21世纪网站开发技术丛书（3）

本光盘内容包括：
本版电子书

中文版

JSP Database Programming Guide

JSP 数据库编程指南

北京希望电子出版社　总策划
布雷恩·赖特　著
赵明昌　译

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

21世纪网站开发技术丛书（3）

本光盘内容包括：
本版电子书

中文版

JSP Database Programming Guide

JSP 数据库编程指南

北京希望电子出版社 总策划
布雷恩·赖特 著
赵明昌 译

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

这是一本关于 JSP 数据库编程方面的书。详细介绍了 JSP（Java Server Pages）在大型数据库 Oracle 中应用与开发。当今社会，信息技术飞速发展，人们越来越依靠现代网络技术来实现各种价值，架构自己的个人网站，组建企业的门户网站，进行网上营销、交流和宣传，为满足广大读者需求，本社特地组织了这套国外最新的网站开发技术丛书。

JSP 技术是企业应用编程中的一部分，它基于强大的 Java 语言，具有良好的伸缩性，与 Java Enterprise API 紧密地集成在一起，在开发电子商务方面具有得天独厚的优势，基于 Java 平台构建电子商务平台已经成为当今 IT 领域新时尚。

本版书由 9 章和 3 个附录组成，主要内容包括：JSP 简介、Oracle 的特定 JSP 实现——OracleJSP 介绍、OracleJSP 的基础知识、编程技巧以及扩展功能、OracleJSP 的解释与发布过程、JSP 标签库和 Oracle 的 JML 标签介绍、OracleJSP 对多种字符集的支持、OracleJSP 的应用范例等。三个附录内容为 OracleJSP 在多种环境下的安装与配置、Servlet 与 JSP 技术的背景知识、编译时 JML 标签支持等。

本版书内容新颖、丰富，具有很强的技术通用性、实用性和指导性，尤其是在数据库的二次应用开发方面，可以说是填补了 JSP 和大型通用数据库平台之间的无缝隙的应用和开发的空白。

本版书不但是从事 JSP 和大型通用数据库平台之间应用和开发的广大编程人员的重要指导书，同时也是高等院校师生教学与自学的优秀参考书、科研院所图书馆的馆藏读物。

本光盘内容为本版电子书。

系 列 盘 书 名：21 世纪网站开发技术丛书（3）
盘 书 名：JSP Database Programming Guide JSP 数据库编程指南
总 策 划：北京希望电子出版社
文 本 著 作 者：布雷恩•赖特 著 赵明昌 译
责 任 编 辑：王玉玲
C D 制 作 者：希望多媒体开发中心
C D 测 试 者：希望多媒体测试部
出版、发行者：北京希望电子出版社
地 址：北京中关村大街 26 号，100080
网址：www.bhp.com.cn
E-mail: lwm@hope.com.cn
电话：010-62562329,62541992,62637101,62637102,62633308,62633309（发行）
010-62613322-215（门市） 010-62547735（编辑部）
经 销：各地新华书店、软件连锁店
排 版：希望图书输出中心
CD 生 产 者：北京中新联光盘有限责任公司
文 本 印 刷 者：北京广益印刷厂
开本 / 规格：787 毫米×1092 毫米 1/16 开本 16.875 印张 381 千字
版次 / 印次：2001 年 6 月第 1 版 2001 年 6 月第 1 次印刷
本 版 号：ISBN 7-900071-27-X/TP • 26
定 价：38.00 元（1CD，含配套书）

译 者 的 话

本书专门介绍 JSP（JavaServer Pages）在大型数据库 Oracle 中应用与开发。今天，越来越多的人使用 JSP 技术来构建高效的电子商务系统，开发各种中间交易系统，创建高水平的企业网站。JSP 技术是企业应用编程中的一部分，它基于强大的 Java 语言，具有良好的伸缩性，与 Java Enterprise API 紧密地集成在一起，在开发电子商务方面具有得天独厚的优势，基于 Java 平台构建电子商务平台已经成为当今 IT 领域新时尚。这种技术的原理是：利用 Microsoft SQL Server 7.0、Oracle 8.0 或者 Sybase 等数据库系统作为后台数据仓库，用 Servlet 等高性能服务器端程序作为后台总控程序，而 JSP 程序在前台运行。Servlet 接受用户的输入，分别调用不同的 JSP 程序向客户端反馈信息，JSP/Servlet 通过 HTTP 连接在服务器端和客户段传递数据，JSP/Servlet 并不使用 JDBC 技术直接访问数据库系统，而是把参数传递给事先编制好的 JavaBeans 和 EJB 组件，由它们对数据库进行操作，这样就把系统内部的数据封装保护起来了。JavaBeans 和 EJB 组件还可以把事务分发到另一个组件中去处理，最后把数据库返回的结果由 JSP/Servlet 送到客户端浏览器显示出来。这样的模式很容易实现分布式网络计算，因此许多企业应用都能够作成 JavaBeans 组件，可以重复利用，这样既封装了某些关键的操作，又方便了开发者，提高了开发速度，网站的伸缩性、安全性也得到了很好的保证。

