快 学 易 用 新 软 件 丛 书

Learn
them quickly
Apply
what you learn easily

快學易用

郭 健 编著
马圣超

JSP

北京邮电大学出版社
www.buptpress.com

前　言

就在几年前，架设一个能够和用户进行交互的动态网站还是一件不太容易的事情。你不得不使用当时仅有的 CGI 技术。CGI 技术不仅难于掌握，而且由于它的可移植性和代码的可重用性低下，你必须把大量的精力都花费在代码的调试和维护上。为了弥补 CGI 的种种缺陷，Sun 公司在 Java 技术的基础上推出了 Servlet 技术。相对于 CGI，Servlet 更容易开发和维护，运行效率更高，系统开销更小。更重要的是，由于 Servlet 是架筑在 Java 技术基础之上的，所以它承袭了 Java 的很多优点，比如面向对象、安全性和健壮性、良好的可移植性、代码可重用等。然而 Servlet 技术并没有实现程序逻辑与外观显示的分离，而这恰恰是后来 ASP、PHP 等技术的主要优势。于是，Sun 公司在 Servlet 技术的基础上，不失时机地推出了 JSP(JavaServer Pages)。和 ASP、PHP 一样，JSP 也是一种服务器端的动态脚本技术。但是由于 JSP 继承了 Java 和 Servlet 技术的优点，因此在安全性、稳定性、可移植性和运行效率等诸多方面，JSP 比 ASP 和 PHP 都更胜一筹。在国外，JSP 已经成为建设电子商务平台的主流技术；在国内，JSP 也在日益得到流行和普及。

随着 Web 技术的不断发展，传统的软件体系结构 Client/Server 逐渐被三层或多层的 Client/Server 结构所取代，而客户端被弱化为一个图形用户接口，成为一个瘦客户机。而这个图形用户接口最典型的就是一个浏览器，也就是说，越来越多的应用程序都会被移植到 Web 上来开发。所以不仅是网站，今后将会更多的应用程序开发会用到动态 Web 技术，而 JSP 将是你的最佳选择。

本书共分 13 章，循序渐进地讲述了 JSP 技术所涉及的方方面面，并结合大量实例介绍了 JSP 的技术细节和开发过程。本书涉及到了 Java、Java Servlet、JDBC、XML、Tag Library 等多种与 JSP 相关的技术。相信不论你是一个有经验的编程人员，还是 Web 编程的新手，都能从本书中获得帮助。

在本书编写的过程中，得到了很多人的热心帮助。马亚宁、唐本亭为本书的编写整理了大量的资料，罗卫国、周健成完成了文字校对工作，张启宇、刘光宇负责调试了本书中的所有代码。

由于时间仓促和限于作者水平，书中难免有疏漏和错误，恳请读者批评指正。

作　者

2001 年 2 月

目　录

第一章　初识 JSP

本章内容包括：

☆　第一个 JSP 程序
☆　动态 Web 技术概览
☆　JSP 技术的发展历史
☆　JSP 的工作机制
☆　为什么选择 JSP

1.1　第一个 JSP 程序

准备好了吗？我们现在就开始。先来看一个程序：

程序 1.1：helloworld.jsp

```
<%@ page language="java" %>
<html>
<head>
<title>Hello World!  </title>
</head>
<body>
<%
        String Msg = "My First JSP Program.";
        out.print("Hello World!");
%>
<h2> <%=Msg%> </h2>
</body>
</html>
```

这是一个简单的 JSP 程序，程序的功能是在浏览器中输出两行文本，运行效果如图 1.1 所示。

仔细观察一下这个程序，会发现在 HTML 代码中间多了一些包含在符号“<%”和“%>”之间的脚本。熟悉 ASP 或 PHP 的读者一定会觉得这种结构似曾相识。是的，JSP 与 ASP 和 PHP 一样，也是一种直接嵌入 HTML 的动态 Web 编程技术。符号“<%”和“%>”之间的部分就是 JSP 脚本，它所使用的脚本语言是 Java。

图 1.1 程序 helloworld.jsp 的运行结果

1.2 动态 Web 技术概览

Internet 技术的飞速发展,特别是 20 世纪 90 年代中后期 Web 技术在 Internet 上的广泛应用,正在使计算机软件业承受着一场巨大的冲击。传统的软件体系结构 Client/Server 模式已经不适应 Internet 应用的需要,这样,人们就将两层的 Client/Server 结构扩展为三层或多层的结构。典型的三层软件体系结构如图 1.2 所示。

