1、Tomcat的优化经验

答：去掉对web.xml的监视，把JSP提前编辑成Servlet；有富余物理内存的情况下，加大Tomcat使用的JVM内存。

2、什么是Servlet？

答：可以从两个方面去看Servlet：

a、API：有一个接口servlet，它是servlet规范中定义的用来处理客户端请求的程序需要实现的顶级接口。

b、组件：服务器端用来处理客户端请求的组件，需要在web.xml文件中进行配置。

3、Servlet的生命周期，并说出Servlet和CGI的区别？

答：servlet有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。

这个生存期由javax.servlet.servlet接口中的init、service、destroy方法表达。

web容器加载servlet，生命周期开始。通过调用servlet的init()方法进行servlet的初始化。通过调用service()方法实现

根据请求的不同调用不同的do\*\*()方法。结束服务，web容器调用servlet的destroy()方法。

区别：

Servlet处于服务器进程中，它通过多线程运行service()方法，一个实例可以服务于多个请求，并且实例一般不会被销毁；而

CGI对每个请求都产生一个新的进程，服务完成后就销毁，所有效率低于Servlet。

4、Servlet的基本架构

答：

public class ServletName extends HttpServlet

{

//初始化方法

public void init()

{

……

}

//GET请求处理的方法

public void doGet(HttpServletRequest request,HttpServletResponse response)

{

……

}

//POST请求处理的方法

public void doPost(HttpServletRequest request,HttpServletResponse response)

{

……

}

//结束服务的方法

public void destroy()

{

……

}

}

5、Servlet API中forware()和redirect()的区别？

答：a、浏览器显示

重定向会改变URL地址，请求转发不会改变URL地址。

b、资源共享

重定向不可以资源共享，请求转发可以。

c、功能

重定向可以用URL绝对路径访问其他Web服务器的资源，而请求转发只能在一个Web应用程序内进行资源转发，即服务器内部的一种操作。

d、效率

重定向效率低，相当于再一次请求；请求转发效率相对较高，跳转仅发生在服务器端。

6、什么情况下调用doGet()和doPost()?

答：默认情况是调用doGet()方法，JSP页面中的Form表单的method属性设置为post的时候，调用的为doPost()方法；为get的时候，调用deGet()方法。

7、request.getParameter()和request.getAttribute()的区别？

答：

a、request.getParameter()获取的类型是String；

request.getAttribute()获取的类型是Object

b、request.getPrameter()获取的是POST/GET传递的参数值和URL中的参数；

request.getAttribute()获取的是对象容器中的数据值/对象

c、request.setAttribute()和request.getAttribute()可以发送、接收对象；

request.getParamter()只能接收字符串，官方不开放request.setParamter()（也就是没有这个方法）

setAttribute()和getAttribute()的传参原理：

setAttribute()是应用服务器把这个对象放在该页面所对应的一块内存中去，当你的页面服务器重定向到另外一个页面时，

应用服务器会把这块内存拷贝到另一个页面所对应的那块内存中。这个就可以通过getAttribute()获取到相应的参数值或者对象。

8、JSP有哪些内置对象？作用分别是什么？

答：JSP一共有9个内置对象：

request：负责得到客户端请求的信息，对应类型：javax.servlet.http.HttpServletRequest

response:负责向客户端发出响应，对应类型：javax.servlet.http.HttpServletResponse

session:负责保存同一客户端一次会话过程中的一些信息，对应类型：javax.servlet.http.httpsession

out：负责管理对客户端的输出，对应类型：javax.serlvet.jsp.jspwriter

application:表示整个应用环境的信息，对应类型：javax.servlet.servletcontext

config:表示ServletConfig，对应类型：javax.servlet.servletconfig

exception:表示页面中发生的异常，可以通过它获得页面异常信息，对应类型：java.lang.exception

pagecontext:表示这个JSP页面上下文，对应类型：javax.servlet.jsp.pagecontext

page:表示当前JSP页面本身。

9、JSP有哪些动作？作用分别是什么？

答：JSP一共有以下6中基本动作

JSP：include （当页面被请求的时候引入一个文件）

JSP：forward （将请求转到另一个页面）

JSP：useBean （获得JavaBean的一个实例）

JSP：setProperty （设置JavaBean的属性）

JSP：getProperty （获得JavaBean的属性）

JSP：plugin （根据浏览器类型为Java插件生成object或者embed两种标记）

10、JSP常用指令有哪些？

答：page、include、taglib

a、page指令：定义页面的一些属性，常用属性：

contentType="text/html;charset=utf-8"; 向浏览器端输出数据的编码

pageEncoding="utf-8"; JSP编译成java文件时所用的编码

session="true" 是否自动创建session

b、include指令：引入一个静态的JSP页面

c、taglib指令：引入一个标签库

11、JSP中动态include和静态include的区别？

答 ： a、静态include：语法：<%@ include file="文件名" %>，相当于复制，编辑时将对应的文件包含进来，当内容变化时，不会再一次对其编译，不易维护。

b、动态include：语法：<jsp:include page="文件名">,能够自动检查被包含文件，当客户端对JSP文件进行请求时，会重新将对应的文件包含进来，进行实时的更新。

12、页面间对象传递的方法？

答：request、session、application、cookie等。

13、JSP和Servlet有哪些相同点和不同点，他们之间的联系是什么？

答：JSP是Servlet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偏重于视图，Servlet偏重于业务逻辑。

14、 详细描述MVC。

答：基于java的web应用系统采用MVC设计模型，即用Model（模型）、View（视图）和Controller（控制）分离设计，这是目前web应用服务系统的主流设置方向。

Model：处理业务逻辑的模块。

View：负责页面显示，显示Model的处理结果给用户，主要实现数据到页面的转换过程。

Controller：负责每个请求的分发，把Form数据传递给Model进行处理，处理完成后，把处理结果返回给相应的View显示给用户。

15、MVC各个部分都有哪些技术实现？如何实现？

答：MVC是Model-View-Controller的简写。Model代表应用的业务逻辑（通过JavaBean、EJB组件实现），View是引用的表示面（由JSP产生），Controller是提供应用的处理过程机制

（一般是一个Servelt），通过这种设计模型把应用逻辑、处理过程和显示逻辑分成不同的组件来实现。这些组件可以交互和重用。

16、我们在web应用开发过程中经常遇到输出某种编码的字符，如iso-8859-1等，如何输出一个某种编码的字符串？

答：下面这个例子是将iso-8859-1转换成utf-8编码方式。

public String Translate(String str)

{

String s="";

try{

s=new String(str.getBytes("iso-8859-1"),"UTF-8");

s=str.trim();

}catch(Exception e){

System.err.println(e.getMessage());

}

return s;

}

17、get和post的区别？

a、get是用来从服务器上获取数据，而post是用来向服务器传递数据；

b、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，并且两者用"？"连接，变量之间用"&"连接；

而post是将表单中的数据放在form的数据体中，按照变量与值对应的方式，传递到action所指定的URL。

c、get是不安全的，因为在传输过程中，数据是被放在请求的URL中;而post的所有操作对用户来说都是不可见的。

d、get传输的数据量小，这主要应为受url长度限制;而post可以传输大量的数据，所有上传文件只能用post提交。

e、get限制form表单的数据集必须为ASCII字符；而post支持整个IS01 0646字符集。

f、get是form表单的默认方法。

18、javascript常用的方面？

答：数据校验、页面特殊效果等。

19、常用的web容器和开发工具？

答：容器：Tomcat、Weblogic

工具：Eclipse、Jbuilder

20、JSP的四种范围？

答：a、page是代表一个页面相关的对象和属性。一个页面由一个编译好的java servlet类（可以带有include指令，但不可以带有include动作）表示。这既包括servlet又包括编译成servlet的jsp页面。

b、request是代表与web客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个web组件（由于forware指令和include动作的关系）

c、session是代表与用于某个web客户机的一个用户体验相关的对象和属性。一个web回话也可以经常跨域多个客户机请求。

d、application是代表与整个web应用程序相关的对象和属性。这实质上是跨域整个web应用程序，包括多个页面、请求和回话的一个全局作用域。

21、应用服务器有哪些？

答：BEA Weblogic Server、IBM WebSphere Application Server、Oracle9i Application Server、JBoss、Tomcat。

22、几种回话跟踪技术？

答：cookie、url重写、hidden表单隐藏域、session

23、Servlet执行时一般实现哪几个方法？

答：

public void init(ServletConfig config){}

public ServletConfig getServeltConfig(){}

public String ServletInfo(){}

public void service(HttpServletRequest request,HttpServletResponse response){}

public void destroy(){}

24、简述HttpSession的作用、使用方法，可用代码说明。

答：HttpSession中可以跟踪并存储用户信息，把值设置到属性中，有两个方法：setAttribute("变量/对象","变量/对象");getAttribute("变量/对象");例如：在一个方法中用session.setAttribute("username",username);在session中设置一个属性名为username的变量或者对象，然后可以在同一个session范围内使用session.getAttribute("username"); 来得到username属性或者对象。

25、J2EE是什么？

答：J2EE现在叫做Java EE：是（Java platform,Enterprice Edition 企业版）的简称。企业版本开发和部署服务器应用程序。

J2EE是Sun公司提出的多层（multi-tiered）、分布式（distributed）、基于组件（component-base）的企业级应用模型（Enterprice Application Model）。

在这样一个应用系统中，可按照功能划分不同的组件，这些组件又可在不同的计算机上，并且处理不同的层次（tier）中。所属层次包括客户层（client tier）、web层和组件、business层和组件、企业信息系统（EIS）层。

26、J2EE是技术、平台还是框架？

答：J2EE本身是一个标准，一个为企业分布式应用的开发提供的标准平台。

J2EE也是一个框架，包括JDBC、JNDI、RMI、JMS、EJB、JTA等技术。

27、cookie和session的作用、区别、应用范围。

答：a、cookie数据保存在客户端，session数据保存在服务端。

b、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，相当重要的数据，应该使用session保存到服务端。

c、session会在一定时间内保持在服务器上，但是会占用内存资源，当访问的用户过多，会加重服务器的负载，考虑到减轻服务器的压力，可以将不重要的数据放在cookie中持久的保存。

d、单个cookie保存的数据不能超过4k，很多浏览器都限制站点最多保存20个cookie。

28、JSTL是什么？优点有哪些？

答：JSTL（JSP Standard Tag Library，JSP标准标签库）是一个不断完善的开放源代码的JSP标签库，由四个定制标记库（core、format、xml、sql）和一对通用标记库验证器（ScriptFreeTLV和PermittedTaglibsTLV）组成。优点有：

a、在应用程序服务器之间提供了一致的接口，最大程度的提高了web应用在各应用服务器之间的移植。

b、简化了JSP和web应用程序的开发。

c、以一种统一的方式减少了JSP中scriptlet代码数据，可以达到没有任何scriptlet代码的代码。在我们公司的项目中是不允许任何scriptlet出现在JSP中。

d、允许JSP设计工具与web应用程序开发的进一步集成。相信不久就会有支持JSTL的IDE开发工具出现。

29、J2EE的优越性主要表现在那些方面？

答：a、J2EE基于Java技术，与平台无关。

b、J2EE提供了一个优秀的企业级应用程序框架，对快速高质量的开发系统打下了基础。

c、J2EE提供相当专业的通用软件服务。

d、J2EE拥有开放标准，许多大型公司实现了该规范支持的应用服务器。如：BEA、IBM、Oracle等。

30、JSP页面是如何被执行的？JSP执行效率比Servlet低吗？

答：当客户端向一个JSP页面发出请求时，Web Container将JSP转换成Servlet的源代码（只是第一次请求时），然后编译转化后的Servlet并加载到内存中执行，执行的结果Response到客户端。

JSP只在第一次执行的时候会转化为Servlet，以后的每次执行Web容器都是直接执行编译后的Servlet，所有JSP和Servlet只有第一次执行的时候不一样，JSP慢一点，以后的执行都是相同的。

31、JSP如何处理运行时异常（run-time）exception？

答：可是使用页面的errorPaga属性捕捉没有处理的运行时异常，然后自动转向到一个错误处理页面，代码如下：

<%@page errorPage="错误页面URL"%>

如果在页面请求时出现运行时异常时，以上代码会将代码转向到错误页面，在错误页面里面，可以通过以下代码定义这个页面是错误处理页面：

<%@page isErrorPage="true"%>

这样描述错误信息的Throwable对象就可以在错误页面里面访问到。

32、如果JSP表单元素的值为空，如何避免null出现在页面上？

答：可以写一个简单的函数对空值进行处理，判断值是否为空，如果为空就返回空字符串。实例代码如下：

<%!

String blanknull(String e)

{

return (e==null)?"":s;

}

%>

<%

<input type="text" name="username" value="<%=blanknull(username)%>"

%>

33、如何避免JSP页面自动生成session对象？为什么要这么做？

答：在默认情况下，在对一个JSP页面发出请求时，如果session还没有建立，那么JSP页面会自动为请求创建一个session对象，但是session是比较消耗资源的，如果没有必要保持和使用session，就不应该创建session，例如一些只用来宣传产品的网页，往往没必要使用session来保存信息，可以在JSP中使用page指令进行设置，避免JSP页面为每个请求都自动创建session。实例代码如下：

<%@page session="false"%>

34、在servlet和JSP之间能共享session对象吗？

答：Yes

HttpSession session=request.getsession(false);

session.getAttribute"变量名"); //获取保存在session容器中的变量

35、使用JSP连接数据库连接缓冲池的最好的方法是什么？

答：a、使用JDBC2.0中带有此服务的Driver

b、使用提供此服务的application server

36、在JSP中如何写文本文件？

答：使用PrintWriter对象：

<%@page improt="java.io.\*"%>

<%

String str="HelloWorld!";

File file=new File("1.txt");

file.createNewFile();

try{

FileOutPutStream fops=new FileOutPutStream(file);

PrintWriter pw=new PrintWriter(fops);

pw.println(str);

pw.flush();

pw.close();

}

%>

37、JSP的缺点？

答：a、大多数的Servlet引擎不支持connection pooling。

b、Servlet引擎没有标准。

c、JSP与其他脚本语言的交互。

d、不易对Java程序进行调试。

38、在JSP页面中如何删除一个cookie？

答：<%

Cookie cookie=new Cookie("cookie名"，"cookie值");

cookie.setMaxAge(0);

cookie.setPath("/");

response.addCookie(cookie);

%>

39、如何实现servlet的单线程模式？

答：在JSP中使用page指令进行设置，具体代码如下：

<%@page isThreadSafe="false"%>

40、HttpServlet类中主要方法有哪些？

41、XML文档定义有几种形式？他们之间有何本质区别？解析XML文档有哪几种方式？

答：a、两种形式:DTD、schema

b、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。）

c、有DOM、SAX、STAX等。

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

SAX：是事件驱动型的XML解析方式。它顺序读取XML文件，不需要一次全部装载这个XML文件。当遇到向文件开头、文件结束或者标签开头、标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理XML文件，适合对XML的顺序访问。

STAX：Streaming API for XML

42、你在项目中用到了XML技术的哪些方面？如何实现的？

答：用到了数据存储、信息配置两方面。

在做数据交换平台时，将不能数据源的数据组装成XML文件，然后将XML文件压缩打包加密后通过网络传送给接受者，接收解密与解压缩后再同XML文件中还原相关信息进行处理。

在做软件配置时，利用XML可以和方便的进行，软件的各种配置参数都存储在XML文件中。

43、EL表达式的功能，为什么要用EL表达式？（Expression Language）

答：功能：a、从四个域对象中取出数据数据显示。

b、取出请求参数数据显示。

原因：

在页面中用jsp脚本和jsp表达式来获取数据显示比较麻烦

a、需要判断

b、可能需要强转

44、JSTL的功能，为什么要用JSTL？（JavaServer Pages Standard Tag Library）

答：功能：主要用于基本输入输出、流程控制、循环、XML文件剖析、数据库查询及国际化和文字格式标准化的应用等。

原因：在jsp页面做条件判断或者循环操作并输出时，比较费力。

45、为什么要使用自定义标签？MyTag如何实现？

答：原因：

a、不想在jsp中编写java代码

b、JSTL标签库不能满足实际项目的需求

实现的流程：

a、编写标签处理器类（SimpleTagSupport的实现类）

重写doTag()

b、编写标签库文件（WEB-INF/xxx.tld）

整个文件的定义：<short-name> uri

标签的定义：<tag>

c、在jsp页面中使用标签：

导入标签库（xxx.tld）

使用标签

46、说说自动登录功能的编码实现？

答：a、登录功能是用session实现的，就是向session对象中保存当前用户的对象。

b、自动功能是用cookie实现的，就是登录时将用户的信息保存为持久cookie。

c、下次访问时，读取请求中如果有用户信息的cookie就可以自动登录。

47、如何防止表单重复提交？

答：使用session技术：

a、在regist.jsp页面中生成一个为一个随机值，将其保存到session中，同时将其保存为表单的隐藏域的值。

b、在处理注册的请求时，获取session中的值，获取请求参数的值，比较两者是否相同，如果相同说明不是重复提交，请求通过同时删除session中保存的的值，如果不相同则是重复提交，不能通过。

48、Request对象的主要方法有哪些？

答：setAttribute(String name,Object)：设置名字为name的request的参数值

getAttribute(String name)：返回由name指定的属性值

getAttributeNames()：返回request对象所有属性的名字集合，结果是一个枚举的实例

getCookies()：返回客户端的所有Cookie对象，结果是一个Cookie数组

getCharacterEncoding()：返回请求中的字符编码方式

getContentLength()：返回请求的Body的长度

getHeader(String name)：获得HTTP协议定义的文件头信息

getHeaders(String name)：返回指定名字的request Header的所有值，结果是一个枚举的实例

getHeaderNames()：返回所以request Header的名字，结果是一个枚举的实例

getInputStream()：返回请求的输入流，用于获得请求中的数据

getMethod()：获得客户端向服务器端传送数据的方法

getParameter(String name)：获得客户端传送给服务器端的有name指定的参数值

getParameterNames()：获得客户端传送给服务器端的所有参数的名字，结果是一个枚举的实例

getParametervalues(String name)：获得有name指定的参数的所有值

getProtocol()：获取客户端向服务器端传送数据所依据的协议名称

getQueryString()：获得查询字符串

getRequestURI()：获取发出请求字符串的客户端地址

getRemoteAddr()：获取客户端的IP地址

getRemoteHost()：获取客户端的名字

getSession([Boolean create])：返回和请求相关Session

getServerName()：获取服务器的名字

getServletPath()：获取客户端所请求的脚本文件的路径

getServerPort()：获取服务器的端口号

removeAttribute(String name)：删除请求中的一个属性

49、Tomcat根目录下有哪些文件？

答：a、config：配置文件存放路径

b、webapps：项目部署的目录

c、bin Tomcat运行需要的脚本与jar的目录

d、lib：运行项目时所需要的jar的目录

e、word：部署项目的缓存目录

f、temp：临时文件的存放目录

g、logs：记录日志的目录

50、什么是Tomcat，怎样启动停止，配置文件，日志文件的存储？

答：Tomcat是一种web服务器，java编写的web项目可以部署在上面，用户在客户端请求时，都是将请求发到Tomcat上，Tomcat在将请求发到对应的项目上。

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下：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下：右键选择stop server

配置文件在Tomcat的config目录下

日志文件在Tomcat的logs目录下

51、实例化servlet有几种方式？

答：有两种方式

a、第一次请求时，实例化servlet对象

b、在web.xml文件中，在servlet标签中添加loadon-startup标签，tomcat启动时就会实例化servlet对象。

52、JSP乱码如何解决？

答：a、JSP页面乱码

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

b、表单提交时出现乱码

request.setCharacterEncoding("utf-8");

c、数据库出现乱码

jdbc:mysql://localhost:3306:/user?useSSL=false&useUnicode=true&characterEncoding=utf-8;

其实我一般的处理的方法就是配置一个过滤器对每个JSP页面进行字符集处理。

53、session和application的区别？

答：a、两者的作用范围不同

session是用户级别的，application是web应用程序级别的，是一个全局作用域。

一个用户一个session，每个用户的session不同，用户所访问的网站多个页面共享同一个session。

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，但是一个web应用程序的多个用户之间共享一个application。

b、生命周期不同

session：用户首次在网站中发出请求创建session，用户注销、离开网站或者关闭浏览器消亡。

application：启动web服务器创建，关闭web服务器销毁。

54、在Eclipse中调试时，怎样查看一个变量的值？

答:在要查看的变量前先设置断点，然后选中变量，右键选择debug as →Java Application，打开debug视图，这时在Variables窗口中可以看到变量当前的值。

如果是局部变量，也可以在局部变量窗口中查看。要知道一个方法被调用的方法调用链，可以在方法栈中查看。

55、你们公司使用的代码配置管理工具是什么？

答：除了说以前使用的CVS,现在新项目使用SVN了，还简要说一下使用的过程，如果有可能，还说说仓库的概念和如何使用锁之类的细节。

56、你们项目总金额多少，多少人开发，总共花了多少个月？

答：像巴巴运动网这种规模的项目，可以说是4、5个人开发了四五个月，费用则是四五十万。按每人每月两万收入去计算就差不多了。

57、什么是重量级？什么是轻量级？

答：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对象。

重量级意味着不能随意的创建和销毁它的实例，会占用太多的资源。

58、如何从form表单中获取checkbox的值？

答：可以在页面将checkbox的name属性设置相同，value属性取每个条目的id，在后台使用request.getParametervalues（"name"）;获取一组值。

59、过滤器有哪些作用？

答：a、可以验证客户是否来自可信的网络

b、可以对客户提交的数据进行重写编码

c、可以从系统里获得配置的信息

d、可以过滤掉客户某些不应该出现的词汇

e、验证客户是否登录

f、验证客户的浏览器是否支持当前的应用

g、可以记录系统的日志

60、过滤器的用法？（对客户端的请求统一编码和对客户端进行验证）

答：实现Filter接口，同时重写Filter接口的三个方法：

init(FilterConfig config) 用于获取FilterConfig对象

doFilter(ServletRequest request,ServeltResponse response,FilterChain chain) 进行过滤处理一些业务

destroy() 销毁Filter

61、C/S与B/S的区别？

答：有如下8个方面的不同：

a、硬件环境的不同

C/S：一般建立在专用的网络上，小范围里的网络环境，局域网之间再通过专门服务器提供连接和数据交换服务。

B/S：建立在广域网之上的，不必是专门的网络硬件环境，有比C/S更强的适应范围，一般只要有操作系统和浏览器就行。

b、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。一般高度机密的信息系统采用C/S结构适宜。

B/S：建立在广域网之上，对安全的控制能力相对弱，可能面向不可知的用户。

c、对程序架构不同

C/S：程序可以更加注重流程，可以对权限多层次校验，对系统运行速度可以较少考虑。

B/S：对安全以及访问速度的多重的考虑，建立在需要更加优化的基础之上。

d、软件重用不同

C/S：程序可以不可避免的整体性考虑，构件的重用性不如在B/S要求下的构件的重用性好.

