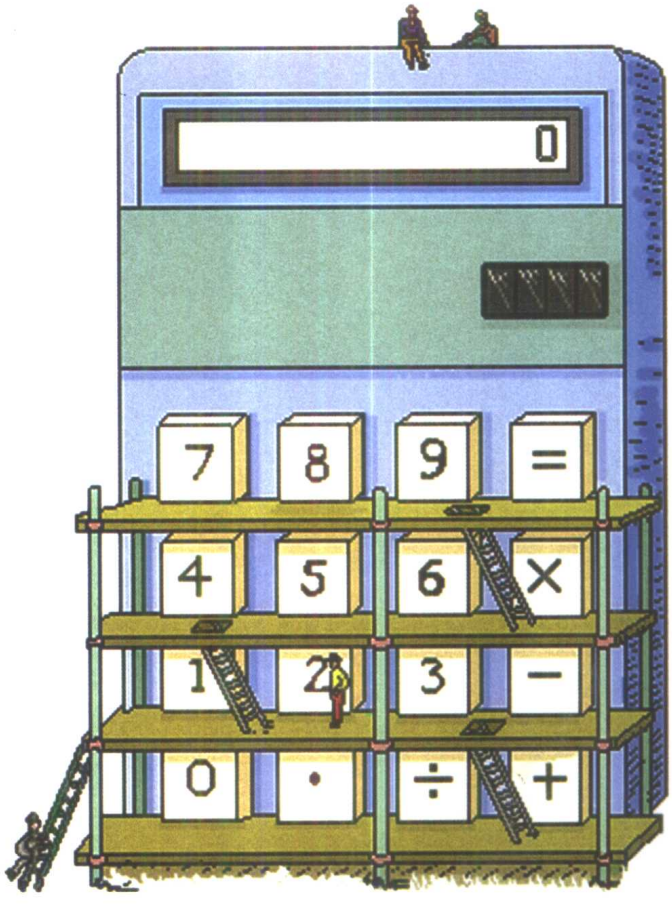
HZ BOOKS

Core JSP

Sun公司核心技术丛书

Sun microsystems

JSP核心技术



附赠 CD-ROM

(美) Damon Hougland
Aaron Tavistock 著

马朝晖 等译

机械工业出版社
China Machine Press

Prentice Hall PTR

Sun公司核心技术丛书

JSP核心技术

（美） Damon Hougland
Aaron Tavistock 著
马朝晖 等译

机械工业出版社
China Machine Press

本书深入介绍了JSP核心技术，包括JSP语法、Servlet与Bean之间的交互、应用程序的分割和部署等内容。全书浓缩精炼、简明实用，既适合Java程序员阅读，也可供其他有关人员学习参考。

本书版权登记号：图字：01-2001-0694

图书在版编目（CIP）数据

JSP核心技术/（美）霍兰德（Hougland, D.），（美）塔维斯托克（Tavistock, A.）著；马朝晖等译. -北京：机械工业出版社，2001.8
（Sun公司核心技术丛书）
书名原文：Core JSP
ISBN 7-111-09055-1

Ⅰ.J… Ⅱ.①霍… ②塔… ③马… Ⅲ.主页制作-技术，JSP Ⅳ.TP393.092

中国版本图书馆CIP数据核字（2001）第039509号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：张 庆
北京昌平奔腾印刷厂印刷·新华书店北京发行所发行
2001年8月第1版第1次印刷
787mm × 1092mm 1/16 · 16印张
印数：0 001-5 000册
定价：42.00元（附光盘）

序　言

在近几年中，大量的软件开发活动已经从客户迁移到了服务器。以客户为中心的模型（即，客户执行复杂的程序来显示和格式化数据）被认为不再适合于主要的企业应用程序。首要的原因是部署——将客户程序部署到大量的桌面，并且在应用程序发生变化时重新部署它们，是非常烦人的。取而代之的是，将应用程序重新设计为使用一个Web浏览器作为“终端”。应用程序本身驻留在服务器上，它将数据格式化为适合用户浏览的Web页面，并且处理用户填入Web表单的响应。

如果你打算开发一个Web应用程序，就需要选择一种具有几种重要功能的技术。你需要能够方便地产生大量动态HTML。你还需要访问数据库和其他资源。此技术必须提供一种有利于性能和稳定性的体系结构基础。最后，你必须能够以一种有利于未来的增长和可维护性的方式将你的程序逻辑分解。

最初的Web应用程序使用CGI（通用网关接口）机制和一些服务器端脚本，这些脚本一般是用Perl写的，偶而会用C、Python、PHP或其他语言。这个方法有很多问题。CGI机制的伸缩性很差，因为每个Web请求都会衍生一个新的服务器进程。进程间的通信（例如，为了共享数据库连接这样的资源）对于程序非常棘手。最后，外来的编程语言可能具有一定的特色，但是它们干不了“重活”。数据库访问或安全性这样的特性通常不是语言的一部分，而是由非标准的第三方库提供的。这使程序员不但受制于语言本身的实现者，而且还要看各种第三方库的提供者的“脸色”。

一段时间以来，Java程序员对Servlet的能力很满意，因为它在一定程度上解决了以上问题。Servlet是用Java写的，而Java是一种得到广泛支持的语言。Java具有数据库访问、连网、多线程、安全性等内置功能。每个Servlet都在它自己的线程中执行，这就避免了产生服务器进程的开销。Servlet可以轻松地共享会话状态和数据库连接这样的资源。Servlet主要的缺点是建立HTML很麻烦。所有HTML必须通过编程产生，即必须使用语句输出所有文本和标记。特别是，这意味着页面必须由程序员自己建立。我们都知道当程序员打算插手Web设计时会发生什么。

