把字节码再“翻译”成所对应的硬件平台能够执行的代码。因此对于 Java 编程者来说，不需要考虑硬件平台是什么。所以 Java 可以跨平台。

3、String 是基本数据类型吗？我可不可以写个类继承于 String？

不是，Strng 是引用类型；String 是 final 的类，是不可以被继承的。

4、谈谈&和&&的区别？

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

&&还具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式。

&还可以用作位运算符，当&操作符两边的表达式不是 boolean 类型时，&表示按位与操作。

5、Switch 语句里面的条件可不可以是 byte、long、String？ 使用时候还应注意什么？

switch 里面的条件必须是能隐式的转化成为 Int 的故 long 和 String 不行，byte 可以；使用 Switch 时候还应注意它的穿透，即每个 case 后要跟 break；

6、short s1=1;s1=s1+1;有什么错？ short s1 = 1;s1+=1 有什么错？

对于 short s1 = 1; s1 = s1 + 1; 由于 s1+1 运算时会自动提升表达式的类型，所以结果是 int 型，再赋值给 short 类型 s1 时，编译器将报告需要强制转换类型的错误。

对于 short s1 = 1; s1 += 1; 由于 += 是 java 语言规定的运算符，java 编译器会对它进行特殊处理，因此可以正确编译。

7、char 为什么能存贮一个汉字？

char 型变量是用来存储 Unicode 编码的字符的，unicode 编码字符集中包含了全世界所有的字体。

8、用最效率的办法算出 2 乘以 8 等于几？

2<<3 位移运算是最底层的运算，他直接操作的是二进制，故效率最快。

9、final 关键字的用法？

final 用于修饰类、属性、方法

final 修饰的类，不能被继承

final 修饰的属性，是常量，值不可以改变

final 修饰的方法，不可以被重写

10、静态变量和实例变量的区别？

静态变量也称为类变量，归全类共有，它不依赖于某个对象，可通过类名直接访问；而实例变量必须依存于某一实例，只能通过对象才能访问到它。

11、面向对象的基本特征是什么？

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：子类拥有父类一切非私有的属性和方法。

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)多态性：同一种事物的不同种表现形式。

12、作用域 public,private,protected,以及不写时的区别？

作用域	当前类	同包	子孙类	其他
public	√	√	√	√
protected	√	√	√	×

default	√	√	×	×
private	√	×	×	×

不写时默认为 default。

13、Overload 和 Override 的区别。

（Overload）重载：发生在同一个类之中，方法名相同、参数列表不同，与返回值无关、与 final 无关、与修饰符无关、与异常无关。

（Override）重写：发生在子类 and 父类之间，方法名相同、参数列表相同、返回值相同、不能是 final 的方法、重写的方法不能有比父类方法更为严格的修饰符权限、重写的方法所抛出的异常不能比父类的更大。

如果父类私有的方法，子类拥有方法签名相同的方法，子类不属于重写父类的方法，该方法属于子类的新方法。

14、构造器可不可以被重载或重写？

构造器不能被继承，故不能被重写、但可以被重载。

15、Java 中有没有多继承？

java 中没有多继承，但是可以多实现，即一个类实现多个接口。

虽然没有多继承，但是 java 中接口可以近似的实现多继承，那就是接口；接口和接口之间可以进行多继承。

16、抽象类和接口的区别？

- 1) 抽象类继承与 object 接口不继承 object.
- 2) 抽象类有构造器，接口中没有构造器。
- 3) 抽象类中可以有普通成员变量和常量，接口中只能有常量，而且只能是 public static final 不写默认。

- 4) 抽象类中可以有抽象方法，也可以由普通的方法，接口中只能有抽象的方法而且修饰符只能是 `public abstract` 不写默认。
- 5) 抽象类中可以有 `final` 的方法,接口中不能有 `final` 的方法。
- 6) 抽象类只能是单继承，多实现，接口是可以多继承其他接口，但是不能实现接口，和不能继承其他类。
- 7) 抽象类中可以有静态的方法，接口中不可以。

17、java 中实现多态的机制是什么？

重写、重载、父类的声明指向子类的对象。

18、int 和 integer 的区别？

`int` 是 java 的基本数据类型,`integer` 是 1.4 版本后提供的基本类型包装类，当两者作为成员变量时，初始值分别为;`int` 是 0; `integer` 是 `null`;其中 `integer` 提供了一些对整数操作的方法，还定义了 `integer` 型数值的最值，其他基本类型也有对应的包装类，基本类型包装类的出现，使得 java 完全面向对象。

19、String 和 StringBuffer 的区别？StringBuffer 和 StringBuilder 区别？

`String` 是不可变的，对 `String` 类的任何改变都会返回一个新的 `String` 对象。

`StringBuffer` 是可变的，对 `StringBuffer` 中的内容修改都是当前这个对象。

`String` 重写了 `equals` 方法和 `hashCode` 方法，`StringBuffer` 没有重写 `equals` 方法。`String` 是 `final` 的类。`StringBuffer` 不是。

`String` 创建的字符串是在常量池中，创建的变量初始化一次，如果再对该字符串改变会产生新的字符串地址值，`StringBuffer` 是在堆中创建对象，当对字符串改变时不会产生新的字符串地址值，如果对字符串进行频繁修改的话建议使用 `StringBuffer`，以节省内存。

`StringBuffer` 和 `StringBuilder`，`StringBuffer` 是线程安全的，`StringBulider` 是线程不安全的。当

不考虑并发问题时候，请使用 `StringBulider`。

20、`String s=new String(“xyz”);`创建了几个 `String Object`?

两个对象，一个是"xyz",一个是指向"xyz"的引用对象 s。

21、数组中有没有 `length()` 方法，`String` 中有没有 `length`（）方法?

数组中没有 `length()` 方法,但是有 `length` 属性，`String` 中有 `length()` 方法

22、`try {}`里有一个 `return` 语句，那么紧跟在这个 `try` 后的 `finally {}`里的 `code` 会不会被执行，什么时候被执行，在 `return` 前还是后?

这道题很有争议，我是通过 `debug` 模式分为两种情况进行测试的.

1) `finally` 中没有 `return` 时候:

会先执行 `try` 里面的，`return` 会执行但是没有真正的 `return` 此时去执行了 `finally` 里面的，然后再返回来执行 `return`.

2) `finally` 中有 `return` 时候(其实这种情况不符合编程规范，会报黄线警告):

会先执行 `try` 里面的，`return` 会执行但是没有真正的 `return` 此时去执行了 `finally` 里面的，然后执行 `finally` 里面的 `return`，直接返回。

23、`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。

`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

24、‘==’ 和 `equals` 的区别？

‘==’ 比较的是两个变量的内容和在内存中的地址值是否全部相等，如果要比较两个基本数据类型那必须用‘==’

`equals` 如果没有重写，则和‘==’的意义一样，如果重写了，则会按照重写的内容进行比较，`javaBean` 规定当重写 `equals` 时候必须重写 `hashCode`，如果不重写会出现对象相同但是 `hashCode` 不同，这样会出现问题，eg:`HashSet` 存储元素时候是按照 `hashCode`，如果重写 `equals` 不重写 `hashCode` 会导致同一个对象，存储了两次。

25、`error` 和 `exception` 有什么区别？

`error` 表示恢复不是不可能但是很困难的情况下的一种严重问题，例如程序书写错误，虚拟机错误等，`exception` 是一种设计和实现问题，如果程序运行正常，从不会发生的情况。`error` 是可以避免的，`exception` 是不可避免的。

26、`heap` 和 `stack` 有什么区别。

java 的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。

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

27、GC 是什么？为什么要有 GC？

GC 是垃圾收集的意思（Garbage Collection），内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java 提供的 GC 功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java 语言没有提供释放已分配内存的显示操作方法。

28、什么是内部类？分为哪几种？

内部类是指在一个外部类的内部再定义一个类。内部类作为外部类的一个成员，并且依附于外部类而存在的。内部类可为静态，可用 `protected` 和 `private` 修饰（而外部类只能使用 `public` 和缺省的包访问权限）。

内部类主要有以下几类：成员内部类、局部内部类、静态内部类、匿名内部类。

29、为什么需要内部类？

典型的情况是，内部类继承自某个类或实现某个接口，内部类的代码操作创建其的外围类的对象。所以你可以认为内部类提供了某种进入其外围类的窗口。

使用内部类最吸引人的原因是：每个内部类都能独立地继承自一个（接口的）实现，所以无论外围类是否已经继承了某个（接口的）实现，对于内部类都没有影响。如果没有内部类提供的可以继承多个具体的或抽象的类的能力，一些设计与编程问题就很难解决。从这个角度看，内部类使得多重继承的解决方案变得完整。接口解决了部分问题，而内部类有效地实现了“多重继承”。

30、内部类可以引用它的包含类的成员吗？有没有什么限制？

完全可以。如果不是静态内部类，那没有什么限制！

如果你把静态嵌套类当作内部类的一种特例，那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员，例如，下面的代码：

```
class Outer
{
    static int x;

    static class Inner
    {
        void test()
        {
            syso(x);
        }
    }
}
```

答题时，也要能察言观色，揣摩提问者的心思，显然人家希望你说的是静态内部类不能访问外部类的成员，但你一上来就顶牛，这不好，要先顺着人家，让人家满意，然后再说特殊情况，让人家吃惊。

31、Anonymous Inner Class (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)?

可以继承其他类或实现其他接口。不仅是**可以**，而是**必须**!

32、使用 java 命令查看 java 虚拟机版本

```
java -version
```

33、数字转字符有多少种方式，分别是什么

- 1) String.valueOf()
- 2) "" + 数字
- 3) Integer.toString()

34、Java 创建对象有几种方式

- 1) new 关键字
- 2) 反射
- 3) 克隆
- 4) 反序列化

35、写出验证 Email 的正则表达式

```
public static boolean checkEmail(String email) {  
    if(email == null || "".equals(email)) {  
        return false;  
    }  
    boolean temp = email.matches("\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*");  
    return temp;  
}
```

36、空指针异常是怎样的？怎么解决

空指针异常会抛出： java.lang.NullPointerException

解决方法：判断使用对象是否为 null

```
if(obj != null) {  
  
    }  
}
```

37、Asp 与 Java 有什么不一样？

1、ASP 是微软公司从自有技术发展出来的，一般仅能用于 Windows 平台，并总是作为微软 Internet Information Server 的强有力的基本特性出现，其开放性较差。

Java 是 Sun 公司推出具有跨平台性的开发语言，能在 Windows 平台于 Linux 平台运行，开发简便、安全性极高，java 所有源码都是开放的。

2、用 java 开发业务较复杂的程序，相对来说要轻松一些，java 有丰富的开源框架和套现设计模式。

ASP 只有 .net 框架，与 java 开源框架如：spring、Struts、hibernate、mybatis 等比较会差一些。

3、java 使用 oracle 数据库，都能跨平台，一般高端机都运行在 Linux 或 Unix 下，注定 oracle 是大型数据库。

ASP 使用 SQL Server 数据库，运行在 Windows 平台下，而 Windows 能够安装的硬件是有限的，SQL Server 数据库只能是中型数据库。

4、java 运行需要 JVM 翻译，所以 java 执行效率没有 .net 高

综述，.net 适合做网站，中型的！java 适合做后台，企业系统，大型网站！

38、面向对象与面向过程的区别

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描述某个事物在整个解决问题的步骤中的行为。

例如五子棋，面向过程的设计思路就是首先分析问题的步骤：1、开始游戏，2、黑子先走，3、绘制画面，4、判断输赢，5、轮到白子，6、绘制画面，7、判断输赢，8、返回步骤2，9、输出最后结果。把上面每个步骤用分别的函数来实现，问题就解决了。

而面向对象的设计则是从另外的思路来解决问题。整个五子棋可以分为 1、黑白双方，这两方的行为是一模一样的，2、棋盘系统，负责绘制画面，3、规则系统，负责判定诸如犯规、输赢等。第一类对象（玩家对象）负责接受用户输入，并告知第二类对象（棋盘对象）棋子布局的变化，棋盘对象接收到了棋子的 i 变化就要负责在屏幕上面显示出这种变化，同时利用第三类对象（规则系统）来对棋局进行判定。

可以明显地看出，面向对象是以功能来划分问题，而不是步骤。同样是绘制棋局，这样的行为在面向过程的设计中分散在了总多步骤中，很可能出现不同的绘制版本，因为通常设计人员会考虑到实际情况进行各种各样的简化。而面向对象的设计中，绘图只可能在棋盘对象中出现，从而保证了绘图的一致。

功能上的统一保证了面向对象设计的可扩展性。比如我要加入悔棋的功能，如果要改动面向过程的设计，那么从输入到判断到显示这一连串的步骤都要改动，甚至步骤之间的循序都要进行大规模调整。如果是面向对象的话，只用改动棋盘对象就行了，棋盘系统保存了黑白双方的棋谱，简单回溯就可以了，而显示和规则判断则不用顾及，同时整个对对象功能的调用顺序都没有变化，改动只是局部的。

39、说出十种常见的异常

<code>java.lang.NullPointerException</code>	空指针异常
<code>java.lang.ClassNotFoundException</code>	类找不到异常
<code>java.lang.ArithmeticException</code>	数学运算异常
<code>java.lang.ArrayIndexOutOfBoundsException</code>	数组下标越界
<code>java.lang.ClassCastException</code>	类型转换异常

java.lang.NumberFormatException	数字转换异常
java.sql.SQLException	操作数据库异常
java.io.IOException	输入输出流异常
java.io.FileNotFoundException	文件找不到异常
java.lang.InstantiationError	实例化异常

40、一个静态方法，里面可不可以用 **this** 和 **super** 关键字

不能，因为 **this** 代表的是调用这个方法的对象的引用，**super** 代表当前父类对象的引用，而静态方法是属于类的，不属于对象，静态优先于对象，静态方法成功加载后，对象还不一定存在。

三、JavaScript/JQuery/Ajax 部分

1、用 js 和 jQuery 怎么进行表单验证

Js 验证表单，可以通过 `document` 对象获取表单中输入框的内容进行验证。

jQuery 验证表单：可以通过 jQuery 选择器获取表单中输入框的内容进行验证，

现在基于 jQuery 第三方验证框架非常多，也可以通过 jQuery 第三方验证框架进行验证。

2、Java 和 Javascript 区别在哪？

Java 与 JavaScript 是目前比较流行的两种语言，单从表面上看，两者名称很相似，于是许多初学者容易将二者混淆，或者直接归为一类，其实不然，虽然两者有着紧密的联系，但确是两个完全不同的语言。接下来，笔者仅就她们的几个主要差别归纳起来。

一．开发厂商

众所周知，Java 是 SUN 公司推出的程序设计语言，特别适合于 Internet 应用程序开发，其前身是 Oak 语言，而 JavaScript 则是 NetScape 公司的产品，是为了扩展 NetScape Navigator

功能而开发的一种可嵌入 Web 页面中的解释性语言，其前身是 Live Script。由于 Java 的普及，NetScape 和 SUN 两家公司签订合同后才将其命名为 JavaScript。

二. 面向对象与基于对象

Java 是一种真正的纯面向对象编程语言，在 Java 中，一切都是对象；JavaScript 是一种脚本语言，由于她本身提供了非常丰富的内部对象供程序员使用，因而它是基于对象的语言。

三. 开发和运行环境的不同

若希望利用 Java 编写程序并使之运行，必须事先在系统内安装相应版本的 JDK 和 JVM，保证代码能够得到编译和运行的环境；而编写 JavaScript 则相对简单，只需使用某种 HTML 文档编辑器甚至某种字符编辑器（如 Notepad）即可，然后打开浏览器即可运行。

四. 变量的区别

Java 中使用变量在编译之前必须声明其数据类型，因而她采用的是强类型变量；JavaScript 则不用在变量前声明类型，而是由解释器在运行时自动检查，所以她是弱类型变量。

五. 标签的不同

利用 Java 写出的 Applet 小程序，在 HTML 中用<applet>……</applet>来标识；JavaScript 程序在 HTML 中运行，其代码在<Script>……</Script>标签内。

六. 解释与编译

Java 源代码在执行前被编译，因而在网络应用中，必须要求客户端安装有解释平台，也就意味着 Java 应用不与 HTML 文档集成（Applet 小程序例外）；JavaScript 是一种解释性语言，其代码在发往客户端之前不需编译，而是将其嵌入到 HTML 文档中，一起发送给客户端，由浏览器解释执行。

另外，JavaScript 仅是一种解释性语言，并没有复杂的语法和规则，更不支持如 Java 里的继承这样的性质，因此也比 Java 更加容易学习。

3、列举 javascript 的 3 种主要数据类型，2 种复合数据类型和 2 种特殊数据类型。

主要数据类型：string, boolean, number

复合数据类型：function, object

4、谈谈你的 JS 的理解？

JavaScript 是一种脚本语言，它采用小程序段的方式实现编程。像其它脚本语言一样,JavaScript 同样已是一种解释性语言,它提供了一个易的开发过程。它的基本结构形式与 C、C++、VB、Delphi 十分类似。但它不像这些语言一样，需要先编译，而是在程序运行过程中被逐行地解释。它与 HTML 标识结合在一起，从而方便用户的使用操作。

