

JSP网络 开发技术

◆ 杨 珏 卢银娟 赵昊彤 等 编著

使用JSP开发动态Web网站

从安装配置、基础知识到高级应用，全面解析JSP

从Java、Servlet、JavaBeans和JDBC入手，深入学习JSP

大量丰富、生动的实例，展现JSP强大功能

人民邮电出版社
www.pptph.com.cn



00117438

TP393.092

318



JSP 网络

开发技术

◆ 杨珏 卢银娟 赵昊彤 等 编著



人民邮电出版社



北航 C0535747

JSP10/30 v2

内容提要

本书详细讲述了使用 JSP 进行网络开发的方法和应用技巧。全书共包括 13 章, 分为 5 部分。第一部分主要讲述了 JSP 的基本概念和简单的使用, 使读者对 JSP 的应用及特点有一个初步的认识; 第二部分讲述了 JSP 的一些基本功能, 包括基本的语法、语句和内置对象; 第三部分讲述了如何在 JSP 中使用 Java 的对象, 这是灵活使用 JSP 的关键所在; 第四部分讲述了使用 JSP 进行数据库开发的方法, 数据库开发是网络开发的主要任务之一; 最后一部分通过几个网络开发中典型的实例综合讲述了 JSP 的知识要点和应用方法。

本书内容丰富、条理性比较强, 适合从事网络开发工作的技术人员参考阅读。

JSP 网络开发技术

◆ 编 著 杨 珏 卢银娟 赵昊彤 等
责任编辑 张立科

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ pptph.com.cn
网址 <http://www.pptph.com.cn>
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本: 787 × 1092 1/16
印张: 27 75
字数: 696 千字 2001 年 3 月第 1 版
印数: 1 - 6 000 册 2001 年 3 月北京第 1 次印刷

ISBN 7-115-09106-4/TP·2069

定价: 39.00 元

前 言

JSP (Java Server Pages) 是由 Sun 公司于 1999 年 6 月推出的新技术, 是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术, 是 Servlet 2.1 API 的扩展。利用这一技术可以建立先进、安全和跨平台的动态网站, 这个技术还在不断地更新和优化中。

在 Sun 正式发布 JSP 之后, 这种新的 Web 应用开发技术很快引起了人们的关注。JSP 具有与平台无关的优点, 为创建高度动态的 Web 应用提供了一个独特的开发环境。JSP 能够适应市场上包括 Apache WebServer、IIS4.0 在内的 85% 的服务器产品。

用 JSP 技术进行网络开发十分方便和快捷。在传统的网页 HTML 文件中加入 Java 程序片段和 JSP 标记, 就构成了 JSP 网页 (*.jsp)。Web 服务器在遇到访问 JSP 网页的请求时, 首先执行其中的程序片段, 然后将执行结果以 HTML 格式返回给客户。程序片段可以操作数据库、可以重新定向网页、可以发送 E-mail 等等。JSP 具有动态页面与静态页面分离, 能够脱离硬件平台的束缚, 以及编译后运行等优点而大大提高了其执行效率, 逐渐成为因特网上的主流开发工具。

JSP 与 Microsoft 的 ASP 技术非常相似。ASP 的编程语言是 VBScript 或 JavaScript, JSP 使用的是 Java。在 ASP 下, VBScript 代码被 ASP 引擎解释执行; 在 JSP 下, 代码被编译成 Servlet 并由 Java 虚拟机 Java Virtual Machine (JVM) 执行, 这种编译操作仅在对 JSP 页面的第一次请求时发生。事实证明, Java Servlet 是一种开发 Web 应用的理想构架。JSP 以 Servlet 技术为基础, 又在许多方面作了改进, 使用起来更加方便灵活。利用跨平台运行的 JavaBeans 组件, JSP 为分离处理程序逻辑与显示样式提供了比较理想的解决方案。JSP 必将成为 ASP 技术的有力竞争者。

本书在编写时本着循序渐进、由浅入深、易学易用、实用性强的原则, 力求达到既能帮助初学者入门, 也能使 JSP 高手进一步提高编程水平的目的。本书全面介绍了如何利用 JSP 进行网络开发, 内容涉及了 JSP 多方面的知识、技术和应用。

