Java

实用程序设计

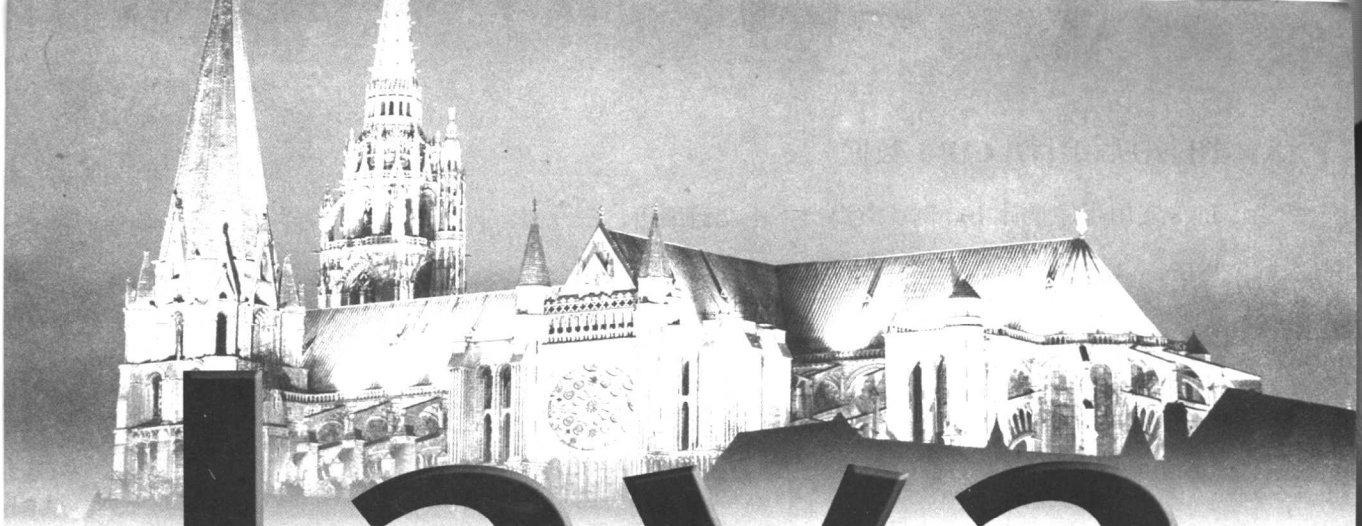
100例

袁海燕 王文涛

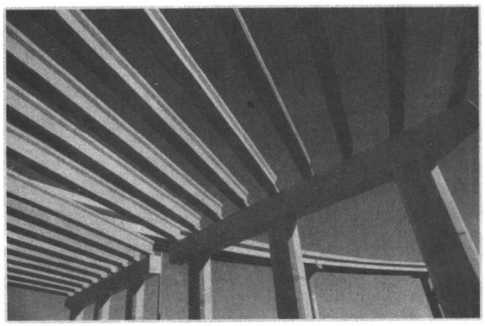
编著

附光盘

人民邮电出版社
POSTS & TELECOM PRESS



Java 实用程序设计 100例



袁海燕 王文涛

编著

人民邮电出版社

图书在版编目（CIP）数据

Java 实用程序设计 100 例 / 袁海燕，王文涛编著. —北京：人民邮电出版社，2005.5

ISBN 7-115-13418-9

Ⅰ. J... Ⅱ. ①袁...②王... Ⅲ. JAVA 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2005）第 041496 号

内 容 提 要

本书通过 100 个精选的实例深入浅出地讲解了 Java 程序设计的主要应用，涵盖图形用户界面、Applet 编程、多媒体处理、输入输出系统、网络编程、数据库编程、安全编程、手机程序设计等内容。