本书由 9 章和 3 个附录组成，主要内容包括：JSP 简介、Oracle 的特定 JSP 实现——OracleJSP 介绍、OracleJSP 的基础知识、编程技巧以及扩展功能、OracleJSP 的解释与发布过程、JSP 标签库和 Oracle 的 JML 标签介绍、OracleJSP 对多种字符集的支持、OracleJSP 的应用范例等。三个附录包括 OracleJSP 在多种环境下的安装与配置、Servlet 与 JSP 技术的背景知识、编译时 JML 标签支持等。

本书的特点是内容新，丰富，具有很强的技术通用性、实用性和指导性，尤其是在数据库的二次应用开发方面，可以说是填补了 JSP 和大型通用数据库平台之间的无缝隙的应用和开发的空白。

本书不但是从事 JSP 和大型通用数据库平台之间应用和开发的广大从业人员的重要指导书，也是高等院校师生教学与自学的优秀参考书、科研院所图书馆的馆藏读物。

参加本书翻译工作的有赵明、尚春红、魏亚峰，感谢方敏、刘立峰、吴晓晖、李威为本书校对，程丽、张云、吴丽为本书录入文稿。由于译者水平有限，书中错误在所难免，请读者批评指正。

目 录

第1章 概 述

本章回顾了 JSP（JavaServer Pges）技术的标准特征和功能，如果要获得更进一步的信息，请参考 Sun 公司的 JSP1.1 技术标准。

（要获得 Oracle 特定的 JSP 实现——OracleJSP 的特征概述，请参看本书第二章。附录 B 也提供了标准的 Servlet 和 JSP 技术的相关背景。）

本章包括以下几部分内容：

- JSP 简介
- JSP 的运行
- JSP 的语法元素一瞥

1.1 JSP 简介

JSP 技术由 Sun 公司提出，利用它可以很方便地在页面中生成动态的内容，使网络应用程序可以输出多姿多彩的动态页面。JSP 技术通常与 Java Servlet 技术相结合，可以在 HTML 页面（或者其他标记语言，如 XML）中内嵌了 Java 代码段并且调用外部 Java 组件。它作为一个前端处理工具，可以使用 JavaBeans 和 EJBs（Enterprise JavaBeans）完美地实现复杂的商业逻辑和动态功能。

JSP 代码与 JavaScript 等网页脚本语言是不同的，在标准的 HTML 页面中可以出现的任何内容都可以在 JSP 页面中出现。

在一个典型的数据库应用中，JSP 页面将会调用某些 JavaBean 或者 Enterprise JavaBean 组件，这些组件可以通过 JDBC 或者 SQLJ 直接或间接地访问数据库。

JSP 页面在运行之前要被解释成 Java Servlet，解释过程是按需进行的，有时可能会提前进行），然后它可以处理 HTTP 请求并生成响应信息，JSP 技术为编写 Servlet 程序提供了更为便利的途径。

另外，JSP 页面和 Servlets 程序是可以相互操作的，也就是说 JSP 页面可以包含从 Servlet 程序输出的内容，可以将内容输出到 Servlet 程序，反过来 Servlet 程序也可以包含从 JSP 页面输出的内容，并且可以将内容输出到 JSP 页面中。

1.1.1 JSP 页面的外观

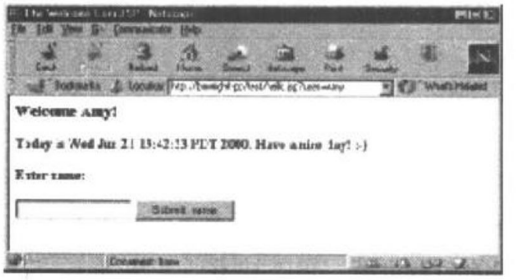
下面是一个简单的 JSP 页面例子（里面所使用的 JSP 语法元素的解释可以参看本书相关部分）。

```
<HTML>
<HEAD><TITLE>The Welcome User JSP</TITLE></HEAD>
<BODY>
<% String user=request.getParameter("user"); %>
<H3>Welcome <%= (user==null) ? "" : user %>!</H3>
<P><B> Today is <%= new java.util.Date() %>. Have a nice day! :-)</B></P>
<B>Enter name:</B>
```

```
<FORM METHOD=get>
<INPUT TYPE="text" NAME="user" SIZE=15>
<INPUT TYPE="submit" VALUE="Submit name">
</FORM>
</BODY>
</HTML>
```

