[JAVA知识点总结 - 1 -](#_Toc477889188)

[================== 一、Java框架知 =================== - 1 -](#_Toc477889189)

[**1.1 Spring的注入方式？** - 1 -](#_Toc477889190)

[**1.2 简单说下Spring IOC 的实现原理？** - 1 -](#_Toc477889191)

[**1.3 分别描述一下Spring的控制和反转？** - 2 -](#_Toc477889192)

[**1.4 简单说下String Aop的实现原理？** - 2 -](#_Toc477889193)

[**1.5 简述Spring中的事务传播行为(事务管理)和隔离级别？** - 3 -](#_Toc477889194)

[**1.6 事务有哪些特性？Spring的事务管理有几种实现方式，如何实现？** - 3 -](#_Toc477889195)

[**1.7 怎样开启Spring 的注解装配？注解装配的作用以及优缺点?** - 4 -](#_Toc477889196)

[**1.8 解释Spring支持的几种Bean的作用域?** - 4 -](#_Toc477889197)

[**1.9 Spring中的核心类都有哪些？各有什么作用？** - 4 -](#_Toc477889198)

[**1.10 简单介绍Spring MVC框架？** - 4 -](#_Toc477889199)

[**1.11 Spring MVC原理、流程以及接口解释？** - 5 -](#_Toc477889200)

[**1.12 Struts1的执行原理？** - 5 -](#_Toc477889201)

[**1.13 Struts2的执行流程、原理？** - 6 -](#_Toc477889202)

[**1.14 Struts1与Struts2的区别与比较？** - 6 -](#_Toc477889203)

[**1.15 简述actionfrom和action在Struts中扮演的角色？** - 7 -](#_Toc477889204)

[**1.16 Hibernate和ibatis的区别？** - 7 -](#_Toc477889205)

[**1.17 Hibernate和mybatis区别？** - 7 -](#_Toc477889206)

[**1.18 Hibernate的工作原理？** - 7 -](#_Toc477889207)

[**1.19 Hibernate的缓存等级及其特点？** - 8 -](#_Toc477889208)

[**1.20 Hibernate中load方法和get方法的区别？** - 8 -](#_Toc477889209)

[**1.21 Hibernate中不看数据库，不看xml文件，不看查询语句，怎么样才能知道表结构？** - 9 -](#_Toc477889210)

[**1.22 Ibatis中#与$的区别？** - 9 -](#_Toc477889211)

[**1.23 MyBatis的优缺点？** - 9 -](#_Toc477889212)

[**1.24 MVC各部分有哪些技术实现？是如何实现的？** - 9 -](#_Toc477889213)

[**1.25 Hibernate中怎样实现类之间的关系？** - 9 -](#_Toc477889214)

[**1.26 ORM是什么？它有什么特点？** - 10 -](#_Toc477889215)

[=============== 二、数据库 ================== - 11 -](#_Toc477889216)

[**2.1 总结一下JDBC？** - 11 -](#_Toc477889217)

[**2.2 JDBC中的PreparedStatement相比Statement的好处？** - 11 -](#_Toc477889218)

[**2.3 数据库三个范式是什么？** - 11 -](#_Toc477889219)

[**2.4 简要介绍数据库中的存储过程以及它的优缺点？** - 11 -](#_Toc477889220)

[**2.5 简要介绍数据库中触发器?** - 12 -](#_Toc477889221)

[**2.6 简说说你知道的数据库中常用函数?** - 12 -](#_Toc477889222)

[**2.7 说说数据连接池的工作机制？** - 13 -](#_Toc477889223)

[**2.8 使用过哪些数据库？各自的特点及应用范围？** - 13 -](#_Toc477889224)

[**2.9 给出几条具体的sql语句的优化建议？至少三条哦** - 13 -](#_Toc477889225)

[================ 三、前端 =================== - 14 -](#_Toc477889226)

[**3.1 jsp和Servlet有哪些相同点和异同点?它们之间的关系是什么？** - 14 -](#_Toc477889227)

[**3.2 表单提交方式post与get的区别？** - 14 -](#_Toc477889228)

[**3.3 什么下调用servlet的doget( )和dopost( )?** - 14 -](#_Toc477889229)

[**3.4 http请求中get和post有何区别？** - 14 -](#_Toc477889230)

[**3.5 页面跳转时使用 forward和redirect传递数据的区别？** - 15 -](#_Toc477889231)

[**3.6 JavaSrcipt中window.onload事件和JQuery ready函数有什么不同？** - 15 -](#_Toc477889232)

[**3.7 简述Servlet的生命周期？** - 15 -](#_Toc477889233)

[**3.8 谈谈对Servlet的理解？** - 16 -](#_Toc477889234)

[**3.9 JSP内置对象有哪些？作用分别是什么？主要有哪些方法？** - 16 -](#_Toc477889235)

[**3.10 Web容器是什么？使用过哪些web容器？各自的优缺点及应用有什么？** - 17 -](#_Toc477889236)

[**3.11 Jsp中动态include与静态include 的区别？** - 18 -](#_Toc477889237)

[**3.12 request.getParameter( )和request.getAttribute( )有何区别？** - 19 -](#_Toc477889238)

[**3.13 CSS中常见的选择器有哪些？** - 19 -](#_Toc477889239)

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*最常见的 22 个 jQuery 面试题\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* - 20 -](#_Toc477889240)

[**1. jQuery 库中的 $() 是什么？** - 20 -](#_Toc477889241)

[**2. 网页上有 5 个 <div> 元素，如何使用 jQuery来选择它们？** - 20 -](#_Toc477889242)

[**3. jQuery 里的 ID 选择器和 class 选择器有何不同？** - 20 -](#_Toc477889243)

[**4. 如何在点击一个按钮时使用 jQuery 隐藏一个图片？** - 21 -](#_Toc477889244)

[**5. $(document).ready() 是个什么函数？为什么要用它？** - 21 -](#_Toc477889245)

[**6. JavaScript window.onload 事件和 jQuery ready 函数有何不同？** - 21 -](#_Toc477889246)

[**7. 如何找到所有 HTML select 标签的选中项？** - 21 -](#_Toc477889247)

[**8. jQuery 里的 each() 是什么函数？是如何使用它的？** - 21 -](#_Toc477889248)

[**9. 你是如何将一个 HTML 元素添加到 DOM 树中的？** - 21 -](#_Toc477889249)

[**10. 你能用 jQuery 代码选择所有在段落内部的超链接吗？** - 22 -](#_Toc477889250)

[**11. $(this) 和 this 关键字在 jQuery 中有何不同？** - 22 -](#_Toc477889251)

[**12. 你如何使用jQuery来提取一个HTML 标记的属性 例如. 链接的href?** - 22 -](#_Toc477889252)

[**13. 你如何使用jQuery设置一个属性值?** - 22 -](#_Toc477889253)

[**14. jQuery中 detach() 和 remove() 方法的区别是什么?** - 22 -](#_Toc477889254)

[**15. 你如何利用jQuery来向一个元素中添加和移除CSS类?** - 22 -](#_Toc477889255)

[**16. 使用 CDN 加载 jQuery 库的主要优势是什么 ?** - 22 -](#_Toc477889256)

[**17. jQuery.get() 和 jQuery.ajax() 方法之间的区别是什么?** - 23 -](#_Toc477889257)

[**18. jQuery 中的方法链是什么？使用方法链有什么好处？** - 23 -](#_Toc477889258)

[**19. 你要是在一个 jQuery 事件处理程序里返回了 false 会怎样？** - 23 -](#_Toc477889259)

[**20. 哪种方式更高效：document.getElementbyId("myId") 还是 $("#myId")？** - 23 -](#_Toc477889260)

[**21. JQuery中的each( )是什么函数？你是如何使用它的？** - 23 -](#_Toc477889261)

[**22. 面试官让简述JQuery和AJAX应该怎么说?** - 23 -](#_Toc477889262)

[============== 四、面向对象 ================ - 24 -](#_Toc477889263)

[**4.1 面向对象的特征有哪些方面？** - 24 -](#_Toc477889264)

[**4.2 多态的好处？** - 24 -](#_Toc477889265)

[**4.3 代码中如何实现多态？** - 24 -](#_Toc477889266)

[**4.4 简述java中实现多态的机制是什么？** - 25 -](#_Toc477889267)

[**4.5 接口的意义？** - 25 -](#_Toc477889268)

[**4.6 抽象类的意义？** - 25 -](#_Toc477889269)

[**4.7 抽象类(abstract class)与接口(interface)有什么区别？** - 25 -](#_Toc477889270)

[**4.8 java面向对象三大特性？五大原则？** - 25 -](#_Toc477889271)

[**4.9 java语言的优势?** - 26 -](#_Toc477889272)

[================ 五、基础知识 ================= - 27 -](#_Toc477889273)

[**5.1 数组有没有length( )方法? String有没有length( )方法？** - 27 -](#_Toc477889274)

[**5.2 请说明值传递和引用传递的区别？Java中String是值传递还是引用传递（引用传递）？** - 27 -](#_Toc477889275)

[**5.3 如何把string转换成double类型？** - 27 -](#_Toc477889276)

[**5.4 switch语句能否作用在byte上，能否作用在long上，能否作用在String上?** - 27 -](#_Toc477889277)

[**5.5 什么是Listener模式？** - 27 -](#_Toc477889278)

[**5.6 分别解释过滤器、拦截器、和监听器是什么？** - 28 -](#_Toc477889279)

[**5.7 java过滤器和拦截器区别？** - 28 -](#_Toc477889280)

[**5.8 静态变量和实例变量的区别?** - 28 -](#_Toc477889281)

[**5.9 java当中的四种引用？** - 28 -](#_Toc477889282)

[**5.10 hql和sql的区别?** - 29 -](#_Toc477889283)

[**5.11 hashmap和hashtable区别？** - 29 -](#_Toc477889284)

[**5.12 LinkedList和ArrayList的区别？** - 29 -](#_Toc477889285)

[**5.13 ArrayList和Vector的区别,HashMap和Hashtable的区别？** - 29 -](#_Toc477889286)

[**5.14 list、map、set区别？** - 29 -](#_Toc477889287)

[**5.15 heap(堆)和stack(栈)有什么区别？** - 30 -](#_Toc477889288)

[**5.16 Integer与int的区别？** - 30 -](#_Toc477889289)

[**5.17 请说明equals和==的区别？** - 30 -](#_Toc477889290)

[**5.19 字节流与字符流的区别？** - 30 -](#_Toc477889291)

[**5.20 同步和异步有何异同，在什么情况下分别使用他们？** - 30 -](#_Toc477889292)

[**5.21 简述你对线程、进程的理解？** - 31 -](#_Toc477889293)

[**5.22 定义一个线程有几种方式？有什么区别？** - 31 -](#_Toc477889294)

[**5.23 启动一个线程是用run( )还是start( )?** - 31 -](#_Toc477889295)

[**5.24 线程的基本状态以及状态之间的关系？** - 31 -](#_Toc477889296)

[**5.25 sleep( )和wait( )有什么区别?** - 32 -](#_Toc477889297)

[**5.26 实现线程同步的有哪些方法？** - 33 -](#_Toc477889298)

[**5.27 请说出作用域public 、private、protected以及不写时的区别？** - 33 -](#_Toc477889299)

[**5.28 String 、StringBuffer、StringBuilder有什么差别？在什么情况下使用它们？** - 33 -](#_Toc477889300)

[**5.29 解释一下序列化与反序列化？** - 33 -](#_Toc477889301)

[**5.30 垃圾回收的优点和原理（并考虑两种回收机制）？** - 34 -](#_Toc477889302)

[**5.31 GC是什么? 为什么要有GC?** - 34 -](#_Toc477889303)

[**5.32 java中会存在内存泄漏吗？请简单描述？** - 34 -](#_Toc477889304)

[**5.33 介绍一下final、finally以及finalize方法？** - 34 -](#_Toc477889305)

[**5.34 java语言如何进行异常处理？ throw、throws、try、catch、finally分别代表什么？** - 35 -](#_Toc477889306)

[**5.35 在什么情况下，finally语句不会执行？** - 36 -](#_Toc477889307)

[**5.36 error和exception有什么区别?** - 36 -](#_Toc477889308)

[**5.37 throw和throws有什么区别？** - 36 -](#_Toc477889309)

[**5.38 说出重写和重载的区别，重写方法是否可以改变返回值的类型?** - 36 -](#_Toc477889310)

[**5.39 B/S和C/S分别是什么？ 有什么区别？** - 36 -](#_Toc477889311)

[**5.40 java中常用的包及其作用?** - 37 -](#_Toc477889312)

[**5.41 java关键字有哪些？** - 37 -](#_Toc477889313)

[**5.42 数组的初始化方式有哪些?简述数组的内存分配方式。** - 37 -](#_Toc477889314)

[**5.43 解释java浮点数据 运算中为何出现精度丢失问题？以及如何解决这个问题？** - 37 -](#_Toc477889315)

[================ 六、手写代码 ================ - 39 -](#_Toc477889316)

[**6.1 写一个Singleton出来？** - 39 -](#_Toc477889317)

[**6.2 判断输入的三个数能否构成三角形？** - 39 -](#_Toc477889318)

[**6.3 判断是否是回文数字的程序？** - 40 -](#_Toc477889319)

[**6.4 冒泡排序：** - 40 -](#_Toc477889320)

[**6.5 九九乘法表：** - 40 -](#_Toc477889321)

[**6.6 存储过程：** - 41 -](#_Toc477889322)

[**6.7 常用SQL语句：** - 42 -](#_Toc477889323)

**JAVA知识点总结**

**================== 一、Java框架知识 ===================**

**1.1 Spring的注入方式？**

接口注入：

  接口注入模式因为具备侵入性，它要求组件必须与特定的接口相关联，实际使用有限。    
Setter 注入：

对于习惯了传统 javabean 开发的程序员，通过 setter 方法设定依赖关系更加直观。

构造器注入：

所有依赖关系在构造函数中集中呈现,依赖关系在构造时由容器一次性设定.

1. <!--使用基于注解的方法自动装配 -->
2. **<context:annotation-config/>**
3. <!--通过属性注入的方式 -->
4. **<bean** id="tire" class="com.springtest.Tire"**>**
5. **<property** name="price" value="1000"**/>**
6. **<property** name="brand" value="长城"**></property>**
7. **</bean>**
8. <!--通过构造器注入 -->
9. **<bean id**="person" class="com.springtest.Person"**>**
10. **<constructor-arg** value="jade yoon"**/>**
11. **<constructor-arg** value="25"**/>**
12. **</bean>**
13. **<bean** id="dao" class = "com.prj.text.UserDao "**>**
14. <!-- 注入list -->
15. **<property** name ="lsst"  **>**
16. <list>
17. <value>郭恒</value>
18. <value>李正琪</value>
19. <value>李静</value>
20. </list>
21. **</property>**
22. <!-- 注入map集合 -->
23. < **property** name ="mmap" >
24. <map>
25. <entry key="CN" value="中国"></entry>
26. <entry key="JP" value="日本"></entry>
27. </map>
28. **</property>**
29. **</bean>**

**1.2 简单说下Spring IOC 的实现原理？**

**(1)Ioc—Inversion of Control，即“控制反转”，不是什么技术，而是一种设计思想。**在[**Java**](http://lib.csdn.net/base/javase)开发中，**Ioc意味着将你设计好的对象交给容器控制，而不是传统的在你的对象内部直接控制。**

**●谁控制谁，控制什么：**传统[Java SE](http://lib.csdn.net/base/javase)程序设计，我们直接在对象内部通过new进行创建对象，是程序主动去创建依赖对象；而IoC是有专门一个容器来创建这些对象，即由Ioc容器来控制对 象的创建；谁控制谁？当然是IoC 容器控制了对象；控制什么？那就是主要控制了外部资源获取（不只是对象包括比如文件等）。

**●为何是反转，哪些方面反转了：**有反转就有正转，传统应用程序是由我们自己在对象中主动控制去直接获取依赖对象，也就是正转；而反转则是由容器来帮忙创建及注入依赖对象；为何是反转？因为由容器帮我们查找及注入依赖对象，对象只是被动的接受依赖对象，所以是反转；哪些方面反转了？依赖对象的获取被反转了。

**(2)DI—Dependency Injection，即“依赖注入”**：**组件之间依赖关系**由容器在运行期决定，形象的说，即由容器动态的将某个依赖关系注入到组件之中。依赖注入的目的并非为软件系统带来更多功能，而是为了提升组件重用的频率，并为系统搭建一个灵活、可扩展的平台。通过依赖注入机制，我们只需要通过简单的配置，而无需任何代码就可指定目标需要的资源，完成自身的业务逻辑，而不需要关心具体的资源来自何处，由谁实现。

　　●谁依赖于谁：当然是**应用程序依赖于IoC容器**；

　　●为什么需要依赖：**应用程序需要IoC容器来提供对象需要的外部资源**；

　　●谁注入谁：很明显是**IoC容器注入应用程序某个对象，应用程序依赖的对象**；

　　●注入了什么：就是**注入某个对象所需要的外部资源（包括对象、资源、常量数据）**。

**IoC和DI**有什么**关系**呢？其实它们**是同一个概念的不同角度描述**，由于控制反转概念比较含糊（可能只是理解为容器控制对象这一个层面，很难让人想到谁来维护对象关系），所以2004年大师级人物Martin Fowler又给出了一个新的名字：“依赖注入”，相对IoC 而言，“依赖注入”明确描述了“被注入对象依赖IoC容器配置依赖对象”。

**1.3 分别描述一下Spring的控制和反转？**

**首先：第一个问题，参与者都有谁？**  
1）对象  
2）IOC/DI容器  
3）某个对象的外部资源  
**第二问题：依赖，谁依赖谁？为什么需要依赖？**  
 ➀应用程序依赖于IOC/DI容器；➁应用程序需要IOC/DI容器来提供对象需要的外部资源。  
**第三个问题：注入，谁注入谁？又注入了什么呢？**  
 ➀IOC/DI容器注入对象；➁注入对象所需要的资源；  
**第四个问题：控制反转，谁控制谁？控制什么？为什么叫反转呢？存在正转吗？**  
 控制:是IOC/DI容器控制对象，主要是控制对象实例的创建，反转是相对于正向而言的，如果要在A里面使用C，在A类中直接(主动)去获取所需要的外部资源C，这种情况被称为正转的。反转就是A类不再主动去获取C，而是被动等待，等待IoC/DI的容器获取一个C的实例，然后反向的注入到A类中。  
**第五个问题：控制反转和依赖注入式同一个概念吗？**  
 依赖注入和控制反转是对同一件事情的不同描述，从某个方面讲，就是它们描述的角度不同。

➀**依赖注入：**是从应用程序的角度在描述，可以把依赖注入描述完整点：应用程序依赖容器创建并注入它所需要的外部资源；

➁**控制反转：**是从容器的角度在描述，描述完整点：容器控制应用程序，由容器反向的向应用程序注入应用程序所需要的外部资源。

**1.4 简单说下String Aop的实现原理？**

**Aop：**是面向切面编程(Aspect Oriented Programming)

**作用：**对代码段进行增强处理

**设计思想：**代理模式(在不改变原程序的基础上为代码段增加新的功能，如：日志输出、事务处理)

**原理：**AOP技术利用一种称为“横切”的技术，解剖封装的对象内部，并将那些影响了多个类的公共行为封装到一个可重用模块，这样就能减少系统的重复代码，降低模块间的耦合度，并有利于未来的可操作性和可维护性，AOP把软件系统分为两个部分：核心关注点和横切关注点。业务处理的主要流程是核心关注点，与之关系不大的部分是横切关注点。横切关注点的一个特点是，他们经常发生在核心关注点的多处，而各处都基本相似。比如权限认证、日志、事务处理。

**1.5 简述Spring中的事务传播行为(事务管理)和隔离级别？**

事务隔离级别是由谁实现的? 答：数据库系统

**事务传播：**

(1) Required：默认的事务传播行为，表示必须有逻辑事务，否则新建一个事务，使用PROPAGATION\_REQUIRED指定，表示如果当前存在一个逻辑事务，则加入该逻辑事务，否则将新建一个逻辑事务：

(2) RequiresNew：创建新的逻辑事务，使用PROPAGATION\_REQUIRES\_NEW指定，表示每次都创建新的逻辑事务（物理事务也是不同的）因此外部事务可以不受内部事务回滚状态的影响独立提交或者回滚。

(3) Supports：支持当前事务，使用PROPAGATION\_SUPPORTS指定，指如果当前存在逻辑事务，就加入到该逻辑事务，如果当前没有逻辑事务，就以非事务方式执行。

(4)NotSupported：不支持事务，如果当前存在逻辑事务，就把当前事务暂停，以非事务方式执行。

，使用PROPAGATION\_NOT\_SUPPORTED指定。

(5)Mandatory（受托者）：使用PROPAGATION\_MANDATORY指定，如果当前有事务，使用当前事务执行，如果当前没有事务，则抛出异常（IllegalTransactionStateException）。

(6)Never：不支持事务，如果当前存在是事务则抛出IllegalTransactionStateException异常，使用PROPAGATION\_NEVER指定。