参加本书编写的人员包括: 杨珏、卢银娟、张建强、赵昊彤、何旭洪、朱志杰、陈薇、乔志宏、邵屹、栗强、李楠舟、秦伟、刘宝华、邢庆子、谢国锋、袁亚玓、程卫峰、闫华文、徐旻、林勇、宋征、刘韬、吴频、程凡、岳进、王远、王保东、黄成崑、钱力鹏。

由于作者的水平有限, 加上时间紧促, 书中错误之处在所难免, 请广大读者批评指正。

编者

2000 年 12 月

目 录

第一章 JSP 的基础知识.....	1
1.1 JSP 产生的背景	1
1.1.1 WWW 的历史.....	1
1.1.2 HTML 的局限性.....	2
1.1.3 交互的 Web 应用程序	3
1.2 JSP 简介	4
1.2.1 什么是 JSP	4
1.2.2 JSP 的基本思想	4
1.2.3 JSP 的设计目标	5
1.3 JSP 与 ASP、PHP 的比较.....	7
1.3.1 常用动态网页语言简介	7
1.3.2 技术特点比较.....	7
1.3.3 JSP 的优势	10
1.3.4 JSP、ASP 和 PHP 的前景分析	12
第二章 JSP 的工作环境.....	13
2.1 JSP 的运作方式	13
2.2 JSP 环境的安装	14
2.2.1 安装 TCP/IP 通信协议.....	15
2.2.2 在 Windows 98 下安装服务器.....	18
2.2.3 JSDK 服务器参数设置.....	24
2.2.4 在 Windows 2000 或 NT 下安装服务器	24
2.2.5 在 Linux 或 Unix 下安装服务器	25
2.3 JSP 的开发工具	27
2.3.1 网页开发的工具	27
2.3.2 JSP 编辑工具	28
第三章 从一个简单的例子来认识 JSP	30
3.1 建立 JSP 程序执行环境	30
3.1.1 启动 JSP 引擎	30
3.1.2 配置 JSDK 服务器.....	31
3.2 编辑 JSP 页面	32
3.2.1 生成 HTML 文档.....	32
3.2.2 编辑 JSP 程序代码	33
3.2.3 运行 JSP 程序	35

第四章 JSP 基本语句	36
4.1 JSP 语法概述	36
4.1.1 JSP 的编译指示语句	36
4.1.2 JSP 的脚本语言	37
4.1.3 内置对象 (Implicit Objects)	38
4.1.4 动作 (Actions)	38
4.2 注释语句	39
4.2.1 发送注释到客户端	39
4.2.2 服务器端的注释	41
4.3 JSP 中的声明	42
4.3.1 声明变量	42
4.3.2 给一个变量赋值	44
4.4 include 指令	47
4.4.1 包含静态文档	47
4.4.2 包含动态文档	50
4.4.3 <jsp:include />指令	51
4.5 page 指令	52
4.6 taglib 指令	54
4.6.1 JSP 与 XML	54
4.6.2 taglib 指令	55
4.7 动作指令	56
4.7.1 forward 指令	56
4.7.2 useBean 指令	56
4.7.3 plugin 指令	59
第五章 JSP 语法基础	62
5.1 数据类型	62
5.1.1 标识符和保留字	62
5.1.2 数据类型概述	62
5.1.3 整型数据	64
5.1.4 浮点型 (实型) 数据	65
5.1.5 字符型数据	66
5.1.6 布尔型数据	68
5.1.7 各种类型数据间的混合运算	68
5.2 运算符和表达式	69
5.2.1 运算符概述	69
5.2.2 算术运算符	69
5.2.3 关系运算符	71
5.2.4 布尔逻辑运算符	72

