Prentice Hall

advanced JAVASERVER PAGES™

JSP 高级开发与应用

- 用 servlet 和 JSP 技术设计和实现灵活、可扩展且易于维护的应用程序
- 掌握强大的身份验证和国际化技术
- 掌握 XML 和 XSLT 与 JSP 技术集成
- 基于 JSP 模板，通过模块化的组件开发健壮的应用程序

Sun microsystems

[美] David M.Geary 著　　贺民 译

Java™2 Platform, Enterprise Edition Series

092

Pearson Education

科学出版社

科海电脑技术丛书

JSP 高级开发与应用

(Advanced JavaServer Pages)

[美] David M.Geary 著

贺 民 译

科 学 出 版 社

2002

著作权合同登记号：01-2002-2467

内 容 简 介

本书深入详实地介绍了 JSP 高级编程，即使用 bean、servlet 和 JSP 设计和实现灵活、可扩展且易于维护的应用程序。

全书从说明如何实现 JSP 定制标记入手，随后介绍了 HTML 表单、JSP 模板、模型 1 和模型 2 结构、简单的模型 2 框架、事件处理、国际化、安全性、数据库和 XML，最后提供了一个完整的实例研究，进一步说明如何综合运用本书介绍的技术开发有价值的 Web 应用程序。

书中在介绍各种高级概念的过程中，提供了大量实用的代码，读者可直接使用。本书适用于具有 servlet 和 JSP 基础知识的 Java 开发人员。

图书在版编目（CIP）数据

JSP 高级开发与应用/（美）吉尔里（Geary,D.M.）著；贺民译 －北京：科学出版社，2002.7

ISBN 7-03-010678-4

Ⅰ.J… Ⅱ.①吉…②贺… Ⅲ.JAVA 语言－主页制作－程序设计

Ⅳ.TP393.092

中国版本图书馆 CIP 数据核字（2002）第 052156 号

科学出版社出版

北京东黄城根北街16号

邮政编码：100717

北京朝阳科普印刷厂印刷

科学出版社发行　各地新华书店经销

*

2002 年 9 月第一版　开本：异 16

2002 年 9 月第一次印刷　印张：24.875

印数：1-4 000　字数：557 200

定价：42.00 元

前　言

1999 年 3 月发布图形 Java 中的 Swing 卷之后不久，Java 客户端的大批研究者转而去研究 Java 服务器端的开发和应用。在认真分析了这种情形后，我开始研究服务器端 Java 技术，这方面的技术将是我下一本书的内容。起先，我迷上了 XML、XSLT 和 Java 技术，并且花了大量的时间研究这些技术。尽管这些技术都各有其独到之处，但是，它们不能直接用来开发 Web 应用程序，而我需要的是能够直接开发 Web 应用程序的技术。后来，我找到了这种技术，它就是 servlet。

说实话，我对 servlet 并不是十分满意。软件开发人员真打算通过一些 servlet 中 print（打印）语句产生的 HTML 程序来创建用户界面吗？至少我这个软件开发人员不是这样做的。自从 1984 年以来，我使用过许多面向对象的语言和非常优秀的用户界面工具包开发软件。我曾经用 Smalltalk、Eiffel 和 NeXTSTEP 开发过应用程序，就我自身的体验来讲，使用 HTML，尤其是使用从 servlet 手工生成的 HTML 来开发应用程序，简直是不可能的事情。接下来，我发现了另外一种可以直接开发 Web 应用程序的技术，即 JSP。

虽然 1999 年 JSP 技术还处于发展的初期，但它的潜力是显而易见的。JSP 将 Java 语言和 HTML 语言结合起来，处理各种有趣的问题。在 JSP 1.0 规范的 Future Directions 中，下面这句话引起了我的注意：JSP 1.0 规范将考虑增加可移植的标记扩展机制，该机制允许在任何 JSP 页面中使用标记描述。这真是太好了。使用定制标记可以封装 Java 代码，当然也就可以以标记的形式创建定制组件，这样，封装的 Java 代码就可以与 HTML 联合起来使用。从那时起，我便希望编写一本有关 JSP 的书。