2) 基于对象的语言。

JavaScript 是一种基于对象的语言，同时以可以看作一种面向对象的。这意味着它能运用自己已经创建的对象。因此，许多功能可以来自于脚本环境中对象的方法与脚本的相互作用。

3) 简单性

JavaScript 的简单性主要体现在：首先它是一种基于 Java 基本语句和控制流之上的简单而紧凑的设计，从而对于学习 Java 是一种非常好的过渡。其次它的变量类型是采用弱类型，并未使用严格的数据类型。

4) 安全性

JavaScript 是一种安全性语言，它不允许访问本地的硬盘，并不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互。从而有效地防止数据的丢失。

5) 动态性的

JavaScript 是动态的，它可以直接对用户或客户输入做出响应，无须经过 Web 服务程序。它对用户的反映响应，是采用以事件驱动的方式进行的。所谓事件驱动，就是指在主页(Home Page)中执行了某种操作所产生的动作，就称为“事件”(Event)。比如按下鼠标、移动窗口、选择菜单等都可以视为事件。当事件发生后，可能会引起相应的事件响应。

6) 跨平台性

JavaScript 是依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可正确执行。从而实现了“编写一次,走遍天下”的梦想。实际上 JavaScript 最杰出之处在于可以用很小的程序做大量的事。无须有高性能的电脑，软件仅需一个字处理软件及一浏览器，无须 WEB 服务器通道，通过自己的电脑即可完成所有的事情。

5、ajax 的优点？

使用 ajax 的最大优点，就是能在不更新整个页面的前提下维护数据。这使得 web 应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

6、简述一下 ajax 调试代码查找错误的方法？

这是 js 调试代码存在已久的问题，简单的我们可以使用浏览器提供的错误提示框，还有可以使用 DW CS4 提供的纠正错误，或者通过专业的插件，如 firebug 等

7、简述 ajax 中 Js 脚本缓存问题该如何解决？

这个问题是大家遇到最常见的问题之一，因为修改了 js 内容调试的时候并不能显示新写代码的结果，是因为 Js 为了加速页面执行，当前页面会使用缓存保持当前调用的相同的连接，为了开发时调试方便可以在连接地址后面增加一个随机函数。

8、Ajax 应用和传统的 web 应用有何不同？

在传统的 javascript 中，如果想得到服务器端数据库或文件上的信息，或者发送客户端信息到服务器，需要建立一个 HTML form 然后 Post 或者 get 提交数据到服务端。用户需要点击 submit 来发送或者接受数据信息，然后等待服务器响应请求，页面重写加载，因为服务器每次都要返回一个新的页面，所以传统的 web 应用有可能会很慢而且用户交互不友好。

使用 ajax 就可以使 javascript 通过 XMLHttpRequest 对象直接与服务器进行交互。通过 XMLHttpRequest，一个 web 页面可以发送一个请求到 web 服务器并且接受 web 服务器返回的信息(不需要加载任何界面)，展示给用户的还是同一个页面，用户感觉不到页面刷新，也看不到 Javascript 后台进行的发送请求和接受的响应。

9、javascript 的作用？

表单验证、网页特效、网页游戏

10、为什么要有 jquery？

1) jQuery 是 JavaScript 的轻量级框架，对 JavaScript 进行了很好的封装，很多复杂的 JavaScript 代码不用写了，直接调用就可以，使开发简单、高效。

2) jQuery 强大的选择器封装了 DOM，操作网页元素更简单了。

3) 在大型 JavaScript 框架中，jQuery 对性能的理解最好，大小不超过 30KB。

4) 完善的 ajax 有着出色的浏览器兼容性，任何浏览器使用 ajax 都能兼容。

5) 基于 jQuery 开发的插件目前已经有大约数千个。开发者可使用插件来进行表单确认、图表种类、字段提示、动画、进度条等任务。

11、jQuery 选择器有多少种？

基本：

`$("#myElement")` ID 选择器

`$("div")` 标签选择器

`$(".myClass")` 类选择器

`$("*")` 通配符选择器

层级选择器

过滤选择器

子元素选择器

12、jquery 选择器有哪些优势？

简单的写法(‘#id’)用来代替 `document.getElementById()`。

支持 css 选择器。

完善的处理机制，就算写错了 Id 也不会报错。

13、你是如何使用 jquery 中的 ajax 的？

如果是常规的 ajax 程序的话，使用 load()、\$.get()、\$.post()，一般我会使用的是\$.post()方法，如果需要设定，beforeSend（提交前回调函数），error（失败后处理），success(成功后处理)，及 complete（请求完成后处理）毁掉函数等，这个时候我会使用\$.ajax()

14、jquery 中的\$.get 和\$.post 请求区别？

- 1) \$.get 方法使用 get 方法来进行一步请求，\$.post 是使用 post 方法来进行请求。
- 2) get 请求会讲参数跟在 url 后进行传递，而 post 请求则是作为 Http 消息的实体内容发送给 web 服务器的，这种传递是对用户不可见的。
- 3) get 方式传输的数据大小不能超过 2kb 而 post 请求要大的多
- 4) get 方式请求的数据会被浏览器缓存起来，因此有安全问题

15、jquery 中如何操作样式的？

addClass()来追加样式，removeClass()来删除样式，toggle()来切换样式。

16、如何设置和获取 HTML 和文本的值？

Html()方法，类似于 innerHTML 属性，可以用来读取或者设置某个元素中的 HTML 内容，text()类似于 innerText 属性，可以用来读取或这是某个元素的文本内容，val()可以用来设置和获取元素的值。

17、Jquery 能做些什么？

- 1) 获取页面元素
- 2) 修改页面的外观

- 3) 修改页面的内容
- 4) 响应页面的操作
- 5) 为页面添加动态效果
- 6) 无需刷新页面，即可从服务器获取信息
- 7) 简化常见的 javascript 的任务

18、在 ajax 中 data 主要有哪几种？

html 拼接、json 数组、form 表单经过 `serialize()` 序列化的

19、jQuery 中 ajax 由几部分组成？

- 1) 请求 url
- 2) 请求参数
- 3) 请求类型，get 或 post
- 4) 回调函数
- 5) 传输类型，html 或 json 等

20、js 和 jQuery 获取 value 值得区别

Js 获取输入框的 value 值：

```
var value = document.getElementById("输入框 ID 属性").value;
```

jQuery 获取输入框的 value 值：

```
var value = $("input").val();
```

```
var value = $("input").attr("value");
```

四、jsp/servlet 部分

1、Tomcat 的优化经验

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

2、Tomcat 根目录下有哪些文件

- 1) config 配置文件存放的路径
- 2) webapps 项目部署的目录
- 3) bin tomcat 运行需要的脚本与 jar 包的目录
- 4) lib 运行项目时所需要的 jar 包的目录
- 5) work 部署项目的缓存目录
- 6) temp 临时文件存放的目录
- 7) logs 记录日志的目录

3、什么是 TOMCAT,怎样启动停止，配置文件，日志文件的存储。

tomcat 其实是一种 web 服务器，java 编写的 web 项目可以部署在其上，用户在客户端请求时，都是先将请求发送到 tomcat 上，tomcat 再将请求发送到对应的项目上。

启动 tomcat

在 Windows 下：进入 bin 目录，双击 startup.bat

在 Linux 下：cd 进入 bin 目录，sh startup.sh

在开发工具 eclipse 中，右键选择 Debug Server 或者 Run Server

停止 tomcat

在 Windows 下：进入 bin 目录，双击 shutdown.bat

在 Linux 下：cd 进入 bin 目录，sh shutdown.sh

在开发工具 eclipse 中，选择服务器 stop Server

配置文件在 tomcat 的 config 文件夹下

日志文件在 logs 文件夹下

4、解释一下什么是 servlet;什么是 servlet 容器;

在 web 容器中运行的服务器端 java 程序，主要用于响应 HTTP 请求。Servlet 一般用于 mvc 中的控制器部分。

用来管理 servlet 生命周期的应用程序如(tomcat webloc 等)

5、说一说 Servlet 的生命周期，执行过程?

Servlet 生命周期分为实例化、初始化、响应请求调用 service()方法、消亡阶段调用 destroy()方法。

执行过程如下：

- 1) 当浏览器发送一个请求地址，tomcat 会接收这个请求
- 2) tomcat 会读取项目中的 web.xml 中的配置
- 3) 当请求地址符合 servlet-mapping 标签映射的地址，会进入这个 servlet
- 4) servlet 首先会实例化（构造），然后初始化执行 init()方法，init()方法至始至终执行一次，servlet 对象是单实例
- 5) 根据请求的方式是 get 或 post，在 service()方法中调用 doGet()或 doPost()方法，完成此次请求
- 6) 当服务器停止，会调用 destroy()方法，销毁实例

6、实例化 servlet 有几种方式

Servlet 实例化有两种，如下：

- 1) 第一次请求时，实例化 servlet 对象
- 2) 在 web.XML 文件中的<Servlet></Servlet>之间添加<loadon-startup>1</loadon-startup>，tomcat 启动时就会实例化 servlet 对象

7、HTTP 请求的 GET 与 POST 方式的区别

Form 中的 get 和 post 方法,在数据传输过程中分别对应了 HTTP 协议中的 GET 和 POST 方法。二者主要区别如下:

- 1) Get 是用来从服务器上获得数据,而 Post 是用来向服务器上传数据;
- 2) Get 将表单中数据按照 variable=value 的形式,添加到 action 所指向的 URL 后面,并且两者使用“?”连接,而各个变量之间使用“&”连接;Post 是将表单中的数据放在 form 的数据体中,按照变量和值相对应的方式,传递到 action 所指向 URL;
- 3) Get 是不安全的,因为在传输过程,数据被放在请求的 URL 中;Post 的所有操作对用户来说都是不可见的;
- 4) Get 传输的数据量小,这主要是因为受 URL 长度限制;而 Post 可以传输大量的数据,所以在上传文件只能使用 Post;
- 5) Get 限制 Form 表单的数据集必须为 ASCII 字符,而 Post 支持整个 ISO10646 字符集;
- 6) Get 是 Form 的默认方法。

8、请写一个 Servlet 的基本架构。

```
public class ServletName extends HttpServlet {  
    public void doPost(HttpServletRequest request,  
                        HttpServletResponse response)  
        throws ServletException, IOException {  
    }  
    public void doGet(HttpServletRequest request,  
                      HttpServletResponse response)  
        throws ServletException, IOException {  
    }  
}
```

9、forward 和 redirect 的区别？

forward 是容器中控制权的转向，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 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，并且从浏览器的地址栏中可以看到跳转后的链接地址。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。

总结：forward 是程序内部跳转，不会跳出 tomcat 服务器，redirect 可以外部跳转，从一个服务器跳转到另一个服务器。

10、servlet 中怎么定义 forward 和 redirect

转发：request.getRequestDispatcher ("demo.jsp"). forward(request, response);

重定向：response.sendRedirect("demo.jsp");

11、过滤器有哪些作用？

可以验证客户是否来自可信的网络，可以对客户提交的数据进行重新编码，可以从系统里获得配置的信息，可以过滤掉客户的某些不应该出现的词汇，可以验证用户是否登录，可以验证客户的浏览器是否支持当前的应用，可以记录系统的日志等等。

12、JSP 的常用指令？

```
<%@page language=" java" contentType=" text/html;charset=gb2312" session="
true" buffer=" 64kb" autoFlush=" true" isThreadSafe=" true" info=" text" errorPage="
error.jsp" isErrorPage=" true" isELIgnored=" true" pageEncoding=" gb2312" import="
java.sql.*" %>
```

isErrorPage: 是否能使用 Exception 对象；isELIgnored: 是否忽略 EL 表达式；

<%@include file=" filename" %>

<%@taglib prefix=" c" uri=" http://....." %>

13、JSP 和 Servlet 中的请求转发分别如何实现？

JSP 中的请求转发可利用 forward 动作实现：<jsp:forward />;

Servlet 中实现请求转发的方式为：

getServletContext().getRequestDispatcher(path).forward(req,res)。

14、JSP 乱码如何解决？

1) JSP 页面乱码

<%@ page contentType=" text/html ; charset=utf-8" %>

2) 表单提交中文时出现乱码

request.setCharacterEncoding("utf-8");

3) 数据库连接出现乱码

是数据库连接中加入 useUnicode=true&characterEncoding=utf-8;

15、session 和 application 的区别？

1) 两者的作用范围不同：

Session 对象是用户级的，而 Application 是应用程序级别的

一个用户一个 session 对象，每个用户的 session 对象不同，在用户所访问的网站多个页面之间共享同一个 session 对象

一个 Web 应用程序一个 application 对象，每个 Web 应用程序的 application 对象不同，但一个 Web 应用程序的多个用户之间共享同一个 application 对象。

两者的生命周期不同：

session 对象的生命周期：用户首次访问网站创建，用户离开该网站 (不一定要关闭浏览器) 消亡。

application 对象的生命周期：启动 Web 服务器创建，关闭 Web 服务器销毁。

16、jsp 有哪些内置对象?作用分别是什么?

JSP 共有以下 9 种基本内置组件

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

17、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 OBJECT 或 EMBED 标记

18、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别?

动态 INCLUDE 用 jsp:include 动作实现，<jsp:include page=included.jsp flush=true />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，先将嵌入的 jsp 页面编译，然后把编译后的内容放入到主页面进行处理，编译两次。

静态 INCLUDE 用 include 伪码实现，使用 jsp 指令引用 `<%@ include file=included.htm %>`，不会检查所含文件的变化，适用于包含静态页面，先将内容先包含到主页面然后在一起编译，只编译一次。

19、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。

20、页面传递对象的方法？

request、session、application、cookie 等

21、Cookie 和 session 区别？

- 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。

22、Filter 的生命周期与执行过程

生命周期如下：

- 1、执行构造方法，实例化
- 2、执行 init 方法，初始化
- 3、执行 doFilter 方法，过滤用户请求
- 4、当 tomcat 关闭时，执行 destroy 方法，进行销毁

构造方法与 init 方法在 tomcat 启动时就执行，至始至终只执行一次，

filter 对象是单实例

执行过程如下：

- 1、浏览器发送一个请求，会到达 tomcat
- 2、tomcat 会根据项目中的 web.xml 中的 Filter 过滤路径配置，过滤请求
- 3、过滤到用户请求会进入 Filter 类中的 doFilter 方法
- 4、在 doFilter 方法中实现业务逻辑，最后调用 doFilter(request, response)
传递用户请求
- 5、到达用户请求的页面

23、Tomcat 系统内存怎么配置

Windows

在 tomcat 的 bin/catalina.bat 最前面加入 set JAVA_OPTS=-Xms128m -Xmx350m

Linux：

在 tomcat 目录下的 bin/catalina.sh 添加：JAVA_OPTS='-Xms512m -Xmx1024m'

-Xms：初始值-Xmx：最大值-Xmn：最小值

24、JSTL 标签库包含哪些？

1、核心库（逻辑库）	用于页面上业务逻辑处理
2、函数库	利用 jstl 内置函数，获取结果
3、格式化库	格式化时间、格式化数字
4、SQL 库（几乎不用）	操作数据库

五、数据库部分

1、触发器的作用？

触发器是一中特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

2、什么是存储过程？用什么来调用？

存储过程是一个预编译的 SQL 语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次 SQL，使用存储过程比单纯 SQL 语句执行要快。

调用：

- 1) 可以用一个命令对象来调用存储过程。
- 2) 可以供外部程序调用，比如：java 程序。

3、存储过程的优缺点？

优点：

- 1) 存储过程是预编译过的，执行效率高。
- 2) 存储过程的代码直接存放于数据库中，通过存储过程名直接调用，减少网络通讯。

- 3) 安全性高，执行存储过程需要有一定权限的用户。
 - 4) 存储过程可以重复使用，可减少数据库开发人员的工作量。
- 缺点：移植性差

4、存储过程与函数的区别

存储过程	函数
用于在数据库中完成特定的操作或者任务 (如插入、删除等)	用于特定的数据(如选择)
程序头部声明用 <code>procedure</code>	程序头部声明用 <code>function</code>
程序头部声明时不需描述返回类型	程序头部声明时要描述返回类型，而且 PL/SQL 块中至少要包括一个有效的 <code>return</code> 语句
可以使用 <code>in/out/in out</code> 三种模式的参数	可以使用 <code>in/out/in out</code> 三种模式的参数
可作为一个独立的 PL/SQL 语句来执行	不能独立执行，必须作为表达式的一部分调用
可以通过 <code>out/in out</code> 返回零个或多个值	通过 <code>return</code> 语句返回一个值，且返回值要与声明部分一致，也可以是通过 <code>out</code> 类型的参数带出的变量
SQL 语句(DML 或 SELECT)中不可调用存储过程	SQL 语句(DML 或 SELECT)中可以调用函数

