1. JSP：**Java Server Pages**。可以理解为：JSP=HTML + JAVA 代码片段

2.**设定 Eclipse 中 JSP 的编码为 UTF-8**，以及换行符为 unix 格式。

3.**JSP 编译运行**。当第一次加载 JSP 页面时，因为要将 JSP 文件转换为 Servlet 类，所以响应速度较慢。

当再次请求时，JSP 容器就会直接执行第一次请求时产生的 Servlet，而不会再重新转换 JSP 文件，所以其执行速度较快。

4.JSP 页面组成

JSP 代码放在特定的标签中，然后嵌入到 HTML 代码中。

开始标签、结束标签和元素内容 三部分统称为 **JSP 元素（Elements）**。

JSP 元素可分成三种不同的类型：

◦脚本元素（Scripting）

◦指令元素（Directive）

◦动作元素（Action）

注意：JSP 的标签必须要完结。

5.三种元素的说明：

A.脚本元素 规范 JSP 网页所使用的 Java 代码，

包括：HTML 注释、隐藏注释、声明、表达式和脚本段。

B.指令元素 是针对 JSP 引擎的，并不会直接产生任何看得见的输出。

包括：inc1ude 指令、page 指令和 taglib 指令。

C.动作元素 利用 XML 语法格式的标记来控制 Servlet 引擎的行为。

6.JSP 注释：

A.html 的注释 ----发送到客户端，在浏览器右键源码中可见

<!--注释[<%=表达式%>]-->

B.jsp 隐藏的注释 ----不发送到客户端，不可见

<%--注释--%>

C.java 隐藏注释 ----客户端也看不见

存在于<% %>标签中的 Java 单行注释和多行注释

7.在 JSP 程序中用到的变量和方法是需要声明的,声明的语法如下：核心格式:**<%! ....%>**

<%! 声明；[声明；]...%> //必须以；结尾

例如：

<%! int i=6;%>

<%! int a,b,c;double d=6.0;%>

<%! Date d=new Date(); %>

8.JSP 的表达式

是由变量、常量组成的算式，它将 JSP 生成的数值嵌入 HTML 页面，用来直接输出 Java 代码的值。

表达式的语法规则如下： **<%= 表达式 %>**

注意：

A.不能用一个分号(“；”)来作为表达式的结束符。

B.“<%=”是一个完整的标记，中间不能有空格。

C.表达式元素包含任何在 Java 语言规范中有效的表达式。

说明，在开发中要尽量使用表达式输出代替 out.println()输出！

9.JSP 脚本段（Scriplet）是一段 Java 代码。可以包含任意行合法的脚本语句，脚本段是一个代码片断，在服务器处理请求过程中被执行。其语法规则如下：

<% 代码 %>

10.会在 Eclispe 中调出这些片段小提示。在光标处插入对应的标签。

11.include 指令：

在 JSP 网页中插入其他的文件，有两种方式：

◦**include 指令**

include 指令称为文件加载指令，可以将其他的文件插入 JSP 网页，被插入的文件可以是 JSP 文件、HTML 文件或者其他文本文件，但是必须保证插入后形成的新文件符合 JSP 页面的语法规则。

其指令形式如下：

<%@ include file=”相对地址” %>

include 指令只有一个属性：file。

◦**jsp:include 动作。**

12.**page 指令称为页面指令**，几乎在所有 JSP 页面顶部都会看到 page 指令。

```
<%@ page language=”脚本语言”
   extends=”继承的父类名称”
   import=”导入的 java 包或类的名称”
   session=”true/false”
   buffer=”none/8kB/自定义缓冲区大小”
   autoflush=”true/false”
   isThreadSafe=” true/false”
   info=”页面信息”
   errorPage=”发生错误时所转向的页面相对地址”
   isErrorPage=”true/false”
   contentType=”MIME 类型和字符集”
%>
```

import

◦用来导入将要用到的一个或多个包/类，例如：

<%@ page import=” java.util.Date” %>

<%@ page import=" java.util.*" %>

isErrorPage：这个属性的默认值为 “false”

isErrorPage 用来指定目前的 JSP 网页是否是另一个 JSP 网页的错误处理页，通常与 errorPage 属性配合使用。特别说明：如果页面错误的提示内容没有使用 exception 对象，值为 false 也是可以的，但是要使用 exception 对象，必须为 true。