在近几年中，一个逐渐流行起来的方式是使用Web服务器脚本语言，如Netscape LiveWire和微软的ASP（Active Server Pages）。使用这些系统，程序员可以将代码片段嵌入Web页面。页面本身由Web设计人员专门设计。Web服务器在为页面服务时执行这些代码片段，并且将每个代码片段产生的HTML插入。这个方式的优点（也就是它流行起来的原因）是可以非常快速地得到结果。但是，这种开始时看起来很有趣而且高效率的方式后来成了维护方面的噩梦。当你将表示逻辑（HTML页面的静态部分）和业务逻辑（代码片段）混合在一起时，如果需求发生变化，改变页面会非常困难。在重新设计页面时，Web设计人员不知道如何移动代码。这使重新设计成为非常费事的活动，需要程序员与Web设计人员进行频繁的交流。最后，要记住：你被限制在某个Web服务器上了。例如，如果你用ASP开发了一个Web应用程序，而后来打算使用Apache，而不

是微软的IIS，那么你就有麻烦了。

这本书的主题——JSP技术解决了这些问题。JSP具有与Servlet一样的优点——实际上，JSP页面就是Servlet。你可以使用Java的全部能力来实现你的业务逻辑，而不是局限于某些脚本语言。通过使用Bean、XML转换和标记库，JSP使你可以分离表示逻辑和业务逻辑。例如，在一个结构良好的JSP应用程序中，同样的业务逻辑可以配用多种界面，这使你的用户可以选择使用正规的浏览器或使用WAP（无线访问协议）的移动电话。

本书教你如何用JSP建立稳固而且可伸缩的Web应用程序。它讨论了JSP语法、JSP从Servlet继承的特性（如会话管理）、Servlet和Bean之间的交互，以及大量有用的Java主题，如JDBC（Java数据库连接性）和XML。最后（也是最重要的），你将学习应用程序的分割和部署——这些主题帮助你创建经得起测试的稳固的应用程序，而不是粗制滥造的产品。

与其他书不同，本书采用完全以JSP为中心的方式，与Sun微系统公司在它的Java Enterprise规划中提出的建议相一致。这是非常合理的方案，而且很强大。其他书先讨论Servlet，而将JSP作为Web编程的第二种方法。本书告诉你为什么JSP页面在Web编程技术中处于较高的位置。JSP页面可以做Servlet所能做的任何事，而且JSP具有更高层的功能，使你能够将注意力集中于业务问题。在使用Servlet时，要想拥有这样的功能，就必须进行大量烦人的编程和组织工作。

秉承核心系列丛书的精神，本书包含许多现实的建议，这些建议在在线文档中没有。本书作者不会停留在烦人的语法和细节上。与许多计算机书籍作者不同，本书作者做了大量的工作，将精华与琐事分开了。你将不会浪费时间研究你用不到的特性，但是你会发现本书覆盖了建立实际应用程序时所需要的所有主题。我确信，你会发现本书真的非常有用。我希望你喜欢它，并且有机会使用它建立出色的Web应用程序。

Cay Horstmann
San Jose,August 2000

本书的翻译工作主要由马朝晖和陈美红承担，潘浩、楼涵、董小蕾、王悦、张磊、王志强、林志国参与了部分翻译、录入和校对工作。

目　录

序言
第1章　简介 ……1
1.1　Web的历史 ……1
1.1.1　静态页面 ……1
1.1.2　动态页面 ……2
1.2　JSP概述 ……4
1.2.1　模板页面 ……5
1.2.2　静态数据与动态元素 ……5
1.2.3　简单的JSP页面 ……6
1.2.4　JSP文件 ……6
1.3　Java的能力 ……7
1.3.1　只写一次，随处运行 ……7
1.3.2　Java API ……8
1.3.3　安全 ……8
1.3.4　可伸缩性 ……8
1.3.5　可扩充性 ……9
1.3.6　组件 ……9
1.4　了解HTTP ……9
1.4.1　HTTP基础 ……9
1.4.2　HTTP请求 ……9
1.4.3　HTTP响应 ……10
第2章　脚本元素 ……12
2.1　Scriptlet元素 ……12
2.2　Expression元素语法 ……13
2.3　Declaration元素语法 ……15
2.4　嵌入式流控制语句 ……15
2.4.1　判断语句 ……18
2.4.2　循环语句 ……20
2.4.3　异常语句 ……21
2.5　注释语法 ……21
2.6　应用脚本元素：Calendar.jsp ……23
第3章　动作和指令 ……31
3.1　Action元素语法 ……31
3.1.1　标准动作 ……32
3.1.2　JavaBean 动作 ……32
3.1.3　Resource 动作 ……35
3.2　指令 ……40
3.2.1　指令语法 ……40
3.2.2　page指令 ……40
3.2.3　include指令 ……43
3.2.4　tablib指令 ……44
第4章　JSP引擎内幕 ……45
4.1　幕后 ……45
4.1.1　重新编译 ……45
4.1.2　Servlet与JSP的关系 ……46
4.2　多线程和持久性 ……46
4.2.1　持久性 ……46
4.2.2　线程的危险 ……47
4.2.3　线程安全 ……48
4.3　隐含对象 ……49
4.3.1　out对象 ……49
4.3.2　request对象 ……50
4.3.3　response对象 ……51
4.3.4　pageContext对象 ……51
4.3.5　session对象 ……51
4.3.6　application对象 ……52
4.3.7　config对象 ……52
4.3.8　page对象 ……52
4.3.9　exception对象 ……52
4.4　JSP的生命期 ……52
4.4.1　jspInit ( ) ……53
4.4.2　jspDestroy ( ) ……53
4.4.3　JSP的生命期概述 ……53
4.4.4　使用jspInit ( ) 和jspDestroy ( )