如本例所示，在一个 JSP 页面中，JSP 元素以标签<%开始，以标签%>结束。本例中，Java 代码从 HTTP request 对象中获取用户名信息，接着打印出用户名并且得到当前的日期。

在此例中，比如用户输入的名字是“Amy”，那么 JSP 页面的输出如下图所示。



1.1.2 JSP 编码与 Servlet 编码，哪个更方便

在 HTML 页面中内嵌 Java 代码和 Java 调用与直接在 Servlet 程序中编写 Java 代码相比起来更为方便。JSP 语法允许你更快捷地编写动态 Web 页面内容，与 Java Servlet 语法相比需要更少的代码。

下面是一个用来对比 Servelet 代码和 JSP 代码的例子：

Servelet 代码：

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Hello extends HttpServlet
{
  public void doGet(HttpServletRequest rq, HttpServletResponse rsp)
  {
    rsp.setContentType("text/html");
    try {
      PrintWriter out = rsp.getWriter();
      out.println("<HTML>");
      out.println("<HEAD><TITLE>Welcome</TITLE></HEAD>");
      out.println("<BODY>");
      out.println("<H3>Welcome!</H3>");
      out.println("<P>Today is "+new java.util.Date()+".</P>");
      out.println("</BODY>");
      out.println("</HTML>");
    } catch (IOException ioe)
    {
      // (error processing)
    }
```

```
    }
}
```

（有关 HttpServlet 抽象类 Response，HttpServletResquest 接口和 HttpServletResponse 接口的背景信息，请参看本书“Servlet 接口”部分。）

JSP 代码：

```
<HTML>
<HEAD><TITLE>Welcome</TITLE></HEAD>
<BODY>
<H3>Welcome!</H3>
<P>Today is <%= new java.util.Date() %>.</P>
</BODY>
</HTML>
```

要知道 JSP 的语法是多么地简单，它可以省略掉传统 Java 代码中的 imports 语句和 try...catch 代码块，并且 JSP 的解释器会为你自动处理大量 servlet 编码工作，比如它直接或间接地实现了标准的 javax.servlet.jsp.HttpJspPage 接口并且添加必要的代码以获得一个 HTTP Session 对象。

另外，JSP 页面里面的 HTML 代码可以直接出现，而不必像 Servlet 代码那样必须嵌在 Java 的 print 语句里，因此你可以使用任何流行的 HTML 创作工具来创建 JSP 页面。

1.1.3　分隔商业逻辑与页面内容——JavaBeans 的调用

JSP 技术的最大优点就是可以将网页的静态内容（HTML 代码）开发与网页的动态内容（Java 代码）开发分隔开来，从而可以使得精通 HTML 但对 Java 不很精通的开发人员专门负责网页静态内容的开发，而那些对 Java 很在行但却不熟悉 HTML 的开发人员就可以专注于网页的动态内容的开发。

在一个典型的 JSP 页面中，大部分的 Java 代码和商业逻辑将不会在内嵌的代码段中，相反，它主要通过调用 JavaBeans 或 Enterprise JavaBeans 组件来实现。

JSP 提供了下面的语法来定义和创建一个 JavaBeans 对象的实例：

```
<jsp:useBean id="pageBean" class="mybeans.NameBean" scope="page" />
```

本例创建了类 mybean.NameBean 的一个实例：PageBean（scope 参数将会在本章的后面作出解释。），在 JSP 页面中此句的后面，你就可以如下例般使用这个实例了：

