Servlet 面试集锦

1、**Servlet如何得到服务器的信息？**

Servlet 可以使用如下四种方法来得到 server 的 name, port 和 info

```
publicString ServletRequest.getServerName()
publicString ServletRequest.getServerPort()
publicString ServletContext.getServerInfo()
publicString ServletRequest.getAttributes(String name)
```

如下代码实现用 servlet 取得 server 的信息并输出到客户端浏览器：

```
importjava.io.*;
importjava.util.*;
importjavax.servlet.*;
importjavax.servlet.http.*;
public classDemoServerSnoopextendsGenericServlet{

publicvoidservice(ServletRequest req , ServletResponse res)
throwsServletException,IOException{

res.setContentType("text/plain");
PrintWriter out= res.getWriter();
out.println("req.getServerName()"+ req.getServerName());
out.println("req.getServerPort()"+ req.getServerPort());

out.println("ServletContext().getServerInfo()"+
getServletContext().getServerInfo());

out.println("getServerInfo() name:"+
getServerInfoName(getServletContext().getServerInfo()));

out.println("getServerInfo() version:"+
getServerInfoVersion(getServletContext().getServerInfo()));

out.println("getServerContext().getAttribute(\"attribute\")"+
getServletContext().getAttribute("attribute"));
}
privateString getServerInfoName(String serverInfo){
```

```
intslash = serverInfo.indexOf('/');
if(slash==-1)
returnserverInfo;
else
return(String)serverInfo.subSequence(0,slash);
}
privateString getServerInfoVersion(String serverInfo){

intslash = serverInfo.indexOf('/');
if(slash==-1)
return null;
else
returnserverInfo.substring(slash +1);
}}
```

2、Servlet如何得到客户端机器的信息？

Servlet 可以使用 getRemoteAddr()和 getRemoteHost()来得到客户端的 IP 地址和 host, 代码如下所示：

```
publicString ServletRequest.getRemoteAddr()
publicStirng ServletRequest.getRemoteHost()
```

用这些方法来访问客户端有所限制，如下代码实现了对客户端配置进行检查并把相关消息发送到客户端的功能：

```
importjava.io.*;
importjava.util.*;
importjavax.servlet.*;
importjavax.servlet.http.*;
public classDemoExportRestrictionextendsHttpServlet{
publicvoiddoGet(HttpServletRequest req,HttpServletResponse res)
throwsServletException,IOException{

res.setContentType("text/plain");
PrintWriter out= res.getWriter();

//得到客户端的hostname
String remoteHost = req.getRemoteHost();

//查看客户端是否允许这样的操作
if(!isHostAllowed(remoteHost)){
out.println("Access <BLINK>ACCESS DENIED </BLINK>");
```

```
}else{
out.println("access granted");
}
}
privatebooleanisHostAllowed(String host) {
return(host.endsWith(".com"))||
(host.indexOf('.')==-1);//没有域名 ok
}
}
```

3、一家美资企业的 java servlet 面试题

filter 的作用是什么？主要实现什么方法？doFilter 方法里面的2个参数 request 和 response 他问这两个接口的全称是什么？

1.Filter 使用户可以改变一个 request 和修改一个 response. Filter 不是一个 servlet,它不能产生一个 response,它能够在一个 request 到达 servlet 之前预处理 request,也可以在离开 servlet 时处理 response. 换种说法,filter 其实是一个”servlet chaining”(servlet 链).一个 filter 包括:

1. 在 servlet 被调用之前截获;
2. 在 servlet 被调用之前检查 servlet request;
3. 根据需要修改 request 头和 request 数据;
4. 根据需要修改 response 头和 response 数据;
5. 在 servlet 被调用之后截获.

2. request 的全称是 HttpServletRequest response 的全称是 HttpServletResponse .

4、JDBC 操作数据库的基本流程是什么？

所有的 JDBC 应用程序都具有下面的基本流程:

1、建立到数据库的连接。

2、执行 SQL 语句。

3、处理结果。

4、从数据库断开连接。

下面我们就来仔细看一看每一个步骤。

建立到数据库的连接

通过 JDBC 使用数据库的第一步就是建立一个连接。JDBC 连接是由 URL 指定的，它的格式如下:

jdbc:<subprotocol>:<subname>

其中 subprotocol 是被请求的数据库连接的类型（如 ODBC，ORACLE，Informix 等等），而 subname 提供了所要建立的连接的一些附加信息。当 JDBC 驱动程序管理器收到一个连接的 URL 时，所有已知的 JDBC 驱动程序会被询问是否可以为这个 URL 服务。请求一个通过 JDBC-ODBC 桥到叫做 MyData 的 ODBC 数据源的连接的例子如下:

Connection con = DriverManage.getconnection(” jdbc:odbc:MyData”);

看上去一切都很顺利，但是 JDBC 驱动程序管理器是怎么知道哪些 JDBC 驱动程序在当前的系统中可用

呢？有两种机制可以通知驱动程序管理器一个 JDBC 驱动程序可以使用：sql.drivers 属性和 JDBC 驱动程序注册。

驱动程序管理器引用 sql.drivers 系统属性来取得当前系统中可用的 JDBC 驱动程序列表。这个系统属性包含一些用冒号隔开的 JDBC 驱动程序的类名，通过这个类名，驱动程序管理器可以试图满足一个连接请求。

使用驱动程序注册更为常见。这种方法使你对你要使用的 JDBC 驱动程序有更多的控制。所有的 JDBC 驱动程序在实例化的时候都必须在驱动程序管理器中注册自己，注册可以通过下列两个方法来实现：

1.Class.forName(”foo.Driver”).newInstance();

2.new foo.Driver();

我个人比较喜欢使用 Class.forName()这种方法，不过这两种方法的效果是相同的。JDBC 驱动程序用驱动程序管理器注册自己，这样，它就可以为连接请求服务了。

执行 SQL 语句

在数据库连接成功建立之后，我们就可以执行那些完成实际工作的 SQL 语句了。在执行 SQL 语句之前，我们必须首先创建一个语句对象，这个对象提供了到特定数据库 SQL 引擎的接口。有下列三种不同类型的语句对象：

1.Statement——基本的语句对象，它提供了直接在数据库中执行 SQL 语句的方法。对于那些只执行一次的查询以及 DDL 语句如 CREATE TABLE，DROP TABLE 等等来说，statement 对象就足够了。

2.Prepared statement——这种语句对象用于那些需要执行多次，每次仅仅是数据取值不同的 SQL 语句，它还提供了一些方法，以便指出语句所使用的输入参数。