的计数器 ……54
4.5 编译JSP ……55
4.6 JSP的性能调整 ……57
4.6.1 避免串联追加 ……58
4.6.2 小心使用synchronized ( ) ……58
第5章 获取信息 ……59
5.1 请求 ……59
5.2 HTTP请求和JSP ……60
5.2.1 参数 ……60
5.2.2 头信息 ……63
5.2.3 cookie ……68
5.2.4 属性 ……68
5.2.5 服务器信息 ……69
第6章 发送信息 ……70
6.1 使用response对象 ……70
6.1.1 HTTP状态 ……70
6.1.2 HTTP头 ……71
6.1.3 response对象的其他方法 ……73
6.2 设置cookie ……75
6.2.1 建立cookie ……75
6.2.2 发送cookie ……76
6.2.3 使用cookie ……76
6.3 处理错误 ……79
第7章 跟踪会话 ……83
7.1 在请求之间跟踪数据 ……83
7.1.1 隐藏的表单字段 ……83
7.1.2 JavaMail的简述 ……84
7.1.3 邮件表单示例 ……84
7.1.4 隐藏的图文框 ……88
7.1.5 重写URL ……89
7.1.6 cookie ……90
7.2 HttpSession API ……90
7.2.1 基础 ……90
7.2.2 使用session对象 ……91
7.2.3 会话的生命期 ……94
7.3 会话和身份 ……96
第8章 JavaBean ……103
8.1 组件模型 ……103
8.2 JavaBean ……104
8.2.1 JavaBean约定 ……104
8.2.2 其他需求 ……107
8.2.3 扩展特性 ……108
8.2.4 内部检查 ……108
8.2.5 在JSP中使用JavaBean ……109
8.3 验证JavaBean ……111
8.3.1 JNDI是什么 ……111
8.3.2 LDAP是什么 ……112
8.3.3 LDAPAuthBean ……112
8.4 组件和组件框架 ……117
8.5 企业JavaBean ……117
8.5.1 会话Bean和实体Bean ……118
8.5.2 企业JavaBean的编程限制 ……118
第9章 数据库连接性 ……119
9.1 什么是JDBC ……119
9.1.1 ODBC ……119
9.1.2 对象与关系型 ……120
9.2 理解关系型模型 ……120
9.3 JDBC ……125
9.4 JDBC和JSP ……128
9.5 JDBC和JavaBean ……139
第10章 JSP和XML ……157
10.1 XML基础知识 ……157
10.2 使JSP成为XML文档 ……160
10.3 XSL基础知识：XSL、XSLT和XPath ……163
10.3.1 学习XSLT ……164
10.3.2 XML分析程序和XSLT处理程序 ……165
10.4 XML与XSLT相结合 ……165
10.5 从JSP产生XML ……168
第11章 定制标记 ……175
11.1 定制标记的基本知识 ……175
11.1.1 定制标记的语法 ……175
11.1.2 播放程序 ……176
11.2 定制标记API ……178
11.2.1 用于定制标记的特殊方法和常量 ……178

11.2.2 理解标记库描述符文件 ……179
11.2.3 打包时需要考虑的特殊因素 ……180
11.3 创建新的定制标记 ……180
11.4 创建合成的标记 ……184
第12章 设计并部署JSP应用程序 ……192
12.1 通过分解来减少复杂性 ……192
12.1.1 构造JSP组件体系结构的方法 ……192
12.1.2 JSP设计模型 ……195
12.2 JSP应用程序的部署 ……199
12.2.1 ZIP、JAR、WAR和EAR ……199
12.2.2 对WAR和EAR文件的支持 ……200
12.2.3 WAR文件入门 ……201
12.2.4 Sun DeployTool ……204
12.2.5 最好的策略 ……207
第13章 未来 ……208
13.1 J2EE的影响 ……208
13.2 JSP和GUI工具 ……208
13.3 XML、XML、更多的XML ……209
13.4 前进方向 ……209
附录A JSP API快速参考 ……210
附录B JSP应用程序 ……223
附录C 示例数据库 ……232
关于CD-ROM ……246

第1章 简 介

本章主题：

- Web的历史。
- JavaServer Pages。
- Java的能力。
- 简单的JSP。
- 了解HTTP。

在当今的环境中，动态内容是任何Web站点成功的关键。用户想要并且也需要得到适合他们情况的特定信息。随着Web成为企业计算和电子商务的标准平台，Web开发毫无疑问正在变得更为复杂。Web服务器正在变成Web应用程序服务器。数据库和应用程序的组合体正在整个因特网上迅速发展。内容提供商要求解决方案可以被很容易地建立并且快速投入市场，而且功能强大且灵活。