5.2.5	位运算符	74
5.2.6	赋值运算符	78
5.2.7	条件运算符	80
5.2.8	表达式	81
5.3	控制语句	81
5.3.1	分支语句	82
5.3.2	循环语句	89
5.4	使用数组	96
5.4.1	一维数组	96
5.4.2	多维数组	102
第六章	JSP 的内置对象	105
6.1	request 对象	106
6.1.1	获取用户提交信息	106
6.1.2	其他常用的方法	118
6.2	response 对象	124
6.2.1	使用 cookie	124
6.2.2	向客户端发送信息	125
6.2.3	处理 HTTP 文件头	127
6.2.4	response 对象的其他方法	128
6.3	out 对象	131
6.4	session 对象	132
6.4.1	session 对象的方法	132
6.4.2	使用 session 对象的方法	133
6.5	application 对象	142
6.5.1	application 对象的方法	142
6.5.2	用 application 对象制作的留言板	143
第七章	使用 Java 对象	149
7.1	面向对象的程序设计	149
7.1.1	Java 程序设计的特点	149
7.1.2	类和对象	150
7.1.3	包和接口	158
7.1.4	JSP 和 Java 方法	162
7.2	Math 类的方法	163
7.2.1	数学常量	163
7.2.2	随机方法	164
7.2.3	数学计算	170
7.3	字符串对象	177
7.3.1	字符串对象	177

7.3.2	处理字符串的 String 对象	179
7.3.3	创建 StringBuffer 类字符串对象	193
7.3.4	StringBuffer 类处理字符串	193
7.4	使用数据类型对象	201
7.4.1	字符类对象的方法	201
7.4.2	对用户的输入进行判别	205
7.5	建立自己的方法和对象	208
7.6	JSP 中的例外处理	210
7.6.1	例外处理	210
7.6.2	运行时例外	211
7.6.3	捕获例外	212
7.6.4	抛弃例外	214
第八章	文件的输入、输出处理	216
8.1	输入、输出流	216
8.1.1	什么是流	216
8.1.2	流控制类	216
8.1.3	InputStream 和 OutputStream	217
8.2	处理文件	220
8.2.1	建立 File 类对象	220
8.2.2	目录和文件的生成	220
8.2.3	文件和目录操作	226
8.3	访问文本文件	233
8.3.1	顺序读文件	233
8.3.2	顺序写文件	235
8.3.3	随机访问文件	238
8.4	以字符流的方式访问文件	244
8.4.1	Reader 对象	244
8.4.2	Writer 对象	245
第九章	数据库编程基础	248
9.1	数据库系统	248
9.1.1	关系数据库系统	248
9.1.2	用 ODBC 创建数据源	249
9.2	使用 SQL Server 7.0	252
9.2.1	SQL Server 7.0 的特点	252
9.2.2	SQL Server 7.0 使用简介	253
9.3	使用 SQL 语言	254
9.3.1	SQL 语言简介	254
9.3.2	使用 SQL 从表中取记录	255

9.3.3	操作多个表	256
9.3.4	操作字段	257
9.3.5	排序查询结果	258
9.3.6	取出不同的目录	259
9.3.7	用 SQL 创建新表	259
9.3.8	字段类型	260
9.3.9	字段属性	263
9.3.10	删除和修改表	264
9.3.11	建立索引	265
9.3.12	用 SQL 建立索引	266
9.3.13	SQL 核心语句	267
9.3.14	其他常用的 SQL 表达式、函数和过程	271
9.3.15	操作日期和时间	275
第十章	使用 JDBC 进行数据库编程	279
10.1	JDBC 简介	279
10.1.1	JDBC 的特点	279
10.1.2	JDBC 的两种主要接口	281
10.1.3	JDBC 产品	281
10.1.4	JDBC-ODBC 桥驱动程序	282
10.2	简单的例子	283
10.2.1	使用 JDBC-ODBC 桥	283
10.2.2	mysql 的 jdbc 驱动程序	287
10.3	JDBC 接口	289
10.3.1	java.sql.Connection 接口	289
10.3.2	java.sql.DriverManager 类	292
10.3.3	java.sql.Statement 接口	294
10.3.4	java.sql.ResultSet 接口	297
10.3.5	java.sql.PreparedStatement 接口	300
10.3.6	java.sql.CallableStatement 接口	303
10.4	映射 SQL 和 Java 类型	305
10.4.1	SQL 数据类型映射为 Java 类型	305
10.4.2	JDBC 类型	306
10.4.3	映射示例	310
10.4.4	动态数据存取	312
10.5	JSP 数据库编程实例	313
第十一章	使用 JavaBeans	320
11.1	JavaBeans 基础	320
11.1.1	JavaBeans 简介	320