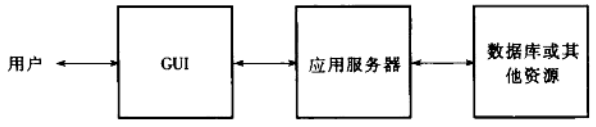

图 1.2 典型的三层软件体系结构

在这种模式中,应用逻辑完全转移到应用服务器上来完成,而客户端弱化成一个只提供用户接口的 GUI(图形用户界面),即瘦客户机。Web 技术正是基于这样一种模式发展而来的,它实际上采用的就是三层的 Browser/Server 结构。Web 服务器用来处理用户的请求,当用户通过浏览器向它发出请求时,Web 服务器经过分析和处理后把结果送回给客户浏览器。当然,这个过程当中有可能要进行 Web 服务器和数据库服务器或其他可访问资源的交互。

Web 技术发展的初期,它只是用作静态信息的发布,完成这个功能只需要简单的 HTML 语言,所以这个时期 HTML 发展迅速。Web 服务器将用户请求的页面直接发送给客户端而不需要做任何处理,HTML 代码的解释完全在客户浏览器中进行。但随着技术的发展,人们已经不再满足 Web 只是简单的信息发布平台,而是想将它作为一种计算平台,在这种平台下完成一些复杂的功能。于是,动态 Web 技术出现了。

动态 Web 技术的原理是:用户通过客户浏览器向 Web 服务器发出页面请求,Web 服务器根据请求文件的类型将该文件送交给相应的执行引擎。执行引擎开始扫描整个文件,运行其中的程序,并将运行的结果重新提交给 Web 服务器,然后由 Web 服务器将提交回的运行结果以 HTML 代码的形式送回客户端浏览器。这样服务器根据用户的需求动态地生成页面并送回给用户,于是就实现了客户端与服务器端的交互。

到目前为止,有以下几种主要的动态 Web 技术。

CGI

通用网关接口 CGI(Common Gateway Interface)是最早应用的动态 Web 技术,几乎所有的 Web 服务器都支持 CGI。当客户端向 Web 服务器发出运行 CGI 程序的请求时,Web 服务器会通过 CGI 接口创建一个网关进程来执行 CGI 程序,运行后的结果通过 Web 服务器返回到客户端。CGI 只是一种工作机制,其程序并不针对某种特定语言。CGI 程序可以使用 C/C++、Perl、Unix Shell、Visual Basic 等语言编写,但最常用的还是 Perl 语言,这得益于它拥有强大的字符串处理能力。CGI 技术虽然已经发展得很成熟了,但它仍存在着一些不可避免的问题:

- CGI 技术不易学习和掌握,而且程序编写起来非常繁琐。比如,页面中所有的 HTML 标记都需要用程序输出。
- CGI 程序的维护非常困难,代码的可重用性低下。
- Web 服务器必须为每一个 CGI 请求创建一个独立的进程,如果同时有多个用户请求,就会给服务器带来极大的负担。
- CGI 的进程不能常驻内存,只要有新的请求,Web 服务器就会重新建立一个新的进程,由于进程的创建与终止都需要系统开销,所以这样做的效率是非常低下的。
- 安全漏洞多,这是由无数网站被黑客攻击的事实证明了的。

服务器专用 API

服务器 API 是一些大公司提供的针对某种特定 Web 服务器而开发的一套面向 Internet 服务的 API 接口,实现了 CGI 的全部功能,并对其进行了扩展。比较常见的有 Netscape 公司的 NSAPI、Microsoft 公司的 ISAPI、Oracle 公司的 Web Server API 等。这里只讨论比较常见的 ISAPI,其他的也都大同小异。

ISAPI(Internet Server Application Program Interface)是 Microsoft 公司提供的,它的工作原理和 CGI 大体相同,都是首先通过交互式页面取得用户的输入信息,然后提交服务器进行后台处理。但是,二者的实现机制却截然不同:

- 用 ISAPI 创建的应用程序以动态链接库(DLL)形式存在。这些动态链接库被装入到 Web 服务器的内存,与 Web 服务器进程处于同一个系统空间。一个 ISAPI 的 DLL 可以在被用户请求激活后长驻内存,等待用户的另一个请求,直到 Web 服务器终止运行。
- 用 ISAPI 创建的应用程序是基于线程的。对于每一个来自浏览器的客户请求,Web 服务器都将为它创建一个线程,由该线程来完成和客户端的交互过程。由于创建线程在占用操作系统资源和时间方面都比创建进程少得多,所以 ISAPI 能极大地提高交互式 Web 应用程序的性能。