(7)Nested：嵌套事务支持，使用PROPAGATION\_NESTED指定，如果当前存在事务，则在嵌套事务内执行，如果当前不存在事务，则创建一个新的事务，嵌套事务使用[数据库](http://lib.csdn.net/base/mysql)中的保存点来实现，即嵌套事务回滚不影响外部事务，但外部事务回滚将导致嵌套事务回滚。

**隔离级别：**

(1)更新丢失（Lost Update）：两个事务都企图去更新一行数据，导致事务抛出异常退出，两个事务的更新都白费了。

(2)脏数据（Dirty Read）：如果第二个应用程序使用了第一个应用程序修改过的数据，而这个数据处于未提交状态，这时就会发生脏读。第一个应用程序随后可能会请求回滚被修改的数据，从而导致第二个事务使用的数据被损坏，即所谓的“变脏”。

(3)不可重读（Unrepeatable Read）：一个事务两次读同一行数据，可是这两次读到的数据不一样，就叫不可重读。如果一个事务在提交数据之前，另一个事务可以修改和删除这些数据，就会发生不可重读。

(4)幻读（Phantom Read）：一个事务执行了两次查询，发现第二次查询结果比第一次查询多出了一行，这可能是因为另一个事务在这两次查询之间插入了新行。

**1.6 事务有哪些特性？Spring的事务管理有几种实现方式，如何实现？**

**特性：**(1)原子性：一个事务中所有对数据库的操作是一个不可分割的操作序列，要么全做要么全不做；

(2)一致性：数据不会因为事务的执行而遭到破坏；

(3)隔离性：一个事物的执行，不受其他事务的干扰，即并发执行的事物之间互不干扰；

(4)持久性：一个事物一旦提交，它对数据库的改变就是永久的；

事务的两种实现方式：编码方式、声明式事务管理方式

**注：**(1)基于AOP技术实现的声明式事务管理实质是：在方法执行前后进行拦截，然后再目标方法开始之前创建并加入事务，执行完目标方法后根据执行情况提交或回滚事务.

(2)声明式事务管理又有两种实现方式：

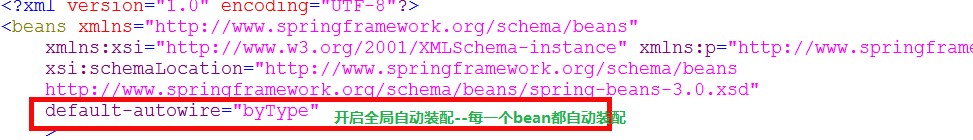
➀基于xml配置文件的方式；

➁在业务方法上进行@Transaction注解，将事务规则应用到业务逻辑中；

**1.7 怎样开启Spring 的注解装配？注解装配的作用以及优缺点?**

在ApplicationContext.xml中配置：

**<bean** id="dao" class = "com.prj.text.UserDao autowire=”byName"/**>**  或者

****

**自动装配作用：**简化bean之间的连接配置 ；

**自动装配(autowire)类型：**

➀根据名称 byName：属性名与bean的id名称必须相同

➁根据类型 byType：属性类型与bean中有相同的类型

➂根据类型通过构造器 constructor ：自动装配类型、构造器参数类型一致的bean

➃默认是无自动装配 none

**优缺点：**自动装配可以减轻配置的工作量，但同时使得配置文件的可读性变得很差，因为你不可能从配置文件中获知这些对象之间得依赖关系，从而维护困难！

**1.8 解释Spring支持的几种Bean的作用域?**

在spring中定义Bean时,除了可以对bean的属性进行注入，还可以为bean指定一个作用域，直接决定了spring创建该组件实例的策略，进而影响程序的运行效率和数据安全，主要分为以下5种：

➀Singleton：默认值，容器中只实例化一个；

➁Proptotype： 每次从容器中获取bean，都会创建一个新的实例(spring在每次获取该组件时，都会创建一个新的实例，避免因为共同一个实例产生的问题)；

➂request ：用于web环境，每次HTTP请求都会创建一个；

➃session ：用于web环境，同一个会话共享同一个实例，不同会话使用不同实例；

➄global session：在portlet(门户组件)的web应用中，同一个全局会话共享一个实例，对于非portlet环境，等同于session；

**1.9 Spring中的核心类都有哪些？各有什么作用？**

(1)BeanFactory：产生一个新的实例，可以实现单例模式；

(2)BeanWrapper：提供统一的get及set方法；

(3) ApplicationContext:提供框架的实现，包括BeanFactory的所有功能；

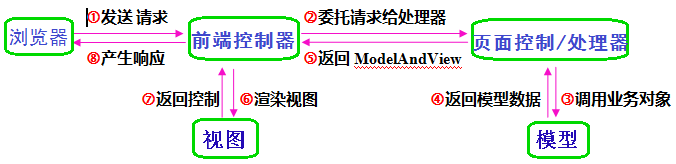
**1.10 简单介绍Spring MVC框架？**

(1) [spring](http://lib.csdn.net/base/javaee) MVC属于SpringFrameWork的后续产品，是Spring衍生而来的一个子类，（用来替换Struts）与Spring框架可以更完美的整合，轻量级框架，也是一个真正意义上的MVC设计模式。

(2)Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

(3)Spring MVC框架主要由DispatcherServlet、处理器映射、处理器(控制器)、视图解析器、视图组成。

**1.11 Spring MVC原理、流程以及接口解释？**



SpringMVC运行原理:

(1)客户端请求提交到DispatcherServlet；

(2)由DispatcherServlet控制器查询一个或多个HandlerMapping，找到处理请求的Controller；

(3)DispatcherServlet将请求提交到Controller；

(4)Controller调用业务逻辑处理后，返回ModelAndView；

(5) DispatcherServlet查询一个或多个ViewResoler视图解析器，找到ModelAndView指定的视图；

(6) 视图负责将结果显示到客户端；

DispatcherServlet是整个Spring MVC的核心。它负责接收HTTP请求组织协调Spring MVC的各个组成部分。其主要工作有以下三项：

       ➀ 截获符合特定格式的URL请求。  
       ➁ 初始化DispatcherServlet上下文对应的WebApplicationContext，并将其与业务层、持久化层的WebApplicationContext建立关联。  
       ➂ 初始化Spring MVC的各个组成组件，并装配到DispatcherServlet中。

**SpringMVC接口解释**

**(1)DispatcherServlet接口：**Spring提供的前端控制器，所有的请求都有经过它来统一分发。在DispatcherServlet将请求分发给Spring Controller之前，需要借助于Spring提供的HandlerMapping定位到具体的Controller。

**(2)HandlerMapping接口：**能够完成客户请求到Controller映射。

**(3)Controller接口：**为并发用户处理上述请求，因此实现Controller接口时，必须保证线程安全并且可重用。

Controller将处理用户请求，这和Struts Action扮演的角色是一致的。一旦Controller处理完用户请求，则返回ModelAndView对象给DispatcherServlet前端控制器，ModelAndView中包含了模型（Model）和视图（View）。

从宏观角度考虑，DispatcherServlet是整个Web应用的控制器；从微观考虑，Controller是单个Http请求处理过程中的控制器，而ModelAndView是Http请求过程中返回的模型（Model）和视图（View）。

**(4)ViewResolver接口：**Spring提供的视图解析器（ViewResolver）在Web应用中查找View对象，从而将相应结果渲染给客户。

**1.12 Struts1的执行原理？**

(1)初始化：struts框架的总控制器ActionServlet是一个Servlet，它在web.xml中配置成自动启动的Servlet，在启动时总控制器会读取配置文件(struts-config.xml)的配置信息，为struts中不同的模块初始化相应的对象。(面向对象思想)

(2)发送请求：用户提交表单或通过URL向WEB服务器提交请求，请求的数据用HTTP协议传给web服务器。

(3)form填充：struts的总控制器ActionServlet在用户提交请求时将数据放到对应的form对象中的成员变量中。

(4)派发请求：控制器根据配置信息对象ActionConfig将请求派发到具体的Action，对应的formBean一并传给这个Action中的excute()方法。

(5)处理业务：Action一般只包含一个excute()方法，它负责执行相应的业务逻辑(调用其它的业务模块)完毕后返回一个ActionForward对象。服务器通过ActionForward对象进行转发工作。

(6)返回响应：Action将业务处理的不同结果返回一个目标响应对象给总控制器。

(7)查找响应：总控制器根据Action处理业务返回的目标响应对象，找到对应的资源对象，一般情况下为jsp页面。

(8)响应用户：目标响应对象将结果传递给资源对象，将结果展现给用户。

**1.13 Struts2的执行流程、原理？**



**struts2的工作原理：**

在Struts2框架中的处理大概分为以下几个步骤

(1)客户端初始化一个指向Servlet容器（例如Tomcat）的请求 ；

(2)这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做ActionContextCleanUp的可选过滤器，这个过滤器对于Struts2和其他框架的集成很有帮助，例如：SiteMesh Plugin） ；

(3)接着FilterDispatcher被调用，FilterDispatcher询问ActionMapper来决定这个请是否需要调用某个Action ；

(4) 如果ActionMapper决定需要调用某个Action，FilterDispatcher把请求的处理交给ActionProxy ；

(5)ActionProxy通过Configuration Manager询问框架的配置文件，找到需要调用的Action类 ；

(6) ActionProxy创建一个ActionInvocation的实例；

(7) ActionInvocation实例使用命名模式来调用，在调用Action的过程前后，涉及到相关拦截器（Intercepter）的调用；

(8)一旦Action执行完毕，ActionInvocation负责根据struts.xml中的配置找到对应的返回结果。返回结果通常是（但不总是，也可 能是另外的一个Action链）一个需要被表示的JSP或者FreeMarker的模版。在表示的过程中可以使用Struts2 框架中继承的标签。在这个过程中需要涉及到ActionMapper。

**struts2的工作流程：**

(1)用户从浏览器发送HTTP请求到服务器；

(2) 根据web.xml配置，该请求被FilterDispatcher接收；

(3) 根据struts.xml配置，找到需要调用的Action类和方法， 并通过IoC方式，将值注入给Aciton ；(4) Action调用业务逻辑组件处理业务逻辑，这一步包含表单验证；

(5) Action执行完毕，根据struts.xml中的配置找到对应的返回结果result，并跳转到相应页面；

(6) 返回HTTP响应到客户端浏览器。

**1.14 Struts1与Struts2的区别与比较？**

struts2相对于struts1来说简单了很多，并且功能强大了很多，我们可以从几个方面来看：

**(1)从体系结构来看：**struts2大量使用拦截器来处理请求，从而允许与业务逻辑控制器 与 servlet-api分离，避免了侵入性；而struts1.x在action中明显的侵入了servlet-api.

**(2)从线程安全分析：**struts2.x是线程安全的，每一个对象产生一个实例，避免了线程安全问题；而struts1.x在action中属于单线程。

**(3)性能方面：**struts2.xml[测试](http://lib.csdn.net/base/softwaretest)可以脱离web容器，而struts1.x依赖servlet-api，测试需要依赖web容器。

**(4)请求参数封装对比：**struts2.x使用ModelDriven模式，这样我们 直接 封装model对象，无需要继承任何struts2的基类，避免了侵入性。

**(5)标签的优势：**标签库几乎可以完全替代JSTL的标签库，并且 struts2.x支持强大的ognl表达式。

  当然，struts2和struts1相比，在文件上传，数据校验等方面也方便了好多。

**1.15 简述actionfrom和action在Struts中扮演的角色？**

这是典型的mvc题目（简单说）：action是处理器，处理一个请求的 actionform是处理器要处理的表单(也就是你前台提交的参数信息) ；

➀actionform，是用来保存表单数据的，是框架中的容器层部分。

➁action，是用来处理各种逻辑的后台业务层部分，主要是处理前台传来的数据或者是在数据库中调用的数据，并将处理结果返回给前台显示页面。

**1.16 Hibernate和ibatis的区别？**

(1)两者都是开源框架：ibatis非常简单易学，hibernate相对较复杂，门槛较高。

(2)当系统二次开发(不能控制和修改数据库结构)时，ibatis的灵活性比hibernate更适合 ；

(3)系统数据处理量巨大，性能要求极为苛刻，必须通过sql语句（或存储过程）才能达到系统性能设计指标时ibatis会有更好的可控性和表现。   
 (4)ibatis需要手写sql语句，也可以生成一部分，hibernate则基本上可以自动生成，偶尔会写一些hql。同样的需求,ibatis的工作量比hibernate要大很多。类似的，如果涉及到数据库字段的修改，hibernate修改的地方很少，而ibatis要把那些sql mapping的地方逐个修改。    
 (5)hibernate现在已经是主流o/r mapping框架，从文档的丰富性产品的完善性，版本的开发速度都要强于ibatis。

**1.17 Hibernate和mybatis区别？**

|  |  |
| --- | --- |
| **mybatis** | **Hibernate** |
| (1)是一个sql语句映射的框架；  (2)注重POJO与SQL之间的映射关系，不会在运行期自动生成SQL；  (3)自动化程度低，手工映射sql 比较灵活、直观能一眼看到sql.  (4) 把sql语句与java代码分离了，sql语句在xml文件配置的，它能建立面向对象的域模型和关系数据模型之间的映射；  (5)需要开发人员熟练掌握sql语句； | (1)主流的ORM框架，提供了从POJO到数据库表的全套映射机制；  (2)会自动生成全套sql语句；  (3)因为自动化程度高，映射配置较为复杂，API也相对复杂，灵活性低；  (4) 对jdbc进行了封装，在分层结构中处于持久化层，通过对象操作数据它大大简化了dao层的编码工作；  (5)开发人员不必关注sql底层语句开发； |

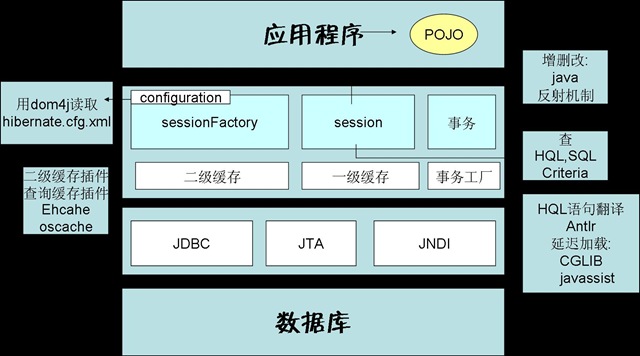
**1.18 Hibernate的工作原理？**

(1)配置好[hibernate](http://www.so.com/s?q=hibernate&ie=utf-8&src=wenda_link)的配置文件和与类对应的配置文件后，启动服务器;

(2)服务器通过[实例化](http://www.so.com/s?q=%E5%AE%9E%E4%BE%8B%E5%8C%96&ie=utf-8&src=wenda_link)Configeration对象，读取hibernate.cfg.xml文件的配置内容，并根据相关的需求建好表或者和表建立好[映射](http://www.so.com/s?q=%E6%98%A0%E5%B0%84&ie=utf-8&src=wenda_link)关系;

(3)通过实例化的Configeration对象就可以建立[sessionFactory](http://www.so.com/s?q=sessionFactory&ie=utf-8&src=wenda_link)实例，进一步，通过sessionFactory实例可以创建[session对象](http://www.so.com/s?q=session%E5%AF%B9%E8%B1%A1&ie=utf-8&src=wenda_link);

(4)得到session之后，便可以对数据库进行增删改查操作了，除了比较复杂的全文搜索外，简单的操作都可以通过hibernate封装好的session内置方法来实现;

(5)此外，还可以通过事物管理，表的关联来实现较为复杂的数据库设计  
优点：hibernate相当于[java](http://www.so.com/s?q=java&ie=utf-8&src=wenda_link)类和数据库表之间沟通的桥梁，通过这座桥我们就可以做很多事情了。

**Hibernate使用的基本流程：**配置Configuration对象，产生SessionFactory，创建session对象，启动事务，完成CRUD操作，提交事务，关闭session；

**1.19 Hibernate的缓存等级及其特点？**

**(1)一级缓存：**由于Hibernate一级缓存的生命周期跟Session的生命周期一样，所以也可以理解为一级Hibernate缓存是session缓存。当使用Session查询一个对象时，先从自己内部查看，若不存在才去访问数据库，并将结果保存在自己内部。一个Session代表一次会话过程，与一个数据库连接相关联，并且session线程不安全，多个线程共享时容易出现问题，所以Session不能长时间打开，通常仅用于一个事务当中，事务结束时就关闭Session。

**(2)二级缓存：**是进程或集群范围内的缓存，可以被所有的Session共享，生命周期和SessionFactory一样，SessionFactory对象的生命周期对应程序的整个进程，因此它属于进程范围的缓存。一个SessionFactory关联一个二级缓存，也就是一个二级缓存只能负责缓存一个数据库中的数据。

在Hibernate中使用二级缓存要在hibernate.cfg.xml配置文件中配置：

➀使用哪个厂家的缓存产品；

➁配置缓存产品的的配置文件；

➂Hibernate中配置哪些实体对象要纳入到二级缓存的管理中。

**Hibernate二级缓存策略的一般过程如下：**

1) 条件查询的时候，总是发出一条select \* from table\_name where …. （选择所有字段）这样的SQL语句查询数据库，一次获得所有的数据对象。  
 2) 把获得的所有数据对象根据ID放入到第二级缓存中。  
 3) 当Hibernate根据ID访问数据对象时，首先从Session一级缓存中查；查不到，如果配置了二级缓存，那么从二级缓存中查；查不到，再查询数据库，把结果按照ID放入到缓存。  
 4) 删除、更新、增加数据的时候，同时更新缓存。  
Hibernate二级缓存策略，是针对于ID查询的缓存策略，对于条件查询则毫无作用。为此，Hibernate提供了针对条件查询的Query Cache。

**1.20 Hibernate中load方法和get方法的区别？**

load加载方法：Users user = (Users)session.load(Users.class, userId)；  
get加载方法：  Users user = (Users)session.get(Users.class, userId)；

**区别：**

(1)如果数据库中没有userId对应的记录，通过get方法加载，则返回的是null值；如果通过load方法加载，则返回一个代理对象，当通过user对象调用某个方法（比如user.getPassword()）时，会抛出异常：org.hibernate.ObjectNotFoundException。  
 (2)load支持延迟加载，get不支持延迟加载。

**1.21 Hibernate中不看数据库，不看xml文件，不看查询语句，怎么样才能知道表结构？**

看表结构对应的类文件，比如UserInfo表对应的UserInfo.java文件

**1.22 Ibatis中#与$的区别？**

|  |  |
| --- | --- |
| $ | **#(**一般能用#的就别用$. ) |
| (1) $将传入的数据直接显示生成在sql中。如：order by $user\_id$，如果传入的值是111,那么解析成sql时的值为order by user\_id,  如果传入的值是id，则解析成的sql为order by id；  (2) $方式无法方式sql注入，$方式一般用于传入数据库对象．例如传入表名； | (1) #将传入的数据都当成一个字符串，会对自动传入的数据加一个双引号。如：order by #user\_id#，如果传入的值是111,那么解析成sql时的值为order by "111", 如果传入的值是id，则解析成的sql为order by "id".，不可以；  (2) #方式能够很大程度防止sql注入； |

**1.23 MyBatis的作用、工作原理、工作流程以及优缺点？**

**作用：**MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以对配置和原生Map使用简单的 XML 或注解，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

**工作原理：**

(1) 使用连接池

(2)统一sql 存取xml

(3)参数封装和结果映射

(4)sql语句的复用封装

注：当我们需要创建SqlSession对象并需要执行SQL语句时，这时候MyBatis才会去调用dataSource对象来创建java.sql.Connection对象。也就是说， Connection对象的创建一直延迟到执行SQL语句的时候。

**执行过程：**

（1）在mybatis中，通过MapperProxy动态代理咱们的dao， 也就是说， 当咱们执行自己写的dao里面的方法的时候，其实是对应的mapperProxy在代理，关联到mapper 再执行mapper.xml中的sql语句===

➀创建SqlSession对象，首先，SqlSessionFactoryBuilder去读取mybatis的配置文件，然后build一个DefaultSqlSessionFactory。通过SqlSessionFactory去获取SqlSession对象；

➁调用连接池dataSource对象来创建java.sql.Connection对象；

**优点：**

(1) 易于上手和掌握。

(2) sql写在xml里，便于统一管理和优化。

(3) 解除sql与程序代码的耦合。

(4)提供映射标签，支持对象与[数据库](http://lib.csdn.net/base/mysql)的orm字段关系映射

(5)提供对象关系映射标签，支持对象关系组建维护

(6)提供xml标签，支持编写动态sql。

**缺点：**

(1) sql工作量很大，尤其是字段多、关联表多时，更是如此。

(2) sql依赖于数据库，导致数据库移植性差。

(3) 由于xml里标签id必须唯一，导致DAO中方法不支持方法重载。

(4) 参数的数据类型支持不完善。（如参数为Date类型时，容易报没有get、set方法，需在参数上加@param）；

(5)多参数时，使用不方便，功能不够强大。（目前支持的方法有map、对象、注解@param以及默认采用012索引位的方式；

**总结:**

mybatis的优点同样是mybatis的缺点，正因为mybatis使用简单，数据的可靠性、完整性的瓶颈便更多依赖于程序员对sql的使用水平上了。sql写在xml里，虽然方便了修改、优化和统一浏览，但可读性很低，调试也非常困难，也非常受限。

mybatis没有hibernate那么强大，但mybatis最大的优点就是简单小巧易上手，方便浏览修改sql语句。

**1.24 MVC各部分有哪些技术实现？是如何实现的？**

MVC 是 Model－View－Controller 的简写，它是一个存在于服务器表达层的模型，它将应用分开，改变应用之间的高度耦合。

(1) 模型(Model)：应用对象（ 通过JavaBean、EJB 组件实现）。

模型是应用程序的主体部分。 模型代表了业务数据和业务逻辑； 当数据发生改变时，它要负责通知视图部分；一个模型能为多个视图提供数据。由于同一个模型可以被多个视图重用，所以提高了应用的可重用性。

