第 8 章　kAWT 编程

AWT（Abstract Window Toolkit，抽象窗体工具箱）是 Java 中提供的一个标准的界面设计类库。AWT 提供了按钮、滚动条、窗体、文本框等界面元素，使用 AWT 可以开发基于窗体的用户界面。kAWT 是可以在 KVM 上运行的简版的 AWT。J2ME 提供的 KVM 运行平台和其他 Java 版本的运行平台在很多地方是不同的，因此将运行在其他 Java 平台上的应用程序移植到 KVM 上需要注意很多方面的问题。其中，最重要的是 KVM 仅提供了比较简单的事件处理能力。在 KVM 这样的略微显得有些弱的事件处理能力的基础上运行 AWT 可能不太现实。因此，必须对标准的 AWT 进行剪裁，使它能适应在 KVM 上的运行，这就是 kAWT。

相对于 AWT 而言，kAWT 是一个轻量级的界面开发工具。kAWT 和 AWT 主要的不同包括：

- kAWT 中每个组件仅仅支持一个事件 listener，而 AWT 中一个组件支持多个事件 listener。
- kAWT 中的许多调用不是线程安全的，而 AWT 是线程安全的。
- kAWT 是使用纯 Java 书写的轻量级界面设计工具，适合在 KVM 上运行。

8.1　在 J2ME 中使用 kAWT

在 J2ME 中并没有包含 kAWT，要在 MIDlet 中使用 kAWT，就必须安装 kAWT 程序包。使用 kAWT 的一个很好的平台是 J2ME Wireless Toolkit。下面介绍如何在 J2ME Wireless Toolkit 中使用 kAWT。

8.1.1　在 J2ME Wireless Toolkit 中使用 kAWT

为了在 J2ME Wireless Toolkit（如果希望使用 kAWT，就需要使用 J2ME Wireless Toolkit 1.03 以上的版本）中使用 kAWT，可以按照下面的步骤进行：

（1）创建一个新的工程，假设名字为 KawtDemo。

（2）在 J2ME Wireless Toolkit 中已经支持外加的 Java 类库，因此可以将 kAWT 的类库 kawt_midp.zip 拷贝到 c:\j2mewtk\apps\KawtDemo\lib 目录下（假设 J2ME Wireless Toolkit 的安装目录为 c:\j2mewtk）。

（3）将 KawtDemo.java 拷贝到 c:\j2mewtk\apps\KawtDemo\src 目录下。

注意

kawt_midp.zip 和 KawtDemo.java 可以在提供的源代码中找到。

上述步骤完成以后，就可以在 J2ME Wireless Toolkit 中编译和运行 KawtDemo 了。由于在 KawtDemo 中使用了 kAWT 类库，因此在编译时 J2ME Wireless Toolkit 会提示报警信息：“KawtDemo.java 使用或覆盖一个不鼓励使用的 API”。由于在 KawtDemo 中需要使用附加的 kAWT 库，因此不要理会这个报警信息。

KawtDemo 的运行效果如图 8-1 所示。

图 8-1　KawtDemo 的运行效果

8.1.2　命令行方式使用 kAWT

kAWT 除了可以在 J2ME Wireless Toolkit 环境中运行以外，当然也可以在命令行方式下编译、校验和运行。下面介绍在命令行环境下运行 kAWT 的例子程序 KawtHello 的步骤。KawtHello.java 是一个使用 kAWT 编写的在屏幕上显示“Hello,World!”的程序，代码如下：

```
import java.awt.*;
import java.awt.event.*;
import de.kawt.*;

public class KawtHello extends KAWTlet implements ActionListener {
    Frame frame = new Frame ("kAWT Hello v1.0");     // 生成框架
    Label lbl1 = new Label("Hello,World!");     // 生成标签
    Label lbl2 = new Label("2002.1.10 UESTC");     // 生成标签

    public KawtHello(){
        frame.add("Center",lbl1);
        frame.add("South",lbl2);
        frame.addWindowListener(new Closer(this));
        frame.pack();
    }

    public void startApp(){
        frame.show ();
    }

    public void destroyApp(boolean uncond){
        frame.dispose ();
    }

    public void actionPerformed(ActionEvent e){
    }

    public static void main (String [] argv){
        new KawtHello().startApp();
    }
}
```