ISAPI 运行的速度要比 CGI 更快,功能也更强,比如它提供了过滤器应用程序接口。但它仍然有以下不足:

- ISAPI 更加难学,开发和维护也更复杂。
- 可移植性不好,只能运行在特定的平台。
- ISAPI 进程和服务器进程处于同一系统进程空间,因此,服务器端程序的一个违规操作就可能导致 Web 服务器的崩溃。

Java Servlet

Java Servlet(简称 Servlet)是运行在服务器端的 Java 小程序,能够像 CGI 一样动态地扩充 Web 服务器的功能。Servlet 的工作原理是:用户向 Web 服务器请求一个包含 Servlet 的页面,Web 服务器将该 Servlet 请求提交给 Servlet 执行引擎进行处理并将处理后的结果通过 Web 服务器再次送回客户端。

由于 Servlet 是用 Java 语言编写,所以它具有很多与生俱来的优点:

- Servlet 继承了 Java 的几乎所有优点:与平台无关、稳定安全、易开发等。
- Servlet 以 Java 字节码的形式存在,只有在 Servlet 第一次被请求的时候才会被装入。一旦被装入,它便运行于后台,对于以后的用户请求,Servlet 会创建一个线程来完成服务而无需重新装载 Servlet。这样一来就使得 Servlet 的运行速度很快(这有点像 ISAPI)。

Java Servlet 可以说是 JSP 技术的前身,与 JSP 关系密切。我们还会在第十二章中详细地讨论 Java Servlet。

ASP

ASP(Active Server Pages)是 Microsoft 公司推出的一种直接嵌入 HTML 的服务器端脚本技术。"服务器端脚本"是一种全新的思想,以后的 PHP 和 JSP 都是类似的技术。ASP 的工作原理是:用户通过客户浏览器请求 ASP 文件,Web 服务器将该文件交给 ASP 解释引擎,ASP 解释引擎读取整个文件,解释执行其中的脚本程序并将执行结果(HTML 代码)传回给用户浏览器。ASP 具有以下优点:

- ASP 脚本程序可以使用 VBScript 或 JavaScript 编写,这些语言相对简单,而且有一定的用户群,所以比较容易掌握和普及。
- ASP 实现了程序逻辑与外观显示的分离:用户可以首先设计 Web 界面,然后将主要的精力放到应用程序的开发上,接着将程序部分嵌入到事先设计好的 HTML 代码中。这样就使得 ASP 程序的开发变得更容易,维护也更加方便。
- 由于 ASP 是解释执行的,所以它不需要手工编译。
- ASP 内建有许多功能强大的对象,这些对象可以使开发者摆脱许多繁琐的工作。
- ASP 可以通过组件技术方便地进行功能扩充,这些组件可以是 ASP 自带的,也可以是第三方厂家开发的组件。
- ASP 本身提供了 ActiveX 数据库对象(ADO)组件,它使得 ASP 可以通过 ODBC 访问任何支持 ODBC 接口的数据库。

然而,ASP 也并不完美。ASP 只能运行于 Microsoft Windows/IIS/PWS 平台(UNIX 下可以通过 Chilisoft 插件来支持 ASP,但功能有限),而且它在稳定性和安全性上也不能让人满意。

PHP

和 ASP 相似,PHP(Professional Hypertext Preprocessor)也是一种运行于服务器端的动态脚本技术。它的工作原理和 ASP 大致相同,只不过脚本的解析使用的是 PHP 引擎。PHP 具有以下特点:

- 在语言方面,PHP 采用了类似 C 和 Perl 的语法格式和框架,易于入门,凡是拥有 C、Perl 编程经验的程序员都可以很快掌握。
- PHP 是免费的而且源代码也是公开的,任何用户都可以修改这些源代码,加入一些新的功能模块,所以它能够使 PHP 的功能不断扩充。

- PHP 的另一个优势是跨平台，它可以在 Microsoft Windows 平台和大多数 UNIX 平台上流畅地运行，支持多种不同平台上的 Web 服务器。
- 另外，PHP 的库函数丰富，广泛支持现有的多种数据库，这使得 PHP 的开发更加简单方便。