于是，我开始编写 JSP 的入门书。当写完本书的第 1 章时，我认识到两点：首先，市面上 JSP 的入门书已经非常多了，我不希望本书重蹈别人的覆辙；其次，也是更重要的是，第 1 章写得比较晦涩，我不喜欢阅读晦涩的书，因此就放弃了写 JSP 入门书，而决定写这本更实用的书来替代 JSP 入门书。

本书内容

从书名可以看出，本书介绍 JSP（JavaServer Pages，Java 服务器页面）高级编程方面的内容，中心主题是使用 bean、servlet 和 JSP 设计和实现灵活、可扩展且易于维护的应用程序。

本书首先介绍如何实现 JSP 定制标记，其中的内容是多数 JSP 入门书所没有的。创建定制标记的能力证明了 JSP 功能的强大，因为通过创建定制标记，软件开发人员和页面制作者

可以并行工作，而无需相互依赖。随后的各章介绍了 HTML 表单、JSP 模板、模型 1 和模型 2 结构、简单的模型 2 框架、事件处理、国际化、安全性、数据库和 XML。本书最后提供了一个完整的案例研究，说明如何使用本书介绍的技术开发有价值的 Web 应用程序。

本书使用的servlet和JSP API

本书代码符合 Servlet 2.2 和 JSP 1.1 规范。尽管 Servlet 2.3 和 JSP 1.2 规范在 2000 年 11 月首次发布了草案，但在本书即将出版时，Servlet 2.3 和 JSP 1.2 规范仍处于不断变化的状态。由于 servlet 过滤器是 Servlet 2.3 规范最重要的新增功能，所以在附录 A 中介绍了该主题。值得注意的是，当你读到这本书时，附录中包含的代码非常可能会发生变化。

本书代码的测试方式

本书的所有代码都用 Tomcat 3.2.1 进行过测试。如果使用 Tomcat 3.2.1 时不能正常工作，比如 9.3 节的例子，书中会指出来。

由于 Tomcat 是 Servlet 和 JSP 规范的参考实现工具，所以，本书的所有代码应该能够适用于符合 Servlet 2.2 和 JSP 1.1 或更高版本规范的任何 servlet 容器。如果本书示例不能与你的 servlet 容器一起工作，极有可能是 servlet 容器存在问题。

本书的所有代码还利用 Resin 1.2 进行了测试，Resin 1.2 是非常优秀的 servlet 容器，访问 http://www.caucho.com 站点可以获得它。通常来说，应当针对多个 servlet 容器测试代码的正确性和可移植性。

本 书 读 者

本书适用于具有 servlet 和 JSP 基础知识的 Java 开发人员。对多数 Java 开发人员来说，在学习了有关 servlet 和 JSP 入门书之后，就应当阅读本书。而 servlet 和 JSP 新手应当首先阅读下列书籍：

- 《Core Servlets and JSP》，作者 Marty Hall，由 Sun Microsystems Press 出版
- 《Java Servlet Programming》，作者 Jason Hunter，由 O'Reilly 出版
- 《Web Development with JavaServer Pages》，作者是 Fields 和 Kolb，由 Manning 出版

针对基本理解了设计模式和 UML（统一建模语言）的读者，本书演示了在基于 JSP 的 Web 应用程序中如何实现多种设计模式，并使用 UML 类图和顺序图表明类之间的相关性和相互作用。有关设计模式和 UML 的内容，请参见本书第 6 章最后列出的参考书目。

本书不适用于页面制作者。对于没有 Java 背景知识的页面制作者来讲，可选择阅读前面列出的任何一本书。

本书的编写方式

面向对象软件的设计过程是一个需要多次反复的过程。为了建立不断发展的系统，开始时需要创建几个类，接着就是一直操作这些类，对以前创建的类和刚刚创建的类都反复操作，以此建立一个不断完善的系统。用面向对象的说法，这个过程叫做“重构”(refactor)。

我做过 15 年的软件工程师，通常以编写软件代码的方式来写书。本书的每一章都很通俗易懂，最后将各章连接成一个最终的产品，就是你手中的这本书了。

有关本书的编写方式，可以参见我在 JavaWorld 上写的介绍 JSP 模板的文章。那篇文章取自本书介绍模板的第 4 章，可以从中看出“模板”一章的来龙去脉。该章及其使用的代码都支持重构。