本书实例覆盖面广，具有较强的示范性和实用价值，适合于已经初步掌握 Java 编程概念及方法的读者阅读。

Java 实用程序设计 100 例

◆ 编　　著　袁海燕　王文涛

责任编辑　刘　浩

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

读者热线　010-67132692

北京鸿佳印刷厂印刷

新华书店总店北京发行所经销

◆ 开本：787×1092　1/16

印张：22.25

字数：543 千字　　2005 年 5 月第 1 版

印数：1－5 000 册　　2005 年 5 月北京第 1 次印刷

ISBN 7-115-13418-9/TP・4667

定价：38.00 元（附光盘）

前　言

Java 具有功能强大、开发快速、跨平台等优秀特性，获得了广大程序员的青睐。从 J2SE、J2EE 到 J2ME，Java 能满足不同层次应用的需要。现有讲解程序设计的书籍，大多是从语法、算法等角度出发，适合于系统知识的学习。本书精心挑选了实际开发中最为常用的案例，以 8 个主题的方式写作而成，针对性较强，方便实用。

而且每个例子都有思考与扩展部分，读者还可学到该例用法之外的扩展性知识。

全书共分为 8 章。

第 1 章“图形用户界面”主要讲解与 Swing 相关的应用，包括控件定制、组件布局、字体处理、快捷菜单、剪贴板的使用、拖动处理、文件选择器的使用等。

第 2 章“Applet 编程”介绍 Applet 编程技巧，主要包括各种文字与图片特效的制做、闹钟程序和万年历的开发等。

第 3 章“多媒体处理”讲解 Java 多媒体框架的使用方法，包括声音与视频的播放、图片与图形的处理等。

第 4 章“Java 输入/输出系统”讲解 Java 的 I/O 系统、主流的压缩与解压缩算法的使用、压缩中对中文文件名的处理等。

第 5 章“Java 网络编程”着重于对 Java 网络功能进行介绍，包括基于 TCP 和 UDP 的服务器与客户端、组播技术的使用以及常用协议（HTTP、FTP、Telnet）的程序设计。

第 6 章“Java 数据库编程”讲解 JDBC 的使用方法、数据库的常用操作、Servlet 编程等知识。

第 7 章“Java 安全编程”讲解 Java 安全体系各个层次的内容，从基础的访问权限控制、各种加密算法的使用到数字签名、数字证书。

第 8 章“Java 手机程序设计”涵盖了 J2ME 应用的主要方面，包括各个组件的使用方法、底层 Canvas 绘图、数据操作和网络连接等。

本书附带的光盘内容丰富，包含每个例子的源代码、编译后的 Class 文件、所需相关资源及效果图片，读者可对照学习。

由于编写时间仓促，作者水平有限，书中不足甚至错误在所难免，恳请读者批评指正（可以发邮件至 book_better@sina.com）。

编者

2005.05

目　录

第1章　图形用户界面

Java 2 平台包含一个复杂的跨平台用户界面，体系结构采用 MVC 的设计模式，即“模型—视图—控制器”（Model-View-Controller）模式，模型用来保存内容，视图用来显示内容，控制器用来控制用户输入。

Java 2 包括众多的组件。其中，Swing 组件是用纯 Java 实现的轻量级组件，没有本地代码，不依赖操作系统的支持，具有比 AWT 组件更强的实用性。Swing 在不同的平台上表现一致，并且提供本地窗口系统不支持的其他特性。它具有可插入的外观，程序在平台上运行时能够有不同的外观。

本章以丰富实用的例程深入浅出地讲解了 Java 的图形用户界面，主要包括采用 MVC 模式对控件的定制、Java 的布局管理器、Look and Feel（不同风格的用户界面）、控件之间的消息传递、剪贴板的使用方法、文字及图形的拖动处理等，对常用的控件，如文本输入框、密码验证框、下拉框和表格等也做了详尽说明。

实例 1　产生自己的控件

目的与效果

在设计应用程序用户界面时，标准组件往往不能满足用户需求，通常需要对组件进行继承、扩展或者替换。在设计 Web 应用程序或 Applet 时，也可通过定制组件来克服标准组件的局限。

本例以一个带图标和提示文本的下拉框为例，讲解如何通过继承下拉框 JComboBox，来满足应用程序的特定需要，并在此基础上实现其他一些功能，如产生颜色选择下拉框或自动排序下拉框等。本例的实现效果如图 1-1 所示。