JSP

JSP(Java Server Pages)是由 Sun 公司倡导、许多公司一起参与而建立的动态 Web 技术，也是本书将要讨论的内容。JSP 由 Java Servlet 技术发展而来，它拥有 Servlet 技术一切优良的特性，却比 Servlet 的开发更加灵活方便。我们在第一节已经看到了一个 JSP 程序的小例子，相信大家已经有了一个初步的印象。下面我们就从头开始，来学习这个被认为是未来发展方向的动态 Web 技术。

1.3　JSP 技术的发展历史

JSP 的发展历程是这样的：

- 1998 年 4 月，Sun 公司发布 JSP 0.90 规范。
- 1999 年 1 月，发布 JSP 0.92 规范。
- 1999 年 11 月，发布 JSP 1.1 规范，同时推出 JSWDK 1.0.1 和 Java Servlet API 2.2。
- 2000 年 9 月，发布 JSP 1.2 规范和 Java Servlet API 2.3。

如果想了解最新的 JSP 动态，请访问 Sun 公司关于 JSP 的网站(如图 1.3 所示)，网址是：http://java.sun.com/products/jsp/

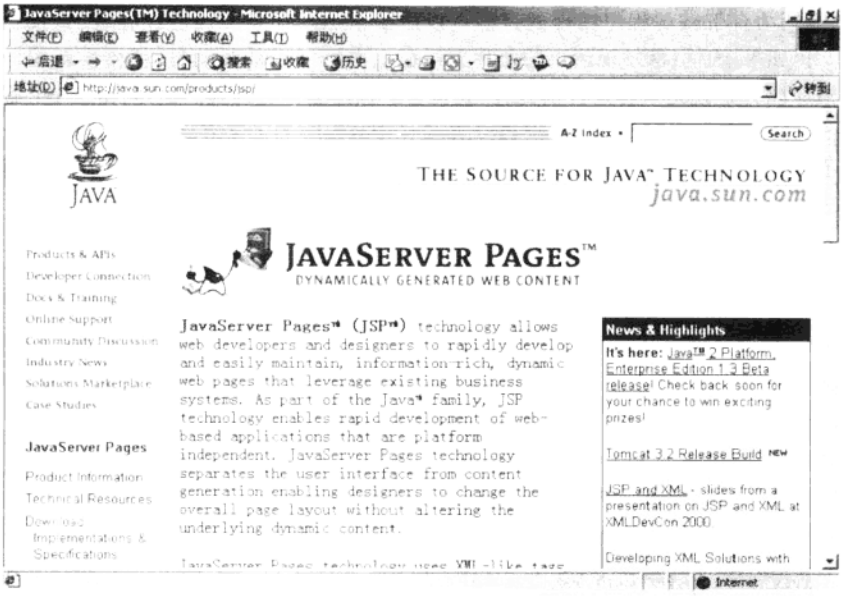

图 1.3　Sun 公司关于 JSP 的网站

1.4 JSP 的工作机制

JSP 是一种直接嵌入 HTML 的服务器端的动态脚本技术，它的工作方式与 ASP 和 PHP 相似，如图 1.4 所示。当用户请求一个 JSP 页面时，Web 服务器将该文件交给 JSP 引擎执行并将结果送回客户浏览器。但是，JSP 采取了完全不同的运行方式：编译运行。其过程如图 1.5 所示。

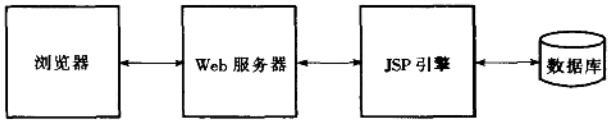

图 1.4 JSP 的工作方式

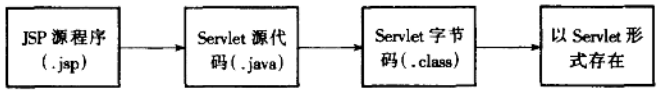

图 1.5 JSP 程序的运行过程

当用户向一个 JSP 文件接收第一个用户请求时，JSP 引擎首先将 JSP 程序转成 Servlet 源代码，接着用 Java 编译器将其编译成 Java 字节码，再由 Java 虚拟机解释执行该字节码并将执行后生成的 HTML 文本传给请求的客户浏览器。JSP 文件在第一次被请求时，会执行的比较慢，这是因为有 Java 源程序编译的过程。以后由于 Servlet 字节码已经存在，直接执行就可以了。但如果 JSP 引擎发现请求的 JSP 文件在上次编译后被改动过，就会重新编译该文件。严格地说，JSP 还是解释运行的，虽然它包含了程序编译的过程。但是，由于 JSP 解释执行的是 Servlet 字节码，这就比 ASP 和 PHP 直接解释源代码的工作方式效率要高多了。