3.Callable statement——这种语句对象被用来访问数据库中的存储过程。它提供了一些方法来指定语句所使用的输入输出参数。

下面是一个用语句类来执行 SQL SELECT 语句的一个例子：

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(”SELECT * FROM MyTable”);
```

处理结果

在执行了一个 SQL 语句之后，我们必须处理得到的结果。有些语句仅仅返回一个整形数，指出受到影响的行数（比如 UPDATE 和 DELETE 语句）。SQL 查询（SELECT 语句）返回一个含有查询结果的结果集。结果集由行和列组成，各列数据可以通过相应数据库类型的一系列 get 方法（如 getString，getInt，getDate 等等）来取得。在取得了一行数据的所有数据之后，我们可以调用 next()方法来移到结果集中的下一条记录。JDBC 规范的1.1版只允许 forward-onle(只向前)型的游标，而在 JDBC2.0中有更健壮的游标控制功能，我们可以向后移动游标而且可以将游标移动到指定行。

从数据库断开连接

在结果集、语句和连接对象用完以后，我们必须正确地关闭它们。连接对象、结果集对象以及所有的语句对象都有 close()方法，通过调用这个方法，我们可以确保正确释放与特定数据库系统相关的所有资源。

有些开发者喜欢将引用乱放，然后用一个垃圾收集程序专门负责正确清除对象。我强烈建议大家在使用了 JDBC 驱动程序之后调用 close()方法。这样可以尽可能的减少由于挂起的对象残留在数据库系统中而

造成的内存泄漏。

5、为什么要使用 servlet

servlet 可以很好地替代公共网关接口(Common Gateway Interface，CGI)脚本。通常 CGI 脚本是用 Perl 或者 C 语言编写的，它们总是和特定的服务器平台紧密相关。而 servlet 是用 Java 编写的，所以它们一开始就是平台无关的。这样，Java 编写一次就可以在任何平台运行(write once,run anywhere)的承诺就同样可以在服务器上实现了。servlet 还有一些 CGI 脚本所不具备的独特优点：

■servlet 是持久的。servlet 只需 Web 服务器加载一次，而且可以在不同请求之间保持服务(例如一次数据库连接)。与之相反，CGI 脚本是短暂的、瞬态的。每一次对 CGI 脚本的请求，都会使 Web 服务器加载并执行该脚本。一旦这个 CGI 脚本运行结束，它就会被从内存中清除，然后将结果返回到客户端。CGI 脚本的每一次使用，都会造成程序初始化过程(例如连接数据库)的重复执行。

■servlet 是与平台无关的。如前所述，servlet 是用 Java 编写的，它自然也继承了 Java 的平台无关性。

■servlet 是可扩展的。由于 servlet 是用 Java 编写的，它就具备了 Java 所能带来的所有优点。Java 是健壮的、面向对象的编程语言，它很容易扩展以适应你的需求。servlet 自然也具备了这些特征。

■servlet 是安全的。从外界调用一个 servlet 的惟一方法就是通过 Web 服务器。这提供了高水平的安全性保障，尤其是在你的 Web 服务器有防火墙保护的时候。

■setvlet 可以在多种多样的客户机上使用。由于 servlet 是用 Java 编写的，所以你可以很方便地在 HTML 中使用它们，就像你使用 applet 一样。在本书中您将看到这一点。

6、**什么是 servlet**

servlet 可以被认为是服务器端的 applet。servlet 被 Web 服务器加载和执行，就如同 applet 被浏览器加载和执行一样。servlet 从客户端(通过 Web 服务器)接收请求，执行某种作业，然后返回结果。

使用 servlet 的基本流程如下：

• 客户端(很可能是 Web 浏览器)通过 HTTP 提出请求。

• Web 服务器接收该请求并将其发给 servlet。如果这个 servlet 尚未被加载，Web 服务器将把它加载到 Java 虚拟机并且执行它。

• servlet 将接收该 HTTP 请求并执行某种处理。

• servlet 将向 Web 服务器返回应答。

• Web 服务器将从 servlet 收到的应答发送给客户端。

由于 servlet 是在服务器上执行，通常与 applet 相关的安全性的问题并不需实现。servlet 使相当数量的不可能或者至少是很难由 applet 实现的功能的实现成为可能。与现有系统通过 CORBA，RMI，socket 和本地(native)调用的通信就是其中的一些例子。另外，一定要注意：Web 浏览器并不直接和 servlet 通信,servlet 是由 Web 服务器加载和执行的。这意味着如果你的 Web 服务器有防火墙保护,那么你的 servlet 也将得到防火墙的保护。

7、**什么是 servlet 链？**

与 UNIX 和 DOS 命令中的管道类似，你也可以将多个 servlet 以特定顺序链接起来。在 servlet 链中，一个 servlet 的输出被当作下一个 servlet 的输入，而链中最后一个 servlet 的输出被返回到浏览器。

servlet 链接提供了将一个 servlet 的输出重定向为另一个 servlet 的输入的能力。这样，你就可以划分

工作，从而使用一系列servlet来实现它。另外，你还可以将servlet组织在一起以提供新的功能。

8、Java Web开发面试题一套

选择题(没有注明多选，则为单选)，每题2分

1.下面哪个不是Form的元素？（ D ）

A. Input　B: textarea　C: select　D: table

2. HTML页面中，下面哪个表示空格（ B ）

A.&　B. 　C.©　D.<

3.<td align=”XXX”>中的align属性是什么含义？（ C ）

A. 加粗　B　斜体　C　对齐方式　D　边框样式

4.单选按钮是下列哪一个？（　D　）

A.<input name=”sex” type=”text” value=”0” />

B. <input name=”sex” type=”checkbox” value=”0” />

C. <input name=”sex” type=”option” value=”0” />

D.<input name=”sex” type=”radio” value=”0” />

5.下边哪个不是JSP内置对象？（ C ）

A. Session　B　request　C　cook　D　out

6. 下边哪个是JSP指令标记（ C ）

A.<%……%>

B. <%!……%>

C. <%@……%>

D.<%=……%>

7. 当在JSP文件中要使用到ArrayList对象时，应在JSP文件中加入以下哪个语句？（ C)

A.<jsp:include file=”java.util.*” />

B. <jsp:include page=”java.util.*” />

C. <%@ page import=”java.util.*” />

D.<%@ page include=”java.util.*” />

8.关于JavaBean的说法，哪个是正确的？（ D ）

A. JavaBean的具体类可以不是public的

B. JavaBean可以只提供一个带参数的构造器

C. JavaBean可以象Html标记一样不关闭

D. JavaBean可以保存状态

9. JavaBean的生命周期中，哪个是用来跟踪用户会话的（　A　）

A. session

B. request

C. page

D. application

10. 要在session对象中保存属性，可以使用以下哪个语句（　B　）

A. session.getAttribute(“key”,”value”)

B. session.setAttribute(“key”,”value”)

C. session.setAttribute(“key”)

D. session.getAttribute(“key”)

11. Jsp:forward 和 sendredirect 都是用来做页面跳转的，描述错误的是？（ ）

1 forward 之后可以使用原来的 request 对象，而且效率更高。

2 sendredirect 之后不可以使用原来的 request 对象，而且效率低。

3 forward 地址栏不变化，只能在 web 应用程序内的页面间跳转。

4 forward 地址栏变化，可以跳转到任何页面和机器。

12. 关于两种请求，下列说法正确的是？（）

5 get 请求是默认的

6 get 请求处理的数据量大小不受限制

7 post 请求地址栏里是能够看到数据的

8 post 请求可以由 doGet 方法处理

13. 如果 Tomcat 安装后，想要修改它的端口号，可以通过修改<tomcat 安装目录>/conf 下的___文件来实现。（ ）

9 web.xml

10 server.xml

11 server-minimal.xml

12 tomcat-user.xml

14. JSP 分页代码中，哪个步需次序是正确的？（ ）

13 先取总记录数，得到总页数，再取所有的记录，最后显示本页的数据。

14 先取所有的记录，得到总页数，再取总记录数，最后显示本页的数据。

15 先取总记录数，得到总记录数，再取所有的记录，最后显示本页的数据。

16 先取本页的数据，得到总页数，再取总记录数，最后显示本页的数据。

15.下边哪个不是 MVC 中的组成？()

17 JavaBean

18 FrameWork

19 JSP

20 Servlet

16..客户化标签库文件 myTags.tld,应当保存在哪一个目录下（ ）

21 web 应用程序的根目录

22 自定义目录

23 WEB-INF 目录

24 Lib 目录

17. Oracle 数据库的 JDBC 驱动程序类名及其完整包路径为（ ）

25 jdbc.driver.oracle.OracleDriver

26 jdbc.oracle.driver.OracleDriver

27 driver.oracle. jdbc.OracleDriver

28 oracle.jdbc.driver.OracleDriver

18. 如果需要删除 session 中的某个属性 key,可以调用下面的____方法( )

29 remove(“key”);

30 removeAttribute(“key”)

31 invalidate()

32 logout()

19. 下面有关 JSP 和 Servlet 关系的论述正确的有: (两个正确答案) ( )

33 JSP 能够实现的功能均可由 Servlet 实现

34 Servlet 能够实现的功能均可由 JSP 实现

35 Servlet 更适做表现层

36 JSP 更适合做控制层

20. 下面那一个是正确使用 JavaBean 的方法()

A.<jsp:useBean id=”address” class=”AddressBean />

B. <jsp:useBean name=”address” class=”AddressBean />

C. <jsp:useBean bean=”address” class=”AddressBean />

D. <jsp:useBean beanName=”address” class=”AddressBean />

21. 销毁一个 session 对象,应当调用下面哪一个方法( )

A. session.invalidate()

B. session.expunge()

C. session.destroy()

D. session.end()

22. ActionForm Bean 中验证表单数据方法的返回类型是( )

A. ActionError

B. ActionErrors

C. ActionForward

D. ActionMapping

23. 以下哪个说法是正确的? (多选)( )

A. 每个 Http 请求对应一个单独的 ActionServlet 实例

B. 对于每个请求访问 HelloAction 的 Http 请求, struts 框架只创建一个单独的 HelloAction 实例

C. 每个子应用对应一个单独的 RequestProcessor 实例

D. 每个子应用对应一个单独的 web.xml 文件

24. 在 struts 应用的控制器中包含哪些组件(两个正确答案) ( )

A. JSP

B. ActionServlet

C. Action

D. 客户化标签

25. 对于以下这段配置 Actionservlet 的代码，哪些说法是正确的？(三个正确答案)

`（　　）

<servlet>

<servlet-name>action</servlet-name>

<servlet-class>org.apache.struts.action.ActionServlet</servlet-class>

<ini-param>

<param-name>config</param-name>

<param-value>/WEB-INF/myconfig.xml</param- value >

</ini-param>

<load-on-startup>0</load-on-startup>

<!—Standard Action Servlet Mapping à

<servlet-mapping>

<servlet-name>action</servlet-name>

<url-pattern>*.do</url-pattern>

</servlet-mapping>

</servlet>

A. Servlet 容器在启动应用时，会初始化这个 ActionServlet

B. 对于所有 URL 中以”*.do”结尾的 Http 请求，都由 ActionServlet 处理

C. 这段代码位于 struts-config.xml 中

D. 这段代码位于 web.xml 中

26. ActionForm 组件的存在范围是(两个正确答案)（　　）

A. application

B. session

C. request

D. page

27. 如果同一页面中有多个提交按钮(<html:submit>)，要求用一个 Action 类处理，应当从下面哪一个扩展更合适（　　）

A. Action

B. DispatchAction

C. LookupDispatchAction

D. ForwardAction

E. SwitchAction

28. 如果同一页面中有多个提交按钮(<html:submit>)，要求用一个 Action 类处理，在

struts-config.xml 文件的<action >元素中最重要的且必须配置的一个属性是(　　)

A. parameter

B. validate

C. key

D. scope

29. 如果在 struts-config.xml 配置了数据源，则在那个组件中可以获得这个数据源对象，进而获得 Connection 对象(　　)

A. Action 组件

B. 视图组件

C. 模型组件

D. JSP 组件

30. struts 的中心控制器组件的完整类名是(　　)

A. org.apache.struts.action.ActionServlet

B. org.apache.struts.action.Action

C. org.apache.struts.action.Switch

D. org.apache.struts.action.CoreController

二. 简述题(总分 40):

1. 简述 Servlet 的生命周期 (5分)

Web 容器加载 servlet，生命收起开始，通过调用 servlet 的的 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

一个 servlet 的生命周期由部署 servlet 的容器控制，当一个请求映射到一个 servlet 是，容器执行下步骤:

1.加载 servlet 类

2.创建一个 servlet 类的实例

3.调用 init 初始化 servlet 实例，

2.调用 service 方法，传递一个请求和响应对象

容器要移除一个 servlet，调用 servlet 的 destroy 方法结束该 servlet

2. 简述<jsp:forward>动作和 response.sendRedirect()的异同 (5分)

重定向分为两类。一类是客户端重定向，一类是服务器端重定向。客户端重定向可以通过设置特定的 HTTP 头，或者写 javaScirpt 脚本实现。

服务器端的重定向方式

服务器的重定向有两种方式，一种是 HttpServletResponse 的 sendRedirect（）方法，一个是使用 RequestDispatcher 的 forward（）方法。

HttpServletResponse.sendRedirect()方法

HttpServletResponse 接口定义了可用于转向的 sendRedirect 方法，这个方法将响应定向到指定的，新的

URL，location 可以是一个绝对的 URL，如 response.sendRedirect(“http://java.sun.com”)也可以使用相对的 URL location 可以是一个绝对的 URL，如 response.sendRedirect(”http://java.sun.com”)也可以使用相对的 URL。如果 location 以“/”开头，则容器认为相对于当前 Web 应用的根，否则，容器将解析为相对于当前请求的 URL。这种重定向的方法，将导致客户端浏览器的请求 URL 跳转。从浏览器中的地址栏中可以看到新的 URL 地址，作用类似于上面设置 HTTP 响应头信息的实现。

RequestDispatcher.forward()方法

RequestDispatcher 是一个 Web 资源的包装器，可以用来把当前 request 传递到该资源，或者把新的资源包括到当前响应中。RequestDispatcher 接口中定义了两个方法，参见如下代码：

