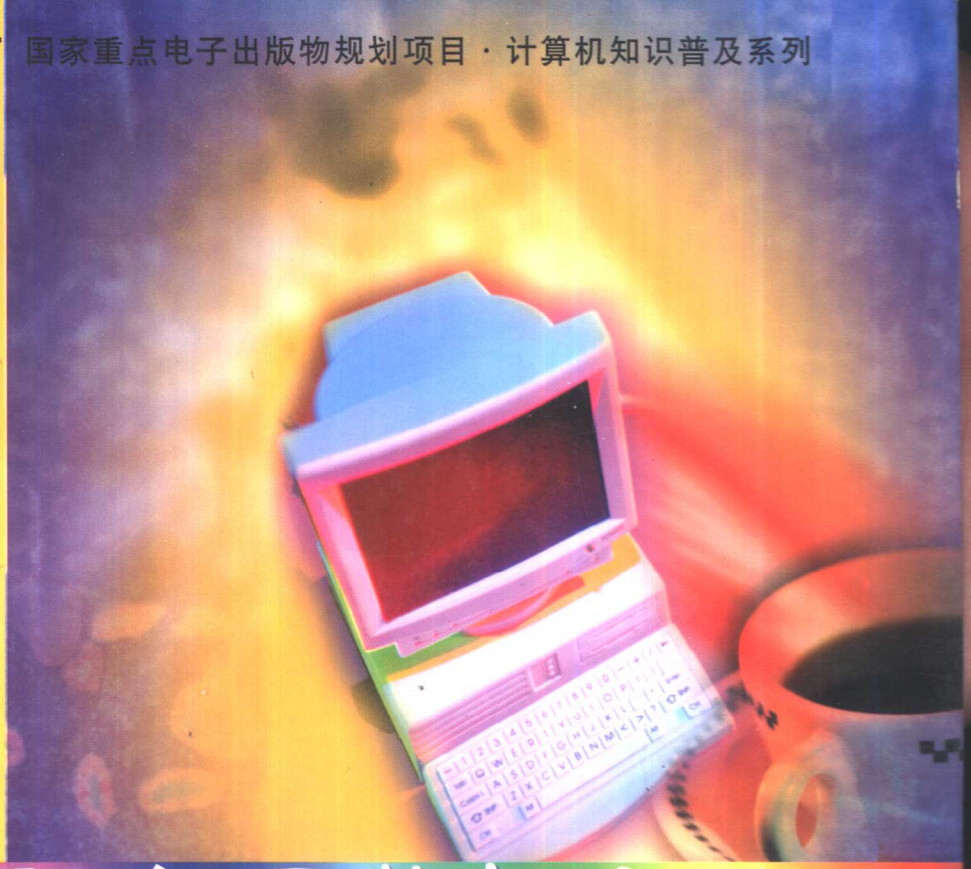
“九五”国家重点电子出版物规划项目·计算机知识普及系列



编程高手成长之路 6

Advanced Programming with JavaServer Pages

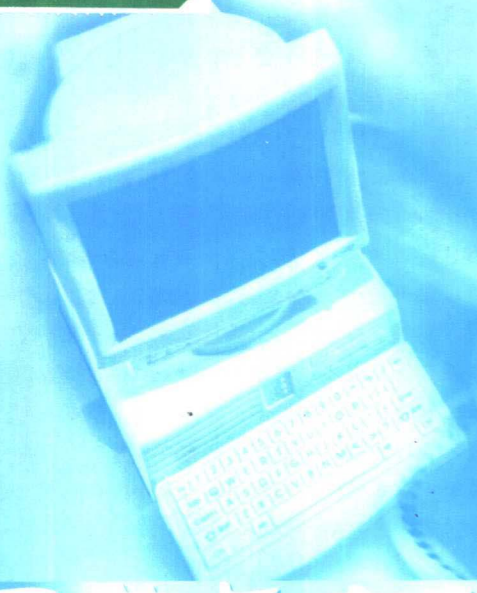
JSP 高级编程

北京希望电子出版社 总策划
北京大学 com 工作室 创作
黄 理 洪 亮 曹林有 张 勇等 编著

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

Advanced Programming with JavaServer Pages

"九五"国家重点电子出版物规划项目·计算机知识普及系列



编程高手成长之路 6

Advanced Programming with JavaServer Pages

JSP 高级编程

北京希望电子出版社 总策划
北京大学com工作室 创 作
黄 理 洪 亮 曹林有 张 勇等 编 著

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

内 容 简 介

这是一本面向中、高级编程人员的自学指导书，其重点放在JSP和其他技术的综合使用方面。全书可分为四大部分：第一部分是JSP技术与J2EE技术（第一章至第四章），着重介绍J2EE技术的代表——EJB技术的基本原理与开发EJB组件的方法。第二部分是JSP技术和XML技术（第五章至第八章），主要介绍了XML技术与JSP技术联合的方式之一——Tag Library。第二部分后面的两章是针对市场上最流行的两个Tag Library的详细介绍，读者可以把它当作参考手册来使用。本作品的第三部分是JDBC新技术及其在JSP/Servlet程序中的应用（第九章和第十章），主要介绍最新的JDBC技术，如JDBC 2.0 JDBC 3.0的新特性，以及鲜为人知而又十分重要的JDBC Optional Pack。本作品的第四部分是JSP网络程序设计，着重介绍如何使用sun.net包、JavaMail API开发访问各种网络服务的JSP程序。本作品四个部分之间互为关联又相对独立，读者可以选择阅读某一个部分或者是通读全文。

本版作品是由经验丰富的使用JSP组建网站的程序员编著，内文附有丰富的实例源码，供读者学习参考。全书具有语言简明扼要，内容丰富，范例典型，理论与实践相结合的特点，不但是从事用JSP进行网站开发和设计的初、中级读者的自学指导书，同时也可作为社会网页设计或编程培训班的教材。

说明：与本书配套的面向初、中级用户的书《JSP深入编程》也已正式出版，欢迎选购。

本版CD为配套书。

系 列 书 名："九五"国家重点电子出版物规划项目　希望计算机知识普及系列