为了让大家有个更直观的认识，我们来看一下第一节中的 helloworld 程序在服务器端自动转换成的 Servlet 源代码。

程序 1.2：0002fch_00031_0002fhelloworld_0002ejsphelloworld_jsp_0.java

```
package ch_00031;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Vector;
```

```
import org.apache.jasper.runtime.*;
import java.beans.*;
import org.apache.jasper.JasperException;
public class _0002fch_00031_0002fhelloworld_0002ejsphelloworld_jsp_0
        extends HttpJspBase {
    static {
    }
    public _0002fch_00031_0002fhelloworld_0002ejsphelloworld_jsp_0() {
    }

    private static boolean _jspx_inited = false;
    public final void _jspx_init() throws JasperException {
    }
    public void _jspService(HttpServletRequest request, HttpServletResponse re-
    sponse)
        throws IOException, ServletException {
        JspFactory _jspxFactory = null;
        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        String _value = null;
        try {
        if (_jspx_inited == false) {
            _jspx_init();
            _jspx_inited = true;
        }
        _jspxFactory = JspFactory.getDefaultFactory();
        response.setContentType("text/html;charset=8859_1");
        pageContext = _jspxFactory.getPageContext(this, request, response,
        "", true, 8192, true);
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
```

```
            out = pageContext.getOut( );
        // HTML
        // begin [file="C:\\tomcat\\webapps\\ROOT\\ch1\\helloworld.jsp";
        // from=(0,27);to=(6,0)]
            out.write("\r\n<html>\r\n<head>\r\n<title>Hello World!
                    </title>
                    \r\n</head>\r\n<body>\r\n");
        // end
        // begin [file="C:\\tomcat\\webapps\\ROOT\\ch1\\helloworld.jsp";
        // from=(6,2);to=(9,0)]
                String Msg = "My First JSP Program." ;
                out.print("Hello World!");
        // end
        // HTML
        // begin [file="C:\\tomcat\\webapps\\ROOT\\ch1\\helloworld.jsp";
        // from=(9,2);to=(10,4)]
            out.write("\r\n<h2>");
        // end
        // begin [file="C:\\tomcat\\webapps\\ROOT\\ch1\\helloworld.jsp";
        // from=(10,7);to=(10,10)]
                out.print(Msg);
        // end
        // HTML
        // begin [file="C:\\tomcat\\webapps\\ROOT\\ch1\\helloworld.jsp";
        // from=(10,12);to=(13,0)]
                out.write("</h2>\r\n</body>\r\n</html>\r\n");
        // end
    } catch (Exception ex) {
      if (out.getBufferSize( ) != 0)
            out.clearBuffer( );
      pageContext.handlePageException(ex);
    } finally {
                out.flush( );
                _jspxFactory.releasePageContext(pageContext);
        }
    }
}
```

这个程序存放在 Tomcat(有关 Tomcat 的内容会在第二章讨论)的 work 目录,如果你使用

Tomcat 来运行这个程序，就会在这个目录找到这个 .java 文件。

从这个文件中，我们可以清楚地看到一个 JSP 程序是怎样被“翻译”成 Servlet 源代码的。

代码中最重要的地方就是 _jspService(HttpServletRequest request, HttpServletResponse response)这个方法。JSP 引擎在处理每个客户请求时，会将针对这个请求的请求对象 request 和响应对象 response 作为参数传给 _jspService( )方法。_jspService( )方法再根据请求对象来获取客户端的信息并利用响应对象来设置传回的结果。

代码看不懂没关系，我们将在第十二章详细讨论 Java Servlet 的结构。

1.5 为什么选择 JSP

在国外，JSP/Servlet 已经成为架设电子商务网站的主流技术；而在国内，JSP/Servlet 技术也正日益普及和流行。为什么 JSP 会在众多的动态 Web 技术中脱颖而出呢？