本书的阅读方法

本书不是小说，我不希望读者从头到尾阅读。考虑到多数读者不会按本书的顺序阅读各章节，所以本书的每一章几乎都是独立的，只有一个例外，即第 6 章，其中介绍了简单的模型 2 框架，它以介绍模型 2 结构的第 5 章为基础。第 6 章是第 5 章示例的深入展开，而第 5 章则是第 6 章的基础。

本书的最后一章提供了一个综合的案例研究，该案例使用本书介绍的所有技术来实现一个有价值的 Web 应用程序。你可以先阅读或者浏览一下这一章，感受一下本书介绍的技术；也可以最后再读这一章，以便掌握如何综合运用本书介绍的技术。当然也可以尝试两种方式一起使用。

本书的定制标记库

本书讨论了大约 50 个 JSP 定制标记的实现方法，范围从国际化的标记到使用 XML 的文档对象模型分析 XML 的标记。这些标记没有法定的限制，因此，你可以以自己认为合适的方式使用这些标记。有关如何下载这些标记的内容，请参见随后的“本书代码”。

本书介绍定制标记有两个目的。首先，说明如何实现自己的定制标记。其次，加深读者对本书概念的理解。需要注意的是，定制标记不是本书的重点，而标记所涉及的概念才是重要的。例如，在介绍国际化的章节中，大部分内容是在介绍基于 JSP 的 Web 应用程序中如何对文本、数字、日期和货币进行国际化处理，最后几页介绍了如何实现完成国际化的两个定制标记。该

章的核心是国际化的概念，而不是定制标记。

本书代码

本书的所有代码以及定制标记库可以从 http://www.phptr.com/advjsp 下载。

本书约定

下表列出了本书编码约定。

编码约定

约定	示例
类名首字母大写	public class ClassName
方法名第一个单词首字母小写，其他单词首字母大写	getLength
变量名第一个单词首字母小写，其他单词首字母大写	private int length, private int bufferLength

注意，对本书而言，引用的大部分方法是无参量的，但在讨论这些方法时，也介绍了相关的参量。

目 录

第 1 章　定制标记基础

XML 是一种热门技术，这主要是因为它是一种创建标记的简单元语言。XML 标记表示特殊域的数据，例如，下面的 XML 代码段表示了一个 CD 集合：

```
<cd_collection>
  <cd>
    <artist>Radiohead</artist>
    <title>OK Computer</title>
    <price>$14.99</price>
  </cd>
  ...
</cd_collection>
```

与 XML 相似，JSP 也可以用来创建标记。但不同的是，XML 标记表示的是数据，而 JSP 定制标记表示的是特殊域的功能[①]。例如，示例 1.1 列出的 JSP 代码段使用定制标记显示数据库中的表：

示例 1.1　使用定制标记访问数据库

```
<html><title>Database Example</title>
<head>
   <%@ taglib uri='/WEB-INF/tlds/database.tld' prefix='database'%>
</head>
<body>

<database:connect database='F:/databases/sunpress'>
  <database:query>
    SELECT *  FROM Customers, Orders
  </database:query>

  <table border='2' cellpadding='5'>
    <database:columnNames columnName='column_name'>
      <th><%= column_name%></th>
    </database:columnNames>

    <database:rows><tr>
      <tr><database:columns columnValue='column_value'>
```

① 术语“定制标记”将系统内置的标记与用户实现的（定制）标记区分开来。XML 没有内置的标记，因此也就没有必要进行区别。

```
            <td><%= column_value %></td>
          </database:columns></tr>
        </database:rows>
      </table>
    </database:connect>

    </body>
    </html>
```

示例 1.1 使用 HTML 和 JSP 定制标记来创建 HTML 表。connect 标记创建数据库连接，query 标记执行数据库查询，rows、columns 和 columnNames 对查询结果进行迭代。

