

商业开发代码库系列

Java 案例开发集锦

袁然 郑自国 邹丰义 编著



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

Java 案例开发集锦

Java是面向对象的跨平台开发语言，也是一种平台，1995年由Sun公司发布，现已成为Internet中最受欢迎、最有影响的编程语言之一。Java连同Internet，WWW正在改变应用软件的开发和使用方式，一切都要围绕着网络，围绕着平台无关。

本书直接从精选的案例入手，不再拘泥于传统编程语言的概念，而是将软件开发思想和经验寓于案例之中，在使读者了解编程语言的同时，可轻松地进入编程实战状态，快速地成为编程高手。

本书的特点包括：

- 直接通过案例的实现显示Java的强大功能
- 提供完整的源代码，读者只需稍做修改便可应用于自己的开发任务
- 真实再现Java编程的全部过程

最新推出“商业开发代码库系列”丛书：

Visual FoxPro案例开发集锦

Visual Basic案例开发集锦

Visual C++案例开发集锦

Java案例开发集锦

ASP案例开发集锦

JSP案例开发集锦

Delphi案例开发集锦

C++ Builder案例开发集锦

PowerBuilder案例开发集锦

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

ISBN 7-121-01599-4



9 787121 015991 >



责任编辑：徐云鹏

责任美编：李倩

ISBN 7-121-01599-4

定价：48.00元(含光碟1张)

商业开发代码库系列

Java案例开发集锦

袁然 郑自国 邹丰义 编著

电子工业出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书精选了几十个来自实战中的Java案例，每个案例配有详细的实现步骤和完整的源代码解释。通过精确分析Java在页面特效、文件处理、游戏、动画处理、数据库、网络、安全、Web服务器部署等方面案例开发过程，在描述Java应用技术知识的同时，展现Java的强大编程功能。

本书结构合理、内容丰富，可以作为Java编程爱好者提高编程水平的参考书，也可作为大中专院校计算机专业学生学习的辅助教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

Java案例开发集锦/袁然，郑自国，邹丰义编著.一北京：电子工业出版社，2005.9
(商业开发代码库系列)

ISBN 7-121-01599-4

I. J… II. ①袁… ②郑… ③邹… III. Java语言—程序设计 IV. TP312

中国版本图书馆CIP数据核字（2005）第085077号

责任编辑：徐云鹏

特约编辑：卢国俊

印 刷：北京天竺颖华印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路173信箱 邮编：100036

北京市海淀区翠微东里甲2号 邮编：100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：31.75 字数：810千字

印 次：2005年9月第1次印刷

定 价：48.00元（含光碟1张）

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：010-68279077。质量投诉请发邮件至zts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

前　　言

距离Sun公司第一次发布Java已经整整10年了，10年相对于计算机飞速发展的进程来说不算短，它足以淘汰掉许多技术，也足以考验真正的强者。时至今日，Java已成为Internet中最受欢迎、最具影响的编程语言之一，并且聚集了大批的拥趸，笔者也是其中之一。在与Java多年朝夕相处的日子里，我们积累了丰富的经验和教训，现呈献出来，希望能对大家有所帮助。

信息的价值在于使用和共享，Internet和Web是信息使用和共享的最快捷、最便宜的方式，Java连同Internet、WWW正在改变应用软件的开发和使用方式，一切都要围绕着网络，与平台无关。Java是面向对象的程序设计语言，通过解释方式来执行，具有跨平台性、简单、面向对象、分布式、可靠、安全、结构中立性、可移植性、高性能、多线程、动态性等许多优点。Java不仅是一种开发语言，还是一个平台，Java平台包括两部分内容：Java虚拟机和Java API。

本书从实际编程出发，共收集了几十个案例，每个案例详细讲解了实现的步骤和制作的要点，并提供了源代码。本书所有案例均调试运行通过，运行环境和配置在书中或者配书光碟中都有说明。我们通过案例学习Java的应用，以便能尽快进入实战状态。

全书共分10章，第1章 Java与Applet，收集了10个应用Applet技术的案例，从案例实现步骤的讲解中反映Applet技术的重点。

第2章 Java与特效，通过12个页面特效和文字特效的例子，说明Java在处理特效方面的独到之处，利用Java能轻松制作又酷又炫的网页特效。

第3章 XML与其他，XML是数据描述的标准，Java和XML的组合具有广泛而深远的意义，本章收录了3个关于XML处理的程序和其他6个小应用。

第4章 Java与游戏，Java与游戏的结合已成为新一代网络游戏的主角。本章有6个好玩的游戏供读者欣赏，同时讲解了在编写游戏过程中用到的方法和技巧。

第5章 Java与文件操作，提供了10个文件处理的案例。计算机上的各种资源，无论是外围设备，还是硬盘的数据资料，都是以“文件”方式进行访问处理的，文件操作是编程中的重要内容之一，Java在这方面提供了丰富的类和方法。

第6章 Java与安全，Java本身具有很强的安全性，本章共介绍了7个加密和数字证书方面的案例。

第7章 Java与数据库，通过6个案例讲解数据库操作在Java中的实现方法。

第8章 Java与Servlet，Servlet运行在服务器端，本章共有8个案例，说明如何部署Servlet。

第9章 Java与网络，本章提供8个案例来说明网络设计语言——Java在网络应用上的得心应手。

第10章介绍了4个综合案例。

本书案例中涉及到Java的类、对象、接口、继承、线程、异常处理、I/O流、Applet、Servlet、NET、XML、Security、DB、File、JavaBean、Socket、JVM等内容，同时也体现了MVC的编程思想。

本书的分类并不严格，比如第1章，Java与Applet是按照应用技术进行组织的，后面章节中的特效应用和有些游戏是按应用类型分类的，但也用到了Applet技术。笔者想全力展现Java的博大精深与魅力，但因为水平有限，有时辞不达意，敬请读者谅解。

本书的编写过程得到了电子工业出版社和北京美迪亚电子信息有限公司和山东大众信息产业有限公司的大力协助及许多网友的支持，在此表示衷心的感谢。参加本书编校的还有王明杰、平凡、韩冬佑、王艳、郭莹、刘有志、杨萍、陈圣琳、向小平等。由于时间仓促，且作者水平有限，书中的错误在所难免，真诚希望广大读者提出宝贵意见。

谨向阅读本书的读者朋友们表示诚挚的敬意和谢意。

目 录

第1章 Java与Applet	1
案例1 图形按钮	1
案例2 模拟工具条	8
案例3 Applet与Applet在页内的通信	13
案例4 电子相册	16
案例5 百叶窗效果	19
案例6 波浪彩虹文字	27
案例7 3D立体渐层文字	35
案例8 飞行文字	38
案例9 聚光灯效果	41
案例10 伸展文	45
本章小结	49
第2章 Java与特效	50
案例1 火焰招牌	50
案例2 闪电中的城市	54
案例3 激光绘图	60
案例4 水面倒影	66
案例5 图片放大镜	71
案例6 浮动的气泡	76
案例7 烟花汇演	81
案例8 星空模拟	88
案例9 阴影跑马灯	94
案例10 下雪的图片	98
案例11 动态分割线	102
案例12 飞流直下	106
本章小结	109
第3章 XML与其他	110
案例1 将HTML文件转成XML文件	110
案例2 把XML文件转换成可浏览的HTML格式文件	113
案例3 用JDOM解析XML文件	116

案例4 Java编制的时钟.....	120
案例5 简单日历	124
案例6 系统内存状态监视程序.....	130
案例7 简单的计算器	134
案例8 多线程断点续传	144
案例9 笛卡儿曲线	153
本章小结	155
第4章 Java与游戏	156
案例1 幸运52游戏	156
案例2 “速算24”扑克游戏	163
案例3 拼图游戏	172
案例4 贪吃蛇游戏	189
案例5 打球游戏	196
案例6 棒打猪头	207
本章小结	219
第5章 Java与文件操作	220
案例1 目录列表的显示	220
案例2 检查与创建目录	223
案例3 文件复制	227
案例4 文件查看器	231
案例5 字符串的查找和替换	235
案例6 利用RandomAccessFile类来实现文件的追加	242
案例7 用Zip进行多文件保存	244
案例8 用JUNIT建立测试类	247
案例9 用Java保存位图文件	252
案例10 获取文件属性	261
本章小结	265
第6章 Java与安全	266
案例1 用户登录验证的完整程序	266
案例2 MD5的JavaBean实现	272
案例3 用公钥计算消息摘要的验证码	278
案例4-1 Java中数字证书的生成及维护方法	283
案例4-2 数字证书的签发（签名）	288
案例4-3 利用数字证书给Applet签名	296
案例5 利用DES加密解密	303
本章小结	311

第7章 Java与数据库	312
案例1 在Applet中应用JDBC访问数据库	312
案例2 通过JDBC-ODBC桥连接数据库	323
案例3 通过Tomcat数据源访问数据库	327
案例4 JDBC连接池的实现	329
案例5 用JavaBean实现MySQL的分页显示	340
案例6 利用JDBC – ODBC查看查询结果	348
本章小结	353
第8章 Java与Servlet	354
案例1 利用Servlet打开非HTML格式的文档	354
案例2 Servlet和JSP的通信	359
案例3 Servlet和Servlet的通信	367
案例4 Servlet动态生成图像	370
案例5 用Servlet连接数据库	373
案例6 用Servlet实现页面注册和登录	377
案例7 运用Servlet实现BBS功能	388
案例8 倾听Web服务器信息	394
本章小结	399
第9章 Java与网络	400
案例1 显示你的IP	400
案例2 用Socket进行客户与服务器通信	404
案例3 利用UDP Socket技术实现IP多点传送	409
案例4 利用Java API发送E-mail	416
案例5 从Mail Server删除一条消息	427
案例6 在Java程序中实现FTP的功能	431
案例7 一个最简单的聊天程序	437
案例8 代理服务器的实现	446
本章小结	452
第10章 Java综合实例	453
案例1 用Java实现zip压缩解压缩	453
案例2 简易图书管理系统	460
案例3 UFO攻击游戏	467
案例4 制作一个MP3播放器	488
本章小结	500

第1章

Java与Applet



本章内容

- 案例1 图形按钮
- 案例2 模拟工具条
- 案例3 Applet与Applet在页内的通信
- 案例4 电子相册
- 案例5 百叶窗效果
- 案例6 波浪彩虹文字
- 案例7 3D立体渐层文字
- 案例8 飞行文字
- 案例9 聚光灯效果
- 案例10 伸展文字
- 本章小结



案例1 图 形 按 钮

案例运行效果与操作

本案例是一个用Java Applet实现页面响应鼠标事件的例子。实现了鼠标移入、移出、点击事件的响应功能。程序运行后，界面如图1-1所示。

把鼠标移到图片上面，效果如图1-2所示，并伴随播放声音。如果把鼠标移出，将返回图1-1的状态。

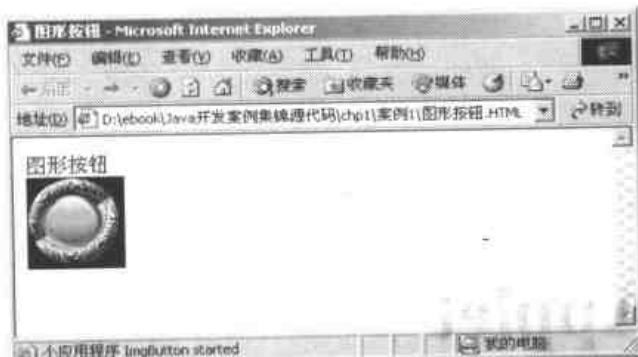


图1-1 运行界面



图1-2 鼠标移到图片上面的效果

单击图片，程序会再次更改图片的显示效果，如图1-3所示。

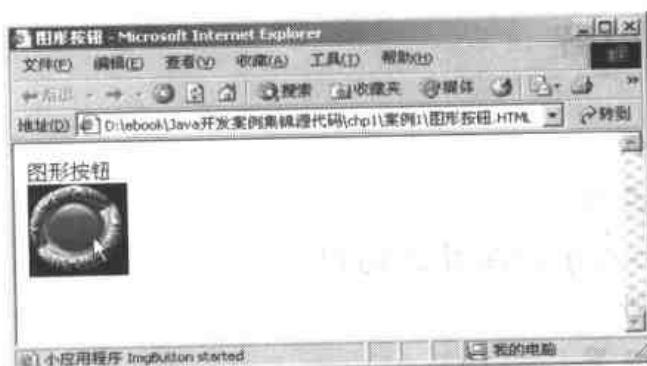


图1-3 单击图片产生的效果

制作要点

1. 获取HTML中的param的参数值
2. Applet的声音播放AudioClip类的使用
3. 鼠标事件的监听
4. MediaTracker类的使用

步骤详解

本例采用了Java小应用程序Java Applet，Java Applet首先需要嵌入到网页中，执行时需下载到用户本地进行执行，用户端需要安装Java虚拟机。

1. 获取HTML中的param的参数值。

我们先来看一下HTML的源代码：

```
<applet code=ImgButton.class width=75 height=75>
<param name=soundA value="midiA.mid">
<param name=soundB value="midiB.mid">
```

```

<param name=Images1      value="button1.gif">
<param name=Images2      value="button2.gif">
<param name=Images3      value="button3.gif">
<param name=URL value=http://www.dzwww.com>
</applet>

```

我们需要在Java中取得这些参数，就要用java.applet.Applet类的方法：

```
public String getParameter(String name)
```

例如Applet标记在HTML网页中是这样的：

```

<applet code="Clock" width=50 height=50>
<param name=Color value="blue">
</applet>

```

我们在程序中调用方法getParameter("Color")，这个方法将返回值“blue”。

```

String colorStr = "";
colorStr = getParameter("Color");
if(colorStr==null){
    colorStr="blue"      ;
}

```

2. Applet的声音播放 AudioClip类的使用。

图像格式各种各样，如BMP、GIF和JPEG等。声音文件也一样，WAV和AU是最常用的两种声音文件。在JDK1.2之前Java仅支持AU文件，从JDK1.2开始，Java提供了对WAV、MIDI等声音文件的支持。

使用Applet播放声音时，需首先定义AudioClip对象，getAudioClip方法能把声音赋予AudioClip对象，如果只想播放一遍声音，应调用AudioClip类的play方法；如果想循环播放，应选用AudioClip类的loop方法。

```

sound1 = getAudioClip(getDocumentBase(), param);
sound1.play(); //播放一次
sound1.loop(); //循环播放

```

3. 鼠标点击事件的监听。

类java.awt.event.MouseListener共有以下5个方法来实现鼠标点击事件：

方法摘要

void	mouseClicked(MouseEvent e)	当用户按下并松开鼠标按钮时发生。 用户在选择或双击图标的时候通常会点击鼠标按钮，如果在松开鼠标之前移动鼠标，点击不会导致鼠标相应事件出现。 因为点击鼠标是按下鼠标和松开鼠标的结合，在事件分配给mouseClicked()方法之前，mousePressed()和mouseReleased()方法已同时被调用。
------	-----------------------------------	--

void	mouseEntered(MouseEvent e)当鼠标离开当前组件并进入所监听的组件时激活事件。
void	mouseExited(MouseEvent e)当鼠标离开所监听的组件时发生。
void	mousePressed(MouseEvent e)当用户按下鼠标按钮时发生。
void	mouseReleased(MouseEvent e)当用户松开鼠标按钮时发生。

①处理鼠标移入事件：

```
offG.drawImage(img1, 0, 0, width, height, this);
```

②处理鼠标按钮松开事件：

```
offG.drawImage(img2, 0, 0, width, height, this);
repaint();
soundB.play();
getAppletContext().showDocument(url);
```

③单击鼠标左键事件：

```
offG.drawImage(img3, 0, 0, width, height, this);
repaint();
soundA.play();
System.out.println("soundB play");
```

4. MediaTracker类的使用。

Java专门提供了用于跟踪包括图像和声音等多媒体对象的ImageObserver类和MediaTracker类。

在本书程序中主要用到的是跟踪多幅图像状态的MediaTracker类。

```
imageTracker = new MediaTracker(this);
try {
    //开始装载所有图片，多媒体对象imageTracker会跟踪所有指定图片的状态。
    //这个方法会等待，直到所有指定的图片都转载完成，可能会抛出异常
    InterruptedException
    imageTracker.waitForID(0);
} catch (InterruptedException e) {
}
```

MediaTracker的主要方法及用途

返回类型	方法	用途
	MediaTracker (Component comp)	构造方法，为指定的组件comp创建一个MediaTracker对象

返回类型	方法	用途
void	<code>addImage (Image image, int id)</code>	将图片加入到MediaTracker的监视队列中去, image为要被监视的图像对象, id为监视图像在监视队列中的标识号
void	<code>AddImage(Image image,int id,int w,int h)</code>	将图片加入到MediaTracker的监视队列中去, w为所监视的图像对象的宽度, h为所监视的图像对象的高度
boolean	<code>CheckAll()</code>	检查所有被监视的图像对象是否已经完成了装载过程。如果所有图像对象已经完成了装载, 则返回true, 否则为false
boolean	<code>CheckAll(Boolean load)</code>	检查所有被监视的图像对象是否已经完成了装载过程。如果load为true, 则开始装载那些还没有被装载的图像
boolean	<code>CheckId(int id)</code>	检查在监视队列中所有标识号为id的图像对象是否已经完成了装载过程, 如果已经完成了装载过程, 返回true
boolean	<code>CheckId (int id,Boolean load)</code>	检查在监视队列中所有标识号为id的图像对象是否已经完成了装载过程, 如果load为true, 且在监视队列中的标识号为id的图像没有被装载的话, 则开始装载此图像对象
boolean	<code>IsErrorHandler(int id)</code>	检查在监视队列中标识号为id的图像对象是否装载有错误, 如有错误, 则返回true
void	<code>RemoveImage (Image image)</code>	从监视队列中移去指定的图像对象image
void	<code>RemoveImage (Image image,int id)</code>	从监视队列中移去指定id的图像对象image
void	<code>RemoveImage(Image image,int id,int width, int height)</code>	从监视队列中移去指定id的图像对象image, 且这个图像对象被指定了大小
void	<code>WaitForAll()</code>	开始装载监视队列中所有没有被装载的图像对象, 如果装载不成功, 则会抛出一个异常
boolean	<code>WaitForAll(long ms)</code>	开始装载监视队列中所有没有被装载的图像对象。但是, 如果装载时间超时, 则会取消装载, 并返回false

返回类型	方法	用途
void	WaitForId(int id)	开始装载监视队列中标识号为id的图像对象，如果装载不成功，则会抛出一个异常
boolean	WaitForId(int id, long ms)	开始装载监视队列中标识号为id的图像对象。但是如果装载时间超时，则会取消装载，并返回false

程序源代码与解释

```

/*
*本案例是一个用java applet实现页面响应鼠标事件的例子。
*实现了鼠标移入、移出、点击事件的响应的功能。
*/
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

//页面响应鼠标事件
public class ImgButton extends Applet implements MouseListener {
    private int width, height;                                //定义图像按钮的宽度和高度
    private Image offI, img1, img2, img3;                      //定义Image图像
    private Graphics offG;                                     //定义Graphics图像
    private MediaTracker imageTracker;                         //定义图像跟踪器
    private boolean isMouseEnter = false, isPress = false;      //定义鼠标状态变量
    private Color ButtonColor = Color.yellow, lightC, darkC;   //定义按钮颜色
    private URL url;                                         //定义超链接变量
    private AudioClip soundA, soundB;                         //定义声音文件变量，以便读取声音文件
    private String param;                                     //定义参数变量，以便从HTML文件获取参数
    public void init() {                                      //Applet的初始化方法
        param = new String();
        //从HTML文件获取声音文件参数
        param = getParameter("soundA");
        if (param == null)
            param = "midiA.mid";
        soundA = getAudioClip(getDocumentBase(), param);
        param = getParameter("soundB");
        if (param == null)
            param = "midiB.mid";
        soundB = getAudioClip(getDocumentBase(), param);
        //从HTML文件获取图形按钮尺寸
    }
}

```

```

width = getSize().width;
height = getSize().height;
//从HTML文件获取超链接参数
param = getParameter("URL");
try {
    url = new URL(param);
} catch (MalformedURLException e) {
}
//创建图像并加载到图像跟踪器
offI = createImage(width, height);
offG = offI.getGraphics();
imageTracker = new MediaTracker(this);
param = getParameter("Images1");
img1 = getImage(getCodeBase(), param);
imageTracker.addImage(img1, 0);
param = getParameter("Images2");
img2 = getImage(getCodeBase(), param);
imageTracker.addImage(img2, 0);
param = getParameter("Images3");
img3 = getImage(getCodeBase(), param);
imageTracker.addImage(img3, 0);
try {
    imageTracker.waitForID(0);
} catch (InterruptedException e) {
}
//安装鼠标事件监听器
addMouseListener(this);
}
public void start() { //Applet的启动方法
    offG.drawImage(img1, 0, 0, width, height, this);
    repaint();
}
public void mouseClicked(MouseEvent e) { //点击鼠标事件
}
public void mousePressed(MouseEvent e) { //点住鼠标不放产生的事件
    offG.drawImage(img3, 0, 0, width, height, this);
    repaint(); //重画图像
    soundA.play(); //播放声音文件
}
public void mouseReleased(MouseEvent e) { //松开鼠标产生的事件
    offG.drawImage(img2, 0, 0, width, height, this);
    repaint(); //重画图像
    soundB.play(); //播放声音文件
}

```

```

        getAppletContext().showDocument(url); //链接到指定网页
    }

    public void mouseEntered(MouseEvent e) { //鼠标移入事件
        offG.drawImage(img2, 0, 0, width, height, this);
        repaint();
    }

    public void mouseExited(MouseEvent e) { //鼠标移出事件
        offG.drawImage(img1, 0, 0, width, height, this);
        repaint();
    }

    public void paint(Graphics g) { //画出图像
        g.drawImage(offI, 0, 0, width, height, this);
    }
}

```



案例2 模拟工具条

案例运行效果与操作

本案例是使用Java Applet响应鼠标事件，点击图标后转入相关的页面。程序运行后，界面如图1-4所示。



图1-4 运行界面

制作要点

1. URL显示
2. 鼠标点击事件的显示

步骤详解

在HTML源文件中定义好参数，由Applet获得参数，点击图片后，根据获得的参数，转入相关页面。

1. URL显示。

步骤如下：

- ① 使用URL类按接收字符串生成URL对象。
- ② 通过Applet类中的getAppletContext()方法取得小应用程序的AppletContext。
- ③ 通过AppletContext接口的showDocument方法显示URL内容。

该方法的具体格式和参数说明如下：

```
void showDocument(URL url, String target)
```

其中，target参数用于指定文件内容显示的位置，其具体值及作用如下：

“-self”	当前帧
“-parent”	父帧
“-top”	最顶部的帧
“-blank”	开辟新浏览器窗口显示
“帧名”	在指定的帧中显示

2. 鼠标点击事件。

参考本章案例1：图形按钮。

3. 嵌入Applet的HTML页面源码。

```
<HTML>
<HEAD>
<TITLE>
模拟工具条
</TITLE>
</HEAD>
<BODY>
模拟工具条<BR>
<applet code="SimToolBar.class" width=52 height=45>
<PARAM name="target" value="http://www.dzwww.com">
<PARAM name="enterImage" value="home2.gif">
<PARAM name="exitImage" value="home.gif">
<PARAM name="where" value="xinwen">
</applet>
<applet code="SimToolBar.class" width=52 height=45>
<PARAM name="target" value="http://bbs.dzwww.com">
<PARAM name="enterImage" value="next.gif">
<PARAM name="exitImage" value="next2.gif">
</applet>
<applet code="SimToolBar.class" width=52 height=45>
<PARAM name="target" value="http://www.baidu.com">
<PARAM name="enterImage" value="search2.gif">
```

```
<PARAM name="exitImage" value="search.gif">
<PARAM name="where" value="search_engine">
</applet>
<applet code="SimToolBar.class" width=52 height=45 >
<PARAM name="target" value="http://www.google.com">
<PARAM name="enterImage" value="channel2.gif">
<PARAM name="exitImage" value="channel.gif">
</applet>
</BODY>
</HTML>
```

程序源代码与解释

```
/*
 *本案例是使用java applet响应鼠标事件，点击图标后转入相关的页面。
 */
import java.awt.*;
import java.net.*;
import java.applet.*;
import java.awt.event.*;
import java.util.StringTokenizer;
public class SimToolBar extends Applet implements MouseListener {
    private MediaTracker myTracker;          //定义图像跟踪器
    private Image enterImage, exitImage;      //定义图像变量，分别代表鼠标处于不同位置时的图像
    private boolean onButton = false;          //定义布尔变量，初始化鼠标状态（是否在按钮上）
    private boolean clickButton = false;        //定义布尔变量，初始化鼠标状态（是否点击按钮）
    private int onImages = 0;                  //定义整型变量，指示鼠标位置
    private URL myURL;                      //定义超链接变量
    private String target, myString, where;   //定义字符串变量，确定超链接的目的地址和目的网页
    public void init() {                     //Applet的初始化方法
        //从HTML文件获取超链接参数（目的网页）
        if ((myString = getParameter("where")) != null)
            where = myString;
        else
            where = null;
        //加载图像并用图像跟踪器跟踪
        myTracker = new MediaTracker(this);
        enterImage = getImage(getDocumentBase(), getParameter("enterImage"));
```

```

myTracker.addImage(enterImage, 0);
exitImage = getImage(getDocumentBase(), getParameter("exitImage"));
myTracker.addImage(exitImage, 0);
//异常处理
try {
    myTracker.waitForAll();
} catch (InterruptedException e) {
}
//从HTML文件获取超链接参数（目的网址）
target = getParameter("target");
try {
    myURL = new URL(getDocumentBase(), target);      //获得转入页面的url
    System.out.println(target);
} catch (MalformedURLException mal) {
}
addMouseListener(this);      //加入鼠标事件监听器
}

public void start() {          //Applet的启动方法
    repaint();                  //重画
}

public void mouseClicked(MouseEvent e) {      //鼠标点击事件
}

public void mousePressed(MouseEvent e) {      //鼠标按下不放事件
    clickButton = true;
    repaint();
}

public void mouseReleased(MouseEvent e) {      //鼠标松开事件
    if (clickButton && onButton) {
        clickButton = false;
        repaint();
        //链接到指定网页
        if (where != null)
            getAppletContext().showDocument(myURL, where);
        else
            getAppletContext().showDocument(myURL);
    } else {
        clickButton = false;
        repaint();
    }
}

public void mouseEntered(MouseEvent e) {      //鼠标移入事件
    onButton = true;
}

```

```

        repaint();
    }

    public void mouseExited(MouseEvent e) {      //鼠标移出事件
        onButton = false;
        repaint();
    }

    public void drawImg(Graphics g, Image theImage) { //本方法实现了画出按钮图
像的功能
        int a;
        if (onImages == 3)
            a = 1;
        else
            a = 0;
        //确定图像宽度、高度和位置
        int imageWidth = theImage.getWidth(this);
        int imageHeight = theImage.getHeight(this);
        int imageX = (getSize().width / 2) - (imageWidth / 2);
        int imageY = (getSize().height / 2) - (imageHeight / 2);
        //画出图像
        imageY = (getSize().height / 2) - (imageHeight / 2) + 2;
        g.drawImage(theImage, imageX + a, imageY + a, this);
    }

    //本方法实现了在不同按钮边界画线的功能，增强显示效果
    public void drawButton(Graphics g) {
        Color topAndLeft;
        Color bottomAndRight;
        if (onImages == 2) {                      //鼠标在按钮上但没有点击时
            topAndLeft = Color.lightGray;        //按钮图像上侧、左侧边界颜色浅灰,
不明显
            bottomAndRight = Color.darkGray;     //按钮图像上侧、左侧边界颜色深灰,
明显
        } else {
            topAndLeft = Color.darkGray;
            bottomAndRight = Color.lightGray;
        }
        //根据设定的颜色画线
        g.setColor(topAndLeft);
        g.drawLine(0, 0, getSize().width, 0);
        g.drawLine(0, 1, 0, getSize().height);
        g.setColor(bottomAndRight);
        g.drawLine(getSize().width - 1, 1, getSize().width - 1,
                  getSize().height);
        g.drawLine(1, getSize().height - 1, getSize().width,

```

```

        getSize().height - 1);
    }

    public void paint(Graphics g) {           //画出图像
        if (!onButton) {                     //鼠标不在按钮上时的图像
            onImages = 1;
            drawImg(g, exitImage);          //调用drawImg方法，画出按钮图像
        } else if (onButton && !clickButton) { //鼠标在按钮上但没有点击时的图像
            onImages = 2;
            drawButton(g);                //调用drawButton方法，画出按钮边界
            drawImg(g, enterImage);       //调用drawImg方法，画出按钮图像
        } else {                           //鼠标点击按钮时的图像
            onImages = 3;
            drawButton(g);                //调用drawButton方法，画出按钮边界
            drawImg(g, enterImage);       //调用drawImg方法，画出按钮图像
        }
    }
}

```



案例3 Applet与Applet在页内的通信

案例运行效果与操作

本案例实现两个Applet在页内的通信，通过页面显示如图1-5（左）所示。

单击“发送到Receiver”，第二个Applet收到信息，如图1-5（右）所示。

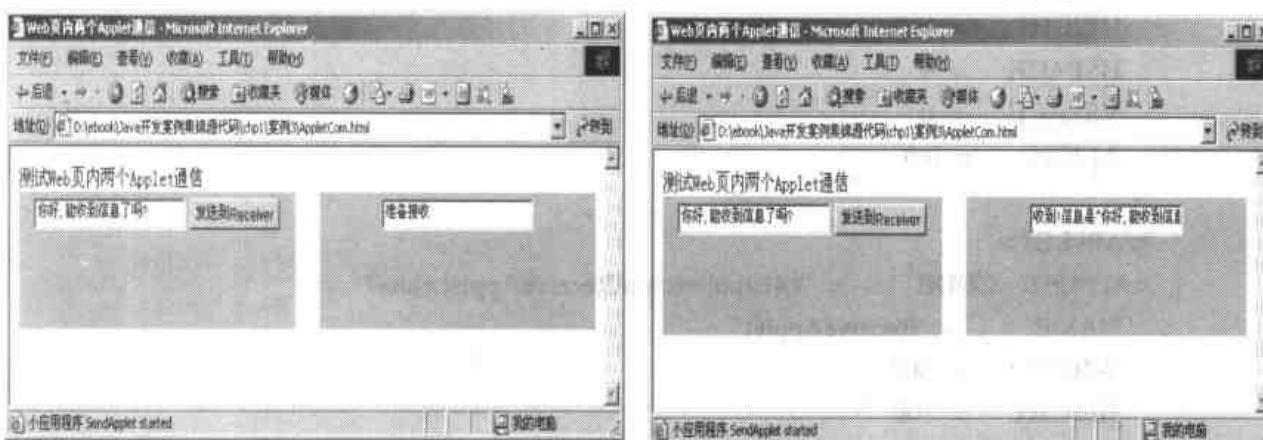


图1-5 “Web页内两个Applet通信”界面

制作要点

1. 接口java.applet.Applet类的使用

2. getApplet()方法的使用

步骤详解

1. Applet的通信。

Applet类中可以取得网页的上下文句柄，因此，同网页内的两个Applet是可以互相访问的。

GetApplet()方法返回与发出调用的Applet在同一个HTML页面上的Applet，它可以按名称找到同一个HTML页面中的另一个Applet。

```
ReceiveApplet receiveApplet =  
    (ReceiveApplet)getContext().getApplet("ReceiveApplet");
```

2. 嵌入Applet的HTML页面源码。

注意，在本例中ReceiveApplet的HTML代码一定要加上name="ReceiveApplet"，否则无法取得ReceiveApplet的句柄。

```
<HTML>  
<HEAD>  
<TITLE>  
Web页内两个Applet通信  
</TITLE>  
</HEAD>  
<BODY>  
测试Web页内两个Applet通信<BR>  
<APPLET CODE      = "twoappletscom\SendApplet.class"  
        NAME      = "SendApplet"  
        WIDTH     = 300  
        HEIGHT    = 100  
        HSPACE    = 0  
        VSPACE    = 0  
        ALIGN     = left  
>  
</APPLET>  
<APPLET CODE      = "twoappletscom\ReceiveApplet.class"  
        NAME      = "ReceiveApplet"  
        WIDTH     = 300  
        HEIGHT    = 100  
        HSPACE    = 0  
        VSPACE    = 0  
        ALIGN     = right  
>  
</APPLET>
```

3. 编译顺序

编译的时候先编译ReceiveApplet，这样SendApplet中才能引用ReceiveApplet这个类。

程序源代码与解释

第一个Applet:

```

package twoappletscom;           //两个Applet放在一个包中，以便于调用
// Web页内两个Applet通信，在同一网页中建立两个Applet的通信。
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class SendApplet extends Applet {
    TextField tfSend = new TextField("你好,能收到信息了吗?", 20); // 定义文本框
    Button btnSend = new Button("发送到Receiver"); // 定义发送按钮
    public void init() {                         //Applet的初始化方法
        setLayout(new FlowLayout());             //设置布局
        add(tfSend);                           //加入文本框
        add(btnSend);                          //加入按钮
    }
    public boolean action(Event ev, Object obj) { // 处理按钮事件
        if (ev.target instanceof Button) {       //发生了按钮事件
            String msgSend = tfSend.getText();   //从文本框中取出文本
            ReceiveApplet receiveApplet = (ReceiveApplet) getAppletContext()
                .getApplet("ReceiveApplet");      //建立ReceiveApplet的实例
            if (receiveApplet != null) {          //存在ReceiveApplet实例
                receiveApplet.AppendText(msgSend); //调用ReceiveApplet中方法
                return true;
            }
            else {
                tfSend.setText("没有找到ReceiveApplet");
                return false;
            }
        }
        return false;
    }
}

```

第二个Applet:

```

package twoappletscom;           //两个Applet放在一个包中，以便于调用
import java.awt.*;
import java.awt.event.*;

```

```

import java.applet.*;
public class ReceiveApplet extends Applet {
    TextField tfReceive = new TextField("准备接收", 20); //定义文本框
    public void init() { //ReceiveApplet的初始化方法
        setLayout(new FlowLayout()); //设置布局
        add(tfReceive); //加入文本框
    }
    public void AppendText(String msg) { //声明公共方法以便SendApplet调用
        tfReceive.setText("收到!信息是\"" + msg + "\""); //设置显示信息
    }
}

```



案例4 电子相册

案例运行效果与操作

本案例是使用Java Applet实现的电子相册效果。首先建立一个下拉框，改变下拉框里的内容，页面上显示不同的图片。程序运行后，界面如图1-6所示。



图1-6 运行界面

制作要点

1. 下拉式列表（Choice）的使用
2. 类java.awt.Component中方法Boolean action(Event evt, Object what)的使用

步骤详解

1. 下拉式列表的使用。

Choice类提供一个下拉列表，当用户选择某项时，会产生ItemEvent事件。用法如下：

```
Choice personchoice = new Choice();
Personchoice.add("张三");
Personchoice.add("李四");
```

2. 方法action()的使用。

在Java Applet程序中，单击按钮、键入文本、使用鼠标或执行任何与界面相关的动作时，就发生一个事件，小程序就会做出适当的反应。

我们通过使用action()方法来获得Java Applet小程序运行时所发生的事件。语句格式如下：

```
public boolean action(Event evt, Object arg)
{
    //逻辑处理
    return true;
    //处理完毕，不需要其他方法再做处理
}
```

action()方法中含有两个参数：一个是Event类的一个对象evt；另一个是Object类的一个对象arg。Event对象告诉我们发生了哪种事件，而Object对象将进一步告诉我们有关该事件的情况。当有Event监听的事件发生时，Java Applet便自动调用该action()方法。

3. 嵌入Applet的HTML页面源码。

```
<applet code= "ElectroAlbum.class" width= "400" height="400">
<param name="TotalPic" value="5">
<param name="Text1" value="照片一">
<param name="Text2" value="照片二">
<param name="Text3" value="照片三">
<param name="Text4" value="照片四">
<param name="Text5" value="照片五">
<param name="Picture1" value="Pic1.jpg">
<param name="Picture2" value="Pic2.jpg">
<param name="Picture3" value="Pic3.jpg">
<param name="Picture4" value="Pic4.jpg">
<param name="Picture5" value="Pic5.jpg">
</applet>
```

程序源代码与解析

```
/*
*本案例是使用java applet实现的电子相册效果,
*首先建立了一个下拉框，改变下拉框里的内容，页面上显示不同的图片。
```

```

*/
import java.applet.*;
import java.awt.*;
public class ElectroAlbum extends Applet {
    private Choice myChoice;           //定义一个下拉框
    private String[] myString1, myString2; //定义二个字符串数组
    private int totalPics;             //定义整型变量，代表相册中的照片总数
    private Image offI;                //定义图像变量
    private Image[] img;               //定义图像数组，用来存放各个照片
    private Graphics offG;             //定义图像变量
    private MediaTracker imagetracker; //定义图像跟踪器
    public void init() {               //Applet的初始化方法
        this.setLayout(null);
        myChoice = new Choice();         //建立下拉框实例
        myChoice.setBounds(10, 10, 290, 20); //设置下拉框
        //从Html文件中提取TotalPic参数值
        totalPics = Integer.parseInt(getParameter("TotalPic"));
        System.out.println(totalPics);
        myString1 = new String[totalPics]; //建立实例，存放相片说明文本
        myString2 = new String[totalPics]; //建立实例，存放相片的图像文件名
        img = new Image[totalPics];       //建立实例，存放相片图像
        for (int i = 0; i < totalPics; i++) {
            myString1[i] = new String("");
            myString2[i] = new String("");
        }
        //从Html文件中提取图像，并加入图像跟踪器
        String s = new String("");
        imagetracker = new MediaTracker(this);
        for (int i = 0; i < totalPics; i++) {
            s = getParameter("Text" + (i + 1));
            myString1[i] = s;
            System.out.println(myString1[i]);
            myChoice.addItem(s); //向下拉列表中增加选项
            s = getParameter("Picture" + (i + 1));
            myString2[i] = s;
            img[i] = getImage(getDocumentBase(), s);
            imagetracker.addImage(img[i], 0);
            System.out.println(myString2[i]);
        }
        try {
            imagetracker.waitForID(0);
        } catch (InterruptedException e) {
        }
    }
}

```

```

        add(myChoice);

        offI = createImage(getSize().width, getSize().height - 40);
        offG = offI.getGraphics();
        offI = img[0];
        offG.drawImage(offI, 0, 0, this);
        repaint();
    }

    public void paint(Graphics g) { //画出图像
        g.drawImage(offI, 10, 40, this);
    }

    //使用action()方法来获得Java Applet小程序运行时所发生的事件
    public boolean action(Event e, Object o) {
        if (e.target == myChoice) {
            offG.setColor(this.getBackground()); //设置背景
            offG.fillRect(0, 40, getSize().width, getSize().height - 40); //填充
            offI = img[myChoice.getSelectedIndex()]; //选择要显示的图像
            offG.drawImage(offI, 0, 0, this); //显示图像
            repaint(); //重画
        }
        return true;
    }
}

```



案例5 百叶窗效果

案例运行效果与操作

本案例是利用Java Applet动态改变显示的图片，并产生百叶窗效果，如图1-7所示。



图1-7 运行界面

制作要点

1. 类PixelGrabber的使用
2. 使用类MemoryImageSource创建图像

步骤详解**1. 类PixelGrabber的使用。**

当我们获得了图像后，可以通过java.awt.image.PixelGrabber包中的PixelGrabber方法来将图像中的像素信息完全读取出来，其用法如下：

```
PixelGrabber(Image img, int x, int y, int w, int h, int[] pix, int off, int scansize)
```

其中是要读取的图像，x/y/>是要读取图像中的左上角坐标，w/h/>分别是从x/y开始起的距离，其实x,y,w,h就是一个矩形，pix是保存像素的数组，off是读取图像时开始的位置，scansize是指扫描的宽度，一般来说和w相等。

下面通过一个例子来讲解类PixelGrabber的使用。

在网页设计过程中，有时为了模拟单色VGA的显示效果，或为了进行某种形象的夸张设计，而将彩色图形画面变为黑白显示，这就需要使用彩色到黑白的灰度变换技术。

灰度变换的算法其实很简单，只要提取每个像素点的红、绿、蓝三原色，然后根据公式：灰度值 = 红色亮度值 × 30% + 绿色亮度值 × 59% + 蓝色亮度值 × 11%，计算出一个灰度值，并将其作为红、绿、蓝三原色的新值重新写回显存即可。

具体步骤如下：

- 调用Graphics对象的drawImage()方法，在applet中显示出一幅名为TEST.JPG的彩色图像，drawImage()的调用形式为：g.drawImage(name,x,y, width, height, this)。
- 定义一个数组存放这幅彩色图像的RGB值，数组大小就是图像的像素个数。使用PixelGrabber()获取每个像素点的RGB值。
- 使用灰度变换公式计算出每个像素点的灰度值，并将其作为新的RGB值存放回数组中，再调用createImage()方法构造出新的黑白图像。
- 显示该黑白图片。为了防止变换过程中的闪烁现象，程序中还使用了双缓冲技术，即先在虚屏中画好图像，再一次性地显示出来。程序中使用鼠标来控制彩色到黑白的变换。

灰度变换的tt4.java源程序如下所示：

```
import java.applet.*;
import java.awt.*;
import java.awt.image.*;
public class tt4 extends Applet
{
    Image art,Buf;
    int onced=0;
```

```

boolean is_color=true;
Graphics Bufg;//使用双缓冲区技术抑制闪烁;
Dimension xy=null;
public void init()
{ art=getImage(getDocumentBase(),"test.jpg");
  resize(640, 480);//装入图片;
}
public void paint(Graphics g)
{ if (onced==0)
  //如果是第一次装入图片，则直接显示,
  { g.drawImage(art,0,0,this);
  }
  if ((onced==1)||(onced==2))
  //如果正在进行灰度变换，则提示等待,
  { g.setColor(new Color(255,200,0));
    g.drawString("running!", 1, 30);
  }
  if (onced==3)
  //如果灰度变换完毕，则显示结果;
  { if (is_color) g.drawImage(Buf,0,0,this);
    else g.drawImage(art,0,0,this);
    is_color=!is_color;//在彩色与黑白之间变化,
  }
}
public boolean mouseDown(Event evt, int x, int y)
{ if (onced==0)
  { onced=1;
  }
  repaint();//用鼠标触发事件;
  return true;
}
public boolean mouseUp(Event evt, int x, int y)
{ if (onced==1)
  { onced=2;
    int wd=art.getWidth(this); //取得图片宽;
    int ht=art.getHeight(this); //取得图片高;
    GetPixels(art,0,0,wd,ht); //调用灰度变换方法;
  }
  return true;
}
public void GetPixels(Image img,int x,int y,int
w,int h)
{ int[]pixels=new int[w*h];

```

```
//定义一块内存空间;
int gray;
PixelGrabber pg=new PixelGrabber(img,x,y,w,h,
pixels,0,w);
try{pg.grabPixels();
}
catch(InterruptedException)
{System.err.println("interrupted waiting for
pixels!");
return;
}
for(int j=0;j<w;j++)
{
gray=(int)((pixels[w*j+i] >> 16)&0xff)*0.3;
gray+=(int)((pixels[w*j+i]>>8)&0xff)*0.59;
gray+=(int)((pixels[w*j+i] )&0xff)*0.11;
//由红，绿，蓝值得到灰度值;
pixels[w*j+i]=(255<<24)|(gray<<16)|(gray<<8)|gray;
}
}
Image pic=createImage(new MemoryImageSource
(w,h,pixels,0,w));
Bufg.drawImage(pic,0,0,this); //显示黑白图片;
onced=3;
repaint();
}
public void update(Graphics g)
{
if(xy==null)
{
xy=this.size();
Buf=createImage(xy.width,xy.height);
Bufg=Buf.getGraphics();
}
paint(g);//修改update方法，避免闪烁;
}
}
```

将源程序编译后，得到tt4.class文件，利用Java Applet Wizard自动生成的HTML文件（tt4.html），就可在浏览器中欣赏它的效果了。

2. 包含applet的HTML文件如下所示。

```
<HTML>
<HEAD>
<TITLE>
百叶窗效果
```

```

</TITLE>
</HEAD>
<BODY>
百叶窗效果<BR>
<applet code="Shutter.class" width=320 height=220>
<param name="image1" value="Pic1.jpg">
<param name="image2" value="Pic2.jpg">
<param name="image3" value="Pic3.jpg">
<param name="image4" value="Pic4.jpg">
<param name="image5" value="Pic5.jpg">
<param name="delay" value="3000">
</applet>
</BODY>
</HTML>

```

3. 使用类MemoryImageSource创建图像。

MemoryImageSource类是ImageProducer接口的一个实现，使用数组生成图象的像素值。MemoryImageSource同样能用于管理随时间改变的内存图像，使得动画和自定义显示效果成为可能。

```

//使用数组产生新的图片
myImageToShow = createImage(new MemoryImageSource(
    myIMGWidth, myIMGHeight, pixA, 0, myIMGWidth));

```

程序源代码与释义

```

/**
 * 本案例是利用java applet动态改变显示的图片，并产生百叶窗效果
 */
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.awt.image.*;
public class Shutter extends Applet implements Runnable { //用Runnable接口实现多线程
    private Image myIMG[], myImageToShow; //定义图像数组和要显示的图像变量
    private MediaTracker imageTracker; //定义图像跟踪器
    //定义图像宽度、高度、图像总数、当前图像编号、下一个图像编号
    private int myIMGWidth, myIMGHeight, totalImage = 5, currentImage, nextImage;
    private Thread mythread; //定义线程
    private int delay; //定义延时
    //定义整型数组，分别用来存放各图像的像素
    private int totalPix, pix1[], pix2[], pix3[], pix4[], pix5[], pixA[], pixB[];
}

```

```

public void init() {                                //Applet的初始化方法
    this.setBackground(Color.black);      //设置背景
    //从Html文件中提取图像，并加入图像跟踪器
    myIMG = new Image[totalImage];
    imageTracker = new MediaTracker(this);
    String s = new String("");
    for (int i = 0; i < totalImage; i++) {
        s = getParameter("image" + (i + 1));
        myIMG[i] = getImage(getCodeBase(), s);
        imageTracker.addImage(myIMG[i], 0);
    }
    try {
        imageTracker.waitForID(0);
    } catch (InterruptedException e) {
    }
    //设置延时3秒钟
    if (getParameter("delay") == null) {
        delay = 3000;
    } else {
        delay = Integer.parseInt(getParameter("delay"));
    }
    //以第一幅图像的尺寸为显示尺寸
    myIMGWidth = myIMG[0].getWidth(this);
    myIMGHeight = myIMG[0].getHeight(this);
    totalPix = myIMGWidth * myIMGHeight; //要显示的总像素数
    //获取第一幅图像的像素
    pix1 = new int[totalPix];
    PixelGrabber PG1 = new PixelGrabber(myIMG[0], 0, 0, myIMGWidth,
                                         myIMGHeight, pix1, 0, myIMGWidth);
    try {
        PG1.grabPixels();
    } catch (InterruptedException ex) {
    }
    //获取第二幅图像的像素
    pix2 = new int[totalPix];
    PixelGrabber PG2 = new PixelGrabber(myIMG[1], 0, 0, myIMGWidth,
                                         myIMGHeight, pix2, 0, myIMGWidth);
    try {
        PG2.grabPixels();
    } catch (InterruptedException ex) {
    }
    //获取第三幅图像的像素

```

```

pix3 = new int[totalPix];
PixelGrabber PG3 = new PixelGrabber(myIMG[2], 0, 0, myIMGWidth,
myIMGHeight, pix3, 0, myIMGWidth);
try {
    PG3.grabPixels();
} catch (InterruptedException ex) {
}
//获取第四幅图像的像素
pix4 = new int[totalPix];
PixelGrabber PG4 = new PixelGrabber(myIMG[3], 0, 0, myIMGWidth,
myIMGHeight, pix4, 0, myIMGWidth);
try {
    PG4.grabPixels();
} catch (InterruptedException ex) {
}
//获取第五幅图像的像素
pix5 = new int[totalPix];
PixelGrabber PG5 = new PixelGrabber(myIMG[4], 0, 0, myIMGWidth,
myIMGHeight, pix5, 0, myIMGWidth);
try {
    PG5.grabPixels();
} catch (InterruptedException ex) {
}
currentImage = 0; //当前图像设置为第一幅
pixA = new int[totalPix]; //整型数组存放当前图像像素
pixB = new int[totalPix]; //整型数组存放要显示图像像素
myImageToShow = myIMG[0]; //要显示的图像设置为第一幅
mythread = new Thread(this);
mythread.start(); //启动线程
}
public void paint(Graphics g) { //画出要显示的图像
    g.drawImage(myImageToShow, 0, 0, this);
}
public void update(Graphics g) { //重载update方法，防止闪烁
    paint(g);
}
public void run() { //Applet的运行方法
    if (mythread == null) { //如果没有线程在运行
        mythread = new Thread(this);
        mythread.start(); //启动一个线程
    }
    while (true) { //无限循环
}

```

```
try {
    mythread.sleep(delay); //线程延时
    nextImage = ((currentImage + 1) % totalImage); //确定下一幅图像
    if (currentImage == 0) { //显示第一幅图片
        //使用数组复制arraycopy方法，复制图片像素
        //复制第一幅图片像素到pixA
        System.arraycopy(pic1, 0, pixA, 0, totalPix);
        //复制第二幅图片像素到pixB
        System.arraycopy(pic2, 0, pixB, 0, totalPix);
        //用MemoryImageSource方法将像素数组pixA对应到图像
        myImageToShow = createImage(new MemoryImageSource(
            myIMGWidth, myIMGHeight, pixA, 0, myIMGWidth));
        //重画以显示图像
        repaint();
    }
    if (currentImage == 1) { //显示第二幅图片
        System.arraycopy(pic2, 0, pixA, 0, totalPix);
        System.arraycopy(pic3, 0, pixB, 0, totalPix);
        myImageToShow = createImage(new MemoryImageSource(
            myIMGWidth, myIMGHeight, pixA, 0, myIMGWidth));
        repaint();
    }
    if (currentImage == 2) { //显示第三幅图片
        System.arraycopy(pic3, 0, pixA, 0, totalPix);
        System.arraycopy(pic4, 0, pixB, 0, totalPix);
        myImageToShow = createImage(new MemoryImageSource(
            myIMGWidth, myIMGHeight, pixA, 0, myIMGWidth));
        repaint();
    }
    if (currentImage == 3) { //显示第四幅图片
        System.arraycopy(pic4, 0, pixA, 0, totalPix);
        System.arraycopy(pic5, 0, pixB, 0, totalPix);
        myImageToShow = createImage(new MemoryImageSource(
            myIMGWidth, myIMGHeight, pixA, 0, myIMGWidth));
        repaint();
    }
    if (currentImage == 4) { //显示第五幅图片
        System.arraycopy(pic5, 0, pixA, 0, totalPix);
        System.arraycopy(pic1, 0, pixB, 0, totalPix);
        myImageToShow = createImage(new MemoryImageSource(
            myIMGWidth, myIMGHeight, pixA, 0, myIMGWidth));
        repaint();
    }
}
```



案例6 波浪彩虹文字

案例运行效果与操作

本案例采用Java Applet实现文字颜色的不停改变，并产生波浪效果。程序运行后，界面如图1-8所示。

制作要点

1. 类Font的使用
 2. 类FontMetrics的使用



图1-8 运行界面

步骤详解

- 字体（Font类）Font类代表字体。它对所有的字体处理进行了抽象，允许动态地选择字体，因此为AWT或Swing中文字的显示提供了灵活性。通过类Graphics或组件的方法getFont()或setFont()可以获取或设置当前使用的字体（Font类的对象）。类Graphics2D也继承了类Graphics的这两个方法。对于某种字体来说，可以设置它为普通（Font.PLAIN）、粗体（Font.BOLD）、斜体（Font.ITALIC）或粗斜体（Font.BOLD+Font.ITALIC）的字体样式，设置字号（字体的大小），同时还可以获得该字体的名称（用字符串来表示）。

下例中显示了系统支持的一些字体。例如，通过

```
GraphicsEnvironment ge=GraphicsEnvironment.getLocalGraphicsEnvironment();
String[] fontList=ge.getAvailableFontFamilyNames();
```

来得到用户系统的图形环境中所支持的字体列表。

下面的例子说明了不同字体的输出。

```
import java.applet.Applet;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GraphicsEnvironment;
public class AllFonts extends Applet {
    String fontList[];
    public void init() {
        GraphicsEnvironment ge = GraphicsEnvironment
            .getLocalGraphicsEnvironment();
        fontList = ge.getAvailableFontFamilyNames();
    }
    public void paint(Graphics g) {
```

```

        Dimension d = getSize();
        int x = d.width / 4, y = d.height / 10 - 5;
        for (int i = 0; i < 10; i++) {
            Font f;
            f = new Font(fontList[i], Font.PLAIN, 12);
            g.setFont(f);
            g.drawString(fontList[i], 5, y * (i + 1));
            f = new Font(fontList[i], Font.BOLD, 12);
            g.setFont(f);
            g.drawString(fontList[i], x, y * (i + 1));
            f = new Font(fontList[i], Font.ITALIC, 12);
            g.setFont(f);
            g.drawString(fontList[i], x + x, y * (i + 1));
            f = new Font(fontList[i], Font.BOLD + Font.ITALIC, 12);
            g.setFont(f);
            g.drawString(fontList[i], x + x + x, y * (i + 1));
        }
    }
}

```

运行结果如图1-9所示。



图1-9 运行结果

上例中使用以下构造方法创建了一个Font对象：Font f=new Font(String name,int style,int size)。其中name为字体的字面名或逻辑字体名，style为字体的样式，size为字体的大小。在Java中，字体名可以有三种表示方式：

(1) 逻辑字体名 (LogicalName)

在Java早期版本中使用。为了保持向后兼容，现在仍然保留。如AWT定义了5个逻辑字体名：

SansSerif

Serif

Monospaced

Dialog

DialogInput

另外，Java早期版本中把Helvetica、TimesRoman、Courier和ZapfDingbats用作逻辑字体名。为了向后兼容，这些字体名仍被用作逻辑字体名，尽管Helvetica是一个真正的字体名（字体的字面名），而TimesRoman和ZapfDingbats真正的字体名分别是Times Roman和Zapf Dingbats。

(2) 字体的族名 (FamilyName)

一个字体族中包含了多个字体。

(3) 字体的字面名 (FaceName)

是真正的字体名。一个字体的字面名属于一个字体族，像Helvetica、Helvetica Bold、Helvetica Italic、Helvetica Ultra Condensed都是属于Arial族的一部分。可以通过调用GraphicsEnvironment.getAllFonts()获得安装在系统上的所有真正的字体。可通过下面的方法获得字体的逻辑名、族名和字面名：

获得字体的字面名（比如Helvetica Bold）：

```
String getFontName()
```

获得字体的家族名（比如Helvetica）：

```
String getFamily()
```

如果该字体是用一个逻辑字体名创建的，就得到逻辑字体名；反之，则得到字体的字面名。

```
String getName()
```

2. 字模 (FontMetrics类)。

FontMetrics类表示字模，这个类提供了一系列方法，通过调用它们可以得到用某种字体表示的一个字符串在屏幕上的特定尺寸。一个字符可以由三条线：基线（base line）、底线（descender line）和顶线（ascender line）来分割，ascent指明基线和顶线之间的像素数，descent指明基线和底线之间的像素数。另外，前进宽度（advance width）指明两个相邻字符起始位置之间的间隔，通常为一个字符的宽度；行间隔（leading）指明上一行字符的底线与下一行字符顶线之间的距离，高度（height）为ascent、descent和leading之和。如图1-10所示。

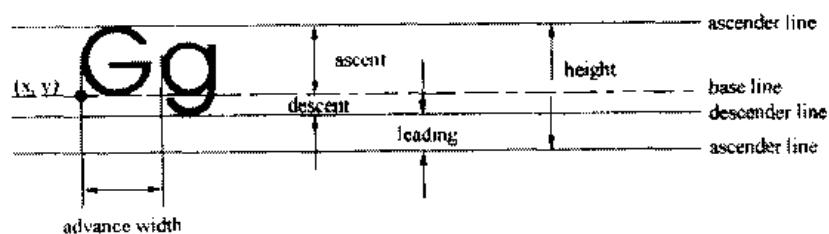


图1-10 字模说明

字模名称示意图中用(x,y)指定一个字符在组件中的起始位置时，并不是指明字符左上角的位置，而是指明字符的水平基线(base line)的起始点。通过Graphics类或组件的方法getFontMetrics()可以获取当前字体的字模(一个FontMetrics类的对象)。注意，Graphics2D类也继承了Graphics类的该方法。当得到一个FontMetrics类的对象(字模)后，可以通过FontMetrics对象的方法获取字体的各种信息，这些方法包括getAscent()、getDescent()、getHeight()和getLeading()等。同时，方法stringWidth()、charWidth()等还可以返回一串字符或一个字符的宽度。下面的例子是在一个窗口的中央显示一行字母。例中使用粗体和粗斜体字母来混合显示这些文字，为了测量用多种字体样式显示的一个字符串，需要调用FontMetrics类的一些方法。

下面介绍利用FontMetrics类的方法精确输出文字。

```
//利用FontMetrics类的方法精确输出文字。
import java.awt.Dimension;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import javax.swing.JApplet;
import javax.swing.JPanel;

public class DisplayString extends JApplet {
    public void init() {
        setContentPane(new DisplayPanel());
    }
}
class DisplayPanel extends JPanel {
    private Font f;
    private Font fi;
    private FontMetrics fm;
    private FontMetrics fim;
    public void setFonts(Graphics g) {
        if (f != null)
            return;
        f = new Font("SansSerif", Font.BOLD, 14);
        fi = new Font("SansSerif", Font.BOLD + Font.ITALIC, 14);
        fm = g.getFontMetrics(f);
        fim = g.getFontMetrics(fi);
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        setFonts(g);
        String s1 = "Not a ";
        String s2 = "Hello World";
        g.drawString(s1, 100, 100);
        g.drawString(s2, 100, 150);
    }
}
```

```

        String s3 = " Program";
        int w1 = fm.stringWidth(s1);
        int w2 = fm.stringWidth(s2);
        int w3 = fm.stringWidth(s3);
        Dimension d = getSize();
        int cx = (d.width - w1 - w2 - w3) / 2;
        int cy = (d.height - fm.getHeight()) / 2 + fm.getAscent();
        g.setFont(f);
        g.drawString(s1, cx, cy);
        cx += w1;
        g.setFont(fi);
        g.drawString(s2, cx, cy);
        cx += w2;
        g.setFont(f);
        g.drawString(s3, cx, cy);
    }
}

```

运行结果如图1-11所示。



图1-11 运行结果

3. 嵌入Applet的HTML页面源码。

```

<applet code="WaveText.class" width=600 height=100>
    <param name=word value="波浪彩虹文字示例">
</applet>

```

程序源代码与解释

```

/**
 * 本案例是采用java applet实现文字颜色的不停改变，并产生波浪效果。
 */
import java.awt.*;
import java.applet.*;

```

```

//由Runnable接口实现多线程
public class WaveText extends Applet implements Runnable {
    String myString = null; //定义字符串，代表显示的文字
    int direct = 1;
    int Hrad = 12;
    int Vrad = 12;
    Thread mythread = null; //定义一个线程
    char[] words; //定义存放显示文字的字符数组
    int phase = 0; //定义相角
    Image offI; //定义图像
    Graphics offG; //定义图形
    Color[] colors; //定义颜色数组
    private Font f; //定义字体
    private FontMetrics fm; //定义字模
    public void init() { //Applet的初始化方法
        String param = null;
        myString = getParameter("word"); //从Html中获得文字
        setBackground(Color.black); //设置黑色背景
        words = new char[myString.length()];
        myString.getChars(0, myString.length(), words, 0); //将字符串中的各个字符保存到数组中
        //双缓冲技术在内存中创建图像
        offI = createImage(getSize().width, getSize().height);
        offG = offI.getGraphics();
        f = new Font("TimesRoman", Font.BOLD, 36); //设置字体
        fm = getFontMetrics(f);
        offG.setFont(f);
        float hue; //颜色的色元
        colors = new Color[myString.length()];
        for (int i = 0; i < myString.length(); i++) {
            hue = ((float) i) / ((float) myString.length());
            colors[i] = new Color(Color.HSBtoRGB(hue, 1.0f, 1.0f)); //颜色分配
        }
    }
    public void start() { //Applet的启动方法
        if (mythread == null) { //如果线程为空，则
            mythread = new Thread(this);
            //开始新的线程
            mythread.start(); //启动线程
        }
    }
    //终止线程
}

```

```

public void stop() { //Applet的停止方法
    if (mythread != null) { //如果线程不为空，则
        mythread.stop(); //终止线程，使它
        mythread = null; //为空
    }
}

//运行线程
public void run() { //Applet的运行方法
    while (mythread != null) {
        try {
            mythread.sleep(200); //让线程沉睡200毫秒
        } catch (InterruptedException e) {
        }
        repaint(); //重新绘制界面
    }
}

public void update(Graphics g) { //重写update方法，解决闪烁问题
    int x, y;
    double angle;
    offG.setColor(Color.black); //设置颜色
    offG.fillRect(0, 0, getSize().width, getSize().height); //填充
    phase += direct; //相角增加
    phase %= 8;
    //将要显示的各个字符按照不同的角度和颜色显示出来
    for (int i = 0; i < myString.length(); i++) {
        angle = ((phase - i * direct) % 8) / 4.0 * Math.PI; //弧度转换角度
        x = 20 + fm.getMaxAdvance() * i + (int) (Math.cos(angle) * Hrad); //X坐标
        y = 60 + (int) (Math.sin(angle) * Vrad); //Y坐标
        offG.setColor(colors[(phase + i) % myString.length()]); //设置颜色
        offG.drawChars(words, i, 1, x, y); //显示字符
    }
    paint(g); //调用paint方法显示图像
}

public void paint(Graphics g) { //画出图像
    g.drawImage(offI, 0, 0, this);
}
}

```



案例7 3D立体渐层文字

案例运行效果与操作

本案例是采用Java Applet实现文字的3D效果，并随着时间的改变，不停地改变字体的颜色。程序运行后，界面如图1-12所示。



图1-12 运行界面

制作要点

1. 随机产生RGB颜色值
2. 使用for循环实现3D文字渐层显示

步骤详解

1. 3D文字效果的实现，使文字颜色伴随时间改变。

利用随机数定义颜色：

```
R = (int) (255 * Math.random());
G = (int) (255 * Math.random());
B = (int) (255 * Math.random());
try {
    mythread.sleep(3000);
} catch (InterruptedException ex) {
}
offG.setColor(Color.blue);
offG.fillRect(0, 0, width, height);
repaint();
for (int i = 0; i < 10; i++) {
    offG.setColor(new Color((255-(255-R)*i/10),
        (255-(255-G)*i/10), (255-(255-G)*i/10)));
}
```

```
    offG.drawString(text, x - i, y - i);
    repaint();
```

2. 嵌入Applet的HTML页面源码。

```
<HTML>
<HEAD>
<TITLE>
3D立体渐层文字
</TITLE>
</HEAD>
<BODY>
3D立体渐层文字示例<BR>
<applet code=Text3D.class width=250 height=60>
<param name=Text value=3D立体渐层文字>
</applet>
</BODY>
</HTML>
```

程序源代码与解析

```
/*
*本案例是采用java applet实现文字的3D效果，并
*随着时间的改变，不停的改变字体的颜色。
*/
import java.awt.*;
import java.applet.*;
//由Runnable接口实现多线程
public class Text3D extends Applet implements Runnable {
    private Image myimage;
    private Image offI;
    private Graphics offG;
    private Thread mythread = null;
    private MediaTracker myimageTracker;
    private int height, width;
    private String text;
    private int fontSize;
    private Font font;
    public void init() { //Applet的初始化方法
        width = this.size().width;
        height = this.size().height;
        this.setBackground(Color.gray);
        myimage = createImage(width, height);
```

```

text = new String("Hello");
//从Html文件中提取要显示的文本
String myString = new String(getParameter("Text"));
if (myString != null)
    text = myString;
fontSize = 30;
font = new Font("TimesRoman", Font.BOLD, fontSize); //设置字体
//创建内存图像并用图像跟踪器跟踪
myimageTracker = new MediaTracker(this);
myimageTracker.addImage(myimage, 0);
try {
    myimageTracker.waitForID(0);
} catch (InterruptedException e) {
}
offI = createImage(width, height);
offG = offI.getGraphics();
}

public void start() { //Applet的启动方法
    if (mythread == null) {
        mythread = new Thread(this);
        mythread.start();
    }
}

public void run() { //Applet的运行方法
    int x = 15;
    int y = height - 15;
    int R, G, B; //定义变量，保存RGB信息
    offG.setFont(font);
    while (mythread != null) {
        //利用随机数定义颜色
        R = (int) (255 * Math.random());
        G = (int) (255 * Math.random());
        B = (int) (255 * Math.random());
        try {
            mythread.sleep(3000); //延时3秒
        } catch (InterruptedException ex) {
        }
        offG.setColor(Color.blue); //设置颜色
        offG.fillRect(0, 0, width, height); //填充
        repaint();
        for (int i = 0; i < 10; i++) { //分成10层进行显示
            offG.setColor(new Color((255-(255-R)*i/10),

```

```

        (255-(255-G)*i/10), (255-(255-G)*i/10));
    offG.drawString(text, x - i, y - i); //在不同位置(层次)显示字符串
    repaint();
}
try {
    mythread.sleep(50);
} catch (InterruptedException e) {
}
}
}

public void update(Graphics g) {           //重写update方法，消除闪烁
    paint(g);
}

public void paint(Graphics g) {           //画出图像
    g.drawImage(offI, 0, 0, this);
}
}

```

案例8 飞行文字



案例运行效果与操作

本案例是采用Java Applet实现文字的由小变大、由远及近，让人感觉文字是慢慢飞出来的。程序运行后，界面如图1-13和图1-14所示。



图1-13 运行界面

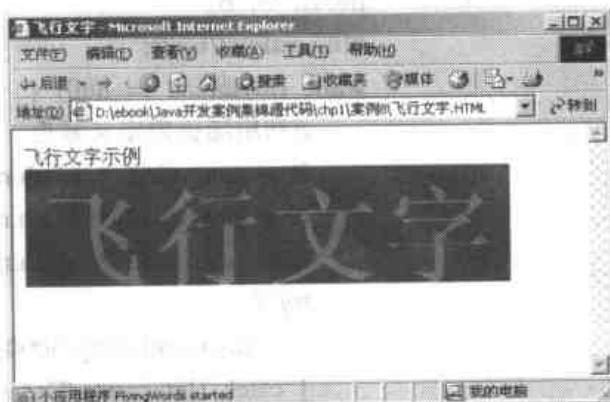


图1-14 点击菜单产生的界面

制作要点

字模类FontMetrics的使用

步骤详解

- 实现文字的逐渐变大，重心不断下移。

```

myGraphic.setColor(Color.blue);           //设置颜色
myGraphic.fillRect(0, 0, getSize().width, getSize().height); //设置位置
font = new Font("TimesRoman", Font.BOLD, fontSize); //定义字体
myGraphic.setFont(font);
myGraphic.setColor(Color.red);
FontMetrics fm = myGraphic.getFontMetrics(font);
int fontHeight = fm.getHeight();
int w;
int baseLine = getSize().height / 2 + fontHeight / 2;
w = fm.stringWidth(myString);
w = (getSize().width - w) / 2;
myGraphic.drawString(myString, w, baseLine -= 20);
g.drawImage(myImage, 0, 0, this);
fontSize++;                                //字体尺寸增加

```

- 嵌入Applet的HTML页面源码。

```

<HTML>
<HEAD>
<TITLE>
飞行文字
</TITLE>
</HEAD>
<BODY>
飞行文字示例<BR>
<applet code=FlyingWords.class width=400 height=100 VIEWASTEXT>
<param name="text" value="飞行文字">
</applet>
</BODY>
</HTML>

```

程序源代码与解析

```

/**
 * 本案例是采用java applet实现文字的由小变大
 * 由远及近，让人感觉文字是慢慢飞出来的
 */
import java.awt.*;
import java.applet.*;

```

```

//由Runnable接口实现多线程
public class FlyingWords extends Applet implements Runnable {
    private Image myImage;
    private Graphics myGraphic;
    private Font font;
    private String myString;
    private Thread mythread;
    private int fontSize;
    public void init() {                                //Applet的初始化方法
        String myText;
        myImage = createImage(getSize().width, getSize().height);
        myGraphic = myImage.getGraphics();
        //从Html文件中提取要显示的文本
        myText = getParameter("text");
        if (myText == null)
            myString = "你好！";
        else
            myString = myText;
        font = new Font("TimesRoman", Font.BOLD, 10);
    }
    public void start() {                                //Applet的启动方法
        if (mythread == null) {
            mythread = new Thread(this);
            mythread.start();
        }
    }
    public void update(Graphics g) {                    //重写update方法，消除闪烁
        paint(g);
    }
    public void paint(Graphics g) {                     //画出图像（文字）
        myGraphic.setColor(Color.blue);                //设置图像颜色
        myGraphic.fillRect(0, 0, getSize().width, getSize().height); //填充图像
        font = new Font("TimesRoman", Font.BOLD, fontSize); //设置字体
        myGraphic.setFont(font);
        myGraphic.setColor(Color.red);                 //设置字体颜色
        FontMetrics fm = myGraphic.getFontMetrics(font); //设置字模
        int fontHeight = fm.getHeight();
        int w;
        int baseLine = getSize().height / 2 + fontHeight / 2; //设置基线
        w = fm.stringWidth(myString);
        w = (getSize().width - w) / 2;
        myGraphic.drawString(myString, w, baseLine -= 20); //在图像上画出字符串
    }
}

```

```

        g.drawImage(myImage, 0, 0, this);
        fontSize++; //字体尺寸增加
    }

    public void run() { //Applet的运行方法
        while (true) { //无限循环
            repaint();
            if (fontSize > getSize().height) //如果字体尺寸超过图像尺寸，则
                fontSize = 0; //字体尺寸置0
            try {
                mythread.sleep(50);
            } catch (InterruptedException e) {
            }
        }
    }
}

```



案例9 聚光灯效果

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个聚光灯效果的文字特效，如图1-15所示。



图1-15 聚光灯效果的文字特效

制作要点

1. 和HTML网页通信
2. 字模类FontMetrics的使用

步骤详解

1. 初始化Applet。

```
myText = getParameter("text"); //读取页面参数，即需要显示的文字
if (myText == null)
    myText = "你好！";
fonts = getParameter("fontSize"); //读取字体大小
if (fonts == null)
    fontSize = 20;
else
    fontSize = Integer.parseInt(fonts);
font = new Font("TimesRoman", Font.BOLD, fontSize);
FontMetrics fm = getFontMetrics(font);
fontHeight = fm.getHeight();
baseLine = getSize().height / 2 + fontHeight / 3;
myTextWidth = fm.stringWidth(myText);
w = fm.stringWidth(myText);
w = (getSize().width - w) / 2;
spotPosition = w;
setBackground(Color.black); //设置背景颜色
```

2. 启动Applet。

```
if (mythread == null) {
    //创建并启动线程
    mythread = new Thread(this);
    mythread.start();
```

3. 启动线程。

```
repaint(); //重新输出Applet,调用update()方法
try { Thread.sleep(50); }
catch(InterruptedException e) { }
```

4. 输出Applet。

```
g.setFont(font);
g.setColor(Color.blue);
g.drawString(myText, w, baseLine); //第一遍显示
g.clipRect(spotPosition, 0, myTextSize, getSize().height); //设置裁剪区域
g.setColor(Color.white);
g.drawString(myText, w, baseLine); //第二遍显示
spotPosition = (spotPosition + 1) % (myTextWidth + 100); //移动聚光灯位置
```

5. 调用Applet的网页。

```
<HTML>
<HEAD>
```

```

<TITLE>
聚光灯效果
</TITLE>
</HEAD>
<BODY>
聚光灯效果示例<BR>
<applet code="SpotLight.class" width=300 height=50 VIEWASTEXT>
<param name="text" value="聚光灯效果示例">
<param name="fontSize" value="30">
</applet>
</BODY>
</HTML>

```

程序源代码与解析

```

/**
 *本案例实现一个聚光灯效果的文字特效
 */
import java.awt.*;
import java.applet.*;
public class SpotLight extends Applet implements Runnable {
    private String myText;
    private Font font;
    private int fontSize;
    private Thread mythread;
    private int spotPosition = 50;           //聚光灯位置
    private int myTextSize = 20;
    private int myTextWidth = 0;
    private int fontHeight, baseLine, w;
    public void init() {
        String fonts, temp;
        myText = getParameter("text");      //读取页面参数，即需要显示的文字
        if (myText == null)
            myText = "你好！";
        fonts = getParameter("fontSize");  //读取字体大小
        if (fonts == null)
            fontSize = 20;
        else
            fontSize = Integer.parseInt(fonts);
        font = new Font("TimesRoman", Font.BOLD, fontSize);
        FontMetrics fm = getFontMetrics(font);
        fontHeight = fm.getHeight();
    }
}

```

```

baseLine = getSize().height / 2 + fontHeight / 3;
myTextWidth = fm.stringWidth(myText);
w = fm.stringWidth(myText);
w = (getSize().width - w) / 2;
spotPosition = w;
setBackground(Color.black); //设置背景颜色
}
public void start() {
    //创建并启动线程
    if (mythread == null) {
        mythread = new Thread(this);
        mythread.start();
    }
}
public void stop() {      //停止applet
    mythread.stop();
    mythread = null;
}
public void run() {      //启动线程
    while (true) {
        repaint();
        try {
            mythread.sleep(50);
        } catch (InterruptedException e) {
        }
    }
}
public void update(Graphics g) {
    //重新输出applet
    paint(g);
}
//利用clipRect()方法，每次调用显示方法paint()时，
//先用蓝色画一遍文字，再用白色在裁剪区中画一遍文字
public void paint(Graphics g) {
    //输出applet
    g.setFont(font);
    g.setColor(Color.blue);
    g.drawString(myText, w, baseLine); //第一遍显示
    g.clipRect(spotPosition, 0, myTextSize, getSize().height); //设置裁剪区域
    g.setColor(Color.white);
    g.drawString(myText, w, baseLine); //第二遍显示
    spotPosition = (spotPosition + 1) % (myTextWidth + 100); //移动聚光灯位置
}
}

```



案例10 伸展文字

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个自小变大的伸展文字特效，如图1-16、图1-17和图1-18所示。

图1-16 伸展文字特效（1）

伸展文字示例

图1-17 伸展文字特效（2）

伸展文字示例

图1-18 伸展文字特效（3）

制作要点

1. 类Graphics的使用
2. 类Font的使用
3. 类Runnable的使用
4. 双缓冲技术消除闪烁

步骤详解

1. 初始化Applet。

```
font = new Font("TimesRoman", Font.BOLD, 30);
width = this.size().width;
height = this.size().height;
myHeight = height / 3;
yPosition = myHeight;
myText = getParameter("words");           //读取页面参数
if (myText != null)
    words = myText;
myImage = createImage(width, height);
myGraphic = myImage.getGraphics();
```

2. 启动applet。

```
mythread = new Thread(this);
mythread.start();
```

3. 启动线程。

蓝色直线：

```
myGraphic.setColor(Color.blue);
for (int i = width; i >= 0; i--) {
    myGraphic.fillRect(i, height / 3, width, height / 10);
    repaint();
    mythread.sleep(10);
}
```

黄色直线：

```
myGraphic.setColor(Color.yellow);
for (int i = 0; i <= width; i++) {
    myGraphic.fillRect(0, height / 3, i, height / 10);
    repaint();
    mythread.sleep(10);
}
```

蓝底白字：

```
myGraphic.setColor(Color.blue);
myGraphic.fillRect(0, 0, width, height);
myGraphic.setFont(font);
myGraphic.setColor(Color.white);
myGraphic.drawString(words, 10, 35);
times++; //自增
```

黄底黑字：

```
myGraphic.setColor(Color.yellow);
myGraphic.fillRect(0, 0, width, height);
myGraphic.setFont(font);
myGraphic.setColor(Color.black);
myGraphic.drawString(words, 10, 35);
times = 0; //交替
```

4. 利用drawImage()方法输出Applet。

Graphics类的drawImage()方法用来显示Image对象。为了提高图像的显示效果，许多Applet都采用双缓冲技术：首先把图像装入内存，然后再显示在屏幕上。

```
g.drawImage(myImage, 0, yPosition, width, myHeight, this);
```

5. 调用applet的网页。

```
<HTML>
<HEAD>
<TITLE>
伸展文字
```

```

</TITLE>
</HEAD>
<BODY>
伸展文字示例<BR>
<applet code="ExtendWords.class" width=240 height=50 VIEWASTEXT>
<param name=words value="伸展文字示例">
</applet>
</BODY>
</HTML>

```

程序源代码与解释

```

/*
*使用Applet，在浏览器中显示一个自小变大的伸展文字特效
*/
import java.awt.*;
import java.applet.*;
public class ExtendWords extends Applet implements Runnable {
    private Image myImage;
    private Graphics myGraphic;
    private int width = 0, height = 0;
    private String myText, words;
    private Thread mythread;
    private int xPosition = 0, yPosition = 0, myHeight;
    private int times = 0;
    private Font font;
    public void init() {                                //初始化applet
        font = new Font("TimesRoman", Font.BOLD, 30); //设置字体
        width = this.size().width;
        height = this.size().height;
        myHeight = height / 3;
        yPosition = myHeight;
        myText = getParameter("words");                //读取页面参数
        if (myText != null)
            words = myText;
        myImage = createImage(width, height);
        myGraphic = myImage.getGraphics();
    }
    public void start() {                            //启动applet
        if (mythread == null) {
            mythread = new Thread(this);
            mythread.start();
        }
    }
}

```

```

        }

    public void update(Graphics g) {
        paint(g);
    }

    public void paint(Graphics g) {           //输出applet
        g.drawImage(myImage, 0, yPosition, width, myHeight, this);
    }

    public void run() {                     //启动线程
        try {
            while (true) {
                yPosition = 0;
                myHeight = height;
                myGraphic.setColor(Color.white);
                myGraphic.fillRect(0, 0, width, height);
                repaint();
                mythread.sleep(100);
                if (times == 0) {           //蓝色动态直线
                    myGraphic.setColor(Color.blue);
                    for (int i = width; i >= 0; i--) {
                        myGraphic.fillRect(i, height / 3, width, height / 10);
                        repaint();
                        mythread.sleep(10);
                    }
                } else if (times == 1) {      //黄色动态直线
                    myGraphic.setColor(Color.yellow);
                    for (int i = 0; i <= width; i++) {
                        myGraphic.fillRect(0, height / 3, i, height / 10);
                        repaint();
                        mythread.sleep(10);
                    }
                }
                yPosition = height / 3;
                myHeight = height / 3;
                for (int i = height / 3; i >= 0; i--) {
                    xPositon = 0;
                    yPosition--;
                    myHeight = myHeight + 2;
                    if (times == 0) {           //蓝底白字
                        myGraphic.setColor(Color.blue);
                        myGraphic.fillRect(0, 0, width, height);
                        myGraphic.setFont(font);
                        myGraphic.setColor(Color.white);
                        myGraphic.drawString(words, 10, 35);
                    }
                }
            }
        }
    }
}

```

```
        times++; //自增
    } else if (times == 1) { //黃底黑字
        myGraphic.setColor(Color.yellow);
        myGraphic.fillRect(0, 0, width, height);
        myGraphic.setFont(font);
        myGraphic.setColor(Color.black);
        myGraphic.drawString(words, 10, 35);
        times = 0; //交替
    }
    repaint();
    mythread.sleep(100);
}
mythread.sleep(2500);
}
} catch (InterruptedException e) {
}
}
```

本 章 小 结

一个完整的Java Applet包含4个方法：Init()、Start()、Stop()、Destroy()（即：初始化、开始、停止、清除），构成一个完整的生命周期，其运行次序也是由上而下顺序执行。它被部署在Web服务器（如IIS）上，当客户端请求Web页时，浏览器从Web服务器上将其下载到本地客户端；然后，浏览器创建该Applet类的实例并调用其init()方法。从安全角度考虑，Applet没有访问本地文件的权限。由于Applet是被浏览器执行的，所以Applet不需要一个main()方法。本章所选的案例利用Java Applet技术模拟网页功能和一些文字特效的实现，以及Applet之间的通信。一般来讲，编制Java Applet离不开4个方面的问题：（1）多线程的使用；（2）布局管理器的选择；（3）数据输入、输出流的应用；（4）设置、监听鼠标与键盘事件。在后面的章节中，我们还将会涉及到Applet的知识。

第2章

Java与特效



本章内容

- 案例1 火焰招牌
- 案例2 闪电中的城市
- 案例3 激光绘图
- 案例4 水面倒影
- 案例5 图片放大镜
- 案例6 浮动的气泡
- 案例7 烟花汇演
- 案例8 星空模拟
- 案例9 阴影跑马灯
- 案例10 下雪的图片
- 案例11 动态分割线
- 案例12 飞流直下
- 本章小结



案例1 火焰招牌

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中

会显示一个燃烧着的火焰招牌，如图2-1所示。



图2-1 火焰招牌特效

制作要点

1. 类Graphics的使用
2. 类MediaTracker的使用
3. 类Image的使用
4. Applet中支持线程的用法

步骤详解

1. 初始化Applet。

初始化Applet，加载背景图片、前景图片：

```
String imageName = getParameter("Image1");
String imageName2 = getParameter("Image2");
Bimg = getImage(getDocumentBase(),imageName);
```

获取背景图片、前景图片：

```
Fimg = getImage(getDocumentBase(),imageName2);
imageTracker = new MediaTracker(this);
```

创建一个媒体跟踪器的实例，为其添加背景图片和前景图片：

```
imageTracker.addImage(BackImage,0);
imageTracker.addImage(ForeImage,0);
```

加载图片：

```
imageTracker.waitForID(0);
```

2. 实现火焰招牌特效。

线程调用开始：

```
thread.sleep(0);
```

在不同的坐标位置输出图像以产生燃烧效果：

```
for(int j=0; j < virtualI.getHeight(this); j = j + tileHeight)
    for(int i=0; i < virtualI.getWidth(this); i = i + tileWidth)
```

```

        virtualG.drawImage(bimg, i, j, this);
        virtualG.drawImage(fimg, x, y, width, height, this);
        offG.drawImage(virtualI,-x,-y,this);
    
```

3. 调用applet的网页。

```

<applet code="FlameSign.class" width="319" height="134" VIEWASTEXT>
    <PARAM name="Image1" value="b.jpg">
    <PARAM name="Image2" value="f.gif">
</applet>

```

程序源代码与解析

```

import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
//实现Runnable接口，在applet中支持线程
public class FlameSign extends Applet implements Runnable
{
    private Image bimg, fimg, offI, virtualI;
    private Graphics offG, virtualG;
    private Thread thread = null;
    private MediaTracker imageTracker;
    private int height, width, X, Y;
    public void init() {
        //初始化applet，加载背景图片、前景图片
        String imageName = getParameter("Image1");
        String imageName2 = getParameter("Image2");
        bimg = getImage(getDocumentBase(), imageName);      //获取背景图片
        fimg = getImage(getDocumentBase(), imageName2);      //获取前景图片
        imageTracker = new MediaTracker(this);           //创建一个媒体跟踪器的实例
        //将图片加入到MediaTracker的监视队列中去，image为要被监视的图像对象,
        //0为监视图像在监视队列中的标识号
        imageTracker.addImage(bimg, 0);
        imageTracker.addImage(fimg, 0);
        width = this.getSize().width; //设置Applet宽度
        height = this.getSize().height; //设置Applet高度
        try {
            imageTracker.waitForID(0); //加载图片
        } catch (InterruptedException e) {
        }
        offI = createImage(width, height);
    }
}

```

```

offG = offI.getGraphics();
virtualI = createImage(width * 2, height * 2);
virtualG = virtualI.getGraphics();
}

public void start() {
    //启动Applet, 创建并启动一个线程
    if (thread == null) {
        thread = new Thread(this); //以applet初始化线程
        thread.start(); //启动线程, 调用run()方法
    }
}

public void run() {
    //线程调用开始
    int x = 0, y = 0;
    int tileWidth = bimg.getWidth(this); //设置招牌宽度
    int tileHeight = bimg.getHeight(this); //设置招牌高度
    while (thread != null) {
        try {
            Thread.sleep(10);
            x = virtualI.getWidth(this) - width;
            y = virtualI.getHeight(this) - height;
            //在不同的坐标位置输出图像以产生燃烧效果
            for (; (x > 0) && (y > 0); x--, y--) {
                if ((x == 0) || (y == 0)) {
                    x = virtualI.getWidth(this) - width;
                    y = virtualI.getHeight(this) - height;
                }
                //输出图像, 产生燃烧特效
                for (int j = 0; j < virtualI.getHeight(this); j = j
                     + tileHeight)
                    for (int i = 0; i < virtualI.getWidth(this); i = i
                         + tileWidth)
                        virtualG.drawImage(bimg, i, j, this);
                virtualG.drawImage(fimg, x, y, width, height, this);
                offG.drawImage(virtualI, -x, -y, this);
                //输出Applet, 调用update()方法
                repaint();
            }
        } catch (InterruptedException e) {
        }
    }
}

```

```

public void update(Graphics g) {           //调用paint()方法
    paint(g);
}
public void paint(Graphics g) {           //输出Applet
    g.drawImage(offI, 0, 0, this);
}
}

```



案例2 闪电中的城市

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个闪电中的城市如图2-2所示。



图2-2 闪电中的城市

制作要点

1. 类Graphics的使用
2. 类Image的使用

步骤详解

1. 初始化Applet。

初始化Applet，加载图片：

```

image = getImage(getCodeBase(),"City.gif");
buffer = this.createImage(getSize().width, getSize().height);

```

2. 实现闪电中的城市特效。

线程休眠，时间随机产生：

```
Thread.sleep((int)(Integer.parseInt(delay)*1000*Math.random()));
```

无闪电标记变量置位：

```
no_thunder = false;
```

初始化闪电数据:

```
createThunder();
```

输出图像:

```
drawBuffer();
g = this.getGraphics();
g.drawImage(buffer, 0, 0, this);
```

线程休眠1秒:

```
Thread.sleep(1000);
```

无闪电标记变量置位:

```
no_thunder = true;
```

3. 输出闪电以及城市的图像。

画出City.gif:

```
g.drawImage(image, 0, 0, this);
```

白色闪电:

```
g.setColor(whiteSky); //设置背景色为白色
```

黄色闪电:

```
g.setColor(yellowSky); //设置背景色为黄色
g.fillRect(0, 0, getSize().width, getSize().height); //填充背景色
```

输出闪电折线:

```
g.drawLine(light[i], i, light[i-1], i-1);
```

4. 生成闪电的坐标数组数据。

随机产生闪电出现的位置:

```
light[i] = light[i-1] + ((Math.random() > 0.5) ? 1 : -1);
b1[i] = light[i];
b2[i] = light[i];
```

5. 调用Applet的网页:

```
<applet code="Thunder.class" width="500" height="125">
</applet>
```

程序源代码与解答

```

import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;

//实现Runnable接口，在Applet中支持线程
public class CityInThunder extends Applet implements Runnable
{
    private Thread thread = null;                      //Applet支持的线程
    private boolean no_thunder = true;                  //没有闪电的标志变量
    private boolean thunder = true;                     //有闪电的标志变量
    private int[] light;
    private int[] b1;
    private int[] b2;
    private Color whiteSky = new Color(0,0,65), yellowSky = new Color(144,40,40);
    //白色闪电、黄色闪电
    private Image buffer, image;
    private String delay = "3";

    public void init()
    {
        //初始化Applet
        image = getImage(getCodeBase(),"City.gif");
        light = new int[getSize().height];
        b1 = new int[getSize().height];
        b2 = new int[getSize().height];
        buffer = this.createImage(getSize().width, getSize().height);
    }

    public void paint(Graphics g)
    {
        int i, thr;
        if (no_thunder)                                //没有闪电
        {
            g.setColor(Color.black);                   //设背景色为黑色
            g.fillRect(0 , 0 , getSize().width, getSize().height); //填充背景色
            g.drawImage(image,0,0,this);               //输出City.gif
        }
        else                                            //有闪电
        {
            if(thunder)                               //白色闪电