11.1.2	Beans 开发工具包简介	325
11.1.3	BeanBox	326
11.2	在 JSP 中使用 Bean	330
11.2.1	用 JAR 文件包装 Bean	330
11.2.2	构造简单的 Bean	332
11.2.3	在 JSP 中使用 Bean	335
11.2.4	在 JSP 中得到和改变 Bean 属性	337
11.3	进一步使用 JavaBeans	342
第十二章	JSP 和 Servlet	348
12.1	Servlet 概述	348
12.1.1	什么是 Servlet	348
12.1.2	Java Servlet 特点及应用范围	349
12.1.3	Servlet 的结构	350
12.2	开发 Servlet 程序	351
12.2.1	Servlet 开发环境	351
12.2.2	Servlet 开发过程	352
12.2.3	处理表单数据	355
12.2.4	读取 HTTP 请求头	359
12.2.5	HTTP 应答状态	361
12.2.6	设置 HTTP 应答头	366
12.3	JSP 与 Servlet	370
第十三章	综合实例	372
13.1	用户管理和记事本	372
13.1.1	程序的结构	372
13.1.2	建立数据库和数据源	372
13.1.3	用户登录注册页面	373
13.1.4	用户注册程序	374
13.1.5	用户登录程序	381
13.1.6	记事本程序	384
13.2	发送 E-mail	389
13.2.1	程序结构	389
13.2.2	定制 E-mail 页面	390
13.2.3	发送 E-mail 的页面	391
13.2.4	Email.java	393
13.3	网上交友	393
13.3.1	程序结构	394
13.3.2	创建 friend 数据库和数据源	394
13.3.3	程序代码	395

13.3.4 运行效果	407
13.4 聊天室程序	410
13.4.1 程序结构	410
13.4.2 登录聊天室	412
13.4.3 聊天室程序	418
13.4.4 离开聊天室程序	431

第一章 JSP 的基础知识

1.1 JSP 产生的背景

Internet 在其短暂的历史中获得了巨大的成功,原因就在于它为人们提供了广泛和快速的信息。World Wide Web (WWW 万维网)就是其中最常用的一种网络资源。

Internet 的一个优势是其平台独立性。网络服务器和它的用户能够使用各种 Internet 服务来相互通信,而不用管他们使用的平台是 DOS、Unix、Linux 或者是各种版本的 Microsoft Windows。在 Internet 上漫游时,用户通常不用知道服务器的情况。对于客户端来说,所需的只是通过拨号或指定线路与 Internet 连接和拥有客户端的软件(如用于 World Wide Web 界面的网页浏览器 Internet Explorer 或 Netscape)。

1.1.1 WWW 的历史

WWW 是 Internet 上最为流行的信息检索服务,它把各种类型的信息(静止图像,文本,声音和影像)有机地集成起来,供用户阅读、查找,是一种基于超链接(Hyperlink)的超文本(Hypertext)系统。WWW 已经实现的部分,给计算机网络上的用户提供了一种手段,以简单的方式去访问各种媒体。用来访问 WWW 的最流行的浏览器软件 Internet Explorer 和 Netscape,改变了人们观察和创建信息的方法,为 Internet 的发展注入了强有力的生命力。

早在 1945 年, Vannevar Bush (Hypertext 之父)提出了最早的超文本的雏形,之后很多人都研究了不同的超文本系统;直到 1989 年, CERN (欧洲粒子物理实验室)的科学家 Tim Berners-Lee 首先提出了 WWW 的概念,并将它作为高能物理学界的科学家传输新方法和新成果的工具;1990 年末,第一个 WWW 软件在计算机上实现了;1992 年, CERN 正式发布了 WWW;1992 年, Marc Andreessen 等人写了 NCSA MOSAIC 程序,使 WWW 浏览器的发展日渐成熟;到 1994 年,在 Internet 上传送的 WWW 数据首次超过 FTP 的数据量,并一跃成为访问 Internet 资源的最流行的手段。