```
Hello <%= pageBean.getNewName() %> !
```

（如果 pageBean 的 newName 属性为“Julie”，那么上句输出“Hello Julie!”。）

将商业逻辑从页面内容中分隔出来将有利于更便利的分工合作，Java 专家可以专门负责商业逻辑和动态内容，比如编写和维护 NameBean 的代码；而 HTML 专家可以负责页面的布局和内容的表达，比如编写和维护 jsp 文件的代码。这样做可以做到责任明确，提高开发的效率。

JavaBeans 使用标签 useBean 去声明 JavaBean 的实例对象，并且 getProperty 和 setProperty 去访问 bean 的属性，关于这方面的详细内容将在“JSP 的 Actions 和<jsp:> 标签集”中讨论。

1.1.4 JSP 页面和其他标记语言

JSP 技术最典型的应用是动态 HTML 输出，但是 Sun 公司的 JSP1.1 技术标准也支持其他类型结构下的、基于文本的文档输出。由于 JSP 解释器不处理 JSP 元素以外的文本，因此任何选用于页面的文本也同样可以出现在 JSP 页面中。

JSP 页面从 HTTP request 对象得到相关信息，通过 SQL 数据库查询从数据服务器上得到相关数据，然后处理这些信息并将它们组合在一起输出到 HTTP response 对象，输出的内容可以是 HTML 格式、DHTML 格式、XHTML 格式或者 XML 格式。

1.2 JSP 的运行

本节简单地介绍了 JSP 页面的运行过程，包括按需解释（在 JSP 页面第一次运行时需要解释），JSP 容器和 Servlet 容器的作用以及出错处理。

注意： 术语 JSP 容器（JSP container）出现于 Sun 公司的 JSP 1.1 技术标准中，用以替代早期标准中的术语 JSP 引擎（JSP engine），这两个术语是同义的。

1.2.1 JSP 容器概论

一个 JSP 容器就是一个程序实体，它用来解释、执行和处理 JSP 页面并且将请求信息传递给 JSP 页面。

JSP 容器的组成随着 JSP 的不同实现也不尽相同，但本质上它是由许多 servlet 和 serlets 的集合构成的，因此 JSP 容器是被 Servlet 容器所运行的。

如果 Web Server 是用 Java 语言开发的话，那么 JSP 容器可以被集成到 Web Server 中，否则此容器可以与 Web Server 建立关联并被其使用。

1.2.2 JSP 页面和按需解释

假设在一个典型的按需解释场合下，JSP 页面通常是通过下面步骤来运行的：

1. 用户通过 HTTP 请求一个后缀名是.jsp 的 URL，此 URL 连接到了一个 JSP 页面上。
2. Web Server 的 Servlet 容器发现在 URL 中有.jsp 文件扩展名，就调用 JSP 容器来进行处理。
3. 如果此页面是第一次被请求，JSP 容器要定位此 JSP 页面文件并解释它，解释的过程包括将 JSP 源文件处理成 Servlet 代码文件(.java)以及编译.java 文件生成 Servlet 的.call 文件。

   JSP 解释器生成的 servlet 类是实现了 javax.servlet.jsp.HttpJspPage 接口的类(由 JSP 容器提供）的一个子类。这个 servlet 类被叫做页面实现类，本书中将页面实现类的实例称为 JSP 页面实例。

   将 JSP 页面解释成 servlet 的同时会自动生成一些 servlet 代码，如 javax.servlet.jsp.HttpJspPage 的实现代码以及它的 service 方法代码，这将极大地减轻编程工作量。
4. JSP 容器运行页面实现类的实例，此时 servlet（即 JSP 页面实例）就会处理 HTTP

请求，生成对应的 HTTP 响应并传回给客户端。

注意： 上面的步骤粗略地描述了 JSP 页面的运行过程，正如前面所提到过的，每个销售商都有自己的 JSP 容器实现，但是它都是由一个 servlet 或者一套 servlet 来组成的。例如，可能有一个前端的 servlet 专门用来定位于 JSP 页面文件，一个解释的 servlet 专门用来解释和编译；以及一个封装的 servlet 类，它可以被每个页面实现类所继承（因为一个解释后的页面并不是一个纯 servlet，所以不能直接由 servlet 容器来运行）。这些 servlet 要想正常运行都必须要有一个 servlet 容器。

1.2.3　请求一个 JSP 页面

一个 JSP 页面可以通过两种方法来请求：要么是直接通过 URL 请求，要么是通过另一个网页或 servlet 来间接请求。

直接请求 JSP 页面

在一个 servlet 或 HTML 页面中，最终用户可以直接通过 URL 来请求一个 JSP 页面，例如，假设在 myapp 应用程序的根目录下存在 HelloWorld.JSP 页面文件，如下所示：

```
myapp/dir1/HelloWorld.jsp
```

如果 Web Server 使用端口 8080，那么可以通过下面 URL 来请求此页面：

```
http:// hostname:8080/myapp/dir1/HelloWorld.jsp
```

（应用程序的根目录是在应用程序的 servlet 环境中指定的。）

当最终用户第一次请求 HelloWorld.jsp 文件时，JSP 容器将解释并运行此页面，对以后的再次请求，JSP 容器只运行此页面，解释过程已经不再需要。

间接请求 JSP 页面

与 servlet 一样，JSP 页面也可以间接地运行，例如从一个标准的 HTML 页面连接至 JSP 页面，或者从其他的 JSP 页面、servlet 中引用另一个 JSP 页面。

当一个 JSP 页面中通过 JSP 语句来调用另外一个 JSP 页面时，路径的指定可以有两种方法：一种是相对于应用程序的路径——叫做 context–relative 或者 application–relative 路径；另外一种是相对于调用页面的路径——叫做 page–relative 路径。application–relative 路径以“/”起始，而 page–relative 路径则没有“/”。

应该注意到，这两种路径与在 URL 或 HTML 链接中指定的路径都不相同。接着上一小节的例子，与直接 URL 请求具有相同效果的 HTML 链接路径如下所示：