JavaServer Pages（JSP）是一项令人兴奋的新技术，它提供高效创建动态内容的强大能力。JSP是一种表示层技术，它允许静态Web内容与Java代码混合在一起。JSP允许使用静态HTML，但增加了Java编程语言的功能和灵活性。

JSP不改变静态数据，所以页面布局和外观可以继续使用当前的方法进行设计。这样就可以清楚地划分页面设计和应用程序。JSP还可以将Web应用程序分割成独立的组件。这就允许使用HTML和进行设计而无需了解产生动态数据的Java代码的详细知识。业务不再必须交给原本就缺乏的了解图形设计、布局、应用程序编程和软件设计的软件开发人员。

正如其名称所暗示的，JSP使用Java编程语言创建动态内容。Java的面向对象设计、平台无关性和内存保护模型等特性允许快速的应用程序开发。内置的连网和企业应用程序编程接口（API）使Java成为设计客户-服务器应用程序的理想语言。另外，Java通过支持JavaBean和企业JavaBean组件模型提供极其高效的代码重用。

1.1 Web的历史

要理解JSP页面的能力，首先必须看一下过去创建Web页面的方法。

1.1.1 静态页面

使用静态页面时，客户向服务器请求Web页面，而服务器将被请求的文件发送给客户。客户收到的文件副本与服务器上的文件完全相同（见图1-1）。

传统的Web页面是静态的、不变的。在正常条件下，它们总是保持原样。多个客户对同一静态页面的请求会得到相同的结果。

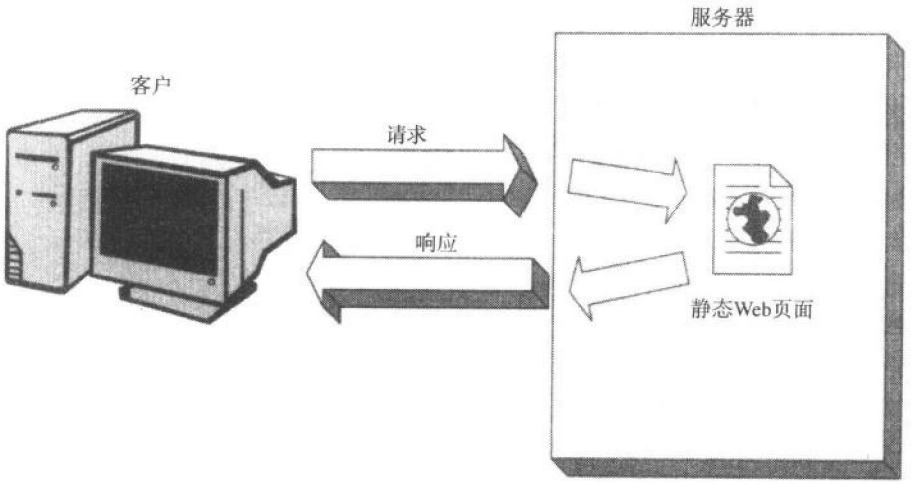


图1-1　标准HTML文档

1.1.2　动态页面

从HTML的早期开始，就已经存在动态处理数据的需求。最初，这种需求只不过是需要提供反馈表单。当今的Web站点需要的远比静态内容和反馈表单要多得多。动态数据对于Web上的所有东西（从在线银行到播放比赛）都很重要。

定义动态内容就是，在Web页面被请求时创建它们，并根据特定的规则改变其内容。例如，显示当前时间的Web页面就是动态的，因为它的内容随当前的时间而改变。动态页面由服务器上的应用程序生成，接收来自客户的输入并做出适当的响应（见图1-2）。

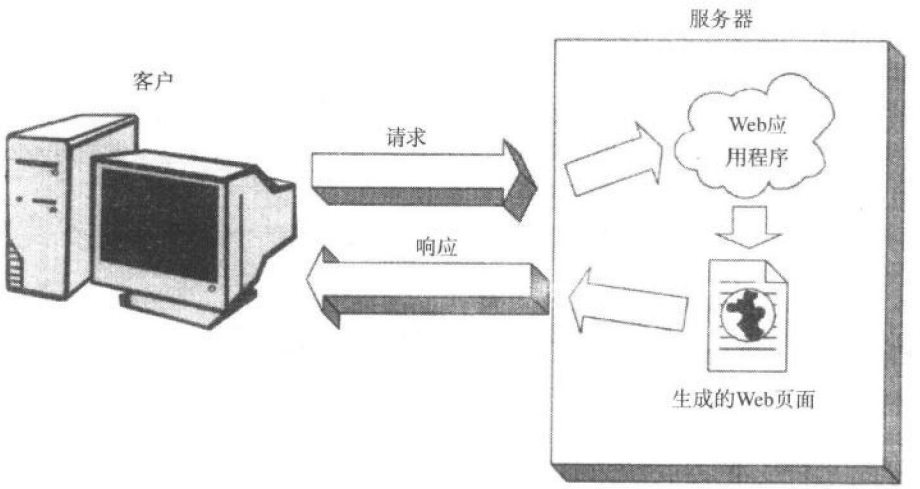


图1-2　动态HTML文档

几年来，已经采用了几种产生动态数据的方法。了解其他方法以及它们的优缺点有助于理

解JSP页面。

1. 通用网关接口

通用网关接口（CGI）可能是当今Web应用程序最常用的形式。在Web服务器时代的早期，CGI就被设计出来了，它允许请求被发送到外部程序。这些外部程序几乎可以用任何编程语言编写——最常用的是C、C++、Perl和Python。