```

```

        g.setColor(whiteSky);           //设背景色为白色
    else
        g.setColor(yellowSky);        //黄色闪电
    g.fillRect(0, 0, getSize().width, getSize().height);      //填充背景色
    //输出闪电图像
    thr = (int) (0.8F * getSize().height);
    for (i = 1; i < getSize().height; i++)
    {
        if (i < thr)
        {   //输出闪电周围的灰色矩形
            g.setColor(Color.darkGray);
            g.drawRect(light[i]-4, i, 3, 1);
            g.drawRect(light[i]+2, i, 3, 1);
            g.setColor(Color.gray);
            g.drawRect(light[i]-1, i, 1, 1);
            g.drawRect(light[i]+1, i, 1, 1);
        }
        if(thunder)
        {   //白色闪电
            g.setColor(Color.white);
        }
        else //黄色闪电
        {
            g.setColor(Color.yellow);
            g.drawLine(light[i], i, light[i-1], i-1);
            if (b1[i] >= 0)
            {   //输出闪电折线
                g.drawLine(b1[i], i, b1[i-1], i-1);
            }
            if (b2[i] >= 0)
            {   //输出闪电折线
                g.drawLine(b2[i], i, b2[i-1], i-1);
            }
        }
        //输出城市图像city.gif
        g.drawImage(image,0,0,this);
        thunder = !thunder;
    }
}

void drawBuffer()
{
    //调用paint()方法
    Graphics g;
}

```

```

        g = buffer.getGraphics();
        paint(g);
    }

    public void start()
    {
        //启动applet, 创建并启动线程
        if (thread == null)
        {
            thread = new Thread(this);
            thread.start();
        }
    }

    public void stop()
    {
        if (thread != null)
        {
            thread.stop();
            thread = null;
        }
    }

    void createThunder()
    {
        //生成闪电的坐标数组数据
        int i;
        int bs1, bs2; //开始位置的坐标
        int be1, be2; //结束位置的坐标
        light[0] = (int) (Math.random() * getSize().width); //随机产生闪电出现的位置
        b1[0] = light[0];
        b2[0] = light[0];
        bs1 = (int) (Math.random() * getSize().height) + 1;
        bs2 = (int) (Math.random() * getSize().height) + 1;
        be1 = bs1 + (int) (0.5 * Math.random() * getSize().height) + 1;
        be2 = bs2 + (int) (0.5 * Math.random() * getSize().height) + 1;
        for (i = 1; i<getSize().height; i++)
        {
            light[i] = light[i-1] + ((Math.random() >0.5)?1:-1);
            b1[i] = light[i];
            b2[i] = light[i];
        }
        for (i = bs1; i<getSize().height; i++)
        {
    }

```

```

        b1[i] = b1[i-1] + ((Math.random() >0.5)?2:-2);
    }
    for (i = bs2; i<getSize().height; i++)
    {
        b2[i] = b2[i-1] + ((Math.random() >0.5)?2:-2);
    }
    for (i = be1; i<getSize().height; i++)
    {
        b1[i] = -1;
    }
    for (i = be2; i<getSize().height; i++)
    {
        b2[i] = -1;
    }
}
public void run()
{
    //启动进程
    Graphics g;
    while (true)
    {
        try
        {
            //输出图像
            drawBuffer();
            g = this.getGraphics();
            g.drawImage(buffer, 0, 0, this);
            //线程休眠，时间随机产生
            Thread.sleep((int) (Integer.parseInt(delay) * 1000 * Math.random()));
            //无闪电标记变量置位
            no_thunder = false;
            //创建闪电
            createThunder();
            //输出图像
            drawBuffer();
            g = this.getGraphics();
            g.drawImage(buffer, 0, 0, this);
            //线程休眠1秒
            Thread.sleep(1000);
            //无闪电标记变量置位
            no_thunder = true;
        }
    }
}

```

```

        catch (InterruptedException e)
        {
            stop(); //发生异常，停止线程
        }
    }
}

```



案例3 激光绘图

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个激光绘图的过程，如图2-3所示。

图2-3展现的是激光绘图尚未完全绘制完成的状态，整个特效是一个动态的绘制过程。



图2-3 激光绘图

制作要点

1. `java.util.Random`类的使用
2. 类`PixelGrabber`的使用
3. 类`MediaTracker`的使用

步骤详解

1. 随机数类Random的使用。

Java实用工具类库中的类`java.util.Random`提供了产生各种类型随机数的方法。它可以产生`int`、`long`、`float`、`double`以及Gaussian等类型的随机数。这也是它与`java.lang.Math`中的方法`Random()`的最大不同之处，后者只产生`double`型的随机数。

类`Random`中的方法十分简单，它只有两个构造方法和6个普通方法。

构造方法：

- (1) `public Random()`
- (2) `public Random(long seed)`

Java产生随机数需要有一个基值`seed`，在第一种方法中默认的基值是将系统时间作为`seed`。

普通方法：

- (1) `public synchronized void setSeed(long seed)`

该方法是设定基值`seed`。

- (2) `public int nextInt()`

该方法是产生一个整型随机数。

(3) public long nextLong()

该方法是产生一个long型随机数。

(4) public float nextFloat()

该方法是产生一个float型随机数。

(5) public double nextDouble()

该方法是产生一个double型随机数。

(6) public synchronized double nextGaussian()

该方法是产生一个double型的Gaussian随机数。

例如，RandomApp.java的代码如下

```
import java.lang.*;
import java.util.Random;

public class RandomApp{
    public static void main(String args[]){
        Random ran1=new Random();
        Random ran2=new Random(12345);
        //创建了两个类Random的对象。
        System.out.println("The 1st set of random numbers:");
        System.out.println("\t Integer:"+ran1.nextInt());
        System.out.println("\t Long:"+ran1.nextLong());
        System.out.println("\t Float:"+ran1.nextFloat());
        System.out.println("\t Double:"+ran1.nextDouble());
        System.out.println("\t Gaussian:"+ran1.nextGaussian());
        //产生各种类型的随机数
        System.out.print("The 2nd set of random numbers:");
        for(int i=0;i<5;i++){
            System.out.println(ran2.nextInt()+" ");
            if(i==2) System.out.println();
            //产生同种类型的不同的随机数。
            System.out.println();
        }
    }
}
```

运行结果：

```
E:\java RandomApp
The 1st set of random numbers:
    Integer:-173899656
    Long:8056223819738127077
    Float:0.6293638
```

Double:0.7888394520265607

Gaussian:0.5015701094568733

The 2nd set of random numbers:1553932502

-2090749135

-287790814

-355989640

-716867186

2. 类PixelGrabber的使用参见第1章案例5。

3. 调用Applet的网页。

```
<APPLET code="LaserPic.class" width="250" height="250">
<PARAM name="Image" value="cafe.gif">
</APPLET>
```

程序源代码与解释

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Point;
import java.awt.image.PixelGrabber;
import java.util.Random;

public class LaserPic extends Applet
{
    private Image image;
    private int x = 25,y = 25;
    private Random random;
    private int width,height,w,h,image_size,pixels[];

    public void init()
    {
        //初始化Applet
        random=new Random();
        //从网页获取图片文件名参数
        String imageName=getParameter("Image");
        //加载图片
        image=getImage(getDocumentBase(),imageName);
        MediaTracker imageTracker=new MediaTracker(this);
        imageTracker.addImage(image,0);
```

```

try
{
    imageTracker.waitForID(0);
}
catch (InterruptedException e)
{
}
}

public void start()
{
    //启动Applet,然后调用paint()方法

    width=getSize().width;           //Applet宽度
    height=getSize().height;         //Applet高度
    w=image.getWidth(this);          //图片宽度
    h=image.getHeight(this);         //图片高度
    //图片输出位置
    x = (width - w)/2;
    y = (height - h)/2;
    //图片大小
    image_size=w*h;
    //创建图片的像素数组
    pixels=new int[image_size];
    //创建一个像素获取器的实例，并将其与像素数组关联
    PixelGrabber pg=new PixelGrabber(image,0,0,w,h,pixels,0,w);
    try
    {
        //解析图片的像素信息
        pg.grabPixels();
    }
    catch (InterruptedException e)
    {
    }
}
}

public void paint(Graphics g)
{
    g.setColor(Color.white);
    g.fillRect(0,0,getSize().width,getSize().height);
    //调用drawImage()方法，在相应的位置输出图片
    drawImage(g,image,x,y);
}

private void drawImage(Graphics g,Image image,int x,int y)

```

```

{
    //输出图片
    while(true)
    {
        g.setColor(Color.white);
        g.fillRect(0,0,getSize().width,getSize().height);
        try
        {
            int one_time=w;                                //图片宽度
            int S_x=0,S_y=0;
            S_x=(int)(random.nextFloat()*width);
            S_y=(int)(random.nextFloat()*height);
            Laser[] nextlot=new Laser[one_time];
            int k=0,l=0;
            int step=1,start=0;
            float f=random.nextFloat();
            step=(f<0.8)?34759:(f<0.9?1:image_size-1);      //步长
            //start=(int)(random.nextFloat()*image_size);      /起始位置
            f=random.nextFloat();
            start=(f<0.5)?image_size:0;
            //如果f小于0.5,则起始位置为图片大小,否则为0
            intsofar=0;
            //初始化nextlot数组
            for (k=start;l<image_size;l++,k+=step)
            {
                Thread.sleep(2);
                if (k<0) k+=image_size;
                k%=image_size;
                int row=k/w;
                int col=k%w;
                Color colr=new Color(pixels[k]);
                int finishx=x+col;
                int y1=y+row;
                nextlot[sofar]=new Laser(colr,new Point(S_x,S_y),
                new Point(finishx,y1));
               sofar++;
                if (sofar==one_time)
                {
                    Track(g,nextlot);
                   sofar=0;
                }
            }
        }
    }
}

```

```

        }
        catch (Exception e){}
        g.setPaintMode();
        g.drawImage(image,x,y,this);
        try
        {
            Thread.sleep(2000);
        }
        catch (InterruptedException e){}
    }
}

private synchronized void Track(Graphics g,Laser[] nextlot)
{
    Color back=Color.white;
    g.setXORMode(back);
    for (int pass=0;pass<2;pass++)
    {
        for (int pixnr=0;pixnr<nextlot.length;pixnr++)
        {
            Laser p=nextlot[pixnr];
            if (!close(p.c,back))
            {
                g.setColor(p.c);
                g.drawLine(p.start.x,p.start.y,p.finish.x,p.finish.y);
            }
            if (pass==1)
            {
                g.setColor(p.c);
                g.drawLine(p.finish.x,p.finish.y,p.finish.x,p.finish.y);
            }
        }
        Thread.yield();
    }
}

private boolean close(Color c1,Color c2)
{
    return (Math.abs(c1.getRed()-c2.getRed()) + Math.abs(c1.getGreen()-c2.getGreen()) +
           Math.abs(c1.getBlue()-c2.getBlue()))<0xff;
}

class Laser
{
}

```

```

public Color c;
public Point start,finish;
public Laser(Color c,Point start,Point finish)
{
    this.c=c;
    this.start=start;
    this.finish=finish;
}
}

```



案例4 水面倒影

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该applet的网页，浏览器中会显示一个水面倒影的特效，如图2-4所示。

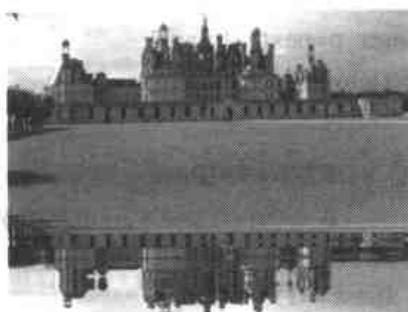


图2-4 水面倒影特效

制作要点

1. 类Math的使用
2. 类MediaTracker的使用
3. 类Graphics的使用
4. 类Image的使用

步骤详解

1. 初始化Applet。
从HTML文件中获取图片文件名：

```

param = getParameter("image");
if (param != null)

```

```
name = param;
```

2. 启动线程。

加载图片：

```
currentImg = 0;
g = getGraphics();
MediaTracker imageTracker = new MediaTracker(this);
String strImage;
image = getImage(getDocumentBase(), name);
imageTracker.addImage(image, 0);
```

输出图片倒影：

```
if (refimage != null) {
    g.drawImage(refimage, (-currentImg * imageW), imageH, this);
    g.drawImage(refimage, ((frames - currentImg) * imageW), imageH, this);
```

输出正向图片：

```
g.drawImage(image, 0, -1, this);
```

3. 实现水波特效。

```
refimage = createImage((frames + 1) * imageW, imageH);
refraction = refimage.getGraphics();
refraction.drawImage(back, frames * imageW, 0, this);
for (phase = 0; phase < frames; phase++) {
    p1 = 2 * Math.PI * (double) phase / (double) frames;
    x = (frames - phase) * imageW;
    for (int i = 0; i < imageH; i++) {
        y = (int) ((imageH / 14)
                    * ((double) i + 28.0)
                    * Math.sin((double) ((imageH / 14) * (imageH - i))
                               / (double) (i + 1) + p1) / (double) imageH);
        if (i < -y)
            refraction.copyArea(frames * imageW, i, imageW, 1, -x, 0);
        else
            refraction.copyArea(frames * imageW, i + y, imageW, 1, -x, -y);
```

4. 调用Applet的网页。

```
<applet code=Ripple.class width=420 height=330>
    <param name=image value="sunset.jpg">
</applet>
```

程序源代码与解析

```

import java.applet.*;
import java.awt.*;

public class Ripple extends Applet implements Runnable {
    Thread thread = null;
    private Graphics g, refraction;
    private Image image, refimage;
    private int currentImg;
    private int imageW = 0, imageH = 0;
    private int ovalW = 0, ovalH = 0;
    private boolean finishLoad = false;
    private final int frames = 12;
    private String name = "";

    public void init() {
        //初始化Applet
        String param;
        //从HTML文件中获取图片文件名
        param = getParameter("image");
        if (param != null)
            name = param;
    }

    public void paint(Graphics g) {
        //如果已经载入图片，则返回
        if (!finishLoad)
            return;
        //输出倒影图片
        if (refimage != null) {
            g.drawImage(refimage, (-currentImg * imageW), imageH, this);
            g.drawImage(refimage, ((frames - currentImg) * imageW), imageH, this);
        }
        //输出正向图片
        g.drawImage(image, 0, -1, this);
    }

    public void start() {
        //启动Applet，创建并启动线程
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }
}

```

```

        }
    }

    public void run() {
        //启动线程
        //加载图片
        currentImg = 0;
        g = getGraphics();
        MediaTracker imageTracker = new MediaTracker(this);
        String strImage;
        image = getImage(getDocumentBase(), name);
        imageTracker.addImage(image, 0);
        try {
            imageTracker.waitForAll();
            finishLoad = !imageTracker.isErrorAny();
        } catch (InterruptedException e) {
        }
        //图片宽度、图片高度
        imageW = image.getWidth(this);
        imageH = image.getHeight(this);
        //生成倒影
        createRipple();
        //重新输出Applet
        repaint();
        while (true) {
            try {
                if (!finishLoad)
                    return;
                if (refimage != null) {
                    g.drawImage(refimage, (-currentImg * imageW), imageH, this);
                    g.drawImage(refimage, ((frames - currentImg) * imageW)
                               , imageH, this);
                }
                g.drawImage(image, 0, -1, this);
                if (++currentImg == frames)
                    currentImg = 0;
                Thread.sleep(50);
            } catch (InterruptedException e) {
                stop();
            }
        }
    }
}

```

```

    }

    public void createRipple() {
        //产生水波特效

        Image back = createImage(imageW, imageH + 1);
        Graphics offg = back.getGraphics();
        int phase = 0;
        int x, y;
        double p1;
        offg.drawImage(image, 0, 1, this);
        for (int i = 0; i < (imageH >> 1); i++) {
            offg.copyArea(0, i, imageW, 1, 0, imageH - i);
            offg.copyArea(0, imageH - 1 - i, imageW, 1, 0, -imageH + 1 + (i << 1));
            offg.copyArea(0, imageH, imageW, 1, 0, -1 - i);
        }
        refimage = createImage((frames + 1) * imageW, imageH);
        refraction = refimage.getGraphics();
        refraction.drawImage(back, frames * imageW, 0, this);

        for (phase = 0; phase < frames; phase++) {
            p1 = 2 * Math.PI * (double) phase / (double) frames;
            x = (frames - phase) * imageW;
            for (int i = 0; i < imageH; i++) {
                y = (int) ((imageH / 14)
                            * ((double) i + 28.0)
                            * Math.sin((double) ((imageH / 14) * (imageH - i))
                                       / (double) (i + 1) + p1) / (double) imageH);
                if (i < -y)
                    refraction.copyArea(frames * imageW, i, imageW, 1, -x, 0);
                else
                    refraction.copyArea(frames * imageW, i + y, imageW, 1, -x, -y);
            }
        }
        offg.drawImage(image, 0, 1, this);
        image = back;
    }
}

```



案例5 图片放大镜

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个图片放大镜的特效，随着放大镜的移动，放大框内的图像被放大，如图2-5所示。

Chr	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
●	0	00		NUL	32	20	空格	64	40	A	96	60
^A	1	01		SOH	33	21	!	65	41	B	97	61
^B	2	02		STX	34	22	“	66	42	C	98	62
^C	3	03		ETX	35	23	”	67	43	D	99	63
^D	4	04		ENQ	36	24	◆	68	44	E	100	64
^E	5	05		ACK	37	25	◆	69	45	F	101	65
^F	6	06		SYN	38	26	◆	70	46	G	102	66
^G	7	07		BBEL	39	27	◆	71	47	H	103	67
^H	8	08		BS	40	28	◆	72	48	I	104	68
^I	9	09		HT	41	29	◆	73	49	J	105	69
^J	10	0A		LF	42	2A	◆	74	4A	K	106	6A
^K	11	0B		VIS	43	2B	◆	75	4B	L	107	6B
^L	12	0C		FF	44	2C	◆	76	4C	M	108	6C
^M	13	0D		CR	45	2D	◆	77	4D	N	109	6D
^N	14	0E		SG	46	2E	◆	78	4E	O	110	6E
^O	15	0F		SP	47	2F	◆	79	4F	P	111	6F
^P	16	10		DEL	48	30	◆	80	50	Q	112	70
^Q	17	11		DC1	49	31	◆	81	51	R	113	71
^R	18	12		DC2	50	32	◆	82	52	S	114	72
^S	19	13		DC3	51	33	◆	83	53	T	115	73
^T	20	14		DC4	52	34	◆	84	54	U	116	74
^U	21	15		NAK	53	35	◆	85	55	◆	117	75

图2-5 图片放大镜特效

制作要点

1. 鼠标移动事件
2. 双缓冲技术处理图像
3. 类MediaTracker的使用

步骤详解

1. Mouse移动事件。

鼠标移动主要通过接口MouseMotionListener来实现：

mouseDragged() 当用户按下鼠标按钮，并在松开之前进行移动时发生。在mouseDragged()后松开鼠标不会导致mouseClicked()。

mouseMoved() 当鼠标在组件上移动而不是拖动时发生。

我们在这要使用到MouseListener接口。实现接口后，要在init()函数中加入监听器addMouseListener()，来监听对Applet的响应事件。

通过鼠标位置设置放大镜的位置：

```
redrawGlass(boxX, boxY, e.getX(), e.getY());
```

设置放大镜的当前位置：

```
boxX = e.getX();
boxY = e.getY();
```

若放大镜溢出Applet，则进行调整：

```
if (boxX > (width - boxW / 2))
    boxX = width - boxW / 2;
if (boxY > (height - boxH / 2))
    boxY = height - boxH / 2;
```

输出放大镜：

```
drawGlass();
```

2. 初始化Applet。

添加鼠标事件监听：

```
addMouseMotionListener(this);
```

3. 实现放大镜特效。

```
Graphics temp;
```

复制g的一个实例：

```
temp = g.create();
```

为temp限制一个矩形区域：

```
temp.clipRect(boxX, boxY, boxW, boxH);
```

输出放大的图像：

```
temp.drawImage(back, -boxX, -boxY, width * 2, height * 2, null);
```

输出放大镜边框：

```
g.setColor(Color.white);
g.drawRect(boxX, boxY, boxW - 1, boxH - 1);
```

4. 重画放大镜。

```
temp.clipRect(newX, newY + boxH, boxW + oldX - newX, oldY - newY);
temp.drawImage(offi, 0, 0, null);
temp = g.create();
temp.clipRect(newX + boxW, newY, oldX - newX, boxH + oldY - newY);
temp.drawImage(offi, 0, 0, null);
```

5. 调用Applet的网页:

```
<applet code="Bigger.class" width=500 height= 400 VIEWASTEXT>
<param name="picture_name" value=beautiful.jpg>
</applet>
```

程序源代码与解释

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;

public class Magnifier extends Applet implements MouseMotionListener {
    Graphics g;
    Image offi;
    Image back;
    String picture_name;
    MediaTracker imageTracker;
    int boxX = 0, boxY = 0;           //放大镜初始位置
    int boxW = 100, boxH = 100;      //放大镜宽度、高度
    int width, height;
    public void init() {
        //初始化Applet
        //加载图片
        g = getGraphics();
        picture_name = new String();
        picture_name = getParameter("picture_name");
        if(picture_name == null)
            picture_name = "beautiful.jpg";
        imageTracker = new MediaTracker(this);
        back = getImage(getCodeBase(), picture_name);
        //设置放大后的图像的大小
        offi = createImage(300,200);
        Graphics offg = offi.getGraphics();
        offg.drawImage(back, 0, 0, this);
        //添加鼠标事件监听
        addMouseMotionListener(this);
    }
}
```

```

public void mouseDragged(MouseEvent e) {
    //鼠标拖拽事件处理
}
public void mouseMoved(MouseEvent e) {
    //处理鼠标移动事件
    //通过鼠标位置设置放大镜的位置
    redrawGlass(boxX, boxY, e.getX(), e.getY());
    //设置放大镜的当前位置
    boxX = e.getX();
    boxY = e.getY();
    //若放大镜溢出Applet，则进行调整
    if (boxX > (width - boxW / 2))
        boxX = width - boxW / 2;
    if (boxY > (height - boxH / 2))
        boxY = height - boxH / 2;
    //输出放大镜
    drawGlass();
}
void drawGlass() {
    Graphics temp;
    //复制g的一个实例
    temp = g.create();
    //为temp限制一个矩形区域
    temp.clipRect(boxX, boxY, boxW, boxH);
    //输出放大后的图像
    temp.drawImage(back, -boxX, -boxY, width * 2, height * 2, null);
    //输出放大镜边框
    g.setColor(Color.white);
    g.drawRect(boxX, boxY, boxW - 1, boxH - 1);
}
void redrawGlass(int oldX, int oldY, int newX, int newY) {
    //清除已经画过的矩形框和放大的图像
    Graphics temp;
    temp = g.create();
    if (newX <= oldX && newY <= oldY) {
        temp.clipRect(newX, newY + boxH, boxW + oldX - newX, oldY - newY);
        temp.drawImage(offi, 0, 0, null);
        temp = g.create();
        temp.clipRect(newX + boxW, newY, oldX - newX, boxH + oldY - newY);
        temp.drawImage(offi, 0, 0, null);
    }
}

```

```

        } else if (newX > oldX && newY <= oldY) {
            temp.clipRect(oldX, newY + boxH, boxW + newX - oldX, oldY - newY);
            temp.drawImage(offi, 0, 0, null);
            temp = g.create();
            temp.clipRect(oldX, newY, newX - oldX, boxH + oldY - newY);
            temp.drawImage(offi, 0, 0, null);
        } else if (newX > oldX && newY > oldY) {
            temp.clipRect(oldX, oldY, boxW + newX - oldX, newY - oldY);
            temp.drawImage(offi, 0, 0, null);
            temp = g.create();
            temp.clipRect(oldX, oldY, newX - oldX, boxH + newY - oldY);
            temp.drawImage(offi, 0, 0, null);
        } else {
            temp.clipRect(newX, oldY, boxW + oldX - newX, newY - oldY);
            temp.drawImage(offi, 0, 0, null);
            temp = g.create();
            temp.clipRect(newX + boxW, oldY, oldX - newX, boxH + newY - oldY);
            temp.drawImage(offi, 0, 0, null);
        }
    }

    public boolean imageUpdate(Image img, int flags, int x, int y, int w, int h) {
        if (flags == ALLBITS) {
            width = back.getWidth(this);
            height = back.getHeight(this);
            offi = createImage(width + boxW / 2, height + boxH / 2);
            Graphics offg = offi.getGraphics();
            offg.setColor(Color.lightGray);
            offg.fillRect(0, 0, width + boxW / 2, height + boxH / 2);
            offg.drawImage(back, 0, 0, this);
            repaint();
            return false;
        } else
            return true;
    }

    public void paint(Graphics g) {
        //输出背景图片
        g.drawImage(back, 0, 0, this);
        //画放大镜
        drawGlass();
    }
}

```



案例6 浮动的气泡

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个浮动的气泡的特效，如图2-6所示。

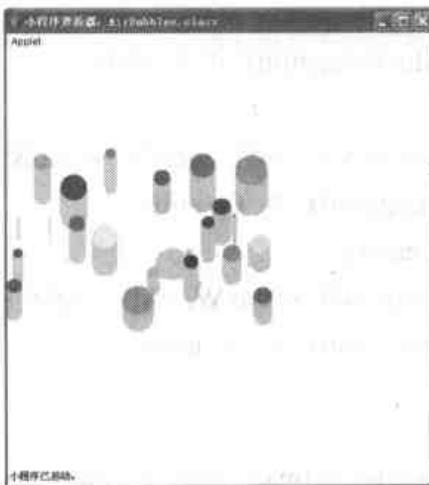


图2-6 浮动的气泡的特效

制作要点

类Graphics的使用

步骤详解

1. Graphics类的主要方法及用途，如表2-1所示。
2. 初始化、输出Applet。
3. 输出气泡。

输出一个圆形：

```
for (i = x - r; i <= x + r; i++) {
    g.setColor(col);
    g.drawLine(i, y - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),
               i, y + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
```

表2-1 Graphics类的主要方法及用途

方法	用途
<code>drawLine(int x1, int y1, int x2, int y2)</code>	从坐标(x1,y1)到(x2,y2)画一条直线，返回值类型为void
<code>drawOval(int x, int y, int width, int height)</code>	以(x,y)为左上角，画一个宽width、高height的椭圆，返回值类型为void
<code>drawPolygon(int[] xPoints, int[] yPoints, int nPoints)</code>	以xPoints的项为各节点的x坐标，yPoints的项为各节点的y坐标画多边形。节点数为nPoints，返回值类型为void
<code>drawPolygon(Polygon p)</code>	画多边形。多边形的信息由p来描述，返回值类型为void
<code>drawPolyline(int[] xPoints, int[] yPoints, int nPoints)</code>	以xPoints的项为各节点的x坐标，yPoints的项为各节点的y坐标画相互连接的多条线。节点数为nPoints，返回值类型为void
<code>drawRect(int x, int y, int width, int height)</code>	画以坐标(x,y)为左上角、宽为width、高为height的矩形。返回值类型为void
<code>drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	画以坐标(x,y)为左上角、宽为width、高为height的矩形，矩形的角是圆弧，圆弧的宽为arcWidth，高为arcHeight。返回值类型为void
<code>drawString(AttributedCharacterIterator iterator, int x, int y)</code>	在坐标(x,y)处画出字符串。字符串的值和属性在iterator里定义。返回值类型为void
<code>drawString(String str, int x, int y)</code>	在坐标(x,y)处画出字符串str。返回值类型为void
<code>fill3DRect(int x, int y, int width, int height, boolean raised)</code>	填充以坐标(x,y)为左上角、宽为width、高为height的三维矩形。若raised为true，则矩形是突出显示的，否则为凹下去的矩形。返回值类型为void
<code>fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)</code>	填充以坐标(x,y)为左上角、宽为width、高为height、弧度为arcAngle的圆弧区域。返回值类型为void
<code>fillOval(int x, int y, int width, int height)</code>	填充以(x,y)为左上角、宽为width、高为height的椭圆区域，返回值类型为void
<code>fillPolygon(int[] xPoints, int[] yPoints, int nPoints)</code>	填充以xPoints的项为各节点的x坐标，yPoints的项为各节点的y坐标的多边形区域。多边形的节点数为nPoints。返回值类型为void
<code>fillPolygon(Polygon p)</code>	填充多边形区域。多边形的信息由p来描述，返回值类型为void
<code>fillRect(int x, int y, int width, int height)</code>	填充以坐标(x,y)为左上角、宽为width、高为height的矩形区域。返回值类型为void

方法	用途
fillRoundRect(int x, int y, int width, int height,int arcWidth, int arcHeight)	填充以坐标(x,y)为左上角、宽为width、高为height的矩形区域，矩形的角是圆弧，圆弧的宽为arcWidth，高为arcHeight。返回值类型为void
getColor()	得到图的当前颜色。返回值类型为Color
getFont()	得到图的当前字体。返回值类型为Font
setColor(Color c)	设置图的当前颜色。返回值类型为void
setFont(Font font)	设置图的当前字体。返回值类型为void

4. 移动气泡。

输出气泡的上半部分：

```
for (i = x - r; i <= x + r; i++) {
    g.setColor(col);
    g.drawLine(i, y - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))), 
               i, y + step
               - (int) (Math.sqrt(r * r - ((i - x) * (i - x))));
```

输出气泡的下半部分：

```
for (i = x - r; i <= x + r; i++) {
    g.setColor(Color.lightGray);
    g.drawLine(i, y + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))), 
               i, y + step
               + (int) (Math.sqrt(r * r - ((i - x) * (i - x))));
```

5. 调用Applet的网页。

```
<applet
  code= AirBubbles.class
  name=Bubbles
  width=300
  height=300  VIEWASTEXT>
  <param name=Speed value="2">
  <param name=MaxBubbles value="10">
</applet>
```

程序源代码与解析

```
import java.applet.Applet;
import java.awt.Color;
```

```

import java.awt.Graphics;

public class AirBubbles extends java.applet.Applet implements Runnable {
    Thread artist = null;
    int bubble = 0, thisbubble = 0;           //气泡数量，当前气泡编号
    int MAXBUBBLES = 25;                     //最大气泡数量
    int stepper = 4;                         //气泡移动计数
    int record[][] = new int[MAXBUBBLES][5]; //记录气泡的二维数组
    public void init() {
        //初始化Applet
        //设定Applet尺寸
        resize(500, 500);
    }
    public void draw_bubble(int x, int y, int r, Color col, Graphics g) {
        //输出气泡
        int i;
        //输出一个圆形
        for (i = x - r; i <= x + r; i++) {
            g.setColor(col);
            g.drawLine(i, y - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),  

                      i, y + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
        }
    }
    public void move_bubble(int x, int y, int r, Color col, int step, Graphics g) {
        //移动气泡
        int i;
        //输出气泡的上半部分
        for (i = x - r; i <= x + r; i++) {
            g.setColor(col);
            g.drawLine(i, y - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),  

                      i, y + step  

                      - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
        }
        //输出气泡的下半部分
        for (i = x - r; i <= x + r; i++) {
            g.setColor(Color.lightGray);
            g.drawLine(i, y + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),  

                      i, y + step  

                      + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
        }
    }
    public void paint(Graphics g) {
        int i, j, tmp;

```

```

        if (bubble < MAXBUBBLES || thisbubble < MAXBUBBLES) {
            record[thisbubble][0] = (int) (Math.random() * 300);
            record[thisbubble][1] = 320;
            record[thisbubble][2] = (int) (Math.random() * 400) / 20;
            record[thisbubble][3] = (int) (Math.random() * 255);
            record[thisbubble][4] = (int) (Math.random() * 255);
            //输出气泡
            draw_bubble(record[thisbubble][0], record[thisbubble][1], record[thisbubble][2],
                        new java.awt.Color(record[thisbubble][3], record
                [thisbubble][4], 255), g);
            //如气泡数小于最大值，则总气泡数自增1，当前气泡编号自增1
            if (bubble < MAXBUBBLES) {
                bubble++;
                thisbubble++;
            } else //气泡数等于最大值
                thisbubble = MAXBUBBLES;
        }
        for (i = 0; i < bubble; i++) {
            if (i % 5 <= stepper) {
                record[i][1] -= 1;
                //移动气泡
                move_bubble(record[i][0], record[i][1], record[i][2],
                            new java.awt.Color(record[i][3], record[i][4], 255), 1,
                            g);
                for (j = 0; j < i; j++) {
                    tmp = ((record[i][1] - record[j][1])
                            * (record[i][1] - record[j][1]) + (record[i][0] -
                            record[j][0]) * (record[i][0] - record[j][0]));
                    if (j != i && Math.sqrt(tmp) < record[i][2] + record[j][2]) {
                        for (tmp = record[i][2]; tmp >= -1; tmp = tmp - 2)
                            draw_bubble(record[i][0], record[i][1],
                                        record[i][2] - tmp, Color.lightGray, g);
                        draw_bubble(record[j][0], record[j][1], record[j][2],
                                    new java.awt.Color(record[j][3], record[j][4],
                            255), g);
                        record[i][1] = -1;
                        record[i][2] = 0;
                    }
                }
            }
            if (record[i][1] + record[i][2] < 0 && bubble >= MAXBUBBLES) {
                thisbubble = i;

```

```

        }
        stepper = (int) (Math.random() * 10);
    }
}

public void update(Graphics g) {
    paint(g);
}

public void start() {
    //启动Applet，创建并启动线程
    if (artist == null) {
        artist = new Thread(this);
        artist.start();
    }
}

public void stop() {
    //结束Applet
    artist = null;
}

public void run() {
    //启动线程
    while (artist != null) {
        try {
            Thread.sleep(200);
        } catch (InterruptedException e) {
        }
        repaint();
    }
    artist = null;
}
}

```



案例7 烟花汇演

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个烟花汇演的特效，如图2-7所示。

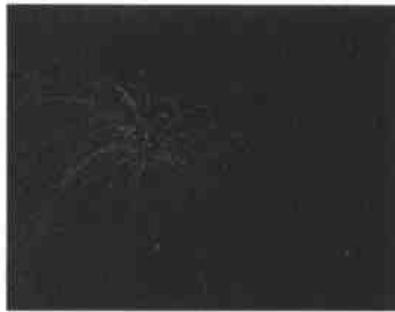


图2-7 烟花汇演特效

制作要点

1. 类Image的使用
2. 类Graphics的使用
3. 双缓冲技术处理闪烁

步骤详解

1. Rocket类。

随机生成烟花的颜色:

```
Color color;
c = (int) (random.nextDouble() * 64) - 32 + red;
```

输出烟花:

```
g.drawLine(ox + x, oy - y, ox + x, oy - y);
```

2. 调用Applet的网页。

```
<APPLET CODE=" Beacon.class" WIDTH=500 HEIGHT=400>
<PARAM NAME="AnimationSpeed" VALUE=5>
<Param NAME="RocketSoundtrack" VALUE=fire.au>
<Param NAME="RocketStyleVariability" VALUE=10>
<Param NAME="MaxRocketNumber" VALUE=100>
<Param NAME="MaxRocketExplosionEnergy" VALUE=900>
<Param NAME="MaxRocketPatchNumber" VALUE=200>
<Param NAME="MaxRocketPatchLength" VALUE=200>
<Param NAME="Gravity" VALUE=800>
</applet>
```

程序源代码与解释

```
import java.applet.Applet;
import java.awt.Color;
```

```

import java.awt.Graphics;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Random;

public class Beacon extends Applet implements Runnable
{
    public int AnimationSpeed, //烟花绽放的速度
        MissileStyleVariability,
        MaxMissileNumber,
        MaxMissileExplosionEnergy,
        MaxMissilePatchNumber,
        MaxMissilePatchLength,
        Gravity;
    public String MissileSoundtrack;
    private int mx, my;
    private Thread thread = null;
    private Missile Missile[];
    public void init()
    {
        //初始化Applet
        int i;
        //读取网页中参数
        String p;
        p = getParameter("AnimationSpeed");
        AnimationSpeed = (p == null) ? 100 : Integer.valueOf(p).intValue();

        MissileSoundtrack = getParameter("MissileSoundtrack");
        p = getParameter("MissileStyleVariability");
        MissileStyleVariability = (p == null) ? 20 : Integer.valueOf(p)
            .intValue();
        p = getParameter("MaxMissileNumber");
        MaxMissileNumber = (p == null) ? 5 : Integer.valueOf(p).intValue();
        p = getParameter("MaxMissileExplosionEnergy");
        MaxMissileExplosionEnergy = (p == null) ? 500 : Integer.valueOf(p)
            .intValue();
        p = getParameter("MaxMissilePatchNumber");
        MaxMissilePatchNumber = (p == null) ? 50 : Integer.valueOf(p).intValue();
        p = getParameter("MaxMissilePatchLength");
        MaxMissilePatchLength = (p == null) ? 100 : Integer.valueOf(p)
            .intValue();
        p = getParameter("Gravity");
        Gravity = (p == null) ? 20 : Integer.valueOf(p).intValue();
        mx = size().width - 1;
        my = size().height - 1;
    }
}

```

```
//初始化Missile数组
Missile = new Missile[MaxMissileNumber];
for (i = 0; i < MaxMissileNumber; i++)
    Missle[i] = new Missile(mx, my, Gravity);
}

public void start()
{//启动Applet
    if (thread == null)
    {
        thread = new Thread(this);
        thread.start();
    }
}

public void stop()
{//停止Applet
    if (thread != null)
    {
        thread.stop();
        thread = null;
    }
}

public void run()
{//启动线程
    int i,
    e = (int) (Math.random() * MaxMissileExplosionEnergy * 3 / 4) +
    MaxMissileExplosionEnergy / 4 + 1,
    p = (int) (Math.random() * MaxMissilePatchNumber * 3 / 4) +
    MaxMissilePatchNumber / 4 + 1,
    l = (int) (Math.random() * MaxMissilePatchLength * 3 / 4) +
    MaxMissilePatchLength / 4 + 1;//配置烟花参数
    long s = (long) (Math.random() * 10000);
    boolean sleep;
    Graphics g = getGraphics();
    URL u = null;
    while (true)
    {
        try {
            thread.sleep(100 / AnimationSpeed);
        }
        catch (InterruptedException x) {
        }
        sleep = true;
```

```

        for (i = 0; i < MaxMissileNumber; i++)
            sleep = sleep && Missile[i].sleep;
        if (sleep && Math.random() * 100 < MissileStyleVariability)
        {
            e = (int) (Math.random() * MaxMissileExplosionEnergy * 3 / 4) +
                MaxMissileExplosionEnergy / 4 + 1;
            p = (int) (Math.random() * MaxMissilePatchNumber * 3 / 4) +
                MaxMissilePatchNumber / 4 + 1;
            l = (int) (Math.random() * MaxMissilePatchLength * 3 / 4) +
                MaxMissilePatchLength / 4 + 1;
            s = (long) (Math.random() * 10000);
        }
        for (i = 0; i < MaxMissileNumber; i++)
        {
            if (Missile[i].sleep && Math.random() * MaxMissileNumber * 1 < 1)
            {
                try {
                    u = new URL(getDocumentBase(), MissileSoundtrack);
                }
                catch (MalformedURLException x) {
                }
                play(u);
                Missile[i].init(e, p, l, s);
                Missile[i].start();
            }
            Missile[i].show(g); //输出烟花
        }
    }
    public void paint(Graphics g)
    {
        g.setColor(Color.black);
        g.fillRect(0, 0, mx + 1, my + 1);
    }
}
class Missile
{
    public boolean sleep = true;
    private int energy, patch, length,
    mx, my,
    gravity,
    ox, oy,

```

```

    vx[], vy[],
    x, y,
    red, blue, green,
    t;
    private Random random;
    //构造函数
    public Missile(int a, int b, int g)
    {
        mx = a;
        my = b;
        gravity = g;
    }
    //初始化
    public void init(int e, int p, int l, long seed)
    {
        int i;
        energy = e;
        patch = p;
        length = l;
        random = new Random(seed);
        vx = new int[patch];
        vy = new int[patch];
        red = (int) (random.nextDouble() * 128) + 128;
        blue = (int) (random.nextDouble() * 128) + 128;
        green = (int) (random.nextDouble() * 128) + 128;
        ox = (int) (Math.random() * mx / 2) + mx / 4;
        oy = (int) (Math.random() * my / 2) + my / 4;
        for (i = 0; i < patch; i++)
        {
            vx[i] = (int) (Math.random() * energy) - energy / 2;
            vy[i] = (int) (Math.random() * energy * 7 / 8) - energy / 8;
        }
    }
    public void start()
    {
        t = 0;
        sleep = false;
    }
    public void show(Graphics g)
    {//输出烟花
        if (!sleep)
            if (t < length)

```

```

    int i, c;
    double s;
    Color color;
    c = (int) (random.nextDouble() * 64) - 32 + red;
    if (c >= 0 && c < 256)
        red = c;
    c = (int) (random.nextDouble() * 64) - 32 + blue;
    if (c >= 0 && c < 256)
        blue = c;
    c = (int) (random.nextDouble() * 64) - 32 + green;
    if (c >= 0 && c < 256)
        green = c;
    color = new Color(red, blue, green);
    for (i = 0; i < patch; i++)
    {
        s = (double) t / 100;
        x = (int) (vx[i] * s);
        y = (int) (vy[i] * s - gravity * s * s);
        g.setColor(color);
        g.drawLine(ox + x, oy - y, ox + x, oy - y);
        if (t >= length / 2)
        {
            int j;
            for (j = 0; j < 2; j++)
            {
                s = (double) ((t - length / 2) * 2 + j) / 100;
                x = (int) (vx[i] * s);
                y = (int) (vy[i] * s - gravity * s * s);
                g.setColor(Color.black);
                g.drawLine(ox + x, oy - y, ox + x, oy - y);
            }
        }
        t++;
    }
    else
    {
        sleep = true;
    }
}

```



案例8 星空模拟

案例运行效果与操作

本案例是一个Java小应用程序（Applet）的例子。运行调用该Applet的网页，浏览器中会显示一个星空的特效，如图2-8所示。



图2-8 星空特效和

制作要点

1. 类Image的使用
2. 类Graphics的使用
3. 双缓冲技术处理闪烁

步骤详解

1. Star类。

构造函数星星：

```
Star( int width, int height, int depth, int type )
{
    this.type = type;
    H = width/2;
    V = height/2;
    x = (int)(Math.random()*width) - H;
    y = (int)(Math.random()*height) - V;
    if( (x == 0) && (y == 0) ) x = 10;
    z = (int)(Math.random()*depth);
}
```

2. 初始化Applet。

创建星体数组：

```
pol = new Star[stars];
```

初始化星体数组：

```
for( int i = 0; i < stars; i++ )
    pol[i] = new Star( Width, Height, 100, type );
```

3. 启动线程。

设置旋转角度：

```
rot += dx;
dx += ddx;
if( dx > max ) ddx=-defddx;
if( dx < -max) ddx=defddx;
try { Thread.sleep( speed ); }
catch (InterruptedException e){}
repaint();
```

4. 输出星体。

```
g.setColor( Color.black );
g.fillRect( 0, 0, Width, Height );
for( int i = 0; i < stars; i++ )
    pol[i].Draw( g, rot );
```

5. 调用Applet的网页。

```
<applet code=StarField.class width=300 height=300>
<PARAM NAME=STARS VALUE=50>
<PARAM NAME=SPEED VALUE=25>
</applet>
```

星体类：

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;

class Star {
    int H, V;
    int x, y, z;
    int type;
    //构造函数
    Star( int width, int height, int depth, int type )
```

```

    {
        this.type = type;
        H = width/2;
        V = height/2;
        x = (int)(Math.random()*width) - H;
        y = (int)(Math.random()*height) - V;
        if( (x == 0) && (y == 0) ) x = 10;
        z = (int)(Math.random()*depth);
    }
    public void Draw( Graphics g, double rot )
    {//输出星体
        double X, Y;
        int h, v, hh, vv;
        int d;
        z=2;
        if( z < -63 ) z = 100;
        hh = (x*64)/(64+z);
        vv = (y*64)/(64+z);
        X = (hh*Math.cos(rot))-(vv*Math.sin(rot));
        Y = (hh*Math.sin(rot))+(vv*Math.cos(rot));
        h = (int)X+H;
        v = (int)Y+V;
        if( (h < 0) || (h > (2*H))) z = 100;
        if( (v < 0) || (v > (2*H))) z = 100;
        GrayMe(g);
        if( type == 0 )//形状1
        {
            d=(100-z)/50;
            if( d == 0 ) d = 1;
            g.fillRect( h, v, d, d );
        }
        else //其他形状
        {
            d=(100-z)/20;
            g.drawLine( h-d, v, h+d, v );
            g.drawLine( h, v-d, h, v+d );
            if( z < 50 ) {
                d/=2;
                g.drawLine( h-d, v-d, h+d, v+d );
                g.drawLine( h+d, v-d, h-d, v+d );
            }
        }
    }
}

```

```

public void GrayMe(Graphics g)
    {//为星体着色
    if ( z > 50 )
    {
        g.setColor( Color.gray );
    }
    else if ( z > 25 )
    {
        g.setColor( Color.lightGray );
    }
    else
    {
        g.setColor( Color.white );
    }
}
}

```

星空类，继承自Applet，实现Runnable接口支持线程：

```

public class StarField extends java.applet.Applet implements Runnable
{
    int          Width, Height;
    Thread      me = null;
    boolean      suspend = false;
    Image        im;
    Graphics    offscreen;
    double       rot, dx, ddx;      //旋转角度,
    int          speed, stars, type; //运动速度、星体数量、星体形状
    double       defddx, max;
    Star        pol[];           //星体数组
    public void init()
    {//初始化Applet
        rot = 0;
        dx=0;
        ddx=0;
        Width = size().width;
        Height = size().height;
        //读取HTML页面中的若干参数
        String theSpeed = getParameter( "speed" );
        Show( "speed", theSpeed );
        speed = (theSpeed == null ) ? 50 : Integer.valueOf( theSpeed ).intValue();
        String theStars = getParameter( "stars" );
        Show( "stars", theStars );
    }
}

```

```

stars = (theStars == null) ? 30 : Integer.valueOf( theStars ).intValue();

String theType = getParameter( "type" );
Show( "type", theType );
type = (theType == null) ? 0 : Integer.valueOf( theType ).intValue();

String theRot = getParameter( "spin" );
Show( "spin", theRot );
rot = (theRot == null) ? 0 : Double.valueOf( theRot ).doubleValue();

String theMax = getParameter( "maxspin" );
Show( "maxspin", theRot );
max = (theMax == null) ? .1 : Double.valueOf( theMax ).doubleValue();

String theddx = getParameter( "ddx" );
Show( "ddx", theddx );
defddx = (theddx == null) ? .005 : Double.valueOf( theddx ).doubleValue();
try
{
    im = createImage( Width, Height );
    offscreen = im.getGraphics();
}
catch( Exception e )
{
    offscreen = null;
}

//创建星体数组
pol = new Star[stars];
//初始化星体数组
for( int i = 0; i < stars; i++ )
    pol[i] = new Star( Width, Height, 100, type );
}

public void paint( Graphics g )
{
    if( offscreen != null )
    {
        paintMe( offscreen );
        g.drawImage( im, 0, 0, this );
    }
    else
    {
        paintMe( g );
    }
}

```

```

        }

    public void paintMe( Graphics g )
        //输出星体
        g.setColor( Color.black );
        g.fillRect( 0, 0, Width, Height );
        for( int i = 0; i < stars; i++ )
            pol[i].Draw( g, rot );
    }

    public void start()
        //启动Applet, 创建并启动线程
        if( me == null )
    {
        me = new Thread( this );
        me.start();
    }
}

public void stop()
    //结束Applet
    if( me != null )
{
    me.stop();
    me = null;
}
}

public void run()
    //启动线程
    while( me != null )
    {
        //设置旋转角度
        rot += dx;
        dx += ddx;
        if( dx > max ) ddx=-defddx;
        if( dx < -max) ddx=defddx;
        try { Thread.sleep( speed ); }
        catch (InterruptedException e){}
        repaint();
    }
}

public void update( Graphics g )
{
    paint( g );
}

```

```

public boolean mouseDown( java.awt.Event evt, int x, int y )
{
    //处理鼠标单击事件
    ddx = (ddx == 0) ? defddx : 0;
    return true;
}
public void Toggle( )
{
    //线程挂起
    if( me != null )
    {
        if( suspend )
        {
            me.resume();
        }
        else
        {
            me.suspend();
        }
        suspend = !suspend;
    }
}
public void Show( String theString, String theValue )
{
    if( theValue == null )
    {
        System.out.println( theString + " : null");
    }
    else
    {
        System.out.println( theString + " : " + theValue );
    }
}
}

```



案例9 阴影跑马灯

案例运行效果与操作

本案例是采用Java Applet实现文字的跑马灯效果，并产生阴影。程序运行后，界面如图2-9所示。



图2-9 运行界面

制作要点

1. 类Graphics的使用
2. 类Runnable的使用
3. 类MediaTracker的使用

步骤详解

1. 控制线程运行间隔时间。

定义要显示的图片和位置：

```
offG.drawImage(img,x,0,this);
```

重画图片：

```
repaint();
thread.sleep(50); 线程休眠50毫秒
```

2. 嵌入Applet的html页面源码。

```
<applet code=ShadowText.class width=300 height=50 VIEWASTEXT>
<param name="Text" value=恭贺新禧>
<param Name="FontSize" value=40>
<param Name="Back" value="0000ff">
<param Name="Fore" value="ff0000">
<param Name="Shadow" value="660066">
</applet>
```

程序源代码与解释

```
import java.applet.Applet;
import java.awt.Color;
```

```
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;

public class ShadowText extends Applet implements Runnable
{
    private Image img;
    private Image offI;
    private Graphics offG;
    private Thread thread = null;
    private MediaTracker imageTracker;
    private int height,width;
    private String text;
    private int FontSize;
    private Font font;
    private int textColor, backcolor, shadowcolor;
    public void init() //初始化
    {
        width = this.size().width;
        height = this.size().height;
        String s = new String(getParameter("Text"));
        text = new String("Hello");
        if(s != null)
            text = s;
        FontSize = 30;
        s = new String(getParameter("FontSize"));
        if(s != null)
            FontSize = Integer.parseInt(s);
        s = getParameter("Fore");
        textColor = (s==null) ? 0x000000 : Integer.parseInt(s, 16);
        s = getParameter("Back");
        backcolor = (s==null) ? 0x000000 : Integer.parseInt(s, 16);
        s = getParameter("shadow");
        shadowcolor = (s==null) ? 0x000000 : Integer.parseInt(s, 16);
        this.setBackground(new Color(backcolor));
        img = createImage(width,height);
        Graphics temp = img.getGraphics();
        temp.setColor(new Color(backcolor));
        temp.fillRect(0,0,width,height);
        temp.setColor(new Color(shadowcolor));
        font = new Font("TimesRoman",Font.BOLD,FontSize);
        temp.setFont(font);
```

```
temp.drawString(text,10,height*3/4);
temp.setColor(new Color(textcolor));
temp.drawString(text,10-3,height*3/4 - 3);
imageTracker = new MediaTracker(this);
imageTracker.addImage(img,0);
try
{
    imageTracker.waitForID(0);
}
catch(InterruptedException e){}
offI = createImage(width,height);
offG = offI.getGraphics();
}
public void start()
{
    if(thread == null)
    {
        thread = new Thread(this);
        thread.start();
    }
}
public void run()
{
    int x=width;
    while(thread != null)
    {
        try
        {
            offG.drawImage(img,x,0,this);
            repaint();
            thread.sleep(50);
        }
        catch(InterruptedException e){}
        x-=3;
        if(x < -width)
        {
            x = width;
        }
    }
}
public void update(Graphics g)
{
    paint(g);
```

```

    }
    public void paint(Graphics g)
    {
        g.drawImage(offI,0,0,this);
    }
}

```



案例10 下雪的图片

案例运行效果与操作

本案例实现在图片上飘雪花的特效，如图2-10所示。



图2-10 下雪特效

制作要点

1. 多线程的使用
2. 类java.util.Random的使用

步骤详解

1. 嵌入Applet的HTML页面源码。

```

<applet code="SnowPic.class" align="baseline" width="300" height="200">
<!--[Tip2]:图文件的位置与名称-->
<param name="graphic" value="Bridge.jpg">
<!--[Tip3]:雪下的数量-->
<param name="snows" value="100">
<param name="snowSize" value="4">
<!--[Tip4]:下雪的速度, 值越小速度越快-->
<param name="threadsleep" value="40">
</applet>

```

2. 用随机数产生雪点。

```

rand = new Random();
    snowX[i] = rand.nextInt() % (dim.width / 2)
+ dim.width / 2;
    snowY[i] = rand.nextInt() % (dim.height / 2)
+ dim.height / 2;

```

程序源代码与解释

```

import java.applet.Applet;
import java.awt.*;
import java.util.Random;

public class SnowPic extends Applet implements Runnable
{
    Thread mainThread ;
    Image offScreen, gAlg[] ;
    Random rand ;
    int stopFlag, snows, wind, threadSleep, snowSize ;
    int[] snowX, snowY ;
    long stopTime = 0;
    Dimension dim ;
    MediaTracker mt ;
    public SnowPic() {}
    int getParameter(String s1, int s2)
    {
        String s = getParameter(s1) ;
        return (s != null) ? Integer.parseInt(s) : s2 ;
    }
    int getParameter(String s1, int s2, int max, int min)
    {
        String s = getParameter(s1) ;
        if (s != null)
        {
            if ((s2 = Integer.parseInt(s)) > max) return max ;
            else if (s2 < min) return min ;
            else return s2 ;
        } else return s2 ;
    }
    String getParameter(String s1, String s2)
    {
        String s = getParameter(s1) ;
        return (s != null) ? s : s2 ;
    }
}

```

```
public void init()
{
    rand = new Random();
    dim = getSize();
    offScreen = createImage(dim.width, dim.height);
    snows = getParameter("snows", 100, 500, 0);
    snowSize = getParameter("snowSize", 3, 10, 3);
    threadSleep = getParameter("threadsleep", 80, 1000, 10);
    snowX = new int[snows];
    snowY = new int[snows];
    for(int i = 0; i < snows; i++)
    {
        snowX[i] = rand.nextInt() % (dim.width / 2)
        + dim.width / 2;
        snowY[i] = rand.nextInt() % (dim.height / 2)
        + dim.height / 2;
    }
}
mt = new MediaTracker(this);
gAlc = new Image[1];
gAlc[0] = getImage(getDocumentBase(),
    getParameter("graphic", "test.gif"));
mt.addImage(gAlc[0], 0);
try
{
    mt.waitForID(0);
}
catch(InterruptedException _ex) { return; }
stopFlag = 0;
}
public void start()
{
    if(mainThread == null)
    {
        mainThread = new Thread(this);
        mainThread.start();
    }
}
public void stop()
{
    mainThread = null;
}
public void run()
```

```

    {
        while(mainThread != null)
        {
            try
            {
                Thread.sleep(threadSleep);
            }
            catch(InterruptedException _ex) { return ; }
            repaint();
        }
    }
    public void drawBackSnow(Graphics g)
    {
        g.setColor(Color.white);
        for(int i = 0; i < snows; i++)
        {
            g.fillOval(snowX[i], snowY[i], snowSize, snowSize);
            snowX[i] += rand.nextInt() % 2 + wind;
            snowY[i] += (rand.nextInt() % 6 + 5) / 5 + 1;
            if(snowX[i] >= dim.width) snowX[i] = 0;
            if(snowX[i] < 0) snowX[i] = dim.width - 1;
            if(snowY[i] >= dim.height || snowY[i] < 0)
            {
                snowX[i] = Math.abs(rand.nextInt() % dim.width);
                snowY[i] = 0;
            }
        }
        wind = rand.nextInt() % 5 - 2 ;
    }
    public void paint(Graphics g)
    {
        offScreen.getGraphics().setColor(Color.black);
        offScreen.getGraphics().fillRect(0, 0, dim.width,
                                         dim.height);
        offScreen.getGraphics().drawImage(gAlc[0], 0, 0, this);
        drawBackSnow(offScreen.getGraphics());
        g.drawImage(offScreen, 0, 0, null);
    }
    public void update(Graphics g)
    {
        paint(g);
    }
}

```



案例11 动态分割线

案例运行效果与操作

本案例是利用Java Applet和4幅图片实现动画效果，并在页面的两端来回跑动。程序运行后，界面如图2-11所示。



图2-11 运行界面

制作要点

接口java.lang.Runnable类的使用

步骤详解

1. 线程执行。

如果有一个类，它已继承了某个类，又想实现多线程，那就可以通过实现Runnable接口来实现。

1) Runnable接口只有一个run()函数。

2) 把一个实现了Runnable接口的对象作为参数产生一个Thread对象，再调用Thread对象的start()函数就可以执行并行操作。如果在产生一个Thread对象时，以一个Runnable接口的实现类的对象作为参数，那么在调用start()函数时，start()会调用Runnable接口的实现类中的run()函数。

实现Runnable接口的类必须要覆盖run()方法。覆盖的run()方法中记述了用线程执行的处理。

启动线程：

```
public void start() {
    if (thread == null) {
        thread = new Thread(this);
        thread.start();
    }
}
```

```

    }
}

```

2. 运行Applet，按照边界实现动画效果。

```

offG.setColor(Color.white);
offG.fillRect(0,0,width,height);
offG.setColor(fg.brighter().brighter());
offG.drawLine(0,(height-ImageH)/2+ImageH,width,(height-ImageH)/2+ImageH);
offG.setColor(fg.darker().darker());
offG.drawLine(0,(height-ImageH)/2+ImageH+1,width,(height-ImageH)/2+ImageH+1);
offG.drawImage(temp,x,y,this);
repaint();

```

3. 嵌入Applet的HTML页面源码。

```

<applet code=Ruler.class width=400 height=60>
</applet>

```

程序源代码与解析

```

import java.applet.*;
import java.awt.*;

public class Ruler extends Applet implements Runnable
{
    private Image IRight1,IRight1,ILeft2,IRight2,temp;
    private Image offI;
    private Graphics offG;
    private Thread thread = null;
    private MediaTracker imageTracker;
    private int height,width;
    public void init()
    {
        IRight1 = getImage(getDocumentBase(),"cat1.gif");
        IRight2 = getImage(getDocumentBase(),"cat2.gif");
        ILeft1 = getImage(getDocumentBase(),"cat3.gif");
        ILeft2 = getImage(getDocumentBase(),"cat4.gif");
        imageTracker = new MediaTracker(this);
        imageTracker.addImage(IRight1,0);
        imageTracker.addImage(ILeft1,0);
        imageTracker.addImage(IRight2,0);
        imageTracker.addImage(ILeft2,0);
        width = this.size().width;
    }
}

```

```
height = this.size().height;
try
{
    imageTracker.waitForID(0);
}
catch(InterruptedException e){}
offI = createImage(width,height);
offG = offI.getGraphics();
}

public void start()
{
    if(thread == null)
    {
        thread = new Thread(this);
        thread.start();
    }
}
public void run()
{
    Color fg = this.getForeground();
    int ImageW,ImageH,x=0,y=0;
    boolean forward = true;
    ImageW = IRight1.getWidth(this);
    ImageH = IRight1.getHeight(this);
    y= (height-ImageH)/2;
    fg = Color.green;
    try
    {
        while(thread != null)
        {
            thread.sleep(200);
            if(forward)
            {
                x+=19;
                if((x%2)==1)
                {
                    temp = IRight1;
                }
                else
                {
                    temp = IRight2;
                }
            }
        }
    }
}
```

```

        if(x>=(width-ImageW))
        {
            forward = false;
        }
        //System.out.println(x);
    }
    else
    {
        x=19;
        if((x%2)==1)
        {
            temp = ILeft1;
        }
        else
        {
            temp = ILeft2;
        }
        if(x == 0)
        {
            forward = true;
        }
    }
    offG.setColor(Color.white);
    offG.fillRect(0,0,width,height);
    offG.setColor(fg.brighter().brighter());
    offG.drawLine(0,(height-ImageH)/2+ImageH,width,(height-ImageH)/2+ImageH);
    offG.setColor(fg.darker().darker());
    offG.drawLine(0,(height-ImageH)/2+ImageH+1,width,(height-ImageH)/2+ImageH+1);
    offG.drawImage(temp,x,y,this);
    repaint();
}
catch(InterruptedException e){}
}
public void update(Graphics g)
{
    paint(g);
}
public void paint(Graphics g)
{
    g.drawImage(offI,0,0,this);
}
}

```



案例12 飞流直下

案例运行效果与操作

本案例是利用Java Applet实现瀑布飞流直下的动画效果，与本章案例8——星空模拟类似。程序运行后，界面如图2-12所示。

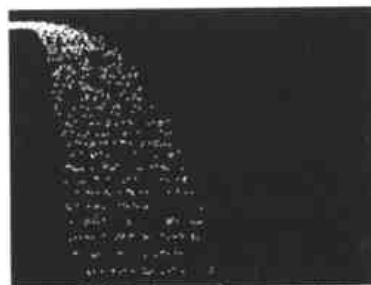


图2-12 运行界面

制作要点

1. 接口Runnable类的使用
2. Applet对线程的支持技术

步骤详解

1. 嵌入Applet的HTML页面源码。

```
<Applet code="WaterFall.class" width=320 height=250>
</Applet>
```

2. 多线程的应用。

多线程实际上就是启动另一个进程，其运行的过程独立于主程序之外，并从主程序的Start()方法载入，由Run()方法调用执行。实现多线程的方法有两种，即：创建Thread类的子类（类的继承）和实现Runnable接口。

在Java语言中，一个子类只能继承一个超类（父类）。由于要编写的Java Applet是应用于网页中的，首先必须继承浏览器类（java.applet），因此，在本例中通过实现Runnable接口的方法来实现多线程，实现的语句如下：

```
public class WaterFall extends Applet implements Runnable
```

首先，在Init()方法中对要创建的线程进行定义及初始化：

```
Thread pThread;
```

然后，在Start()和Stop()方法中加入以下代码：

```
public void start()
{
    pThread = new Thread(this); //创建一个新线程
    pThread.start();          //启动该线程
}
public void stop()
{
    pThread = null;           //调用stop()方法，释放系统资源
}
```

程序源代码与解释

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

public class WaterFall extends Applet implements Runnable
{
    final int Max = 1000;
    wparticle p[];
    int AppletWidth,AppletHeight,XCenter,YCenter;
    Image OffScreen;
    Graphics drawOffScreen;
    Thread pThread;

    public void init()
    {
        setBackground(Color.black);
        AppletWidth = getSize().width;
        AppletHeight = getSize().height;
        p = new wparticle[Max];
        for(int i=0; i<Max; i++)
            p[i] = new wparticle();
        OffScreen = createImage(AppletWidth,AppletHeight);
        drawOffScreen = OffScreen.getGraphics();
    }
    public void start()
    {
        pThread = new Thread(this);
        pThread.start();
    }
    public void stop()
```

```
{  
    pThread = null;  
}  
public void update(Graphics g)  
{  
    paint(g);  
}  
public void paint(Graphics g)  
{  
    g.drawImage(OffScreen,0,0,this);  
}  
public void run()  
{  
    boolean reset = false;  
    int i, t = 0;  
    while(true)  
    {  
        drawOffScreen.clearRect(0,0,AppletWidth,AppletHeight);  
        drawOffScreen.setColor(Color.white);  
        drawOffScreen.drawLine(0,15,10,15);  
        for(i=0; i<Max; i++)  
        {  
            drawOffScreen.fillOval((int)p[i].X,(int)p[i].Y,3,3);  
            p[i].X = p[i].X + p[i].Vx;  
            if(p[i].X > 10)  
            {  
                p[i].Y += p[i].Vy*p[i].time / 1000;  
                p[i].Vy = (int) 9.8*p[i].time;  
                p[i].time++;  
            }  
            if(p[i].Y > AppletHeight)  
            {  
                p[i].reset();  
            }  
        }  
        repaint();  
        try {  
            Thread.sleep(100);  
        }  
        catch (InterruptedException e) { }  
    }  
}
```

```
}

class wparticle
{
    double X,Y;
    double Vx,Vy;
    int time;
    public wparticle()
    {
        reset();
    }
    public void reset()
    {
        X = (int) (Math.random() * -40);
        Y = (int) (Math.random() * 5 + 10);
        Vx = Math.random()*3 + 1.0;
        Vy = 0;
        time = 0;
    }
}
```

本 章 小 结

动态特效一般是通过在不同的坐标位置显示相应的图片（或图形）而获得的，通过Applet对线程的支持，即可得到精美的动画效果。

第3章

XML与其他



本章内容

- 案例1 将HTML文件转换成XML文件
- 案例2 把XML文件转换成可浏览的HTML格式文件
- 案例3 用JDOM解析XML文件
- 案例4 Java编制的时钟
- 案例5 简单日历
- 案例6 系统内存状态监视程序
- 案例7 计算器
- 案例8 多线程文件下载
- 案例9 笛卡儿曲线
- 本章小结



案例1 将HTML文件转成XML文件

案例运行效果与操作

本案例是一个用Java实现的将HTML文件转换为XML文件的例子，实现了将HTML文件转换为XML文件的功能，这个功能非常有用。在现实应用中，由于浏览器的容错性，大量的HTML网页格式很不规范，将其转换为XML文件之后，将大大方便网络信息的抓取。程序运行后，界面如图3-1所示。



图3-1 运行界面

在Eclipse中直接运行该程序，就会看到图3-1，程序产生的XML文件和错误提示信息都保存在D盘temp文件夹下，如图3-2所示。

名称	大小	类型
name.jpg	6 KB	JPEG 图像
jtidyerror.txt	1 KB	文本文档
jtidytext.html	3 KB	HTML Document
audit.log	8 KB	文本文档
class.log	19 KB	文本文档
com.log	23 KB	文本文档
trace.log	31 KB	文本文档
text.html	2 KB	HTML Document
error.txt	1 KB	文本文档
medical.xml	3 KB	XML 文档

图3-2 XML文件和错误信息

制作要点

1. 输入/输出流缓冲区BufferedInputStream()的应用
2. Java扩展标准库org.w3c.tidy.Tidy的使用
3. URL的使用

步骤详解

1. 实例设置输出的文件是XML格式的。

告诉Tidy将HTML转换为XML。

JTidy提供HTML语法检查和HTML的“pretty printing（漂亮打印）”，它还允许将一个HTML文件作为输入，然后将其转换为XML。JTidy读取输入文件，如果发现有任何不匹配或遗漏的闭合标记，将纠正这些标记，最后输出一个格式良好的XML文档。

```
Tidy tidy = new Tidy();
tidy.setXmlOut(true);
```

2. 提供一个输入URL。

```
this.url = url;
```

还有一个输出和错误文件:

```
this.outFileName = outFileName;
this.errOutFileName = errOutFileName;
```

3. 进行文件转换。

JTidy对文件转换的方法进行了良好的包装，大家只要调用parse方法就可以了。

```
try {
    //将错误信息保存到文件中
    tidy.setErrout(new PrintWriter(new FileWriter(errOutFileName),true));
    u = new URL(url);
    //创建一个输入输出流
    in = new BufferedInputStream(u.openStream());
    out = new FileOutputStream(outFileName);
```

转换文件:

```
tidy.parse(in, out);
```

程序源代码与解释

```
import java.io.BufferedInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.URL;
import org.w3c.tidy.Tidy;
//HTML文件转换成XML输出
public class HTML2XML {
    private String url;
    private String outFileName;
    private String errOutFileName;
    public HTML2XML(String url, String outFileName, String errOutFileName) {
        this.url = url;
        this.outFileName = outFileName;
        this.errOutFileName = errOutFileName;
    }
    public void convert() {
        URL u;
        BufferedInputStream in;
```

```

FileOutputStream out;
Tidy tidy = new Tidy();
//告诉Tidy将HTML转换为XML
tidy.setXmlOut(true);
try {
    //将错误信息保存到文件中
    tidy.setErrout(new PrintWriter(new FileWriter(errOutFileName),
                                   true));
    u = new URL(url);
    //创建一个输入输出流
    in = new BufferedInputStream(u.openStream());
    out = new FileOutputStream(outFileName);
    //转换文件
    tidy.parse(in, out);
    in.close();
    out.close(); //释放
} catch (IOException e) {
    System.out.println(this.toString() + e.toString());
}
}

public static void main(String[] args) {
    //参数：HTML文件的URL，输出文件和错误文件名
    System.out.println("程序开始运行.....");
    TestHTML2XML t = new TestHTML2XML(
        "http://localhost/servlet-study/testmail.jsp",
        "d:\\temp\\html2xml.xml", "d:\\temp\\error.txt");
    t.convert();
    System.out.println("程序运行结束.....");
}
}
}

```



案例2 把XML文件转换成可 浏览的HTML格式文件

案例运行效果与操作

本案例将指定的XML文件转换成可浏览的HTML格式文件。程序执行结果如图3-3所示。
对应的HTML文件如下：

```

<html
  xmlns="http://www.w3.org/1999/xhtml"><head><title>Schedule</title></head><body><?xml
version="1.0" encoding="UTF-8"?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><h2
align="center">Dr.&nbsp;Zhu's Schedule</h2><hr xmlns="" /><h3
xmlns="">Appointment</h3><p xmlns="">03/15/2005 from 09:30
until 10:30</p><table xmlns=""><tr><td>Subject:</td><td>Interview
potential new hire</td></tr><tr><td>Location:</td><td>Rm
103</td></tr><tr><td>Note:</td><td>Ask Bob for an updated resume.</td></tr></table><hr
xmlns="" /><h3 xmlns="">Appointment</h3><p xmlns="">03/15/2005
from 15:30 until 16:30</p><table
xmlns=""><tr><td>Subject:</td><td>Dr. Appointment</td></tr><tr><td>Location:
</td><td>No.6 LuoYuan Avenue</td></tr><tr><td>Note:</td><td /></tr></table><hr
xmlns="" /><h3 xmlns="">Appointment</h3><p
xmlns="">03/16/2005 from 11:30 until 12:30</p><table
xmlns=""><tr><td>Subject:</td><td>Lunch
w/Boss</td></tr><tr><td>Location:</td><td>QiluQanbao Ting on News
Hotel</td></tr><tr><td>Note:</td><td /></tr></table></body></html>

```

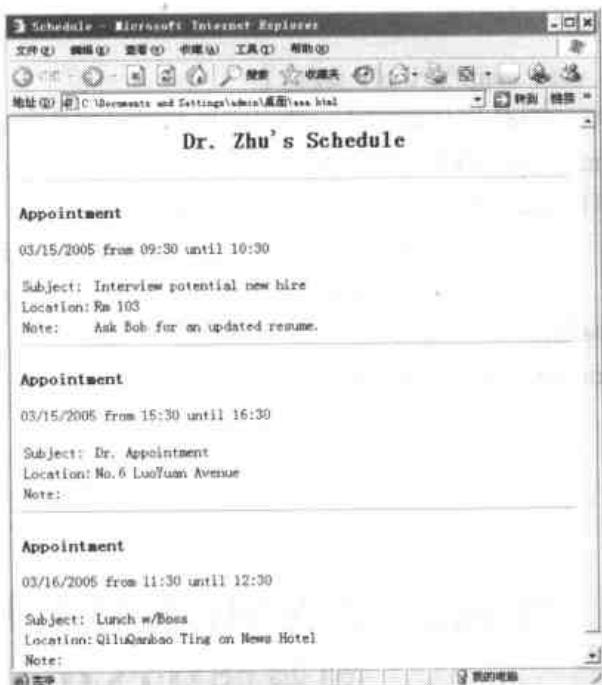


图3-3 HTML文件浏览

制作要点

1. javax.xml.transform.stream.StreamSource的用法
2. javax.xml.transform.stream.StreamResult的用法

步骤详解

1. 读入源文件。

```
Source xmlSource = new StreamSource(xmlFile);
Source xsltSource = new StreamSource(xsltFile);
```

2. 利用java.xml.transform包中的方法处理XSLT变换。

```
TransformerFactory transFact =
    TransformerFactory.newInstance();
Transformer trans = transFact.newTransformer(xsltSource);
```

3. 输出转换后的文件。

```
trans.transform(xmlSource, new StreamResult(System.out))
```

程序源代码与解析

```
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class Transform {
    //本程序执行XSLT转换，将结果输出到标准输出
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println(
                "Usage: java Transform [xmlfile] [xsltfile]");
            System.exit(1);
        }
        File xmlFile = new File(args[0]);
        File xsltFile = new File(args[1]);
        Source xmlSource = new StreamSource(xmlFile);
        Source xsltSource = new StreamSource(xsltFile);
        TransformerFactory transFact =
            TransformerFactory.newInstance();
        Transformer trans = transFact.newTransformer(xsltSource);
        trans.transform(xmlSource, new StreamResult(System.out));
    }
}
```



案例3 用JDOM解析XML文件

案例运行效果与操作

本案例利用JDOM实现XML的解析。

sample.xml如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<car vin="123fhg5869705iop90">
    <!--Description of a car-->
    <make>Toyota</make>
    <model>Celica</model>
    <year>1997</year>
    <color>green</color>
    <license state="CA">1ABC234</license>
</car>
```

输出内容如下：

```
parse successfull
vin = 123fhg5869705iop90
打印全部节点
vin = 123fhg5869705iop90
#text =
    #comment = Description of a car
#text =
    #text = Toyota
#text =
    #text = Celica
#text =
    #text = 1997
#text =
    #text = green
#text =
    state = CA
#text = 1ABC234
#text =
```

制作要点

1. 利用JAXP解析XML文件
2. DOM的用法

步骤详解

1. 建立解析器工厂，利用这个工厂获得一个具体的解析器对象。

```
factory = DocumentBuilderFactory.newInstance();
```

使用DocumentBuilderFactory的目的是为了创建与具体解析器无关的程序，当DocumentBuilderFactory类的静态方法newInstance()被调用时，它根据一个系统变量来决定具体使用哪一个解析器。因为所有的解析器都服从于JAXP所定义的接口，所以无论具体使用哪一个解析器，代码都是一样的。所以，当在不同的解析器之间进行切换时，只需更改系统变量的值，而不用更改任何代码。

2. 利用这个解析器对XML文档进行解析。

当获得一个工厂对象后，使用它的静态方法newDocumentBuilder()方法可以获得一个DocumentBuilder对象，这个对象代表了具体的DOM解析器。但具体是哪一种解析器，微软的或者IBM的，对于程序而言并不重要。

```
docBuilder = factory.newDocumentBuilder();
```

然后，就可以解析了：

```
doc = docBuilder.parse(in);
```

DocumentBuilder的parse()方法接受一个XML文档名作为输入参数，返回一个Document对象，这个Document对象就代表了一个XML文档的树模型。以后所有对XML文档的操作都与解析器无关，直接在这个Document对象上进行操作就可以了。

3. 获得Node对象。

```
root = doc.getDocumentElement();
elementName = root.getNodeName();
```

然后就可以输出打印了。

程序源代码与解释

```
import java.io.FileInputStream;
import javax.xml.parsers.*;
import org.w3c.dom.*;
//Java读取XML文档，利用DOM来读取一个XML文档的内容，并输出到标准输出
public class TestXML {
```

```
public static void main(String[] args) {  
    Document doc;  
    DocumentBuilderFactory factory;  
    DocumentBuilder docBuilder;  
  
    Element root;  
    String elementName;  
  
    FileInputStream in;  
    String fileName;  
    try{  
        //获取XML文件  
        fileName = System.getProperty("user.dir");  
        fileName = fileName+"/sample.xml";  
        in = new FileInputStream(fileName);  
        //解析XML文件，生成Document对象  
        factory = DocumentBuilderFactory.newInstance();  
        factory.setValidating(false);  
        docBuilder = factory.newDocumentBuilder();  
        doc = docBuilder.parse(in);  
        //解析成功  
        System.out.println("parse successfull");  
        //获取XML文档的根节点  
        root = doc.getDocumentElement();  
        elementName = root.getNodeName();  
        //打印根节点的属性  
        printAttributes(root);  
        //打印该文档全部节点  
        System.out.println("打印全部节点");  
        printElement(root,0);  
    }  
    catch(Exception exp){  
        exp.printStackTrace();  
    }  
}  
//打印某个节点的全部属性  
public static void printAttributes(Element elem){  
    NamedNodeMap attributes;  
    int i,max;  
    String name,value;  
    Node curNode;  
  
    attributes = elem.getAttributes();  
    max = attributes.getLength();  
  
    for(i=0;i<max;i++){  
        curNode = attributes.item(i);  
    }  
}
```

```
name = curNode.getNodeName();
value = curNode.getNodeValue();
System.out.println("\t"+name+" = "+value);
}
}

//打印所有节点的名称和值
//该方法采用递归方式打印文档的全部节点
public static void printElement(Element elem,int depth){
String elementName;
NodeList children;
int i,max;
Node curChild;
Element curElement;
String nodeName,nodeValue;

//elementName = elem.getNodeName();
//获取输入节点的全部子节点
children = elem.getChildNodes();

//按一定格式打印输入节点
for(int j=0;j<depth;j++){
System.out.print(" ");
}
printAttributes(elem);

//采用递归方式打印全部子节点
max = children.getLength();
for(i=0;i<max;i++){

curChild = children.item(i);

//递归退出条件
if(curChild instanceof Element){
curElement = (Element)curChild;
printElement(curElement,depth+1);
}
else{
nodeName = curChild.getNodeName();
nodeValue = curChild.getNodeValue();

for(int j=0;j<depth;j++)System.out.print(" ");
System.out.println(nodeName+" = "+nodeValue);
}
}
}
}
```



案例4 Java编制的时钟

案例运行效果与操作

本案例是一个用Java实现的简单动态时钟程序，以图形和数字两种方式来显示当前时间。程序运行后，界面如图3-4所示。

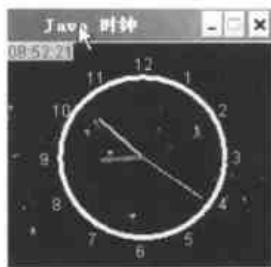


图3-4 时钟

制作要点

- 类包java.util.Calendar的应用
- 将时间转换成图形表示的弧度，按照一定的公式来计算
- GregorianCalendar()的用法

步骤详解

- 监听事件。

```
ActionListener drawClock = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        repaint();
    }
};
```

创建一个时间计数器，每一秒触发一次：

```
new Timer(delay,drawClock).start();
```

- 刷新。

首先擦除：

```
g.setColor(backgroundColor);
g.drawLine(x0,y0,oldm_x,h-oldm_y);
```

重新按照计算所得绘制：

```
x = (int)(r*0.7*Math.cos(RAD*mm))+x0;
```

```

y = (int)(r*0.7*Math.sin(RAD*mm))+y0-2*T;
g.setColor(Color.green );
g.drawLine(x0,y0,x,h-y);
oldm_x = x;
oldm_y = y;
old_m = mm;
g2D.setStroke(new BasicStroke(3.4f))

```

3. 显示界面。

构建一个窗体，显示风格可自行确定。

程序源代码与解密

```

import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.*;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class Clock extends JFrame implements ActionListener {
    int x,y,x0,y0,r,h,olds_x,olds_y,oldm_x,oldm_y,oldh_x,oldh_y,ss,mm,hh,old_m,old_h,ang;
    final double RAD = Math.PI/180; //度数转换成弧度的比例

    //构造函数创建了一个窗体
    public Clock() {
        super("Java时钟");
        setDefaultCloseOperation(3);
        Image image = getToolkit().getImage("clock.gif");
        setIconImage(image);
        setSize(200,200);
        setBackground(Color.BLACK );
        setLocation(300,150);
        setResizable(false);
        show();
        int delay = 1000;
        //创建一个监听事件
        ActionListener drawClock = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                repaint();
            }
        };
        //创建一个时间计数器，每一秒触发一次
        new Timer(delay,drawClock).start();
    }
}

```

```

        }

        //实现 ActionListener接口必须实现的方法
        public void actionPerformed(ActionEvent e) { }

        //绘制图形
        public void paint(Graphics g) {
            Graphics2D g2D = (Graphics2D) g;
            Insets insets = getInsets();
            int L = insets.left/2 ,T = insets.top/2 ;
            h = getSize().height ;
            g.setColor(Color.white );
            //画圆
            g2D.setStroke(new BasicStroke(4.0f));
            g.drawOval(L+40,T+40,h-80,h-80);
            r=h/2-40;
            x0=40+r-5+L;
            y0=40+r-5-T;
            ang=60;
            //绘制时钟上的12个字
            for (int i=1; i<=12; i++) {
                x=(int)((r+10)*Math.cos(RAD*ang)+x0);
                y=(int)((r+10)*Math.sin(RAD*ang)+y0);
                g.drawString(""+i,x,h-y);
                ang-=30;
            }
            //获得现在的时间
            Calendar now = new GregorianCalendar();
            int nowh = now.get(Calendar.HOUR_OF_DAY );
            int nowm = now.get(Calendar.MINUTE );
            int nows = now.get(Calendar.SECOND );
            String st;
            if(nowh<10) st = "0"+nowh ; else st = ":"+nowh;
            if(nowm<10) st += ":0"+nowm; else st += ":"+nowm;
            if(nows<10) st += ":0"+nows; else st += ":"+nows;
            //在窗体上显示时间
            g.setColor(Color.pink);
            g.fillRect(L,T,50,28);
            g.setColor(Color.blue );
            g.drawString(st,L+2,T+26);
            //计算时间与度数的关系
            ss = 90-nows*6;
            mm = 90-nowm*6;
            hh = 90-nowh*30-nowm/2;
        }
    }
}

```

```

x0 = r+40+L;
y0 = r+40+T;
g2D.setStroke(new BasicStroke(1.2f));
//擦除秒针
if(olds_x>0) {
    g.setColor(getBackground());
    g.drawLine(x0,y0,olds_x,h-olds_y);
}
else {
    old_m = mm;
    old_h = hh;
}
//绘制秒针
x = (int)(r*0.9*Math.cos(RAD*ss))+x0;
y = (int)(r*0.9*Math.sin(RAD*ss))+y0-2*T;
g.setColor(Color.yellow );
g.drawLine(x0,y0,x,y);
olds_x = x;
olds_y = y;
g2D.setStroke(new BasicStroke(2.2f));
//擦除分钟
if(old_m!=mm) {
    g.setColor(getBackground());
    g.drawLine(x0,y0,oldm_x,h-oldm_y);
}
//绘制分钟
x = (int)(r*0.7*Math.cos(RAD*mm))+x0;
y = (int)(r*0.7*Math.sin(RAD*mm))+y0-2*T;
g.setColor(Color.green );
g.drawLine(x0,y0,x,y);
oldm_x = x;
oldm_y = y;
old_m = mm;
g2D.setStroke(new BasicStroke(3.4f));
//擦除时针
if(old_h!=hh) {
    g.setColor(getBackground());
    g.drawLine(x0,y0,oldh_x,h-oldh_y);
}
//绘制时针
x = (int)(r*0.5*Math.cos(RAD*hh))+x0;
y = (int)(r*0.5*Math.sin(RAD*hh))+y0-2*T;

```

```

        g.setColor(Color.red );
        g.drawLine(x0,y0,x,h-y);
        oldh_x = x;
        oldh_y = y;
        old_h = hh;
    }

    public static void main(String[] args) {
        Clock clock = new Clock();
    }
}

```



案例5 简单日历

案例运行效果与操作

本案例实现一个日历的功能，可显示当前的年、月、日和星期。也可以查询，选择年、月后，显示当时的日历。程序运行后，界面如图3-5所示。

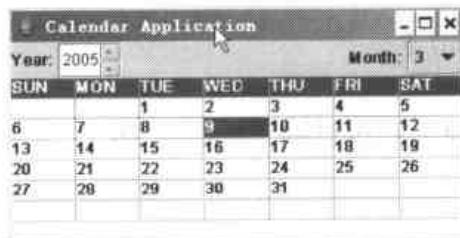


图3-5 日历

制作要点

1. AWT与Swing的使用
2. 应用JApplet生成界面

步骤详解

1. JApplet生成界面。

```
public class MyCalendar extends JApplet {
```

因为Swing库类的单线程原则，JApplet和Applet有一点不同。

2. 获得当前时间。

```
calendar = Calendar.getInstance()
```

Calendar.getInstance()可以获得当前时间，可以使用它的get(int)方法取得年、月、日。

3. 目标时间的查询。

利用set()方法设置年、月、日、时间、分秒的值域。

```
calendar.set(Calendar.DAY_OF_MONTH, day > maxDay ? maxDay : day);
```

程序源代码与解释

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.*;

public class MyCalendar extends JApplet {
    //定义星期天到星期六
    public static final String WEEK_SUN = "SUN";
    public static final String WEEK_MON = "MON";
    public static final String WEEK_TUE = "TUE";
    public static final String WEEK_WED = "WED";
    public static final String WEEK_THU = "THU";
    public static final String WEEK_FRI = "FRI";
    public static final String WEEK_SAT = "SAT";
    //设置背景颜色
    public static final Color background = Color.white;
    //设置前景颜色
    public static final Color foreground = Color.black;
    //设置题头星期的背景颜色和前景颜色
    public static final Color headerBackground = Color.blue;
    public static final Color headerForeground = Color.white;
    //设置被选中的日期的背景颜色和前景颜色
    public static final Color selectedBackground = Color.blue;
    public static final Color selectedForeground = Color.white;

    private JPanel cPane;
    private JLabel yearsLabel; //年
    private JSpinner yearsSpinner; //年调控
    private JLabel monthsLabel; //月
    private JComboBox monthsComboBox; //月下拉框
    private JTable daysTable; //用来显示日期的table
    private AbstractTableModel daysModel;
    private Calendar calendar;
    //构造方法初始化panel
```

```

public MyCalendar() {
    cPane = (JPanel) getContentPane();
}
//初始化，对所有的空间进行布局。
public void init() {
    cPane.setLayout(new BorderLayout());
    //使用border布局管理器
    calendar = Calendar.getInstance();
    calendar = Calendar.getInstance();
    yearsLabel = new JLabel("Year: ");
    yearsSpinner = new JSpinner();
    yearsSpinner.setEditor(new JSpinner.NumberEditor(yearsSpinner, "0000"));
    yearsSpinner.setValue(new Integer(calendar.get(Calendar.YEAR)));
    //增加监听 监听年份的改变
    yearsSpinner.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent changeEvent) {
            int day = calendar.get(Calendar.DAY_OF_MONTH);
            calendar.set(Calendar.DAY_OF_MONTH, 1);
            calendar.set(Calendar.YEAR, ((Integer) yearsSpinner.getValue()).intValue());
            int maxDay = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
            calendar.set(Calendar.DAY_OF_MONTH, day > maxDay ? maxDay : day);
            updateView();
        }
    });
}

JPanel yearMonthPanel = new JPanel();
cPane.add(yearMonthPanel, BorderLayout.NORTH);
yearMonthPanel.setLayout(new BorderLayout());
yearMonthPanel.add(new JPanel(), BorderLayout.CENTER);
JPanel yearPanel = new JPanel();
yearMonthPanel.add(yearPanel, BorderLayout.WEST);
yearPanel.setLayout(new BorderLayout());
yearPanel.add(yearsLabel, BorderLayout.WEST);
yearPanel.add(yearsSpinner, BorderLayout.CENTER);

monthsLabel = new JLabel("Month: ");
//向月份下拉框中增加内容
monthsComboBox = new JComboBox();
for (int i = 1; i <= 12; i++) {
    monthsComboBox.addItem(new Integer(i));
}
monthsComboBox.setSelectedIndex(calendar.get(Calendar.MONTH));
//监听月份的改变
monthsComboBox.addActionListener(new ActionListener() {