(2) 视图(View)：数据的展现（ 由 JSP 页面产生）。

视图是用户看到并与之交互的界面。视图向用户显示相关的数据，并能接收用户的输入数据，但是它并不进行任何实际的业务处理。视图可以向模型查询业务状态，但不能改变模型。视图还能接受模型发出的数据更新事件，从而对用户界面进行同步更新。   
 (3) 控制器(Controller)：逻辑处理、控制实体数据在视图上展示、调用模型处理业务请求（一般是一个 Servlet）。

当 Web 用户单击 Web 页面中的提交按钮来发送 HTML 表单时，控制器接收请求并调用相应的模型组件去处理请求，然后调用相应的视图来显示模型返回的数据。

注：控制器完成页面逻辑，通过实体来与界面层完成通话；而C层直接与三层中的BLL进行对话。

**1.25三层架构和MVC是同一概念吗？**

三层架构和MVC是有明显区别的，MVC应该是展现模式(三个加起来以后才是三层架构中的UI层) 三层架构(3-tier application) 通常意义上的三层架构就是将整个业务应用划分为：表现层（UI）、业务逻辑层（BLL）、数据访问层（DAL）。区分层次的目的即为了“高内聚，低耦合”的思想。

(1)**表现层（UI）**：通俗讲就是展现给用户的界面，即用户在使用一个系统的时候他的所见所得。

(2)**业务逻辑层（BLL）**：针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。

(3)**数据访问层（DAL）**：事务直接操作数据库，针对数据的增添、删除、修改、更新、查找等。

**1.26 Hibernate中怎样实现类之间的关系？**

类与类之间的关系主要体现在表与表之间的关系进行操作，它们都是对对象进行操作，我们程序中把所有的表与类都映射在一起，它们通过配置文件中的many-to-one、one-to-many、many-to-many，来实现类之间的关系。

**1.27 ORM是什么？它有什么特点？**

**概念：**对象关系映射（Object Relational Mapping，简称ORM）模式是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系数据库中。   ORM的方法论基于三个核心原则：   
　　 (1)简单：以最基本的形式建模数据。   
　　 (2) 传达性：数据库结构被任何人都能理解的语言文档化。   
　　 (3)精确性：基于数据模型创建正确标准化了的结构。

**作用：**ORM解决的主要问题是对象关系的映射。域模型和关系模型分别是建立在概念模型的基础上的。域模型是面向对象的，而关系模型是面向关系的。一般情况下，一个持久化类和一个表对应，类的每个实例对应表中的一条记录，类的每个属性对应表的每个字段。   
 **ORM技术特点：**  
        (1)提高了开发效率。由于ORM可以自动对Entity对象与数据库中的Table进行字段与属性的映射，所以我们实际可能已经不需要一个专用的、庞大的数据访问层。   
        (2)ORM提供了对数据库的映射，不用sql直接编码，能够像操作对象一样从数据库获取数据。

**ORM的缺点 :**  
        ORM的缺点是会牺牲程序的执行效率和会固定思维模式。   
        从系统结构上来看,采用ORM的系统一般都是多层系统，系统的层次多了，效率就会降低。ORM是一种完全的面向对象的做法，而面向对象的做法也会对性能产生一定的影响。

**=============== 二、数据库 ==================**

**2.1 总结一下JDBC？**

**JDBC概念：**

JDBC是一种Java数据库的链接技术能实现Java程序对各种数据库的访问。由一组使用Java语言编写的类和接口组成，这些类和接口称为JDBC　API，它们位于包Java.sql以及Java.sql中。

**JDBC的作用：**

(1)建立和数据库之间的连接。

(2)将编好的SQL语句发送到数据库执行。

(3)对数据库返回的执行结果进行处理。

**JDBC的工作原理**(**Java访问数据库的具体步骤**)：

➀加载（注册）数据库驱动：

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

➁获取数据库的连接对象：Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "帐号","密码");

➂获取执行对象：Statement st=conn.createStatement();

➃准备SQL语句

➄处理结果集(返回响应行数)：查询执行st.executeQuery(sql)、增删改执行st.executeUpdate(sql);

➅关闭数据库：依次将ResultSet、Statement、PreparedStatement、Connection对象关 闭，释放所占用的资源。

**2.2 JDBC中的PreparedStatement相比Statement的好处？**

(1)用PreparedStatement来代替Statement代码多出几行，但代码的可读性、可维护性提高了；

(2)PreparedStatement是预编译语句，有可能被重复调用，被编译器编译后的执行代码被缓存下来,下次调用时只要是相同的预编译语句就不需要编译，直接将参数传入编译过的代码中执行即可；

(3)极大地提高了安全性，可以有效的避免SQL注入；

(4)有预编译功能，相同操作批量数据效率较高。

**2.3 数据库三个范式是什么？**

即： 属性唯一,   记录唯一,   表唯一

第一范式（1NF）：数据库表中的字段都是单一属性的，不可再分。这个单一属性由基本类型构成，包括整型、实数、字符型、逻辑型、日期型等。

第二范式（2NF）：数据库表中不存在非关键字段对任一候选关键字段的部分函数依赖（部分函数依赖指的是存在组合关键字中的某些字段决定非关键字段的情况），也即所有非关键字段都完全依赖于任意一组候选关键字。

第三范式（3NF）：在第二范式的基础上，数据表中如果不存在非关键字段对任一候选关键字段的传递函数依赖则符合第三范式。所谓传递函数依赖，指的是如果存在"A → B → C"的决定关系，则C传递函数依赖于A。因此，满足第三范式的数据库表应该不存在如下依赖关系： 关键字段 → 非关键字段x → 非关键字段y 。

**2.4 简要介绍数据库中的存储过程以及它的优缺点？**

|  |  |
| --- | --- |
| **定义** | 存储过程（Stored Procedure）是一组为了完成特定功能的SQL 语句集，经编译后存储在数据库中用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。 |
| **应用场景** | 存储过程一般用于处理比较复杂的任务, 可以大大降低耗时，其编译机制也提高了[数据库](http://lib.csdn.net/base/mysql)执行速度。 |
| **优点** | (1)存储过程只在创建时进行编译一次，以后每次执行都不需再重新编译，而一般SQL语句每执行一次就编译一次,所以使用存储过程可提高数据库执行速度；  (2)当对数据库进行复杂操作时(如对多个表进行Update,Insert,Query,Delete时），可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用；  (3)存储过程可以重复使用；  (4)安全性高，可设定存储过程的使用权； |
| **缺点** | (1)如果需要对输入存储过程的参数进行更改，或者要更改由其返回的数据，则您仍需要更新程序集中的代码以添加参数、更新 GetValue() 调用，这时候估计比较繁琐了。  (2)可移植性差 ：由于存储过程将应用程序绑定到 SQL Server，因此使用存储过程封装业务逻辑将限制应用程序的可移植性。 |

**2.5 简要介绍数据库中触发器?**

**(1)定义：**触发器 是一种特殊类型的存储过程，一般的存储过程是通过存储过程名直接调用，而触发器主要是通过事件(增、删、改)进行触发而被执行的。其在表中数据发生变化时自动强制执行。常见的触发器有两种：after(for)、instead of，用于insert、update、delete事件。

after(for) :  表示执行代码后，执行触发器；

instead of :  表示执行代码前，用已经写好的触发器代替你的操作；

**(2)触发器分类：**

➀DML触发器：ORACLE可以在DML语句进行触发，可以在DML操作前或操作后进行触发，并且可以对每个行或语句操作上进行触发。

➁ 替代触发器：由于在ORACLE里，不能直接对由两个以上的表建立的视图进行操作。所以给出了替代触发器。它就是ORACLE 8专门为进行视图操作的一种处理方法。

➂ 系统触发器:ORACLE 8i 提供了第三种类型的触发器叫系统触发器。它可以在ORACLE数据库系统的事件中进行触发，如ORACLE系统的启动与关闭等。

**(3)触发器的优点 :**

➀触发器是自动的：它们在对表的数据作了任何修改（比如手工输入或者应用程序采取的操作）之后立即被激活。

➁触发器可以通过数据库中的相关表进行层叠更改。例如，可以在 titles 表的 title\_id 列上写入一个删除触发器，以使其它表中的各匹配行采取删除操作。该触发器用 title\_id 列作为唯一键，在 titleauthor、sales 及 roysched 表中对各匹配行进行定位。

➂触发器可以强制限制，这些限制比用 CHECK 约束所定义的更复杂。与 CHECK 约束不同的是，触发器可以引用其它表中的列。例如，触发器可以回滚试图对价格低于 10 美元的书（存储在 titles 表中）应用折扣（存储在 discounts 表中）的更新。

**2.6 触发器与存储过程的区别：**

触发器与存储过程的主要区别在于运行方式。

(1)存储过程必须有用户、应用程序或者触发器来显示的调用并执行;

(2)触发器是当特定时间出现的时候，自动执行或者激活的，与连接用数据库中的用户、或者应用程序无关。当一行被插入、更新或者删除时触发器才执行，同时还取决于触发器是怎样创建的，当UPDATE发生时使用一个更新触发器，当INSERT发生时使用一个插入触发器，当DELETE发生时使用一个删除触发器。

**2.7 简说说你知道的数据库中常用函数?**

(1)**日期函数：**

day()：函数返回date\_expression 中的日期值；  
month() ：函数返回date\_expression 中的月份值；  
year()：函数返回date\_expression 中的年份值；

(2)**字符串函数：**

ASCII()：函数返回字符表达式最左端字符的ASCII 码值；  
char()：函数用于将ASCII 码转换为字符，输入0 ~ 255 之外的ASCII 会返回一个NULL 值；  
lower()：函数把字符串全部转换为小写；  
upper()：函数把字符串全部转换为大写；  
str()：函数把数值型数据转换为字符型数据；

(3)**统计函数：**  
AVG：求平均值  COUNT：统计数目 MAX：求最大值 MIN：求最小值 SUM：求和

(4)**算术函数：**三角函数、反三角函数

**2.8 说说数据连接池的工作机制？**

**概念：**对于需要频繁地跟数据库交互的应用程序，可以在创建了Connection对象，并操作完数据库后，可以不释放掉资源，而是将它放到内存中，当下次需要操作数据库时，可以直接从内存中取出Connection对象，不需要再创建了，这样就极大地节省了创建Connection对象的资源消耗，我们将在内存中存放Connection对象的容器称之为 连接池（Connection Pool）；

**作用：**数据库连接池负责分配，管理并释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而不是再重新建立一个新的数据库连接。同时，它还负责释放空闲时间超过最大空闲时间的数据库连接，避免因为没有释放数据库连接而引起的数据库连接遗漏。

**原理：**在J2EE中，服务器在启动时会创建一定数量的池连接，并一直维持不少于此数目的连接池。当客户程序需要访问数据库时，就可以直接从池中获取与数据库的连接（获取一个空闲的连接）。而不用去创建一个新的连接。同时将该连接标记为忙状态。当使用完毕后再将该连接标记为空闲状态。这样其他用户就可以使用这个连接了。如果当前没有空闲的连接，那么服务器就会根据配置参数在池中创建一定数量的连接。

**优点：**(1)采用这种方法对数据库连接进行管理后可以大幅缩短用户的响应时间，提高运行效率。

(2)为了提高数据库操作的性能，数据库连接池会释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。

**2.9 使用过哪些数据库？各自的特点及应用范围？**

**Oracle:**

(1)Oracle支持大并发，大访问量，是OLTP最好的工具

(2)Oracle是大型数据库而Mysql是中小型数据库，同时Mysql是开源的而Oracle价格非常高

(3)oracle默认不自动提交，需要用户手动提交。

(4)oracle逻辑备份时不锁定数据，且备份的数据是一致的。  
**mysql：**

(1) Mysql是开源的，默认是自动提交。  
(2)mysql逻辑备份时要锁定数据，才能保证备份的数据是一致的，影响业务正常的dml使用

**2.10 给出几条具体的sql语句的优化建议？至少三条哦**

(1) 查询尽量用确定的列名，少用\*号；

(2) 创建表的时候,应尽量建立主键,根据主键查询数据；

(3) 大数据表删除，用truncate table代替delete；

(4) 尽量多用commit语句提交事务，可以及时释放资源、解锁、释放日志空间、减少管理花费。

**2.11 MySQL数据库中的字段类型varchar和char的主要区别是什么？哪种字段的查找效率要高，为什么?**

Char的长度是固定的；varchar的长度是可变的，节省存储空间。

Char的查找效率高一些，因为varchar是非定长，必须先查找长度，然后进行数据的提取，比char定长类型多了一个步骤，所以效率低一些。

**================ 三、前端 ===================**

**3.1 jsp和Servlet有哪些相同点和异同点?它们之间的关系是什么？**

|  |  |
| --- | --- |
| **相同点** | jsp经编译后就变成了servlet(jsp本质就是servlet,jvm只能识别java的类，不能识别jsp代码，web容器将jsp的代码编译成jvm能够识别的java类)； |
| **异同点** | (1)jsp更擅长表现于页面显示，servlet更擅长于逻辑控制；  (2)setvlet中没有内置对象，jsp中的内置对象都是必须通过HttpServletRequest对象，HttpServletResponse对象及HttpServlet对象得到； |
| **关系** | jsp是servlet的一种简化，使用jsp只需要完成程序员需用输出到客户端的内容，jsp中的java脚本如何镶嵌到一个类中，由jsp容器完成，而servlet则是个完整的java类，这个类的service方法用于生成对客户端的响应； |

**3.2 表单提交方式post与get的区别？**

|  |  |  |
| --- | --- | --- |
| 比较 | **post** | **get** |
| 是否在URL中显示参数 | 否 | 是 |
| 数据传递是否有长度限制 | 无 | 有 |
| 数据安全性 | 高 | 低 |
| URL是否可以传播 | 否 | 是 |

**3.3 什么下调用servlet的doget( )和dopost( )?**

在servlet开发中，以doGet()和doPost()分别处理get和post方法。首先判断请求时是get还是post,如果是get就调用doGet(), 如果是post就调用doPost()；

注：一般来说我们是用不到doGet方法的，doGet方法提交表单的时候会在url后边显示提交的内容，所以不安全。而且doGet方法只能提交256个字符(1024字节)，而doPost没有限制，因为get方式数据的传输载体是URL（提交方式能form，也能任意的URL链接），而POST是HTTP头键值对（只能以form方式提交），通常我们使用的都是doPost方法。

**3.4 http请求中get和post有何区别？**

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| **点击返回/刷新按钮** | 没有影响 | 数据会重新发送（浏览器将会提示用户“数据被从新提交”） |
| **添加书签** | 可以 | 不可以 |
| **缓存** | 可以 | 不可以 |
| **编码类型(Encoding type)** | application/x-www-form-urlencoded | application/x-www-form-urlencoded or multipart/form-data. 请为二进制数据使用multipart编码 |
| **历史记录** | 有 | 没有 |
| **长度限制** | 有 | 没有 |
| **数据类型限制** | 只允许ASCII字符类型 | 没有限制。允许二进制数据 |
| **安全性** | 查询字符串会显示在地址栏的URL中，不安全，请不要使用GET请求提交敏感数据 | 因为数据不会显示在地址栏中，也不会缓存下来或保存在浏览记录中，所以看POST求情比GET请求安全，但也不是最安全的方式。如需要传送敏感数据，请使用加密方式传输 |
| **可见性** | 查询字符串显示在地址栏的URL中，可见 | 查询字符串不会显示在地址栏中，不可见 |

**什么是HTTP?**

超文本传输协议(HyperText Transfer Protocol -- HTTP)是一个设计来使客户端和服务器顺利进行通讯的协议。HTTP在客户端和服务器之间以request-response protocol（请求-回复协议）工作。

GET - 从指定的服务器中获取数据；

POST - 提交数据给指定的服务器处理；

**3.5 页面跳转时使用 forward和redirect传递数据的区别？**

|  |  |  |
| --- | --- | --- |
|  | **Forward** | **redirect** |
| **语法** | request.getRequestDispather(“URL”).forward(  request,response) | Response.sendRedirect(“URL”) |
| **地址栏显示** | forward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从哪里来的,所以它的地址栏还是原来的地址.  URL不改变，服务器内部跳转。 | redirect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.所以地址栏显示的是新的URL.  2次请求、响应，返回结果为第二次请求所对应的URL地址。 |
| **数据共享** | forward转发页面和转发到的页面可以共享request里面的数据. | redirect不能共享数据. |
| **运用地方** | forward:一般用于用户登陆的时候,根据角色转发到相应的模块. | redirect:一般用于用户注销登陆时返回主页面和跳转到其它的网站等. |
| **效率** | Forward效率高. | Redirect效率低. |
|  | 直接转发方式（Forward），客户端和浏览器只发出一次请求，Servlet、HTML、JSP或其它信息资源，由第二个信息资源响应该请求，在请求对象request中，保存的对象对于一个每个信息资源是共享的。 | 间接转发方式（Redirect）实际是两次HTTP请求，服务器端在响应第一次请求的时候，让浏览器再向另外一个URL发出请求，从而达到转发的目的。 |

**3.6 JavaSrcipt中window.onload事件和JQuery ready函数有什么不同？**

|  |  |
| --- | --- |
| **window.onload** | **JQuery ready** |
| (1) window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行；  (2) 不能同时编写多个，如果有多个window.onload方法，只会执行一个；  (3) window.onload没有简化写法； | (1)$(document).ready()是DOM结构绘制完毕后就执行，不必等到加载完毕；  (2)可以同时编写多个，并且都可以得到执行  (3) $(document).ready(function(){})可以简写成$(function(){}) ； |

**3.7 简述Servlet的生命周期？**



➀init（）：在Servlet的生命周期中，仅执行一次init()方法。它是在服务器装入Servlet时执行的，负责初始化Servlet对象。可以配置服务器，以在启动服务器或客户机首次访问Servlet时装入Servlet。无论有多少客户机访问Servlet，都不会重复执行init（）。

➁service（）：它是Servlet的核心，负责响应客户的请求。每当一个客户请求一个HttpServlet对象，该对象的Service()方法就要调用，而且传递给这个方法一个“请求”（ServletRequest）对象和一个“响应”（ServletResponse）对象作为参数。在HttpServlet中已存在Service()方法。默认的服务功能是调用与HTTP请求的方法相应的do功能。

➂destroy（）： 仅执行一次，在服务器端停止且卸载Servlet时执行该方法。当Servlet对象退出生命周期时，负责释放占用的资源。一个Servlet在运行service()方法时可能会产生其他的线程，因此需要确认在调用destroy()方法时，这些线程已经终止或完成。

**工作原理**：

首先简单解释一下Servlet接收和响应客户请求的过程：客户发送一个请求，调用service()方法对对请求的方式进行了匹配，选择调用doGet,doPost等这些方法，然后再进入对应的方法中调用逻辑层的方法，实现对客户的响应。

注：➀在Servlet接口没有doGet（）、doPost（）等这些方法的，HttpServlet中定义了这些方法，但是都是返回error信息，所以，我们每次定义一个Servlet的时候，都必须实现doGet或doPost等这些方法。

➁每一个自定义的Servlet都必须实现Servlet的接口，Servlet接口中定义了五个方法，其中比较重要的三个方法涉及到Servlet的生命周期，分别init(),service(),destroy()方法。GenericServlet是一个通用的，不特定于任何协议的Servlet,它实现了Servlet接口。而HttpServlet继承于GenericServlet，因此HttpServlet也实现了Servlet接口，所以定义Servlet的时候只需要继承HttpServlet即可。

➂Servlet接口和GenericServlet是不特定于任何协议的，而HttpServlet是特定于HTTP协议的类，所以HttpServlet中实现了service()方法，并将请求ServletRequest、ServletResponse 强转为HttpRequest 和 HttpResponse。

**3.8 谈谈对Servlet的理解？**

Java Servlet 是运行在 Web 服务器或应用服务器上的程序，它是作为来自 Web 浏览器或其他 HTTP 客户端的请求和 HTTP 服务器上的数据库或应用程序之间的中间层。

**3.9 JSP内置对象有哪些？作用分别是什么？主要有哪些方法？**

  输出输入对象：request对象、response对象、out对象；

  通信控制对象：pageContext对象、session对象、application对象；

Servlet对象：page对象、config对象；

错误处理对象：exception对象；

(1)request对象：客户端的请求信息被封装在request对象中，此请求包含来自GET/POST的请求参数，通过它才能了解到客户的需求，然后做出响应。它是HttpServletRequest类的实例。