```
public interface RequestDispatcher {
void forward(ServletRequest request, ServletResponse response);
void include(ServletRequest request, ServletResponse response);
}
```

forward()方法将当前的 request 和 response 重定向到该 RequestDispacher 指定的资源。这在实际项目中大量使用，因为完成一个业务操作往往需要跨越多个步骤，每一步骤完成相应的处理后，转向到下一个步骤。比如，通常业务处理在 Servlet 中处理，处理的结果转向到一个 JSP 页面进行显示。这样看起来类似于 Servlet 链的功能，但是还有一些区别。一个 RequestDispatcher 对象可以把请求发送到任意一个服务器资源，而不仅仅是另外一个 Servlet。 include()方法将把 Request Dispatcher 资源的输出包含到当前输出中。

注意，只有在尚未向客户端输出响应时才可以调用 forward()方法，如果页面缓存不为空，在重定向前将自动清除缓存。否则将抛出一个 IllegalStateException 异常。

如何得到 RequestDispatcher

有三种方法可以得到 Request Dispatcher 对象。

1.javax.servlet. ServletRequest 的 getRequestDispatcher(String path)方法，其中 path 可以是相对路径，但不能越出当前 Servlet 上下文。如果 path 以“/”开头，则解析为相对于当前上下文的根。

2.javax.servlet. ServletContext 的 getRequestDispatcher(String path)方法，其中 path 必须以“/”开头，路径相对于当前的 Servlet 上下文。可以调用 ServletContext 的 getContext(String uripath)得到另一个 Servlet 上下文，并可以转向到外部上下文的一个服务器资源链接。

3.使用 javax.servlet. ServletContext 的 getNamedDispatcher(String name)得到名为 name 的一个 Web 资源，包括 Servlet 和 JSP 页面。这个资源的名字在 Web 应用部署描述文件 web.xml 中指定。

这三种方法的使用有细微的差别。比如，下面是一个应用的配置文件 web.xml：

```
<?xml version=”1.0″ ?>
<!DOCTYPE web-app PUBLIC “-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN”
“http://java.sun.com/j2ee/dtds/web-app_2_2.dtd” >
<web-app>
<servlet>
<servlet-name>FirstServlet</servlet-name>
<servlet-class>org. javaresearch.redirecttest.ServletOne</servlet-class>
```

</servlet>

<servlet>

<servlet-name>SecondServlet</servlet-name>

<servlet-class>org.javaresearch. redirecttest.ServletTwo</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>FirstServlet</servlet-name>

<url-pattern>/servlet/firstservlet/</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>SecondServlet</servlet-name>

<url-pattern>/servlet/secondservlet/</url-pattern>

</servlet-mapping>

</web-app>

其中定义了两个Servlet,名字分别为FirstServlet和SecondServlet,对应的类分别为org.javaresearch.redirecttest.ServletOne和org. javaresearch.redirecttest.ServletTwo。可以在浏览器中通过类似于下面的链接访问:

http://localhost:8080/servlet/firstservlet/

使用1中方法，例如在firstservlet可以写入下面的代码:

RequestDispatcher rd = request.getRequestDispatcher(”secondservlet”);

rd.forward(request, response);

此时控制权将转向到第二个Servlet了。

使用2中的方法，可以从Servlet Context中得到RequestDispatcher代码如下:

RequestDispatcher rd = getServletContext().getRequest

Dispatcher(”/servlet/secondservlet”);

rd.forward(request, response);

使用3中的方法，从上面的web. xml配置文件可以看到定义了两个Servlet，名字分别为FirstServlet和SecondServlet，所以可以得到命名的Dispatcher:

RequestDispatcher rd = getServletContext().getNamedDispatcher(”SecondServlet”);

rd.forward(request, response);

这样也可以重定向到SecondServlet了。

JSP页面中的重定向

JSP在解析后编译为一个Servlet运行，所以在JSP中也可以使用上面的重定向代码，并且，JSP还提供了更便利的操作，如下:

<jsp:forward page= “nextpage.jsp”/>

JSP页面执行到这儿，将终止当前的处理，将控制权交由nextpage.jsp。

如何选择

RequestDispatcher.forward()方法和HttpServletResponse.sendRedirect()方法的区别是：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Request Dispatcher.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HttpServletResponse.sendRequest()方法。

3.。用图形画出一个web.应用程序部署在Tomcat5的基本目录结构，说明每个目录下
保存哪些文件。(8分)

4. 提交表单请求时，post和get有哪些区别? (5分)

以Get方式请求方式传输,所带参数附加在请求URL后直接传给服务器,并可以从服务器端的QUERY_STRING这个环境变量中读取；如果以POST请求方式传输则参数被打包在数据包中传给服务器

使用Post方法数据由标准的输入设备读入，使用Get方法，数据由CGI变量Query_STRING传递给表单数据处理程序，即附加在请求地址的后面，在浏览器的地址栏可看到servlet会自动将以上两种方法得到的数据进行处理，对于两种方法servlet处理方法是一样的，用户只要调用HttpServletRequest的getParameter（）方法，给出变量名称即可取出变量的值。

5.。MVC的各个部分都由那些技术来实现? (9分)

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

6. 什么叫DTO,简述基于struts的应用在模型层和视图层之间加入DTO的优点
(8分)

Data access object 数据访问对象

主要用来封装对数据库的访问，通过她可以把POJO持久化PO，用PO组装出来的VO，DTO

PO : persistent Object持久对象，可以看成与数据库中的表相映射的java对象，最简单的PO就是对应数据库中某个表的一条记录，多个记录可以用PO的集合。PO应该不包含任何对数据库的操作。

VO:value object值对象 通常用于业务层之间的数据传递，和PO一样也是仅仅包含数据而已，但应是抽象出的的业务对象，可以和表对应，也可以不

DAO: data access object数据访问对象，此对象用于访问数据库，通常与PO结合使用，DAO中包含了各种数据库的操作方法，通过它的方法，结合PO对数据库进行相关的操作

BO: business object 业务对象，封装业务逻辑的java对象，通过调用DAO方法，结合PO，VO进行业务操作

POJO: plain ordinary java object 简单屋规则java对象

9/**如何解决Java中的中文乱码问题?**

一、中文问题的来源

计算机最初的操作系统支持的编码是单字节的字符编码，于是，在计算机中一切处理程序最初都是以单字节编码的英文为准进行处理。随着计算机的发展，为了适应世界其它民族的语言（当然包括我们的汉字),

人们提出了UNICODE编码，它采用双字节编码，兼容英文字符和其它民族的双字节字符编码，所以，目前，大多数国际性的软件内部均采用UNICODE编码，在软件运行时，它获得本地支持系统（多数时间是操作系统）默认支持的编码格式，然后再将软件内部的UNICODE转化为本地系统默认支持的格式显示出来。Java的JDK和JVM即是如此，我这里说的JDK是指国际版的JDK，我们大多数程序员使用的是国际化的JDK版本，以下所有的JDK均指国际化的JDK版本。我们的汉字是双字节编码语言，为了能让计算机处理中文，我们自己制定的gb2312、GBK、GBK2K等标准以适应计算机处理的需求。所以，大部分的操作系统为了适应我们处理中文的需求，均定制有中文操作系统，它们采用的是GBK,GB2312编码格式以正确显示我们的汉字。如：中文Win2K默认采用的是GBK编码显示，在中文WIN2k中保存文件时默认采用的保存文件的编码格式也是GBK的，即，所有在中文WIN2K中保存的文件它的内部编码默认均采用GBK编码，注意：GBK是在GB2312基础上扩充来的。

由于Java语言内部采用UNICODE编码，所以在JAVA程序运行时，就存在着一个从UNICODE编码和对应的操作系统及浏览器支持的编码格式转换输入、输出的问题，这个转换过程有着一系列的步骤，如果其中任何一步出错，则显示出来的汉字就会出是乱码，这就是我们常见的JAVA中文问题。

同时，Java是一个跨平台的编程语言，也即我们编写的程序不仅能在中文windows上运行，也能在中文Linux等系统上运行，同时也要求能在英文等系统上运行（我们经常看到有人把在中文win2k上编写的JAVA程序，移植到英文Linux上运行）。这种移植操作也会带来中文问题。

还有，有人使用英文的操作系统和英文的IE等浏览器，来运行带中文字符的程序和浏览中文网页，它们本身就不支持中文，也会带来中文问题。

有，几乎所有的浏览器默认在传递参数时都是以UTF-8编码格式来传递，而不是按中文编码传递，所以，传递中文参数时也会有问题，从而带来乱码现象。

总之，以上几个方面是JAVA中的中文问题的主要来源，我们把以上原因造成的程序不能正确运行而产生的问题称作：JAVA中文问题。

2、JAVA编码转换的详细过程

我们常见的JAVA程序包括以下类别：

*直接在console上运行的类(包括可视化界面的类)

*JSP代码类（注：JSP是Servlets类的变型）

*Servelets类

*EJB类

*其它不可以直接运行的支持类

这些类文件中，都有可能含有中文字符串，并且我们常用前三类JAVA程序和用户直接交互，用于输出和输入字符，如：我们在JSP和Servlet中得到客户端送来的字符，这些字符也包括中文字符。无论这些JAVA类的作用如何，这些JAVA程序的生命周期都是这样的：

*编程人员在一定的操作系统上选择一个合适的编辑软件来实现源程序代码并以.java扩展名保存在操作系统中，例如我们在中文win2k中用记事本编辑一个java源程序；

*编程人员用JDK中的javac.exe来编译这些源代码，形成.class类(JSP文件是由容器调用JDK来编译的)；

*直接运行这些类或将这些类布署到WEB容器中去运行，并输出结果。

那么，在这些过程中，JDK 和 JVM 是如何将这些文件如何编码和解码并运行的呢？

这里，我们以中文 win2k 操作系统为例说明 JAVA 类是如何来编码和被解码的。

第一步，我们在中文 win2k 中用编辑软件如记事本编写一个 Java 源程序文件(包括以上五类 JAVA 程序)，程序文件在保存时默认采用了操作系统默认支持 GBK 编码格式(操作系统默认支持的格式为 file.encoding 格式)形成了一个.java 文件，也即，java 程序在被编译前，我们的 JAVA 源程序文件是采用操作系统默认支持的 file.encoding 编码格式保存的，java 源程序中含有中文信息字符和英文程序代码；要查看系统的 file.encoding 参数，可以用以下代码：