5、索引的作用？和它的优点缺点是什么？

索引就是一种特殊的查询表，数据库的搜索可以利用它加速对数据的检索。它很类似与现实生活中书的目录，不需要查询整本书内容就可以找到想要的数据库。索引可以是唯一的，创建索引允许指定单个列或者是多个列。缺点是它减慢了数据录入的速度，同时也增加了数据库的尺寸大小。

6、什么样的字段适合建索引

- 1、经常被查询的字段
- 2、不为空且字段值不重复
- 3、字段的值不经常被修改

7、索引类型有哪些？

逻辑上：

Single column 单行索引

Concatenated 多行索引

Unique 唯一索引

NonUnique 非唯一索引

Function-based 函数索引

Domain 域索引

物理上：

Partitioned 分区索引

NonPartitioned 非分区索引

B-tree：

Normal 正常型 B 树

Rever Key 反转型 B 树

Bitmap 位图索引

8、什么是事务？什么是锁？

事务就是被绑定在一起作为一个逻辑工作单元的 SQL 语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过 ACID 测试，即原子性，一致性，隔离性和持久性。

锁：在所有的 DBMS 中，锁是实现事务的关键，锁可以保证事务的完整性和并发性。与现实生活中锁一样，它可以使某些数据的拥有者，在某段时间内不能使用某些数据或数据结构。当然锁还分级别的。

9、什么叫视图？游标是什么？

视图：是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，视图通常是有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更容易，相比多表查询。

游标：是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

10、视图的优缺点

优点：

- 1) 对数据库的访问，因为视图可以有选择性的选取数据库里的一部分。
- 2) 用户通过简单的查询可以从复杂查询中得到结果。
- 3) 维护数据的独立性，视图可从多个表检索数据。
- 4) 对于相同的数据可产生不同的视图。

缺点：

性能：查询视图时，必须把视图的查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，即使是视图的一个简单查询，也把它变成一个复杂的结合体，需要花费一定的时间。

11、列举几种表连接方式,有什么区别？

内连接、自连接、外连接（左、右、全）、交叉连接

内连接：只有两个元素表相匹配的才能在结果集中显示。

外连接:

左外连接:左边为驱动表,驱动表的数据全部显示,匹配表的不匹配的不会显示。

右外连接:右边为驱动表,驱动表的数据全部显示,匹配表的不匹配的不会显示。

全外连接:连接的表中不匹配的数据全部会显示出来。

交叉连接:笛卡尔效应,显示的结果是链接表数的乘积。

12、主键和外键的区别?

主键在本表中是唯一的、不可唯空的,外键可以重复可以唯空;外键和另一张表的主键关联,不能创建对应表中不存在的外键。

13、在数据库中查询语句速度很慢,如何优化?

1.建索引

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

5.尽量用 PreparedStatement 来查询, 不要用 Statement

14、数据库三范式是什么?

第一范式 (1NF): 字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式。

数据库表中的字段都是单一属性的,不可再分。例如,姓名字段,其中的姓和名必须作为一个整体,无法区分哪部分是姓,哪部分是名,如果要区分出姓和名,必须设计成两个独立的字段。

第二范式 (2NF): 是在第一范式 (1NF) 的基础上建立起来的,即满足第二范式 (2NF) 必须先满足第一范式 (1NF)。

要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列,以

存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式（3NF）：必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

所以第三范式具有如下特征：

- 1，每一列只有一个值
- 2，每一行都能区分。
- 3，每一个表都不包含其他表已经包含的非主关键字信息。

例如，帖子表中只能出现发帖人的 id，而不能出现发帖人的姓名，还同时出现发帖人姓名，否则，只要出现同一发帖人 id 的所有记录，它们中的姓名部分都必须严格保持一致，这就是数据冗余。

15、union 和 union all 有什么不同？

UNION 在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史表 UNION。

UNION ALL 只是简单的将两个结果合并后就返回。这样，如果返回的两个结果集中有重复的数据，那么返回的结果集就会包含重复的数据了。

从效率上说，UNION ALL 要比 UNION 快很多，所以，如果可以确认合并的两个结果集中不包含重复的数据的话，那么就使用 UNION ALL。

16、char、varchar2、varchar 有什么区别？

Char 的长度是固定的，而 varchar2 的长度是可以变化的，比如，存储字符串“abc”对

于 char(20)，表示你存储的字符将占 20 个字节，包含 17 个空，而同样的 varchar2 (20) 只占了 3 个字节，20 只是最大值，当你存储的字符小于 20 时，按实际长度存储。

char 的效率要被 varchar2 的效率高。

目前 varchar 是 varchar2 的同义词，工业标准的 varchar 类型可以存储空字符串，但是 oracle 不能这样做，尽管它保留以后这样做的权利。Oracle 自己开发了一个数据类型 varchar2，这个类型不是一个标准的 varchar，他将在数据库中 varchar 列可以存储空字符串的特性改为存储 null 值，如果你想有向后兼容的能力，oracle 建议使用 varchar2 而不是 varchar

17、Oracle 和 Mysql 的区别？

- 1) 库函数不同。
- 2) Oracle 是用表空间来管理的，Mysql 不是。
- 3) 显示当前所有的表、用户、改变连接用户、显示当前连接用户、执行外部脚本的语句的不同。
- 4) 分页查询时候时候，mysql 用 limit oracle 用 rownum
- 5) sql 的语法的不同。

18、Oracle 语句有多少类型

Oracle 语句分三类：DDL、DML、DQL、DCL。

DDL (Data Definition Language) 数据定义语言，包括：

Create 语句：可以创建数据库和数据库的一些对象。

Drop 语句：可以删除数据表、索引、触发程序、条件约束以及数据表的权限等。

Alter 语句：修改数据表定义及属性。

Truncate 语句：删除表中的所有记录,包括所有空间分配的记录被删除。

DML (Data Manipulation Language) 数据操控语言，包括：

Insert 语句：向数据表张插入一条记录。

Delete 语句：删除数据表中的一条或多条记录，也可以删除数据表中的所有记录，但是它的操作对象仍是记录。

Update 语句：用于修改已存在表中的记录的内容。

DQL（Data Query Language）数据操控语言，包括：

Select 语句：用于查询已存在表中的记录的内容。

DCL（Data Control Language）数据库控制语言，包括：

Grant 语句：允许对象的创建者给某用户或某组或所有用户（PUBLIC）某些特定的权限。

Revoke 语句：可以废除某用户或某组或所有用户访问权限

TCL（Transaction Control Language）事务控制语言

commit：提交事务

rollback：回滚事务

19、oracle 分页语句

使用 rownum，两种如下：

第一种：

```
select * from (select t.*,rownum row_num from mytable t) b where b.row_num between 1 and 10
```

第二种：

```
select * from ( select a.*, rownum rn from mytable a where rownum <= 10 ) where rn >= 1
```

使用 rowid，如下：

```
select * from scott.emp where rowid in (select rd from (select rowid as rd ,rownum as rn from scott.emp ) where rn<=6 and rn>3)
```

20、从数据库中随机取 50 条

```
select * from (select * from t_example order by dbms_random.random) where rownum <= 50
```

21、order by 与 group by 的区别

order by 排序查询、asc 升序、desc 降序

group by 分组查询、having 只能用于 group by 子句、作用于组内，having 条件子句可以直接跟函数表达式。使用 group by 子句的查询语句需要使用聚合函数。

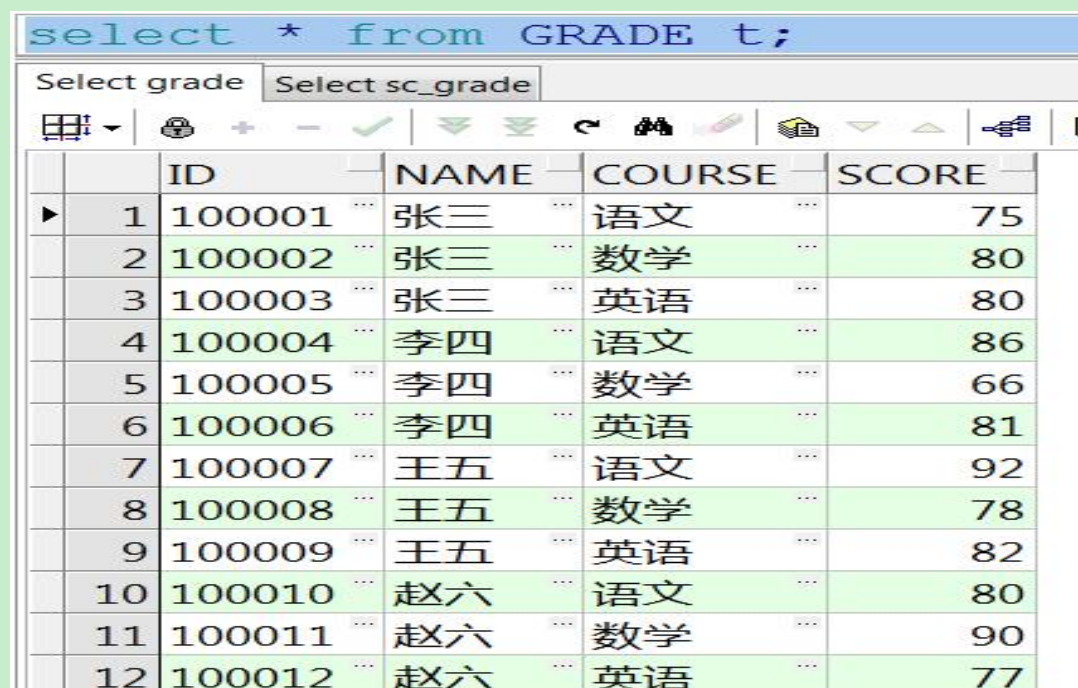
22、commit 在哪里会运用

oracle 的 commit 就是 DML 语句提交数据（这里是释放锁不是锁表），在未提交前你前面的操作更新的都是内存，没有更新到物理文件中。

执行 commit 从用户角度讲就是更新到物理文件了，事实上 commit 时还没有写 data file，而是记录了 redo log file，要从内存写到 data 物理文件，需要触发检查点，由 DBWR 这个后台进程来写，这里内容有点多的，如果不深究的话你就理解成 commit 即为从内存更新到物理文件。

23、行转列、列换行怎么转

行转列



	ID	NAME	COURSE	SCORE
1	100001	张三	语文	75
2	100002	张三	数学	80
3	100003	张三	英语	80
4	100004	李四	语文	86
5	100005	李四	数学	66
6	100006	李四	英语	81
7	100007	王五	语文	92
8	100008	王五	数学	78
9	100009	王五	英语	82
10	100010	赵六	语文	80
11	100011	赵六	数学	90
12	100012	赵六	英语	77

1) 使用 decode 函数

```
select name,  
       sum(decode(course, '语文', score, 0)) as 语文,  
       sum(decode(course, '数学', score, 0)) as 数学,  
       sum(decode(course, '英语', score, 0)) as 英语  
from GRADE group by name;
```

2) 使用 case when 语句

```

select name,
sum(case course when '语文' then score else 0 end) as 语文,
sum(case course when '数学' then score else 0 end) as 数学,
sum(case course when '英语' then score else 0 end) as 英语
from GRADE group by name;

```

```

select name,
sum(case when course='语文' then score else 0 end) as 语文,
sum(case when course='数学' then score else 0 end) as 数学,
sum(case when course='英语' then score else 0 end) as 英语
from GRADE group by name;

```

列转行

select * from SC_GRADE t;						
Select grade Select sc_grade						
	ID	NAME	CN_SCORE	MATH_SCORE	EN_SCORE	
1	10001	张三	75	80	80	
2	10002	王五	92	78	82	
3	10003	赵六	80	90	77	
4	10004	李四	86	66	81	
5	10005	李明	69	75	80	

```

select name, '语文' as course, cn_score as score from SC_GRADE
union all
select name, '数学' as course, math_score as score from SC_GRADE
union all
select name, '英语' as course, en_score as score from SC_GRADE
order by name;

```


24、什么是 PL/SQL?

PL/SQL 是一种程序语言，叫做过程化 SQL 语言（Procedural Language/SQL）。PL/SQL 是 Oracle 数据库对 SQL 语句的扩展。在普通 SQL 语句的使用上增加了编程语言的特点，所以 PL/SQL 把数据操作和查询语句组织在 PL/SQL 代码的过程性单元中，通过逻辑判断、循环等操作实现复杂的功能或者计算。PL/SQL 只有 Oracle 数据库有。MySQL 目前不支持 PL/SQL 的。

25、序列的作用

Oracle 使用序列来生成唯一编号，用来处理一个表中自增字段。Oracle 序列是原子对象，并且是一致的。也就是说，一旦您访问一个序列号，Oracle 将在处理下一个请求之前自动递增下一个编号，从而确保不会出现重复值。

26、表和视图的关系

视图其实就是一条查询 sql 语句，用于显示一个或多个表或其他视图中的相关数据。

表就是关系数据库中实际存储数据用的。

27、oracle 基本数据类型

1) 字符串类型

char、nchar、varchar、varchar2、nvarchar2

2) 数字类型

number、integer

3) 浮点类型

binary_float、binary_double、float

4) 日期类型

date、timestamp

5) LOB 类型

blob、clob、nclob、bfile

28、drop、truncate、delete 区别

TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。

TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

TRUNCATE,DELETE,DROP 放在一起比较：

TRUNCATE TABLE：删除内容、释放空间但不删除定义。

DELETE TABLE:删除内容不删除定义，不释放空间。

DROP TABLE：删除内容和定义，释放空间。

29、oracle 获取系统时间

```
select to_char(sysdate, 'yyyy-MM-dd HH24:mi:ss') from dual;
```

30、oracle 怎么去除去重

使用 distinct 关键字

31、合并查询有哪些？

- union(并集去重复)、
- union all(并集不去重复)、
- intersect(交集)、
- minus(差集)

32、SQL 语句执行顺序

关键字：select、from、join、on、where、group by、having、order by、distinct 执行顺序

- 1、from
- 2、join
- 3、on
- 4、where
- 5、group by 分组字段
- 6、having 表达式
- 7、select
- 8、distinct
- 9、order by

33、null 的含义

在我们不知道具体有什么数据的时候，也即未知，可以用 NULL，我们称它为“空”，ORACLE 中，含有空值的表列长度为零。

ORACLE 允许任何一种数据类型的字段为“空”，除了以下两种情况：

- 1、主键字段（primary key），
- 2、定义时已经加了 NOT NULL 限制条件的字段

说明：

- 1、等价于没有任何值、是未知数。
- 2、NULL 与 0、空字符串、空格都不同。
- 3、对空值做加、减、乘、除等运算操作，结果仍为“空”。
- 4、NULL 的处理使用 NVL 函数。
- 5、比较时使用关键字用 “is null” 和 “is not null”。
- 6、空值不能被索引，所以查询时有些符合条件的数据可能查不出来，count(*)中，用 nvl(列名,0)处理后再查。
- 7、排序时比其他数据都大（索引默认是降序排列，小→大），所以 NULL 值总是排在最后。

34、mysql 分页

Select * from 表名 limit 起始位置, 查询数量

35、MySQL、SqlServer、oracle 写出字符存储、字符串转时间

1、在 MySQL、SqlServer、oracle 中两个单引号代表字符

2、字符串转时间

Oracle: 使用 to_date('字符串时间','yyyy-MM-dd HH24:mi:ss')函数

SqlServer: CONVERT(数据类型,日期值,日期样式 ID)

MySQL: date_format(日期值,'%Y-%c-%d %h:%i:%s');

36、update 语句可以修改结果集中的数据吗？

在 oracle 中是可以的，在 mysql 中不可以。

如下语句：

```
update (select * from table1 t1 left join table2 t2 on t1.字段 = t2.字段 ) set 字段名 = 值
where 条件字段 = 条件值
```

六、Java 高级部分

1、java 中有几种方法可以实现一个线程？用什么关键字修饰同步方法？stop()和 suspend()方法为何不推荐使用？

实现线程有两种方式：1.继承 Thread 类，重写 run 方法，在调用 start 方法。

实现 Runnable 接口，重写 run 方法。在传给 Thread 构造器，调用时调用 Thread 的 start 方法。

用 synchronized 关键字修饰同步方法。

不使用 stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在。suspend()方法容易发生死锁。调用 suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被"挂起"的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用 suspend()，而应在自己的 Thread 类中置入一个标志，

指出线程应该活动还是挂起。若标志指出线程应该挂起，便用 `wait()` 命其进入等待状态。若标志指出线程应当恢复，则用一个 `notify()` 重新启动线程。