(2)response对象 ：包含了响应客户端请求的有关信息。

(3)session对象 : 指的是客户端与服务器的一次会话，从客户连到服务器的一个WebApplication开始，直到客户端与服务器断开连接为止。

(4)out 对象 ：它是JspWriter类的实力，是向客户端输出内容常用的对象

(5)page对象 : 它指当前JSP页面本身，有点类似this指针，它是java.lang.object类的一个实例

(6)application对象 : 它实现了用户间的数据共享，可以改变全局变量，它从服务器启动开始直到服务器关闭。

(7)exception对象 : 它是一个例外对象当一个页面的运行过程中发生了例外，就产生这个对象。

(8)pageContext对象 : 它提供了对JSP页所有对象及名字空间的访问。

(9)config对象 :它是在一个Servlet初始化时，JSP引擎向它传递信息用的。

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | | | 说明 |
| request对象 | | | |
| **void setAttribute(String key,Object obj)** | | | **设置属性的属性值** |
| **Object getAttribute(String name)** | | | **返回指定属性的属性值** |
| **String getCharacterEncoding( )** | | | **返回字符编码方式** |
| **String getParameter(String name)** | | | **返回name指定参数的参数值** |
| **String[] getParameterValues(String name)** | | | **返回包含参数name的所有值的数组** |
| response对象 | | | |
| **String getCharacterEncoding( )** | | | **返回响应用的是何种字符编码** |
| **sendRedirect(“URL”)** | | | **重新定向客户端的请求** |
| session对象 | | | |
| **Long getCreationTime( )** | **返回SESSION创建时间** | | |
| **public String getId( )** | **返回SESSION创建时JSP引擎为它设的惟一ID号** | | |
| **String[] getValueNames( )** | **返回一个包含此SESSION中所有可用属性的数组** | | |
| **void invalidate( )** | **取消SESSION，使SESSION不可用(使Session失效)** | | |
| page对象 | | | |
| **Class getClass( )** | **返回此Object的类** | | |
| **String toString ( )** | **把此Object对象转换成String类的对象** | | |
| **Object clone( )** | **克隆此Object对象** | | |
| **void notify( )** | **唤醒一个等待的线程** | | |
| **Void wait(int timeout)** | **使一个线程处于等待直到timeout结束或被唤醒** | | |
| application对象 | | | |
| **void setAttribute(String name,Object obj)** | | **设定属性的属性值** | |
| **Object getAttribute(String name)** | | **返回给定名的属性值** | |
| **void removeAttribute(String name)** | | **删除一属性及其属性值** | |
| **URL getResource(String path)** | | **返回指定资源(文件及目录)的URL路径** | |
| **ServletContext getContext(String uripath)** | | **返回指定WebApplication的application对象** | |

**3.10 Web容器是什么？使用过哪些web容器？各自的优缺点及应用有什么？**

容器是一种服务调用规范框架，J2EE 大量运用了容器和组件技术来构建分层的企业级应用。在 J2EE 规范中，相应的有 WEB[Container](http://lib.csdn.net/base/docker) 和 EJB Container 等。

 WEB 容器给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使 JSP、SERVLET 直接跟容器中的环境变量交互，不必关注其它系统问题（从这个角度来说，web 容器应该属于[架构](http://lib.csdn.net/base/architecture)上的概念）。web 容器主要由 WEB 服务器来实现。开发Java Web应用所采用的服务器主要是与JSP/Servlet兼容的Web服务器，比较常用的有Tomcat、Resin、JBoss、WebSphere 和 WebLogic 等;

Tomcat 服务器：是一个小型、轻量级的支持JSP和Servlet 技术的Web服务器；

JBoss服务器：JBoss是一个种遵从JavaEE规范的、开放源代码的、纯Java的EJB服务器；

【Web服务器】

  Web服务器是运行及发布Web应用的容器，只有将开发的Web项目放置到该容器中，才能使网络中的所有用户通过浏览器进行访问。Web 服务器（Web Server）可以处理 HTTP 协议。当 Web 服务器接收到一个 HTTP 请求，会返回一个 HTTP 响应，例如送回一个 HTML 页面。

Web 服务器仅仅提供一个可以执行服务器端程序和返回(程序所产生的)响应的环境，而不会超出职能范围。Web 服务器主要是处理需要向浏览器发送 HTML 的请求以供浏览。

【应用程序服务器（The Application Server）】

    根据定义，作为应用程序服务器，要求可以通过各种协议（包括 HTTP 协议）把商业逻辑暴露给（expose）客户端应用程序。应用程序使用此商业逻辑就像你调用对象的一个方法或过程（语言中的一个函数）一样。

【serverlet】

Servlet（Server Applet），全称 [Java](http://lib.csdn.net/base/java)Servlet，未有中文译文。是用 [Java](http://lib.csdn.net/base/javase) 编写的服务器端程序。其主要功能在于交互式地浏览和修改数据，生成动态 Web 内容。狭义的 Servlet 是指 Java 语言实现的一个接口，广义的 Servlet 是指任何实现了这个 Servlet 接口的类，一般情况下，人们将 Servlet 理解为后者。

 Servlet 运行于支持 Java 的应用服务器中。从实现上讲，Servlet 可以响应任何类型的请求，但绝大多数情况下 Servlet 只用来扩展基于 HTTP 协议的 Web 服务器。

【Tomcat】

      Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好 Apache 服务器，可利用它响应对 HTML 页面的访问请求。实际上 Tomcat 部分是Apache 服务器的扩展，但它是独立运行的，所以当你运行 tomcat 时，它实际上作为一个与 Apache 独立的进程单独运行的。

**3.11 Jsp中动态include与静态include 的区别？**

|  |  |
| --- | --- |
| <%@include%>静态 | <jsp:include>动态 |
| <%@ include file="included.htm" %> | <jsp:include page="included.jsp" flush="true" /> |
| (1)静态用include伪码实现,定不会检查所含文件的变化，适用于包含静态页面。  (2) 使用静态[include](https://www.baidu.com/s?wd=include&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknHm1nj9hmHcLuHKBmHP-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1csn1f4nH03P6)时，不会解析所要包含的页面（你例子中的[include](https://www.baidu.com/s?wd=include&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknHm1nj9hmHcLuHKBmHP-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1csn1f4nH03P6)d.htm）即不管你的[include](https://www.baidu.com/s?wd=include&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknHm1nj9hmHcLuHKBmHP-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1csn1f4nH03P6)d.htm中有什么，我的任务就是把你包含并显示，其他的一概不管；  @ include是先把文件包含起来，然后统一编译 (先包含，后编译) | (1)动态用jsp:include动作实现，它总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。  (2) 使用 动态[include](https://www.baidu.com/s?wd=include&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknHm1nj9hmHcLuHKBmHP-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1csn1f4nH03P6)时，会先解析所要包含的页面（你例子中的[include](https://www.baidu.com/s?wd=include&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknHm1nj9hmHcLuHKBmHP-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1csn1f4nH03P6)d.jsp），解析后在和主页面放到一起显示；  (3) jsp:include是先编译一下included.jsp文件，然后再包含 (先编译，后包含) |

动作包含的属性：  
page：指定所包含资源的相对url路径，该资源必须时同一web应用程序的组成部分。  
flush：指定在执行include动作后是否应刷新缓冲区，在jsp1.1中，该属性必须设置为真。

动态include优点：

➀引入和同步一个动态的页面，使jsp页面更具灵活性。  
➁能和不同页面之间进行信息的交互和快捷的实现方式。  
➂改变了原始的所有页面编码都放在一个jsp上，使不同的功能分别写在不同页里，通过动态include方式引用到页面，更易于编码，更易于管理。

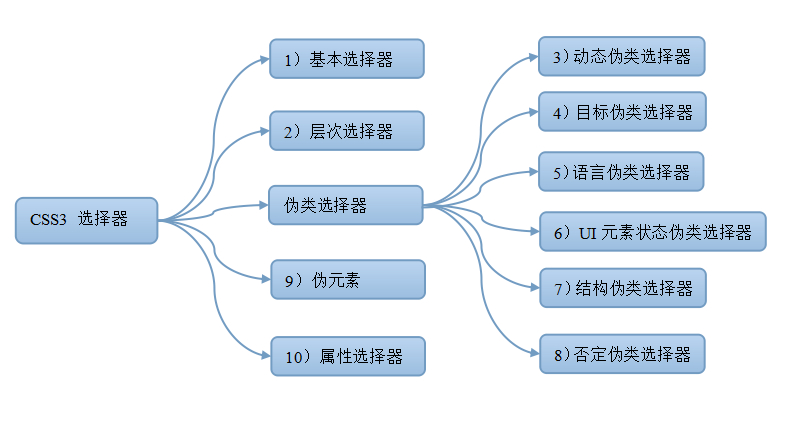
动态include不足：

动态的引入时需要频繁的变化和页面信息的更新和交互，要占用大量的资源开销。降低页面的访问速度。如果在没必要动态引入的情况下，不要使用动态include。

**3.12 request.getParameter( )和request.getAttribute( )有何区别？**

|  |
| --- |
| **request.getAttribute( )---对象** |
| (1) getAttribute 返回的是 Object，返回request范围内存在的对象，需进行转换,可用   setAttribute 设置成任意对象，使用很灵活 ;  (2) getAttribute：用于服务器端重定向时，即在 sevlet 中使用了 forward 函数， getAttribute 只能收到程序用 setAttribute 传过来的值。  (3) )request.setAttribute()和getAttribute()方法传递的数据只会存在于Web容器内部，在具有转发关系的W eb组件之间共享； |
| **request.getParameter( )---标签、页面或url参数** |
| (1) getParameter 返回的是 String类型的数据，用于读取提交的表单中的值 ，即获取  POST/GET 传递的参数值, getParameter 显然只能传字符串; (2) getParameter：用于客户端重定向时，即点击了链接或提交按扭时传值用，即用于在用表单 或 url 重定向传值时接收数据用。  (3) request.getParameter()方法传递的数据，会从Web客户端传到Web服务器端，代表HTTP请求数据； |

**3.13 CSS中常见的选择器有哪些？**



**(1)基本选择器：**

|  |  |  |
| --- | --- | --- |
| **类型** | **选择器** | **功能描述** |
| 标签选择器 | 没有任何标点 | 直接的标签名的匹配eg:$("input").css(); |
| 元素选择器 | E | 选择指定类型的HTML元素 |
| ID选择器 | #id | 选择指定ID属性值为“id”的任意类型元素eg:$("#userName").css(); |
| 类选择器 | .class | 选择指定class属性值为“class”的任意类型的任意多个元素  eg:$(".tab").css(); |
| 群组选择器 | selector1,selectorN | 将每一个选择器匹配的元素集合并 |
| 通配选择器 | \* | 匹配该元素下面的所有的元素eg:$("form \*").css(); |

**(2) 层次选择器:**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | | **功能描述** | |
| 后代选择器（包含选择器）a b | | | 选择匹配的b元素，E里所有是F的直系子元素eg:$("a b").css(); |
| 子选择器a>b | | 选择匹配的F元素，E下的所有F的元素eg:$("a>b").css(); | |
| 相邻兄弟选择器a+next | | 紧接着该元素的下一个名为next的元素eg: $("a+next").css(); | |
| 通用选择器a~b | a之后的所有的名称为b的所有后代元素（直系和间接的都是）eg:$("a~b").css(); | | |

**CSS选择器总结：**

(1)  标签选择器直接应用于HTML标签。

(2)  类选择器可在页面中多次使用。

(3)  ID选择器在同一个页面中只能使用一次。

**CSS中的优先级：**

* 行内样式 > 内部样式表 > 外部样式表
* ID选择器 > 类选择器 > 标签选择器

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*最常见的 22 个 jQuery 面试题\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1. jQuery 库中的 $() 是什么？**

$()函数是 jQuery() 函数的别称，乍一看这很怪异，还使 jQuery 代码晦涩难懂。一旦你适应了，你会爱上它的简洁。$() 函数用于将任何对象包裹成 jQuery 对象，接着你就被允许调用定义在 jQuery 对象上的多个不同方法。你甚至可以将一个选择器字符串传入$() 函数，它会返回一个包含所有匹配的 DOM 元素数组的 jQuery 对象。这个问题我已经见过好几次被提及，尽管它非常基础，它经常被用来区分一个开发人员是否了解 jQuery。

**2. 网页上有 5 个 <div> 元素，如何使用 jQuery来选择它们？**

另一个重要的 jQuery 问题是基于选择器的。jQuery 支持不同类型的选择器，例如 ID 选择器、class 选择器、标签选择器。鉴于这个问题没提到 ID 和 class，你可以用标签选择器来选择所有的 div 元素。jQuery 代码：$("div")，这样会返回一个包含所有 5 个 div 标签的 jQuery 对象。

**3. jQuery 里的 ID 选择器和 class 选择器有何不同？**

如果你用过 CSS，你也许就知道 ID 选择器和 class 选择器之间的差异，jQuery 也同样如此。ID 选择器使用 ID 来选择元素，比如 #element1，而 class 选择器使用 CSS class 来选择元素。当你只需要选择一个元素时，使用 ID 选择器，而如果你想要选择一组具有相同 CSS class 的元素，就要用 class 选择器。在面试过程中，你有很大几率会被要求使用 ID 选择器和 class 选择器来写代码。下面的 jQuery 代码使用了 ID 选择器和 class 选择器：

$('#LoginTextBox') // Returns element wrapped as jQuery object with id='LoginTextBox'

$('.active') // Returns all elements with CSS class active.

正如你所见，从语法角度来说，ID 选择器和 class 选择器的另一个不同之处是，前者用字符”#”而后者用字符”.”。

注：与类选择器不同，在一个HTML文档中，[ID](https://www.baidu.com/s?wd=ID&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkrHmLP1ubPjPhuhP-njR40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHf4PjbLn1f4)选择器只能使用一次。而类选择器可以使用多次。

**4. 如何在点击一个按钮时使用 jQuery 隐藏一个图片？**

这是一个事件处理问题。jQuery为按钮点击之类的事件提供了很好的支持。可通过以下代码隐藏一个通过ID或class定位到的图片。你需要知道如何为按钮设置事件并执行hide() 方法，代码如下：

$('#ButtonToClick').click(function(){

$('#ImageToHide').hide();

});

**5. $(document).ready() 是个什么函数？为什么要用它？**

ready() 函数用于在文档进入ready状态时执行代码。当DOM 完全加载（例如HTML被完全解析DOM树构建完成时），jQuery允许你执行代码。使用$(document).ready()的最大好处在于它适用于所有浏览器，jQuery帮你解决了跨浏览器的难题。

**6. JavaScript window.onload 事件和 jQuery ready 函数有何不同？**

这个问答是紧接着上一个的。JavaScript window.onload 事件和 jQuery ready 函数之间的主要区别是，前者除了要等待 DOM 被创建还要等到包括大型图片、音频、视频在内的所有外部资源都完全加载。如果加载图片和媒体内容花费了大量时间，用户就会感受到定义在 window.onload 事件上的代码在执行时有明显的延迟。

另一方面，jQuery ready() 函数只需对 DOM 树的等待，而无需对图像或外部资源加载的等待，从而执行起来更快。

使用 jQuery $(document).ready() 的另一个优势是你可以在网页里多次使用它，浏览器会按它们在 HTML 页面里出现的顺序执行它们，相反对于 onload 技术而言，只能在单一函数里使用。鉴于这个好处，用 jQuery ready() 函数比用 JavaScript window.onload 事件要更好些。

**7. 如何找到所有 HTML select 标签的选中项？**

这是面试里比较棘手的 jQuery 问题之一。这是个基础的问题，但是别期望每个 jQuery 初学者都知道它。你能用下面的 jQuery 选择器获取所有具备multiple=true 的 <select >标签的选中项：

$('[name=NameOfSelectedTag] :selected')

这段代码结合使用了属性选择器和:selected选择器，结果只返回被选中的选项。你可按需修改它，比如用 id 属性而不是 name 属性来获取<select>标签。

**8. jQuery 里的 each() 是什么函数？是如何使用它的？**

each() 函数就像是 Java 里的一个 Iterator，它允许你遍历一个元素集合。你可以传一个函数给 each() 方法，被调用的 jQuery 对象会在其每个元素上执行传入的函数。有时这个问题会紧接着上面一个问题，举个例子，如何在 alert 框里显示所有选中项。我们可以用上面的选择器代码找出所有选中项，然后我们在 alert 框中用 each() 方法来一个个打印它们，代码如下：

$('[name=NameOfSelectedTag] :selected').each(function(selected) {

alert($(selected).text()); });

其中 text() 方法返回选项的文本。

**9. 你是如何将一个 HTML 元素添加到 DOM 树中的？**

你可以用 jQuery 方法 appendTo() 将一个 HTML 元素添加到 DOM 树中。这是 jQuery 提供的众多操控 DOM 的方法中的一个。你可以通过 appendTo() 方法在指定的 DOM 元素末尾添加一个现存的元素或者一个新的 HTML 元素。

**10. 你能用 jQuery 代码选择所有在段落内部的超链接吗？**

这是另一个关于选择器的 jQuery 面试题。就像其他问题那样，只需一行 jQuery 代码就能搞定。你可以使用下面这个 jQuery 代码片段来选择所有嵌套在段落（<p>标签）内部的超链接（<a>标签）……

**11. $(this) 和 this 关键字在 jQuery 中有何不同？**

这对于很多 jQuery 初学者来说是一个棘手的问题，其实是个简单的问题。$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法，比如用 text() 获取文本，用val() 获取值等等。而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法，直到它被 $() 函数包裹，例如 $(this)。

[**jQuery**](http://lib.csdn.net/base/jquery)中this与$(this)的用法区别.先看以下代码: $("#textbox").hover( function() { this.title = "Test"; }, fucntion() { this.title = "OK”; } ); 这里的this其实是一个Html 元素(textbox)，textbox有text属性，所以这样写是完全没有什么问题的。 但是如果将this换成$(this)就不是那回事了，就会报错了。 以下写法是错误的： $("#textbox").hover( function() { $(this).title ＝ "Test"; }, function() { $(this).title = "OK"; } ); 这里的$(this)是一个JQuery对象，而jQuery对象沒有title 属性，因此这样写是错误的。 JQuery拥有attr()方法可以get/set DOM对象的属性，所以正确的写法应该是这样： 正确的写法： $("#textbox").hover( function() { $(this).attr('title', 'Test'); }, function() { $(this).attr('title', 'OK'); } ); 使用JQuery的好处是它包裝了各种浏览器版本对DOM对象的操作，因此统一使用$(this)而不再用this应该是比较不错的选择。

**12. 你如何使用jQuery来提取一个HTML 标记的属性 例如. 链接的href?**

attr() 方法被用来提取任意一个HTML元素的一个属性的值. 你首先需要利用jQuery选择及选取到所有的链接或者一个特定的链接，然后你可以应用attr()方法来获得他们的href属性的值。下面的代码会找到页面中所有的链接并返回href值：

$('a').each(function(){

alert($(this).attr('href'));

});

**13. 你如何使用jQuery设置一个属性值?**

前面这个问题的后续问题是，attr()方法和jQuery中的其它方法一样，能力不止一样. 如果你在调用attr()的同时带上一个值 例如. attr(name, value), 这里name是属性的名称，value是属性的新值。

**14. jQuery中 detach() 和 remove() 方法的区别是什么?**

尽管detach()和remove()方法都被用来移除一个DOM元素, 两者之间的主要不同在于detach()会保持对过去被解除元素的跟踪, 因此它可以被取消解除, 而remove()方法则会保持过去被移除对象的引用. 你也还可以看看用来向DOM中添加元素的appendTo()方法.

**15. 你如何利用jQuery来向一个元素中添加和移除CSS类?**

通过利用addClass()和removeClass() 这两个 jQuery 方法。动态的改变元素的class属性可以很简单例如. 使用类“.active"来标记它们的未激活和激活状态，等等.