```
<a href="/myapp/dir1/HelloWorld.jsp" /a>
```

具有相同效果的使用 application –relative 路径的 JSP 语句如下：

```
<jsp:include page="/dir1/HelloWorld.jsp" flush="true" />
```

具有相同效果的使用 page–relative 路径的 JSP 语句如下：（调用和被调用的页面处于同一目录下。）

```
<jsp:forward page="HelloWorld.jsp" />
```

（“JSP 的 Actions 和<jsp:> 标签集”中将讨论 jsp:include 和 jsp:forward 语句。）

1.3 JSP 的语法元素一瞥

在“JSP 页面的外观”中曾经有一个简单的例子，其中使用了一部分 JSP 语法。本节将进一步介绍 JSP 的各种语法元素，包括下面的内容：

- 指令语句，用来表示应将 JSP 作为一个整体来看待。
- 脚本元素，也就是 Java 代码段，用来声明变量，添加表达式、代码段以及为程序编写注释等。
- 对象和作用域，Java 的对象可以显式创建也可以隐式创建，每个对象都有一个作用域，在此作用域里可以访问这个对象的属性和方法，出了此作用域，对象就会被释放。作用域一般有页面作用域及会话作用域等。
- 动作（action），用来创建对象或者在 JSP 的响应中改变输出流，也可既创建对象又改变输出流。

上面所列主题的基本语法和例子在本节后续部分将有进一步介绍，要获得更多的信息，请参看 Sun 公司的 JSP1.1 技术标准。

注意： JSP 的指令、声明、表达式以及脚本等的，都有对应的与 XML 兼容的语法，请参看“可替换 XML 的 JSP 语法”以获得更多信息。

1.3.1 指令语句

指令语句所设置的属性是针对整个 JSP 页面的，它主要用来控制解释和运行页面时所需的参数信息。指令语句的基本语法如下：

```
<%@ directive attribute1="value1" attribute2=" value2"... %>
```

JSP 1.1 标准支持的指令语句有：

- page——使用此语句来设置任意与页面相关的属性，比如使用的脚本语言，需要继承的类，引入的包，错误处理页面以及页面输出缓存的大小等，并且它一次可以设置多个属性。下面是一个例子：

```
<%@ page language="java" import="packages.mypackage"
errorPage="boof.jsp" %>
```

如果要将 JSP 页面的输出缓冲大小设为 20kb（缺省值为 8kb），请参照下例：

```
<%@ page buffer="20kb" %>
```

如果在页面中不使用缓冲，请参照下例：

```
<%@ page buffer="none" %>
```

注意：

  - 使用错误处理页面的 JSP 页必须被缓冲，如果出错从而转向错误处理页面时，可以清除缓冲内容，不向浏览器输出。
  - 在 OracleJSP 中，java 是缺省的语言设置，但是在程序中显式地声明是个良好的编程习惯。

- include——使用此语句可以将另外一个包含有内容和代码的 JSP 文件插入到当前的 JSP 页面中。指定 JSP 文件的路径时必须使用相对于当前 JSP 页面的 URL 路径，

下面是一个例子：

```
<%@ include file="/jsp/userinfopage.jsp" %>
```

include 语句既可以指定相对于页面的路径，也可以指定相对于环境的路径。

注意：

- 此处所讲的 include 语句是所谓的“静态包含”，在本章后续部分还将讨论使用 jsp:include 语句的动态包含。静态包含在 JSP 页面被解释的时候生效，而动态包含在 JSP 页面被请求的时候生效。
- include 语句只能在相同的 servlet 环境下使用。

- taglib——使用此语句可以在 JSP 页面中添加定制的 JSP 标签库。一些厂商为了扩展 JSP 功能而提供他们自己的标签集，使用 taglib 语句就可以使用这些定制的标签了。Taglib 是要指定标签库的描述文件的路径以及为了使用它们而加的前缀，请看下面的例子：

```
<%@ taglib uri="/oracustomtags" prefix="oracust" %>
```

在此页面的后面，如果想使用 oracustomtags 标签库中的标签，那么就必须使用 Oracust 后缀，如下例所示（假设标签库中有一个标签叫 dbaseAccess）：