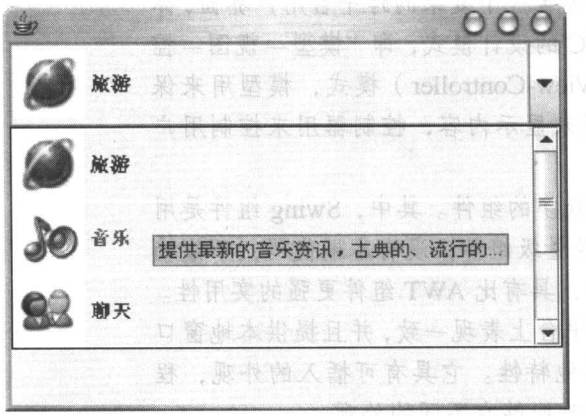


图 1-1　控件效果图

实例分析

1. ListCellRenderer 接口介绍

要定制 JComboBox 控件，需要实现 ListCellRenderer 接口。ListCellRenderer 接口提供 getListCellRendererComponent 方法，方法原型为：

```
    getListCellRendererComponent(JList list, Object value, int index, boolean
isSelected, boolean cellHasFocus)
```

方法返回 Component 对象，list 为下拉框；value 为单元的值，在本例中为一元数组，分别存放将绘制的图片与文字；index 为单元的索引值；isSelected 表示单元是否被选中；cellHasFocus 表示单元是否有焦点。

在绘制每个单元之前，系统调用该方法，得到一个 Component 对象，并将该对象绘制在正确的位置。因为返回的是 Component 对象，所以可以通过扩展任意一个

Component 对象来改变 JComboBox、JLabel 等控件的外观。

可以通过继承 JLabel 来实现带图标的下拉框，在 JLabel 中绘制图标和文本。调用 JLabel 的 setIcon、setText、setToolTipText 和 setBorder 方法分别设置图标、显示文本、提示文本和边界，并返回该 JLabel 对象，代码如下。

```
public Component getListCellRendererComponent(JList list, Object obj, int row,
boolean
        sel, boolean hasFocus){
           Object[] cell = (Object[])obj;
           setIcon((Icon)cell[0]);
           setText(cell[1].toString());
           setToolTipText(cell[2].toString());
           setBorder(new LineBorder(Color.WHITE));
           if(sel){
              setForeground(Color.MAGENTA);
           }
           else{
              setForeground(list.getForeground());
           }
           return this;
        }
```

2. 使用定制组件

使用定制组件与使用标准组件有两点不同。一是传入的参数为一维数组（addItem 为一维数组），数组内容分别是图标、显示文本和提示文本；二是调用 setRenderer 方法设置单元绘制器为已经实现的 IconRenderer，如下所示。

```
iconComboBox = new JComboBox();
iconComboBox.setMaximumRowCount(3);
iconComboBox.setRenderer(new IconRenderer());
for(int i=0;i<obj.length;i++){
   iconComboBox.addItem(obj[i]);
}
```

3. 事件处理

在本例中事件处理用匿名类来实现。在调用 addItem 方法时设置的参数为一维数组，所以在调用 getSelectedItem 时返回的也是一维数组，如下所示。

```
iconComboBox.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent evt){
        Object[] obj = (Object[])iconComboBox.getSelectedItem();
```

```
    iconLabel.setIcon((Icon)obj[0]);
    iconLabel.setText(obj[1].toString());
      }
    });
```

程序清单

1. 定制单元绘制器，实现 ListCellRenderer 接口

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;
//带图标组合框的单元绘制器,从 JLabel 类扩展,实现 ListCellRenderer 接口
public class IconRenderer extends JLabel implements ListCellRenderer{
  ……//代码见前面讲解部分，或光盘第 1 章实例 1
}
```

2. 自定义组合框的演示程序代码

```
Object[][] obj={
            {new ImageIcon("1.gif"),"旅游","提供旅游的最新信息"},
            {new ImageIcon("2.gif"),"音乐","提供最新的音乐资讯,古典的、流行的..."},
            {new ImageIcon("3.gif"),"聊天","与朋友聊天"},
            {new ImageIcon("4.gif"),"影视","影视娱乐"},
            {new ImageIcon("5.gif"),"家居","家居世界"},
          };
        iconComboBox = new JComboBox();
        iconComboBox.setMaximumRowCount(3);  //设置最大可视行数
        iconComboBox.setRenderer(new IconRenderer()); //设置单元绘制器
        for(int i=0;i<obj.length;i++){   //增加数组中的所有元素到下拉框中
          iconComboBox.addItem(obj[i]);
} //完整代码见光盘第 1 章实例 1
```

思考与扩展

- 怎样设计一个类似于 Word 中的字体选择下拉框？