用浏览器去访问某一个服务器,是一种浏览器/服务器(Browser/Server)的工作模式。基于浏览器客户端的应用程序相对于传统的基于客户机/服务器(Client/Server)的应用程序而言有几个好处。这些好处包括了几乎没有任何限制的客户端的访问和极其简化的应用程序部署及管理(要更新一个应用程序,管理人员只需要更改一个基于服务器的程序,而不是成千上万一个安装在客户端的应用程序)。这样,软件工业迅速地向建造基于浏览器客户端的多层次应用程序迈进。

WWW 以超文本传输协议 HTTP (Hypertext Transport Protocol) 和 HTML 为基础,允许用户在客户机上发出请求,在服务器和浏览器之间传输超媒体信息。浏览器的作用就是把从

服务器传回的超媒体信息展现在用户面前，它知道如何去解释和显示在 WWW 上找到的 HTML 文件。此外，大多数浏览器都可以自动调用其他应用程序（Helper Applications Program），以显示特殊类型的文档，如 audio 或者 video 的文件格式。

1.1.2 HTML 的局限性

随着 Web 应用的飞速发展，传统的静态 HTML 语言已经越来越不适应需要。由此导致了人们开始在传统的静态 HTML 上增添新的功能，这一切新功能最终导致了动态 HTML 的产生。

HTML 为开发信息文档提供了独立的平台。利用 HTML 可以方便地将相互独立的文本文件和图形文件综合形成 Web 页。

的确，动画 GIF 可以提供 Web 页上的动画效果，但必须事先在静态区指定对象移动路径。DHTML 使用户可以通过加载来操作运动单元。开发复杂的用户接口和布局是极其繁琐的：如果采用单像素控件技术，就不得不使用布局微调控制。HTML 出现以来，Web 设计受到三方面的制约：文本格式、布局控制和动态改变文档内容的能力。

● 文本格式的局限

任何试图扩展文件格式的人都被 HTML 的格式限制弄得焦头烂额。这个问题从 Web 页一开始就存在。建立 Web 页是为了更快、更方便地向客户提供信息。HTML 没有显示布局和定位控制的功能，而这些恰好是设计精美可视化产品必不可少的。HTML 有一个深刻的教训，当用户需要显示黑体字符时，必须使用标志。虽然可以用光标来选择那些标志，但使用起来很不方便，就使它的可视化不那么完美。

绝望之余，设计者不得不在 Web 设计时玩一些小把戏。大多数是用单像素控制技术对齐 Web 页上的单元，另外一些人则用表格设计一些网格，但两者都不是很好的解决方法，HTML 的缺陷已搅扰设计者很长时间了。更令人难堪的是，设计者无法控制 WWW 的字体，而这也是用户使用浏览器浏览 Web 页必不可少的。如果没有这种控制，各浏览器对 Web 页的影响就有细微的差别，如果人们使用表格、缩进这样的 HTML 技巧创建最好的布局，就使得测试格式变得混乱且不可读。

● 布局控制的限制

在格式方面，静态 HTML 使设计者无法控制文本内容的定位和格局。虽然 HTML 可以很方便地组合文本和图形，但不具备报纸、杂志那样漂亮的布局。要实现图像定位以及文本环绕图像这样的格式几乎是不可能的。设计者可以使用<TABLE>标志，经过繁琐的过程，使 HTML 具有一些布局上的特征，但这一过程往往是极其乏味的。

● 对动态改变内容无能为力

由于不允许在加载后更改页面的内容，HTML 实际上也限制 Web 页的内容。事实上，问题不仅如此。静态 HTML 的 Web 页不具备使浏览器接受布局改变的机制。例如，当用户在浏览时改变窗口大小，文本被刷新，但图像的大小并不随之改变。

例如，音乐爱好者希望从 Web 页上看到近期音乐会的座次表和舞台设计。利用静态 HTML 可以绘出座次的 GIF 和 JPEG 图像。但 DHTML 还可以提供座次布局，用户单击座次，就会出现本座次所能看到的舞台景象。DHTML 可以使用户在本地机上进行交互式操作，而不必使用复杂的服务器。