**16. 使用 CDN 加载 jQuery 库的主要优势是什么 ?**

这是一个稍微高级点儿的jQuery问题。好吧，除了报错节省服务器带宽以及更快的下载速度这许多的好处之外, 最重要的是，如果浏览器已经从同一个CDN下载类相同的 jQuery 版本, 那么它就不会再去下载它一次. 因此今时今日，许多公共的网站都将jQuery用于用户交互和动画, 如果浏览器已经有了下载好的jQuery库，网站就能有非常好的展示机会。

**17. jQuery.get() 和 jQuery.ajax() 方法之间的区别是什么?**

ajax() 方法更强大，更具可配置性, 让你可以指定等待多久，以及如何处理错误。get( ) 方法是一个只获取一些数据的专门化方法。

**18. jQuery 中的方法链是什么？使用方法链有什么好处？**

方法链是对一个方法返回的结果调用另一个方法，这使得代码简洁明了，同时由于只对 DOM 进行了一轮查找，性能方面更加出色。

**19. 你要是在一个 jQuery 事件处理程序里返回了 false 会怎样？**

这通常用于阻止事件向上冒泡。

**20. 哪种方式更高效：document.getElementbyId("myId") 还是 $("#myId")？**

第一种，因为它document.getElementbyId("myId")直接调用了 JavaScript 引擎。

**21. JQuery中的each( )是什么函数？你是如何使用它的？**

对于jQuery对象，只是把each方法简单的进行了委托：把jQuery对象作为第一个参数传递给jQuery的each方法.换句话说：jQuery提供的each方法是对参数一提供的对象的中所有的子元素逐一进行方法调用each()函数是基本上所有的框架都提供了的一个工具类函数，通过它，你可以遍历对象、数组的属性值并进行处理。jQuery和jQuery对象都实 现了该方法，而jQuery对象提供的each方法则是对jQuery内 部的子元素进行逐个调用。

**22. 面试官让简述JQuery和AJAX应该怎么说?**

首先jQuery和Ajax都是基于Javascript的，jQuery是轻量级的javascript库，兼容[css3](https://www.baidu.com/s?wd=css3&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPHTYrjR1nH79nAmsnvRz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWcvPWfYP1c1rHTYrH0LnHDYr0)和各种浏览器，能够方便的处理[Html元素](https://www.baidu.com/s?wd=Html%E5%85%83%E7%B4%A0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPHTYrjR1nH79nAmsnvRz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWcvPWfYP1c1rHTYrH0LnHDYr0)，[表单验证](https://www.baidu.com/s?wd=%E8%A1%A8%E5%8D%95%E9%AA%8C%E8%AF%81&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPHTYrjR1nH79nAmsnvRz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWcvPWfYP1c1rHTYrH0LnHDYr0)、时间处理等；AJAX全称为“Asynchronous JavaScript and XML”（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术；意思是在事件触发之后，浏览器可以做其他事情，与发生事件的控件的无关控件不受影响，Ajax是一种用于创建更好更快以及交互性更强的Web应用程序技术。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，减少网络阻塞。

**============== 四、面向对象 ================**

**4.1 面向对象的特征有哪些方面？**

面向对象编程有三大特性：封装、继承、多态。

**(1)封装（Encapsulation）:**使用方法把类的数据隐藏起来，遵循了java 一切皆为对象的基本概念和核心思想，达成对类的封装，让普通类型上升为对象级；封装控制用户对类的修改和访问数据的程度，增强了程序的可维护性。

**(2)继承（Implementation）:**发生在类和类之间，可分为单继承和多层继承描述父子对象之间的血缘关系， 最终达到的目的是：一旦使用了继承，子类便获得了父类所有的成员（变量和方法），但是父类的私有的方法和属性不能被访问，父类的final成员不能被重写。 一旦使用了继承，父类的实例能指向派生类（子类）的引用。

**(3)多态（Multipart）:**对象在运行期和编译期具有的两种状态，使代码具有灵活性和重用性。编译时多态，也就是函数重载，所谓函数重载就是指同一个函数名可以对应多个函数的实现具体调用哪个按照由参数个数，参数类型等来决定。运行时多态：用一个基类的指针或引用来操纵多个派生类型的能力被称为多态性。

通俗的理解：

      封装：是面向对象方法的重要原则，就是把对象的属性和行为（数据）结合为一个独立的整体，并尽可能隐藏对象的内部实现细节，就是把不想告诉或者不该告诉别人的东西隐藏起来，把可以告诉别人的公开，别人只能用我提供的功能实现需求，而不知道是如何实现的。增加安全性

       继承：是面向对象最显著的一个特性，继承是从已有的类中派生出新的类称为子类，子类继承父类的数据属性和行为，并能根据自己的需求扩展出新的行为，提高了代码的复用性。

多态：指允许不同的对象对同一消息做出相应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式（发送消息就是函数调用）。封装和继承几乎都是为多态而准备的，在执行期间判断引用对象的实际类型，根据其实际的类型调用其相应的方法。

抽象：表示对问题领域进行分析、设计中得出的抽象的概念，是对一系列看上去不同，但是本质上相同的具体概念的抽象，在java中抽象用 abstract 关键字来修饰，用 abstract 修饰类时，此类就不能被实例化，从这里可以看出，抽象类就是为了继承而存在的，如果定义了一个抽象类而不去继承它，那么等于白白创建了这个抽象类，因为你不能用它来做任何事情，用 abstract 修饰方法时，此方法就是抽象方法，抽象方法必须存在于抽象类中，抽象方法没有方法体，对于一个父类来说，如果它的某个方法在父类中实现出来没有任何意义，必须根据子类的实际需求来进行不同的实现，那么就可以将这个方法声明为抽象方法，抽象方法的修饰符必须为 public 或者 protected ，应为用 private，则不能被子类继承，子类便无法实现该方法，缺省情况下默认为 public 。

**4.2 多态的好处？**

允许不同类对象对同一消息做出响应，即同一消息可以根据发送对象的不同而采用多种不同的行为方式(发送消息就是函数调用)，主要有以下优点:

1. 可替换性:多态对已存在代码具有可替换性.

2. 可扩充性:增加新的子类不影响已经存在的类结构.

3. 接口性:多态是超类通过方法签名,向子类提供一个公共接口,由子类来完善或者重写它来实现的.

4. 灵活性:

5. 简化性:

**4.3 代码中如何实现多态？**

实现多态主要有以下三种方式:

1. 实现接口

2. 继承父类重写方法

3. 同一类中进行方法重载

**4.4 简述java中实现多态的机制是什么？**

(父类)引用变量 (子类)实例对象 程序内存运行的是实例对象的方法；

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

**4.5 接口的意义？**

接口的意义用三个词就可以概括:规范,扩展,回调。

**4.6 抽象类的意义？**

抽象类的意义可以用三句话来概括：

1. 为其他子类提供一个公共的类型；

2. 封装子类中重复定义的内容；

3. 定义抽象方法,子类虽然有不同的实现,但是定义时一致的；

**4.7 抽象类(abstract class)与接口(interface)有什么区别？**

|  |  |  |
| --- | --- | --- |
|  | **Abstract class** | **Interface** |
| 实例化 | 不能 | 不能 |
| 构造方法 | 可以有 | 不能有 |
| 普通成员变量 | 可以有 | 没有 |
| 方法 | 可以包含普通方法与静态方法 | 所有方法都必须是抽象的，且不能包含静态方法 |
| 修饰符 | 抽象方法访问类型可以是Public、protected，非abstract方法，必须实现 | 抽象方法访问类型只能是public 且默认public abstract类型 |
| 包含静态成员变量 | 可以包含，访问类型可以任意，默认是friendly 型，其值可以在子类中重新定义，也可以重新赋值 | 可以包含，但只能是(默认)public static final 型，且必须给其初值，实现类中不能重新定义，不能改变其值。 |
| 设计理念 | 表示的是“is-a”关系 | 表示的是“like-a”关系 |
| 实现 | 要用extends继承抽象类（单继承） | 要用implements实现接口(多实现) |

**4.8 java面向对象三大特性？五大原则？**

面向对象三大特性：

**(1)封装(Encapsulation)**

封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

**2)继承(Inheritance)**

继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

**(3)多态(Polymorphism)**

多态就是指一个类实例的相同方法在不同情形有不同表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。

面向对象五大原则：

**(1)单一职责原则（Single-Resposibility Principle）**

•一个类应该仅有一个引起它变化的原因

**(2)开放封闭原则（Open-Closed principle）**

•对扩展是开放的，对更改是封闭的！

**(3)里氏替换原则（Liskov-Substituion Principle）**

•子类可以替换父类并且出现在父类能够出现的任何地方,贯彻GOF倡导的面向接口编程

**(4)依赖倒置原则（Dependecy-Inversion Principle）**

•传统的结构化编程中，最上层的模块通常都要依赖下面的子模块来实现，也称为高层依赖低层！

所以DIP原则就是要逆转这种依赖关系，让高层模块不要依赖低层模块，所以称之为依赖倒置原则！

**(5)ISP 接口隔离原则(Interface-Segregation Principle)**

•使用多个专门的接口比使用单个接口要好的多！

**4.9 java语言的优势?**

(1)Java是一种可以撰写跨平台应用程序的面向对象的程序设计语言

(2)java是纯面向对象开发，功能强大，分支众多;

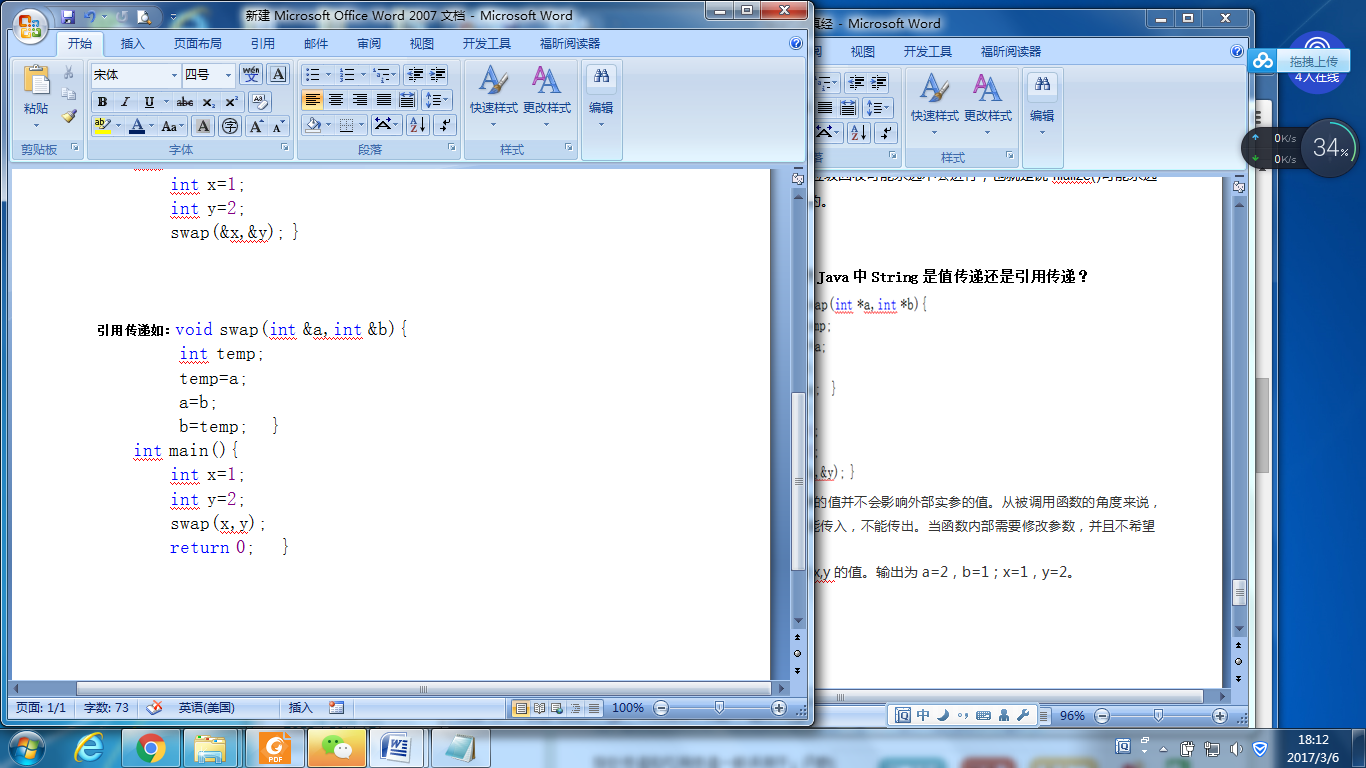
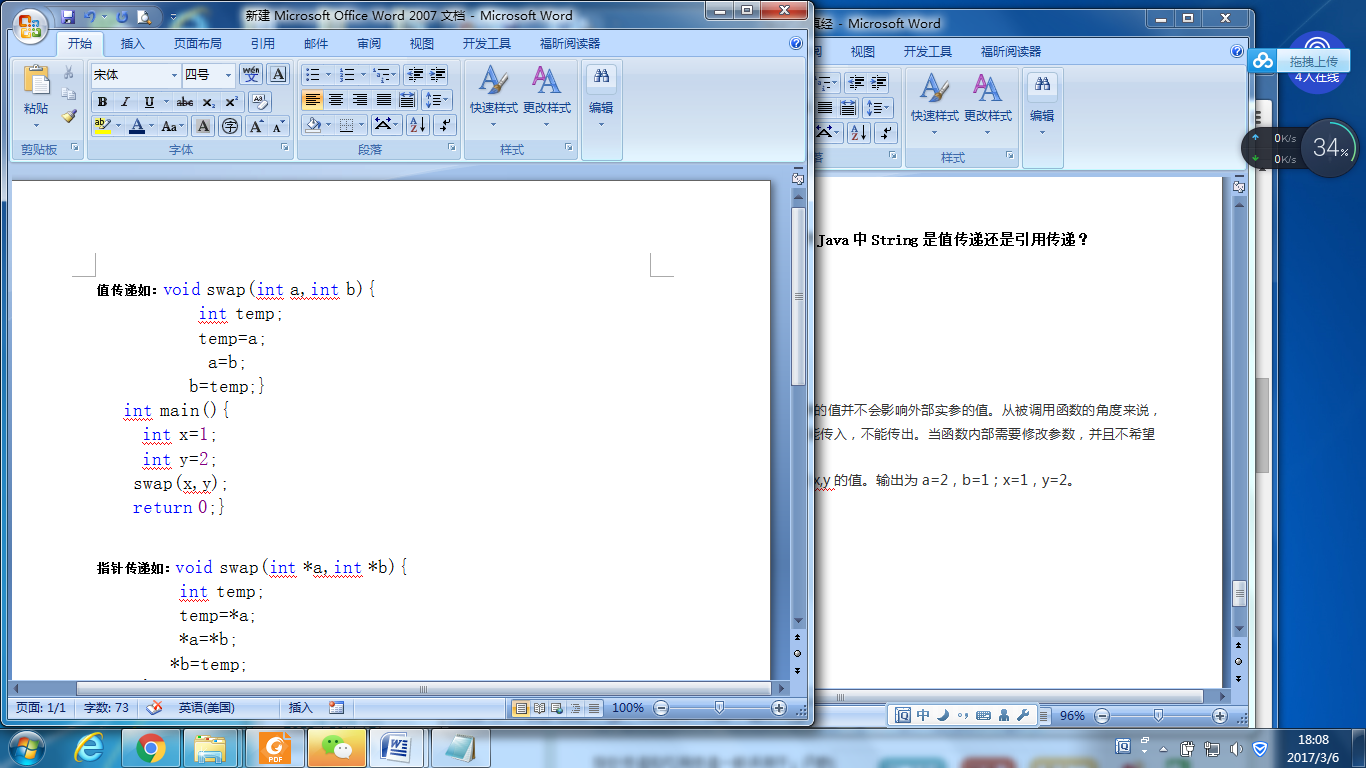
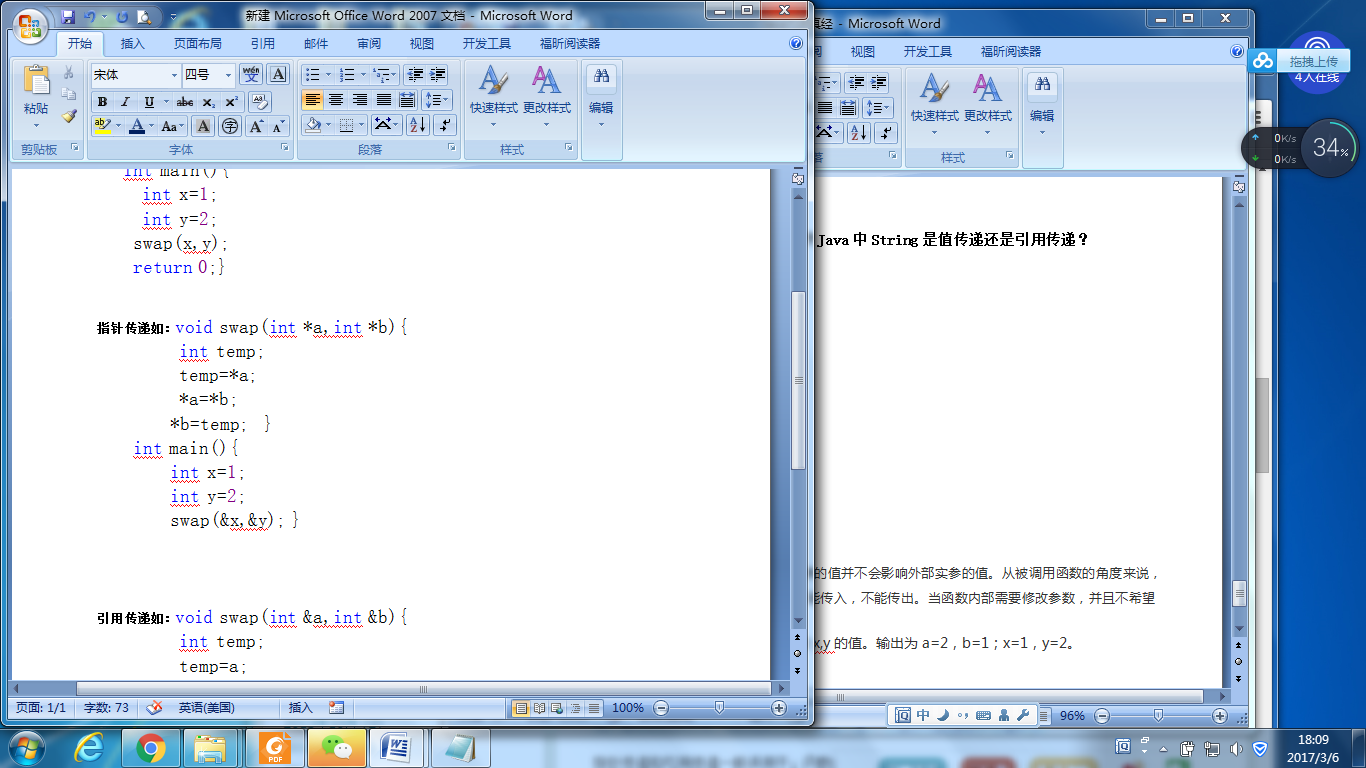
(3)Java稳定，应用广泛

**================ 五、基础知识 =================**

**5.1 数组有没有length( )方法? String有没有length( )方法？**

答：数组没有length()方法，只有length的属性。String有length()方法。

**5.2 请说明值传递和引用传递的区别？Java中String是值传递还是引用传递（引用传递）？**

****

**(1) 值传递：**形参是实参的拷贝，改变形参的值并不会影响外部实参的值。从被调用函数的角度来说，值传递是单向的（实参->形参），参数的值只能传入，不能传出。当函数内部需要修改参数，并且不希望这个改变影响调用者时，采用值传递。

结论：swap只交换了a,b，并不会改变x,y的值。输出为a=2，b=1；x=1，y=2。

**(2) 指针传递：**简单地说，a是一个指向外部实参地址的指针，\*a是指针的内容，如果改变了\*a也必然导致外部实参的改变。

输出结果是a=2，b=1；x=2，y=1;

**(3)引用传递：**形参a,b的地址也与x,y相同，这样一来，交换a,b就相当于交换x,y。

输出结果是a=2，b=1；x=2，y=1;

**总结：**指针传递和引用传递一般适用于：函数内部修改参数能影响调用者。对比值传递，指针/引用传递可以将改变由形参“传给”实参（实际上就是直接在实参的内存上修改，不像值传递将实参的值拷贝到另外的内存地址中才修改）。

int x=1;

int \*y=&x; //用于指针传递，y有自己独立的内存地址，存储的内容是x的地址，\*y是x的值

int &z=x; //用于引用传递，可以理解为z就是x，x就是z，只不过名字不一样

**5.3 如何把string转换成double类型？**

答案：String s="12345";  
double d;  
d=Double.parseDouble(s);

**5.4 switch语句能否作用在byte上，能否作用在long上，能否作用在String上?**

switch能作用在byte、char、short和int上，long不行，JDK1.7后可以作用在String上。

因为switch(expr1)中expr1只能是一个整数表达式或者枚举常量，整数表达式可以是int或integer类型，由于byte、char、short都可以隐式转换为int，但是long、String不符合switch语法，且不能隐式转换为int；

**5.5 什么是Listener模式？**

① **Listener接听器：**它是基于观察者模式设计的，Listener 的设计对开发 Servlet 应用程序提供了一种快捷的手段，能够方便的从另一个纵向维度控制程序和数据；

②事件源经过事件的封装传给监听器，当事件源触发事件后，监听器接收到事件对象可以回调事件的方法。

③Listener是Servlet的监听器，它可以监听客户端的请求、服务端的操作等。通过监听器，可以自动激发一些操作，比如监听在线的用户的数量。

**5.6 分别解释过滤器、拦截器、和监听器是什么？**

**(1)过滤器(**Filter**)：**它是Servlet技术中最实用的技术，实现了javax.servlet.Filter接口的服务器端，主要应用于对客户请求HttpServletRequest预处理(请求到达 Servlet 之前拦截请求)与对HttpServletResponse请求响应的后处理(响应到达客户端之前拦截响应)，是典型的处理链。它随着Web应用程序的启动而启动，只初始化一次，只要在web.xml中文件中配置好要拦截的客户端请求，就可以对请求、响应统一设置编码集 ；