B/S：对的多重结构，要求构件相对独立的功能。

e、系统维护不同

C/S：程序由于整体性，必须整体考察，处理出现的问题以及系统升级。

B/S：构件组成，方面构件个别的更换,实现系统的无缝升级。

f、处理问题不同

　　C/S：程序可以处理用户面固定，并且在相同区域，安全要求高需求，与操作系统相关。

　　B/S：建立在广域网上，面向不同的用户群，分散地域，这是C/S无法作到的。

g、用户接口不同

　　C/S：多是建立的Window平台上，表现方法有限，对程序员普遍要求较高。

　　B/S：建立在浏览器上，有更加丰富和生动的表现方式与用户交流；并且大部分难度减低,减低开发成本。

h、信息流不同

　　C/S：程序一般是典型的中央集权的机械式处理，交互性相对低。

　　B/S：信息流向可变化，B-B B-C B-G等信息、流向的变化，更像交易中心。

62、什么是web容器？

答：给处于其中的应用程序组件（JSP、Servlet）提供一个环境，是JSP、Servlet直接跟容器中的变量交互，不必关注其他系统问题。

主要有web服务器来实现。例如：tomcat、weblogic、sphere、JBoss等。该容器提供的接口严格遵守J2EE规范中的web application标准。

我们把遵守以上标准的web服务器叫做J2EE的web容器。

63、什么是EJB容器？

答：Enterprice Java Bean容器。它提供给运行在其中的组件EJB各种管理功能。只要满足J2EE规范的EJB放入该容器，马上就会被容器

进行高效率的管理。并且可以通过现成的接口来获得系统级别的服务。例如：邮件服务、事务管理……

64、开发中都用到了哪些设计模式？用在什么场合？

答：每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，你可以无数次的使用

哪些已有的解决方案，无需再重复相同的工作。主要用到了MVC设计模式。用来开发JSP/Servlet或者J2EE的相关应用。简单工厂模式等。

65、说说你所熟悉的或听说过的J2EE中的几种常用模式？及对设计模式的一些看法。

答：Session Facade Pattern：使用SessionBean访问EntityBean。

Message Facade Pattern：实现异步调用

EJB Command Pattern：使用Command.JavaBeans取代SessionBean，显示轻量级访问。

Data Transfer Object Factory：通过DTO Factory简化EntityBean数据提供特性

Generic Attribute Access:通过AttributeAccess接口简化EntityBean数据提供特性

Business Interface：通过远程（本地）接口和Bean类实现相同接口规范业务逻辑一致性

EJB架构的设计好坏将直接影响系统的性能、可扩展性、可维护性、组件可重用性及开发效率。项目越复杂，项目队伍余越庞大，则越能体现良好设计的重要性。

66、J2EE常用的设计模式？说明工厂模式。

答:Java中一共有23中设计模式：

Factory（工厂模式）、Builder（建造模式）、Factory Method（工厂方法模式）、ProtoType（原始模型模式）、Singleton（单例模式）、

Facade（门面模式）、Adapter（适配器模式）、Bridge（桥梁模式）、Composite（合成模式）、Decorator（装饰模式）、

FlyWeight（享元模式）、Proxy（代理模式）、Command（命令模式）、Interpreter（解释器模式）、Visitor（访问者模式）、

Iterator（迭代子模式）、Mediator（调停者模式）、Memento（备忘录模式）、Observer（观察者模式）、State（状态模式）、

Strategy（策略模式）、Template Method（模板方法模式）、Chain Of Responsibility（责任链模式）、

工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某个类的实例，通常一组类中有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

67、EJB2.0有哪些内容？分别用在什么场合？EJB2.0和EJB2.1的区别？

答：规范内容包括Bean提供者，应用程序装配者，EJB容器，EJB配置工具，EJB服务提供者，系统管理员。这里面，EJB容器是EJB之所有能够运行的核心。

EJB容器管理着EJB的创建，撤消，激活，去活，与数据库的连接等等重要的核心工作。

68、EJB与JAVA BEAN的区别？

答:Java Bean 是可复用的组件，对Java Bean并没有严格的规范，理论上讲，任何一个Java类都可以是一个Bean。但通常情况下，

由于Java Bean是被容器所创建（如Tomcat)的，所以Java Bean应具有一个无参的构造器，另外，通常Java Bean还要实现

Serializable接口用于实现Bean的持久性。Java Bean实际上相当于微软COM模型中的本地进程内COM组件，它是不能被跨进程访问的。

Enterprise Java Bean 相当于DCOM，即分布式组件。它是基于Java的远程方法调用（RMI）技术的，所以EJB可以被远程访

跨进程、跨计算机）。但EJB必须被布署在诸如Webspere、WebLogic这样的容器中，EJB客户从不直接访问真正的EJB组件，

而是通过其容器访问。EJB容器是EJB组件的代理，EJB组件由容器所创建和管理。客户通过容器来访问真正的EJB组件。

69、EJB的基本架构

//Remote Interface接口的代码

package Beans;

import javax.ejb.EJBObject;

import java.rmi.RemoteException;

public interface Add extends EJBObject {

//some method declare

}

//Home Interface接口的代码

package Beans;

import java.rmi.RemoteException;

import jaax.ejb.CreateException;

import javax.ejb.EJBHome;

public interface AddHome extends EJBHome {

//some method declare

}

//EJB类的代码

package Beans;

import java.rmi.RemoteException;

import javax.ejb.SessionBean;

import javx.ejb.SessionContext;

public class AddBean Implements SessionBean {

//some method declare

}

70、存储过程和函数的区别

答：存储过程是用户定义的一系列sql语句的集合，涉及特定表或者其他对象的任务，用户可以调用存储过程，而函数通常是数据库

已定义的方法，它接受参数并返回某种类型的值并且不涉及特定用户表。

71、什么是事务？

答:事务时作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为ACID（原子性、一致性、隔离性和持久性）属性，

只有这样才能成为一个事务：

原子性：事务必须是原子工作单元，对于其数据修改，要么全都执行，要么全都不执行。

一致性：事务在完成时，必须使所有的数据保持一致的状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如B树索引或双向链表）都必须是正确的。

隔离性：由并发事务所做的修改必须与任何其他并发事务所做的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前

的状态，要么是另一并发事务修改它之后的状态，事务不会查看中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播

一系列事务，以使数据结束时的状态与原始事务执行的状态相同。

持久性：事务完成后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

72、触发器分为事前触发和事后触发，两种触发有什么区别？语句级触发和行级触发有什么区别？

答：事前触发运行于触发事件发生之前，而事后触发运行于触发事件发生之后。通常事前触发器可以获取事件之前和新的字段值。

语句级触发可以在语句执行前或后执行，而行级触发在触发器所影响的每一行触发一次。

73、EJB容器提供的服务

答：主要提供声明周期管理、代码产生、持续性管理、安全、事务管理、锁和并发行管理等服务。

74、EJB的三个角色和三个对象

答：角色：EJB角色主要包括Bean开发者、应用组装者、部署者、系统管理员、EJB容器提供者、EJB服务器提供者。

对象：Remote（Local）接口、Home（LocalHome）接口、Bean类

75、EJB的几种类型？

答：会话（Session） Bean，实体（Entity）Bean、消息驱动的（Message Driven）Bean

会话Bean又可分为有状态（Stateful）和无状态（Stateless）

实体Bean可分为Bean管理的持续性（BMP）和容器管理的持续性（CMP）两种

76、Bean实例的生命周期

答：对于Stateless Session Bean、Entity Bean、Message Driven Bean一般存在缓冲池管理，而对于Entity Bean和

Stateful Session Bean存在Cache管理，通常包含创建实例，设置上下文、创建EJB Object（create）、业务方法调用、

remove等过程，对于存在缓冲池管理的Bean，在create之后实例并不从内存清除，而是采用缓冲池机制不断重用实例；

而对于存在Cache管理的Bean，则通过激活或去激活机制保持Bean的状态并限制内存中实例数量。

77、激活机制

答：以Stateful Session Bean为例：其Cache大小决定了内存中可以同时存在的Bean实例的数量，根据MRU或NRU算法，

实例在激活和去激活状态之间迁移，激活机制是当客户端调用某个EJB实例业务方法时，如果对应的EJB Object发现自己

没有捆绑对应的Bean实例，则从其去激活Bean存储中（通过序列化机制存储实例）回复（激活）此实例。状态变迁前会

调用对应的EJB Active 和EJB Passivate方法。

78、remote接口和home接口的作用

答：remote接口中定义了业务方法，用于EJB用户调用业务方法。

home是EJB工厂用于创建、移除和查找EJB实例

79、客服端调用EJB对象的几个基本步骤

答:a、设置JNDI服务工厂以及JNDI服务地址系统属性

b、查找home接口

c、从home接口调用create方法创建remote接口

d、通过remote接口调用其业务方法

1 Ajax你以前用过么?简单介绍一下

AJAX = 异步 JavaScript 和 XML。

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、开心网等等。

2 Ajax可以做异步请求么?

可以.ajax请求默认是异步的.如果想同步 把 async设置为false就可以了默认是true

如果是jquery

$.ajax({

url: some.php,

async: false,

success : function(){

}

});

如果是原生的js

xmlHttp.open("POST",url,false);

3 一网站如果有大量的人登陆访问。那么会产生很多的session，如果你是程序员你该怎么办。

session默认保存在内存中，内存资源宝贵，session数据量大导致内存利用率高，以下方案解决session内存存储问题：

1、可以设置session超时时间，达到超时时间session自动清空

<session-config>

<session-timeout>20</session-timeout>

</session-config>

2、将session中的数据序列化到硬盘中

3、不使用session，使用cookie（此方法存在安全性问题）

4 Jsp有哪些内置对象？作用分别是什么？

Page，pageContext，request，response，session，application，out，config，exception

Page指的是JSP被翻译成Servlet的对象的引用.

pageContext对象可以用来获得其他8个内置对象,还可以作为JSP的域范围对象使用.pageContext中存的值是当前的页面的作用范围》

request代表的是请求对象,可以用于获得客户机的信息,也可以作为域对象来使用，使用request保存的数据在一次请求范围内有效。

Session代表的是一次会话，可以用于保存用户的私有的信息,也可以作为域对象使用，使用session保存的数据在一次会话范围有效

Application：代表整个应用范围,使用这个对象保存的数据在整个web应用中都有效。

Response是响应对象,代表的是从服务器向浏览器响应数据.

Out:JSPWriter是用于向页面输出内容的对象

Config：指的是ServletConfig用于JSP翻译成Servlet后 获得Servlet的配置的对象.

Exception:在页面中设置isErrorPage=”true”，即可使用，是Throwable的引用.用来获得页面的错误信息

5 什么是cookie？Session和cookie有什么区别？

Cookie是会话技术,将用户的信息保存到浏览器的对象.

Session也是会话技术,将Session的信息保存到服务器的对象.Session是基于Cookie的 利用Cookie向浏览器回写JSessionID.

2.6 ajax如何实现异步定时5秒刷新？

setInterval(function() {

$("#content").load(location.href+" #content>\*","");

}, 5000);

1、jsp和servlet的区别、共同点、各自应用的范围？？

JSP是Servlet技术的扩展，本质上就是Servlet的简易方式。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。在struts框架中,JSP位于MVC设计模式的视图层,而Servlet位于控制层.

2、cookie和session的作用、区别、应用范围，session的工作原理？？？

Cookie:主要用在保存客户端，其值在客户端与服务端之间传送，不安全，存储的数据量有限。

Session:保存在服务端，每一个session在服务端有一个sessionID作一个标识。存储的数据量大，安全性高。占用服务端的内存资源。

3、jstl是什么？优点有哪些？？

JSTL（JSP Standard　Tag　Library　，JSP标准标签库)是一个不断完善的开放源代码的JSP标签库，由四个定制标记库（core、format、xml 和 sql）和一对通用标记库验证器（ScriptFreeTLV 和 PermittedTaglibsTLV）组成。优点有：

1、 在应用程序服务器之间提供了一致的接口，最大程序地提高了WEB应用在各应用服务器之间的移植。

2、 简化了JSP和WEB应用程序的开发。

3、 以一种统一的方式减少了JSP中的scriptlet代码数量，可以达到没有任何scriptlet代码的程序。在我们公司的项目中是不允许有任何的scriptlet代码出现在JSP中。

4、 允许JSP设计工具与WEB应用程序开发的进一步集成。相信不久就会有支持JSTL的IDE开发工具出现。

4、j2ee的优越性主要表现在哪些方面？MVC模式

a、 J2EE基于JAVA 技术，与平台无关

b、 J2EE拥有开放标准，许多大型公司实现了对该规范支持的应用服务器。如BEA ,IBM,Oracle等。

c、 J2EE提供相当专业的通用软件服务。

d、 J2EE提供了一个优秀的企业级应用程序框架，对快速高质量的开发系统打下了基础。

Model模型：应用程序的主体部分，用于表示业务逻辑。

View视图：应用程序中用户界面相关的部分，是用户看到并与之交互的界面。

Controller控制器：用于根据用户的输入，控制用户界面数据显示，更新Model对象状态。

MVC模式的出现不仅实现了功能模块和显示模块的分离，同时还提够了应用系统的可维护、可扩展性、可移植性、和组建的可复用性。

5、Struts的优点

a、实现MVC模式，结构清晰，使开发者只需关注业务逻辑的实现。

b、有丰富的tag可以用，能大大提够开发效率，缩短开发时间。

c、页面导航。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很大的好处

d、提供Exception处理机制

e、支持L18N

6、为什么要用struts？

　　JSP、Servlet、JavaBean技术的出现给我们构建强大的企业应用系统提供了可能。但用这些技术构建的系统非常的繁乱，所以在此之上，我们需要一个规则、一个把这些技术组织起来的规则，这就是框架，Struts便应运而生。

　　基于Struts开发的应用由3类组件构成：控制器组件、模型组件、视图组件

7、Sturt1的核心类、核心标签库？

ActionServlet 控制器、ActionMapping状态改变事件 、 Action控制器的一部分、ActionForward用户指向、ActionForm状态改变的数据

Html标签、bean标签、logic标签、tiles标签、nested标签

8、struts1与sturts2的区别（struts2是struts1和webwork的结合体）

1、struts1要求Action类继承一个抽象基类，而不是接口。

struts2的action类可以实现一个action接口，也可以实现其他接口。

2、sturts1 action是单例模式，线程是安全的。

struts2 action线程是不安全的，action为每一个请求都生成了一个实例。

3、sturts1过去依赖serlet API，不容易测试。

struts2不依赖于容器，允许Action脱离容器单独被测试。

4、Struts1 使用ActionForm对象捕获输入。所有的ActionForm必须继承一个基类。

Struts 2直接使用Action属性作为输入属性，消除了对第二个输入对象的需求。

5、Struts1 整合了JSTL，因此使用JSTL EL。这种EL有基本对象图遍历，但是对集合和索引属性的支持很弱。

Struts2可以使用JSTL，但是也支持一个更强大和灵活的表达式语言－－"Object Graph Notation Language" (OGNL).

6、Struts 1使用标准JSP机制把对象绑定到页面中来访问。

Struts 2 使用 "ValueStack"技术，使taglib能够访问值而不需要把你的页面（view）和对象绑定起来。

7、Struts 1 ActionForm 属性通常都是String类型。Struts1使用Commons-Beanutils进行类型转换。

Struts2 使用OGNL进行类型转换。提供基本和常用对象的转换器。

8、Struts 1支持在ActionForm的validate方法中手动校验，或者通过Commons Validator的扩展来校验。

Struts2支持通过validate方法和XWork校验框架来进行校验。

9、Struts1支持每一个模块有单独的Request Processors（生命周期），但是模块中的所有Action必须共享相同的生命周期。

Struts2支持通过拦截器堆栈（Interceptor Stacks）为每一个Action创建不同的生命周期。堆栈能够根据需要和不同的Action一起使用。

9、过滤器和拦截器的区别

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、过滤器依赖于servlet容器，而拦截器不依赖于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

5、在action的生命周期中，拦截器可以多次被调用，而过滤器只在容器初始化时调用一次

拦截器 ：是在面向切面编程的就是在你的service或者一个方法，前调用一个方法，或者在方法后调用一个方法比如动态代理就是拦截器的简单实现，在你调用方法前打印出字符串（或者做其它业务逻辑的操作），也可以在你调用方法后打印出字符串，甚至在你抛出异常的时候做业务逻辑的操作。

过滤器：是在Java web中，你传入的request,response提前过滤掉一些信息，或者提前设置一些参数，然后再传入servlet或者struts的 action进行业务逻辑，比如过滤掉非法url（不是login.do的地址请求，如果用户没有登陆都过滤掉）,或者在传入servlet或者 struts的action前统一设置字符集，或者去除掉一些非法字符.

10、hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得java程序员可以随心所欲的使用对象编程思维来操纵数据库。

工作原理：

1.读取并解析配置文件2.读取并解析映射信息，创建SessionFactory 3.打开Sesssion 4.创建事务Transation 5.持久化操作6.提交事务7.关闭Session 8.关闭SesstionFactory

优点有：

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的主流持久化框架，是一个优秀的ORM实现。他很大程度的简化DAO层的编码工作

3、 Hibernate使用Java反射机制而不是字节码增强程序来实现透明性。

4、 Hibernate的性能好，映射的灵活性比较出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

11、hibernate的核心类是什么？？重要方法是什么？？

Configuration、SessionFactory

Session如下方法 Save、 load、 Update、Delete

Query q=CreateQuery(“from Customer where customerName=:customerName”)

beginTransaction、close、Transaction、Commit()

12、session.load()和session.get()的区别

Session.load/get方法均可以根据指定的实体类和id从数据库读取记录，并返回与之对应的实体对象。其区别在于：

如果未能发现符合条件的记录，get方法返回null，而load方法会抛出一个ObjectNotFoundException。

13、hql和sql的区别【可以这样说，hibernate是面向对象语言与关系型数据库之间的桥梁，他使得程序员可以不用关心底层数据库连接的代码，而可以专心写业务逻辑。】

sql是面向数据库表查询

hql是面向对象查询的,其form子句返回的是对象的实例。

14、hibernate与jdbc之间的区别【可以这样说，hibernate是面向对象语言与关系型数据库之间的桥梁，他使得程序员可以不用关心底层数据库连接的代码，而可以专心写业务逻辑。】

Hibernate作为一个O/R Mapping,比JDBC具备的优势有：

1.编程思想上，更加符合人的逻辑思维习惯，面向对象比面向过程更加容易理解，测试和维护

2.开发维护速度上，Hibernate显著的快，代码量显著小

3.通过Annotation进行数据库的字段加密

4.对Sql不熟的菜鸟来说可以自动调优

5.结合spring，通过声明式事务可以省略事务的控制，事务以横切面形式出现

Jdbc比Hibernate具备的优势有：

1.大数据量访问时，Jdbc的效率显著快

2.直接操作数据库比较灵活

15、Hibernate是如何延迟加载？

当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。

16、说下Hibernate的缓存机制

　　1. 内部缓存存在Hibernate中又叫一级缓存，属于应用事物级缓存

　　2. 二级缓存：

　　a) 应用及缓存

　　b) 分布式缓存

c) 第三方缓存的实现

17、spring工作机制及为什么要用?【spring是一个轻量的控制反转和面向切面的容器框架】

　　1.springmvc把所有的请求都提交给DispatcherServlet,它会委托应用系统的其他模块负责对请求进行真正的处理工作。

　　2.DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller.

　　3.DispatcherServlet把请求提交到目标Controller

　　4.Controller进行业务逻辑处理后，会返回一个ModelAndView

　　5.Dispathcher查询一个或多个ViewResolver视图解析器,找到ModelAndView对象指定的视图对象

　　6.视图对象负责渲染返回给客户端。

IoC就是由容器来控制业务对象之间的依赖关系。控制反转的本质，是控制权由应用代码转到了外部容器，控制器的转移既是所谓的反转。控制权的转移带来的好处就是降低了业务对象之间的依赖程度，即实现了解耦。

DI/IOC,对持久层和表示层的控制与分配，增加系统的灵活性和稳定性. AOP,面向切面,利用代理对程序的有效管理.

spring是一个轻量级的IOC和AOP框架，通过spring的IOC实现松耦合，而作为一个AOP框架他又能分离系统服务，实现内聚开发 Spring 最好的地方是它有助于您替换对象。有了 Spring，只要用 JavaBean 属性和配置文件加入依赖性（协作对象）。然后可以很容易地在需要时替换具有类似接口的协作对象。}

Spring对多种ORM框架提供了很好的支持

线程的相关题

http://blog.csdn.net/liuxiaogangqq/article/details/8124866

阿里云>

云栖社区>

正文

经典java面试题

一、你对MVC的理解，MVC有什么优缺点？结合Struts，说明在一个Web应用如何去使用？

答：

MVC设计模式（应用观察者模式的框架模式）

M: Model(Business process layer)，模型，操作数据的业务处理层,并独立于表现层(Independent of presentation)。

V: View(Presentation layer)，视图，通过客户端数据类型显示数据,并回显模型层的执行结果。

C: Controller(Control layer)，控制器，也就是视图层和模型层桥梁，控制数据的流向，接受视图层发出的事件，并重绘视图

MVC框架的一种实现模型

模型二(Servlet-centric)：

JSP+Servlet+JavaBean，以控制为核心，JSP只负责显示和收集数据，Sevlet，连接视图和模型，将视图层数据，发送给模型层，JavaBean，分为业务类和数据实体，业务类处理业务数据，数据实体，承载数据，基本上大多数的项目都是使用这种MVC的实现模式。

StrutsMVC框架(Web application frameworks)

Struts是使用MVC的实现模式二来实现的，也就是以控制器为核心。

Struts提供了一些组件使用MVC开发应用程序：

Model：Struts没有提供model类。这个商业逻辑必须由Web应用程序的开发者以JavaBean或EJB的形式提供

View：Struts提供了action form创建form bean, 用于在controller和view间传输数据。此外，Struts提供了自定义JSP标签库，辅助开发者用JSP创建交互式的以表单为基础的应用程序，应用程序资源文件保留了一些文本常量和错误消息，可转变为其它语言，可用于JSP中。

Controller：Struts提供了一个核心的控制器ActionServlet，通过这个核心的控制器来调用其他用户注册了的自定义的控制器Action，自定义Action需要符合Struts的自定义Action规范，还需要在struts-config.xml的特定配置文件中进行配置，接收JSP输入字段形成Action form，然后调用一个Action控制器。Action控制器中提供了model的逻辑接口。

二、什么是WebService？

答：

WebService是一个SOA（面向服务的编程）的架构，它是不依赖于语言，不依赖于平台，可以实现不同的语言间的相互调用，通过Internet进行基于Http协议的网络应用间的交互。

WebService实现不同语言间的调用，是依托于一个标准，webservice是需要遵守WSDL（web服务定义语言）/SOAP（简单请求协议）规范的。

WebService=WSDL+SOAP+UDDI（webservice的注册）

Soap是由Soap的part和0个或多个附件组成，一般只有part，在part中有Envelope和Body。

Web Service是通过提供标准的协议和接口，可以让不同的程序集成的一种SOA架构。

Web Service的优点

(1) 可以让异构的程序相互访问（跨平台）

(2) 松耦合

(3) 基于标准协议（通用语言，允许其他程序访问）

Web Service的基本原理

(1) Service Provider采用WSDL描述服务

(2) Service Provider 采用UDDI将服务的描述文件发布到UDDI服务器（Register server）

(3) Service Requestor在UDDI服务器上查询并 获取WSDL文件

(4) Service requestor将请求绑定到SOAP，并访问相应的服务。

三、什么是中间件？

中间件就是程序中可织入的，可重用的，与业务逻辑无关的各种组件。

中间件（middleware）是基础软件的一大类，属于可复用软件的范畴。顾名思义，中间件处于操作系统软件与用户的应用软件的中间。中间件在操作系统、网络和数据库之上，应用软件的下层，总的作用是为处于自己上层的应用软件提供运行与开发的环境，帮助用户灵活、高效地开发和集成复杂的应用软件。

　　在众多关于中间件的定义中，比较普遍被接受的是IDC表述的：中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通信。

分类：数据访问中间件，远程调用中间件，消息中间件，交易中间件，对象中间件。

举例：

1，RMI（Remote Method Invocations, 远程调用）

2，Load Balancing(负载均衡，将访问负荷分散到各个服务器中)

3，Transparent Fail-over(透明的故障切换)

4，Clustering(集群,用多个小的服务器代替大型机）

5，Back-end-Integration(后端集成，用现有的、新开发的系统如何去集成遗留的系统)

6，Transaction事务（全局/局部）全局事务（分布式事务）局部事务（在同一数据库联接内的事务）

7，Dynamic Redeployment(动态重新部署,在不停止原系统的情况下，部署新的系统）

8，System Management(系统管理)

9，Threading(多线程处理)

10，Message-oriented Middleware面向消息的中间件（异步的调用编程）

11，Component Life Cycle(组件的生命周期管理)