```
public class FontCellRenderer implements ListCellRenderer{
        public Component getListCellRendererComponent(JList list, Object
```

```
value, int index, boolean isSelected, boolean cellHasFocus)  {
            JLabel label = new JLabel();  //用于返回的 Component，这里是一个 JLabel
            Font font = (Font)value;  //得到字体
            label.setText(font.getFamily());  //设置显示文本为字体名称
            label.setFont(font);  //设置显示字体
            return label;
    }}
```

◆ 怎样自定义一个颜色选择的组合框？

只需要写一个 Renderer 即可，示例如下：

```
public class ColorComboBoxRenderer extends JPanel implements ListCellRenderer
{
        public Component getListCellRendererComponent(JList list, Object obj, int
row, boolean sel, boolean hasFocus){
            if(obj instanceof Color){  //如果对象为 Color 对象,则设置背景色
                setBackground((Color)obj)  //设置背景
            }
            return this;
    }}
```

实例 2　控件的排布示例

目的与效果

在设计用户界面时，按钮、菜单和工具栏等各种控件如何在屏幕上显示，它们的尺寸如何，位置怎样安排，都需要程序员考虑。而 Java 做为跨平台的语言，有时一个程序需要面对不同的屏幕尺寸、不同的分辨率，为适应这些需要，Java 采用了布局管理器来安排控件的位置。

本例展示了最常用的布局管理器 BorderLayout（如图 2-1 所示）和 GridBagLayout（如图 2-2 所示），说明了这些布局管理器的使用方法和适用场合。通过组合使用这些布局管理器，可以设计出简洁、实用的用户界面。

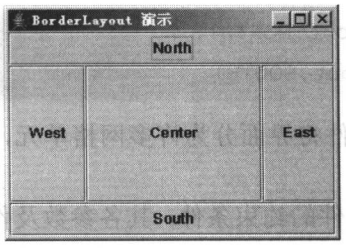


图 2-1　BorderLayout

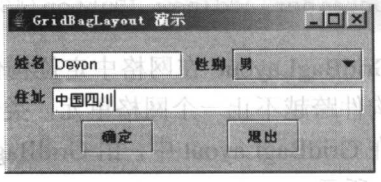


图 2-2　GridBagLayout

1. 设置布局管理器

Java 在布局管理方面采用了容器和布局管理分离的方案，即容器只负责将组件放入其中，而不管这些组件件是如何放置的。布局管理交给专门的布局管理器类（LayoutManager）来完成。使用布局管理器，当窗口缩放时，组件位置、大小也随之合理调整。

布局管理器惟一的任务就是将组件安放（即确定其尺寸和位置）进容器里（Container）。每个容器都有一个 java.awt.Container 的子类，并提供 setLayout 方法来指定要被使用的布局管理器。对 JFrame 而言，使用布局管理器时，首先要调用 JFrame 的 getContentPane 得到容器，然后用容器的 setLayout 方法设置布局管理器，如下所示。

```
Container container = getContentPane();
container.setLayout( new BorderLayout() );
```

BorderLayout 将组件按东、南、西、北、中 5 个区域放置，每个方向最多只能放置一个组件。

GridBagLayout 是功能最强大、最复杂的布局管理器，使用它可指定组件放置的具体位置及占用的单元格数目，如下所示。

```
Container contentPane = getContentPane();
contentPane.setLayout(new GridBagLayout());
```

2. 用布局管理器放置组件

描述 BorderLayout5 个区域的常量如下所示。

- BorderLayout.EAST：东
- BorderLayout.SOUTH：南
- BorderLayout.WEST：西
- BorderLayout.NORTH：北
- BorderLayout.CENTER：中

增加组件时只需要指定组件放置的区域即可。调用容器的 Add 方法时，可传入两个参数：要加入的组件以及组件在 BorderLayout 中的区域，如下所示。