编程高手成长之路（6）
书　　名：JSP高级编程　Advanced Programming with JavaServer Pages
总 策 划：北京希望电子出版社
文本著作者：北京大学com工作室 创作　黄理 洪亮 曹林有 张勇等　编著
责任编辑：马红华
CD制作者：希望多媒体开发中心
CD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）　010-62613322-308（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　杜海燕
CD生产者：北京中新联光盘有限责任公司
文本印刷者：北京媛明印刷厂
开本/规格：787毫米×1092毫米　1/16开本　38.75印张　900千字
版次/印次：2001年10月第1版　2001年10月第1次印刷
印　　数：0001—5000册
本 版 号：ISBN 7-980007-78-6
定　　价：55.00元（本版CD）

前　言

JSP（JavaServer Pages）是目前十分流行的一种技术，主要运行于开发服务端的脚本程序和动态生成网站的内容。它与目前同样流行的 ASP 技术、PHP 技术是同样性质的、同一层次的，它们在网站的建设中所起到的作用是一样的。但是 JSP 技术与后面两种技术相比，有着十分突出的优越性。关于 JSP 技术与 ASP 技术、PHP 技术的比较，我们在书中另有论述，这里只想强调一点：JSP 技术有 J2EE 平台支持，发展前途不可限量。众所周知，J2EE 平台提供了 Java 企业应用编程接口（Java Enterprise APIs），为企业计算以及电子商务应用系统提供了有关的技术和强大的类库支持，J2EE 平台包含十几种技术，JSP 技术正是其中的一种核心技术。J2EE 的发展势头十分迅猛，在可以预见的将来，Sun 的 J2EE 平台可能是唯一可以与微软的.Net 构架相互抗衡的平台。在这个意义上说，基于 J2EE 平台的 JSP 技术与基于.Net 平台的 ASP•ASP+技术之争，不正好就是 J2EE 平台与.Net 平台之争的折射吗？因此，JSP 技术以及它的基础——J2EE 技术十分值得我们去关注。在国外，采用 JSP+J2EE 技术构架电子商务网站已经是大行其道了，应用得十分普遍。在国内，这一项技术还是方兴未艾，采用这一项技术架构的网站还不多，不过大致的趋势已经出现了，这真是一个令人兴奋的消息。为了帮助广大读者了解 JSP/J2EE 技术并掌握 JSP/J2EE 技术，我们编写了《JSP 深入编程》和《JSP 高级编程》这两本书。前者侧重于 JSP 技术的基础知识与基本应用，后者侧重于 JSP 技术和其他技术联合使用，本书正是其中第二本书。