2、`sleep()` 和 `wait()` 有什么区别？

`sleep` 是线程类（`Thread`）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用 `sleep` 不会释放对象锁。`wait` 是 `Object` 类的方法，对此对象调用 `wait` 方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出 `notify` 方法（或 `notifyAll`）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

3、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？

分几种情况：

- 1）其他方法前是否加了 `synchronized` 关键字，如果没加，则能。
- 2）如果这个方法内部调用了 `wait`，则可以进入其他 `synchronized` 方法。
- 3）如果其他个方法都加了 `synchronized` 关键字，并且内部没有调用 `wait`，则不能。
- 4）如果其他方法是 `static`，它用的同步锁是当前类的字节码，与非静态的方法不能同步，因为非静态的方法用的是 `this`。

4、线程的基本概念

一个程序中可以有多个执行线索同时执行，一个线程就是程序中的一条执行线索，每个线程上都关联有要执行的代码，即可以有多个程序代码同时运行，每个程序至少都有一个线程，即 `main` 方法执行的那个线程。如果只是一个 `cpu`，它怎么能够同时执行多段程序呢？这是从宏观上来看的，`cpu` 一会执行 `a` 线索，一会执行 `b` 线索，切换时间很快，给人的感觉是 `a, b` 在同时执行，好比大家在同一个办公室上网，只有一条链接到外部网线，其实，这条网线一会为 `a` 传数据，一会为 `b` 传数据，由于切换时间很短暂，所以，大家感觉都在同时上网。

5、什么是多线程

线程是程序执行流的最小单元，相对独立、可调度的执行单元，是系统独立调度和分派 CPU 的基本单位。在单个程序中同时运行多个线程完成不同的工作，称为多线程。

6、程序、进程、线程之间的关系

程序是一段静态的代码，是应用软件执行的蓝本。

进程是程序一次动态执行的过程，它对应了从代码加载、执行完毕的一个完整过程，这也是进程开始到消亡的过程。

线程是进程中独立、可调度的执行单元，是执行中最小单位。

一个程序一般是一个进程，但可以一个程序中有多个进程。

一个进程中可以有多个线程，但只有一个主线程。

Java 应用程序中默认的主线程是 main 方法，如果 main 方法中创建其他线程，JVM 就会执行其他的线程。

7、创建线程有几种方式，分别是什么？

创建线程有三种方式：

1) 是继承 Thread 类，创建格式如下：

```
Thread thread = new Thread();
```

2) 是实现 Runnable 接口，创建格式如下：

```
Thread thread = new Thread(new Runnable());
```

其实 Thread 类实现了 Runnable 接口

3) 通过线程池方式，获取线程

```
package com.myjava.thread;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ThreadPool {

    private static int POOL_NUM = 10;

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

```

        ExecutorService executorService = Executors.newFixedThreadPool(5);

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

            RunnableThread thread = new RunnableThread();

            executorService.execute(thread);

        }

    }

}

class RunnableThread implements Runnable{

    private    int THREAD_NUM = 10;

    public void run() {

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

            System.out.println("线程"+Thread.currentThread()+i);

        }

    }

}

```

8、线程的生命周期

创建--运行--中断--死亡

创建：线程构造

运行：调用 start()方法，进入 run()方法

中断：sleep()、wait()

死亡：执行完 run()方法或强制 run()方法结束，线程死亡

9、线程 currentThread()与 interrupt()方法的使用

currentThread()方法是获取当前线程

interrupt()唤醒休眠线程，休眠线程发生 InterruptedException 异常

10、线程状态

- 1) 新建状态 (New): 新创建了一个线程对象。
- 2) 就绪状态 (Runnable): 线程对象创建后, 其他线程调用了该对象的 start()方法。该状态的线程位于可运行线程池中, 变得可运行, 等待获取 CPU 的使用权。
- 3) 运行状态 (Running): 就绪状态的线程获取了 CPU, 执行程序代码。
- 4) 阻塞状态 (Blocked): 阻塞状态是线程因为某种原因放弃 CPU 使用权, 暂时停止运行。
- 5) 死亡状态 (Dead): 线程执行完了或者因异常退出了 run()方法, 该线程结束生命周期。

11、什么是 java 序列化, 如何实现 java 序列化?

通俗的说, 就是可以将内存中 Java 对象可以写在硬盘上(序列化到硬盘上), 反序列化就是讲硬盘的内容读取到内存中去; java 是通过实现 Serializable 接口, 实现的序列化, Serializable 接口里面没有任何的方法, 只是个标示接口。

12、编写一个程序, 将 d:\java 目录下的所有.java 文件复制到 d:\jad 目录下, 并将原来文件的扩展名从.java 改为.jad。

答: listFiles 方法接受一个 FileFilter 对象, 这个 FileFilter 对象就是过滤的策略对象, 不同的人提供不同的 FileFilter 实现, 即提供了不同的过滤策略。

```
import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FilenameFilter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;


public class Jad2Java {
```



```

public static void main(String[] args) throws Exception {

    File srcDir = new File("java");

    if(!(srcDir.exists() && srcDir.isDirectory()))

        throw new Exception("目录不存在");

    File[] files = srcDir.listFiles(

        new FilenameFilter(){

            public boolean accept(File dir, String name) {

                return name.endsWith(".java");

            }

        }

    );

    System.out.println(files.length);

    File destDir = new File("jad");

    if(!destDir.exists()) destDir.mkdir();

    for(File f :files){

        FileInputStream fis = new FileInputStream(f);

        String destFileName = f.getName().replaceAll("\\.java$", ".jad");

        FileOutputStream fos = new FileOutputStream(new File(destDir,destFileName));

        copy(fis,fos);

        fis.close();

        fos.close();

    }

}

private static void copy(InputStream ips,OutputStream ops) throws Exception{

    int len = 0;

    byte[] buf = new byte[1024];

```

```

        while((len = ips.read(buf)) != -1){
            ops.write(buf,0,len);
        }

    }

}

```

由本题总结的思想及策略模式的解析：

1.

class jad2java{

1. 得到某个目录下的所有的 java 文件集合

1.1 得到目录 File srcDir = new File("d:\\java");

1.2 得到目录下的所有 java 文件： File[] files = srcDir.listFiles(new MyFileFilter());

1.3 只想得到.java 的文件： class MyFileFilter implements FileFilter{

public boolean accept(File pathname){

return pathname.getName().endsWith(".java")

}

}

2.将每个文件复制到另外一个目录，并改扩展名

2.1 得到目标目录，如果目标目录不存在，则创建之

2.2 根据源文件名得到目标文件名，注意要用正则表达式，注意.的转义。

2.3 根据表示目录的 File 和目标文件名的字符串，得到表示目标文件的 File。

//要在硬盘中准确地创建一个文件，需要知道文件名和文件的目录。

2.4 将源文件的流拷贝成目标文件流，拷贝方法独立成为一个方法，方法的参数采用抽象流的形式。

//方法接受的参数类型尽量面向父类，越抽象越好，这样适应面更宽广。

}

分析 listFiles 方法内部的策略模式实现原理

```

File[] listFiles(FileFilter filter){

    File[] files = listFiles();

    //ArrayList acceptedFilesList = new ArrayList();

    File[] acceptedFiles = new File[files.length];

    int pos = 0;

    for(File file: files){

        boolean accepted = filter.accept(file);

        if(accepted){

            //acceptedFilesList.add(file);

            acceptedFiles[pos++] = file;

        }

    }

    Arrays.copyOf(acceptedFiles,pos);

    //return (File[])acceptedFilesList.toArray();

}

```

13、java 中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

字节流，字符流。字节流继承于 InputStream OutputStream，字符流继承于 InputStreamReader OutputStreamWriter。在 java.io 包中还有许多其他的流，主要是为了提高性能和使用方便

14、字节流与字符流的区别

把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为 IO 流，对应的抽象类为 OutputStream 和 InputStream ，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。

在应用中，经常要完全是字符的一段文本输出或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。

底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向 IO 设别写入或读取字符串提供了一点点方便。

字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，其实是转成该字符的某种编码的字节形式，读取也是反之的道理。

讲解字节流与字符流关系的代码案例：

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintWriter;

public class IOTest {
    public static void main(String[] args) throws Exception {
        String str = "中国人";
        /*FileOutputStream fos = new FileOutputStream("1.txt");

        fos.write(str.getBytes("UTF-8"));
        fos.close();*/

        /*FileWriter fw = new FileWriter("1.txt");
        fw.write(str);
        fw.close();*/
```

```

        PrintWriter pw = new PrintWriter("1.txt","utf-8");

        pw.write(str);

        pw.close();

    }

    /*FileReader fr = new FileReader("1.txt");

    char[] buf = new char[1024];

    int len = fr.read(buf);

    String myStr = new String(buf,0,len);

    System.out.println(myStr);*/

    /*FileInputStream fr = new FileInputStream("1.txt");

    byte[] buf = new byte[1024];

    int len = fr.read(buf);

    String myStr = new String(buf,0,len,"UTF-8");

    System.out.println(myStr);*/

    BufferedReader br = new BufferedReader(

        new InputStreamReader(

            new FileInputStream("1.txt"),"UTF-8"

        )

    );

    String myStr = br.readLine();

    br.close();

    System.out.println(myStr);

}

```

总结：很简单，字符流的底层就是字节流。而字符流主要是读取文本文件内容的，可以一个字符一个字符的读取，也可以一行一行的读取文本文件内容。而字节流读取单位为 **byte**。byte 作为计算机存储最基本单位，可以用字节流来读取很多其他格式的文件，比如图片视频等等。基于 B/S 和 C/S 的文件传输都可以采用字节流的形式。

15、怎么判断指定路径是否为目录

```
File f = new File(fileName); //构造文件 File 类  
f.isDirectory(); //判断是否为目录
```

16、怎么获取指定路径下的全部文件

```
File f = new File(filePath); //构造文件 File 类  
String[] fileName = f.list(); //获取目录下的文件名  
File[] files = f.listFiles(); //获取目录下的文件
```

17、Java 怎么读取文件和写入文件

读取文件：

```
public class FileRead {  
    /**  
     * 1、找到指定的文件  
     * 2、根据文件创建文件的输入流  
     * 3、创建字节数组  
     * 4、读取内容，放到字节数组里面  
     * 5、关闭输入流  
     * @param args  
     */  
    public static void main(String[] args) {  
        File file = new File("E:" + File.separator + "hello.txt"); //构建指定文件  
        InputStream in = null;  
        try {  
            in = new FileInputStream(file); //根据文件创建文件的输入流  
            byte[] data = new byte[1024]; //创建字节数组  
            in.read(data); //读取内容，放到字节数组里面  
            System.out.println(new String(data));  
        }  
    }  
}
```

```

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            in.close(); //关闭输入流
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}

```

写入文件：

```

public class FileWriter {

    /**
     * 文件的输出流，用来写入文件内容
     * 1、找到指定的文件
     * 2、根据文件创建文件的输出流
     * 3、把内容转换成字节数组
     * 4、向文件写入内容
     * 5、关闭输出流
     * @param args
     */

    public static void main(String[] args) {

        File file = new File("E:" + File.separator + "hello.txt"); //构建指定文件
        OutputStream out = null;

        try {

            out = new FileOutputStream(file); 根据文件创建文件的输出流

            String message = "黄晓明与 bady 结婚了，扬子和黄圣依有女儿了。";

```

```

        byte[] mesByte = message.getBytes(); //把内容转换成字节数组
        out.write(mesByte);    //向文件写入内容
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally{
        try {
            out.close(); //关闭输出流
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}
}

```

18、java 怎么复制文件

```

public class FileCopy {
    /**
     * 实现思路
     * 1、构建源文件与目标文件
     * 2、源文件创建输入流，目标文件创建输出流
     * 3、创建字节数组
     * 4、使用循环，源文件读取一部分内容，目标文件写入一部分内容，直到写完所有内
容
     * 5、关闭源文件输入流，目标文件输出流
     * @param args
     */
    public static void main(String[] args) {

```



```

//构建源文件

File file = new File("E:" + File.separator + "helloworld.txt");

//构建目标文件

File fileCopy = new File("D:" + File.separator + "helloworld.txt");

InputStream in = null;

OutputStream out = null;

try{

    //目标文件不存在就创建

    if(!(fileCopy.exists())) {

        fileCopy.createNewFile();

    }

    //源文件创建输入流

    in = new FileInputStream(file);

    //目标文件创建输出流

    out = new FileOutputStream(fileCopy, true);

    //创建字节数组

    byte[] temp = new byte[1024];

    int length = 0;

    //源文件读取一部分内容

    while((length = in.read(temp)) != -1) {

        //目标文件写入一部分内容

        out.write(temp, 0, length);

    }

} catch(IOException e) {

    e.printStackTrace();

} finally {

    try {

        in.close(); //关闭源文件输入流

        out.close(); //关闭目标文件输出流

    } catch(IOException e) {

```

```

        e.printStackTrace();
    }
}
}
}
}

```

19、用 JDBC 如何调用存储过程

代码如下：

```

package com.huawei.interview.lym;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Types;

public class JdbcTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Connection cn = null;
        CallableStatement cstmt = null;
        try {
            //这里最好不要这么干，因为驱动名写死在程序中了
            Class.forName("com.mysql.jdbc.Driver");
            //实际项目中，这里应用 DataSource 数据，如果用框架，

```

```

//这个数据源不需要我们编码创建，我们只需 DataSource ds = context.lookup()

//cn = ds.getConnection();

cn = DriverManager.getConnection("jdbc:mysql:///test","root","root");

cstmt = cn.prepareStatement("call insert_Student(?,?,?)");

cstmt.registerOutParameter(3,Types.INTEGER);

cstmt.setString(1, "wangwu");

cstmt.setInt(2, 25);

cstmt.execute();

//get 第几个，不同的数据库不一样，建议不写

System.out.println(cstmt.getString(3));

} catch (Exception e) {

    // TODO Auto-generated catch block

    e.printStackTrace();

}

finally

{

    /*try{cstmt.close();}catch(Exception e){}

    try{cn.close();}catch(Exception e){}*/

    try {

        if(cstmt != null)

            cstmt.close();

        if(cn != null)

            cn.close();

    } catch (SQLException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    }

}

}

```

20、JDBC 中的 PreparedStatement 相比 Statement 的好处

1) 提高性能：在使用 preparedStatement 对象执行 sql 时候，命令被数据库编译和解析，然后被放到命令缓冲区，然后每当执行同一个 preparedStatement 时候，他就被再解析一次，但不会在编译，在缓冲区中可以发现预编译的命令，并且可以重新使用。

如果你要写 Insert update delete 最好使用 preparedStatement，在有大量用户的企业级应用软件中，经常会执行相同的 sql,使用 preparedStatement 会增加整体的性能。

2) 安全性：PreparedStatement 可以防止 sql 注入。

21、写一个用 jdbc 连接实例。

```
package com.seecen.stream;

import java.sql.*;

public class TestJDBC {

    /**
     * 1、实例话驱动类
     * 2、建立到数据库的连接
     * 3、将数据发送到数据库中
     * 4、执行语句（select 语句）
     * 5、关闭
     * @param args
     */

    public static void main(String[] args) {

        ResultSet rs = null;

        Statement stmt = null;

        Connection conn = null;

        try {

            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```

        conn = DriverManager.getConnection("jdbc:oracle:thin:@192.168.0.1:1521:yuewei",
"scott", "tiger");

        stmt = conn.createStatement();

        rs = stmt.executeQuery("select * from dept");

        while(rs.next()) {

            System.out.println(rs.getString("deptno"));

        }

    } catch (ClassNotFoundException e) {

        e.printStackTrace();

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        try {

            if(rs != null) {

                rs.close();

                rs = null;

            }

            if(stmt != null) {

                stmt.close();

                stmt = null;

            }

            if(conn != null) {

                conn.close();

                conn = null;

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

}

```

```
}
```

22、ArrayList 和 Vector 的区别？

这两个类都实现了 List 接口（List 接口继承了 Collection 接口），他们都是有序集合，即存储在这两个集合中的元素的位置都是有顺序的，相当于一种动态的数组，我们以后可以按位置索引号取出某个元素，并且其中的数据是允许重复的，这是 HashSet 之类的集合的最大不同处，HashSet 之类的集合不可以按索引号去检索其中的元素，也不允许有重复的元素（本来题目问的与 hashset 没有任何关系，但为了说清楚 ArrayList 与 Vector 的功能，我们使用对比方式，更有利于说明问题）。

接着才说 ArrayList 与 Vector 的区别，这主要包括两个方面：

（1）同步性：

Vector 是线程安全的，也就是说它的方法之间是线程同步的，而 ArrayList 是线程程序不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用 ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用 Vector，因为不需要我们自己再去考虑和编写线程安全的代码。

备注：对于 Vector&ArrayList、Hashtable&HashMap，要记住线程安全的问题，记住 Vector 与 Hashtable 是旧的，是 java 一诞生就提供了的，它们是线程安全的，ArrayList 与 HashMap 是 java2 时才提供的，它们是线程不安全的。所以，我们讲课时先讲老的。

（2）数据增长：

ArrayList 与 Vector 都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加 ArrayList 与 Vector 的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector 默认增长为原来两倍，而 ArrayList 的增长策略在文档中没有明确规定（从源代码看到的是增长为原来的 1.5 倍）。ArrayList 与 Vector 都可以设置初始的空间大小，Vector 还可以设置增长的空间大小，而 ArrayList 没有提供设置增长空间的方法。

总结：即 Vector 增长原来的一倍，ArrayList 增加原来的 0.5 倍。

23、List、Set 和 Map 的区别？

- 1) List 和 Set 是 Collection 的子接口,map 不是。
- 2) List 的底层是数组的方式实现，Set 是散列表的方式实现，map 是键值对的方式。
- 3) list 是有序可重复的，Set 是无序不可重复的，map 是有序，key 不重复，value 可重复
- 4) list 和 Set 可直接使用 iterator 来进行遍历，map 只能通过先遍历 Key 在遍历 value.