```

```

public void actionPerformed(ActionEvent actionEvent) {
    int day = calendar.get(Calendar.DAY_OF_MONTH);
    calendar.set(Calendar.DAY_OF_MONTH, 1);
    calendar.set(Calendar.MONTH, monthsComboBox.getSelectedIndex());
    int maxDay = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
    calendar.set(Calendar.DAY_OF_MONTH, day > maxDay ? maxDay : day);
    updateView();
}
});

JPanel monthPanel = new JPanel();
yearMonthPanel.add(monthPanel, BorderLayout.EAST);
monthPanel.setLayout(new BorderLayout());
monthPanel.add(monthsLabel, BorderLayout.WEST);
monthPanel.add(monthsComboBox, BorderLayout.CENTER);

daysModel = new AbstractTableModel() {
//设置行7
    public int getRowCount() {
        return 7;
    }
    //设置列7
    public int getColumnCount() {
        return 7;
    }
    public Object getValueAt(int row, int column) {
        if (row == 0) {
            return getHeader(column);
        }
        row--;
        Calendar calendar = (Calendar) MyCalendar.this.calendar.clone();
        calendar.set(Calendar.DAY_OF_MONTH, 1);
        int dayCount = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
        int moreDayCount = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        int index = row * 7 + column;
        int dayIndex = index - moreDayCount + 1;
        if (index < moreDayCount || dayIndex > dayCount) {
            return null;
        } else {
            return new Integer(dayIndex);
        }
    }
};

```

```

daysTable = new CalendarTable(daysModel, calendar);
//设置每个cell可以被选中
daysTable.setCellSelectionEnabled(true);
daysTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

daysTable.setDefaultRenderer(daysTable.getColumnClass(0), new TableCellRenderer() {
    public Component getTableCellRendererComponent(JTable table, Object
value, boolean isSelected,
        boolean hasFocus, int row, int column) {
        String text = (value == null) ? "" : value.toString();
        JLabel cell = new JLabel(text);
        cell.setOpaque(true);
        if (row == 0) {
            cell.setForeground(headerForeground);
            cell.setBackground(headerBackground);
        } else {
            if (isSelected) {
                cell.setForeground(selectedForeground);
                cell.setBackground(selectedBackground);
            } else {
                cell.setForeground(foreground);
                cell.setBackground(background);
            }
        }
        return cell;
    }
});
updateView();
cPane.add(daysTable, BorderLayout.CENTER);
}

public static String getHeader(int index) {
    switch (index) {
    case 0:
        return WEEK_SUN;
    case 1:
        return WEEK_MON;
    case 2:
        return WEEK_TUE;
    case 3:
        return WEEK_WED;
    case 4:
        return WEEK_THU;
    }
}

```

```

        case 5:
            return WEEK_FRI;
        case 6:
            return WEEK_SAT;
        default:
            return null;
    }
}

public void updateView() {
    daysModel.fireTableDataChanged();
    daysTable.setRowSelectionInterval(calendar.get(Calendar.WEEK_OF_MONTH),
                                      calendar.get(Calendar.WEEK_OF_MONTH));
    daysTable.setColumnSelectionInterval(calendar.get(Calendar.DAY_OF_WEEK) - 1,
                                         calendar.get(Calendar.DAY_OF_WEEK) - 1);
}
//设置日历的table
public static class CalendarTable extends JTable {

    private Calendar calendar;

    public CalendarTable(TableModel model, Calendar calendar) {
        super(model);
        this.calendar = calendar;
    }

    public void changeSelection(int row, int column, boolean toggle, boolean extend) {
        super.changeSelection(row, column, toggle, extend);
        if (row == 0) {
            return;
        }
        Object obj = getValueAt(row, column);
        if (obj != null) {
            calendar.set(Calendar.DAY_OF_MONTH, ((Integer)obj).intValue());
        }
    }
}

//让applet做一个可执行的程序来运行
public static void main(String[] args) {
    JFrame frame = new JFrame("Calendar Application");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    MyCalendar myCalendar = new MyCalendar();
    myCalendar.init();
    frame.getContentPane().add(myCalendar);
}

```

```

        frame.setSize(240, 172);
        frame.setVisible(true);
    }
}

```



案例6 系统内存状态监视程序

案例运行效果与操作

本案例动态显示了当前机器内存使用情况，不断刷新，空闲内存所占比例。程序运行后，如图3-6所示。

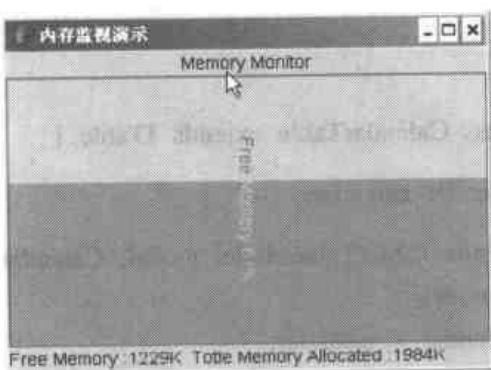


图3-6 “内存监视演示”界面

制作要点

1. 多线程的应用
2. getRuntime()的用法

步骤详解

Java的一个主要优点就是通过垃圾收集器（Garbage Collection，简称GC）自动管理内存的回收，不需要通过调用函数来释放内存。Java的内存管理就是对象的分配和释放问题。在Java中，程序员需要通过关键字new为每个对象申请内存空间（基本类型除外），所有的对象都在堆（Heap）中分配空间。另外，对象的释放是由GC决定和执行的。在Java中，内存的分配是由程序完成的，而内存的释放是由GC完成的，这种收支两条线的方法确实简化了程序员的工作。但同时，它也加重了JVM的工作，这也是Java程序运行速度较慢的原因之一。GC为了能够正确释放对象，必须监控每一个对象的运行状态，包括对象的申请、引用、被引用、赋值等。

系统的内存是否释放还是需要关注的，可以通过Java提供的方法得到当前系统内存的使用情况：

```
totle=(int)(Runtime.getRuntime().totalMemory()/1024);
free=(int)(Runtime.getRuntime().freeMemory()/1024);
```

程序源代码与解释

```
MainFrame.java
package jmemorydemo;

import java.awt.AWTEvent;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.SwingConstants;

public class MainFrame extends JFrame {
    private JPanel contentPane;
    private BorderLayout borderLayout1 = new BorderLayout();
    private JProgressBar jProgressBar1 = new JProgressBar();
    private JLabel jLabel1 = new JLabel();
    private JLabel jLabel2 = new JLabel();
    //构造方法
    public MainFrame() {
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try {
            jbInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    //组件初始化
    private void jbInit() throws Exception {
        contentPane = (JPanel) this.getContentPane();
        contentPane.setLayout(borderLayout1);
        this.setSize(new Dimension(304, 215));
        this.setTitle("内存监视演示");
        jLabel1.setFont(new java.awt.Font("Dialog", 0, 14));
        jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
        jLabel1.setText("Memory Monitor");
        contentPane.add(jLabel1, "Center");
        contentPane.add(jProgressBar1, "South");
    }
}
```

```

//将进度条设置成为垂直状态
jProgressBar1.setOrientation(JProgressBar.VERTICAL);
jProgressBar1.setFont(new java.awt.Font("Dialog", 0, 14));
jProgressBar1.setToolTipText("");
//设置进度条能够显示字符串文本
jProgressBar1.setStringPainted(true);

jLabel2.setFont(new java.awt.Font("Dialog", 0, 14));
jLabel2.setText("");
contentPane.add(jProgressBar1, BorderLayout.CENTER);
contentPane.add(jLabel1, BorderLayout.NORTH);
contentPane.add(jLabel2, BorderLayout.SOUTH);
//调用类ProgressThread
ProgressThread pThread = new ProgressThread(this.jProgressBar1,
    this.jLabel2);
pThread.start();
}

//覆盖方法processWindowEvent (WindowEvent e) , 当窗口关闭时, 程序能够正常结束
protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}
}

ProgressThread.java
package jmemorydemo;

import javax.swing.JLabel;
import javax.swing.JProgressBar;
public class ProgressThread extends Thread {
    JProgressBar pbar;//进度条
    JLabel label;//标签
    public ProgressThread(JProgressBar pbar, JLabel label) {
        this.pbar = pbar;
        this.label = label;
    }
    public void run() {
        /** @todo Override this java.lang.Thread method */
        int min = 0;
        int max = 100;
        int free = 0;
        int totle = 0;
    }
}

```

```

int status = 0;
pbar.setMinimum(min);//设置进度条的最小长度为0
pbar.setMaximum(max);//设置进度条的最大长度为100
pbar.setValue(status);//设置进度条当前的值
while (true) {
    try {
        totle = (int) (Runtime.getRuntime().totalMemory() / 1024);
        //获取总的内存数
        free = (int) (Runtime.getRuntime().freeMemory() / 1024);
        //获取空闲的内存数
    } catch (Exception e) {
        e.printStackTrace();
    }
    //设置标签的显示文本内容
    label.setText("Free Memory :"
        + (int) (Runtime.getRuntime().freeMemory() / 1024) + "K"
        + " Total Memory Allocated :"
        + (int) (Runtime.getRuntime().totalMemory() / 1024) + "K");
    status = (int) (free * 100 / totle);
    pbar.setValue(status);
    //设置进度条显示的内容
    pbar.setString("Free Memory " + status + "%");
    try {
        Thread.sleep(1000);
    } catch (InterruptedException err) {
    }
}
}

JmemoryDemo.java
package jmemorydemo;

import java.awt.Dimension;
import java.awt.Toolkit;

public class JMemoryDemo {
    private boolean packFrame = false;
    //构造方法
    public JMemoryDemo() {
        MainFrame frame = new MainFrame();
        //校验frame的预设值,
        if (packFrame) {
            frame.pack();
        }
    }
}

```

```

    } else {
        frame.validate();
    }
    //让主窗口启动后位于屏幕的中央
    //取得屏幕的长度宽度
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    //取得当前frame的长度和宽度
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
        frameSize.width = screenSize.width;
    }
    //设置frame的显示的位置
    frame.setLocation((screenSize.width - frameSize.width) / 2,
                      (screenSize.height - frameSize.height) / 2);
    frame.setVisible(true);
}
//main()方法
public static void main(String[] args) {
    new JMemoryDemo();
}
}

```



案例7 简单的计算器

案例运行效果与操作

本案例是一个计算器的小例子，它实现简单加减乘除四则运算、乘方、倒数和平方根的小程序，还可以用存储计算。操作步骤为，输入运算数，点击运算符，然后输出结果。例如，求 $56 * 65$ 的值，如图3-7所示。首先输入56，单击运算符号“*”，再输入乘数65，点击运算符“=”，得出结果如图3-8。



图3-7 运行界面

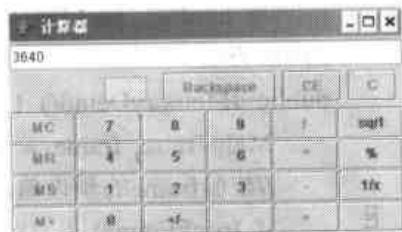


图3-8 结果界面

制作要点

Java布局的设计

步骤详解

Java布局管理器。

Java提供了多种方法进行布局管理如FlowLayout()和BorderLayout()、GridLayout()、GridBagLayout()等。为了布局美观整洁，本例中三种布局方法都已用到。

FlowLayout()。顾名思义，也就是流布局方式。这是一种顺其自然的方式：从左到右，一个个对象地摆放，摆不下，就摆到下一行。简单但无法控制。

计算器中输入条和第二行布局使用了FlowLayout：

```
panel2.setLayout(new FlowLayout(FlowLayout.RIGHT));
```

BorderLayout()边框布局管理器，先将“容器”设置为“边框布局控制模式”，当调用setLayout方法为容器设置布局控制模式时，参数设置为BorderLayout。例如：

```
JPanel panel1=(JPanel)getContentPane();
panel1.setLayout(new BorderLayout());
```

然后，可以在使用容器的add方法添加部件时，附加上位置参数，使得该部件显示在指定的位置上。位置参数分别是：

BorderLayout.NORTH——位置为北

BorderLayout.SOUTH——位置为南

BorderLayout.EAST——位置为东

BorderLayout.WEST——位置为西

BorderLayout.CENTER——位置为中心

计算器的整体面板布局使用BorderLayout：

```
panel.add(mainMenu, BorderLayout.NORTH);
panel.add(textAnswer, BorderLayout.CENTER);
panel.add(panel1, BorderLayout.SOUTH);
panel.setLayout(new BorderLayout());
```

GridLayout()，网格布局管理器GridLayout (int rows,int cols)，指定行数和列数，整个容器分配成行数*列数个单元。

计算器的数字面板和运算符布局使用GridLayout：

```
panel3.setLayout(new GridLayout(4, 6));
```

GridBagLayout()是Java语言中最强大的布局管理器，它不像网格布局一样，需要所有的部件的大小、形状相同。而且还可以将某一个部件放在一个固定的位置上。

程序源代码与解释

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DecimalFormat;

public class Calculator implements ActionListener { //导入动作监听接口
//设计面板中的单位
    JFrame frame;
    JTextField textAnswer;
    JPanel panel, panel1, panel2, panel3;
    JMenuBar mainMenu;
    JTextField textMemory;
    JLabel labelMemSpace; //控制面板的形状
    JButton buttonBk, buttonCe, buttonC;
    JButton button[];
    JButton buttonMC, buttonMR, buttonMS, buttonMAdd;
    JButton buttonDot, buttonAddAndSub, buttonAdd, buttonSub, buttonMul,
           buttonDiv, buttonMod;
    JButton buttonSqrt, buttonDao, buttonEqual;
    JMenu editMenu, viewMenu, helpMenu;
    JMenuItem copyItem, pasteItem, tItem, sItem, numberGroup, topHelp,
              aboutCal;
    DecimalFormat df; //设置数据输出精度
    boolean clickable; //控制当前能否按键
    double memoryd; //使用内存中存储的数字
    int memoryi;
    double vard, answerd; //用来保存double型数据的中间值(vard)和最后结果(answerd)
    short key = -1, prekey = -1; //key用来保存当前进行何种运算， prekey用来保存前次
    进行何种运算
    String copy; //做复制用
    JTextArea help; //帮助
    JScrollPane scrollHelp;
    //构造函数
    public Calculator() {
        clickable = true;
        answerd = 0;
        frame = new JFrame("计算器");
```

```

df = new DecimalFormat("0##########"); //设置数据输出精度
(对于double型值)

textAnswer = new JTextField(15);
textAnswer.setText("");
textAnswer.setEditable(false);
textAnswer.setBackground(new Color(255, 255, 255));
panel = new JPanel();
frame.getContentPane().add(panel);
panel1 = new JPanel();
panel2 = new JPanel();
panel.setLayout(new BorderLayout());
//设计整个面板
mainMenu = new JMenuBar();
panel.add(mainMenu, BorderLayout.NORTH);
panel.add(textAnswer, BorderLayout.CENTER);
panel.add(panel1, BorderLayout.SOUTH);
panel1.setLayout(new BorderLayout());
/*
.....代码省略，添加按键
*/
panel1.add(panel2, BorderLayout.NORTH);
panel2.setLayout(new FlowLayout(FlowLayout.RIGHT));
/*
.....代码省略，添加按键
*/
panel3 = new JPanel();
panel1.add(panel3, BorderLayout.CENTER);
button = new JButton[10];
/*
.....代码省略
*/
//建立各个按键，并设置前景色的监听
//将所有行为与监听绑定
panel3.setLayout(new GridLayout(4, 6));
panel3.add(buttonMC);
/*
.....代码省略
*/
//设置各个按键的监听
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.pack();
frame.show();

```

```

        }

    //设置各个按钮行为
    public void actionPerformed(ActionEvent event) {
        boolean sign = false; //判断是否是double型数参与运算，是为true，不是为false
        Object temp = event.getSource();
        try {
            //如果按下数据按钮，将按下的按钮代表的数据插入的当前文本框字符串
           之后
            for (int i = 0; i <= 9; i++)
                if (temp == button[i] && clickable == true)
                    textAnswer.setText(textAnswer.getText()
                        + Integer.toString(i));
            //按下"."按钮时，判断当前文本框内字符串中含不含".", 如果已含，则不允许再插入"."
            if (temp == buttonDot && clickable == true) {
                boolean isDot = false;
                if (textAnswer.getText().length() == 0)
                    isDot = true;
                for (int i = 0; i < textAnswer.getText().length(); i++)
                    if ('.' == textAnswer.getText().charAt(i)) {
                        isDot = true;
                        break;
                    }
                if (isDot == false)
                    textAnswer.setText(textAnswer.getText() + ".");
            }
            if ((temp == buttonAdd || temp == buttonSub || temp == buttonMul || temp
            == buttonDiv)
                && clickable == true) {
                //"+操作
                if (temp == buttonAdd) {
                    switch (prekey) {
                    case 0:
                        answerd += Double.parseDouble(textAnswer.getText());
                        break;
                    case 1:
                        answerd -= Double.parseDouble(textAnswer.getText());
                        break;
                    case 2:
                        answerd *= Double.parseDouble(textAnswer.getText());
                        break;
                    case 3:
                }
            }
        }
    }
}

```

```

        if (Double.parseDouble(textAnswer.getText()) == 0) {
            textAnswer.setText("除数不能为零");
            clickable = false;
        } else
            answerd /= Double.parseDouble(textAnswer.getText());
        break;
    default:
        answerd = Double.parseDouble(textAnswer.getText());
    }
    textAnswer.setText("");
    prekey = key = 0;
}
//"-操作
if (temp == buttonSub) {
    switch (prekey) {
/*
.....代码省略
*/
    prekey = key = 1;
}
//"*"操作
if (temp == buttonMul) {
    switch (prekey) {
/*
.....代码省略
*/
    prekey = key = 2;
}
//"/"操作
if (temp == buttonDiv) {
    switch (prekey) {
/*
.....代码省略
*/
    prekey = key = 3;
}
}
// "="操作
if (temp == buttonEqual && clickable == true) {
    //如果连续按"=",则进行连续运算
    if (prekey == 5) {
        if (key == 0) {

```

```

        answerd += vard;
        textAnswer.setText(df.format(answerd));
    }
    if (key == 1) {
        answerd -= vard;
        textAnswer.setText(df.format(answerd));
    }
    if (key == 2) {
        answerd *= vard;
        textAnswer.setText(df.format(answerd));
    }
    if (key == 3) {
        if (Double.parseDouble(textAnswer.getText()) == 0) {
            textAnswer.setText("除数不能为零");
            clickable = false;
        } else {
            answerd /= vard;
            textAnswer.setText(df.format(answerd));
        }
    }
} else {
/*
.....代码省略
*/
}
prekey = 5;
}
//%"操作，对第二个操作数除以100
if (temp == buttonMod && clickable == true) {
    if (answerd == 0) {
        String s = textAnswer.getText();
        textAnswer.setText(s);
    } else {
        boolean isDot = false;
        for (int i = 0; i < textAnswer.getText().length(); i++) {
            if ('.' == textAnswer.getText().charAt(i)) {
                isDot = true;
                break;
            }
        }
        //如果是double数，除100
        if (isDot == true) {

```

```

        double dtemp = Double.parseDouble(textAnswer.getText());
        dtemp = dtemp / 100.0;
        textAnswer.setText(Double.toString(dtemp));
    } else {
        //如果是int数但能被100整除，则去掉末尾两个零
        if (Integer.parseInt(textAnswer.getText()) % 100 == 0) {
            int itemp = Integer.parseInt(textAnswer.getText());
            itemp /= 100;
            textAnswer.setText(Integer.toString(itemp));
        }
        //如果是int数，但不能被100整除，则按double数处理
        else {
            double dtemp = Double.parseDouble(textAnswer
                .getText());
            dtemp = dtemp / 100.0;
            textAnswer.setText(Double.toString(dtemp));
        }
    }
}
}

//开根号运算
if (temp == buttonSqrt && clickable == true) {
    String s = textAnswer.getText();
    if (s.charAt(0) == '-') {
        textAnswer.setText("负数不能开根号");
        clickable = false;
    } else
        textAnswer.setText(Double.toString(java.lang.Math
            .sqrt(Double.parseDouble(textAnswer.getText()))));
}
}

//倒数运算
if (temp == buttonDuo && clickable == true) {
    if (textAnswer.getText().charAt(0) == '0'
        && textAnswer.getText().length() == 1) {
        textAnswer.setText("零不能求倒数");
        clickable = false;
    } else {
        /*
         * .....代码省略
        */
    }
}
}

//按下"+/-"按钮时处理

```

```

        if (temp == buttonAddAndSub && clickable == true) {
            boolean isNumber = true;
            String s = textAnswer.getText();
            for (int i = 0; i < s.length(); i++)
                if (!(s.charAt(i) >= '0' && s.charAt(i) <= '9'
                    || s.charAt(i) == '.' || s.charAt(i) == '-')) {
                    isNumber = false;
                    break;
                }
            if (isNumber == true) {
                //如果当前字符串首字母有‘-’号，代表现在是个负数，再按下时。
                //则将首符号去掉
                if (s.charAt(0) == '-') {
                    textAnswer.setText("");
                    for (int i = 1; i < s.length(); i++) {
                        char a = s.charAt(i);
                        textAnswer.setText(textAnswer.getText() + a);
                    }
                }
                //如果当前字符串第一个字符不是符号，则添加一个符号在首字母处
                else
                    textAnswer.setText('-' + s);
            }
        }
        //计算器有关内存操作
        //‘MC’的操作，将内存清0
        if (temp == buttonMC && clickable == true) {
            memoryd = memoryi = 0;
            textMemory.setText("");
        }
        //‘MS’的操作，将当前文本框内容保存入内存，显示‘M’
        if (temp == buttonMS && clickable == true) {
            boolean isDot = false;
            textMemory.setText(" M");
            for (int i = 0; i < textAnswer.getText().length(); i++)
                if ('.' == textAnswer.getText().charAt(i)) {
                    isDot = true;
                    break;
                }
            //如果是double，则存入memoryd（double存储器）
            if (isDot == true) {
                memoryd = Double.parseDouble(textAnswer.getText());
            }
        }
    }
}

```

```

        memoryi = 0; //保证存储器中存放最新的值
    }
    //如果是int,则存入memoryi(int存储器)
    else {
        memoryi = Integer.parseInt(textAnswer.getText());
        memoryd = 0; //保证存储器中存放最新的值
    }
}

//MR'的操作，将存储器中的信息输出
if (temp == buttonMR && clickable == true) {
    if (memoryd != 0)
        textAnswer.setText(Double.toString(memoryd));
    if (memoryi != 0)
        textAnswer.setText(Integer.toString(memoryi));
}

//M+'的功能，将当前文本框里的数据和存储器中数据相加后，再存入存储器
if (temp == buttonMAdd && clickable == true) {
    boolean isDot = false;
    for (int i = 0; i < textAnswer.getText().length(); i++) {
        if ('.' == textAnswer.getText().charAt(i)) {
            isDot = true;
            break;
        }
    }
    if (memoryi != 0) { //存储中是一个int型数
        if (isDot == false) //被加数是一个int型数
            memoryi += Integer.parseInt(textAnswer.getText());
        else { //被加数是一个double型数，则将int存储器中数传入
            double存储器与当前数相加，int存储器清零
            memoryd = memoryi
                + Double.parseDouble(textAnswer.getText());
            memoryi = 0;
        }
    } else
        memoryd += Double.parseDouble(textAnswer.getText());
}

//按下'Backspace'键，利用循环将当前字符串中的最后一个字母删除
if (temp == buttonBk && clickable == true) {
    String s = textAnswer.getText();
    textAnswer.setText("");
    for (int i = 0; i < s.length() - 1; i++) {
        char a = s.charAt(i);
        textAnswer.setText(textAnswer.getText() + a);
    }
}

```

```

        }
    }
    //按下'CE'按钮，将当前文本框内数据清除
    if (temp == buttonCe) {
        textAnswer.setText("");
        clickable = true;
    }
    //按下'C'按钮，文本框内数据清除，同时var,answer清0
    if (temp == buttonC) {
        vard = answerd = 0;
        textAnswer.setText("");
        clickable = true;
    }
}
//输入中如果有操作非法，比如按下两次'+', 捕获异常
catch (Exception e) {
    textAnswer.setText("操作非法");
    clickable = false;
}
}
//主函数
public static void main(String args[]) {
    new Calculator();
}
}
}

```



案例8 多线程断点续传

案例运行效果与操作

本案例实现网络文件传输，在客户端请求Web服务器传输指定文件，并将文件保存。程序运行后，控制台信息如图3-9所示。

同时，下载的文件已经存储在指定路径上。

制作要点

1. Java.net包的使用
2. IO包中的RandomAccessFile类的用法
3. 多线程的应用



图3-9 控制台的信息

步骤详解

1. 文件断点的记录。

```
URL url = new URL(sURL);
HttpURLConnection httpConnection = (HttpURLConnection)url.openConnection();
```

设置User-Agent:

```
httpConnection.setRequestProperty("User-Agent","NetFox");
String sProperty = "bytes="+nStartPos+"-";
```

设置断点续传的开始位置:

```
httpConnection.setRequestProperty("RANGE",sProperty);
Utility.log(sProperty);
```

获得输入流:

```
InputStream input = httpConnection.getInputStream();
```

2. 保存下载文件。

采用IO包中的RandomAccessFile类保存获得的文件，首先定位文件指针，然后写到文件中：

```
oSavedFile = new RandomAccessFile(sName,"rw");
this.nPos = nPos;
oSavedFile.seek(nPos);
```

RandomAccessFile还定义了DataInput和DataOutput的接口，通过这两个接口可以将数值

从文件中读出或写入文件。

程序源代码与解释

```
//Java实现网络文件传输，在客户端请求Web服务器传输指定文件，并将文件保存。  
  
import javax.swing.UIManager;  
import java.awt.*;  
import java.io.DataOutputStream;  
import java.net.HttpURLConnection;  
import java.io.IOException;  
import java.io.FileInputStream;  
import java.net.URL;  
import java.io.File;  
import java.io.DataInputStream;  
import java.io.FileOutputStream;  
import java.io.InputStream;  
import java.io.Serializable;  
import java.io.RandomAccessFile;  
  
public class HttpDownload {  
    private String sWebAddr = "http://kent.dl.sourceforge.net/sourceforge/jamper/Sample.zip";  
    //定义Web地址和文件名  
    //例如，传输http://cs.bnu.edu.cn/source/vb/地址的vb.zip则sWebAddr = "http://cs.bnu.edu.cn/source/vb/vb.zip"  
    private String sSavePath = "d:\\temp";           //定义存放文件的路径  
    private String sSaveName = "sample.zip";         //定义文件名  
  
    class CatchFile extends Thread {    //传输文件线程类  
        Catchinfo siteInfoBean = null; //文件信息Bean  
        long[] nPos;  
        long[] nStartPos; //开始位置  
        long[] nEndPos; //结束位置  
        PartCatch[] partCacth; //子线程对象  
        long nFileLength; //文件长度  
        boolean bFirst = true; //是否第一次取文件  
        boolean bStop = false; //停止标志  
        File tmpFile; //文件传输临时信息  
        DataOutputStream output; //输出到文件的输出流  
  
        public CatchFile(Catchinfo bean) throws IOException{  
            siteInfoBean = bean;  
            tmpFile = new File(bean.getSFilePath()+File.separator + bean.getSFileName() +  
".info");
```

```

if(tmpFile.exists()){
    bFirst = false;
    read_nPos();
}
else{
    nStartPos = new long[bean.getNSplitter()];
    nEndPos = new long[bean.getNSplitter()];
}
}

public void run(){
    //获得文件长度，分割文件
    try{
        if(bFirst){
            nFileLength = getFileSize();
            if(nFileLength == -1){
                System.err.println("File Length is not known!");
            }
            else
                if(nFileLength == -2){
                    System.err.println("File is not access!");
                }
                else{
                    for(int i=0;i<nStartPos.length;i++){
                        nStartPos[i] = (long)(i*(nFileLength/nStartPos.length));
                    }
                    for(int i=0;i<nEndPos.length-1;i++){
                        nEndPos[i] = nStartPos[i+1];
                    }
                    nEndPos[nEndPos.length-1] = nFileLength;
                }
        }
        //启动子线程
        partCacth = new PartCatch[nStartPos.length];
        for(int i=0;i<nStartPos.length;i++){
            partCacth[i] = new PartCatch(siteInfoBean.getSSiteURL(),
                siteInfoBean.getSFilePath() + File.separator + siteInfoBean.getSFileName(),
                nStartPos[i],nEndPos[i],i);
            Additon.log("Thread " + i + " , nStartPos = " + nStartPos[i] + ", nEndPos =
" + nEndPos[i]);
            partCacth[i].start();
        }
        //等待子线程结束,int count = 0;是否结束while循环
    }
}

```

```

        boolean breakWhile = false;
        while(!bStop){
            write_nPos();
            Additon.sleep(500);
            breakWhile = true;
            for(int i=0;i<nStartPos.length;i++){
                if(!partCacth[i].bDownOver){
                    breakWhile = false;
                    break;
                }
            }
            if(breakWhile)
                break;
        }
        System.out.println("文件传输结束！");
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

//获得文件长度
public long getFileSize(){
    int nFileLength = -1;
    try{
        URL url = new URL(siteInfoBean.getSSiteURL());
        HttpURLConnection httpConnection = (HttpURLConnection)url.openConnection();
        httpConnection.setRequestProperty("User-Agent","NetFox");
        int responseCode=httpConnection.getResponseCode();
        if(responseCode>=400){
            processErrorCode(responseCode);
            return -2; // -2为Web服务器响应错误
        }
        String sHeader;
        for(int i=1;;i++){
            sHeader=httpConnection.getHeaderFieldKey(i);
            if(sHeader!=null){
                if(sHeader.equals("Content-Length")){
                    nFileLength = Integer.parseInt(httpConnection.getHeaderField(sHeader));
                    break;
                }
            }
        }
    }
}

```

```

        else
            break;
    }
}

catch(IOException e){e.printStackTrace();}
catch(Exception e){e.printStackTrace();}

Additon.log(nFileLength);

return nFileLength;
}

//保存传输信息(文件指针位置)
private void write_nPos(){
    try{
        output = new DataOutputStream(new FileOutputStream(tmpFile));
        output.writeInt(nStartPos.length);
        for(int i=0;i<nStartPos.length;i++){
            // output.writeLong(nPos[i]);
            output.writeLong(partCacth[i].nStartPos);
            output.writeLong(partCacth[i].nEndPos);
        }
        output.close();
    }
    catch(IOException e){e.printStackTrace();}
    catch(Exception e){e.printStackTrace();}
}

//读取保存的下载信息(文件指针位置)
private void read_nPos(){
    try{
        DataInputStream input = new DataInputStream(new FileInputStream(tmpFile));
        int nCount = input.readInt();
        nStartPos = new long[nCount];
        nEndPos = new long[nCount];
        for(int i=0;i<nStartPos.length;i++){
            nStartPos[i] = input.readLong();
            nEndPos[i] = input.readLong();
        }
        input.close();
    }
    catch(IOException e){e.printStackTrace();}
    catch(Exception e){e.printStackTrace();}
}

private void processErrorCode(int nErrorCode){
}

```

```

        System.err.println("Error Code : " + nErrorCode);
    }

    //停止文件传输
    public void siteStop(){
        bStop = true;
        for(int i=0;i<nStartPos.length;i++)
            partCacth[i].partStop();
    }
}

class PartCatch extends Thread {
    String sURL; //定义文件传输时使用的变量
    long nStartPos; //分段文件传输开始位置
    long nEndPos; //分段文件传输结束位置
    int nThreadID; //子线程ID
    boolean bDownOver = false; //完成文件传输
    boolean bStop = false; //停止文件传输
    StoreFile fileAccess = null;

    public PartCatch(String sURL,String sName,long nStart,long nEnd,int id) throws
    IOException{
        this.sURL = sURL;
        this.nStartPos = nStart;
        this.nEndPos = nEnd;
        nThreadID = id;
        fileAccess = new StoreFile(sName,nStartPos);
    }

    public void run(){
        while(nStartPos < nEndPos && !bStop){
            try{
                URL url = new URL(sURL);
                HttpURLConnection httpConnection = (HttpURLConnection)url.openConnection ();
                httpConnection.setRequestProperty("User-Agent","NetFox");
                String sProperty = "bytes="+nStartPos+"-";
                httpConnection.setRequestProperty("RANGE",sProperty);
                Additon.log(sProperty);
                InputStream input = httpConnection.getInputStream();
                byte[] b = new byte[1024];
                int nRead;
                nStartPos=200;
                while
                    ((nRead=input.read(b,0,1024)) > 0 && nStartPos < nEndPos && !bStop)
                {

```

```

        nStartPos += fileAccess.write(b,0,nRead);
    }
    Additon.log("Thread " + nThreadID + " is over!");
    bDownOver = true;
}
catch(Exception e){
    e.printStackTrace();
}
}

public void logResponseHead(HttpURLConnection con){
for(int i=1;i++){
    String header=con.getHeaderFieldKey(i);
    if(header!=null)
        Additon.log(header+" : "+con.getHeaderField(header));
    else
        break;
}
}

public void partStop(){
    bStop = true;
}
}

class StoreFile implements Serializable{ //定义访问文件类
    RandomAccessFile oSavedFile;
    long nPos;

    public StoreFile() throws IOException{
        this("",0);
    }

    public StoreFile(String sName,long nPos) throws IOException{
        oSavedFile = new RandomAccessFile(sName,"rw");
        this.nPos = nPos;
        oSavedFile.seek(nPos);
    }

    public synchronized int write(byte[] b,int nStart,int nLen){
        int n = -1;
        try{
            oSavedFile.write(b,nStart,nLen);
            n = nLen;
        }
    }
}

```

```

        }
        catch(IOException e){
        }
        return n;
    }
}

public HttpDownload(){
    try{
        Catchinfo bean = new Catchinfo(sWebAddr,sSavePath,sSaveName,5);
        CatchFile fileFetch = new CatchFile(bean);
        fileFetch.start();
    }
    catch(Exception e){
        e.printStackTrace ();
    }
}

public static void main(String[] args){
    new HttpDownload();
}
}

class Catchinfo {      //定义获取和设置相关文件信息类
    private String sSiteURL;    //定义URL变量
    private String sFilePath; //定义存文件路径变量
    private String sFileName; //定义文件名变量
    private int nSplitter; //定义传输文件计数值

    public Catchinfo(){
        this("", "", "", 5);      //设置传输文件计数值
    }

    public Catchinfo(String sURL, String sPath, String sName, int nSplitter){
        sSiteURL = sURL;
        sFilePath = sPath;
        sFileName = sName;
        this.nSplitter = nSplitter;
    }

    public String getSSiteURL(){
        return sSiteURL;
    }

    public void setSSiteURL(String value){
        sSiteURL = value;
    }
}

```

```

public String getSFilePath(){
    return sFilePath;
}
public void setSFilePath(String value){
    sFilePath = value;
}
public String getSFileName(){
    return sFileName;
}
public void setSFileName(String value){
    sFileName = value;
}
public int getNSplitter(){
    return nSplitter;
}
public void setNSplitter(int nCount){
    nSplitter = nCount;
}

```



案例9 笛卡儿曲线

案例运行效果与操作

非常有名的笛卡儿数学公式 $(x^2 + y^2 - 2ax)^2 = 4a^2(x^2 + y^2)$ ，即心形曲线，由Java画出。如图3-10所示。

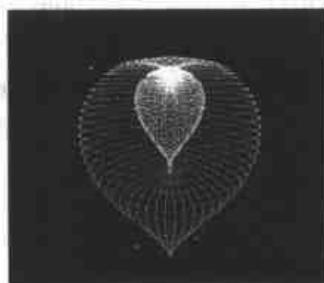


图3-10 心形曲线

制作要点

1. `getGraphics()`方法的使用
2. `CreateImage()`方法的用法

步骤详解

笛卡儿数学公式

笛卡儿心形线是一个圆在一个同样半径的圆的圆周上滚动，其上的一定点在滚动过程中的轨迹曲线。

它的数学方程：

$$x = a(2\cos(t) - \cos(2t))$$

$$y = a(2\sin(t) - \sin(2t))$$

$$\text{极坐标方程: } r = 2a(1 + \cos(\theta))$$

算法实现：

```
r=Math.PI/45*i*(1-Math.sin(Math.PI/45*j))*18;
x=r*Math.cos(Math.PI/45*j)*Math.sin(Math.PI/45*i)+AppletWidth/2;
y=-r*Math.sin(Math.PI/45*j)+AppletHeight/4;
```

程序源代码与解释

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;

public class Descartes extends Applet {
    int AppletWidth, AppletHeight;
    Image OffScreen;
    Graphics drawOffScreen;
    public void init() {
        setBackground(Color.black); //设置背景为黑色
        AppletWidth = getSize().width; //获取applet的宽度
        AppletHeight = getSize().height; //获取applet的高度
        OffScreen = createImage(AppletWidth, AppletHeight);
        //创建一幅使用了双缓冲技术的不可见的画布。
        drawOffScreen = OffScreen.getGraphics();
    }
    public void paint(Graphics g) {
        //设置前景色为白色
        drawOffScreen.clearRect(0, 0, AppletWidth, AppletHeight);
        drawOffScreen.setColor(Color.white);
        int i, j;
        double x, y, r;
        for (i = 0; i <= 90; i++)
```

```
for (j = 0; j <= 90; j++) {
    r = Math.PI / 45 * i * (1 - Math.sin(Math.PI / 45 * j)) * 18;
    x = r * Math.cos(Math.PI / 45 * j) * Math.sin(Math.PI / 45 * i)
        + AppletWidth / 2;
    y = -r * Math.sin(Math.PI / 45 * j) + AppletHeight / 4;
    drawOffScreen.fillOval((int) x, (int) y, 2, 2);
    //绘制椭圆形
}
g.drawImage(OffScreen, 0, 0, this);
//生成图片
}
}
```

本 章 小 结

建立Web应用程序的最佳方式是使之成为三层应用程序，从而巧妙地区分出三个组成部分：用户界面、计算逻辑与数据存储，而Java与XML的组合提供建立三层应用程序的最佳手段。可以说，Java语言与XML的结合具有广泛而深远的意义，这方面的知识是当今程序设计人员必须掌握的。

第4章

Java与游戏



本章内容

- 案例1 幸运52游戏
- 案例2 “速算24”扑克游戏
- 案例3 拼图游戏
- 案例4 贪吃蛇游戏
- 案例5 打球游戏
- 案例6 棒打猪头
- 本章小结



案例1 幸运52游戏

案例运行效果与操作

幸运52游戏是让用户对物品的价格进行估测。在对价格进行估测的时候，系统会给出用户估测的价格是高还是低的信息，用户根据这些信息重新调整所估测的价格。游戏的初始界面如图4-1所示。

单击“开始游戏”按钮，网页就会自动调出商品的图像，如图4-2所示为刚刚开始的一盘游戏，其中的商品是IXUS50的数码相机。

在输入框中输入对该商品价格的估测，然后单击“确定”按钮，这时会弹出一个对话框，告诉你猜的价格是高了还是低了，如图4-3所示。

如果你猜中了，则对话框会给出“恭喜你”的消息，如图4-4所示。

幸运52游戏

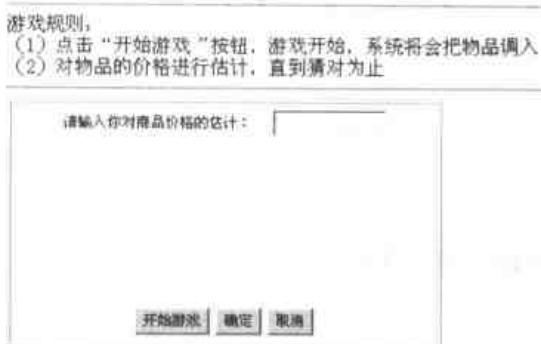


图4-1 游戏的初始界面



图4-2 开始游戏

幸运52游戏



图4-3 猜不中的界面

幸运52游戏



图4-4 猜中后的界面

制作要点

- 类的继承
- Panel类的方法，创建MyImgPanel类来显示图片
- Frame类的应用，显示对话框
- 事件处理，实现监听器接口

步骤详解

- 编写HTML页面。

```
<html>
<HEAD>
<TITLE>
幸运52游戏
</TITLE>
</HEAD>
<BODY>
```

```
<h2>幸运52游戏</h2>
<hr>
游戏规则: <br>
(1) 单击“开始游戏”按钮, 游戏开始, 系统将会把物品调入<br>
(2) 对物品的价格进行估计, 直到猜对为止<br>
<hr>
<TABLE border=1>
<TR>
<TD>
<APPLET CODE="Lucky52.class" WIDTH="350" HEIGHT="200">
</APPLET>
</TD>
</TR>
</TABLE>
</BODY>
</html>
```

2. 编写一个用来显示图片的Panel。

Java中不提供直接用来显示图片的Panel, 所以, 要编写一个用来显示图片的类。这个类从Panel类中继承, 它具有Panel类的一切特征, 还能用来显示图片。

```
String fileName = "IXUS50.jpg"; //要显示的图片名称
Image myImage; //要显示的图片对象
```

把图片从文件中装载到myImage对象中去:

```
URL url = null; // 创建URL对象
try {
    url = Class.forName("Lucky52").getResource("IXUS50.jpg");
} catch (Exception e) {
}
myImage = getToolkit().getImage(url);
MediaTracker myTracker = new MediaTracker(this);
myTracker.addImage(myImage, 1);
```

重载Panel以显示图片:

```
g.drawImage(myImage, 135, 30, 55, 75, this);
```

3. 编写Applet主界面。

```
Panel pnlNorth = new Panel();
MyImgPanel pnlCenter = new MyImgPanel();
Panel pnlBottom = new Panel();
TextField txtField = new TextField(10);
```

```
Label label = new Label("请输入你对商品价格的估计: ");
private int truePrice = 3250;
```

采用BorderLayout布局:

```
super();
this.setLayout(new BorderLayout());
pnlNorth.add(label);
pnlNorth.add(txtField);
add(pnlNorth, BorderLayout.NORTH);
add(pnlCenter, BorderLayout.CENTER);
Button btnStart = new Button("开始游戏");
Button btnOk = new Button("确定");
Button btnCancel = new Button("取消");
btnStart.setActionCommand("start");
btnStart.addActionListener(this);
btnOk.setActionCommand("ok");
btnOk.addActionListener(this);
btnCancel.setActionCommand("cancel");
btnCancel.addActionListener(this);
pnlBottom.add(btnStart);
pnlBottom.add(btnOk);
pnlBottom.add(btnCancel);
add(pnlBottom, BorderLayout.SOUTH);
setBackground(Color.white);
```

4. 编写弹出式对话框。

通过继承Frame类来构建一个对话框，用来显示所输入的价格与实际价格的关系。设置Frame的大小、显示时所在的位置，并显示：

```
setTitle("猜测结果显示");
Panel myPanel = new Panel();
add(myPanel);
myPanel.add(label);
label.setText(strMsg);
setSize(150, 100);           //设置对话框大小
setLocation(300, 200);        //设置Frame显示时所在的位置
Button btnOk = new Button("确定");
btnOk.addActionListener(this);
myPanel.add(btnOk);
show();                      //显示对话框
```

5. 添加对事件的处理。

通过监听器来实现对事件的处理。首先在Applet类中实现ActionListener的接口，用来

处理ActionEvent事件。当产生ActionEvent事件时，就会调用接口中定义的ActionPerformed方法。然后调用组件的addActionListener()方法来安装监听器，用setActionCommand（String对象）来区分不同组件产生的ActionEvent事件。

程序源代码与解释

```
import java.applet.*;
import java.awt.*;
import java.net.*;
import java.awt.event.*;

//实现ActionListener监听器接口，用来处理ActionEvent事件
public class Lucky52 extends Applet implements ActionListener {
    //声明类的成员变量，以便在事件处理时改变其属性
    Panel pnlNorth = new Panel();
    MyImgPanel pnlCenter = new MyImgPanel();
    Panel pnlBottom = new Panel();
    TextField txtField = new TextField(10);
    Label label = new Label("请输入你对商品价格的估计：");
    private int truePrice = 3250;

    public Lucky52() {
        super();
        this.setLayout(new BorderLayout());           //采用BorderLayout布局
        pnlNorth.add(label);
        pnlNorth.add(txtField);
        add(pnlNorth, BorderLayout.NORTH);
        add(pnlCenter, BorderLayout.CENTER);
        Button btnStart = new Button("开始游戏");
        Button btnOk = new Button("确定");
        Button btnCancel = new Button("取消");
        btnStart.setActionCommand("start");
        btnStart.addActionListener(this);             //安装事件监听器
        btnOk.setActionCommand("ok");
        btnOk.addActionListener(this);                //安装事件监听器
        btnCancel.setActionCommand("cancel");
        btnCancel.addActionListener(this);            //安装事件监听器
        pnlBottom.add(btnStart);
        pnlBottom.add(btnOk);
        pnlBottom.add(btnCancel);
        add(pnlBottom, BorderLayout.SOUTH);
    }

    public void actionPerformed(ActionEvent e) {
        if ("start".equals(e.getActionCommand())) {
            String str = txtField.getText();
            if (str == null || str.length() < 1) {
                JOptionPane.showMessageDialog(this, "请输入商品的价格");
            } else {
                try {
                    int num = Integer.parseInt(str);
                    if (num < 0) {
                        JOptionPane.showMessageDialog(this, "请输入正确的商品价格");
                    } else if (num > truePrice) {
                        JOptionPane.showMessageDialog(this, "你输入的商品价格过高");
                    } else if (num < truePrice) {
                        JOptionPane.showMessageDialog(this, "你输入的商品价格过低");
                    } else {
                        JOptionPane.showMessageDialog(this, "恭喜你，猜对了！");
                    }
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(this, "请输入正确的商品价格");
                }
            }
        } else if ("ok".equals(e.getActionCommand())) {
            String str = txtField.getText();
            if (str == null || str.length() < 1) {
                JOptionPane.showMessageDialog(this, "请输入商品的价格");
            } else {
                try {
                    int num = Integer.parseInt(str);
                    if (num < 0) {
                        JOptionPane.showMessageDialog(this, "请输入正确的商品价格");
                    } else if (num > truePrice) {
                        JOptionPane.showMessageDialog(this, "你输入的商品价格过高");
                    } else if (num < truePrice) {
                        JOptionPane.showMessageDialog(this, "你输入的商品价格过低");
                    } else {
                        JOptionPane.showMessageDialog(this, "恭喜你，猜对了！");
                    }
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(this, "请输入正确的商品价格");
                }
            }
        } else if ("cancel".equals(e.getActionCommand())) {
            System.exit(0);
        }
    }
}
```

```

        setBackground(Color.white);
    }

    //事件处理
    public void actionPerformed(ActionEvent evt) {
        if (evt.getActionCommand().equals("start")) {
            pnlCenter.initImg();
            label.setText("请输入你对商品价格的估计：");
            pnlCenter.repaint();
        } else if (evt.getActionCommand().equals("ok")) {
            int guessPrice = 0;
            try {
                guessPrice = Integer.parseInt(txtField.getText().trim());
                String guess = comparePrice(guessPrice);
                new MsgDlg(guess);
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else if (evt.getActionCommand().equals("cancel")) {
            txtField.setText("");
        }
    }

    //判断所猜价格与实际价格的关系
    public String comparePrice(int guessPrice) {
        if (guessPrice == truePrice) {
            return "猜对了，恭喜你！";
        } else if (guessPrice > truePrice) {
            return "猜的价格过高，请重新猜！";
        } else if (guessPrice < truePrice) {
            return "猜的价格过低，请再加价！";
        }
        return "出错了！";
    }
}

class MyImgPanel extends Panel {
    String fileName = "IXUS50.jpg"; //要显示的图片名称
    Image myImage; //要显示的图片对象
    public void initImg() { //把图片从文件中装载到myImage对象中去
        URL url = null; //创建URL对象
        try {

```

```
url = Class.forName("Lucky52").getResource("IXUS50.jpg");
} catch (Exception e) {
}
myImage = getToolkit().getImage(url);
MediaTracker myTracker = new MediaTracker(this);
myTracker.addImage(myImage, 1);
try {
    myTracker.waitForAll();
    myTracker.checkAll();
} catch (Exception e) {
}
}

public void paint(Graphics g) {          //重载Panel以显示图片
    g.drawImage(myImage, 135, 30, 55, 75, this);
}
}

//实现ActionListener监听器接口，用来处理.ActionEvent事件
class MsgDlg extends Frame implements ActionListener {
    Label label = new Label();

    public MsgDlg(String strMsg) {
        super();
        setTitle("猜测结果显示");
        Panel myPanel = new Panel();
        add(myPanel);
        myPanel.add(label);
        label.setText(strMsg);
        setSize(150, 100);           //设置Frame的大小
        setLocation(300, 200);       //设置Frame显示时所在的位置
        Button btnOk = new Button("确定");
        btnOk.addActionListener(this); //安装事件监听器
        myPanel.add(btnOk);
        show();                     //显示对话框
    }

    public void actionPerformed(ActionEvent evt) {
        this.dispose();           //隐藏对话框
    }
}
```



案例2 “速算24” 扑克游戏

案例运行效果与操作

“速算24”扑克游戏是一个锻炼玩家心算和快速反应能力的Applet游戏。在游戏给出4张牌后，玩家应迅速给出一个算式，使得计算结果等于24。游戏能够在玩家输入表达式之后判断出输入是否合法，计算结果是否为24，并给出相应的提示信息。游戏的初始界面如图4-5所示。



图4-5 游戏的初始界面

单击“开始游戏”按钮，游戏就会随机发出4张牌。图4-6所示为刚刚开始的一盘游戏，给出的4张纸牌分别为“5”、“Q”、“2”、“9”。

在输入框中输入计算表达式，单击“确定”按钮，此时如果输入非法，如 $5 + + 12 + 2 + 9$ ，这时就会弹出一个对话框，提示表达式不合法，要求重新输入，如图4-7所示。



图4-6 开始游戏

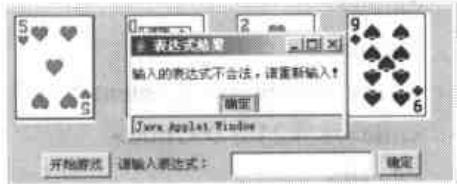


图4-7 输入的表达式不合法

如果输入合法，但数值不等于24，如 $5+12+2+9$ ，单击“确定”按钮，这时就会弹出一个对话框，提示输入的表达式结果，要求重新输入，如图4-8所示。

如果输入的表达式的值是24，如 $5+12-2+9$ ，单击“确定”按钮，这时就会弹出一个对话框，提示祝贺信息，并自动开始一盘新的游戏，如图4-9所示。

速算24游戏

游戏规则：

- (1) 点击“开始游戏”的按钮，游戏开始，系统将会发出四张牌。
- (2) 用户将发出的四张牌用+、-、×、/组合，输入到输入框里头。
- (3) 点击确定按钮，游戏将会计算你输入的表达式是否正确，并且给出提示。
- (4) 如果输入的表达式不正确，则会重新输入。



图4-8 输入结果错误

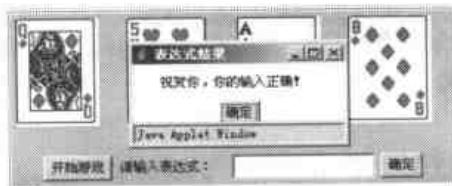


图4-9 输入正确

制作要点

1. Applet类的应用
2. Panel类的方法，创建MyImgPanel类来显示图片
3. Frame类的应用，显示对话框
4. 事件处理，实现监听器接口
5. 数组的应用技巧

步骤详解

1. 写HTML页面。

在页面里首先给出游戏规则，然后把Applet嵌入到一个表单中。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>速算24游戏</title>
</head>
<body>
<h2>速算24游戏</h2>
<body>
<hr>
游戏规则：<br>
(1) 单击“开始游戏”的按钮，游戏开始，系统将会发出四张牌<br>
```

(2) 用户将发出的四张牌用+,-,*,/组合起来，并把组合的表达式输入到输入框里头

 (3) 单击确定按钮，游戏将会计算你输入的表达式是否合法，或者结果是否正确，并且给出提示。

(4) 如果输入的表达式不正确，则会让用户重新输入；如果输入的表达式正确，则重新开始游戏。


```
<hr>
<center>
<TABLE border=1>
<TR>
<TD>
<APPLET CODE="Calculate24.class" WIDTH="400" HEIGHT="160">
</APPLET>
</TD>
</TR>
</TABLE>
</center>
</body>
</html>
```

2. 编写一个用来显示图片的Panel。

和本章案例1类似，本案例也要编写一个用来显示图片的Panel类。首先准备14张图片，分别为A~K的扑克牌正面图和一张背面图，进行装载。使用Graphic类的drawImage方法在面板上画上4张扑克牌，用myStatus数组保存当前4张牌的大小。设置图片尺寸、对象：

```
final int IMGSIZE = 100;
Image[] myImage = new Image[14];
Calculate24 mycal24;
```

装载图片：

```
//初始时装载背面图
url = Class.forName("Calculate24").getResource("img/back.JPG");
url = Class.forName("Calculate24").getResource("img/" + i + ".JPG");
```

在指定位置画出图片：

```
g.drawImage(myImage[mycal24.myStatus[i]], i * IMGSIZE + 5, 5, this);
```

3. 编写Applet主界面。

```
int[] myStatus = new int[4]; //定义一个数组
setLayout(new BorderLayout()); //采用BorderLayout布局
pnlCenter = new MyImgPanel(this);
```

```

pnlCenter.initImg();
add(pnlCenter, BorderLayout.CENTER);
pnlBottom = new Panel();
add(pnlBottom, BorderLayout.SOUTH);
Button btnStart = new Button("开始游戏");
Button btnOk = new Button("确定");
txtField = new TextField(15);
Label label = new Label("请输入表达式：");
pnlBottom.add(btnStart);
pnlBottom.add(label);
pnlBottom.add(txtField);
pnlBottom.add(btnOk);
btnStart.setActionCommand("start");
btnStart.addActionListener(this);           //安装事件监听器
btnOk.setActionCommand("ok");
btnOk.addActionListener(this);           //安装事件监听器

```

4. 增加对表达式处理的方法。

该方法返回类型为int，当表达式为非法时返回-1，否则返回表达式的值。

对表达式的处理分为三部分：

- 判断表达式的合法性
- 判断表达式的数字就是扑克牌上的数字
- 计算表达式的值

5. 添加对事件的处理。

通过监听器来实现对事件的处理。首先在Applet类中实现ActionListener的接口，用来处理ActionEvent事件。当产生ActionEvent事件时，就会调用接口中定义的ActionPerformed的方法。然后调用组件的addActionListener()方法来安装监听器，用setActionCommand（String对象）来区分不同组件产生的ActionEvent事件。

程序源代码与解释

```

import java.applet.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

//实现ActionListener监听器接口，用来处理ActionEvent事件
public class Calculate24 extends Applet implements ActionListener {
    //声明类的成员变量，以便在事件处理时改变其属性
    Panel pnlBottom;
    MyImgPanel pnlCenter;

```

```

TextField txtField;

//通过数组来保存当前4张牌的大小，并由此确定所画的扑克牌
int[] myStatus = new int[4];

public Calculate24() {
    setLayout(new BorderLayout());
    pnlCenter = new MyImgPanel(this);
    pnlCenter.initImg();
    add(pnlCenter, BorderLayout.CENTER);
    pnlBottom = new Panel();
    add(pnlBottom, BorderLayout.SOUTH);
    Button btnStart = new Button("开始游戏");
    Button btnOk = new Button("确定");
    txtField = new TextField(15);
    Label label = new Label("请输入表达式：");
    pnlBottom.add(btnStart);
    pnlBottom.add(label);
    pnlBottom.add(txtField);
    pnlBottom.add(btnOk);
    btnStart.setActionCommand("start");
    btnStart.addActionListener(this);           //安装事件监听器
    btnOk.setActionCommand("ok");
    btnOk.addActionListener(this);           //安装事件监听器
}

//对4张扑克牌的状态进行初始化
public void init() {
    for (int i = 0; i < 4; i++) {
        myStatus[i] = 0;
    }
}

//按钮事件处理
public void actionPerformed(ActionEvent evt) {
    //按下开始按钮
    if (evt.getActionCommand().equals("start")) {
        for (int i = 0; i < 4; i++) {
            myStatus[i] = (int) (Math.random() * 13) + 1; //随机产生4张牌
        }
        pnlCenter.repaint();
    }
    //按下确定按钮
    else if (evt.getActionCommand().equals("ok")) {
}

```

```

try {
    int result = calculateString(txtField.getText().trim());
    if (result == -1) {
        txtField.setText("");
        txtField.requestFocus();
        new MsgDlg("你输入的表达式不合法，请重新输入！");
    } else if (result != 24) {
        txtField.setText("");
        txtField.requestFocus();
        new MsgDlg("你输入的表达式的值为" + result + ",请重新输入!");
    } else if (result == 24) {
        txtField.requestFocus();
        new MsgDlg("祝贺你，你的输入正确！");
        for (int i = 0; i < 4; i++) {
            myStatus[i] = (int) (Math.random() * 13) + 1;
        }
        pnlCenter.repaint();
        txtField.setText("");
        txtField.requestFocus();
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

public int calculateString(String str) {
    //判断表达式的合法性
    int[] numbers = new int[4];
    String[] operators = new String[4];
    for (int i = 0; i < 4; i++) {
        operators[i] = "";
    }

    String tempStr, myString = "";
    int numberNo = 0, operatorNo = 0;
    for (int i = 0; i < str.length(); i++) {
        tempStr = str.substring(i, i + 1);
        if (isNumber(tempStr)) {
            myString += tempStr;
        } else if (isOperator(tempStr)) {
            if (numberNo >= 4 || operatorNo >= 3)
                return -1;
            try {

```

```

        numbers[numberNo] = Integer.parseInt(myString);
    } catch (Exception e) {
    }
    myString = "";
    numberNo++;
    operators[operatorNo] = tempStr;
    operatorNo++;
} else {
    return -1;
}
}
if (myString.length() != 0 && numberNo == 3) {
    try {
        numbers[numberNo] = Integer.parseInt(myString);
    } catch (Exception e) {
        return -1;
    }
} else
    return -1;
//判断表达式的数字就是扑克牌上的数字
int tempStatus[] = new int[4];
for (int i = 0; i < 4; i++) {
    tempStatus[i] = myStatus[i];
}
for (int i = 0; i < 4; i++) {
    int j = 0;
    boolean existed = false;
    while (j < 4 && !existed) {
        if (tempStatus[j] == numbers[i]) {
            tempStatus[j] = -1;
            existed = true;
        }
        j++;
    }
    if (!existed)
        return -1;
}
//计算表达式的值
int result = numbers[0];
for (int i = 0; i < 3; i++) {
    if (operators[i].equals("+")) {
        result += numbers[i + 1];
    }
}

```

```

        } else if (operators[i].equals("-")) {
            result -= numbers[i + 1];
        } else if (operators[i].equals("*")) {
            result *= numbers[i + 1];
        } else if (operators[i].equals("/")) {
            result /= numbers[i + 1];
        }
    }
    return result;
}

private boolean isNumber(String str) {
    if (str.equals("0") || str.equals("1") || str.equals("2")
        || str.equals("3") || str.equals("4") || str.equals("5")
        || str.equals("6") || str.equals("7") || str.equals("8")
        || str.equals("9")) {
        return true;
    } else {
        return false;
    }
}

private boolean isOperator(String str) {
    if (str.equals("+") || str.equals("-") || str.equals("*")
        || str.equals("/")) {
        return true;
    } else {
        return false;
    }
}

class MyImgPanel extends Panel {
    final int IMGSIZE = 100;
    Image[] myImage = new Image[14];
    Calculate24 mycal24;
    //装载图片
    public MyImgPanel(Calculate24 cal24) {
        mycal24 = cal24;
    }
    public void initImg() {
        URL url = null;
        try {
            url = Class.forName("Calculate24").getResource("img/back.JPG");
        }
    }
}

```

```

        } catch (Exception e) {
            e.printStackTrace();
        }
        myImage[0] = getToolkit().getImage(url);
        for (int i = 1; i <= 13; i++) {
            try {
                url = Class.forName("Calculate24").getResource(
                    "img/" + i + ".JPG");
                System.out.println(url.toString());
            } catch (Exception e) {
            }
            myImage[i] = getToolkit().getImage(url);
        }
        MediaTracker mt = new MediaTracker(this);
        for (int i = 0; i <= 13; i++) {
            mt.addImage(myImage[i], i);
        }
        try {
            mt.wait();
            mt.checkAll();
        } catch (Exception e) {
        }
    }
    public void paint(Graphics g) {
        for (int i = 0; i < 4; i++) {
            g.drawImage(myImage[mycal24.myStatus[i]], i * IMGSIZE + 5, 5, this);
        }
    }
}

class MsgDig extends Frame implements ActionListener {
    Label label = new Label();

    public MsgDig(String strMsg) {
        super();
        setTitle("表达式结果");
        Panel p = new Panel();
        add(p);
        p.add(label);
        label.setText(strMsg);
        setSize(200, 100);
        setLocation(400, 300);
        Button btnOk = new Button("确定");
    }
}

```

```

        btnOk.addActionListener(this);
        p.add(btnOk);
        show();           //显示对话框
    }

    public void actionPerformed(ActionEvent evt) {
        this.dispose();      //隐藏对话框
    }
}

```



案例3 拼 图 游 戏

案例运行效果与操作

拼图游戏也是一个Applet小程序。这个游戏将一张大图打散成9张小图，然后在游戏里任意挑选8张小图，贴在9个位置中的任意位置。通过鼠标和键盘来移动打乱的8张小图，让其还原成原来的顺序，玩家胜利，游戏就结束了。一打开页面，游戏即自动开始，在游戏过程中，游戏所用时间会实时显示在浏览器的状态栏。在游戏结束后，会算出玩家的得分。拼图游戏的初始界面如图4-10所示。

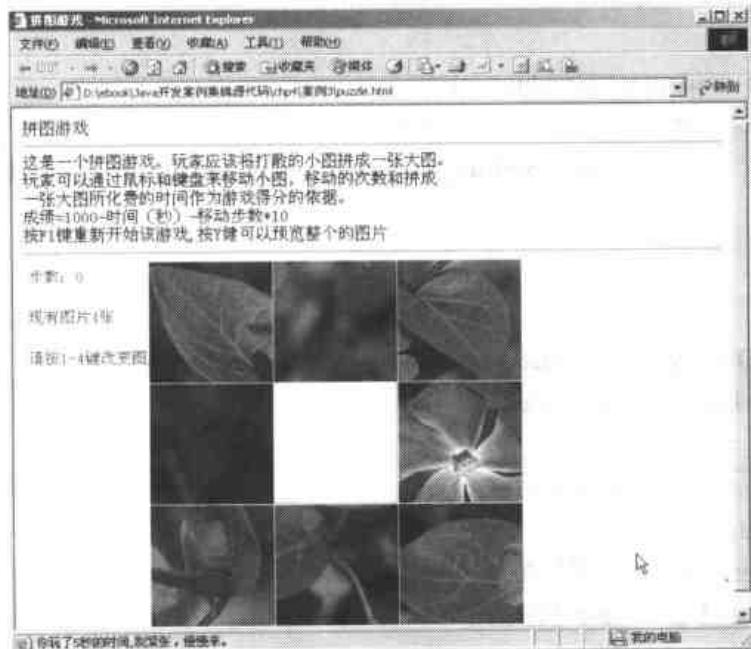


图4-10 游戏的初始界面

当玩家将各个小图的顺序排好之后，可以看到如图4-11所示的界面。



图4-11 游戏的结束界面

制作要点

1. Applet类的应用，使用getParameter方法获取param标记的值
2. Graphics对象的应用，利用getGraphics和drawImage等方法实现大图的分割
3. Math包中random方法的应用，使游戏每次初始化状态都不一样
4. 鼠标和键盘事件处理，实现监听器接口
5. 重载update方法，消除闪烁问题
6. 实现Runnable接口，创建多线程的应用

步骤详解

1. 嵌入Applet的HTML页面源码。

```

<Html>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>
    拼图游戏
</title>
</HEAD>
<body>
    拼图游戏
    <hr>
    这是一个拼图游戏。玩家应该将打散的小图拼成一张大图。<br>
    玩家可以通过鼠标和键盘来移动小图，移动的次数和拼成<br>
    一张大图所花费的时间作为游戏得分的依据。<br>
    成绩 = 1000 - 时间（秒） - 移动步数*10<br>

```

按F1键重新开始该游戏,按Y键可以预览整个的图片

```
<hr>
<applet code="puzzle.class" width="480" height="360">
</applet>
</body>
</html>
```

2. 编写Applet主界面。

1) 准备图片并进行装载

准备大小为 360×360 像素的图片, 将其分成 3×3 的小图, 每张小图的大小为 120×120 像素。使用数组来表示大图和9个小图。小图装载时, 首先用`createImage(int width, int height)`方法为每个小图的对象建立一个实例, 再用`getGraphics`方法得到它, 最后用`drawImage`方法将大图的某一个固定区域的图画到每个小图的图像上。

遍历9个小图对象, 用来装载其中的每个。

```
myImage[i]=createImage(IMAGE_WIDTH,IMAGE_HEIGHT);
```

创建实例, 为每个图片对象分配内存空间:

```
Graphics g=myImage[i].getGraphics();
```

获得`Graphics`对象, 算出该小图对象对应大图的哪一块区域, 并往小图上画。

```
g.drawImage(myImageAll,0,0,IMAGE_WIDTH,IMAGE_HEIGHT,nRow*IMAGE_WIDTH,
nCol*IMAGE_HEIGHT,(nRow+1)*IMAGE_WIDTH,(nCol+1)*IMAGE_HEIGHT,this);
```

2) 画出界面

让界面在左边留下一定的区域来显示游戏的一些信息, 然后在右边将拼图的图片画出。定义一个二维数组来标识各个小图的排列情况, 遍历该数组, 按数组里所表述的小图排列顺序画出拼图区域。

```
g.setColor(Color.white);
//将当前颜色变为白色
g.fillRect(0,0,DELTAX,IMAGE_HEIGHT*3);
//填充左边的提示信息区域
g.setFont(new Font("宋体",Font.PLAIN,15));
//设置字体
g.setColor(Color.blue);
//设置颜色
g.drawString("步数: "+nStep,10,20);
//在坐标(10,20)处画出字符串, 来显示现在走了多少步。
g.setColor(Color.white);
g.fill3DRect(x,y,IMAGE_WIDTH,IMAGE_HEIGHT,true);
else
```

```

        g.drawImage(myImage [myImageNo[i][j]],x,y,this);
        g.drawRect(x,y,IMAGE_WIDTH,IMAGE_HEIGHT);
    
```

3. 事件处理。

1) 鼠标事件处理

首先判断出鼠标单击的是哪个小图，然后用directionCanMove方法判断其可以往哪个方向移动，若可以移动，则用move方法移动该小图。

判断小图可以往哪个方向移动：

```

//在4个方向上依次判断有没有拼图，若没有，则返回该方向值
//在4个方向上依次判断有没有拼图，若有，则返回一个整型值标志不能移动
if((nCol-1)>=0)
    if(myImageNo[nRow][nCol-1]==NO_IMAGE)
        return DIRECTION_UP;
if((nCol+1)<=2)
    if(myImageNo[nRow][nCol+1]==NO_IMAGE)
        return DIRECTION_DOWN;
if((nRow-1)>=0)
    if(myImageNo[nRow-1][nCol]==NO_IMAGE)
        return DIRECTION_LEFT;
if((nRow+1)<=2)
    if(myImageNo[nRow+1][nCol]==NO_IMAGE)
        return DIRECTION_RIGHT;
    
```

移动拼图：

```

//依次判断是哪个方向，然后按照不同的方向对myImageNo进行不同的操作
case DIRECTION_UP:
    myImageNo[nRow][nCol-1]=myImageNo[nRow][nCol];
    myImageNo[nRow][nCol]=NO_IMAGE;
case DIRECTION_DOWN:
    myImageNo[nRow][nCol+1]=myImageNo[nRow][nCol];
    myImageNo[nRow][nCol]=NO_IMAGE;
case DIRECTION_LEFT:
    myImageNo[nRow-1][nCol]=myImageNo[nRow][nCol];
    myImageNo[nRow][nCol]=NO_IMAGE;
case DIRECTION_RIGHT:
    myImageNo[nRow+1][nCol]=myImageNo[nRow][nCol];
    myImageNo[nRow][nCol]=NO_IMAGE;
    
```

2) 键盘事件处理

首先判断出按下了哪个方向键，将方向存到nDirection中，然后用move方法先判断哪个小图可以往方向nDirection移动，然后移动该小图。

```

        if(myImageNo[i][j]==NO_IMAGE)
            nNoImageRow=i;
            nNoImageCol=j;
        //以上判断哪个拼图可以往方向nDirection移动
        //可以移动的拼图的位置为第nNoImageCol行、第nNoImageRow列
    
```

往某个方向移动拼图:

```

switch(nDirection)
    case DIRECTION_UP:
        if(nNoImageCol==3) return false;
        myImageNo[nNoImageRow][nNoImageCol]=myImageNo[nNoImageRow][nNoImageCol+1];
        myImageNo[nNoImageRow][nNoImageCol+1]=NO_IMAGE;
    case DIRECTION_DOWN:
        if(nNoImageCol==0) return false;
        myImageNo[nNoImageRow][nNoImageCol]=myImageNo[nNoImageRow][nNoImageCol-1];
        myImageNo[nNoImageRow][nNoImageCol-1]=NO_IMAGE;
    case DIRECTION_LEFT:
        if(nNoImageRow==3) return false;
        myImageNo[nNoImageRow][nNoImageCol]=myImageNo[nNoImageRow+1][nNoImageCol];
        myImageNo[nNoImageRow+1][nNoImageCol]=NO_IMAGE;
    case DIRECTION_RIGHT:
        if(nNoImageRow==0) return false;
        myImageNo[nNoImageRow][nNoImageCol]=myImageNo[nNoImageRow-1][nNoImageCol];
        myImageNo[nNoImageRow-1][nNoImageCol]=NO_IMAGE;
    
```

4. 编写随机产生初始状态的代码。

用Math包里的random方法来产生随机数，用这个随机数来初始化游戏的状态：

```
nImgNo=(int)(Math.random()*9);
```

若此图已经分配，则给这个位置重新分配小图：

```

myImageNo[i][j]=nImgNo;
nHasDistrib[nImgNo]=1;
System.out.println("test..");
    
```

初始化玩家走的步数：

```

myImageNo[(int)(Math.random()*3)][(int)(Math.random()*3)]=NO_IMAGE;
nStep=0;
    
```

5. 用多线程技术实现计时器来记录游戏时间。

通过实现Runnable接口来实现多线程，定义一个计时器线程对象thTimer，通过实现接口中的run方法来显示游戏时间。

```

    thTimer.sleep(990);
    String sTemp="你玩了"+nTime+"秒的时间，";
    if(nTime>200)sTemp=sTemp+"时间用的很长了，你可要加油啦!";
    else sTemp=sTemp+"别紧张，慢慢来。";
    showStatus(sTemp);
    if(!bWantStartNewGame)nTime++;
    //如果游戏已经结束，则不将时间累加

```

6. 编写改变不同图片的代码。

用HTML的param标记来实现玩家随便改变大图的功能，最多可以改变9张图片，用数字键来选择。在Applet里使用getParameter方法取得param标记的值，并根据用户的选则装载不同的图片。

修改后的HTML代码：

```

<Html>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>
    拼图游戏
</title>
</HEAD>
<body>
    拼图游戏
    <hr>
    这是一个拼图游戏。玩家应该将打散的小图拼成一张大图。<br>
    玩家可以通过鼠标和键盘来移动小图，移动的次数和拼成<br>
    一张大图所花费的时间作为游戏得分的依据。<br>
    成绩 = 1000 - 时间（秒） - 移动步数 * 10<br>
    按F1键重新开始该游戏，按Y键可以预览整个的图片
    <hr>
    <applet code="Puzzle.class" width="480" height="360">
        <!-- param标记NumOfImage param的标记表示有多少张图片
            剩下的param标记为每张图的名字
        -->
        <param name=NumOfImage value="4">
        <param name=Image1 value="puzzle1">
        <param name=Image2 value="puzzle2">
        <param name=Image3 value="puzzle3">
        <param name=Image4 value="puzzle4">
    </applet>
</body>
</html>

```

7. 编写游戏声音代码。

准备合适的*.mid声音文件，装载后，根据所选小图能否移动来进行播放。

程序源代码与解释

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class Puzzle extends Applet implements MouseListener, KeyListener,
    Runnable {
    Image[] myImage = new Image[9];
    //9个用来装入每个拼图的图片对象
    Image myImageAll;
    //总的大图片
    int myImageNo[][] = new int[3][3];
    //标志现在各个拼图的排列情况
    final int NO_IMAGE = -1;
    //此位置没有拼图
    final int IMAGE_WIDTH = 120;
    //每张小图的宽
    final int IMAGE_HEIGHT = 120;
    //每张小图的高
    final int DIRECTION_UP = 1;
    final int DIRECTION_DOWN = 2;
    final int DIRECTION_LEFT = 3;
    final int DIRECTION_RIGHT = 4;
    final int DIRECTION_NONE = -1;
    final int DELTAX = 120;
    //标志提示信息区的宽度
    boolean bWantStartNewGame = false;
    //游戏是否结束，是否需要开始新游戏
    int nStep = 0;
    //已经走的步数
    int nTime = 0;
```

```

//已经玩过的时间，以秒为单位
Thread thTimer;

//计时器线程
int nScore = 0;

//玩家所得的分数
int NumOfImage = 0;

//拼图底图所使用的图片的个数
String ImageName[] = new String[9];

//记录拼图底图的名字
boolean bOnShowAll = false;

//预览的开关
AudioClip myAudioClip1, myAudioClip2;

//装载要播放的声音对象
public void init() {
    String param = getParameter("NumOfImage");
    try {
        NumOfImage = Integer.parseInt(param);
    } catch (Exception e) {
        NumOfImage = 1;
        System.out.println("无法将参数值转换为整型。");
    }
    for (int i = 0; i < NumOfImage; i++) {
        ImageName[i] = getParameter("Image" + (i + 1)) + ".jpg";
        System.out.println(ImageName[i]);
        if (ImageName[i] == null)
            ImageName[i] = "puzzle.jpg";
    }
    System.out.println(param);
    MediaTracker mediaTracker = new MediaTracker(this);
    myImageAll = getImage(getDocumentBase(), "PuzzleImage/" + ImageName[0]);
    //装载总的大图片
    mediaTracker.addImage(myImageAll, 1);
    try {
        mediaTracker.waitForID(1);
    } catch (Exception e) {
        System.out.println("图片装载出错");
    }
    if (mediaTracker.isErrorAny())
        System.out.println("图片装载出错");
    for (int i = 0; i < 9; i++) {

```

```

    }

    myImage[i] = createImage(IMAGE_WIDTH, IMAGE_HEIGHT);
    Graphics g = myImage[i].getGraphics();
    int nRow = i % 3;
    int nCol = i / 3;
    g.drawImage(myImageAll, 0, 0, IMAGE_WIDTH, IMAGE_HEIGHT, nRow
        * IMAGE_WIDTH, nCol * IMAGE_HEIGHT, (nRow + 1)
        * IMAGE_WIDTH, (nCol + 1) * IMAGE_HEIGHT, this);
    System.out.println("init " + i);
}

System.out.println("init over");
thTimer = new Thread(this);
//为线程分配内存空间
thTimer.start();
//开始线程
initData();
myAudioClip1 = getAudioClip(getCodeBase(), "Sound/move.mid");
myAudioClip2 = getAudioClip(getCodeBase(), "Sound/notmove.mid");
addMouseListener(this);
addKeyListener(this);
}

public void initImageAgain(int nImgNo) {
//nImgNo为要装载的图像是第几个图像
if (nImgNo > NumOfImage) {
    showStatus("你要的图片不存在！！");
    return;
}
MediaTracker mediaTracker = new MediaTracker(this);
myImageAll = getImage(getDocumentBase(), "PuzzleImage/"
    + ImageName[nImgNo]);
mediaTracker.addImage(myImageAll, 1);
try {
    mediaTracker.waitForAll();
} catch (Exception e) {
    System.out.println("图片装载出错");
}
for (int i = 0; i < 9; i++) {
    myImage[i] = createImage(IMAGE_WIDTH, IMAGE_HEIGHT);
    Graphics g = myImage[i].getGraphics();
    int nRow = i % 3;
    int nCol = i / 3;
    g.drawImage(myImageAll, 0, 0, IMAGE_WIDTH, IMAGE_HEIGHT, nRow
        * IMAGE_WIDTH, nCol * IMAGE_HEIGHT, (nRow + 1)
        * IMAGE_WIDTH, (nCol + 1) * IMAGE_HEIGHT, this);
    System.out.println("init " + i);
}

System.out.println("init over");
thTimer = new Thread(this);
//为线程分配内存空间
thTimer.start();
//开始线程
initData();
myAudioClip1 = getAudioClip(getCodeBase(), "Sound/move.mid");
myAudioClip2 = getAudioClip(getCodeBase(), "Sound/notmove.mid");
addMouseListener(this);
addKeyListener(this);
}
}

```

```

        g.drawImage(myImageAll, 0, 0, IMAGE_WIDTH, IMAGE_HEIGHT, nRow
                    * IMAGE_WIDTH, nCol * IMAGE_HEIGHT, (nRow + 1)
                    * IMAGE_WIDTH, (nCol + 1) * IMAGE_HEIGHT, this);
        System.out.println("dfsdfdfdsdf" + i);
    }
}

public void update(Graphics g) { //重载update方法，消除闪烁
    paint(g);
}

public void paint(Graphics g) {
    g.setColor(Color.white);
    //将当前颜色变为白色
    g.fillRect(0, 0, DELTAX, IMAGE_HEIGHT * 3);
    //填充左边的提示信息区域
    g.setFont(new Font("宋体", Font.PLAIN, 15));
    //设置字体
    g.setColor(Color.blue);
    //设置颜色
    g.drawString("步数：" + nStep, 5, 20);
    g.drawString("现有图片" + NumOfImage + "张", 5, 60);
    g.drawString("请按1-" + NumOfImage + "键改变图片", 5, 100);
    //在坐标 (10, 20) 画出字串，来显示现在走了多少步。
    g.setColor(Color.white);
    if (bOnShowAll) {
        int x = DELTAX;
        int y = 0;
        g.drawImage(myImageAll, x, y, this);
        return;
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            int x = i * IMAGE_WIDTH + DELTAX;
            int y = j * IMAGE_HEIGHT;
            if (myImageNo[i][j] == NO_IMAGE)
                g.fill3DRect(x, y, IMAGE_WIDTH, IMAGE_HEIGHT, true);
            else {
                g.drawImage(myImage[myImageNo[i][j]], x, y, this);
                g.drawRect(x, y, IMAGE_WIDTH, IMAGE_HEIGHT);
            }
        }
    }
}

```

```

        checkStatus();
        if (bWantStartNewGame) {
            //如果游戏结束，玩家将拼图的顺序排对之后
            nScore = 1000 - nStep * 10 - nTime;
            g.setColor(Color.blue);
            g.drawString("请按任意键重新开始", 5, 140);
            g.setColor(Color.red);
            g.setFont(new Font("宋体", Font.PLAIN, 40));
            g.drawString("你赢了" + nScore + "分", 70 + DELTAX, 160);
            g.drawString("祝贺你！", 110 + DELTAX, 210);
        }
    }

    public Puzzle() {
    }

    public void initData() {
        int[] nHasDistrib = new int[9];
        for (int i = 0; i < 9; i++)
            nHasDistrib[i] = 0;
        for (int j = 0; j < 3; j++) {
            for (int i = 0; i < 3; i++) {
                int nCount = j * 3 + i;
                int nImgNo = -1;
                do {
                    nImgNo = (int) (Math.random() * 9);
                } while (nHasDistrib[nImgNo] == 1);
                //1代表已经分配了这张图片
                myImageNo[i][j] = nImgNo;
                nHasDistrib[nImgNo] = 1;
                System.out.println("test..");
            }
        }
        myImageNo[(int) (Math.random() * 3)][(int) (Math.random() * 3)] = NO_IMAGE;
        nStep = 0;
        nTime = 0;
        //清空计时器
    }

    public int directionCanMove(int nCol, int nRow) {
        if ((nCol - 1) >= 0)
            if (myImageNo[nRow][nCol - 1] == NO_IMAGE)
                return DIRECTION_UP;
        if ((nCol + 1) <= 2)
    }
}

```

```

        if (myImageNo[nRow][nCol + 1] == NO_IMAGE)
            return DIRECTION_DOWN;
        if ((nRow - 1) >= 0)
            if (myImageNo[nRow - 1][nCol] == NO_IMAGE)
                return DIRECTION_LEFT;
        if ((nRow + 1) <= 2)
            if (myImageNo[nRow + 1][nCol] == NO_IMAGE)
                return DIRECTION_RIGHT;
        return DIRECTION_NONE;
    }

    public void move(int nCol, int nRow, int nDirection) {
        switch (nDirection) {
            case DIRECTION_UP:
                myImageNo[nRow][nCol - 1] = myImageNo[nRow][nCol];
                myImageNo[nRow][nCol] = NO_IMAGE;
                break;
            case DIRECTION_DOWN:
                myImageNo[nRow][nCol + 1] = myImageNo[nRow][nCol];
                myImageNo[nRow][nCol] = NO_IMAGE;
                break;
            case DIRECTION_LEFT:
                myImageNo[nRow - 1][nCol] = myImageNo[nRow][nCol];
                myImageNo[nRow][nCol] = NO_IMAGE;
                break;
            case DIRECTION_RIGHT:
                myImageNo[nRow + 1][nCol] = myImageNo[nRow][nCol];
                myImageNo[nRow][nCol] = NO_IMAGE;
                break;
        }
    }

    public boolean move(int nDirection) {
        int nNoImageCol = -1;
        int nNoImageRow = -1;
        int i = 0;
        int j = 0;
        while (i < 3 && nNoImageRow == -1) {
            while (j < 3 && nNoImageCol == -1) {
                if (myImageNo[i][j] == NO_IMAGE) {
                    nNoImageRow = i;
                    nNoImageCol = j;
                }
            }
        }
    }
}

```

```

        j++;
    }
    j = 0;
    i++;
}
//以上判断哪个拼图可以往方向nDirection移动
//可以移动的拼图的位置为第nNoImageCol行，第nNoImageRow列
System.out.println(nNoImageCol + " , " + nNoImageRow);
switch (nDirection) {
case DIRECTION_UP:
    if (nNoImageCol == 3)
        return false;
    myImageNo[nNoImageRow][nNoImageCol] = myImageNo[nNoImageRow]
[nNoImageCol + 1];
    myImageNo[nNoImageRow][nNoImageCol + 1] = NO_IMAGE;
    break;
case DIRECTION_DOWN:
    if (nNoImageCol == 0)
        return false;
    myImageNo[nNoImageRow][nNoImageCol] = myImageNo[nNoImageRow]
[nNoImageCol - 1];
    myImageNo[nNoImageRow][nNoImageCol - 1] = NO_IMAGE;
    break;
case DIRECTION_LEFT:
    if (nNoImageRow == 3)
        return false;
    myImageNo[nNoImageRow][nNoImageCol] = myImageNo[nNoImageRow + 1]
[nNoImageCol];
    myImageNo[nNoImageRow + 1][nNoImageCol] = NO_IMAGE;
    break;
case DIRECTION_RIGHT:
    if (nNoImageRow == 0)
        return false;
    myImageNo[nNoImageRow][nNoImageCol] = myImageNo[nNoImageRow - 1]
[nNoImageCol];
    myImageNo[nNoImageRow - 1][nNoImageCol] = NO_IMAGE;
    break;
}
return true;
}

public void mouseClicked(MouseEvent e) {

```

```

//鼠标单击事件
if (bOnShowAll)
    return;
if (bWantStartNewGame) {
    initData();
    repaint();
    bWantStartNewGame = false;
    return;
}

int nX = e.getX() - DELTAX;
int nY = e.getY();
int nCol = nY / IMAGE_HEIGHT;
int nRow = nX / IMAGE_WIDTH;
System.out.println("col:" + nCol + " row:" + nRow);
int nDirection = directionCanMove(nCol, nRow);
if (nDirection != DIRECTION_NONE) {
    move(nCol, nRow, nDirection);
    nStep++;
    myAudioClip1.play();
    repaint();
} else {
    myAudioClip2.play();
}
}

public void mouseEntered(MouseEvent e) {
    //鼠标移入事件
}

public void mouseExited(MouseEvent e) {
    //鼠标移出事件
}

public void mousePressed(MouseEvent e) {
    //鼠标按住不放事件
}

public void mouseReleased(MouseEvent e) {
    //鼠标松开按钮事件
}

public void keyPressed(KeyEvent e) {
    //键盘按键事件
    if (bOnShowAll) {

```

```

        if (e.getKeyCode() == KeyEvent.VK_Y) {
            bOnShowAll = false;
            repaint();
        }
        return;
    }
    System.out.println("press key" + e.getKeyCode() + "      "
        + e.getText(e.getKeyCode()));
    System.out.println(KeyEvent.VK_LEFT);
    if (bWantStartNewGame) {
        initData();
        bWantStartNewGame = false;
        repaint();
        return;
    }
    int nDirection = DIRECTION_NONE;
    switch (e.getKeyCode()) {
    case KeyEvent.VK_DOWN:
        nDirection = DIRECTION_DOWN;
        break;
    case KeyEvent.VK_UP:
        nDirection = DIRECTION_UP;
        break;
    case KeyEvent.VK_LEFT:
        System.out.println("left111... ");
        nDirection = DIRECTION_LEFT;
        break;
    case KeyEvent.VK_RIGHT:
        System.out.println("left... ");
        nDirection = DIRECTION_RIGHT;
        break;
    case KeyEvent.VK_F1:
        //F1键按下，重新开始游戏
        initData();
        //init();
        repaint();
        return;
    case KeyEvent.VK_1:
    case KeyEvent.VK_2:
    case KeyEvent.VK_3:
    case KeyEvent.VK_4:
    case KeyEvent.VK_5:

```

```

        case KeyEvent.VK_6:
        case KeyEvent.VK_7:
        case KeyEvent.VK_8:
        case KeyEvent.VK_9:
            int nImgNo = e.getKeyCode() - KeyEvent.VK_1;
            if (nImgNo < NumOfImage) {
                System.out.println(nImgNo);
                initImageAgain(nImgNo);
                initData();
                repaint();
            }
            return;
        case KeyEvent.VK_Y:
            if (bOnShowAll)
                bOnShowAll = false;
            else
                bOnShowAll = true;
            repaint();
            return;
        default:
            return;
        }
        boolean bCanMove = move(nDirection);
        if (bCanMove) {
            nStep++;
            myAudioClip1.play();
            repaint();
        } else {
            myAudioClip2.play();
        }
    }

    public void keyReleased(KeyEvent e) {
        //按键松开事件。
    }

    public void keyTyped(KeyEvent e) {
        //按键敲入事件
    }

    public void checkStatus() {
        boolean bWin = true;
        //定义成员， 默认值为真
        int nCorrectNum = 0;

```

```

        for (int j = 0; j < 3; j++) {
            for (int i = 0; i < 3; i++) {
                if (myImageNo[i][j] != nCorrectNum
                    && myImageNo[i][j] != NO_IMAGE)
                    bWin = false;
                nCorrectNum++;
            }
        }
        //比较拼图是否都放到了正确的位置上
        //如果有一个没有放到正确位置上，则游戏就不能结束
        if (bWin)
            bWantStartNewGame = true;
    }

    public static void main(String[] args) {           //主方法
        Puzzle Puzzle1 = new Puzzle();
        Puzzle1.init();
        Puzzle1.start();
    }

    public void run() {           //Applet的运行方法
        while (Thread.currentThread() == thTimer) {
            try {
                thTimer.sleep(990);
                String sTemp = "你玩了" + nTime + "秒的时间，";
                if (nTime > 200)
                    sTemp = sTemp + "时间用的很长了，你可要加油啦！";
                else
                    sTemp = sTemp + "别紧张，慢慢来。";
                showStatus(sTemp);
                if (!bWantStartNewGame)
                    nTime++;
            } catch (Exception e) {
            }
        }
    }
}