```
container.add(new JButton("North"), BorderLayout.NORTH);
container.add(new JButton("South"), BorderLayout.SOUTH);
```

GridBagLayout 在网格中定位构件，根据相关约束条件将界面分为许多网格单元，并且允许构件跨越不止一个网格单元，类似于表格的结构。

在 GridBagLayout 中，由 GridBagConstraints 来确定组件的约束条件，其各参数及作用如表 2-1 所示。

表 2-1 参数说明

参 数 名	作 用	可 选 值
anchor	组件在它的显示区（组件所在网格）中的位置	CENTER、EAST、NORTH、NORTHEAST、NORTHWEST、SOUTH、SOUTHEAST、SOUTHWEST、WEST
fill	组件填充它所占据的网格单元的方式	BOTH、HORIZONTAL、VERTICAL、NONE
gridx、gridy	组件相对于左上角网格单元的位置	RELATIVE 或在网格中表示一个 *x,y* 位置的整数值
gridwidth、gridheight	分配给组件的网格单元数量	RELATIVE,REMAINDER,整数值
ipadx、ipady	内部填充增加构件的首选尺寸	整数值（表示像素个数）
insets	构件的边缘和网格边缘之间的部分	Insets 对象
Weightx,wdighty	组件所在网格单元相对于同行或同列的其他组件的值	double 值

程序清单

1. BorderLayout 演示

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class BorderLayoutDemo extends JFrame{
    public BorderLayoutDemo(){  //构造函数
        Container container = getContentPane();  //得到容器
        container.setLayout( new BorderLayout() ); //设置布局管理器为 Borderlayout
        container.add(new JButton("North"), BorderLayout.NORTH);  //增加按钮
        container.add(new JButton("South"), BorderLayout.SOUTH);
        container.add(new JButton("East"), BorderLayout.EAST);
        container.add(new JButton("West"), BorderLayout.WEST);
        container.add(new JButton("Center"), BorderLayout.CENTER);
        setTitle("BorderLayout 演示");  //设置窗口标题
        setSize(280,200);  //设置主窗口尺寸
        setVisible(true);  //设置主窗口可视
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //关闭窗口时退出程序
    }
    public static void main( String args[] ){
        new BorderLayoutDemo();
    }
}
```

2. GridBagLayout 演示

```
import java.awt.*;
```

```
import java.awt.event.*;
import javax.swing.*;
public class GridBagLayoutDemo extends JFrame {
public GridBagLayoutDemo(){  //构造函数
      Container contentPane = getContentPane();  //得到容器
      contentPane.setLayout(new GridBagLayout());  //设置布局管理器
      JLabel labelName=new JLabel("姓名");  //姓名标签
      JLabel labelSex=new JLabel("性别");  //性别标签
      JLabel labelAddress=new JLabel("住址");  //住址标签
      JTextField textFieldName = new JTextField();  //性名文本域
      JTextField textFieldAddress = new JTextField(); //地址文本域
      JComboBox comboBoxSex = new JComboBox();  //性别组合框
      JButton buttonConfirm=new JButton("确定");  //确定按钮
      JButton buttonCancel=new JButton("退出");  //退出按钮
      //增加各个组件
      contentPane.add(labelName,       newGridBagConstraints(0, 0, 1, 1, 0.0, 0.0
          ,GridBagConstraints.CENTER,    GridBagConstraints.HORIZONTAL, new
Insets(0, 5, 0, 0), 0, 0));
      ……. //省略程序代码附在光盘中第 1 章 实例 2 控件的排布示例
      contentPane.add(textFieldAddress,     new GridBagConstraints(1, 1, 3, 1,
0.0, 0.0
          ,GridBagConstraints.NORTHWEST, GridBagConstraints.HORIZONTAL, new
Insets(5, 5, 0, 5), 0, 0));
      setTitle("GridBagLayout 演示");  //设置窗口标题
      setSize(300,140);  //设置窗口尺寸
      setVisible(true);  //设置窗口可见
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //关闭窗口时退出程序
    }
    public static void main(String args[])   {
       new GridBagLayoutDemo();
    }
}
```

思考与扩展

◆ 如何使用 FlowLayout?