CGI是在Web上创建动态内容的一个解决方案，但它的几个特性使它非常低效。CGI程序通常使用大量的系统资源，不只是CPU时间，还有大量内存。对于发送到CGI程序的每个请求，Web服务器要装载、运行和卸载整个CGI。CGI程序独立于Web服务器，并且不能写到Web服务器的日志或了解服务器的特性。另外，CGI页面常常不能轻易地移植到其他平台。

自从CGI程序实现以来，几个厂商开发了独特的方法来解决它固有的限制。由Open Market开发的FastCGI是CGI的替代品，它创建单一持久的进程用于每个CGI请求。有了它的帮助，只有一个进程用于每个CGI程序，并且如果此进程需要启动一个解释器（如Perl），那么会消耗更多的资源。要了解FastCGI的详细情况，请访问Fast Engine Web站点（http://www.fastengines.com/）。

改进CGI性能的另一个方法是在Web服务器中嵌入解释器。这就允许Web服务器预编译和预衍生CGI。它也允许固有的API挂到Web服务器。这些方法的最大局限是，它们与特定的语言、平台和Web服务器相关联。例如，mod_perl用于Apache Web服务器，由ActiveState开发的PerlEx用于微软的Internet Information Server（IIS，因特网信息服务系统）。要了解mod_perl的详细情况，请访问Perl/Apache Web站点（http://perl.apache.org）。要了解PerlEx的详细情况，请访问Active State的Web站点（http://www.activestate.com/plex）。

所有这些克服CGI程序限制的尝试已经提高了CGI程序的性能，但CGI进程自身的限制是造成系统障碍的主要原因。

2. 服务器API

创建动态Web应用程序的另一个方法是专用于Web服务器的API。Netscape为它的服务器套件提供了Web Application Interface（WAI，以前称为NSAPI）。微软为它的Internet Information Server提供了ISAPI。Apache Web服务器有一个基于模块的编程接口，它允许模块被装载到httpd可执行程序中。

通过与固有的Web服务器紧密地集成，所有这些服务器API提供非常高的速度和资源收益。但是，它们所建立的解决方案也被限制为用于某个平台和Web服务器。另外，服务器API扩展可能会带来一些安全问题。因为API扩展作为Web服务器自身的一部分运行，所以API扩展发生的问题可能会使Web服务器崩溃。

3. 客户端脚本解决方案

有一类Web开发技术是将代码发到用户处，并且在用户的机器上运行。这些技术对于开发动态内容是非常有用的工具，但是它们有严格的限制。

客户必须以所期望的方式支持脚本语言。微软的IE和Netscape的Navigator解释客户端脚本的方式不同，这可能在客户的浏览器上产生奇怪的差异。

VBScript就是一种客户端脚本语言。VBScript基于微软的Visual Bsic。当前只有微软的IE支持VBScript。

JavaScript（也称为JScript和ECMAScript）在创建完全在客户端运行的Web应用程序方面起到重要作用。欧洲计算机生产商协会（ECMA）近来通过结合Netscape和微软开发的流行版本（ECMA-262），已经将JavaScript标准化。

JavaScript被限制为只用在支持此语言的Web浏览器上；Netscape和Opera Web浏览器直接支持JavaScript，而微软支持JavaScript标准的一个版本——JScript。JavaScript与JSP页面的作用非常不同，但这两种语言可以结合使用来创建令人惊异的结果。要了解JavaScript的详细信息，请阅读Janice Winsor和Brian Freeman的《Jumping JavaScript》（Pearson Technical Reference/Prentice Hall，1998）。

4. 服务器端脚本解决方案

有几种创建Web应用程序常见的脚本解决方案。这些脚本在页面被发送到用户前在服务器上运行。

Netscape的服务器端脚本解决方案称为Server Side JavaScript（SSJS）。在SSJS中，JavaScript在服务器上执行以修改HTML页面，并且脚本被预编译以提高服务器的性能。SSJS在Netscape Web服务器的几个不同版本上可以使用。要了解SSJS的详细信息，请访问http://developer.netscape.com/tech/javascript/ssjs/ssjs.html。

微软的服务器提供Active Server Pages（ASP）。ASP页面与JSP非常相似，它允许开发者将VBScript或JScript代码直接嵌入到Web页面中。ASP页面必须在每次运行时进行编译，这和CGI脚本的主要缺点一样。只有运行微软的Internet Information Server 3.0或更高版本，开发人员才能使用ASP。

显然，主要的脚本解决方案最大的缺点是：它们是专用的。这里讨论的所有解决方案都依赖于某种Web服务器或特定的厂商。

5. Java Servlet

Java Servlet是CGI程序和脚本语言强大的替代品。Servlet极其类似于专用的服务器API，但因为它们是用Java语言编写的，所以可以被轻易地移植到任何支持Servlet API的环境中。因为它们运行在Java虚拟机中，所以可以绕过影响服务器API的安全问题。

Servlet运行于Servlet引擎内。各个Servlet作为Web服务器进程中的一个线程运行。这个解决方案比CGI程序所实现的多个服务器进程要高效得多。通过在线程中运行，Servlet的可伸缩性也很好，并且因为它们是Web服务器进程的一部分，所以可以和Web服务器更近地交互。