本书按顺序讲述了以下知识点：

- JavaBeans 的基础知识。
- EJB 的结构框架。
- 会话 EJB 的开发•部署•测试•应用。
- CMP 模式，BMP 模式的实体 EJB 的开发、部署、测试、应用。
- J2EE 体系结构。
- CORBA,RMI,JNDI 技术入门。
- XML,XSL,CSS 语法介绍。
- WML,XHTML 简介。
- XML+JSP 的开发模式。
- Tag Library 的开发、应用、运行原理。
- javax.servlet.jsp.tagext 包的详细说明。
- JRun Tag Library 的使用手册。
- Jakarta Tag Library 的使用手册。
- JDBC 2.0/3.0 新特性介绍。
- JDBC Optional Pack 介绍（含 RowSet 包、CachedRowSet 包的介绍）。
- JSP 网络程序开发（访问 SMTP,FTP,News 等服务，Socket 技术应用介绍）
- JavaMail 技术完全指南。

总的来说，本书可以分为四大部分：第一部分是 JSP 技术与 J2EE 技术（第一章至第四章），着重介绍 J2EE 技术的代表——EJB 技术的基本原理与开发 EJB 组件的方法。第二部分是 JSP 技术和 XML 技术（第五章至第八章），主要介绍了 XML 技术与 JSP 技术联合的方式之一———Tag Library。第二部分后面的两章是针对市场上最流行的两个 Tag Library 的详细介绍，读者可以把它当作参考手册来使用。本书的第三部分是 JDBC 新技术及其在 JSP/Servlet 程序中的应用（第 9 章和第 10 章），主要介绍最新的 JDBC 技术，如 JDBC 2.0/JDBC 3.0 的新特性，以及鲜为人知而又用处极大的 JDBC Optional Pack。本书的第四部分是 JSP 网络程序设计，着重介绍如何使用 sun.net 包、JavaMail API 开发访问各种网络服务的 JSP 程序。本书这四个部分之间互为关联又相互独立，读者可以单独阅读某一个部分或者是通读全书。

顾名思义，本书不是关于 JSP 技术的入门书籍，本书要求读者必须有 JSP,Java 基础，否则阅读起来可能会有很大的困难。作者建议读者不妨参考《JSP 深入编程》，因为这两本书是配套编写的，在知识体系结构上有一定的承接性。

本书虽然名为《JSP 高级编程》，但是真正涉及到 JSP 程序编写技巧方面的章节并不多，这是因为 JSP 技术的核心内容很少很少，除了基本语法、编译指令、操作指令和内部对象以外，就没有别的东西了，要发挥 JSP 技术的长处，开发功能强大的 JSP 程序，单单靠 JSP 技术本身是不可能的。JSP 技术必须和其他相关的 Java 技术结合起来，例如 JDBC,EJB,RMI,CORBA,JavaMail 等技术，才有可能开发出功能强大的程序。本书重点介绍的就是上述技术的基本原理和开发方法，至于如何把这些技术和 JSP 技术结合起来，开发运行于服务端的应用程序与 JSP 程序，书中讲的很少，但是读者应该有这方面的经验。况且只要明了这些技术的基本原理与开发的方法，把它们和 JSP 技术结合起来是一件十分简单的事情，不需要浪费过多的笔墨去介绍这方面的知识。