因为在 KawtHello.java 中使用了 kAWT 的类库，所以在编译 KawtHello.java 时必须使 Java 的编译器能访问到 kAWT 的类库。kAWT 的类库打包在文件 kawt_midp.zip（kawt_midp.zip 可以在提供的源代码中找到）中，因此必须在 J2ME 的环境变量 classpath 中加上 kawt_midp.zip 的路径。假设 kawt_midp.zip 存放在 c:\kawt 目录下，那么在 classpath 中就必须加上 c:\kawt\kawt_midp.zip。一个典型的 classpath 如下：

```
CLASSPATH=c:\j2me\midp-fcs\classes;c:\kawt\kawt_midp.zip;.
```

CLASSPATH 设置完毕以后，可以在 DOS 环境中输入下列命令：

```
javac KawtHello.java
midp KawtHello
```

上述命令执行后同样可以看到如图 8-2 所示的 KawtHello 运行效果。

图 8-2　KawtHello 的运行效果

8.1.3　KawtHello 的分析

在 KawtHello.java 中，首先引入 kAWT 的类库，代码如下：

```
import de.kawt.*;
```

每一个基于 kAWT 的程序都是从 KAWTlet 类派生出来的，并且为了能响应用户的输入，就必须实现 ActionListener 接口。KawtHello 的声明代码如下：

```
public class KawtHello extends KAWTlet implements ActionListener {
```

接下来，在 KawtHello 中生成了一个 Frame 类。Frame 类是在 kAWT 中屏幕显示的框架，每一个 KAWTlet（像 MIDlet 一样，KAWTlet 表示使用 kAWT 编写的程序）中都有一个 Frame 类，用来形成屏幕的框架。生成 Frame 类的代码如下：

```
Frame frame = new Frame ("kAWT Hello v1.0");
```

随后，KawtHello 生成两个 Label 用来显示“Hello,World!”和“2002.1.10 UESTC”信息，并且在 KawtHello 的构造函数中将两个 Label 加入到 Frame 中。这部分代码如下：

```
Label lbl1 = new Label("Hello,World!");
Label lbl2 = new Label("2002.1.10 UESTC");

public KawtHello(){
    frame.add("Center",lbl1);
    frame.add("South",lbl2);
    frame.addWindowListener(new Closer(this));
    frame.pack();
}
```

在 KAWTlet 中，同样有 startApp()和 destroyApp()方法，它们的作用与在 MIDlet 中的作用是相同的。在 startApp()中通过 Frame 类的 show()方法将 Frame 显示出来。startApp()和 destroyApp()方法的代码如下：

```
public void startApp(){
    frame.show ();
}

public void destroyApp(boolean uncond){
    frame.dispose ();
}
```

actionPerformed()方法类似于 MIDlet 中的 commandAction()方法，actionPerformed()方法的作用是处理在 KAWTlet 中的用户输入。在 KawtHello 中 actionPerformed()方法没有任何需要处理的事件，因为在 Frame 的构造函数中已经加上了下面的代码来处理用户的退出请求：

```
frame.addWindowListener(new Closer(this));
```

MIDlet 的起点是 startApp()函数，而在 KAWTlet 中，程序的起点是 main()函数，KawtHello 中 main()函数也负责启动 KAWTlet，代码如下：

```
public static void main (String [] argv){
    new KawtHello().startApp();
}
```

8.1.4　定制编译 kAWT 类库

如果希望编译自己的 kAWT 类库，则可以下载 kAWT 的源文件 kawt_src_zip（kawt_src_zip 在提供的源代码中可以找到），使用 Winzip 将 kawt_src_zip 解压到一个目录中，假设是 c:\kawt 目录。为了编译 kAWT，系统中必须安装 JDK 1.3 以上版本和 J2ME Wireless Toolkit 1.03 以上版本。在编译之前，必须设置一些环境变量来指引编译器找到 MIDP 的 API 类库和 previerify 程序，典型的环境变量设置如下所示（假设 JDK 的安装目录是 c:\jdk1.3，J2ME Wireless Toolkit 的安装目录是 c:\j2mewtk）：

```
MIDP_BCP=c:\j2mewtk\lib\midpapi.zip
KAWT=c:\kawt
KAWT_DRIVE=c:
PATH=%PATH%;c:\j2mewtk\bin;c:\jdk1.3\bin;
```