JSP 是目前最优秀的动态 Web 技术，这么说并没有言过其实。

开放的源代码和开发过程

Sun 公司将以前开发 Java 时所用的社团性、开放性的开发过程移植到 JSP 的开发上来。Sun 授权了工具提供商(如 Macromedia)，结盟公司(如 Apache、Netscape)，最终用户，协作商及其他。最近，Sun 将最新版本的 JSP 和 Java™ Servlet(JSP 1.1 和 Java Servlet 2.2)的源代码发放给了 Apache，以求 JSP 与 Apache 紧密的相互发展。Apache、Sun 和许多其他的公司及个人公开成立一个健壮的咨询机构以便任何公司和个人都能免费取得信息。

直接嵌入 HTML

JSP 使用 HTML 或 XML 来设计 Web 文档的显示格式，再将 JSP 脚本直接嵌入 HTML 或 XML 代码，用以生成动态的内容。这样使得文档的显示格式和内容分离，也就简化了开发过程。程序员在不同的开发阶段可以将精力集中到不同的部分，而不是像以前那样设计程序时既要考虑程序逻辑，又要考虑页面的生成格式。另外一个好处就是程序的可维护性增强了。如果你只想改变页面的显示格式，而原有的程序逻辑还是可用的，那么你只需要修改当中的 HTML 代码就可以了；反之，如果你只想增加程序的功能，那么也只需修改嵌入的 JSP 代码。

服务器端运行的脚本

这是相对客户端脚本而言的。大家以前使用过 JavaScript，它就是一种完全的客户端的脚本语言。客户端脚本完成的功能很有限，它所有的程序逻辑都是由浏览器解析的，也正因为这一点，它需要把程序的源代码传给浏览器，这样你辛辛苦苦编写的代码就轻易地被别人“享用”了。JSP 程序运行于服务器端，由服务器中的 JSP 引擎解析其中的 JSP 脚本，只是将脚本运行后的结果传回给浏览器，JSP 的代码在客户端是不可见的。脚本在服务器端运行更易于与服务器端的各种资源进行交互，完成更加复杂的程序逻辑。

编译运行的机制

JSP 程序是经过编译后运行的，这种编译过程只在该 JSP 程序第一次被请求时发生，所以代码的执行效率较 ASP 和 PHP 有很大提高。虽然 JSP 程序只是被编译成字节码，运行时还需要由 Java 虚拟机解释执行，但它通过服务器端的 Cache 机制，使得字节码的访问效率得

到提升。

跨平台

JSP 的技术特点和工作机制保证了它的跨平台性。JSP 使用 Java 和 Servlet 技术作基础，而 Java 的最大优势就是跨平台性，这使得 JSP 是真正意义上的可跨平台，而不仅仅只是有广泛的不同平台上的服务器软件的支持。

使用 Java

JSP 使用 Java 作为脚本语言，它承继了 Java 几乎所有优秀的特性，诸如简单、稳定、安全、多线程等等。我们会在第三章详细地讨论 Java。

可重用的组件

JSP 通过 JavaBeans 来扩充程序的功能。JavaBeans 是一种可移植的、与平台无关的分布式组件模型。与其他组件模型如 Microsoft 公司的 COM/DCOM 相比，JavaBeans 更易于开发和维护。JavaBeans 通过把程序中需要进行复杂计算或完成某项特定功能的模块分离出来封装成 JavaBeans，而这些模块通常是可以复用的，这就大大简化了程序的开发过程。而且使用 JavaBeans 更安全，你可以将 JavaBeans 放置到不可访问的目录中。

可扩展标签库

JSP 能够定制标签库，使得 JSP 标签可以进行扩展。这是除了 JavaBean 技术以外另一个使得 JSP 组件化的技术。目前已有大量的第三方厂商开发的标签库可供免费使用。

数据库支持

JSP 可以通过 JDBC 访问任何支持 JDBC 接口的数据库。几乎所有的数据库厂商都开发了各种不同类型的 JDBC 驱动程序(JDBC 驱动程序有四种类型)。另外由 Sun 公司开发的 JDBC－ODBC bridge 使得 JSP 程序可以和所有带有 ODBC 驱动程序的数据库建立连接，这样 JSP 就能够支持几乎所有的数据库。

讲了这么多，相信你一定找到答案了。选择了 JSP，意味着你的开发将会变得轻松惬意。当然，你首先要学好这门技术。从下一章开始我们将从搭建 JSP 的开发环境讲起，逐步引导你轻松地掌握 JSP。