```



案例4 贪吃蛇游戏

案例运行效果与操作

贪吃蛇是一款非常经典的手机游戏，本案例用Applet实现了4种级别的玩法，通过方向键控制蛇的动向，吃掉前面的食物，当碰到墙壁时，失败。图4-12为游戏开始界面。

本例中有4个级别，分别是，“初级”、“中级”、“高级”、“专家级”，单击希望进入的级别，则进入图4-13所示的游戏界面。

移动方向键，贪吃蛇开始运动，控制方向键来控制贪吃蛇的动向，向着食物的方向前进，如果一不小心，碰到墙壁，游戏则结束，界面下方会显示本轮的得分。如图4-14所示。



图4-12 游戏开始界面

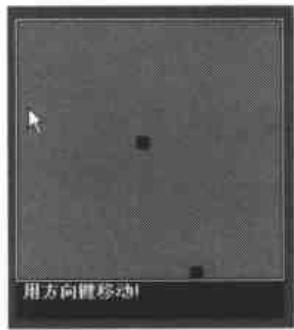


图4-13 进入界面

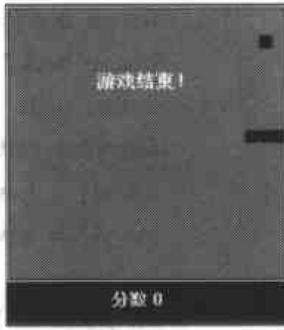


图4-14 游戏结束界面

制作要点

1. Applet类的应用
2. Graphics对象的应用，以及getImage和drawImage等方法的使用
3. 鼠标和键盘事件处理

步骤详解

1. 嵌入Applet的HTML页面源码：

```
<html>
<body bgcolor="#333333">
<center>
<applet code="snake.class" align=center width="202" height="232"></applet>
</center>
</html>
```

2. 蛇的移动。

蛇本身在直行时由小段组成，碰到拐弯变成两段，每段由许多小黑格组成，这里说的小黑格是指一个图片文件。

```
dot[z] = getImage(getCodeBase(), "dot.gif");
```

3. 键盘的移动。

```
if ((key == Event.UP) && (!down)){ up = true; right = false; left = false;if(!started)
started=true;}
if ((key == Event.DOWN) && (!up)){down = true; right = false; left = false;if(!started)
started=true;}
```

程序代码与解析

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

public class Snake extends Applet implements Runnable { //用Runnable接口实现多线程
    Image dot[] = new Image[400];
    Image back;
    Image offI;
    Graphics offG;
    int x[] = new int[400];
    int y[] = new int[400];
    int rtemp = 1;
    int game = 1;
    int count = 0;
    int score = 0;
    int add = 1;
    int level,z,n;
    Button level1 = new Button("初级");
    Button level2 = new Button("中级");
    Button level3 = new Button("高级");
    Button level4 = new Button("专家级");
    String stemp;
    String s,t;
    boolean go[] = new boolean[400];
    boolean left = false;
    boolean right = false;
    boolean up = false;
    boolean down = false;
    boolean started = false;
    Thread setTime;
    public void init() { //Applet的初始化方法
        //添加按钮组件
        add(level1);
        add(level2);
```

```

        add(level3);
        add(level4);
        //设置背景色为黑色
        setBackground(Color.black);
        back = getImage(getCodeBase(), "screan.gif"); //背景图案
        for (z = 0; z < 400; z++) { //蛇和食物的图案
            dot[z] = getImage(getCodeBase(), "dot.gif");
        }
    }

    public void update(Graphics g) { //重载update方法，消除闪烁
        Dimension d = this.size();
        if (offl == null) {
            offl = createImage(d.width, d.height); //双缓冲技术
            offG = offl.getGraphics();
        }
        offG.clearRect(0, 0, d.width, d.height);
        paint(offG);
        g.drawImage(offl, 0, 0, null);
    }

    public void paint(Graphics g) { //画出图像
        g.drawImage(back, 0, 0, this);
        g.setColor(Color.white);
        if (started) { //显示分数信息
            g.setFont(new Font("TimesRoman", 1, 12));
            t = "分数 " + score + "";
            g.drawString(t, 75, 220);
        }
        if (game == 1) {
            g.setFont(new Font("TimesRoman", 1, 13));
            s = "游戏模式选择";
            g.drawString(s, 65, 30);
            //移动按钮位置
            level1.move(75, 50);
            level2.move(75, 90);
            level3.move(75, 130);
            level4.move(70, 170);
        }
        if ((game == 2) || (game == 3)) {
            if (!started) {
                g.setFont(new Font("TimesRoman", 1, 13));
                t = "用方向键移动!";
                g.drawString(t, 5, 215);
            }
        }
    }
}

```

```

        }
        for (z = 0; z <= n; z++) {
            g.drawImage(dot[z], x[z], y[z], this);
        }
    }
    if (game == 3) {
        g.setFont(new Font("TimesRoman", 1, 13));
        s = "游戏结束!";
        g.drawString(s, 65, 60);
    }
}
public void run() { //Applet的运行方法
    for (z = 4; z < 400; z++) {
        go[z] = false;
    }
    for (z = 0; z < 4; z++) {
        go[z] = true;
        x[z] = 91;
        y[z] = 91;
    }
    n = 3;
    game = 2;
    score = 0;
    level1.move(70, -100);
    level2.move(70, -100);
    level3.move(70, -100);
    level4.move(70, -100);
    left = false;
    right = false;
    up = false;
    down = false;
    locateRandom(4);
    while (true) {
        if (game == 2) {
            if ((x[0] == x[n]) && (y[0] == y[n])) {
                go[n] = true;
                locateRandom((n + 1));
                score += add;
            }
            for (z = 399; z > 0; z--) {
                if (go[z]) {
                    x[z] = x[(z - 1)];

```

```

y[z] = y[(z - 1)];
if ((z > 4) && (x[0] == x[z]) && (y[0] == y[z])) {
    game = 3;
}
}

if (left) {
    x[0] -= 10;
}
if (right) {
    x[0] += 10;
}
if (up) {
    y[0] -= 10;
}
if (down) {
    y[0] += 10;
}
}

if (y[0] > 191) {
    y[0] = 191;
    game = 3;
}
if (y[0] < 1) {
    y[0] = 1;
    game = 3;
}
if (x[0] > 191) {
    x[0] = 191;
    game = 3;
}
if (x[0] < 1) {
    x[0] = 1;
    game = 3;
}
}

if (game == 3) {
    if (count < (1500 / level)) {
        count++;
    } else {
        count = 0;
        game = 1;
        repaint();
    }
}

```

```

        setTime.stop();
    }
}
repaint();
try {
    setTime.sleep(level);
} catch (InterruptedException e) {
}
}

public void locateRandom(int turn) { //产生随机位置
    rtemp = (int) (Math.random() * 20);
    x[turn] = ((rtemp * 10) + 1);
    rtemp = (int) (Math.random() * 20);
    y[turn] = ((rtemp * 10) + 1);
    n++;
}

public boolean keyDown(Event e, int key) {
    if ((key == Event.LEFT) && (!right)) {
        left = true;
        up = false;
        down = false;
        if (!started)
            started = true;
    }
    if ((key == Event.RIGHT) && (!left)) {
        right = true;
        up = false;
        down = false;
        if (!started)
            started = true;
    }
    if ((key == Event.UP) && (!down)) {
        up = true;
        right = false;
        left = false;
        if (!started)
            started = true;
    }
    if ((key == Event.DOWN) && (!up)) {
        down = true;
        right = false;
    }
}

```

```
    left = false;
    if (!started)
        started = true;
    }
    return true;
}

public boolean action(Event event, Object obj) { //设置不同级别的行为
    stemp = (String) obj;
    if (stemp.equals("初级")) {
        add = 2;
        level = 100;
        setTime = new Thread(this);
        setTime.start();
        return true;
    }
    if (stemp.equals("中级")) {
        add = 5;
        level = 70;
        setTime = new Thread(this);
        setTime.start();
        return true;
    }
    if (stemp.equals("高级")) {
        add = 10;
        level = 40;
        setTime = new Thread(this);
        setTime.start();
        return true;
    }
    if (stemp.equals("专家级")) {
        add = 20;
        level = 20;
        setTime = new Thread(this);
        setTime.start();
        return true;
    }
    return false;
}
```



案例5 打球游戏

案例运行效果与操作

本案例是一个非常好玩的Applet游戏，共有40个级别的玩法。目标是在限定的步骤中将围墙内的所有小球击碎，两个或两个以上的小球在同一水平或者竖直线上时，小球便会粉碎。图4-15便是其中的Level 2的界面。

通过方向键移动小球，使其尽可能多地处在同一水平或竖直线上，直至小球全部消失。如图4-16所示。

如果在规定的步骤内小球没有消失，则游戏失败。如图4-17所示。

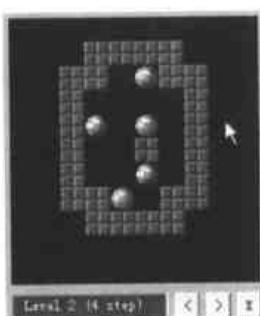


图4-15 运行界面1

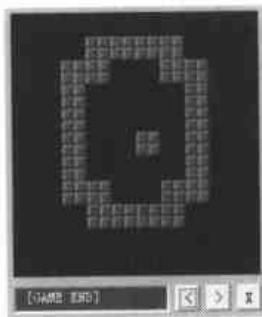


图4-16 运行界面2

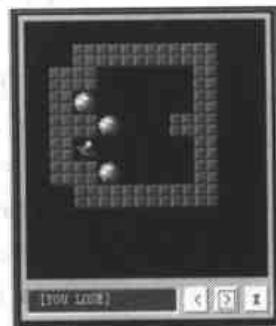


图4-17 失败界面

制作要点

1. Applet类的用法
2. catch (InterruptedException e)中断异常捕获

步骤详解

1. 嵌入Applet的html页面源码。

```
<applet code=Eliminator width=210 height=260>
</applet>
```

2. 修改游戏数据。

如果感兴趣的话，可以利用data和maxstep参数修改游戏级别，例如：

```
<applet name=Eliminator width=210 height=260>
<param name=data0 value=",,#####,.o #,#.o #,#####">
<param name=maxstep0 value=3>
</applet>
```

可将游戏级别改为0。

围墙的特性值修改如下：

```
# : 围墙
* : 无球
. : 球 #1
o : 球 #2
O : 球 #3
@ : 球 #4
+ : 球 #5
space : 空域
, : 输入下一行数据
```

程序源代码与解释

```
/*
*Applet小游戏
*/
import java.applet.*;
import java.awt.*;
import java.awt.image.*;
import java.net.*;
import java.util.*;
// Runnable接口实现多线程
public class BeatBall extends Applet implements Runnable
{
    static int MAX_X=9, MAX_Y=10, MAX_LVL=100, MAX_IMG=11, MAX_HIST=100;
    static int X_0 = 5, Y_0 = 5; //静态变量说明
    Thread game0 = null;
    private Image[] xImg = new Image[MAX_IMG+1];
    private int xPos, yPos, gLevel, mvPos, maxMove;
    private byte[][] currCanvas = new byte[MAX_Y+1][MAX_X+1];
    private byte[] xMove = new byte[MAX_HIST];
    private byte firstTime;

    private Button btn1 = new Button("X");
    private Button btnN = new Button(">");
    private Button btnP = new Button("<");
    //定义按钮
    private static String stGameEnd = " [GAME END]";
    private static String stYouLost = " [YOU LOST]";
    //定义游戏状态
```

```

private static String widthe = "# * .oO@+,123456789abcdefgij";
private static int xCoord[][] =
{{0, 0,20,20}, {20, 0,20,20}, {40, 0,20,20}, {60, 0,20,20}, {80, 0,20,20},
{100, 0,20,20},
{100,20,20,20}, {0,20,20,20}, {20,20,20,20}, {40,20,20,20}, {60,20,20,20},
{80,20,20,20}};

private static String biDat[] = {
};

private static int aMove[] =
{4,4,4,4,4,4,4,4,6,6,6,6,6,6,6,6,7,7,7,8,8,8,8,8,9,9,9,9,9,9,9,9,10,};

private Image mainImg;
private Graphics g;
private TextField txt1;

public void init()
{
    String param;
    int i;
    if (g==null) g = getGraphics();
    URL url = getCodeBase();
    String str = "";
    int b = 0;
    MediaTracker tracker = new MediaTracker(this);
    mainImg = getImage(url, "Balls.gif");
    tracker.addImage(mainImg,0);
    for (i=0; i<=MAX_IMG; i++)
    {
        xImg[i] = extractImage(xCoord[i]);
        tracker.addImage(xImg[i], i+1);
    }
    try { tracker.waitForAll(); }
    catch (InterruptedException e) {}
    //设置布局
    setLayout(new BorderLayout());
    txt1 = new TextField(17);
    txt1.setForeground(new Color(255,255,204));
    txt1.setBackground(new Color(102,51,51));
    txt1.setEditable(false);
    Panel p = new Panel();
    p.add(txt1);
    p.add(btnP);
}

```

```

p.add(btnN);
p.add(btn1);
add("South", p);//布局格式
gLevel = 0;
dat2Canvas(0);
firstTime = 1;
}
public void start() //启动线程
{
if (game0==null)
{
    game0 = new Thread(this);
    game0.start();
}
}
public void stop()
{
if (game0 != null)
{
    game0.stop();
    game0 = null;
}
}
public void run()
{
}
public boolean dat2Canvas(int dlvl)
{
int i,x,y,tmp0;
byte z = 0, oz = 0;
String param;
try
{
    gLevel += dlvl;
    if (gLevel<0) gLevel = 0;
    try { i = Integer.parseInt(getParameter("maxmove"+gLevel)); }
    catch (NumberFormatException e) { i = 0; }
    param = getParameter("data"+gLevel);
    if (i>0) maxMove = i;
    if (param==null || i<=0)
        { param = biDat[gLevel]; maxMove = aMove[gLevel]; }
}
}

```

```

        catch (ArrayIndexOutOfBoundsException exc)
        {
            gLevel -= dlvl;
            return false;
        }
        for (y=0; y<=MAX_Y; y++)
        {
            for (x=0; x<=MAX_X; x++)
            {
                currCanvas[y][x] = 0;
            }
        }
        x = y = 0;
        for (i=0; i<param.length(); i++)
        {
            z = (byte)(widthe.indexOf(param.charAt(i)));
            if (z>MAX_IMG)
            {
                if ((tmp0 = (z-MAX_IMG-1))>0)
                {
                    for (int j=0; j<tmp0; j++)
                    {
                        try { currCanvas[y][x+j] = oz; }
                        catch (ArrayIndexOutOfBoundsException exc) { j = tmp0; }
                    }
                }
            }
            else
            {
                x = 0; y++;
                if (y>MAX_Y) break;
            }
        }
        else if (x<=MAX_X)
        {
            if (z<0) z = 0;
            currCanvas[y][x++] = z;
            oz = z;
        }
    }
    for (i=0; i<MAX_HIST; i++) { xMove[i] = 0; }
    mvPos = 0;

    txt1.setText(" Level "+gLevel+" ("+maxMove+" step max.)");
    if (firstTime==1) firstTime = 0;
    return true;
}

```

```

public boolean action(Event e, Object arg)
{
    int h = -1;
    if (e.target==btnP) h = 44;
    else if (e.target==btnN) h = 46;
    else if (e.target==btn1) h = 27;
    if (h>0) keyDown(e, h);
    return true;
}
public boolean keyDown(Event e, int c)//键盘动作处理
{
    int dx = 0;
    int dy = 0;
    int ch = 0;
    int h = 0;
    switch (c)
    {
        case Event.UP: dy = -1; h = 1; break;
        case Event.DOWN: dy = 1; h = 3; break;
        case Event.LEFT: dx = -1; h = 5; break;
        case Event.RIGHT: dx = 1; h = 7; break;
        case 27:
            if (mvPos==0 && firstTime==0) { firstTime = 1; paint(null); return true; }
            h = -100; break;
        case 44: h = -101; break;
        case 46: h = -99; break;
        case 14: h = -90; break;
        case 16: h = -110; break;
        default:
            txt1.setText(" Use ^P,<,>^N,^R, or BS key.");
            break;
    }
    if (dx==0 && dy==0)
    {
        if (h<0) { if (dat2Canvas(h+100)) paint(null); }
        return true;
    }
    if (mvPos>=maxMove)
        { txt1.setText(stYouLost); return true; }
    h = 1; ch = 0;
    while(h>0)
    {

```

```

ch=0;
if ((h = moveBall(dx,dy))>0)
{ ch = 1; h = checkBoard(); }
}
if (ch==1) { mvPos++; txt1.setText(" Have "+(maxMove-mvPos)+" more step."); }
else return true;

h = GameEnd();
if (h==1) { txt1.setText(stGameEnd); h = -98; }
else if (h== -1 || mvPos>=maxMove)
{ txt1.setText(stYouLost); h = -99; }

if (h<=-98)
{
try { game0.sleep(1500); }
catch (InterruptedException exc) {}
if (dat2Canvas(h+99)) paint(null);
}

return true;
}

public int moveBall(int dx, int dy)//球的移动处理
{
int x,y,lx,ly,x2,y2;
byte z0,z1;
int h = 0;
if (dx != 0)
{
    ly = Y_0;
    for (y=0; y<=MAX_Y; y++)
    {
        if (dx==1) x = MAX_X-1; else x = 1;
        while (x>0 && x<MAX_X)
        {
            if ((z0 = currCanvas[y][x])>=6)
            {
                x2 = x; lx = X_0+20*x;
                while (x2>0 && x2<MAX_X && (z1=currCanvas[y][x2+dx])==0)
                {
                    try { game0.sleep(10); }
                    catch (InterruptedException e) {}
                    h++;
                    currCanvas[y][x2] = 0;
                    g.drawImage(xImg[0],lx,ly,this);
                    x2 += dx; lx += 20*dx;
                }
            }
        }
    }
}

```

```

        currCanvas[y][x2] = z0;
        g.drawImage(xImg[z0],lx,ly,this);
    }
}
x -= dx;
}
ly += 20;
}
}
if (dy != 0)
{
    lx = X_0;
    for (x=0; x<=MAX_X; x++)
    {
        if (dy==1) y = MAX_Y-1; else y = 1;
        while (y>0 && y<MAX_Y)
        {
            if ((z0 = currCanvas[y][x])>=6)
            {
                y2 = y; ly = Y_0+20*y;
                while (y2>0 && y2<MAX_Y && (z1=currCanvas[y2+dy][x])==0)
                {
                    try { game0.sleep(10); }
                    catch (InterruptedException e) {}
                    h++;
                    currCanvas[y2][x] = 0;
                    g.drawImage(xImg[0],lx,ly,this);
                    y2 += dy; ly += 20*dy;
                    currCanvas[y2][x] = z0;
                    g.drawImage(xImg[z0],lx,ly,this);
                }
            }
            y -= dy;
        }
        lx += 20;
    }
}
return h;
}
public int checkBoard()//是否越界
{
    byte z0,z1,z2;
}

```

```

int h = 0;
int h2 = 0;
z1 = 100;
for (int y=0; y<MAX_Y; y++)
{
    for (int x=0; x<MAX_X; x++)
    {
        z0 = (byte)(currCanvas[y][x] % z1);
        if (z0>=7)
        {
            z2 = (byte)(z0+z1);
            if (z0==(byte)(currCanvas[y][x+1] % z1))
            {
                currCanvas[y][x] =
                currCanvas[y][x+1] = z2;
                h++;
            }
            if (z0==(byte)(currCanvas[y+1][x] % z1))
            {
                currCanvas[y][x] =
                currCanvas[y+1][x] = z2;
                h++;
            }
        }
    }
}
if (h==0) return 0;
for (int z=2; z<=6; z++)
{
    try { game0.sleep(100); }
    catch (InterruptedException e) {}
    int ly = Y_0;
    for (int y=0; y<MAX_Y; y++)
    {
        int lx = X_0;
        for (int x=0; x<MAX_X; x++)
        {
            if ((z0 = currCanvas[y][x])>MAX_IMG)
            {
                if (z<6) g.drawImage(xImg[z],lx,ly,this);
                else
                {

```

```

        currCanvas[y][x] = 0;
        g.drawImage(xImg[0],lx,ly,this);
    }
}
lx += 20;
}
ly += 20;
}
}
return h;
}
public int GameEnd()//游戏结束处理
{
    int x,y;
    int h = 1;
    int h0[] = new int[MAX_IMG+1];
    for (y=0; y<=MAX_IMG; y++) h0[y] = 0;
    for (y=0; y<=MAX_Y; y++)
    {
        for (x=0; x<=MAX_X; x++)
        {
            h0[currCanvas[y][x]]++;
        }
    }
    y = 7;
    while (y<=MAX_IMG)
    {
        x = h0[y];
        if (x==1) { h = -1; y = MAX_IMG+1; }
        else if (h==1 && x>1) h = 0;
        y++;
    }
    return h;
}
private Image extractImage(int[] xyCoord)
{
    Image newImage;
    ImageFilter filter;
    ImageProducer producer;
    filter = new CropImageFilter(xyCoord[0],xyCoord[1],xyCoord[2],xyCoord[3]);
    producer = new FilteredImageSource(mainImg.getSource(), filter);
    newImage = createImage(producer);
}

```

```

        return newImage;
    }

    public void paint(Graphics g)
    {
        if (g==null) g = getGraphics();
        int y2 = Y_0;
        for (int y=0; y<=MAX_Y; y++)
        {
            int x2 = X_0;
            for (int x=0; x<=MAX_X; x++)
            {
                byte z = currCanvas[y][x];
                if (firstTime==1 || z>MAX_IMG) z = 0;
                g.drawImage(xImg[z],x2,y2,this);
                x2 += 20;
            }
            y2 += 20;
        }
        if (firstTime==1) doFirstTime();
    }

    public void doFirstTime()//进入游戏时设置，说明规则
    {
        int y = 20, x = 15;
        g.setColor(new Color(255,255,204));
        String str = "* 打球游戏 *";
        g.drawString(str,15,y);
        txt1.setText(str);
        y += 20; g.drawString("",15,y);
        y += 20; g.drawString("游戏目标：打碎所有球",15,y);
        y += 15; g.drawString("",15,y);
        y += 15; g.drawString("移动球：方向键",15,y);
        y += 15; g.drawString("",15,y);
        y += 15; g.drawString("打碎规则：水平或竖直两个以上的球",15,y);
        y += 15; g.drawString("",15,y);
        y += 15; g.drawString("",15,y);
        y += 15; g.drawString("WIN：所有球击碎",15,y);
        y += 30; g.drawString("",15,y);
        y += 15; g.drawString("LOST：球有剩余",15,y);
        y += 15; g.drawString("",15,y);
    }
}

```



案例6 棒打猪头

案例运行效果与操作

本游戏的窗口中不断出现非常可爱的小猪，只要用榔头（鼠标）打中猪的头，小猪便会消失，根据打的速度和个数计算得分，是非常有意思的一款练习反应速度的小游戏。图4-18为游戏运行界面。

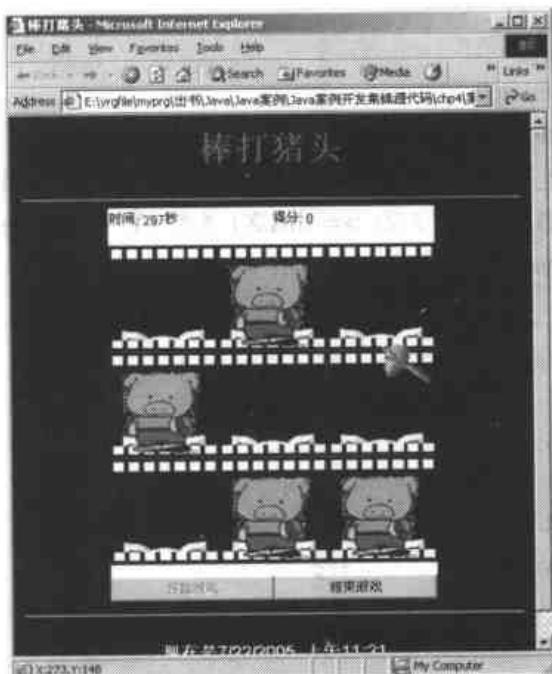


图4-18 游戏界面

单击“结束游戏”结束，屏幕上显示得分和所用的时间。

制作要点

1. 抽象类Abstract class的应用及抽象类的继承
2. Applet类的应用
3. 多线程的应用

步骤详解

1. 定义抽象类Abstract class。

```
abstract class SuperSprite
```

2. 抽象类的继承。

```
class HammerSprite extends SuperSprite
```

3. 嵌入applet的html源码。

```
<Applet code="HitPigHeadv.class" width=300 height=370></Applet>
```

4. 图形输出。

```
g.drawImage(Frame,X,Y,Game);
if(showPig == true)
    g.drawImage(SpriteImage,X + 12,Y + 18,Game);
```

5. 鼠标监听来控制击打。

```
addMouseListener(this);
addMouseMotionListener(this);
if((this.X + P_Width >= X) && (this.Y + (P_Height / 2) >= Y) &&
(X + (H_Width / 2) >= this.X) && (Y + (H_Height / 2) >= this.Y)
&& showPig)
```

程序源代码与解析

```
import java.awt.*;
import java.util.*;
import java.applet.*;
import java.awt.event.*;

abstract class SuperSprite
{
    int      X,Y,width,height;
    boolean visible,active;
    abstract public void paintSprite(Graphics g);
    abstract public void updateState();
    public int getX()
    {
        return X;
    }
    public int getY()
    {
        return Y;
    }
    public void setLocation(int X,int Y)
    {
        this.X = X;
        this.Y = Y;
    }
}
```

```
        }
        public int getWidth()
        {
            return width;
        }
        public int getHeight()
        {
            return height;
        }
        public void setSize(int width,int height)
        {
            this.width = width;
            this.height = height;
        }
        public boolean canVisible()
        {
            return visible;
        }
        public void setVisible(boolean v)
        {
            visible = v;
        }
        public boolean canMove()
        {
            return active;
        }
        public void setMove(boolean m)
        {
            active = m;
        }
    }
    class PigSprite extends SuperSprite
    {
        int seed;
        Image SpriteImage,Frame;
        Applet Game;
        Random R;
        boolean showPig;
        public PigSprite(Image SpriteImage,Image Frame,Applet Game)
        {
            R = new Random();
            this.SpriteImage = SpriteImage;
```

```

        this.Frame      = Frame;
        this.Game       = Game;
        showPig        = false;
        setVisible(true);
        setMove(true);
    }
    public void updateState()
    {
        if(active == true)
        {
            if(R.nextInt(seed) % 100 < 1)
            {
                if(showPig == false)
                    showPig = true;
            }
            else if(R.nextInt(seed) % 100 > 95)
            {
                if(showPig == true)
                    showPig = false;
            }
        }
    }
    public void paintSprite(Graphics g)
    {
        if(visible == true)
        {
            g.drawImage(Frame,X,Y,Game);
            if(showPig == true)
                g.drawImage(SpriteImage,X + 12,Y + 18,Game);
        }
    }
    public void setSeed(int seed)
    {
        this.seed = seed;
    }
    public boolean hit(int X,int Y,int P_Width,int P_Height,int H_Width,
                      int H_Height)
    {
        if((this.X + P_Width >= X) && (this.Y + (P_Height / 2) >= Y) &&
           (X + (H_Width / 2) >= this.X) && (Y + (H_Height / 2) >= this.Y)
           && showPig)
    }
}

```

```

        showPig = false;
        return true;
    }
    else
        return false;
}
}

class HammerSprite extends SuperSprite
{
    Image hammer1,hammer2,currentImage;
    Applet Game;
    public HammerSprite(Image hammer1,Image hammer2,Applet Game)
    {
        this.hammer1 = hammer1;
        this.hammer2 = hammer2;
        this.Game = Game;
        currentImage = hammer1;
        setLocation(0,0);
        setVisible(false);
        setMove(false);
    }
    public void updateState()
    {
        if(currentImage == hammer1)
            currentImage = hammer2;
        else
            currentImage = hammer1;
    }
    public void paintSprite(Graphics g)
    {
        if(visible == true)
            g.drawImage(currentImage,X,Y,Game);
    }
}

public class HitPigHeadv extends Applet
    implements Runnable,MouseListener,MouseMotionListener,ActionListener
{
    int AppletWidth,AppletHeight,FrameWidth,FrameHeight,
        countX,countY,HammerWidth,HammerHeight,score,
        CurrentSecond,GameSecond;
    Image frame,pig,hammer1,hammer2,OffScreen,PigHead1,bklImage,
        PigHead2;
}

```

```
Thread      newThread;
Graphics    drawOffScreen;
MediaTracker MT;
PigSprite   pigSprite[];
HammerSprite hammerSprite;
Panel       Status,Control;
Label       Time,Score;
Button      start,end;
boolean     StartGame,EndGame;
StartScreen S_Screen;
CloseDialog CD;
GregorianCalendar time;
public void init()
{
    addMouseListener(this);
    addMouseMotionListener(this);
    CD = new CloseDialog(this,new Frame());
    Time = new Label("时间: 0");
    Score = new Label("得分: 0");
    end = new Button("结束游戏");
    start = new Button("开始游戏");
    end.addActionListener(this);
    start.addActionListener(this);
    end.setEnabled(false);
    Status = new Panel();
    Control = new Panel();
    Status.setLayout(new GridLayout(1,2));
    Control.setLayout(new GridLayout(1,2));
    Status.add(Time);
    Status.add(Score);
    Control.add(start);
    Control.add(end);
    setLayout(new BorderLayout());
    add(Status,BorderLayout.NORTH);
    add(Control,BorderLayout.SOUTH);
    AppletWidth = getSize().width;
    AppletHeight = getSize().height;
    countX      = 3;
    countY      = 3;
    score       = 0;
    GameSecond  = 0;
    CurrentSecond = -1;
```

```

StartGame      = true;
EndGame       = false;
//定义图像
MT      = new MediaTracker(this);
pig     = getImage(getDocumentBase(),"Images/pig.gif");
frame   = getImage(getDocumentBase(),"Images/frame.gif");
hammer1 = getImage(getDocumentBase(),"Images/hammer1.gif");
hammer2 = getImage(getDocumentBase(),"Images/hammer2.gif");
PigHead1 = getImage(getDocumentBase(),"Images/pighead1.gif");
PigHead2 = getImage(getDocumentBase(),"Images/pighead2.gif");
bkImage = getImage(getDocumentBase(),"Images/009.jpg");
MT.addImage(pig,0);
MT.addImage(frame,0);
MT.addImage(hammer1,0);
MT.addImage(hammer2,0);
MT.addImage(PigHead1,0);
MT.addImage(PigHead2,0);
MT.addImage(bkImage,0);
try
{
    MT.waitForAll();
}
catch(InterruptedException E){}
FrameWidth  = frame.getWidth(this);
FrameHeight = frame.getHeight(this);
pigSprite  = new PigSprite[9];
for(int i=0;i<9;i++)
{
    pigSprite[i] = new PigSprite(pig,frame,this);
    pigSprite[i].setLocation(i%countX*FrameWidth,
                           i/countY*FrameHeight);
    pigSprite[i].setSeed(i+100);
}
hammerSprite = new HammerSprite(hammer1,hammer2,this);
HammerWidth  = hammer1.getWidth(this);
HammerHeight = hammer1.getHeight(this);
S_Screen     = new StartScreen(AppletWidth,AppletHeight,this,
                           PigHead1,PigHead2,bkImage);
OffScreen   = createImage(AppletWidth,AppletHeight);
drawOffScreen = OffScreen.getGraphics();
resize(FrameWidth*countX,FrameHeight*countY + 70);
}

```

```
public void start()
{
    newThread = new Thread(this);
    newThread.start();
}

public void stop()
{
    newThread = null;
}

public void paint(Graphics g)
{
    drawOffScreen.clearRect(0,0,AppletWidth,AppletHeight);
    if(StartGame)
        S_Screen.paintScreen(drawOffScreen);
    else
    {
        for(int i=0;i<9;i++)
            pigSprite[i].paintSprite(drawOffScreen);
        hammerSprite.paintSprite(drawOffScreen);
    }
    g.drawImage(OffScreen,0,35,this);
}

public void update(Graphics g)
{
    paint(g);
}

public void run()
{
    while(newThread != null)
    {
        repaint();
        try
        {
            Thread.sleep(33);
        }
        catch(InterruptedException E){}
        if(StartGame)
            S_Screen.UpdateStatus();
        else
        {
            time = new GregorianCalendar();
            if(CurrentSecond != time.get(Calendar.SECOND))

```

```

    {
        CurrentSecond = time.get(Calendar.SECOND);
        GameSecond++;
        Time.setText("时间: " + GameSecond + "秒");
    }
    for(int i=0;i<9;i++)
        pigSprite[i].updateState();
}
}

public void endGame(boolean isEndGame)
{
    EndGame = isEndGame;
}

public void mouseExited(MouseEvent e)
{
    hammerSprite.setVisible(false);
    hammerSprite.setLocation(0,0);
}

public void mouseClicked(MouseEvent e){ }

public void mouseEntered(MouseEvent e)
{
    hammerSprite.setVisible(true);
    hammerSprite.setLocation(e.getX() - (HammerWidth / 2),
                           e.getY() - (HammerHeight / 2) + 35);
    showStatus("X:" + e.getX() + "," + "Y:" + e.getY());
}

public void mousePressed(MouseEvent e)
{
    if(StartGame) return;
    hammerSprite.updateState();
    int X = hammerSprite.getX();
    int Y = hammerSprite.getY();
    for(int i=0;i<9;i++)
    {
        if(pigSprite[i].hit(X,Y,FrameWidth,FrameHeight,HammerWidth,
                            HammerHeight) == true)
        {
            score = score + 10;
            Score.setText("得分: " + score);
        }
    }
}

```

```
        }
        public void mouseReleased(MouseEvent e)
        {
            if(StartGame) return;
            hammerSprite.updateState();
        }
        public void mouseMoved(MouseEvent e)
        {
            hammerSprite.setLocation(e.getX() - (HammerWidth / 2),
                                     e.getY() - (HammerHeight / 2) - 35);
            showStatus("X:" + e.getX() + "," + "Y:" + e.getY());
        }
        public void mouseDragged(MouseEvent e)
        {
            hammerSprite.setLocation(e.getX() - (HammerWidth / 2),
                                     e.getY() - (HammerHeight / 2) - 35);
        }
        public void actionPerformed(ActionEvent e)
        {
            if(e.getSource() == start)
            {
                StartGame = false;
                start.setEnabled(false);
                end.setEnabled(true);
            }
            if(e.getSource() == end)
            {
                newThread = null;
                CD.show();
                if(EndGame)
                {
                    score = 0;
                    GameSecond = 0;
                    CurrentSecond = -1;
                    StartGame = true;
                    EndGame = false;
                    start.setEnabled(true);
                    end.setEnabled(false);
                    Time.setText("时间: 0");
                    Score.setText("得分: 0");
                }
                newThread = new Thread(this);
            }
        }
    }
```

```

        newThread.start();
    }
}
}

class StartScreen
{
    int      width,height,StringWidth,StringHeight,Ascent,Descent,X,Y;
    int      ImageLeftBound,ImageRightBound,ImageX,ImageY,ImageWidth,
            ImageHeight,VX;
    Font    F1,F2,F3;
    Image   Normal,Hit,currentImage,bkImage;
    String  ChineseTitle,EnglishTitle,PressEnter;
    Applet  Game;
    Random  R;
    boolean showPressEnter;
    FontMetrics FM;
    public StartScreen(int AppletWidth,int AppletHeight,HitPigHeadv2 Game,
                        Image normal,Image hit, Image bk)
    {
        R      = new Random();
        F1     = new Font("TimesRoman",Font.BOLD,72);
        F2     = new Font("TimesRoman",Font.BOLD + Font.ITALIC,36);
        F3     = new Font("TimesRoman",Font.BOLD,20);
        ChineseTitle      = "棒打猪头";
        EnglishTitle      = "Hit Pig's Head";
        width             = AppletWidth;
        height            = AppletHeight;
        Normal            = normal;
        Hit               = hit;
        bkImage           = bk;
        ImageWidth        = Normal.getWidth(Game);
        ImageHeight       = Normal.getHeight(Game);
        ImageLeftBound    = 25;
        ImageRightBound   = AppletWidth - (25 + ImageWidth);
        ImageX            = ImageRightBound;
        VX                = -2;
        this.Game          = Game;
        currentImage      = Normal;
        showPressEnter    = true;
    }
    public void UpdateStatus()
    {

```

```

        ImageX = ImageX + VX;
        if(ImageX <= ImageLeftBound)
        {
            currentImage = Hit;
            ImageX      = ImageLeftBound;
            VX          = -VX;
        }
        if(ImageX >= ImageRightBound)
        {
            currentImage = Normal;
            ImageX      = ImageRightBound;
            VX          = -VX;
        }
    }
    public void paintScreen(Graphics g)
    {
        g.setFont(F1);
        FM           = g.getFontMetrics();
        Ascent       = FM.getAscent();
        Descent     = FM.getDescent();
        StringWidth = FM.stringWidth(ChineseTitle);
        StringHeight = Ascent + Descent;
        X            = (width - StringWidth) / 2;
        Y            = Ascent;
        g.drawImage(bkImage, 0, 0, Game);
        g.setColor(Color.white);
        g.drawString(ChineseTitle,X,Y);
        Y           = StringHeight;
        g.drawLine(X,Y,X+StringWidth,Y);
        X           = X + 30;
        Y           = Y + 5;
        g.drawLine(X,Y,X+StringWidth-60,Y);
        g.setFont(F2);
        FM           = g.getFontMetrics();
        Ascent       = FM.getAscent();
        Descent     = FM.getDescent();
        StringWidth = FM.stringWidth(EnglishTitle);
        StringHeight = Ascent + Descent;
        X            = (width - StringWidth) / 2;
        Y            = Y + Ascent;
        g.drawString(EnglishTitle,X,Y);
        ImageY      = Y + Descent + 30;
    }
}

```

```

        g.drawImage(currentImage,ImageX,ImageY,Game);
    }
}

class CloseDialog extends Dialog implements ActionListener
{
    Panel      P1,P2;
    Button     B1,B2;
    HitPigHeadv2 Game;
    public CloseDialog(HitPigHeadv2 Game,Frame owner)
    {
        super(owner,"离开游戏...",true);
        this.Game = Game;
        setLayout(new GridLayout(2,1));
        P1 = new Panel();
        P2 = new Panel();
        P1.add(new Label("真的要离开吗????"));
        P2.add(B1 = new Button("确定"));
        P2.add(B2 = new Button("取消"));
        B1.addActionListener(this);
        B2.addActionListener(this);
        add(P1);
        add(P2);
        pack();
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == B1)
            Game.endGame(true);
        else if(e.getSource() == B2)
            Game.endGame(false);
        hide();
    }
}

```

本 章 小 结

事实上，Java和游戏市场已紧密结合成为新一代网络游戏的主角。而且Java和游戏市场的结合已经在视讯游戏业界扮演越来越重要的角色，也正是由于Java多媒体技术、多线程技术的简单易用，具有很好的移植性，才使得Java在当前手机游戏开发中成为首选的开发语言。

第5章

Java与文件操作



本章内容

- 案例1 目录列表的显示
- 案例2 检查与创建目录
- 案例3 文件复制
- 案例4 文件查看器
- 案例5 字符串的查找和替换
- 案例6 用RandomAccessFile类实现文件的追加
- 案例7 用Zip进行多文件保存
- 案例8 用JUNIT建立测试类
- 案例9 用Java保存位图文件
- 案例10 获知文件的属性
- 本章小结



案例1 目录列表的显示

案例运行效果与操作

File类代表目录内一系列文件的名字，可用list()方法查询这个文件集，返回的是一个字符串数组。若想得到一个不同的目录列表，只需创建一个不同的File对象即可。本案例将向大家完整地展示如何使用这个类，其中包括相关的FilenameFilter（文件名过滤器）接口。如果

想观看一个目录列表。可用两种方式列出File对象。若在不含自变量（参数）的情况下调用list()，会获得File对象包含的一个完整列表。然而，若想对这个列表进行某些限制，就需要使用一个“目录过滤器”，该类的作用是指出如何选择File对象来完成显示。

在不含自变量（参数）的情况下，运行效果如图5-1所示。



图5-1 运行界面1

在包含自变量（参数）的情况下，运行效果如图5-2所示。



图5-2 运行界面2

制作要点

1. 用DirectoryFilter类的方法，实现FilenameFilter接口，创建一个“目录过滤器”
2. File类的基本使用：显示目录列表

步骤详解

1. 创建DirectoryFilter类。

DirectoryFilter类实现了FilenameFilter接口。FilenameFilter接口非常简单，格式如下：

```
public interface FilenameFilter {
```

```
    boolean accept(文件目录, 字串名);
}
```

它为这种类型的所有对象都提供了一个名为accept()的方法。之所以要创建这样一个类，原因就是把accept()方法提供给list()方法，使list()能够“回调”accept()，从而判断应将哪些文件名包含到列表中。因此，通常将这种技术称为“回调”，有时也称为“算子”（也就是说，**DirectoryFilter**是一个算子，因为它惟一的作用就是容纳一个方法）。由于list()采用一个**FilenameFilter**对象作为自己的自变量，所以我们能传递实现了**FilenameFilter**的任何类的一个对象，用它决定（甚至在运行期）list()方法的行为方式。回调的目的是在代码的行为上提供更大的灵活性。

accept()方法必须接纳一个File对象，用它指示用于寻找一个特定文件的目录，并接纳一个String，其中包含了要寻找的文件的名字。可决定使用或忽略这两个参数之一，但有时至少要使用文件名。list()方法准备为目录对象中的每个文件名调用accept()，核实哪个应包含在内——具体由accept()返回的“布尔”结果决定。

为确定我们操作的只是文件名，其中没有包含路径信息，必须采用String对象，并在它的外部创建一个File对象。然后调用getName()，它的作用是去除所有路径信息（采用与平台无关的方式）。随后，accept()用String类的indexOf()方法检查文件名内部是否存在搜索字串“myString”。若在字串内找到myString，那么返回值就是myString的起点索引；但假如没有找到，返回值就是-1。

2. 应用File类中的list方法，显示目录列表。

list()方法返回的是一个数组。可查询这个数组的长度，然后在其中遍历，选定数组元素。

程序源代码与解析

```
/*
 * 本案例使用File类的list()方法，实现了目录的列表显示。
 */
import java.io.*;
public class DirectoryList {
    public static void main(String[] args) { //主方法
        try {
            File path = new File(".");
            //定义一个File对象
            String[] myList; //定义一个字符串数组
            if (args.length == 0) //不含自变量则显示所有文件
                myList = path.list();
            else //利用过滤器显示相关文件
                myList = path.list(new DirectoryFilter(args[0]));
            for (int i = 0; i < myList.length; i++) //输出文件列表
                System.out.println(myList[i]);
        }
    }
}
```

```

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class DirectoryFilter implements FilenameFilter { //文件过滤器方法
    String myString;

    DirectoryFilter(String myString) {
        this.myString = myString;
    }

    public boolean accept(File dir, String name) {
        //去除所有路径信息
        String f = new File(name).getName();
        return f.indexOf(myString) != -1;
    }
}
}

```



案例2 检查与创建目录

案例运行效果与操作

File类并不仅仅是对现有目录路径、文件或者文件组的一个表示，也可用一个File对象新建一个目录，甚至创建一个完整的目录路径。或用它了解文件的属性（长度、上一次修改日期、读/写属性等），检查一个File对象到底代表一个文件还是一个目录，以及删除一个文件等等。该案例完整展示了如何运用File类剩下的这些方法。

创建目录的运行画面如图5-3所示。

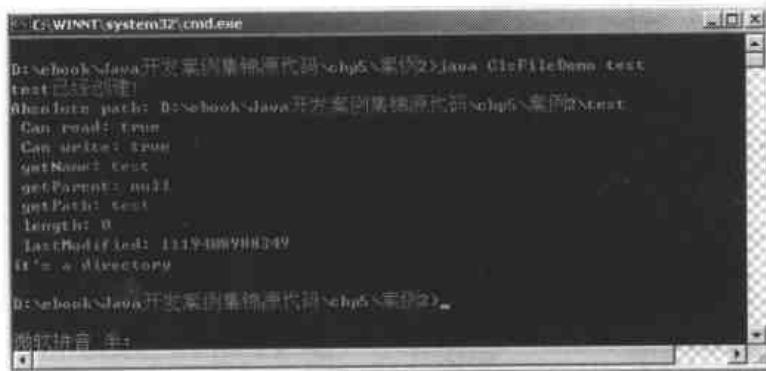


图5-3 创建一个目录

由图中可见，成功创建了test目录，同时，详细列出了目录的属性。

删除目录的运行画面如图5-4所示。由图中可见，成功删除了test目录。



图5-4 删除一个目录

重命名目录的运行画面如图5-5所示。



图5-5 对目录进行重命名

不带参数时的运行画面如图5-6所示。

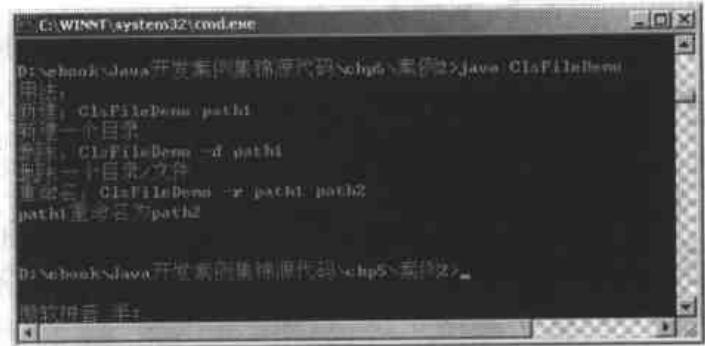


图5-6 不带参数时的运行画面

由图中可见，如果运行时不带参数，则给出程序的使用方法，提示用户。

操作要点

1. File类的基本使用：显示目录属性
2. File类的基本使用：创建、删除或者重命名一个目录

步骤讲解

1. 为便于使用，在用户不输入自变量的情况下，显示类的用法信息。
2. 在fileInfo()中，首先应用各种文件属性方法来显示与文件或目录路径有关的信息。
3. main()应用的第一个方法是renameTo()，利用它可以重命名（或移动）一个文件至一个全新的路径（该路径由参数决定），它属于另一个File对象。这也适用于任何长度的目录。
4. 调用delete()方法，删除一个目录路径。
5. 调用mkdirs()方法，使得本案例可以创建任意复杂程度的一个目录路径。

核心代码解析

```
/*
 * 本案例演示用File类来创建目录并进行文件操作
 */
import java.io.*;

public class ClsFileDemo {
    private final static String myUsage = "用法: \n" + "新建: ClsFileDemo path1 \n"
        + "新建一个目录\n" + "删除: ClsFileDemo -d path1 \n" + "删除一个目录/文件\n"
        + "重命名: ClsFileDemo -r path1 path2\n" + "path1重命名为path2\n";

    private static void myUsage() {
        System.err.println(myUsage); //显示程序的使用方法
        System.exit(1);
    }

    private static void fileInfo(File f) { //该方法用来显示属性信息
        System.out.println( //显示属性信息
            "Absolute path: " + f.getAbsolutePath() + "\n Can read: "
            + f.canRead() + "\n Can write: " + f.canWrite()
            + "\n getName: " + f.getName() + "\n getParent: "
            + f.getParent() + "\n getPath: " + f.getPath()
            + "\n length: " + f.length() + "\n lastModified: "
            + f.lastModified());
        if (f.isFile())
            System.out.println("it's a file");
        else if (f.isDirectory())
            System.out.println("it's a directory");
    }
}
```

```

    }

    public static void main(String[] args) { //主方法
        if (args.length < 1)
            myUsage(); //无输入参数时显示用法信息
        if (args[0].equals("-r")) {
            if (args.length != 3)
                myUsage();
            File old = new File(args[1]), rname = new File(args[2]);
            old.renameTo(rname); //重命名
            System.out.println(old + " 重命名为: " + rname);
            fileInfo(old);
            fileInfo(rname);
            return; //退出
        }
        int count = 0;
        boolean del = false;
        if (args[0].equals("-d")) {
            count++;
            del = true;
        }
        for (; count < args.length; count++) {
            File f = new File(args[count]);
            if (f.exists()) { //存在则显示目录信息或删除目录
                System.out.println(f + " 已经存在");
                if (del) {
                    System.out.println("正在删除..." + f);
                    f.delete();
                    System.out.println(f + "已经删除！");
                }
            } else { //不存在则创建目录
                if (!del) {
                    f.mkdirs();
                    System.out.println(f + "已经创建！");
                }
            }
            fileInfo(f); //调用fileInfo方法
        }
    }
}

```



案例3 文件复制

案例运行效果与操作

本案例实现了文件复制功能，并且定义了一个可供其它程序使用的文件复制方法copyFile()。

如果目标文件存在，首先要确定它是可写的文件，并且在写入前询问用户。

如果目标是一个目录，则将源文件名作为目标文件名。

运行效果如图5-7所示：



图5-7 运行界面

制作要点

1. File类的使用
2. FileInputStream()输入流方法的使用
3. FileOutputStream()输出流方法的使用

步骤详解

1. 利用File类查看文件属性。

在复制文件之前，需要用File类进行一些测试，读取文件的各种属性，以确定一切正常。

首先定义两个文件，它们都是File类的实例：

```

File resourceFile = new File (resourceFileName);
File targetFile= new File (targetFileName);

```

确定源文件存在，并且可读：

```
if (!resourceFile.exists())
```

```
abort("未发现源文件: " + resourceFileName);
if (!resourceFile.isFile())
    abort("不能复制目录: " + resourceFileName);
if (!resourceFile.canRead())
    abort("源文件: " + resourceFileName + "不是可读文件!");
```

如果目标是一个目录，则将源文件名作为目标文件名：

```
if (targetFile.isDirectory())
    targetFile= new File(targetFile, resourceFile.getName());
```

2. 应用FileInputStream()输入流方法和FileOutputStream()输出流方法，进行文件复制。在JDK 1.0中，通常是用InputStream & OutputStream这两个基类来进行读写操作的。InputStream中的FileInputStream类似一个文件句柄，通过它来对文件进行操作，类似的，在OutputStream中有FileOutputStream这个对象。这二者可以对文本文件或二进制文件进行操作。

开始复制文件：

```
resource = new FileInputStream(resourceFile); //创建输入流
target = new FileOutputStream(targetFile); //创建输出流
byte[] buffer = new byte[4096]; //每次读取4K字符
```

将一段字节流读到缓存中，然后将它们写出来：

```
while ((byteNum = resource.read(buffer)) != -1) //读到文件末尾
    target.write(buffer, 0, byteNum); //写入
```

完整代码与解析

```
/*
 * 本案例能独立实现复制文件,
 * 并且定义了一个可供其它程序使用的文件复制方法copyFile()
 */
import java.io.*;
public class FileCopy {
    //程序的main()方法
    public static void main(String[] args) {
        if (args.length != 2) //检查参数
            System.err.println("使用方法: java FileCopy <源文件名> <目的文件名>");
        else {
            //调用copyFile()方法, 实现复制文件, 并显示错误信息
            try {
                copyFile(args[0], args[1]);
            } catch (IOException e) {
```

```

        System.err.println(e.getMessage());
    }
}

//静态方法copyFile()真正实现文件复制
//在复制文件之前，进行一些测试，以确定一切正常
public static void copyFile(String resourceFileName, String targetFileName)
    throws IOException {
    //首先定义两个文件，它们都是File类的实例
    File resourceFile = new File (resourceFileName);
    File targetFile= new File (targetFileName);
    //首先确定源文件存在，并且是可读的文件
    if (!resourceFile.exists())
        abort("未发现源文件: " + resourceFileName);
    if (!resourceFile.isFile())
        abort("不能复制目录: " + resourceFileName);
    if (!resourceFile.canRead())
        abort("源文件: " + resourceFileName +"不是可读文件！");
    //如果目标是一个目录，则将源文件名作为目标文件名
    if (targetFile.isDirectory())
        targetFile= new File(targetFile, resourceFile.getName());
    //如果目标文件存在，首先要确定它是可写的文件
    //并且在写入前询问用户
    //如果目标文件不存在，那么要确定该目录存在并且是可写的
    if (targetFile.exists()) {
        if (!targetFile.canWrite())
            abort("目标文件不是可写文件: " + targetFileName);
        //询问用户是否覆盖目标文件
        System.out.print("是否覆盖现在的文件 " + targetFile.getName() + "? (Y/N): ");
        System.out.flush();
        //得到用户的反馈信息
        BufferedReader in = new BufferedReader(new InputStreamReader(
            System.in));
        String response = in.readLine();
        //检测用户反馈的信息
        //如果得到的答案是否定的，那么取消复制操作
        if (!response.equals("Y") && !response.equals("y"))
            abort("用户取消复制操作。");
    } else {
        //如果文件不存在，检查目录是否存在，该目录是否是可写的
        //如果getParent()方法返回空值，说明该目录为当前目录
        String parent = targetFile.getParent(); //目标目录
    }
}

```

```

        if (parent == null) //如果是空值，使用当前目录
            parent = System.getProperty("user.dir");
        File directory = new File(parent); //将它转换成文件
        if (!directory.exists())
            abort("目标目录 " + parent + "不存在。");
        if (directory.isFile())
            abort("指定的目标 " + parent + "不是目录。");
        if (!directory.canWrite())
            abort("指定的目标目录 " + parent + "不是可写的。");
    }
    //开始复制文件
    FileInputStream resource = null; //读取源文件的数据流
    FileOutputStream target = null; //写入目标文件的数据流
    try {
        resource = new FileInputStream(resourceFile); //创建输入流
        target = new FileOutputStream(targetFile); //创建输出流
        byte[] buffer = new byte[4096]; //每次读取4K字符
        int byteNum; //缓存中的字节数目
        //将一段字节流读到缓存中，然后将它们写出来，
        //循环直到文件末尾（当read()返回-1时）停止
        while ((byteNum = resource.read(buffer)) != -1)
            //读取流直到文件末尾
            target.write(buffer, 0, byteNum); //写入
            System.out.print("文件复制成功！");
    }
    //关闭流
    finally {
        if (resource != null)
            try {
                resource.close();
            } catch (IOException e) {
                ;
            }
        if (target != null)
            try {
                target.close();
            } catch (IOException e) {
                ;
            }
    }
}
//抛出异常

```

```

private static void abort(String msg) throws IOException {
    throw new IOException("文件复制: " + msg);
}
}

```



案例4 文件查看器

案例运行效果与操作

本案例创建一个包含TextArea区域的窗口，在该区域中显示打开的文本文件的内容，初始运行画面如图5-8所示：

单击“打开文件”按钮，会弹出一个对话框，要求选取要显示的文件，如图5-9所示：

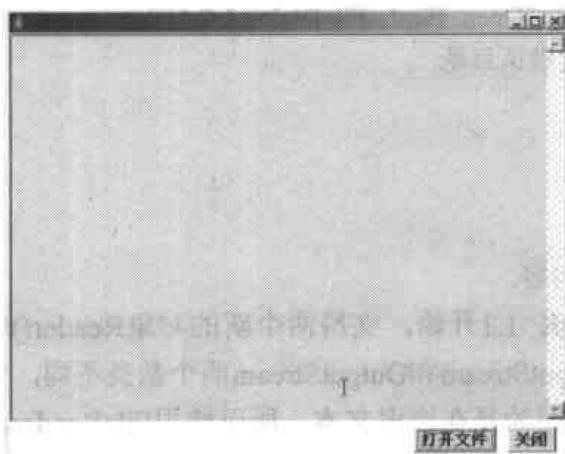


图5-8 初始运行画面

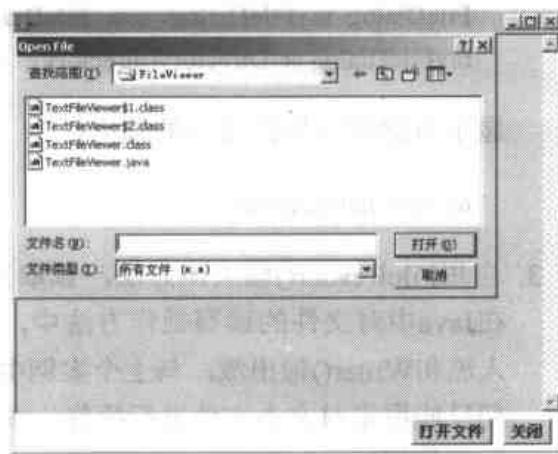


图5-9 “打开文件”对话框

在对话框中选择要显示的文件名，该文件内容就会显示在TextArea文本域中，如图5-10所示：

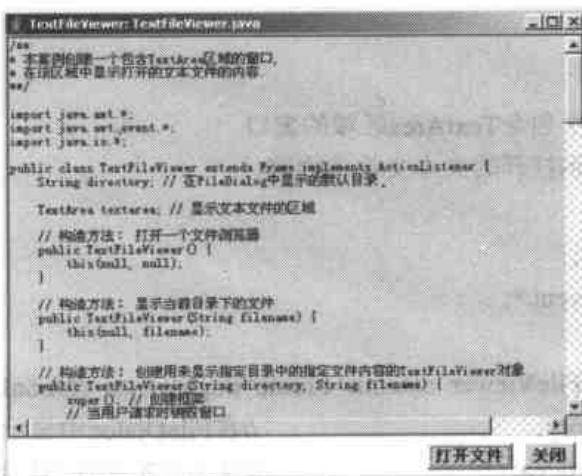


图5-10 显示文件内容

制作要点

1. File类的使用
2. FileReader()输入流方法的使用
3. FileDialog类的使用

制作步骤

1. File类用来获取文件属性，用法同前面的案例。
2. 应用FileDialog类处理与用户的交互。

FileDialog类显示出一个对话框窗口，用户可以从中选择文件。

因为它是一个模式对话框，当应用调用它的 show 方法来显示对话框时，它会阻塞应用的其余部分直到用户选择了一个文件。

创建一个文件对话框，提示输入新的文件：

```
FileDialog myFileDialog=new FileDialog(this,"Open File",FileDialog.LOAD);
myFileDialog.setDirectory(directory); //设置默认目录
```

显示对话框并等待用户的输入：

```
myFileDialog.show();
```

3. 应用FileReader()输入流方法，读取文件内容。

在Java中对文件的读写操作方法中，从JDK 1.1开始，支持两个新的对象Reader()输入流和Writer()输出流，与上个案例中的InputStream和OutputStream两个基类不同，它们只能用来对文本文件进行操作。本案例因为是在读取文本，所以使用FileReader，而不使用FileInputStream。

```
FileReader in = null;
in = new FileReader(f); //创建一个用来读取文件的字符流
```

制作代码编写

```
/*
 * 本案例创建一个包含TextArea区域的窗口
 * 在该区域中显示打开的文本文件的内容
 */
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class TextFileViewer extends Frame implements ActionListener {
    String directory; //在FileDialog中显示的默认目录
    TextArea textarea; //显示文本文件的区域
```

```
//构造方法： 打开一个文件浏览器
public TextFileViewer() {
    this(null, null);
}

//构造方法： 显示当前目录下的文件
public TextFileViewer(String filename) {
    this(null, filename);
}

//构造方法： 创建用来显示指定目录中的指定文件内容的TextFileViewer对象
public TextFileViewer(String directory, String filename) {
    super(); //创建框架
    //当用户请求时销毁窗口
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            dispose();
        }
    });
    //创建一个用来显示文件内容的TextArea区域
    textarea = new TextArea("", 24, 80);
    textarea.setFont(new Font("MonoSpaced", Font.PLAIN, 12));
    textarea.setEditable(false);
    this.add("Center", textarea);
    //创建一个包含两个按钮控件的面板
    Panel myPanel = new Panel();
    myPanel.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 5));
    this.add(myPanel, "South");
    //创建按钮控件，并且处理单击按钮事件
    Font font = new Font("SansSerif", Font.BOLD, 14);
    Button btnOpenFile = new Button("打开文件");
    Button btnClose = new Button("关闭");
    btnOpenFile.addActionListener(this);
    btnOpenFile.setActionCommand("open");
    btnOpenFile.setFont(font);
    btnClose.addActionListener(this);
    btnClose.setActionCommand("close");
    btnClose.setFont(font);
    myPanel.add(btnOpenFile);
    myPanel.add(btnClose);
    this.pack();
    //指明目录
    if (directory == null) {
        File f;
```

```

        if ((filename != null) && (f = new File(filename)).isAbsolute()) {
            directory = f.getParent();
            filename = f.getName();
        } else
            directory = System.getProperty("user.dir");
    }
    this.directory = directory; //记住目录
    setFile(directory, filename); //载入并显示文件
}
//载入并且显示指定目录中的指定文件内容
public void setFile(String directory, String filename) {
    if ((filename == null) || (filename.length() == 0))
        return;
    File f;
    FileReader in = null;
    //读取并且显示文件内容
    //因为是在读取文本，所以使用FileReader，而不使用InputStream
    try {
        f = new File(directory, filename); //创建一个file对象
        in = new FileReader(f);           //和一个用来读取它的字符流
        char[] buffer = new char[4096];   //每次读取4K字符
        int len;                         //每次读入的字符数
        textarea.setText("");           //清除文本区域
        while ((len = in.read(buffer)) != -1) { //读取一批字符
            String s = new String(buffer, 0, len); //转化为一个字符串
            textarea.append(s);                 //显示文本
        }
        this.setTitle("TextFileViewer: " + filename); //设置窗口标题
        textarea.setCaretPosition(0);           //设置文件起始位置
    }
    //显示错误信息
    catch (IOException e) {
        textarea.setText(e.getClass().getName() + ": " + e.getMessage());
        this.setTitle("TextFileViewer: " + filename + ": I/O Exception");
    }
    //关闭输入流
    finally {
        try {
            if (in != null)
                in.close();
        } catch (IOException e) {
        }
    }
}

```

```

        }

    }

    //处理单击按钮事件
    public void actionPerformed(ActionEvent e) {
        String cmd = e.getActionCommand();
        if (cmd.equals("open")) { // 如果用户单击了“打开”按钮
            //创建一个文件对话框，提示输入新的文件
            FileDialog myFileDialog = new FileDialog(this, "Open File", FileDialog.LOAD);
            myFileDialog.setDirectory(directory); //设置默认目录
            //显示对话框并等待用户的输入
            myFileDialog.show();
            directory = myFileDialog.getDirectory(); //记住新的默认目录
            setFile(directory, myFileDialog.getFile()); //载入并显示选择
            myFileDialog.dispose(); //关闭对话框
        } else if (cmd.equals("close")) //如果用户单击了“关闭”按钮
            this.dispose(); //关闭窗口
    }

    //TextFileViewer可以被其他程序所使用
    //也可以加上main()方法，成为一个独立程序。
    static public void main(String[] args) throws IOException {
        //创建一个TextFileViewer对象
        Frame myFrame = new TextFileViewer((args.length == 1) ? args[0] : null);
        //当TextFileViewer窗口关闭时，退出
        myFrame.addWindowListener(new WindowAdapter() {
            public void windowClosed(WindowEvent e) {
                System.exit(0);
            }
        });
        //弹出一个窗口
        myFrame.show();
    }
}

```



案例5 字符串的查找和替换

案例运行效果与操作

本案例实现文本的查找和替换功能，类似于Word的查找和替换。它利用Panel的嵌套实现界面，核心是两个方法成员，一个查找方法，一个替换方法。

运行画面如图5-11所示：

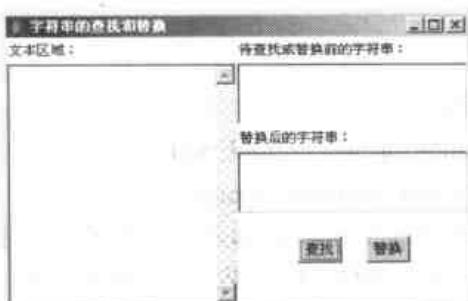


图5-11 运行画面

制作要点

1. Panel类的使用（编写主界面）
2. StringBuffer类的使用
3. 事件处理，实现监听器接口

步骤详解

1. 编写主界面：

```
myFrame = new Frame("字符串的查找和替换");
```

2. 将字符串转化成StringBuffer型进行操作：

字符串的使用贯穿于绝大多数应用程序，不管是作为用户界面的标识或在后台处理从数据库取回的值。通常，这些值并不符合要求，需要处理。可以使用String类，但是它并不是设计成处理动态值的：String对象是常量字符串。一旦被初始化和赋值，它的值和所分配的内存就被固定了。如果硬要改变它的值，将会产生一个包含新值的新String对象。这就是String对象会消耗掉很多资源的原因。而StringBuffer类正好填补了这个需求并使得系统资源的利用更加有效。StringBuffer类被设计用于创建和操作动态字符串信息。为该对象分配的内存会自动扩展以容纳新增的文本。有三种方法来创建一个新的StringBuffer对象：使用初始化字符串、设定大小以及使用默认构造函数：

```
StringBuffer sb=new StringBuffer();
StringBuffer sb=new StringBuffer(30);
StringBuffer sb=new StringBuffer("Builder.com");
```

第一行创建了不包含任何文本的对象，默认的容量是16个字符。类的第二个实例也不包含文本，容量是30个字符，最后一行创建了一个拥有初始化值的对象。StringBuffer类位于java.lang基础包中，因此要使用它的话不需要特殊的引入语句。一旦创建了StringBuffer类的对象，就可以使用StringBuffer类的大量方法和属性。

本案例使用第三种方法：将s1转化成StringBuffer型进行操作：

```
repStr = new StringBuffer(s1);
```

3. 添加对事件的处理。

通过监听器来实现对事件的处理。首先在类中实现MouseListener和WindowListener的接口，用来处理MouseEvent或者WindowEvent事件，然后根据需要重写MouseListener和WindowListener接口中的方法。当产生相应事件时，就会调用接口中定义的这些方法。最后调用组件的addMouseListener()或者addWindowListener()方法来安装监听器，用getLabel()方法来区分不同组件产生的事件。

程序源代码与解析

```
//chp5
import java.awt.*;
import java.awt.event.*;

public class StringFindReplace implements MouseListener, WindowListener {
    private Frame myFrame;
    private TextArea myTextArea;
    private TextField myTextField1, myTextField2;
    private Button myButton1;
    private Button myButton2;
    private Panel myPanel1, myPanel2, myPanel3, myPanel4, myPanel5;
    private Label myLabel1, myLabel2, myLabel3;
    Dialog myDialog;
    Label textLab;
    Button myButton = new Button("确认");
    public static void main(String args[]) {
        StringFindReplace myExam = new StringFindReplace();
        myExam.create();
    }
    //create()方法用于创建主界面和创建有关查找和替换字符串结果信息的对话框
    public void create() {
        //下面是创建主界面
        myFrame = new Frame("字符串的查找和替换");
        //myTextArea代表界面上的文本区域
        myTextArea = new TextArea();
        //myTextField1代表用于输入待查找或替换前的字符串的文本框
        myTextField1 = new TextField();
        //myTextField2代表用于输入替换后的字符串的文本框
        myTextField2 = new TextField();
        //初始化两个按钮对象，分别用来实现“查找”和“替换”操作
        myButton1 = new Button("查找");
        myButton2 = new Button("替换");
        //下面的三个Label对象用于显示有关的提示信息
        myLabel1 = new Label("文本区域: ");
    }
}
```

```
myLabel2 = new Label("待查找或替换前的字符串: ");
myLabel3 = new Label("替换后的字符串: ");
//接下来的5个Panel对象用于控制主界面上各组件的位置和大小
myPanel1 = new Panel();
myPanel2 = new Panel();
myPanel3 = new Panel();
myPanel4 = new Panel();
myPanel5 = new Panel();
//myPanel1用于控制文本区域和相关提示信息的相对位置
myPanel1.setLayout(new BorderLayout());
myPanel1.add("North", myLabel1);
myPanel1.add("Center", myTextArea);
//myPanel2用于控制第一个文本框和相关提示信息的相对位置
myPanel2.setLayout(new BorderLayout());
myPanel2.add("North", myLabel2);
myPanel2.add("Center", myTextField1);
//myPanel3用于控制第二个文本框和相关提示信息的相对位置
myPanel3.setLayout(new BorderLayout());
myPanel3.add("North", myLabel3);
myPanel3.add("Center", myTextField2);
//myPanel4用于控制“查找”按钮和“替换”按钮的相对位置
myPanel4.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));
myPanel4.add(myButton1);
myPanel4.add(myButton2);
//myPanel5用于控制myPanel2、myPanel3和myPanel4的相对位置
myPanel5.setLayout(new GridLayout(3, 1));
myPanel5.add(myPanel2);
myPanel5.add(myPanel3);
myPanel5.add(myPanel4);
//最后通过myFrame控制整体的效果
myFrame.setLayout(new GridLayout(1, 2));
myFrame.add(myPanel1);
myFrame.add(myPanel5);
//为“查找”按钮、“替换”按钮和主窗口添加事件监听器
myButton1.addMouseListener(this);
myButton2.addMouseListener(this);
myFrame.addWindowListener(this);
//下面两个语句设置主界面的大小并让主界面可见
myFrame.setSize(400, 260);
myFrame.setVisible(true);
/*
 * 下面的语句用于创建一个对话框，当用户单击“查找”按钮或者“替换”
 * 按钮后，根据结果显示
*/
```

```

* 一个有关结果信息的对话框，对话框上有一个Label组件和一个确认按钮，Label
组件用于显示 查找或替换字符串的次数
*/
myDialog = new Dialog(myFrame);
myDialog.setLayout(new FlowLayout(FlowLayout.CENTER, 40, 20));
textLab = new Label("");
myDialog.add(textLab);
myDialog.add(myButton);
myButton.addMouseListener(this);
myDialog.setSize(200, 100);
}
/** 方法mouseClicked用于处理鼠标单击的事件，也就是当鼠标单击事件发生后，  

程序就会进入该方法中执行 */
public void mouseClicked(MouseEvent e) {
    //下面这个语句用于获得事件源按钮
    Button myBTN = (Button) (e.getSource());
    //下面的if语句处理事件源是“查找”按钮或“替换”按钮时的情况
    if (myBTN.getLabel() == "查找" || myBTN.getLabel() == "替换") {
        /* String型变量myString1和myString2获得文本区域和第一个文本框中的文  

字内容 */
        String myString1 = myTextArea.getText();
        String myString2 = myTextField1.getText();
        //变量matchNum代表字符串匹配的次数，初始值为0
        int matchNum = 0;
        //cursorPos代表光标当前的位置
        int cursorPos = myTextArea.getCaretPosition();
        //实例化一个Match类的对象
        Match myClass = new Match();
        //下面的if语句处理单击“查找”按钮事件
        if (myBTN.getLabel() == "查找") {
            //通过调用Match类的方法strFind计算出字符串匹配的次数
            matchNum = myClass.strFind(myString1, myString2, cursorPos);
            //下面的一行语句重新设置对话框上Label对象的文本内容
            textLab.setText("共找到" + matchNum + "处");
            myDialog.show();
        }
        //接下来的if语句处理单击“替换”按钮事件
        if (myBTN.getLabel() == "替换") {
            //变量myString3获得第二个文本框中的字符串
            String myString3 = myTextField2.getText();
            //通过调用Match类中的strReplace按钮计算字符串匹配次数
            matchNum = myClass.strReplace(myString1, myString2, myString3,
                cursorPos);
        }
    }
}

```

```

        //重新设置对话框上Label对象的文本内容
        textLab.setText("共替换到" + matchNum + "处");
        //下面的语句用于刷新字符串替换后文本区域的文字显示
        StringBuffer taText = myClass.repStr;
        myTextArea.setText(taText.toString());
        myDialog.show();
    }
}

//下面的if语句用于处理事件源为确认按钮时的情况
if (myBTN.getLabel() == "确认") {
    //单击确认按钮后，信息提示对话框消失，主界面显示
    myDialog.hide();
    myFrame.show();
}

//下面的语句重写MouseListener和WindowListener接口中的方法
public void mouseEntered(MouseEvent e) {
}
public void mouseExited(MouseEvent e) {
}
public void mousePressed(MouseEvent e) {
}
public void mouseReleased(MouseEvent e) {
}
//重写windowClosing方法，关闭窗口时，程序退出
public void windowClosing(WindowEvent e) {
    System.exit(0);
}
public void windowOpened(WindowEvent e) {
}
public void windowIconified(WindowEvent e) {
}
public void windowDeiconified(WindowEvent e) {
}
public void windowClosed(WindowEvent e) {
}
public void windowActivated(WindowEvent e) {
}
public void windowDeactivated(WindowEvent e) {
}
}
/** 类Match用于处理有关字符串查找和替换算法 */
class Match {

```

```

StringBuffer repStr;

/* 方法strFind用于实现字符串查找，返回匹配的次数 */
public int strFind(String s1, String s2, int pos) {
    /* 变量i和j分别表示主串和模式串中当前字符串的位置，k表示匹配次数 */
    int i, j, k = 0;
    //pos代表主串中开始比较的位置
    i = pos;
    j = 0;
    while (i < s1.length() && j < s2.length()) {
        if (s1.charAt(i) == s2.charAt(j)) {
            ++i;
            ++j;
            if (j == s2.length()) {
                //j==s2.length()表示字符串匹配成功，匹配次数加1
                k = k + 1;
                //将指示主串和模式串中当前字符的变量i和j进行回退
                i = i - j + 1;
                j = 0;
            }
        } else {
            i = i - j + 1;
            j = 0;
        }
    }
    return k;
}

/* 方法strReplace用于实现字符串替换操作，返回替换的次数 */
public int strReplace(String s1, String s2, String s3, int pos) {
    /* 变量i和j分别表示主串和模式串中当前字符串的位置，k表示匹配次数 */
    int i, j, k = 0;
    i = pos;
    j = 0;
    //将s1转化成StringBuffer型进行操作
    repStr = new StringBuffer(s1);

    while (i < repStr.length() && j < s2.length()) {
        if (repStr.charAt(i) == s2.charAt(j)) {
            ++i;
            ++j;
            if (j == s2.length()) {
                /* j==s2.length()表示字符串匹配成功，匹配次数加1，此外对主串进行字符串替换 */
                k = k + 1;
            }
        }
    }
}

```

```
        repStr.replace(i - j, i, s3);
        //将j进行重新赋值开始新的比较
        j = 0;
    }
} else {
    i = i - j + 1;
    j = 0;
}
return k;
}
```



案例6 利用RandomAccessFile类 来实现文件的追加

案例运行效果与操作

RandomAccessFile是个很好用的类，功能十分强大，本案例利用它的length()和seek()方法来轻松实现文件的追加。

程序运行画面如图5-12所示。

```
C:\WINNT\system32\cmd.exe

D:\chapter6>java AppendText
Value 1:xici,this is line0
Value 2:xici,this is line1
Value 3:xici,this is line2
Value 4:xici,this is line3
Value 5:xici,this is line4
Value 6:xici,this is line5
Value 7:xici,this is line6
Value 8:xici,this is line7
Value 9:xici,this is line8
Value 10:xici,this is line9
Value 11:lala.append line

D:\chapter6>java AppendText
Value 1:xici,this is line0
Value 2:xici,this is line1
Value 3:xici,this is line2
Value 4:xici,this is line3
Value 5:xici,this is line4
Value 6:xici,this is line5
Value 7:xici,this is line6
Value 8:xici,this is line7
Value 9:xici,this is line8
Value 10:xici,this is line9
Value 11:lala.append line
Value 12:lala.append line

D:\chapter6>
```

图5-12 运行画面

由图中可见，每运行一次程序，则加入一行文本，实现了文件的追加功能。

制作要点

1. writeBytes()方法的使用
2. length()和seek()方法的使用
3. 异常处理

步骤详解

1. 先使用writeBytes()方法，写入十行；
2. 用length()读出长度（以byte为单位）；
3. 再用seek()移动到文件末尾；
4. 继续添加一行；
5. 最后显示记录。

程序源代码与解析

```
/*
 *本案例利用RandomAccessFile类的length()和seek()方法,
 *轻松实现文本的追加
 */
import java.io.*;
public class AppendText {
    public static void main(String[] args) {
        try{
            //输出到d:\chapter6\test.txt文件中
            RandomAccessFile rf1 = new
            RandomAccessFile("d:\\chapter6\\test.txt","rw");
            for (int i = 0; i<10; i++){
                rf1.writeBytes("xixi,this is line"+i+"\n"); //先写入10行
            }
            rf1.close();
            int i = 0;
            String record = new String();
            RandomAccessFile rf2 = new
            RandomAccessFile("d:\\chapter6\\test.txt","rw");
            rf2.seek(rf2.length());
            rf2.writeBytes("lala,append line"+"\n"); //追加1行
            rf2.close();
            RandomAccessFile rf3 = new
                RandomAccessFile("d:\\chapter6\\test.txt","r");
            while ((record = rf3.readLine()) != null) {
                i++;
                System.out.println("Value "+i+":"+record); //输出
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
    }  
    rf3.close();  
}catch(Exception e){}  
}  
}
```



案例7 用Zip进行多文件保存

案例运行效果与操作

本案例能根据需要控制任意数量的命令行参数。除此之外，它展示了如何用Checksum类来计算和校验文件的“校验和”（Checksum）。可选用两种类型的Checksum：Adler32（速度要快一些）和CRC32（慢一些，但更准确）。

程序运行画面如图5-13、图5-14所示：

```
C:\WINNT\system32\cmd.exe
B:\chapters>java ZipCompress test.txt
Writing file test.txt
Checksum: D2115883B
Reading file
Checksum: A
reading file test.txt
checksum in line0
checksum in line1
checksum in line2
checksum in line3
checksum in line4
checksum in line5
checksum in line6
checksum in line7
checksum in line8
checksum in line9
Checksum: B
File: test.txt
```

图5-13 单个文件的压缩保存

图5-14 多个文件的压缩保存

制作要点

1. `putNextEntry()`和`getNextEntry()`方法的使用
2. `setComment()`方法的使用
3. `CheckedInputStream`和`CheckedOutputStream`输入/输出流方法的使用
4. `Enumeration`（枚举）方法的使用

步骤详解

1. 对于要加入压缩档的每一个文件，调用`putNextEntry()`，并将其传递给一个`ZipEntry`对象。`ZipEntry`对象包含了一个功能全面的接口，利用它可以获取和设置`Zip`文件内哪个特定的`Entry`（入口）上能够接受的所有数据：名字、压缩后和压缩前的长度、日期、CRC校验和、额外字段的数据、注释、压缩方法以及它是否一个目录入口等等。然而，尽管`CheckedInputStream`和`CheckedOutputStream`同时提供了对Adler32和CRC32校验和的支持，但是`ZipEntry`只支持CRC的接口。
2. 为解压文件，`ZipInputStream`提供了一个`getNextEntry()`方法，能返回下一个`ZipEntry`（前提是下一个`ZipEntry`存在）。作为一个更简洁的方法，可以用`ZipFile`对象读取文件。该对象有一个`entries()`方法，可以为`ZipEntry`返回一个`Enumeration`（枚举）。
3. 为读取校验和，必须拥有对关联的`Checksum`对象的访问权限。在这里保留了指向`CheckedOutputStream`和`CheckedInputStream`对象的一个句柄。但是，也可以只占有指向`Checksum`对象的一个句柄。
4. Zip流中一个令人困惑的方法是`setComment()`。正如前面展示的那样，我们可在写一个文件时设置注释内容，但却没有办法取出`ZipInputStream`内的注释。看起来，似乎只能通过`ZipEntry`逐个人口地提供对注释的完全支持。
5. 使用GZIP或Zip库时并不仅仅限于文件——可以压缩任何东西，包括要通过网络连接发送的数据。

程序源代码与解释

```
//ch5: ZipCompress.java
import java.io.*;
import java.util.*;
import java.util.zip.*;

public class ZipCompress {
    public static void main(String[] args) {
        try {
            //压缩输出到当前目录下的test.zip文件
            FileOutputStream f = new FileOutputStream("test.zip");
            //采用Adler32校验和
            CheckedOutputStream csum =
                new CheckedOutputStream(f, new Adler32());

```

```
ZipOutputStream out =new ZipOutputStream(
new BufferedOutputStream(csum));
//添加压缩文件的注释
out.setComment("A test of Java Zipping");
//依次对每个文件进行压缩
for(int i = 0; i < args.length; i++) {
System.out.println("Writing file " + args[i]);
BufferedReader in =new BufferedReader(new FileReader(args[i]));
out.putNextEntry(new ZipEntry(args[i]));
int c;
while((c = in.read()) != -1)
out.write(c);
in.close();
}
//校验和仅当文件关闭后才有效
out.close();
System.out.println("Checksum: " +csum.getChecksum().getValue());
//从压缩文件中释放文件
System.out.println("Reading file");
FileInputStream fi =new FileInputStream("test.zip");
CheckedInputStream csumi =new CheckedInputStream(fi, new Adler32());
ZipInputStream in2 =new ZipInputStream(new BufferedInputStream(csumi));
ZipEntry ze;
System.out.println("Checksum: " +csumi.getChecksum().getValue());
while((ze = in2.getNextEntry()) != null) {
System.out.println("Reading file " + ze);
int x;
while((x = in2.read()) != -1)
System.out.write(x); //显示文件内容
}
in2.close();
//另一种打开和读取压缩文件的方法：枚举方法
ZipFile zf = new ZipFile("test.zip");
Enumeration e = zf.entries();
while(e.hasMoreElements()) {
ZipEntry ze2 = (ZipEntry)e.nextElement();
System.out.println("File: " + ze2);
}
} catch(Exception e) {
e.printStackTrace();
}
}
```



案例8 用JUNIT建立测试类

案例运行效果与操作

JUnit是一个用来在项目中进行测试和调试的工具。本案例演示如何在Eclipse中建立JUnit测试，如何测试Hello World这样简单的程序。

系统初始界面如图5-15所示。



图5-15 系统初始界面

单击工具栏中的 按钮，对当前测试用例进行测试，在左侧的Junit窗格会显示出测试结果，如图5-16、图5-17所示：

在JUnit的窗口中显示了一个绿条，参看图5-17。绿条证明测试成功。

制造一个错误，例如把HelloWorld.java里的return “Hello World” 改成return “HelloWorld”，编译后运行测试，看到结果报告错误。如图5-18所示。此时JUnit窗口显示一个红条，表明测试出错误，如图5-19所示。具体的错误原因参看图中的故障跟踪部分。

双击故障跟踪中的错误，会弹出比较结果界面，便于了解错误所在，如图5-20所示。

制作要点

1. 被测程序和方法的编写
2. JUnit库的引入
3. JUnit测试类的编写和使用



图5-16 测试无错误的界面

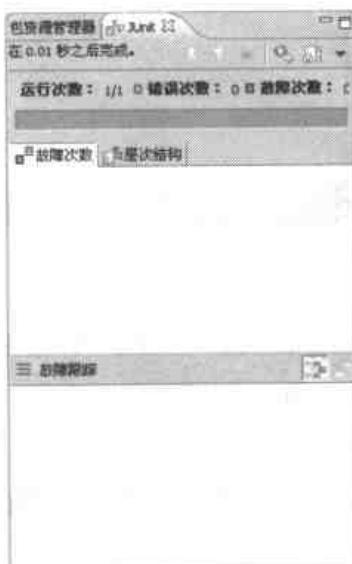


图5-17 Junit窗格: 无错误的界面

4. JUnit测试的运行配置

步骤详解

1. 创建项目。

运行Eclipse，新建一个workplace项目，单击“文件▶新建▶项目”，选择Java项目，单击“下一步”。起一个项目名称，例如mypro。单击“完成”。这样就完成新项目的建立了。

2. 创建被测程序。

现在开发“Hello World”程序。在mypro的标题上面单击右键，选择“新建▶包”，单击下一步。起一个包名，例如hello。单击完成。在hello的标题上面单击右键，选择“新建▶类”，单击下一步。起一个类名HelloWorld.java，单击完成。双击刚刚创建的Hello-World.java类，在编码窗口中输入以下代码：

```
public class HelloWorld{
    public String sayHello(){
        return "Hello World";
    }
    public static void main(String[] args){
        HelloWorld world=new HelloWorld();
        System.out.println(world.sayHello());
    }
}
```

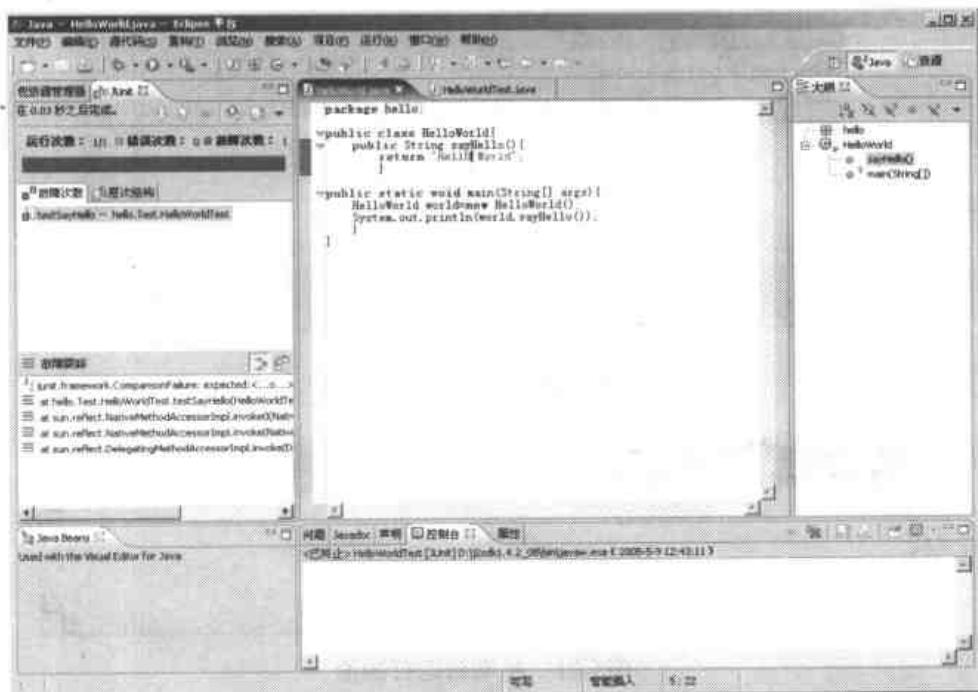


图5-18 测试有错误的界面

单击运行，控制台输出如图5-21所示，表明程序调试运行成功。

3. 建立测试类。

现在写测试类，来测试HelloWorld类里的sayHello方法。在hello的标题上面单击右键，选择“新建▶包”，单击下一步。起一个包名，例如Test，单击完成。在Test的标题上面单击右键，选择新建->JUnit测试用例，出现如图5-22所示对话框：

单击“是”按钮，弹出如图5-23所示窗口：

选择合适的源文件夹和包，起一个类名HelloWorldTest.java，单击完成。双击刚刚创建的HelloWorldTest.java类，在编码窗口中输入以下代码：



图5-19 Junit窗格：有错误的界面

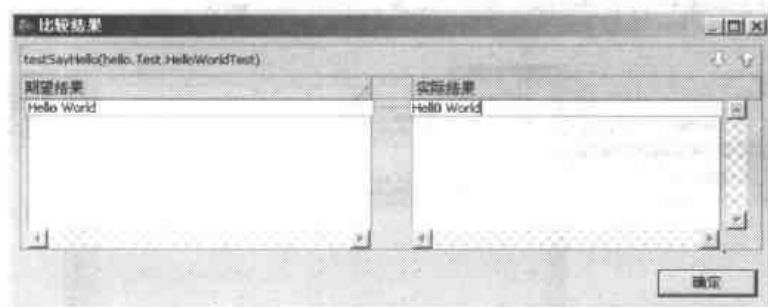


图5-20 比较结果界面

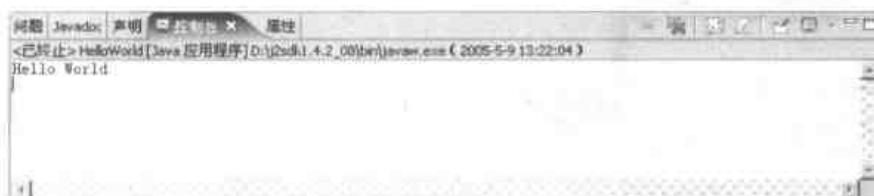


图5-21 控制台运行画面