12，Resource pooling（资源池）

13，Security（安全）

14，Caching（缓存）

四、什么是典型的软件三层结构？软件设计为什么要分层？软件分层有什么好处？

答：(1) Presentation layer（表示层）

(1) 表示逻辑（生成界面代码）

(2) 接收请求

(3) 处理业务层抛出的异常

(4) 负责规则验证（数据格式，数据非空等）

(5) 流程控制

(2) Service layer（服务层/业务层）

(1) 封装业务逻辑处理，并且对外暴露接口

(2) 负责事务，安全等服务

(3) Persistence layer（持久层）

(1) 封装数据访问的逻辑，暴露接口

(2) 提供方便的数据访问的方案（查询语言，API，映射机制等）

Domain layer（域层）

(1) 业务对象以及业务关系的表示

(2) 处理简单的业务逻辑

(3) 域层的对象可以穿越表示层，业务层，持久层

软件分层结构使得代码维护非常方便，设计明确，各层独立，专注自己擅长的领域。

五、什么是OOP？OOP相对于面向过程编程有哪些优点？

OOP，Object-Oriented Programming，面向对象编程不同于面向过程编程：

（1）OOP关注对象和角色，也就是事物的本质

1）OOP把客观世界中的对象抽象成对应的类；

2）通过类构造实例；

3）通过依赖、继承、实现等形式建立对象间的通信关系

（2）OOP易于扩展，增加或改变业务的功能，无需大幅改动改变源代码

（3）OOP易于建模，OOP就是软件架构师在计算机高级语言中对客观世界的抽象和再现，人们可以很好地理解和建立起计算机中的抽象模型

六、谈谈Overload和Override的区别。

答：

方法的重写Overriding和重载Overloading是Java多态性的不同表现。重写Overriding是父类与子类之间多态性的一种表现，重载Overloading是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写(Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded的方法是可以改变返回值的类型。

七、谈谈HashMap和Hashtable的区别。

答： HashMap是轻量级的（线程不安全的，效率高的）集合，Hashtable是重量级的（线程安全的，效率低的）集合。

都属于Map接口的类，实现了将惟一键映射到特定的值上。

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

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

八、请问abstract class和interface有什么区别？

答：

1) 接口没有任何的实现,而抽象类它可以有部分的实现也可以没有;

2) 如果需要复用或者共享部分代码,应该用抽象类而不是接口;

3) 继承无法解决类型的转换问题,接口就是为了解决这一问题而产生的(Java的单继承是接口产生的根本原因) 接口是一种抽象的第三方规范,跟对象没什么直接关系。

九、请问软件开发中的设计模式你会使用哪些？

答：我熟悉的设计模式有单例模式，DAO模式，模板方法，工厂模式，委托代理模式，MVC模式等。

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。

Singleton模式一般形式:

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，在类初始化时实例话，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。

十、类的核心特性有哪些？

类具有封装性、继承性和多态性。

封装性：

类的封装性为类的成员提供公有、缺省、保护和私有等多级访问权限，目的是隐藏类中的私有变量和类中方法的实现细节。

继承性：

类的继承性提供从已存在的类创建新类的机制，继承（inheritance）使一个新类自动拥有被继承类（父类）的全部成员。

多态性：

类的多态性提供类中方法执行的多样性，多态性有两种表现形式：重载和覆盖。

十一、请问类与对象有什么区别？

类

类就是某一种事物的一般性的集合体，是相同或相似的各个事物共同特性的一种抽象。

对象

在面向对象概念中，对象（Object）是类的实例（instance）。对象与类的关系就像变量与数据类型的关系一样。

十二、写出生产者消费者模型的实现

十三、用软件工程和Java来叙述购物车（shopping cart）系统实现

十四、文件系统的几个类:FileInputStream、FileOutputStream、FileReader、FileWriter的区别

十五、请写出Java API中最常用的五个包，并且各举两个类加以说明。

十六、请问你在“电信运营支撑系统”中遇到了哪些问题？你是怎么解决的？做这个项目你最大的收获是什么？

十七、请说出几个常用的异常类

答：NullpointException（空指针异常）

ClassNotFoundException（类找不到异常）

ClassCastException（类型转换异常）

IllegalArgumentException（非法参数异常）

ArithmeticException（算术异常）

NumberFormatException（数据格式异常）

IndexOutOfBoundsException（数组下标越界异常）

IllegalStateException（非法语句异常）

十八、什么是进程和线程？两者的区别是什么？

十九、请简要描述一下你做的ShoppingCart项目。

二十、请问网络7层协议，tcp/ip4 层协议是什么？

二十一、Java面向对象的四大特点

二十二、public,private,protected，default访问权限（可见性）

答：

public：全局可见

protected：继承体系结构之间可见

default（或不写）：同包可见

private：本类可见。

二十三、名词解释

CRM

[Customer Relationship Management, 客户关系管理]

ERP

[Enterprise Resource Planning, 企业资源规划]

OSS

[Operation Support System, 运营支撑系统]

BSS

[Business Support System, 数据定义语言]

BOSS

[Business Operation Support System, 数据定义语言]

OA

[Office Automatization, 办公自动化]

HTTP

OOP

SOA

GUI

DDL

[Data Definition Language, 数据定义语言]

DML

[Data Manipulation Language, 数据操作语言]

WYSIWYG

所见即所得 [What You See is What You Get]

CMP

CMT

B2B

C2C

B2C

IDE

DOM

Client/Server

CMM

ORM

MIS

MVC

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

OLE

CORBA

CORBA 标准是公共对象请求代理结构(Common Object Request Broker Architecture)，由对象管理组织 (Object Management Group，缩写为 OMG)标准化。它的组成是接口定义语言(IDL), 语言绑定(binding:也译为联编)和允许应用程序间互操作的协议。 其目的为：

用不同的程序设计语言书写

在不同的进程中运行

为不同的操作系统开发。

UML

UML，标准建模语言，包含用例图,静态图(包括类图、对象图和包图),行为图,交互图(顺序图,合作图),实现图等。

XML

CMMI

JRE

J2EE

J2EE是Sun公司提出的多层(multi-diered),分布式(distributed),基于组件(component-base)的企业级应用模型(enterpriese application model).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietn tier)组件,web层和组件,Business层和组件,企业信息系统(EIS)层。

JDK

AOP

OO

Container

W3C

JMS

Domain

POJO

JVM

JNDI

JTA

SOAP

[Simple Object Access Protocol，简单对象访问协议]

WSDL

JDO

JDO是Java对象持久化的新的规范，为Java Data Object的简称,也是一个用于存取某种数据仓库中的对象的标准化API。JDO提供了透明的对象存储，因此对开发人员来说，存储数据对象完全不需要额外的代码（如JDBC API的使用）。这些繁琐的例行工作已经转移到JDO产品提供商身上，使开发人员解脱出来，从而集中时间和精力在业务逻辑上。另外，JDO很灵活，因为它可以在任何数据底层上运行。JDBC只是面向关系数据库（RDBMS)JDO更通用，提供到任何数据底层的存储功能，比如关系数据库、文件、XML以及对象数据库（ODBMS）等等，使得应用可移植性更强。

TDD

DAO

IoC

[Inversion of Control, 控制反转]

RMI

DNS

[Internet Domain Name System, 因特网域名系统]

URL

URI

二十四、数据库连接池的工作机制

二十五、互联网提供哪些服务？

二十六、请写出JSP的几个隐含内置对象

答：JSP中隐含内置对象

名称 类型 注释和范围

request javax.servlet.http.HttpServletRequest request

response javax.servlet.http.HttpServletResponse response

page javax.lang.Object page

Exception java.lang.Throwable page

pageContext javax.servlet.jsp.PageContext page

session javax.servlet.http.HttpSession session

application javax.servlet.ServletContext ServletContext

out javax.servlet.jsp.JspWriter OutputStream

config javax.servlet.ServletConfig ServletConfig

JSP共有以下9种基本内置组件（可与ASP的6种内部组件相对应）：

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

page JSP 网页本身

exception 针对错误网页，未捕捉的例外

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

二十七、请你谈谈SSH整合

答：SSH：

Struts（表示层）+Spring（业务层）+Hibernate（持久层）

Struts：

Struts是一个表示层框架，主要作用是界面展示，接收请求，分发请求。

在MVC框架中，Struts属于VC层次，负责界面表现，负责MVC关系的分发。（View：沿用JSP，HTTP，Form，Tag，Resourse ；Controller：ActionServlet，struts-config.xml，Action）

Hibernate：

Hibernate是一个持久层框架，它只负责与关系数据库的操作。

Spring：

Spring是一个业务层框架，是一个整合的框架，能够很好地黏合表示层与持久层。

二十八、应用服务器与Web Server的区别

二十九、Java Servlet API中forward() 与redirect()的区别

答：

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

三十、写一个简单的C/S结构程序，Java 的通信编程，编程题(或问答)，用JAVA SOCKET编程，读服务器几个字符，再写入本地显示？

答:Server端程序:

package test;

import java.net.\*;

import java.io.\*;

public class Server

{

private ServerSocket ss;

private Socket socket;

private BufferedReader in;

private PrintWriter out;

public Server()

{

try

{

ss=new ServerSocket(10000);

while(true)

{

socket = ss.accept();

String RemoteIP = socket.getInetAddress().getHostAddress();

String RemotePort = ":"+socket.getLocalPort();

System.out.println("A client come in!IP:"+RemoteIP+RemotePort);

in = new BufferedReader(new

InputStreamReader(socket.getInputStream()));

String line = in.readLine();

System.out.println("Cleint send is :" + line);

out = new PrintWriter(socket.getOutputStream(),true);

out.println("Your Message Received!");

out.close();

in.close();

socket.close();

}

}catch (IOException e)

{

out.println("wrong");

}

}

public static void main(String[] args)

{

new Server();

}

};

Client端程序:

package test;

import java.io.\*;

import java.net.\*;

public class Client

{

Socket socket;

BufferedReader in;

PrintWriter out;

public Client()

{

try

{

System.out.println("Try to Connect to 127.0.0.1:10000");

socket = new Socket("127.0.0.1",10000);

System.out.println("The Server Connected!");

System.out.println("Please enter some Character:");

BufferedReader line = new BufferedReader(new

InputStreamReader(System.in));

out = new PrintWriter(socket.getOutputStream(),true);

out.println(line.readLine());

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

System.out.println(in.readLine());

out.close();

in.close();

socket.close();

}catch(IOException e)

{

out.println("Wrong");

}

}

public static void main(String[] args)

{

new Client();

}

};

阿里云>

云栖社区>

正文

50 道Java 线程面试题(经典）\_java

下面是 Java 线程相关的热门面试题，你可以用它来好好准备面试。

1) 什么是线程？

线程是操作系统能够进行运算调度的最小单位，它被包含在进程之中，是进程中的实际运作单位。程序员可以通过它进行多处理器编程，你可以使用多线程对运算密集型任务提速。比如，如果一个线程完成一个任务要 100 毫秒，那么用十个线程完成改任务只需 10 毫秒。Java 在语言层面对多线程提供了卓越的支持，它也是一个很好的卖点。欲了解更多详细信息请点击这里。

2) 线程和进程有什么区别？

线程是进程的子集，一个进程可以有很多线程，每条线程并行执行不同的任务。不同的进程使用不同的内存空间，而所有的线程共享一片相同的内存空间。别把它和栈内存搞混，每个线程都拥有单独的栈内存用来存储本地数据。更多详细信息请点击这里。

3) 如何在 Java 中实现线程？

在语言层面有两种方式。java.lang.Thread 类的实例就是一个线程但是它需要调用 java.lang.Runnable 接口来执行，由于线程类本身就是调用的 Runnable 接口所以你可以继承 java.lang.Thread 类或者直接调用 Runnable 接口来重写 run ()方法实现线程。更多详细信息请点击这里.

4) 用 Runnable 还是 Thread？

这个问题是上题的后续，大家都知道我们可以通过继承 Thread 类或者调用 Runnable 接口来实现线程，问题是，那个方法更好呢？什么情况下使用它？这个问题很容易回答，如果你知道 Java 不支持类的多重继承，但允许你调用多个接口。所以如果你要继承其他类，当然是调用 Runnable 接口好了。更多详细信息请点击这里。

6) Thread 类中的 start () 和 run () 方法有什么区别？

这个问题经常被问到，但还是能从此区分出面试者对 Java 线程模型的理解程度。start ()方法被用来启动新创建的线程，而且 start ()内部调用了 run ()方法，这和直接调用 run ()方法的效果不一样。当你调用 run ()方法的时候，只会是在原来的线程中调用，没有新的线程启动，start ()方法才会启动新线程。更多讨论请点击这里

7) Java 中 Runnable 和 Callable 有什么不同？

Runnable 和 Callable 都代表那些要在不同的线程中执行的任务。Runnable 从 JDK1.0 开始就有了，Callable 是在 JDK1.5 增加的。它们的主要区别是 Callable 的 call () 方法可以返回值和抛出异常，而 Runnable 的 run ()方法没有这些功能。Callable 可以返回装载有计算结果的 Future 对象。我的博客有更详细的说明。

8) Java 中 CyclicBarrier 和 CountDownLatch 有什么不同？

CyclicBarrier 和 CountDownLatch 都可以用来让一组线程等待其它线程。与 CyclicBarrier 不同的是，CountdownLatch 不能重新使用。点此查看更多信息和示例代码。

9) Java 内存模型是什么？

Java 内存模型规定和指引 Java 程序在不同的内存架构、CPU 和操作系统间有确定性地行为。它在多线程的情况下尤其重要。Java 内存模型对一个线程所做的变动能被其它线程可见提供了保证，它们之间是先行发生关系。这个关系定义了一些规则让程序员在并发编程时思路更清晰。比如，先行发生关系确保了：

线程内的代码能够按先后顺序执行，这被称为程序次序规则。

对于同一个锁，一个解锁操作一定要发生在时间上后发生的另一个锁定操作之前，也叫做管程锁定规则。

前一个对 Volatile的写操作在后一个volatile的读操作之前，也叫volatile变量规则。

一个线程内的任何操作必需在这个线程的 start ()调用之后，也叫作线程启动规则。

一个线程的所有操作都会在线程终止之前，线程终止规则。

一个对象的终结操作必需在这个对象构造完成之后，也叫对象终结规则。

可传递性

我强烈建议大家阅读《Java 并发编程实践》第十六章来加深对 Java 内存模型的理解。

10) Java 中的 volatile 变量是什么？

volatile 是一个特殊的修饰符，只有成员变量才能使用它。在 Java 并发程序缺少同步类的情况下，多线程对成员变量的操作对其它线程是透明的。volatile 变量可以保证下一个读取操作会在前一个写操作之后发生，就是上一题的 volatile 变量规则。点击这里查看更多 volatile 的相关内容。

11) 什么是线程安全？Vector 是一个线程安全类吗？ （详见这里)

如果你的代码所在的进程中有多个线程在同时运行，而这些线程可能会同时运行这段代码。如果每次运行结果和单线程运行的结果是一样的，而且其他的变量的值也和预期的是一样的，就是线程安全的。一个线程安全的计数器类的同一个实例对象在被多个线程使用的情况下也不会出现计算失误。很显然你可以将集合类分成两组，线程安全和非线程安全的。Vector 是用同步方法来实现线程安全的， 而和它相似的 ArrayList 不是线程安全的。

12) Java 中什么是竞态条件？ 举个例子说明。

竞态条件会导致程序在并发情况下出现一些 bugs。多线程对一些资源的竞争的时候就会产生竞态条件，如果首先要执行的程序竞争失败排到后面执行了，那么整个程序就会出现一些不确定的 bugs。这种 bugs 很难发现而且会重复出现，因为线程间的随机竞争。一个例子就是无序处理，详见答案。

13) Java 中如何停止一个线程？

Java 提供了很丰富的 API 但没有为停止线程提供 API。JDK 1.0 本来有一些像 stop ()， suspend () 和 resume ()的控制方法但是由于潜在的死锁威胁因此在后续的 JDK 版本中他们被弃用了，之后 Java API 的设计者就没有提供一个兼容且线程安全的方法来停止一个线程。当 run () 或者 call () 方法执行完的时候线程会自动结束，如果要手动结束一个线程，你可以用 volatile 布尔变量来退出 run ()方法的循环或者是取消任务来中断线程。点击这里查看示例代码。

14) 一个线程运行时发生异常会怎样？

这是我在一次面试中遇到的一个很刁钻的 Java 面试题, 简单的说，如果异常没有被捕获该线程将会停止执行。

Thread.UncaughtExceptionHandler 是用于处理未捕获异常造成线程突然中断情况的一个内嵌接口。当一个未捕获异常将造成线程中断的时候 JVM 会使用 Thread.getUncaughtExceptionHandler ()来查询线程的 UncaughtExceptionHandler 并将线程和异常作为参数传递给 handler 的 uncaughtException ()方法进行处理。

15） 如何在两个线程间共享数据？

你可以通过共享对象来实现这个目的，或者是使用像阻塞队列这样并发的数据结构。这篇教程《Java 线程间通信》(涉及到在两个线程间共享对象)用 wait 和 notify 方法实现了生产者消费者模型。

16) Java 中 notify 和 notifyAll 有什么区别？

这又是一个刁钻的问题，因为多线程可以等待单监控锁，Java API 的设计人员提供了一些方法当等待条件改变的时候通知它们，但是这些方法没有完全实现。notify ()方法不能唤醒某个具体的线程，所以只有一个线程在等待的时候它才有用武之地。而 notifyAll ()唤醒所有线程并允许他们争夺锁确保了至少有一个线程能继续运行。我的博客有更详细的资料和示例代码。

17) 为什么 wait, notify 和 notifyAll 这些方法不在 thread 类里面？

这是个设计相关的问题，它考察的是面试者对现有系统和一些普遍存在但看起来不合理的事物的看法。回答这些问题的时候，你要说明为什么把这些方法放在 Object 类里是有意义的，还有不把它放在 Thread 类里的原因。一个很明显的原因是 JAVA 提供的锁是对象级的而不是线程级的，每个对象都有锁，通过线程获得。如果线程需要等待某些锁那么调用对象中的 wait ()方法就有意义了。如果 wait ()方法定义在 Thread 类中，线程正在等待的是哪个锁就不明显了。简单的说，由于 wait，notify 和 notifyAll 都是锁级别的操作，所以把他们定义在 Object 类中因为锁属于对象。你也可以查看这篇文章了解更多。

18) 什么是 ThreadLocal 变量？

ThreadLocal 是 Java 里一种特殊的变量。每个线程都有一个 ThreadLocal 就是每个线程都拥有了自己独立的一个变量，竞争条件被彻底消除了。它是为创建代价高昂的对象获取线程安全的好方法，比如你可以用 ThreadLocal 让 SimpleDateFormat 变成线程安全的，因为那个类创建代价高昂且每次调用都需要创建不同的实例所以不值得在局部范围使用它，如果为每个线程提供一个自己独有的变量拷贝，将大大提高效率。首先，通过复用减少了代价高昂的对象的创建个数。其次，你在没有使用高代价的同步或者不变性的情况下获得了线程安全。线程局部变量的另一个不错的例子是 ThreadLocalRandom 类，它在多线程环境中减少了创建代价高昂的 Random 对象的个数。查看答案了解更多。

19) 什么是 FutureTask？

在 Java 并发程序中 FutureTask 表示一个可以取消的异步运算。它有启动和取消运算、查询运算是否完成和取回运算结果等方法。只有当运算完成的时候结果才能取回，如果运算尚未完成 get 方法将会阻塞。一个 FutureTask 对象可以对调用了 Callable 和 Runnable 的对象进行包装，由于 FutureTask 也是调用了 Runnable 接口所以它可以提交给 Executor 来执行。

20) Java 中 interrupted 和 isInterruptedd 方法的区别？

interrupted () 和 isInterrupted ()的主要区别是前者会将中断状态清除而后者不会。Java 多线程的中断机制是用内部标识来实现的，调用Thread.interrupt ()来中断一个线程就会设置中断标识为 true。当中断线程调用静态方法Thread.interrupted ()来检查中断状态时，中断状态会被清零。而非静态方法 isInterrupted ()用来查询其它线程的中断状态且不会改变中断状态标识。简单的说就是任何抛出 InterruptedException 异常的方法都会将中断状态清零。无论如何，一个线程的中断状态有有可能被其它线程调用中断来改变。

21) 为什么 wait 和 notify 方法要在同步块中调用？

主要是因为 Java API 强制要求这样做，如果你不这么做，你的代码会抛出 IllegalMonitorStateException 异常。还有一个原因是为了避免 wait 和 notify 之间产生竞态条件。

22) 为什么你应该在循环中检查等待条件?

处于等待状态的线程可能会收到错误警报和伪唤醒，如果不在循环中检查等待条件，程序就会在没有满足结束条件的情况下退出。因此，当一个等待线程醒来时，不能认为它原来的等待状态仍然是有效的，在 notify ()方法调用之后和等待线程醒来之前这段时间它可能会改变。这就是在循环中使用 wait ()方法效果更好的原因，你可以在 Eclipse 中创建模板调用 wait 和 notify 试一试。如果你想了解更多关于这个问题的内容，我推荐你阅读《Effective Java》这本书中的线程和同步章节。

23) Java 中的同步集合与并发集合有什么区别？

同步集合与并发集合都为多线程和并发提供了合适的线程安全的集合，不过并发集合的可扩展性更高。在 Java1.5 之前程序员们只有同步集合来用且在多线程并发的时候会导致争用，阻碍了系统的扩展性。Java5 介绍了并发集合像 ConcurrentHashMap，不仅提供线程安全还用锁分离和内部分区等现代技术提高了可扩展性。更多内容详见答案。

24） Java 中堆和栈有什么不同？

为什么把这个问题归类在多线程和并发面试题里？因为栈是一块和线程紧密相关的内存区域。每个线程都有自己的栈内存，用于存储本地变量，方法参数和栈调用，一个线程中存储的变量对其它线程是不可见的。而堆是所有线程共享的一片公用内存区域。对象都在堆里创建，为了提升效率线程会从堆中弄一个缓存到自己的栈，如果多个线程使用该变量就可能引发问题，这时 volatile 变量就可以发挥作用了，它要求线程从主存中读取变量的值。

更多内容详见答案。

25） 什么是线程池？ 为什么要使用它？

创建线程要花费昂贵的资源和时间，如果任务来了才创建线程那么响应时间会变长，而且一个进程能创建的线程数有限。为了避免这些问题，在程序启动的时候就创建若干线程来响应处理，它们被称为线程池，里面的线程叫工作线程。从 JDK1.5 开始，Java API 提供了 Executor 框架让你可以创建不同的线程池。比如单线程池，每次处理一个任务；数目固定的线程池或者是缓存线程池（一个适合很多生存期短的任务的程序的可扩展线程池）。更多内容详见这篇文章。

26） 如何写代码来解决生产者消费者问题？

在现实中你解决的许多线程问题都属于生产者消费者模型，就是一个线程生产任务供其它线程进行消费，你必须知道怎么进行线程间通信来解决这个问题。比较低级的办法是用 wait 和 notify 来解决这个问题，比较赞的办法是用 Semaphore 或者 BlockingQueue 来实现生产者消费者模型，这篇教程有实现它。

27） 如何避免死锁？

Java 多线程中的死锁

死锁是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。这是一个严重的问题，因为死锁会让你的程序挂起无法完成任务，死锁的发生必须满足以下四个条件：

互斥条件：一个资源每次只能被一个进程使用。

请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

不剥夺条件：进程已获得的资源，在末使用完之前，不能强行剥夺。

循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。

避免死锁最简单的方法就是阻止循环等待条件，将系统中所有的资源设置标志位、排序，规定所有的进程申请资源必须以一定的顺序（升序或降序）做操作来避免死锁。这篇教程有代码示例和避免死锁的讨论细节。

28) Java 中活锁和死锁有什么区别？

这是上题的扩展，活锁和死锁类似，不同之处在于处于活锁的线程或进程的状态是不断改变的，活锁可以认为是一种特殊的饥饿。一个现实的活锁例子是两个人在狭小的走廊碰到，两个人都试着避让对方好让彼此通过，但是因为避让的方向都一样导致最后谁都不能通过走廊。简单的说就是，活锁和死锁的主要区别是前者进程的状态可以改变但是却不能继续执行。

29） 怎么检测一个线程是否拥有锁？

我一直不知道我们竟然可以检测一个线程是否拥有锁，直到我参加了一次电话面试。在 java.lang.Thread 中有一个方法叫 holdsLock ()，它返回 true 如果当且仅当当前线程拥有某个具体对象的锁。你可以查看这篇文章了解更多。

30) 你如何在 Java 中获取线程堆栈？

对于不同的操作系统，有多种方法来获得 Java 进程的线程堆栈。当你获取线程堆栈时，JVM 会把所有线程的状态存到日志文件或者输出到控制台。在 Windows 你可以使用 Ctrl + Break 组合键来获取线程堆栈，Linux 下用 kill -3 命令。你也可以用 jstack 这个工具来获取，它对线程 id 进行操作，你可以用 jps 这个工具找到 id。