能够动态改变网页的内容是 DHTML 的重要特性。传统 HTML 也能提供同样的功能,例如,用图像映像代表座位,当用户单击这一区域时,图像数据便传送到 CGI 进行编排,这样便在浏览器上显示出从所选择座次看到的舞台图像。但要完成这些功能,本地机必须与服务器进行多次交互式操作,这势必会产生性能瓶颈。

1.1.3 交互的 Web 应用程序

随着网络技术的高速发展,Web 已经从一个大部分显示静态信息的网络演化成了能够进行股票交易和购书操作等的一种基础设施的网络。在各种各样的应用程序中,对于可能使用的基于 Web 的客户端,看上去没有任何限制。这些基于 Web 的应用程序要求开发技术上的不断改进。静态 HTML 对于显示相对静态的内容是不错的选择;新的挑战在于创建交互的基于 Web 的应用程序,用户在浏览信息的同时也需要向服务器反馈一些信息,在这些程序中,页面的内容是基于用户的请求或者系统的状态,而不是预先定义的文字。

对于这个问题的一个早期解决方案是使用 CGI-BIN 接口;开发人员编写与接口相关的单独的程序,以及基于 Web 的应用程序,后者通过 Web 服务器来调用前者。这个方案存在严重的扩展性问题——每个新的 CGI 要求在服务器上新增一个进程。如果多个用户并发地访问该程序,这些进程将消耗掉该 Web 服务器所有的可用资源,使系统性能降低到极其低下的地步。

某些 Web 服务器供应商已经尝试通过为他们的服务器提供“插件”和 API 来简化 Web 应用程序的开发难度。但是这些解决方案是与特定的 Web 服务器相关的,例如,微软的 Active Server Pages (ASP) 技术使得在 Web 页面上创建动态内容更加容易,但是也只能工作在微软的 IIS 和 Personal Web Server 上。

还存在其他的解决方案,但是都不能使一个普通的页面设计者轻易地掌握。例如使用像 Java Servlets 这样的技术,就可以使得用 Java 语言编写交互应用程序的服务器端的代码变得很容易。一个 Java Servlet 就是一个基于 Java 技术的运行在服务器端的程序(与 Applet 不同,后者运行在浏览器端)。开发人员能够编写出这样的 Servlet,以接收来自 Web 浏览器的 HTTP 请求,动态地生成响应(可能要查询数据库来完成这项请求),然后发送包含 HTML 或 XML 文档的响应到浏览器。

采用这种方法,整个网页必须都在 Java Servlets 中制作。如果开发人员或者 Web 管理人员想要调整页面显示,就不得不编辑并重新编译该 Java Servlet,即使在逻辑上已经能够运行了。采用这种方法,生成带有动态内容的页面仍然需要应用程序的开发技巧。

很显然,目前所需要的是一个业界范围内的创建动态内容页面的解决方案。这个方案将解决当前方案所受到的限制,即能够在任何 Web 或应用程序服务器上运行、将应用程序逻辑和页面显示分离、能够快速地开发和测试、简化开发基于 Web 的交互式应用程序的过程。

Java Server Pages (JSP) 技术就是被设计用来满足这样的要求的。

1.2 JSP 简介

1.2.1 什么是 JSP

JSP 的英文全称为 Java Server Pages, 是由 Sun Microsystems Inc. 公司倡导, 由多家公司参与建立的一种动态网页技术标准。

与 ASP (Active Serve Pages)、PHP 一样, JSP 在动态网页的建设中具有强大而特别的功能。SUN 公司在 Java 的基础下开发出的 JSP 具有动态页面与静态页面分离, 能够脱离硬件平台的束缚, 以及编译后运行等优点而大大提高了其执行效率, 逐渐成为因特网上的主流开发工具。