```
package hello.Test;  
import hello.*;  
import junit.framework.*;  
  
public class HelloWorldTest extends TestCase{  
    public HelloWorldTest(String name){  
        super(name);  
    }  
    public static void main(String args[]){  
        junit.textui.TestRunner.run(HelloWorldTest.class);  
    }  
}
```

```
//实现对SayHello方法的测试
public void testSayHello(){
    HelloWorld world=new HelloWorld();
    assertEquals("Hello World",world.sayHello());
}
```

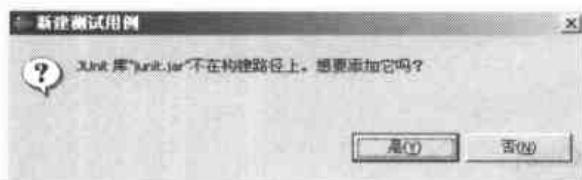


图5-22 新建测试用例对话框



图5-23 新建测试用例窗口

4. 运行测试

在菜单栏中，单击“运行▶运行...”，弹出如图5-24所示窗口：



图5-24 运行配置窗口

单击“运行”按钮即可进行测试。

程序源代码与解析

```

//HelloWorld.java:
package hello;
public class HelloWorld{
    public String sayHello(){          //返回测试字符串的方法
        return "Hello World";
    }
    public static void main(String[] args){ //主方法
        HelloWorld world=new HelloWorld();
        System.out.println(world.sayHello()); //调用sayHello方法
    }
}

//HelloWorldTest.java:
package hello.Test;
import hello.*;
import junit.framework.*;

public class HelloWorldTest extends TestCase{
    public HelloWorldTest(String name){
        super(name);
    }
    public static void main(String args[]){      //主方法
        junit.textui.TestRunner.run(HelloWorldTest.class);
    }
    //实现对SayHello方法的测试
    public void testSayHello(){
        HelloWorld world=new HelloWorld();
        assertEquals("Hello World",world.sayHello());
    }
}

```

**案例9 用Java保存位图文件****案例运行效果与操作**

如果在 Microsoft Windows环境中工作，那么创建位图文件的功能将提供许多方便。虽然Java 提供了几种打开图像的机制，但保存图像并不是其强项；它不支持创建位图。本案例讲述如何将图像保存在24位位图文件中。

运行画面如图5-25所示。

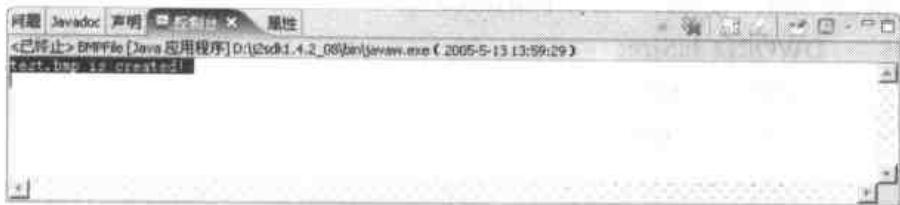


图5-25 运行画面

制作要点

1. 像素捕获器PixelGrabber类的使用
2. 类型强制转换的应用
3. 位操作的应用
4. FileOutputStream输出流的应用，生成位图文件

步骤详解

1. 研究24位位图文件的格式，确定BMPFile类的属性。

位图文件分为三个部分。

第1部分：位图文件的标头

标头包含位图文件的类型大小信息和版面信息。结构如下：

```
typedef struct tagBITMAPFILEHEADER {
    UINT bfType;
    DWORD bfSize;
    UINT bfReserved1;
    UINT bfReserved2;
    DWORD bfOffBits;
}BITMAPFILEHEADER;
```

下面是对这个清单中的代码元素的说明：

bfType: 指定文件类型，其值始终为BM。

bfSize: 指定整个文件的大小（以字节为单位）。

bfReserved1: 保留一必须为0。

bfReserved2: 保留一必须为0。

bfOffBits: 指定从BitmapFileHeader到图像首部的字节偏移量。

也就是说位图标头的用途就是标识位图文件。读取位图文件的每个程序都使用位图标头来进行文件验证。

第2部分：位图信息标头

随后的标头称为信息标头，其中包含图像本身的属性。

下面说明如何指定Windows 3.0（或更高版本）设备独立位图（DIB）的大小和颜色格式：

```

typedef struct tagBITMAPINFOHEADER {
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
} BITMAPINFOHEADER;

```

以上代码清单的每个元素说明如下：

biSize: 指定BITMAPINFOHEADER结构所需的字节数。

biWidth: 指定位图的宽度（以象素为单位）。

biHeight: 指定位图的高度（以象素为单位）。

biPlanes: 指定目标设备的位面数。这个成员变量的值必须为1。

biBitCount: 指定每个象素的位数。其值必须为1、4、8或24。

biCompression: 指定压缩位图的压缩类型。在24位格式中，该变量被设置为0。

biSizeImage: 指定图像的大小（以字节为单位）。如果位图的格式是BI_RGB，则将此成员变量设置为0是有效的。

biXPelsPerMeter: 为位图指定目标设备的水平分辨率（以“象素/米”为单位）。应用程序可用该值从最符合当前设备特征的资源群组中选择一个位图。

biYPelsPerMeter: 为位图指定目标设备的垂直分辨率（以“象素/米”为单位）。

biClrUsed: 指定位图实际所用的颜色表中的颜色索引数。如果biBitCount设为24，则biClrUsed指定用来优化Windows调色板性能的参考颜色表。

biClrImportant: 指定对位图的显示有重要影响的颜色索引数。如果此值为0，则所有颜色都很重要。

现在已定义了创建图像所需的全部信息。

第3部分：图像

在24位格式中，图像中的每个象素都由存储为BRG的三字节RGB序列表示。每个扫描行都被补足到4位。为了使这个过程稍复杂一点，图像是自底而上存储的，即第一个扫描行是图像中的最后一个扫描行。下图显示了标头（BITMAPHEADER）和（BITMAPINFOHEADER）以及部分图像。各个部分由垂线分隔：

```

0000000000 4D42 B536 0002 0000 0000 0036 0000 | 0028
0000000020 0000 0107 0000 00E0 0000 0001 0018 0000
0000000040 0000 B500 0002 0EC4 0000 0EC4 0000 0000
0000000060 0000 0000 0000 | FFFF FFFF FFFF FFFF

```

0000000100 FFFF FFFF FFFF FFFF FFFF FFFF FFFF

根据上述讨论，确定我们所编写的BMPFile类中的属性如下：

位图文件标头：

```
private byte bitmapFileHeader [] = new byte [14];
private byte bfType [] = {'B', 'M'};
```

位图信息标头：

```
private byte bitmapInfoHeader [] = new byte [40];
private int biSize = BITMAPINFOHEADER_SIZE;
private int biSizeImage = 0x030000;
private int biXPelsPerMeter = 0x0;
private int biYPelsPerMeter = 0x0;
```

2. 编写save方法，利用FileOutputStream输出流依次输出位图文件的文件标头、信息表头和图像。
3. 编写convertImage方法，使用像素捕获器从内存读取图像的像素信息，并实现将内存图像转换为位图格式（BRG）。

java.awt.image.PixelGrabber包中的PixelGrabber方法的用法如下：

```
PixelGrabber(Image img, int x, int y, int w, int h, int[] pix, int off, int scansize)
```

其中img是要读取的图像，x/y是要读取图像中的左上角坐标，w/h分别是从x/y开始起的距离，其实x,y,w,h就是一个矩形，pix是保存像素的数组，off是读取图像时开始的位置，scansize是指扫描的宽度，一般来说和w相等。

利用填充，每个扫描行都被补足到4位

```
pad = (4 - ((parWidth * 3) % 4)) * parHeight;
biSizeImage = ((parWidth * parHeight) * 3) + pad; //图像部分的大小
```

整个位图文件的大小（三个部分累加）

```
bfSize = biSizeImage + BITMAPFILEHEADER_SIZE +
BITMAPINFOHEADER_SIZE;
```

4. 编写writeBitmap方法，将像素捕获器返回的图像转换为所需的格式。

不需填充的情况

```
if (pad == 4) pad = 0;
```

扫描行在位图文件中是反向存储的

```
rowIndex = size - biWidth;
```

向每个像素点写入RGB信息

```

value = bitmap [rowIndex];
rgb [0] = (byte) (value & 0xFF);           //强制转换成byte型
rgb [1] = (byte) ((value >> 8) & 0xFF);    //右移8位
rgb [2] = (byte) ((value >> 16) & 0xFF);   //右移16位

```

5. 编写writeBitmapFileHeader和writeBitmapInfoHeader方法，将位图文件标头和位图信息标头写入文件中。其中要编写类型转换方法，将整数转换为单字或双字，再进行写入。
6. 编写Main方法，读入一个本地图像文件到内存中，然后用上述方法将其保存为一个位图文件。

程序源代码与解析

```

//ch5:BMPFile.java:
import java.awt.*;
import java.io.*;
import java.awt.image.*;

public class BMPFile extends Component {
    //--- 私有常量
    private final static int BITMAPFILEHEADER_SIZE = 14;
    private final static int BITMAPINFOHEADER_SIZE = 40;
    //--- 私有变量声明
    private static boolean saveSucess = true; //存储成功的标志
    //--- 位图文件标头
    private byte bitmapFileHeader [] = new byte [14];
    private byte bfType [] = {'B', 'M'};
    private int bfSize = 0;
    private int bfReserved1 = 0;
    private int bfReserved2 = 0;
    private int bfOffBits =
        BITMAPFILEHEADER_SIZE + BITMAPINFOHEADER_SIZE;
    //--- 位图信息标头
    private byte bitmapInfoHeader [] = new byte [40];
    private int biSize = BITMAPINFOHEADER_SIZE;
    private int biWidth = 0;
    private int biHeight = 0;
    private int biPlanes = 1;
    private int biBitCount = 24;
    private int biCompression = 0;
    private int biSizeImage = 0x030000;
}

```

```

private int biXPelsPerMeter = 0x0;
private int biYPelsPerMeter = 0x0;
private int biClrUsed = 0;
private int biClrImportant = 0;

//--- 位图原始数据
private int bitmap [];

//--- 文件部分
private FileOutputStream fo;

//--- 缺省构造函数
public BMPFile() {
}

public static void main(String args[]){
    Toolkit tkit=Toolkit.getDefaultToolkit();
    Image myjpeg=tkit.getImage("d:\\chapter5\\test.jpg");
    BMPFile myfile=new BMPFile();
    //输出到一个文件
    myfile.saveBitmap("d:\\chapter5\\test.bmp",myjpeg,450,350);
    if (saveSucess=true)
        System.out.println("test.bmp is created!");
}

public void saveBitmap (String parFilename, Image parImage, int
parWidth, int parHeight) {
    try {
        fo = new FileOutputStream (parFilename);
        save (parImage, parWidth, parHeight);
        fo.close ();
    }
    catch (Exception saveEx) {
        saveEx.printStackTrace ();
        saveSucess=false;
    }
}

/*
 * save方法是该进程的主方法
 * 该方法将调用convertImage方法以将内存图像转换为字节数组
 * writeBitmapFileHeader方法创建并写入位图文件标头
 * writeBitmapInfoHeader创建信息标头
 * writeBitmap写入图像
 *
 */

```

```

private void save (Image parImage, int parWidth, int parHeight) {
    try {
        convertImage (parImage, parWidth, parHeight);
        writeBitmapFileHeader ();
        writeBitmapInfoHeader ();
        writeBitmap ();
    }
    catch (Exception saveEx) {
        saveEx.printStackTrace ();
    }
}
/*
 * convertImage 将内存图像转换为位图格式 (BRG)。
 * 它还计算位图信息标头所用的某些信息。
 *
 */
private boolean convertImage (Image parImage, int parWidth, int parHeight) {
    int pad;                                //定义保存填充值的变量
    bitmap = new int [parWidth * parHeight];  //定义保存像素的数组
    PixelGrabber pg = new PixelGrabber (parImage, 0, 0, parWidth, parHeight,
                                        bitmap, 0, parWidth);
    try {
        pg.grabPixels ();
    }
    catch (InterruptedException e) {
        e.printStackTrace ();
        return (false);
    }
    //利用填充，每个扫描行都被补足到4位
    pad = (4 - ((parWidth * 3) % 4)) * parHeight;
    biSizeImage = ((parWidth * parHeight) * 3) + pad; //图像部分的大小
    //整个位图文件的大小（三个部分累加）
    bfSize = biSizeImage + BITMAPFILEHEADER_SIZE +
    BITMAPINFOHEADER_SIZE;
    biWidth = parWidth;
    biHeight = parHeight;
    return (true);
}
/*
 * writeBitmap将像素捕获器返回的图像转换为所需的格式
 * 请记住：扫描行在位图文件中是反向存储的
 * 每个扫描行必须补足为4个字节

```

```

/*
private void writeBitmap () {
    int size;
    int value;
    int j;
    int i;
    int rowCount;
    int rowIndex;
    int lastRowIndex;
    int pad;
    int padCount;
    byte rgb [] = new byte [3];
    size = (biWidth * biHeight) - 1;
    //每个扫描行必须补足为4个字节
    pad = 4 - ((biWidth * 3) % 4);
    if (pad == 4)           //不需填充的情况
        pad = 0;
    rowCount = 1;
    padCount = 0;
    rowIndex = size - biWidth;      //扫描行在位图文件中是反向存储的
    lastRowIndex = rowIndex;
    try {
        //向每个像素点写入RGB信息
        for (j = 0; j < size; j++) {
            value = bitmap [rowIndex];
            rgb [0] = (byte) (value & 0xFF);      //强制转换成byte型
            rgb [1] = (byte) ((value >> 8) & 0xFF); //右移8位
            rgb [2] = (byte) ((value >> 16) & 0xFF); //右移16位
            fo.write (rgb);
            if (rowCount == biWidth) {
                padCount += pad;
                for (i = 1; i <= pad; i++) {
                    fo.write (0x00);
                }
                rowCount = 1;
                rowIndex = lastRowIndex - biWidth;
                lastRowIndex = rowIndex;
            }
            else
                rowCount++;
            rowIndex++;
        }
    }
}

```

```

        //更新文件大小
        bfSize += padCount - pad;
        biSizeImage += padCount - pad;
    }
    catch (Exception wb) {
        wb.printStackTrace ();
    }
}

// writeBitmapFileHeader将位图文件标头写入文件中
private void writeBitmapFileHeader () {
    try {
        fo.write (bfType);
        fo.write (intToDWord (bfSize));
        fo.write (intToWord (bfReserved1));
        fo.write (intToWord (bfReserved2));
        fo.write (intToDWord (bfOffBits));
    }
    catch (Exception wbfh) {
        wbfh.printStackTrace ();
    }
}

//writeBitmapInfoHeader将位图信息标头写入文件中
private void writeBitmapInfoHeader () {
    try {
        fo.write (intToDWord (biSize));
        fo.write (intToDWord (biWidth));
        fo.write (intToDWord (biHeight));
        fo.write (intToWord (biPlanes));
        fo.write (intToWord (biBitCount));
        fo.write (intToDWord (biCompression));
        fo.write (intToDWord (biSizeImage));
        fo.write (intToDWord (biXPelsPerMeter));
        fo.write (intToDWord (biYPelsPerMeter));
        fo.write (intToDWord (biClrUsed));
        fo.write (intToDWord (biClrImportant));
    }
    catch (Exception wbih) {
        wbih.printStackTrace ();
    }
}

/** intToWord将整数转换为单字

```

```

    * 返回值存储在一个双字节数组中
    */
private byte [] intToWord (int parValue) {
    byte retValue [] = new byte [2];
    retValue [0] = (byte) (parValue & 0x00FF);
    retValue [1] = (byte) ((parValue >> 8) & 0x00FF);
    return (retValue);
}

/**intToDWord 将整数转换为双字
 *返回值存储在一个4字节数组中
 */
private byte [] intToDWord (int parValue) {
    byte retValue [] = new byte [4];
    retValue [0] = (byte) (parValue & 0x00FF);
    retValue [1] = (byte) ((parValue >> 8) & 0x000000FF);
    retValue [2] = (byte) ((parValue >> 16) & 0x000000FF);
    retValue [3] = (byte) ((parValue >> 24) & 0x000000FF);
    return (retValue);
}
}

```



案例10 获取文件属性

案例运行效果与操作

本案例实现指定文件的文件名、文件类型、是否只读、是否隐藏、最后修改时间等属性。程序运行后，如图5-26所示。

在菜单条内输入文件路径和名称，如图5-27所示。

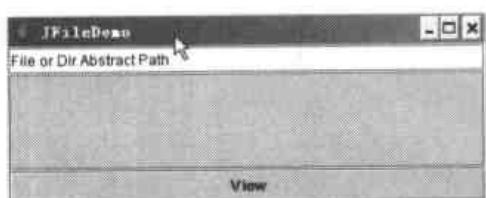


图5-26 运行界面

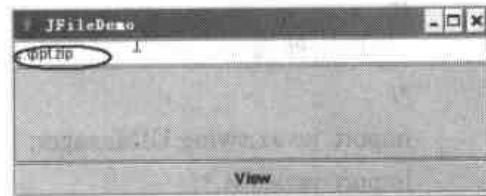


图5-27 输入文件名和路径

单击 **View**，得到如图5-28的结果。

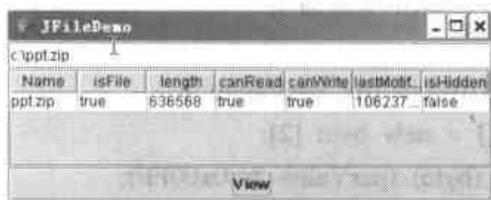


图5-28 结果显示

制作要点**1. File的属性用法****步骤详解**

- 通过exists()判断文件是否存在。

```
if (!theFile.exists())
    return null;
```

- 应用File类中的文件属性值，显示文件属性。

把文件属性放在一数组中，通过输出数组来显示。

```
data[0][0] = theFile.getName();
data[0][1] = String.valueOf(theFile.isFile());
data[0][2] = String.valueOf(theFile.length());
data[0][3] = String.valueOf(theFile.canRead());
data[0][4] = String.valueOf(theFile.canWrite());
data[0][5] = String.valueOf(theFile.lastModified());
data[0][6] = String.valueOf(theFile.isHidden());
return data;
```

程序源代码与解释

```
//chp5
/*
 *返回文件属性
 */
import javax.swing.UIManager;
import java.awt.*;
import javax.swing.JTable;
import java.awt.event.ActionEvent;
import javax.swing.JTextField;
import javax.swing.JScrollPane;
import java.awt.event.WindowEvent;
import javax.swing.JPanel;
```

```

import java.io.File;
import javax.swing.JFrame;
import javax.swing.JButton;

public class FileAttr {
    private boolean packFrame = false;

    //构造方法
    public FileAttr() {
        MainFrame frame = new MainFrame();
        if (packFrame) {
            frame.pack();
        }
        else {
            frame.validate();
        }
        //居中处理
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height) {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
        frameSize.height) / 2);
        frame.setVisible(true);
    }
    //Main()方法
    public static void main(String[] args) {
        new FileAttr();
    }
}
class MainFrame extends JFrame {
    private JPanel contentPane;
    private BorderLayout borderLayout1 = new BorderLayout();
    private JTextField jTextField1 = new JTextField();
    private JScrollPane jScrollPane1 = new JScrollPane();
    private JTable jTable1;
    private JButton jButton1 = new JButton();
    File file;
    //构造窗体
    public MainFrame() {

```

```

enableEvents(AWTEvent.WINDOW_EVENT_MASK);
try {
    jbInit(); //调用jbInit()方法
}
catch(Exception e) {
    e.printStackTrace();
}
//部件初始化
private void jbInit() throws Exception {
    contentPane = (JPanel) this.getContentPane();
    jTextField1.setText("File or Dir Abstract Path"); //设置文本域的文本
    contentPane.setLayout(borderLayout1); //设置布局
    this.setSize(new Dimension(394, 164)); //设置尺寸
    this.setTitle("JFileDemo"); //设置标题
    jScrollPane1.setAutoscrolls(true); //自动滚屏
    jButton1.setText("View"); //按钮名称
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            jButton1ActionPerformed(e);
        }
    });
    contentPane.add(jButton1, BorderLayout.SOUTH); //加入按钮
    contentPane.add(jTextField1, BorderLayout.NORTH); //加入文本域
    contentPane.add(jScrollPane1, BorderLayout.CENTER); //加入ScrollPane
}
//处理窗口事件
protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}
Object[] getFileInfoType(){
    return new Object[]{"Name","isFile","length","canRead","canWrite", "lastModified",
".isHidden"};
}
Object[][] getFileInfo(File file){
    File theFile = file;
    Object[][] data = new Object[1][7];
    //用File类的各种方法获取文件属性，保存到data数组中
    if (!theFile.exists())
        return null;
}

```

```
data[0][0] = theFile.getName();
data[0][1] = String.valueOf(theFile.isFile());
data[0][2] = String.valueOf(theFile.length());
data[0][3] = String.valueOf(theFile.canRead());
data[0][4] = String.valueOf(theFile.canWrite());
data[0][5] = String.valueOf(theFile.lastModified());
data[0][6] = String.valueOf(theFile.isHidden());

return data;
}
void jButton1ActionPerformed(ActionEvent e) { //处理按钮
    file = new File(this.JTextField1.getText());
    this.JTable1 = new JTable(this.getFileInfo(file),this.getFileInfoType());
    jScrollPane1.setViewportView(JTable1);
}
}
```

本 章 小 结

文件操作在编程中必不可少，Java IO流标准类库能满足文件操作的许多基本要求：可以通过控制台、文件、内存块甚至因特网进行读写，可以创建新的输入和输出对象类型（通过从InputStream和OutputStream继承）等等。

第6章

Java与安全



本章内容

- 案例1 用户登录验证的完整程序
- 案例2 MD5的JavaBean实现
- 案例3 用公钥计算消息摘要的验证码
- 案例4-1 Java中数字证书的生成及维护方法
- 案例4-2 数字证书的签发（签名）
- 案例4-3 利用数字证书给Applet签名
- 案例5 利用DES加密解密
- 本章小结



案例1 用户登录验证的完整程序

案例运行效果与操作

本案例是一个实现用户登录验证的例子。用户输入用户名、密码，如果用户名和密码均正确，那么系统提示登录成功，否则系统自动提示用户名错误或者是密码错误。当然，在实际应用中为了安全起见，无论是用户名还是密码错误，系统均不出现详细的提示。

在IE地址栏中输入http://localhost/myforum/uservalidate/login.jsp，出现如下画面，如图6-1所示。

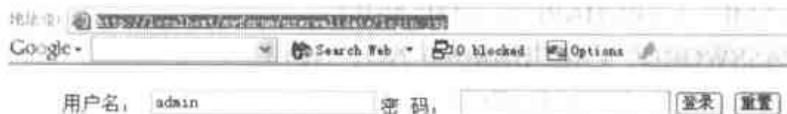


图6-1 运行界面

输入用户名和密码之后，单击“登录”按钮，系统跳转到对应的页面，判断用户名和密码是否正确给出提示信息。



图6-2 成功登录

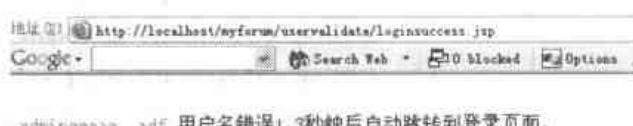


图6-3 用户名错误

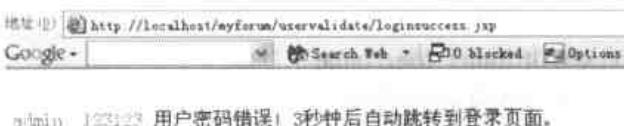


图6-4 用户密码错误

制作要点

1. Session()的用法
2. 数据库的连接

步骤详解

1. 创建数据库表格，如表6-1所示。

表6-1 数据库表的结构

用户表forumuser					
用户id	自动编号	UserID	否	(不能)	为初始值为1
用户名	字符串	UserName	否	(不能为空)	
用户密码	字符串	UserPassword	否	(不能为空)	
注册日期	字符串	RegisterDate			
注册IP	字符串	RegisterIP			

```
CREATE TABLE "DB2ADMIN"."FORUMUSER" (
  "USERID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
  ( START WITH +1 , INCREMENT BY +1 , CACHE 20 )
```

```
"USERNAME" VARCHAR(20) NOT NULL ,
"USERPASSWORD" VARCHAR(64) NOT NULL
"REGISTERDATE" VARCHAR(64)
"REGISTERIP" CHAR(15) )
```

2. 连接数据库。

具体方法参见第7章Java与数据库。

3. 验证登录信息。

```
if (tempUserPassword.equals(userPassword)) {
    session.setAttribute("userName", "admin");
    session.setMaxInactiveInterval(120);
    //设置session的有效时间120秒。
    temp = 1;//说明用户名和密码均正确
```

如果不相等：

```
session.setAttribute("userName", "");
temp = -2; //说明密码错误
```

程序源代码与解析

登录JSP页面login.jsp:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en" "http://www.w3.org/TR/REC-
html40/strict.dtd">
<%@ page contentType="text/html;charset=gb2312" info="我的第一个论坛！"
errorPage="error.jsp"%>
<%@ page import="simpleforum.database.QueryHelp"%>
<%@ page import="simpleforum.util.ParamUtil"%>
<%@ page import="simpleforum.user.UserValidate"%>
<html>
<head>
    <title>青年论坛</title>
    <%@ include file="/simpleforum/meta.jsp"%>
    <link rel='stylesheet' type='text/css' href='./css/forum.css'>
</head>
<body>
    <%@ include file="/simpleforum/header.jsp"%>
    <%
        String contextPath = request.getContextPath();
    %>
    <table width="100%">
```

```

<tr>
    <td><a href="<%={contextPath%}/simpleforum/index.jsp">论坛主页
</a></td>
</tr>
</table>
<%
String userName = ParamUtil.getParameter(request,"userName");
String userPassword = ParamUtil.getParameter(request,"userPassword");
int flag = UserValidate.userValidate(session,userName,userPassword);
if(flag > 0){
    String sql="select * from forumforum order by issueNum desc";
    java.util.Vector vector = QueryHelp.getHelp(sql);
    if(vector.size()<=0){
%>
<table width="100%">
    <tr>
        <td>本论坛暂时没有讨论区，谢谢关注！！！2秒钟后自动转入讨论区添
加页面！</td>
    </tr>
</table>
<meta http-equiv='refresh' content='2; url=addforum.jsp'>
<%
}else{
    out.println("<br><font color='red'>登录成功，2秒钟自动转入论坛主页！");
}
%>
<meta http-equiv='refresh' content='2; url=login.jsp'>
<%
}else{
    out.println("<br><br><font color='red'>登录失败，2秒钟后将自动转到登录页
面！</font>"+<br><br>flag="+flag);
}
%>
<meta http-equiv='refresh' content='2; url=login.jsp'>
<%
}
%>
<%@ include file="/simpleforum/footer.jsp"%>
</body>
</html>

```

登录验证页面loginsuccess.jsp:

```

<!doctype html public "-//w3c//dtd html 4.0 transitional//en" "http://www.w3.org/TR/REC-
html40/strict.dtd">
<%@ page contentType="text/html;charset=gb2312" info="我的第一个论坛！" %>
<%@ page import="simpleforum.user.UserValidate"%>
<html>
<head>
    <title> 判断登录是否成功 </title>
</head>
<body>
<%
    String userName = request.getParameter("userName")==null ? "" : request.getParameter
("userName");
    String userPassword = request.getParameter("userPassword")==null ? "" :
request.getParameter("userPassword");
    //因为前面已经校验过了，所以取到的值不会是空字符串“”；
    out.println("<br><font color=\"red\\>" + userName + "&nbsp;&nbsp;" + userPassword +
"</font>");
    int flag = UserValidate.userValidate(session,userName,userPassword);
    if(flag > 0){
        out.println("登录成功！");
    }else if (flag == -2){
        out.println("用户名密码错误！3秒钟后自动跳转到登录页面。");
    }else{
        out.println("用户名错误！3秒钟后自动跳转到登录页面。");
    }
<%
    <meta http-equiv='refresh' content='3; url=login.jsp'>
<%
    }
<%
    }
</body>
</html>

```

验证用户身份的Java源代码:

```

//chp6
public class UserValidate {
    /**

```

```

* 用来校验用户名和用户密码是否正确。
*
* @param request
* @param session
* @param userName
* @param userPassword
* @return
*/
public static int userValidate(HttpServletRequest session, String userName,
                               String userPassword) {
    int temp = 0;
    String tempUserName = ""; //用户名默认为空
    String tempUserPassword = ""; //密码默认为空
    String sql = "select username,userpassword from forumuser where username='"
                + userName + "'";
    //System.out.println("sql=" + sql);
    Vector vector = QueryHelp.getHelp(sql);
    //连接数据库的源代码可以在配书光碟中找到
    if (vector.size() > 0) {
        for (int i = 0; i < vector.size(); i++) {
            Hashtable hash = (Hashtable) vector.elementAt(i);
            tempUserPassword = (String) hash.get("USERPASSWORD");
        }
        if (tempUserPassword.equals(userPassword)) {
            session.setAttribute("userName", "admin");
            session.setMaxInactiveInterval(120);
            //设置session的有效时间120秒。
            temp = 1;//说明用户名和密码均正确。
        } else {
            session.setAttribute("userName", "");
            temp = -2; //说明密码错误
        }
    } else {
        temp = -1;
        //用户名错误
        return temp;
    }
    return temp;
}

```



案例2 MD5的JavaBean实现

案例运行效果与操作

本案例是一个用Java实现MD5加密的例子。程序运行界面如图6-5所示。

```

public static void main(String[] args){
    MD5 md = new MD5();
    System.out.println("我爱你Java:" + md.calcMD5("我爱你"));
    //我爱你Java: 16236e0315e4ae31c3b28c9a1ccb46 MD5.java
}

```

控制台
已经启动 MD5 Java 应用程序] C:\java\jre1.5.0_03\bin\javaw.exe { 2005-5-9 16:16:35 }
我爱你Java:16236e0315e4ae31c3b28c9a1ccb46

图6-5 运行界面

制作要点

MD5的算法

步骤详解

1. MD5简介。

MD5的全称是Message-Digest Algorithm 5，在20世纪90年代初由MIT的计算机科学实验室和RSA Data Security Inc发明，经MD2、MD3和MD4发展而来。

Message-Digest泛指字节串（Message）的Hash变换，就是把一个任意长度的字节串变成一定长的大整数。请注意，这里使用了“字节串”而不是“字符串”这个词，是因为这种变换只与字节的值有关，与字符集或编码方式无关。

MD5将任意长度的“字节串”变成一个128bit的大整数，而且，它是一个不可逆的字符串变换算法。换句话说就是，即使你看到源程序和算法描述，也无法将一个MD5的值变换回原始的字符串，从数学原理上说，是因为原始的字符串有无穷多个，这有点像不存在反函数的数学函数。

MD5的典型应用是对一段Message（字节串）产生fingerprint（指纹），以防止被“篡改”。举个例子，你将一段话写在一个叫readme.txt的文件中，并对这个readme.txt产生一个MD5的值并记录在案，然后你可以传播这个文件给别人，别人如果修改了文件中的任何内容，你对这个文件重新计算MD5时就会发现。如果再有一个第三方的认证机构，用MD5还可以防止文件作者的“抵赖”，这就是所谓的数字签名应用。MD5还广泛用于加密和解密技术上，在很多操作系统中，用户的密码是以MD5值（或类似的其他算法）的方式保存的，用户Login的时候，系统把用户输入的密码计算成MD5值，然后再去和系统中保存的MD5值进行比较，而系统并不“知道”用户的密码是什么。

一些黑客破获这种密码的方法是一种被称为“跑字典”的方法。有两种方法得到字典，一种是日常搜集的用做密码的字符串表，另一种是用排列组合方法生成的，先用MD5程序计算出这些字典项的MD5值，然后再用目标的MD5值在这个字典中检索。假设密码的最大长度为8，同时密码只能是字母和数字，共 $26 + 26 + 10 = 62$ 个字符，排列组合出的字典的项数则是 $P^1_{62} + P^2_{62} \dots + P^8_{62}$ ，那也已经是一个天文数字了，存储这个字典就需要TB级的磁盘组。而且，这种方法还有一个前提，就是在能获得目标账户的密码MD5值的情况下才可以。

在软件的加密保护中很多软件采用MD5保护，但是由于MD5算法为不可逆算法，所以所有的软件都是使用MD5算法作为一个加密的中间步骤，比如对用户名做一个MD5变换，结果再进行一个可逆的加密变换。做注册机时，只要先用MD5变换，然后再用一个逆算法即可。所以，对于破解者来说只要能看出是MD5就很容易了。

MD5代码的特点明显，跟踪时很容易发现，如果软件采用MD5算法，在数据初始化的时候必然用到以下的4个常数：

0x67452301;

0xefcdab89;

0x98badcfe;

0x10325476;

若常数不等，则可能是变形的MD5算法，或者根本就不是这个算法。

2. MD5算法。

第一步：增加填充

增加padding，使得数据长度（bit为单位）模512为448。如果数据长度正好是模512为448，增加512个填充bit，也就是说填充的个数为1~512。第一个bit为1，其余全部为0。

第二步：补足长度

将数据长度转换为64bit的数值，如果长度超过64bit所能表示的数据长度的范围，只保留最后64bit，增加到前面填充的数据后面，使得最后的数据为512bit的整数倍。也就是32bit的16倍的整数倍。在RFC1321中，32bit被称为一个word。

第三步：初始化变量

用到4个变量，分别为A、B、C、D，均为32bit长。初始化为：

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

第四步：数据处理

首先定义4个辅助函数：

F(X,Y,Z) = XY \vee not(X) Z

G(X,Y,Z) = XZ \vee Y not(Z)

H(X,Y,Z) = X xor Y xor Z

$$I(X,Y,Z) = Y \text{ xor } (X \vee \text{not}(Z))$$

其中：XY表示按位与，X v Y表示按位或，not(X)表示按位取反，xor表示按位异或。函数中的X、Y、Z均为32bit。

定义一个需要用到的数组T(i),i取值为1~64,T(i)等于abs(sin(i))的4 294 967 296倍的整数部分,i为弧度。假设前三步处理后的数据长度为32*16*Nbit

第五步：输出

最后得到的ABCD为输出结果，共128bit。A为低位，D为高位。

程序源代码与解析

```
//chp6

/*
 * MD5.java
 */
public class MD5 {
    /*
     * Convert a 32-bit number to a hex string with ls-byte first
     */
    String hex_chr = "0123456789abcdef";

    private String rhex(int num) {
        String str = "";
        for (int j = 0; j <= 3; j++)
            str = str + hex_chr.charAt((num >> (j * 8 + 4)) & 0x0F)
                    + hex_chr.charAt((num >> (j * 8)) & 0x0F);
        return str;
    }

    /*
     * Convert a string to a sequence of 16-word blocks, stored as an array.
     * Append padding bits and the length, as described in the MD5 standard.
     */
    private int[] str2blks_MD5(String str) {
        int nblk = ((str.length() + 8) >> 6) + 1;
        int[] blks = new int[nblk * 16];
        int i = 0;
        for (i = 0; i < nblk * 16; i++) {
            blks[i] = 0;
        }
        for (i = 0; i < str.length(); i++) {
            blks[i >> 2] |= str.charAt(i) << ((i % 4) * 8);
        }
        blks[i >> 2] |= 0x80 << ((i % 4) * 8);
    }
}
```

```

        blks[nblk * 16 - 2] = str.length() * 8;

        return blks;
    }

/*
 * Add integers, wrapping at 2^32
 */
private int add(int x, int y) {
    return ((x & 0x7FFFFFFF) + (y & 0x7FFFFFFF)) ^ (x & 0x80000000)
        ^ (y & 0x80000000);
}

/*
 * Bitwise rotate a 32-bit number to the left
 */
private int rol(int num, int cnt) {
    return (num << cnt) | (num >>> (32 - cnt));
}

/*
 * These functions implement the basic operation for each round of the
 * algorithm.
 */
private int cmn(int q, int a, int b, int x, int s, int t) {
    return add(rol(add(add(a, q), add(x, t)), s), b);
}

private int ff(int a, int b, int c, int d, int x, int s, int t) {
    return cmn((b & c) | ((~b) & d), a, b, x, s, t);
}

private int gg(int a, int b, int c, int d, int x, int s, int t) {
    return cmn((b & d) | (c & (~d)), a, b, x, s, t);
}

private int hh(int a, int b, int c, int d, int x, int s, int t) {
    return cmn(b ^ c ^ d, a, b, x, s, t);
}

private int ii(int a, int b, int c, int d, int x, int s, int t) {
    return cmn(c ^ (b | (~d))), a, b, x, s, t);
}

/*
 * Take a string and return the hex representation of its MD5.
 */
public String calcMD5(String str) {

```

```

int[] x = str2blks_MD5(str);
int a = 0x67452301;
int b = 0xEFCDAB89;
int c = 0x98BADCFE;
int d = 0x10325476;

for (int i = 0; i < x.length; i += 16) {
    int olda = a;
    int oldb = b;
    int oldc = c;
    int olld = d;

    a = ff(a, b, c, d, x[i + 0], 7, 0xD76AA478);
    d = ff(d, a, b, c, x[i + 1], 12, 0xE8C7B756);
    c = ff(c, d, a, b, x[i + 2], 17, 0x242070DB);
    b = ff(b, c, d, a, x[i + 3], 22, 0xC1BDCEEE);
    a = ff(a, b, c, d, x[i + 4], 7, 0xF57C0FAF);
    d = ff(d, a, b, c, x[i + 5], 12, 0x4787C62A);
    c = ff(c, d, a, b, x[i + 6], 17, 0xA8304613);
    b = ff(b, c, d, a, x[i + 7], 22, 0xFD469501);
    a = ff(a, b, c, d, x[i + 8], 7, 0x698098D8);
    d = ff(d, a, b, c, x[i + 9], 12, 0x8B44F7AF);
    c = ff(c, d, a, b, x[i + 10], 17, 0xFFFF5BB1);
    b = ff(b, c, d, a, x[i + 11], 22, 0x895CD7BE);
    a = ff(a, b, c, d, x[i + 12], 7, 0x6B901122);
    d = ff(d, a, b, c, x[i + 13], 12, 0xFD987193);
    c = ff(c, d, a, b, x[i + 14], 17, 0xA679438E);
    b = ff(b, c, d, a, x[i + 15], 22, 0x49B40821);

    a = gg(a, b, c, d, x[i + 1], 5, 0xF61E2562);
    d = gg(d, a, b, c, x[i + 6], 9, 0xC040B340);
    c = gg(c, d, a, b, x[i + 11], 14, 0x265E5A51);
    b = gg(b, c, d, a, x[i + 0], 20, 0xE9B6C7AA);
    a = gg(a, b, c, d, x[i + 5], 5, 0xD62F105D);
    d = gg(d, a, b, c, x[i + 10], 9, 0x02441453);
    c = gg(c, d, a, b, x[i + 15], 14, 0xD8A1E681);
    b = gg(b, c, d, a, x[i + 4], 20, 0xE7D3FBC8);
    a = gg(a, b, c, d, x[i + 9], 5, 0x21E1CDE6);
    d = gg(d, a, b, c, x[i + 14], 9, 0xC33707D6);
    c = gg(c, d, a, b, x[i + 3], 14, 0xF4D50D87);
    b = gg(b, c, d, a, x[i + 8], 20, 0x455A14ED);
    a = gg(a, b, c, d, x[i + 13], 5, 0xA9E3E905);
    d = gg(d, a, b, c, x[i + 2], 9, 0xFCEFA3F8);
    c = gg(c, d, a, b, x[i + 7], 14, 0x676F02D9);
}

```

```

    b = gg(b, c, d, a, x[i + 12], 20, 0x8D2A4C8A);
    a = hh(a, b, c, d, x[i + 5], 4, 0xFFFFA3942);
    d = hh(d, a, b, c, x[i + 8], 11, 0x8771F681);
    c = hh(c, d, a, b, x[i + 11], 16, 0x6D9D6122);
    b = hh(b, c, d, a, x[i + 14], 23, 0xFDE5380C);
    a = hh(a, b, c, d, x[i + 1], 4, 0xA4BEEA44);
    d = hh(d, a, b, c, x[i + 4], 11, 0x4BDECFA9);
    c = hh(c, d, a, b, x[i + 7], 16, 0xF6BB4B60);
    b = hh(b, c, d, a, x[i + 10], 23, 0xBEBFBC70);
    a = hh(a, b, c, d, x[i + 13], 4, 0x289B7EC6);
    d = hh(d, a, b, c, x[i + 0], 11, 0xEAA127FA);
    c = hh(c, d, a, b, x[i + 3], 16, 0xD4EF3085);
    b = hh(b, c, d, a, x[i + 6], 23, 0x04881D05);
    a = hh(a, b, c, d, x[i + 9], 4, 0xD9D4D039);
    d = hh(d, a, b, c, x[i + 12], 11, 0xE6DB99E5);
    c = hh(c, d, a, b, x[i + 15], 16, 0x1FA27CF8);
    b = hh(b, c, d, a, x[i + 2], 23, 0xC4AC5665);

    a = ii(a, b, c, d, x[i + 0], 6, 0xF4292244);
    d = ii(d, a, b, c, x[i + 7], 10, 0x432AFF97);
    c = ii(c, d, a, b, x[i + 14], 15, 0xAB9423A7);
    b = ii(b, c, d, a, x[i + 5], 21, 0xFC93A039);
    a = ii(a, b, c, d, x[i + 12], 6, 0x655B59C3);
    d = ii(d, a, b, c, x[i + 3], 10, 0x8F0CCC92);
    c = ii(c, d, a, b, x[i + 10], 15, 0xFFEFF47D);
    b = ii(b, c, d, a, x[i + 1], 21, 0x85845DD1);
    a = ii(a, b, c, d, x[i + 8], 6, 0x6FA87E4F);
    d = ii(d, a, b, c, x[i + 15], 10, 0xFE2CE6E0);
    c = ii(c, d, a, b, x[i + 6], 15, 0xA3014314);
    b = ii(b, c, d, a, x[i + 13], 21, 0x4E0811A1);
    a = ii(a, b, c, d, x[i + 4], 6, 0xF7537E82);
    d = ii(d, a, b, c, x[i + 11], 10, 0xBD3AF235);
    c = ii(c, d, a, b, x[i + 2], 15, 0x2AD7D2BB);
    b = ii(b, c, d, a, x[i + 9], 21, 0xEB86D391);

    a = add(a, olda);
    b = add(b, oldb);
    c = add(c, oldc);
    d = add(d, olde);

}

return rhex(a) + rhex(b) + rhex(c) + rhex(d);
}

public static void main(String[] args){

```

```

        MD5 md = new MD5();
        System.out.println("我爱你 Java: "+md.calcMD5("我爱你"));
        //我爱你: 282368f0315e4dc3fc9be28c9a1ccb46    MD5.java
    }
}

```



案例3 用公钥计算消息摘要的验证码

案例运行效果与操作

在计算机安全通信过程中，常使用消息摘要和消息验证码来保证传输的数据未曾被第三方修改。本案例将利用消息摘要和消息验证码（公钥）来验证传送的消息摘要是否正确。运行结果如图图6-6、图6-7、图6-8所示。

```

//常用的有MD5,SHA算法等
md.update(str.getBytes("UTF-8")); //传入原始字符串
byte[] re = md.digest(); //计算消息摘要放入byte数组
//下面把消息摘要转换为字符串

```

控制台
已终止 DigestPass [Java 应用程序] C:\java\jre1.5.0_03\bin\javaw.exe [2005-07-a9745449ca52906557fb6b8172e321f]

图6-6 计算字符串的消息摘要

```

//也可使用别名：升全功能计算
//初始化时是从开始即开始计算，但我们可以开始时关闭，然后
//din.on(false);
int b;

```

控制台
已终止 DigestInput [Java 应用程序] C:\java\jre1.5.0_03\bin\javaw.exe [2005-04e37ed0bea7db18061377d70a527b75]

图6-7 计算文件的消息摘要

```

m.update(str.getBytes("UTF-8"));
byte[] re = m.doFinal(); //生成消息码
//下面把消息码转换为字符串

```

控制台
已终止 MyMac [Java 应用程序] C:\java\jre1.5.0_03\bin\javaw.exe [2005-5-10 615cc3b8a1bf64180869ddaa82238119]

图6-8 通过共同密钥计算消息摘要

制作要点

1. 输入/输出流的使用
2. java.security的使用

步骤详解

1. 计算消息摘要。

消息摘要是对原始数据按照一定算法进行计算后得到的结果，它主要检测原始数据是否被修改过。消息摘要与加密不同，加密是对原始数据进行变换，可以从变换后的数据中获得原始数据，而消息摘要是从原始数据中获得一部分信息，它比原始数据少得多。因此，消息摘要可以被看做是原始数据的指纹。

下面用一段程序计算字符串的消息摘要：

```

String str = "Hello,I sent to you 800 yuan.";
MessageDigest md = MessageDigest.getInstance("MD5");//常用的有MD5、SHA算法等
md.update(str.getBytes("UTF-8"));//传入原始字串
byte[] re = md.digest();//计算消息摘要放入byte数组中

```

下面把消息摘要转换为字符串：

```

String result = "";
for (int i = 0; i < re.length; i++) {
    result += Integer.toHexString((0x000000ff & re[i]) | 0xfffff00)
        .substring(6);
}
System.out.println(result);           //07a9745449ca5296557f56b8172e391f

```

当对一个文件进行加密时，上述方法便不再适用。下面一段程序来计算输入/输出流中的消息摘要。使用输入/输出流可以自己控制何时开始和关闭计算摘要，也可以不控制，将全过程进行计算。

```

String fileName = "test.txt";
MessageDigest md = MessageDigest.getInstance("MD5");
FileInputStream fin = new FileInputStream(fileName);
DigestInputStream din = new DigestInputStream(fin, md);//构造输入流
//DigestOutputStream dout = new DigestOutputStream(fout,md);

```

可以从开始时就进行计算，如可以在开始时关闭，然后从某一部分开始，例如：

```

while (b = din.read() != -1) {
    //做一些对文件的处理
    //if(b=='$') din.on(true); //当遇到文件中的符号$时才开始计算
}

```

test.txt文本的内容为“下面一段程序计算从输入/输出流中计算消息摘要。”。

2. 消息验证码。

当A和B通信时，A将数据传给B时，同时也将数据的消息摘要传给B，B收到后可以用该消息摘要验证A传来的消息是否正确。这时会产生问题，即，如果传递过程中别人修改了数据时，同时也修改了消息摘要。B就无法确认数据是否正确。消息验证码可以解决这一问题。

使用消息验证码的前提是A和B双方有一个共同的密钥，这样A可以将数据计算出来的消息摘要加密后发给B，以防止消息摘要被改。由于使用了共同的密钥，所以称为“验证码”。

下面的程序即可利用共同的密钥来计算消息摘要的验证码：

```

public class MyMac {
    public static void main(String[] args) throws Exception {
        //这是一个消息摘要串
    }
}

```

```

String str = "TestString";
//共同的密钥编码，这个可以通过其他算法计算出来
byte[] kb = { 11, 105, -119, 50, 4, -105, 16, 38, -14, -111, 21, -95,
              70, -15, 76, -74, 67, -88, 59, -71, 55, -125, 104, 42 };
//获取共同的密钥
SecretKeySpec k = new SecretKeySpec(kb, "HMACSHA1");
//获取Mac对象
Mac m = Mac.getInstance("HmacMD5");
m.init(k);
m.update(str.getBytes("UTF-8"));
byte[] re = m.doFinal(); //生成消息码

```

使用以上两种技术可以保证数据没有经过改变，但接收者还无法确定数据是否确实是某个人发来的。尽管消息码可以确定数据是某个有同样密钥的人发来的，但这要求双方具有共享的密钥，若有一组用户共享，我们就无法确定数据的来源了。

数字签名可以解决这一问题。数字签名利用非对称加密技术，发送者使用私钥加密数据产生的消息摘要（签名），接收者使用发送者的公钥解密消息摘要以验证签名是否是某个人的。由于私钥只有加密者才有，因此如果接收者用某个公钥解密了某个消息摘要，就可以确定这段消息摘要必然是对应的私钥持有者发来的。

使用数字签名的前提是接收数据者能够确信验证签名时（用发送者的私钥加密消息摘要）所用的公钥确实是某个人的（因为有可能有人假告公钥）。数字证书可以解决这个问题。

数字证书含有两部分数据：一部分是对应主体（单位或个人）的信息，另一部分是这个主体所对应的公钥。即数字证书保存了主体和它的公钥的一一对应关系。同样，数字证书也有可能被伪造，如何判定数字证书的内容的真实性呢？有效的数字证书必须经过权威CA的签名，即权威CA验证数字证书的内容的真实性，然后再在数字证书上使用自己的私钥签名（相当于在证书盖章确认）。

这样，当用户收到这样的数字证书后，会用相应的权威CA的公钥验证该证书的签名（因为权威的CA的公钥在操作系统中已经安装）。根据非对称加密的原理，如果该证书不是权威CA签名的，将不能通过验证，即该证书是不可靠的。

若通过验证，即可证明此证书含的信息（发信人的公钥和信息）是无误的。于是可以信任该证书，便可以通过该证书内含的公钥来确认数据确实是发送者发来的。

于是，双方通信时，A把数据的消息摘要用自己的私钥加密（即签名），然后把自己的数字证书和数据及签名后的消息摘要一起发送给B，B处查看A的数字证书，如果A的数字证书是经过权威CA验证可靠的，便信任A，可使用A的数字证书中附带的A的公钥解密消息摘要（这一过程同时确认了发送数据的人又可以解密消息摘要），然后通过解密后的消息摘要验证数据是否正确无误、没被修改过。

利用这一原理，可以突破Java Applet小程序在浏览器中的权限，由于默认的Applet权限控制不允许它访问操作系统级。于是我们可以用我们的数字证书来给applet签名，然后客户端收到该Applet时，系统会自动查看给该Applet签名的数字证书并提供给终

端用户，判定是否信任该数字证书，如果用户信任，则该Applet便有了访问系统的权限。

程序源代码与解析

```
//chp6

/**
 * 下面一段程序计算一段字符串的消息摘要
 * DigestPass.java
 */
import java.security.MessageDigest;

public class DigestPass {
    public static void main(String[] args) throws Exception {
        String str = "Hello,I sent to you 800 yuan.";
        MessageDigest md = MessageDigest.getInstance("MD5");//常用的有MD5,SHA
        算法等
        md.update(str.getBytes("UTF-8"));//传入原始字符串
        byte[] re = md.digest();//计算消息摘要放入byte数组中
        //下面把消息摘要转换为字符串
        String result = "";
        for (int i = 0; i < re.length; i++) {
            result += Integer.toHexString((0x000000ff & re[i]) + 0xfffffff00)
                .substring(6);
        }
        System.out.println(result);
        //07a9745449ca5296557f56b8172e391f
    }
}

//chp6
/**
 * 下面一段程序计算从输入/输出流中计算消息摘要
 * DigestInput.java
 */
import java.io.FileInputStream;
import java.security.DigestInputStream;
import java.security.MessageDigest;

public class DigestInput {
    public static void main(String[] args) throws Exception {
        String fileName = "test.txt";
        MessageDigest md = MessageDigest.getInstance("MD5");
    }
}
```

```

        FileInputStream fin = new FileInputStream(fileName);
        DigestInputStream din = new DigestInputStream(fin, md); //构造输入流
        //DigestOutputStream dout = new DigestOutputStream(fout,md);
        //使用输入/输出流可以自己控制何时开始和关闭计算摘要
        //也可以不控制，将全过程计算
        //从开始时进行计算，也可以在开始时关闭，然后从某一部分开始，如：
        //din.on(false);
        int b;
        while ((b = din.read()) != -1) {
            //做一些对文件的处理
            //if(b=='$') din.on(true); //当遇到文件中的符号$时才开始计算
        }
        byte[] re = md.digest(); //获得消息摘要
        //下面把消息摘要转换为字符串
        String result = "";
        for (int i = 0; i < re.length; i++) {
            result += Integer.toHexString((0x000000ff & re[i]) | 0xfffff00)
                .substring(6);
        }
        System.out.println(result);
        //04e37ed6bea7db16051377d70a527b75
    }
}

/**
 *      下面的程序即可利用共同的密钥来计算消息摘要的验证码MyMac.java
 */
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class MyMac {
    public static void main(String[] args) throws Exception {
        //这是一个消息摘要串
        String str = "TestString";
        //共同的密钥编码，这个可以通过其它算法计算出来
        byte[] kb = { 11, 105, -119, 50, 4, -105, 16, 38, -14, -111, 21, -95,
                      70, -15, 76, -74, 67, -88, 59, -71, 55, -125, 104, 42 };
        //获取共同的密钥
        SecretKeySpec k = new SecretKeySpec(kb, "HMACSHA1");
        //获取Mac对象
        Mac m = Mac.getInstance("HmacMD5");
        m.init(k);
        m.update(str.getBytes("UTF-8"));
    }
}

```

```

byte[] re = m.doFinal(); //生成消息码
//下面把消息码转换为字符串
String result = "";
for (int i = 0; i < re.length; i++) {
    result += Integer.toHexString((0x000000ff & re[i]) + 0xfffffff00)
        .substring(6);
}
System.out.println(result);
//615ce3b8ebef6e189869dda622981191
}
}
}

```



案例4-1 Java中数字证书的生成及维护方法

步骤详解

Java中的keytool.exe可以用来创建数字证书，所有的数字证书均是以逐条（采用别名区别）的形式存入证书库中的，证书库中的一条证书包含该条证书的私钥、公钥和对应的数字证书的信息。证书库中的一条证书可以导出数字证书文件，数字证书文件只包括主体信息和对应的公钥。

每一个证书库都一个文件组成，它有访问密码，在首次创建时，会自动生成证书库，并要求指定访问证书库的密码。

在创建证书的时候，需要填写证书的一些信息及其对应的私钥密码。这些信息包括 CN = xx, OU = xx, O = xx, L = xx, ST = xx, C = xx，它们的意思是：

CN (Common Name) 名字与姓氏

OU (Organization Unit) 组织单位名称

O (Organization) 组织名称

L (Locality) 城市或区域名称

ST (State) 州或省份名称

C (Country) 国家名称

可以采用交互式工具提示输入以上信息，也可以采用参数 -dname “CN = xx, OU = xx, O = xx, L = xx, ST = xx, C = xx” 来自动创建。

1. 创建实例。

通过如下口令创建数字证书：

```
keytool -genkey -alias wnjCA -keyalg RSA -keysize 1024 -keystore wnjCALib -validity 3650
```

图6-9中最后一步，我们输入的是CN，代表中国的缩写，也可以直接输入“中国”两个字。



图6-9 创建证书

2. 证书操作方法。

- 证书显示

```
keytool -list -keystore wmjCALib
```

将显示wmjCALib证书库的所有证书列表，如图6-10所示。



图6-10 显示证书

又如：

```
keytool -list -alias wmjCA -keystore wmjCALib
```

将显示wmjCALib证书库中别名为wmjCA的证书的信息，如图6-11所示。

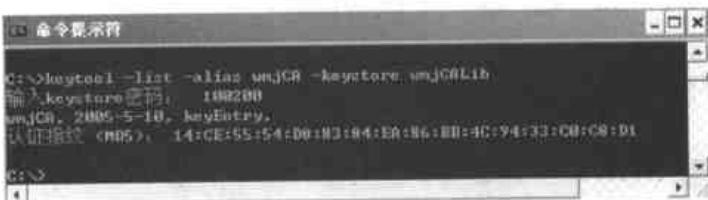


图6-11 显示wmjCALib证书库中别名为wmjCA的证书的信息

又如：

```
keytool -list -v -alias wmjCA -keystore wmjCALib
```

将显示证书的详细信息（-v参数），如图6-12所示。

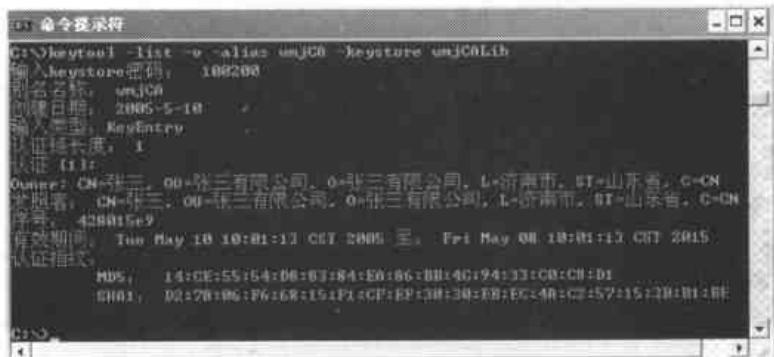


图6-12 显示证书的详细信息

- 将证书导出到证书文件

```
keytool -export -alias wmjCA -file wmjCA.cer -keystore wmjCALib
```

将把证书库wmjCALib中别名为wmjCA的证书导出到wmjCA.cer证书文件中。它包含证书主体的信息及证书的公钥，但不包括私钥，可以公开，如图6-13所示。

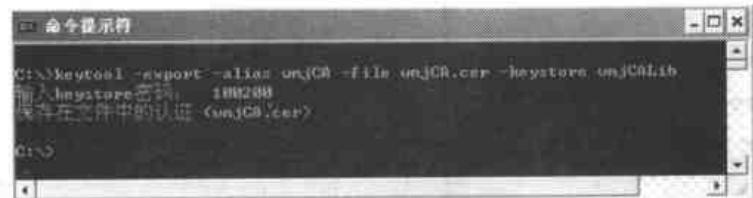


图6-13 导出证书文件（1）

上面导出的证书文件是以二进制编码的文件，无法用文本编辑器正确显示，因此不利于公布证书，可以加上-rfc参数，以一种可打印的编码输出。

```
keytool -export -alias wmjCA -file wmjCA.cer -keystore wmjCALib -storepass 100200 -rfc
```

这个命令在命令行中指定了证书库的访问密码，同时指定以可查看编码的方式输出，如图6-14所示。



图6-14 导出证书文件（2）

3. 通过证书文件查看证书的信息。

执行

```
keytool -printcert -file wmjCA.cer
```

查看到的证书信息如图6-15所示。

```
C:\>keytool -printcert -file wmjCA.cer
Owner: CN=张三, O=张三有限公司, L=济南市, ST=山东省, C=CN
颁发者: CN=张三, O=张三有限公司, L=济南市, ST=山东省, C=CN
序列号: 420015e9
有效期至: Tue May 10 10:01:13 CST 2005 至 Fri May 08 10:01:13 CST 2015
认证指纹:
MD5: 14:CE:55:54:D8:83:84:EA:B6:BB:4C:9A:33:CB:CB:8D
SHA1: D2:7B:06:F6:6B:15:F1:CF:EF:38:L3H:EB:4C:96:C2:57:15:2B:81:BE
C:\>
```

图6-15 查看证书信息

在Windows下查看证书，如图6-16、图6-17、图6-18所示。



图6-16 查看证书（1）



图6-17 查看证书（2）



图6-18 查看证书（3）

- **删除证书**

keytool的命令行参数`-delete`可以删除密钥库中的条目，如：

```
keytool -delete -alias wmjCA -keystore wmjCALib
```

这条命令将删除wmjCALib库中的wmjCA证书，如图6-19所示。



图6-19 删除证书

- **修改证书条目口令。**

执行

```
keytool -keypasswd -alias wmjCA -keystore wmjCALib
```

将修改证书口令，如图6-20所示。

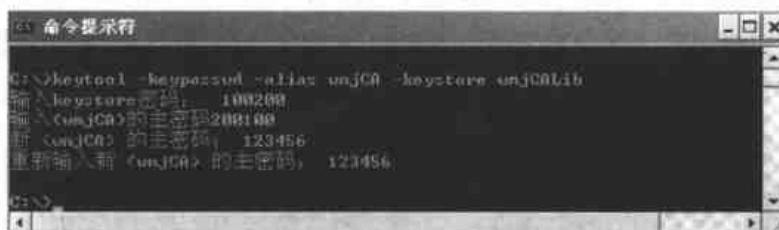


图6-20 交互式修改证书口令

可以用非交互的方式修改 wmjCALib证书库中条目为wmjCA的证书。如：

```
keytool -keypasswd -alias wmjCA -keypass 123456 -new 200100 -storepass
100200 -keystore wmjCALib
```

这一行命令以非交互式的方式修改库中别名为wmjCA的证书的密码为新密码123456，行中的200100是指该条证书的原密码，100200是指证书库的密码，如图6-21所示。

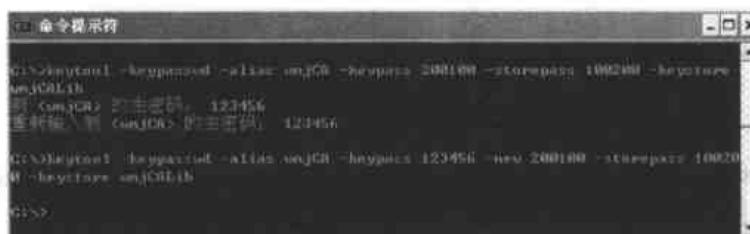


图6-21 直接修改证书口令



案例4-2 数字证书的签发（签名）

案例运行效果与操作

在案例4-1中创建好的数字证书，还没有经过权威CA的证实（即签名）。一般情况下，需要将这些证书发送给权威的CA，并申请其签名以确认数字证书让客户信任。

下面我们假设自己是一个权威的数字证书认证机构CA，这个机构将采用自己的私钥来签发其他的证书。签发过程是这样的：我们自己是CA，自己有一个自签的数字证书存入数字证书库中。在数字证书库中的这个CA数字证书含有私钥、公钥和这个CA的主体信息。下面这条指令可以创建一个CA的自签数字证书：

```
keytool -genkey -dname "CN=张三有限公司,OU=张三有限公司,O=张三有限公司,L=济南市,ST=山东省,C=中国" -alias MissionCA -keyalg RSA -keysize 1024 -keystore wmjCALib -keypass 200100 -storepass 100200 -validity 3650
```

在图6-22中，我们在 wmjCALib这个数字证书库中创建了一个别名为：missionCA、有效期为3650天、算法为RSA且密钥长度为1024的数字证书，这个证书的私钥密码为：200100，证书库的访问密码为：100200。这条别名为missionCA的证书代表我们自己的权威CA，即，美森系统软件有限公司这个权威CA。以后我们将用这个证书来签名其他的数字证书。



图6-22 创建数字证书

现在要给自己申请一个数字证书，可以这么做：先在数字证书库中创建一个证书：

```
keytool -genkey -dname "CN=王明杰,OU=张三有限公司,O=张三有限公司,L=济南市,ST=山东省,C=中国" -alias wMJCA -keyalg RSA -keysize 1024 -keystore wmjCALib -keypass 200100 -storepass 100200 -validity 3650
```

如图6-23所示，这样创建了一个别名为wmjCA的数字证书，我们可以将它导出为cer文件，如图6-24所示。

```
keytool -export -alias wMJCA -file wMJCA.cer -keystore wmjCALib -storepass 100200 -rfc
```

接着，我们可以用上一步生成的CA自签证书来签名这个数字证书。CA签名数字证书的过程需用以下程序来进行，这个程序是自解释的，见本案例的程序源代码和解释。

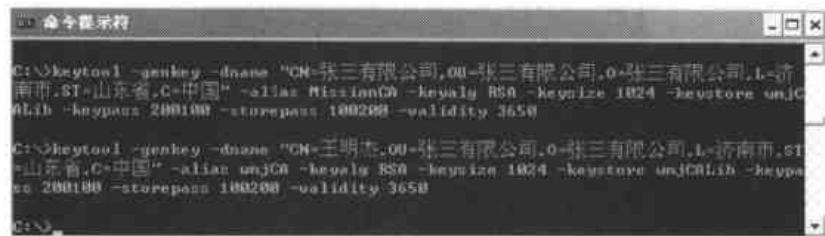


图6-23 运行界面



图6-24 导出界面

运行程序后，即可运用MissionCA证书来签发wmjCA证书。运行后，在wmjCALib中增加一条别名为wmjCA_Signed的数字证书，我们将它导出为cer文件，如图6-25和图6-26所示。



图6-25 运行界面

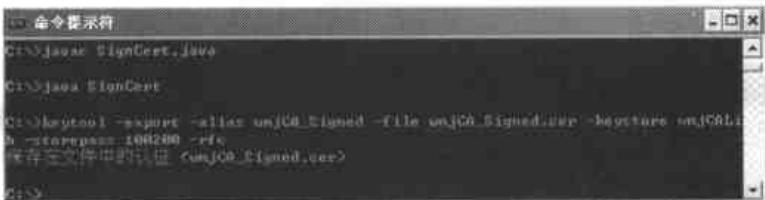


图6-26 导出界面

```
keytool -export -alias wmjCA_Signed -file wmjCA_Signed.cer -keystore wmjCALib -storepass 100200 -rfc
```

至此，已经用CA的证书为我们的数字证书签名了。在Windows中，双击导出的wmjCA_Signed.cer文件，出现如图6-27所示界面：

图6-27中“证书信息”一栏显示“不能验证该证书”，原因是我们的这个数字证书的签发者MissionCA证书没有安装到系统中。可以将证书库中别名为MissionCA的自签数字证书导出为cer文件，然后安装到系统中，如图6-28所示。



图6-27 打开cer文件



图6-28 导出自签数字证书

```
keytool -export -alias MissionCA -file MissionCA.cer -keystore wmjCALib -storepass  
100200 -rfc
```

在图6-29中单击“安装证书”按钮，将证书安装到系统中，安装过程如图6-30、图6-31、图6-32、图6-33、图6-34所示。

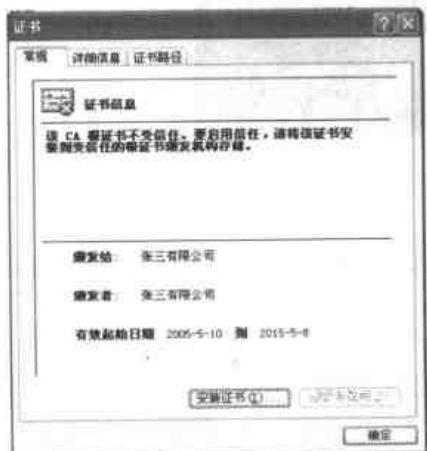


图6-29 打开证书文件

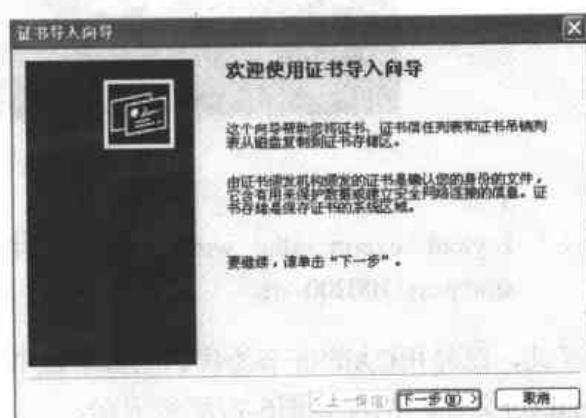


图6-30 安装界面1

再次双击查看此证书wmjCA_Signed.cer，结果如图6-35所示。



图6-31 安装界面2

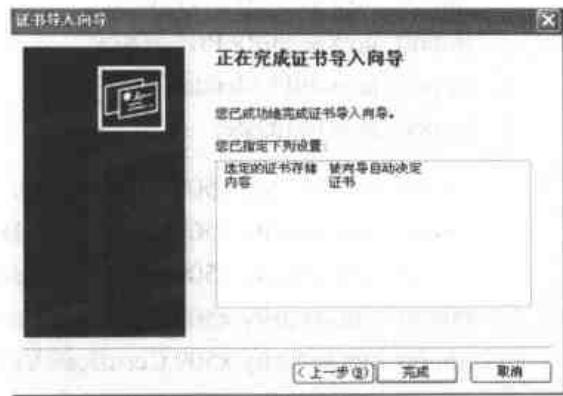


图6-32 安装界面3

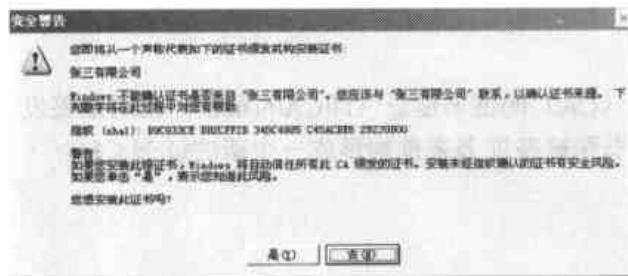


图6-33 安装界面4



图6-34 安装界面5

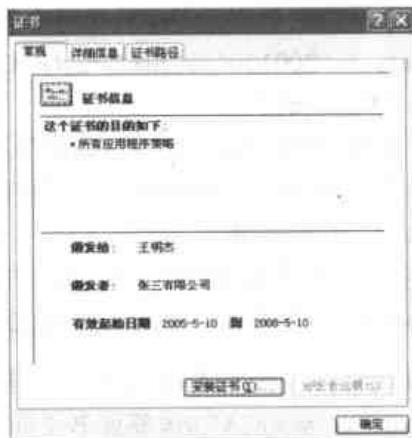


图6-35 查看证书

到此为止，我们已经获得了一个由我们自己的 CA 签名颁发的个人数字证书，并且将自己的 CA 证书安装到系统中成为系统信任的根证书。于是，以后只要是由我们的这个 CA 证书签名颁发的数字证书都会受到系统的信任。

程序源代码与解释

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```

import java.security.KeyStore;
import java.security.PrivateKey;
import java.util.Calendar;
import java.util.Date;

import sun.security.x509.AlgorithmId;
import sun.security.x509.CertificateAlgorithmId;
import sun.security.x509.CertificateIssuerName;
import sun.security.x509.CertificateSerialNumber;
import sun.security.x509.CertificateValidity;
import sun.security.x509.X500Name;
import sun.security.x509.X509CertImpl;
import sun.security.x509.X509CertInfo;

/**
 * <p>Description: 该程序根据签发者（CA）的证书信息（即CA的私钥）来对被签发者的证书进行签名，过程是使用CA的证书和被签证书来重构形成一个新的证书</p>
 */

public class SignCert {
    public static void main(String[] args) throws Exception {
        char[] storepass = "100200".toCharArray();
        //存放CA证书和被签证书的证书库的访问密码
        char[] cakeypass = "200100".toCharArray(); //CA数字证书条目的访问密码
        String alias = "MissionCA";
        //CA证书在证书库中的别名，这个CA的证书用来签名其他的证书
        String name = "wmjCALib"; //存放CA证书和被签证书的证书库的名字
        String newLib = "SignedLib";
        //新证书库的名字，如果需要将签名后的证书放入新库，这就是新库的名字
        char[] newLibPass = "100200".toCharArray(); //设置新库的访问密码

        String cerFileName = "wmjCA.cer"; //被签证书的证书文件名
        String aliasName = "wmjCA"; //被签证书在证书库中的alias别名
        char[] namePass = "200100".toCharArray();
        //被签证书的条目在证书库的私钥密码
        int n = 3; //被签证书的有效期，以年为单位，以当前时间开始计算
        int sn = 200505001;
        //序列号可自己定义，这里定义为2005年5月签发，是本年度CA签发的第多少个，以001的形式计算，要求惟一
        String afteraliasName = "wmjCA_Signed";
        //签名后新产生的被签过名的证书在库中的别名
        char[] afterNewPass = "200100".toCharArray();
        //签名后新产生的被签过名的证书在库的条目的私钥密码
        //装载证书库
        FileInputStream in = new FileInputStream(name);
    }
}

```

```

KeyStore ks = KeyStore.getInstance("JKS");//JKS为证书库的类型
ks.load(in, storepass);
//从证书库中读出签发者(CA)的证书
java.security.cert.Certificate cl = ks.getCertificate(alias);
//读出一个CA证书,这里的l是字母,而不是数字1
PrivateKey privateKey = (PrivateKey) ks.getKey(alias, cakeypass);
//根据别名和证书密码读出CA证书的私钥
in.close();
//从证书库中读出的签发者(CA)的证书中提取签发者的信息
byte[] encodl = cl.getEncoded();//提取证书的编码,这里l是字母不是数字1
X509CertImpl cimpl = new X509CertImpl(encodl);
//这里l是字母不是数字1,根据证书的编码创建X509CertImpl类型的对象
//根据上面的对象获得X509CertInfo类型的对象,该对象封装了证书的全部内容
X509CertInfo cinfo_first = (X509CertInfo) cimpl.get(X509CertImpl.NAME
        + " ." + X509CertImpl.INFO);
//然后获得X500Name类型的签发者信息
X500Name issuer = (X500Name) cinfo_first.get(X509CertInfo.SUBJECT + "."
        + CertificateIssuerName.DN_NAME);
//获取待签发的证书,即获取被签发者的证书
//可从密钥库中获取,也可从导出的证书文件中获取,这里给出两种方式
// //////////////////////////////////////////////////////////////////
//方式一:采用从导出的cer文件中获取start
// //////////////////////////////////////////////////////////////////
/*
 *
 * CertificateFactory cf = CertificateFactory.getInstance("X.509");
 * //X.509是使用最多的一种数字证书标准
 *
 * FileInputStream in2 = new FileInputStream(cerFileName);//被签证书文件
 *
 * java.security.cert.Certificate c2 = cf.generateCertificate(in2);
 * //生成需要被签的证书 in2.close(); byte[] encod2 = c2.getEncoded();
 * X509CertImpl cimpl2 = new X509CertImpl(encod2);
 *
 * //获得被签证书的详细内容,然后根据这个证书生成新证书
 *
 * X509CertInfo cinfo_second =
 * (X509CertInfo)cimpl2.get(X509CertImpl.NAME+"."+X509CertImpl.INFO);
 */
//end方式一
// //////////////////////////////////////////////////////////////////

```

```

///////////
//方式二：从证书库中读出被签的证书 start
/////////
java.security.cert.Certificate c3 = ks.getCertificate(aliasName);
//从证书库中读出被签证书，然后生成新的证书
byte[] encod3 = c3.getEncoded();
X509CertImpl cimp3 = new X509CertImpl(encod3);
X509CertInfo cinfo_second = (X509CertInfo) cimp3.get(X509CertImpl.NAME
+ "." + X509CertImpl.INFO);

//end方式二

//设置新证书的有效期，使之为当前向后n年有效，新证书的截止日期不能超过
CA证书的有效日期

Date beginDate = new Date();
Calendar cal = Calendar.getInstance();
cal.setTime(beginDate);
//cal.add(cal.YEAR, n);
cal.add(Calendar.YEAR, n);
Date endDate = cal.getTime();
CertificateValidity cv = new CertificateValidity(beginDate, endDate);
cinfo_second.set(X509CertInfo.VALIDITY, cv);

//设置新证书的序列号

CertificateSerialNumber csn = new CertificateSerialNumber(sn);
cinfo_second.set(X509CertInfo.SERIAL_NUMBER, csn);

//设置新证书的签发者
cinfo_second.set(X509CertInfo.ISSUER + "."
+ CertificateIssuerName.DN_NAME, issuer);

//新的签发者是从CA的证书中读出来的

//设置新证书的算法，指定CA签名该证书所使用的算法为md5WithRSA

AlgorithmId algorithm = new AlgorithmId(
    AlgorithmId.md5WithRSAEncryption_oid);
cinfo_second.set(CertificateAlgorithmId.NAME + "."
+ CertificateAlgorithmId.ALGORITHM, algorithm);

//创建新的签名后的证书

X509CertImpl newcert = new X509CertImpl(cinfo_second);

//签名，使用CA证书的私钥进行签名，签名使用的算法为MD5WithRSA
newcert.sign(privateKey, "MD5WithRSA");//这样便得到了经过CA签名后的证书

//把新证书存入证书库

```

```

//把新生成的证书存入一个新的证书库，也可以存入原证书库，  

//存入新证书库，则新证书库中不仅包含原证书库中的所有条目，而且新增加了一个这次产生的条目。注意，这时，新产生的签名后的证书只包括公钥和主体信息及签名信息，不包括私钥信息。这里给出两种方式。  

/////////////////////////////////////////////////////////////////////////  

//方式一：存入新密钥库  

/////////////////////////////////////////////////////////////////////////  

/*
 *
 * ks.setCertificateEntry(afteraliasName,newcert); FileOutputStream out =
 * new FileOutputStream(newLib);
//存入新库signedLib，并设置新库的库访问密码
 * ks.store(out,newLibPass); out.close();
 */
/////////////////////////////////////////////////////////////////////////  

//end 方式一
/////////////////////////////////////////////////////////////////////////  

//也可以采用另外一种方式，存入原证书库中
//存入原库中，即在原证书库中增加一条证书，它是原证书经过签名后的证书
//这个新证书含有私钥和私钥密码
/////////////////////////////////////////////////////////////////////////  

//方式二：存入原密钥库
/////////////////////////////////////////////////////////////////////////  

//先在原库中读出被签证书的私钥
PrivateKey prk = (PrivateKey) ks.getKey(aliasName, namePass);
java.security.cert.Certificate[] cchain = { newcert };
//存入原来的库，第一个参数为原证书的私钥，第二个参数为新证书的私钥密码，  

第三个参数为新证书
ks.setKeyEntry(afteraliasName, prk, afterNewPass, cchain); //用新密钥替代原来  

没有签名的证书的密码
FileOutputStream out2 = new FileOutputStream(name);
ks.store(out2, storepass);//存入原来的库中，第二个参数为该库的访问密码
//end 方式二
}
}

```



案例4-3 利用数字证书给Applet签名

步骤详解

现在假设要给xx公司做一个项目，这个项目中需要用到Applet，且这些applet需要特权以实现一些特殊的功能（如读出客户端用户系统中C:\WINDOWS\system.ini文件中的内容并显示）。那么，我们可以颁发一个数字证书，并给这个数字证书签名，然后用签名后的数字证书来签名我们的Applet，使客户信任。具体过程如下：

- 生成一个用于此项目签名Applet的数字证书，别名定为：Mission_Water，如图6-36所示。

```
keytool -genkey -dname "CN=张三软件-水公司项目,OU=张三有限公司,O=张三有限公司,L=济南市,ST=山东省,C=中国" -alias Mission_Water -keyalg RSA -keysize 1024 -keystore wmjCALib -keypass 200100 -storepass 100200 -validity 3650
```



图6-36 数字证书生成界面

- 用我们的CA（MissionCA）来签发这个数字证书，运行下面的程序，如图6-37所示。



图6-37 运行界面

程序代码见案例4-1程序源代码与解释。

程序会在wmjCALib证书库中产生一个别名为Mission_Water_Signed的数字证书，这个证书是经过我们的CA（MissionCA）签发的。

然后在DOS下运行指令。

```
keytool -export -alias Mission_Water_Signed -file Mission_Water_Signed.cer -keystore wmjCALib -storepass 100200 -rfc
```

如图6-38所示。

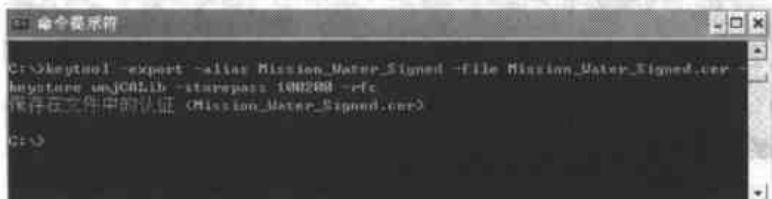


图6-38 导出界面

导出这个证书，会在C盘根目录下生成一个Mission_Water_Signed.cer文件。

3. 用签发后的数字证书来签名我们的Applet。

首先我们制作一个简单的Applet：

```
public class ShowFileApplet extends JApplet {
```

编译此Applet文件，如图6-39所示，会在当前目录下生成一个com\applet的目录结构，在applet目录下有一个ShowFileApplet.class，进入当前目录，执行，如图6-40所示。

```
Jar cvf myapplet.jar com/applet/*.*
```



图6-39 编译程序



图6-40 执行程序

于是在当前目录下产生了一个myapplet.jar文件。然后再在当前目录下新建一个applet目录，专门存放applet的jar文件，把前面生成的数字证书库wmjCALib文件也复制到applet目录下面来，同时把刚才生成的myapplet.jar文件也移到applet目录下面来。然后进入该目录执行：

```
jarsigner -keystore wmjCALib myapplet.jar Mission_Water_Signed
```

如图6-41所示。

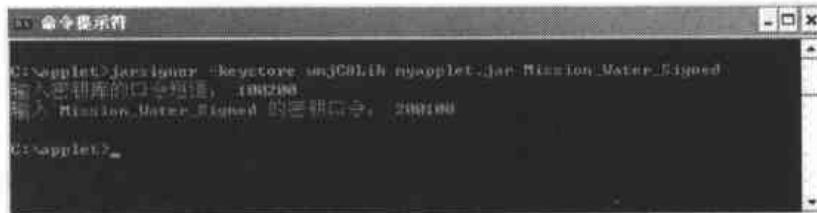


图6-41 执行界面

即用这个经我们的CA签发的数字证书Mission_Water_Signed给这个Applet签了名。

4. 运行我们的Applet。

首先来写一个html文件以运行这个签名后的Applet，内容如下：

```
<!-- ShowFileApplet.html -->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312">
<title>HTMLTestPage</title>
</head>
<body>
    applet将会显示，如果你的浏览器支持Java<br>
    <applet archive = "/applet/myapplet.jar"
        code = "com.applet.ShowFileApplet.class"
        name = "TestApplet"
        width = "400"
        height = "300"
        hspace = "0"
        vspace = "0"
        align = "middle">
    </applet>
</body>
</html>
```

双击此html文件，系统出现警告提示框，如图6-42所示。



图6-42 运行界面1

单击按钮“是(Y)”，就会看到下面的界面，如图6-43所示。



图6-43 运行界面2

这个HTML文件可以运行Applet，但如果浏览器不支持Java，即没有安装JRE，它不会提示用户去下载安装。我们可以用Java自带的htmlconverter工具转换一下这个HTML文件，如图6-44、图6-45、图6-46所示。转换后的文件可以在支持Java2的浏览器中运行Applet（不管该浏览器是否设置了使用Java2运行Applet，它都会在Java2环境中运行Applet，如果浏览器不支持Java2，则会自动下载所需的文件）。



图6-44 运行界面



图6-45 转换界面

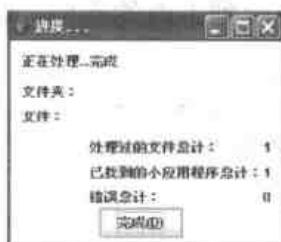


图6-46 运行界面

转换后的HTML代码如下：

```
<!-- ShowFileApplet.html -->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312">
<title>HTML TestPage</title>
</head>
<body>
    applet将会显示，如果你的浏览器支持Java<br>
    <!--"CONVERTED_APPLET"-->
    <!-- HTML CONVERTER -->
    <object
        classid = "clsid:CAFEEFAC-0015-0000-0003-ABCDEFFEDCBA"
        codebase = "http://java.sun.com/update/1.5.0/jinstall-1_5_0_03-windows-
        i586.cab#Version=5,0,30,7"
        WIDTH = "400" HEIGHT = "300" NAME = "TestApplet" ALIGN = "middle"
        VSPACE = "0" HSPACE = "0" >
        <PARAM NAME = CODE VALUE = "com.applet.ShowFileApplet.class" >
        <PARAM NAME = ARCHIVE VALUE = "./applet/myapplet.jar" >
        <PARAM NAME = NAME VALUE = "TestApplet" >
        <param name = "type" value = "application/x-java-applet;jpi-version=1.5.0_03">
        <param name = "scriptable" value = "false">

        <comment>
        <embed
            type = "application/x-java-applet;jpi-version=1.5.0_03" \
            CODE = "com.applet.ShowFileApplet.class" \
            ARCHIVE = "./applet/myapplet.jar" \
            NAME = "TestApplet" \
            WIDTH = "400" \
            HEIGHT = "300" \
            ALIGN = "middle" \
            VSPACE = "0" \
            HSPACE = "0"
```

```

scriptable = false
pluginspage = "http://java.sun.com/products/plugin/index.html#download">
<noembed>

</noembed>
</embed>
<comment>
</object>

<!--
<APPLET CODE = "com.applet.ShowFileApplet.class" ARCHIVE = "./applet/myapplet.jar"
WIDTH = "400" HEIGHT = "300" NAME = "TestApplet" ALIGN = "middle" VSPACE
= "0" HSPACE = "0">

</APPLET>
-->
<!--"END_CONVERTED_APPLET"-->

</body>
</html>

```

再次运行ShowFileApplet.html（小程序只使用该特定的JRE版本。若没有安装此版本，该版本将被自动下载。否则，将重新指定用户进入手动下载页进行下载），结果如图6-47所示。



图6-47 运行界面

程序源代码与解释

```

//chp6;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.TextArea;
import java.io.BufferedReader;

```

```

import java.io.FileReader;
import java.io.IOException;
import javax.swing.JApplet;
public class ShowFileApplet extends JApplet {
    private boolean isStandalone = false;
    private String content = "文件的内容是：" //自定义的提示信息
    private String fileName = "C:\WINDOWS\system.ini";//读出这个文件的内容
    private TextArea ta = new TextArea(10, 80);//自定义的输出框
    public String getParameter(String key, String def) {
        return isStandalone ? System.getProperty(key, def) :
            (getParameter(key) != null ? getParameter(key) : def);
    }
    public ShowFileApplet() {
    }
    public void init() {
        try {
            jbInit();
            myInit();//自己定义的方法
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setSize(new Dimension(400, 300));
    }
    /**
     * 自定义的初始化方法，读入系统中的一个文件的内容并保存起来，然后，增加一个
     * 可视化的输出框
     */
    private void myInit() {
        String s;
        BufferedReader in;
        try {
            in = new BufferedReader(new FileReader(fileName));
            while ((s = in.readLine()) != null) {
                content += s + "\n";
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        System.out.println(content);
    }
}

```

```

        ta.setText(content);
        getContentPane().add(ta);
    }
    //重载的方法，输出内容
    public void paint(Graphics g) {
        ta.setText(content);
    }
    public String getAppletInfo() {
        return "Applet Information";
    }
    public String[][] getParameterInfo() {
        return null;
    }
    //static initializer for setting look & feel
    static {
        try {
        }
        catch (Exception e) {
        }
    }
}

```



案例5 利用DES加密解密

案例运行效果与操作

本案例实现对指定文件进行DES加/解密的过程。程序运行后，界面如图6-48所示。

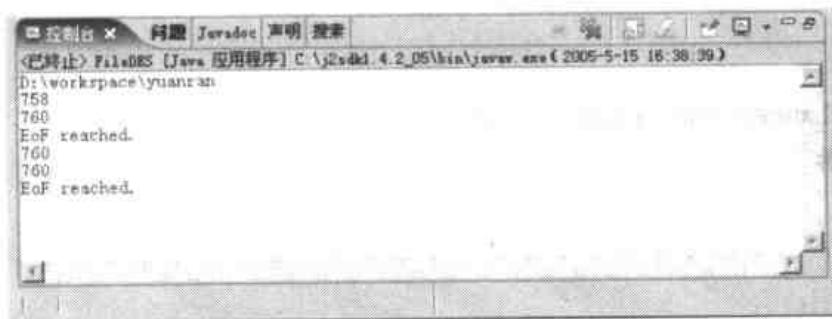


图6-48 控制台信息

加密的原文如下：

第一幕发现 协勤约住可疑男子

昨日下午4:30许，渝中区莲花街社区两位协勤正在辖区巡逻，当两人行至七星岗苏宁电器商场门前时，发现一名30岁左右的中年男子正鬼鬼祟祟地往身上藏约8厘米长的两把匕首。男子形迹可疑！两协勤当即不动声色地尾随其后，观察其进一步的动向。

孰料，在前行不足100米后，警觉性较高的该男子发现自己已被跟踪，便撒腿就跑。协勤及巡逻民警立即向这名持刀男子逃跑方向追去。

第二幕夺刀 闻进重庆抢出菜刀

一阵狂跑后，该男子冲进重庆宾馆厨房内，从案板上抓起一把大号菜刀，疯狂地挥舞着从宾馆内冲了出来，并提起菜刀朝两位追赶他的协勤砍去。经过一番躲闪，两协勤发现情况不妙，便迅速将此情况向渝中公安分局巡警支队三大队报告，并请求增援，正在解放碑步行街执勤的巡警们接报后，立即赶到事发现场，并一起追赶上向逃窜边不停挥舞菜刀的可疑男子。

密文输出如下：