```
public class ShowSystemDefaultEncoding {
public static void main(String[] args) {
String encoding = System.getProperty(”file.encoding”);
System.out.println(encoding);
}}
```

第二步，我们用 JDK 的 javac.exe 文件编译我们的 Java 源程序，由于 JDK 是国际版的，在编译的时候，如果我们没有用-encoding 参数指定我们的 JAVA 源程序的编码格式，则 javac.exe 首先获得我们操作系统默认采用的编码格式，也即在编译 java 程序时，若我们不指定源程序文件的编码格式，JDK 首先获得操作系统的 file.encoding 参数(它保存的就是操作系统默认的编码格式，如 WIN2k，它的值为 GBK)，然后 JDK 就把我们的 java 源程序从 file.encoding 编码格式转化为 JAVA 内部默认的 UNICODE 格式放入内存中。然后，javac 把转换后的 unicode 格式的文件进行编译成.class 类文件，此时.class 文件是 UNICODE 编码的，它暂放在内存中，紧接着，JDK 将此以 UNICODE 编码的编译后的 class 文件保存到我们的操作系统中形成我们见到的.class 文件。对我们来说，我们最终获得的.class 文件是内容以 UNICODE 编码格式保存的类文件，它内部包含我们源程序中的中文字符串，只不过此时它已经由 file.encoding 格式转化为 UNICODE 格式了。

这一步中，对于 JSP 源程序文件是不同的，对于 JSP，这个过程是这样的：即 WEB 容器调用 JSP 编译器，JSP 编译器先查看 JSP 文件中是否设置有文件编码格式，如果 JSP 文件中没有设置 JSP 文件的编码格式，则 JSP 编译器调用 JDK 先把 JSP 文件用 JVM 默认的字符编码格式(也即 WEB 容器所在的操作系统的默认的 file.encoding)转化为临时的 Servlet 类，然后再把它编译成 UNICODE 格式的 class 类，并保存在临时文件夹中。如：在中文 win2k 上，WEB 容器就把 JSP 文件从 GBK 编码格式转化为 UNICODE 格式，然后编译成临时保存的 Servlet 类，以响应用户的请求。

第三步，运行第二步编译出来的类，分为三种情况：

A、 直接在 console 上运行的类

B、 EJB 类和不可以直接运行的支持类(如 JavaBean 类)

C、 JSP 代码和 Servlet 类

D、 JAVA 程序和数据库之间

下面我们分这四种情况来看。

A、直接在 console 上运行的类

这种情况，运行该类首先需要 JVM 支持，即操作系统中必须安装有 JRE。运行过程是这样的：首先 java 启动 JVM，此时 JVM 读出操作系统中保存的 class 文件并把内容读入内存中，此时内存中为 UNICODE 格式的

class 类，然后 JVM 运行它，如果此时此类需要接收用户输入，则类会默认用 file.encoding 编码格式对用户输入的串进行编码并转化为 unicode 保存入内存（用户可以设置输入流的编码格式）。程序运行后，产生的字符串（UNICODE 编码的）再回交给 JVM，最后 JRE 把此字符串再转化为 file.encoding 格式(用户可以设置输出流的编码格式)传递给操作系统显示接口并输出到界面上。

以上每一步的转化都需要正确的编码格式转化，才能最终不出现乱码现象。

B、EJB 类和不可以直接运行的支持类(如 JavaBean 类)

由于 EJB 类和不可以直接运行的支持类，它们一般不与用户直接交互输入和输出，它们常常与其它的类进行交互输入和输出，所以它们在第二步被编译后，就形成了内容是 UNICODE 编码的类保存在操作系统中了，以后只要它与其它的类之间的交互在参数传递过程中没有丢失，则它就会正确的运行。

C、JSP 代码和 Servlet 类

经过第二步后，JSP 文件也被转化为 Servlets 类文件，只不过它不像标准的 Servlets 一校存在于 classes 目录中，它存在于 WEB 容器的临时目录中，故这一步中我们也把它做为 Servlets 来看。

对于 Servlets，客户端请求它时，WEB 容器调用它的 JVM 来运行 Servlet，首先，JVM 把 Servlet 的 class 类从系统中读出并装入内存中，内存中是以 UNICODE 编码的 Servlet 类的代码，然后 JVM 在内存中运行该 Servlet 类，如果 Servlet 在运行的过程中，需要接受从客户端传来的字符如：表单输入的值和 URL 中传入的值，此时如果程序中没有设定接受参数时采用的编码格式，则 WEB 容器会默认采用 ISO-8859-1编码格式来接受传入的值并在 JVM 中转化为 UNICODE 格式的保存在 WEB 容器的内存中。Servlet 运行后生成输出，输出的字符串是 UNICODE 格式的，紧接着，容器将 Servlet 运行产生的 UNICODE 格式的串（如 html 语法，用户输出的串等）直接发送到客户端浏览器上并输出给用户，如果此时指定了发送时输出的编码格式，则按指定的编码格式输出到浏览器上，如果没有指定，则默认按 ISO-8859-1编码发送到客户的浏览器上。

D、Java 程序和数据库之间

对于几乎所有数据库的 JDBC 驱动程序，默认的在 JAVA 程序和数据库之间传递数据都是以 ISO-8859-1为默认编码格式的，所以，我们的程序在向数据库内存储包含中文的数据时，JDBC 首先是把程序内部的 UNICODE 编码格式的数据转化为 ISO-8859-1的格式，然后传递到数据库中，在数据库保存数据时，它默认即以 ISO-8859-1保存，所以，这是为什么我们常常在数据库中读出的中文数据是乱码。

3、分析常见的 JAVA 中文问题几个必须清楚的原则

首先，经过上面的详细分析，我们可以清晰地看到，任何 JAVA 程序的生命期中，其编码转换的关键过程是在于：最初编译成 class 文件的转码和最终向用户输出的转码过程。

其次，我们必须了解 JAVA 在编译时支持的、常用的编码格式有以下几种：

*ISO-8859-1，8-bit，同8859_1,ISO-8859-1,ISO_8859_1等编码

*Cp1252，美国英语编码，同 ANSI 标准编码

*UTF-8，同 unicode 编码

*GB2312，同 gb2312-80,gb2312-1980等编码

*GBK ，同 MS936，它是 gb2312的扩充

及其它的编码，如韩文、日文、繁体中文等。同时，我们要注意这些编码间的兼容关体系如下：

unicode 和 UTF-8编码是一一对应的关系。GB2312可以认为是 GBK 的子集，即 GBK 编码是在 gb2312上扩展来的。同时，GBK 编码包含了20902个汉字，编码范围为：0×8140-0xfefe，所有的字符可以一一对应到

UNICODE2.0中来。

再次，对于放在操作系统中的.java 源程序文件，在编译时，我们可以指定它内容的编码格式，具体来说用-encoding 来指定。注意：如果源程序中含有中文字符，而你用-encoding 指定为其它的编码字符，显然是要出错的。用-encoding 指定源文件的编码方式为 GBK 或 gb2312，无论我们在什么系统上编译含有中文字符的 JAVA 源程序都不会有问题，它都会正确地将中文转化为 UNICODE 存储在 class 文件中。

然后，我们必须清楚，几乎所有的 WEB 容器在其内部默认的字符编码格式都是以 ISO-8859-1为默认值的，同时，几乎所有的浏览器在传递参数时都是默认以 UTF-8的方式来传递参数的。所以，虽然我们的 Java 源文件在出入口的地方指定了正确的编码方式，但其在容器内部运行时还是以 ISO-8859-1来处理的。

4、中文问题的分类及其建议最优解决办法

了解以上 JAVA 处理文件的原理之后，我们就可以提出了一套建议最优的解决汉字问题的办法。

我们的目标是：我们在中文系统中编辑的含有中文字符串或进行中文处理的 JAVA 源程序经编译后可以移值到任何其它的操作系统中正确运行，或拿到其它操作系统中编译后能正确运行，能正确地传递中文和英文参数，能正确地和数据库交流中英文字符串。

我们的具体思路是：在 JAVA 程序转码的入口和出口及 JAVA 程序同用户有输入输出转换的地方限制编码方法使之正确即可。

具体解决办法如下：

1、 针对直接在 console 上运行的类

对于这种情况，我们建议在程序编写时，如果需要从用户端接收用户的可能含有中文的输入或含有中文的输出，程序中应该采用字符流来处理输入和输出，具体来说，应用以下面向字符型节点流类型：

对文件：FileReader，FileWrieter

其字节型节点流类型为：FileInputStream，FileOutputStream

对内存（数组）：CharArrayReader，CharArrayWriter

其字节型节点流类型为：ByteArrayInputStream，ByteArrayOutputStream

对内存（字符串）：StringReader，StringWriter

对管道：PipedReader，PipedWriter

其字节型节点流类型为：PipedInputStream，PipedOutputStream

同时，应该用以下面向字符型处理流来处理输入和输出：

BufferedWriter，BufferedReader

其字节型的处理流为：BufferedInputeStream，BufferedOutputStream

InputStreamReader，OutputStreamWriter

其字节型的处理流为：DataInputStream，DataOutputStream

其中 InputStreamReader 和 InputStreamWriter 用于将字节流按照指定的字符编码集转换到字符流，如：

InputStreamReader in = new InputStreamReader(System.in,” GB2312″);

OutputStreamWriter out = new OutputStreamWriter (System.out,” GB2312″);

例如：采用如下的示例 JAVA 编码就达到了要求：

```
//Read.java
import java.io.*;
```

```
public class Read {
public static void main(String[] args) throws IOException {
String str = " 中文测试，这是内部硬编码的串"+" test english character";
String strin= "";
BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in," gb2312"));  //设置输入接口按中文编码
BufferedWriter stdout = new BufferedWriter(new OutputStreamWriter(System.out," gb2312"));  //设置输出接口按中文编码
stdout.write("请输入:");
stdout.flush();
strin = stdin.readLine();
stdout.write("这是从用户输入的串:"+strin);
stdout.write(str);
stdout.flush();
}}
```

同时，在编译程序时，我们用以下方式来进行：

```
javac -encoding gb2312 Read.java
```

2、 针对EJB类和不可以直接运行的支持类(如JavaBean类)

由于这种类它们本身被其它的类调用，不直接与用户交互，故对这种类来说，我们的建议的处理方式是内部程序中应该采用字符流来处理程序内部的中文字符串（具体如上面一节中一样），同时，在编译类时用-encoding gb2312参数指示源文件是中文格式编码的即可。

3、 针对Servlet类

针对Servlet，我们建议用以下方法：

在编译Servlet类的源程序时，用-encoding指定编码为GBK或GB2312，且在向用户输出时的编码部分用response对象的setContentType("text/html;charset=GBK");或gb2312来设置输出编码格式，同样在接收用户输入时，我们用request.setCharacterEncoding("GB2312")；这样无论我们的servlet类移植到什么操作系统中，只有客户端的浏览器支持中文显示，就可以正确显示。如下是一个正确的示例：