```
<oracust:dbaseAccess>
...
</oracust:dbaseAccess>
```

从这些例子中可以看到，它们使用了 XML 风格 start-tag 和 end-tag 语法。JSP 标签库和标签库描述文件将在本章后面部分介绍，并且在第 7 章中会有更详细的讨论。

1.3.2 脚本元素

JSP 脚本元素包括以下几种，它们都是在 JSP 页面中可以出现的 Java 代码段：

- 声明——用来声明 JSP 页面所使用的方法或成员变量的语句。

  JSP 声明语句出现在<%!...%>声明标签中，它使用标准的 Java 语法来声明成员变量或方法，并且导致在生成的 servlet 代码中出现对应的声明语句。请看下面的一个例子：

```
<%! double f1=0.0; %>
```

  这个例子声明了一个成员变量 f1，同时在 JSP 解释器生成的 servlet 类代码中，变量 f1 也在顶层类中被声明。

注意：相对于成员变量来说，方法变量是在 JSP 小代码段中声明的（有关小代码段，后面将对其作出进一步解释）。

- 表达式——标准的 Java 表达式，首先经过计算，然后将结果转换成字符串类型，显示在页面中它们出现的地方。JSP 表达式不以分号（;）结束，并且需要包含在<%...%>标签中。下面是一个实例：

```
<P><B> Today is <%= new java.util.Date() %>. Have a nice day! </B></P>
```

注意：在请求期属性（如 jsp:setproperty 语句）中的 JSP 表达式不需要转换成字符串类型。

- 小代码段——所谓小代码段，就是 Java 代码和标记语言混合在一块形成的页面的

一部分。

在小代码段中，可以包括多种形式的 Java 代码，你可以使用一行完整的代码，也可以使用连续多行代码，甚至还可以使用不完整的半行代码，使它们与 JSP 页面中的 HTML 代码交叉出现，以完成一定的功能。

JSP 小代码段必须出现在<%...%>标签中，并且使用标准的 Java 语法。

例子 1：

```
<% if (pageBean.getNewName().equals("")) { %>
   I don't know you.
<% } else { %>
   Hello <%= pageBean.getNewName() %>.
<% } %>
```

在这个例子中，有三个单行的 JSP 代码与两行 HTML 代码混合出现，在第四行的 HTML 代码中还包括一个 JSP 表达式，注意到它不需要分号作为结束标记，并且 JSP 语法允许 HTML 代码插入到 if 和 else 分支语句之间，如果满足条件的话，HTML 内容才会被输出。

上面的例子中假设使用了一个 JavaBean 对象 pageBean。

例子 2：

```
<% if (pageBean.getNewName().equals("")) { %>
   I don't know you.
   <% empmgr.unknownemployee();
} else { %>
   Hello <%= pageBean.getNewName() %>.
   <% empmgr.knownemployee();
} %>
```

这个例子在小代码段中添加了更多 Java 代码，并且它假设有如下的前提条件：使用 JavaBean 对象 pageBean，事先已给定义并实例化了对象 empgr；empgr 对象有合适的方法去处理已知或未知的雇员信息。

注意：相对于声明成员变量来说，要使用 JSP 小代码段以声明方法变量，如下例所示：

```
<% double f2=0.0; %>
```

本例的小代码段声明了一个方法变量 f2，与成员变量不同的是，在 JSP 解释器所生成的 servlet 类代码中，f2 被声明成 service 方法的一个变量，而在前面曾经讲到过，成员变量是 servlet 类的一个变量。

要获得这两种变量之间比较的内容，请参考“方法变量声明与成员变量声明的比较。”

- 注释——即开发人员在 JSP 代码里所做的注释，它和 Java 代码里的注释是相同的。注释语句要出现在<%...%>标签中，与 HTML 注释不同的是，当用户查看页面的源代码时，这些注释语句是不可见的。

  实例：

```
<%-- Execute the following branch if no user name is entered. --%>
```

1.3.3 JSP 对象与作用域

在本书中，术语 JSP 对象指的是在 JSP 页面中声明或访问的 Java 类的实例。JSP 对象可以是下列情况中的一种：

- 显式的——显式对象在 JSP 页面的代码中声明并创建，根据你选择的作用域设置，它可以被本页面或其他页面所访问。
- 隐式的——隐式对象由潜在的 JSP 机制所创建，根据特定对象类型的继承作用域设置，它可以被 JSP 页面中的小代码段或表达式所访问。
  作用域将在后面的“对象作用域”一部分详细讨论。

显式对象

显式对象通常是使用 jsp:useBean 语句来声明和创建的一个 JavaBean 实例。关于 jsp:useBean 语句和其他的动作语句的详细描述，请参看“JSP 的 Actions 和<jsp:> 标签集”，但是本处也提供了如下的例子：