```
f 踏?| (稍?"6 萎嫿嫿嫿?药 E 进
珪 b?"最<4 g 跳落 ?? 遮掩坪 iyy 遮蔽 x
I! E?L?y@折十Y 奉秆e^ 邪漫 7 告 2FjQ`?F 騞 x + 猥隋?4 ?妃 v 困 t S 嫣 A 困-7 遮蔽
+ ?辛噏澧!F 狹播 D 榆岩歡避雷境鹿?N 撇 6 露?AP 露 A ?乱?達權 tb4 3-s 嫣
(?)?| V?d- 5 張祁側 : r 迎 j 隘?傀?異離保 l (0 隘- 前 fL 札os 舒 L 摆 qL 遮蔽
XH<!1?"?回. 婴| は 斯述;呪唐越| r ?uF?| ?挫觀 g 梅寵綱 C4v 紀>妃 <熟e 遮蔽
1Z! PR?2@ 嗟獲邇< 遮 ryc 遮, 婴 pJ?@缺|
B1 X 句| ?始 R, T 摆 1S 婴- / 呕 €
- `0 鮑? 真 Y- 遮蟲季??/ 遮校 Sw 探戒季漢! ?u@节 pt b 開著獎惊債]K, ?鍼髮 R? 困
- 始, 1座;恭 sAh"詩??e 異領達] ?鮀 味孽_ - 鮀 k 順 a 嘴鹹?侏? (蘋? n\雷 S_??!, 遮曉
@1 開&KY 樣~領做 0#辛C 遮育|
幹廢瑞CX?Mv 脂 oh 開 2- <4 / | e, 遮 rs 潤? 遮?sk?遮 x 易+6 遮- ?(
肺?w 脣? 遮瓈心妙?飄 w91 寶唇??t? 遮開初- ;) 惡? B 遮@H Y_T + ++
齒近?蘋?e 遮?0 亂圓 X(
```

解密后的文件输出与原文件相同。

制作要点

1. `java.nio.channels.FileChannel`的用法
2. Des加密算法

步骤详解**DES算法介绍：**

DES算法的人口参数有3个：`Key`、`Data`、`Mode`。其中，`Key`为8字节共64位，是DES算法的工作密钥；`Data`也为8字节64位，是要被加密或被解密的数据；`Mode`为DES的工作方式，有两种：加密或解密。

DES算法的工作流程：

如Mode为加密，则用Key把数据Data进行加密，生成Data的密码形式（64位）作为DES的输出结果；如Mode为解密，则用Key把密码形式的数据Data解密，还原为Data的明码形式（64位）作为DES的输出结果。在通信网络的两端，双方约定一致的Key，在通信的源点用Key对核心数据进行DES加密，然后以密码形式在公共通信网（如电话网）中传输到通信网络的终点，数据到达目的地后，用同样的Key对密码数据进行解密，便再现了明码形式的核心数据。这样，便保证了核心数据（如PIN、MAC等）在公共通信网中传输的安全性和可靠性。

DES算法把64位的明文输入块变为64位的密文输出块，它所使用的密钥也是64位，整个算法的主流图如下：

把输入的64位数据块按位重新组合，并把输出分为L0、R0两部分，每部分各长32位，其置换规则如下：

58,50,12,34,26,18,10,2,60,52,44,36,28,20,12,4,
62,54,46,38,30,22,14,6,64,56,48,40,32,24,16,8,
57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,
61,53,45,37,29,21,13,5,63,55,47,39,31,23,15,7,

即将输入的第58位换到第一位，第50位换到第2位，依此类推，最后一位是原来的第7位。L0、R0则是换位输出后的两部分，L0是输出的左32位，R0是右32位，例如：设置换前的输入值为D1D2D3……D64，则经过初始置换后的结果为：L0=D58D50…D8；R0=D57D49…D7。

经过16次迭代运算后，得到L16、R16，将此作为输入，进行逆置换，即得到密文输出。逆置换正好是初始置的逆运算，例如，第1位经过初始置换后，处于第40位，而通过逆置换，又将第40位换回到第1位，其逆置换规则如下所示：

40,8,48,16,56,24,64,32,39,7,47,15,55,23,63,31,
38,6,46,14,54,22,62,30,37,5,45,13,53,21,61,29,
36,4,44,12,52,20,60,28,35,3,43,11,51,19,59,27,
34,2,42,10,50,18,58 26,33,1,41,9,49,17,57,25,

放大换位表

32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10,11,
12,13,12,13,14,15,16,17,16,17,18,19,20,21,20,21,
22,23,24,25,24,25,26,27,28,29,28,29,30,31,32, 1,

单纯换位表

16,7,20,21,29,12,28,17, 1,15,23,26, 5,18,31,10,
2,8,24,14,32,27, 3, 9,19,13,30, 6,22,11, 4,25,

在f(Ri,Ki) 算法描述图中，S1,S2...S8为选择函数，其功能是把6bit数据变为4bit数据。下面给出选择函数Si (i=1,2.....8) 的功能表：

S1:

14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,

0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13,

S2:

15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9,

S3:

10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12,

S4:

7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14,

S5:

2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3,

S6:

12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,
9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13,

S7:

4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12,

S8:

13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11,

在此以S1为例说明其功能，我们可以看到：在S1中，共有4行数据，命名为0、1、2、3行；每行有16列，命名为0、1、2、3、……、14、15列。

现设输入为：D = D1D2D3D4D5D6

令：列 = D2D3D4D5

行 = D1D6

然后在S1表中查得对应的数，以4位二进制表示，这就是选择函数S1的输出。下面给出子密钥Ki（48bit）的生成算法。

初始Key值为64位，但DES算法规定，其中第8、16...64位是奇偶校验位，不参与DES运算。故Key实际可用位数便只有56位。即：经过缩小选择换位表1的变换后，Key的位数由64位变成了56位，此56位分为C0、D0两部分，每部分各28位然后分别进行第1次循环左移，得到C1、D1，将C1（28位）、D1（28位）合并得到56位，再经过缩小选择换位2，从而便得到了密钥K0（48位）。以此类推，便可得到K1、K2...K15。不过需要注意的是，16次循环左移对应的左移位数要依据下述规则进行：

循环左移位数

1,1,2,2,2,2,2,1,2,2,2,2,2,2,1

以上介绍了DES算法的加密过程。DES算法的解密过程也是一样的，区别仅仅在于第一次迭代时用子密钥K15，第二次用K14……最后一次用K0，而算法本身并没有任何变化。

程序源代码与解析

```
//chp6
//FileDES.java

import java.io.*;
import java.nio.*;
import java.nio.channels.FileChannel;
public class FileDES{
    private static final boolean enc=true;      //加密
    private static final boolean dec=false;      //解密
    private String sourceFileName;
    private String targetFileName;
    private String inKey;
    private boolean encdes;
    private File sourceFile;
    private File targetFile;
    private Des des;
    private void getpath(){
        String pathname;
        int pos=sourceFileName.lastIndexOf("/");
        pathname=sourceFileName.substring(0,pos); //取得文件路径
        File dir=new File(pathname);
        if (!dir.exists()) {                  //判断是否存在
            System.out.println("不存在");
        }
    }
}
```

```

        System.err.println(pathname+" is not exist");
        System.exit(1);
    }else if(!dir.isDirectory()){ //判断是否为目录
        System.err.println(pathname+" is not a directory");
        System.exit(1);
    }

    pos=targetFileName.lastIndexOf("/");
    pathname=targetFileName.substring(0,pos);
    dir=new File(pathname);
    if (!dir.exists()){
        if(!dir.mkdirs()){
            System.out.println ("can not creat directory:"+pathname);
            System.exit(1);
        }
    }else if(!dir.isDirectory()){
        System.err.println(pathname+" is not a directory");
        System.exit(1);
    }
}

private static int bufcontent(FileChannel channel,ByteBuffer buf) throws I
OException{
    long byteLeft=channel.size()-channel.position();
    if(byteLeft==0L)
        return -1;
    buf.position(0);
    buf.limit(buf.position()+(byteLeft<8 ? (int)byteLeft :8));
    return channel.read(buf);
}

private void fileAction(boolean flag){
    des=new Des(inKey);
    FileOutputStream outputFile=null;
    try {
        outputFile=new FileOutputStream(sourceFile,true);
    }catch (java.io.FileNotFoundException e) {
        e.printStackTrace(System.err);
    }
    FileChannel outChannel=outputFile.getChannel();

    try{
        if(outChannel.size()%2!=0){
            ByteBuffer bufTemp=ByteBuffer.allocate(1);

```

```

        bufTemp.put((byte)32);
        bufTemp.flip();
        outChannel.position(outChannel.size());
        outChannel.write(bufTemp);
        bufTemp.clear();
    }
} catch(Exception ex){
    ex.printStackTrace(System.err);
    System.exit(1);
}
FileInputStream inFile=null;
try{
    inFile=new FileInputStream(sourceFile);
} catch(java.io.FileNotFoundException e){
    e.printStackTrace(System.err);
}
outputFile=null;
try {
    outputFile=new FileOutputStream(targetFile,true);
} catch (java.io.FileNotFoundException e) {
    e.printStackTrace(System.err);
}

FileChannel inChannel=inFile.getChannel();
outChannel=outputFile.getChannel();

ByteBuffer inBuf=ByteBuffer.allocate(8);
ByteBuffer outBuf=ByteBuffer.allocate(8);

try{
    String sourceStr;
    String targetStr;
    while(true){
        if (bufcontent(inChannel,inBuf)==-1) break;
    sourceStr=((ByteBuffer)(inBuf.flip())).asCharBuffer().toString();
        inBuf.clear();
        if (flag)      //若为true，则加密
            targetStr=des.enc(sourceStr,sourceStr.length());
        else          //若为false，则解密
            targetStr=des.dec(sourceStr,sourceStr.length());
        outBuf.clear();
        if (targetStr.length()==4){
            for (int i = 0; i<4; i++) {

```

```

        outBuf.putChar(targetStr.charAt(i));
    }
    outBuf.flip();
} else{
    outBuf.position(0);
    outBuf.limit(2*targetStr.length());
    for (int i = 0; i<targetStr.length(); i++) {
        outBuf.putChar(targetStr.charAt(i));
    }
    outBuf.flip();
}

try {
    outChannel.write(outBuf);
    outBuf.clear();
} catch (java.io.IOException ex) {
    ex.printStackTrace(System.err);
}
}
System.out.println (inChannel.size());
System.out.println (outChannel.size());
System.out.println ("EoF reached.");
inFile.close();
outputFile.close();
} catch(java.io.IOException e){
    e.printStackTrace(System.err);
    System.exit(1);
}
}

public FileDES(String sourceFileName,String targetFileName,String inKey,boolean encdes){
    this.sourceFileName=sourceFileName;
    this.targetFileName=targetFileName;
    this.encdes=encdes;
    getpath();
    sourceFile=new File(sourceFileName);
    targetFile=new File(targetFileName);
    this.inKey=inKey;
    if (encdes==enc)
        fileAction(enc); //执行加密操作
    else
        fileAction(dec); //执行解密操作
}

```

```
}

public static void main(String[] args){
    String srcfile=System.getProperty("user.dir")+"/soucefile.doc";
    //原始文件
    String cypfile=System.getProperty("user.dir")+"/cryptfile.doc";
    //加密后的密文文件
    String trgfile=System.getProperty("user.dir")+"/targetfile.doc";
    //解密后文件
    String passWord="ZYWX1234";           //工作密匙
    new FileDES(srcfile,cypfile,passWord,true);
    new FileDES(cypfile,trgfile,passWord,false);
}

}
```

本 章 小 结

无论在开发过程，还是应用部署，任何时候都少不了安全问题。在因特网广泛影响我们日常生活的今天，给安全性也带来了巨大问题。Java平台的基本语言和库扩展都提供了用于编写安全应用程序的极佳基础。其实，也可以说Java本身就是安全的语言和安全的计算平台，它的无指针特点、代码检验以及“沙箱”机制等，确保了系统的安全。当然，这并不意味着在开发中就不再需要使用其他的安全机制了。

第7章

Java与数据库



本章内容

- 案例1 在Applet中应用JDBC访问数据库
- 案例2 通过JDBC-ODBC桥连接数据库
- 案例3 通过Tomcat数据源访问数据库
- 案例4 JDBC连接池的实现
- 案例5 用Javabean来实现MySQL的分页显示
- 案例6 利用JDBC-ODBC查看查询结果
- 本章小结



案例1 在Applet中应用 JDBC访问数据库

案例运行效果与操作

本案例是一个通过JDBC连接后台数据库的例子，用JApplet来显示。程序运行后，界面如图7-1所示。

环境介绍：Java编写使用的是Eclipse 3.0.1，数据库MySQL 4.10，JDK版本是1.5.0_01，Tomcat版本是5.5.4。

在JDBC Driver下拉框中选择“com.mysql.jdbc.Driver”，在Database URL中输入“jdbc:mysql://localhost/mvnforum?useServerPrepStmts=false”，在User中输入用户名“root”，在Pass-

word中输入数据库的连接密码。



图7-1 运行界面

单击 **Connect** 按钮，即可连接到MySQL数据库，在文本框中如果出现下面的提示信息：

```
Using JDBC driver: com.mysql.jdbc.Driver
Database jdbc:mysql://localhost/mvnforum connected.
```

则说明数据库已经连接成功。

单击 **Disconnect** 按钮，就会断开数据库的连接。文本框的提示信息变为：

```
Using JDBC driver: com.mysql.jdbc.Driver
Database jdbc:mysql://localhost/mvnforum connected.
Database disconnected.
```

在最下面的SQL文本框中输入：“select * from mvnforumforum”，然后单击 **Execute Query** 按钮，文本框中出现如下信息：

```
Executing query: select * from mvnforumforum
```

表的字段名如下：

```
ForumID CategoryID LastPostMemberName ForumName ForumDesc ForumCreationDate
ForumModifiedDate ForumLastPostDate ForumOrder ForumType ForumFormatOption
ForumOption ForumStatus ForumModerationMode ForumPassword ForumThreadCount
ForumPostCount
```

```
1 1 Admin测试第一个随便写东西2005-04-05 08:39:14.0 2005-04-05 08:39:14.0 2005-04-05 08:48:01.0 0 0 0 0 0 1 4
```

看到上述信息，说明数据查询成功。

制作要点

1. JApplet类的用法
2. JDBC的Connection接口对象的使用
3. GetConnection()方法
4. 元数据类ResultSetMetaData的使用

步骤详解

1. JApplet的制作。

```
public class HelloWorldJ extends JApplet {  
    public void init(){  
        HelloWorldPanel contentPane = new HelloWorldPanel();  
        setContentPane(contentPane);  
    }  
    class HelloWorldPanel extends JPanel{  
        public void paintComponent(Graphics g){  
            super.paintComponent(g);  
            g.drawString("Hello World!",25,25);  
        }  
    }  
}
```

2. 通过JDBC连接数据库。

连接数据库：

```
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {  
    if (connected) {  
        taResponse.append("Database already connected.\n");  
    } else {
```

从下拉框中取得驱动程序：

```
driver = (String) cbDriver.getSelectedItem();
```

取得数据库连接的URL：

```
url = tfUrl.getText();
```

取得数据库的用户名：

```
user = tfUser.getText();
```

注册数据库驱动，通过JDBC的方式连接数据库：

```
Class.forName(driver).newInstance();
connection = DriverManager.getConnection(url, user, password);
```

连接数据库：

```
if (connection != null) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Database " + url
                + " connected.\n");
            connected = true;
        }
    });
}
```

3. 元数据类ResultSetMetaData的使用。

通过这个类可以将数据库表的每个字段的名字从数据库中调出来，显示在JTextArea中：

```
query = tfSql.getText();
Statement stmt = connection.createStatement();
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        taResponse.append("Executing query: " + query + "\n");
    }
});
rs = stmt.executeQuery(query);
```

使用元数据：

```
ResultSetMetaData rsmd = rs.getMetaData();
```

4. 匿名类的使用。

在这里使用了匿名类：

```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        taResponse.append("Database " + url + " connected.\n");
    }
});
```

程序源代码与解析

```
//chp7
/*
 * DBApplet.java 用JApplet做的Applet和一般的Applet有些不同，主要是因为Swing库类
 * 的单线程原则
 */
```

* 所以当JApplet的界面生成后，由其他线程（一般是消息分发线程）试图改变界面（如
setText()）

* 可能会导致问题发生。这时可以借助SwingUtilities.invokeLater()

* 或SwingUtilities.invokeAndWait()来解决。下面是一个用JApplet访问数据库的例子

*/

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.SwingUtilities;

public class DBApplet extends javax.swing.JApplet {

    final static private String[] jdbcDriver = { "com.mysql.jdbc.Driver", "com.informix.jdbc
.IfxDriver",
        "sun.jdbc.odbc.JdbcOdbcDriver",
        "com.borland.datastore.jdbc.DataStoreDriver",
        "com.sybase.jdbc.SybDriver", "oracle.jdbc.driver.OracleDriver",
        "COM.ibm.db2.jdbc.net.DB2Driver", "interbase.interclient.Driver",
        "weblogic.jdbc.mssqlserver4.Driver" };

    private boolean connected = false;
    //判断是否连接成功
    private Connection connection = null;
    private ResultSet rs = null;
    private String query = null;
    //查询的SQL语句

    private String rsLine = null;
    private String driver = null;
    //数据库驱动程序

    private String url = null;
    //数据库连接URL

    private String user = null;
    //数据库用户名

    private String password = null;
    //数据库用户密码

    public DBApplet() {
        initComponents();
        postInit();
    }
```

```

}

private void postInit() {
    for (int i = 0; i < jdbcDriver.length; i++) {
        cbDriver.addItem(jdbcDriver[i]);
    }
}

private void initComponents() {
    jScrollPane1 = new javax.swing.JScrollPane();
    taResponse = new javax.swing.JTextArea();
    //显示SQL查询结果的文本区域
    jPanel2 = new javax.swing.JPanel();
    jPanel1 = new javax.swing.JPanel();
    jLabel6 = new javax.swing.JLabel();
    tfSql = new javax.swing.JTextField();
    //输入sql语句的文本区域。
    btnExecute = new javax.swing.JButton();
    //SQL执行按钮
    jPanel3 = new javax.swing.JPanel();
    jLabel3 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    cbDriver = new javax.swing.JComboBox();
    //数据库驱动下拉框
    jLabel7 = new javax.swing.JLabel();
    tfUrl = new javax.swing.JTextField();
    //连接数据库的url
    jLabel9 = new javax.swing.JLabel();
    tfUser = new javax.swing.JTextField();
    //录入用户名的文本框
    jLabel10 = new javax.swing.JLabel();
    tfPassword = new javax.swing.JTextField();
    //录入用户密码的文本框
    btnConnect = new javax.swing.JButton();
    //连接button
    btnDisconnect = new javax.swing.JButton();
    //释放数据库连接的button
    setFont(new java.awt.Font("Verdana", 0, 12));
    //设置字体
    jScrollPane1.setViewportView(taResponse);
    getContentPane().add(jScrollPane1, java.awt.BorderLayout.CENTER);
    getContentPane().add(jPanel2, java.awt.BorderLayout.EAST);
}

```

```
jLabel6.setText("SQL:");
//label Sql:
jPanel1.add(jLabel6);

tfSql.setPreferredSize(new java.awt.Dimension(300, 21));
jPanel1.add(tfSql);

btnExecute.setText("Execute Query");
btnExecute.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnExecuteActionPerformed(evt);
    }
});
jPanel1.add(btnExecute);
//将jPanel1放到上边
getContentPane().add(jPanel1, java.awt.BorderLayout.SOUTH);

jPanel3.setPreferredSize(new java.awt.Dimension(550, 100));
jPanel3.setMinimumSize(new java.awt.Dimension(550, 100));
jPanel3.setMaximumSize(new java.awt.Dimension(550, 100));

jLabel3.setText("JDBC Driver:");
jPanel3.add(jLabel3);
jPanel3.add(jPanel4);

cbDriver.setPreferredSize(new java.awt.Dimension(450, 26));
cbDriver.setMinimumSize(new java.awt.Dimension(100, 26));
jPanel3.add(cbDriver);

jLabel7.setText("Database URL:");
jPanel3.add(jLabel7);
tfUrl.setPreferredSize(new java.awt.Dimension(450, 21));
jPanel3.add(tfUrl);

jLabel9.setText("User:");
jPanel3.add(jLabel9);

tfUser.setPreferredSize(new java.awt.Dimension(100, 21));
jPanel3.add(tfUser);

jLabel10.setText("Password:");
jPanel3.add(jLabel10);
//label PassWord:
tfPassword.setPreferredSize(new java.awt.Dimension(100, 21));
jPanel3.add(tfPassword);

btnConnect.setPreferredSize(new java.awt.Dimension(89, 27));
btnConnect.setMaximumSize(new java.awt.Dimension(89, 27));
```

```

btnConnect.setText("Connect");
btnConnect.setMinimumSize(new java.awt.Dimension(89, 27));
btnConnect.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnConnectActionPerformed(evt);
    }
});
jPanel3.add(btnConnect);

btnDisconnect.setText("Disconnect");
btnDisconnect.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDisconnectActionPerformed(evt);
    }
});
jPanel3.add(btnDisconnect);
//放在布局管理器的北边
getContentPane().add(jPanel3, java.awt.BorderLayout.NORTH);

}

//执行查询的SQL语句
private void btnExecuteActionPerformed(java.awt.event.ActionEvent evt) {
    if (!connected) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                taResponse.append("No database connected.\n");
            }
        });
    } else {
        if (connection == null) {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    taResponse.append("Database connection error.\n");
                }
            });
        } else {
            try {
                query = tfSql.getText();
                Statement stmt = connection.createStatement();
                SwingUtilities.invokeLater(new Runnable() {
                    public void run() {
                        taResponse.append("Executing query: " + query
                            + "\n");
                    }
                });
            }
        }
    }
}

```

```

        }
    });
    rs = stmt.executeQuery(query);
    //使用元数据
    ResultSetMetaData rsmd = rs.getMetaData();
    //还使用了元数据来判断有多少个字段
    int count = rsmd.getColumnCount();
    int i;
    rsLine = "\n 表的字段名如下:\n";
    for(int it=1;it<=count;it++){
        rsLine += rsmd.getColumnName(it) + " ";
    }
    rsLine += "\n";
    while (rs.next()) {
        for (i = 1; i <= count; i++) {
            rsLine += rs.getString(i) + " ";
        }
        rsLine += "\n";
    }
    rsLine += "\n";
    stmt.close();
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append(rsLine);
        }
    });
} catch (SQLException e) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Query failed.\n");
        }
    });
    e.printStackTrace();
}
}
}
}

//释放数据库连接
private void btnDisconnectActionPerformed(java.awt.event.ActionEvent evt) {
    if (connected) {
        try {
            if (connection != null) {

```

```

        connection.close();
        connection = null;
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                taResponse.append("Database disconnected.\n");
            }
        });
    }
} catch (SQLException e) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Database disconnecting error.\n");
        }
    });
    e.printStackTrace();
}
connected = false;
driver = null;
url = null;
user = null;
password = null;
} else {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Database already disconnected.\n");
        }
    });
}
}

//连接数据库
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {
    if (connected) {
        taResponse.append("Database already connected.\n");
    } else {
        driver = (String) cbDriver.getSelectedItem();
        url = tfUrl.getText();
        user = tfUser.getText();
        password = tfPassword.getText();
        try {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    taResponse

```

```

        .append("Using JDBC driver: " + driver + "\n");
    }
});

//注册数据库驱动通过，JDBC的方式连接数据库
Class.forName(driver).newInstance();
connection = DriverManager.getConnection(url, user, password);
//连接数据库
if (connection != null) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Database " + url
                + " connected.\n");
        }
    });
    connected = true;
}
} catch (ClassNotFoundException e) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Cannot load the driver.\n");
        }
    });
    e.printStackTrace();
} catch (SQLException e) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            taResponse.append("Cannot connect to the database.\n");
        }
    });
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}
}

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextArea taResponse;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel1;

private javax.swing.JLabel jLabel6;

```

```

private javax.swing.JTextField tfSql;
private javax.swing.JButton btnExecute;
private javax.swing.JPanel jPanel3;
private javax.swing.JLabel jLabel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JComboBox cbDriver;
private javax.swing.JLabel jLabel7;
private javax.swing.JTextField tfUrl;
private javax.swing.JLabel jLabel9;
private javax.swing.JTextField tfUser;
private javax.swing.JLabel jLabel10;
private javax.swing.JTextField tfPassword;
private javax.swing.JButton btnConnect;
private javax.swing.JButton btnDisconnect;
}

```



案例2 通过JDBC-ODBC桥连接数据库

案例运行效果与操作

按照图7-2的指示进行操作，或者在DOS窗口中执行命令：java JdbcOdbcBridge，就会看到图7-3所示的结果。

制作要点

1. ResultSet接口的用法
2. Statement类的使用

步骤详解

1. 配置ODBC数据源。

在Windows“开始”菜单中依次选择“控制面板▶管理工具▶数据源”，大家会看到如图7-4所示的对话框，然后根据你使用的数据库进行数据源的配置。



图7-2 执行Java程序

```

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//注册驱动程序
Properties prop = new Properties();
prop.put("user", "dbadmin");
prop.put("password", "rootwmp37");
prop.put("charSet", "gb2312");

conn = DriverManager.getConnection("jdbc:odbc:myforum", prop);
//conn = DriverManager.getConnection("jdbc:odbc:myforum", "dbadmin", "rootwmp37");
//获得连接
stmt = conn.createStatement();
//创建会话声明
//stmt.executeUpdate("select * from forumpost");
rst = stmt.executeQuery("select * from forumpost");
while(rst.next()){
    System.out.print(rst.getString(1)+"--");
    System.out.print(rst.getString(2)+"--");
    System.out.print(rst.getString(3)+"--");
    System.out.print(rst.getString(4));
    System.out.println();
}

```

图7-3 运行界面

2. 通过JDBC-ODBC桥连接数据库。

图7-5是通过JDBC-ODBC桥连接数据库的示意图，首先要建立好数据库myforum，然后任意建立一个4个字段的数据库表forumpost。建好之后，要在Windows中配置一下ODBC数据源。

JDBC编写数据库程序的方法如下：

- (1) 建立数据源：建立数据源是指建立ODBC数据源。
- (2) 建立连接：与数据库建立连接的标准方法是调用。

```
DriverManager.getConnection(String url, String user, String password)
```

DriverManager类用于处理驱动程序的调入，并对新的数据库连接提供支持。



图7-4 ODBC数据源管理器

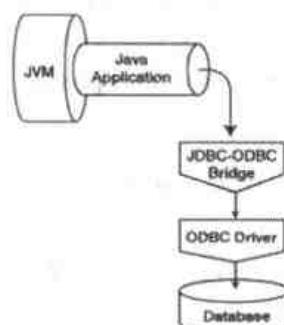


图7-5 JDBC-ODBC桥示意图

(3) 执行SQL语句。JDBC提供了Statement类来发送SQL语句，Statement类的对象由createStatement方法创建；SQL语句发送后，返回的结果通常存放在一个ResultSet类的对象中，ResultSet可以看做是一个包含由SQL返回的列名和相应值的表。ResultSet对象中维持了一个指向当前行的指针，通过一系列的getXXX方法，可以检索当前行的各个列，从而显示出来。

程序中给出了三种连接方式：

第一种：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
conn = DriverManager.getConnection("jdbc:odbc:myforum","db2admin","****");
```

第二种：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//注册驱动程序
Properties prop = new Properties();
prop.put("user", "db2admin");
prop.put("password", "****");
prop.put("charSet", "gb2312");
conn = DriverManager.getConnection("jdbc:odbc:myforum", prop);
```

第三种：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
conn = DriverManager.getConnection("jdbc:odbc:myforum");
```

程序源代码与解释

```
//chp7
import java.sql.Connection;
```

```

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

/**
 *通过JDBC-ODBC桥访问数据库
 */
public class JdbcOdbcBridge {

    public static void main(String args[]) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rst = null;
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            //注册驱动程序
            Properties prop = new Properties();
            prop.put("user", "root");
            prop.put("password", "***");
            prop.put("charSet", "gb2312");
            //设置用户名、密码、字符集
            conn = DriverManager.getConnection("jdbc:odbc:myforum", prop);
            //conn = DriverManager.getConnection("jdbc:odbc:myforum","root","***");
            //conn = DriverManager.getConnection("jdbc:odbc:myforum");
            //获得连接，上面的三种方式均可。
            stmt = conn.createStatement();
            //创建会话声明
            //stmt.execute("select * from forumpost");
            rst = stmt.executeQuery("select * from forumpost");
            while(rst.next()){
                System.out.print(rst.getString(1)+"--");
                System.out.print(rst.getString(2)+"--");
                System.out.print(rst.getString(3)+"--");
                System.out.print(rst.getString(4));
                System.out.println();
            }
            //执行SQL语句
        } catch (ClassNotFoundException e) {
            System.out.println(e.getMessage());
        } catch (SQLException e) {
    }
}

```

```
e.printStackTrace();
System.out.println(e.getErrorCode());
} finally {
    //关闭所有的连接
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException e) {
        }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException e) {
        }
}
}
```



案例3 通过Tomcat数据源访问数据库

案例运行效果与操作

在IE地址栏输入http://localhost/myfourm/test.jsp，大家会看到如图7-6所示的界面。



图7-6 运行界面

制作要点

- 1. Connection接口的用法
 - 2. Statement类的使用

步骤详解

- ## 1. Server.Xml的配置

Server.xml位于<%tomcat_home%>/conf下面。例如： C:\Java\Tomcat554\conf。

```

<Server port="8005" shutdown="SHUTDOWN">
    <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
    <GlobalNamingResources> .
        <Environment name="simpleValue" type="java.lang.Integer" value="30"/>
        <Resource name="UserDatabase" auth="Container"
            type="org.apache.catalina.UserDatabase"
            description="User database that can be updated and saved"
            factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
            pathname="conf/tomcat-users.xml" />
    </GlobalNamingResources>
    <Service name="Catalina">
        <Connector port="80" maxThreads="150" minSpareThreads="25" maxSpareThreads
="75"
            enableLookups="false" redirectPort="8443" acceptCount="100"
            connectionTimeout="20000" disableUploadTimeout="true" />
        <Connector port="8009"
            enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
    <Engine name="Catalina" defaultHost="localhost">
        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
            resourceName="UserDatabase"/>
        <Host name="localhost" appBase="webapps"
            unpackWARs="true" autoDeploy="true"
            xmlValidation="false" xmlNamespaceAware="false">
            <Context path="/myforum" docBase="myforum"
                debug="5" reloadable="true" crossContext="true">
                <Resource name="jdbc/MyDataSource" auth="Container" type="javax.sql.DataSource"
                    maxActive="100" maxIdle="30" maxWait="10000"
                    username="root" password="yourpassword" driverClassName="com.mysql
.jdbc.Driver"
                    url="jdbc:mysql://localhost/myforum?useServerPrepStmts=false"/>
            </Context>
        </Host>
    </Engine>
    </Service>
</Server>

```

推荐源代码与解答

为了方便起见，我们将数据库的连接写在JSP里面，当然你也可以写在JavaBean中。

```

<%@ page contentType="text/html; charset=gb2312" language="java" import="javax.naming.  

.Context,javax.sql.DataSource,javax.naming.InitialContext,java.sql.*"%>  

<%  

    DataSource ds=null;  

    try  

    {  

        Context initCtx=new InitialContext();  

        Context envCtx=(Context)initCtx.lookup("java:comp/env");  

        //给出JNDI上下文，返回数据源  

        ds=(DataSource)envCtx.lookup("jdbc/MyDataSource");  

        //获得数据库连接  

        if(ds!=null)  

        {  

            out.println("connection is ok!");  

            out.println("<br>");  

            Connection conn=ds.getConnection();  

            Statement stmt=conn.createStatement();  

            ResultSet rs=stmt.executeQuery("select * from forumuser");  

            while(rs.next())  

            {  

                out.println(rs.getString("UserName"));  

                out.println(rs.getString("UserPassword"));  

                out.println("<br>");  

            }  

        }  

        else  

            out.println("fail!");  

    }  

    catch(Exception e){out.println(e);}  

%>

```



案例4 JDBC连接池的实现

案例运行效果与操作

连接池是一种特殊的数据库连接类型，在关闭时不会被删除，不像常规的 Connection 对象（当常规的连接不再被引用时，垃圾收集器能删除它们）。相反，连接被缓存以备将来再次使用，从而可能带来大幅度的性能提升。本案例只体现了一个连接池的思想，一个设计良好的连接池要比这个复杂得多。

本案例是实现数据库查询结果的显示，运行结果如图7-7所示。

生产日报		生产管理			计划生产数
编号	姓名	生产日期	生产领班人	预计生产数	最后生产人/生产时间
1	王伟国	2005年04月15日 17:21 星期五	王伟国	0	2005年04月15日 17:21 星期五
2	王伟	2005年04月15日 17:20 星期五	王伟	0	2005年04月15日 17:20 星期五
3	王伟	2005年04月15日 17:20 星期五	王伟	0	2005年04月15日 17:20 星期五
4	王伟	2005年04月15日 17:20 星期五	王伟	0	2005年04月15日 17:20 星期五
5	王伟	2005年04月15日 17:20 星期五	王伟	0	2005年04月15日 17:20 星期五
6	王伟	2005年04月15日 17:20 星期五	王伟	0	2005年04月15日 17:20 星期五
7	王伟	2005年04月15日 17:20 星期五	王伟	0	2005年04月15日 17:20 星期五
8	王伟	2005年04月15日 16:50 星期五	王伟	1	2005年04月15日 16:50 星期五
9	王伟	2005年04月15日 16:50 星期五	王伟	0	2005年04月15日 16:50 星期五
10	王伟	2005年04月15日 16:50 星期五	王伟	0	2005年04月15日 16:50 星期五
11	王伟	2005年04月15日 16:49 星期五	王伟	0	2005年04月15日 16:49 星期五
12	王伟	2005年04月15日 16:49 星期五	王伟	0	2005年04月15日 16:49 星期五

图7-7 运行界面

制作要点

1. 连接池的使用
2. DriverManager连接数据库的方式

步骤详解

1. 使用JDBC连接数据库。
参见案例1在Applet中应用JDBC连接数据库。

2. 单件模式的使用。

单件设计模式：

```
public class Company {
    private static Company instance = new Company();
    private String name;
```

获取“单件”：

```
public static Company getCompany() {
    return instance;
}
//Constructor
private Company()
```

```

        this.name = "initial";
    }
    返回名称
    public String getName() {
        return name;
    }
    设置名称
    public void setName(String name) {
        this.name = name;
    }
}

```

3. 连接池的使用。

```

ConnecionManager cm = ConnecitonManager.getInstance();
Connection conn = cm.getConnection();
.....
.....

```

所有服务停止后，调用下面的方法释放所有的数据库连接：

```
cm.release();
```

程序源代码与解释

DBOptions.java用来存放连接池的配置信息，代码如下：

```

//chp7
/**
 *
 * DBOptions.java
 */
public class DBOptions {
    //设置JDBC驱动
    private String driverClassName = "com.mysql.jdbc.Driver";
    //数据库连接URL
    private String databaseURL = "jdbc:mysql://localhost/mvnforum?useServerPrepStmts
=false";
    //用户名
    private String databaseUser = "root";
    //密码
    private String databasePassword = "mysqlwmj";
    //设置最大连接数
    private int maxConnection = 20;
    //设置最长等待时间
}

```

```
private int maxTimeToWait = 2000;// 2 seconds

/**
 * @return Returns the databasePassword.
 */
public String getDatabasePassword() {
    return databasePassword;
}

/**
 * @param databasePassword
 *          The databasePassword to set.
 */
public void setDatabasePassword(String databasePassword) {
    this.databasePassword = databasePassword;
}

/**
 * @return Returns the databaseURL.
 */
public String getDatabaseURL() {
    return databaseURL;
}

/**
 * @param databaseURL
 *          The databaseURL to set.
 */
public void setDatabaseURL(String databaseURL) {
    this.databaseURL = databaseURL;
}

/**
 * @return Returns the databaseUser.
 */
public String getDatabaseUser() {
    return databaseUser;
}

/**
 * @param databaseUser
 *          The databaseUser to set.
 */
public void setDatabaseUser(String databaseUser) {
    this.databaseUser = databaseUser;
}
```

```
/*
 * @return Returns the driverClassName.
 */
public String getDriverClassName() {
    return driverClassName;
}

/**
 * @param driverClassName
 *         The driverClassName to set.
 */
public void setDriverClassName(String driverClassName) {
    this.driverClassName = driverClassName;
}

/**
 * @return Returns the maxConnection.
 */
public int getMaxConnection() {
    return maxConnection;
}

/**
 * @param maxConnection
 *         The maxConnection to set.
 */
public void setMaxConnection(int maxConnection) {
    this.maxConnection = maxConnection;
}

/**
 * @return Returns the maxTimeToWait.
 */
public int getMaxTimeToWait() {
    return maxTimeToWait;
}

/**
 * @param maxTimeToWait
 *         The maxTimeToWait to set.
 */
public void setMaxTimeToWait(int maxTimeToWait) {
    this.maxTimeToWait = maxTimeToWait;
}
}
```

ConnectionManager.java用来管理连接池的使用，连接池是采用内部类DBConnectionPool.java来实现的，代码如下：

```
package simpleforum.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Enumeration;
import java.util.Vector;

/**
 *
 * ConnectionManager.java
 */
public class ConnectionManager {

    /**
     * 使用了单件模式访问连接池，客户端只能通过ConnectionManager.getInstance()得到一个实例
     * 当客户端关闭时，应该调用release()方法关闭所有的数据库连接，并做好其他的清理工作
     */
    private static final int TIME_BETWEEN_RETRIES = 500; // 0.5 second

    // static variable
    static private ConnectionManager instance = null; // The single instance

    // instance variable
    private DBConnectionPool pool = null;

    /**
     * 私有构造方法 初始化数据库的连接
     */
    private ConnectionManager() {
        DBOptions option = new DBOptions();
        String driverClassName = option.getDriverClassName();
        try {
            Class.forName(driverClassName).newInstance();
        } catch (Exception e) {
            System.out.println("fatal ERROR: ConnectionManager: Unable to load driver = "
                    + driverClassName);
        }
        String url = option.getDatabaseURL();
        String user = option.getDatabaseUser();
    }

    /**
     * 公有方法 得到连接池的实例
     */
    public static ConnectionManager getInstance() {
        if (instance == null) {
            instance = new ConnectionManager();
        }
        return instance;
    }

    /**
     * 公有方法 从连接池中获得连接
     */
    public Connection getConnection() {
        return pool.getConnection();
    }

    /**
     * 公有方法 将连接归还给连接池
     */
    public void release(Connection conn) {
        pool.release(conn);
    }

    /**
     * 公有方法 关闭连接池
     */
    public void close() {
        pool.close();
    }
}
```

```

String password = option.getDatabasePassword();
int maxConnection = option.getMaxConnection();

//pool是实例变量，总是创建
pool = new DBConnectionPool(url, user, password, maxConnection);
}

/**
 * 返回一个单独的实例，如果这个方法是第一次被调用就创建一个新的实例
 *
 * @return ConnectionManager The single instance.
 */

public static synchronized ConnectionManager getInstance() {
    if (instance == null) {
        instance = new ConnectionManager();
    }
    return instance;
}

/**
 * 返回一个连接
 * 如果没有可以使用的连接，且连接数还没有达到最大，就创建一个新的连接
 *
 * @return Connection The connection or null
 */

Connection getConnection() {
    return pool.getConnection();
}

/**
 * 返回一个连接
 * 如果没有可以使用的连接，且连接数还没有达到最大，就创建一个新的连接
 * 如果连接数达到了最大值，按指定的时间等待，如果这个时间内有连接被释放了，就返回一个连接
 * 如果没有连接被释放，就返回null
 * @param time
 *          The number of milliseconds to wait
 * @return Connection The connection or null
 */

Connection getConnection(long time) {
    return pool.getConnection(time);
}

/**
 * 关闭所有开着的数据库连接

```

```

*
* @return true if the pool is empty and balance false if the pool has
*         returned some connection to outside
*/
boolean release() {
    return pool.release();
}

/**
* 连接池使用内部类来实现
*/
class DBConnectionPool {
    private int checkedOut = 0;
    private Vector freeConnections = new Vector();
    private int maxConn = 0;
    private String password = null;
    private String URL = null;
    private String user = null;
    /**
     * Creates new connection pool. NOTE: new an instance of this class is
     * lightweight since it does not create any connections
     *
     * @param URL
     *          The JDBC URL for the database
     * @param user
     *          The database user, or null
     * @param password
     *          The database user password, or null
     * @param maxConn
     *          The maximal number of connections, or 0 for no limit
     */
    public DBConnectionPool(String URL, String user, String password,
                           int maxConn) {
        this.URL = URL;
        this.user = user;
        this.password = password;
        this.maxConn = maxConn;
    }
}

```

```

    * Checks in a connection to the pool. Notify other Threads that may be
    * waiting for a connection.
    *
    * @param con
    *          The connection to check in
    */
synchronized void freeConnection(Connection con) {
    if (con != null) {//make sure that the connection is not null
        if (checkedOut <= 0) {
            try {
                System.out.println("ConnectionManager: about to close the
orphan connection.");
                con.close();
            } catch (SQLException ex) {
            }
        } else {
            freeConnections.addElement(con);
            checkedOut--;
            notifyAll();
        }
    }
}

/**
 * Checks out a connection from the pool. If no free connection is
 * available, a new connection is created unless the max number of
 * connections has been reached. If a free connection has been closed by
 * the database, it's removed from the pool and this method is called
 * again recursively.
 */
synchronized Connection getConnection() {
    Connection con = null;

    while ((freeConnections.size() > 0) && (con == null)) {
        con = (Connection) freeConnections.firstElement();
        freeConnections.removeElementAt(0);
        try {
            if (con.isClosed()) {
                System.out.println("Removed bad connection in
DBConnectionPool.");
                con = null; // to make the while loop to continue
            }
        } catch (SQLException e) {
    }
}

```

```

        con = null; // to make the while loop to continue
    }
} // while

if (con == null) { // cannot get any connection from the pool
    if (maxConn == 0 || checkedOut < maxConn) {
        // maxConn = 0 means unlimited connections
        con = newConnection();
    }
}
if (con != null) {
    checkedOut++;
}
return con;
}

/**
 * 返回一个连接
 * 如果没有可以使用的连接，且连接数还没有达到最大，就创建一个新的连接
 * 如果连接数达到了最大值，按指定的时间等待，如果这个时间内有连接被释
放了，就返回一个连接
 * 如果没有连接被释放，就返回null
 * @param timeout
 *          The timeout value in milliseconds
 */
Connection getConnection(long timeout) {
    long startTime = System.currentTimeMillis();
    Connection con;
    while ((con = getConnection()) == null) {
        long elapsedTime = System.currentTimeMillis() - startTime;
        if (elapsedTime >= timeout) {
            return null;
        }

        long timeToWait = timeout - elapsedTime;
        //每次等待的时间不能超过TIME_BETWEEN_RETRIES
        if (timeToWait > TIME_BETWEEN_RETRIES)
            timeToWait = TIME_BETWEEN_RETRIES;
        try {
            Thread.sleep(timeToWait);
        } catch (InterruptedException e) {
        }
    }
    return con;
}

```

```

    }

    /**
     * 关闭所有可利用的连接
     *
     * @return true if the pool is empty and balance false if the pool has
     *         returned some connection to outside
     */
    synchronized boolean release() {
        boolean retValue = true;
        Enumeration allConnections = freeConnections.elements();
        while (allConnections.hasMoreElements()) {
            Connection con = (Connection) allConnections.nextElement();
            try {
                con.close();
            } catch (SQLException e) {
                System.out.println("Cannot close connection in
DBConnectionPool.");
            }
        }
        freeConnections.removeAllElements();
        if (checkedOut != 0) {
            retValue = false;
            System.out.println("ConnectionManager: the built-in connection
pool is not balanced.");
        }
        checkedOut = 0;
        return retValue;
    }

    /**
     * 使用指定的用户名和密码创建一个数据库连接
     */
    private synchronized Connection newConnection() {
        Connection con = null;
        try {
            //匿名登录，不提倡使用
            if (user == null) {
                con = DriverManager.getConnection(URL);
            } else {
                con = DriverManager.getConnection(URL, user, password);
            }
            con.setAutoCommit(true);
        }
    }
}

```

```
        } catch (SQLException e) {
            System.out.println(
                "Cannot create a new connection in
DBConnectionPool. URL = "
                + URL+ e.getMessage());
            return null;
        }
        return con;
    }
}
```



案例5 用JavaBean实现MySQL的分页显示

案例运行效果与操作

在IE地址栏输入http://localhost/myforum/page3.jsp，将看到如图7-8所示的界面。单击“下一页”，将看到如图7-9所示的界面。



图7-8 运行界面



图7-9 单击“下一页”后的界面

单击“尾页”，将看到如图7-10所示的界面。



图7-10 单击“尾页”后的界面

制作要点

1. PreparedStatement()的使用
2. 元数据类ResultSetMetaData的用法

步骤详解

1. 制作思路。

将分页的所有操作均封装在JavaBean中，向JavaBean中传入必要的参数即可实现分页。

2. 通过Tomcat数据源连接数据库。

写在JavaBean中的例子：

```
public class DataBaseConn {
    public static synchronized Connection getConnection(){
        try {
            Context initCtx = new javax.naming.InitialContext();
            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            DataSource ds = (DataSource) envCtx.lookup("jdbc/MyDataSource");
            return ds.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (NamingException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

3. 封装SQL语句操作。

```
public class QueryHelp {
    * 传入一个查询SQL语句，然后将结果放入Vector中
    public static Vector getHelp(String querySql) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultset = null;
        Vector vector = new Vector();
        //这里不能使用空句柄
        //教训，最好不要使用null，除非你能够通过下面的语句取得一个对象
        //String对象也是如此
        try {
            connection = DataBaseConn.getConnection();
```

```

connection.setReadOnly(true);
statement = connection.prepareStatement(querySql);
resultset = statement.executeQuery();
ResultSetMetaData resultsetmetadata = resultset.getMetaData();
int i = resultsetmetadata.getColumnCount();
String as[] = new String[i];
for (int j = 1; j <= i; j++)
    as[j - 1] = (resultsetmetadata.getColumnName(j)).toUpperCase();
//所有的key值均为字段的大写
Hashtable hashtable;
for (; resultset.next(); vector.addElement(hashtable)) {
    hashtable = new Hashtable();
    for (int k = 1; k <= i; k++) {
        Object obj = resultset.getObject(k);
        String s = as[k - 1];
        if (obj != null)
            hashtable.put(s, obj);
        else
            hashtable.put(s, "");
    }
}
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    try {
        if (resultset != null)
            resultset.close();
    } catch (SQLException e) {
    }
    try {
        if (statement != null)
            statement.close();
    } catch (SQLException e) {
    }
    try {
        if (connection != null)
            connection.close();
    } catch (SQLException e) {
    }
}
}
return vector;
}
*/

```

```

* 使用本方法应该注意，只能使用统计函数count(*), 例如:
* select count(*) from forumpost where forumid=1;
* @param querySql
* @return
*/
public static int getInt(String querySql) {
    int temp = 0;
    Connection conn = DataBaseConn.getConnection();
    PreparedStatement statement = null;
    ResultSet rst = null;
    try{
        statement = conn.prepareStatement(querySql);
        rst = statement.executeQuery();
        while(rst.next()){
            temp = rst.getInt(1);
        }
    }catch(SQLException e){
        System.out.println("出现异常: \n"+ "QueryHelp.getInt(\"+querySql+)\n");
        System.out.println("下面是详细信息: \n");
        e.printStackTrace();
    }finally{
        try {
            if (rst != null)
                rst.close();
        } catch (SQLException e) {
        }
        try {
            if (statement != null)
                statement.close();
        } catch (SQLException e) {
        }
        try {
            if (conn != null)
                conn.close();
        } catch (SQLException e) {
        }
    }
    return temp;
}

```

第7章 代码与解析

实现分页的JavaBean如下：

```
//chp7
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Hashtable;
import java.util.Vector;

import simpleforum.database.DataBaseConn;
import simpleforum.database.QueryHelp;

public class SimplePageDb2 {
    private int curPage; //当前第几页;
    private int maxPage; //共有多少页;
    private int maxRowCount; //总共多少行
    private int rowsPerPage=10; //每页有多少行, 默认为10行
    private Vector vector = new Vector(); //用来存放最后的查询结果
    private String countSql = ""; //统计总的查询结果数目的SQL
    private String selectSql = ""; //查询SQL语句
    private String formName = "item"; //form表单的名字, 默认为item

    public void setFormName(String formName){
        this.formName = formName;
    }

    public String getFormName(){
        return this.formName;
    }

    public void setRowsPerPage(int rowsPerPage){
        this.rowsPerPage=rowsPerPage;
    }

    public int getRowsPerPage(){
        return this.rowsPerPage;
    }

    public void setCurPage(int curPage){
        this.curPage=curPage;
    }

    public int getCurPage(){
        return this.curPage;
    }
}
```

```

}

//设置查询总数的SQL语句，当然不是必需的
public void setCountSql(String countSql){
    this.countSql = countSql;
}
public String getCountSql(){
    return this.countSql;
}
public void setSelectSql(String selectSql){
    this.selectSql=selectSql;
}
public String getSelectSql(){
    return this.selectSql;
}
//返回查询结果的总记录数
public void setMaxRowCount(){
    this.maxRowCount = QueryHelp.getInt(getCountSql());
}
public int getMaxRowCount(){
    return this.maxRowCount;
}

//根据总数计算总共有多少页
public void setMaxPage() {
    this.maxPage = (this.maxRowCount % this.rowsPerPage == 0) ?
this.maxRowCount
                    / this.rowsPerPage
                    : this.maxRowCount / this.rowsPerPage + 1;
}
public int getMaxPage(){
    return this.maxPage;
}

//返回查询的结果
//修改这个方法，根据不同数据库的特点实现分页显示
public Vector getResult(){
    setMaxRowCount();//总结果数
    setMaxPage();//总的页数
    String objectSql=this.selectSql+" limit "+(getCurPage()-1)*getRowsPerPage()+" ,
"+getCurPage()*getRowsPerPage();
    Vector = QueryHelp.getHelp(objectSql);
    return vector;
}

```

```

//修改这个方法可以改变分页的显示样式
public String getPageInfo(){
    StringBuffer sb = new StringBuffer();
    //首先输出页面中JS实现的页面跳转
    sb.append("<script languange=\"javascript\">")
        .append("function Jumping(){")
        .append("    document."+getFormName()+"submit();")
        .append("    return;")
        .append("}")
        .append("function gotoPage(pagenum){")
        .append("    document."+getFormName()+"jumpPage.value=pagenum;")
        .append("    document."+getFormName()+"submit();")
        .append("    return;")
        .append("}")
        .append("</script>");

    //这个可以改变分页显示的样式
    sb.append( "<table>")
        .append("<tr>")
        .append("<td>")
        .append("每页"+getRowsPerPage()+"行")
        .append("共"+getMaxRowCount()+"行")
        .append("第"+getCurPage()+"页")
        .append("共"+getMaxPage()+"页")
        .append("<br>")
        .append("");
    if(getCurPage() == 1){
        sb.append(" 首页 &ampnbsp上一页");
    } else {
        sb.append("<a href=\"javascript:gotoPage(1)\">首页</a>")
            .append("&ampnbsp&ampnbsp<a href=\"javascript:gotoPage(\"+(getCurPage()-1)+\")\">上一页</a>");
    }
    if(getCurPage() == getMaxPage()){
        sb.append("&ampnbsp&ampnbsp下一页&ampnbsp&ampnbsp&ampnbsp尾页");
    } else {
        sb.append("<a href = \"javascript:gotoPage(\"+(getCurPage()+1)+\")\">下一页</a>")
            .append("&ampnbsp&ampnbsp<a href = \"javascript:gotoPage(\"+getMaxPage()+(\"尾页</a>\"");
    }
    sb.append("&ampnbsp&ampnbsp转到第")
        .append("<select name=\"jumpPage\" onchange=\"Jumping()\">");

```

```

        for(int i = 1;i<=getMaxPage();i++){
            if(i==getCurPage()){
                sb.append("<option selected value=\""+i+"\">" + i + "</option>");
            } else {
                sb.append("<option value=\""+i+"\">" + i + "</option>");
            }
        }
        sb.append("</select>页")
        .append("</td>")
        .append("</tr>")
        .append("</table>")
        .append("");
    return sb.toString();
}
}

```

分页JavaBean在JSP中的使用代码如下：

```

<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="simpleforum.page.*"%>
<html>
<head>
    <title> 分页测试 </title>
</head>
<%
    String countSql = "select count(*) from forumpost where answerPostId= 0 ";
    String jumpPage = (request.getParameter("jumpPage")==null)? "1":request.getParameter("jumpPage");
    String objSql = "select * from forumpost where answerPostId=0 ";
    SimplePageDb2 sp = new SimplePageDb2();
    sp.setRowsPerPage(6); //设置每页显示多少行记录
    sp.setCurPage(Integer.parseInt(jumpPage)); //设置要跳转到哪一页
    sp.setCountSql(countSql); //设置统计所有记录说的SQL
    sp.setSelectSql(objSql); //设置要查询的SQL
    sp.setFormName("item2"); //设置form表单的名字
    java.util.Vector vector = sp.getResult(); //将查询结果放在Vector中
    //pageBean这个名字不要改变，否则分页无法使用
%>
<body>
    <form name="item2" method="post" action="page3.jsp">
        <table border="1" bordercolor="blue">
            <tr>

```

```

        <td>顺号</td>
        <td>帖子标题</td>
        <td>发言人</td>
    </tr>
<%
int start = (Integer.parseInt(jumpPage)-1)*sp.getRowsPerPage()+1;
String PostName = "";
String SpokesMan = "";
out.println("vector.size()="+vector.size());
for(int i=0 ;i<vector.size();i++){
    java.util.Hashtable hash =(java.util.Hashtable)vector.elementAt(i);
    PostName = (String)hash.get("POSTNAME");
    SpokesMan = (String)hash.get("SPOKESMAN");
}
%>
<tr>
    <td><%=start+i%></td>
    <td><%=PostName%></td>
    <td><%=SpokesMan%></td>
</tr>
<%
}
%>
</table>
<%=sp.getPageInfo()%>
</form>
</body>
</html>

```



案例6 利用JDBC – ODBC查看查询结果

案例运行效果与操作

本案例实现数据库SQL语句查询结果的显示，程序运行后，界面如图7-11所示。界面上方显示输入的SQL语句，下方则是执行语句返回的结果。

制作要点

1. JDBC的Connection接口的用法
2. ResultSet对象返回SQL语句的执行结果

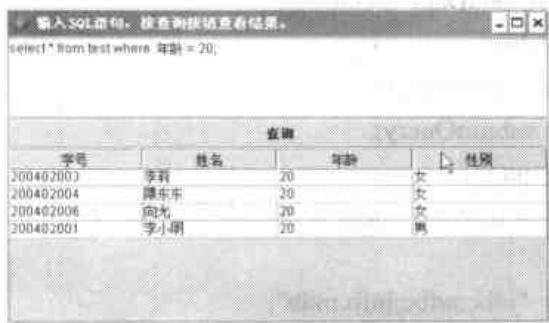


图7-11 运行界面

步骤详解

- 利用JDBC-ODBC进行数据库连接:

```
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
connection = DriverManager.getConnection(url, username, password );
```

- 执行SQL语句，利用ResultSet对象返回结果:

```
statement = connection.createStatement();
resultSet = statement.executeQuery( query );
```

- 使用Vector数组存储多条记录:

```
Vector currentRow = new Vector();
```

程序源代码与解释

```
//chp7
/*
 * 显示查询数据库结果
 */

import java.applet.*;
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class QueryEvent extends JFrame {
    private Connection connection;
    //数据库变量定义
    private Statement statement;
    private ResultSet resultSet;
```

```
private ResultSetMetaData metaData;
private JTable table;
private JTextArea inputQuery;
private JButton submitQuery;
public QueryEvent() {
    super( "输入SQL语句, 按查询按钮查看结果" );
    //输出标题
    String url = "jdbc:odbc:info.mdb";
    String username = "infodb";
    String password = "123456";
    try {
        Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
        //加载odbc数据驱动
        connection = DriverManager.getConnection(url, username, password );
    }
    catch ( ClassNotFoundException cnfex ) {
        System.err.println( "装载 JDBC/ODBC 驱动程序失败。" );
        cnfex.printStackTrace();
        System.exit( 1 );
    }
    catch ( SQLException sqlex ) {
        System.err.println( "无法连接数据库" );
        sqlex.printStackTrace();
        System.exit( 1 );
    }
    String test="select * from test where 年龄 = 20;";
    inputQuery = new JTextArea( test, 4, 30 );
    submitQuery = new JButton( "查询" );
    submitQuery.addActionListener(
        new ActionListener() {
            public void actionPerformed( ActionEvent e )
            {
                getTable();
            }
        }
    );
    //定义Button监听
};
JPanel topPanel = new JPanel();
//创建布局界面
topPanel.setLayout( new BorderLayout() );
topPanel.add( new JScrollPane( inputQuery ), BorderLayout.CENTER );
topPanel.add( submitQuery, BorderLayout.SOUTH );
table = new JTable();
```

```

Container c = getContentPane();
c.setLayout( new BorderLayout() );
c.add( topPanel, BorderLayout.NORTH );
c.add( table, BorderLayout.CENTER );
getTable();
setSize( 400, 300 );
show();
//显示窗口
}
private void getTable() {
    try {
        String query = inputQuery.getText();
        statement = connection.createStatement();
        resultSet = statement.executeQuery( query ); //执行SQL语句
        QueryResultSet( resultSet ); //显示查询结果
    }
    catch ( SQLException sqlex ) {
        sqlex.printStackTrace();
    }
}
private void QueryResultSet( ResultSet row ) throws SQLException {
    boolean moreRecords = row.next();
    //定位到第一条记录
    if ( ! moreRecords ) {
        JOptionPane.showMessageDialog( this, "无记录" );
        setTitle( "无记录显示" );
        return;
    } //如果没有记录，则显示消息
    Vector columnHeads = new Vector(); //声明向量对象并实例化向量
    Vector rows = new Vector();
    try {
        ResultSetMetaData recor = row.getMetaData();
        //获取字段的名称
        for ( int i = 1; i <= recor.getColumnCount(); ++i )
            columnHeads.addElement( recor.getColumnName( i ) );
        do { //获取记录集
            rows.addElement( getNextRow( row, recor ) );
        } while ( row.next() );
        //显示查询结果
        table = new JTable( rows, columnHeads );
        JScrollPane scroller = new JScrollPane( table );
    }
}

```

```
        Container c = getContentPane();
        c.remove(1);
        c.add( scroller, BorderLayout.CENTER );
        c.validate();
    }
    catch ( SQLException sqlex ) {
        sqlex.printStackTrace();
    }
}
private Vector getNextRow( ResultSet row,ResultSetMetaData recor ) throws
SQLException{
    Vector currentRow = new Vector();
    for ( int i = 1; i <= recor.getColumnCount(); ++i )
        currentRow.addElement( row.getString( i ) );
    return currentRow;
    //返回一条记录
}
public void disconnect() {
    try {
        connection.close();
        //关闭数据库连接
    }
    catch ( SQLException sqlex ) {
        System.err.println( "无法关闭数据库连接" );
        sqlex.printStackTrace();
    }
}
public static void main( String args[] ) {
    final QueryEvent ben = new QueryEvent();
    ben.addWindowListener(
        new WindowAdapter() {
            public void windowClosing( WindowEvent e ) {
                ben.disconnect();
                System.exit( 0 );
            }
        }
    );
}
}
```

本 章 小 结

目前，数据库连接最常用的是案例3中所讲解的方法，即通过Web中间件自带的数据源连接数据库，这种方式操作简单、编程方便、效率高，但是有一个缺点，就是无法通过应用程序监控数据库的连接的情况。通过JDBC连接池可以避免上述的缺陷，可是这种方式对程序员的要求比较高，感兴趣的朋友可以尝试一下编写自己的连接池组件。案例5是一个在B/S结构应用开发过程中经常遇到的分页问题，读者可以针对不同的数据库写成更佳的分页JavaBean。

第8章

Java与Servlet



本章内容

- 案例1 利用Servlet打开非HTML格式的文档
- 案例2 Servlet和JSP的通信
- 案例3 Servlet与servlet的通信
- 案例4 Servlet动态生成图像
- 案例5 用Servlet连接数据库
- 案例6 通过Servlet实现页面注册和登录
- 案例7 运用Servlet实现BBS功能
- 案例8 倾听Web服务器信息
- 本章小结



案例1 利用Servlet打开非 HTML格式的文档

案例运行效果与操作

Java Servlet编程可以很方便地将HTML文件发送到客户端Web浏览器。然而许多站点还允许访问非HTML格式的文档，包括Adobe PDF、Microsoft Word和Micorsoft Excel等。事实上这些非HTML格式只要能用MIME类型表示，就可以利用Servlet来发送。本案例将以PDF和Microsoft Word文件为例，介绍如何使用Servlet传送非HTML格式文件，以及与防火墙交互的方法。

首先在IE地址栏里输入http://localhost/应用程序名称/download.jsp，单击“下载pdf文档”链接，就会出现如图8-1所示的画面。然后单击“保存”按钮，将pdf文档保存到本地磁盘上。



图8-1 运行界面

制作要点

1. 使用 javax.servlet.http.HttpServletResponse接口描述返回客户端的HTTP响应
2. 通过javax.servlet.ServletResponse接口，Servlet响应客户端的一个MIME实体，可能是一个HTML页、图像数据或其他MIME格式

步骤详解

1. Servlet输出流。

Servlet的响应有：

- 一个输出流，浏览器根据它的内容类型（如text/HTML）进行解释。
- 一个HTTP错误响应，重定向到另一个URL、Servlet或JSP。

只要将文件写到Servlet的输出流中，就可以利用Servlet在浏览器中打开一个文件。尽管这看起来非常简单，但在打开非HTML格式文档（比如二进制数据或多媒體文件）的时候，仍要注意一些要点。

首先要从获得Servlet的输出流开始：

```
ServletOutputStream out = res.getOutputStream();
```

2. MIME类型。

互联网上使用MIME（Multipurpose Internet Mail Extensions，多用途互联网邮件扩展协议）来传送混合格式、多媒体和二进制数据文件。如果要在Servlet的response对象中打开某个文档，就必须设置该文档的MIME类型。下面这个例子中我们将打开PDF文档。

Web浏览器使用MIME类型来识别非HTML文档，并决定如何显示该文档内的数据。

将插件（plug-in）与MIME类型结合使用，则当Web浏览器下载MIME类型指示的文

档时，就能够启动相应插件处理此文档。某些MIME类型还可以与外部程序结合使用，浏览器下载文档后会启动相应的外部程序。

PDF文件的MIME类型是“application/pdf”。要用Servlet来打开一个PDF文档，需要将response对象中header的content类型设置成“application/pdf”：

```
// MIME type for pdf doc
res.setContentType( "application/pdf" );
```

另外，若要打开一个Microsoft Word文档，你就要将response对象的content类型设置成“application/msword”：

```
// MIME type for MSWord doc
res.setContentType( "application/msword" );
```

如果是一个Excel文档，则使用MIME类型“application/vnd.ms-excel”。其中，vnd表示该应用程序的制造者，必须将它包含在MIME类型里才能够打开该类文档。

有时候浏览器不能识别文档的MIME类型。通常这是由于没有安装这些文档需要的插件而导致的。这种情况下，浏览器会弹出一个对话框，询问用户是否需要打开该文件或是将它保存到本地磁盘上。

3. content-disposition。

一种叫做content-disposition的HTTP response header允许Servlet指定文档表示的信息。使用这种header，就可以将文档指定成单独打开（而不是在浏览器中打开），还可以根据用户的操作来显示。如果用户要保存文档，还可以为该文档建议一个文件名。这个建议名称会出现在Save As对话框的“文件名”栏中。如果没有指定，则对话框中就会出现Servlet的名字。

在Servlet中，需要将header设置成下面这样：

```
res.setHeader("Content-disposition","attachment; filename=" +"Example.pdf" );
```

如果要打开的是Microsoft Word文件，可以设成：

```
res.setHeader("Content-disposition","attachment; filename" +"Example.doc" );
```

4. 封装非HTML文档。

完成上述工作后，剩下的就非常简单了。根据待传送文件的名字，创建一个java.net.URL对象。交给URL构造器的字符串必须是指向该文件的一个有效URL地址。在这个例子中，需要打开Adobe employment格式的文档：

```
String fileURL = "http://localhost/servlet-study/attachments/adobeapp.pdf";
```

URL字符串也可以类似于http://www.gr.com/pub/somefile.doc或http://www.gr.com/pub/somefile.xls。但必须确保待传送文件类型与先前在HTTP response对象中设置的MIME类型一致：

```
URL url = new URL( fileURL );
```

5. 防火墙。

如果需要通过防火墙，最后一件要考虑的事情就是你的URL链接。首先应当搜集所用代理服务器的相关信息，例如主机名称和端口号等。

如果使用的是Java2，应该从URL对象类中创建一个URLConnection对象，并设置下列系统属性：

```
URLConnection conn = url.openConnection();
```

如果代理服务器需要认证，则使用用户名和密码连接外网：

```
String authentication = "Basic " + new
sun.misc.BASE64Encoder().encode("username:password".getBytes());
System.getProperties().put("proxySet", "true");
System.getProperties().put("proxyHost", PROXY_HOST);
```

代理服务器主机：

```
System.getProperties().put("proxyPort", PROXY_PORT);
```

代理服务器端口：

```
conn.setRequestProperty("Proxy-Authorization", authentication);
```

如果使用的是JDK 1.1，则不能设置这些系统属性。这种情况下，可以根据所用代理服务器的信息创建java.net.URL对象（假设不需认证）：

```
url = new URL("http", PROXY_HOST, Integer.parseInt(PROXY_PORT), fileURL );
```

6. 深入工作。

开始阅读传送的文档之前，首先要从URLConnection（或URL）对象中获得输入流InputStream。在这个例子中，用BufferedInputStream将InputStream封装起来。

如果采用URLConnection，可以尝试如下代码：

```
BufferedInputStream bis = new BufferedInputStream(conn.getInputStream());
```

如果使用URL，则可用下列代码：

```
BufferedInputStream bis = new BufferedInputStream(url.openStream());
```

一旦完成上述操作后，只要简单地将InputStream中的字节写入到Servlet的输出流OutputStream中即可：

```
BufferedOutputStream bos = new BufferedOutputStream(out);
byte[] buff = new byte[2048];
```

```

int bytesRead;
// Simple read/write loop.
while(-1 != (bytesRead = bis.read(buff, 0, buff.length))) {bos.write(buff, 0, bytesRead);}

```

在最后的代码块中，关闭这些流。

程序源代码与解析

```

//chp8

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.netURLConnection;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/** 
 * 这个Servlet用于打开非HTML文件
 */
public class OpenNonHtml extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        //通过ServletOutputStream打开一个输出流。
        ServletOutputStream out = response.getOutputStream();
        //设置数据mime类型
        response.setContentType("application/pdf"); // MIME type for pdf doc
        //为fileURL创建输入流
        String fileURL = "http://localhost/servlet-study/attachments/adobeapp.pdf";
        //Content-disposition header -不在浏览器中打开，设置"Save As..." 的文件名
        response.setHeader("Content-disposition", "attachment;filename="
            + "Example.pdf");
        //在这里随意定义文件的名字，例如“Example.pdf”
    }
}

```

```
//PROXY_HOST和PROXY_PORT为代理主机和代理端口，  
//可不需验证可通过防火墙，否则，需要设置系统属性  
//并使用URLConnection.getInputStream().  
BufferedInputStream bis = null;  
BufferedOutputStream bos = null;  
try {  
    URL url = new URL(fileURL);  
// 防火墙  
    URLConnection conn = url.openConnection();  
    // 使用Buffered Stream进行读写  
    bis = new BufferedInputStream(conn.getInputStream());  
    bos = new BufferedOutputStream(out);  
    byte[] buff = new byte[2048];  
    int bytesRead;  
    // 简单的读写循环  
    while (-1 != (bytesRead = bis.read(buff, 0, buff.length))) {  
        bos.write(buff, 0, bytesRead);  
    }  
} catch (final MalformedURLException e) {  
    System.out.println("MalformedURLException.");  
    throw e;  
} catch (final IOException e) {  
    System.out.println("IOException." + e.getMessage());  
    throw e;  
} finally {  
    if (bis != null)  
        bis.close();  
    if (bos != null)  
        bos.close();  
}
```



案例2 Servlet和JSP的通信

本例来讲解Servlet和Jsp之间的通信，主要的文件有addemployee.jsp、addemployeesuccess.jsp、EmployeeDataBean.java、AddEmployeeServlet.java。Web应用程序在Tomcat下的目录结构如图8-2所示。



图8-2 目录结构

案例运行效果与操作

启动Tomcat，在IE的地址栏中录入http://localhost/servlet-study/sjsp/index，就会看到图8-3所示的界面。

单击“提交”按钮，就会看到图8-4所示的界面。说明职工添加成功，即Servlet和JSP通信成功。

图8-3展示了运行界面。在地址栏输入http://localhost/servlet-study/sjsp/index。下方是一个表单，包含以下输入框：
 ID号：123
 姓名：ZhangSan
 地址：City of QingDao
 年龄：23
 [提交] [重填]

图8-3 运行界面

图8-4展示了结果显示界面。上方显示了通过三种方式（request、session）从Servlet接收到的数据：
 第一种方式得到数据：
 ID号：123
 姓名：ZhangSan
 地址：City of QingDao
 年龄：23
 通过request取到的：
 ID号：123
 姓名：ZhangSan
 地址：City of QingDao
 年龄：23
 通过session取到的：
 ID号：123
 姓名：ZhangSan
 地址：City of QingDao
 年龄：23
 提交：“employee”对象。

图8-4 结果显示

制作要点

Servlet通过方法控制页面的流转。

步骤详解

1. 总的流程。

AddEmployeeServlet.java通过request.getParameter(String arg0)方法从JSP页面采集信息，然后将这些信息放在DataBean EmployeeDataBean.java中，再将DataBean对象放

入request对象的一个属性中去，Servlet通过方法来控制页面的扭转，即根据不同的请求跳转到指定的页面。

```
request.getRequestDispatcher("/test/sjsp/*.jsp").forward(request, response)
```

2. Servlet的部署。

配置web.xml，部署Servlet：

```
<servlet>
    <servlet-name>AddEmployeeServlet</servlet-name>
    <servlet-class>ninth.AddEmployeeServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AddEmployeeServlet</servlet-name>
    <url-pattern>/sjsp/*</url-pattern>
</servlet-mapping>
```

程序源代码与解释

Addemployee.jsp页面主要用来采集职工数据，包括ID号、用户名、地址、年龄。

```
<%@ page contentType="text/html;charset=gb2312" language="java" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
    <title>jsp和Servlet通信</title>
</head>
<%
    String contextPath = request.getContextPath();
%>
<body>
    <form method="post" action="<%=contextPath%>/sjsp/AddEmployeeServlet">
        <table border="1" cellpadding="1" cellspacing="3">
            <tr>
                <td>ID号：</td>
                <td><input type="text" name="id"></td>
            </tr>
            <tr>
                <td>姓名：</td>
                <td><input type="text" name="name"></td>
            </tr>
            <tr>
                <td>地址：</td>
                <td><input type="text" name="address"></td>
            </tr>
        </table>
    </form>
</body>
```

```

        </tr>
        <tr>
            <td>年龄: </td>
            <td><input type="text" name="age"></td>
        </tr>
        <tr>
            <td><input type="submit" name="submit" value="提交"></td>
            <td><input type="reset" name="reset" value="重填"></td>
        </tr>
    </table>
</form>
</body>
</html>

```

EmployeeDataBean.java是一个DataBean，用来存放职工信息。

```

//chp8
/**
 *
 * EmployeeDataBean.java
 */
public class EmployeeDataBean {
    private String id="";
    private String name="";
    private String address="";
    private String age="";

    /**
     * @return Returns the address.
     */
    public String getAddress() {
        return address;
    }
    /**
     * @param address The address to set.
     */
    public void setAddress(String address) {
        this.address = address;
    }
    /**
     * @return Returns the age.
     */
    public String getAge() {

```

```

        return age;
    }
    /**
     * @param age The age to set.
     */
    public void setAge(String age) {
        this.age = age;
    }
    /**
     * @return Returns the id.
     */
    public String getId() {
        return id;
    }
    /**
     * @param id The id to set.
     */
    public void setId(String id) {
        this.id = id;
    }
    /**
     * @return Returns the name.
     */
    public String getName() {
        return name;
    }
    /**
     * @param name The name to set.
     */
    public void setName(String name) {
        this.name = name;
    }
}

```

AddEmployeeServlet.java是一个Servlet，用来从addemployee.jsp采集数据，控制页面的流转。

```

//chp8
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```

```
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 */
public class AddEmployeeServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html; charset=gb2312");
        String contextPath = request.getContextPath();
        //取得请求的上下文
        HttpSession session = request.getSession();
        //取得session
        String pathInfo = request.getPathInfo();
        //取得请求的路径
        System.out.println("request.getPathInfo()=" + pathInfo);
        // /index /AddEmployeeServlet
        if (pathInfo.trim().equals("/AddEmployeeServlet")) {
            System.out.println("requestPath=" + requestPath);
            String id = request.getParameter("id");
            String name = request.getParameter("name");
            String address = request.getParameter("address");
            String age = request.getParameter("age");
            EmployeeDataBean edb = new EmployeeDataBean();
            //将获得的参数放入DataBean edb中
            edb.setId(id);
            edb.setName(name);
            edb.setAddress(address);
            edb.setAge(age);
            //将edb对象放入请求的属性“employee”中
            request.setAttribute("employee", edb);
            //将edb对象放入session的属性“employee”中
            session.setAttribute("employee", edb);
            //根据请求将页面跳转到指定的页面
            request.getRequestDispatcher("/test/sjsp/addemployeesuccess.jsp").forward
                (request, response);
        } else if (pathInfo.trim().equals("/index")){
            request.getRequestDispatcher("/test/sjsp/addemployee.jsp").forward(request,
                response);
        }else{
            request.getRequestDispatcher("/test/sjsp"+pathInfo).forward(request,
                response);
        }
    }
}
```

```

        }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        doGet(request, response);
    }
}

```

addemployeesuccess.jsp使用了三种方式来展示从Servlet传来的数据。

```

<%@ page contentType="text/html; charset=gb2312" language="java"%>
<%@ page import="ninth.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<jsp:useBean id="edb" class="ninth.EmployeeDataBean" />
<jsp:useBean id="employee" type="ninth.EmployeeDataBean" scope="request"/>
<html>
<head>
    <title> jsp和Servlet通信</title>
</head>
<%
    String contextPath = request.getContextPath();
    edb = (EmployeeDataBean)request.getAttribute("employee");
%>
<body>
    第一种方式得到数据:
        <table border="1" cellpadding="1" cellspacing="3">
            <tr>
                <td>ID号: </td>
                <td><%=employee.getId()%></td>
            </tr>
            <tr>
                <td>姓名: </td>
                <td><%=employee.getName()%></td>
            </tr>
            <tr>
                <td>地址: </td>
                <td><%=employee.getAddress()%></td>
            </tr>
            <tr>
                <td>年龄: </td>
                <td><%=employee.getAge()%></td>
            </tr>
        </table>

```

通过request取到的:

```
<br>
<table border="1" cellpadding="1" cellspacing="3">
<tr>
    <td>ID号: </td>
    <td><%=edb.getId()%></td>
</tr>
<tr>
    <td>姓名: </td>
    <td><%=edb.getName()%></td>
</tr>
<tr>
    <td>地址: </td>
    <td><%=edb.getAddress()%></td>
</tr>
<tr>
    <td>年龄: </td>
    <td><%=edb.getAge()%></td>
</tr>
</table>

<%
    edb = (EmployeeDataBean)session.getAttribute("employee");
%>
```

通过session取到的:

```
<br>
<table border="1" cellpadding="1" cellspacing="3">
<tr>
    <td>ID号: </td>
    <td><%=edb.getId()%></td>
</tr>
<tr>
    <td>姓名: </td>
    <td><%=edb.getName()%></td>
</tr>
<tr>
    <td>地址: </td>
    <td><%=edb.getAddress()%></td>
</tr>
<tr>
    <td>年龄: </td>
    <td><%=edb.getAge()%></td>
</tr>
```

```

</table>
<br>
<hr width="100%">
<a href="<%={contextPath%}/delete/DeleteEmployeeServlet">删除“employee”
对象</a>
</body>
</html>

```



案例3 Servlet和Servlet的通信

案例运行效果与操作

Servlet和Servlet之间的通信主要是通过Session对象来实现的，本例与案例2的联系非常紧密。

本案例通过两个Session实现了DeleteEmployeeServlet和AddEmployeeServlet之间的通信。启动Tomcat，在IE的地址栏中录入http://localhost/servlet-study/sjsp/index，就会看到图8-5所示的画面。

单击“提交”按钮，运行结果如图8-6所示。说明记录添加成功，即Servlet和JSP通信成功。

The screenshot shows a web browser window with the URL http://localhost/servlet-study/sjsp/index. Below the address bar is a form for entering employee information:

ID号:	123
姓名:	ZhangSan
地址:	City of QingDao
年龄:	23
<input type="button" value="提交"/> <input type="button" value="重填"/>	

图8-5 运行界面

The screenshot shows the same browser window after the form has been submitted. It displays three sets of data corresponding to different ways of retrieving data:

- 第一种方式得到数据:**

ID号:	123
姓名:	ZhangSan
地址:	City of QingDao
年龄:	23
- 通过request取到的:**

ID号:	123
姓名:	ZhangSan
地址:	City of QingDao
年龄:	23
- 通过session取到的:**

ID号:	123
姓名:	ZhangSan
地址:	City of QingDao
年龄:	23

At the bottom, there is a link: [删除“employee”对象](#).

图8-6 运行结果

图8-6的最后一行是“删除employee对象”，单击它之后的结果如图8-7所示。

The screenshot shows the browser again with the URL http://localhost/servlet-study/delete/DeleteEmployeeServlet. The page displays the message: "Servlet之间通信成功！！！ “employee” 对象被删除。"

图8-7

最能证明Servlet是否通信成功的是Tomcat的logs目录下的stdout.log日志，如图8-8所示：

```

2012-05-22 10:11:20 30 40 50
151 request.getPathInfo()=/AddEmployeeServlet
152 request.getPathInfo()
153 ServletPath-/delete
154 删掉"employee"对象开始...
155 Euro
156 adf
157 adf
158 adf
159 "employee"对象已经被删除
160

```

图8-8 日志文件

通过log日志我们可以准确地判断出Servlet和Servlet之间通信成功。

制作要点

利用javax.servlet.http.HttpSession描述Session

步骤详解

1. Session对象的使用。

Session，中文经常翻译为会话，其本义是指有始有终的一系列动作/消息，比如打电话时从拿起电话拨号到挂断电话这中间的一系列过程，可以称之为一个Session。有时我们可以看到这样的话“在一个浏览器会话期间……”这里的会话一词用的就是其本义，是指从一个浏览器窗口打开到关闭的过程。

然而当Session一词与网络协议相关联时，它又往往隐含了“面向连接”和/或“保持状态”这样两个含义。“面向连接”指的是通信双方在通信之前要先建立一个通信的渠道，比如打电话，直到对方接了电话通信才能开始，与此相对的是写信，在你把信发出去的时候并不能确认对方的地址是否正确，通信渠道不一定能建立，但对发信人来说，通信已经开始了。“保持状态”则是指通信的一方能够把一系列的消息关联起来，使得消息之间可以互相依赖，比如一个服务员能够认出再次光临的老顾客并且记得上次这个顾客还欠店里一块钱。

而到了Web服务器蓬勃发展的时代，Session在Web开发语境下的语义又有了新的扩展，它的含义是指一类用来在客户端与服务器之间保持状态的解决方案。有时候Session也用来指这种解决方案的存储结构，如“把xxx保存在Session里”。由于各种用于Web开发的语言在一定程度上都提供了对这种解决方案的支持，所以在某种特定语言的语境下，Session也被用来指代该语言的解决方案，比如经常把Java里提供的javax.servlet.http.HttpSession简称为session。

鉴于这种混乱已不可改变，本文中session一词的运用也会根据上下文有不同的含义，请大家注意分辨。

2. 通过Session来实现两个Servlet之间的通信。

```

Object o = session.getAttribute("employee");
EmployeeDataBean edb = (EmployeeDataBean)o;
session.removeAttribute("employee");

```

程序源代码与解释

DeleteEmployeeServlet.java实现了和AddEmployeeServlet.java之间的通信。

```

//chp8

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
public class DeleteEmployeeServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html;charset=gb2312");
        String servletPath = request.getServletPath();
        System.out.println("ServletPath=" + servletPath);
        String pathInfo = (request.getPathInfo()).trim();
        HttpSession session = request.getSession();
        //取得Servlet请求的路径信息
        if (pathInfo.equals("/DeleteEmployeeServlet")) {
            //Object o = request.getAttribute("employee");
            //在这里可以通过Session来实现两个Servlet之间的通信
            Object o = session.getAttribute("employee");
            EmployeeDataBean edb = (EmployeeDataBean)o;
            if (o == null) {
                System.out.println("session.getAttribute(\"employee\")=null");
                request.getRequestDispatcher("/test/sjsp/deleteFailure.jsp")
                    .forward(request, response);
            } else {
                System.out.println("删除 \"employee\" 对象开始...");
                System.out.println(edb.getId());
                System.out.println(edb.getName());
                System.out.println(edb.getAddress());
                System.out.println(edb.getAge());
                session.removeAttribute("employee");
            }
        }
    }
}

```

```

        System.out.println("“employee”对象已经被删除");
        request.getRequestDispatcher("/test/sjsp/deleteEmployee.jsp")
            .forward(request, response);
    }
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    doGet(request, response);
}
}

```



案例4 Servlet动态生成图像

案例运行效果与操作

启动Tomcat，在IE地址栏中输入http://localhost/servlet-study/download.jsp，运行界面如图8-9所示。

Hello World!
Hello World!

[下载pdf文档 通过Servlet生成图片](#)

图8-9 运行界面

单击链接“通过Servlet生成图片”，结果如图8-10所示。



图8-10 运行结果

制作要点

JPEGImageEncoder类的使用

步骤讲解

1. 获取HTTP请求。

```
response.setContentType("image/jpeg");
```

注意，这里的MIME类型：

```
ServletOutputStream out = response.getOutputStream();
```

2. 构造缓冲图像。

绘制图像，但都是黑色的，需要填充颜色：

```
ServletOutputStream out = response.getOutputStream();
BufferedImage image = new BufferedImage(750,30,BufferedImage.TYPE_INT_RGB);
```

3. JPEGImageEncoder类的使用。

非标准类包com.sun.image.codec.jpeg中提供JPEGImageEncoder类编码图像，返回response中的图像。

```
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
encoder.encode(image);
```

源代码与解析**Download.jsp**

```
<%@ page contentType="text/html;charset=gb2312" %>
<html>
<head>
    <title>Servlet生成图片</title>
    <meta http-equiv="content-type" content="text/html; charset=gb2312">
</head>
<body>
<%
    for(int i=0;i<10;i++)
        out.println("<br>Hello World!");
    String contextPath = request.getContextPath();
%>
<hr width="100%">
<a href="<%=contextPath%>/servlet/OpenNonHtml">下载pdf文档</a>
```

```
<a href="<%={contextPath%}/servlet/JPEGServlet">通过Servlet生成图片</a>
</body>
</html>
```

JPEGServlet.java用来生成JPEG图片。

```
/*
 *
 * 利用Servlet生成动态图像
 * JPEGServlet.java
 */

import java.awt.Color;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;
public class JPEGServlet extends HttpServlet{

    /**
     * 处理Http Get request
     */
    public void doGet(HttpServletRequest request,HttpServletResponse response)
            throws ServletException,IOException{
        response.setContentType("image/jpeg");
        //注意这里的MIME 类型
        ServletOutputStream out = response.getOutputStream();
        //构造一个缓冲图像
        BufferedImage image = new BufferedImage(750,30,BufferedImage.TYPE_INT_RGB);
        int per = 0;
        per = Integer.parseInt((String) getInitParameter("size"));

        Graphics graphics = image.getGraphics();
        graphics.setColor(Color.green);
        graphics.fillRect(0,0,750,30);//first draw a rectangle
        graphics.setColor(Color.YELLOW);//change color
        graphics.fillRect(0,0,750*per/100,30);
        JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
```

```

        encoder.encode(image);
        out.close();
    }
    /**
     * 处理Http Post request和doGet一样
     */
    public void doPost(HttpServletRequest request,HttpServletResponse response)
            throws IOException,ServletException{
        doGet(request,response);
    }
}

```



案例5 用Servlet连接数据库

案例运行效果与操作

本案例通过Servlet实现MySQL数据库内容的查询，程序运行后，界面如图8-11所示。



图8-11 提交界面

输入用户名后，单击“提交”，返回查询的结果，如图8-12所示。



图8-12 结果返回界面

制作要点

1. GetConnect()实现数据库的连接
2. HttpServlet的用法

步骤解析

Servlet API所定义的Servlet生命周期类如下，分别对应本例中的相关操作：

1. 创建并初始化Servlet（init()方法）

在init()中，加载数据库驱动，在程序中调用Class.forName()方法：

```
Class.forName("org.gjt.mm.mysql.Driver").newInstance();
```

2. 响应客户程序的服务请求（service()方法）

在doPost()，调用getConnection()，连接数据库，再执行数据库操作。

利用HttpServletRequest和HttpServletResponse对象获得servlet的输入流。

3 Servlet终止运行，释放所有资源（destroy()方法）

在doPost()，调用close()关闭所有连接，释放所有资源。

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.net.*;
import java.sql.*;
import java.text.DateFormat;
import java.util.Locale;

public class DBConnectSample extends HttpServlet {
    public void init() throws ServletException {
        try {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        }
        //创建实例
        catch (Exception e) {
        }
    }
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ServletOutputStream out = res.getOutputStream();
        //用户名
        String userName = req.getParameter("username");
    }
}
```

```

//SQL语句
String sqlStr = "select * from person where username= " + userName + "";
Connection conn = null;
Statement stmt = null;
ResultSet rs = null;
try {
    //数据库连接
    conn = getConnection();
    stmt = conn.createStatement();
    //执行查询操作
    rs = stmt.executeQuery(sqlStr);
    res.setContentType("text/html");
    //输出用户相关信息
    if (rs.next()) {
        out.println("<TABLE WIDTH=\"50%\" ALIGN=\"CENTER\" BORDER=\"1\"");
        out.println("<TR><TD COLSPAN=\"2\">User " + userName + " Info: </TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Name:</TD><TD>" + rs.getString("TRUENAME") + "</TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Birthday:</TD><TD>" + rs.getString("BIRTHDAY") +
        "</TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Identity No:</TD><TD>" + rs.getString("IDENTITY_NO") +
        "</TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Office Telephone:</TD><TD>" + rs.getString
        ("OFFICEPHONE") + "</TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Home Telephone:</TD><TD>" + rs.getString("HOMEPHONE") +
        "</TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Mobilephone:</TD><TD>" + rs.getString("MOBILEPHONE") +
        "</TD>");
        out.println("</TR>");
        out.println("<TR ALIGN=\"LEFT\"");
        out.println("<TD>Email:</TD><TD>" + rs.getString("EMAIL") + "</TD>");
    }
}