当你读完本书以后，我们不能够保证你一定能够成为 JSP 高手，因为本书提到的技术虽然很多，但是由于篇幅的关系以及其他的原因，这些技术讲的都很肤浅，只是相当于入门的水平。读者如果想有更大的进步，最好是深入研究本书所提到的技术，找几个项目来做，当你能够游刃有余地应用这几种技术于 JSP 程序的开发中时，那时你才是真正的精通 JSP 的高手。本书给读者指出努力的方向以及提供入门的知识，剩下的就靠读者自身的努力了。这就是本书命名为《JSP 高级编程》的原因。

本书由北京大学 com 工作室组织编写。由于时间仓促，笔者的水平有限，书中的谬误一定很多。不足之处，请读者指正。

本书的成功出版，首先归功于本书的主要作者北大黄理同学，他深厚的计算机理论积累和丰富的实践经验才使得本书兼具理论指导及实务操作性，其工作的严谨态度以及出色的语言驾驭功底相信读者在阅读本书时自有体会；其次还有感谢北大洪亮同学，其出色的工作为本书增色不少。也感谢其他许许多多人辛勤的劳动与无私的帮助，轻易便可以列出很多：北大计算机系的李积善、水木清华（smth.org）的 javafancy、北大未名站的 javalover，还有 ROBBY•lz.lan、snowleaf 以及可爱的 Rainbow。

本书技术支持的联系方式：

com_pku@263.net

http：//162.105.106.162：8080 注：访问前，需要事先 mail 联系，以便启动服务器。

目 录

第一部分 JSP 技术与 J2EE 技术

第二部分 JSP 技术和 XML 技术

第三部分 JDBC 新技术及其在 JSP/Servlet 中的应用

第四部分 JSP 网络程序设计

第一部分　JSP 技术与 J2EE 技术

第 1 章　JavaBeans 组件技术

本章将要向读者介绍 JavaBeans 组件技术在 JSP 程序开发中的应用。在《JSP 深入编程》中，我们已经介绍了一点关于 JavaBeans 的知识，但是由于体系结构的原因，我们并没有深入讨论它，也许有的读者对此还有些遗憾，不过不要紧，这一章就来弥补读者的这个遗憾。本章中读者需要重点掌握的内容有：

- JavaBeans 的属性
- JavaBeans 的事件模型
- JSP 中与 JavaBeans 相关的操作指令的语法与用法
- JavaBeans 的开发流程
- JavaBeans 的 Scope 属性
- JavaBeans 封装数据库操作

1.1　什么是 JavaBeans

1.1.1　JavaBeans 简介

软件开发的真正目的之一是利用在程序编码方面的投资，以便在同一公司或者不同公司的其他开发中重用程序编码。近年来，编程人员投入大量精力以便建立可重用的软件、可重用的软件组件。早期用在面向对象编程方面中的投资已经在 Java、C#等编程语言的开发中充分实现，很多软件可以不用做很大的改变就可以运行在各种平台上。

JavaBeans 描述了 Java 的软件组件模型，这个模型被设计成使第三方厂家可以生成和销售能够集成到其他开发厂家或者其他开发人员开发的软件产品的 Java 组件。

应用程序开发者可以从开发厂家购买现成的 JavaBeans 组件，拖放到集成开发环境的工具箱中，再将其应用于应用软件的开发，对于 JavaBeans 组件的属性、行为可以进行必要的修改、测试和修订而不必重新编写和编译程序。在 JavaBeans 模型中，JavaBeans 组件可以被修改或者与其他 JavaBeans 组件组合以生成新的 JavaBeans 组件或完整的 Java 应用程序。

Java 应用程序在运行时，最终用户也可以通过 JavaBeans 组件设计者或应用程序开发者所建立的属性存取方法（setXXX 方法和 getXXX 方法）修改 JavaBeans 组件的属性，这些属性可能是颜色和形状等简单属性，也可能是影响 JavaBeans 组件总体行为的复杂属性。

JavaBeans 组件模型使得软件可以设计成便于修改和便于升级。每个 JavaBeans 组件都

包含了一组属性、操作和事件处理器。将若干个 JavaBeans 组件组合起来就可以生成设计者、开发者所需要的特定运行行为，JavaBeans 组件存放于容器或工具库中，供开发者开发应用程序。