**(2)监听器(Listener):**Servlet中的监听器Listener实现了javax.Servlet.ServletContextListener接口的服务器端程序，也是随着Web应用的启动而启动，只初始化一次，随着Web应用的停止而销毁，主要作用：做一些初始化内容添加工作、设置基本内容，比如监听在线的用户的数量。

**(3)拦截器(Interceptor)：**拦截器是Struts在面向切面编程中应用的，基于JAVA的反射机制，是框架中很多核心功能的实现手段，在访问的action执行之前和之后执行代码，实现某项功能，如：接收接入的参数、数据验证、文件上传、国际化等。

**5.7 java过滤器和拦截器区别？**

(1) 拦截器是基于java的反射机制的，而过滤器是基于函数回调。

(2)拦截器不依赖与servlet容器，过滤器依赖与servlet容器。

(3)拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用。

(4)拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问。

(5)在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次。

(6)过滤器是Servlet中的概念，在Web.xml中配置，而拦截器是Struts/webwork中的概念，在Struts.xml中配置。

**5.8 静态变量和实例变量的区别?**

[Java](http://lib.csdn.net/base/javase)类的成员变量有俩种：一种是被static关键字修饰的变量，叫类变量或者**静态变量**；另一种没有static修饰，为**实例变量。**

|  |  |  |
| --- | --- | --- |
|  | 静态变量 | 实例变量 |
| 语法定义区别 | 静态变量前要加static关键字； | 实例变量前则不加static； |
| 程序运行时的区别 | 静态变量属于类变量，只要程序加载了类的字节码，不用创建任何实例对象，就可以直接使用类名来引用； | 实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，即(new)后才能使用这个实例变量； |
| 存储区别 | 静态变量存储在方法区,属于类所有。 | 实例变量存储在堆当中,其引用存在当前线程栈。 |

**5.9 java当中的四种引用？**

强引用,软引用,弱引用,虚引用.不同的引用类型主要体现在GC上:

(1)强引用：如果一个对象具有强引用，它就不会被垃圾回收器回收。即使当前内存空间不足，JVM也不会回收它，而是抛出 OutOfMemoryError 错误，使程序异常终止。如果想中断强引用和某个对象之间的关联，可以显式地将引用赋值为null，这样一来的话，JVM在合适的时间就会回收该对象

(2)软引用：在使用软引用时，如果内存的空间足够，软引用就能继续被使用，而不会被垃圾回收器回收，只有在内存不足时，软引用才会被垃圾回收器回收。

(3)弱引用：具有弱引用的对象拥有的生命周期更短暂。因为当 JVM 进行垃圾回收，一旦发现弱引用对象，无论当前内存空间是否充足，都会将弱引用回收。不过由于垃圾回收器是一个优先级较低的线程，所以并不一定能迅速发现弱引用对象；

(4) 虚引用：顾名思义，就是形同虚设，如果一个对象仅持有虚引用，那么它相当于没有引用，在任何时候都可能被垃圾回收器回收。

**5.10 hql和sql的区别?**

|  |  |
| --- | --- |
| Sql | Hql(Hibernate Query Language) |
| (1) sql是关系数据库查询语言,面对的数据库  (2) sql操作的是数据库表和字段 | (1) hql是Hibernate这样的数据库持久化框架提供的内置查询语言  (2) 而作为面向对象的hql操作的则是持久化类及其属性  (3) HQL仅用于查询数据，不支持insert, update和delete语句 |

**5.11 hashmap和hashtable区别？**

|  |  |
| --- | --- |
| **HashMap** | **HashTable** |
| (1) 线程不安全，效率高一点  (2) 允许有null的键和值  (3) 方法不是Synchronize的要提供外同步  (4) 有containsvalue和containsKey方法  (5) HashMap 是Java1.2 引进的Map interface 的一个实现  (6) HashMap是Hashtable的轻量级实现 | (1) 线程安全，效率稍低  (2) 不允许有null的键和值  (3) 方法是是Synchronize的  (4) 有contains方法  (5) Hashtable 继承于Dictionary 类  (6) Hashtable 比HashMap 要旧 |

HashMap提供外同步:利用Collections类的静态的synchronizedMap()方法，它创建一个线程安全的Map对象，并把它作为一个封装的对象来返回。

**5.12 LinkedList和ArrayList的区别？**

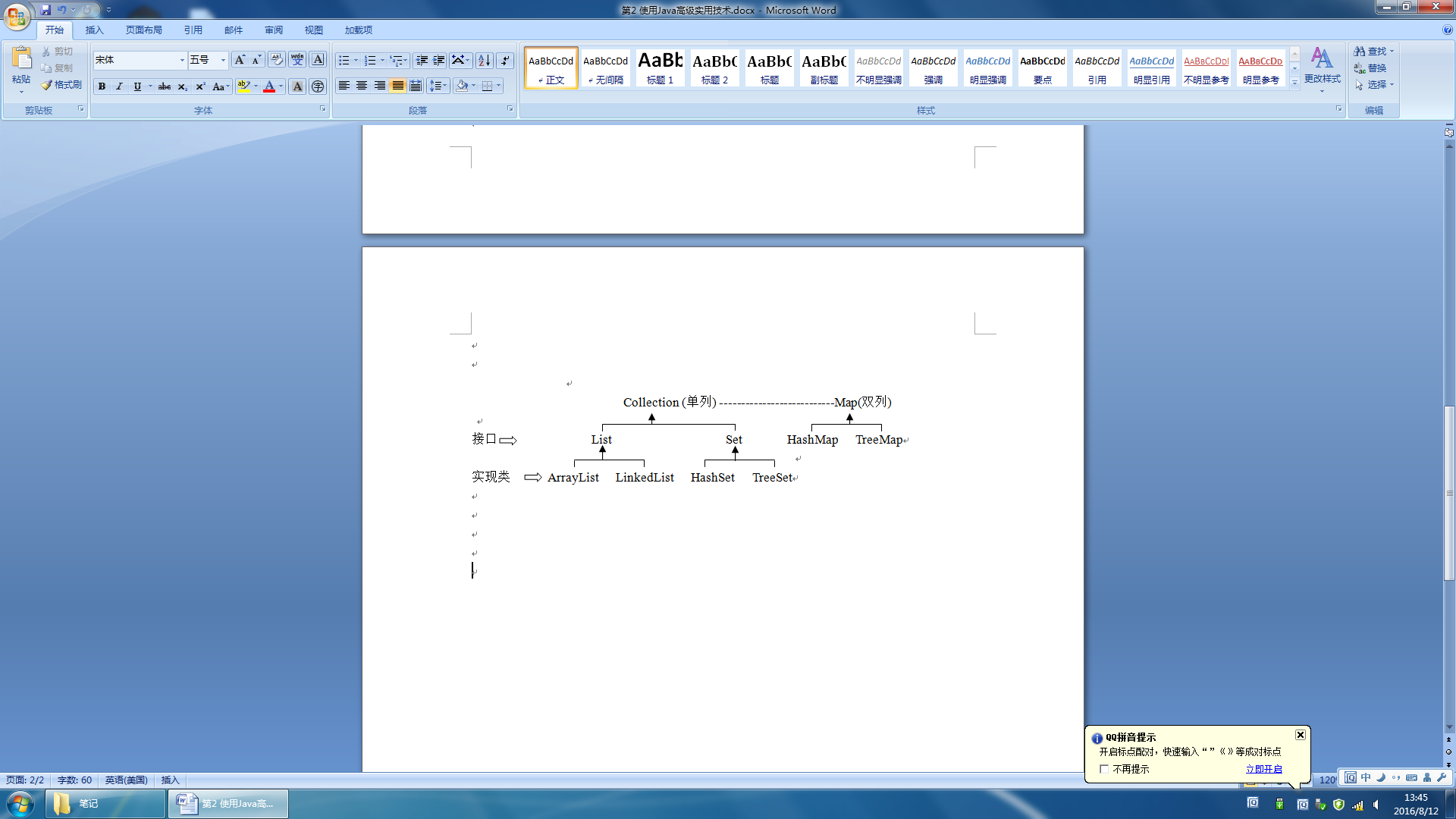
① ArrayList：数组结构的集合，使用数组存储方式-🡪**遍历、随机访问元素效率比较高**；

② LinkedList：链表结构的集合，使用列表式存储-🡪**在中间或前面插入、删除效率高，但查找效率低**；

**5.13 ArrayList和Vector的区别,HashMap和Hashtable的区别？**

(1)ArrayList是JDK1.2的集合类并且线程不安全，Vector是1.0的集合类并且线程安全，二者用法类似。  
 (2)HashMap线程不安全且能放a空键或空值，Hashtable线程安全且不能放空键或空值。

**5.14 list、map、set区别？**

****

相同点：Set、List和Map都处于java.util包中，Set、List和Map都是接口，它们有各自的实现类。 Set的实现类主要有HashSet和TreeSet，List的实现类主要有ArrayList，Map的实现类主要有HashMap和TreeMap。

区别：

(1)List 接口存储一组不唯一、有序(插入顺序)的对象；List中的对象按照索引位置排序，可以有重复对象，允许按照对象在集合中的索引位置检索对象，如通过list.get(i)方式来获得List集合中的元素。

(2)Set接口存储一组唯一、无序的对象；Set中的对象不按特定方式排序，并且没有重复对象。但它的有些实现类能对集合中的对象按特定方式排序，例如TreeSet类，它可以按 照默认排序，也可以通过实现java.util.Comparator<Type>接口来自定义排序方式。

(4)Map接口存储一组键值对象、提供Key(键)到Value(值)的映射；Map中的每一个元素包含一个键对象和值对象，它们成对出现。键对象不能重复，值对象可以重复。

**5.15 heap(堆)和stack(栈)有什么区别？**

JVM的内存区域组成：java把内存分两种，一种是栈内存，另一种是堆内存；

|  |  |
| --- | --- |
| **Heap(堆)** | **Stack(栈)** |
| (1) 堆中放对象；  (2) 堆空间是一个无序的空间；  (3) 堆内存用来存放由new创建的对象和数组以及对象的实例变量。 | (1) 栈中放引用变量；  (2) 栈是先进后出的结构；  (3) 在函数中定义的基本类型变量和对象的引用变量都在函数的栈内存中分配。 |
| **优势：**是堆可以动态分配内存大小，生存期也不必事先告诉编译器，因为它是在运行时动态分配内存的。 | **优势：**(1)是栈存取速度比堆要快，仅次于直接位于CPU中的寄存器。  (2) 存在栈中的数据可以共享。 |
| **缺点：**就是要在运行时动态分配内存，存取速度较慢。 | **缺点：**是存在栈中的数据大小与生存期必须是确定的，缺乏灵活性。 |

**说明：**java的基本数据类型共有8种，即int,short,long,byte,float,double,boolean,

char(注意并没有String)。这种类型的定义是通过诸如int a = 3；long b = 255L；的形式来定义的。如int a = 3；这里的a是一个指向int类型的引用，指向3这个字面值。这些字面值的数据，由于大小可知，生存期可知(这些字面值定义在某个程序块里面，程序块退出后，字段值就消失了)，出于追求速度的原因，就存在于栈中。

**注：**在函数（代码块）中定义一个变量时，java就在栈中为这个变量分配内存空间，当超过变量的作用域后，java会自动释放掉为该变量所分配的内存空间；在堆中分配的内存由java虚拟机的自动垃圾回收器来管理。

**5.16 Integer与int的区别？**

|  |  |
| --- | --- |
| **int** | **Integer** |
| (1) int是基本数据类型；  (2) 基本数据类型存储在栈内存中； | (1) Integer为int的包装类；  (2) 包装类拥有方法和属性； |
| 联系：在JDK1.5后，包装类和基本数据类型可以实现自动转换，包装类可以通过intValue来转换成基本数据类型，也可以通过new Integer( )将基本数据类型转换为包装类。 | |

若Int i=1;int j=1;

➀那么判断条件(i= =j)是否为true? 答：是

➁如果i和j都是integer类型呢？答：是，因为Integer i=10这种赋值方式会自动调用Integer的valueOf()缓存(如果值大于127不会缓存)为基本类型。

**5.17 请说明equals和==的区别？**

|  |  |
| --- | --- |
| **==** | **equals** |
| (1) == 是一个运算符；  (2) 用于基本数据类型的比较，判断内存地址(栈)是否相同；  (3)比较引用地址 | (1) equals则是string对象的方法，可以（.点）出来  (2) 用于判断两个变量是否是对同一个对象的引用，即堆中的内容是否相同，返回值为布尔类型;  (4)比较字符串的值(内容)； |
| 总结：(1)同一对象，"=="和equals结果相同；  (2)不同对象，内容相同，"=="返回false，equals返回true；  (3)数值类型存储在栈内存中，而引用类型的变量在栈中存储的仅仅是引用变量的地址，其内容则是存在堆内存中。 | |

**5.18 字符串的compareTo方法结果是什么？**

答：使用compareTo()方法可以比较字符串与字符串之间的大小关系。如果当前字符串小于指定字符串，则返回一个小于0的数值；否则返回一个大于0的数值；如果两个字符串相等，则返回0。

**5.19 字节流与字符流的区别？**

**字节流：**用于读取或写出二进制数据，比如图片、影像等数据。

**字符流：**用于读取或写出字符数据，比如传输字符串。  
  **注：**所有的数据都可以通过字节流来进行处理，不过如果是字符数据，用字节流还需要进行转换后传输，如果使用字符流可以方便数据的转换。

**5.20 同步和异步有何异同，在什么情况下分别使用他们？**

|  |  |  |
| --- | --- | --- |
|  | **同步** | **异步** |
| 定义 | 指同一时间只能一个线程执行该方法，其他线程需要等待。 | 指多个线程可以同时执行某个方法，并共享同一资源。 |
| 异同 | 同步可以让访问的资源具有安全性，因为同一时间只能一个线程对其进行访问。但是效率不高。 | 异步会对访问的资源造成不稳定，如：多个线程同时访问一个资源，一个在修改、一个在删除、一个在读取，这样可能会造成资源的混乱。但因同时运行，执行效率会提高。 |
| 举例 | 比如：银行取款 | 比如：迅雷看看，一边下载，一边播放。 |

**5.21 简述你对线程、进程的理解？**

|  |  |  |
| --- | --- | --- |
|  | **进程** | **线程** |
| 概念 | (1)进程是系统运行程序的基本单位；  (2)进程(Process)是程序的一次动态执行过程，它对应了从代码加载、执行至执行完毕的一个完整过程，这个过程也是进程本身从产生、发展至消亡的过程。 | (1)线程是进程中执行运算的最小单位；  (2)线程是进程内部的一个执行单元，可完成一个独立的任务的顺序控制流程，如果在一个进程中同时运行了多个线程，用来完成不同的工作，则称之为**多线程。** |
| 特点 | (1)每一个进程都有自己独立的一块内存空间、一组系统资源；  (2)每一个进程的内部数据和状态都是完全独立的； | (1)多线程程序可以带来更好的用户体验，避免因程序执行过慢而导致的计算机死机或者白屏的情况；  (2)多线程程序可以很大程度的提高计算机系统的利用率，如迅雷的多线程下载；  (3)多线程之间抢占资源，并发进行。 |
| 联系 | (1)一个进程在其执行过程中可以产生多个线程，而线程必须在某个进程内执行，不可能有一个线程脱离进程单独存在。  (2)一个进程中至少要有一个线程；  (3)资源分配给进程，同一进程的所有线程共享该进程的所有资源；  (4)处理器分配给线程，即真正在处理机上运行的是线程。 | |

**5.22 定义一个线程有几种方式？有什么区别？**

(1)继承自java.lang.Thread类，extends Thread只能继承这一个类，而使用接口还可以继承其他类；

(2)实现java.lang.Runnable接口，implements Runnable可以实现多线程共享资源；

创建线程的**步骤：a.**定义线程**b.**创建线程对象**c.**调用start启动线程**d.**终止线程**；**

**注：** ➀多线程有两种实现方法：继承Thread类与实现Runnable。  
 ➁接口同步的实现方法有两种： synchronized、wait与notify。

**5.23 启动一个线程是用run( )还是start( )?**

start( )方法启动线程，run( )方法是线程执行的主方法。

**5.24 线程的基本状态以及状态之间的关系？**

**线程生命周期的4种状态**：新生状态、可运行状态、阻塞状态、死亡状态；

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

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

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

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

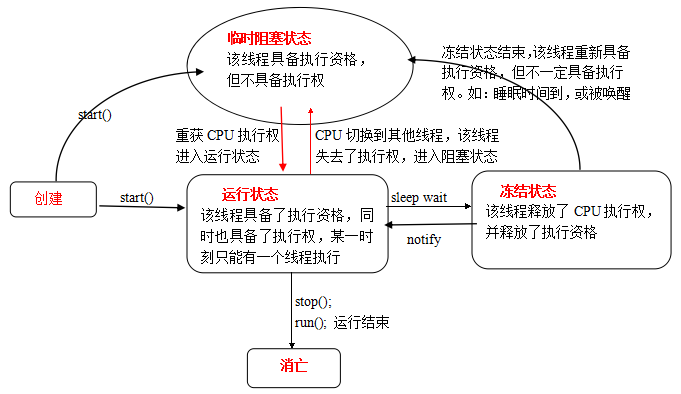
自然终止：正常运行run()方法后终止

异常终止：调用stop()方法让一个线程终止运行

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

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

正在等待：调用wait()方法。（调用motify()方法回到就绪状态）;被另一个线程所阻塞：调用suspend()方法。（调用resume()方法恢复）



**注：**(1)一个程序中可以有多条执行线索同时执行，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，即可以有多段程序代码同时运行，每个程序至少都有一个线

程，即main方法执行的那个主线程。

(2)如果只是一个cpu，它怎么能够同时执行多段程序呢？

答：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；

**5.25 sleep( )和wait( )有什么区别?**

|  |  |
| --- | --- |
| **sleep( )** | **wait( )** |
| (1) sleep是Thread类的方法;  (2) sleep是自动唤醒；  (3) sleep不会释放同步锁；  (4) sleep可以用在任意方法中； | (1) wait是Object类的方法；  (2) wait需要其他线程来唤醒；  (3) wait会释放同步锁；  (4) wait只能用在同步方法或同步块中； |
| 注：Sleep（） 不会释放对象锁 到时自动恢复 ，wait（）会释放对象锁 进入等待此对象的等待锁定池 发出notify（）方法后 才进入等待锁定池准备对象锁的获取进入运行状态。 | |

**5.26 实现线程同步的有哪些方法？**

**(1)同步方法:**即有synchronized关键字修饰的方法。

由于java的每个对象都有一个内置锁，当用此关键字修饰方法时，内置锁会保护整个方法。在调用该方法前，需要获得内置锁，否则就处于阻塞状态。

 如： public synchronized void save(){}

注： synchronized关键字也可以修饰静态方法，此时如果调用该静态方法，将会锁住整个类

**(2)同步代码块:**即有synchronized关键字修饰的语句块。

  被该关键字修饰的语句块会自动被加上内置锁，从而实现同步，如： synchronized(object){ }

注：同步是一种高开销的操作，因此应该尽量减少同步的内容。通常没有必要同步整个方法，使用synchronized代码块同步关键代码即可。

**(3) 使用特殊域变量(volatile)实现线程同步:** 需要同步的变量加上volatile；

**(4) 使用重入锁实现线程同步**

**(5) 使用局部变量实现线程同步**

**5.27 请说出作用域public 、private、protected以及不写时的区别？**

(1)**public：**公共修饰符，表示任意类都可以访问。  
(2)**protected：**为受保护的修饰符，表示同类、同包以及不同包但是父子关系的是可以访问。  
(3)**private：**为私有修饰符，表示只有同类中可以访问，出了这个类就不能访问了。

(4)**不写：**表示默认修饰符friendly，或者称为package修饰符，该修饰符表示只有同类或同包下的类可以访问，出了这个包就不能访问了。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 作用域 | 同一package | 当前类 | 子孙类 | 其他package |
| public | √ | √ | √ | √ |
| protected | √ | √ | √ | × |
| friendly | √ | √ | × | × |
| private | √ | × | × | × |

**5.28 String 、StringBuffer、StringBuilder有什么差别？在什么情况下使用它们？**

|  |  |  |
| --- | --- | --- |
| **String** | **StringBuffer** | **StringBuilder** |
| (1) 字符串常量；  (2) 字符串的基本类，该字符串是不可变的；  (3) 如果要操作少量的数据用 String。 | (1) 字符串变量，存储在堆中；  (2)线程安全；  (3) 多线程下操作大量数据用StringBuffer字符串缓冲区安全； | (1) 字符串变量；  (2)线程不安全；  (3) 单线程下操作大量数用 StringBuilder字符串缓冲区效率高； |
|  | 他们是可改变的对象即可以对字符串的内容进行改变 ，每当我们用它们对字符串做操作时，实际上是在一个对象上操作的; | |