Servlet是CGI程序的极其强大的替代品。它们可以用于扩展任何类型的服务器。内置的线程和安全支持使Servlet成为扩展服务器服务的可靠工具。

所有主要的Web服务器现在都支持Servlet。使用Java Servlet的主要缺点在于它们的能力。Java编程语言既强大又复杂，学习Java对于一般的Web开发人员是一个艰难的任务。

1.2 JSP概述

JSP（Javaserver Pages）是进行Web开发的强大工具。JSP技术使用服务器端脚本，这些脚本实际上被转换为Servlet，并且在运行前进行编译。这向开发人员提供了创建强大的Java Servlet的脚本编程接口。

JSP页面提供的标记使开发人员不必编写复杂的Java代码就可以执行大多数动态内容操作。高级的开发人员可以添加完整的Java语言功能来在JSP页面中执行高级操作。

1.2.1 模板页面

显然，使页面进行动态响应的最高效的方法是修改静态页面。理想情况下，将特殊部分添加到页面，这些部分由服务器动态修改。在这种情况下，页面变得更像是服务器在发送前处理的页面模板。它们现在不再是普通的Web页面，而是服务器页面了。

在服务器页面上，客户请求Web页面，服务器用新的数据替换模板的某些部分，然后将修改后的页面发送到客户（见图1-3）。

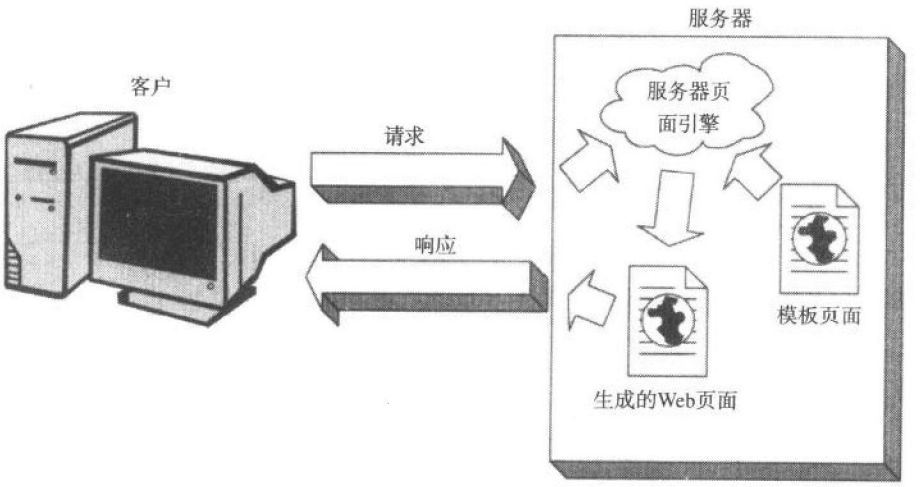


图1-3 服务器页面

因为处理是在服务器上进行的，客户收到的看起来像是静态数据。从客户的角度来看，服务器页面与标准的Web页面间并无差别。这样创建的动态页面解决方案不消耗任何客户资源并且完全独立于浏览器。

1.2.2 静态数据与动态元素

因为JSP页面是围绕静态页面设计的，它们可以由与标准Web页面一样的静态数据组成。JSP页面使用HTML或XML建立页面的格式和布局。只要是通常的Web页面可以包含的数据，JSP页面就可以包含。

为了替换页面的某些部分，服务器需要能够识别应该改变的部分。JSP页面通常有一组特殊的标记，用于表示页面的哪些部分应该由服务器修改。JSP使用<%标记标明JSP部分的开始，使用%>标记标明JSP部分的结束。JSP将这两个标记间的所有内容解释为特殊部分。

JSP页面通常既包含静态内容又包含动态元素。重要的是，要理解两者间的差别。静态数据在服务器页面内从不改变，而动态元素总是要在发送到客户前被解释和替换。

1.2.3 简单的JSP页面

了解事物的最简单的方法通常是亲眼看看它。清单1-1显示了一个非常简单的JSP页面。

清单1-1 simpleDate.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Final//EN">
<HTML>
<HEAD>
<TITLE>A simple date example</TITLE>
</HEAD>