```
//HelloWorld.java
package hello;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
public void init() throws ServletException { }
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException,
ServletException
```

```
{
request.setCharacterEncoding(" GB2312" ); //设置输入编码格式
response.setContentType(" text/html;charset=GB2312" ); //设置输出编码格式
PrintWriter out = response.getWriter(); //建议使用 PrintWriter 输出
out.println(" <hr>" );
out.println(" Hello World! This is created by Servlet!测试中文!" );
out.println(" <hr>" );
}
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException,
ServletException
{
request.setCharacterEncoding(" GB2312" ); //设置输入编码格式
response.setContentType(" text/html;charset=GB2312" ); //设置输出编码格式
String name = request.getParameter(" name" );
String id = request.getParameter(" id" );
if(name==null) name=" ";
if(id==null) id=" ";
PrintWriter out = response.getWriter(); //建议使用 PrintWriter 输出
out.println(" <hr>" );
out.println(" 你传入的中文字串是:" + name);
out.println(" <hr>你输入的 id 是:" + id);
out.println(" <hr>" );
}
public void destroy() { }
}
```

请用 javac -encoding gb2312 HelloWorld.java 来编译此程序。

测试此 Servlet 的程序如下所示:

```
<%@page contentType=" text/html; charset=gb2312" %>
<%request.setCharacterEncoding(" GB2312" );%>
<html><head><title></title>
<Script language=" JavaScript" >
function Submit() {
//通过 URL 传递中文字符串值给 Servlet
document.base.action = " ./HelloWorld?name=中文" ;
document.base.method = " POST" ;
document.base.submit();
}
```