JavaBeans 就是一个可以复用软件模型。JavaBeans 在某个容器中运行，提供具体的操作性能。JavaBeans 是建立应用程序的建筑模块。大多数常用的 JavaBeans 通常是中小型控制程序，但我们也可以编写包装整个应用程序运行逻辑的 JavaBeans 组件 ，并将其嵌入到复合文档中，以便实现更为复杂的功能。

一般来说，JavaBeans 可以表示为简单的 GUI 组件，可以是按钮组件、游标、菜单等等。这些简单的 JavaBeans 组件提供了告诉用户什么是 JavaBeans 的直观方法。但我们也可以编写一些不可见的 JavaBeans，用于接受事件和在幕后工作，例如访问数据库，执行查询操作的 JavaBeans，它们在运行时刻不需要任何可视的界面。在 JSP 程序中所用的 JavaBeans 一般以不可见的组件为主。可见的 JavaBeans 一般用于编写 Applet 程序或者 Java 应用程序。

1.1.2 JavaBeans 属性

JavaBeans 的属性与一般 Java 程序中所指的属性，或者说与所有面向对象的程序设计语言中对象的属性是同一个概念，在程序中的具体体现就是类中的变量。在 JavaBeans 的设计中，按照属性的不同作用又细分为 4 类：Simple 属性、Index 属性、Bound 属性与 Constrained 属性。

Simple 属性

一个 Simple 类型的属性表示一个伴随有一对 getXXX()、setXXX()方法的变量。属性的名称与和该属性相关的 getXXX()、setXXX()方法相对应。例如：如果有 setX()和 getX()方法，则暗指有一个名为"X"的属性，如果有一个方法名为 isX()，则通常暗指"X"是一个布尔类型的属性。请看下面的程序清单 1.1(JavaBean1.java)。

程序清单 1.1

```
//File Name:JavaBean1.java
//Author:fancy
//Date:2001.3.29
//Note:create a simple javabean

public class JavaBean1
{
    String ourString= "Hello";

    public JavaBean1()
    {

    }

    public void setoutString(String newString)
```

```
    {
        ourString=newString;
    }
    public String getoutString()
    {
        return ourString;
    }
}
```

在程序清单 1.1(JavaBean1.java)中，我们定义了一个 JavaBean——JavaBean1，其实也就是定义了一个 JavaBean1 类。JavaBean1 有一个名为 outString 的字符串类型的属性。与这个属性相对应的方法为 setoutString()和 getoutString()，使用这两个方法可以存取 outString 属性的值。

Indexed 属性

一个 Indexed 类型的 JavaBeans 属性表示一个数组值。使用与该属性相对应的 setXXX()方法和 getXXX()方法可以存取数组中某个元素的数值。同时，我们也可以使用另两个同名方法一次设置或取得整个数组的值(即属性的值)。请看程序清单 1.2。

程序清单 1.2

```
//File Name:JavaBean2.java
//Author:fancy
//Date:2001.3.29
//Note:create a indexed javabean

public class JavaBean2
{
    int[] dataSet={1，2，3，4，5，6};

    public void JavaBean2()
    {
    }

    public void setDataSet(int[] x)
    {
        dataSet=x;
    }

    public void setDataSet(int index，int x)
    {
        dataSet[index]=x;
    }

    public int[] getDataSet()
```

```
    {
        return dataSet;
    }

    public int getDataSet(int x)
    {
        return dataSet[x];
    }
}
```

在程序清单 1.2(JavaBean2.java)中，定义了 JavaBean——JavaBean2，JavaBean2 具有属性 dataSet，dataSet 属性是一个整型数组，JavaBean2.java 定义了 4 个方法以存取 dataSet 属性的值，它们分别是： setDataSet(int[] x)、setDataSet(int index，int x)、getDataSet(int x)、getDataSet()，其中 setDataSet(int[] x)方法可以一次设定 dataSet 属性的值，getDataSet()方法可以一次获取 dataSet 属性的值，该方法的返回值是一个整型数组，getDataSet(int x)方法可以获取 dataSet 属性中某个指定的元素的值，该方法的返回值为整型数据，与这个方法相对的方法是 setDataSet(int index，int x)方法，使用这个方法可以指定 dataSet 属性中某个特定元素的值。