如示例 1.1 所示，JSP 定制标记非常复杂。query 标记按照 SQL 解释其主体内容，并且，connection 和 query 标记要与页面中的其他标记协同工作。columns 和 columnNames 标记在查询结果中迭代，并创建由 JSP 开发人员命名的脚本编程变量。在示例 1.1 的代码段中，创建的脚本编程变量名是 column_name 和 column_value。

定制标记具有如下特征：

- 可以有主体，也可以为空
- 可以以任意深度嵌套到其他定制标记中
- 进行流控制，例如，利用 if 语句和迭代
- 处理主体的内容，例如过滤和编辑
- 可以在同一页面与其他标记协同工作
- 访问页面信息（请求、响应和会话等）
- 创建脚本编程变量

JSP 定制标记遵守 XML 规范，因此，JSP 定制标记实际上也是 XML 标记。这非常重要，因为这意味着可以在 XML 和 HTML 工具中使用 JSP 定制标记。

在示例 1.1 中，所有定制标记的开始标记和结束标记之间都包含主体内容，但定制标记也可以只包含开始和结束标记，而主体内容是空的，如下所示：

```
<prefix:someTag/>
```

所有的 JSP 定制标记按组存放在标记库中。使用与库相关的前缀区分这些定制标记，这样就可以同时使用不同库中的同名标记。

JSP 1.1 规范正着手按照新版本规范设置标准标记库[①]，标准标记库中将包含用于数据库访问的标记和许多具有其他用途的标记。

> **JSP 提示：定制标记是 JSP 最强大的功能**
>
> 利用定制标记，软件开发人员和页面设计人员可以独立地自由工作。页面设计人

① 最初是 JSR（Java 规范请求），请参见 http://java.sun.com。

员可以将精力集中在使用标记集（比如 HTML、XML 或 JSP）创建网站上，而软件开发人员则可以将精力集中在实现底层功能上，如国际化或数据库访问，这样，页面设计人员随后就能够以定制标记的形式访问数据库。

1.1　使用定制标记——JSP文件

图 1.1 显示了一个 JSP 页面，这个页面使用了最简单的定制标记，该标记既没有属性，也没有主体。这个标记是一个计数器，用于计算访问 JSP 页面的次数。

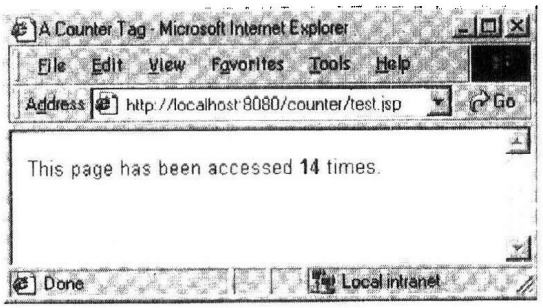

图 1.1　计数器标记

示例 1.2.a 中给出了与图 1.1 对应的 JSP 文件的代码。

示例 1.2.a　/test.jsp

```
<html><head><title>A Counter Tag</title></head>
<body>
<%@ taglib uri='/WEB-INF/tlds/counter.tld' prefix='util' %>
This page has been accessed <b><util:counter/></b> times.
</body></html>
```

在示例 1.2.a 中，taglib 指令指定了定义标记库及其标记的 URI，这个 URI 指向一个标记库描述信息（TLD,Tag Library Descriptor），也就是 counter.tld 的 tld 扩展。taglib 指令还需要用 prefix 属性来指定用于访问该库标记的前缀，在示例 1.2.a 中，这个前缀是 util，因此可以用 <util:counter/>来访问计数器标记。

示例 1.2.a 中的 TLD 位于 WEB-INF/tlds 下。虽然这里不是必须要定位 TLD，不过我们还是推荐你这样做。

1.2 定义定制标记——TLD

TLD 是定义一个标记库及其标记的 XML 文档。示例 1.2.b 列出了示例 1.2.a 中所用标记的 TLD 文件。

示例 1.2.b /WEB-INF/tlds/counter.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC
  "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
  "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>Sun Microsystems Press Tag Library</shortname>
    <info>This tag library has a single counter tag</info>