<BODY COLOR=#ffffff>
The time on the server is
<%= new java.util.Date() %>
</BODY>
</HTML>
```

如果你不知道这个JSP页面正在做什么，不必太担心，后面的章节将对此进行讨论。重要的是，注意此页面中的两类数据：静态数据和动态数据。理解两者间的差别可以为创建JSP页面打下基础。

当客户请求这个JSP页面时，客户将收到HTML文档。转换如图1-4所示。

在被编译并且被发送到浏览器后，此页面应该看起来像图1-5。

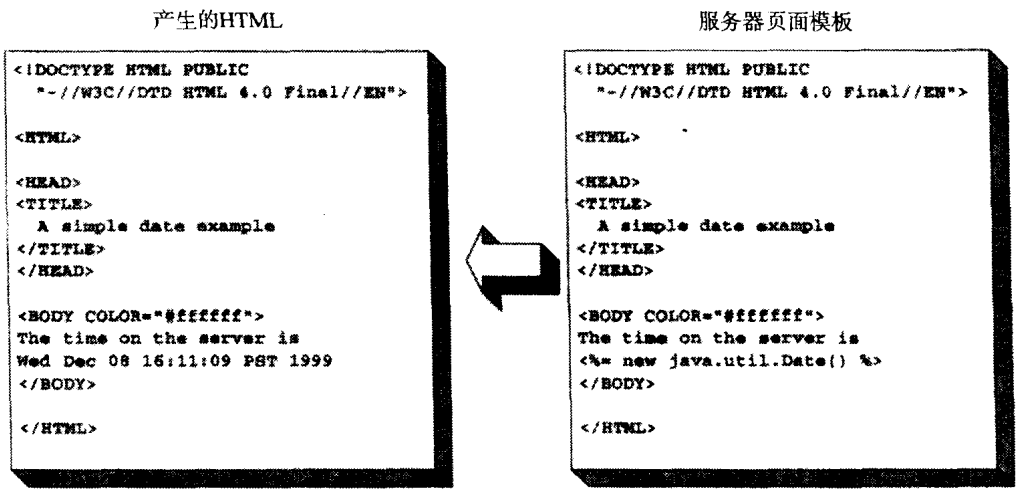


图1-4 服务器页面转换为HTML数据

1.2.4 JSP文件

大多数理解JSP的Web服务器将寻找特定的文件扩展名。通常，文件名以.jsp结尾的文件将由JSP引擎解释和处理。实际的扩展名是可以配置的，但是为了清楚起见，本书将一直使用扩展名.jsp。

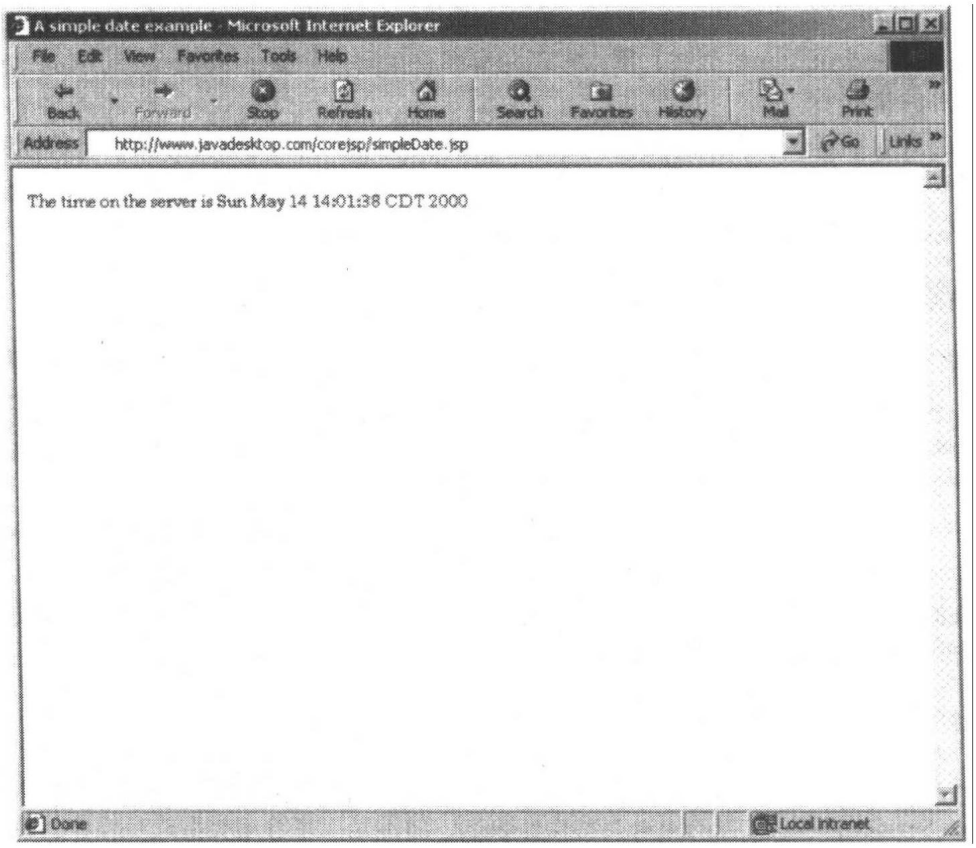


图1-5 一个简单的JSP页面

注意：没有JSP标记的JSP页面

因为JSP可以处理与HTML文件相同的静态数据，任何HTML都可以将扩展名改为.jsp。如果JSP服务器正在运行，这些文件现在将通过JSP引擎运行。因为没有特殊的标记标明动态部分，这个文档的表现将与原始的HTML文件一样；但是它会消耗更多的资源，因为JSP引擎会尝试分析和执行这个文件。

1.3 Java的能力

JSP页面从底层的Java语言和Java Servlet技术继承了许多优点。它们也通过集成组件模型获得了其他开发方法的优点。除了这些优点，JSP规范的设计也相当优秀，允许可扩充性和与其他语言和规范的集成。

1.3.1 只写一次，随处运行

因为JSP页面利用了Java语言，所以它们自动获得了许多优点。首先（也是最重要的）是最高级的可移植性，这由完美定义且被广为接受的Java API提供。在一个平台上开发出来的JSP页

面可以被部署在许多种系统上。例如，在Windows NT系统上开发出来的JSP页面经过JSP Reference Implementation的测试，可以被轻易地部署在运行Allaire Software的JRun Application Server的Linux机器上。

而且，JSP页面避免了跨平台Java开发的几个令人讨厌的方面。因为JSP页面运行在服务器上，所以应用程序不必用几种不同的客户平台进行测试，而Java applet常常需要这么做。有时令人讨厌的Java GUI系统（如AWT和Swing）开发在JSP页面中也可以避免。

1.3.2 Java API

在你编写JSP页面时，注意到的第一件事可能是：JSP作者可以利用Java API的全部能力。核心的Java API提供如下能力：连网、多线程、数据库连接、国际化、图像操作、对象串行化、远程方法调用、CORBA访问等等。对Java的标准扩展，例如Java Naming and Directory Interface（JNDI）和Java Mail API，为Web应用程序提供了强大的扩充功能。

使用各个软件厂商提供的Java类、JavaBean和企业JavaBean组件，可以轻易地向Web应用程序添加强大的代码。使用JavaBean组件框架，JSP页面可以组成多层应用程序的表示层。

JSP页面可以写成直接与小应用程序通信，允许相同的代码在服务器和客户两端进行平衡。这开创了客户/服务器应用程序开发的全新局面。

1.3.3 安全

从Java语言继承的另一个优点是强类型安全性。与大多数常见的脚本语言不同，JSP页面和底层的Java Servlet API以数据的原有类型操作数据，而不是以字符串类型进行操作。Java还使用自动处理废弃集合和无用指针来避免许多内存问题。