Bound 属性

一个 Bound 类型的 JavaBean 组件的属性具有这样的特性：当该种属性的值发生变化时，必须通知其它的 JavaBeans 组件对象。每次 JavaBeans 组件对象的属性值改变时，这种属性就引发一个 PropertyChange 事件(属性改变事件，在 Java 程序中，事件也被看作是一个对象)。这个事件中封装了发生属性改变事件的属性名、属性的原值、属性变化后的新值。这个事件将被传递到其它的 JavaBeans 组件中，至于接收事件的 JavaBeans 组件对象应该做什么动作由其自己定义。请看程序清单 1.3(JavaBean3.java)。

程序清单 1.3

```
//File Name:JavaBean3.java
//Author:fancy
//Date:2001.3.29
//Note:create a bound javabean

import java.beans.*;
public class JavaBean3
{
    String ourString= "Hello";
    private PropertyChangeSupport changes = new PropertyChangeSupport(this);
    public void setString(String newString)
    {
        String oldString = ourString;
        ourString = newString;
        changes.firePropertyChange("ourString", oldString, newString);
```

```
    }

    public String getString()
    {
        return ourString;
    }

    public void addPropertyChangeListener(PropertyChangeListener l)
    {
        changes.addPropertyChangeListener(l);
    }

    public void removePropertyChangeListener(PropertyChangeListener l)
    {
        changes.removePropertyChangeListener(l);
    }
}
```

读者对程序清单 1.3(JavaBean3.java)的运行逻辑一定感到十分迷惑吧，那好，下面我们就来详细解释 JavaBean3.java 程序的含义。程序首先创建了 PropertyChangeSupport 类型的对象 changes，这是最关键的一步操作，changes 对象主要用于向监听者对象发送信息：当前的 JavaBean 对象已经发生了属性改变的事件。在 JavaBean3.java 程序中，除了普通的存取 JavaBeans 属性值的 setXXX()、getXXX()等方法以外，还定义了如下的方法：

```
public void addPropertyChangeListener(PropertyChangeListener l);
public void removePropertyChangeListener(PropertyChangeListener l);
```

第一个方法(addPropertyChangeListener()方法)其实是调用 changes 对象的 addPropertyChangeListener()方法，使一个事件监听者对象和当前 JavaBean 对象绑定起来，并把它添加到监听者队列中去。充当当前 JavaBean 对象的事件监听者，如果当前 JavaBean 对象发生了属性值改变的事件，那么 changes 对象会依次通知监听者队列中的每一个对象，当然也通知了这个事件监听者对象，让它对这个事件做出反映。

第二个方法(removePropertyChangeListener()方法)和前者的作用相反，该方法其实是调用 changes 对象的 removePropertyChangeListener()方法，从监听者队列中移除某个特定的事件监听者对象，此事件监听者对象一旦从监听者队列中删除，那么 changes 对象将不会把属性值改变的事件通知它，它再也没有办法对属性值发生改变的事件作出响应了。

getString()方法可以返回属性值，setString()方法用于设定属性值，setString()方法的代码如下所示：

```
String oldString = ourString;
ourString = newString;
changes.firePropertyChange("ourString", oldString, newString);
```

在上面的代码中，首先新定义一个字符串 oldString，用于保存属性的原值，然后把新值赋给属性值，这样会产生 JavaBeans 组件属性值改变的事件，最后调用 changes 对象的 firePropertyChange()方法，通知监听者队列里的所有事件监听者对象，当前的 JavaBean 对

象发生了属性值改变的事件，属性的名称、属性的新值、属性的原值，都被作为该方法的参数，一并传给监听者对象，由它们根据这些信息，对此事件作出响应。

Bound 类型的属性就是这样使用的。

Constrained 属性

JavaBeans 组件的 Constrained 类型的属性具有这样的性质：当这个属性的值将要发生变化但是还没有发生变化的时候，与这个属性已经建立了某种监听关系的其它 Java 对象可以否决属性值的改变。此 Constrained 类型的属性的事件监听者对象将会通过抛出 PropertyVetoException 异常事件来阻止该属性值的改变。读者请看程序清单 1.4(JavaBean4.java)。