31) JVM 中哪个参数是用来控制线程的栈堆栈小的

这个问题很简单， -Xss 参数用来控制线程的堆栈大小。你可以查看 JVM 配置列表来了解这个参数的更多信息。

32） Java 中 synchronized 和 ReentrantLock 有什么不同？

Java 在过去很长一段时间只能通过 synchronized 关键字来实现互斥，它有一些缺点。比如你不能扩展锁之外的方法或者块边界，尝试获取锁时不能中途取消等。Java 5 通过 Lock 接口提供了更复杂的控制来解决这些问题。 ReentrantLock 类实现了 Lock，它拥有与 synchronized 相同的并发性和内存语义且它还具有可扩展性。你可以查看这篇文章了解更多

33） 有三个线程 T1，T2，T3，怎么确保它们按顺序执行？

在多线程中有多种方法让线程按特定顺序执行，你可以用线程类的 join ()方法在一个线程中启动另一个线程，另外一个线程完成该线程继续执行。为了确保三个线程的顺序你应该先启动最后一个(T3 调用 T2，T2 调用 T1)，这样 T1 就会先完成而 T3 最后完成。你可以查看这篇文章了解更多。

34) Thread 类中的 yield 方法有什么作用？

Yield 方法可以暂停当前正在执行的线程对象，让其它有相同优先级的线程执行。它是一个静态方法而且只保证当前线程放弃 CPU 占用而不能保证使其它线程一定能占用 CPU，执行 yield ()的线程有可能在进入到暂停状态后马上又被执行。点击这里查看更多 yield 方法的相关内容。

35） Java 中 ConcurrentHashMap 的并发度是什么？

ConcurrentHashMap 把实际 map 划分成若干部分来实现它的可扩展性和线程安全。这种划分是使用并发度获得的，它是 ConcurrentHashMap 类构造函数的一个可选参数，默认值为 16，这样在多线程情况下就能避免争用。欲了解更多并发度和内部大小调整请阅读我的文章 How ConcurrentHashMap works in Java。

36） Java 中 Semaphore 是什么？

Java 中的 Semaphore 是一种新的同步类，它是一个计数信号。从概念上讲，从概念上讲，信号量维护了一个许可集合。如有必要，在许可可用前会阻塞每一个 acquire ()，然后再获取该许可。每个 release ()添加一个许可，从而可能释放一个正在阻塞的获取者。但是，不使用实际的许可对象，Semaphore 只对可用许可的号码进行计数，并采取相应的行动。信号量常常用于多线程的代码中，比如数据库连接池。更多详细信息请点击这里。

37）如果你提交任务时，线程池队列已满。会时发会生什么？

这个问题问得很狡猾，许多程序员会认为该任务会阻塞直到线程池队列有空位。事实上如果一个任务不能被调度执行那么 ThreadPoolExecutor's submit ()方法将会抛出一个 RejectedExecutionException 异常。

38) Java 线程池中 submit () 和 execute ()方法有什么区别？

两个方法都可以向线程池提交任务，execute ()方法的返回类型是 void，它定义在 Executor 接口中， 而 submit ()方法可以返回持有计算结果的 Future 对象，它定义在 ExecutorService 接口中，它扩展了 Executor 接口，其它线程池类像 ThreadPoolExecutor 和 ScheduledThreadPoolExecutor 都有这些方法。更多详细信息请点击这里。

39) 什么是阻塞式方法？

阻塞式方法是指程序会一直等待该方法完成期间不做其他事情，ServerSocket 的 accept ()方法就是一直等待客户端连接。这里的阻塞是指调用结果返回之前，当前线程会被挂起，直到得到结果之后才会返回。此外，还有异步和非阻塞式方法在任务完成前就返回。更多详细信息请点击这里。

40) Swing 是线程安全的吗？ 为什么？

你可以很肯定的给出回答，Swing 不是线程安全的，但是你应该解释这么回答的原因即便面试官没有问你为什么。当我们说 swing 不是线程安全的常常提到它的组件，这些组件不能在多线程中进行修改，所有对 GUI 组件的更新都要在 AWT 线程中完成，而 Swing 提供了同步和异步两种回调方法来进行更新。点击这里查看更多 swing 和线程安全的相关内容。

41） Java 中 invokeAndWait 和 invokeLater 有什么区别？

这两个方法是 Swing API 提供给 Java 开发者用来从当前线程而不是事件派发线程更新 GUI 组件用的。InvokeAndWait ()同步更新 GUI 组件，比如一个进度条，一旦进度更新了，进度条也要做出相应改变。如果进度被多个线程跟踪，那么就调用 invokeAndWait ()方法请求事件派发线程对组件进行相应更新。而 invokeLater ()方法是异步调用更新组件的。更多详细信息请点击这里。

42) Swing API 中那些方法是线程安全的？

这个问题又提到了 swing 和线程安全，虽然组件不是线程安全的但是有一些方法是可以被多线程安全调用的，比如 repaint ()， revalidate ()。 JTextComponent 的 setText ()方法和 JTextArea 的 insert () 和 append () 方法也是线程安全的。

43) 如何在 Java 中创建 Immutable 对象？

这个问题看起来和多线程没什么关系， 但不变性有助于简化已经很复杂的并发程序。Immutable 对象可以在没有同步的情况下共享，降低了对该对象进行并发访问时的同步化开销。可是 Java 没有@Immutable 这个注解符，要创建不可变类，要实现下面几个步骤：通过构造方法初始化所有成员、对变量不要提供 setter 方法、将所有的成员声明为私有的，这样就不允许直接访问这些成员、在 getter 方法中，不要直接返回对象本身，而是克隆对象，并返回对象的拷贝。我的文章 how to make an object Immutable in Java 有详细的教程，看完你可以充满自信。

44） Java 中的 ReadWriteLock 是什么？

一般而言，读写锁是用来提升并发程序性能的锁分离技术的成果。Java 中的 ReadWriteLock 是 Java 5 中新增的一个接口，一个 ReadWriteLock 维护一对关联的锁，一个用于只读操作一个用于写。在没有写线程的情况下一个读锁可能会同时被多个读线程持有。写锁是独占的，你可以使用 JDK 中的 ReentrantReadWriteLock 来实现这个规则，它最多支持 65535 个写锁和 65535 个读锁。

45) 多线程中的忙循环是什么?

忙循环就是程序员用循环让一个线程等待，不像传统方法 wait ()， sleep () 或 yield () 它们都放弃了 CPU 控制，而忙循环不会放弃 CPU，它就是在运行一个空循环。这么做的目的是为了保留 CPU 缓存，在多核系统中，一个等待线程醒来的时候可能会在另一个内核运行，这样会重建缓存。为了避免重建缓存和减少等待重建的时间就可以使用它了。你可以查看这篇文章获得更多信息。

46）volatile 变量和 atomic 变量有什么不同？

这是个有趣的问题。首先，volatile 变量和 atomic 变量看起来很像，但功能却不一样。Volatile 变量可以确保先行关系，即写操作会发生在后续的读操作之前， 但它并不能保证原子性。例如用 volatile 修饰 count 变量那么 count++ 操作就不是原子性的。而 AtomicInteger 类提供的 atomic 方法可以让这种操作具有原子性如 getAndIncrement ()方法会原子性的进行增量操作把当前值加一，其它数据类型和引用变量也可以进行相似操作。

47) 如果同步块内的线程抛出异常会发生什么？

这个问题坑了很多 Java 程序员，若你能想到锁是否释放这条线索来回答还有点希望答对。无论你的同步块是正常还是异常退出的，里面的线程都会释放锁，所以对比锁接口我更喜欢同步块，因为它不用我花费精力去释放锁，该功能可以在 finally block 里释放锁实现。

48） 单例模式的双检锁是什么？

这个问题在 Java 面试中经常被问到，但是面试官对回答此问题的满意度仅为 50%。一半的人写不出双检锁还有一半的人说不出它的隐患和 Java1.5 是如何对它修正的。它其实是一个用来创建线程安全的单例的老方法，当单例实例第一次被创建时它试图用单个锁进行性能优化，但是由于太过于复杂在 JDK1.4 中它是失败的，我个人也不喜欢它。无论如何，即便你也不喜欢它但是还是要了解一下，因为它经常被问到。你可以查看 how double checked locking on Singleton works 这篇文章获得更多信息。

49） 如何在 Java 中创建线程安全的 Singleton？

这是上面那个问题的后续，如果你不喜欢双检锁而面试官问了创建 Singleton 类的替代方法，你可以利用 JVM 的类加载和静态变量初始化特征来创建 Singleton 实例，或者是利用枚举类型来创建 Singleton，我很喜欢用这种方法。你可以查看这篇文章获得更多信息。

50) 写出 3 条你遵循的多线程最佳实践

这种问题我最喜欢了，我相信你在写并发代码来提升性能的时候也会遵循某些最佳实践。以下三条最佳实践我觉得大多数 Java 程序员都应该遵循：

给你的线程起个有意义的名字。

这样可以方便找 bug 或追踪。OrderProcessor, QuoteProcessor or TradeProcessor 这种名字比 Thread-1. Thread-2 and Thread-3 好多了，给线程起一个和它要完成的任务相关的名字，所有的主要框架甚至 JDK 都遵循这个最佳实践。

避免锁定和缩小同步的范围

锁花费的代价高昂且上下文切换更耗费时间空间，试试最低限度的使用同步和锁，缩小临界区。因此相对于同步方法我更喜欢同步块，它给我拥有对锁的绝对控制权。

多用同步类少用 wait 和 notify

首先，CountDownLatch, Semaphore, CyclicBarrier 和 Exchanger 这些同步类简化了编码操作，而用 wait 和 notify 很难实现对复杂控制流的控制。其次，这些类是由最好的企业编写和维护在后续的 JDK 中它们还会不断优化和完善，使用这些更高等级的同步工具你的程序可以不费吹灰之力获得优化。

多用并发集合少用同步集合　　这是另外一个容易遵循且受益巨大的最佳实践，并发集合比同步集合的可扩展性更好，所以在并发编程时使用并发集合效果更好。如果下一次你需要用到 map，你应该首先想到用 ConcurrentHashMap。我的文章 Java 并发集合有更详细的说明。

51) 如何强制启动一个线程？

这个问题就像是如何强制进行 Java 垃圾回收，目前还没有觉得方法，虽然你可以使用 System.gc ()来进行垃圾回收，但是不保证能成功。在 Java 里面没有办法强制启动一个线程，它是被线程调度器控制着且 Java 没有公布相关的 API。

52) Java 中的 fork join 框架是什么？

fork join 框架是 JDK7 中出现的一款高效的工具，Java 开发人员可以通过它充分利用现代服务器上的多处理器。它是专门为了那些可以递归划分成许多子模块设计的，目的是将所有可用的处理能力用来提升程序的性能。fork join 框架一个巨大的优势是它使用了工作窃取算法，可以完成更多任务的工作线程可以从其它线程中窃取任务来执行。你可以查看这篇文章获得更多信息。

53） Java 多线程中调用 wait () 和 sleep ()方法有什么不同？

Java 程序中 wait 和 sleep 都会造成某种形式的暂停，它们可以满足不同的需要。wait ()方法用于线程间通信，如果等待条件为真且其它线程被唤醒时它会释放锁，而 sleep ()方法仅仅释放 CPU 资源或者让当前线程停止执行一段时间，但不会释放锁。你可以查看这篇文章获得更多信息。

阿里云>

云栖社区>

正文

50 道Java 线程面试题(经典）\_java

下面是 Java 线程相关的热门面试题，你可以用它来好好准备面试。

1) 什么是线程？

线程是操作系统能够进行运算调度的最小单位，它被包含在进程之中，是进程中的实际运作单位。程序员可以通过它进行多处理器编程，你可以使用多线程对运算密集型任务提速。比如，如果一个线程完成一个任务要 100 毫秒，那么用十个线程完成改任务只需 10 毫秒。Java 在语言层面对多线程提供了卓越的支持，它也是一个很好的卖点。欲了解更多详细信息请点击这里。

2) 线程和进程有什么区别？

线程是进程的子集，一个进程可以有很多线程，每条线程并行执行不同的任务。不同的进程使用不同的内存空间，而所有的线程共享一片相同的内存空间。别把它和栈内存搞混，每个线程都拥有单独的栈内存用来存储本地数据。更多详细信息请点击这里。

3) 如何在 Java 中实现线程？

在语言层面有两种方式。java.lang.Thread 类的实例就是一个线程但是它需要调用 java.lang.Runnable 接口来执行，由于线程类本身就是调用的 Runnable 接口所以你可以继承 java.lang.Thread 类或者直接调用 Runnable 接口来重写 run ()方法实现线程。更多详细信息请点击这里.

4) 用 Runnable 还是 Thread？

这个问题是上题的后续，大家都知道我们可以通过继承 Thread 类或者调用 Runnable 接口来实现线程，问题是，那个方法更好呢？什么情况下使用它？这个问题很容易回答，如果你知道 Java 不支持类的多重继承，但允许你调用多个接口。所以如果你要继承其他类，当然是调用 Runnable 接口好了。更多详细信息请点击这里。

6) Thread 类中的 start () 和 run () 方法有什么区别？

这个问题经常被问到，但还是能从此区分出面试者对 Java 线程模型的理解程度。start ()方法被用来启动新创建的线程，而且 start ()内部调用了 run ()方法，这和直接调用 run ()方法的效果不一样。当你调用 run ()方法的时候，只会是在原来的线程中调用，没有新的线程启动，start ()方法才会启动新线程。更多讨论请点击这里

7) Java 中 Runnable 和 Callable 有什么不同？

Runnable 和 Callable 都代表那些要在不同的线程中执行的任务。Runnable 从 JDK1.0 开始就有了，Callable 是在 JDK1.5 增加的。它们的主要区别是 Callable 的 call () 方法可以返回值和抛出异常，而 Runnable 的 run ()方法没有这些功能。Callable 可以返回装载有计算结果的 Future 对象。我的博客有更详细的说明。

8) Java 中 CyclicBarrier 和 CountDownLatch 有什么不同？

CyclicBarrier 和 CountDownLatch 都可以用来让一组线程等待其它线程。与 CyclicBarrier 不同的是，CountdownLatch 不能重新使用。点此查看更多信息和示例代码。

9) Java 内存模型是什么？

Java 内存模型规定和指引 Java 程序在不同的内存架构、CPU 和操作系统间有确定性地行为。它在多线程的情况下尤其重要。Java 内存模型对一个线程所做的变动能被其它线程可见提供了保证，它们之间是先行发生关系。这个关系定义了一些规则让程序员在并发编程时思路更清晰。比如，先行发生关系确保了：

线程内的代码能够按先后顺序执行，这被称为程序次序规则。

对于同一个锁，一个解锁操作一定要发生在时间上后发生的另一个锁定操作之前，也叫做管程锁定规则。

前一个对 Volatile的写操作在后一个volatile的读操作之前，也叫volatile变量规则。

一个线程内的任何操作必需在这个线程的 start ()调用之后，也叫作线程启动规则。

一个线程的所有操作都会在线程终止之前，线程终止规则。

一个对象的终结操作必需在这个对象构造完成之后，也叫对象终结规则。

可传递性

我强烈建议大家阅读《Java 并发编程实践》第十六章来加深对 Java 内存模型的理解。

10) Java 中的 volatile 变量是什么？

volatile 是一个特殊的修饰符，只有成员变量才能使用它。在 Java 并发程序缺少同步类的情况下，多线程对成员变量的操作对其它线程是透明的。volatile 变量可以保证下一个读取操作会在前一个写操作之后发生，就是上一题的 volatile 变量规则。点击这里查看更多 volatile 的相关内容。

11) 什么是线程安全？Vector 是一个线程安全类吗？ （详见这里)

如果你的代码所在的进程中有多个线程在同时运行，而这些线程可能会同时运行这段代码。如果每次运行结果和单线程运行的结果是一样的，而且其他的变量的值也和预期的是一样的，就是线程安全的。一个线程安全的计数器类的同一个实例对象在被多个线程使用的情况下也不会出现计算失误。很显然你可以将集合类分成两组，线程安全和非线程安全的。Vector 是用同步方法来实现线程安全的， 而和它相似的 ArrayList 不是线程安全的。

12) Java 中什么是竞态条件？ 举个例子说明。

竞态条件会导致程序在并发情况下出现一些 bugs。多线程对一些资源的竞争的时候就会产生竞态条件，如果首先要执行的程序竞争失败排到后面执行了，那么整个程序就会出现一些不确定的 bugs。这种 bugs 很难发现而且会重复出现，因为线程间的随机竞争。一个例子就是无序处理，详见答案。

13) Java 中如何停止一个线程？

Java 提供了很丰富的 API 但没有为停止线程提供 API。JDK 1.0 本来有一些像 stop ()， suspend () 和 resume ()的控制方法但是由于潜在的死锁威胁因此在后续的 JDK 版本中他们被弃用了，之后 Java API 的设计者就没有提供一个兼容且线程安全的方法来停止一个线程。当 run () 或者 call () 方法执行完的时候线程会自动结束，如果要手动结束一个线程，你可以用 volatile 布尔变量来退出 run ()方法的循环或者是取消任务来中断线程。点击这里查看示例代码。

14) 一个线程运行时发生异常会怎样？

这是我在一次面试中遇到的一个很刁钻的 Java 面试题, 简单的说，如果异常没有被捕获该线程将会停止执行。

Thread.UncaughtExceptionHandler 是用于处理未捕获异常造成线程突然中断情况的一个内嵌接口。当一个未捕获异常将造成线程中断的时候 JVM 会使用 Thread.getUncaughtExceptionHandler ()来查询线程的 UncaughtExceptionHandler 并将线程和异常作为参数传递给 handler 的 uncaughtException ()方法进行处理。

15） 如何在两个线程间共享数据？

你可以通过共享对象来实现这个目的，或者是使用像阻塞队列这样并发的数据结构。这篇教程《Java 线程间通信》(涉及到在两个线程间共享对象)用 wait 和 notify 方法实现了生产者消费者模型。

16) Java 中 notify 和 notifyAll 有什么区别？

这又是一个刁钻的问题，因为多线程可以等待单监控锁，Java API 的设计人员提供了一些方法当等待条件改变的时候通知它们，但是这些方法没有完全实现。notify ()方法不能唤醒某个具体的线程，所以只有一个线程在等待的时候它才有用武之地。而 notifyAll ()唤醒所有线程并允许他们争夺锁确保了至少有一个线程能继续运行。我的博客有更详细的资料和示例代码。

17) 为什么 wait, notify 和 notifyAll 这些方法不在 thread 类里面？

这是个设计相关的问题，它考察的是面试者对现有系统和一些普遍存在但看起来不合理的事物的看法。回答这些问题的时候，你要说明为什么把这些方法放在 Object 类里是有意义的，还有不把它放在 Thread 类里的原因。一个很明显的原因是 JAVA 提供的锁是对象级的而不是线程级的，每个对象都有锁，通过线程获得。如果线程需要等待某些锁那么调用对象中的 wait ()方法就有意义了。如果 wait ()方法定义在 Thread 类中，线程正在等待的是哪个锁就不明显了。简单的说，由于 wait，notify 和 notifyAll 都是锁级别的操作，所以把他们定义在 Object 类中因为锁属于对象。你也可以查看这篇文章了解更多。

18) 什么是 ThreadLocal 变量？

ThreadLocal 是 Java 里一种特殊的变量。每个线程都有一个 ThreadLocal 就是每个线程都拥有了自己独立的一个变量，竞争条件被彻底消除了。它是为创建代价高昂的对象获取线程安全的好方法，比如你可以用 ThreadLocal 让 SimpleDateFormat 变成线程安全的，因为那个类创建代价高昂且每次调用都需要创建不同的实例所以不值得在局部范围使用它，如果为每个线程提供一个自己独有的变量拷贝，将大大提高效率。首先，通过复用减少了代价高昂的对象的创建个数。其次，你在没有使用高代价的同步或者不变性的情况下获得了线程安全。线程局部变量的另一个不错的例子是 ThreadLocalRandom 类，它在多线程环境中减少了创建代价高昂的 Random 对象的个数。查看答案了解更多。

19) 什么是 FutureTask？

在 Java 并发程序中 FutureTask 表示一个可以取消的异步运算。它有启动和取消运算、查询运算是否完成和取回运算结果等方法。只有当运算完成的时候结果才能取回，如果运算尚未完成 get 方法将会阻塞。一个 FutureTask 对象可以对调用了 Callable 和 Runnable 的对象进行包装，由于 FutureTask 也是调用了 Runnable 接口所以它可以提交给 Executor 来执行。

20) Java 中 interrupted 和 isInterruptedd 方法的区别？

interrupted () 和 isInterrupted ()的主要区别是前者会将中断状态清除而后者不会。Java 多线程的中断机制是用内部标识来实现的，调用Thread.interrupt ()来中断一个线程就会设置中断标识为 true。当中断线程调用静态方法Thread.interrupted ()来检查中断状态时，中断状态会被清零。而非静态方法 isInterrupted ()用来查询其它线程的中断状态且不会改变中断状态标识。简单的说就是任何抛出 InterruptedException 异常的方法都会将中断状态清零。无论如何，一个线程的中断状态有有可能被其它线程调用中断来改变。

21) 为什么 wait 和 notify 方法要在同步块中调用？

主要是因为 Java API 强制要求这样做，如果你不这么做，你的代码会抛出 IllegalMonitorStateException 异常。还有一个原因是为了避免 wait 和 notify 之间产生竞态条件。

22) 为什么你应该在循环中检查等待条件?