注意：

A.在一个页面中可以使用多个<%@ page %>指令，分别描述不同的属性

B.每个属性只能用一次，但是 import 指令可以多次使用。

C.<%@ page%>指令区分大小写。

可以在 Tomcat 的安装目录/conf/web.xml 中查看到所有已知的 MIME 类型。

13.**taglib 指令**

用来定义一个标记库以及标记的前缀，其语法规则如下：

<%@ taglib uri=" URIToLibrary" prefix=" 标记前缀" %>

14.JSP 动作：

JSP 动作功能：

A.JSP 动作元素用来控制 JSP 引擎的行为

B.可以动态插入文件、重用 JavaBean 组件、导向另一个页面等。

JSP 动作元素：----注意大小写敏感

jsp:include 动作：在页面得到请求时包含一个文件。
jsp:forward 动作：引导请求者进入新的页面。
jsp:plugin 动作：连接客户端的 Applet 或 Bean 插件。
jsp:useBean 动作：应用 JavaBean 组件。
jsp:setProperty 动作：设置 JavaBean 的属性。
jsp:getProperty 动作：获取 JavaBean 的属性并输出。

15.jsp:include 动作

在即将生成的页面上**动态的插入文件**，它在页面运行时才将文件插入，对被插入文件进行处理。也就是说它是在页面产生时插入文件，其语法如下：

<jsp:include page=”文件相对路径” flush=”true”/>或

<jsp:include page=“文件相对路径” flush=“true”>

<jsp:param name=“参数名 1” value=“参数值 1”/>

<jsp:param name=“参数名 2” value=“参数值 2”/>

…

</jsp:include>

<jsp:include page="文件相对路径" flush=“true”/>或

<jsp:include page="文件相对路径" flush=“true”/>

<jsp:include page="文件相对路径" value=“参数值 1”/>

</jsp:include>

说明：flush 属性值为布尔型，若值为 false 表示这个网页全部读进来之后才输出，用户体验就不好；通常设为 true，buffer 满了就输出。

被包含页面可以使用 request.getParameter()方法进行参数接收！

16.include 指令和 jsp:include 动作区别：

A．**include 指令是静态的**，是在 JSP 文件被转换成 Servlet 的时候引入文件，它把被插入文件插到当前位置后再进行编译

B．**jsp:include 动作是动态的**，插入文件的时间是在页面被请求的时候。JSP 引擎不把插入文件和原 JSP 文件合并成一个新的 JSP 文件，而是在运行时把被插入文件包含进来。

注意：如果包含页面时需要传递参数，则只能使用 jsp:include 动作

特别说明：被包含的页面如果有图片等标签，当图片不能显示时，要特别注意，当前路径的基准是外面的 JSP 页面，而不是被包含页面的位置基准！

建议优先使用动态包含！

17.jsp:forward 动作 ----服务器跳转

用于停止当前页面的执行，转向另一个 HTML 或 JSP 页面。

在执行中 JSP 引擎不再处理当前页面剩下的内容，缓冲区被清空。

在客户端看到的是原页面的地址，而实际显示的是另一个页面的内容。

```
<jsp:forward page="文件名"/>或
<jsp:forward page="文件名">
<jsp:param name="参数名 1" value="参数值 1"/>
<jsp:param name="参数名 2" value="参数值 2"/>
…
</jsp:forward>
```

服务器跳转：客户端浏览器的地址栏显示路径不变！

18.jsp:useBean 动作

◦jsp:useBean 动作用来装载一个将要在 JSP 页面中使用的 JavaBean。

它创建一个 JavaBean 实例并指定其名字和作用范围。

◦实际工程中常用 JavaBean 做组件开发，而在 JSP 中只需要声明并使用这个组件，这样可以较大限度地实现静态内容和动态内容的分离，这也是 JSP 的优点之一。

在 JSP 中实例化一个 bean 的最简单的方法如下：

<jsp:useBean id="bean 的名称" scope="有效范围" class="包名.类名" />

scope 属性的取值有四种： page, request, session 和 application，默认值是 page。或者：

<jsp:useBean id="bean 的名称" scope="有效范围" class="包名.类名" >

实体

< /jsp:useBean >

实体的内容可以是：

◦合法的 JSP 程序代码

◦ <jsp:setProperty>和<jsp:getProperty>标签

◦一般的 HTML 代码

注意：这种实例化形式下，只有当第一次实例化 bean 时才执行实体部分，如果是利用现有的 bean 实例则不执行实体部分。

jsp:useBean 并非总是意味着创建一个新的 bean 实例。

JSP 引擎根据 useBean 中 id 属性指定的名字，在一个同步块中，查找内置对象 pageContext 中是否包含该 id 指定的名字和 scope 指定的作用域的对象。

如果该对象存在，JSP 引擎把这样一个对象分配给用户。

如果不存在则创建新的 bean 实例。

19.一个类作为 JavaBean 需要符合的要求：

A.类必须放在包中，在 web 中没有包的类是不存在的；
B.类必须声明为 public 的；
C.必须要有无参构造方法，无参构造方法可以是显示定义或者隐式继承；
D.对类的属性使用 private 封装，提供 setter 和 getter 方法；特殊情况下可以不提供 setter 方法！