程序清单 1. 4

```
//File Name:JavaBean4.java
//Author:fancy
//Date:2001.3.29
//Note:create a Constrained javabean

    import java.beans.*;
    public class JavaBean4
    {
    private PropertyChangeSupport changes=new PropertyChangeSupport(this);
    private VetoableChangeSupport vetos=new VetoableChangeSupport(this);
    int ourPriceInCents;

    public void setPriceInCents(int newPriceInCents)
          throws PropertyVetoException
    {
          int oldPriceInCents=ourPriceInCents;
          vetos.fireVetoableChange("priceInCents",
                new Integer(oldPriceInCents),
                new Integer(newPriceInCents));
          ourPriceInCents=newPriceInCents;
          changes.firePropertyChange("priceInCents",
                new Integer(oldPriceInCents),
                new Integer(newPriceInCents));
    }

    public void addVetoableChangeListener(VetoableChangeListener l)
    {
          vetos.addVetoableChangeListener(l);
    }

    public void removeVetoableChangeListener(VetoableChangeListener l)
```

```
    {
        vetos.removeVetoableChangeListener(l);
    }

    public void addPropertyChangeListener(PropertyChangeListener l)
    {
        changes.addPropertyChangeListener(l);
    }

    public void removePropertyChangeListener(PropertyChangeListener l)
    {
        changes.removePropertyChangeListener(l);
    }
}
```

程序清单 1.4(JavaBean4.java)比起程序清单 1.3(JavaBean3.java)来说，显得更为晦涩难解，在程序清单 1.4 中，定义了一个 JavaBean——JavaBean4，它有一个 Constrained 类型的属性是 ourPriceInCents，这是一个整型数据。为什么说它是 Constrained 类型的属性呢?请读者注意，在程序的开始部分，我们分别定义了 PropertyChangeSupport 类型的对象 changes 和 VetoableChangeSupport 类型的对象 vetos，changes 对象的作用和程序清单 1.3 中 changes 对象的作用一样，在这里我们就不讨论它的用法了。在这里我们主要讨论 vetos 对象的用法。

vetos 对象主要用于通知事件否决者对象，某个 JavaBean 对象的属性值将要发生变化，让它们投票表决是否允许这个事件的发生。在 JavaBean4.java 中，定义了这样的两个方法，分别是：

public void addVetoableChangeListener(VetoableChangeListener l);

public void removeVetoableChangeListener(VetoableChangeListener l);

前者可以往事件否决者对象队列中添加新的事件否决者对象，作为 JavaBean4 组件对象的事件否决者，一旦成为 JavaBean4 对象的事件否决者，就可以在事件发生之前，否决事件的发生。

第二个方法与第一个方法的作用相反，它可以将某个特定的事件否决者对象从事件否决者对象列表中删除。被删除的事件否决者对象就再也没有权利否决事件的发生，除非它们再次被添加到事件否决者队列中去。

在 JavaBean4.java 程序中，读者需要特别注意 setPriceInCents()方法的实现。在 setPriceInCents()方法中，首先把 ourPriceInCents 属性的原值给保存下来。然后调用 vetos 对象的 fireVetoableChange()方法，通知事件否决者对象队列中的每一个事件否决者对象，告诉它们：JavaBean4 对象即将发生属性改变的事件，发生此事件的属性是 ourPriceInCents，属性的新值为 newPriceInCents，属性的原值为 oldPriceInCents，(实际上还没有改变属性值)。事件否决者对象会根据这些信息，投票表决是否允许该事件的发生，如果有任何一个事件否决者对象否决了这个事件发生的可能性，那么 setPriceInCents()方法将会抛出 PropertyVetoException 异常，程序的运行将会中断，下面的代码将不会执行，也就是说属性值将会保持原来的值。如果事件否决者不否决事件的发生，那么程序将会继续往下执行，