```
<jsp:useBean id="pageBean" class="mybeans.NameBean" scope="page" />
```

在这个例子中 jsp:useBean 语句定义一个对象 pageBean，它是 mybeans 包里名字叫“NameBean”类的一个实例。有关 scope 参数的详细讨论请参看下一节“对象作用域”。

在 Java 小代码段或声明中也可以创建对象，所用语法和在 Java 程序中创建一个 Java 类实例相同。

对象作用域

在 JSP 中声明的对象，不管它是显式对象还是隐式对象，都只能在某一特定的作用域下被访问。如果是用 jsp:useBean 语句创建的显式 JavaBean 对象，你可以通过下面所示的语法来显式地设置它的作用域：

```
scope=" scopevalue"
```

下面列举了四种可能的作用域范围：

- scope=“page”——对象只能在创建它的 JSP 页面中被访问。
  值得注意的是，当用户为了运行一个 JSP 页面而刷新它时，所有具有页面作用域的对象都要重新生成新的实例。
- scope=“request”——对象可以在与创建它的 JSP 页面监听的 HTTP 请求相同的任意一个 JSP 页面中被访问。
- scope=“session”——对象可以在与创建它的 JSP 页面共享相同的 HTTP 会话的任意一个 JSP 页面中被访问。
- scope=“application”——对象可以在与创建它的 JSP 页面属于相同的网络应用程序的任意一个 JSP 页面中被访问。

隐式对象

在 JSP 页面中，有很多的隐式对象可以直接使用，它们是由 JSP 机制所自动创建的 Java 类实例，通过这些隐式对象，JSP 页面可以实现与用户的动态交互。

下面列出了可用的隐式对象，关于这些对象的方法及如何使用它们，请参看 Sun 公司

的 Java 文档，这些文档可以在如下 URL 中找到：

```
http://java.sun.com/products/servlet/2.2/javadoc/index.html
```

- page
  page 对象是 JSP 页面实现类的一个实例，它在页面被解释的时候自动创建，在 JSP 页面中，page 与 this 是相同的。
- request
  request 对象是 javax.servlet.http.HttpSevrletRequest 接口实现类的一个实例，它代表着一个 HTTP 请求。（javax.servlet.http.HttpSevrletRequest 接口扩展了 javax.servlet.SevrletRequest 接口。）
- response
  response 对象是 javax.servlet.http.HttpSevrletResponse 接口实现类的一个实例，它代表着一个 HTTP 响应。（javax.servlet.SevrletResponse 接口扩展了 javax.servlet.SevrletResponse 接口。）
- pageContext
  pageContext 对象是 javax.servlet.jsp.PageContext 类的一个实例，它代表着 JSP 页面的上下文环境，用来存储和访问 JSP 页面中所具有 page 作用域的对象。
  Pagecontext 对象的作用域为 page，也就是说它只能在与其关联的 JSP 页面内被访问，而不能在其他地方访问。
- session
  session 对象是 javax.servlet.http.HttpSession 类的一个实例，它代表着一个 HTTP 会话。
- application
  application 对象是 javax.servlet.ServletContext 类的一个实例，它代表着网络应用程序的 servlet 环境。
  application 对象具有全局作用域，只要和应用程序在同一个 Java 虚拟机中，任何 JSP 页面都可以访问 application 对象。（程序员应该具有 Java 虚拟机和服务器架构的基础知识。例如，在 Oracle Servlet Engine 架构中，每个用户的应用程序都在他（或她）自己的 Java 虚拟机中运行。）
- out
  out 对象是 javax.servlet.jsp.JspWriter 类的一个实例，它用来将页面内容写到输出流中（javax.servlet.jsp.JspWriter 类扩展了 java.io.Writer 类）。
  对于一个特定的请求来说，out 对象被关联到此请求的 response 对象上。
- config
  config 对象是 javax.servlet.ServletConfig 类的一个实例，它代表着一个 JSP 页面的 servlet 配置信息。（一般来讲，servlet 容器在初始化期间使用 ServletConfig 类的实例为 servlet 提供信息，这些信息中有一部分就是 ServletContext 的实例。）
- exception（仅用于 JSP 错误处理页面）
  exception 对象是 java.lang.Exception 类的一个实例，它代表着 JSP 页面中没有被捕捉到的异常，这些异常被另外一个 JSP 页面抛出并且导致错误处理页面被调用。

此对象只适用于 JSP 错误处理页，所谓错误处理页就是当一个 JSP 页面抛出异常时所自动转向的页面。在错误处理页的实例中访问，提供一些关于异常的信息。

隐式对象的使用

上一小节所讨论的隐式对象是非常有用的，下面的例子介绍了如何使用 request 对象从 HTPP 请求参数中查询并显示用户名。