处于等待状态的线程可能会收到错误警报和伪唤醒，如果不在循环中检查等待条件，程序就会在没有满足结束条件的情况下退出。因此，当一个等待线程醒来时，不能认为它原来的等待状态仍然是有效的，在 notify ()方法调用之后和等待线程醒来之前这段时间它可能会改变。这就是在循环中使用 wait ()方法效果更好的原因，你可以在 Eclipse 中创建模板调用 wait 和 notify 试一试。如果你想了解更多关于这个问题的内容，我推荐你阅读《Effective Java》这本书中的线程和同步章节。

23) Java 中的同步集合与并发集合有什么区别？

同步集合与并发集合都为多线程和并发提供了合适的线程安全的集合，不过并发集合的可扩展性更高。在 Java1.5 之前程序员们只有同步集合来用且在多线程并发的时候会导致争用，阻碍了系统的扩展性。Java5 介绍了并发集合像 ConcurrentHashMap，不仅提供线程安全还用锁分离和内部分区等现代技术提高了可扩展性。更多内容详见答案。

24） Java 中堆和栈有什么不同？

为什么把这个问题归类在多线程和并发面试题里？因为栈是一块和线程紧密相关的内存区域。每个线程都有自己的栈内存，用于存储本地变量，方法参数和栈调用，一个线程中存储的变量对其它线程是不可见的。而堆是所有线程共享的一片公用内存区域。对象都在堆里创建，为了提升效率线程会从堆中弄一个缓存到自己的栈，如果多个线程使用该变量就可能引发问题，这时 volatile 变量就可以发挥作用了，它要求线程从主存中读取变量的值。

更多内容详见答案。

25） 什么是线程池？ 为什么要使用它？

创建线程要花费昂贵的资源和时间，如果任务来了才创建线程那么响应时间会变长，而且一个进程能创建的线程数有限。为了避免这些问题，在程序启动的时候就创建若干线程来响应处理，它们被称为线程池，里面的线程叫工作线程。从 JDK1.5 开始，Java API 提供了 Executor 框架让你可以创建不同的线程池。比如单线程池，每次处理一个任务；数目固定的线程池或者是缓存线程池（一个适合很多生存期短的任务的程序的可扩展线程池）。更多内容详见这篇文章。

26） 如何写代码来解决生产者消费者问题？

在现实中你解决的许多线程问题都属于生产者消费者模型，就是一个线程生产任务供其它线程进行消费，你必须知道怎么进行线程间通信来解决这个问题。比较低级的办法是用 wait 和 notify 来解决这个问题，比较赞的办法是用 Semaphore 或者 BlockingQueue 来实现生产者消费者模型，这篇教程有实现它。

27） 如何避免死锁？

Java 多线程中的死锁

死锁是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。这是一个严重的问题，因为死锁会让你的程序挂起无法完成任务，死锁的发生必须满足以下四个条件：

互斥条件：一个资源每次只能被一个进程使用。

请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

不剥夺条件：进程已获得的资源，在末使用完之前，不能强行剥夺。

循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。

避免死锁最简单的方法就是阻止循环等待条件，将系统中所有的资源设置标志位、排序，规定所有的进程申请资源必须以一定的顺序（升序或降序）做操作来避免死锁。这篇教程有代码示例和避免死锁的讨论细节。

28) Java 中活锁和死锁有什么区别？

这是上题的扩展，活锁和死锁类似，不同之处在于处于活锁的线程或进程的状态是不断改变的，活锁可以认为是一种特殊的饥饿。一个现实的活锁例子是两个人在狭小的走廊碰到，两个人都试着避让对方好让彼此通过，但是因为避让的方向都一样导致最后谁都不能通过走廊。简单的说就是，活锁和死锁的主要区别是前者进程的状态可以改变但是却不能继续执行。

29） 怎么检测一个线程是否拥有锁？

我一直不知道我们竟然可以检测一个线程是否拥有锁，直到我参加了一次电话面试。在 java.lang.Thread 中有一个方法叫 holdsLock ()，它返回 true 如果当且仅当当前线程拥有某个具体对象的锁。你可以查看这篇文章了解更多。

30) 你如何在 Java 中获取线程堆栈？

对于不同的操作系统，有多种方法来获得 Java 进程的线程堆栈。当你获取线程堆栈时，JVM 会把所有线程的状态存到日志文件或者输出到控制台。在 Windows 你可以使用 Ctrl + Break 组合键来获取线程堆栈，Linux 下用 kill -3 命令。你也可以用 jstack 这个工具来获取，它对线程 id 进行操作，你可以用 jps 这个工具找到 id。

31) JVM 中哪个参数是用来控制线程的栈堆栈小的

这个问题很简单， -Xss 参数用来控制线程的堆栈大小。你可以查看 JVM 配置列表来了解这个参数的更多信息。

32） Java 中 synchronized 和 ReentrantLock 有什么不同？

Java 在过去很长一段时间只能通过 synchronized 关键字来实现互斥，它有一些缺点。比如你不能扩展锁之外的方法或者块边界，尝试获取锁时不能中途取消等。Java 5 通过 Lock 接口提供了更复杂的控制来解决这些问题。 ReentrantLock 类实现了 Lock，它拥有与 synchronized 相同的并发性和内存语义且它还具有可扩展性。你可以查看这篇文章了解更多

33） 有三个线程 T1，T2，T3，怎么确保它们按顺序执行？

在多线程中有多种方法让线程按特定顺序执行，你可以用线程类的 join ()方法在一个线程中启动另一个线程，另外一个线程完成该线程继续执行。为了确保三个线程的顺序你应该先启动最后一个(T3 调用 T2，T2 调用 T1)，这样 T1 就会先完成而 T3 最后完成。你可以查看这篇文章了解更多。

34) Thread 类中的 yield 方法有什么作用？

Yield 方法可以暂停当前正在执行的线程对象，让其它有相同优先级的线程执行。它是一个静态方法而且只保证当前线程放弃 CPU 占用而不能保证使其它线程一定能占用 CPU，执行 yield ()的线程有可能在进入到暂停状态后马上又被执行。点击这里查看更多 yield 方法的相关内容。

35） Java 中 ConcurrentHashMap 的并发度是什么？

ConcurrentHashMap 把实际 map 划分成若干部分来实现它的可扩展性和线程安全。这种划分是使用并发度获得的，它是 ConcurrentHashMap 类构造函数的一个可选参数，默认值为 16，这样在多线程情况下就能避免争用。欲了解更多并发度和内部大小调整请阅读我的文章 How ConcurrentHashMap works in Java。

36） Java 中 Semaphore 是什么？

Java 中的 Semaphore 是一种新的同步类，它是一个计数信号。从概念上讲，从概念上讲，信号量维护了一个许可集合。如有必要，在许可可用前会阻塞每一个 acquire ()，然后再获取该许可。每个 release ()添加一个许可，从而可能释放一个正在阻塞的获取者。但是，不使用实际的许可对象，Semaphore 只对可用许可的号码进行计数，并采取相应的行动。信号量常常用于多线程的代码中，比如数据库连接池。更多详细信息请点击这里。

37）如果你提交任务时，线程池队列已满。会时发会生什么？

这个问题问得很狡猾，许多程序员会认为该任务会阻塞直到线程池队列有空位。事实上如果一个任务不能被调度执行那么 ThreadPoolExecutor's submit ()方法将会抛出一个 RejectedExecutionException 异常。

38) Java 线程池中 submit () 和 execute ()方法有什么区别？

两个方法都可以向线程池提交任务，execute ()方法的返回类型是 void，它定义在 Executor 接口中， 而 submit ()方法可以返回持有计算结果的 Future 对象，它定义在 ExecutorService 接口中，它扩展了 Executor 接口，其它线程池类像 ThreadPoolExecutor 和 ScheduledThreadPoolExecutor 都有这些方法。更多详细信息请点击这里。

39) 什么是阻塞式方法？

阻塞式方法是指程序会一直等待该方法完成期间不做其他事情，ServerSocket 的 accept ()方法就是一直等待客户端连接。这里的阻塞是指调用结果返回之前，当前线程会被挂起，直到得到结果之后才会返回。此外，还有异步和非阻塞式方法在任务完成前就返回。更多详细信息请点击这里。

40) Swing 是线程安全的吗？ 为什么？

你可以很肯定的给出回答，Swing 不是线程安全的，但是你应该解释这么回答的原因即便面试官没有问你为什么。当我们说 swing 不是线程安全的常常提到它的组件，这些组件不能在多线程中进行修改，所有对 GUI 组件的更新都要在 AWT 线程中完成，而 Swing 提供了同步和异步两种回调方法来进行更新。点击这里查看更多 swing 和线程安全的相关内容。

41） Java 中 invokeAndWait 和 invokeLater 有什么区别？

这两个方法是 Swing API 提供给 Java 开发者用来从当前线程而不是事件派发线程更新 GUI 组件用的。InvokeAndWait ()同步更新 GUI 组件，比如一个进度条，一旦进度更新了，进度条也要做出相应改变。如果进度被多个线程跟踪，那么就调用 invokeAndWait ()方法请求事件派发线程对组件进行相应更新。而 invokeLater ()方法是异步调用更新组件的。更多详细信息请点击这里。

42) Swing API 中那些方法是线程安全的？

这个问题又提到了 swing 和线程安全，虽然组件不是线程安全的但是有一些方法是可以被多线程安全调用的，比如 repaint ()， revalidate ()。 JTextComponent 的 setText ()方法和 JTextArea 的 insert () 和 append () 方法也是线程安全的。

43) 如何在 Java 中创建 Immutable 对象？

这个问题看起来和多线程没什么关系， 但不变性有助于简化已经很复杂的并发程序。Immutable 对象可以在没有同步的情况下共享，降低了对该对象进行并发访问时的同步化开销。可是 Java 没有@Immutable 这个注解符，要创建不可变类，要实现下面几个步骤：通过构造方法初始化所有成员、对变量不要提供 setter 方法、将所有的成员声明为私有的，这样就不允许直接访问这些成员、在 getter 方法中，不要直接返回对象本身，而是克隆对象，并返回对象的拷贝。我的文章 how to make an object Immutable in Java 有详细的教程，看完你可以充满自信。

44） Java 中的 ReadWriteLock 是什么？

一般而言，读写锁是用来提升并发程序性能的锁分离技术的成果。Java 中的 ReadWriteLock 是 Java 5 中新增的一个接口，一个 ReadWriteLock 维护一对关联的锁，一个用于只读操作一个用于写。在没有写线程的情况下一个读锁可能会同时被多个读线程持有。写锁是独占的，你可以使用 JDK 中的 ReentrantReadWriteLock 来实现这个规则，它最多支持 65535 个写锁和 65535 个读锁。

45) 多线程中的忙循环是什么?

忙循环就是程序员用循环让一个线程等待，不像传统方法 wait ()， sleep () 或 yield () 它们都放弃了 CPU 控制，而忙循环不会放弃 CPU，它就是在运行一个空循环。这么做的目的是为了保留 CPU 缓存，在多核系统中，一个等待线程醒来的时候可能会在另一个内核运行，这样会重建缓存。为了避免重建缓存和减少等待重建的时间就可以使用它了。你可以查看这篇文章获得更多信息。

46）volatile 变量和 atomic 变量有什么不同？

这是个有趣的问题。首先，volatile 变量和 atomic 变量看起来很像，但功能却不一样。Volatile 变量可以确保先行关系，即写操作会发生在后续的读操作之前， 但它并不能保证原子性。例如用 volatile 修饰 count 变量那么 count++ 操作就不是原子性的。而 AtomicInteger 类提供的 atomic 方法可以让这种操作具有原子性如 getAndIncrement ()方法会原子性的进行增量操作把当前值加一，其它数据类型和引用变量也可以进行相似操作。

47) 如果同步块内的线程抛出异常会发生什么？

这个问题坑了很多 Java 程序员，若你能想到锁是否释放这条线索来回答还有点希望答对。无论你的同步块是正常还是异常退出的，里面的线程都会释放锁，所以对比锁接口我更喜欢同步块，因为它不用我花费精力去释放锁，该功能可以在 finally block 里释放锁实现。

48） 单例模式的双检锁是什么？

这个问题在 Java 面试中经常被问到，但是面试官对回答此问题的满意度仅为 50%。一半的人写不出双检锁还有一半的人说不出它的隐患和 Java1.5 是如何对它修正的。它其实是一个用来创建线程安全的单例的老方法，当单例实例第一次被创建时它试图用单个锁进行性能优化，但是由于太过于复杂在 JDK1.4 中它是失败的，我个人也不喜欢它。无论如何，即便你也不喜欢它但是还是要了解一下，因为它经常被问到。你可以查看 how double checked locking on Singleton works 这篇文章获得更多信息。

49） 如何在 Java 中创建线程安全的 Singleton？

这是上面那个问题的后续，如果你不喜欢双检锁而面试官问了创建 Singleton 类的替代方法，你可以利用 JVM 的类加载和静态变量初始化特征来创建 Singleton 实例，或者是利用枚举类型来创建 Singleton，我很喜欢用这种方法。你可以查看这篇文章获得更多信息。

50) 写出 3 条你遵循的多线程最佳实践

这种问题我最喜欢了，我相信你在写并发代码来提升性能的时候也会遵循某些最佳实践。以下三条最佳实践我觉得大多数 Java 程序员都应该遵循：

给你的线程起个有意义的名字。

这样可以方便找 bug 或追踪。OrderProcessor, QuoteProcessor or TradeProcessor 这种名字比 Thread-1. Thread-2 and Thread-3 好多了，给线程起一个和它要完成的任务相关的名字，所有的主要框架甚至 JDK 都遵循这个最佳实践。

避免锁定和缩小同步的范围

锁花费的代价高昂且上下文切换更耗费时间空间，试试最低限度的使用同步和锁，缩小临界区。因此相对于同步方法我更喜欢同步块，它给我拥有对锁的绝对控制权。

多用同步类少用 wait 和 notify

首先，CountDownLatch, Semaphore, CyclicBarrier 和 Exchanger 这些同步类简化了编码操作，而用 wait 和 notify 很难实现对复杂控制流的控制。其次，这些类是由最好的企业编写和维护在后续的 JDK 中它们还会不断优化和完善，使用这些更高等级的同步工具你的程序可以不费吹灰之力获得优化。

多用并发集合少用同步集合　　这是另外一个容易遵循且受益巨大的最佳实践，并发集合比同步集合的可扩展性更好，所以在并发编程时使用并发集合效果更好。如果下一次你需要用到 map，你应该首先想到用 ConcurrentHashMap。我的文章 Java 并发集合有更详细的说明。

51) 如何强制启动一个线程？

这个问题就像是如何强制进行 Java 垃圾回收，目前还没有觉得方法，虽然你可以使用 System.gc ()来进行垃圾回收，但是不保证能成功。在 Java 里面没有办法强制启动一个线程，它是被线程调度器控制着且 Java 没有公布相关的 API。

52) Java 中的 fork join 框架是什么？

fork join 框架是 JDK7 中出现的一款高效的工具，Java 开发人员可以通过它充分利用现代服务器上的多处理器。它是专门为了那些可以递归划分成许多子模块设计的，目的是将所有可用的处理能力用来提升程序的性能。fork join 框架一个巨大的优势是它使用了工作窃取算法，可以完成更多任务的工作线程可以从其它线程中窃取任务来执行。你可以查看这篇文章获得更多信息。

53） Java 多线程中调用 wait () 和 sleep ()方法有什么不同？

Java 程序中 wait 和 sleep 都会造成某种形式的暂停，它们可以满足不同的需要。wait ()方法用于线程间通信，如果等待条件为真且其它线程被唤醒时它会释放锁，而 sleep ()方法仅仅释放 CPU 资源或者让当前线程停止执行一段时间，但不会释放锁。你可以查看这篇文章获得更多信息。

一、 SSH框架相关的面试题:

1. BeanFactory的作用是什么？ [中]

BeanFactory是配置、创建、管理bean的容器，有时候也称为bean上下文。Bean与bean的依赖关系，也是由BeanFactory负责维护的。

2. bean在spring的容器中两种基本行为是什么? [中]

Singleton:单态

Non-singleton或prototype:原型

3. spring配置文件中的ref元素的作用是什么?它的两个基本属性是什么? [中]

用于指定属性值为spring容器中的其它bean.两个基本属性是bean和local

4. 什么是DWR?它有哪些功能？ [中]

DWR(Direct Web Remoting)是一个WEB远程调用框架. 可以在客户端利用JavaScript直接调用服务端的Java方法并返回值给JavaScript DWR根据Java类来动态生成JavaScrip代码.

支持Dom Trees,支持Spring,支持commons-logging

5. Ajax的工作原理 ？　［难］

在会话的开始，浏览器加载Ajax引擎

请求动作通过JavaScript调用Ajax引擎来代替.

引擎负责绘制用户界面以及与服务器端通讯。

Ajax引擎采用异步交互过程－－不用等待服务器的通讯。

6. XMLHttpRequest对象是什么，它有哪些常用方法？ [难]

一个 JavaScript 对象。是Ajax 的核心。

该对象的方法和属性。

open()：建立到服务器的新请求。

send()：向服务器发送请求。

abort()：退出当前请求。

readyState：提供当前 HTML 的就绪状态。

responseText：服务器返回的请求响应文本。.

7. XMLHttpRequest有哪些属性， 它的值分代表什么？[ 中]

8. 什么是Ajax? [易]

AJAX（Asynchronous JavaScript and XML），它不是一门新的语言或技术，而是多种技术的综合，包括：

Javascript

XHTML

CSS

DOM

XML

XSTL

XMLHttpRequest

9. 同步与异步的区别？ [ 难]

普通B/S模式（同步）AJAX技术（异步）

同步：提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事，而异步则是 请求通过事件触发->服务器处理->处理完毕

同步是阻塞模式，异步是非阻塞模式。

同步(发送方发出数据后，等接收方发回) 异步(发送方发出数据后，不等接收方发回响应)

10. Struts+Hibernate+Spring开发模式中Hibernate扮演的角色 [中]

答：Hibernate扮演的是数据持久层 它的作用是实现持久化对象和数据库表

之间的映射，形成持久化对象和数据库表中数据的一个转换平台

11. 什么是Hibernate 的抓取策略 [难]

抓取策略是指当应用程序需要利用关联关系进行对象获取的时候。

12. 何为容器 [中]

容器就是符合某种规范的能够提供一系列服务的管理器。

13. Spring实现了那几种模式 [中]

答: 工厂模式 和 单例模式

14 :Hibernate实体对象生命周期的三种状态 [难]

答 Transient(瞬态) ，persistent(持久态)和Detached(游历态)

15: HQL查询参数的处理方法 [中]

答：直接将参数拼写为HQL语句

通过参数名称来标识参数

16. Hibernate实体之间的关联关系的三种形式 [中]

答 一对一关联

一对多关联

多对多关联

17. Hibernate具用的三种检索方式 [难]

答：HQL检索方式

QBC检索方式

SQL检索方式

18. spring 中<Bean>的depends-on属性是什么 [中]

Depends-on用于当前Bean初始化之前显示的强制一个或多个bean被初始化

19. spring 中的BeanWrapper类是什么 [难]

BeanWrapper类是一个对JavaBean进行各种操作的工具类

BeanWrapper本身是一个接口BeanWrapperImpl实现了BeanWrapper

20 . <set>元素有一个cascade属性，如果希望Hibernate级联保存集合中的对象，casecade属性应该取什么值？ [难]

答:save-update

21. Hibernate中session什么时候是游离态 [中]

答:session.close();方法后

22. Hibernate中映射文件<hibernate-mapping> inverse属性和cascade属性的区别 [中]

答:inverse属性只存在于集合标记的元素中集合元素包括<set/>,<map/>,<list/>,<array/>,<bag/>

Inverse属性的作用是是否将对集合对象的修改反射到数据库中 inverse=”false”时修改反射到数据库中 inverse=”true” 时修改不反射到数据库中

Cascade属性的作用是描述关联对象进行操作时的级联特性，因此只有涉及到关系的元素才有cascade属性

23. : Hibernate中Session对象的load()方法和get()方法的区别（请列举出两点）[难]

答：①记录不存在时 get()方法会返回空（null），而load()方法将会抛出一个HibernateException异常 ② load()方法查询数据时会先找Hibernate的内部缓存和二级缓

存中的现有数据，get()方法在内部缓存中没有达到相对应的数据时装直接执行SQL语句

进行查询

24. : Hibernate中HQL属于什么语言 [易]

答：HQL是面向对象的查询语言，它可以查询以对象形式存在的数据。

25. Hibernate简介以及主要功能 [中]

答：Hibernate是采用ORM模式实现数据持久层的一个优秀的JAVA组件，它提供了

强大，高效的将JAVA对象进行持久化操作的服务

26. 简述Hibernate的优点 [难]

答：开源和免费的License，我可以在需要的时候研究源代码，改写源代码，进行功能的定制。轻量级封装，避免引入过多复杂的问题，调试容易，也减轻程序员的负担。

具有可扩展性，API开放，当本身功能不够用的时候，可以自己编码进行扩展。

27. 怎样构建SessionFactory [难]

答: Hibernate的SessionFactory接口提供Session类的实例，Session类用于完成对数据库的操作。

由于SessionFactory实例是线程安全的（而Session实例不是线程安全的），所以每个操作都可以共用同一个SessionFactory来获取Session。Hibernate配置文件分为两种格式，一种是xml格式的配置文件，另一种是Java属性文件格式的配置文件

28. :从XML文件读取配置信息构建SessionFactory的具体步骤如下。 [难]

（1）创建一个Configuration对象，并通过该对象的configura()方法加载Hibernate配置文件，代码如下。

Configuration config = new Configuration().configure();

configura()方法：用于告诉Hibernate加载hibernate.cfg.xml文件。Configuration在实例化时默认加载classpath中的hibernate.cfg.xml，当然也可以加载名称不是hibernate.cfg.xml的配置文件，例如wghhibernate.cfg.xml，可以通过以下代码实现。

Configuration config = new Configuration().configure("wghhibernate.cfg.xml");

（2）完成配置文件和映射文件的加载后，将得到一个包括所有Hibernate运行期参数的Configuration实例，通过Configuration实例的buildSessionFactory()方法可以构建一个惟一的SessionFactory，代码如下:

SessionFactory sessionFactory = config.buildSessionFactory();

构建SessionFactory要放在静态代码块中，因为它只在该类被加载时执行一次。

29. 写出使用构造方法进行注入的关键代码 [难]

<bean id=”constructBean” class=”com.huang.ConstructBean”>

<constructor-arg><ref bean=”bean\_1”/></ constructor-arg >

<constructor-arg><ref bean=”bean\_2”/></ constructor-arg >

<constructor-arg><ref bean=”bean\_3”/></ constructor-arg >

</bean>

30. 什么是IOC？ [难]

不创建对象，但是描述创建它们的方式。在代码中不直接与对象和服务连接，但在配置文件中描述哪一个组件需要哪一项服务。容器（在Spring 框架中是 IOC 容器） 负责将这些联系在一起。

就是由容器控制程序之间的关系，而非传统实现中，由程序代码直接操控，控制权由应用代码中转到了外部容器，控制权的转移，就是所谓的反转。

31. 编程题: 写一个Singleton出来。 [难]

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。

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，在类初始化时实例话，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。

[java] view plain copy

print?

public class Singleton {

private Singleton(){}

　　 private static Singleton instance = new Singleton();

　　 public static Singleton getInstance() {

　　　　 return instance;

　　 }

}

第二种形式:

public class Singleton {

　　private static Singleton instance = null;

　　public static synchronized Singleton getInstance() {

　　if (instance==null)

　　　　instance＝new Singleton();

return instance; 　　}

}

[java] view plain copy

print?

String tempStr = "";

try {

tempStr = new String(str.getBytes("ISO-8859-1"), "GBK");

tempStr = tempStr.trim();

}

catch (Exception e) {

System.err.println(e.getMessage());

}

return tempStr;

}

32. J2EE是技术还是平台还是框架？ [中]

J2EE本身是一个标准，一个为企业分布式应用的开发提供的标准平台。

J2EE也是一个框架，包括JDBC、JNDI、RMI、JMS、EJB、JTA等技术。

33. 我们在web应用开发过程中经常遇到输出某种编码的字符，如iso8859-1等，如何输出一个某种编码的字符串？ [难]

Public String translate (String str) {

String tempStr = "";

try {

tempStr = new String(str.getBytes("ISO-8859-1"), "GBK");

tempStr = tempStr.trim();

}

catch (Exception e) {

System.err.println(e.getMessage());

}

return tempStr;

}

３４．如何使用静态工厂方法来创建Bean的实例 [易]

<bean id=”staticFactoryBean” class=”com.huang.StaticFactoryBean”

factory-method=”静态方法名”/>

相当于: StaticFactoryBean staticFactoryBean=StaticFactoryBean.静态方法名

使用实例化的工厂方法来创建Bean的实例

<!—注意此时bean的class属性为空-->

<bean id=”dynamicFactory”class=”com.huang.DynamicFactory”/>

<bean id=”dynamiceFactoryBean” factory-bean=”dynamicFactory”

Factory-method=”方法名”/>

相当于: DynamicFactory dynamicFactory=new DynamicFactory();

DynamiceFactoryBean dynamiceFactoryBean=dynamicFactory.方法名

35. 从Java属性文件读取配置信息构建SessionFactory的具体步骤如下: [难]

（1）创建一个Configuration对象，此时Hibernate会默认加载classpath中的配置文件hibernate.properties，代码如下。

Configuration config = new Configuration();

（2）由于在配置文件中缺少相应的配置映射文件的信息，所以此处需要通过编码方式加载，这可以通过Configuration对象的addClass()方法实现，具体代码如下。

config.addClass(BranchForm.class);

addClass()方法用于加载实体类。

（3）完成配置文件和映射文件的加载后，将得到一个包括所有Hibernate运行期参数的Configuration实例，通过Configuration实例的buildSessionFactory()方法可以构建一个惟一的SessionFactory，代码如下。

SessionFactory sessionFactory = config.buildSessionFactory();

36. spring框架的7个模块是什么？ [ 难]

答: (1) spring AOP --面象切面编程

(2)spring DAO --数据访问对象

(3)spring ORM --对象关系影射

(4)spring Contect -- 上下文配置，向Spring框架提供上下文信息

(5)spring WEB - -WEB上下文模块

(6)\spring WEB-MVC --实现了MVC

(7)spring CORE –核心容器提供Spring框架基本功能

37. 什么是AOP 请祥述 [中]

答: 是面向切面编程

AOP 把软件系统分为两个部分：核心关注点和横切关注点。所谓的核心关注点，是业务处理的主要流程，也就是说这个解决方案要做的事。所谓横切关注点，是与核心业务无关的部分，它把常发生在核心关注点的多处，而各处基本相似，如日志，事务，权限等 。

38. Hinbernate和EJB的区别 [中]

答：Hibernate可以用在任何JDBC使用的场合，例如Java应用程序的数据库访问代码，DAO接口的实现类，甚至可以是BMP里面的访问数据库的代码。从这个意义上来说，Hibernate和EB不是一个范畴的东西，也不存在非此即彼的关系。

39. 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 [难]

答：表对应类

记录对应表的对象

表的字段对应类的属性

40. 事务隔离级别是由谁实现的？ [难]

答：数据库系统‘

41. 什么是IOC [难]

答：IOC 是控制反转，实现了对象之间的依赖关系的转移成而使程序的菘耦合

42. 在Spring中给属性有几种赋值方式 请祥叙 [难]