上述环境变量设置好后，可以将当前目录切换到 c:\kawt\bin 目录下，然后执行 make_kawt_midp.bat 即可完成 kAWT 的编译工作。编译完成后，在 c:\ kawt\build\midp 目录下能找到生成的 kawt_midp.zip 文件。kawt_midp.zip 文件适用于 MIDP 的 kAWT 类库。

8.2　kAWT API 简介

kAWT 提供了一系列的类库，这些类库形成了如表 8-1 所示的几个包。

表 8-1　kAWT 提供的包

包名	描述
java.awt	包含了 kAWT 中创建用户界面和绘制图形的所有类
java.awt.event	提供了所有事件处理类
java.awt.image	提供显示和处理图像的类
de.kawt	提供了在 J2ME 中特殊的 kAWT 类
de.kawt.shell	提供了在 J2ME 中特殊的文件处理类

java.awt 包含了 kAWT 中创建用户界面和绘制图形的所有类，典型的类如表 8-2 所示。

表 8-2　java.awt 包中包含的类

包名	描述
AWTEvent	所有 kAWT 事件的基础类
Canvas	代表用户绘制图形的屏幕
Checkbox	显示 checkbox 或者 radio button 部件
Dialog	显示在屏幕上的对话框
Scrollbar	显示在窗口或者对话框中的滚动框
TextArea	文本输入区，支持输入文本，文本可以自动换行

在 de.kawt 包中提供了 KAWTlet 类。KAWTlet 类是所有使用 kAWT 编写的程序的基础类。kAWT 的类库包含很多的内容，在本节中进行了简单的介绍。如果希望了解更详细的内容，则可参考 kAWT 的文档（在提供的源代码中可以找到）。

8.3 kAWT 编程例子

在本节中通过一个最典型的 kAWT 例子来介绍使用 kAWT 编程。在 8.1 中介绍了 KAWTlet 的整体框架，在 8.2 节中介绍了 kAWT 的 API，本节是这两部分知识的应用。

KawtDemo——展示 kAWT 界面设计的威力。

在 KawtDemo 中演示了 Panels、Buttons、Checkboxes 和 Textfields 等屏幕部件的使用方法，这个例子是学习 kAWT 界面编程的好例子。KawtDemo 的运行效果如图 8-3 所示。