```
<H3> Welcome <%= request.getParameter("username") %> ! <H3>
```

1.3.4　JSP 的 Actions 和<jsp:> 标签集

当 JSP 页面运行时，JSP 的动作元素会导致某些类型的动作被触发，比如初始化一个 Java 对象并使它在此页面中可用，这样的动作元素包括下面几项：

- 创建一个 JavaBean 实例并且访问它的属性。
- 将页面转移到另外一个 HTTP 页面、JSP 页面或 servlet 中。
- 在 JSP 页面中包括一个外部的资源。

动作元素使用一套标准的 JSP 标签，它以<jsp: 做为开始标记。尽管使用在本章的前面所讨论的以<%语法开始的标签已经足够用来编写 JSP 页面，但是<jsp:标签仍然提供了更强大的功能和更方便的用法。

动作元素使用的语法和 XML 语句所使用的语法是相似的，都有开始和结束标签，如下例所示：

```
<jsp:sampletag attr1=" value1" attr2=" value2" ... attrN=" valueN">
... body...
</jsp:sampletag>
```

如果没有正文的话，动作元素以空标签来作为结束标记，如下例所示：

```
<jsp:sampletag attr1=" value1", ..., attrN=" valueN" />
```

JSP 标准包括下列标准的动作标签，它们在这儿只是被简单介绍和讨论了一下：

- jsp:useBean

  jsp:useBean 动作创建一个指定 JavaBean 类的实例，并且给实例指定一个名字，定义它的作用域。

  例子：

  ```
  <jsp:useBean id="pageBean" class="mybeans.NameBean" scope="page" />
  ```

  这个例子创建了类 mybeans.NameBean 的一个实例的属性（这些实例必须已经用 useBean 语句定义过）。

- jsp:setProperty

  jsp:setProperty 动作设置一个或多个 bean 属性（bean 必须是已在 UseBean 动作中指定的）。你可以直接指定某一个属性的值，或者从一个相关联 HTTP 请求中得到一个参数值赋给某一个属性，甚至还可以循环查询 HTTP 请求参数的值，把它们分别赋给多个属性。

  下面的例子将 pageBean 实例（在前面的 useBean 例子中定义）的 user 属性设成“Smith”：

  ```
  <jsp:setProperty name="pageBean" property="user" value="Smith" />
  ```

下面的例子根据HTTP请求中名字叫username的参数的值来设置pageBean的user属性：

```
<jsp:setProperty name="pageBean" property="user" param="username" />
```

或者，如果HTTP请求的参数名字和pageBean的属性名字相同（都为user），那么可以如下简单地设置user的属性：

```
<jsp:setProperty name="pageBean" property="user" />
```

下面的例子将导致循环查询HTTP请求参数，如果发现HTTP请求参数中有和pageBean的属性名字相同的，就将pageBean的那个属性值设为HTTP请求参数中对应的那个值：

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

- jsp:getProperty

  jsp:getProperty语句读出一个JavaBean实例的属性值，将它转换为Java字符串，并且将字符串的值放入输出缓冲中以便它能被显示出来。（此语句所谈的JavaBean字符串必需已经用jsp:useBean语句定义过。）为了使字符串转换能正确进行，简单类型（如int、char等）可以直接转换，而对象类型则必须调用在java.lang.Object类中指定的toString()方法来转换。

  下面的例子将pageBean的user属性的值填入输出对象中：

  ```
  <jsp:getProperty name="pageBean" property="user" />
  ```

- jsp:param

  jsp:param语句一般和jsp:include，jsp:forward或jsp:plugin语句联合使用，关于jsp:include，jsp:forward，jsp:plugin语句的用法将在稍后介绍。

  对于jsp:forward和jsp:include语句来说，jsp:param语句提供了名字/值这一可选语法来设置HTTP请求对象的参数值，如果指定了新的参数和值，它们将被添加到HTTP请求对象中，并且新的值将覆盖掉旧的值。

  下面的例子将HTTP请求对象中的username参数的值设为“Smith”：

  ```
  <jsp:param name="username" value="Smith" />
  ```

注意：在JSP 1.0标准中，jsp:include或者jsp:forward不支持jsp:param标签。

- jsp:include

  jsp:include语句将外部的静态或者动态资源插入到页面中，当此页面被请求显示时，插入的外部资源也将被显示出来。在指定资源文件时应该使用相对URI路径。（要么使用页面相对路径，要么使用应用程序相对路径。）

  在Sun公司的JSP 1.1标准中规定，使用jsp:include语句时必须将flash属性设为true，从而使得当jsp:include动作被执行时，缓冲区里的输出马上被送到浏览器里（在当前的JSP实现里，将flash属性设成flase是无效的）。

  你也可以在jsp:include的正文区用jsp:param来设定一些参数值，如下面两个例子所示：

  例子一：

  ```
  <jsp:include page="/templates/userinfopage.jsp" flush="true" />
  ```

  例子二：