1. **jsp有哪些内置对象?作用分别是什么?**

答：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

pageContext 网页的属性是在这里管理

session 与请求有关的会话期

application servlet 正在执行的内容

out 用来传送回应的输出

config servlet的构架部件

page JSP网页本身

exception 针对错误网页，未捕捉的例外

2.jsp有哪些动作?作用分别是什么?

答:JSP共有以下 6 种基本动作

jsp:include：在页面被请求的时候引入一个文件。

jsp:useBean：寻找或者实例化一个JavaBean。

jsp:setProperty：设置JavaBean的属性。

jsp:getProperty：输出某个JavaBean的属性。

jsp:forward：把请求转到一个新的页面。

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

3.forward 和redirect的区别

答：forward是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。

4.JSP中动态INCLUDE与静态INCLUDE的区别？

答：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" %>

5.两种跳转方式分别是什么?有什么区别?

答：有两种，分别为：

<jsp:include page="included.jsp" flush="true">

<jsp:forward page= "nextpage.jsp"/>

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

6.JSP的内置对象及方法。

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

servlet笔试题目

1.说一说Servlet的生命周期?
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接口的init,service和destroy方法表达。 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。

2.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。

3.Servlet的基本架构
答：

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

4.什么情况下调用doGet()和doPost()？
答：Jsp页面中的form标签里的method属性为get时调用doGet()，为post时调用doPost()。

5.servlet的生命周期

答：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。

6.如何现实servlet的单线程模式

答：<%@ page isThreadSafe="false"%>

7. 页面间对象传递的方法

答：request，session，application，cookie等

8.四种会话跟踪技术

答：会话作用域ServletsJSP 页面描述

page否是代表与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（可以带有任何的 include 指令，但是没有 include 动作）表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面

request是是代表与 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 forward 指令和 include 动作的关系）

session是是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求

application是是代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域

9.Request对象的主要方法

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

10.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？

答：

```
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;
}
```

11.Servlet执行时一般实现哪几个方法？

答：

```
public void init(ServletConfig config)
public ServletConfig getServletConfig()
public String getServletInfo()
public void service(ServletRequest request,ServletResponse response)
public void destroy()
```

12.说出数据连接池的工作机制是什么？

答：J2EE服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

13.Class.forName的作用?为什么要用?

答：调用该访问返回一个以字符串指定类名的类的对象。

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：

<!--＃i nclude file="data.inc"-->

但是如果 data.inc 中包含 JSP CODE ，我们可以使用：

<%@include file="data.inc"%>

2、如何执行一个线程安全的 JSP?

只需增加如下指令

<%@ page isThreadSafe="false" %>

3、JSP 如何处理 HTML FORM 中的数据?

通过内置的 request 对象即可，如下：

```
<%
String item = request.getParameter("item");
int howMany = new Integer(request.getParameter("units")).intValue();
%>
```

4、在 JSP 如何包含一个静态文件?

静态包含如下：<%@ include file="copyright.html" %>

动态包含如下：<jsp:include page="copyright.html" flush="true"/>

5、在 JSP 中如何使用注释?

主要有四中方法：

1。<%-- 与 --%>

2。//

3。/**与**/

4。<!--与-->

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：

```
<%
response.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);
String newLocn="/newpath/index.html";
response.setHeader("Location",newLocn);
%>
```

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：

```
<%
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
%>
```

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：

```
<%
Cookie mycookie = new Cookie("aName","aValue");
response.addCookie(mycookie);
```

```
%>
```

9、在 JSP 中如何删除一个 COOKIE?

```
<%
Cookie killMyCookie = new Cookie("mycookie", null);
killMyCookie.setMaxAge(0);
killMyCookie.setPath("/");
response.addCookie(killMyCookie);
%>
```

10、在一个 JSP 的请求处理中如何停止 JSP 的执行

如下例：

```
<%
if (request.getParameter("wen") != null) {
// do something
} else {
return;
}
%>
```

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：

```
<%!
public String howBadFrom(HttpServletRequest req) {
HttpSession ses = req.getSession();
...
return req.getRemoteHost();
}
%>
<%
out.print("in general,lao lee is not baddie ");
%>
<%= howBadFrom(request) %>
```

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

```
<%@ page session="true" %>
<%
Integer num = new Integer(100);
session.putValue("num",num);
String url =response.encodeURL("hello2.jsp");
%>
<a href="/<";%=url%>>hello2.jsp</a>
```

hello2.jsp

```
<%@ page session="true" %>
```

```
<%
Integer i= (Integer )session.getValue("num");
out.println("Num value in session is "+i.intValue());
%>
```

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。

```
<%@ page import="sun.net.smtp.SmtpClient, java.io.*" %>
<%
String from="ybwen@sina.com";
String to="hewenjun@yeah.net, lei@who.com.cn";
try{
SmtpClient client = new SmtpClient("mail.xxxxx.xxx");
client.from(from);
client.to(to);
PrintStream message = client.startMessage();
message.println("To: " + to);
message.println("Subject: Sending email from JSP!");
message.println("This was sent from a JSP page!");
message.println();
message.println("Cool! :-)");
message.println();
message.println("Good Boy");
message.println("Im in genius.com");
message.println();
client.closeServer();
}
catch (IOException e){
System.out.println("ERROR SENDING EMAIL:"+e);
}
%>
```

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 request,HttpServletResponse response) {
try {
```

```
// do something
} catch (Exception ex) {
try {
sendErrorRedirect(request,response,"/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

```
<%@ page import="javax.naming.*, javax.rmi.PortableRemoteObject,
foo.AccountHome, foo.Account" %>
<%!
//定义一个对 SessionBeanHome 接口实例的全局引用
AccountHome accHome=null;

public void jspInit() {
//获得 Home 接口实例
InitialContext cntxt = new InitialContext( );
Object ref= cntxt.lookup("java:comp/env/ejb/AccountEJB");
accHome = (AccountHome)PortableRemoteObject.narrow(ref,AccountHome.class);
}
%>
<%
//实例化 SessionBean
Account acct = accHome.create();
//调用远程方法
acct.doWhatever(...);
//  如此等等
%>
```