FlowLayout 将组件按从左到右、从上到下的顺序依次排列，如果一行放不下，可以在下一行继续放置。使用 FlowLayout 只能控制组件布局的顺序。

```
JPanel panel = new JPanel(new FlowLayout());  //初始化组件，设置布局管理器
panel.add(new Jbutton("button"));  //增加组件
panel.add(new JTextField());  //增加组件
```

◆ 如何使用 BoxLayout?

BoxLayout 就像整齐的盒子，可水平放置，也可垂直放置，每个盒子中有一个组件。

```
Box box = Box.createHorizontalBox();  //创建水平方向的 Box
box.add(component1);  //增加组件 1
box.add(component2);  //增加组件 2
```

◆ 怎样使用绝对坐标定位组件位置?

可用 component 的 setbounds 方法来设置组件的边界。

```
JPanel panel = new JPanel(null);  //初始化组件
component.setBounds(x, y, w, h);  //设置边界，依次为 x 坐标、y 坐标、宽度、高度
panel.add(component);  //增加组件到 panel 上去
```

◆ 如何固定一个组件的尺寸?

调用要固定尺寸的组件的 SetMinimumSize、setPreferredSize 和 setMaximumSize 3 个方法，把尺寸设为一致。

```
JButton button=new JButton();
Dimension dim = new Dimension(200,200);
button.setMaximumSize(dim);
button.setMinimumSize(dim);
button.setPreferredSize(dim);
```

实例 3　控件的相互控制与消息传递

目的与效果

组件之间消息的传递与相互控制是程序设计中的重要内容。本例通过一个简单的聊天程序界面，来说明在 Java 程序设计中各个组件之间如何相互传递消息并相互控制。

程序运行效果如图 3-1 所示，单击“清除”按钮，则消息输入文本框中的信息清除掉；单击“发送”按钮，则把消息发送到信息显示框中。

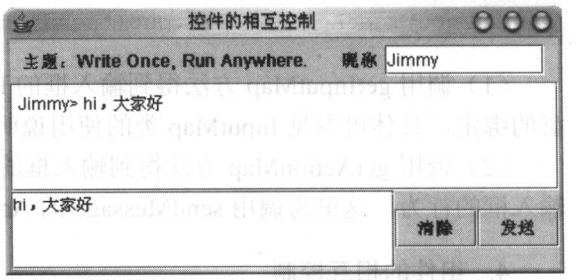


图 3-1　信息传递窗口

实例分析

1. 通过匿名类实现

可以通过匿名类实现“清除”按钮的事件监听，在 actionPerformed 方法中，当单击按钮时进行相应的处理。本例中直接调用消息输入框（jtaInput）的 setText 方法，设置输入框的文本为空，以达到清除输入框中已输入文本的目的，代码如下所示。

```
jbClear.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
            jtaInput.setText("");
    }
});
```

2. 通过 Action 实现

在“发送”按钮的事件处理中，通过设置其行为（Action）来实现。首先，需要定义一个 Action，实现其 actionPerformed 方法。本例中，发生事件时能通过调用自定义的 sendMessage 方法来发送消息。

对于“发送”按钮，可调用 setAction 方法设置有按钮事件发生时的行为，代码如下所示。

```
Action sendMessage = new AbstractAction(){
      public void actionPerformed(ActionEvent e){
            sendMessage();
      }
 };
 jbSend.setAction(sendMessage);
```

3. 处理键盘事件

设置文本输入框在用户按下“Enter”键时发出消息，代码如下所示。

```
jtaInput.getInputMap().put(KeyStroke.getKeyStroke("ENTER"),"send");
jtaInput.getActionMap().put("send",sendMessage);
```

（1）调用 getInputMap 方法得到输入框的 InputMap，再用该 InputMap 的 put 方法实现键盘的绑定，具体可参见 InputMap 类的使用说明。本例中，绑定的键为“Enter”键。

（2）调用 getActionMap 方法得到输入框的 ActionMap，再用 ActionMap 的 put 方法设置输入框的行为，这里为调用 sendMessage 的 Action。

4. 组件的相互控制

在 sendMessage 方法中，调用聊天信息显示框的 insert 方法插入最新收到的信息。本例