```

```

        out.println("</TR>");
        out.println("<TR ALIGN='LEFT'>");
        out.println("<TD>Address:</TD><TD>" + rs.getString("ADDRESS") + "</TD>");
        out.println("</TR>");
        out.println("</TABLE>");
    }
}

catch (SQLException se) {
    System.out.println("查询数据库记录时出现异常。");
}
finally {
    try {
        if (conn != null)
            conn.close();
        if (stmt != null)
            stmt.close();
        if (rs != null)
            rs.close();
    } catch (Exception e) {
    }//如果创建了连接，则关闭所有连接
}
}

private Connection getConnection() {
    Connection conn = null;
    String dbUrl = "jdbc:mysql://10.21.88.88/mysql3235";
    //数据库的JDBC URL
    String dbUser = "root";
    //数据库账号
    String dbPassword = "yyoa2";
    //数据密码
    try {
        conn = DriverManager.getConnection(dbUrl, dbUser, dbPassword);
        //数据库连接
    }
    catch (SQLException sqlexception) {
        System.out.println("数据库连接异常。");
    }
    return conn;
    //返回一个数据库连接
}
}
}

```



案例6 用Servlet实现页面注册和登录

案例运行效果与操作

本案例通过Servlet实现页面注册和登录。本例在Tomcat 4.1, JDK 1.4下调试通过。启动Tomcat后，在IE地址栏输入相应地址，如图8-13所示。



图8-13 登录注册页面

单击“新用户注册”，并填写具体内容，如图8-14所示。

单击“确定”后，再回到登录页面，输入用户名和密码，用户信息显示如图8-15所示。

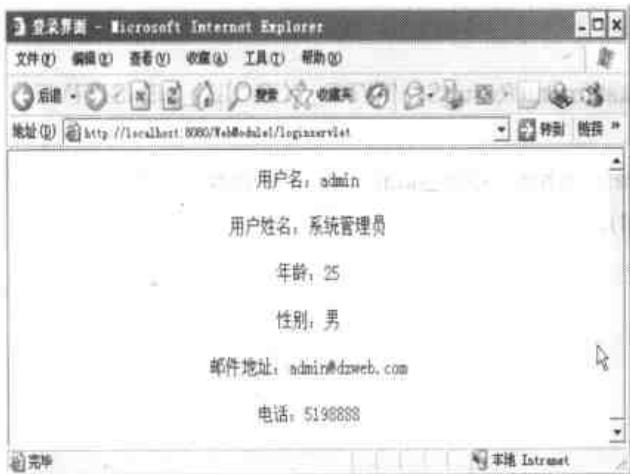


图8-14 注册界面



图8-15 用户信息显示

制作要点

1. HttpServlet中的doGet()和doPost()方法的应用
2. destroy()方法在Servlet中的用法
3. 直接通过URL调用Servlet

步骤详解

1. destroy()方法。

destroy()方法仅执行一次，即在服务器停止且卸载Servlet时执行该方法。当服务器卸载Servlet时，将在所有service()方法调用完成后，或在指定的时间间隔过后调用destroy()方法。一个Servlet在运行service()方法时可能会产生其他的线程，因此在调用destroy()方法时，必须确认这些线程已终止或完成。

2. HTTP中用方法控制servlet动作。

```
<form name="form1" method="post" action="loginservlet">
```

如果HTTP请求方法为GET，则默认情况下就调用doGet()。当一个客户通过HTML表单发出一个HTTP POST请求时，doPost()方法被调用。与POST请求相关的参数作为一个单独的HTTP请求从浏览器发送到服务器。当需要修改服务器端的数据时，应该使用doPost()方法：

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    doPost(request, response);
}
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {}
```

3. 数据库数据的返回。

```
Statement statement = connection.createStatement	ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
String sql = "select * from userbase where user_name = '" + name + "'";
ResultSet rs = statement.executeQuery(sql);
```

程序源代码与解析

LOGIN.htm

```
LOGIN.htm
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="post" action="loginservlet">
<p>&nbsp;</p>
```

```

<table width="47%" border="0" align="center">
  <tr>
    <td>用户名: </td>
    <td>
      <input type="text" name="username">
    </td>
  </tr>
  <tr>
    <td>密码: </td>
    <td>
      <input type="password" name="password">
    </td>
  </tr>
</table>
<p align="center">
  <input type="submit" name="Submit" value="确定">
  <input type="reset" name="Reset" value="重写">
</p>
</form>

<p align="center"><a href="register.htm">新用户注册</a></p>
</body>
</html>

```

Register.htm

```

<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="post" action="registerservlet">
  <div align="center">
    <table width="75%" border="0">
      <tr>
        <td>用户名: </td>
        <td>
          <input type="text" name="name">
        </td>
      </tr>
      <tr>
        <td>密码: </td>
        <td>

```

```
        <input type="password" name="password">
    </td>
</tr>
<tr>
    <td>密码确认: </td>
    <td>
        <input type="password" name="pass_confirm">
    </td>
</tr>
<tr>
    <td>用户姓名: </td>
    <td>
        <input type="text" name="truelname">
    </td>
</tr>
<tr>
    <td>用户年龄: </td>
    <td>
        <input type="text" name="age">
    </td>
</tr>
<tr>
    <td>用户性别: </td>
    <td>
        <select name="sex">
            <option value="男">男</option>
            <option value="女">女</option>
        </select>
    </td>
</tr>
<tr>
    <td>用户E-Mail:</td>
    <td>
        <input type="text" name="mail">
    </td>
</tr>
<tr>
    <td>用户电话: </td>
    <td>
        <input type="text" name="tel">
    </td>
</tr>
```

```

</table>
<input type="submit" name="Submit" value="确定">
<input type="reset" name="Reset" value="重写">
</div>
</form>
</body>
</html>

Login.java
package testserv;

/**
 * 用servlet登录程序
 */
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class loginServlet extends HttpServlet {
    static final private String CONTENT_TYPE = "text/html; charset=GBK";
    Connection connection;                                //创建连接对象
    Vector userinfo;                                     //定义向量存入个人注册信息
    Vector userbaseinfo;                                //定义向量存入用户名及密码

    public void init() throws ServletException {
        connection = null;
        userinfo = new Vector();
        userbaseinfo = new Vector();
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String name = request.getParameter("username");           //获得登录用户名
        String password = request.getParameter("password");       //获得登录密码
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>登录界面</title></head>");
        out.println("<body>");
    }
}

```

```

boolean loginning= false;
if ((loginning=login(name, password))==true)           //判断数据库中是否有用户名
{
    boolean right_pass = validate(password);           //检测用户密码是否正确
    if (right_pass)
    {
        getUserInfo(name, password);      //调用个人注册信息，分别赋值给字符串变量
        String truename = String.valueOf(userinfo.elementAt(0));
        String age = String.valueOf(userinfo.elementAt(1));
        String sex = String.valueOf(userinfo.elementAt(2));
        String mail = String.valueOf(userinfo.elementAt(3));
        String tel = String.valueOf(userinfo.elementAt(4));
        //显示个人注册信息
        out.println("<p align=center>用户名: " + name + "</p>");
        out.println("<p align=center>用户姓名: " + truename + "</p>");
        out.println("<p align=center>年龄: " + age + "</p>");
        out.println("<p align=center>性别: " + sex + "</p>");
        out.println("<p align=center>邮件地址: " + mail + "</p>");
        out.println("<p align=center>电话: " + tel + "</p>");
    }
    else          //处理密码不正确情况
    {
        out.println("密码不正确！<br>");
        out.println("<p align=center><a href='login.htm'>重新登录</a></p>");
    }
}
else          // 处理用户名不存在情况
{
    out.println("用户名不存在！<br>");
    out.println("<p align=center><a href='login.htm'>重新登录</a></p>");
}
out.println("</body></html>");
}

public void destroy() {
}

public boolean login(String name, String password)
{
    boolean loginning= false;
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //实例化JDBC-ODBC桥的驱动
        String url = "jdbc:odbc:login";                 //设置连接字符串
    }
}

```

```

connection = DriverManager.getConnection(url); //连接数据库
Statement statement = connection.createStatement (ResultSet.TYPE_SCROLL_
SENSITIVE,ResultSet.CONCUR_READ_ONLY);
String sql = "select * from userbase where user_name = '" + name + "'";
ResultSet resultset = statement.executeQuery(sql);
int recordCount = 0;
while(resultset.next())
{
    recordCount++;
}
if (recordCount>0) //判断数据库表userbase中是否有用户信息
{
    userbaseinfo.clear();
    resultset.first();
    userbaseinfo.addElement(resultset.getString("user_name"));
//将个人信息存入向量对象userbase_Vec中
    userbaseinfo.addElement(resultset.getString("user_password"));
    loginning= true;
}
catch(Exception ex)
{
    ex.printStackTrace();
}//捕捉异常
return loginning;
}

public void getUserInfo(String name, String password)
{
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //实例化JDBC-ODBC桥的驱动
    String url = "jdbc:odbc:login"; //设置连接字符串
    connection = DriverManager.getConnection(url); //连接数据库
//创建Statement接口对象
    Statement statement = connection.createStatement();
    String sql = "select * from userinfo where user_name = '" + name + "' and user_
password = '" + password + "'";
    ResultSet resultset = statement.executeQuery(sql);
    resultset.next();
    userinfo.clear();
//将个人注册信息存入向量用户信息中
    userinfo.addElement(resultset.getString("user_truename"));
    userinfo.addElement(resultset.getString("user_age"));
}

```

```

        userinfo.addElement(resultset.getString("user_sex"));
        userinfo.addElement(resultset.getString("user_mail"));
        userinfo.addElement(resultset.getString("user_tel"));
    }
    catch(Exception ex) //捕捉异常
    {
        ex.printStackTrace();
    }
}

public boolean validate(String password) //定义验证密码函数
{
    boolean pass = false;
    if (password.equals(String.valueOf(userbaseinfo.elementAt(1))))
    {
        pass = true;
    }
    return pass;
}
}

register.java
package testserv;
/**
 * 用servlet登录注册
 */
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class registerServlet extends HttpServlet {
    static final private String CONTENT_TYPE = "text/html; charset=GBK";
    Connection connection;
    //定义Connection接口对象connection
    public void init() throws ServletException {
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

```

```

//获得个人注册信息并存入对应的字符串变量中
String name = new String(request.getParameter("name").getBytes("8859_1"));
String password = new String(request.getParameter("password").getBytes("8859_1"));
String pass_confirm = new String(request.getParameter("pass_confirm").getBytes("8859_1"));
String truename = new String(request.getParameter("truename").getBytes("8859_1"));
String age_str = new String(request.getParameter("age").getBytes("8859_1"));
int age = Integer.parseInt(age_str);
String sex = new String(request.getParameter("sex").getBytes("8859_1"));
String mail = new String(request.getParameter("mail").getBytes("8859_1"));
String tel = new String(request.getParameter("tel").getBytes("8859_1"));
response.setContentType(CONTENT_TYPE);
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>用户注册</title></head>");
out.println("<body>");
boolean existed = checkUser(name);           //检测数据库中是否存在重名
if (existed)                                //有重名处理情况
{
    out.println("<p align=center>对不起，此用户名已经存在！</p>");
    out.println("<p align=center><a href='register.htm'>返回</a></p>"); //返回注册页
}
else                                         //不存在重名处理情况
{
    boolean identical = checkPass(password, pass_confirm);
    //检测密码和密码确认是否一致
    if (!identical)                          //处理密码不一致情况
    {
        out.println("<p align=center>密码不一致！</p>");
        out.println("<p align=center><a href='register.htm'>返回</a></p>"); //返回注册页
    }
    else                                     //处理密码一致情况
    {
        String sql_base = "insert into userbase(user_name, user_password) values (" +
name + ", " + password + ")";
        String sql_info = "insert into userinfo(user_name, user_password, user_truename,
user_age, user_sex, user_mail, user_tel) values ";
        sql_info+= "(" + name + ", ";
        sql_info+= "" + password + ", ";
        sql_info+= "" + truename + ", ";
        sql_info+= "" + age + ", ";
        sql_info+= "" + sex + ", ";
        sql_info+= "" + mail + ", ";
    }
}

```

```

        sql_info += "" + tel + ")");
        boolean success = addRecord(sql_base, sql_info);
        //向表userbase和userinfo中添加个人注册信息
        if (success)
        {
            out.println("注册成功！<br>");
        }
        else
        {
            out.println("注册失败，请稍后再试！<br>");
        }
        out.println("<p align=center><a href='login.htm'>返回主页</a></p>");
    }
}
out.println("</body></html>");
}

public void destroy() {

}

public boolean checkUser(String name)          //检测是否存在用户名函数
{
    boolean existed = false;
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");      //实例化JDBC-ODBC桥的驱动
        String url = "jdbc:odbc:login";                      //设置连接字符串
        connection = DriverManager.getConnection(url);         //连接数据库
        //创建Statement接口对象
        Statement statement = connection.createStatement();
        String sql = "select * from userbase where user_name = " + name + "";
        ResultSet rs = statement.executeQuery(sql);
        int recordCount = 0;
        while(rs.next())
        {
            recordCount++;
        }
        if (recordCount>0)
        {
            existed = true;
        }
    }
    catch(Exception ex)                                //处理异常
    {
        ex.printStackTrace();
    }
}

```

```

        }
        return existed;
    }
}

//检测密码和密码确认是否一致函数
public boolean checkPass(String password, String pass_confirm)
{
    boolean identical = true;
    if (!password.equals(pass_confirm))
    {
        identical = false;
    }
    return identical;
}

public boolean addRecord(String sql_base, String sql_info) //添加记录函数
{
    boolean made = false;
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //实例化JDBC-ODBC桥的驱动
        String url = "jdbc:odbc:login"; //设置连接字符串
        connection = DriverManager.getConnection(url); //连接数据库
        //创建Statement接口对象
        Statement statement = connection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
        statement.executeUpdate(sql_base);
        //向userbase表中插入数据
        statement.close();
        //关闭statement对象
        statement = connection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
        statement.executeUpdate(sql_info);
        //向userinfo表中插入数据
        statement.close();
        //关闭statement对象
        made = true;
    }
    catch(Exception ex)
    {
        ex.printStackTrace();
    } //异常处理
    return made;
}
}

```



案例7 运用Servlet实现BBS功能

案例运行效果与操作

本案例使用Servlet实现一个简单的留言板功能，包括留言、保存和查看留言。启动Tomcat后，输入留言板对应的Servlet地址，界面如图8-16所示。



图8-16 编辑留言

输入留言后，单击“保存留言”，界面如图8-17所示。

单击“查看留言”，则显示刚才的留言信息，如图8-18所示。



图8-17 保存留言



图8-18 查看留言

制作要点

Servlet的service()方法

步骤详解

1. 总的流程。

本例共应用三个Servlet类，分别是leavewordshome（留言）、leavewords（保存留言）和displaywords（查看留言）。

2. Service()方法。

Servlet的主要功能是接受从浏览器发送过来的HTTP请求（request），并返回HTTP响应（response）。这个工作是在service()方法中完成的。service()方法包括从request对象获得客户端数据和向response对象创建输出service()方法，默认的服务功能是调用与HTTP请求的方法相应的do功能。

```
protected void service(HttpServletRequest request, HttpServletResponse response)
```

3. 对应的web.xml配置如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc./DTD Web Application 2.3/
/EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <display-name>WebApp</display-name>
    <servlet>
        <servlet-name>showinfo</servlet-name>
        <servlet-class>simplebbs.ShowInfo</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>inputinfo</servlet-name>
        <servlet-class>simplebbs.InputInfo</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>inputinfohome</servlet-name>
        <servlet-class>simplebbs.InputInfoHome</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>showinfo</servlet-name>
        <url-pattern>/showinfo</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>inputinfo</servlet-name>
        <url-pattern>/inputinfo</url-pattern>
```

```

</servlet-mapping>
<servlet-mapping>
    <servlet-name>inputinfohome</servlet-name>
    <url-pattern>/inputinfohome</url-pattern>
</servlet-mapping>
</web-app>

```

4. servlet用方法控制页面跳转。

```
<form name=\\"showinfo\\" action=\\"showinfo\\" method=\\"POST\\\">
```

留言板代码与解释

留言板InputInfoHome.java，输入留言：

```

/*
 *输入信息
 */
package simplebbs;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class InputInfoHome extends HttpServlet {
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        java.io.PrintWriter out=new java.io.PrintWriter(response.getOutputStream());
        out.print("<html>");
        out.print("<head><title>Servlet留言板</title></head>");
        out.print("<body>");
        out.print("<h3>Servlet留言板</h3>");
        out.print("<form name=\\"inputform\\" action=\\"inputinfo\\" method=\\"POST\\\">");
        out.print("<br>");
        out.print("<td align=\\"right\\"><b>姓名：</b></td>");
        out.print("<td align=\\"left\\">");
        out.print("<input type=\\"text\\" name=\\"cName\\" SIZE=\\"20\\" value=\\"\\\" required=\\"\\\">");
        out.print("</td><br>");
        out.print("<td align=\\"right\\"><B>Email地址：</B> </td>");
        out.print("<td align=\\"left\\">");
        out.print("<input type=\\"text\\" name=\\"cEmail\\" SIZE=\\"20\\" value=\\"\\\">");
        out.print("</td><br>");
        out.print("</tr>");
        out.print("<tr><td align=\\"right\\">");
        out.print("<b>主题：</b></td><td colspan=\\"3\\">");

```

```

        out.print("<input type=\"text\" name=\"cTopic\" SIZE=\"40\" value=\"\"></td><br></tr>");
        out.print("<tr><td colspan=\"4\" align=\"center\">");
        out.print("<textarea name=\"cWords\" rows=\"10\" cols=\"80%\"></textarea>");
        out.print("</td><br></tr>");
        out.print("<input type=\"submit\" name=\"action\" value=\"保存留言\">");
        out.print("</form>");
        out.print("</body></html>");
        out.flush();
    }
}

InputInfo.java, 保存留言
/*
 * 输入信息保存至数据库
 */
package simplebbs;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
public class InputInfo extends HttpServlet {
    public void init() throws ServletException {}
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        java.io.PrintWriter out=new java.io.PrintWriter(response.getOutputStream());
        out.print("<html>");
        out.print("<head><title>保存留言</title></head>");
        out.print("<body>");
        out.print("<center><h3>谢谢留言</h3></center>");
        out.print("<hr>");
        out.print("<h4>留言信息</h4>");
        String getDate=new java.util.Date().toString();
        String getName=request.getParameter("cName");
        String getEmail=request.getParameter("cEmail");
        String getWords=request.getParameter("cWords");
        String getTopic=request.getParameter("cTopic");
        //检查客户的留言信息是否完整
        if(getName.length()<1){           out.print("请输入姓名");
        }else{
            if(getEmail.length()<3){
                out.print("请输入正确的Email地址");
            }
            else{

```

```

        if(getTopic.length()<1){
            out.print("请输入主题");
        }
        else{
            if(getWords.length()<1){
                out.print("没有留言");
            }
            else{ //客户输入完整的信息则响应 在HTML中让客户看到自己的留言
                out.print("<table>");
                out.print("<tr><td align=\"right\">姓名: </td><td>");
                out.print(getName);
                out.print("</td></tr>");
                out.print("<tr><td align=\"right\">Email地址: </td><td>");
                out.print(getEmail);
                out.print("</td></tr>");
                out.print("<tr><td align=\"right\">主题: </td><td>");
                out.print(getTopic);
                out.print("</td></tr>");
                out.print("<tr><td valign=\"top\" align=\"right\">留言: </td><td>");
                out.print(getWords);
                out.print("</td></tr>");
                out.print("<tr><td align=\"right\">日期: </td><td>");
                out.print(getDate);
                out.print("</td></tr></table>");
                try{
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                    Connection cn=DriverManager.getConnection("jdbc:odbc:info");
                    String str="INSERT INTO info VALUES(";
                    String data=getName+",""+getDate+",""+getEmail+",""+getTopic+",""+getWords+"");
                    Statement st=cn.createStatement();
                    st.executeUpdate(str+data);
                    st.close();
                    cn.close();
                    out.print("<center><h3>成功保存留言</h3></center>");
                }
                catch(Exception e){
                    out.print(e.getMessage());
                } }}}
out.print("<hr>"); //输出响应HTML文件的尾部信息
out.print("<form name=\"showinfo\" action=\"showinfo\" method=\"POST\">");
out.print("<input type=\"submit\" name=\"action\" value=\"查看留言\">");
out.print("</form>");
```

```

        out.print("</body></html>");
        out.flush();
    }
}

显示留言, ShouInfo.java
/*
*保存在数据库中的信息回显
*/
package simplebbs;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.Vector;

public class ShowInfo extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        java.io.PrintWriter out=new java.io.PrintWriter(response.getOutputStream());
        out.print("<html>");
        //输出响应的HTML 文件头部信息
        out.print("<head><title>查看留言</title></head>");
        out.print("<body>");
        out.print("<h2>查看留言</h2>");
        out.print("<hr>");
        //创建数据库连接取得库中保存的留言信息
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection cn=DriverManager.getConnection("jdbc:odbc:info");
            //数据驱动
            String str="SELECT * FROM info";
            Statement statement=cn.createStatement();
            ResultSet resultSet=statement.executeQuery(str);
            String name,date,email,topic,words;
            while(resultSet.next()){
                name=resultSet.getString("name");
                date=resultSet.getString("wdate");
                email=resultSet.getString("email");
                topic=resultSet.getString("topic");
                words=resultSet.getString("words");
                //输出相关信息
                out.print("<table>");

```

```

        out.print("<tr><td align=\"right\">");
        out.print("<b>主题：</b></td><td colspan=\"3\"><b>");
        out.print(topic);
        out.print("</b></td></tr>");
        out.print("<tr><td align=\"right\">姓名：</td><td colspan=\"3\">");
        out.print(name);
        out.print("</td></tr>");
        out.print("<tr><td align=\"right\">Email地址：</td><td colspan=\"3\">");
        out.print(email);
        out.print("</td></tr>");
        out.print("<tr><td valign=\"top\" align=\"right\">留言：</td><td colspan=\"3\">");
        out.print(words);
        out.print("</td></tr>");
        out.print("<tr><td align=\"right\">日期：</td><td>");
        out.print(date);
        out.print("</td></tr>");
    }
    statement.close();
    cn.close();
    //关闭
}
catch(Exception e){
    out.print(e.getMessage());
}
out.print("<form name=\"inputinfohome\" action=\"inputinfohome\" method=\\\"POST\\\"");
out.print("<input type=\"submit\" name=\"action\" value=\"留言主页\\\"");
out.print("</form>");
out.print("</body></html>");
out.flush();
}
}
}

```



案例8 偷听Web服务器信息

案例运行效果与操作

本案例利用Servlet探询Web服务器的信息，返回客户发送给服务器的请求行和头部信息，以及一些可访问的HTTP信息等，如服务器名称、地址、端口等。启动Tomcat，在地址栏内输入http://localhost:8080/webapp/snoopservlet，返回当前Web服务器信息，如图8-19所示。

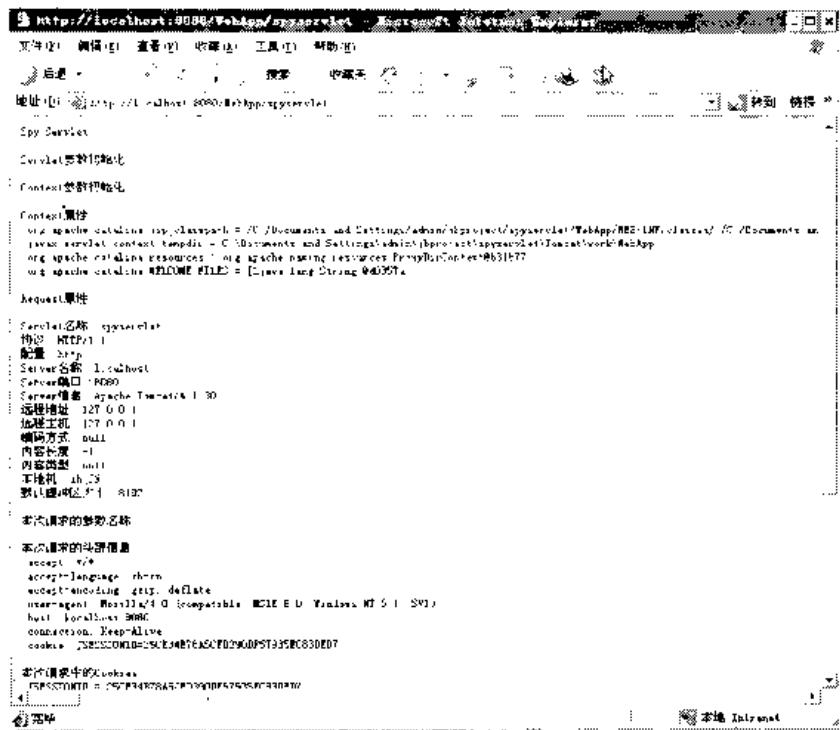


图8-19 Servlet运行界面

制作要点

1. HttpServlet类的使用
2. Javax.servlet类包的使用

步骤详解

1. HttpServletRequest接口用来处理一个对Servlet的HTTP格式的请求信息。

客户请求的http信息由HttpServletRequest类提供的方法返回，如：

```
request.getProtocol();
request.getScheme();
request.getServerName();
request.getServerPort();
context.getServerInfo();
request.getRemoteAddr();
request.getRemoteHost();
```

2. Sevlet中的中文显示。

Servlet中文经常显示乱码，需要指定字符集。

```
response.setContentType("text/plain;charset=gb2312");
```

3. 对应的web.xml配置。

```
<web-app>
    <display-name>webapp</display-name>
    <servlet>
        <servlet-name>spyservlet</servlet-name>
        <servlet-class> spyservlet.SpyServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>spyservlet</servlet-name>
        <url-pattern>/spyservlet</url-pattern>
    </servlet-mapping>
</web-app>
```

程序源代码与解释

```
//chp8
/*
 *利用servlet返回请求信息
 */
package httpserv;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;
import javax.servlet.*;
import javax.servlet.http.*;

public class SpyServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        PrintWriter out = response.getWriter();
        response.setContentType("text/plain;charset=gb2312");
        out.println("侦听Servlet");
        out.println();
        out.println("Servlet参数初始化:");
        Enumeration e = getInitParameterNames();
        while (e.hasMoreElements()) {
            String key = (String)e.nextElement();
            String value = getInitParameter(key);
            out.println(" " + key + " = " + value);
        }
        out.println();
    }
}
```

```
out.println("Context参数初始化:");
ServletContext context = getServletContext();
Enumeration num1 = context.getInitParameterNames();
while (num1.hasMoreElements()) {
    String key = (String)num1.nextElement();
    Object value = context.getInitParameter(key);
    out.println(" " + key + " = " + value);
}
out.println();

out.println("Context属性:");
num1 = context.getAttributeNames();
while (num1.hasMoreElements()) {
    String key = (String)num1.nextElement();
    Object value = context.getAttribute(key);
    out.println(" " + key + " = " + value);
}
out.println();

out.println("Request属性:");
e = request.getAttributeNames();
while (e.hasMoreElements()) {
    String key = (String)e.nextElement();
    Object value = request.getAttribute(key);
    out.println(" " + key + " = " + value);
}
out.println();

out.println("Servlet名称: " + getServletName());
out.println("协议: " + request.getProtocol());
out.println("配置: " + request.getScheme());
out.println("Server名称: " + request.getServerName());
out.println("Server端口: " + request.getServerPort());
out.println("Server信息: " + context.getServerInfo());
out.println("远程地址: " + request.getRemoteAddr());
out.println("远程主机: " + request.getRemoteHost());
out.println("编码方式: " + request.getCharacterEncoding());
out.println("内容长度: " + request.getContentLength());
out.println("内容类型: " + request.getContentType());
out.println("本地机: " + request.getLocale());
out.println("默认缓冲区大小: " + response.getBufferSize());
out.println();

out.println("本次请求的参数名称:");
e = request.getParameterNames();
```

```
while (e.hasMoreElements()) {
    String key = (String)e.nextElement();
    String[] values = request.getParameterValues(key);
    out.print(" " + key + " = ");
    for(int i = 0; i < values.length; i++) {
        out.print(values[i] + " ");
    }
    out.println();
}
out.println();
out.println("本次请求的头部信息:");
e = request.getHeaderNames();
while (e.hasMoreElements()) {
    String key = (String)e.nextElement();
    String value = request.getHeader(key);
    out.println(" " + key + ": " + value);
}
out.println();
out.println("本次请求中的Cookies:");
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for (int i = 0; i < cookies.length; i++) {
        Cookie cookie = cookies[i];
        out.println(" " + cookie.getName() + " = " + cookie.getValue());
    }
}
out.println();
out.println("Request Is Secure: " + request.isSecure());
out.println("Auth类型: " + request.getAuthType());
out.println("HTTP方法: " + request.getMethod());
out.println("远程用户: " + request.getRemoteUser());
out.println("请求URI: " + request.getRequestURI());
out.println("Context路径: " + request.getContextPath());
out.println("Servlet路径: " + request.getServletPath());
out.println("路径信息: " + request.getPathInfo());
out.println("路径转化: " + request.getPathTranslated());
out.println("查询串: " + request.getQueryString());

out.println();
HttpSession session = request.getSession();
out.println("请求会话Id: " +
request.getRequestedSessionId());
```

```
out.println("当前会话Id: " + session.getId());
out.println("会话创建时间: " + session.getCreationTime());
out.println("会话最后访问时间: " + session.getLastAccessedTime());
out.println("会话最大停止时间间隔: " + session.getMaxInactiveInterval());
out.println();
out.println("会话值: ");
Enumeration names = session.getAttributeNames();
while (names.hasMoreElements()) {
    String name = (String) names.nextElement();
    out.println(" " + name + " = " + session.getAttribute(name));
}
}
```

本 章 小 结

Servlet是一种独立于平台和协议的服务器端Java应用程序，可以生成动态的Web页面。和Applet对比，Servlet运行于服务器端，由Web服务器进行加载，所以需要一个单独的Web服务器（如Tomcat）作为容器，该Web服务器必须包含支持Servlet的Java虚拟机。

客户端程序可以是一个Web浏览器，或者是其他的可以连接上Internet的程序，它会访问Web服务器并发出请求。这个请求被运行在Web服务器上的Servlet引擎处理，并返回响应到Servlet。Servlet通过HTTP将这个响应转发到客户端。在功能上，Servlet与CGI、NSAPI有点类似，但是，与它们不同的是：Servlet具有平台无关性。

案例2中Servlet和JSP之间的通信实际上也可以体现出当前MVC思想。

第9章

Java与网络



本章内容

- 案例1 显示你的IP
- 案例2 用Socket进行客户与服务器通信
- 案例3 利用UDP Socket技术实现IP多点传送
- 案例4 利用Java API发送E-mail
- 案例5 从Mail Server删除一条消息
- 案例6 在java程序中实现FTP的功能
- 案例7 一个简单的聊天程序
- 案例8 代理服务器的实现
- 本章小结



案例1 显示你的IP

案例运行效果与操作

本案例讲解了如何通过Java API来显示本地机器的IP地址和网络上任何一台可以访问到的服务器的IP地址，运行界面如图9-1所示。

在图9-1中，可以看到两行信息：

LocalHost IP is: IBM-D6AFAB4A0FE/10.63.2.182

Server IP is :www.sina.com.cn/61.135.153.182

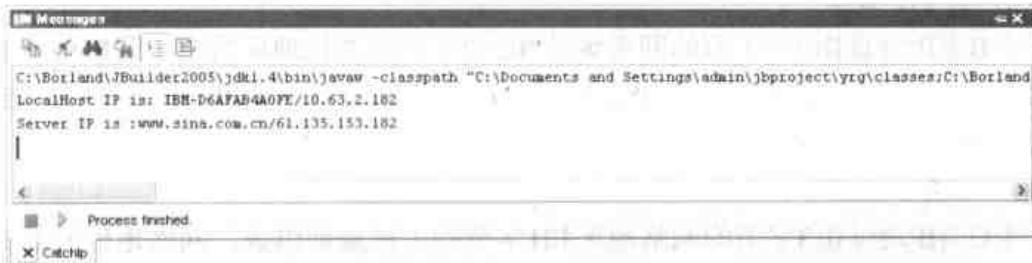


图9-1 运行界面

第一行显示的是本地机器在局域网中的IP地址，上面显示的是A类私有IP地址。另外还可以取到网络上任何一个服务器的IP地址。例如本例中的：www.sina.com.cn/61.135.153.182，读者可以扩展本案例，任意修改所要获取的网络服务器地址，或者另外编写一个输入框来得到需要的服务器，以获得其网络地址，如图9-2所示。

```
1. package Catchip;
2. import java.net.*;
3.
4. public class Catchip {
5.     public InetAddress getServerIP() {
6.         try {
7.             ServerIP = InetAddress.getByName("www.sina.com.cn");
8.             //取得 www.sina.com.cn 的 IP 地址
9.         } catch (UnknownHostException e) {
10.             System.out.println(e.getMessage());
11.         }
12.         return ServerIP;
13.     }
14. }
15. /*
16. * Catchip.java
17. */
18. 
```

图9-2 网络地址的获得

制作要点

1. 类java.net.InetAddress的用法
2. getLocalHost()方法的使用，它返回一个InetAddress对象
3. getAddress()方法的使用，它则返回一个长度为4的字节数组（IP地址为4字节）

步骤详解

1. 相关知识，IPV4的地址分类。

IP地址根据网络ID的不同分为5种类型：A类地址、B类地址、C类地址、D类地址和E类地址。

(1) A类IP地址

一个A类IP地址由1字节的网络地址和3字节主机地址组成，网络地址的最高位必须是“0”，地址范围从1.0.0.0到126.0.0.0。可用的A类网络有126个，每个网络能容纳1亿多个主机。

(2) B类IP地址

一个B类IP地址由2个字节的网络地址和2个字节的主机地址组成，网络地址的最高位必须是“10”，地址范围从128.0.0.0到191.255.255.255。可用的B类网络有16382个，每个网络能容纳6万多个主机。

(3) C类IP地址

一个C类IP地址由3字节的网络地址和1字节的主机地址组成，网络地址的最高位必须是“110”。范围从192.0.0.0到223.255.255.255。C类网络可达209万余个，每个网络能容纳254个主机。

(4) D类地址用于多点广播（Multicast）

D类IP地址第一个字节以“1110”开始，它是一个专门保留的地址。它并不指向特定的网络，目前这一类地址被用在多点广播（Multicast）中。多点广播地址用来一次寻址一组计算机，它标识共享同一协议的一组计算机。

(5) E类IP地址

以“11110”开始，为将来使用保留。

全零（“0.0.0.0”）地址对应于当前主机。全“1”的IP地址（“255.255.255.255”）是当前子网的广播地址。

在IP地址3种主要类型里，各保留了3个区域作为私有地址，其地址范围如下：

A类地址：10.0.0.0~10.255.255.255

B类地址：172.16.0.0~172.31.255.255

C类地址：192.168.0.0~192.168.255.255

2. 类java.net.InetAddress。

为了获得所在网络的IP地址，或者网络中其他主机的IP地址，Java提供了丰富的类库，可以用java.net软件包中的java.net.InetAddress类加以实现。这个类没有被public修饰的构造方法，但是可以通过4种static方法来产生它的实例，如表9-1所示。

表9-1 方法介绍

方法摘要	
static InetAddress	getByAddress(byte[] addr)
static InetAddress	getByAddress(String host, byte[] addr)
static InetAddress	getByName(String host)
static InetAddress	getLocalHost()

3. 取得本机地址。

```
LocalIP = InetAddress.getLocalHost();
```

4. 取得要访问的机器地址。

```
ServerIP = InetAddress.getByName("www.sina.com.cn");
```

程序源代码与解释

```

//chp 9
/**
 * 获取IP地址
 */

import java.io.IOException;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class CatchIp {
    private InetAddress LocalIP = null;
    private InetAddress ServerIP = null;
    public static void main(String args[]) {
        CatchIp mytest;
        mytest = new CatchIp();
        System.out.println("LocalHost IP is: " + mytest.catchLocalIP());
        System.out.println("Server IP is :" + mytest.catchServerIP());
    }
    //取得本机IP地址
    public InetAddress catchLocalIP() {
        try {
            LocalIP = InetAddress.getLocalHost();
        } catch (UnknownHostException e) {
        }
        return LocalIP;
    }
    //取得服务器网络地址
    public InetAddress catchServerIP() {
        try {
            ServerIP = InetAddress.getByName("www.sina.com.cn");
            //取得 www.sina.com.cn 的IP地址
        } catch (UnknownHostException e) {
            System.out.println(e.getMessage());
        }
        return ServerIP;
    }
}//end class CatchIp

```



案例2 用Socket进行客户与服务器通信

案例运行效果与操作

Socket（套接字）是现在网络上的两个程序通过一个双向的通信连接实现数据交换的通道，这个双向链路的一端称为一个Socket。Socket通常用来实现客户方和服务方的连接，一个程序将一段信息写入Socket中，该Socket将这段信息发送给另外一个Socket，使这段信息能传送到其他程序中。下面首先介绍如何用Socket进行Client/Server程序设计。

首先运行服务器端程序，在DOS窗口对应的目录下面输入“java TalkServer”，然后再打开一个新的DOS窗口，输入“java TalkClient”，如图9-3所示。

```

C:\WINDOWS\system32>cmd.exe
D:\80JavaProgram\changxianxiang>java TalkServer
Client:Hello server
Hello Client
Client:How are you?
Fine,thank you.
Client:See you later.
Ok
Client null
Bye
D:\80JavaProgram\changxianxiang>

D:\命令提示符

D:\80JavaProgram\changxianxiang>java TalkClient
Hello server
Server:Hello Client
How are you?
Fine,thank you.
See you later.
Goodbye
Bye
D:\80JavaProgram\changxianxiang>

```

图9-3 运行界面

注意，首先运行的是Server端，然后才是Client端，顺序不能颠倒。如果先运行Client程序，那么Client端将无法找到服务器，就会出现异常，如图9-4所示。

```

D:\命令提示符

D:\80JavaProgram\changxianxiang>java TalkClient
IOException:
Can't find server.Program end.
D:\80JavaProgram\changxianxiang>

```

图9-4 出现异常

制作要点

1. Socket()的应用
2. ServerSocket()类的应用

步骤详解

1. 通信过程。

使用Socket进行Client/Server程序设计的一般连接过程是这样的：Server端监听某个端口是否有连接请求，Client端向Server端发出连接请求，Server端向Client端发回Accept（接受）消息，一个连接就建立起来了。Server端和Client端都可以通过send()、write()等方法与对方通信。对于一个功能齐全的Socket，其工作过程包含以下4个基本的步骤：

- (1) 创建Socket；
- (2) 打开连接到Socket的输入/输出流；
- (3) 按照一定的协议对Socket进行读/写操作；
- (4) 关闭Socket。

其中，第(3)步是程序员用来调用Socket和实现程序功能的关键步骤，其他三步在各种程序中基本相同。

2. 创建Socket。

Java在java.net包中提供了Socket和serverSocket两个类，分别用来表示双向连接的客户端和服务端。这是两个包装得非常好的类，使用很方便。其构造方法如下：

```
Socket(InetAddress address, int port);
Socket(InetAddress address, int port, boolean stream);
Socket(String host, int port);
Socket(String host, int port, boolean stream);
Socket(SocketImpl impl);
Socket(String host, int port, InetAddress localAddr, int localPort);
Socket(InetAddress address, int port, InetAddress localAddr, int localPort);
ServerSocket(int port);
ServerSocket(int port, int backlog);
ServerSocket(int port, int backlog, InetAddress bindAddr);
```

其中address、host和port分别是双向连接中另一方的IP地址、主机名和端口号；stream指明Socket是流Socket还是数据报Socket；localPort表示本地主机的端口号；localAddr和bindAddr是本地机器的地址（ServerSocket的主机地址）；impl是Socket的父类，既可以用来创建ServerSocket，又可以用来创建Socket；count则表示服务端所能支持的最大连接数。例如，对于客户端程序，可以通过生成一个Socket对象打开Socket：

```
Socket client;
client=new Sokcet ("Machine name", Portnumber);
```

注意：在选择端口时，必须小心。每一个端口提供一种特定的服务，只有给出正确的端口，才能获得相应的服务。0~1023的端口号为系统所保留，例如大家熟知的HTTP服务端口号为80，telnet服务端口号为21，FTP服务的端口号为23。所以在选择端口号时，最好选择一个大于1023的数以防止发生冲突。

在创建Socket时如果发生错误，将产生IOException异常，在程序中必须对其进行处理。如：

```
try( Socket socket=new Socket("kps", 2000); )  
catch(IOException e){ System.out.println("Error: "+e); }
```

对于服务方，通过生成一个ServerSocket对象打开Socket，然后调用方法accept()，准备接受客户发来的连接请求。同样，选择端口号时也要防止发生冲突。

```
ServerSocket server=null;  
Try{  
    Server=new ServerSocket(2000);  
}  
catch(IOException e){  
    System.out.println("can not listen to : "+e);  
}  
Socket socket=null;  
Try{  
    Socket=server.accept();  
}catch(IOException e){  
    System.out.println ("Error: "+e);  
}
```

accept()是阻塞性的方法，所谓阻塞性方法就是说该方法被调用后，将等待客户的请求，直到有一个客户启动并请求连接到相同的端口，然后 accept()返回一个对应于客户的Socket。这时，客户方和服务方都建立了用于通信的Socket，接下来就是由各个Socket分别打开各自的输入/输出流。

3. 打开输入/输出流。

类Socket提供了方法getInputStream()和getOutputStream()来得到对应的输入/输出流以进行读/写操作，这两个方法分别返回InputStream和OutputStream类对象。为了便于读/写数据，可以在返回的输入/输出流对象上建立过滤流，如DataInputStream、DataOutputStream或PrintStream类对象。对于文本方式流对象，可以采用InputStreamReader和OutputStreamWriter、PrintWriter等处理。例如：

```
PrintStream os=new PrintStream(  
    new BufferedOutputStream (socket.getOutputStream()));  
DataInputStream is=new DataInputStream(socket.getInputStream());  
PrintWriter out=new PrintWriter(socket.getOutputStream(),true);  
BufferedReader in=new BufferedReader(new InputStreamReader  
(Socket.getInputStream()));
```

4. 关闭Socket。

每一个Socket存在时，都将占用一定的资源，在使用完Socket对象时，要将其关闭。

关闭Socket可以调用Socket的close()方法。在关闭Socket之前，应将与Socket相关的所有的输入/输出流全部关闭，以释放所有的资源。而且要注意关闭的顺序，应该首先关闭与Socket相关的所有输入/输出，然后再关闭Socket。

程序源代码与解释

```
//chp 9
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.net.ServerSocket;

public class TalkClient {
    public static void main(String[] args) {
        try {
            //创建socket
            Socket socket = new Socket("127.0.0.1", 4700);
            BufferedReader sin = new BufferedReader(new InputStreamReader(
                System.in));
            //从键盘读入信息
            PrintWriter os = new PrintWriter(socket.getOutputStream());
            //打开连接socket输出流
            BufferedReader is = new BufferedReader(new InputStreamReader(socket
                .getInputStream())); //打开连接socket输入流
            String readline;
            readline = sin.readLine();
            while (!readline.equals("bye")) {
                os.println(readline); //输出通过键盘的录入信息
                os.flush();
                System.out.println("Server:" + is.readLine());
                //按照一定的协议读写
                readline = sin.readLine();
            }
            os.close();
            is.close();
            socket.close();
            //最后全部关闭
        } catch (UnknownHostException e) {
            System.out.println("Cann't find server.Program end.");
            System.exit(0);
        }
    }
}
```

```

        e.printStackTrace();
    } catch (IOException e) {
        System.out.println("Can't find server.Program end.");
        System.exit(0);
    } catch (Exception e) {
        e.getMessage();
    }
}

//Server类
public class TalkServer {
    public static void main(String[] args) {
        try {
            ServerSocket server = null;
            try {
                server = new ServerSocket(4700);
            } catch (IOException e) {
                System.out.println("can not listen to: " + e);
                e.getMessage();
            }
            Socket socket = null;
            try {
                socket = server.accept();
            } catch (Exception e) {
                System.out.println("Error " + e);
            }
            String line;
            BufferedReader is = new BufferedReader(new InputStreamReader(socket
                .getInputStream())); //输入流
            PrintWriter os = new PrintWriter(socket.getOutputStream());
            //输出流
            BufferedReader sin = new BufferedReader(new InputStreamReader(
                System.in));
            System.out.println("Client:" + is.readLine());
            line = sin.readLine();
            while (!line.equals("bye")) {
                os.println(line);
                os.flush();
                System.out.println("Client " + is.readLine());
                line = sin.readLine();
            }
            //读入一行，输出一行
        }
    }
}

```

```

        is.close();
        os.close();
        //关闭输入/输出流
        socket.close();
        //关闭socket，注意关闭顺序
        server.close();
    } catch (Exception e) {
        System.out.println("Error:" + e);
        e.getMessage();
    }
    //捕获异常
}
}

```



案例3 利用UDP Socket技术 实现IP多点传送

案例运行效果与操作

本案例是一个数据报通信的例子。程序运行时，服务器端会使用UDP协议将自己的源代码广播给所有的客户端程序。

首先在DOS对应的目录下输入“java MulticastServer2”，如图9-5所示。如果先运行客户端程序“java MulticastClient2”程序会抛出异常。

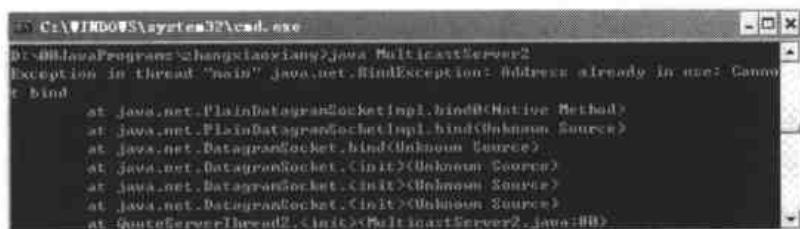


图9-5 运行界面1

然后再打开两个新的DOS窗口，大家会看到如图9-6所示的界面：两个客户端的DOS窗口的内容几乎是同时显示的。

制作要点

1. 类DatagramSocket()用于在程序之间建立传送数据报的通信连接
2. 类DatagramPacket()则用来表示一个数据报

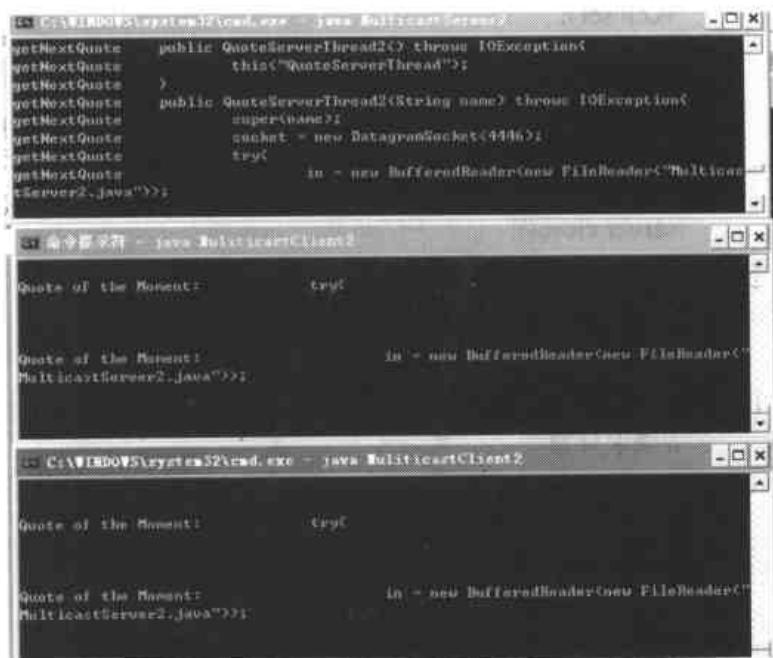


图9-6 运行界面2

步骤详解

1. 数据报通信和流式通信。

当用户进行网络编程时，有两种通信可供选择：数据报通信和流式通信。数据报通信协议UDP（User Datagram Protocol）是一种无连接的协议，每个数据报都是独立的信息，包括完整的源地址或目的地址，它在网络上以任何可能的路径传往目的地，因此，能否到达目的地、到达目的地的时间以及内容的正确性都是不能被保证的。流式通信协议TCP（Transfer control protocol）与UDP不同，它是面向连接的协议，发送方和接收方成对的两个Socket之间必须建立连接，以便在TCP协议的基础上进行通信，当一个Socket（通常都是Server Socket）等待建立连接时，另一个Socket可以要求进行连接，一旦这两个Socket连接起来，它们就可以进行双向数据传输，双方都可以进行发送或接收操作。究竟应使用哪种协议，这取决于不同的应用情况。

使用UDP时，每个数据报中都给出了完整的地址信息，因此无需建立发送方和接收方的连接。对于TCP协议，由于它是一个面向连接的协议，在Socket之间进行数据传输之前必然要建立连接，所以在TCP中多了一个建立连接的时间。使用UDP传输数据时是有大小限制的，每个被传输的数据报必须限定在64KB之内。而TCP没有这方面的限制，一旦连接建立起来，双方的Socket就可以按统一的格式传输大量的数据。UDP是一个不可靠的协议，发送方所发送的数据报并不一定以相同的次序到达接收方。而TCP是一个可靠的协议，它确保接收方完全正确地获取发送方所发送的全部数据。总之，TCP在网络通信上有极强的生命力，例如远程连接（Telnet）和文件传输（FTP）都需要不定长度的数据被可靠地传输。相比之下，UDP操作简单，而且仅需较少的监护，因此通常用于局域网高可靠性的分散系统中的Client/Server应用程序。

2. DatagramSocket和DatagramPacket。

java.net包中提供了DatagramSocket和DatagramPacket两个类，用来支持数据报通信，DatagramSocket用于在程序之间建立传送数据报的通信连接，DatagramPacket则用来表示一个数据报。先来看一下DatagramSocket的构造方法：

```
DatagramSocket();
DatagramSocket(int port);
DatagramSocket(int port, InetAddress laddr);
```

其中，port指明Socket所使用的端口号，如果未指明端口号，则把Socket连接到本地主机上一个可用的端口。laddr指明一个可用的本地地址。给出端口号时要保证不发生端口冲突，否则会生成SocketException类异常。注意上述的两个构造方法都声明抛出非运行时异常SocketException，程序中必须进行处理，或者捕获，或者声明抛出。用数据报方式编写Client/Server程序时，无论在客户器还是服务器，首先都要建立一个DatagramSocket对象，用来接收或发送数据报，然后使用DatagramPacket类对象作为传输数据的载体。下面看一下DatagramPacket的构造方法：

```
DatagramPacket(byte buf[],int length);
DatagramPacket(byte buf[], int length, InetAddress addr, int port);
DatagramPacket(byte[] buf, int offset, int length);
DatagramPacket(byte[] buf, int offset, int length, InetAddress address,int port);
```

其中，buf中存放数据报数据，length为数据报中数据的长度；addr和port指明目的地址；offset指明数据报的位移量。在接收数据前，应该采用上面的第一种方法生成一个DatagramPacket对象，给出接收数据的缓冲区及其长度。然后调用DatagramSocket的方法receive()等待数据报的到来，receive()将一直等待，直到收到一个数据报为止。

```
DatagramPacket packet=new DatagramPacket(buf, 256);
Socket.receive (packet);
```

发送数据前，也要先生成一个新的DatagramPacket对象，这时要使用上面的第二种构造方法，在给出存放发送数据的缓冲区的同时，还要给出完整的目的地址，包括IP地址和端口号。发送数据是通过DatagramSocket的方法send()实现的，send()根据数据报的目的地址来寻径，以传递数据报。

```
DatagramPacket packet=new DatagramPacket(buf, length, address, port);
Socket.send(packet);
```

在构造数据报时，要给出InetAddress类参数。类InetAddress在java.net包中定义，用来表示一个Internet地址，可以通过它提供的类方法getByName()从一个表示主机名的字符串获取该主机的IP地址，然后再获取相应的地址信息。

程序源代码与解析

```

//chp 9
/**
 * DatagramSocket只允许数据报发送一个目的地址, java.net
 * 包中提供了一个类MulticastSocket, 允许数据报以广播的方式
 * 发送到该端口的所有客户。MulticastSocket用在客户端, 监听
 * 服务器广播的数据
 * MulticastClient2.java
 */
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
public class MulticastClient2 {
    public static void main(String[] args) throws Exception{
        MulticastSocket socket = new MulticastSocket(4446);
        InetAddress address = InetAddress.getByName("230.0.0.1");
        //注意: 客户方使用的是MulticastSocket, 而且这个Socket需要捆绑在特定
        //的组地址上, 用JoinGroup()方法捆绑, 这里使用的组地址为: 230.0.0.1
        //注意, 这里的地址是广播地址
        socket.joinGroup(address);
        DatagramPacket packet = null;
        //get a few quotes
        for(int i =0;i<5; i++){
            byte[] buf = new byte[256];
            packet = new DatagramPacket(buf,buf.length);
            socket.receive(packet);
            String received = new String(packet.getData());
            System.out.println("Quote of the Moment:"+received);
        }
        socket.leaveGroup(address);
        socket.close();
    }
}
/**
 *可以通过这个方法将服务器收到的信息广播给大家
 * 首先, java MulticastServer2
 * 然后, java MulticastClient
 * 服务器最终会把从文件中读出的信息广播给大家
 * MulticastServer2.java
 */

```

```

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Date;
public class MulticastServer2 {
    public static void main(String[] args) throws java.io.IOException{
        new MulticastServerThread().start();
    }
}
class MulticastServerThread extends QuoteServerThread2{
    private long FIVE_SECONDS = 5000;
    //public MulticastServerThread
    public MulticastServerThread() throws IOException{
        super("MulticastServerThread");
    }
    boolean moreQuotes = true;
    String dString = null;
    public void run(){
        while(moreQuotes){
            try{
                byte[] buf = new byte[256];
                Date date = null;
                date = new Date();
                if(dString == null){
                    dString = new String(date.toString());
                } else {
                    dString = getNextQuote();
                }
                buf = dString.getBytes();
                //send it
                InetAddress group = null;
                try {
                    group = InetAddress.getByName("230.0.0.1");
                } catch (UnknownHostException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

        DatagramPacket packet = new DatagramPacket
(buf,buf.length,group,4446);
        socket.send(packet);
//sleep for a while
try{
    sleep((long)(Math.random()*FIVE_SECONDS));
}catch(InterruptedException e){
    e.getMessage();
}
}catch(IOException e){
    e.printStackTrace();
    moreQuotes = false;
}
}
socket.close();
}
}

class QuoteServerThread2 extends Thread{
protected DatagramSocket socket = null;
protected BufferedReader in = null;
protected boolean moreQuotes = true;
public QuoteServerThread2() throws IOException{
    this("QuoteServerThread");
}
public QuoteServerThread2(String name) throws IOException{
    super(name);
    socket = new DatagramSocket(4446);
    try{
        in = new BufferedReader(new FileReader("one-liners.txt"));
    }catch(FileNotFoundException e){
        System.err.println("Could not open quote file.Serving time instead.");
    }
}
public void run(){
while(moreQuotes){
    try{
        byte[] buf = new byte[256];
        //receive request
        DatagramPacket packet = new DatagramPacket(buf,buf.length);
        socket.receive(packet);
        //figure out response
    }
}
}
}

```

```

        String dString = null;
        if(in == null){
            dString = new Date().toString();
        }else{
            dString = getNextQuote();
        }
        buf = dString.getBytes();

        //send the response to the client at "address" and "port"
        InetAddress address = packet.getAddress();
        int port = packet.getPort();
        packet = new DatagramPacket(buf,buf.length,address,port);
        socket.send(packet);

    }catch(IOException e){
        e.printStackTrace();
        moreQuotes = false;
    }
}
socket.close();
}

protected String getNextQuote(){
    String value = null;
    try{
        if((value = in.readLine())==null){
            in.close();
            moreQuotes = false;
            value="No more quotes.Goodbye.";
        }
    }catch(IOException e){
        value = "IOException occurred in server";
    }finally{
        System.out.println("getNextQuote "+value);
    }
    return value;
}
}

```



案例4 利用Java API发送E-mail

案例运行效果与操作

在地址栏中，发送邮件的JSP页面如图9-7所示：输入邮件服务器、用户名、用户密码、邮件主题、收件人、发送人、邮件内容、附件，然后单击“发送”按钮。

邮件服务器	mail.keylab.net
用户名	wangqj@keylab.net
密码	*****
邮件主题	邮件发送测试
收件人	wangmj@keylab.net
发送人	wangqj@keylab.net
邮件内容	大家好！邮件测试。
附件	d:\\myforum.txt
<input type="button" value="发送"/> <input type="button" value="重置"/>	

图9-7 邮件发送界面

会看到“邮件发送成功！！！”的提示，如图9-8所示。

大家好！邮件测试。
wangmj@keylab.net
wangmj@keylab.net
wangmj@keylab.net
大家好！邮件测试。
d:\\myforum.txt
邮件发送成功！！！

图9-8 发送成功

在Outlook中，单击“发送/接收”按钮，就收到了刚才发送的测试邮件，如图9-9所示。



图9-9 收到邮件

制作要点

1. DataHandler()类的应用
2. FileDataSource()的使用
3. 类包Java.mail的作用

步骤详解

1. 邮件服务器身份验证。

通过设置mail.smtp.auth的值为true来解决：

```
Properties props = new Properties();
if (host != null && !host.trim().equals(""))
    props.put("mail.smtp.host", host);
else
    throw new Exception("没有指定发送邮件服务器");
if (needAuth)
    props.put("mail.smtp.auth", "true");
Session s = Session.getInstance(props, null);
```

2. 通过多种方式添加附件。

- 将本地磁盘上的文件作为附件

例如：send.addAttachFromFile("d:\\bb.txt", "aa.txt"), 其中d:\\bb.txt为邮件附件，在收信人的地方以aa.txt这个文件名来显示。

```
BodyPart mdp = new MimeBodyPart();
FileDataSource fds = new FileDataSource(filename);
DataHandler dh = new DataHandler(fds);
mdp.setFileName(MimeUtility.encodeWord(showname, "gb2312", null));
mdp.setDataHandler(dh);
attaches.add(mdp);
```

- 将互联网上的一个文件作为附件发送给收信人

例如：send.addAttachFromUrl("http://localhost/servlet-study/attachments/adobeapp.pdf", "first.pdf");，将String中的内容存放入文件showname，并将这个文件作为附件发送给收件人，@param string为邮件的内容，@param showname显示的文件的名字

```
BodyPart mdp = new MimeBodyPart();
DataHandler dh = new DataHandler(string, TEXT);
mdp.setFileName(MimeUtility.encodeWord(showname, "gb2312", null));
mdp.setDataHandler(dh);
attaches.add(mdp);
```

程序源代码与解析

```
//chp 9
/*
 * SendMail.java
```

```
/*
import java.net.URL;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Properties;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.activation.URLDataSource;
import javax.mail.BodyPart;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeUtility;

//import javax.mail.internet.MailDateFormat;

public class SendMail {
    public static String TEXT = "text/plain; charset=gb2312";
    public static String HTML = "text/html; charset=gb2312";
    private String host; //邮件服务器
    private String user; //用户名
    private String pass; //用户密码
    private String from; //发信人
    private String to; //收信人
    private String cc; //Carbon Copy, 抄送邮件给某人
    private String bc; //bcc Blind Carbon Copy, 暗送给某人
    private String subject; //邮件主题
    private BodyPart body; //邮件内容
    private boolean needAuth; //是否需要认证
    private List attaches; //邮件附件

    /**
     * 构造方法
     *
     */
    public SendMail() {
        needAuth = true;
        attaches = new ArrayList();
    }
}
```

```

/**
 * 构造方法
 * @param host
 */
public SendMail(String host) {
    needAuth = true;
    attaches = new ArrayList();
    this.host = host;
}

/**
 * 构造方法
 * @param host
 * @param user
 * @param pass
 */
public SendMail(String host, String user, String pass) {
    needAuth = true;
    attaches = new ArrayList();
    this.host = host;
    this.user = user;
    this.pass = pass;
}

//设置邮件服务器是否需要认证
public void setNeedAuth(boolean needAuth) {
    this.needAuth = needAuth;
}

public void setFrom(String from) {
    this.from = from;
}

public void setTo(String to) {
    this.to = to;
}

public String getPass() {
    return pass;
}

public String getUser() {
    return user;
}

public void setPass(String string) {
    pass = string;
}

public void setUser(String string) {
}

```

```
        user = string;
    }

    public String getFrom() {
        return from;
    }

    public String getHost() {
        return host;
    }

    public boolean isNeedAuth() {
        return needAuth;
    }

    public String getSubject() {
        return subject;
    }

    public void setHost(String string) {
        host = string;
    }

    public void setBlindTo(String bc) {
        this.bc = bc;
    }

    public void setCopyTo(String cc) {
        this.cc = cc;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    /**
     * 设置邮件内容的形式
     * @param string
     * @param contentType
     */
    public void setBody(String string, String contentType) {
        try {
            body = new MimeBodyPart();
            DataHandler dh = new DataHandler(string, contentType);
            body.setDataHandler(dh);
        } catch (Exception exception) {
        }
    }

    /**
     * 设置邮件内容的格式为文本格式
     * @param string
     */
```

```

/*
    public void setBodyAsText(String string) {
        setBody(string, TEXT);
    }
    /**
     * 以HTML的形式存放内容
     * @param string
     */
    public void setBodyAsHTML(String string) {
        setBody(string, HTML);
    }
    /**
     * 从文件中自动导入邮件内容
     * @param filename
     */
    public void setBodyFromFile(String filename) {
        try {
            BodyPart mdp = new MimeBodyPart();
            FileDataSource fds = new FileDataSource(filename);
            DataHandler dh = new DataHandler(fds);
            mdp.setDataHandler(dh);
            attaches.add(mdp);
        } catch (Exception exception) {
        }
    }
    /**
     * 从一个URL导入邮件的内容
     * @param url
     */
    public void setBodyFromUrl(String url) {
        try {
            BodyPart mdp = new MimeBodyPart();
            URLDataSource ur = new URLDataSource(new URL(url));
            DataHandler dh = new DataHandler(ur);
            mdp.setDataHandler(dh);
            attaches.add(mdp);
        } catch (Exception exception) {
        }
    }
    /**
     * 将String中的内容存放到文件showname，并将这个文件作为附件发送给收件人
     * @param string为邮件的内容
     */

```

```

* @param showname显示的文件的名字
*/
public void addAttachFromString(String string, String showname) {
    try {
        BodyPart mdp = new MimeBodyPart();
        DataHandler dh = new DataHandler(string, TEXT);
        mdp.setFileName(MimeUtility.encodeWord(showname, "gb2312", null));
        mdp.setDataHandler(dh);
        attaches.add(mdp);
    } catch (Exception exception) {
    }
}
/***
 * filename为邮件附件
 * 在收信人的地方以showname这个文件名来显示
 * @param filename
 * @param showname
 */
public void addAttachFromFile(String filename, String showname) {
    try {
        BodyPart mdp = new MimeBodyPart();
        FileDataSource fds = new FileDataSource(filename);
        DataHandler dh = new DataHandler(fds);
        mdp.setFileName(MimeUtility.encodeWord(showname, "gb2312", null));
        mdp.setDataHandler(dh);
        attaches.add(mdp);
    } catch (Exception exception) {
    }
}
/***
 * 将互联网上的一个文件作为附件发送给收信人
 * 并在收信人处显示文件的名字为showname
 * @param url
 * @param showname
 */
public void addAttachFromUrl(String url, String showname) {
    try {
        BodyPart mdp = new MimeBodyPart();
        URLDataSource ur = new URLDataSource(new URL(url));
        DataHandler dh = new DataHandler(ur);
        mdp.setFileName(MimeUtility.encodeWord(showname, "gb2312", null));
        mdp.setDataHandler(dh);
    }
}

```

```

        attaches.add(mdp);
    } catch (Exception exception) {
    }
}

/**
 * 发送邮件,需要身份认证
 * @throws Exception
 */
public void send() throws Exception {
    try {
        Properties props = new Properties();
        if (host != null && !host.trim().equals(""))
            props.put("mail.smtp.host", host);
        else
            throw new Exception("没有指定发送邮件服务器");
        if (needAuth)
            props.put("mail.smtp.auth", "true");
        Session s = Session.getInstance(props, null);
        MimeMessage msg = new MimeMessage(s);
        //设置邮件主题
        msg.setSubject(subject);
        //设置邮件发送时间
        msg.setSentDate(new Date());
        //制定发件人
        if (from != null)
            msg.addFrom(InternetAddress.parse(from));
        else
            throw new Exception("没有指定发件人");
        //指定收件人
        if (to != null)
            msg.addRecipients(javax.mail.Message.RecipientType.TO,
                InternetAddress.parse(to));
        else
            throw new Exception("没有指定收件人地址");
        //指定抄送给谁
        if (cc != null)
            msg.addRecipients(javax.mail.Message.RecipientType.CC,
                InternetAddress.parse(cc));
        //制定暗抄送给谁
        if (bc != null)
            msg.addRecipients(javax.mail.Message.RecipientType.BCC,
                InternetAddress.parse(bc));
    }
}

```

```
Multipart mm = new MimeMultipart();
//设置邮件的附件
if (body != null)
    mm.addBodyPart(body);
for (int i = 0; i < attaches.size(); i++) {
    BodyPart part = (BodyPart) attaches.get(i);
    mm.addBodyPart(part);
}
//设置邮件的内容
msg.setContent(mm);
//保存所有改变
msg.saveChanges();
//发送邮件服务器 (SMTP) , 简单邮件传输协议
Transport transport = s.getTransport("smtp");
//访问邮件服务器
transport.connect(host, user, pass);
//发送信息
transport.sendMessage(msg, msg.getAllRecipients());
//关闭邮件传输
transport.close();
} catch (Exception e) {
    throw new Exception("发送邮件失败:", e);
}
```

Testmail.jsp 测试邮件界面代码如下：

```
<%@ page contentType="text/html; charset=gb2312" language="java" %>
<%@ page import="ten.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
    <title>发送邮件测试</title>
    <meta name="copyright" content="邮件发送测试. All rights reserved.">
    <meta name="Keywords" content="jsp,servlet,mail">
    <meta name="Description" content="发送邮件测试">
    <meta http-equiv="content-type" content="text/html; charset=gb2312">
</head>
<body>
<form method="post" action="sendmail.jsp">
    <table border="1" cellpadding="0" cellspacing="3" align="center">
        <tr>
```

```

<td>邮件服务器</td><td><input type="text" name="host" value="mail.keylab.net"></td>
</tr>
<tr>
    <td>用户名</td><td><input type="text" name="user" value="wangmj@keylab.net"></td>
</tr>
<tr>
    <td>密码</td><td><input type="password" name="password" value="root3wmj"></td>
</tr>
<tr>
    <td>邮件主体</td><td><input type="text" name="subject" value="邮件发送测试"></td>
</tr>
<tr>
    <td>收件人</td><td><input type="text" name="to" value="wangmj@keylab.net"></td>
</tr>
<tr>
    <td>发送人</td><td><input type="text" name="from" value="wangmj@keylab.net"></td>
</tr>
<tr>
    <td>邮件内容</td><td><input type="text" name="bodyashtml" value="大家好！邮件测试。"></td>
</tr>
<tr>
    <td>附件</td><td><input type="text" name="attachfile" value="d:\\myforum.txt"></td>
</tr>
<tr>
    <td><input type="submit" name="submit" value="发送" ></td>
    <td><input type="reset" name="reset" value="重置" ></td>
</tr>
</table>
</form>
</body>
</html>

```

Sendmail.jsp发送邮件的JSP程序：

```
<%@ page contentType="text/html;charset=gb2312" language="java" %>
```

```

<%@ page import="ten.*;simpleforum.util.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>发送邮件测试</title>
<meta name="copyright" content="邮件发送测试. All rights reserved.">
<meta name="Keywords" content="jsp,servlet,mail">
<meta name="Description" content="发送邮件测试">
<meta http-equiv="content-type" content="text/html; charset=gb2312">
</head>
<body>
<%
String host = ParamUtil.getParameter(request,"host");
String user = ParamUtil.getParameter(request,"user");
String password = ParamUtil.getParameter(request,"password");
String subject = ParamUtil.getParameter(request,"subject");
String to = ParamUtil.getParameter(request,"to");
String from = ParamUtil.getParameter(request,"from");
String bodyashtml = ParamUtil.getParameter(request,"bodyashtml");
String attachfile = ParamUtil.getParameter(request,"attachfile");
out.println("<br>" + host);
out.println("<br>" + user);
out.println("<br>" + subject);
out.println("<br>" + to);
out.println("<br>" + from);
out.println("<br>" + bodyashtml);
out.println("<br>" + attachfile);
try {
    SendMail send = new SendMail();
    send.setHost(host);
    send.setUser(user);
    send.setPass(password);
    send.setSubject(subject);
    send.setTo(to);
    send.setFrom(from);
    send.setBodyAsHTML(bodyashtml);
    send.addAttachFromFile(attachfile,"aa.txt");
    send.send();
    out.println("<br>邮件发送成功！！！");
} catch (Exception e) {
    out.println("<br>" + e.getMessage());
    out.println("<br>邮件发送失败！！！");
}

```

```

    }
%
```

```

</body>
</html>
```



案例5 从Mail Server删除一条消息

案例运行效果与操作

本案例实现邮件的接收，并选择接收的邮件是否删除，如果选择是，将删掉所选择的邮件。

首先在Eclipse中运行该程序，将所有收到的邮件列出来，大家会看到共有5封邮件，如图9-10所示。再次启动该程序，输入“YES”删除其中的两封邮件，大家会看到图9-11所示的界面。

```

System.out
    .println("Do you want to delete message? [YES to delete]
    主要部分将不再输出，而下部保留在此示例的框架中
String line = reader.readLine();

控制台 x * 重置 *
已转让 PaletaMessageExample [Java 应用程序] C:\java\jre1.5.0_03\bin\javaw.exe (2005-5-8 15:39:34)
0: wangqiang@keylab.net 邮件发送测试
java.mail.internet.MimeMultipart@100d7c
Do you want to delete message? [YES to delete]
1: wangqiang@keylab.net 邮件发送测试
java.mail.internet.MimeMultipart@128e20
Do you want to delete message? [YES to delete]
2: wangqiang@keylab.net 邮件发送测试
java.mail.internet.MimeMultipart@100d7c
Do you want to delete message? [YES to delete]
3: wangqiang@keylab.net 邮件发送测试
java.mail.internet.MimeMultipart@100d7c
Do you want to delete message? [YES to delete]
4: wangqiang@keylab.net 邮件发送测试
java.mail.internet.MimeMultipart@100d7c
Do you want to delete message? [YES to delete]
```

图9-10 运行界面

最后再次运行该程序，列出所有的邮件，现在现只剩下三封邮件了（如图9-12所示）。

制作要点

1. SetFlag()方法的运用，设Flags.Flag.DELETED标志
2. Java.mail API的使用技巧

步骤详解

1. 登录时邮箱身份验证。

```

System.out.println("Do you want to delete message? [YES to delete]
是否要将消息删除。是否要删除邮件的消息?")
String line = reader.readLine();
if (line.equals("YES"))
    break;
}

// 按照白 X
[2005-5-6 15:41:09] DeleteMessageExample [Java 应用程序] C:\java\jral 5.0_03\bin\javaw.exe
0: wmjtest@keylab.net 邮件发送测试
javax.mail.internet.MimeMultipart@6bd433
Do you want to delete message? [YES to delete]
是否要将消息删除。是否要删除邮件的消息?

1: wmjtest@keylab.net 邮件发送测试
javax.mail.internet.MimeMultipart@6bd433
Do you want to delete message? [YES to delete]
是否要将消息删除。是否要删除邮件的消息?

2: wmjtest@keylab.net 邮件发送测试
javax.mail.internet.MimeMultipart@1100d7a
Do you want to delete message? [YES to delete]
是否要将消息删除。是否要删除邮件的消息?

3: wmjtest@keylab.net 邮件发送测试
javax.mail.internet.MimeMultipart@6bf0f07
Do you want to delete message? [YES to delete]
是否要将消息删除。是否要删除邮件的消息?

```

图9-11 删除界面

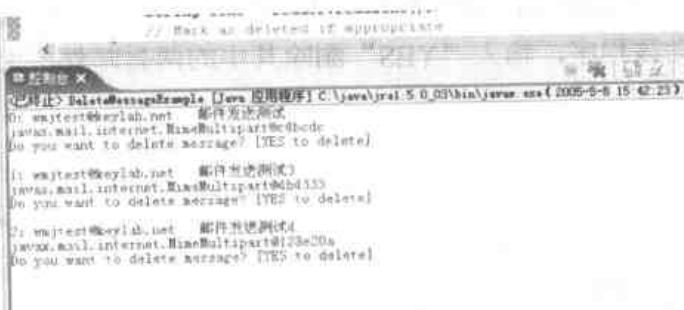


图9-12 运行界面

输入邮件账户和密码:

```

String host = "mail.keylab.net";
String from = "wmjtest@keylab.net";
String to = "wmjtest@keylab.net";
String username = "wmjtest@keylab.net";
String password = "wmjtest123";

```

获得系统属性:

```
Properties props = new Properties();
```

设置邮件服务器:

```

props.put("mail.smtp.host", host);
props.put("mail.smtp.auth", "true");

```

2. 通过pop3协议接收邮件。

获取javax.mail.Session实例，进而获得javax.mail.Store实例。

利用javax.mail.Store实例获得默认收件箱（INBOX）javax.mail.Folder实例。

使用javax.mail.Folder对象提取新邮件，存储为javax.mail.Message对象数组。

获取store:

```
Store store = session.getStore("pop3");
store.connect(host, username, password);
```

获取文件夹:

```
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_WRITE);
```

3. 删除邮件。

最基本的处理方式就是调用setFlag()方法，然后设置Flags.Flag.DELETED标志为真。

```
BufferedReader reader = new BufferedReader(new InputStreamReader(
    System.in));
```

获取目录:

```
Message message[] = folder.getMessages();
for (int i = 0, n = message.length; i < n; i++) {
    System.out.println(i + ": " + message[i].getFrom()[0] + "\t"
        + message[i].getSubject() + "\n" +
        message[i].getContent());
    System.out
        .println("Do you want to delete message? [YES to delete]");
```

邮件将被直接删除，而不是保留在服务器的垃圾箱中：

```
String line = reader.readLine();
```

标志删除位:

```
if ("YES".equals(line)) {
    message[i].setFlag(Flags.Flag.DELETED, true);
```

程序源代码与解析

```
//chp 9
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Properties;

import javax.mail.Flags;
import javax.mail.Folder;
```

```

import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Store;

public class DeleteMessageExample {
    public static void main(String args[]) throws Exception {
        //String host = "smtp.sina.com.cn";
        String host = "mailserver";
        String from = "username@mailserver.com ";
        String to = "username@mailserver.com";
        String username = "username";
        String password = "****";
        // 获取系统属性
        Properties props = new Properties();
        // 设置Mail Server
        props.put("mail.smtp.host", host);
        props.put("mail.smtp.auth", "true"); //这样才能通过验证
        // Get session
        Session session = Session.getDefaultInstance(props);
        //Session session = Session.getInstance(System.getProperties(), null);
        // Get the store
        Store store = session.getStore("pop3");
        store.connect(host, username, password);
        // Get folder
        Folder folder = store.getFolder("INBOX");
        folder.open(Folder.READ_WRITE);
        BufferedReader reader = new BufferedReader(new InputStreamReader(
            System.in));
        // Get directory
        Message message[] = folder.getMessages();
        for (int i = 0, n = message.length; i < n; i++) {
            System.out.println(i + ": " + message[i].getFrom()[0] + "\t"
                + message[i].getSubject() + "\n" +
                message[i].getContent());
            System.out
                .println("Do you want to delete message? [YES to delete]");
            //注意，邮件将被直接删除，而不是保留在服务器的垃圾箱中
            String line = reader.readLine();
            // Mark as deleted if appropriate
            if ("YES".equals(line)) {
                message[i].setFlag(Flags.Flag.DELETED, true);
            }
        }
    }
}

```

```
// Close connection  
folder.close(true);  
store.close();  
}  
}
```

案例6 在Java程序中实现FTP的功能



案例运行效果与操作

本案例讲解了如何通过Java API实现FTP的功能。功能Jar包可以从<http://jakarta.apache.org/commons/net/>来下载包，如图9-13所示。



图9-13 下载界面

本案例的主要任务就是通过FTP服务器来下载一个文件，同时将服务器上所有的文件或者文件夹的名称打印出来。

首先，要有一个FTP服务器，网络上有很多FTP软件，Windows操作系统本身也具有FTP功能。登录FTP服务器，可以看到在文件夹“ht”下面有一个压缩文件isic-0.05.tar，如图9-14所示。

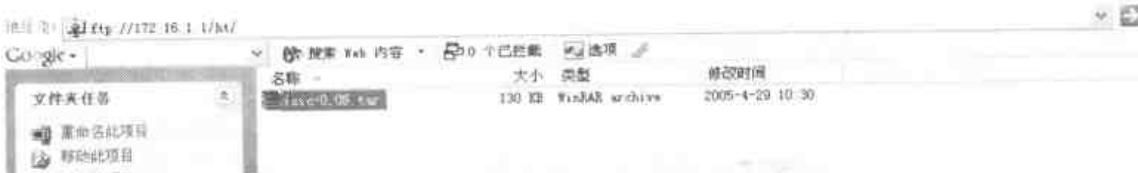


图9-14 登录到FTP服务器

运行FTPTest，文件jsjc-0.05.tar已经被保存到d盘根目录下面，如图9-15所示。



图9-15 运行界面

如图9-16和图9-17所示，程序已经将FTP服务器根目录下的所有文件或者文件夹打印出来了。



图9-16 运行界面



图9-17 输出界面

制作要点

1. IOException的用法
2. passive mode的使用

步骤详解

1. 连接服务器并登录。

```
ftest.connect(server);
ftest.login(username, password);
```

2. 下载文件并保存到本地硬盘上。

由于现在的FTP服务器一般都连接在防火墙后面，因此要注意编写的程序能够穿越防火墙，默认状态下多使用passive mode。

```
if (binaryTransfer)
    ftp.setFileType(FTP.BINARY_FILE_TYPE);
ftp.enterLocalPassiveMode();
if (storeFile) {
    InputStream input;
    input = new FileInputStream(local);
    // 下载文件并存放到路径input下
    ftp.storeFile(remote, input);
    input.close();
} else {
    OutputStream output;
    output = new FileOutputStream(local);
```

从服务器接收文件并保存到指定输出流中：

```
ftp.retrieveFile(remote, output);
//要自动关闭输出流示例对象
output.close();
ftp.logout();
```

源代码与解析

```
//chp 9
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

```

import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.net.SocketException;

import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPConnectionClosedException;
import org.apache.commons.net.ftp.FTPFile;
import org.apache.commons.net.ftp.FTPReply;

public final class FtpTest {

    public static final void main(String[] args) {
        int base = 0;
        boolean storeFile = false, binaryTransfer = false, error = false;
        String server, username, password, remote, local;
        FTPClient ftp;
        server = "172.16.1.1";
        username = "public";
        password = "public";
        remote = "ht/isic-0.05.tar";
        local = "d:\isic-0.05.tar";

        ftp = new FTPClient();
        ftp.addProtocolCommandListener(new PrintCommandListener(
            new PrintWriter(System.out)));
        //-----新添加的-----
        FTPClient ftest = new FTPClient();
        try {
            ftest.connect(server);
            ftest.login(username, password);
            FTPFile[] files = ftest.listFiles();
            System.out.println("输出FTP服务器根目录下的文件或文件夹名字开始: \n");
            for (int i = 0; i < files.length; i++) {
                System.out.println(IsoToCh(files[i].getName()));
            }
            System.out.println("输出FTP服务器根目录下的文件或文件夹名字结束。 \n");
        } catch (SocketException e1) {
            // TODO自动生成catch块
            e1.printStackTrace();
        } catch (IOException e1) {
            // TODO自动生成catch块
            e1.printStackTrace();
        }
        //-----
    }
}

```

```

try {
    int reply;
    ftp.connect(server);
    System.out.println("Connected to " + server + ":");

    // 尝试连接后，需要检验返回代码以验证是否连接成功
    reply = ftp.getReplyCode();

    if (!FTPReply.isPositiveCompletion(reply)) {
        ftp.disconnect();
        System.err.println("FTP server refused connection.");
        System.exit(1);
    }
} catch (IOException e) {
    if (ftp.isConnected()) {
        try {
            ftp.disconnect();
        } catch (IOException f) {
            // do nothing
        }
    }
    System.err.println("Could not connect to server.");
    e.printStackTrace();
    System.exit(1);
}

__main: try {
    if (!ftp.login(username, password)) {
        ftp.logout();
        error = true;
        break __main;
    }

    System.out.println("Remote system is " + ftp.getSystemName());

    if (binaryTransfer)
        ftp.setFileType(FTP.BINARY_FILE_TYPE);

    // 默认状态下使用passive mode，因为大多都在防火墙后面
    ftp.enterLocalPassiveMode();

    if (storeFile) {
        InputStream input;
        input = new FileInputStream(local);
        ftp.storeFile(remote, input);
        input.close();
    }
}

```

```

        }
    } else {
        OutputStream output;
        output = new FileOutputStream(local);
        ftp.retrieveFile(remote, output);
        output.close();
    }
    ftp.logout();
} catch (FTPConnectionClosedException e) {
    error = true;
    System.err.println("Server closed connection.");
    e.printStackTrace();
} catch (IOException e) {
    error = true;
    e.printStackTrace();
} finally {
    if (ftp.isConnected()) {
        try {
            ftp.disconnect();
        } catch (IOException f) {
            // do nothing
        }
    }
}
System.exit(error ? 1 : 0);
} // end main
/**
 * iso8859_1转换为gb2312编码
 * @param param
 * @return
 */
public static String IsoToCh(String param) {
    String temp = new String(param);
    //这里都需要使用new方法，否则WebSphere不兼容
    try {
        temp = new String(temp.getBytes("iso8859_1"), "gb2312");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    return temp;
}
}

```



案例7 一个最简单的聊天程序

案例运行效果与操作

本案例是一个简单的聊天程序，利用Socket实现服务器端和客户端的握手、连接、对话。

服务器端启动后，等待客户端的连接，如图9-18所示。

客户端启动后，与服务器端通信成功，如图9-19所示。

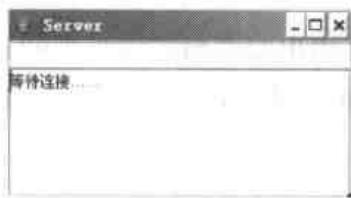


图9-18 运行界面

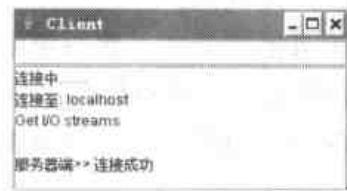


图9-19 连接成功界面

相应的，服务端也显示一个客户端连接上来，并等待接受输入流，如图9-20所示。

连接成功后，两端可以进行简单的通话，图9-21和图9-22所示。

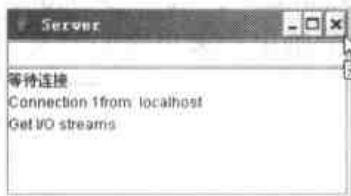


图9-20 连接上一个客户端

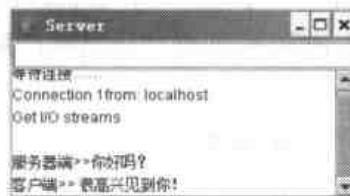


图9-21 Server端信息显示

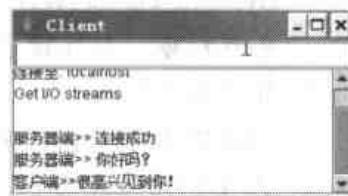


图9-22 Client端信息显示

通话完毕后，关闭连接。

制作要点

1. ServerSocket类的方法，创建多线程的应用
2. GetInputStream()输入流方法的使用
3. GetOutputStream()输出流方法的使用
4. 输入/输出流缓冲

步骤详解

1. 建立服务器。

用服务器需要使用的端口号作为参数来创建服务器对象。

```
server = new ServerSocket(4000)
```

这个服务器使用4000端口。当一个客户端程序建立一个Socket连接，所连接的端口号为4000时，服务器对象Server便响应这个连接，并且server.accept()方法会创建一个Socket对象。服务器端便可以利用这个Socket对象与客户进行通信。

```
connection = server.accept()
```

进而得到输入流和输出流，并进行封装：

```
inputS = new ObjectInputStream(server.getInputStream());
outputS = new ObjectOutputStream( server.getOutputStream() );
```

2. 建立客户端代码。

相比服务器端而言，客户端要简单一些，客户端只需用服务器所在机器的地址以及服务器的端口作为参数创建一个Socket对象。得到这个对象后，就可以用“建立服务器”部分介绍的方法实现数据的输入和输出。

```
toclient = new Socket( InetAddress.getByName( chatServer ), 4000 );
outputS = new ObjectOutputStream(client.getOutputStream());
inputS = new ObjectInputStream(client.getInputStream());
```

以上的程序代码建立了一个Socket对象，这个对象连接到IP地址为InetAddress.getByName(chatServer)值的主机上（本案例中，chatserver为Localhost）、端口为4000的服务器对象。并且建立了输入流和输出流，分别对应服务器的输出和客户端的写入。

3. 建立用户界面。

可以根据用户喜好来建立，这里不再赘述。

4. 本案例可支持多线程。

```
try
{ file://建立服务器
ServerSocket server = new ServerSocket(9998);
int i=1;
for(;;)
{
    Socket incoming = server.accept();
    new ServerThread(incoming,i).start();
    i++;
}
}catch (IOException ex){ ex.printStackTrace(); }
```

循环检测是否有客户连接到服务器上，如果有，则创建一个线程来服务这个客户，这个线程的名称是ServerThread，这个类扩展了Thread类，它的编写方法与服务器的写法相同。

程序源代码与解密

服务器端：

```
/* 聊天程序之服务器端ChatServer.java
 * 建立服务器端，接收客户端连接，
 * 发送信息、关闭连接
 */
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ChatServer extends JFrame {
    private JTextField inputBox;
    private JTextArea outFrame;
    private ObjectOutputStream outputS;
    private ObjectInputStream inputS;
    private ServerSocket toserver;
    private Socket server;
    private int counter = 1;

    public ChatServer()
    {
        super( "Server" );
        Container container = getContentPane();
        inputBox = new JTextField();
        inputBox.setEnabled( false );
        //输出
        inputBox.addActionListener(
        //监听
        new ActionListener() {
            //发送信息
            public void actionPerformed( ActionEvent event )
            {
                sendMsg( event.getActionCommand() );
            }
        }
    );
    container.add( inputBox, BorderLayout.NORTH );
```

```

        //输出框
        outFrame = new JTextArea();
        container.add( new JScrollPane( outFrame ),
                BorderLayout.CENTER );
        setSize( 280, 160 );
        setVisible( true );
    }