**5.29 解释一下序列化与反序列化？**

**(1)概念：**

**Java序列化**是指把Java对象转换为字节序列的过程；

**Java反序列化**是指把字节序列恢复为Java对象的过程；

**(2)为什么需要序列化与反序列化？**

因为当两个进程进行远程通信时，可以相互发送各种类型的数据，包括文本、图片、音频、视频等， 而这些数据都会以二进制序列的形式在网络上传送。那么当两个Java进程进行通信时，要实现进程间的对象传送这就需要Java序列化与反序列化了。换句话说，一方面，发送方需要把这个Java对象转换为字节序列，然后在网络上传送；另一方面，接收方需要从字节序列中恢复出Java对象。

**(3)如何实现Java序列化与反序列化？**

1)JDK类库中序列化API：

  ➀java.io.ObjectOutputStream：表示对象输出流，它的writeObject(Object obj)方法可以对参数指定的obj对象进行序列化，把得到的字节序列写到一个目标输出流中。

➁java.io.ObjectInputStream：表示对象输入流，它的readObject( )方法源输入流中读取字节序列，再把它们反序列化成为一个对象，并将其返回。

2)实现序列化的**要求**：

只有实现了Serializable或Externalizable接口的类的对象才能被序列化，否则抛出异常。

3）JDK类库中序列化的步骤：

步骤一：创建一个对象输出流，它可以包装一个其它类型的目标输出流，如文件输出流：

ObjectOutputStream out = new ObjectOutputStream(new fileOutputStream(“D:\\objectfile.obj”));

步骤二：通过对象输出流的writeObject()方法写对象：

out.writeObject(“Hello”);

out.writeObject(new Date());

4）JDK类库中反序列化的步骤：

步骤一：创建一个对象输入流，它可以包装一个其它类型输入流，如文件输入流：

ObjectInputStream in = new ObjectInputStream(new fileInputStream(“D:\\objectfile.obj”));

步骤二：通过对象输出流的readObject()方法读取对象：

String obj1 = (String)in.readObject();

Date obj2 = (Date)in.readObject();

**5.30 垃圾回收的优点和原理（并考虑两种回收机制）？**

Java语言中一个显著的特点就是引入了垃圾回收机制，使c++程序员最头疼的内存管理的问题迎刃而解，它使得Java程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java中的对象不再有"作用域"的概念，只有对象的引用才有"作用域"。

**优点：**垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。

**原理：垃圾回收器**通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有**分代复制垃圾回收**和**标记垃圾回收**，**增量垃圾回收**。

**5.31 GC是什么? 为什么要有GC?**

GC是垃圾收集的意思（Gabage 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。

**5.32 java中会存在内存泄漏吗？请简单描述？**

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

**5.33 什么叫java的内存泄露？**

在java中，内存泄漏就是存在一些被分配的对象，这些对象有两个特点，首先这些对象是可达的，即在有向图中存在通路可以与其相连（仍存在该内存对象的引用如:Object o）；其次这些对象是无用的，即程序以后不会再使用这些对象。如果对象满足这两个条件,就可以判定为Java中的内存泄漏，这些对象不会被GC所回收，然而它却占用内存。

如： Vector v=new Vector(10);  
for (int i=1;i<100; i++) {  
 Object o=new Object();  
 v.add(o);  
 o=null;   
}

**5.34 介绍一下final、finally以及finalize方法？**

(1)**final：**修饰符（关键字），可以修饰类（不能被继承）、属性（常量）、和方法（不能被重写）。因此一个类不能abstract修饰，又被final修饰。将变量或方法声明为final，可以保证它们在使用中不被改变。被声明为final的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改。被声明为final的方法也同样只能使用，不能重载。

注：1.被final修饰的类不可以被继承；

2.被final修饰的方法不可以被重写；

3.被final修饰的变量不可以被改变.如果修饰引用,那么表示引用不可变,引用指向的内容可变；

4.被final修饰的方法，JVM会尝试将其内联,以提高运行效率；

5.被final修饰的常量，在编译阶段会存入常量池中；

(2)**finally：**是异常处理块中的代码块，表示无论如何都会执行的代码块。异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。一般异常处理块需要。

(3)**finalize( )：**是Object类的方法，该方法在对象被垃圾回收之前执行的方法。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此Java中所有类都从Object类中继承finalize()方法。子类覆盖 finalize( ) 方法以整理系统资源或者执行其他清理工作。

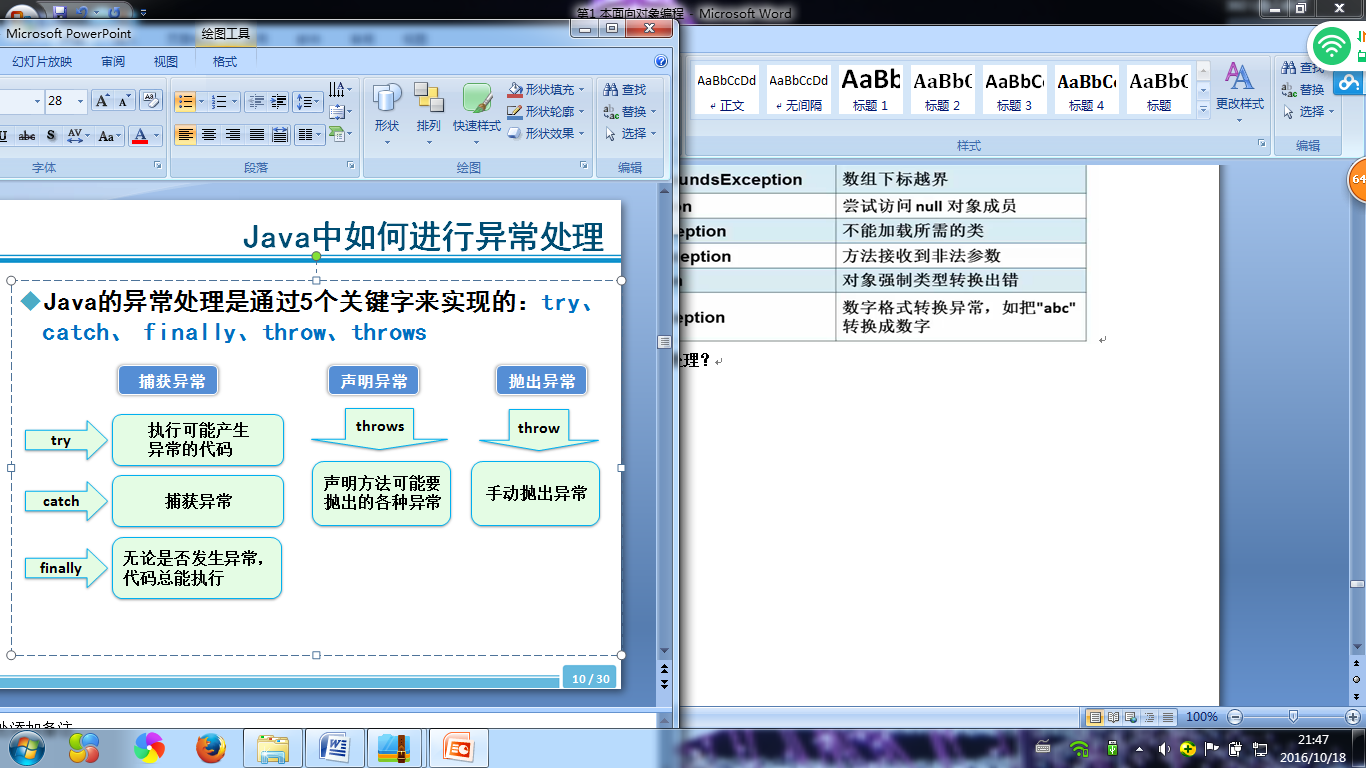
注：**当垃圾回收器(garbage colector)决定回收某对象时，就会运行该对象的finalize()方法。**值得注意的是，finalize( )方法并不能等同与析构函数。Java中是没有析构函数的。但是在Java中很不幸，如果内存总是充足的，那么垃圾回收可能永远不会进行，也就是说filalize( )可能永远不被执行，显然指望它做收尾工作是靠不住的。

**5.35 java语言如何进行异常处理？ throw、throws、try、catch、finally分别代表什么？**

(1)传统处理方式；

(2)异常处理机制：

Java的异常处理是通过5个关键字来实现的：try、catch、 finally、throw、throws



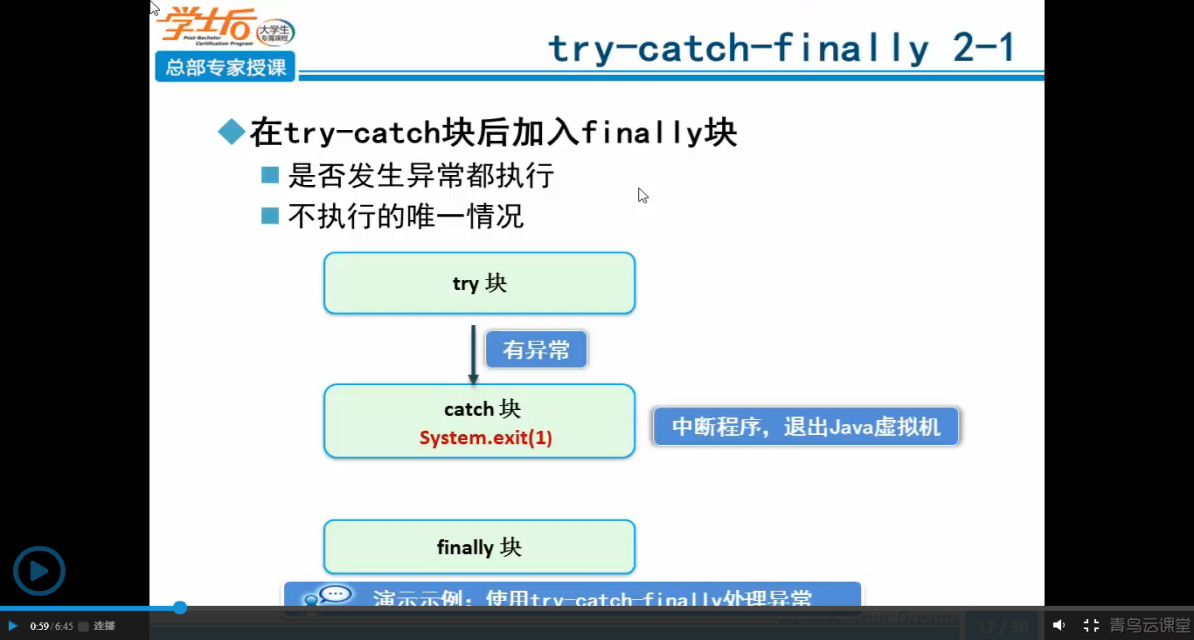
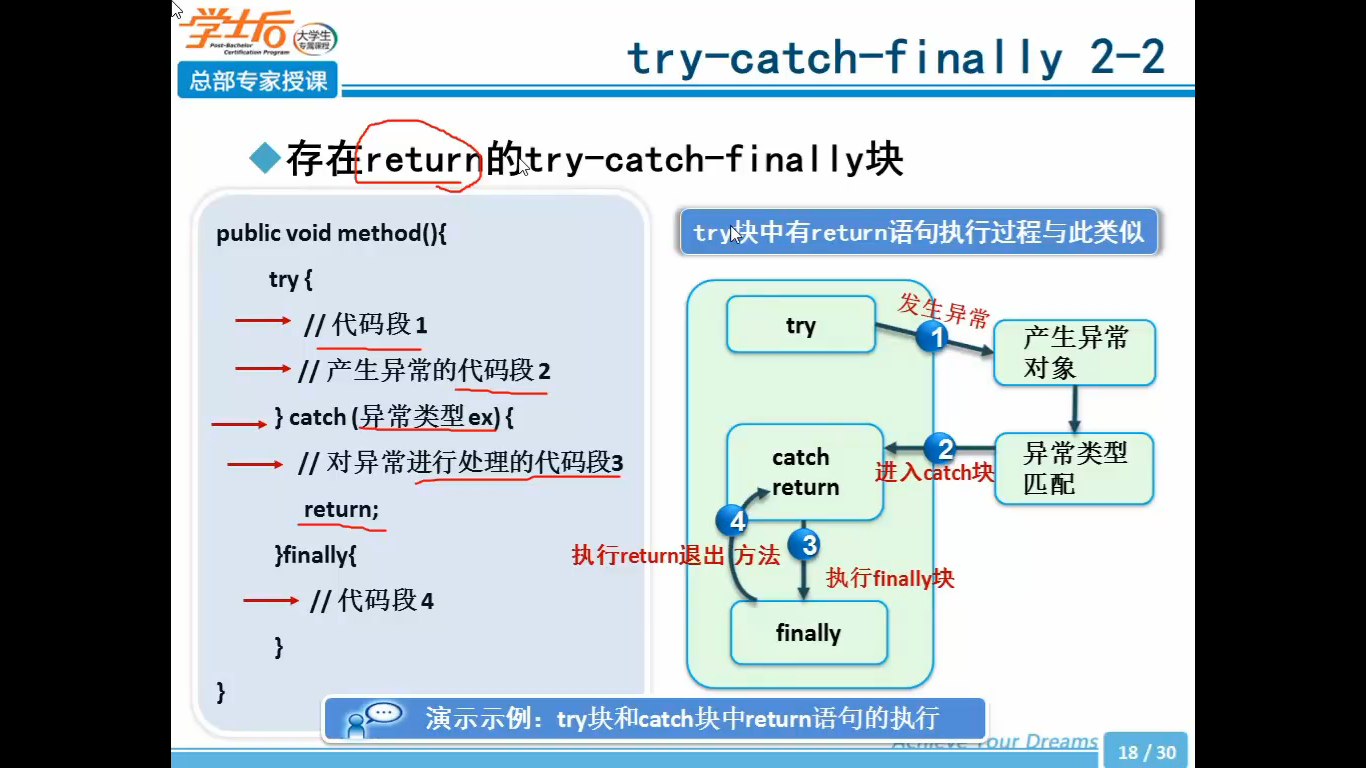
**➀**在**try**语句中对这个异常对象进行捕获。

**➁**产生的异常对象与**catch**语句中各个异常类型进行匹配，如果匹配成功，则执行catch语句中的代码。

**➂ finally**如果没有中途退出最后是一定会执行到的。

**➃throw：**关键字一般用于方法内部，用于手动抛出一个异常类对象；

**➄throws：**关键字通常出现在方法声明中，用来指定该方法可能抛出的异常，跟在方法名后面写，可声明抛出多个异常，异常之间用逗号隔开，调用者可处理异常，也可继续抛出；

****

**5.36 在什么情况下，finally语句不会执行？**

答：如果在try内部执行一条System.exit(0)语句终止应用程序的执行，则finally中的语句不会被执行。

**5.37 error和exception有什么区别?**

**error：**是系统错误很难恢复，代码不能处理的错误，比如内存溢出、堆栈溢出等。  
**exception：**是程序异常，可以通过代码try-catch进行处理，比如空指针异常，数组越界等。

**5.38 throw和throws有什么区别？**

**throw：**关键字一般用于方法内部，用于抛出一个异常类对象，一旦异常被抛出后，throw语句后的程序代码将不会被执行。

**throws：**关键字通常出现在方法声明中，用来指定该方法可能抛出的异常。如果可能有多个异常抛出，可以使用逗号将它们分隔开。

**5.39 说出重写和重载的区别，重写方法是否可以改变返回值的类型?**

答：方法的重写和重载是Java多态性的不同表现。

|  |  |
| --- | --- |
| Override(重写) | Overload(重载) |
| (1) 如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写。  (2) 重写是父类与子类之间多态性的一种表现；  (3)可以改变返回值类型，但改变后的类型必须是修改前类型的子类型，如：父类中Object重写时返回类型为String； | (1)重载：在同一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型；  (2) 重载是一个类中多态性的一种表现；  (3重载的方法是可以改变返回值的类型； |

**5.40 B/S和C/S分别是什么？ 有什么区别？**

**概念：**

**C/S：**是Client/Server的缩写。客户端需要安装专用的客户端软件。

**B/S：**是Brower/Server的缩写。客户机上只要安装一个浏览器。在这种结构下，用户界面完全通过WWW浏览器。

**区别：**

(1)硬件环境不用：C/S一般建立在专用的网络上，小范围里的网络环境；B/S建立在广域网之上的，不必是专门的网络硬件环境；

(2)对安全要求不同： C/S一般面向相对固定的用户群，对信息安全的控制能力很强B/S建立在广域网之上，对安全的控制能力相对弱；

(3)处理问题不同：C/S程序可以处理用户面固定，并且在相同的区域； B/S建立在广域网上，面向不同的用户群；

**5.41 java中常用的包及其作用?**

➀java.lang.\*：提供利用 Java [编程语言](https://www.baidu.com/s?wd=%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzP1IWnHfsuHwWnANhmhPh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErHnLnjmYnjn1njcsrHbLrjDz)进行程序设计的基础类；

➁java.util.\* : 包含集合框架、collection 类、事件模型、日期和时间设施、国际化和各种实用工具类；

➂java.io : 提供了全面的IO接口， 包括：文件读写、[标准设备](https://www.baidu.com/s?wd=%E6%A0%87%E5%87%86%E8%AE%BE%E5%A4%87&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzP1IWnHfsuHwWnANhmhPh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErHnLnjmYnjn1njcsrHbLrjDz)输出等；

➃java.sql.\*：提供使用 JavaTM [编程语言](https://www.baidu.com/s?wd=%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzP1IWnHfsuHwWnANhmhPh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErHnLnjmYnjn1njcsrHbLrjDz)访问并处理存储在数据源（通常是关系数据库）中数据的 API；

**5.42 java关键字有哪些？**

Java语言有51个关键字，其中const和goto虽然被保留但未使用：

(1)数据类型：Boolean    int    long    short    byte    float    double    char    class    interface

(2)流程控制：if     else    do    while    for    switch    case    default    break    continue    return    try

  catch    finally

(3)修饰符： public    protected    private    final    void     static    strictfp     abstract     transient

synchronized     volatile    native

(4)动作：  package    import     throw    throws     extends    implements    this    Super    instanceof    new

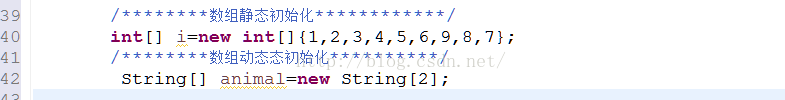
(5)保留字：  true     false    null    goto    const

**5.43 数组的初始化方式有哪些?简述数组的内存分配方式。**

java初始化数组的方式有两种：

➀静态初始化方式：前者是给定数组对象的初始值，长度则由系统根据给定的初始值自行确定；

➁动态初始化方式：先给定数组的长度，对象的初始值由系统分配。例如：



系统按如下规则为数组对象分配初始值：

      (1)整型数据类型byte，short，int，long，数组对象的初始值为0。

      (2) 浮点类型float，double，数组对象的初始值为0.0。

      (3) 字符类型char，数组对象的初始值为'\u0000'。

      (4)布尔类型boolean，数组对象的初始值为false。

      (5)引用数据类型类，接口，数组，数组元素的初始值为null。

**5.44 解释java浮点数据 运算中为何出现精度丢失问题？以及如何解决这个问题？**

因为计算机本身是二进制的，你使用float和double这些节约了内存，对精度就没那么多要求了，要是你对精度有要求，为了能精确的表示、计算浮点数，Java提供了java.math.BigDecimal浮点和java.math.BigInteger 整型两个类用来对超过16位有效位的数进行精确的运算。

**5.45 描述一下JVM加载class文件的原理机制?**  
   答：JVM中类的装载是由ClassLoader和它的子类来实现的,Java ClassLoader 是一个重要的Java运行时系统组件，它负责在运行时查找和装入类文件的类，类装载器所做的工作实质是把类文件从硬盘读取到内存中。一个应用程序总是由n多个类组成，Java程序启动时，并不是一次把所有的类全部加载后再运行，它总是先把保证程序运行的基础类一次性加载到jvm中，其它类等到jvm用到的时候再加载，这样的好处是节省了内存的开销，因为java最早就是为嵌入式系统而设计的，内存宝贵，这是一种可以理解的机制，而用到时再加载这也是java动态性的一种体现 。

**什么是哈希表？**  
   答：哈希表又称散列表，是一种能将关键字映射成存储地址的记录存储技术。要存储数据，先设计一种[**算法**](http://lib.csdn.net/base/31)（哈希函数），然后根据数据记录的关键字计算出各记录的哈希码，这个哈希码作为与关键字相关的记录数据的索引。要取出记录，只需要根据哈希函数重新计算，得到哈希码后，便可直接到相应位置去存取。

**java项目开发流程：**

1、项目启动 (制定项目计划周期)

2、需求调研 (收集客户资料,编写需求文档)

3、系统设计/详细设计(基础平台和应用模块)

4、程序开发

5、测试 试用、培训、维护

**================ 六、手写代码 ================**

**6.1 写一个Singleton出来？**

public class ObserverManager {

private ObserverManager() { //私有的默认构造函数

}

//( 1） 饿汉式单例模式 : 饿汉式单例类.在类初始化时，已经自行实例化

private static ObserverManager mInstance = new ObserverManager(); //已经自行实例化

public static ObserverManager getInstance() {  //静态工厂方法

return mInstance;