```
</Script>
</head>
<body bgcolor=”#FFFFFF” text=”#000000″ topmargin=”5″>
<form name=”base” method = “POST” target=”_self”>
<input name=”id” type=”text” value=”" size=”30″>
<a href = “JavaScript:Submit()”>传给 Servlet</a>
</form></body></html>
//testchinese.jsp
<%@page pageEncoding=”GB2312″ %>
<%@page contentType=”text/html; charset=gb2312″ %>
<%request.setCharacterEncoding(”GB2312″);%>
<%
String action = request.getParameter(”ACTION”);
String name = “”;
String str = “”;
if(action!=null && action.equals(”SENT”))
{
name = request.getParameter(”name”);
str = request.getParameter(”str”);
}
%>
<html>
<head>
<title></title>
<Script language=”JavaScript”>
function Submit()
{
document.base.action = “?ACTION=SENT&str=传入的中文”;
document.base.method = “POST”;
document.base.submit();
}
</Script>
</head>
<body bgcolor=”#FFFFFF” text=”#000000″ topmargin=”5″>
<form name=”base” method = “POST” target=”_self”>
<input type=”text” name=”name” value=”" size=”30″>
<a href = “JavaScript:Submit()”>提交</a>
```

</form>

<%

if(action!=null && action.equals(”SENT”))

{

out.println(”
你输入的字符为:”+name);

out.println(”
你通过URL传入的字符为:”+str);

}

%>

</body>

</html>

由于大多数本地测试环境是TOMCAT，现也将其中文问题一并附上。

Tomcat中文问题- -

在tomcat5中发现了以前处理tomcat4的方法不能适用于处理直接通过url提交的请求,上网找资料终于发现了最完美的解决办法，不用每个地方都转换了，而且无论get,和post都正常。写了个文档，贴出来希望跟我有同样问题的人不再像我一样痛苦一次:-)

问题描述:

1 表单提交的数据，用request.getParameter(“xxx”)返回的字符串为乱码或者？？

2 直接通过url如http://localhost/a.jsp?name=中国，这样的get请求在服务端用request.getParameter(“name”)时返回的是乱码；按tomcat4的做法设置Filter也没有用或者用request.setCharacterEncoding(”GBK”);也不管用

原因:

1 tomcat的j2ee实现对表单提交即post方式提示时处理参数采用缺省的iso-8859-1来处理

2 tomcat对get方式提交的请求对query-string 处理时采用了和post方法不一样的处理方式。(与tomcat4不一样,所以设置setCharacterEncoding(“gbk”))不起作用。

解决办法:

首先所有的jsp文件都加上:

1 实现一个Filter.设置处理字符集为GBK。(在tomcat的webapps/servlet-examples目录有一个完整的例子。请参考web.xml和SetCharacterEncodingFilter的配置。)

1) 只要把%TOMCAT安装目录%/webappsservlets-examplesWEB-INFclassesfiltersSetCharacterEncodingFilter.class文件拷到你的webapp目录/filters下，如果没有filters目录，就创建一个。

2)在你的web.xml里加入如下几行: <filter>

<filter-name>Set Character Encoding</filter-name>

<filter-class>filters.SetCharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>GBK</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>Set Character Encoding</filter-name>

<url-pattern>/*</url-pattern>

</filter-mapping>

3)完成.

2 get 方式的解决办法

1) 打开 tomcat 的 server.xml 文件, 找到区块, 加入如下一行:

URIEncoding=”GBK”

完整的应如下:

<Connector

port=”80″ maxThreads=”150″ minSpareThreads=”25″ maxSpareThreads=”75″

enableLookups=”false” redirectPort=”8443″ acceptCount=”100″

debug=”0″ connectionTimeout=”20000″

disableUploadTimeout=”true”

URIEncoding=”GBK”

/>

2)重启 tomcat,一切 OK。

执行如下 jsp 页页测试是否成功

<%@ page contentType=”text/html;charset=gb2312″ %>

<%@ page import=”java.util.*” %>

<%

String q=request.getParameter(”q”);

q = q == null? “没有值” : q;

%>

<HTML>

<HEAD><TITLE>新闻列表显示</TITLE>

<META http-equiv=Content-Type content=”text/html; charset=gb2312″ >

<META http-equiv=pragma content=no-cache>

<body>

你提交了:

<%=q%>

<form action=”tcnchar.jsp” method=”post”>

输入中文:<input type=”text” name=”q”><input type=”submit” value=”确定”>

通过 get 方式提交

</form>

</BODY></HTML>

测试结果如果你输入文本框或者点超链都会显示:你提交了”中国”,说明成功!!!!!

10/**forward 与 sendRedirect 区别是什么？**

1.RequestDispatcher.forward()　　是在服务器端起作用,当使用 forward()时,Servletengine 传递 HTTP 请求从当前的 Servlet or JSP 到另外一个 Servlet, JSP 或普通 HTML 文件, 也即你的 form 提交至 a.jsp, 在 a.jsp 用到了 forward()重定向至 b.jsp,此时 form 提交的所有信息在 b.jsp 都可以获得,参数自动传递。但 forward()无法重定向至有 frame 的 jsp 文件,可以重定向至有 frame 的 html 文件,同时 forward()无法在后面带参数传递, 比如 servlet?name=value, 这样不行,可以程序内通过 request.setAttribute(”name”,value)来传至下一个页面。　　重定向后浏览器地址栏 URL 不变, 因为完成一个业务操作往往需要跨越多个步骤, 每一步骤完成相应的处理后, 转向到下一个步骤。比如, 通常业务处理在 Servlet 中处理, 处理的结果转向到一个 JSP 页面进行显示。这样看起来类似于 Servlet 链的功能, 但是还有一些区别。一个 RequestDispatcher 对象可以把请求发送到任意一个服务器资源,而不仅仅是另外一个 Servlet。　　注意, 只有在尚未向客户端输出响应时才可以调用 forward()方法, 如果页面缓存不为空, 在重定向前将自动清除缓存。否则将抛出一个异常

例:servlet 文件中重定向

public void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException{　　response.setContentType(”text/html;charset=gb2312″);　　ServletContext sc = getServletContext();　　RequestDispatcher rd = null;　　rd = sc.getRequestDispatcher(”/index.jsp”);　　rd.forward(request, response);　}

2.response.sendRedirect()　　是在用户的浏览器端工作, sendRedirect()可以带参数传递, 比如 servlet?name=value 传至下个页面, 同时它可以重定向至不同的主机上, 且在浏览器地址栏上会出现重定向页面的 URL。HttpServletResponse 接口定义了可用于转向的 sendRedirect()方法。代码如下: public void sendRedirect(java.lang.String location)throws java.io.IOException

这个方法将响应定向到参数 location 指定的、新的 URL。location 可以是一个绝对的 URL, 如 response.sendRedirect (”http://java.sun.com”)也可以使用相对的 URL。如果 location 以”/”开头, 则容器认为相对于当前 Web 应用的根, 否则, 容器将解析为相对于当前请求的 URL。这种重定向的方法, 将导致客户端浏览器的请求 URL 跳转。从浏览器中的地址栏中可以看到新的 URL 地址, 作用类似于上面设置 HTTP 响应头信息的实现

3.如何得到 RequestDispatcher 有三种方法可以得到 Request Dispatcher 对象。

(1).javax.servlet.ServletRequest 的 getRequestDispatcher(String path)方法, 其中 path 可以是相对路径, 但不能越出当前 Servlet 上下文。如果 path 以”/”开头, 则解析为相对于当前上下文的根。

(2).javax.servlet.ServletContext 的 getRequestDispatcher(String path)方法, 其中 path 必须以”/”开头, 路径相对于当前的 Servlet 上下文。可以调用 ServletContext 的 getContext(String uripath)得到另一个 Servlet 上下文, 并可以转向到外部上下文的一个服务器资源链接。

(3).使用 javax.servlet.ServletContext 的 getNamedDispatcher(String name)得到名为 name 的一个 Web

资源，包括 Servlet 和 JSP 页面。这个资源的名字在 Web 应用部署描述文件 web.xml 中指定。

这三种方法的使用有细微的差别。比如，下面是一个应用的配置文件 web.xml：

<!DOCTYPE web-app PUBLIC “ -//Sun Microsystems, Inc.//DTD Web Application 2.3//EN ” “http://java.sun.com/dtd/web-app_2_3.dtd” >

<web-app> <servlet> <servlet-name>FirstServlet</servlet-name>

<servlet-class>org.javaresearch.redirecttest.ServletOne</-servlet-class> </servlet>

<servlet> <servlet-name>SecondServlet</servlet-name>

<servlet-class>org.javaresearch.redirecttest.ServletTwo</servlet-class> </servlet>

<servlet-mapping> <servlet-name>FirstServlet</servlet-name>

<url-pattern>/servlet/firstservlet</url-pattern> </servlet-mapping> <servlet-mapping>

<servlet-name>SecondServlet</servlet-name>

<url-pattern>/servlet/secondservlet</url-pattern> </servlet-mapping>

</web-app> 其中定义了两个 Servlet，名字分别为 FirstServlet 和 SecondServlet，对应的类分别为 org.javaresearch.redirecttest.ServletOne 和 org. javaresearch.redirecttest.ServletTwo。可以在浏览器中通过类似于下面的链接访问： http://localhost:8080/servlet/firstservlet

使用(1)中方法，例如在 firstservlet 可以写入下面的代码：

RequestDispatcher rd = request.getRequestDispatcher(”/second/servlet”); rd.forward(request, response);

此时控制权将转向到第二个 Servlet 了。

使用(2)中的方法，可以从 Servlet Context 中得到 RequestDispatcher 代码如下：

RequestDispatcher rd = getServletContext().getRequest Dispatcher(”/servlet/secondservlet”); rd.forward(request, response);

使用(3)中的方法，从上面的 web. xml 配置文件可以看到定义了两个 Servlet，名字分别为 FirstServlet 和 SecondServlet，所以可以得到命名的 Dispatcher：

RequestDispatcher rd = getServletContext().getNamedDispatcher(” SecondServlet ”); rd.forward(request, response);

这样也可以重定向到 SecondServlet 了。

JSP 页面中的重定向,JSP 在解析后编译为一个 Servlet 运行,所以在 JSP 中也可以使用上面的重定向代码,并且，JSP 还提供了更便利的操作，如下：

<jsp:forward page=” nextpage.jsp” />

JSP 页面执行到这儿，将终止当前的处理，将控制权交由 nextpage.jsp。

4.如何选择

RequestDispatcher.forward()方法和 HttpServletResponse.sendRedirect()方法的区别是：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 Request Dispatcher.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

HttpServletResponse.sendRequest()方法。

11/JSP&Servlet技术面试题

1. 描述JSP和Servlet的区别、共同点、各自应用的范围

2. 在Web开发中需要处理HTML标记时，应做什么样的处理，要筛选那些字符（< > & “”）

3. 在JSP中如何读取客户端的请求，如何访问CGI变量，如何确定某个Jsp文件的真实路径。

4. 描述Cookie和Session的作用，区别和各自的应用范围，Session工作原理。

5. 列出Jsp中包含外部文件的方式，两者有何区别。

6. 说明Jsp中errorPage的作用，应用范围。

7. 介绍在Jsp中如何使用JavaBeans。

8. 简单介绍JSP的标记库

9. Jsp和Servlet中的请求转发分别如何实现。

12/**某个公司的面试测试题 两天时间完成**

请在2日内完成该测试；提交完整的eclipse工程和数据库建表语句。

功能简单说明：

用户登录：按设置的用户名登录系统

客户管理：设置客户编号（ajax查询），名称，地址，增删改查 CRUD

员工设置：设置姓名，登录用户名（ajax查询），密码，管理的客户（多选），增删改查

入库扫描：录入一个客户编号，回车ajax显示客户信息

再录入运单号，回车ajax检查是否重复运单，回车保存后数据列表局部刷新，页面下方显示一条数据

订单查询：根据客户和入库时间(可勾选)，查询订单数据，列出：运单号 客户编号 客户名称 客户地址 入库时间

分页显示，包括合计，可以导出excel

4 技术要求：

JSP+servlet+JDBC(标准sql)，允许使用需要的第三方jar包

数据库连接使用连接池方式（tomcat配置或者使用其他java连接池类读取db.properties文件方式）

业务层，数据层严格分开

注释清晰

请保证自己独立完成

13/**一套Java笔试题 包括JSP和JDBC等方面知识**

1.编写一个JSP页面，页面中含有一个表单、文本框、下拉列表框、单选框、复选框。

2.展示JDBC连数据库的调用方法。

3.描述page|request|session|application适用范围的含义。

4.数据类型之间的转换：

(1)String、java.Util.date

(2)String、float

(3)”123″、long

(4)String、TimeStamp

5.通过 JSP 代码片段展示 forword 动作的使用方法.

6.描述 getProperty 和 setProperty 动作的作用

7.post、get 两种方法的区别和特点

8、什么是 taglib，如何在 Jsp 中使用自定义的标记，请结合自己的项目实际描述说明

9.通过 jsp+javascript 代码实例展示如何在一个 jsp 页面中动态显示属性列表,树的层次和每个层次包含的节点树由数据库动态查出来!

14/**某公司的 Java JSP 上机题**

1 请按测试 demo 页面需要实现的功能自行设计数据表，数据库不限（oracle，mysql，sqlserver 均可）

2 所需 JSP 页面可以按测试 demo 功能，自己做 JSP，有录入框和表格即可，页面无需任何美化

3 测试 demo 地址：http://124.42.120.53/test/index.html （无需密码，直接登录）

功能简单说明：

用户登录：按设置的用户名登录系统

客户管理：设置客户编号（ajax 查询），名称，地址，增删改查 CRUD

员工设置：设置姓名，登录用户名（ajax 查询），密码，管理的客户（多选），增删改查

入库扫描：录入一个客户编号，回车 ajax 显示客户信息

再录入运单号，回车 ajax 检查是否重复运单，回车保存后数据列表局部刷新，页面下方显示一条数据

订单查询：根据客户和入库时间(可勾选)，查询订单数据，列出：运单号 客户编号 客户名称 客户地址 入库时间

分页显示，包括合计，可以导出 excel

4 技术要求：

JSP+servlet+JDBC(标准 sql)，允许使用需要的第三方 jar 包

数据库连接使用连接池方式（tomcat 配置或者使用其他 java 连接池类读取 db.properties 文件方式）

业务层，数据层严格分开

注释清晰

请保证自己独立完成

15/**Servlet 都有哪些方法？主要作用是什么？**

HttpServlet 类包含 init()、destroy()、service() 等方法。其中 init() 和 destroy() 方法是继承的。

(1) init() 方法

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

缺省的 init() 方法通常是符合要求的，但也可以用定制 init() 方法来覆盖它，典型的是管理服务器端资源。 例如，可能编写一个定制 init() 来只用于一次装入 GIF 图像，改进 Servlet 返回 GIF 图像和含有多个客户机请求的性能。另一个示例是初始化数据库连接。缺省的 init() 方法设置了 Servlet 的初

始化参数，并用它的 ServletConfig 对象参数来启动配置， 因此所有覆盖 init() 方法的 Servlet 应调用 super.init() 以确保仍然执行这些任务。在调用 service() 方法之前,应确保已完成了 init() 方法。

(2) service() 方法

service() 方法是 Servlet 的核心。每当一个客户请求一个 HttpServlet 对象，该对象的 service() 方法就要被调用，而且传递给这个方法一个“请求”(ServletRequest) 对象和一个“响应”(ServletResponse) 对象作为参数。 在 HttpServlet 中已存在 service() 方法。缺省的服务功能是调用与 HTTP 请求的方法相应的 do 功能。例如， 如果 HTTP 请求方法为 GET ，则缺省情况下就调用 doGet() 。 Servlet 应该为 Servlet 支持的 HTTP 方法覆盖 do 功能。因为 HttpServlet.service() 方法会检查请求方法是否调用了适当的处理方法，不必要覆盖 service() 方法。只需覆盖相应的 do 方法就可以了。

= 当一个客户通过 HTML 表单发出一个 HTTP POST 请求时， doPost （）方法被调用。 与 POST 请求相关的参数作为一个单独的 HTTP 请求从浏览器发送到服务器。当需要修改服务器端的数据时，应该使用 doPost() 方法。

= 当一个客户通过 HTML 表单发出一个 HTTP GET 请求或直接请求一个 URL 时， doGet() 方法被调用。与 GET 请求相关的参数添加到 URL 的后面，并与这个请求一起发送。当不会修改服务器端的数据时，应该使用 doGet() 方法。

Servlet 的响应可以是下列几种类型:

一个输出流，浏览器根据它的内容类型（如 text/HTML ）进行解释。

一个 HTTP 错误响应 ，重定向到另一个 URL 、 servlet 、 JSP 。

(3) destroy() 方法

destroy() 方法仅执行一次，即在服务器停止且卸装 Servlet 时执行该方法。典型的，将 Servlet 作为服务器进程的一部分来关闭。缺省的 destroy() 方法通常是符合要求的，但也可以覆盖它，典型的是管理服务器端资源。例如，如果 Servlet 在运行时会累计统计数据，则可以编写一个 destroy() 方法，该方法用于在未装入 Servlet 时将统计数字保存在文件中。另一个示例是关闭数据库连接。

当服务器卸装 Servlet 时，将在所有 service() 方法调用完成后，或在指定的时间间隔过后调用 destroy() 方法。一个 Servlet 在运行 service() 方法时可能会产生其它的线程，因此请确认在调用 destroy() 方法时，这些线程已终止或完成。

(4) GetServletConfig（）方法

GetServletConfig （）方法返回一个 ServletConfig 对象，该对象用来返回初始化参数和 ServletContext 。 ServletContext 接口提供有关 servlet 的环境信息。

(5) GetServletInfo（）方法

GetServletInfo （）方法是一个可选的方法，它提供有关 servlet 的信息，如作者、版本、版权。

当服务器调用 sevlet 的 Service （）、 doGet （）和 doPost （）这三个方法时，均需要 “请求”和“响应”对象作为参数。“请求”对象提供有关请求的信息，而“响应”对象提供了一个将响应信息返回给浏览器的一个通信途径。 javax.servlet 软件包中的相关类为 ServletResponse 和 ServletRequest ，而 javax.servlet.http 软件包中的相关类为 HttpServletRequest 和 HttpServletResponse 。Servlet

通过这些对象与服务器通信并最终与客户机通信。 Servlet 能通过调用“请求”对象的方法获知客户机环境，服务器环境的信息和所有由客户机提供的信息。 Servlet 可以调用“响应”对象的方法发送响应，该响应是准备发回客户机的。

16/**Java Servlet 的主要功能和作用是什么？**

Servlet 通过创建一个框架来扩展服务器的能力，以提供在 Web 上进行请求和响应服务。当客户机发送请求至服务器时，服务器可以将请求信息发送给 Servlet ，并让 Servlet 建立起服务器返回给客户机的响应。 当启动 Web 服务器或客户机第一次请求服务时，可以自动装入 Servlet 。装入后， Servlet 继续运行直到其它客户机发出请求。 Servlet 的功能涉及范围很广。例如， Servlet 可完成如下功能：

(1) 创建并返回一个包含基于客户请求性质的动态内容的完整的 HTML 页面。

(2) 创建可嵌入到现有 HTML 页面中的一部分 HTML 页面（ HTML 片段）。

(3) 与其它服务器资源（包括数据库和基于 Java 的应用程序）进行通信。

(4) 用多个客户机处理连接，接收多个客户机的输入，并将结果广播到多个客户机上。例如， Servlet 可以是多参与者的游戏服务器。

(5) 当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至 applet 的新连接，并将该连接保持在打开状态。当允许客户机和服务器简单、高效地执行会话的情况下， applet 也可以启动客户浏览器和服务器之间的连接。可以通过定制协议或标准（如 IIOP ）进行通信。

(6) 对特殊的处理采用 MIME 类型过滤数据，例如图像转换和服务器端包括（ SSI ）。

(7) 将定制的处理提供给所有服务器的标准例行程序。例如， Servlet 可以修改如何认证用户。

17/**Request 对象的主要方法有哪些？**

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

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

18/**某银行信息中心的 J2EE 笔试题**

1.jsp 中定义 javabean 的作用域可以在用户会话注销后仍能够访问的是？（多选）

a.page

b.session

c.application

d.request

2.当浏览器第二次访问以下下 jsp 网页时输出结果是什么？