答：有四种 分别是

(1)普通属性赋值

（2）集合属性赋值

（3）Properties赋值

(4)Map 属性赋值

43. 在Spring说说Bean的alias元素和name属性有什么区别 [难]

答: <bean>元素name属性可以一次为<bean>定义多个别名

例：<bean id=”beanName” name=”alias\_1,alias\_2”/>

<alias>元素一次只能定义一个别名

例：<alias name=”beanName” alias=”toName”/>

注意: alias元素name属性是bean实例名

44. Bean 的作用域用几种，请祥叙 [难]

答: 用5种

分别是:

1. singleton IOC容器只会创建一个Bean的唯一的实例

2. prototype IOC容器在每次请求该Bean的时候都创建一个新的实例

3. request 在一次Http请求中 IOC容器会返回该Bean的同一个实例，而对于不同的用户请求，则会返回不同的实例

4. session 在一次Http请求Session 中 IOC容器会返回该Bean的同一个实例，而对于不同的用户Session，则会返回不同的实例

5. global session在一个全局Http请求Session 中

45. 简述什么是ORM [中]

答：ORM的全称是Object-Relational Mapping 翻译成中文就是“对象－关系映射”

ORM组件的主要功能就是实现实体域对象的持久化并封装数据库访问的细节

ORM本身并不是一个组件，它是具用某种功能的组件的总称，也可以说是一种框

架结构

46.：struts中的几个关键对象的作用(说说几个关键对象的作用) [中]

struts中的几个关键对象:Action Global(设置语言靠它了) ModuleConfig(获取mapping)，

47．Action的作用 [中]

Action的作用是接受用户的请求，通过调用业务方法实现业务处理的功能。

48. 在通常情况下软件系统由表示层，业务层，持久层和数据库层组成，Struts属于哪一层？ [难]

Struts属于表示层组件，它的作用主要体现在以下几个方面：

1) 输出用户界面和接收用户的输入，实现与用户的交互。

2) 调用业务方法，完成业务处理，还要包括处理后的显示工作。

48：Struts标签库由哪些组件组成？ [难]

标签库是组合在一起的一组JSP自定义标签。

标签库由以下组件组成：

1) 标签处理程序

2) 标签库描述符(TLD)文件

3) 应用程序部署描述符(web.xml)文件

4) JSP页面中标签库的声明

49. 用<bean:write>读出一个URL和一个名称， [难]

用<html:link>的话怎么组合起来。

即要达到这样的效果

<a href="<bean:write name="data" property="url">">

<bean:write name="data" property="tilte">

</a>

<html:link page="<bean:write name="data" property="url">">

<bean:write name="data" property="title"/>

</html:link> 会报出属性data无值的错误!(page="<bean:write name="data" property="url">"中的data无值。）

50：怎样才能配置＜html:button＞的资源文件？ [难]

在资源文件 ApplicationResourses.properties 中加入 label.login=login

在jsp页面写：

<html:button name="btn\_login"><bean:message key="label.login"/></html:button>

这样显示页面时，button上就会显示label.login的对应内容“login”

51. 说说struts框架,的方法的工作原理或流程 [易]

答: 对于采用Struts框架的web应用， 在web应用启动时会加载并初始化ActionServlet，ActionServlet从struts-config.xml中读取配置信息，

把它们存放到各种配置对象中，例如把Action的映射信息存放在ActionMapping对象中。

当ActionServlet接收到客户请求时，执行以下流程：

1.检索和用户请求匹配的ActionMapping实例，如果不存在，就返回用户请求路径无效信息；

2.如果ActionForm实例不存在，就创建一个ActionForm对象并在其中保存客户提交的表单内容；

3.根据配置信息决定是否调用ActionForm的validate()方法；

4.如果ActionForm的validate()方法返回null或返回一个不包含ActionMessage的ActionErrors对象，就表示表单验证成功；

5.ActionServlet根据ActionMapping实例包含的映射信息将请求转发给Action（如果Action实例不存在，就先创建Action实例），然后调用Action的excute()方法；

6.Action的excute()方法返回一个ActionForward对象，ActionServlet再把客户请求转发给ActionForward对象指向的JSP组件；

7.ActionForward对象指向的JSP组件生成动态网页，返回给客户。

52： strust的Action是不是线程安全的？ [难]

答：线程安全就是你可以在多线程环境下使用它，而不需要你来对它进行特殊的处理。action都是继承至servlet的，由于servlet就是线程不安全的（指多个线程共享一个servlet对象，所以），所以不要再action中定义类变量和实例变量，否则其他线程改变了这些值，可本线程还在使用

53 ：MVC，分析一下struts是如何实现MVC的 [难]

从MVC角度来看看struts的体系结构（Model 2）与工作原理：

　　1）模型（Model）

　　在Struts的体系结构中，模型分为两个部分：系统的内部状态和可以改变状态的操作（事务逻辑）。内部状态通常由一组ActinForm Bean表示。根据设计或应用程序复杂度的不同，这些Bean可以是自包含的并具有持续的状态，或只在需要时才获得数据（从某个数据库）。大型应用程序通常在方法内部封装事务逻辑（操作），这些方法可以被拥有状态信息的bean调用。比如购物车bean，它拥有用户购买商品的信息，可能还有checkOut()方法用来检查用户的信用卡，并向仓库发定货信息。 小型程序中，操作可能会被内嵌在Action类，它是struts框架中控制器角色的一部分。当逻辑简单时这个方法很适合。 建议用户将事务逻辑（要做什么）与Action类所扮演的角色（决定做什么）分开。

2）视图（View）

　　视图主要由JSP建立，struts包含扩展自定义标签库（TagLib），可以简化创建完全国际化用户界面的过程。目前的标签库包括：Bean Tags、HTML tags、Logic Tags、Nested Tags 以及Template Tags等。

　3）控制器（Controller）

　　在struts中，基本的控制器组件是ActionServlet类中的实例servelt，实际使用的servlet在配置文件中由一组映射（由ActionMapping类进行描述）进行定义。对于业务逻辑的操作则主要由Action、ActionMapping、ActionForward这几个组件协调完成的，其中Action扮演了真正的业务逻辑的实现者，ActionMapping与ActionForward则指定了不同业务逻辑或流程的运行方向。struts-config.xml 文件配置控制器。

54 ：简述什么是Struts [中]

Struts只是一个MVC框架（Framework）,用于快速开发Java Web应用。Struts实现的重点在C(Controller)，包括ActionServlet/RequestProcessor和我们定制的Action,也为V(View)提供了一系列定制标签（Custom Tag）。但Struts几乎没有涉及M(Model),所以Struts可以采用JAVA实现的任何形式的商业逻辑。

Spring是一个轻型容器(light-weight container)，其核心是Bean工厂(Bean Factory)，用以构造我们所需要的M(Model)。在此基础之上，Spring提供了AOP（Aspect-Oriented Programming, 面向层面的编程）的实现，用它来提供非管理环境下申明方式的事务、安全等服务；对Bean工厂的扩展ApplicationContext更加方便我们实现J2EE的应用；DAO/ORM的实现方便我们进行数据库的开发；Web MVC和Spring Web提供了Java Web应用的框架或与其他流行的Web框架进行集成。

就是说可将两者一起使用，达到将两者自身的特点进行互补。

55 ：Struts有哪些主要功能：[难]

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

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

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

Struts项目的目标是为创建Java web应用提供一个开放源代码的framework。Struts framework的内核是基于例如Java Servlets, JavaBeans, ResourceBundles, 和 XML，以及各种 Jakarta Commons包的标准技术的灵活的控制层。

Struts提供了它自身的控制器组件，并整合了其他技术，以提供模型和视图。对于模型，同大多数的第三方软件包一样，如Hibernate, iBATIS, 或者 Object Relational Bridge，Struts能够和标准数据连接技术相结合，如JDBC和EJB。对于视图，Struts与JavaServer Pages协同工作，包含JSTL和JSF。

56 ：Stuts框架中控制器组件的类主要有哪些？ [难]

ActionServlet ,

RequestProcessor

, Action,

ActionMapping,

ActionForward

41：Validator的组成与作用 [难]

Validator框架主要包括以下几个部分：

1) 实现各种验证规则的Java类

2) 配置文件

3) 资源文件

4) JSP自定义标签

Validator组件可以很好地解决用户输入数据的验证问题，但它并不是一个独立运行的组件，它可以被嵌入到目前大部分的Web应用开发框架中。Validator组件的验证方式有多种，既可以通过JavaScript脚本实现用户输入数据的页面验证，也可以实现在后台处理程序中的Java验证

57. ActionForm的作用？[难]

ActionForm属于一种数据传输对象，联系了前台页面与后台的Action方法，实现了前台与后台之间的数据转换和传递。它的作用主要体现在以下几个方面：

1) 在显示页面的时候用于完成页面中各种控件的初始化工作。

2) 在用户提交请求的时候，ActionForm又代表了用户所提交的数据，供Action以及后续的业务处理方法使用

ActionForm还有另外一个作用就是对用户提交数据的合法性进行验证

Java 基础就业题库

1. 类和对象的区别? [易]

类是对象的抽象,是模型概念,而对象是实实在在存在的事物,是现实中存在的实体

2. Java类库中八个标准包分别是什么? [易]

java.lang 提供常用的类、接口、一般异常、系统等编程语言的核心内容。

java.util 包含日期、日历、向量、堆栈等实用工具。

java.io 包含输入输出流类、文件类等与输入输出I/O有关的类。

java.awt 包含窗口和屏幕元素类，事件处理接口等与图形用户界面有关的内容。

java.applet 提供为编写applet小程序所需要的类。

java.text 提供与文本有关的类。

java.net 包含url类等与网络传输有关的东西。

java.sql 提供与数据库应用相关的类和接口。

3. 接口和抽象类有什么区别? [中]

接口是公开的，不能包含私有的方法或变量，而抽象类是可以有私有方法或私有变量的，

实现接口的一定要实现接口里定义的所有方法，而实现抽象类可以有选择地重写需要用到的方法，

接口可以实现多重继承，而一个类只能继承一个超类，但可以通过继承多个接口实现多重继承，

接口还有标识（里面没有任何方法，如Remote接口）和数据共享（里面的变量全是常量）的作用.

一般的应用里，最顶级的是接口，然后是抽象类实现接口，最后才到具体类实现。

4. 说说java中的内存分配? [难]

Java把内存分成两种，一种叫做栈内存，一种叫做堆内存

在函数中定义的一些基本类型的变量和对象的引用变量都是在函数的栈内存中分配。当在一段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分配的内存空间，该内存空间可以立刻被另作他用。

堆内存用于存放由new创建的对象和数组。在堆中分配的内存，由java虚拟机自动垃圾回收器来管理。在堆中产生了一个数组或者对象后，还可以在栈中定义一个特殊的变量，这个变量的取值等于数组或者对象在堆内存中的首地址，在栈中的这个特殊的变量就变成了数组或者对象的引用变量，以后就可以在程序中使用栈内存中的引用变量来访问堆中的数组或者对象，引用变量相当于为数组或者对象起的一个别名，或者代号。

引用变量是普通变量，定义时在栈中分配内存，引用变量在程序运行到作用域外释放。而数组＆对象本身在堆中分配，即使程序运行到使用new产生数组和对象的语句所在地代码块之外，数组和对象本身占用的堆内存也不会被释放，数组和对象在没有引用变量指向它的时候，才变成垃圾，不能再被使用，但是仍然占着内存，在随后的一个不确定的时间被垃圾回收器释放掉。这个也是java比较占内存的主要原因。但是在写程序的时候，可以人为的控制

5. Character类有哪些常用方法? [难]

charValue()

返回字符对象对应的值。

digit(char, int)

以指定基数返回字符 ch 对应的数值。

equals(Object)

比较该对象和指定对象。

forDigit(int, int)

确定以指定基数指定的数对应的字符。

getNumericValue(char)

返回此字符对应的 Unicode 的非负整型值。

getType(char)

返回一个表示字符种类的值。

hashCode()

返回此字符对应的哈希码。

isDefined(char)

判定一个字符在 Unicode 中是否有定义。

isDigit(char)

判定指定字符是否为数字。

isIdentifierIgnorable(char)

判定指定字符在 Java 标识符中或 Unicode 标识符中是否应看作是一个可忽略字符。

isISOControl(char)

判定指定字符是否为 ISO 控制字符。

isJavaIdentifierPart(char)

判定指定字符是否为 Java 标识符中除首字符外的字符。

isJavaIdentifierStart(char)

判定指定字符是否可作为 Java 标识符的首字符。

isJavaLetter(char)

判定指定字符是否为 Java 字母，若是，它可作为 Java 语言中一个标识符的首字符。 不推荐使用该方法。

isJavaLetterOrDigit(char)

判定指定字符是否为 Java 字母或数字，若是，它可作为 Java 语言中的一个标识符除首字符外的字符。 不推荐使用该方法。

isLetter(char)

判定指定字符是否为字母。

isLetterOrDigit(char)

判定指定字符是否为字母或数字。

isLowerCase(char)

判定指定字符是否为小写字符。

isSpace(char)

判定指定字符是否为 ISO-LATIN-1 空格。 不推荐使用该方法。

isSpaceChar(char)

判定指定字符是否为 Unicode 空白字符。

isTitleCase(char)

判定指定字符是否为标题字符。

isUnicodeIdentifierPart(char)

判定指定字符是否为 Unicode 标识符中除首字符外的字符。

isUnicodeIdentifierStart(char)

判定指定字符是否可作为 Unicode 标识符首字符。

isUpperCase(char)

判定指定字符是否为大写字符。

isWhitespace(char)

据 Java 语言，判定指定字符是否为空格。

toLowerCase(char)

将给定字符映射为对应的小写字符，若没有对应的小写字符，返回此字符本身。

toString()

返回表示此字符值的串对象。

toTitleCase(char)

将字符参数转换为标题字符。

toUpperCase(char)

将字符参数转换为大写字符。

5.Boolean类有哪些方法?

booleanValue()

返回 Boolean 对象对应的布尔值。

equals(Object)

当且仅当参数非空，且是包含与此对象相同的布尔值的布尔对象时，返回 true。

getBoolean(String)

当且仅当以参数命名的系统属性存在，且等于 "true" 时，返回为 true。

hashCode()

返回此布尔值对应的哈希码。

toString()

返回表示当前布尔值的一个串对象。

valueOf(String)

返回表示指定串的布尔值。

6. String s = new String("xyz");创建了几个String Object? [中]

两个对象，一个是“xyx”,一个是指向“xyx”的引用对象s。

7. String与StringBuffer有什么区别? [难]

从表面看来String类只用一个加号（+）便完成了字符串的拼接，而StringBuffer类却要调用一个append()方法，是否实现起来更简洁，更单纯呢？其实不然，让我们了解一下程序运行内部发生了哪些事情：

经编译后程序的bytecode（字节码）展示出了实质： 在用String类对象直接拼接时，JVM会创建一个临时的StringBuffer类对象，并调用其append()方法完成字符串的拼接，这是因为String类是不可变的，拼接操作不得不使用StringBuffer类（并且－－JVM会将"You are nice."和"I love you so much."创建为两个新的String对象）。之后，再将这个临时StringBuffer对象转型为一个String，代价不菲！可见，在这一个简单的一次拼接过程中，我们让程序创建了四个对象：两个待拼接的String，一个临时StringBuffer，和最后将StringBuffer转型成为的String－－它当然不是最初的str了，这个引用的名称没变，但它指向了新的String对象。

而如果直接使用StringBuffer类，程序将只产生两个对象：最初的StringBuffer和拼接时的String（"I love you so much."），也不再需要创建临时的StringBuffer类对象而后还得将其转换回String对象。

short s1 = 1; s1 = s1 + 1;有什么错?

short s1 = 1; s1 = s1 + 1; （s1+1运算结果是int型，需要强制转换类型）

short s1 = 1; s1 += 1;有什么错?

short s1 = 1; s1 += 1;（可以正确编译）

8. 是否可以继承String类? [难]

String类是final类故不可以继承。

9. throw与throws有什么区别? [中]

throws是用来声明一个方法可能抛出的所有异常信息

throw则是指抛出的一个具体的异常类型。

通常在一个方法（类）的声明处通过throws声明方法（类）可能抛出的异常信息，而在方法（类）内部通过throw声明一个具体的异常信息。

throws通常不用显示的捕获异常，可由系统自动将所有捕获的异常信息抛给上级方法；

throw则需要用户自己捕获相关的异常，而后在对其进行相关包装，最后在将包装后的异常信息抛出。

throws语句

throws用来标明一个成员函数可能抛出的各种"异常"。对大多数Exception子类来说，Java编译器会强迫你声明在一个成员函数中抛出的"异常"的类型。如果"异常"的类型是Error或RuntimeException，或它们的子类，这个规则不起作用，因为这copy;在程序 的正常部分中是不期待出现的。如果你想明确地抛出一个RuntimeException，你必须用throws语句来声明它的类型。这就重新定义了成员函数的定义语法：type method-name(arg-list) throws exception-list { }下面是一段程序，它抛出了一个"异常"， 但既没有捕捉它，也没有用throws来声明。这在编译时将不会通过。

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?

会执行，在return前执行。

10. 常见的runtime exception有哪些?。 [中]

ArithmeticException,

ArrayStoreException,

BufferOverflowException,

BufferUnderflowException,

CannotRedoException, CannotUndoException,

ClassCastException,

CMMException,

ConcurrentModificationException,

DOMException,

EmptyStackException, IllegalArgumentException,

IllegalMonitorStateException,

IllegalPathStateException,

IllegalStateException,

ImagingOpException, IndexOutOfBoundsException,

MissingResourceException,

NegativeArraySizeException,

NoSuchElementException, NullPointerException,

ProfileDataException,

ProviderException,

RasterFormatException,

SecurityException,

SystemException, UndeclaredThrowableException,

UnmodifiableSetException,

UnsupportedOperationException

NullPointException

11. 介绍JAVA中的Collection FrameWork(包括如何写自己的数据结构)? [难]

答：Collection FrameWork如下：

Collection

├List

│├LinkedList

│├ArrayList

│└Vector

│　└Stack

└Set

Map

├Hashtable

├HashMap

└WeakHashMap

Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素（Elements）

Map提供key到value的映射

12. ArrayList和Vector的区别,HashMap和Hashtable的区别 [难]

答：就ArrayList与Vector主要从二方面来说.

一.同步性:Vector是线程安全的，也就是说是同步的，而ArrayList是线程序不安全的，不是同步的

二.数据增长:当需要增长时,Vector 默认增长为原来一培，而ArrayList却是原来的一半

就HashMap与HashTable主要从三方面来说。

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

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

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

13. Math.round(11.5)等於多少? Math.round(-11.5)等於多少? [难]

Math.round(11.5)==12

Math.round(-11.5)==-11

round方法返回与参数最接近的长整数.

14. swtich是否能作用在byte上，是否能作用在long上，是否能作用在String上? [难]

switch（expr1）中，expr1是一个整数表达式。应该是 int、 short、 char byte。

long,string 都不能作用于swtich。

15. GC是什么? 为什么要有GC? [难]

GC是垃圾收集的意思（Gabage Collection）

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。

16. 什么叫方法重载 [中]

方法重载就是一个同名方法，有多种不同的签名

说白了就是一个同名方法可以传入不同个数或类型的参数

之间可以互相调用

17. 数组有没有length()这个方法? String有没有length()这个方法？ [中]

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

18. 什么是抽象类抽象类 [中]

仅提供一个类型的部分实现。抽象类可以有实例变量，以及一个或多个构造函数。抽象类可以同时有抽象方法和具体方法。一个抽象类不会有实例，这些构造函数不能被客户端调用来创建实例。一个抽象类的构造函数可以被其子类调用，从而使一个抽象类的所有子类都可以有一些共同的实现，而不同的子类可以在此基础上有其自己的实现。

19． 抽象类的用途　　［中］

1） 具体类不是用来继承的。 Scott Meyers曾指出，只要有可能，不要丛具体类继承。2） 假设有2个具体类，类A和类B，类B是类A 的子类，那么一个最简单的修改方案是应当建立一个抽象类（或java接口）C，然后让类A和类B成为抽象类C的子类。3） 抽象类应当拥有尽可能多的共同代码。以提高代码的复用率。4） 抽象类应当拥有尽可能少的数据。

20 .java中接口有什么用? [难]

java不允许多重继承，也就是说一个子类只能有一个父类，Son extends FatherA,FatherB 是错误的

为了弥补这点不足，java允许实现多个接口，Son extends FatherA implements AnotherFatherA,AnotherFatherB是允许的

接口中的方法没有实体，就这一点而言就相当于abstact class，如：

interface ainter{

void dosth(int i);

}

ainter是一个接口，它仅仅声明了一个方法dosth，而没有具体实现它

class aclass implements ainter

{

public void dosth(int i) {

//在这里你可以不作任何处理，但是必须实现方法体

}

}

aclass实现了ainter这个接口，因此它必须实现ainter中的方法dosth

21. 什么叫方法重载 [易]

方法重载就是一个同名方法，有多种不同的签名

说白了就是一个同名方法可以传入不同个数或类型的参数

之间可以互相调用

22. 垃圾回收机制的原理？ [难]

在JAVA中 ,JAVA VM 每隔一段时间就会查看每一块由NEW分配的内存空间,看指向它的有效引用是否存在,如果这个引用不存在,系统会自动将这块空间归入空闲内存区.这个过程被称为 垃圾收集.