}

// (2) 懒汉式单例模式 : 在第一次调用的时候实例化

private static ObserverManager mInstance=null;  //注意，这里没有final

public static ObserverManager getInstance() { //静态工厂方法

if (mInstance == null) { 在多线程的访问情况下 不是线程安全的

mInstance = new ObserverManager();

}

return mInstance;

}

**6.2 写一个工厂模式？**

Public interface IFactory{

public IProduct createProduct();

}

Public Class Factory implements IFactory{

public IProduct createProduct(){

return new Product();

}}

Public class client{

Public Static void main (String [] args){

IFactory factory=new Factory();

IProduct product=factory.createProduct();

product.ProductMethod();}}

**6.3 判断输入的三个数能否构成三角形？**

核心代码：

for(int i=0;i<3;i++){  
 System.out.print("请输入三角形的第"+(i+1)+"条边：");  
 num[i] = scan.nextInt();  
 }  
 int temp;  
 for(int j=0;j<num.length-1;j++){  
 if(num[j]<num[j+1]){  
 temp = num[j];  
 num[j] = num[j+1];  
 num[j+1] = temp;  
 }  
 }  
 int a = num[0];I nt b = num[1]; int c = num[2];  
 if(b+c>a){  
 if(a == b&&b == c){  
 System.out.println("等边三角形");  
 }else if(b == c){  
 System.out.println("等腰三角形");  
 }else if(b\*b+c\*c>a\*a){  
 System.out.println("锐角三角形");  
 }else if(b\*b+c\*c==a\*a){  
 System.out.println("直角三角形");  
 }else if(b\*b+c\*c<a\*a){  
 System.out.println("[钝角三角形](https://www.baidu.com/s?wd=%E9%92%9D%E8%A7%92%E4%B8%89%E8%A7%92%E5%BD%A2&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dhm1c1mWD4n1mYmvfdmyR0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdrjn1rjczrHfknHfdnHRvnjR)");  
 }  
 }else{  
 System.out.println("输入的三个数不能构成三角形");  
 }

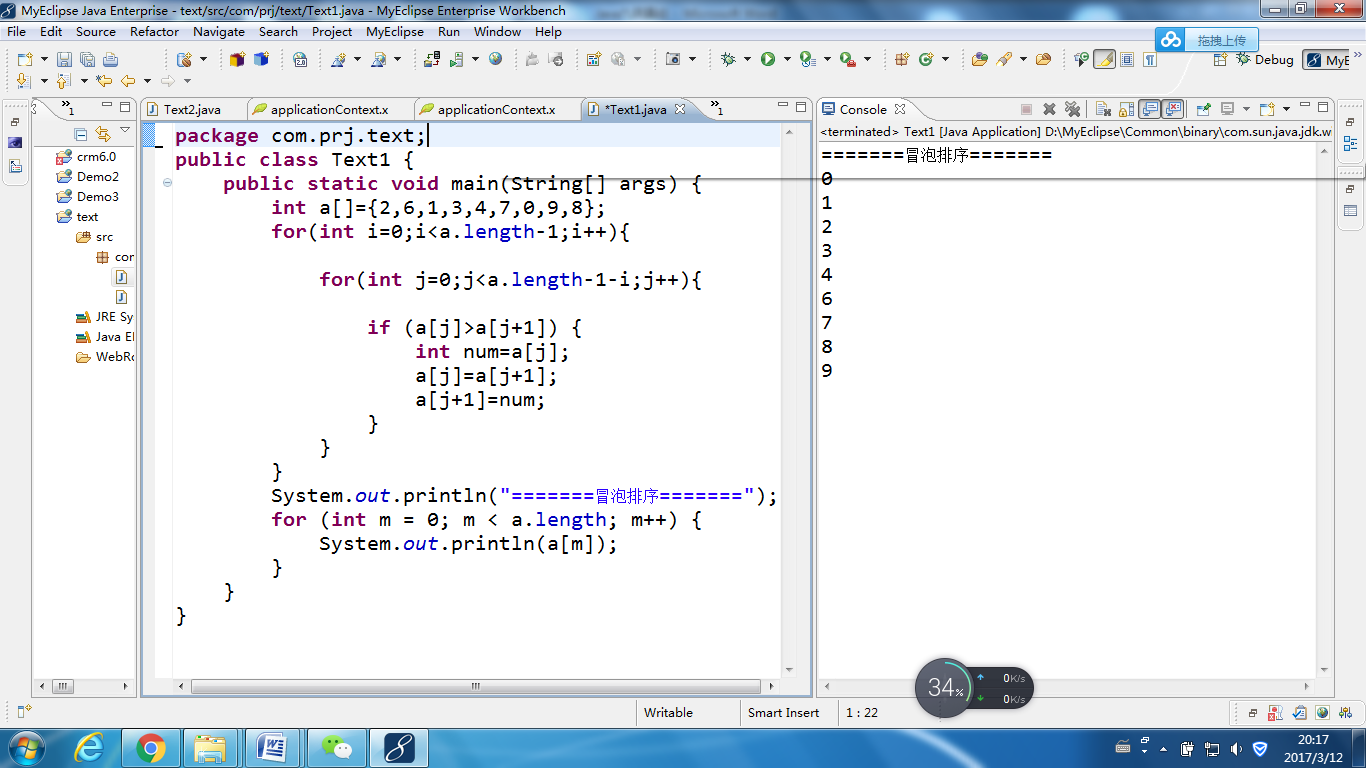
**6.4 判断是否是回文数字的程序？**

public static void main(String[] args) {  
 [Sc](https://www.baidu.com/s?wd=Sc&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykujn1PHD4PHf4nHuBPjwh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH0YPHfYnjnz)anner [sc](https://www.baidu.com/s?wd=sc&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykujn1PHD4PHf4nHuBPjwh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH0YPHfYnjnz)=new [Sc](https://www.baidu.com/s?wd=Sc&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykujn1PHD4PHf4nHuBPjwh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH0YPHfYnjnz)anner(System.in);  
 System.out.println("请输入矩阵1行数：");  
 String a=sc.nextLine();  
 boolean b=false;  
 for(int i=0;i<a.length();i++){  
 if(a.charAt(i)!=a.charAt(a.length()-[i-1](https://www.baidu.com/s?wd=i-1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykujn1PHD4PHf4nHuBPjwh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH0YPHfYnjnz))){  
 b=false;  
 break;  
 }else{  
 b=true;  
 }  
 }  
 if(b){  
 System.out.println(a+"是回文");  
 }else{  
 System.out.println(a+"不是回文");  
 }

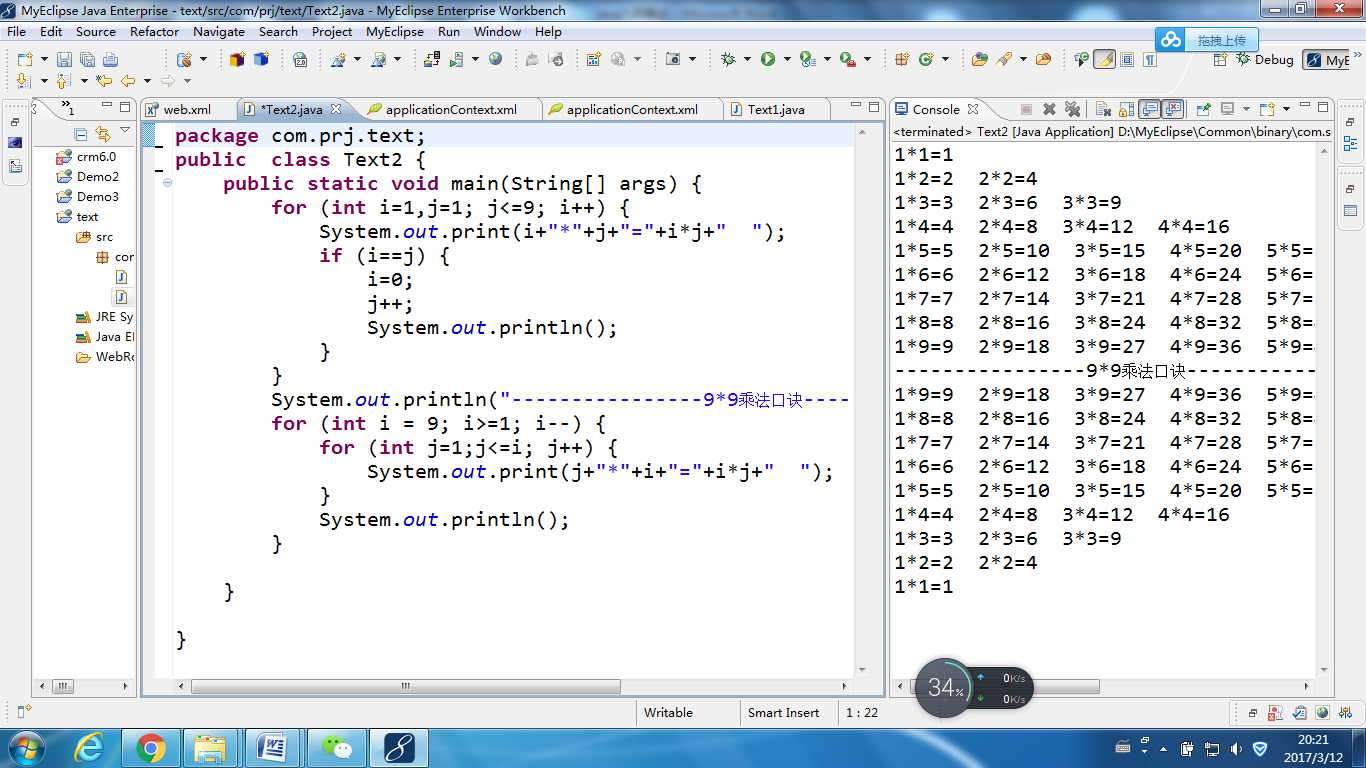
**6.5 选择排序:**

public static void selectSort(int[] numbers) {     
    int size = numbers.length, temp;     
    for (int i = 0; i < size; i++) {     
        int k = i;     
        for (int j = size - 1; j >i; j--)  {     
            if (numbers[j] < numbers[k])  k = j;     
        }     
        temp = numbers[i];     
        numbers[i] = numbers[k];     
        numbers[k] = temp;     
    }     
}

**6.6 冒泡排序：**



**6.6 九九乘法表：**



**6.7 存储过程：**

**create or replace procedure showEmp**(--传递一个员工编号得到相应的名称，薪资，部门名称

eid in number, --out表示输出，in表示输入

ename out varchar2,

sal out number,

depname out varchar2

)

**is**

--申明

nums number(20);--存储人数

depid number(20);--存储部门编号

**begin**

select count(\*) into nums from scott.emp e where e.empno = eid; --查询有无该用户

if nums > 0 then --判断

--a.查询用户名称和薪资

select e.ename,e.sal,e.deptno into ename,sal,depid from scott.emp e where e.empno = eid;

--b.查询部门名称

select d.dname into depname from scott.dept d where d.deptno = depid;

else

--提示

dbms\_output.put\_line('该用户不存在');

end if;

**end;**

----------调用存储过程----------------------

**declare**

a varchar2(20);

b number(20);

c varchar2(20);

**begin**

showEmp(7368,a,b,c);

dbms\_output.put\_line(a);

dbms\_output.put\_line(b);

dbms\_output.put\_line(c);

**end;**

**6.8 两亿条数据如何同时删除？**

 public boolean delUsers(String[] userIds){

·          boolean flag = false;

·          Connection conn = null;

·          PreparedStatement pstmt = null;

·          String sql = "delete from t\_user where user\_id=?";

·          try {

·              conn = DbUtil.getConnection();

·              conn.setAutoCommit(false);

·              pstmt = conn.prepareStatement(sql);

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

·                  pstmt.setString(1,userIds[i].trim());

·                  pstmt.addBatch();

·              }

·              pstmt.executeBatch(); //批量执行

·              conn.commit();//提交事务

·              flag = true;

·          } catch (SQLException e) {

·              try {

·                  conn.rollback(); //进行事务回滚

·              } catch (SQLException ex) {

·              }

·          }finally {

·              DbUtil.close(pstmt);

·              DbUtil.close(conn);

·          }

·          return flag;

·      }

**6.9 如何将1个g的数据最快速度放进C盘中?**

(1)public class Test01 {

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

        getInputStream("C:\\setup.log","D:\\a.txt");

    }

    private static void getInputStream(String pathName,String copyName)throws Exception{

         File file = new File(pathName);

         if(!file.exists())

            throw new RuntimeException("文件不存在呀,你Copy个屁！");

         else{

            getCopy(copyName,new BufferedInputStream(new FileInputStream(file)));

         }

    }

    private static void getCopy(String copyName,BufferedInputStream bis)throws Exception{

         BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(copyName));

         BufferedInputStream biss = bis;

         byte[] b = new byte[biss.available()];

         int len = 0;

         while((len = biss.read(b))!=-1){

             bos.write(b, 0, len);

         }

         bos.close();

         biss.close();

         System.out.println(copyName+"复制成功！");

    }

**6.10 常用SQL语句：**

--------------------创建表----------------------

create table student(

stuid number(20) primary key,

stuname varchar2(20) not null,

stusex varchar2(20) default '男',

classid number(20)

);

----给student表添加字段

alter table student add (stuage number(20));

create table classtable(

cid number(20) primary key,

cname varchar2(20)

);

----------添加外键约束

alter table student add constraint fk\_classid foreign key (classid) references classtable(cid);

------------------新增-------------------------

--insert into 表(字段可省略) values(值);

--commit;事物提交

insert into classtable values(1,'研发20班');

insert into classtable values(2,'研发21班');

insert into classtable values(3,'研发19班');

insert into classtable(cid) values(3);

---------------创建序列

create sequence seq\_cid;

create sequence seq\_stuid;

---------------删除序列

drop sequence seq\_cid;

--属性

--seq\_cid.nextval等价于i++

--seq\_cid.currtval当前序列值

insert into classtable values(seq\_cid.nextval,'研发21班');

commit;

insert into student values(seq\_stuid.nextval,'赵哲','男',1);

insert into student values(seq\_stuid.nextval,'威航飞','男',1);

insert into student values(seq\_stuid.nextval,'张震','男',1);

insert into student values(seq\_stuid.nextval,'张党委','男',2);

insert into student values(seq\_stuid.nextval,'吉明','男',2);

insert into student values(seq\_stuid.nextval,'马萌','男',2);

commit;

---------------------删除数据-----------------------

--and 且--or 或

--sql语句内还可以加入逻辑运算符>, <, >=, <=, !=

delete from classtable;

delete from student;

delete from classtable where cid = 5;

delete from classtable where cid >= 4;

delete from classtable where cid >= 2 and cid<=6;

delete from classtable where cid between 2 and 6;

--当表有主外键关系谁被引用谁是主表

--删除子表数据随意

--删除主表数据不能随意删除，若主表数据被应用则不能被删除，若没被应用可以随意删除

delete from classtable where cid = 3;

------------------------修改--------------------------

--update 表 set要修改的字段=新值,字段=值，字段=值，字段=值 where 条件（省略）

update classtable set cname='研19' where cid = 10;

-----------------------查询--------------------------------

--查询所有

select \* from student;

--条件查询

select \* from classtable where cname='研20' or cid= 10;

select \* from classtable where cname='研20' and cid= 10;

--字段查询

select cid from classtable;

------------排序查询 --asc升（默认） desc降

select \* from student order by stuage asc;

select \* from student order by stuage desc;

----------------给字段起别名

select stuname 姓名,stusex 性别,stuage 年龄 from student;

--------------------给表起别名

select s.stuname,s.stusex,s.stuage from student s;

------------查询字段连接-- || 表示拼接类似java中的+

select stuname||'\*\*\*'||stuage 姓名年龄 from student;

--查询语句中还可以使用运算符

select stuname,stusex,stuage+10 from student;

-----------------oracle中的函数---------------------

-----------------判断字符所在字符串中的位置

select instr('hello','o')\* from dual;

--截取--从第几位开始截取，截取几位数字

select substr('helloworld',3,5)\* from dual;

--四舍五入

select round(3.1455916,2)\* from dual;

------------------获取系统当前时间

select sysdate\* from dual;

-----------------查询日期之间相差月份

select months\_between(sysdate,'23-9月-2015') from dual;

-------------添加月份

select add\_months(sysdate,5) from dual;

--返回下一个日期

select next\_day(sysdate,'星期二') from dual;

--获取当前日期所在月的最后一天的日期

select last\_day('1-10月-2016') from dual;

-------------对指定日期进行四舍五入

----------------------1按照年份,月份，和天的格式四舍五入

select round(sysdate,'year') from dual;

select round(sysdate,'month') from dual;

select round(sysdate+3,'day') from dual;

---------------将数子转为相应格式的字符串

select to\_char(9855.355,'$99999.9') from dual;\*

------------------将字符串转为相应日期

select to\_date('2016-10-10','yyyy-MM-dd') from dual;\*

-----------------将日期转换为字符

Select to\_char(sysdate,'yyyy-MM-dd HH:mm:ss') from dual;\*

Select to\_char(sysdate,'yyyy-MM-dd HH24:mm:ss') from dual;\*

select to\_char(sysdate,'yyyy-MM-dd') from dual;\*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*聚合函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-------------------sum求一列的和

select sum(stuage) from student;

---------------------count求有多少行

select count(\*) from student;

---------------------------------avg平均分

select avg(stuage) from student;

---------------max最大min最小

select max(stuage) 最老 ,min(stuage) 最嫩 from student;

-------------------------求平均分

select round(avg(e.sal),1) from scott.emp e

select \* from scott.emp;

------------------------------模糊查询-----------------------------

select e.ename 名字,e.job 职位,e.hiredate 生日, e.sal 薪资,e.comm 奖金 from scott.emp e;

-- \_匹配一个字符

-- %匹配任意字符

select e.ename 名字,e.job 职位,e.hiredate 生日, e.sal 薪资,e.comm 奖金 from scott.emp e where e.ename like 'SM%';

-----------------------连接查询--------------------------

--内连接分为:等值连接,不等值连接,自然连接

--等值连接

select e.\*,d.dname

from scott.emp e inner join scott.dept d

on e.deptno = d.deptno;

--省略写法

select e.\*,d.dname

from scott.emp e,scott.dept d

where e.deptno = d.deptno;

--不等值连接

create table salgrade(

grade number, --等级编号

losal number, --最低薪资

hisal number --最高薪资

);

insert into salgrade values(1,700,1200);

insert into salgrade values(2,1201,1400);

insert into salgrade values(3,1401,2000);

commit;

select e.ename,e.job,e.sal,s.grade from scott.emp e inner join salgrade s on e.sal between s.losal and s.hisal;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*自然连接\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

select \* from scott.emp natural join scott.dept;

/\*\*\*\*\*\*\*\*\*\*\*\*外连接--可以查询到连接查询看不到的行信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

select \* from scott.emp;

select \* from scott.dept;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*左连接\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

select e.\*,d.dname from scott.dept d left join scott.emp e on e.deptno = d.deptno;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*右连接\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

select e.\*,d.dname from scott.emp e right join scott.dept d on e.deptno = d.deptno;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*交叉连接\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--返回结果集合中的数据行数等于第一表中符合查询条件的数据行数

--乘以第二个表中符合条件的行数

select count(\*) from scott.emp cross join scott.dept;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分页查询\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

select \* from (select 表.\*,rownum r from 表 where rownum<=(当前页数\*每页记录数))

where r>(当前页数-1)\*每页记录数;

如：select \* from (select e.\*,rownum r from scott.emp e where rownum<=(3\*3))

where r>2\*3；

**================ 七、待整理面试题 ================**

1、spring IOC、AOP介绍和原理，AOP代码实现

2、前后台交互方式get、post区别

3、json串转dto方式，list<dto>如何实例化为ArrayList<dto> fastjson:JSON.parseObject(jsonString, Object.class);

4、redis优缺点，比数据库存储过程

5、多线程thread和runnable的区别，编程式多线程和多线程池优缺点

6、object基本方法,wait和notify,hashcode和equals

7、sql查询的一些基本优化

8、zookeeper的使用，为什么？

9、递归是否可用其他代替，for循环和递归哪个复杂度高

10、having的使用，和where一起使用时顺序

11、乐观锁和悲观锁，介绍和实现

1、union和union all，leftjoin使用

2、return后finally是否执行

3、动态代理实现的方式

4、==和equels

5、存储过程参数，如何判断参数是否会返回

6、HashMap原理

7、线程通信，线程同步方法

1、spring组件，web.xml配置，springmvc流程，@Controller是什么类型，dispatchservlet是什么类型

2、面向切面-事务、日志

2、redis数据类型，连接方式，如何实现分布式缓存，优缺点

3、服务治理如何实现的，zookeeper原理

4、rabbitmq原理，消息机制，concurrent并发包

5、负载均衡

6、设计模式

7、quartz实现原理，加锁机制

8、seajs闭包机制

1、dubbo通信协议，负载均衡算法

2、redis序列化的实现，jedispool、c3p0线程池优点

3、MQ消息类型

4、MVC接参方式

5、AOP底层实现原理，事物控制代码实现原理

6、过滤器有什么优点

7、XML解析json工具