```
<%! int a=0; %>
<%
int b=0;
a++;
b++;
%>
a:<%=a %><br>
b:<%=b %>
```

3.web.xml 中不包括哪些定义（多选）

a.默认起始页

b.servlet 启动延迟时间定义

c.error 处理页面

d.jsp 文件改动后重新载入时间

4.需要在两个数据库之间完成事务交易，需要应用服务器中配置那种连接池？（多选）

a.XA Datasource

b.C3P0 的连接池

c.type2的连接池

d.type4方式的连接池。

5.使用纯 java 实现的 jdbc 驱动是哪种类型的驱动（多选）

a.type1 b,type2 c.type3 d,type4

6.线程安全的类是那些？（多选）

a.webwork 1.x 框架中的 action 处理类

b.struts1.x 框架中的 action 处理类

c.普通的 Servlet

d.普通的 Filter

7.下列解析大量数据 XML 的方式那种速度最快？（多选）

a.DOM b. SAX c.JDOM d.JAX

8.JSP 中定义<%@ page contentType=”text/html;charset=utf-8″ pageEncoding=”GBK”%>,
说明（多选）

a.输出的网页编码是 utf8的

b.输出的网页编码是 GBK 的

c.输出的网页采用 utf 编码，显示在页面的字符集使用 GBK

d.输出的网页采用 GBK 编码，显示在页面的字符集使用 utf8编码

9.在 servlet 的 doGet 方法，下面哪几个说法是错误的（多选）

a.redirect 到一个 jsp 页面时，之前 doGet 放在 request 中的对象都会丢失

b.redirect 到一个 jsp 页面时，之前 doGet 放在 session 中的对象都会丢失

c.forward 到一个 jsp 页面时，之前 doGet 放在 request 中的对象都会丢失

d.forward 到一个 jsp 页面时，之前 doGet 放在 session 中的对象都会丢失

10.web.xml 中定义了如下内容，下列哪种说法是错误的？（多选）

<resource-ref>

<description></description>

<res-ref-name>/sys/myresource</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

a.应用服务器需要配置名为/sys/myresource 的数据库连接池，才能为应用提供数据访问

b.应用代码中使用了 look up（”/sys/myresource”）的方式获得资源

c./sys/myresource 的资源的认证方式是在应用服务器中配置的

d./sys/myresource 在 war 的文件部署时需要映射到应用服务器上

11.在 j2ee 服务中，下列哪些 url 进行 GET 方式调用时，可以被配置成 Filter 类拦截（多选）

a./jsp/hello.jsp

b./images/logo.gif

c./webapp/showView.do

d./webapp/showChart

19/**描述 JSP 和 Servlet 的区别、共同点、各自应用的范围**

JSP 在本质上就是 SERVLET,但是两者的创建方式不一样.Servlet 完全是 JAVA 程序代码构成擅长于流程控制和事务处理而通过 Servlet
来生成动态网页;JSP 由 HTML 代码和 JSP 标签构成，可以方便地编写动态网页

因此在实际应用中采用 Servlet 来控制业务流程,而采用 JSP 来生成动态网页.在 struts 框架中,JSP 位于 MVC 设计模式的视图层,而 Servlet 位于控制层.

答案2:

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。

20/介绍一下 javax.servlet.Servlet 接口及其主要方法

Servlet 接口的主要作用是提供 Servlet 生命周期的 init()、service()和 destroy()方法。

servlet 接口中的主要方法有:

1、void init(ServletConfit config)throws ServletException

在 servlet 被载入后和实施服务前由 servlet 引擎进行一次性调用。如果 init()产生溢出 UnavailableException，则 servle 退出服务。

2、ServletConfig getServletConfig()

返回传递到 servlet 的 init()方法的 ServletConfig 对象

void service(ServletRequest request, ServletResponse response)throws ServletException,IOException 处理 request 对象中描述的请求，使用 response 对象返回请求结果

String getServletInfo() 返回描述 servlet 的一个字符串

void destory() 当 servlet 将要卸载时由 servlet 引擎调用，销毁 Servlet 实例。

21/**HttpServlet 类中的主要方法都有哪些？各自的作用是什么？**

HttpServlet 的主要方法有 doGet，doPost，doPut，doDelete，doTrace 等等

Void doGet(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException 由 servlet 引擎调用用处理一个 HTTP GET 请求。输入参数、HTTP 头标和输入流可从 request 对象、response 头标和 response 对象的输出流中获得。

Void doPost(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException 由 servlet 引擎调用用处理一个 HTTP POST 请求。输入参数、HTTP 头标和输入流可从 request 对象、response 头标和 response 对象的输出流中获得。

Void doPut(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException 由 servlet 引擎调用用处理一个 HTTP PUT 请求。本方法中请求 URI 指出被载入的文件位置。

Void doDelete(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException 由 servlet 引擎调用用处理一个 HTTP DELETE 请求。请求 URI 指出资源被删除。

Void doOptions(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException 由 servlet 引擎调用用处理一个 HTTP OPTIONS 请求。返回一个 Allow 响应头标表明此 servlet 支持的 HTTP 方法。一个 servlet 不需要覆盖此方法，因为 HttpServlet 方法已经实现规范所需的功能。

Void doTrace(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException 由 servlet 引擎调用用处理一个 HTTP TRACE 请求。使得请求头标被反馈成响应关标。一个 servlet 不需要覆盖此方法，因为 HttpServlet 方法已经实现 HTTP 规范所需的功能。

Void service(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException Service(Request request,Response response)调用的一个立即方法，带有指定 HTTP 请求和响应。此方法实际上将请求导向 doGet()、doPost()等等。不应该覆盖此方法。

Void service(Request request,Response response)throws ServletException,IOException 将请求和响应对象置入其指定的 HTTP 子类，并调用指定 HTTP 的 service()方法。

22/**JSP&Servlet 技术面试题**

1.描述 JSP 和 Servlet 的区别、共同点、各自应用的范围

2.在 Web 开发中需要处理 HTML 标记时，应做什么样的处理，要筛选那些字符（< > & “ ”）

3.在 JSP 中如何读取客户端的请求，如何访问 CGI 变量，如何确定某个 Jsp 文件的真实路径。

4.描述 Cookie 和 Session 的作用，区别和各自的应用范围，Session 工作原理。

5.列出 Jsp 中包含外部文件的方式，两者有何区别。

6.说明 Jsp 中 errorPage 的作用，应用范围。

7.介绍在 Jsp 中如何使用 JavaBeans。

8.简单介绍 JSP 的标记库

9. Jsp 和 Servlet 中的请求转发分别如何实现。

23/**Java Web 方向的综合面试题**

1. HTML 全称，和 XML 的相同点和不同点。

2. 简述 HTML 的作用，以及掌握 HTML 的要点。

3. CSS 中层叠的含义

4. 写出 CSS 选择符的种类以及它们的层叠次序。

5. 你对盒子模式的理解

6. 什么是 JSP

7. JSP 执行原理，并解释 JSP 第一次执行慢的原因。

8. JSP 和 servlet 的相同点和不同点

9. JSP 页面中代码混乱，如何顺利调试一个页面。

10. 在 JSP 网页开发中，include 包含的好处是什么。要注意些什么问题

11. 你在开发页面时，如果出现图片没有显示，或 CSS 样式没有发挥作用你会如何处理

12. setAttribute 的作用，使用时如何避免错误

13. 谈谈 jsp jsp+javabean,MVC ，框架技术（比如 DWR）的优缺点。

14. 你在写 MVC 增删改查时的步骤。容易出 bug 的地方有哪些

15. 404错误的原因

16. 简述 application session request 生命周期与作用域

17. servlet 转页面的两种方式，不同点是什么。JS 能转页吗。如果能怎么写

18. AJAX 包括哪些技术

19. AJAX 的原理

20. 你对 AJAX 的认识（越多越好）

21. JS 的常规套路

22. javascript 在浏览器的作用

23. JS 的语法中 function 的特点。

24. 陈述 js 中的事件使用方式，不同事件的应用的场合

25. 什么是 DOM 什么是 BOM。DOM 的使用要点

26. JSON 的作用。

27. setInterval 和 setTimeout 的区别

28. open()和 showModalDialog()的区别

29. frameset frame iframe 的区别

30. JS 提交表单的好处

31. 你对 JS 验证的认识

32. 写一个迭代 JS 数组的小例子

33. 写一个迭代 JSON 的小例子

34. 全局函数 parseInt， parseFloat ，eval 的作用

35. 写例子能说明 DOM 增删改查。

36. DWR 的配置流程。

37. 在 dwr 中得到 ServletAPI 要通过哪个类

38. 用 EXTJS 做表示层，后台需要做那些改变

39. 无级树的实现要点。

40. 谈谈 SQL 中在写分组是的要领

41. 你对手册的认识，对英文文档的认识

42. 谈谈你在写 SQL 语句时的分析步骤

43. 描述 PreparedStatement 和 Statement 的区别

44. DatabaseMetadata 和 ResultSetMetadata 的区别

45. RIA 的认识

46. 数据库中索引的作用。索引是否越多越好，为什么？

47. 简述笛卡尔乘积与多表连接查询

48. 子查询在 SQL 语句个子句中的作用

49. 你如何看待编码规范。

50. “在一个团队中要么付出你的汗水，要么付出你的智慧，如果都没有，请你走开”。你对这句话的理解

24/**几道 JSP 面试题(附答案)**

1、如何混合使用 Jsp 和 SSI #include?

在 JSP 中可以使用如下方式包含纯 HTML:

但是如果 data.inc 中包含 JSP CODE ，我们可以使用:

2、如何执行一个线程安全的 JSP?

只需增加如下指令

3、JSP 如何处理 HTML FORM 中的数据?

通过内置的 request 对象即可，如下：

4、在 JSP 如何包含一个静态文件?

静态包含如下：

动态包含如下：

5、在 JSP 中如何使用注释?

主要有四中方法：

1。

2。//

3。/**与**/

4。

6、在 JSP 中如何执行浏览重定向?

使用如下方式即可：response.sendRedirect(”http://ybwen.home.chinaren.com/index.html”);

也能物理地改变 HTTP HEADER 属性，如下：

7、如何防止在 JSP 或 SERVLET 中的输出不被 BROWSER 保存在 CACHE 中?

把如下脚本加入到 JSP 文件的开始即可：

8、在 JSP 中如何设置 COOKIE?

COOKIE 是作为 HTTP HEADER 的一部分被发送的，如下方法即可设置：

9、在 JSP 中如何删除一个 COOKIE?

10、在一个 JSP 的请求处理中如何停止 JSP 的执行

如下例：

11、在 JSP 中如何定义方法

你可以定义方法，但是你不能直接访问 JSP 的内置对象，而是通过参数的方法传递。如下：

12、如果 BROWSER 已关闭了 COOKIES，在 JSP 中我如何打开 SESSION 来跟踪

使用 URL 重写即可，如下：

hello1.jsp

hello2.jsp

hello2.jsp

13、在 JSP 中能发送 EMAIL 吗

可以使用 SUN 的专用包：sun.net.smtp 包。如下脚本使用 SmtpClient 类发送 EMAIL。

14、在 SERVLET 中我能调用一个 JSP 错误页吗

当然没问题，如下展示了如何在一个 SERVLET 控制逻辑单元内调用一个 JSP 错误页面。