24、Collection 和 Collections 的区别。

Collection 是集合类的上级接口，继承与他的接口主要有 Set 和 List.

Collections 是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

25、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，元素重复与否是使用 equals()方法进行判断的。

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。

26、HashMap 与 Hashtable 的区别

- 1) 继承不同

```
public class Hashtable extends Dictionary implements Map
```

```
public class HashMap extends AbstractMap implements Map
```

- 2) Hashtable 中的方法是同步的，而 HashMap 中的方法在缺省情况下是非同步的。在多线程并发的环境下，可以直接使用 Hashtable，但是要使用 HashMap 的话就要自己增加同步处

理了。

3) Hashtable 中, key 和 value 都不允许出现 null 值, 在 HashMap 中, null 可以作为键, 这样的键只有一个; 可以有一个或多个键所对应的值为 null。当 get()方法返回 null 值时, 即可以表示 HashMap 中没有该键, 也可以表示该键所对应的值为 null。因此, 在 HashMap 中不能由 get()方法来判断 HashMap 中是否存在某个键, 而应该用 containsKey()方法来判断。

4) 两个遍历方式的内部实现上不同。

Hashtable、HashMap 都使用了 Iterator。而由于历史原因, Hashtable 还使用了 Enumeration 的方式。

5) 哈希值的使用不同, Hashtable 直接使用对象的 hashCode。而 HashMap 重新计算 hash 值。

6) Hashtable 和 HashMap 它们两个内部实现方式的数组的初始大小和扩容的方式。Hashtable 中 hash 数组默认大小是 11, 增加的方式是 $old * 2 + 1$ 。HashMap 中 hash 数组的默认大小是 16, 而且一定是 2 的指数

27、Java 中有多少种数据结构, 分别是什么?

List: 是列表, 有下标值, 存储元素可以重复, 遍历元素是有序的。

Set: 是散列集, 无下标值, 存储元素不可重复, 遍历元素时无序的。

Map: 是以键值对存储, 一个 key 一个 value, key 不可以重复, value 可以重复。

数组: 指定类型, 固定长度, 元素存储地址是连续的。

树: 元素以树形结构存储, 只有一个根节点。

栈: 元素是先进后出, 后进先出。

向量: 动态数组, 可以存储任何类型元素, 动态长度, 元素存储地址是连续的。

队列: 元素存储是排列有序的, 一定保证先进的先出, 后进的后出。

28、Arraylist 和 linklist 的区别

相同点:

ArrayList 和 Linklist 都是接口 List 的实现类, 里面的数据都是有序可重复的。

区别:

ArrayList: 采用的是数组形式保存对象的，访问速度更快，而 Linklist 的插入和删除元素的速度更快

29、List 遍历方式有多少种

- 1) 下标遍历
- 2) Iterator 遍历
- 3) Foreach 遍历（最快）

30、Map 怎么遍历

先调用 keySet () 方法获取所有的 key，在遍历 key 获取所有的元素

31、怎么获取 Map 所有的 key，所有的 value

Map 调用 keySet () 方法获取所有的 key 值，是一个 Set 集合

Map 调用 values()方法获取所有的 value 值，是一个 List 集合

32、获取 Class 的实例有几种方式

```
Class<?> demo1=Class.forName("Reflect.Demo");    //使用 Class 类
```

```
Class<?> demo2=new Demo().getClass();    //通过对象
```

```
Class<?> demo3=Demo.class;    //通过类
```

33、怎么获取类中所有的方法，所有属性

获取所有方法：

```
Class<?> demo = Class.forName("Reflect.Demo");
```

```
Method[] methods = Demo. getDeclaredMethods();
```

获取所有属性：

```
Class<?> demo = Class.forName("Reflect.Demo");
```

```
Field[] fields = demo .getDeclaredFields();
```

34、JDBC 常用接口有哪些？

- | | |
|---------------------|--|
| 1、Connection | 用来与数据库建立连接 |
| 2、Statement | 用来执行 sql 语句 |
| 3、ResultSet | 用于接收结果集 |
| 4、PreparedStatement | 是 Statement 子接口，执行 sql 语句，预编译，防止 sql 注入，安全性高 |

35、Statement 中 execute、executeUpdate、executeQuery 这三者的区别

Statement 接口提供了三种执行 SQL 语句的方法：executeQuery、executeUpdate 和 execute。使用哪一个方法由 SQL 语句所产生的内容决定。

方法 executeQuery

用于产生单个结果集的语句，例如 SELECT 语句。被使用最多的执行 SQL 语句的方法是 executeQuery。这个方法被用来执行 SELECT 语句，它几乎是使用最多的 SQL 语句。

方法 executeUpdate

用于执行 INSERT、UPDATE 或 DELETE 语句以及 SQL DDL（数据定义语言）语句，例如 CREATE TABLE 和 DROP TABLE。INSERT、UPDATE 或 DELETE 语句的效果是修改表中零行或多行中的一列或多列。executeUpdate 的返回值是一个整数，指示受影响的行数（即更新计数）。对于 CREATE TABLE 或 DROP TABLE 等不操作行的语句，executeUpdate 的返回值总为零。

使用 executeUpdate 方法是因为在 createTableCoffees 中的 SQL 语句是 DDL（数据定义语言）语句。创建表，改变表，删除表都是 DDL 语句的例子，要用 executeUpdate 方法来执行。你也可以从它的名字里看出，方法 executeUpdate 也被用于执行更新表 SQL 语句。实际上，相对于创建表来说，executeUpdate 用于更新表的时间更多，因为表只需要

创建一次，但经常被更新。

方法 `execute`:

用于执行返回多个结果集、多个更新计数或二者组合的语句。因为多数程序员不会需要该高级功能

`execute` 方法应该仅在语句能返回多个 `ResultSet` 对象、多个更新计数或 `ResultSet` 对象与更新计数的组合时使用。当执行某个已存储过程或动态执行未知 SQL 字符串（即应用程序程序员在编译时未知）时，有可能出现多个结果的情况，尽管这种情况很少见。

因为方法 `execute` 处理非常规情况，所以获取其结果需要一些特殊处理并不足为怪。例如，假定已知某个过程返回两个结果集，则在使用方法 `execute` 执行该过程后，必须调用方法 `getResultSet` 获得第一个结果集，然后调用适当的 `getXXX` 方法获取其中的值。要获得第二个结果集，需要先调用 `getMoreResults` 方法，然后再调用 `getResultSet` 方法。如果已知某个过程返回两个更新计数，则首先调用方法 `getUpdateCount`，然后调用 `getMoreResults`，并再次调用 `getUpdateCount`。

对于不知道返回内容，则情况更为复杂。如果结果是 `ResultSet` 对象，则方法 `execute` 返回 `true`；如果结果是 `Java int`，则返回 `false`。如果返回 `int`，则意味着结果是更新计数或执行的语句是 DDL 命令。在调用方法 `execute` 之后要做的第一件事情是调用 `getResultSet` 或 `getUpdateCount`。调用方法 `getResultSet` 可以获得两个或多个 `ResultSet` 对象中第一个对象；或调用方法 `getUpdateCount` 可以获得两个或多个更新计数中第一个更新计数的内容。

当 SQL 语句的结果不是结果集时，则方法 `getResultSet` 将返回 `null`。这可能意味着结果是一个更新计数或没有其它结果。在这种情况下，判断 `null` 真正含义的唯一方法是调用方法 `getUpdateCount`，它将返回一个整数。这个整数为调用语句所影响的行数；如果为 `-1` 则表示结果是结果集或没有结果。如果方法 `getResultSet` 已返回 `null`（表示结果不是 `ResultSet` 对象），则返回值 `-1` 表示没有其它结果。也就是说，当下列条件为真时表示没有结果（或没有其它结果）：

```
((stmt.getResultSet() == null) && (stmt.getUpdateCount() == -1))
```

如果已经调用方法 `getResultSet` 并处理了它返回的 `ResultSet` 对象，则有必要调用方法 `getMoreResults` 以确定是否有其它结果集或更新计数。如果 `getMoreResults` 返回 `true`，则需要再次调用 `getResultSet` 来检索下一个结果集。如上所述，如果 `getResultSet` 返回 `null`，则需要调用 `getUpdateCount` 来检查 `null` 是表示结果为更新计数还是表示没有其

它结果。

当 `getMoreResults` 返回 `false` 时，它表示该 SQL 语句返回一个更新计数或没有其它结果。因此需要调用方法 `getUpdateCount` 来检查它是哪一种情况。在这种情况下，当下列条件为真时表示没有其它结果：

```
((stmt.getMoreResults() == false) && (stmt.getUpdateCount() == -1))
```

下面的代码演示了一种方法用来确认已访问调用方法 `execute` 所产生的全部结果集和更新计数：

```
stmt.execute(queryStringWithUnknownResults);

while (true) {

    int rowCount = stmt.getUpdateCount();

    if (rowCount > 0) { // 它是更新计数

        System.out.println("Rows changed = " + count);

        stmt.getMoreResults();

        continue;

    }

    if (rowCount == 0) { // DDL 命令或 0 个更新

        System.out.println(" No rows changed or statement was DDL

        command");

        stmt.getMoreResults();

        continue;

    }

    // 执行到这里，证明有一个结果集

    // 或没有其它结果

    ResultSet rs = stmt.getResultSet;

    if (rs != null) {

        ...// 使用元数据获得关于结果集列的信息

        while (rs.next()) {

            ...// 处理结果
```

```

stmt.getMoreResults();

continue;

}

break; // 没有其它结果

```

36、jdbc 中怎么做批量处理的？

1、使用 Statement 批量处理

```

public static void batchStatement() {

    Connection conn = null;

    Statement sta = null;

    try {

        conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);

        sta = conn.createStatement();

        for(int i=1; i<=10000; i++) {

            String sql = "insert into t_emp (f_id, f_no, f_name, f_sex, f_salary,
f_birthday, f_departId) values (t_emp_seq.nextval, '" + i + "', '葫芦娃" + i + "', '男', " + i + ",
to_date('2016-09-05','yyyy-MM-dd'), 1)";

            //把 sql 语句加入批处理

            sta.addBatch(sql);

        }

        //统一执行批处理的 sql 语句

        int[] rows = sta.executeBatch();

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        try {

            sta.close();

            conn.close();

        }
    }
}

```

```

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

}

```

2、使用 PreparedStatement 批量处理

```

public static void batchPreparedStatement() {

    Connection conn = null;

    PreparedStatement ps = null;

    try {

        conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);

        ps = conn.prepareStatement("insert into t_emp (f_id, f_no, f_name, f_sex,
f_salary, f_birthday, f_departId) values (t_emp_seq.nextval, ?,?,?,?,?)");

        for(int i=1; i<=10000; i++) {

            ps.setString(1, String.valueOf(10 + i));

            ps.setString(2, "奥特曼" + i);

            ps.setString(3, "女");

            ps.setDouble(4, i);

            ps.setTimestamp(5, new Timestamp(System.currentTimeMillis()));

            ps.setInt(6, 2);

            //加入批处理

            ps.addBatch();

        }

        //统一执行批处理的 sql 语句

        int[] rows = ps.executeBatch();

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        try {

            ps.close();

        }
    }
}

```

```
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
```

37、什么是 json

Json 是一种字符串数据格式，一般用于数据传输格式。

json 字符串中[]对应 JSONArray, {}对应 JSONObject

38、json 与 xml 的区别

(1).可读性方面。

JSON 和 XML 的数据可读性基本相同，JSON 和 XML 的可读性可谓不相上下，一边是建议的语法，一边是规范的标签形式，XML 可读性较好些。

(2).可扩展性方面。

XML 天生有很好的扩展性，JSON 当然也有，没有什么是 XML 能扩展，JSON 不能的。

(3).编码难度方面。

XML 有丰富的编码工具，比如 Dom4j、JDom 等，JSON 也有 json.org 提供的工具，但是 JSON 的编码明显比 XML 容易许多，即使不借助工具也能写出 JSON 的代码，可是要写好 XML 就不太容易了。

(4).解码难度方面。

XML 的解析得考虑子节点父节点，让人头昏眼花，而 JSON 的解析难度几乎为 0。这一点 XML 输的真是没话说。

(5).流行度方面。

XML 已经被业界广泛的使用，而 JSON 才刚刚开始，但是在 Ajax 这个特定的领域，未来的发展一定是 XML 让位于 JSON。到时 Ajax 应该变成 Ajaj(Asynchronous Javascript and JSON)

了。

(6).解析手段方面。

JSON 和 XML 同样拥有丰富的解析手段。

(7).数据体积方面。

JSON 相对于 XML 来讲，数据的体积小，传递的速度更快些。

(8).数据交互方面。

JSON 与 JavaScript 的交互更加方便，更容易解析处理，更好的数据交互。

(9).数据描述方面。

JSON 对数据的描述性比 XML 较差。

(10).传输速度方面。

JSON 的速度要远远快于 XML

39、XML 和 HTML 的区别？

1) 设计上的区别：XML 用来存储数据，重点在于数据本身，HTML 用来定义数据，重在数据的显示模式。

2) XML 可扩展性强，因为他本身就是可拓展性标记语言，可创建个性化的标记语言，提供更多数据操作。

3) XML 语法比 HTML 严格。

4) 起始标签和结束标签要匹配

5) 嵌套标签不能相互嵌套

6) 区分大小写

7) XML 属性必须放在引号中，HTML 可有可无。

8) XML 必须有相应值，但 HTML 可以有不带属性的属性名。

40、XML 文档定义有几种形式？它们之间有何本质区别？

1) 两种形式 dtd schema。

2) 本质区别:schema 本身是 xml 的，可以被 XML 解析器解析(这也是从 DTD 上发展 schema 的根本目的)，

七、框架部分

1、谈谈你对 Struts2 的理解。

1)struts2 是一个 MVC 框架，MVC 是一种开发模式，把业务逻辑代码与视图代码分离，通过控制器连接业务逻辑与视图。MVC 将应用程序分成了视图、模型、控制器三部分，使代码结构层次清晰、降低耦合度、代码重用性高。

2) 结合 Struts2 处理请求的工作流程加以说明：

客户端发送一个请求到服务器，tomcat 会接收这个请求， tomcat 会读取项目中的 web.xml 中的配置，判断请求是否符合 Struts2 过滤器 StrutsPrepareAndExecuteFilter 过滤的路径，如果符合会把这个请求交给 Struts2 处理，StrutsPrepareAndExecuteFilter 会分析请求路径，根据 Struts.xml 中的配置，请求路径匹配 package 标签的 namespace 属性加上 action 标签的 name 属性，跳转到对应的 action 类，默认执行 execute 方法，如果使用动态方法调用，会执行 action 类中的对应方法，方法执行完成后会返回一个字符串，这个字符串对应 Struts.xml 中 action 标签下的 result 标签 name 属性根据 result 标签的配置跳转到对应的 jsp 页面，在 jsp 页面中呈现数据，返回给客户端。

3) 结合 Struts2 优点说明：

- a、实现 MVC 模式，结构清晰,使开发者只关注业务逻辑的实现.
- b、有丰富的 tag 可以用 ,Struts 的标记库(Taglib)，如能灵活动用，则能大大提高开发效率
- c、页面导航使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。
- d、提供 Exception 处理机制 .
- e、数据库链接池管理
- f、支持 I18N

2、谈谈你对 Hibernate 的理解。

1) hibernate 是 ORM 框架, ORM 是对象关系映射,类—>表, 属性—>表中的列,对象—>表中的每一条数据,是为了解决面向对象与关系数据库之间互不匹配现象的技术。使我们编程的思想更面向对象了,不用去考虑关系型数据库。