Java也以其优秀的异常处理功能而闻名。当错误发生时，JSP可以安全地捕获异常并且通知用户，而不是使服务器崩溃。这个内置的特性被认为远比其他Web应用程序环境中实现的附加的扩充和模块要优秀。

最后，Java应用程序服务器可以利用Java安全管理器，防止编写得很差的JSP页面影响服务器的性能或者损害主机文件系统。Java安全管理器控制对资源（这些资源可能被用于损害系统）的访问权，只允许具有适当权力的进程对受保护的资源进行访问。这是Java语言的基础部分。

1.3.4 可伸缩性

Java语言以及Java Servlet API为JSP页面添加了几个可伸缩性组件。在一个JSP页面被装载后，它一般被保存在内存中。当对此JSP页面的新的请求到来时，服务器进行一个简单的方法调用。这和传统的CGI应用程序大不相同，CGI应用程序通常为每个请求产生一个进程和一个解释器。底层的服务器通过利用单独的线程并行处理多个请求，这使JSP具有高度的可伸缩性。

集成进JavaBean组件框架时，JSP页面变得更加可伸缩。例如，一个JDBC JavaBean可以处理来自JSP页面的多个请求，并且维持一个到后端数据库的高效连接。在与企业JavaBeans集成时，这特别高效，企业JavaBean为Web应用程序添加事务和安全服务，以及为Java组件提供中间件支持。

1.3.5 可扩充性

JSP页面的另一个竞争优势是它们的可扩充性。JSP规范本身就是Java Servlet的扩充。在JSP页面中，JSP规范可以被扩充以创建定制的标记。这些标记允许JSP“语言”以可移植的方式进行扩充。例如，可以创建一个定制的标记库，用它包含嵌入式的数据库查询。通过使这些标记库可以移植并且为它们提供一个公用的接口，JSP页面可以内在表达传递组件模型。

JSP规范的作者也为未来的扩充留下了空间，办法就是使JSP所用的元素独立于任何特定的脚本语言。当前，JSP规范只支持用Java语言进行脚本编程，但是JSP引擎可以选择支持其他语言。

JSP与XML（可扩充标记语言）的密切关系也非常重要，因为XML具有可扩充性和高度组织化的结构。正确组成的JSP页面实际上可以被写成合法的XML文档。利用静态模板，可以在JSP页面中进行简单的XML生成工作。动态XML的生成可以使用定制的标记组件、JavaBean或企业JavaBean组件完成。XML也可以作为请求数据被接收，并且被直接发送到定制的标记组件、JavaBean或企业JavaBean组件。

1.3.6 组件

JSP页面的一个相当强大的特性是，它可以集成到JavaBean组件框架。这为由开发团体创建的大型企业应用程序打开了大门。随着Web应用程序变得越来越复杂，利用JSP的组件本质有助于将复杂的任务分解为较简单的、更加可管理的模块。JSP有助于将表示逻辑与业务逻辑分隔开，并且可以将静态数据与动态数据分隔开。

由于JSP具有以组件为中心的本质，Java程序员和非Java程序员都可以利用JSP。这允许Java程序员建立和使用JavaBean，并且利用对这些Bean的适当控制创建动态Web页面。而非Java程序员可以使用JSP标记连接到有经验的Java程序员所建立的JavaBean。

1.4 了解HTTP

JSP规范在JSP所用的协议方面没有定义任何限制。它要求JSP支持HTTP（超文本传输协议）。事实上，当前所有的JSP都利用HTTP发送内容。

JSP作者不需要理解HTTP的复杂细节，但是HTTP给JSP带来了一些限制。下面对HTTP进行简单的描述，详细细节将在后续的章节中给出。

1.4.1 HTTP基础

HTTP是一个通用的轻量协议，用于发送HTML和XML。HTTP是在WWW上发送信息的主要协议。

HTTP也是一个在请求间不保持联系的无状态协议。这意味着当客户向服务器发出请求时，服务器发送响应，然后事务结束。来自同一客户的第二个请求是一个全新的事务，与前一个请求无关。

1.4.2 HTTP请求

客户通过向服务器发送特殊的“请求消息”来请求信息。这个请求包含表示请求本身的一