```
protected void sendErrorRedirect(HttpServletRequest request
HttpServletResponse response String errorPageURL
Throwable e)
throws ServletException IOException {
request.setAttribute (” javax.servlet.jsp.jspException” e);
```

```
getServletConfig().getServletContext().
getRequestDispatcher(errorPageURL).forward(request
response);
}
public void doPost(HttpServletRequest requestHttpServletResponse response) {
try {
// do something
} catch (Exception ex) {
try {
sendErrorRedirect(requestresponse”/jsp/MyErrorPage.jsp”ex);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

15、JSP 和 applet 如何通讯

JSP 如何与 EJB SessionBean 通讯

下面的代码段作了很好的示范

16、当我使用一个结果集时，如何防止字段为”null”的字域显示在我的 HTML 输入文本域中?

可以定义一个简单的函数来达到目的，如下:

然后在 JSP 的 FORM 中，可以这样使用

17、如何中 SERVLET 或 JSP 下载一个文件（如: binarytextexecutable)?

现提供两个解决方案:

A: 使用 HTTP,

B: 在 Servlet 中，通过设置 ContentType 和使用 java.io 包的 Stream 等类可作到.例如:

response.setContentType(”application/x-msword”);

然后想输出缓冲中写一些东东即可。

18、使用 useBean 标志初始化 BEAN 时如何接受初始化参数

使用如下两标签即可:

19、使用 JSP 如何获得客户浏览器的信息?

使用 request.getHeader(String)即可

20、能象调用子程序一样调用 JSP 吗?

当然可以，用

21、当我重编译我的 JSP 使用的一个类后，为什么 JVM 继续使用我的老 CLASS?

与之间的差别?

前一个为静态包含，而后一个为动态包含

22、JSP 的缺点?

1。对 JAVA 程序进行调试没有好东东

2。因大多数的 servlet 引擎不支持 connection pooling

3。Servlet 引擎没有标准

4。JSP 与其它脚本语言的交互

23、JSP 能进行递归调用吗？

当然可以如对 form 的提交给本页

34、如何实现 JSP 的国际化？

为各种版本提供 resource bundles 属性文件即可

25、在 JSP 中如何写文本文件？

使用 PrintWriter 对象，如：

26、如何在 JSP 中包括绝对路径文件？

使用 URLConnection 即可。

27、在 servlets 和 JSP 之间能共享 session 对象吗？

当然可以，

HttpSession session = request.getSession(true);

session.putValue(”variable”"value”);

28、javascript 的变量能复制到 JSP 的 SESSION 中吗？

29、如何设置 cookie 在某一时间后过期？

用 Cookie.setMaxAge(int)

30、如何获得当前的 sessions 数？

可以使用 HttpSessionBindingListeners 来跟踪

31、能设置一些代码在我所有的 JSP 文件之上运行？如果可以，能共享吗？

当然可以可以为你的 JSP 文件定义一个别名:/jsp/=ybwen.genius.myPreprocessingServlet 而以/jsp/为前缀的文件可以使用

32、对一个 JSP 页，如果多个客户端同时请求它，同步可能吗？

在 jsp:useBean 语法中使用 beanName 有何好处？

beanName 使用 Beans.instantiate()初始化 Bean

33、当我使用时，在浏览器的地址栏没有改变？

使用 response.sendRedirect(”newURL”)

34、如何转换 JSP 0.9版本的文件到 JSP1.1?

可使用 sed/awk 即可

35、使用 JSP 能设置 HTML FORM 中输入域的焦点，不用 javascript?

没办法

36、使用 JSP 连接到数据库连接缓冲池的最好方法是什么？

1.使用 JDBC2。0中带有此服务的 Driver

2.使用提供有此服务的 Application server

3.自己写

25/**在 servlets 和 JSP 之间能共享 session 对象吗？**

当然可以，

HttpSession session = request.getSession(true);

session.putValue(”variable”,”value”);

26/**在 JSP 中如何写文本文件？**

使用 PrintWriter 对象，如：

<%@ page import=”java.io.*” %>

<%

String str = “print me”;

String nameOfTextFile = “/usr/anil/imp.txt”;

try {

PrintWriter pw = new PrintWriter(new FileOutputStream(nameOfTextFile));

pw.println(str);

pw.close();

} catch(IOException e) {

out.println(e.getMessage());

}

%>

27/**JSP 的缺点？**

1.对 JAVA 程序进行调试没有好东东

2.因大多数的 servlet 引擎不支持 connection pooling

3.Servlet 引擎没有标准

4.JSP 与其它脚本语言的交互

28/**如何中 SERVLET 或 JSP 下载一个文件（如：binary,text,executable)？**

现提供两个解决方案：

A：使用 HTTP，

B：在 Servlet 中，通过设置 ContentType 和使用 java.io 包的 Stream 等类可作到.例如：

response.setContentType(”application/x-msword”);

然后想输出缓冲中写一些东东即可。

29/**如何防止在 JSP 或 SERVLET 中的输出不被 BROWSER 保存在 CACHE 中？**

把如下脚本加入到 JSP 文件的开始即可：

<%

response.setHeader(”Cache-Control”,”no-store”); //HTTP 1.1

response.setHeader(”Pragma”,”no-cache”); //HTTP 1.0

response.setDateHeader (”Expires”, 0); //prevents caching at the proxy server

%>

30/**Servlet 执行时一般实现哪几个方法？**

public void init(ServletConfig config)

public ServletConfig getServletConfig()

public String getServletInfo()

public void service(ServletRequest request,ServletResponse response)

public void destroy()

31/**Request对象的主要方法**

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

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

JSP是Servlet技术的扩展，本质上是Servlet的简易方式，更强调应用的外表表达。JSP编译后是”类servlet”。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。JSP侧重于视图，Servlet主要用于控制逻辑。

33/**如何现实servlet的单线程模式**

<%@ page isThreadSafe=” false” %>

34/**说出 Servlet 的生命周期，并说出 Servlet 和 CGI 的区别。**

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

与 cgi 的区别在于 servlet 处于服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 servlet。

35/**用 ServletRequest 和 ServletContext 调用 ReqestDispatcher 有什么区别？**

在用 ServletRequest 调用 RequestDispatcher 的时候可以用相对 URL，但是 ServletContext 不行。

36/**当容器调用 servlet 的 destory()方法的时候，servlet 会马上销毁么？ 如果当时这个 servlet 正在执行其他任务或者线程呢？**

是的， 当容器调用 servlet 的 destory()方法的时候，servlet 会马上销毁，但是容器在调用 destory()方法之前，会等 servlet 的 service()方法结束剩余的任务。

37/**我们没有写 servlet 的构造方法，那么容器是怎么创建 servlet 的实例呢？**

容器会自动为 Servlet 写一个无参的构造方法，容器是用 Class.forName(className).newInstance()来创建 servlet 的实例的。

38/**Servlet 的实例是在生命周期什么时候创建的？ 配置 servlet 最重要的是什么？**

Servlet 实例是在 servlet 第一次在容器中被加载的是时候创建的， Init()方法是用来配置这个 servlet 实力的， 这个方法在 servlet 的生命周期中只被调用一次，所以应该把所有 servlet 生命周期中的配置

39/**Servlet 的生命周期？**

Servlet 是一种可以 在 Servlet 容器中运行的组件，那么理所当然就应该有一个从创建到销毁的过程，这个过程我们可以称之为 Servlet 生命周期。Servlet 的生命 周期可以分为加载、实例化、初始化、处理客户请求和卸载五个阶段，体现在方法上主要是 init ()、service () 和 destroy () 三个方法。生 命周期的具体说明如下：

Servlet 容器完成加载 Servlet 类和实例化一个 Servlet 对象

init () 方法完成初始化工作，该方法由 Servlet 容器调用完成

service () 方法处理客户端请求，并返回响应结果

destroy () 方法在 Servlet 容器卸载 Servlet 之前被调用，释放一些资源