2) hibernate 工作流程是: configuration 类读取并解析 hibernate.cfg.xml 配置文件,读取解析映射信息,创建 sessionFactory,打开 session,创建事务,持久化操作,关闭 session,整个应用停止关闭 sessionFactory。

3) 结合 hibernate 优点说明:

- a、程序更加面向对象,提高开发效率
- b、提高了生产率,不用写 SQL 语句
- c、hibernate 使用的是 hql,支持方言配置,方便数据库移植
- d、对 jdbc 代码进行封装,编程更简便了
- e、hibernate 是个轻量级框架,对代码无侵入性

3、你对 Spring 的理解。

1) Spring 是一个轻量级框架,设计原则是非侵入性的。Spring 核心是 IOC 容器,IOC 是一种编程思想,是一种架构艺术,是用来管理控制对象的生命周期和对象之间的关系,通过配置文件进行注入,很好的实现了对象与对象之间解耦。

2) IOC 工作原理: IOC 实现了工厂模式,通过读取 application.xml 配置文件中的<bean>标签的类,注入到 IOC 容器中,通过构造或 set 方法注入,产生 BeanFactory, BeanFactory 通过 getBean 方法获取对象。

3) Spring 还提供了另外一种重要编程思想 AOP, AOP 称为面向切面编程,可以动态的将主线业务逻辑代码与实现功能代码分离,为了更清晰的逻辑,可以让你的业务逻辑去关注自己本身的业务,而不去想一些其他的事情,将日志记录,性能统计,安全控制,事务处理,异常处理等代码从业务逻辑代码中划分出来,通过对这些行为的分离,我们希望能将它们独立到非指导业务逻辑的方法中,进而改变这些行为的时候不影响业务逻辑的代码。

4) Spring 提供了很多第三方框架的整合,如: hibernate、struts、mybatis、web service 等,使用 IOC 管理所有的 Java bean,这样可以让框架与框架之间偶尔度降低,方便项目的

管理，提高开发效率。

4、Struts2 优缺点

优点：

- 1) 实现 MVC 模式，结构清晰,使开发者只关注业务逻辑的实现.
- 2) 有丰富的 tag 可以用 ,Struts 的标记库(Taglib)，如能灵活动用，则能大大提高开发效率
- 3) 页面导航使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。
- 4) 提供 Exception 处理机制 .
- 5) 数据库链接池管理
- 6) 支持 I18N

缺点

- 1) 转到展示层时，需要配置 forward，每一次转到展示层，相信大多数都是直接转到 jsp，而涉及到转向，需要配置 forward，如果有十个展示层的 jsp，需要配置十次 struts，而且还不包括有时候目录、文件变更，需要重新修改 forward，注意，每次修改配置之后，要求重新部署整个项目，而 tomcate 这样的服务器，还必须重新启动服务器，如果业务变更复杂频繁的系统，这样的操作简单不可想象。现在就是这样，几十上百个人同时在线使用我们的系统，大家可以想象一下，我的烦恼有多大。
- 2) Struts 的 Action 必需是 thread—safe 方式，它仅仅允许一个实例去处理所有的请求。所以 action 用到的所有的资源都必需统一同步，这个就引起了线程安全的问题。
- 3) 测试不方便. Struts 的每个 Action 都同 Web 层耦合在一起，这样它的测试依赖于 Web 容器，单元测试也很难实现。不过有一个 Junit 的扩展工具 Struts TestCase 可以实现它的单元测试。
- 4) 类型的转换. Struts 的 FormBean 把所有的数据都作为 String 类型，它可以使用工具 Commons-Beanutils 进行类型转化。但它的转化都是在 Class 级别，而且转化的类型是不可配置的。类型转化时的错误信息返回给用户也是非常困难的。

5) 对 Servlet 的依赖性过强. Struts 处理 Action 时必须需要依赖 ServletRequest 和 ServletResponse, 所有它摆脱不了 Servlet 容器。

6) 前端表达式语言方面.Struts 集成了 JSTL, 所以它主要使用 JSTL 的表达式语言来获取数据。可是 JSTL 的表达式语言在 Collection 和索引属性方面处理显得很弱。

7) 对 Action 执行的控制困难. Struts 创建一个 Action, 如果想控制它的执行顺序将会非常困难。甚至你要重新去写 Servlet 来实现你的这个功能需求。

8) 对 Action 执行前和后的处理. Struts 处理 Action 的时候是基于 class 的 hierarchies, 很难在 action 处理前和后进行操作。

9) 对事件支持不够. 在 struts 中, 实际是一个表单 Form 对应一个 Action 类(或 DispatchAction), 换一句话说: 在 Struts 中实际是一个表单只能对应一个事件, struts 这种事件方式称为 application event, application event 和 component event 相比是一种粗粒度的事件。

5、说说 struts1 与 struts2 的区别。

1) 都是 MVC 的 WEB 框架,

2) struts1 的老牌框架, 应用很广泛, 有很好的群众基础, 使用它开发风险很小, 成本更低! struts2 虽然基于这个框架, 但是应用群众并不多, 相对不成熟, 未知的风险和变化很多, 开发人员相对不好招, 使用它开发项目的风险系数更大, 用人成本更高!

3) struts2 毕竟是站在前辈的基础设计出来, 它会改善和完善 struts1 中的一些缺陷, struts1 中一些悬而未决问题在 struts2 得到了解决。

4) struts1 的前端控制器是一个 Servlet, 名称为 ActionServlet, struts2 的前端控制器是一个 filter, 在 struts2.0 中叫 FilterDispatcher, 在 struts2.1 中叫 StrutsPrepareAndExecuteFilter。

5) struts1 的 action 需要继承 Action 类, struts2 的 action 可以不继承任何类; struts1 对同一个路径的所有请求共享一个 Action 实例, struts2 对同一个路径的每个请求分别使用一个独立 Action 实例对象, 所有对于 struts2 的 Action 不用考虑线程安全问题。

6) 在 struts1 中使用 formbean 封装请求参数, 在 struts2 中直接使用 action 的属性来封装请求参数。

7) struts1 中的多个业务方法放在一个 Action 中时(即继承 DispatchAction 时), 要么都校验, 要么都不校验; 对于 struts2, 可以指定只对某个方法进行校验, 当一个 Action 继承

了 `ActionSupport` 且在这个类中只编写了 `validateXxx()` 方法,那么则只对 `Xxx()` 方法进行校验。

8) 一个请求来了的执行流程进行分析, `struts2` 是自动支持分模块开发, 并可以不同模块设置不同的 url 前缀, 这是通过 `package` 的 `namespace` 来实现的; `struts2` 是支持多种类型的视图; `struts2` 的视图地址可以是动态的, 即视图的名称是支持变量方式的, 举例, 论坛发帖失败后回来还要传递 `boardid`。视图内容显示方面: 它的标签用 `ognl`, 要 `el` 强大很多, 在国际化方面支持分模块管理, 两个模块用到同样的 `key`, 对应不同的消息。

9) 与 `Struts1` 不同, `Struts2` 对用户的每一次请求都会创建一个 `Action`, 所以 `Struts2` 中的 `Action` 是线程安全的。

10) 给我印象最深刻的是: `struts` 配置文件中的 `redirect` 视图的 url 不能接受参数, 而 `struts2` 配置文件中的 `redirect` 视图可以接受参数。

6、struts2 的核心组件有哪些?

- 1) `StrutsPrepareAndExecuteFilter`, 整个 `struts2` 的调度中心, 它对请求进行过滤并决定 `struts2` 是否出来该请求。
- 2) `Struts.xml`: `struts2` 的应用配置文件, 它负责配置系统中用到的 `action`
- 3) `Action`: `struts2` 的动作执行单元实际处理用户的请求, 封装业务所需的数据
- 4) `Result`: `action` 运行后要转向下一个资源, 可以是视图也可以说其他的 `action`
- 5) `Struts` 标签: 用于页面上遍历后台传过来的数据

7、Struts2 的执行过程

- 1) 客户端发送一个请求到服务器, `tomcat` 会接收这个请求
- 2) `tomcat` 会读取项目中的 `web.xml` 中的配置
- 3) 判断请求是否符合 `Struts2` 过滤器 `StrutsPrepareAndExecuteFilter` 过滤的路径
- 4) 如果符合会把这个请求交给 `Struts2` 处理
- 5) `StrutsPrepareAndExecuteFilter` 会分析请求路径, 根据 `Struts.xml` 中的配置, 请求路径匹配 `package` 标签的 `namespace` 属性加上 `action` 标签的 `name` 属性, 跳转到对应的 `action` 类
- 6) 默认执行 `execute` 方法, 如果使用动态方法调用, 会执行 `action` 类中的对应方法, 方法执行完成后会返回一个字符串

- 7) 这个字符串对应 Struts.xml 中 action 标签下的 result 标签 name 属性
- 8) 根据 result 标签的配置跳转到对应的 jsp 页面，在 jsp 页面中呈现数据，返回给客户端

8、为什么要使用 struts2?

- 1) 开源
- 2) mvc 框架
- 3) 纯 pojo 的 action
- 4) 更好的标签特性
- 5) 易测性
- 6) 易扩展性

9、openSession 和 getCurrentSession

1) openSession 从字面上可以看得出来，是打开一个新的 session 对象，而且每次使用都是打开一个新的 session，假如连续使用多次，则获得的 session 不是同一个对象，并且使用完需要调用 close 方法关闭 session。

2) getCurrentSession ，从字面上可以看得出来，是获取当前上下文一个 session 对象，当第一次使用此方法时，会自动产生一个 session 对象，并且连续使用多次时，得到的 session 都是同一个对象，这就是与 openSession 的区别之一，简单而言，getCurrentSession 就是：如果有已经使用的，用旧的，如果没有，建新的。

注意：在实际开发中，往往使用 getCurrentSession 多，因为一般是处理同一个事务（即使用一个数据库的情况），所以在一般情况下比较少使用 openSession 或者说 openSession 是比较老旧的一套接口了；

对于 getCurrentSession 来说，有以下一些特点：

- 1.用途，界定事务边界
- 2.事务提交会自动 close，不需要像 openSession 一样自己调用 close 方法关闭 session
- 3.上下文配置（即在 hibernate.cfg.xml）中，需要配置：

```
<property name="current_session_context_class">thread</property>
```

10、拦截器的作用？拦截器和过滤器的区别？

拦截器是对调用的 action 起作用，它提供类一种机制可以使开发者可以定义在一个 action 执行的前后执行的代码。拦截器只能拦截 action，说白了拦截器其实就是一个 action 的功能块。拦截器可以抽象出一部分代码可以用来完善原来的 action。同时可以减轻代码冗余提高重用率。

过滤器是拦截用户请求，范围被拦截器大。

11、struts.xml 中 result 的 type 有哪些类型？

Dispatcher: struts2 默认的结果类型，把控制权转发给应用程序里的某个资源，不能把控制权转发给一个外部资源，若需要啊控制权重定向到一个外部资源，应该使用 redirect 结果类型。

Redirect 把响应重定向到另一个资源

RedirectAction 把响应重定向到另一个 Action

Freemarker、velocity、chain、httpheader、xslt、plainText、stream、json.

12、一般情况下，关系数据模型与对象模型之间有哪些匹配关系？

表对应类

记录对应对象

表的字段对应类的属性

13、hibernate 数据的三个状态

1) 瞬时状态（临时状态）：当 new 对象时候，处于瞬时状态（如果程序运行完了，该

对象会被垃圾回收)。

2) 持久状态 : 跟 session 有关, 就是持久状态, 持久状态的对象, 任何的修改, 都会影响到数据库中与之对应的数据。

3) 托管状态 (游离状态): 当 session 不在管理对象的时候, 脱离了 session 的管理, 处于托管状态的对象, 修改属性, 对数据库数据没有任何影响。

企业开发中, 使用 `saveOrUpdate(obj)`: 来替代 `save(obj)` 或 `update(obj)` 方法

避免因为状态的改变, 导致方法出错, `saveOrUpdate(obj)`

可以根据 obj 的状态, 来选择是 `save()` 还是 `update()`

14、Hibernate 中 load 和 get 的区别?

如果数据库中, 没有 `userId` 的对象, 如果通过 `get` 方法加载, 则返回的是一个 `Null`; 如果通过 `Load` 则返回一个代理对象, 如果后面代码调用 `user` 对象的某个属性, 会抛出 `objectNotFoundException`

`Load` 支持延迟加载, `get` 不支持。

15、Hibernate 的工作原理?

- 1) `configuration` 类读取并解析 `hibernate.cfg.xml` 配置文件
- 2) 读取解析映射信息, 创建 `sessionFactory`
- 3) 打开 `session`
- 4) 创建事务
- 5) 持久化操作
- 6) 提交事务
- 8) 关闭 `session`
- 9) 整个应用停止, 关闭 `sessionFactory`

16、hibernate 优缺点?

优点:

1) 对 jdbc 访问数据库的代码做了封装, 大大简化了数据访问层繁琐的重复性代码。

2) Hibernate 是一个基于 JDBC 的主流持久性框架, 是一个优秀的 ORM 实现, 他很大程度的简化 DAO 的编码工作, 程序更加面向对象, 提高开发效率。

- 3) 程序更加面向对象, 提高开发效率
- 4) 提高了生产率, 不用写 SQL 语句
- 5) hibernate 使用的是 hql, 支持方言配置, 方便数据库移植
- 6) hibernate 是个轻量级框架, 对代码无侵入性

缺点:

- 1) 效率比 JDBC 略差
- 2) 不适合批量操作
- 3) 对表的操作不够灵活

17、Hibernate 是如何延迟加载的?

hibernate 中存在一些查询方法, 在查询的时候并没有立刻访问数据库查询数据, 而是返回了一个空对象, 这个对象并不是 null 而是经过 new 的对象, 但对象中除了 ID 这种属性外其他属性都是 null, 当程序使用对象时 hibernate 才会真正的发送语句去查询数据库, 将返回的数据填充到对象的属性值。这种将推迟查询队形机制称为延迟加载。

为什么要用延迟加载:

- 1) 推迟的时间内由于没有数据加载可以节约内存空间, 提高内存的使用率。
- 2) 如果对象查询出来并没有使用, 那么延迟加载的对象根本没有访问数据库, 可以减少数据可得访问次数。

如何使用延迟加载

- 1) 在 hibernate 里面有一些方法自动支持延迟加载, 只要调用就可以使用。
- 2) 具有延迟加载的方法如下:

```
session.load();  
  
query.iterate();
```

关联映射属性加载, 属性名是 lazy, 如果查询不存在延迟加载就会抛异常

18、如果优化 Hibernate?

使用双向一对多关联，不使用单向一对多

灵活使用单向一对多

不使用一对一，用多对一取代

配置对象缓存，不适用集合缓存

一对多集合使用 bag，多对多使用 set

继承类使用显式多态

表字段要少，表关联不要怕多，有二级缓存。

19、什么是 ORM?

ORM 是对象关系映射,类—>表, 属性—>表中的列,对象—>表中的每一条数据,

是为了解决面向对象与关系数据库之间互不匹配现象的技术。

优点：使我们编程的思想更面向对象了，不用去考虑关系型数据库

20、Hibernate 的主键生成策略?