20.jsp:setProperty 动作

用来设置已经实例化的 bean 对象的属性， jsp:setProperty 动作的

几种语法规则

第一种是直接将属性值设置为字符串或表达式，形如：

<jsp:setProperty name= “bean 的名称” property= “bean 的属性名称” **value= “属性值” />**

第二种方法用 request 的参数值来设置 JavaBean 的属性值，request 参数的名字和 JavaBean 属性的名字可以不同，其语法规则如下：

<jsp:setProperty name= “bean 的名称” property= “bean 的属性名称”

param= “request 参数的名字” />

第三种方式：使用名称自动匹配

```
<jsp:setProperty name="book" property="bookName" />
<jsp:setProperty name="book" property="bookNum" />
```

第四种方式：使用*通配符：

```
<jsp:setProperty name="book" property="*" />
```

21.jsp:setProperty 动作的用法

A.可以**在 jsp:useBean 元素的外面**使用 jsp:setProperty：

< jsp:useBean id="myName" ... / >

...

< jsp:setProperty name="myName" property="someProperty" ... / >

此时，不管 jsp:useBean 是找到了一个现有的 bean，还是新创建了一个 bean 实例，jsp:setProperty 都会执行。

B.把 jsp:setProperty **放入 jsp:useBean 元素的内部**，如下所示：

< jsp:useBean id="myName" ... >

...

< jsp:setProperty name="myName" property="someProperty" ... / >

< /jsp:useBean >

此时，jsp:setProperty 只有在新建 bean 实例时才会执行，如果是使用现有实例则不执行 jsp:setProperty。

22.jsp:getProperty 动作

用来获取 beans 的属性值，将其转换成字符串，然后输出。其语法规则如下：

<jsp:getProperty name=“bean 的名称” property=“bean 的属性名称”/>

注意：jsp:setProperty 动作和 jsp:getProperty 动作必须与 jsp:useBean 动作一起使用，不能单独使用。

23.在 Eclispe 中运行使用 JavaBean 的 JSP，不一定需要将编译后的 Bean 的 class 文件放在 WEB-INF/classes 下；见下图：

JavaBean 的删除：JavaBean 属于哪个范围对象就调用哪个范围的内置对象的 removeAttribute()方法删除！

request.removeAttribute()

24.如何处理 JSP 页面中的代码<% %>包裹问题？即何时包裹，何时不包裹。

可以认为是:

A.输出指令不用包裹；B.<jsp:... 指令不需要包裹。C.HTML 的标签不需要包裹；

25.JSP 的 9 个内置对象：---容器自动实例化，不需要用户去实例化，可以直接使用。

pageContext---JSP 页面容器、request---得到用户的请求信息、

response---服务器向客户端的响应信息、session---用来保存一个用户的信息

application---表示所有用户的共享信息、config---服务器配置，可以取得 web.xml 中的初始化参数

out---页面输出、page---表示从页面中表示出来的一个 Servlet 实例

exception---表示 JSP 页面所发生的异常，在错误页中才起作用，且错误页的 isErrorPage 的属性必须为 true！

26.JSP 提供的 4 种属性保存范围：

A.page 只在页面中保存属性，跳转之后则无效；**---使用 pageContext 表示**

B.request 只在一次请求中保存属性，客户端跳转无效，但是服务器跳转后依然有效；

C.session 在一次会话中保存，无论何种跳转都可以使用，但是新开的浏览器无法使用；

D.application 在整个服务器上保存，所有用户都可以使用。

27.request 的常用方法：

表 6-5 request 内置对象的常用方法

No.	方 法	类 型	描 述
1	public String getParameter(String name)	普通	接收客户端发来的请求参数内容
2	public String[] getParameterValues(String name)	普通	取得客户端发来的一组请求参数内容

续表

No.	方 法	类 型	描 述
3	public Enumeration getParameterNames()	普通	取得全部请求参数的名称
4	public String getRemoteAddr()	普通	得到客户端的 IP 地址
5	void setCharacterEncoding(String env) throws UnsupportedEncodingException	普通	设置统一的请求编码
6	public boolean isUserInRole(String role)	普通	进行用户身份的验证
7	public Httpsession getSession()	普通	取得当前的 session 对象
8	public StringBuffer getRequestURL()	普通	返回正在请求的路径
9	public Enumeration getHeaderNames()	普通	取得全部请求的头信息的名称
10	public String getHeader(String name)	普通	根据名称取得头信息的内容
11	public String getMethod()	普通	取得用户的提交方式
12	public String getServletPath()	普通	取得访问的路径
13	public String getContextPath()	普通	取得上下文资源路径

28.response 的常用方法：

表 6-6 response 对象的常用方法

No.	方 法	类 型	描 述
1	public void addCookie(Cookie cookie)	普通	向客户端增加 Cookie
2	public void setHeader(String name,String value)	普通	设置回应的头信息
3	public void sendRedirect(String location) throws IOException	普通	页面跳转