23.. 什么叫面向对象? [中]

面向对象是一种新兴的程序设计方法,或者是一种新的程序设计规范(paradigm),其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。从现实世界中客观存在的事物（即对象）出发来构造软件系统，并且在系统构造中尽可能运用人类的自然思维方式。开发一个软件是为了解决某些问题，这些问题所涉及的业务范围称作该软件的问题域。其应用领域不仅仅是软件，还有计算机体系结构和人工智能等。

24. 面向对象的涉及方法有哪些? [中]

OOA－Object Oriented Analysis 　　　　面向对象的分析

　　OOD－Object Oriented Design 　　　　　面向对象的设计

OOI－Object Oriented Implementation 　面向对象的实现

25. jvm工作原理 [难]

运行jvm 字符码的工作是由解释器来完成的。解释执行过程分三步进行：

代码的装入、代码的校验、和代码的执行。

装入代码的工作由“类装载器class loader”完成。类装载器负责装入运

行一个程序需要的所有代码，这也包括程序代码中的类所继承的类和被调

用的类。当类装载器装入一个类时，该类被放在自己的名字空间中。除了

通过符号引用自己名字空间以外的类，类之间没有其他办法可以影响其他

类。在本台计算机的所有类都在同一地址空间中，而所有从外部引进的类

，都有一个自己独立的名字空间。这使得本地类通过共享相同的名字空间

获得较高的运行效率，同时又保证它们与从外部引进的类不会相互影响。

当装入了运行程序需要的所有类后，解释器便可确定整个可执行程序的内

存布局。解释器为符号引用与特定的地址空间建立对应关系及查询表。通

过在这一阶段确定代码的内布局，java很好地解决了由超类改变而使子类

崩溃的问题，同时也防止了代码的非法访问。

随后，被装入的代码由字节码校验器进行检查。校验器可以发现操作数栈

益处、非法数据类型转化等多种错误。通过校验后，代码便开始执行了。

java字节码的执行有两种方式：

1） 即时编译方式：解释器先将字节编译成机器码，然后再执行该机器码

。

2）解释执行方式：解释器通过每次解释并执行一小段代码来完成java字节

码程序的所有操作。

26. .java中输入输出流有哪些相关的类? [难]

Input和Output

1. stream代表的是任何有能力产出数据的数据源，或是任何有能力接收数据的接收源。在Java的IO中，所有的stream（包括Input和Out stream）都包括两种类型：

1.1 以字节为导向的stream

以字节为导向的stream，表示以字节为单位从stream中读取或往stream中写入信息。以字节为导向的stream包括下面几种类型：

1) input stream：

1) ByteArrayInputStream：把内存中的一个缓冲区作为InputStream使用

2) StringBufferInputStream：把一个String对象作为InputStream

3) FileInputStream：把一个文件作为InputStream，实现对文件的读取操作

4) PipedInputStream：实现了pipe的概念，主要在线程中使用

5) SequenceInputStream：把多个InputStream合并为一个InputStream

2) Out stream

1) ByteArrayOutputStream：把信息存入内存中的一个缓冲区中

2) FileOutputStream：把信息存入文件中

3) PipedOutputStream：实现了pipe的概念，主要在线程中使用

4) SequenceOutputStream：把多个OutStream合并为一个OutStream

1.2 以Unicode字符为导向的stream

以Unicode字符为导向的stream，表示以Unicode字符为单位从stream中读取或往stream中写入信息。以Unicode字符为导向的stream包括下面几种类型：

1) Input Stream

1) CharArrayReader：与ByteArrayInputStream对应

2) StringReader：与StringBufferInputStream对应

3) FileReader：与FileInputStream对应

4) PipedReader：与PipedInputStream对应

2) Out Stream

1) CharArrayWrite：与ByteArrayOutputStream对应

2) StringWrite：无与之对应的以字节为导向的stream

3) FileWrite：与FileOutputStream对应

4) PipedWrite：与PipedOutputStream对应

以字符为导向的stream基本上对有与之相对应的以字节为导向的stream。两个对应类实现的功能相同，字是在操作时的导向不同。

如CharArrayReader：和ByteArrayInputStream的作用都是把内存中的一个缓冲区作为InputStream使用，所不同的是前者每次从内存中读取一个字节的信息，而后者每次从内存中读取一个字符。

1.3 两种不现导向的stream之间的转换

InputStreamReader和OutputStreamReader：把一个以字节为导向的stream转换成一个以字符为导向的stream。

2. stream添加属性

2.1 “为stream添加属性”的作用

运用上面介绍的Java中操作IO的API，我们就可完成我们想完成的任何操作了。但通过FilterInputStream和FilterOutStream的子类，我们可以为stream添加属性。下面以一个例子来说明这种功能的作用。

如果我们要往一个文件中写入数据，我们可以这样操作：

FileOutStream fs = new FileOutStream(“test.txt”);

然后就可以通过产生的fs对象调用write()函数来往test.txt文件中写入数据了。但是，如果我们想实现“先把要写入文件的数据先缓存到内存中，再把缓存中的数据写入文件中”的功能时，上面的API就没有一个能满足我们的需求了。但是通过FilterInputStream和FilterOutStream的子类，为FileOutStream添加我们所需要的功能。

2.2 FilterInputStream的各种类型

2.2.1 用于封装以字节为导向的InputStream

1) DataInputStream：从stream中读取基本类型（int、char等）数据。

2) BufferedInputStream：使用缓冲区

3) LineNumberInputStream：会记录input stream内的行数，然后可以调用getLineNumber()和setLineNumber(int)

4) PushbackInputStream：很少用到，一般用于编译器开发

2.2.2 用于封装以字符为导向的InputStream

1) 没有与DataInputStream对应的类。除非在要使用readLine()时改用BufferedReader，否则使用DataInputStream

2) BufferedReader：与BufferedInputStream对应

3) LineNumberReader：与LineNumberInputStream对应

4) PushBackReader：与PushbackInputStream对应

2.3 FilterOutStream的各种类型

2.2.3 用于封装以字节为导向的OutputStream

1) DataIOutStream：往stream中输出基本类型（int、char等）数据。

2) BufferedOutStream：使用缓冲区

3) PrintStream：产生格式化输出

2.2.4 用于封装以字符为导向的OutputStream

1) BufferedWrite：与对应

2) PrintWrite：与对应

3. RandomAccessFile

1) 可通过RandomAccessFile对象完成对文件的读写操作

2) 在产生一个对象时，可指明要打开的文件的性质：r，只读；w，只写；rw可读写

3) 可以直接跳到文件中指定的位置

4. I/O应用的一个例子

import java.io.\*;

public class TestIO{

public static void main(String[] args)

throws IOException{

//1.以行为单位从一个文件读取数据

BufferedReader in =

new BufferedReader(

new FileReader("F:\\nepalon\\TestIO.java"));

String s, s2 = new String();

while((s = in.readLine()) != null)

s2 += s + "\n";

in.close();

//1b. 接收键盘的输入

BufferedReader stdin =

new BufferedReader(

new InputStreamReader(System.in));

System.out.println("Enter a line:");

System.out.println(stdin.readLine());

//2. 从一个String对象中读取数据

StringReader in2 = new StringReader(s2);

int c;

while((c = in2.read()) != -1)

System.out.println((char)c);

in2.close();

//3. 从内存取出格式化输入

try{

DataInputStream in3 =

new DataInputStream(

new ByteArrayInputStream(s2.getBytes()));

while(true)

System.out.println((char)in3.readByte());

}

catch(EOFException e){

System.out.println("End of stream");

}

//4. 输出到文件

try{

BufferedReader in4 =

new BufferedReader(

new StringReader(s2));

PrintWriter out1 =

new PrintWriter(

new BufferedWriter(

new FileWriter("F:\\nepalon\\ TestIO.out")));

int lineCount = 1;

while((s = in4.readLine()) != null)

out1.println(lineCount++ + "：" + s);

out1.close();

in4.close();

}

atch(EOFException ex){

System.out.println("End of stream");

}

//5. 数据的存储和恢复

try{

DataOutputStream out2 =

new DataOutputStream(

new BufferedOutputStream(

new FileOutputStream("F:\\nepalon\\ Data.txt")));

out2.writeDouble(3.1415926);

out2.writeChars("\nThas was pi:writeChars\n");

out2.writeBytes("Thas was pi:writeByte\n");

out2.close();

DataInputStream in5 =

new DataInputStream(

new BufferedInputStream(

new FileInputStream("F:\\nepalon\\ Data.txt")));

BufferedReader in5br =

new BufferedReader(

new InputStreamReader(in5));

System.out.println(in5.readDouble());

System.out.println(in5br.readLine());

System.out.println(in5br.readLine());

}

catch(EOFException e){

System.out.println("End of stream");

}

27. 构造器Constructor是否可被(覆盖)override? [难]

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

28. JAVA反射机制作用是什么? [难]

可以于运行时加载、探知、使用编译期间完全未知的

classes。换句话说，Java程序可以加载一个运行时才得知名称

的class，获悉其完整构造（但不包括methods定义），并生成其

对象实体、或对其fields设值、或唤起其methods1。

29. web应用程序体系结构是怎样的? [难]

一般分为表示层、业务层、数据存取层

30. .GET和POST有什么区别？ [难]

GET是明码传递，POST是暗码传递

31. HTTP协议的特点？ [中]

使用端口发送和接受消息 。端口是协议发送和接收数据的信道或机制 ，80是默认端口

通过这种协议传递数据服务器不会保存连接信息，因此又称为连接信息。

32. 如何通过RandomAccessFile操作文件 [难]

RandomAccessFile rf =

new RandomAccessFile("F:\\nepalon\\ rtest.dat", "rw");

for(int i=0; i<10; i++)

rf.writeDouble(i\*1.414);

rf.close();

rf = new RandomAccessFile("F:\\nepalon\\ rtest.dat", "r");

for(int i=0; i<10; i++)

System.out.println("Value " + i + "：" + rf.readDouble());

rf.close();

rf = new RandomAccessFile("F:\\nepalon\\ rtest.dat", "rw");

rf.seek(5\*8);

rf.writeDouble(47.0001);

rf.close();

rf = new RandomAccessFile("F:\\nepalon\\ rtest.dat", "r");

for(int i=0; i<10; i++)

System.out.println("Value " + i + "：" + rf.readDouble());

rf.close();

}

}

32. 静态方法有什么好处? [难]

（1）在Java里，可以定义一个不需要创建对象的方法，这种方法就是静态方法。要实现这样的效果，只需要在类中定义的方法前加上static关键字。例如：public static int maximum(int n1,int n2)

使用类的静态方法时，注意：

a） 在静态方法里只能直接调用同类中其他的静态成员（包括变量和方法），而不能直接访问类中的非静态成员。这是因为，对于非静态的方法和变量，需要先创建类的实例对象后才可使用，而静态方法在使用前不用创建任何对象。

b） 静态方法不能以任何方式引用this和super关键字，因为静态方法在使用前不用创建任何实例对象，当静态方法调用时，this所引用的对象根本没有产生。

（2）静态变量是属于整个类的变量而不是属于某个对象的。注意不能把任何方法体内的变量声明为静态，例如：

fun()

{

static int i=0;//非法。

}

（3）一个类可以使用不包含在任何方法体中的静态代码块，当类被载入时，静态代码块被执行，且之被执行一次，静态块常用来执行类属性的初始化。例如：

static

{

}

33. 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? [中]

Set里的元素是不能重复的，那么用iterator()方法来区分重复与否。equals()是判断两个对象是否相等。

34. JAVA中的静态方法有什么理解? [难]

在Java里，可以定义一个不需要创建对象的方法，这种方法就是静态方法。要实现这样的效果，只需要在类中定义的方法前加上static关键字。例如：public static int maximum(int n1,int n2)

使用类的静态方法时，注意：

a） 在静态方法里只能直接调用同类中其他的静态成员（包括变量和方法），而不能直接访问类中的非静态成员。这是因为，对于非静态的方法和变量，需要先创建类的实例对象后才可使用，而静态方法在使用前不用创建任何对象。

b） 静态方法不能以任何方式引用this和super关键字，因为静态方法在使用前不用创建任何实例对象，当静态方法调用时，this所引用的对象根本没有产生。

（2）静态变量是属于整个类的变量而不是属于某个对象的。注意不能把任何方法体内的变量声明为静态，例如：

fun()

{

static int i=0;//非法。

}

（3）一个类可以使用不包含在任何方法体中的静态代码块，当类被载入时，静态代码块被执行，且之被执行一次，静态块常用来执行类属性的初始化。例如：

static

{

}

35. JSP中动态INCLUDE与静态INCLUDE的区别？　［难］

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。

静态INCLUDE用include伪码实现,定不会检查所含文件的变化，适用于包含静态页面<%@ include file="included.htm" %>

36. Java有没有goto?

java中的保留字，现在没有在java中使用。

37. 启动一个线程是用run()还是start()? [难]

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

38. swtich是否能作用在byte上，是否能作用在long上，是否能作用在String上? [易]

switch（expr1）中，expr1是一个整数表达式。应该是 int、 short、 char byte。

long,string 都不能作用于swtich。

39. 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? [难]

会执行，在return前执行。

40. 两个对象值相同(x.equals(y) == true)，但却可有不同的hash code，这句话对不对?

不对，有相同的hash code。 [易]

\

41. char型变量中能不能存贮一个中文汉字?为什么? [难]

可以，因为java中以unicode编码，一个char占16个字节，所以放一个中文是没问题的

42. Java中的线程有四种状态分别是是什么？ [难]

运行、就绪、挂起、结束。

43. 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？ [难]

字节流，字符流。

字节流继承于InputStream \ OutputStream，

字符流继承于InputStreamReader \ OutputStreamWriter。

44. 静态变量和实例变量的区别？ [中]

static i = 10; //常量

class A a; a.i =10;//可变

45. 什么是java序列化，如何实现java序列化？ [难]

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

序列化的实现：

将需要被序列化的类实现Serializable接口，该接口没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化

46. 是否可以从一个static方法内部发出对非static方法的调用？ [难]

不可以,如果其中包含对象的method()；不能保证对象初始化.

47. 在JAVA中，如何跳出当前的多重嵌套循环？ [难]

用break; return 方法。

48. 面向对象的特征有哪些方面 [中]

1.抽象：

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

2.继承：

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

3.封装：

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

4. 多态性：

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

49. String是最基本的数据类型吗? [中]

基本数据类型包括byte、int、char、long、float、double、boolean和short。

java.lang.String类是final类型的，因此不可以继承这个类、不能修改这个类。为了提高效率节省空间，我们应该用StringBuffer类

50. int 和 Integer 有什么区别 [中]

Java 提供两种不同的类型：引用类型和原始类型（或内置类型）。Int是java的原始数据类型，Integer是java为int提供的封装类。Java为每个原始类型提供了封装类。

51. 运行时异常与一般异常有何异同？ [ 难]

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

52. &和&&的区别。 [ 中]

&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与（and）。

53. final, finally, finalize的区别。 [中]

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。

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。

Jsp/servlet面试题

1. 四种会话跟踪技术是什么？(jsp) (中)

答：cookie,url重写,session,隐藏表单域。

2.不同客户端可不可以共享Session?不可以 (易)

Jps和Servlet之间可不可以用session传值?可以

3.简介cookie的有关知识 (中)

浏览器与WEB服务器之间是使用HTTP协议进行通信的，当某个用户发出页面请求时，WEB服务器只是简单的进行响应，然后就关闭与该用户的连接。因此当一个请求发送到WEB服务器时，无论其是否是第一次来访，服务器都会把它当作第一次来对待，这样的不好之处可想而知。为了弥补这个缺陷，Netscape 开发出了cookie这个有效的工具来保存某个用户的识别信息，因此人们昵称为“小甜饼”。cookies是一种WEB服务器通过浏览器在访问者的硬盘上存储信息的手段：Netscape Navigator使用一个名为cookies.txt本地文件保存从所有站点接收的Cookie信息；而IE浏览器把Cookie信息保存在类似于C: \windows\cookies的目录下。当用户再次访问某个站点时，服务端将要求浏览器查找并返回先前发送的Cookie信息，来识别这个用户。

　　cookies给网站和用户带来的好处非常多：

1、Cookie能使站点跟踪特定访问者的访问次数、最后访问时间和访问者进入站点的路径

2、Cookie能告诉在线广告商广告被点击的次数 ，从而可以更精确的投放广告

3、Cookie有效期限未到时，Cookie能使用户在不键入密码和用户名的情况下进入曾经浏览过的一些站点

4、Cookie能帮助站点统计用户个人资料以实现各种各样的个性化服务

JSP是使用如下的语法格式来创建cookie的：

　　Cookie cookie\_name =new Cookie(""Parameter"",""Value"");

　　例如：Cookie newCookie =new Cookie(""username"",""zheng""); response.addCookie(newCookie);

4.在浏览器的打开时再新打开一个浏览器还是同一个session吗? (中)

对IE而言不是同一个SESSION

5.描述Cookie和Session的作用，区别和各自的应用范围 (中)

Cookie和Session都可以用来在多个页面之间共享数据,区别是Cookie保存在客户端,可以设置比较长的保存时间.而Session保存在服务器端,通常生存时间较短。如果客户端禁用了Cookie,Cookie将无法工作，而session不受这一影响。一般来说保密性高、保存时间短的信息适合用session来存放，而Cookie适合存放需要长期保存的非敏感数据。

6.如何实现购物车添加新项,又不使购物车中原有项目被取消? (易)

可以将购物车存放在session当中

7.Cookie的过期时间如何设置? (易)

使用setMaxAge(int exp)方法

8.如果BROWSER已关闭了cookies,在JSP中我如何打开session? (中)

如果Cookie被禁用，session将通过url重写的方式来传送，所以不会影响session的使用

9.include指令和include动作的区别 (中)

10. Jsp有哪些常用动作?作用分别是什么?(jsp) (中)

答：JSP有6种常用基本动作 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。

11. "forward与redirect区别? (中)

Forward是在服务器端进行分发,分发后地址栏无变化,用户看不到分发后的地址.而redirect是在客户端进行的页面重定向,地址栏上会显示重定向后的网页.forward的页面只能是在同一个web应用程序之内,而重定向可以定位到外部资源.forward后还是同一个request请求,而重定向后以前的request请求就不存在了.

12.Jsp和Servlet中的请求转发分别如何实现? (易)

Jsp中使用<jsp:forward>动作,servlet中使用RequestDistpatcher对象的forward方法"

13.什么是web容器？(易)

答：给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使JSP,SERVLET直接更容器中的环境变量接口交互，不必关注其它系统问题。主要有WEB服务器来实现。例如：TOMCAT,WEBLOGIC,WEBSPHERE等。该容器提供的接口严格遵守J2EE规范中的WEB APPLICATION 标准。我们把遵守以上标准的WEB服务器就叫做J2EE中的WEB容器。

14.应用服务器有那些？ (中)

答：BEA WebLogic Server，IBM WebSphere Application Server，Oracle9i Application Server，JBoss，Tomcat。"

15.请说出JSP的内置对象及方法 (中)

答：request表示HttpServletRequest对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取cookie,

header, 和session数据的有用的方法。

response表示HttpServletResponse对象，并提供了几个用于设置送回

浏览器的响应的方法（如cookies,头信息等）

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

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

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

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

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

page表示从该页面产生的一个servlet实例。

16.使用JSP如何获得客户浏览器的信息? (易)

request对象的getXXXX方法"

17.B/S与C/S的联系与区别。 (中)

答：C/S是Client/Server的缩写。服务器通常采用高性能的PC、工作站或小型机，并采用大型数据库系统，如Oracle、Sybase、Informix或

SQL Server。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（Browser），如Netscape

Navigator或Internet Explorer，服务器安装Oracle、Sybase、Informix或 SQL

Server等数据库。在这种结构下，用户界面完全通过WWW浏览器实现，一部分事务逻辑在前端实现，但是主要事务逻辑在服务器端实现。浏览器通过Ｗeb

Server 同数据库进行数据交互。

C/S 与 B/S 区别：

１）．硬件环境不同:

　　C/S 一般建立在专用的网络上, 小范围里的网络环境,

局域网之间再通过专门服务器提供连接和数据交换服务.

　　B/S 建立在广域网之上的, 不必是专门的网络硬件环境,例与电话上网, 租用设备.

信息自己管理. 有比C/S更强的适应范围, 一般只要有操作系统和浏览器就行

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.

一般高度机密的信息系统采用C/S 结构适宜. 可以通过B/S发布部分可公开信息.

　　B/S 建立在广域网之上, 对安全的控制能力相对弱, 可能面向不可知的用户。

３）．对程序架构不同

　　C/S 程序可以更加注重流程, 可以对权限多层次校验,

对系统运行速度可以较少考虑.

　　B/S 对安全以及访问速度的多重的考虑, 建立在需要更加优化的基础之上.

比C/S有更高的要求 B/S结构的程序架构是发展的趋势, 从MS的.Net系列的BizTalk

2000 Exchange 2000等, 全面支持网络的构件搭建的系统. SUN 和IBM推的JavaBean

构件技术等,使 B/S更加成熟.

４）．软件重用不同

　　C/S 程序可以不可避免的整体性考虑,

构件的</ div <>

《redis》

（1）什么是redis?

Redis 是一个基于内存的高性能key-value数据库。 (有空再补充，有理解错误或不足欢迎指正)

（2）Reids的特点

Redis本质上是一个Key-Value类型的内存数据库，很像memcached，整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据flush到硬盘上进行保存。因为是纯内存操作，Redis的性能非常出色，每秒可以处理超过 10万次读写操作，是已知性能最快的Key-Value DB。

Redis的出色之处不仅仅是性能，Redis最大的魅力是支持保存多种数据结构，此外单个value的最大限制是1GB，不像 memcached只能保存1MB的数据，因此Redis可以用来实现很多有用的功能，比方说用他的List来做FIFO双向链表，实现一个轻量级的高性 能消息队列服务，用他的Set可以做高性能的tag系统等等。另外Redis也可以对存入的Key-Value设置expire时间，因此也可以被当作一 个功能加强版的memcached来用。

Redis的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，因此Redis适合的场景主要局限在较小数据量的高性能操作和运算上。

（3）Redis支持的数据类型

Redis通过Key-Value的单值不同类型来区分, 以下是支持的类型:

Strings

Lists

Sets 求交集、并集

Sorted Set

hashes

（4）为什么redis需要把所有数据放到内存中？

Redis为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，redis将会越来越受欢迎。

如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

（5）Redis是单进程单线程的

redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销

（6）虚拟内存

当你的key很小而value很大时,使用VM的效果会比较好.因为这样节约的内存比较大.

当你的key不小时,可以考虑使用一些非常方法将很大的key变成很大的value,比如你可以考虑将key,value组合成一个新的value.

vm-max-threads这个参数,可以设置访问swap文件的线程数,设置最好不要超过机器的核数,如果设置为0,那么所有对swap文件的操作都是串行的.可能会造成比较长时间的延迟,但是对数据完整性有很好的保证.

自己测试的时候发现用虚拟内存性能也不错。如果数据量很大，可以考虑分布式或者其他数据库

（7）分布式

redis支持主从的模式。原则：Master会将数据同步到slave，而slave不会将数据同步到master。Slave启动时会连接master来同步数据。

这是一个典型的分布式读写分离模型。我们可以利用master来插入数据，slave提供检索服务。这样可以有效减少单个机器的并发访问数量

（8）读写分离模型

通过增加Slave DB的数量，读的性能可以线性增长。为了避免Master DB的单点故障，集群一般都会采用两台Master DB做双机热备，所以整个集群的读和写的可用性都非常高。

读写分离架构的缺陷在于，不管是Master还是Slave，每个节点都必须保存完整的数据，如果在数据量很大的情况下，集群的扩展能力还是受限于单个节点的存储能力，而且对于Write-intensive类型的应用，读写分离架构并不适合。

（9）数据分片模型

为了解决读写分离模型的缺陷，可以将数据分片模型应用进来。

可以将每个节点看成都是独立的master，然后通过业务实现数据分片。

结合上面两种模型，可以将每个master设计成由一个master和多个slave组成的模型。

（10）Redis的回收策略

volatile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰

no-enviction（驱逐）：禁止驱逐数据

1. 使用Redis有哪些好处？

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

2. redis相比memcached有哪些优势？

(1) memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

(2) redis的速度比memcached快很多

(3) redis可以持久化其数据

3. redis常见性能问题和解决方案：

(1) Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件

(2) 如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次

(3) 为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内

(4) 尽量避免在压力很大的主库上增加从库

(5) 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变。

4. MySQL里有2000w数据，redis中只存20w的数据，如何保证redis中的数据都是热点数据

相关知识：redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。redis 提供 6种数据淘汰策略：

voltile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰

no-enviction（驱逐）：禁止驱逐数据

5. Memcache与Redis的区别都有哪些？

1)、存储方式

Memecache把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。

Redis有部份存在硬盘上，这样能保证数据的持久性。

2)、数据支持类型

Memcache对数据类型支持相对简单。

Redis有复杂的数据类型。

3)、使用底层模型不同

它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。

Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

4），value大小

redis最大可以达到1GB，而memcache只有1MB

6. Redis 常见的性能问题都有哪些？如何解决？

1).Master写内存快照，save命令调度rdbSave函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以Master最好不要写内存快照。

2).Master AOF持久化，如果不重写AOF文件，这个持久化方式对性能的影响是最小的，但是AOF文件会不断增大，AOF文件过大会影响Master重启的恢复速度。Master最好不要做任何持久化工作，包括内存快照和AOF日志文件，特别是不要启用内存快照做持久化,如果数据比较关键，某个Slave开启AOF备份数据，策略为每秒同步一次。

3).Master调用BGREWRITEAOF重写AOF文件，AOF在重写的时候会占大量的CPU和内存资源，导致服务load过高，出现短暂服务暂停现象。

4). Redis主从复制的性能问题，为了主从复制的速度和连接的稳定性，Slave和Master最好在同一个局域网内

7, redis 最适合的场景

Redis最适合所有数据in-momory的场景，虽然Redis也提供持久化功能，但实际更多的是一个disk-backed的功能，跟传统意义上的持久化有比较大的差别，那么可能大家就会有疑问，似乎Redis更像一个加强版的Memcached，那么何时使用Memcached,何时使用Redis呢?