1) sequence,通知 Hibernate 框架，主键的值采用指定序列生成，然后插入数据库，主要用于

Oracle,DB2,不用程序员参与

```
<generator class="sequence">
```

```
<param name="sequence">foo_seq</param>// 必须加上
```

```
</generator>
```

2) identity,通知 hibernate 框架，主键值采用数据库自动增长机制，每次进行 save()操作，hibernate 都会根据（数据库）自增的方式，生成一个 id 值，不用程序员参与，主要用于

mySQL , SQLServer

```
<generator class="identity"></generator>
```

3) uuid(西方常用),hibernate 每次进行 save()操作，都会随机生成一个 32 的不重复的字符串，不用程序员去参与维护，PO 类的 Id 属性必须为 String

4) native 根据 dialect(方言)不同,来自动的选择 identity 或 sequence 智能选择。是企业中

常用的

5) assigned 不推荐使用，程序员要自己维护主键的 Id 值，当数据量很大时候很难维护

21、Hibernate 的级联操作

1) cascade 操作

all:所有情况下都进行级联操作，save-update 和 delete

save-update: 在进行 save()/update()/saveOrUpdate 时候进行级联操作

delete:在进行 delete 时候进行级联操作

all-delete-orphan :适合集合中删除，在返回的集合中执行 remove()操作

none:在任何情况下都不进行级联操作

2) inverse 属性的作用

是否放弃维护关联关系 true 放弃 false 不放弃

22、Hibernate 有哪 5 个核心接口？

Configuration 接口：配置 Hibernate，根据其启动 hibernate，创建 SessionFactory 对象；

SessionFactory 接口：初始化 Hibernate，充当数据存储源的代理，创建 session 对象，sessionFactory 是线程安全的，意味着它的同一个实例可以被应用的多个线程共享，是重量级、二级缓存；

Session 接口：负责保存、更新、删除、加载和查询对象，是线程不安全的，避免多个线程共享同一个 session，是轻量级、一级缓存；

Transaction 接口：管理事务；

Query 和 Criteria 接口：执行数据库的查询。

23、什么是重量级？什么是轻量级？

轻量级是指它的创建和销毁不需要消耗太多的资源，意味着可以在程序中经常创建和销毁 session 的对象；重量级意味不能随意的创建和销毁它的实例，会占用很多的资源。

24、谈谈 Spring 的 IOC 和 DI

首先想说说 IoC (Inversion of Control, 控制倒转)。这是 spring 的核心, 贯穿始终。所谓 IoC, 对于 spring 框架来说, 就是由 spring 来负责控制对象的生命周期和对象间的关系。这是什么意思呢, 举个简单的例子, 我们是如何找女朋友的? 常见的情况是, 我们到处去看哪里有长得漂亮身材又好的 mm, 然后打听她们的兴趣爱好、qq 号、电话号、ip 号、iq 号……, 想办法认识她们, 投其所好送其所要, 然后嘿嘿……这个过程是复杂深奥的, 我们必须自己设计和面对每个环节。传统的程序开发也是如此, 在一个对象中, 如果要使用另外的对象, 就必须得到它 (自己 new 一个, 或者从 JNDI 中查询一个), 使用完之后还要将对象销毁 (比如 Connection 等), 对象始终会和其他的接口或类耦合起来。

那么 IoC 是如何做的呢? 有点像通过婚介找女朋友, 在我和女朋友之间引入了一个第三者: 婚姻介绍所。婚介管理了很多男男女女资料, 我可以向婚介提出一个列表, 告诉它我想找个什么样的女朋友, 比如长得像李嘉欣, 身材像林熙雷, 唱歌像周杰伦, 速度像卡洛斯, 技术像齐达内之类的, 然后婚介就会按照我们的要求, 提供一个 mm, 我们只需要去和她谈恋爱、结婚就行了。简单明了, 如果婚介给我们的人选不符合要求, 我们就会抛出异常。整个过程不再由我自己控制, 而是有婚介这样一个类似容器的机构来控制。Spring 所倡导的开发方式就是如此, 所有的类都会在 spring 容器中登记, 告诉 spring 你是个什么东西, 你需要什么东西, 然后 spring 会在系统运行到适当的时候, 把你想要的东西主动给你, 同时也把你交给其他需要你的东西。所有的类的创建、销毁都由 spring 来控制, 也就是说控制对象生存周期的不再是引用它的对象, 而是 spring。对于某个具体的对象而言, 以前是它控制其他对象, 现在是所有对象都被 spring 控制, 所以这叫控制反转。如果你还不明白的话, 我决定放弃。

IoC 的一个重点是在系统运行中, 动态的向某个对象提供它所需要的其他对象。这一点是通过 DI (Dependency Injection, 依赖注入) 来实现的。比如对象 A 需要操作数据库, 以前我们总是要在 A 中自己编写代码来获得一个 Connection 对象, 有了 spring 我们就只需要告诉 spring, A 中需要一个 Connection, 至于这个 Connection 怎么构造, 何时构造, A 不需要知道。在系统运行时, spring 会在适当的时候制造一个 Connection, 然后像打针一样, 注射到 A 当中, 这样就完成了对各个对象之间关系的控制。A 需要依赖 Connection 才能正常运行, 而这个 Connection 是由 spring 注入到 A 中的, 依赖注入的名字就这么来的。那么 DI 是如何实现的呢? Java 1.3 之后一个重要特征是反射 (reflection), 它允许程序在运行的时

候动态的生成对象、执行对象的方法、改变对象的属性，spring 就是通过反射来实现注入的。

总结:IOC 是用来管理控制对象的生命周期和对象之间的关系,通过配置文件进行注入,很好的实现了对象与对象之间解耦。

IOC 工作原理:

IOC 实现了工厂模式,通过读取 application.xml 配置文件中的<bean>标签的类,注入到 IOC 容器中,通过构造或 set 方法注入,产生 BeanFactory,BeanFactory 通过 getBean 方法获取对象。

25、什么是 AOP?

Aspect Oriented Programming (面向方面编程)

OOP 是面向对象编程,AOP 是在 OOP 基础之上一种更高级的设计思想.

OOP 和 AOP 之间也存在一些区别,OOP 侧重于对象的提取和封装.

AOP 侧重于方面组件,方面组件可以理解成封装了通用功能的组件,

方面组件可以通过配置方式灵活的切入到某一批目标对象方法上.

aop 是面向切面编程,可以动态的将主线业务逻辑代码与实现功能代码分离,没有侵入性.为了更清晰的逻辑,可以让你的业务逻辑去关注自己本身的业务,而不去想一些其他的事情,将日志记录,性能统计,安全控制,事务处理,异常处理等代码从业务逻辑代码中划分出来,通过对这些行为的分离,我们希望能将它们独立到非指导业务逻辑的方法中,进而改变这些行为的时候不影响业务逻辑的代码。

26、Spring 的通知类型有哪些?

通知决定了方面组件功能在目标对象方法上执行的时机.

Spring 框架提供了以下 5 中类型通知.

1) .前置通知<aop:before>

方面功能在目标方法之前调用.

2) 后置通知<aop:afterReturning>

方面功能在目标方法之后调用.(如果目标方法抛出异常则不会执行方面功能)

3) 最终通知<aop:after>

方面功能在目标方法之后调用.(目标方法有无异常都会执行方面功能)

4) 环绕通知<aop:around>

方面功能在目标方法之前和之后调用.

5) 异常通知<aop:afterThrowing>

方面功能在目标方法抛出异常之后调用.

27、什么是 MVC?

MVC 是一种开发模式，把业务逻辑代码与视图代码分离，通过控制器连接业务逻辑与视图。

MVC 将一个应用程序分为三个部分：

Model: 程序开发中的业务逻辑，通常用 `JavaBean` 实现

View: 程序开发中程序用户数据，通常用 `jsp` 表示

Controller: 程序开发中连接业务逻辑与视图的控制器，通常用 `servlet` 表示

优点:

1)代码结构层次清晰

2)就是低耦合

3)重用性高

缺点:

一个应用程序分成了三个部分开发，增加开发工作量。

28、hibernate 查询方式有多少种?

主键查找: `session.get()`或 `load()`

hql 查询: `session.createQuery("hql")`

sql 查询: `session.createSQLQuery("sql")`

criteria 查询 (QBC) : `session.createCriteria()`

29、spring 中 Bean 的 scope

目前，scope 的取值有 5 种。

在 Spring 2.0 之前，有 `singleton` 和 `prototype` 两种

在 Spring 2.0 之后，为支持 web 应用的 ApplicationContext，推出另外三种：request，session 和 global session 类型

singleton：在 IOC 容器中只存在一个实例，在初始化 IOC 容器的时候就被创建

prototype：在 IOC 容器中只存在多个实例，在使用到此对象是被创建

request：使用在 web 应用中，相当于 Servlet 中的 Request

session：使用在 web 应用中，相当于 Servlet 中的 Session

global session：使用在 web 应用中，相当于 Servlet 中的 application

30、SSH 对应 MVC 的哪些层

Struts2：用于处理请求，调用业务逻辑

Hibernate：用于操作数据库，做持久化操作

Spring：用于管理对象，处理对象与对象之间的关系

MVC 是一种开发模式，模型、视图、控制，与 SSH 框架的作用是两个东西，不能相互对应。

31、spring 注入方式有几种

Spring 四种依赖注入方式，常用 1、2 种，

- 1) Set 方法注入
- 2) 构造器注入
- 3) 静态工厂方法注入
- 4) 实例工厂方法注入

32、spring mvc 的工作原理

- 1) 客户端发送一个请求到服务器，tomcat 会接收这个请求
- 2) tomcat 会读取项目中的 web.xml 中的配置
- 3) 判断请求是否符合 spring mvc 核心 servlet(DispatcherServlet)的请求路径
- 4) 如果符合会把请求提交到 DispatcherServlet
- 5) 根据请求的 URL 由 DispatcherServlet 控制器查询一个或多个 HandlerMapping，找到处理请求的 Controller

- 6) DispatcherServlet 将请求提交到 Controller
- 7) Controller 调用业务逻辑处理后，返回 ModelAndView
- 8) DispatcherServlet 查询一个或多个 ViewResolver 视图解析器，找到 ModelAndView 指定的视图
- 9) 视图负责将结果显示到客户端

33、mybatis 的工作原理

- 1) SqlSessionFactoryBuilder 类读取 mybatis-config.xml 配置文件
- 2) 读取解析 Mapper.xml 映射文件，创建 SqlSessionFactory
- 3) 打开 SqlSession
- 4) 持久化操作
- 5) 提交事务
- 6) 关闭 session

34、spring mvc 与 Struts2 的区别

- 1) struts2 框架是类级别的拦截，每次来了请求就创建一个 Action，然后调用 setter getter 方法把 request 中的数据注入

struts2 实际上是通过 setter getter 方法与 request 打交道的

struts2 中，一个 Action 对象对应一个 request 上下文

SpringMVC 不同，SpringMVC 是方法级别的拦截，拦截到方法后根据参数上的注解，把 request 数据注入进去

在 SpringMVC 中，一个方法对应一个 request 上下文

- 2) 由于上边原因：SpringMVC 的 Controller 是单实例的，方法之间基本上独立的，独享 request response 数据，请求数据通过参数获取，处理结果通过 ModelAndView 交回给框架

方法之间不共享变量

而 struts2 的 Action 是多实例的，虽然方法之间也是独立的，但其所有 Action 变量是共享的，这不会影响程序运行，却给我们编码 读程序时带来麻烦，每次请求就创建一个新的 Action，一个 action 对象对应一个 request 上下文

- 3) 由于 Struts2 需要针对每个 request 进行封装，把 request, session 等 servlet 生命周期的变量封装成一个一个 Map，供给每个 Action 使用，并保证线程安全，所以在原则上，是比较耗费内存的。
- 4) 拦截器实现机制上，Struts2 有以自己的 interceptor 机制，SpringMVC 用的是独立的 AOP 方式，这样导致 Struts2 的配置文件量还是比 SpringMVC 大。
- 5) SpringMVC 的入口是 servlet，而 Struts2 是 filter
- 6) spring MVC 和 Spring 是无缝的。从这个项目的管理和安全上也比 Struts2 高
- 7) 设计思想上，Struts2 更加符合 OOP 的编程思想，SpringMVC 就比较谨慎，在 servlet 上扩展。
- 8) SpringMVC 开发效率和性能高于 Struts2。
- 9) SpringMVC 可以认为已经 100%零配置。

35、mybatis 与 hibernate 的区别

第一方面：开发速度的对比

就开发速度而言，Hibernate 的真正掌握要比 Mybatis 来得难些。Hibernate 门槛较高，Mybatis 框架相对简单很容易上手，但也相对简陋些。个人觉得要用好 Mybatis 还是首先要先理解好 Hibernate。

第二方面：开发工作量的对比

Hibernate 和 MyBatis 都有相应的代码生成工具。可以生成简单基本的 DAO 层方法。针对高级查询，Mybatis 需要手动编写 SQL 语句，以及 resultMap。而 Hibernate 有良好的映射机制，开发者无需关心 SQL 的生成与结果映射，可以更专注于业务流程。

第三方面：sql 优化方面

Hibernate 的查询会将表中的所有字段查询出来，这一点会有性能消耗。Hibernate 也可以自己写 SQL 来指定需要查询的字段，但这样就破坏了 Hibernate 开发的简洁性。而 Mybatis 的 SQL 是手动编写的，所以可以按需求指定查询的字段。

第四方面：对象管理的对比

Hibernate 是完整的对象/关系映射解决方案，它提供了对象状态管理（state management）的功能，使开发者不再需要理会底层数据库系统的细节。也就是说，相对于常见的 JDBC/SQL 持久层方案中需要管理 SQL 语句，Hibernate 采用了更自然的面向对象的视角来持久化 Java 应用中的数据。

换句话说，使用 Hibernate 的开发者应该总是关注对象的状态（state），不必考虑 SQL 语句的执行。这部分细节已经由 Hibernate 掌管妥当，只有开发者在进行系统性能调优的时候才需要进行了解。而 MyBatis 在这一块没有文档说明，用户需要对对象自己进行详细的管理。

第五方面：缓存机制

Hibernate 有一级缓存是 Session 缓存，二级缓存是 SessionFactory 级的缓存。

Mybatis 也有查询缓存与二级缓存，

因为 Hibernate 对查询对象有着良好的管理机制，用户无需关心 SQL，所用 hibernate 的缓存机制会比 mybatis 完善。

相同点：Hibernate 与 MyBatis 都可以是通过 SessionFactoryBuider 由 XML 配置文件生成 SessionFactory，然后由 SessionFactory 生成 Session，最后由 Session 来开启执行事务和 SQL 语句。其中 SessionFactoryBuider，SessionFactory，Session 的生命周期都是差不多的。

Hibernate 和 MyBatis 都支持 JDBC 和 JTA 事务处理。

Mybatis 优势

MyBatis 可以进行更为细致的 SQL 优化，可以减少查询字段。

MyBatis 容易掌握，而 Hibernate 门槛较高。

Hibernate 优势

Hibernate 的 DAO 层开发比 MyBatis 简单，Mybatis 需要维护 SQL 和结果映射。

Hibernate 对对象的维护和缓存要比 MyBatis 好，对增删改查的对象的维护要方便。

Hibernate 数据库移植性很好，MyBatis 的数据库移植性不好，不同的数据库需要写不同 SQL。

Hibernate 有更好的二级缓存机制，可以使用第三方缓存。MyBatis 本身提供的缓存机制不佳。

36、spring 和 springmvc 的区别

Spring 是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

这两个核心，可以单独用于任何应用。

spring mvc 类似于 struts 的一个 MVC 开框架，其实都是属于 spring，spring mvc 需要有 spring 的架包作为支撑才能跑起来。

37、java 中的悲观锁和乐观锁

业务逻辑的实现过程中，往往需要保证数据访问的排他性。在某个时间点的数据进行处理，而不希望在进行过程中数据再发生变化。此时，我们就需要通过一些机制来保证这些数据在某个操作过程中不会被外界修改，这样的机制称为“锁”。

悲观锁，正如其名，它指的是对数据被外界修改持保守态度，因此，在整个数据处理过程中，将数据处于锁定状态。悲观锁的实现，往往依靠数据库提供的锁机制，也只有数据库层面才能保证数据的锁定状态。

乐观相对悲观锁而言，乐观锁机制采取了更加宽松的加锁机制。悲观锁大多数情况下依靠数据库的锁机制实现，以保证操作最大程度的独占性。但随之而来的就是数据库性能的大量开销，特别是对长事务而言，这样的开销往往无法承受。乐观锁，大多是基于数据版本记录机制实现。

八、设计模式部分

请写出你所知道的设计模式？

设计模式主要分三个类型:创建型、结构型和行为型。

其中创建型有：

一、Singleton，单例模式：保证一个类只有一个实例，并提供一个访问它的全局访问点

二、Abstract Factory，抽象工厂：提供一个创建一系列相关或相互依赖对象的接口，而无须指定它们的具体类。

三、Factory Method，工厂方法：定义一个用于创建对象的接口，让子类决定实例化哪一个类，Factory Method 使一个类的实例化延迟到了子类。

四、Builder，建造模式：将一个复杂对象的构建与他的表示相分离，使得同样的构建过程可以创建不同的表示。

五、Prototype，原型模式：用原型实例指定创建对象的种类，并且通过拷贝这些原型来创建新的对象。

行为型有：

六、Iterator，迭代器模式：提供一个方法顺序访问一个聚合对象的各个元素，而又不需要暴露该对象的内部表示。

七、Observer，观察者模式：定义对象间一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知自动更新。

八、Template Method，模板方法：定义一个操作中的算法的骨架，而将一些步骤延迟到子类中，TemplateMethod 使得子类可以不改变一个算法的结构即可以重定义该算法得某些特定步骤。

九、Command，命令模式：将一个请求封装为一个对象，从而使你可以用不同的请求对客户进行参数化，对请求排队和记录请求日志，以及支持可撤销的操作。

十、State，状态模式：允许对象在其内部状态改变时改变他的行为。对象看起来似乎改变了他的类。

十一、Strategy，策略模式：定义一系列的算法，把他们一个个封装起来，并使他们

可以互相替换，本模式使得算法可以独立于使用它们的客户。

十二、Chain of Responsibility，职责链模式：使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系