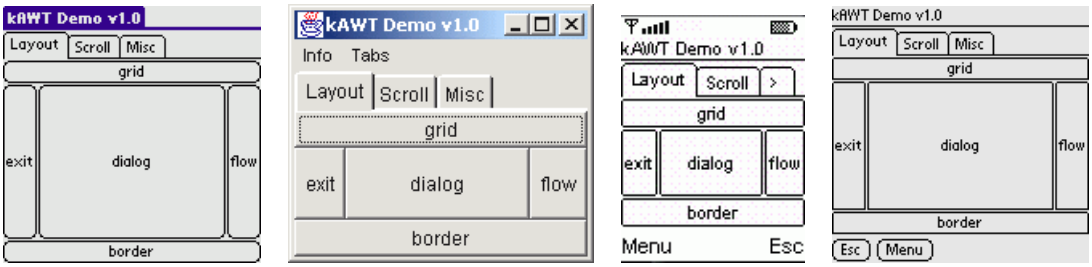

图 8-3 KawtDemo 的运行效果

KawtDemo 的代码如下：

```
import java.awt.*;
import java.awt.event.*;
import de.kawt.*;

public class KawtDemo extends KAWTlet implements ActionListener {
    class LayoutPanel extends Panel implements ActionListener {
        LayoutManager saveBorder;
        KAWTlet kawtlet;
        LayoutPanel () {
            super (new BorderLayout ());
            // 添加所有的按钮
            addButton ("North", "grid");
            addButton ("West", "exit");
            addButton ("Center", "dialog");
            addButton ("East", "flow");
            addButton ("South", "border");
            saveBorder = getLayout ();
            validate ();
        }
        // 添加一个按钮的代码
        void addButton (String where, String label) {
            Button b = new Button (label);
            add (where, b);
            b.addActionListener (this);
            b.setActionCommand (label);
        }
```

```
        // 响应按钮事件
        public void actionPerformed (ActionEvent e) {
            String cmd = e.getActionCommand ();
            if (cmd.equals ("grid"))
                setLayout (new GridLayout (3, 2));
            else if (cmd.equals ("flow"))
                setLayout (new FlowLayout ());
            else if (cmd.equals ("border"))
                setLayout (saveBorder);
            else if (cmd.equals ("dialog")) {
                if (OptionDialog.showConfirmDialog
                    (frame, "Do you like kAWT?", "Sample Dialog")
                    == OptionDialog.NO_OPTION)
                    OptionDialog.showInputDialog
                        (frame, "Why don't you like it?");
            }
            else if (cmd.equals ("exit")) {
                notifyDestroyed ();
        }
            validate ();
        }
    }

    class MiscPanel extends Panel implements ActionListener, ItemListener {
        Checkbox checkBox1;
        Checkbox checkBox2;
        Checkbox titlebox;
        TextField tf;

        MiscPanel () {
            super (new GridLayout (0, 1));

            CheckboxGroup group = new CheckboxGroup ();
            // 生成 CheckBox
            checkBox1 = new Checkbox ("Public text", group, false);
            checkBox1.addItemListener (this);
            checkBox2 = new Checkbox ("Hidden text", group, false);
            checkBox2.addItemListener (this);

            add (checkBox1);
            add (checkBox2);

            titlebox = new Checkbox ("Show Frame Title", true);
            titlebox.addItemListener (this);
            add (titlebox);

            tf = new TextField ("Test");
            add (tf);
            // 生成 Choice
            Choice choice = new Choice ();
            choice.add ("red");
            choice.add ("green");
            choice.add ("blue");
```

```
            add (choice);
            addButton ("beep");
            addButton ("exit");
        }

        void addButton (String label) {
            Button b = new Button (label);
            add (b);
            b.addActionListener (this);
            b.setActionCommand (label);
        }

        public void itemStateChanged (ItemEvent ev) {

            if (ev.getSource () == titlebox) {

                if (ev.getStateChange () == ev.SELECTED)
                    frame.setTitle ("kAWT Demo v0.9903");
                else
                    frame.setTitle (null);
            }
            else if (ev.getStateChange () == ev.SELECTED) {
                tf.setEchoChar (ev.getSource () == checkBox2 ? '*' : '\0');
                tf.setText (tf.getText ());
            }
        }

        public void actionPerformed (ActionEvent ev) {
            if (ev.getActionCommand ().equals ("beep")) {
                Toolkit.getDefaultToolkit ().beep ();
            }
            else {
        destroyApp (true);
        notifyDestroyed ();
        }
    }
    }
    // 演示面板
    class ScrollbarPanel extends Panel implements ActionListener {

        ScrollbarPanel () {
            super (new BorderLayout ());

            Panel center = new Panel (new GridLayout (0, 2));

            java.awt.List list = new java.awt.List ();

            for (int i= 0; i < 16; i++)
                list.add ("item "+i);

            center.add (list);

            TextArea ta = new TextArea ();
```

```
            ta.setText ("This is an example of a simple TextArea with automatic word wrapping in this text
            block and some hard line breaks below.\n01\n02\n03\n04\n05\n06\n07\n08\n09\n10\n11\n12\
            n13\n14\n15\n16\n17\n18\n19\n20\n");
            center.add (ta);

            Button exitB = new Button ("exit");
            exitB.addActionListener (this);
            exitB.setActionCommand ("exit");
            add ("Center", center);
            add ("South", exitB);
            validate ();
        }

        public void actionPerformed (ActionEvent e) {
            String cmd = e.getActionCommand ();
            if (cmd.equals ("exit")) {
destroyApp (true);
                notifyDestroyed ();
}
        }
}


Frame frame = new Frame ("kAWT Demo v1.0");;
Dialog dialog;
TabbedPane tabPane;

// 主程序
public KawtDemo () {
    // 菜单
    MenuBar menuBar = new MenuBar ();
    Menu menu = new Menu ("Info");
    MenuItem mi = new MenuItem ("About kAWT");
    mi.addActionListener (this);
    menu.add (mi);

    mi = new MenuItem ("Memory");
    mi.addActionListener (this);
    menu.add (mi);

menuBar.add (menu);

    menu = new Menu ("Tabs");

for (int i = 1; i <= 3; i++) {
    mi = new MenuItem ("Tab "+i);
    mi.addActionListener (this);
    menu.add (mi);
}
menuBar.add (menu);
    frame.setMenuBar (menuBar);
```

```
        tabPane = new TabbedPane();
        tabPane.addTab ("Layout", new LayoutPanel ());
        //tabPane.addTab ("Color", new MemPanel ());
        tabPane.addTab ("Scroll", new ScrollbarPanel ());
        tabPane.addTab ("Misc", new MiscPanel ());
        frame.add ("Center", tabPane);

        //getRootPane ().add ("Center", new LayoutPanel ());

    frame.addWindowListener (new Closer (this));

        frame.pack ();
    }

    public void startApp () {
    frame.show ();
    }


    public void destroyApp (boolean uncond) {
    frame.dispose ();
    }



    public void actionPerformed (ActionEvent e) {
    String cmd = e.getActionCommand ();
    if (cmd.equals ("About kAWT")) {
        OptionDialog.showMessageDialog
        (frame, "For information about kAWT, please visit\nhttp:// www.kawt.de");
    }
    else if (cmd.equals ("Memory")) {
        int count = 0;
        long free0 = Runtime.getRuntime().freeMemory ();
        long free = free0;
        while (true) {
        Runtime.getRuntime().gc();
        long newFree = Runtime.getRuntime().freeMemory ();
        if (newFree <= free) break;
        count ++;
        free = newFree;
        }
        OptionDialog.showMessageDialog
        (frame, "ini: "+free0+"\ngc"+count+": "+free);
    }
    else {
        int nr = Integer.parseInt (cmd.substring (4));
        tabPane.setSelectedIndex (nr-1);
    }
```

```
        }

        public static void main (String [] argv) {
            new KawtDemo ().startApp ();
        }

    }
```