如果简单地比较Redis与Memcached的区别，大多数都会得到以下观点：

1 、Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。

2 、Redis支持数据的备份，即master-slave模式的数据备份。

3 、Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。

（1）、会话缓存（Session Cache）

最常用的一种使用Redis的情景是会话缓存（session cache）。用Redis缓存会话比其他存储（如Memcached）的优势在于：Redis提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗？

幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用Redis来缓存会话的文档。甚至广为人知的商业平台Magento也提供Redis的插件。

（2）、全页缓存（FPC）

除基本的会话token之外，Redis还提供很简便的FPC平台。回到一致性问题，即使重启了Redis实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似PHP本地FPC。

再次以Magento为例，Magento提供一个插件来使用Redis作为全页缓存后端。

此外，对WordPress的用户来说，Pantheon有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

（3）、队列

Reids在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得Redis能作为一个很好的消息队列平台来使用。Redis作为队列使用的操作，就类似于本地程序语言（如Python）对 list 的 push/pop 操作。

如果你快速的在Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用Redis创建非常好的后端工具，以满足各种队列需求。例如，Celery有一个后台就是使用Redis作为broker，你可以从这里去查看。

（4），排行榜/计数器

Redis在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis只是正好提供了这两种数据结构。所以，我们要从排序集合中获取到排名最靠前的10个用户–我们称之为“user\_scores”，我们只需要像下面一样执行即可：

当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行：

ZRANGE user\_scores 0 10 WITHSCORES

Agora Games就是一个很好的例子，用Ruby实现的，它的排行榜就是使用Redis来存储数据的，你可以在这里看到。

（5）、发布/订阅

最后（但肯定不是最不重要的）是Redis的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用Redis的发布/订阅功能来建立聊天系统！（不，这是真的，你可以去核实）。

Redis提供的所有特性中，我感觉这个是喜欢的人最少的一个，虽然它为用户提供如果此多功能。

1. 使用Redis有哪些好处？

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

2. redis相比memcached有哪些优势？

(1) memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

(2) redis的速度比memcached快很多

(3) redis可以持久化其数据

3. redis常见性能问题和解决方案：

(1) Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件

(2) 如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次

(3) 为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内

(4) 尽量避免在压力很大的主库上增加从库

(5) 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变。

1. 使用Redis有哪些好处？

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

2. redis相比memcached有哪些优势？

(1) memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

(2) redis的速度比memcached快很多

(3) redis可以持久化其数据

3. redis常见性能问题和解决方案：

(1) Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件

(2) 如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次

(3) 为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内

(4) 尽量避免在压力很大的主库上增加从库

(5) 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变。

4. MySQL里有2000w数据，redis中只存20w的数据，如何保证redis中的数据都是热点数据

相关知识：redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。redis 提供 6种数据淘汰策略：

voltile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰

no-enviction（驱逐）：禁止驱逐数据

5. Memcache与Redis的区别都有哪些？

1)、存储方式

Memecache把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。

Redis有部份存在硬盘上，这样能保证数据的持久性。

2)、数据支持类型

Memcache对数据类型支持相对简单。

Redis有复杂的数据类型。

3)、使用底层模型不同

它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。

Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

4），value大小

redis最大可以达到1GB，而memcache只有1MB

6. Redis 常见的性能问题都有哪些？如何解决？

1).Master写内存快照，save命令调度rdbSave函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以Master最好不要写内存快照。

2).Master AOF持久化，如果不重写AOF文件，这个持久化方式对性能的影响是最小的，但是AOF文件会不断增大，AOF文件过大会影响Master重启的恢复速度。Master最好不要做任何持久化工作，包括内存快照和AOF日志文件，特别是不要启用内存快照做持久化,如果数据比较关键，某个Slave开启AOF备份数据，策略为每秒同步一次。

3).Master调用BGREWRITEAOF重写AOF文件，AOF在重写的时候会占大量的CPU和内存资源，导致服务load过高，出现短暂服务暂停现象。

4). Redis主从复制的性能问题，为了主从复制的速度和连接的稳定性，Slave和Master最好在同一个局域网内

7, redis 最适合的场景

Redis最适合所有数据in-momory的场景，虽然Redis也提供持久化功能，但实际更多的是一个disk-backed的功能，跟传统意义上的持久化有比较大的差别，那么可能大家就会有疑问，似乎Redis更像一个加强版的Memcached，那么何时使用Memcached,何时使用Redis呢?

如果简单地比较Redis与Memcached的区别，大多数都会得到以下观点：

1 、Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。

2 、Redis支持数据的备份，即master-slave模式的数据备份。

3 、Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。

（1）、会话缓存（Session Cache）

最常用的一种使用Redis的情景是会话缓存（session cache）。用Redis缓存会话比其他存储（如Memcached）的优势在于：Redis提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗？

幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用Redis来缓存会话的文档。甚至广为人知的商业平台Magento也提供Redis的插件。

（2）、全页缓存（FPC）

除基本的会话token之外，Redis还提供很简便的FPC平台。回到一致性问题，即使重启了Redis实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似PHP本地FPC。

再次以Magento为例，Magento提供一个插件来使用Redis作为全页缓存后端。

此外，对WordPress的用户来说，Pantheon有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

（3）、队列

Reids在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得Redis能作为一个很好的消息队列平台来使用。Redis作为队列使用的操作，就类似于本地程序语言（如Python）对 list 的 push/pop 操作。

如果你快速的在Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用Redis创建非常好的后端工具，以满足各种队列需求。例如，Celery有一个后台就是使用Redis作为broker，你可以从这里去查看。

（4），排行榜/计数器

Redis在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis只是正好提供了这两种数据结构。所以，我们要从排序集合中获取到排名最靠前的10个用户–我们称之为“user\_scores”，我们只需要像下面一样执行即可：

当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行：

ZRANGE user\_scores 0 10 WITHSCORES

Agora Games就是一个很好的例子，用Ruby实现的，它的排行榜就是使用Redis来存储数据的，你可以在这里看到。

（5）、发布/订阅

最后（但肯定不是最不重要的）是Redis的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用Redis的发布/订阅功能来建立聊天系统！（不，这是真的，你可以去核实）。

Redis提供的所有特性中，我感觉这个是喜欢的人最少的一个，虽然它为用户提供如果此多功能。

目录

Spring概述

依赖注入

Spring Beans

Spring注解

Spring的对象访问

Spring面向切面编程

Spring MVC框架

Spring概述

1.什么是Spring？

Spring是一个开源的Java EE开发框架。Spring框架的核心功能可以应用在任何Java应用程序中，但对Java EE平台上的Web应用程序有更好的扩展性。Spring框架的目标是使得Java EE应用程序的开发更加简捷，通过使用POJO为基础的编程模型促进良好的编程风格。

2.Spring有哪些优点？

轻量级：Spring在大小和透明性方面绝对属于轻量级的，基础版本的Spring框架大约只有2MB。

控制反转(IOC)：Spring使用控制反转技术实现了松耦合。依赖被注入到对象，而不是创建或寻找依赖对象。

面向切面编程(AOP)： Spring支持面向切面编程，同时把应用的业务逻辑与系统的服务分离开来。

容器：Spring包含并管理应用程序对象的配置及生命周期。

MVC框架：Spring的web框架是一个设计优良的web MVC框架，很好的取代了一些web框架。

事务管理：Spring对下至本地业务上至全局业务(JAT)提供了统一的事务管理接口。

异常处理：Spring提供一个方便的API将特定技术的异常(由JDBC, Hibernate, 或JDO抛出)转化为一致的、Unchecked异常。

3.Spring框架有哪些模块？

Spring框架的基本模块如下所示：

Core module

Bean module

Context module

Expression Language module

JDBC module

ORM module

OXM module

Java Messaging Service(JMS) module

Transaction module

Web module

Web-Servlet module

Web-Struts module

Web-Portlet module

4.解释核心容器(应用上下文)模块

这是Spring的基本模块，它提供了Spring框架的基本功能。BeanFactory 是所有Spring应用的核心。Spring框架是建立在这个模块之上的，这也使得Spring成为一个容器。

5.BeanFactory – BeanFactory 实例

BeanFactory是工厂模式的一种实现，它使用控制反转将应用的配置和依赖与实际的应用代码分离开来。

最常用的BeanFactory实现是XmlBeanFactory类。

6.XmlBeanFactory

最常用的就是org.springframework.beans.factory.xml.XmlBeanFactory，它根据XML文件中定义的内容加载beans。该容器从XML文件中读取配置元数据，并用它来创建一个完备的系统或应用。

7.解释AOP模块

AOP模块用来开发Spring应用程序中具有切面性质的部分。该模块的大部分服务由AOP Aliance提供，这就保证了Spring框架和其他AOP框架之间的互操作性。另外，该模块将元数据编程引入到了Spring。

8.解释抽象JDBC和DAO模块

通过使用抽象JDBC和DAO模块保证了与数据库连接代码的整洁与简单，同时避免了由于未能关闭数据库资源引起的问题。它在多种数据库服务器的错误信息之上提供了一个很重要的异常层。它还利用Spring的AOP模块为Spring应用程序中的对象提供事务管理服务。

9.解释对象/关系映射集成模块

Spring通过提供ORM模块在JDBC的基础上支持对象关系映射工具。这样的支持使得Spring可以集成主流的ORM框架，包括Hibernate, JDO, 及iBATIS SQL Maps。Spring的事务管理可以同时支持以上某种框架和JDBC。

10.解释web模块

Spring的web模块建立在应用上下文(application context)模块之上，提供了一个适合基于web应用程序的上下文环境。该模块还支持了几个面向web的任务，如透明的处理多文件上传请求及将请求参数同业务对象绑定起来。

11.解释Spring MVC模块

Spring提供MVC框架构建web应用程序。Spring可以很轻松的同其他MVC框架结合，但Spring的MVC是个更好的选择，因为它通过控制反转将控制逻辑和业务对象完全分离开来。

12.Spring的配置文件

Spring的配置文件是一个XML文件，文件包含了类信息并描述了这些类是如何配置和互相调用的。

13.Spring IoC容器是什么？

Spring IOC负责创建对象、管理对象(通过依赖注入)、整合对象、配置对象以及管理这些对象的生命周期。

14.IOC有什么优点？

IOC或依赖注入减少了应用程序的代码量。它使得应用程序的测试很简单，因为在单元测试中不再需要单例或JNDI查找机制。简单的实现以及较少的干扰机制使得松耦合得以实现。IOC容器支持勤性单例及延迟加载服务。

15.应用上下文是如何实现的？

FileSystemXmlApplicationContext 容器加载XML文件中beans的定义。XML Bean配置文件的完整路径必须传递给构造器。

FileSystemXmlApplicationContext 容器也加载XML文件中beans的定义。注意，你需要正确的设置CLASSPATH，因为该容器会在CLASSPATH中查看bean的XML配置文件。

WebXmlApplicationContext：该容器加载xml文件，这些文件定义了web应用中所有的beans。

16.Bean Factory和ApplicationContext有什么区别？

ApplicationContext提供了一种解决文档信息的方法，一种加载文件资源的方式(如图片)，他们可以向监听他们的beans发送消息。另外，容器或者容器中beans的操作，这些必须以bean工厂的编程方式处理的操作可以在应用上下文中以声明的方式处理。应用上下文实现了MessageSource，该接口用于获取本地消息，实际的实现是可选的。

17.Spring应用程序看起来像什么？

一个定义功能的接口

实现包括属性，setter和getter方法，功能等

Spring AOP

Spring的XML配置文件

使用该功能的客户端编程

依赖注入

18.Spring中的依赖注入是什么？

依赖注入作为控制反转(IOC)的一个层面，可以有多种解释方式。在这个概念中，你不用创建对象而只需要描述如何创建它们。你不必通过代码直接的将组件和服务连接在一起，而是通过配置文件说明哪些组件需要什么服务。之后IOC容器负责衔接。

19.有哪些不同类型的IOC(依赖注入)？

构造器依赖注入：构造器依赖注入在容器触发构造器的时候完成，该构造器有一系列的参数，每个参数代表注入的对象。

Setter方法依赖注入：首先容器会触发一个无参构造函数或无参静态工厂方法实例化对象，之后容器调用bean中的setter方法完成Setter方法依赖注入。

20.你推荐哪种依赖注入？构造器依赖注入还是Setter方法依赖注入？

你可以同时使用两种方式的依赖注入，最好的选择是使用构造器参数实现强制依赖注入，使用setter方法实现可选的依赖关系。

Spring Beans

21.什么是Spring Beans？

Spring Beans是构成Spring应用核心的Java对象。这些对象由Spring IOC容器实例化、组装、管理。这些对象通过容器中配置的元数据创建，例如，使用XML文件中定义的创建。

在Spring中创建的beans都是单例的beans。在bean标签中有一个属性为”singleton”,如果设为true，该bean是单例的，如果设为false，该bean是原型bean。Singleton属性默认设置为true。因此，spring框架中所有的bean都默认为单例bean。

22.Spring Bean中定义了什么内容？

Spring Bean中定义了所有的配置元数据，这些配置信息告知容器如何创建它，它的生命周期是什么以及它的依赖关系。

23.如何向Spring 容器提供配置元数据？

有三种方式向Spring 容器提供元数据:

XML配置文件

基于注解配置

基于Java的配置

24.你如何定义bean的作用域？

在Spring中创建一个bean的时候，我们可以声明它的作用域。只需要在bean定义的时候通过’scope’属性定义即可。例如，当Spring需要产生每次一个新的bean实例时，应该声明bean的scope属性为prototype。如果每次你希望Spring返回一个实例，应该声明bean的scope属性为singleton。

25.说一下Spring中支持的bean作用域

Spring框架支持如下五种不同的作用域：

singleton：在Spring IOC容器中仅存在一个Bean实例，Bean以单实例的方式存在。

prototype：一个bean可以定义多个实例。

request：每次HTTP请求都会创建一个新的Bean。该作用域仅适用于WebApplicationContext环境。

session：一个HTTP Session定义一个Bean。该作用域仅适用于WebApplicationContext环境.

globalSession：同一个全局HTTP Session定义一个Bean。该作用域同样仅适用于WebApplicationContext环境.

bean默认的scope属性是’singleton‘。

26.Spring框架中单例beans是线程安全的吗？

不是，Spring框架中的单例beans不是线程安全的。

27.解释Spring框架中bean的生命周期

Spring容器读取XML文件中bean的定义并实例化bean。

Spring根据bean的定义设置属性值。

如果该Bean实现了BeanNameAware接口，Spring将bean的id传递给setBeanName()方法。

如果该Bean实现了BeanFactoryAware接口，Spring将beanfactory传递给setBeanFactory()方法。

如果任何bean BeanPostProcessors 和该bean相关，Spring调用postProcessBeforeInitialization()方法。

如果该Bean实现了InitializingBean接口，调用Bean中的afterPropertiesSet方法。如果bean有初始化函数声明，调用相应的初始化方法。

如果任何bean BeanPostProcessors 和该bean相关，调用postProcessAfterInitialization()方法。

如果该bean实现了DisposableBean，调用destroy()方法。

28.哪些是最重要的bean生命周期方法？能重写它们吗？

有两个重要的bean生命周期方法。第一个是setup方法，该方法在容器加载bean的时候被调用。第二个是teardown方法，该方法在bean从容器中移除的时候调用。

bean标签有两个重要的属性(init-method 和 destroy-method)，你可以通过这两个属性定义自己的初始化方法和析构方法。Spring也有相应的注解：@PostConstruct 和 @PreDestroy。

29.什么是Spring的内部bean？

当一个bean被用作另一个bean的属性时，这个bean可以被声明为内部bean。在基于XML的配置元数据中，可以通过把元素定义在 或元素内部实现定义内部bean。内部bean总是匿名的并且它们的scope总是prototype。

30.如何在Spring中注入Java集合类？

Spring提供如下几种类型的集合配置元素：

list元素用来注入一系列的值，允许有相同的值。

set元素用来注入一些列的值，不允许有相同的值。

map用来注入一组”键-值”对，键、值可以是任何类型的。

props也可以用来注入一组”键-值”对，这里的键、值都字符串类型。

31.什么是bean wiring？

Wiring，或者说bean Wiring是指beans在Spring容器中结合在一起的情况。当装配bean的时候，Spring容器需要知道需要哪些beans以及如何使用依赖注入将它们结合起来。

32.什么是bean自动装配？

Spring容器可以自动配置相互协作beans之间的关联关系。这意味着Spring可以自动配置一个bean和其他协作bean之间的关系，通过检查BeanFactory 的内容里没有使用和< property>元素。

33.解释自动装配的各种模式？

自动装配提供五种不同的模式供Spring容器用来自动装配beans之间的依赖注入:

no：默认的方式是不进行自动装配，通过手工设置ref 属性来进行装配bean。

byName：通过参数名自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byName。之后容器试图匹配、装配和该bean的属性具有相同名字的bean。

byType：通过参数的数据类型自动自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byType。之后容器试图匹配和装配和该bean的属性类型一样的bean。如果有多个bean符合条件，则抛出错误。

constructor：这个同byType类似，不过是应用于构造函数的参数。如果在BeanFactory中不是恰好有一个bean与构造函数参数相同类型，则抛出一个严重的错误。

autodetect：如果有默认的构造方法，通过 construct的方式自动装配，否则使用 byType的方式自动装配。

34.自动装配有哪些局限性？

自动装配有如下局限性：

重写：你仍然需要使用 和< property>设置指明依赖，这意味着总要重写自动装配。

原生数据类型:你不能自动装配简单的属性，如原生类型、字符串和类。

模糊特性：自动装配总是没有自定义装配精确，因此，如果可能尽量使用自定义装配。

35.你可以在Spring中注入null或空字符串吗？

完全可以。

Spring注解

36.什么是Spring基于Java的配置？给出一些注解的例子

基于Java的配置允许你使用Java的注解进行Spring的大部分配置而非通过传统的XML文件配置。

以注解@Configuration为例，它用来标记类，说明作为beans的定义，可以被Spring IOC容器使用。另一个例子是@Bean注解，它表示该方法定义的Bean要被注册进Spring应用上下文中。

37.什么是基于注解的容器配置?

另外一种替代XML配置的方式为基于注解的配置，这种方式通过字节元数据装配组件而非使用尖括号声明。开发人员将直接在类中进行配置，通过注解标记相关的类、方法或字段声明，而不再使用XML描述bean之间的连线关系。

38.如何开启注解装配？

注解装配默认情况下在Spring容器中是不开启的。如果想要开启基于注解的装配只需在Spring配置文件中配置元素即可。

39.@Required 注解

@Required表明bean的属性必须在配置时设置，可以在bean的定义中明确指定也可通过自动装配设置。如果bean的属性未设置，则抛出BeanInitializationException异常。

40.@Autowired 注解

@Autowired 注解提供更加精细的控制，包括自动装配在何处完成以及如何完成。它可以像@Required一样自动装配setter方法、构造器、属性或者具有任意名称和/或多个参数的PN方法。

41. @Qualifier 注解

当有多个相同类型的bean而只有其中的一个需要自动装配时，将@Qualifier 注解和@Autowire 注解结合使用消除这种混淆，指明需要装配的bean。

Spring数据访问

42.在Spring框架中如何更有效的使用JDBC？

使用Spring JDBC框架，资源管理以及错误处理的代价都会减轻。开发人员只需通过statements和queries语句从数据库中存取数据。Spring框架中通过使用模板类能更有效的使用JDBC，也就是所谓的JdbcTemplate(例子)。

43.JdbcTemplate

JdbcTemplate类提供了许多方法，为我们与数据库的交互提供了便利。例如，它可以将数据库的数据转化为原生类型或对象，执行写好的或可调用的数据库操作语句，提供自定义的数据库错误处理功能。

44.Spring对DAO的支持

Spring对数据访问对象(DAO)的支持旨在使它可以与数据访问技术(如 JDBC, Hibernate 及JDO)方便的结合起来工作。这使得我们可以很容易在的不同的持久层技术间切换，编码时也无需担心会抛出特定技术的异常。

45.使用Spring可以通过什么方式访问Hibernate？

使用Spring有两种方式访问Hibernate：

使用Hibernate Template的反转控制以及回调方法

继承HibernateDAOSupport，并申请一个AOP拦截器节点

46.Spring支持的ORM

Spring支持一下ORM：

Hibernate

iBatis

JPA (Java -Persistence API)

TopLink

JDO (Java Data Objects)

OJB

47.如何通过HibernateDaoSupport将Spring和Hibernate结合起来？

使用Spring的SessionFactory 调用LocalSessionFactory。结合过程分为以下三步：

配置Hibernate SessionFactory

继承HibernateDaoSupport实现一个DAO

使用AOP装载事务支持

48.Spring支持的事务管理类型

Spring支持如下两种方式的事务管理：

编程式事务管理：这意味着你可以通过编程的方式管理事务，这种方式带来了很大的灵活性，但很难维护。

声明式事务管理：这种方式意味着你可以将事务管理和业务代码分离。你只需要通过注解或者XML配置管理事务。

49.Spring框架的事务管理有哪些优点？

它为不同的事务API(如JTA, JDBC, Hibernate, JPA, 和JDO)提供了统一的编程模型。

它为编程式事务管理提供了一个简单的API而非一系列复杂的事务API(如JTA).

它支持声明式事务管理。

它可以和Spring 的多种数据访问技术很好的融合。

50.你更推荐那种类型的事务管理？

许多Spring框架的用户选择声明式事务管理，因为这种方式和应用程序的关联较少，因此更加符合轻量级容器的概念。声明式事务管理要优于编程式事务管理，尽管在灵活性方面它弱于编程式事务管理(这种方式允许你通过代码控制业务)。

Spring面向切面编程(AOP)

51.解释AOP

面向切面编程,或AOP允许程序员模块化横向业务逻辑，或定义核心部分的功能，例如日志管理和事务管理。

52.切面(Aspect)

AOP的核心就是切面，它将多个类的通用行为封装为可重用的模块。该模块含有一组API提供 cross-cutting功能。例如,日志模块称为日志的AOP切面。根据需求的不同，一个应用程序可以有若干切面。在Spring AOP中，切面通过带有@Aspect注解的类实现。

53.在Spring AOP中concern和 cross-cutting concern的区别是什么？

Concern(核心逻辑)：表示在应用程序中一个模块的行为。Concern可以定义为我们想要实现的功能。

Cross-cutting concern(横向的通用逻辑)：指的是整个应用程序都会用到的功能，它影响整个应用程序。例如，日志管理（Logging）、安全管理（Security）以及数据交互是应用程序的每个模块都要涉及到的，因此这些都属于Cross-cutting concern。

54.连接点(Join point)

连接点代表应用程序中插入AOP切面的地点。它实际上是Spring AOP框架在应用程序中执行动作的地点。

55.通知(Advice)

通知表示在方法执行前后需要执行的动作。实际上它是Spring AOP框架在程序执行过程中触发的一些代码。

Spring切面可以执行一下五种类型的通知:

before(前置通知)：在一个方法之前执行的通知。

after(最终通知)：当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。

after-returning(后置通知)：在某连接点正常完成后执行的通知。

after-throwing(异常通知)：在方法抛出异常退出时执行的通知。

around(环绕通知)：在方法调用前后触发的通知。

56.切入点(Pointcut)

切入点是一个或一组连接点，通知将在这些位置执行。可以通过表达式或匹配的方式指明切入点。

57.什么是引入？

引入允许我们在已有的类上添加新的方法或属性。

58.什么是目标对象？

被一个或者多个切面所通知的对象。它通常是一个代理对象。也被称做被通知（advised）对象。

59.什么是代理？

代理是将通知应用到目标对象后创建的对象。从客户端的角度看，代理对象和目标对象是一样的。

60.有几种不同类型的自动代理？

BeanNameAutoProxyCreator：bean名称自动代理创建器

DefaultAdvisorAutoProxyCreator：默认通知者自动代理创建器

Metadata autoproxying：元数据自动代理

61.什么是织入？什么是织入应用的不同点？

织入是将切面和其他应用类型或对象连接起来创建一个通知对象的过程。织入可以在编译、加载或运行时完成。

62.解释基于XML Schema方式的切面实现

在这种情况下，切面由使用XML文件配置的类实现。

63.解释基于注解方式(基于@AspectJ)的切面实现

在这种情况下(基于@AspectJ的实现)，指的是切面的对应的类使用Java 5注解的声明方式。

Spring的MVC框架

64.什么是Spring的MVC框架？

Spring提供了一个功能齐全的MVC框架用于构建Web应用程序。Spring框架可以很容易的和其他的MVC框架融合(如Struts)，该框架使用控制反转(IOC)将控制器逻辑和业务对象分离开来。它也允许以声明的方式绑定请求参数到业务对象上。

65.DispatcherServlet

Spring的MVC框架围绕DispatcherServlet来设计的，它用来处理所有的HTTP请求和响应。

66.WebApplicationContext

WebApplicationContext继承了ApplicationContext，并添加了一些web应用程序需要的功能。和普通的ApplicationContext 不同，WebApplicationContext可以用来处理主题样式，它也知道如何找到相应的servlet。

67.什么是Spring MVC框架的控制器？

控制器提供对应用程序行为的访问，通常通过服务接口实现。控制器解析用户的输入，并将其转换为一个由视图呈现给用户的模型。Spring 通过一种极其抽象的方式实现控制器，它允许用户创建多种类型的控制器。

68.@Controller annotation

@Controller注解表示该类扮演控制器的角色。Spring不需要继承任何控制器基类或应用Servlet API。

69.@RequestMapping annotation

@RequestMapping注解用于将URL映射到任何一个类或者一个特定的处理方法上。