十三、Mediator，中介者模式：用一个中介对象封装一些列的对象交互。

十四、Visitor，访问者模式：表示一个作用于某对象结构中的各元素的操作，它使你可以在不改变各元素类的前提下定义作用于这个元素的新操作。

十五、Interpreter，解释器模式：给定一个语言，定义他的文法的一个表示，并定义一个解释器，这个解释器使用该表示来解释语言中的句子。

十六、Memento，备忘录模式：在不破坏对象的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。

结构型有：

十七、Composite，组合模式：将对象组合成树形结构以表示部分整体的关系，Composite 使得用户对单个对象和组合对象的使用具有一致性。

十八、Facade，外观模式：为子系统的一组接口提供一致的界面，Facade 提供了一高层接口，这个接口使得子系统更容易使用。

十九、Proxy，代理模式：为其他对象提供一种代理以控制对这个对象的访问

二十、Adapter,适配器模式：将一类的接口转换成客户希望的另外一个接口，Adapter 模式使得原本由于接口不兼容而不能一起工作那些类可以一起工作。

二十一、Decorator，装饰模式：动态地给一个对象增加一些额外的职责，就增加的功能来说，Decorator 模式相比生成子类更加灵活。

二十二、Bridge，桥模式：将抽象部分与它的实现部分相分离，使他们可以独立的变化。

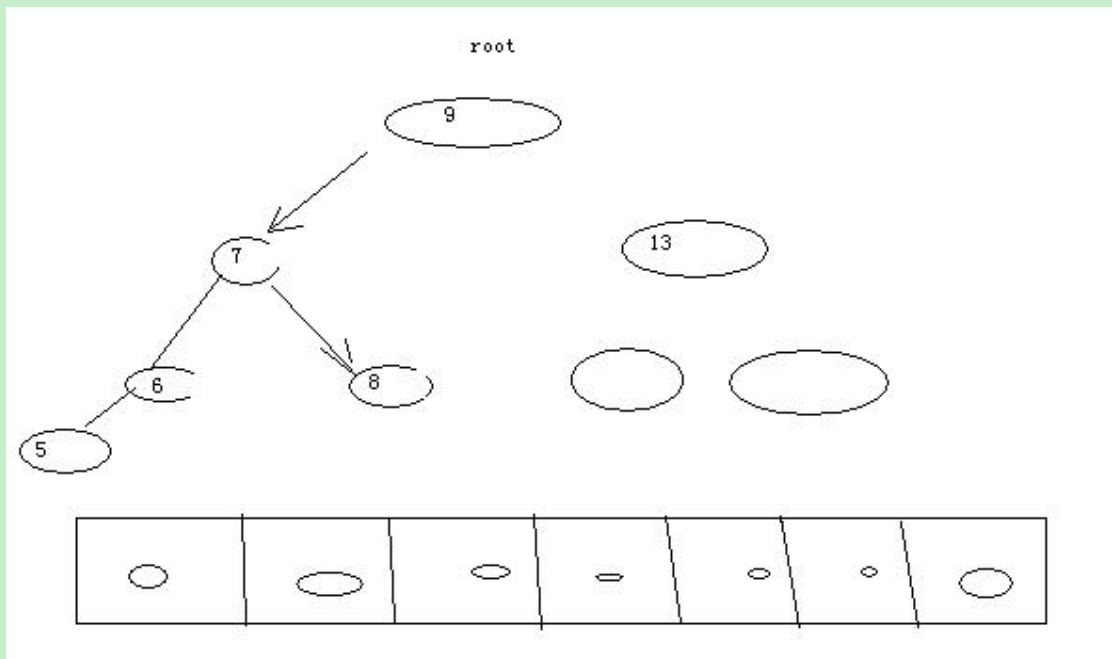
二十三、Flyweight，享元模式

九、算法部分

1、说明生活中遇到的二叉树，用 java 实现二叉树

这是组合设计模式。

我有很多个(假设 10 万个)数据要保存起来，以后还需要从保存的这些数据中检索是否存在某个数据，（我想说出二叉树的好处，该怎么说呢？那就是说别人的缺点），假如存在数组中，那么，碰巧要找的数字位于 99999 那个地方，那查找的速度将很慢，因为要从第 1 个依次往后取，取出来后进行比较。平衡二叉树（构建平衡二叉树需要先排序，我们这里就不作考虑了）可以很好地解决这个问题，但二叉树的遍历（前序，中序，后序）效率要比数组低很多，原理如下图：



代码如下：

```
package com.huawei.interview;
```

```
public class Node {
    public int value;
    public Node left;
    public Node right;

    public void store(int value)
    {
        if(value<this.value)
        {
            if(left == null)
```

```

        {
            left = new Node();
            left.value=value;
        }
        else
        {
            left.store(value);
        }
    }
    else if(value>this.value)
    {
        if(right == null)
        {
            right = new Node();
            right.value=value;
        }
        else
        {
            right.store(value);
        }
    }
}

```

```

public boolean find(int value)
{
    System.out.println("happen " + this.value);
    if(value == this.value)
    {
        return true;
    }
}

```

```

        else if(value>this.value)
        {
            if(right == null) return false;

            return right.find(value);
        }else
        {
            if(left == null) return false;

            return left.find(value);
        }
    }
}

```

```

public void preList()
{
    System.out.print(this.value + ",");

    if(left!=null) left.preList();

    if(right!=null) right.preList();
}

```

```

public void middleList()
{
    if(left!=null) left.preList();

    System.out.print(this.value + ",");

    if(right!=null) right.preList();
}

```

```

public void afterList()
{
    if(left!=null) left.preList();

    if(right!=null) right.preList();

    System.out.print(this.value + ",");
}

```



```

    }

    public static void main(String [] args)
    {
        int [] data = new int[20];

        for(int i=0;i<data.length;i++)
        {
            data[i] = (int)(Math.random()*100) + 1;

            System.out.print(data[i] + ",");

        }

        System.out.println();

        Node root = new Node();

        root.value = data[0];

        for(int i=1;i<data.length;i++)
        {
            root.store(data[i]);

        }

        root.find(data[19]);

        root.preList();

        System.out.println();

        root.middleList();

        System.out.println();

        root.afterList();

    }
}

```

-----又一次临场写的代码-----

```

import java.util.Arrays;

import java.util.Iterator;

```

```

public class Node {

    private Node left;

    private Node right;

    private int value;

    //private int num;

    public Node(int value){

        this.value = value;

    }

    public void add(int value){

        if(value > this.value)

        {

            if(right != null)

                right.add(value);

            else

            {

                Node node = new Node(value);

                right = node;

            }

        }

        else{

            if(left != null)

                left.add(value);

            else

            {

                Node node = new Node(value);

                left = node;

            }

        }

    }

}

```

```

    }
}

public boolean find(int value){
    if(value == this.value) return true;
    else if(value > this.value){
        if(right == null) return false;
        else return right.find(value);
    }else{
        if(left == null) return false;
        else return left.find(value);
    }
}

public void display(){
    System.out.println(value);
    if(left != null) left.display();
    if(right != null) right.display();
}

/*public Iterator iterator(){

}*/

public static void main(String[] args){
    int[] values = new int[8];
    for(int i=0;i<8;i++){
        int num = (int)(Math.random() * 15);

```

```

        //System.out.println(num);

        //if(Arrays.binarySearch(values, num)<0)
        if(!contains(values,num))

            values[i] = num;

        else

            i--;

    }

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

    Node root  = new Node(values[0]);

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

        root.add(values[i]);

    }

    System.out.println(root.find(13));

    root.display();

}

public static boolean contains(int [] arr, int value){

    int i = 0;

    for(;i<arr.length;i++){

        if(arr[i] == value) return true;

    }

    return false;

}

```

**2、第 1 个人 10，第 2 个比第 1 个人大 2 岁，依次递推，
请用递归方式计算出第 8 个人多大？**

```
package cn.itcast;

import java.util.Date;

public class A1 {

    public static void main(String [] args)
    {
        System.out.println(computeAge(8));
    }

    public static int computeAge(int n)
    {
        if(n==1) return 10;
        return computeAge(n-1) + 2;
    }
}

public static void toBinary(int n,StringBuffer result)
{
    if(n/2 != 0)
        toBinary(n/2,result);
    result.append(n%2);
}
```

3、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。

```
public class QuickSort {  
    /**  
     * 快速排序  
     * @param strDate  
     * @param left  
     * @param right  
     */  
    public void quickSort(String[] strDate,int left,int right){  
        String middle,tempDate;  
        int i,j;  
        i=left;  
        j=right;  
        middle=strDate[(i+j)/2];  
        do{  
            while(strDate[i].compareTo(middle)<0&& i<right)  
                i++; //找出左边比中间值大的数  
            while(strDate[j].compareTo(middle)>0&& j>left)  
                j--; //找出右边比中间值小的数  
            if(i<=j){ //将左边大的数和右边小的数进行替换  
                tempDate=strDate[i];  
                strDate[i]=strDate[j];  
                strDate[j]=tempDate;  
                i++;  
                j--;  
            }  
        }while(i<=j); //当两者交错时停止
```

```

if(i<right){
quickSort(strDate,i,right);//从
}
if(j>left){
quickSort(strDate,left,j);
}
}
}
/**
 * @param args
 */
public static void main(String[] args){
String[] strVoid=new String[]{"11","66","22","0","55","22","0","32"};
QuickSort sort=new QuickSort();
sort.quickSort(strVoid,0,strVoid.length-1);
for(int i=0;i<strVoid.length;i++){
System.out.println(strVoid[i]+" ");
}
}
}
}

```

4、金额转换，阿拉伯数字的金额转换成中国传统的形式如：

（¥1011）—>（一千零一拾一元整）输出。

去零的代码：

```

return sb.reverse().toString().replaceAll(" 零 [拾佰仟]"," 零").replaceAll(" 零 + 万 "," 万")
.replaceAll("零+元","元").replaceAll("零+","零");

```

```

public class RenMingBi {

```

```

/**
 * @param args add by zxx ,Nov 29, 2008
 */

private static final char[] data = new char[]{
    '零','壹','贰','叁','肆','伍','陆','柒','捌','玖'
};

private static final char[] units = new char[]{
    '元','拾','佰','仟','万','拾','佰','仟','亿'
};

public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println(
        convert(135689123));
}

public static String convert(int money)
{
    StringBuffer sbf = new StringBuffer();
    int unit = 0;
    while(money!=0)
    {
        sbf.insert(0,units[unit++]);
        int number = money%10;
        sbf.insert(0, data[number]);
        money /= 10;
    }

    return sbf.toString();
}

```



```
}
```

5、从类似如下的文本文件中读取出所有的姓名，并打印出重复的姓名和重复的次数，并按重复次数排序：

```
1,张三,28
2,李四,35
3,张三,28
4,王五,35
5,张三,28
6,李四,35
7,赵六,28
8,田七,35
```

程序代码如下（答题要博得用人单位的喜欢，包名用该公司，面试前就提前查好该公司的网址，如果查不到，现场问也是可以的。还要加上实现思路的注释）：

```
package com.huawei.interview;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.TreeSet;
```

```

public class GetNameTest {

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

        // TODO Auto-generated method stub

        //InputStream ips =
        GetNameTest.class.getResourceAsStream("/com/huawei/interview/info.txt");

        //用上一行注释的代码和下一行的代码都可以，因为 info.txt 与 GetNameTest
        类在同一包下面，所以，可以用下面的相对路径形式

```

```

        Map results = new HashMap();

        InputStream ips = GetNameTest.class.getResourceAsStream("info.txt");
        BufferedReader in = new BufferedReader(new InputStreamReader(ips));
        String line = null;
        try {
            while((line=in.readLine())!=null)
            {
                dealLine(line,results);
            }
            sortResults(results);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    static class User
    {

```

```

public String name;

public Integer value;

public User(String name,Integer value)
{
    this.name = name;
    this.value = value;
}

```

@Override

```

public boolean equals(Object obj) {
    // TODO Auto-generated method stub

```

方法。

//下面的代码没有执行，说明往 `treeSet` 中增加数据时，不会使用到 `equals`

```

        boolean result = super.equals(obj);

        System.out.println(result);

        return result;
    }
}

```

```

private static void sortResults(Map results) {

```

```

    // TODO Auto-generated method stub

```

```

    TreeSet sortedResults = new TreeSet(

```

```

        new Comparator(){

```

```

            public int compare(Object o1, Object o2) {

```

```

                // TODO Auto-generated method stub

```

```

                User user1 = (User)o1;

```

```

                User user2 = (User)o2;

```

```

                /*如果 compareTo 返回结果 0，则认为两个对象相等，新的

```

对象不会增加到集合中去

* 所以，不能直接用下面的代码，否则，那些个数相同的其他姓名就打印不出来。

```
        */

        //return user1.value-user2.value;

        //return

user1.value<user2.value?-1:user1.value==user2.value?0:1;

        if(user1.value<user2.value)
        {
            return -1;
        }else if(user1.value>user2.value)
        {
            return 1;
        }else
        {
            return user1.name.compareTo(user2.name);
        }
    }

}

);

Iterator iterator = results.keySet().iterator();
while(iterator.hasNext())
{
    String name = (String)iterator.next();
    Integer value = (Integer)results.get(name);
    if(value > 1)
    {
        sortedResults.add(new User(name,value));
    }
}
```

```

    }

    printResults(sortedResults);
}

private static void printResults(TreeSet sortedResults)
{
    Iterator iterator = sortedResults.iterator();
    while(iterator.hasNext())
    {
        User user = (User)iterator.next();

        System.out.println(user.name + ":" + user.value);
    }
}

public static void dealLine(String line, Map map)
{
    if(!"".equals(line.trim()))
    {
        String [] results = line.split(",");

        if(results.length == 3)
        {
            String name = results[1];

            Integer value = (Integer)map.get(name);

            if(value == null) value = 0;

            map.put(name, value + 1);
        }
    }
}
}

```

6、写一个 Singleton 出来。

第一种：饱汉模式

```
public class Singleton {  
    private Singleton(){  
          
    }  
  
    //实例化放在静态代码块里可提高程序的执行效率，但也可能更占用空间  
    private final static Singleton instance = new Singleton();  
  
    public static Singleton getInstance(){  
        return instance;  
    }  
}
```

第二种：饥汉模式

```
public class Singleton {  
    private Singleton(){}  
  
    private static instance = null;//new Singleton();  
  
    public static synchronized Singleton getInstance(){  
        if(instance == null)  
            instance = new Singleton();  
        return instance;  
    }  
}
```

第三种：用枚举

```

public enum SingleTon{

    ONE;

}

```

第三：更实际的应用（在什么情况用单例）

```

public class SequenceGenerator{

    //下面是该类自身的业务功能代码

    private int count = 0;

    public synchronized int getSequence(){

        ++count;

    }

    //下面是把该类变成单例的代码

    private SequenceGenerator(){}

    private final static instance = new SequenceGenerator();

    public static SingleTon getInstance(){

        return instance;

    }

}

```

第四：

```

public class MemoryDao

{

    private HashMap map = new HashMap();

    public void add(Student stu1){

        map.put(SequenceGenerator.getInstance().getSequence(),stu1);
    }
}

```

```
}
```

```
//把 MemoryDao 变成单例
```

```
}
```

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

一般 Singleton 模式通常有几种形式:

第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例化，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {  
    private Singleton(){}  
    //在自己内部定义自己一个实例，是不是很奇怪？  
    //注意这是 private 只供内部调用  
    private static Singleton instance = new Singleton();  
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问  
    public static Singleton getInstance() {  
        return instance;  
    }  
}
```

第二种形式:

```
public class Singleton {  
    private static Singleton instance = null;  
    public static synchronized Singleton getInstance() {  
        //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
```



```

//使用时生成实例，提高了效率！

if (instance==null)

    instance=new Singleton();

    return instance;

}

}

```

其他形式:

定义一个类，它的构造函数为 `private` 的，所有方法为 `static` 的。

一般认为第一种形式要更加安全些

7、古典问题：有一对兔子，从出生后第 3 个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少？

这是一个菲波拉契数列问题

```

public class lianxi01 {

    public static void main(String[] args) {

        System.out.println("第 1 个月的兔子对数:  1");

        System.out.println("第 2 个月的兔子对数:  1");

        int f1 = 1, f2 = 1, f, M=24;

        for(int i=3; i<=M; i++) {

            f = f2;

            f2 = f1 + f2;

            f1 = f;

            System.out.println("第" + i +"个月的兔子对数: "+f2);

        }

    }

}

```

```
}
```

8、简单的说个递归

N 阶乘:

```
public int factorial(int m)
{
    if (m < 0)
        return 0;
    else if ( m == 1)
        return 1;
    else if (m > 1)
        return m * factorial(m-1);
}
```

9、什么是平衡二叉树

平衡二叉树是一棵空树或它的左右两个子树的高度差的绝对值不超过 1，并且左右两个子树都是一棵平衡二叉树。

10、怎么判断二叉树是否有环

在遍历二叉树时，能循环到起点指针称为有环。