本章小结

本章介绍了使用 kAWT 在手机上进行编程的基本方法。kAWT 是为 J2ME 设计的可以在手机这样的移动信息设备上运行的 AWT 的微型版本。kAWT 提供了相当强大的功能，本章仅介绍了 kAWT 的一些简单的知识。关于 kAWT 的详细知识，特别是类库中的各种各样的类，可以参考 kAWT 的参考手册（在出版社网站上下载）。

在 J2ME 中并没有包含 kAWT，要在 MIDlet 中使用 kAWT 必须进行必要的安装。本章介绍如何在 J2ME Wireless Toolkit 中使用 kAWT 和在命令行方式下使用 kAWT 的方法。

在本章中详细分析了使用 kAWT 编写的一个例子程序——KawtHello，通过 kAWT 的分析，读者可以了解到如何使用 kAWT 编写应用程序。

本章的最后给出了 kAWT 编程的两个例子。KawtDemo 展示了 kAWT 界面设计的威力，GifDemo 展示了 kAWT 的图形编程功能。

习题八

一、选择题

1．java.awt 包含了 kAWT 中创建用户界面和绘制图形的所有类，请选择以下类的正确描述（　　）。

（1）AWTEvent　　（2）Canvas

（3）Checkbox　　（4）Dialog

（5）Scrollbar　　（6）TextArea

A．代表用户绘制图形的屏幕

B．所有 kAWT 事件的基础类

C．显示 checkbox 或 radio button 部件

D．文本输入区，支持输入文本，文本可以自动换行

E．显示在屏幕上的对话框

F．显示在窗口或对话框中的滚动框

二、思考题

1．请简要说明 kAWT 与 AWT 的区别。

2．在 J2ME Wireless Toolkit 上使用 kAWT，有哪些步骤？