JSP 规范是 Web 服务器、应用服务器、交易系统以及开发工具供应商间广泛合作的结果。SUN 公司开发出这个规范来整合和平衡已经存在的规范, 对 Java 编程环境(例如, Java Servlet 和 JavaBeans)进行支持的技术和工具。其结果是产生了一种新的、开发基于 Web 应用程序的方法, 给予使用基于组件应用逻辑的页面设计者以强大的功能支持。

1.2.2 JSP 的基本思想

在传统的网页 HTML 文件 (*.htm, *.html) 中加入 Java 程序片段 (Scriptlet) 和 JSP 标记 (tag), 就构成了 JSP 网页 (*.jsp)。Web 服务器在遇到访问 JSP 网页的请求时, 首先执行其中的程序片段, 然后将执行结果以 HTML 格式返回给客户。程序片段可以操作数据库、重新定向网页以及发送 E-mail 等, 这就是建立动态网站所需要的功能。所有程序操作都在服务器端执行, 网络上传送给客户端的仅是得到的结果, JSP 对客户浏览器的要求很低, 支持无 Plugin, 无 ActiveX, 无 Java Applet, 甚至无 Frame 的浏览器。

JSP 网页包含以下的组件:

- JSP 指示

JSP 页面使用 JSP 指示将指令传递到 JSP 引擎。其中包括: JSP 页面指示传递与页面相关的信息, 例如缓冲区和线程信息或者出错处理; 语言指示指定脚本语言以及所有的扩展; 包含指示可以被用来在页面中包含一个外部的文件, 被包含的文件可能是另一个 JSP 文件; 标识库指示指出页面可以调用的客户标识库类型。

- JSP 标识

绝大多数 JSP 处理将通过与 JSP 相关的基于 XML 的标识完成。JSP1.0 中包含大量标准标识, 这些标识作为核心标识包括:

`jsp:useBean` 这个标识声明对一个 JavaBeans 组件实例的使用。如果该组件的实例不存在, JavaBeans 组件将实例化和注册这个标识;

`jsp:setProperty` 这个标识在组件的实例中设置一个属性的值;

`jsp:getProperty` 这个标识获取一个组件的实例的属性值, 将其转化为字符串, 并且将它放入隐含对象 “out” 中;

`jsp:include` 和 `jsp:forward`。

1.1 版本将包含更多的标准标识。标识的好处在于它们易于在应用程序间使用和共享。基于标识的语法的真正威力来自于客户标识库的开发,使得工具供应商或其他人员能够为特定的要求创建和分派标识。

- 脚本元件

JSP 页面可以在页面中包含小的脚本,称之为小脚本 (scriptlets)。小脚本是一个代码片段,在请求的处理过程中被执行。小脚本可以和页面中的静态元件组合 (正如上面的例子一样) 起来创建动态生成的页面。

脚本在 “<%” 和 “%>” 标志中被描述。在这对标志中的所有东西都会被脚本描述语言引擎执行,在我们的例子中是主机上的 Java 虚拟机。

JSP 规范支持所有常用的脚本元件,包括表达式和声明。

- JSP 页面的应用模型

JSP 页面由 JSP 引擎执行,引擎安装在 Web 服务器或者使用 JSP 的应用服务器上。JSP 引擎接受客户端对 JSP 页面的请求,并且生成 JSP 页面给客户端的响应。

JSP 页面通常被编译成为 Java Servlet。后者是一个标准的 Java 扩展,在 www.java.sun.com 站点有更详细的描述。页面开发人员能够访问全部的 Java 应用环境,以利用 Java 技术的扩展性和可移植性。

当 JSP 页面第一次被调用时,如果它还不存在,就会被编译成为一个 Java Servlet 类,并且存储在服务器的内存中。这使得以后对该页面的调用有非常快的响应。(避免了 CGI-BIN 为每个 HTTP 请求生成一个新的进程的问题,或是在服务器端使用引用所引起的运行时的语法分析问题。)

JSP 页面可以包含在多种不同的应用体系结构或者模型中。JSP 页面可以应用于由不同协议、组件和格式所组成的联合体中。

例如在一个简单实现中,浏览器直接调用 JSP 页面,JSP 页面自动生成被请求的内容 (可能会调用 JDBC 直接从数据库中获取信息)。JSP 页面能够调用 JDBC 或者 Java Blend 组件来生成结果,并且创建标准的 HTML,作为结果发送回浏览器。

