jsp也是j2ee服务器端执行的java组件。只不过语法不是Java语言的语法。jsp文件就是在html文件中内嵌java代码。

例如 time.jsp

Now: <%= new java.util.Date() %>

这个文件放在web应用的某个文件夹下，浏览器端就可以直接访问。

第一次访问jsp，响应慢是正常的，因为它要转换为java源文件，然后再编译成class文件，然后再部署运行。那么以后再访问就不会慢了。

jsp继承了servlet，所以，它也可以部署，也有配置信息。只不过，它更加方便，可以不用部署。

如果要部署，那么可以在web.xml文件中，声明该jsp组件。

```
<servlet>
        <servlet-name>a</servlet-name>
        <jsp-file>/a.jsp</jsp-file>
</servlet>
```

servlet-mapping元素我就不写了，你可以为jsp组件分配任何合法的访问路径（url-pattern)

jsp文件内嵌java代码，语法包括，jsp 指令，jsp 声明，jsp 表达式， jsp 注释， jsp 脚本

<%@ %> jsp 指令的语法。

<%= %> jsp 表达式的语法。

<%! %> jsp 声明的语法。

<%－ －%> jsp 注释，是两个-，不是一个破折号。两个减号。

<% 合法的Java 代码 %> jsp 脚本的语法。这个java代码片段最终会转换到service 函数内。

Jsp和servlet一样，都是服务器端的组件，也是处理请求和响应的。所以，以后可以不用写servlet类了。servlet编写麻烦，需要类的定义，函数的定义，而jsp有隐含的、预定义的9个变量（考试内容）。

HttpServletRequest request
HttpServletResponse response
HttpSession session
ServletConfig config
ServletContext application
JspWriter out
PageContext pageContext
page, 等价于 this
Exception exception

exception在错误处理页面中存在。

考试重点是JspWriter类，PageContext类。

举个例子, welcome.jsp

```
<%
   String username = session.getAttribute(“username”);
   out.println(username);
%>
```

这个代码比servlet方便。

假如 在B.jsp 中，使用<%@ include file=”A.jsp”%> ，那么就是把 A.jsp 的内容 原封不动 引入到 B.jsp 中。
<jsp:include page="a.html"/>//这是动态包含
对于静态包含，<%@include%>，中包含的文件，只是简单的嵌入到主文件中，就是在jsp页面转化成Servlet时才嵌入到主文件中，因为运行的结果是只生成了一个Servlet。

而对于动态包含<jsp:incude>，如果被包含文件是动态的，那么就会生成两个Servlet，也就是被包含文件也要经过jsp引擎编译执行生成一个Servlet，两个Servlet通过request和reponse进行通信。如果被包含的文件是静态的，那么这种情况和<%@include>就很相似，只生成了一个Servlet，但是他们之间没有进行简单的嵌入，而依然是通过request和reponse进行的通信。

Jsp 指令包括

<%@ page 属性列表 %>
<%@ include file=“afile” %>
<%@ taglib uri = “” prefix=“a” %>

例如：
<%@ page import=“java.util.*, java.net.*, java.io.*". session=“true” isErrorPage=“false” errorPage=“/error.jsp” %>
<%@ include file=“copyrights.html” %>

Jsp action包括
<jsp:include page=“/uri” />
<jsp:forward page=“/uri” />

在后面指定的作用域中找类型为SomeClazz的变量名为a的变量，如果没有则新建一个该变量

<jsp:useBean id=“a” class=“SomeClazz” scope=“page|request|session|application”/>
<jsp:getProperty name=“a” property=“age”/>
<jsp:setProperty name=“a” property=“age” value=“18”/>
<jsp:setProperty name=“a” property=“age” param=“age”/>
<jsp:setProperty name=“a” property=“*”/>

获取指定Javabean对象的属性值。

第四种形式:

<jsp:setProperty name = "JavaBean实例名"

property = "propertyName" param = "request对象中的参数名"

/>

param指定用哪个请求参数作为Bean属性的值。Bean 属性和request参数的名字可以不同。如果当前请求没有参数，则什么事情也不做，系统不会把null传递给Bean属性的set方法。因此，你可以让Bean自己提供默认属性值，只有当请求参数明确指定了新值时才修改默认属性值。

例如，下面的代码片断表示：如果存在numItems请求参数的话，把numberOfItems属性的值设置为请求参数numItems的值；否则什么也不做。

<jsp:setProperty name="orderBean" property="numberOfItems" param="numItems" />

下面是一个简单的例子:

另外，jsp也支持简化的表达式语言(EL)，使得输出数据更加方便。
${ express }

举个例子，a.jsp
<%
request.setAttribute(“a”, “hello”);
%>

${ a }

那么这个文件运行输出hello

脚本变量从page,request,session,application这四个作用域自动寻找，如果没有定义，就什么输出也没有，不报错。
那么变量是怎么“定义”的呢?
pageContext.setAttribute(“var”, new Integer(5));
request.setAttribute(“var”, new Integer(6));
session.setAttribute(“var”, new Integer(7));
application.setAttribute(“var”, new Integer(8));

EL表达式预定义了11个对象，除了pageContext对象是PageContext类型（考试重点），其余都是Map类型！！！（考试必考）

pageContext
param
paramValues
header
headerValues
cookie
initParam
pageScope
requestScope
sessionScope

applicationScope

一共十一个预定义对象。
既然是表达式，那么一定会有运算。大家自行学习支持的运算。
百度百科 EL表达式。

EL表达式中的存取运算符，也是必考的。两个存取运算符分别是：　.　[ ]

上面的代码还可以修改为：

举个例子，a.jsp
```
<%
  request.setAttribute(“a”, “hello”);
%>

${ requestScope.a }
```

其中${requestScope.a}和${a}的区别，大家知道吧?