说明：

A.response.setHeader（"refresh"，＂2”）； //设置两秒一刷新

B.response.setHeader("refresh”，” 3;URL=hello.htm”） ;// 3 秒后跳转到 hello.htm

定时的时间如果为 0 则为立刻跳转。----**定时跳转为客户端跳转**

超链接跳转也是客户端跳转！

C.response.sendRedirect（”hello.htm”）； //直接跳转到 hello.htm

----此跳转也是**客户端跳转！**

29.Cookie 的常用方法：

表 6-7 Cookie 定义的常用方法

No.	方 法	类 型	描 述
1	public Cookie(String name,String value)	构造	实例化 Cookie 对象，同时设置名称和内容
2	public String getName()	普通	取得 Cookie 的名称
3	public String getValue()	普通	取得 Cookie 的内容
4	public void setMaxAge(int expiry)	普通	设置 Cookie 的保存时间，以秒为单位

```
Cookie c1 = new Cookie(”name",”zhangsan”); //定义新的 Cookie 对象
Cookie c2 = new Cookie(”age”,"25"); // 定义新的 Cookie 对象
c1.setMaxAge(60) ; //Cookie 保存 60 秒
c2.setMaxAge(60) ; //Cookie 保存 60 秒
response.addCookie(c1) ; / /向客户端增加 Cookie
response.addCookie(c2) ; / /向客户端增加 Cookie
```

30.session 对象是 javax.servlet.http.HttpSession 接口的实例化对象，

表 6-10 HttpSession 接口的常用方法

No.	方 法	类 型	描 述
1	public String getId()	普通	取得 Session Id
2	public long getCreationTime()	普通	取得 session 的创建时间
3	public long getLastAccessedTime()	普通	取得 session 的最后一次操作时间
4	public boolean isNew()	普通	判断是否是新的 session（新用户）
5	public void invalidate()	普通	让 session 失效
6	public Enumeration getAttributeNames()	普通	得到全部属性的名称

31.application 对象是 javax.servlet.ServletContext 接口的实例化对象，从单词上翻译表示的

是整个 Servlet 的上下文， ServletContext 代表了整个容器的操作

表 6-12 ServletContext 接口的常用方法

No.	方 法	类 型	描 述
1	String getRealPath(String path)	普通	得到虚拟目录对应的绝对路径
2	public Enumeration getAttributeNames()	普通	得到所有属性的名称
3	public String getContextPath()	普通	取得当前的虚拟路径名称

例如：

```
String path = application.getRealPath(”/”); //得到当前虚拟目录下对应的真实路径
```

String path= this.getServletContext().getRealPath("/"); //得到当前虚拟目录下对应的真实路径-----**尽量使用 this.getServletContext()来代替 application 对象。**

32.config 对象是 javax.servlet.ServletConfig 接口的实例化对象，主要的功能是取得一些初
始化的配置信息。

表 6-13 ServletConfig 接口的常用方法

No.	方 法	类 型	描 述
1	public String getInitParameter(String name)	普通	取得指定名称的初始化参数内容
2	public Enumeration getInitParameterNames()	普通	取得全部的初始化参数名称

所有的初始化参数必须在 web.xml 中配置，即如果一个 JSP 文件要想通过初始化参数取得一些信息， 则一定要在 web.xml 文件中完成映射。

33.out 对象是 javax.servlet.jsp.JspWriter 类的实例化对象， 主要功能就是完成页面的输出操作，使用 println（）或 print（）方法输出，但是从实际的开发来看，直接使用 out 对象的几率较小，

表 6-14 out 对象的其他操作

No.	方 法	类 型	描 述
1	public int getBufferSize()	普通	返回 JSP 中缓冲区的大小
2	public int getRemaining()	普通	返回 JSP 中未使用的缓冲区大小

34.pageContext 对象是 javax.servlet.PageContext 类的实例，主要表示一个 JSP 页面的上下文，在此类中除了之前讲解过的属性操作外，还定义了如表 6-15 所示的一些方法。

表 6-15 pageContext 对象的方法

No.	方 法	类 型	描 述
1	public abstract void forward(String relativeUrlPath) throws ServletException,IOException	普通	页面跳转
2	public void include(String relativeUrlPath) throws ServletException,IOException	普通	页面包含
3	public ServletConfig getServletConfig()	普通	取得 ServletConfig 对象
4	public ServletContext getServletContext()	普通	取得 ServletContext 对象
5	public ServletRequest getRequest()	普通	取得 ServletRequest 对象
6	public ServletResponse getResponse()	普通	取得 ServletResponse 对象
7	public HttpSession getSession()	普通	取得 HttpSession 对象