    <tag>
        <name>counter</name>
        <tagclass>tags.CounterTag</tagclass>
        <bodycontent>empty</bodycontent>
    </tag>
</taglib>
```

在上面的 tld 文件中，第 1 行表示这是一个 XML 文档，第 2 行进一步表示文档类型是 taglib，并提供了文档类型定义（DTD,Document Type Definition）的 URL，该 DTD 定义了 taglib 文档的结构。servlet 容器使用这个 DTD 来验证该 taglib 文档。

可以使用<taglib>定义标记库。在示例 1.2.b 中，标记库的版本为 1.0，所以，只有符合 JSP 1.1 或者更高版本的 JSP 实现规范，才能使用该标记库。

除了列出标记库的一些有关信息外，示例 1.2.b 还给出了标记库的简称，页面设计工具通常要使用这些值。

在上例中，<tag>用于定义标记，它具有两个重要的元素：标记的<name>和<tagclass>。<tagclass>指定实现标记功能的 Java 类，这些类的类型通常叫做标记处理程序。

上例中，计数器标记的主体内容为空，这意味着计数器标记具有主体是不合法的。

<taglib>和<tag>除了上面使用的元素之外，还具有许多元素。有关这两个标记及其元素的详细内容，请参见 1.4 节。

1.3 实现定制标记——标记处理程序

标记处理程序通过 javax.servlet.jsp.tagext 实现 Tag 接口。Tag 接口定义了 6 种方法，下面

列出了 3 种最常用的方法[①]：

```
int doStartTag() throws JspException
int doEndTag() throws JspException
void release()
```

servlet 容器按上面的顺序依次调用 Tag 方法。标记的开始和结束部分分别调用 doStartTag 和 doEndTag 方法，这两个方法均返回 integer 常量值，该值是在 Tag 接口中定义的，返回值用来指示 servlet 容器应该如何继续操作。

调用 doEndTag 方法后，servlet 容器调用 release 方法。release 方法将标记处理程序保持的所有资源都释放掉。

示例 1.2.c 中给出了计数器标记的处理程序，我们一起来看看这个处理程序是如何实现上面列出的 Tag 方法的。

示例 1.2.c　/WEB-INF/classes/tags/CounterTag.java

```
package tags;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;
import javax.servlet.http.HttpServletRequest;

public class CounterTag extends TagSupport {
  private int count = 0;
  private File file = null;

  public int doStartTag() throws JspException {
    try {
      checkFile();
      readCount();
      pageContext.getOut().print(++count);
    }
    catch(java.io.IOException ex) {
      throw new JspException(ex.getMessage());
    }
    return SKIP_BODY;
  }
  public int doEndTag() throws JspException {
    saveCount();
    return EVAL_PAGE;
```

① 有关 Tag 接口的详细信息，请参见 1.11.1 小节。

```
    }
    private void checkFile() throws JspException, IOException {
      if(file == null) {
         file  = new File(getCounterFilename());
         count = 0;
      }
      if(!file.exists()) {
         file.createNewFile();
         saveCount();
      }
    }
    private String getCounterFilename() {
      HttpServletRequest req = (HttpServletRequest)pageContext.
                                                    getRequest();
      String servletPath = req.getServletPath();
      String    realPath = pageContext.getServletContext().
                                getRealPath(servletPath);

      return realPath + ".counter";
    }
    private void saveCount() throws JspException {
      try {
         FileWriter writer = new FileWriter(file);
         writer.write(count);
         writer.close();
      }
      catch(Exception ex) {
         throw new JspException(ex.getMessage());
      }
    }
    private void readCount() throws JspException {
      try {
         FileReader reader = new FileReader(file);
         count = reader.read();
         reader.close();
      }
      catch(Exception ex) {
         throw new JspException(ex.getMessage());
      }
    }
}
```

通过在文件中存储计数，上述标记处理程序记录了访问标记（也就是标记的 JSP 页面）的次数。这类文件的名称与相应的 JSP 页面文件名相同，只是多了.counter 后缀。例如，文件/index.jsp 使用一个计数器标记，而相应的/index.jsp.counter 文件将包含一个计数，说明访问/index.jsp 页面的次数。