这个模型基本上用 JSP 页面 (编译成为 Java Servlet) 代替了 CGI-BIN 概念。这个方法拥有下列优点:

- 简单而快速地编程;

- 页面作者可以很容易地根据请求和资源状态生成动态内容。

这个结构在许多应用上工作良好,但不能扩展到这种情况,即大量的基于 Web 的并发客户访问稀少的企业资源。因为每个客户必须建立或者共享一个到可用内容资源的连接。例如,如果 JSP 页面访问数据库,可能生成许多到数据库的连接,这将影响到数据库的性能。

1.2.3 JSP 的设计目标

Java Server Page 技术主要是为了使动态网页的编写更容易、更简单。JSP 设计的目标主要有以下几点:

- 到处可执行

Java Server Page 技术是完全与平台无关的设计,包含它的动态网页与底层的 Server 元件设计。使用者可以在任何作业平台上编写动态网页,并且在任何支持 Java Serve Page 的服务

器系统上执行。当然，这些 JSP 的程序与浏览器端是独立的。

使用者还可以建立 Server 元件，并在 Java Server Page 中使用。目前主要是 JavaBeans 与 Java Servlet，而它们也都符合跨平台的规格。

- 加强元件能力

目前服务器端的动态网页所欠缺的主要就是完整而强大的元件模型，JSP 技术以 Java 的元件模型 JavaBeans 来加强了 Java Server Page 元件的使用能力。

这样将为使用者节省大量的开发时间，因为 JavaBeans 与 Servlet 是早已存在的技术，而元件可以完成多数的工作，这可以帮助开发者将网页设计与程序的编写分开，有助于程序的开发和维护。

- 适合 Java Enterprise 平台用户

Java Server Page 是 Java Enterprise 平台（Java 专注于企业应用方的平台技术，如 JDBC、JNDI、JINI 等）的高度兼容的部分。开发者可以利用 Java 的 Enterprise API 开发企业的各种需求系统，而 Java Server Page 正适合用来作为这些技术的前端。当需要升级程序时，开发者只要升级元件与动态网页部分，这些都存储在服务器中，所以只要修改服务器的资源，所有用户端浏览到的信息就会跟着改变了。

- 更容易建立动态网页

Java Server Page 是嵌入 HTML 文档的，它混合了标准的 HTML 语法与 JSP 自己的语法标识，并以 Java 作为其脚本语言。一个 JSP 文档通常都是以“.jsp”作为扩展名的，并使用服务器端可重用的元件。

- 与 XML 技术集成

JSP 页面可以用于生成 XML 和 HTML 页面。

对于简单的 XML 生成，开发的 JSP 页面中可以包含 XML 标识和 JSP 页面的静态模板部分。对于动态 XML 生成，使用基于服务器的对象和生成 XML 输出的客户化标识。

JSP 页面与 XML 工具是兼容的。Sun 在设计 JSP 规范时使得即使手工操作 JSP 页面也很容易，JSP 规范同时也提供了一个机制，用以创建任意 JSP 页面的一个 XML 版本。通过这种方法，XML 工具能够创作和操作 JSP 页面。

通过将 JSP 标识和元件转换为与 XML 对等的标识和元件，可以使用基于 XML 的工具来操作 JSP 页面。例如，脚本可以被包含在“<%”和“%>”之中，或者基于 XML 标志的“<jsp:scriptlet>”和“</jsp:scriptlet>”中。事实上，经过下面这样简单的几步将 JSP 页面转换为 XML 页面是可能的，这些步骤包括：

- 增加一个 JSP 根元件；

- 将元件和指示转换为 XML 对等的元件和指示；

- 为页面上其他的元件（通常为非 JSP 文件）创建 CDATA 元件。

通过这些与 XML 兼容的方法，创建 HTML 页面的设计者拥有了一个快速创建动态 Web 页面的应用环境。同时，基于 XML 的工具和服务可以与 JSP 页面集成并且和 JSP 兼容的服务器一起工作。