    //处理连接
    public void connectServer()
    {
        try {
            //创建一个ServerSocket.
            toserver = new ServerSocket( 4000, 100 );
            while ( true ) {
                //等待连接
                wait4Connection();
                //获取输入输出流
                getStreams();
                //处理连接
                processConnection();
                //关闭连接
                closeConnection();
                ++counter;
            }
        }
        catch ( EOFException eofException ) {
            System.out.println( "Client terminated connection" );
        }
        //捕获异常
        catch ( IOException ioException ) {
            ioException.printStackTrace();
        }
    }

    private void wait4Connection() throws IOException
    {
        outFrame.setText( "等待连接……\n" );
        connection = toserver.accept();
        outFrame.append( "Connection " + counter +
                "from: " + connection.getInetAddress().getHostName()
        );
    }
}

```

```

private void getStreams() throws IOException
{
    //设置输出流
    outputS = new ObjectOutputStream(
        server.getOutputStream()
    );
    outputS.flush();
    //设置输入流
    inputS = new ObjectInputStream(
        server.getInputStream()
    );
    outFrame.append( "\nGet I/O streams\n" );
}

//处理客户端连接
private void processConnection() throws IOException
{
    //连接成功
    String message = "服务器端>> 连接成功";
    outputS.writeObject( message );
    outputS.flush();
    //输入框
    inputBox.setEnabled( true );
    //处理来自客户端的信息
    do {
        //读取消息
        try {
            message = ( String ) inputS.readObject();
            outFrame.append( "\n" + message );
            outFrame.setCaretPosition(
                outFrame.getText().length()
            );
        }
        catch ( ClassNotFoundException classNotFoundException ) {
            outFrame.append( "\nUnknown object type received" );
        }
    } while ( !message.equals( "客户端>> TERMINATE" ) );
}

//关闭socket, 先关闭输入输出流
private void closeConnection() throws IOException
{
    outFrame.append( "\nUser terminated connection" );
    inputBox.setEnabled( false );
    outputS.close();
}

```

```

        inputS.close();
        server.close();
    }
    //向客户端发送消息
    private void sendMsg( String message )
    {
        try {
            outputS.writeObject( "服务器端>> " + message );
            outputS.flush();
            outFrame.append( "\n服务器端>>" + message );
        }
        catch ( IOException ioException ) {
            outFrame.append( "\nError writing object" );
        }
    }
    //main()方法
    public static void main( String args[] )
    {
        ChatServer process = new ChatServer();
        process.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE );
        process.connectServer();
    }
} // end class ChatServer

```

客户端程序：

```

/* 聊天程序之客户端ChatClient.java
 * 建立客户端以获取服务器端发送信息并显示
 */
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ChatClient extends JFrame {
    private JTextField inputBox;
    private JTextArea outFrame;
    private ObjectOutputStream outputS;
    private ObjectInputStream inputS;
    private String message = "";
    private String chatServer;

```

```

private Socket toclient;
//初始化聊天服务
public ChatClient( String srvhost )
{
    super( "Client" );
    chatServer = srvhost;
    //设置客户端连接的服务器
    Container container = getContentPane();
    inputBox = new JTextField();
    //建立输入框
    inputBox.setEnabled( false );
    inputBox.addActionListener(
        //监听
        new ActionListener() {
            public void actionPerformed( ActionEvent event )
            {
                sendMsg( event.getActionCommand() );
            }
            //向服务器端发送消息
        }
    );
    container.add( inputBox, BorderLayout.NORTH );
    outFrame = new JTextArea();
    container.add( new JScrollPane( outFrame ),
        BorderLayout.CENTER );
    setSize( 280, 160 );
    setVisible( true );
    //输出输出框
}
//连接服务器端
public void connectClient()
{
    try {
        //创建用于连接的Socket
        connect2Server();
        //得到输入输出流
        getStreams();
        //建立连接
        processConnection();
        //关闭连接
        closeConnection();
    }
}

```

```

        catch ( EOFException eofException ) {
            System.out.println( "Server terminated connection" );
        }
        catch ( IOException ioException ) {
            ioException.printStackTrace();
        }
        //捕获异常
    }

    //获取输入输出流
    private void getStreams() throws IOException
    {
        //输出
        outputS = new ObjectOutputStream(
            toclient.getOutputStream());
        outputS.flush();

        //输入
        inputS = new ObjectInputStream(
            toclient.getInputStream());
        outFrame.append( "\nGet I/O streams\n" );
    }

    //连接服务器端
    private void connect2Server() throws IOException
    {
        outFrame.setText( "连接中.....\n" );
        toclient = new Socket(
            InetAddress.getByName( chatServer ), 4000 );

        //连接信息显示
        outFrame.append( "连接至：" +
            toclient.getInetAddress().getHostName() );
    }

    private void processConnection() throws IOException
    {
        //输出框
        inputBox.setEnabled( true );
        do {
            //读入信息并输出
            try {
                message = ( String ) inputS.readObject();
                outFrame.append( "\n" + message );
                outFrame.setCaretPosition(

```

```

        outFrame.getText().length() );
    }
    catch ( ClassNotFoundException classNotFoundException ) {
        outFrame.append( "\nUnknown object type received" );
    }
} while ( !message.equals( "服务器端>> TERMINATE" ) );
}

//关闭输入输出流、关闭连接，注意顺序
private void closeConnection() throws IOException
{
    outFrame.append( "\n关闭连接" );
    outputS.close();
    inputS.close();
    toclient.close();
}

//给服务器端发消息
private void sendMsg( String message )
{
    try {
        outputS.writeObject( "客户端>> " + message );
        outputS.flush();
        outFrame.append( "\n客户端>>" + message );
    }
    catch ( IOException ioException ) {
        outFrame.append( "\nError writing object" );
    }
}

//main()方法
public static void main( String args[] )
{
    ChatClient beginning;
    if ( args.length == 0 )
        beginning = new ChatClient( "127.0.0.1" );
    else
        beginning = new ChatClient( args[ 0 ] );
    beginning.setDefaultCloseOperation(
        JFrame.EXIT_ON_CLOSE );
    beginning.connectClient();
}
} // end class ChatClient

```



案例8 代理服务器的实现

案例运行效果与操作

代理服务器打开一个端口，接收浏览器发来的访问某个站点的请求，从请求的字符串中解析出用户想访问哪个网页，然后通过URL对象建立输入流以读取相应的网页内容，最后按照Web服务器的工作方式将网页内容发送给用户浏览器。

程序运行后，如图9-23所示，说明代理服务已经在808端口启动。

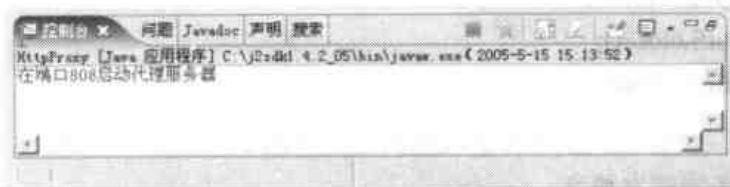


图9-23 代理服务器启动界面

为了测试有无代理服务功能，将代理服务器设置为本机，端口为808，如图9-24所示。

客户端浏览网页时，控制台返回的信息如图9-25所示，显示此URL地址通过代理。同时浏览器也返回用户要求的网页内容。

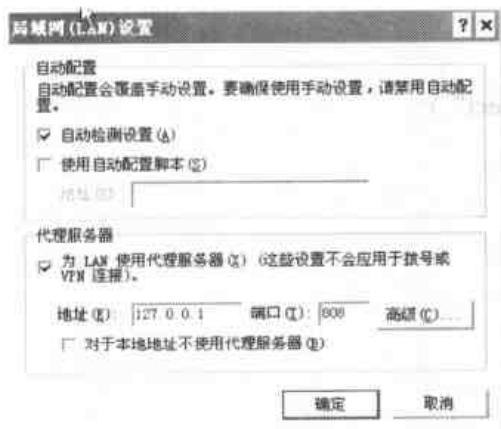


图9-24 测试设定代理服务器

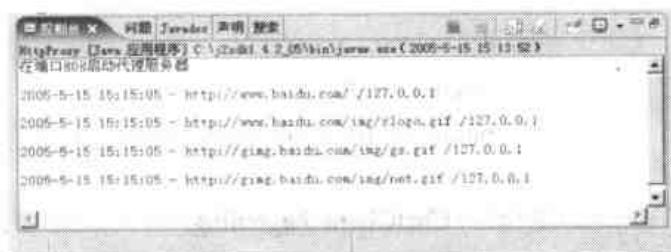


图9-25 显示信息

制作要点

1. 在给定Socket上创建线程
2. 创建类的实例

步骤详解

1. 在给定Socket上创建线程。

```
public SimpleProxy(Socket s) { socket=s; start(); }
```

2. 创建代理服务类的实例。

```
Constructor layer = test.getDeclaredConstructor(sarg);
arg[0]=sersock.accept();
layer.newInstance(arg);
```

3. 可建立二级代理。

HttpProxy的简单派生类：

```
public class SubProxy extends SimpleProxy {}
```

也可以建立日志文件，直接将代理信息记入日志文件。

实际上，这是个网页代理服务器，可以服务于多个连接，但不具有用户验证等功能，在java.net API中，软件可以通过两个属性来支持代理服务器，分别是http.proxyHost 和http.proxyPort。感兴趣的可以试一下。

程序源代码与解释

```
/*
 * 简单代理服务器实现
 */
import java.net.*;
import java.io.*;
import java.util.*;
import java.text.*;
import java.text.DateFormat.*;
import java.lang.reflect.*;

public class SimpleProxy extends Thread {
    static public int Retry_Num=3;
    static public int Connect_Delay=3;
    static public int TimeOut=20;
    static public int BufferSize=1024;
    static public boolean Passport = false;
    static public OutputStream Message=null;

    //传入数据用Socket
    protected Socket socket;
    //上级代理服务器
    static private String Parent=null;
    static private int ParentPort=-100;

    static public void ParentProxy(String parentname, int portnum) {
```

```

        Parent=parentname;
        ParentPort=portnum;
    }

    //在给定Socket上创建一个代理线程
    public SimpleProxy(Socket sock) {
        socket=sock;
        start();
    }

    public void recordLog(int c) throws IOException {
        Message.write(c);
    }

    //记录、输出日志
    public void recordLog(byte[] bytes,int c, int lenth) throws IOException {
        for (int i=0;i<lenth;i++)
            recordLog((int)bytes[c+i]);
    }

    //默认情况下，日志信息输出到标准输出设备

    public String getHostName(String url, String host, int portnum, Socket sock) {
        DateFormat ware=DateFormat.getDateInstance();
        System.out.println(ware.format(new Date()) + " - " + url + " "
            + sock.getInetAddress()+"\n");
        //输出格式即2005-5-16 22:01:06 - http://www.google.com/ /127.0.0.1
        return host;
    }

    //执行操作的线程
    public void run() {
        String strline;
        String strhost;
        int port=80;
        Socket out=null;
        try {
            socket.setSoTimeout(TimeOut);
            InputStream instr=socket.getInputStream();
            OutputStream osstr=null;
            try {
                //获取请求行的内容
                strline="";
                strhost="";
                int state=0;
                boolean blank;

```

```

while (true) {
    int l=instr.read();
    if (l== -1) break;
    if (Passport) recordLog(l);
    blank=Character.isWhitespace((char)l);
    switch (state) {
        case 0:
            if (blank) continue;
            state=1;
        case 1:
            if (blank) {
                state=2;
                continue;
            }
            strline=strline+(char)l;
            break;
        case 2:
            if (blank) continue;
            //跳过空白字符
            state=3;
        case 3:
            if (blank) {
                state=4;
                //只取出主机名称部分
                String host=strhost;
                int n;
                n=strhost.indexOf("//");
                if (n!= -1) strhost=strhost.substring(n+2);
                n=strhost.indexOf('/');
                if (n!= -1) strhost=strhost.substring(0,n);
                //分析可能存在的端口号
                n=strhost.indexOf(":");
                if (n!= -1) {
                    port=Integer.parseInt(strhost.substring(n+1));
                    strhost=strhost.substring(0,n);
                }
                strhost=getHostName(host,strhost,port,socket);
            }
            if (Parent!=null) {
                strhost=Parent;
                port=ParentPort;
                //处理上级代理
            }
    }
}

```

```

        int retry=Retry_Num;
        while (retry--!=0) {
            try {
                out=new Socket(strhost,port);
                break;
            } catch (Exception e) { }
            //等待
            Thread.sleep(Connect_Delay);
        }
        if (out==null) break;
        out.setSoTimeout(TimeOut);
        osstr=out.getOutputStream();
        osstr.write(strline.getBytes());
        osstr.write(' ');
        osstr.write(host.getBytes());
        osstr.write(' ');
        check(instr,out.getInputStream(),osstr,socket.getOutputStream());
        break;
    }
    strhost=strhost+(char)l;
    break;
}
}
}
catch (IOException e) {
}
}
catch (Exception e) {
}
finally {
    try {
        socket.close();
    } catch (Exception e1) {
    }
    //关闭socket
    try {
        out.close();
    } catch (Exception e2) {
    }
    //关闭socket
}
}

void check(InputStream instr0, InputStream instr1,

```

```

OutputStream outstr0, OutputStream outstr1) throws IOException {
    try {
        int len;
        byte bytes[] = new byte[BufferSize];
        while (true) {
            try {
                if ((len=instr0.read(bytes))>0) {
                    outstr0.write(bytes,0,len);
                    if (Passport)
                        recordLog(bytes,0,len);
                        //记录日志
                }
                else if (len<0)
                    break;
            } catch (InterruptedIOException e) {
            }
            try {
                if ((len=instr1.read(bytes))>0) {
                    outstr1.write(bytes,0,len);
                    if (Passport)
                        recordLog(bytes,0,len);
                }
                else if (len<0)
                    break;
            } catch (InterruptedIOException e) {
            }
        }
    } catch (Exception e0) {
        System.out.println("异常: " + e0);
    }
}

static public void startProxy(int port, Class test) {
    ServerSocket sersock;
    Socket sock;
    try {
        sersock = new ServerSocket(port);
        while (true) {
            Class [] sarg = new Class[1];
            Object [] arg = new Object[1];
            sarg[0] = Socket.class;
            try {
                Constructor layer = test.getDeclaredConstructor(sarg);

```

```
    arg[0]=sersock.accept();
    layer.newInstance(arg);
    //创建SimpleProxy 及其派生类的实例
} catch (Exception e) {
    Socket esock = (Socket)arg[0];
    try { esock.close();
    } catch (Exception ec) {}
}
}
} catch (IOException e) {
}
}

//main方法,用于简单测试
static public void main(String args[]) {
    System.out.println("在端口808启动代理服务器\n");
    SimpleProxy.Message=System.out;
    SimpleProxy.Passport=false;
    SimpleProxy.startProxy(808,SimpleProxy.class);
}
}
```

本 章 小 结

Java有很强的Internet网络程序设计功能，似乎专门为网络设计的，用其开发网络软件特别便利。这不仅仅在于java具有很好的安全性和可移植性，还在于java为Internet编程提供了丰富的网络类库支持。本章通过几个实现简单网络功能的案例，涉及了使用Java语言的网络编程方法，但要成为Java网络编程高手，还需要进一步的学习和大量经验的积累。

第10章

Java综合实例



本章内容

案例1 用Java实现zip压缩解压缩

案例2 简易的图书管理系统

案例3 UFO攻击游戏

案例4 制作一个MP3播放器

本章小结



案例1 用Java实现zip压缩解压缩

案例运行效果与操作

本案例是一个用Java实现的压缩、解压缩zip文件的例子，实现了将各种文件压缩成zip文件，并解压缩zip文件的功能。程序运行后，如图10-1所示。

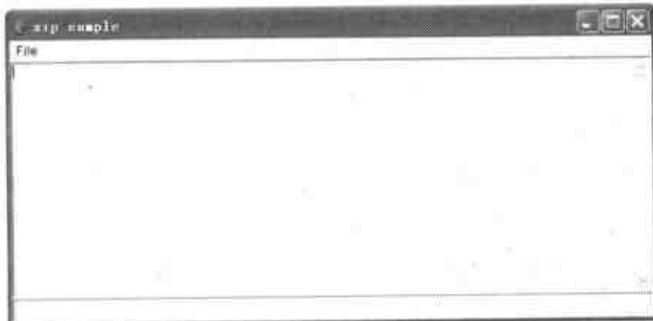


图10-1 运行界面

单击“File”菜单，即可显示出所有的二级菜单“New”、“Open”、“Save”、“Exit”，如图10-2所示。

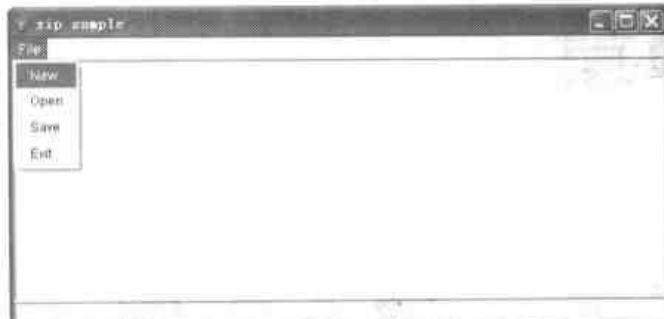


图10-2 主菜单

单击“New”菜单，程序就会随机产生50个Double类型的数字，如图10-3所示。

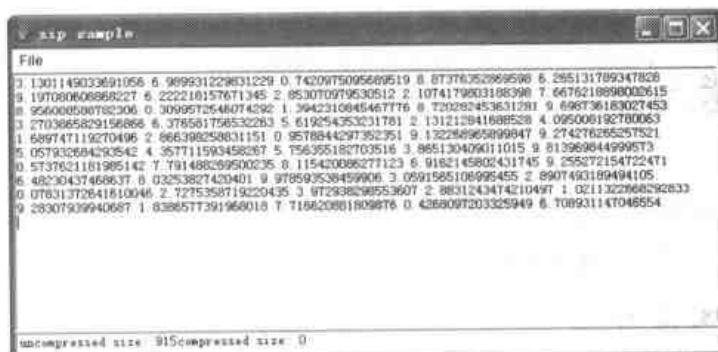


图10-3 随机产生50个Double类型的数字

单击“Save”菜单，程序会自动打开如图10-4所示的窗口，然后在“文件名”后面输入要保存的zip文件的名字，如我们起名为“firstzip”，单击“保存(S)”按钮，大家看到如图10-5所示的界面。下面一栏的提示“uncompressed size: 915 compressed size: 599”是说文件压缩之前是915字节，压缩之后是599字节，压缩率为65.46%。我们会在桌面上找到一个名为“firstzip.zip”的文件，然后可以用WinRAR软件解压缩，和我们生成的数据做一下比较。感兴趣的读者还可以对Java程序和WinRAR软件的压缩率做一下比较。

将文本区域中的文字清空，然后单击“Open”菜单，找到我们保存在桌面的“firstzip.zip”文件，大家会发现我们的程序已经将数据成功解压。

单击“Exit”按钮退出程序。

制作要点

1. 随机函数Math.random()的用法
2. 文件实例的创建以及文件各种应用，包括的打开、读取、保存、关闭等
3. ZipInputStream类的用法，输入过滤流，用来读取ZIP格式文件中的文件
4. ZipOutputStream类的用法，输出过滤流，用来向ZIP格式文件写入文件



图10-4 将50个Double类型的数据压缩保存为zip文件

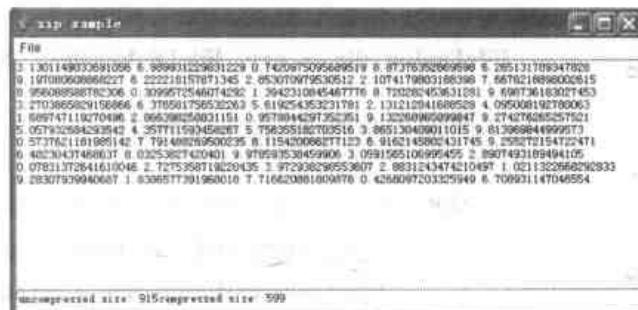


图10-5 文件压缩保存之后

5. Deflater类的用法，使用ZLIB压缩类，支持通常的压缩方式
6. Inflater类的用法，使用ZLIB压缩类，支持通常的解压方式

步骤详解

1. 随机生成50个Double类型的数字。

生成随机数，作为将要压缩的内容，并放在doc字符串变量中。

```
for (int i = 1; i < 51; i++) {
    double rdm = Math.random() * 10;
    doc = doc + new Double(rdm).toString();
    if (i % 5 == 0)
        doc = doc + "\n";
    else
        doc = doc + " ";
```

2. 打开zip文件，将输入流定位在当前entry数据项位置。

打开zip文件，将文件内容读入doc字符串变量中。用文件输入流构建zip压缩输入流：

```
FileDialog dlg = new FileDialog(this, "Open", FileDialog.LOAD);
```

从后缀为.zip文件的入口读取文件，并决定入口数据流数据的位置：

```
ZipInputStream zipis = new ZipInputStream(new FileInputStream(f));
```

将输入流定位在当前entry数据项位置：

```
zipis.getNextEntry();
```

用zip输入流构建DataInputStream：

```
DataInputStream dis = new DataInputStream(zipis);
doc = dis.readUTF();//读取文件内容
```

3. 将数据压缩为zip文件，生成一个ZIP entry，并写入文件输出流中。将输出流定位于entry起始处，用zip输出流构建DataOutputStream。

打开zip文件，将doc字符串变量写入文件中：

```
FileDialog dlg = new FileDialog(this, "Save", FileDialog.SAVE);
```

用文件输出流构建zip压缩输出流：

```
ZipOutputStream zpos = new ZipOutputStream(new FileOutputStream(f));
zpos.setMethod(ZipOutputStream.DEFLATED);//设置压缩方法
zpos.putNextEntry(new ZipEntry("zip"));
```

生成一个zip entry，写入文件输出流中，并将输出流定位于entry起始处：

```
DataOutputStream os = new DataOutputStream(zpos);
```

用zip输出流构建DataOutputStream：

```
os.writeUTF(doc);//将随机生成的数据写入文件中
os.close();//关闭数据流
doczipsize = f.length();//获取压缩文件的长度
showTextandInfo();//显示数据
```

4. 本程序只实现一段随机生成的浮点数字的压缩，用同样的方法可以把压缩输入流定位为一段缓冲区或者是一个文件。

Java提供了java.util.zip包用来兼容zip格式的数据压缩。它提供了一系列的类用来读取、创建、修改zip和Gzip格式的文件，还提供了工具类来计算任意输入流的数目，这可以用来验证输入数据的有效性。读者可以详细查阅这方面的资料以实现更为完善的压缩、解压缩功能。

程序源代码与解释

```
//chp10

import java.awt.FileDialog;
import java.awt.Frame;
import java.awt.Menu;
import java.awtMenuBar;
import java.awtMenuItem;
import java.awtTextArea;
import java.awtTextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.zip.ZipEntry;
import java.util.zip.ZipInputStream;
import java.util.zip.ZipOutputStream;

public class TestZip extends Frame implements ActionListener {
    TextArea textarea; //显示数据文件的多行文本
    TextField infotip; //显示数据文件未压缩大小以及压缩大小单行文本显示域
    String doc; //存储随机生成的数据
    long doczipsize = 0;//压缩数据文件的大小
    public TestZip() {
        //生成菜单
        MenuBar menubar = new MenuBar();
        setMenuBar(menubar);
        Menu file = new Menu("File", true);
        menubar.add(file);
        MenuItem neww = new MenuItem("New");
        neww.addActionListener(this);
        file.add(neww);
        MenuItem open = new MenuItem("Open");
        open.addActionListener(this);
        file.add(open);
        MenuItem save = new MenuItem("Save");
        save.addActionListener(this);
        file.add(save);
        MenuItem exit = new MenuItem("Exit");
        exit.addActionListener(this);
        file.add(exit);
        //随机生成的数据文件的多行文本显示域
        add("Center", textarea = new TextArea());
        //提示文本原始大小、压缩大小的单行文本显示域
        add("South", infotip = new TextField());
        //关闭当前的窗口，释放系统资源
        this.addWindowListener(
            new WindowAdapter(){
                public void windowClosing(WindowEvent e){
                    System.exit(0);
                }
            }
        );
    }
}

```

```

        }
    }

}

public static void main(String args[]) {
    TestZip ok = new TestZip();
    ok.setTitle("zip sample");
    ok.setSize(600, 300);
    ok.setVisible(true);
}

private void randomData() {
    //随机生成50个double数据,并放在doc字符串变量中。
    doc = "";
    for (int i = 1; i < 51; i++) {
        double rdm = Math.random() * 10;
        doc = doc + new Double(rdm).toString();
        if (i % 5 == 0)
            doc = doc + "\n";
        else
            doc = doc + " ";
    }
    doczipsize = 0;
    showTextandInfo();
}

private void openFile() {
    //打开zip文件,将文件内容读入doc字符串变量中。
    FileDialog dlg = new FileDialog(this, "Open", FileDialog.LOAD);
    dlg.setVisible(true);
    //取得文件目录、文件名
    String filename = dlg.getDirectory() + dlg.getFile();
    try {
        //创建一个文件实例
        File f = new File(filename);
        if (!f.exists())
            return;
        //文件不存在，则返回
        //用文件输入流构建ZIP压缩输入流
        ZipInputStream zipis = new ZipInputStream(new FileInputStream(f));
        //读取下一个压缩文件入口，定位入口数据流开始位置。
        //从后缀为.zip文件的入口读取文件，并决定入口数据流数据的位置。
        zipis.getNextEntry(); //将输入流定位在当前entry数据项位置
        DataInputStream dis = new DataInputStream(zipis);
        //用zip输入流构建DataInputStream
        doc = dis.readUTF(); //读取文件内容
    }
}

```

```

        dis.close(); //关闭文件
        doczipsize = f.length(); //获取zip文件长度
        showTextandInfo(); //显示数据
    }
    catch (IOException ioe) {
        System.out.println(ioe);
    }
}

private void saveFile() {
    //打开zip文件，将doc字符串变量写入zip文件中。
    FileDialog dlg = new FileDialog(this, "Save", FileDialog.SAVE);
    dlg.setVisible(true);
    String filename = dlg.getDirectory() + dlg.getFile();
    try {
        //检查文件的后缀是否为“zip”，如果不是以“.zip”结尾则自动增加
        if (!filename.endsWith(".zip")){
            filename += ".zip";
        }
        File f = new File(filename);
        //创建一个文件实例
        //检查文件是否存在，如果不存在就重新创建一个新的文件。
        if (!f.exists())
            f.createNewFile();
        //用文件输出流构建ZIP压缩输出流
        ZipOutputStream zpos = new ZipOutputStream(new FileOutputStream(f));
        zpos.setMethod(ZipOutputStream.DEFLATED); //设置压缩方法
        zpos.putNextEntry(new ZipEntry("zip"));
        //生成一个ZIP entry，写入文件输出流中，并将输出流定位于entry起始处。
        DataOutputStream os = new DataOutputStream(zpos);
        //用ZIP输出流构建DataOutputStream;
        os.writeUTF(doc); //将随机生成的数据写入文件中
        os.close(); //关闭数据流
        doczipsize = f.length(); //获取压缩文件的长度
        showTextandInfo(); //显示数据
    }
    catch (IOException ioe) {
        System.out.println(ioe);
    }
}

private void showTextandInfo() {
    //显示数据文件和压缩信息
    textarea.replaceRange(doc, 0, textarea.getText().length());
    infotip.setText("uncompressed size: " + doc.length());
}

```

```

        + "compressed size: " + doczipsize);
    }

    public void actionPerformed(ActionEvent e) {
        String arg = e.getActionCommand();
        if ("New".equals(arg))
            randomData();
        else if ("Open".equals(arg))
            openFile();
        else if ("Save".equals(arg))
            saveFile();
        else if ("Exit".equals(arg)) {
            dispose(); //关闭窗口
            System.exit(0); //关闭程序
        }
        else {
            System.out.println("无此命令!");
        }
    }
}
}

```



案例2 简易图书管理系统

案例运行效果与操作

本案例是一个Java语言编写的综合应用程序，实现简单图书入库和流通管理、图书证管理和统计查询的功能。主要涉及图形用户接口和数据库访问方面的知识。程序运行后界面如图10-6所示：

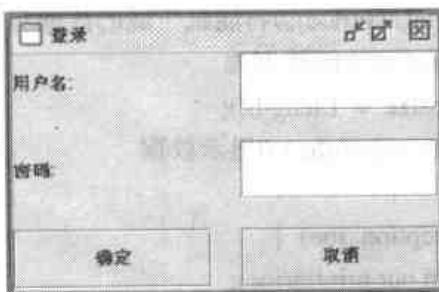


图10-6 登录界面

用户名为admin，密码为123（或者用户名为guest，密码也是guest）。登录成功后，出现程序主界面，如图10-7所示。

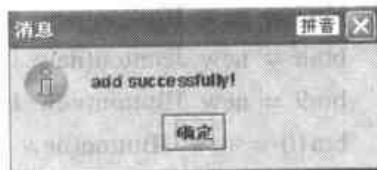


图10-7 主界面

图10-8为图书入库界面。

书名:	Java高级编程	条形码:	7-301-12345-6
作者:	王开诚	分编号:	7-031
分类名:	综合教程	馆藏号:	301
出版社:	电子工业出版社	出版日期:	2005-04-01
入库日期:	2005-5-15	价格:	100
Java 开发经验的积累与实践			
<input type="button" value="添加"/>		<input type="button" value="删除"/>	<input type="button" value="修改"/>

图10-8 图书入库界面



图书录入成功提示

图10-9为图书信息查询界面。图10-10为图书证办理界面。

图书信息查询									
书名		条形码		类别		出版日期		状态	
书名	条形码	分类号	分类名	编架号	出版社	出版日期	入库日期	状态	简介
adf	333	dsf	ad	ads	ed	2001-0	2005-0	在架	ad
aff	444	adf	adsf	ad	ad	2003-0	2005-0	在架	ad
adf	555	adf	adsf	ad	bs	2002-0	2005-0	在架	b
af	666	aaa	f	f	f	2001-0	2005-0	在架	adsf
vvv	131	asf	adsf	asd	asd	2001-0	2005-0	在架	adsf
sdgf	456	gad	政治	sdf	sdf	2001-0	2005-0	在架	adsf
fsdgg	567	fgs	政治	dfg	sdg	2004-0	2004-1	在架	sdg
adf	asf	asd	政治	asd	ad	2004-0	2004-1	在架	adssad
大	微热	认为人	政治	仍然	真	2005-0	2005-0	在架	见书
111	111222	aaa	马克恩	dfa	asd	2004-0	2005-0	在架	ad

图10-9 图书信息查询界面

图书证办理	
借阅证办理	
姓名:	<input type="text"/>
性别:	<input type="text"/>
班级:	<input type="text"/>
证件号码:	<input type="text"/>
进馆日期:	<input type="text"/>
有效日期:	<input type="text"/>
<input type="button" value="添加"/>	<input type="button" value="删除"/>

图10-10 图书证办理界面

制作要点

1. 图形用户接口Frame的用法
2. 数据库访问

步骤详解

1. 构造界面。

前面章节中很多案例都涉及到用户界面的构造，还有各种布局的用法，界面风格设计取决于用户的喜好，这里不再赘述。

2. 设置工具栏。

```
JToolBar JTB = new JToolBar();
工具栏
btn1 = new JButton("A");
设置菜单一的字体
btn1.setFont(new Font("楷体", Font.BOLD + Font.CENTER_BASELINE, 16));
btn2 = new JButton(new ImageIcon("images/register.gif"));
btn3 = new JButton(new ImageIcon("images/new.gif"));
btn4 = new JButton("C");
btn4.setFont(new Font("楷体", Font.BOLD + Font.CENTER_BASELINE, 16));
btn5 = new JButton(new ImageIcon("images/paste.gif"));
btn6 = new JButton(new ImageIcon("images/right.gif"));
btn7 = new JButton(new ImageIcon("images/middle.gif"));
btn8 = new JButton(new ImageIcon("images/left.gif"));
btn9 = new JButton(new ImageIcon("images/sound.gif"));
btn10 = new JButton(new ImageIcon("images/underline.gif"));
```

3. 连接数据库。

连接数据库

```
try{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con=DriverManager.getConnection("jdbc:odbc:library");
stmt=con.createStatement();}
```

验证用户名和密码

```
sqlStr="select * from login where user_name='"+str1+"' and psw='"+str2+"';
ResultSet result=stmt.executeQuery(sqlStr);
if(result.next())
{
    username=result.getString("user_name");
    if(username.equals("guest"))
    {
```

```

MainFrame mainFrame=new MainFrame();
mainFrame.menuItem1.setEnabled(false);
.....
}
else
{ new MainFrame();
}

```

然后关闭连接

```

stmt.close();
con.close();

```

4. 数据库操作。

本案例涉及大量的数据库操作，图书入/出库，流通、图书证管理、及统计查询等。如图书入库：

```

sqlStr1="insert into book (bookname,bannercode,kindnumber,kindname,"+
"positionnumber,publishingcompany,publishtime,putintime,price,"+
"state,introduction,author) values(?,?,?,?,?,?,?,?,?,?,'在架',?,?)";
sqlStr2="delete from book where bannercode=?";
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con=DriverManager.getConnection("jdbc:odbc:library");
pstmt1=con.prepareStatement(sqlStr1);
pstmt2=con.prepareStatement(sqlStr2);

```

程序源代码与解释

程序源代码在附带光碟中，由于太多，这里只列出Login.java，其主要功能为管理员登录，数据库连接、验证。

```

//chp10
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JFrame;

```

```
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class Login extends JFrame {
    private JLabel JLb1;//用户名
    private JLabel JLb2;//密码
    private JButton Ok_btn;//确定按钮
    private JButton Cancel_btn;//取消按钮
    private JTextField jtflduser; //用户名文本框
    private JPasswordField jtpwdfld;//密码文本框
    private JFrame frame;
    private Connection con;
    private Statement stmt;
    public Login() {
        JFrame.setDefaultLookAndFeelDecorated(true);
        frame = new JFrame("登录");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container content = frame.getContentPane();
        //采用GridLayout布局管理器
        content.setLayout(new GridLayout(3, 2, 20, 20));
        JLb1 = new JLabel("用户名:");
        JLb2 = new JLabel("密码:");
        jtflduser = new JTextField();
        jtpwdfld = new JPasswordField();
        Ok_btn = new JButton("确定");
        Cancel_btn = new JButton("取消");
        //为按钮增加ActionListener()监听器
        Ok_btn.addActionListener(new ActionHandler());
        Cancel_btn.addActionListener(new ActionHandler());
        content.add(JLb1);
        content.add(jtflduser);
        content.add(JLb2);
        content.add(jtpwdfld);
        content.add(Ok_btn);
        content.add(Cancel_btn);
        frame.pack();
        frame.setLocationRelativeTo(null);//设定登陆窗口启动时出现在屏幕中央或接近
        屏幕中央
        frame.setSize(300, 200);
```

```

frame.setVisible(true);
//使用jdbc - odbc桥连接数据库，需要配置odbc驱动，数据库的名字为library
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    //注册jdbcodbc驱动
    con = DriverManager.getConnection("jdbc:odbc:library");
    //链接access数据库library
    stmt = con.createStatement();
    //创建statement

} catch (ClassNotFoundException e) {
    System.out.println(e.getMessage());
} catch (SQLException ex) {
    System.out.println(ex.getMessage());
}
}

/**
 * 实现ActionListener监听，激活组件时响应
 */
class ActionHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String str1, str2, username, sqlStr;
        Object obj = e.getSource();
        str1 = jtflduser.getText().trim();
        str2 = new String(jtpwdfld.getPassword()).trim();
        try {
            //点击确定
            if (obj.equals(Ok_btn)) {
                if (str1.equals("")) {
                    JOptionPane.showMessageDialog(frame,
                        "username can't be null!");
                    return;
                }
                //产生登录sql语句
                sqlStr = "select * from login where user_name=" + ""
                    + str1 + "" + " and psw=" + "" + str2 + "";
                ResultSet result = stmt.executeQuery(sqlStr);
                if (result.next()) {
                    username = result.getString("user_name");
                    //如果用户名为guest
                    if (username.equals("guest")) {
                        //为不同的用户赋予不同的权限
                    }
                }
            }
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
}

```

```
MainFrame mainFrame = new MainFrame();
mainFrame.menuItem1.setEnabled(false);
mainFrame.menuItem14.setEnabled(false);
mainFrame.menuItem3.setEnabled(false);
mainFrame.menuItem4.setEnabled(false);
mainFrame.menuItem5.setEnabled(false);
mainFrame.menuItem6.setEnabled(false);
mainFrame.menuItem7.setEnabled(false);
mainFrame.menuItem15.setEnabled(false);

} else {
    //登录成功打开主窗口
    new MainFrame();
}

//关闭登录窗口
frame.dispose();
stmt.close(); //关闭statement
con.close(); //关闭数据库链接

} else {
    JOptionPane.showMessageDialog(frame,
        "username or password is error!");
}

} else if (obj.equals(Cancel_btn)) {
    //点击取消退出应用程序
    System.exit(0);
}

} catch (SQLException ex) {
    System.err.println(ex);
}

}

}

public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        //启动登录窗口
        public void run() {
            new Login();
        }
    });
}
```



案例3 UFO攻击游戏

案例运行效果与操作

本案例是一个用Java实现的UFO攻击小游戏。操作很简单，如图10-11所示，用鼠标左键单击UFO，每击落10架UFO之后，就会增加一架。即开始的时候是1架出现，每当击落10架之后就增加1架，依次类推直至同时出现的UFO架数达到5便不再增加。在攻击UFO的同时，一旦UFO落到了地面，那么游戏也将结束，如图10-12所示。单击鼠标左键游戏重新开始，当然计数器也会重新置零。



图10-11 运行界面



图10-12 UFO落地，游戏结束

制作要点

1. Applet类的构造函数
2. 鼠标和键盘的响应事件
3. 动画处理
4. 声音处理

步骤详解

本案例由下表所示的几个类构成。

表10-1 类及其说明

类名	说明
UFO_Attack	本Applet的主类，完成系统初始化，创建并显示GUI，调用其他类完成与本案例相关的所有代码逻辑
Piece	定义一个基类，完成位置、颜色、速度、高度和宽度等配置，以及绘制图像并将缓冲图像进行显示输出等功能，由Launcher、Missile、UFO、Explosion类进行继承
Launcher	导弹发射架类，继承于Piece类

(续表)

类名	说明
Missile	Missile类，继承于Piece类
UFO	UFO类，继承于Piece类
Explosion	爆炸类，继承于Piece类

1. 编写HTML页面

```
<HTML>
<HEAD>
<TITLE>UFO ATTACK</TITLE>
<META NAME="GENERATOR" Content="Eclipse">
</HEAD>
<BODY>
<applet code="UFO_Attack.class" WIDTH="250" HEIGHT="300" VIEWASTEXT>
</applet>
</BODY>
</HTML>
```

2. 装载图片

Java Applet常用来显示存储在GIF文件中的图像。Java Applet装载GIF图像时需定义Image对象。多数Java Applet使用的是GIF或JPEG格式的图像文件。Applet使用getImage方法把图像文件和Image对象联系起来。

Graphics类的drawImage方法用来显示Image对象。为了提高图像的显示效果，许多Applet都采用双缓冲技术：首先把图像装入内存，然后再显示在屏幕上。

```
tracker = new MediaTracker(this) ; //媒体跟踪器监测图像装载的情况
图片的装载：
bgimg = getImage(this.getCodeBase(),"bgimg.gif") ;
tracker.addImage(bgimg,0) ;
ufostrip = getImage(this.getCodeBase(),"ufostrip.gif") ;
tracker.addImage(ufostrip,0) ;
missile = getImage(this.getCodeBase(),"missile.gif") ;
tracker.addImage(missile,0) ;
missile_explosion = getImage(this.getCodeBase(),"explosionstrip.gif") ;
tracker.addImage(missile_explosion,0) ;
```

3. 动画技巧 - 消除闪烁

本案例使用两种方法来消除闪烁：重载update()或使用双缓冲。

(1) 重载update()

改写update()方法，使该方法不会清除整个画面，只是消除必要的部分。

使动画更加流畅

```
public void update(Graphics g) {
    paint(g);
}
```

(2) 使用双缓冲技术

另一种减小帧之间闪烁的方法是使用双缓冲。其主要原理是创建一个后台图像，将需要绘制的一帧画入图像，然后调用DrawImage()将整个图像一次画到屏幕上；好处是大部分绘制是离屏的，将离屏图像一次绘至屏幕上比直接在屏幕上绘制要有效得多，大大提高做图的性能。

设置绘制的区域

```
set_draw_rectangles(a.paint_area, a.new_area);
```

绘制缓冲区

```
Graphics bg = a.buffer.getGraphics();
bg.clipRect(a.paint_area.x, a.paint_area.y, w, h);
bg.drawImage(a.backdrop, 0, 0, a);
bg.dispose();
```

填充缓冲区

```
a.buf_g.setColor(c);
a.buf_g.fillRect(a.new_area.x, a.new_area.y, w, h);
```

使用新的区域

```
a.paint_area.add(a.new_area);
```

将缓冲绘制到屏幕上

```
Graphics g = a.getGraphics();
g.clipRect(a.paint_area.x, a.paint_area.y, a.paint_area.width, a.paint_area.height);
g.drawImage(a.buffer, 0, 0, a);
g.dispose();
```

4. 播放声音文件

首先定义AudioClip对象，GetAudioClip方法能把声音赋予AudioClip对象，如果仅想把声音播放一遍，应调用AudioClip类的play方法，如果想循环把声音剪辑，应选用AudioClip类的loop方法。

加载声音文件

```
if (explosion == null)
    explosion = getAudioClip(getCodeBase(), "explosion.au");
if (newufo == null)
    newufo = getAudioClip(getCodeBase(), "sonar.au");
if (missile_launch == null)
    missile_launch = getAudioClip(getCodeBase(), "rocket.au");
game_over = true;
```

声音播放

```
newufo.play() ;
missile_launch.play() ;
explosion.play() ;
```

5. 响应鼠标事件。

处理鼠标移动事件

```
mouse_x = x; //返回鼠标所在位置的X坐标
```

处理鼠标的按下事件

```
public boolean mouseDown(Event e, int x, int y) {
```

游戏结束时所做的相应处理

```
if (game_over) {
```

.....

程序源代码与解释

```
//chp10
//引入所需的类包
import java.applet.Applet;
import java.applet.AudioClip;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Event;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Rectangle;
import java.util.Vector; //将类UFO_Attack转换为线程，并实现Runnable接口
public class UFO_Attack extends Applet implements Runnable {
    Image buffer = null; //临时图像缓冲
    Image backdrop = null; //背景幕
    Image bgimg = null; //原背景幕
    Image ufostrip = null; //UFO序列图
    Image missile = null; //导弹序列图
    Image missile_explosion = null; //导弹爆炸序列图
    MediaTracker tracker = null; //媒体跟踪器，用来监测图像的装载
    Graphics buf_g = null; //缓冲中的图像对象
    Graphics bkd_g = null; //背景的图像对象
```

```

Dimension window_size = null; //窗口尺寸
Font font ;
Font font_s ; //显示字的字体
AudioClip explosion = null ; //爆炸声
AudioClip newufo = null ; //新的UFO出现时发出的声音
AudioClip missile_launch = null ; //导弹发射的声音
Thread game = null ; //程序的主线程
boolean game_over = true ; //用来判断游戏结束与否
int mouse_x = 100 ; //鼠标的X坐标，用来控制导弹和发射架的移动
Rectangle paint_area = new Rectangle() ; //对象出现的区域
Rectangle new_area = new Rectangle() ; //对象即将出现的区域
Launcher L = null ; //定义一个导弹发射架
Missile M = null ; //定义一个导弹
Vector UV = new Vector() ; //定义UFO向量，即一个UFO集合
Vector EV = new Vector() ; //定义爆炸向量，即一个爆炸集合
int NU = 1 ; //UFO的数目
int score = 0 ; //玩家所得分数
//相应用对象的颜色设置
Color gunColor;
Color mColor;
Color ufoColor;
Color scoreColor;
Color bgColor;
//UFO_Attack类的初始化
public void init() {
    System.out.println("UFO Attack: A game by Sergio Fanchiotti");
    tracker = new MediaTracker(this) ; //媒体跟踪器监测图像装载的情况
    bgimg = getImage(this.getCodeBase(),"bgimg.gif") ; //图片的装载
    tracker.addImage(bgimg,0) ;
    ufostrip = getImage(this.getCodeBase(),"ufostrip.gif") ;
    tracker.addImage(ufostrip,0) ;
    missile = getImage(this.getCodeBase(),"missile.gif") ;
    tracker.addImage(missile,0) ;
    missile_explosion = getImage(this.getCodeBase(),"explosionstrip.gif") ;
    tracker.addImage(missile_explosion,0) ;
    font = new Font("Helvetica", Font.BOLD, 24) ;
    font_s = new Font("Helvetica", Font.BOLD, 14) ; //显示字的字体设置
    //设置所需的颜色
    bgColor = new Color(0,0,128);
    gunColor = new Color(0,88,0);
    mColor = new Color(255,255,255);
    ufoColor = new Color(255,0,0);
}

```

```
scoreColor = new Color(0,0,255);
}

public void start() {
    //使用十字型光标
    getFrame(this).setCursor(Frame.CROSSHAIR_CURSOR) ;
    window_size = size();      //获取窗口的尺寸
    buffer = null;           //生成缓冲区
    buffer = createImage(window_size.width, window_size.height);
    backdrop = null;
    backdrop = createImage(window_size.width, window_size.height);
    //用背景色来填充缓冲区
    buf_g = buffer.getGraphics();
    buf_g.setColor(bgColor);
    buf_g.fillRect(0, 0, window_size.width, window_size.height);
    //显示初始化信息
    set_say_font(font) ;
    set_say_mode(CENTER) ;
    set_say_style(SHADOW) ;
    say("UFO",10,80) ;
    say("ATTACK") ;
    set_say_font(font_s) ;
    set_say_style(NORMAL) ;
    say("") ;
    say("Click to start") ;
    say("a game") ;
    //将缓冲绘制到屏幕上
    Graphics g = getGraphics() ;
    g.drawImage(buffer,0,0,this) ;
    //初始化导弹发射架
    mouse_x = window_size.width/2 ;
    L = new Launcher(this) ;
    L.set_color(gunColor) ;
    //初始化导弹
    M = new Missile(this) ;
    M.set_color(mColor) ;
    //加载声音文件
    if (explosion == null)
        explosion = getAudioClip(getCodeBase(),"explosion.au") ;
    if (newufo == null)
        newufo = getAudioClip(getCodeBase(),"sonar.au") ;
    if (missile_launch == null)
        missile_launch = getAudioClip(getCodeBase(),"rocket.au") ;
```

```

game_over = true ;
    //声音播放
    newufo.play() ;
missile_launch.play() ;
explosion.play() ;
}
//停止运行函数
public void stop() {
    if (game != null) {
        game.stop() ;
        game = null ; //如果线程正在运行，强行令其停止
    }
    //重新设置光标形状
    getFrame(this).setCursor(Frame.DEFAULT_CURSOR) ;
}
//游戏主线程的执行函数
public void run() {
    //定义本地变量和对象
    UFO U ;
    Explosion E ;
    long count = 0 ;
    long ti = 0 ;
    //等待图片装载完毕
    Graphics g = getGraphics() ;
    g.setColor(Color.red) ;
    g.drawString("Starting Game...", 20,20) ;
    while (tracker.checkAll(true) == false) {
        if ((ti++ % 2) == 0)
            g.setColor(Color.red) ;
        else
            g.setColor(Color.green) ;
        g.drawString("*", 10,22) ;
        try {Thread.sleep(50);} catch (InterruptedException e) {} ;
        //装载超时时强行退出
        if (ti > 1000) break ;
    }
    //捕捉获取图片时的错误信息
    if (tracker.isErrorAny()) {
        showStatus("Error getting images") ;
        return ;
    }
    showStatus("Loading completed") ;//装载成功
}

```

```

        g.dispose() ;
        //绘制背景的缓冲区
        buf_g = backdrop.getGraphics();
        buf_g.drawImage(bgimg,0,0,window_size.width,window_size.height,this) ;
        //将背景的缓冲绘制到屏幕上
        buf_g = getGraphics();
        buf_g.drawImage(backdrop,0,0,this) ;
        // 绘制缓冲区
        buf_g = buffer.getGraphics();
        buf_g.drawImage(bgimg,0,0,window_size.width,window_size.height,this) ;
        repaint(); //重新绘制
        display_score(); //显示玩家得分数
        L.draw(); //绘制导弹发射架
        showStatus("UFO ATTACK");
        //事件循环
        for (;;) {
            ti = System.currentTimeMillis();
            //如果有多余的UFO飞行空间的话可增加一架UFO
            if ((UV.size() < NU) &&
                (Math.random() > (UV.size() == 0 ? 0.90 : 0.98))) {
                newufo.play(); //播放警报声
                U = new UFO(this);
                U.set_color(ufocolor);
                //在相应条件下提高UFO的下降速度
                if (score > 10 && Math.random() > 0.7) U.vy -= 1;
                //在UFO向量中增加一名成员
                UV.addElement(U);
            }
            //在背景上绘制爆炸画面，结束后将其清除
            for (int j=EV.size()-1; j>=0 ; -j) {
                E = (Explosion) EV.elementAt(j);
                if (E.active) {
                    E.draw();
                } //如果爆炸出现就进行其画面的绘制
                else {
                    //结束后从背景上清除，并从爆炸向量中删除
                    E.erase();
                    EV.removeElementAt(j);
                }
            }
            //移动导弹发射架
            L.move();
        }
    }
}

```

```

    //如果导弹存在，移动导弹
    if (M.active() == true) M.move() ;
    //移动每个UFO
    for (int i=0; i < UV.size(); ++i) {
        U = (UFO) UV.elementAt(i) ;
        U.move() ;
    }
    //监控UFO与导弹之间的碰撞
    for (int i=(UV.size()-1); i >=0 ; --i) {
        U = (UFO) UV.elementAt(i) ;
        if (U.active() && M.active() && U.collision(M)) {
            ++score ; //增加玩家的得分
            explosion.stop() ;
            display_score() ;
            explosion.play() ;
            //每击落10架UFO后便增加UFO的最大出现数目，直到数目为5
            if ((NU < 5) && (score % 10) == 1) ++NU ;
            //碰撞发生后，将导弹从背景上清除，并使其active属性为false
            M.active(false) ;
            M.erase() ;
            //将被击中的UFO从背景上清除，并使其active属性为false
            U.active(false) ;
            U.erase() ;
            //显示爆炸的场面
            E = new Explosion(this,U.px,U.py) ;
            //在爆炸向量中添加一员
            EV.addElement(E) ;
        }
        //如果UFO没有被击中，则显示出来，否则，将其从UFO向量中删除
        if (U.active())
            U.draw() ;
        else
            UV.removeElementAt(i) ;
        //如果有任何一个UFO成功着陆，则玩家失败
        if ((U.py - U.h/2) <=0) {
            game_over = true ;
            display_game_over() ;
            return ;
        }
    }
    //如果导弹发射架移动了，重画发射架
    if (L.has_moved() || ((M.py-M.h) < (L.py+L.h)) || (! M.active()) )

```

```

        L.draw() ;
        //如果导弹的active属性值为true，则重画导弹
        if (M.active() == true) M.draw() ;
        //万一CPU速度太快，要使循环维持在20ms以上
        ti = System.currentTimeMillis() - ti ;
        ti = 20 - ti ;
        ti = ti > 0 ? 10 + ti : 10 ;
        Thread.yield() ;
        //处理线程sleep函数的异常
        try {Thread.sleep(ti);}
        catch (InterruptedException e) { } ;
        //每100次循环重新绘制一次
        if ((count = ++count % 500) == 0) {
            repaint() ;
        }
    }
}

//显示玩家的得分
public void display_score() {
    Graphics bkd_g = backdrop.getGraphics();
    bkd_g.clipRect(window_size.width/2, 0, window_size.width/2, 40);
    bkd_g.drawImage(bgimg,0,0,window_size.width,window_size.height,this) ;
    bkd_g.setColor(Color.red) ;
    bkd_g.setFont(font) ;
    String aux = score > 9 ? "" : "0" ;
    bkd_g.drawString(aux+score, window_size.width - 60,30) ;
    bkd_g.dispose() ;
    Graphics bg = buffer.getGraphics() ;
    bg.clipRect(0, 0, window_size.width, 40);
    bg.drawImage(backdrop,0,0,this) ;
    bg.dispose() ;
    Graphics g = getGraphics() ;
    g.clipRect(0, 0, window_size.width, 40);
    g.drawImage(buffer,0,0,this) ;
    g.dispose() ;
}
}

//游戏结束时进行提示
public void display_game_over() {
    set_say_font(font) ;
    set_say_mode(CENTER) ;
    set_say_style(SHADOW) ;
    set_say_pos(10,80) ;
}

```

```

say("GAME OVER");
set_say_font(font_s);
set_say_style(NORMAL);
say("(click to start)");
repaint();
try {Thread.sleep(500);} catch (InterruptedException e) {};
}

//处理鼠标移动事件
public boolean mouseMove(Event e, int x, int y) {
    //返回鼠标所在位置的X坐标
    mouse_x = x;
    return true;
}

//处理鼠标的按下事件
public boolean mouseDown(Event e, int x, int y) {
    //游戏结束时所做的相应处理
    if (game_over) {
        game_over = false;
        if (game != null) {
            game.stop();
            game = null;
        }
        NU = 1;
        score = 0;
        M.active(false);
        UV.removeAllElements();
        EV.removeAllElements();
        //新建一个线程
        game = new Thread(this);
        game.setPriority(Thread.MIN_PRIORITY);
        //线程启动
        game.start();
        buf_g.dispose();
        return true;
    }
    //如果游戏没有结束并且导弹没有被发射，则发射导弹
    if (M != null && !M.active()) {
        missile_launch.stop();
        missile_launch.play();
        M.set_pos(L.px,L.py);
        M.active(true);
    }
}

```

```

        return true;
    }
    //将缓冲绘制到屏幕上
    public void paint(Graphics g) {
        if (buffer != null) g.drawImage(buffer, 0, 0, this);
    }
    //使动画更加流畅
    public void update(Graphics g) {
        paint(g);
    }
    //获取这个Applet的Frame对象
    public Frame getFrame(Component c) {
        while( c != null && !(c instanceof java.awt.Frame) )
            c = c.getParent();
        return (Frame) c;
    }
    //文本显示的常量
    public static final int CENTER = 1 ; // 模式: 居中
    public static final int LEFT = 2 ; // 居左
    public static final int RIGHT = 3 ; // 居右
    public static final int FREE = 0 ; // 居于所给的(x, y)位置上
    public static final int NORMAL = 0 ; // 类型: 正常
    public static final int SHADOW = 1 ; // 带有阴影

    //文本显示变量
    private int say_pos_y = 0 ;
    private int say_pos_x = 0 ;
    private int say_mode = -1 ;
    private int say_style = -1 ;
    private int say_margin = 10 ;
    private Font say_font = null ;

    //文本显示方法
    public void say(String s, int x, int y) {
        set_say_pos(x, y) ;
        say(s) ;
    }
    //文本显示
    public void say(String s) {
        //获取字体的信息
        FontMetrics fm = getFontMetrics(say_font) ;
        //计算x坐标
        switch(say_mode) {
        case CENTER:

```

```

        say_pos_x = (window_size.width - fm.stringWidth(s))/2 ;
        break ;
    case RIGHT:
        say_pos_x = window_size.width - fm.stringWidth(s) - say_margin ;
        break ;
    case LEFT:
    default :
        say_pos_x = say_margin ;
        break ;
    }
    Graphics bg = buffer.getGraphics() ;
    bg.setFont(say_font) ;

    if (say_style == SHADOW) {
        bg.setColor(new Color(150,150,150)) ;
        bg.drawString(s, say_pos_x+2,say_pos_y+1) ;
    }
    //在缓冲内书写字符串
    bg.setColor(Color.white) ;
    bg.drawString(s, say_pos_x,say_pos_y) ;
    //提升相应的 y坐标值
    say_pos_y += (int) (1.2 * fm.getHeight()) ;
    //释放一些资源
    bg.dispose() ;
}
//设置显示模式
public void set_say_mode(int m) {
    say_mode = m ;
}
//设置显示类型
public void set_say_style(int s) {
    say_style = s ;
}
//设置显示所用的字体
public void set_say_font(Font f) {
    say_font = f ;
}
//设置显示的空白边缘
public void set_say_margin(int margin) {
    say_margin = margin ;
}
//设置显示位置的坐标
public void set_say_pos(int x, int y) {

```

```

    say_pos_x = x ;
    say_pos_y = y ;
}
}

//定义Piece类，这是一个基类
class Piece {
    UFO_Attack a ;
    int px,py ; //新位置的x、y坐标
    int opx,opy ; //旧位置的x、y坐标
    int w,h ; //宽度，高度
    int vx,vy ; //x和y方向的速度
    Color c ; //颜色
    boolean active = false ; //活动性
    Image img = null ; //外观
    //位置的设定
    public void set_pos(int x, int y) {
        px = opx = x ;
        py = opy = y ;
    }
    //速度的设置
    public void set_vel(int x,int y) {
        vx = x ;
        vy = y ;
    }
    //高度和宽度的设置
    public void set_size(int x,int y) {
        w = x ;
        h = y ;
    }
    //颜色的设置
    public void set_color(Color c) {
        this.c = c ;
    }
    //绘制区域的设置
    public void set_draw_rectangles(Rectangle o, Rectangle n) {
        int sh = a.window_size.height ;
        int x = px - w/2 ;
        int y = (sh - py) - h/2 ;
        int ox = opx - w/2 ;
        int oy = (sh - opy) - h/2 ;
        o.reshape(ox,oy,w,h) ;
        n.reshape(x,y,w,h) ;
    }
}

```

```

}

//获取active属性值
public boolean active() {
    return active ;
}

//设置active属性值
public void active(boolean s) {
    active = s ;
}

//物体间碰撞的监测
public boolean collision(Piece p) {
    int dpx = Math.abs(px - p.px) ;
    int dpy = Math.abs(py - p.py) ;
    if ((dpx < (Math.max(w/2,p.w/2))+1) && (dpy < (Math.max(h/2,p.h/2)+1)))
        return true ;
    return false ;
}

//对象的绘制函数
public void draw() {
    //设置绘制的区域
    set_draw_rectangles(a.paint_area, a.new_area) ;
    //绘制缓冲区
    Graphics bg = a.buffer.getGraphics() ;
    bg.clipRect(a.paint_area.x, a.paint_area.y, w, h);
    bg.drawImage(a.backdrop,0,0,a) ;
    bg.dispose() ;
    //填充缓冲区
    a.buf_g.setColor(c);
    a.buf_g.fillRect(a.new_area.x, a.new_area.y, w, h);
    //使用新的区域
    a.paint_area.add(a.new_area) ;
    //将缓冲绘制到屏幕上
    Graphics g = a.getGraphics() ;
    g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
    g.drawImage(a.buffer, 0, 0, a);
    g.dispose() ;
}

//对象的清除
public void erase() {
    //设置绘制的区域
    set_draw_rectangles(a.paint_area, a.new_area) ;
    //使用新的区域
}

```

```

    a.paint_area.add(a.new_area) ;
    //将背景复制到缓冲中
    Graphics bg = a.buffer.getGraphics() ;
    bg.clipRect(a.paint_area.x, a.paint_area.y, a.paint_area.width, a.paint_area.height);
    bg.drawImage(a.backdrop,0,0,a) ;
    bg.dispose() ;
    ///将缓冲绘制到屏幕上
    Graphics g = a.getGraphics() ;
    g.clipRect(a.paint_area.x,a.paint_area.y,a.paint_area.width,a.paint_area.height);
    g.drawImage(a.buffer,0,0, a);
    g.dispose() ;
}

}

//定义导弹发射架类，继承于Piece类
class Launcher extends Piece {
    //Launcher类的构造函数
    public Launcher (UFO_Attack a) {
        //导弹发射架属性值的初始化
        this.a = a ;
        w = 12 ;
        h = 22 ;
        px = opx = a.window_size.width/2 ;
        py = opy = w/2+1 ;
        active = true ;
        img = a.missile ;
    }
    //导弹发射架的移动函数
    public void move() {
        opx = px ;
        opy = py ;
        int dx = a.mouse_x - px ;
        int abs_dx = Math.abs(dx) ;
        int step = 1 ;
        if (abs_dx > 10)
            step = 5 ;
        else if (abs_dx > 1)
            step = abs_dx/2 ;
        if (dx != 0) {
            px += step*(dx/abs_dx) ;
            if (px < w/2)
                px = w/2 ;
            else if (px > (a.window_size.width - w/2))

```

```

        px = a.window_size.width - w/2 ;
    }
}
//判断导弹发射架是否移动
public boolean has_moved() {
    if ((px - opx) != 0) return true ;
    return false ;
}
//导弹发射架的绘制
public void draw() {
    set_draw_rectangles(a.paint_area, a.new_area) ;
    Graphics bg = a.buffer.getGraphics() ;
    bg.clipRect(a.paint_area.x, a.paint_area.y, w, h);
    bg.drawImage(a.backdrop,0,0,a) ;
    bg.dispose() ;
    //根据导弹的active属性值进行相应的绘制
    if (a.M.active()) {
        //导弹飞行时，发射架外观为一实心矩形
        a.buf_g.setColor(c);
        a.buf_g.fillRect(a.new_area.x, a.new_area.y, w, h);
    }
    else {
        //否则，发射架外观为一竖立的导弹
        bg = a.buffer.getGraphics() ;
        bg.clipRect(a.new_area.x, a.new_area.y, w, h);
        bg.drawImage(img,a.new_area.x,a.new_area.y,a) ;
        bg.dispose() ;
        // bg = null ;
    }
    //使用新的区域
    a.paint_area.add(a.new_area) ;
    //将缓冲绘制到屏幕上
    Graphics g = a.getGraphics() ;
    g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
    g.drawImage(a.buffer, 0, 0, a);
    g.dispose() ;
}
}

//定义Missile类，继承于Piece类
class Missile extends Piece {
    //导弹类的构造函数
    public Missile (UFO_Attack a) {

```

```

//导弹属性值的初始化
this.a = a ;
px = opx = 0 ;
py = opy = 0 ;
vx = 0 ;
vy = 7 ;
w = 12 ;
h = 22 ;
active = false ;
img = a.missile ;
}

//对象的移动函数
public void move() {
    opx = px ;
    opy = py ;
    px = a.L.px ;
    //使移动的速度更加实际化
    int dx = px - opx ;
    int nvy = vy*vy - dx*dx ;
    if (nvy > 0) nvy = (int) Math.sqrt(nvy) ; // Should exceptions
    if (nvy < 1) nvy = 1 ;
    py += nvy ;
    if (py > a.window_size.height + 2*h) active = false ;
}

int seq = 0 ;
//导弹对象的绘制
public void draw() {
    //设置绘制区域
    set_draw_rectangles(a.paint_area, a.new_area) ;
    //先将变化绘制到缓冲中
    Graphics bg = a.buffer.getGraphics() ;
    bg.clipRect(a.paint_area.x, a.paint_area.y, w, h);
    bg.drawImage(a.backdrop,0,0,a) ;
    bg.dispose() ;
    //由于导弹所用的图片是一个序列图，所以要从中剪切，然后再使用
    //根据seq的值来显示此序列图的一部分
    seq = ++seq % 1 ;
    int dx = px - opx ;
    seq = 0 ;
    if (dx > 0)
        seq = 1 ;
    else if (dx < 0)

```

```

    seq = 2 ;
    //将变化绘制到缓冲上
    bg = a.buffer.getGraphics() ;
    bg.clipRect(a.new_area.x, a.new_area.y, w, h);
    bg.drawImage(img,a.new_area.x-w*seq,a.new_area.y,a) ;
    bg.dispose() ;
    //使用新的区域
    a.paint_area.add(a.new_area) ;
    //将缓冲绘制到屏幕上
    Graphics g = a.getGraphics() ;
    g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
    g.drawImage(a.buffer, 0, 0, a);
    g.dispose() ;
}
}

//定义UFO类，继承于Piece类
class UFO extends Piece {
    //UFO类的构造函数
    public UFO (UFO_Attack a) {
        //UFO属性值的初始化
        this.a = a ;
        vx = (Math.random() > 0.5 ? 1 : -1) ;
        vy = -2 ;
        w = 20 ;
        h = 8 ;
        int aw = a.window_size.width ;
        px = opx = (int) (w/2+1 + (aw-w-2)* Math.random()) ;
        py = opy = a.window_size.height + h/2 + 1 ;
        active = true ;
        img = a.ufostrip ;
    }
    //UFO对象的移动函数
    public void move() {
        opx = px ;
        opy = py ;
        px += vx ;
        py += vy ;
        if (py < -h/2) active = false ;
        if ((px <= w/2) ||
            (px >= (a.window_size.width - w/2)) ||

```

```

        (Math.random() > 0.96) {
            vx = -vx ;
        }
    }
    int seq = 0 ;
    int seq2 = 0 ;
    //UFO对象的绘制函数
    public void draw() {
        //设置对象的绘制区域
        set_draw_rectangles(a.paint_area, a.new_area) ;
        //清除旧的图像
        Graphics bg = a.buffer.getGraphics() ;
        bg.clipRect(a.paint_area.x, a.paint_area.y, w, h);
        bg.drawImage(a.backdrop,0,0,a) ;
        bg.dispose() ;
        //由于UFO用的图片是一个序列图，要进行剪切再使用
        //根据seq2的值选择相应的部分
        if ((++seq2 % 4) == 0) seq = ++seq % 4 ;
        //绘制新的区域到缓冲中
        bg = a.buffer.getGraphics() ;
        bg.clipRect(a.new_area.x, a.new_area.y, w, h);
        bg.drawImage(img,a.new_area.x-w*seq,a.new_area.y,a) ;
        bg.dispose() ;
        //使用新的区域
        a.paint_area.add(a.new_area) ;
        //将缓冲绘制到屏幕上
        Graphics g = a.getGraphics() ;
        g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
        g.drawImage(a.buffer, 0, 0, a);
        g.dispose() ;
    }
}

//定义爆炸类，继承于Piece类
class Explosion extends Piece {
    //Explosion类的构造函数
    public Explosion (UFO_Attack a, int x, int y) {
        //爆炸对象属性值的初始化
        this.a = a ;
        w = 30 ;
        h = 30 ;
        px = opx = x ;
        py = opy = y ;
    }
}

```

```

        active = true ;
        img = a.missile_explosion ;
    }
    int seq = 0 ;
    int seq2 = 0 ;
    //爆炸对象的绘制函数
    public void draw() {
        //设置绘制的区域
        set_draw_rectangles(a.paint_area, a.new_area) ;
        //清除旧的图像
        Graphics bkd_g = a.backdrop.getGraphics();
        bkd_g.clipRect(a.paint_area.x, a.paint_area.y, w, h);
        bkd_g.drawImage(a.bgimg,0,0,a.window_size.width,a.window_size.height,a) ;

        //由于爆炸用的图片是一个序列图，要进行剪切再使用
        //根据seq2的值选择相应的部分
        if ((++seq2 % 4) == 0) seq = ++seq % 5 ;
        //爆炸图最后的部分显示后，将active属性值设为false，并将其清除
        if (seq == 4) active = false ;
        // 将新的区域绘制到缓冲中
        bkd_g.clipRect(a.new_area.x, a.new_area.y, w, h);
        bkd_g.drawImage(img,a.new_area.x-w*seq,a.new_area.y,a) ;
        bkd_g.dispose() ;
        //将变化绘制到缓冲中
        Graphics bg = a.buffer.getGraphics() ;
        bg.clipRect(a.new_area.x,a.new_area.y,w,h);
        bg.drawImage(a.backdrop,0,0,a) ;
        bg.dispose() ;
        //将缓冲绘制到屏幕上
        Graphics g = a.getGraphics() ;
        g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
        g.drawImage(a.buffer, 0, 0, a);
        g.dispose() ;
    }
    //爆炸对象的清除
    public void erase() {
        //设置绘制的区域
        set_draw_rectangles(a.paint_area, a.new_area) ;
        //清除旧的图像
        Graphics bkd_g = a.backdrop.getGraphics();
        bkd_g.clipRect(a.paint_area.x, a.paint_area.y, w, h);
        bkd_g.drawImage(a.bgimg,0,0,a.window_size.width,a.window_size.height,a) ;
        bkd_g.dispose() ;
    }
}

```

```

    // 对缓冲和屏幕执行同样的操作
    super.erase();
}
}

```



案例4 制作一个MP3播放器

案例运行效果与操作

本案例是一个用java实现MP3文件播放的例子。借用winnap的skin，程序运行后，播放界面如图10-13所示，读者可参见www.javazoom.net。



图10-13 运行界面

制作要点

1. javax.sound.sampled的用法
2. javazoom.jlGui的使用

步骤详解

1. 构造界面。

定义各种参数、界面、按钮及其监听，加载播放皮肤，本例中通过定义的类SkinLoader来实现：

```

if (Skin != null)
thePath = Skin;
else thePath = config.getDefaultSkin();
SkinLoader skl = new SkinLoader(thePath);
Try {
    skl.loadImages();
} catch (Exception e)
{
    trace(0,getClass().getName(),"Can't load skin : "+e.getMessage());
    System.exit(0);
}

```

设置按钮并监听，如“Play”：

```
acPlay = new activeComponent(releasedImage[1],pressedImage[1],AWTEvent.MOUSE_EVENT_MASK);
acPlay.setLocation(panelLocation[l++],panelLocation[l++]);
add(acPlay);
acPlay.setActionCommand("Play");
acPlay.addActionListener(this);
```

2. 定义播放对象。

利用JavaSound API播放当前选中的文件。首先要通过AudioSystem类获得一个AudioInputStream。然后，利用AudioInputStream的getFormat()获知音频数据的格式：

```
m_audioInputStream = AudioSystem.getAudioInputStream(file);
m_audioFileFormat = AudioSystem.getAudioFileFormat(file);
```

播放对象为URL方式时：

```
m_audioInputStream = AudioSystem.getAudioInputStream(url);
m_audioFileFormat = AudioSystem.getAudioFileFormat(url);
```

3. 转换格式。

因为MP3格式文件本身是压缩的，再进行播放之前，首先要解码转换，分三个步骤，

- (1) 创建一个解压缩结果的定制AudioFormat（PCM编码），但保留和原压缩流一样的取样率、通道信息等。
- (2) 创建一个AudioInputStream把原来的AudioInputStream转换成新的AudioFormat格式。
- (3) 获得一个处理解码后格式的SourceDataLine。

```
AudioFormat sourceFormat = m_audioInputStream.getFormat();
trace(1,getClass().getName(), "Source format : ", sourceFormat.toString());
AudioFormat targetFormat = new AudioFormat( AudioFormat.Encoding.PCM_SIGNED,
sourceFormat.getSampleRate(),16, sourceFormat.getChannels(),sourceFormat.getChannels() *
2, sourceFormat.getSampleRate(),false);
trace(1,getClass().getName(), "Target format: " + targetFormat);
m_audioInputStream = AudioSystem.getAudioInputStream(targetFormat, m_audioInputStream);
AudioFormat audioFormat = m_audioInputStream.getFormat();
trace(1,getClass().getName(), "Create Line : ", audioFormat.toString());
DataLine.Info info = new DataLine.Info(SourceDataLine.class, audioFormat, AudioSystem
.NOT_SPECIFIED);
m_line = (SourceDataLine) AudioSystem.getLine(info);
```

getLine()方法的返回值是一个与参数中指定的AudioFormat兼容的SourceDataLine。如果不能获得兼容的SourceDataLine，getLine()返回null。

创建和填充一个DataLine.Info结构，调用AudioSystem.getLine()方法，将info结构传递给AudioSystem类工厂。

4. 播放文件。

准备播放，需要判断是否播放空文件。

```
if (m_line == null)
{
    createLine();
    trace(l.getClassName(), "Create Line OK ");
    openLine();
}
```

然后要判断播放状态，对应相应的状态采取相应的动作，例如暂停/恢复。

```
if (playerState == PLAY)
{
    theSoundPlayer.pausePlayback();
    playerState = PAUSE;
    offScreenGraphics.drawImage(IconsImage[1], iconsLocation[0], iconsLocation[1], this);
    offScreenGraphics.drawImage(IconsImage[4], iconsLocation[2], iconsLocation[3], this);
    repaint();
}
```

其实，准备好AudioInputStream和SourceDataLine之后，播放器实际上是用一个循环从AudioInputStream读取数据，然后写入到SourceDataLine，完成了整个的播放功能，当然其中还有很多情况的处理。

程序源代码与解释

下面是部分源码和解释。

```
//构造器
public Player(String Skin, int loglevel, String logfile)
{
    super(new Frame());
    Debug dbg = Debug.getInstance();
    if ( (logfile != null) && (!logfile.equals("")) )
    {
        dbg.init(logfile, loglevel);           //日志特性
    }
    else dbg.setLevel(loglevel);
    config = Config.getInstance();           //配置特性
    if (Skin != null) thePath = Skin;        //皮肤特性
    else thePath = config.getDefaultSkin();
```

```

SkinLoader skl = new SkinLoader(thePath);
try
{
    skl.loadImages();
} catch (Exception e)
{
    trace(0,getClass().getName(),"Can't load skin : "+e.getMessage());
    System.exit(0);
}
imMain = skl.getImage(theMain);
imButtons = skl.getImage(theButtons);
imTitleBar = skl.getImage(theTitleBar);
imText = skl.getImage(theText);
imMode = skl.getImage(theMode);
imNumbers = skl.getImage(theNumbers);
imVolume = skl.getImage(theVolume);
imIcons = skl.getImage(theIcons);
imPosBar = skl.getImage(thePosBar);
WinHeight = imMain.getHeight(this);
WinWidth = imMain.getWidth(this);           //设置皮肤
setSize(WinWidth,WinHeight);
setLocation(OrigineX,OrigineY);
setBackground(Color.black);
show();          //输出
offScreenImage = createImage(WinWidth,WinHeight);
offScreenGraphics = offScreenImage.getGraphics();
offScreenGraphics.drawImage(imMain,0,0,this);
PlaylistFactory plf = PlaylistFactory.getInstance();
playlist = plf.playlist();
playlist.load(config.getPlaylistFilename());
theSoundPlayer = new BasicPlayer(this); //播放列表
readPanel(releasedImage,releasedPanel,pressedImage,pressedPanel,imButtons);
setButtonsPanel(); //按键
readPanel(releasedVolumeImage,releasedVolumePanel,pressedVolumeImage,pressedVolumePanel,imVolume);
setVolumePanel();      //音量/平衡
readPanel(releasedTitleIm,releasedTitlePanel,pressedTitleIm,pressedTitlePanel,imTitleBar);
setTitleBarPanel(); //标题条
readPanel(releasedExitIm,releasedExitPanel,pressedExitIm,pressedExitPanel,imTitleBar);
setExitPanel();      //退出
//模式
readPanel(activeModeImage,activeModePanel,passiveModeImage,passiveModePanel,imMode);
offScreenGraphics.drawImage(passiveModeImage[0], stereoLocation[0],

```

```

stereoLocation[1], this);
    offScreenGraphics.drawImage(passiveModeImage[1], monoLocation[0],
monoLocation[1], this);
    //文件
    sampleRateClearImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
sampleRateClearText)).getBanner();
    bitsRateClearImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
bitsRateClearText)).getBanner();
    clearImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
clearText)).getBanner(0,0,155,6);
    titleImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
titleText)).getBanner(0,0,155,6);
    offScreenGraphics.drawImage(titleImage, titleLocation[0], titleLocation[1], this);
    //数字
    for (int h=0;h<numberIndex.length();h++)
    {
        timeImage[h] = (new taftb(numberIndex, imNumbers, numberWidth,
numberHeight, 0, ""+numberIndex.charAt(h))).getBanner();
    }
    //图标
    readPanel(IconsImage,iconsPanel,null,null,imIcons);
    offScreenGraphics.drawImage(IconsImage[2], iconsLocation[0], iconsLocation[1],
this);
    //进度条
    readPanel(releasedPosIm,releasedPosPanel,pressedPosIm,pressedPosPanel,imPosBar);
    setPosBarPanel();
    repaint();
}
//面板特性
private void readPanel(Image[] releasedImage, int[] releasedPanel,
Image[] pressedImage, int[] pressedPanel, Image imPanel)
{
    /*.....代码省略*/
}
//按钮 Panel的实例，加入窗口并监听
private void setButtonsPanel()
{
    /*.....代码省略*/
}
//标题 Panel的实例，加入窗口并监听
private void setTitleBarPanel()
{
    /*.....代码省略*/
}
// Exit Panel的实例，加入窗口并监听
private void setExitPanel()
{
    /*.....代码省略*/
}
//声音/平衡面板的实例，加入窗口并监听

```

```

private void setVolumePanel()
{ /*.....代码省略*/ }
// PosBar Panel 的实例，加入窗口并监听
private void setPosBarPanel()
{ /*.....代码省略*/ }
//设置播放的音乐文件，随时可以播放
private void setCurrentSong(PlaylistItem pli)
{
    int playerStateMem = playerState;
    if ( (playerState == PAUSE) || (playerState == PLAY) )
    {
        theSoundPlayer.stopPlayback();
        playerState = STOP;
        secondsAmount = 0;
        acPosBar.setLocation(posBarBounds[0],posBarLocation[1]);
        offScreenGraphics.drawImage(IconsImage[2], iconsLocation[0],
iconsLocation[1], this);
        offScreenGraphics.drawImage(IconsImage[4], iconsLocation[2],
iconsLocation[3], this);
    }
    playerState = OPEN;//设置播放状态
/*
*.....代码省略
*收集当前信息，为播放准备并重画界面
*/
    //点击开始播放
    if ((playerStateMem == PLAY) || (playerStateMem == PAUSE))
    {
        acPlay.fireEvent();
    }
}
// 定义BasicPlayerListener接口
//BasicPlayerListene 处理时间事件
public void updateCursor(int secondsAmount, int total)
{
    //显示已播放时间
    /*.....代码省略*/
    //更新进度条位置
    /*.....代码省略*/
}
//处理结束媒体文件事件，首先判断文件是否到达尾部
public void updateMediaState(String state)

```

```

{
    /*.....代码省略*/
}

public void updateMediaData(byte[] data)      //频谱分析
{
}

//事件处理
public void actionPerformed( ActionEvent e )
{
    //      查找处理
    if (e.getActionCommand().equals("Seek"))
    {
        /*.....代码省略*/
    }
    //      合成声音
    else if (e.getActionCommand().equals("Volume"))
    {
        /*.....代码省略*/
    }
    //      合成平衡
    else if (e.getActionCommand().equals("Balance"))
    {
        /*.....代码省略*/
    }
    //选择要装载的文件名或URL
    else if (e.getActionCommand().equals("Eject"))
    {
        if ((playerState == PLAY) || (playerState == PAUSE))
        {
            theSoundPlayer.stopPlayback();
            playerState = STOP;
        }
        if ((playerState == INTT) || (playerState == STOP) || (playerState == OPEN))
        {
            System.gc();
            PlaylistItem pli = null;
            // 本地文件处理
            if (acEject.getMouseButton() == MouseEvent.BUTTON1_MASK)
            {
                FileDialog FD = new FileDialog(new Frame(),"Open file",FileDialog.LOAD);
                FD.show();
                if (FD.getFile() != null) pli = new PlaylistItem(FD.getFile(), FD.getDirectory()+
FD.getFile(), -1, true);
            }
            //非本地文件处理.
            else if (acEject.getMouseButton() == MouseEvent.BUTTON3_MASK)
            {
                UrlDialog UD = new UrlDialog("Open location",100,100,280,120,

```

```

config.getLastURL());
        UD.show();
        if (UD.getFile() != null) pli = new PlaylistItem(UD.getFile(), UD.getURL(), -1, false);
    }
    this.setCurrentSong(pli);
}
offScreenGraphics.drawImage(IconsImage[2], iconsLocation[0], iconsLocation[1], this);
offScreenGraphics.drawImage(IconsImage[4], iconsLocation[2], iconsLocation[3], this);
repaint();
}
else if (e.getActionCommand().equals("Play")) //播放
{
    if (playerState == PAUSE) //处理暂停状态
    {
        theSoundPlayer.resumePlayback();
        playerState = PLAY;
        offScreenGraphics.drawImage(IconsImage[0], iconsLocation[0],
iconsLocation[1], this);
        offScreenGraphics.drawImage(IconsImage[3], iconsLocation[2],
iconsLocation[3], this);
        repaint();
    }
    else if (playerState == PLAY) //处理播放状态
    {
        theSoundPlayer.stopPlayback();
        playerState = PLAY;
        secondsAmount = 0;
        offScreenGraphics.drawImage(timeImage[0], minuteDLocation[0],
minuteDLocation[1], this);
        offScreenGraphics.drawImage(timeImage[0], minuteLocation[0],
minuteLocation[1], this);
        offScreenGraphics.drawImage(timeImage[0], secondDLocation[0],
secondDLocation[1], this);
        offScreenGraphics.drawImage(timeImage[0], secondLocation[0],
secondLocation[1], this);
        repaint();
    }
    if (currentFileOrURL != null) //判断播放文件是否为空
    {
        try
        {
            if (currentIsFile == true) theSoundPlayer.setDataSource(openFile
(currentFileOrURL));
        }
    }
}

```

```

        else theSoundPlayer.setDataSource(new URL(currentFileOrURL));
    } catch (Exception ex)
    {
        trace(0,getClass().getName(),"Cannot read file : "+currentFileOrURL+","
        "+ex.getMessage());
    }
    theSoundPlayer.startPlayback(); //开始播放
}
}
else if ((playerState == STOP) || (playerState == OPEN)) //处理停止、
打开状态
{
    if (currentFileOrURL != null)
    { //播放文件不为空时
        try
        {
            if (currentIsFile == true) theSoundPlayer.setDataSource(openFile
(currentFileOrURL));
            else theSoundPlayer.setDataSource(new URL(currentFileOrURL));
        } catch (UnsupportedAudioFileException uafe)
        {
            trace(0,getClass().getName(),"Stream error :" +currentFileOrURL+",
"+ufae.getMessage());
        }
        catch (LineUnavailableException lue)
        {
            trace(0,getClass().getName(),"Stream error :" +currentFileOrURL+",
"+lue.getMessage());
        }
        catch (IOException ex)
        {
            trace(0,getClass().getName(),"Stream error :" +currentFileOrURL+",
"+ex.getMessage());
        }
        theSoundPlayer.startPlayback(); //开始播放
        theSoundPlayer.setGain((double)gainValue / (double)maxGain);
        theSoundPlayer.setPan((float)balanceValue);
        sampleRateImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
""+Math.round((theSoundPlayer.getAudioFormat()).getSampleRate()/1000))).getBanner();
        bitsRateImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
""+Math.round((theSoundPlayer.getAudioFormat()).getFrameSize()*8*(theSoundPlayer.getAudioFormat()
.getFrameRate()/1000))).getBanner());
    }
}

```

```

offScreenGraphics.drawImage(sampleRateImage, sampleRateLocation[0],
sampleRateLocation[1], this);
offScreenGraphics.drawImage(bitsRateImage, bitsRateLocation[0], bitsRateLocation[1],
this);
if ((theSoundPlayer.getAudioFormat()).getChannels() == 2)
{
    offScreenGraphics.drawImage(activeModeImage[0], stereoLocation[0],
stereoLocation[1], this);
}
else if ((theSoundPlayer.getAudioFormat()).getChannels() == 1)
{
    offScreenGraphics.drawImage(activeModeImage[1], monoLocation[0],
monoLocation[1], this);
}
double lenghtInSecond = 0.0; //播放时间长度
if ((theSoundPlayer.getAudioFileFormat().getFrameLength() != -1) lenghtInSecond =
((theSoundPlayer.getAudioFileFormat().getFrameLength()*theSoundPlayer.getAudioFormat().getFrameSize()*8)/
((theSoundPlayer.getAudioFormat().getFrameSize()*8*(theSoundPlayer.getAudioFormat().getFrameRate())));
else lenghtInSecond = ((theSoundPlayer.getAudioFileFormat().getByteLength()*8)/
(theSoundPlayer.getAudioFormat().getFrameSize()*8*(theSoundPlayer.getAudioFormat().getFrameRate()));
int minutes = (int) (lenghtInSecond/60);
int seconds = (int) (lenghtInSecond - (minutes*60));
titleText = currentSongName.toUpperCase();
if (seconds >= 10) titleText += titleText + " ("+minutes+":"+seconds+ ")";
else titleText += titleText + " ("+minutes+":0"+seconds+ ")";
titleImage = (new taftb(fontIndex, imText, fontWidth, fontHeight, 0,
titleText)).getBanner(0,0,155,6);
//输出
offScreenGraphics.drawImage(clearImage, titleLocation[0], titleLocation[1], this);
offScreenGraphics.drawImage(titleImage, titleLocation[0], titleLocation[1], this);
offScreenGraphics.drawImage(timeImage[0], minuteDLocation[0], minuteDLocation[1],
this);
offScreenGraphics.drawImage(timeImage[0], minuteLocation[0], minuteLocation[1], this);
offScreenGraphics.drawImage(timeImage[0], secondDLocation[0], secondDLocation[1],
this);
offScreenGraphics.drawImage(timeImage[0], secondLocation[0], secondLocation[1], this);
playerState = PLAY; //设置播放状态
offScreenGraphics.drawImage(IconsImage[0], iconsLocation[0], iconsLocation[1], this);
offScreenGraphics.drawImage(IconsImage[3], iconsLocation[2], iconsLocation[3], this);
repaint(); //重画
trace(1,getClass().getName(),titleText);

```

```

        if ((theSoundPlayer.getAudioFileFormat()).getFrameLength() != -1) trace(l,getClass()
        .getName(),"Frames = "+(theSoundPlayer.getAudioFileFormat()).getFrameLength());
        trace(l,getClass().getName(),"FrameRate (Hz) = "+theSoundPlayer
        .getAudioFormat().getFrameRate());
        trace(l,getClass().getName(),"FrameSize (bits) = "+theSoundPlayer.getAudioFormat()
        .getFrameSize()*8);
    }
}
//暂停/恢复
else if (e.getActionCommand().equals("Pause"))
{
    if (playerState == PLAY)           //处理播放状态
    {
        theSoundPlayer.pausePlayback(); //暂停播放
        playerState = PAUSE;          //设置为暂停状态
        offScreenGraphics.drawImage(IconsImage[1], iconsLocation[0],
        iconsLocation[1], this);
        offScreenGraphics.drawImage(IconsImage[4], iconsLocation[2],
        iconsLocation[3], this);
        repaint();
    }
    else if (playerState == PAUSE)     // 暂停状态
    {
        theSoundPlayer.resumePlayback(); //恢复播放
        playerState = PLAY;           //设置为播放状态
        offScreenGraphics.drawImage(IconsImage[0], iconsLocation[0],
        iconsLocation[1], this);
        offScreenGraphics.drawImage(IconsImage[3], iconsLocation[2],
        iconsLocation[3], this);
        repaint(); //重画
    }
}
//停止播放
else if (e.getActionCommand().equals("Stop"))
{
    if ((playerState == PAUSE) || (playerState == PLAY))
    {
        theSoundPlayer.stopPlayback(); //停止播放
        playerState = STOP;          //设置为停止状态
        secondsAmount = 0;
        acPosBar.setLocation(posBarBounds[0],posBarLocation[1]);
    }
}

```

```

        offScreenGraphics.drawImage(IconsImage[2], iconsLocation[0],
iconsLocation[1], this);
        offScreenGraphics.drawImage(IconsImage[4], iconsLocation[2],
iconsLocation[3], this);
        repaint();
    }
}

//下一曲，从播放列表中取得下一首
else if (e.getActionCommand().equals("Next"))
{
    /*.....代码省略*/
}
//前一曲，从播放列表中取得前一首
else if (e.getActionCommand().equals("Previous"))
{
    /*.....代码省略*/
}
//退出
else if (e.getActionCommand().equals("Exit"))
{
    if ((playerState == PAUSE) || (playerState == PLAY))
    {
        theSoundPlayer.stopPlayback(); //停止播放
    }
    dispose();
    System.gc();
    System.exit(0);
}
//按住标题栏拖动窗口
else if (e.getActionCommand().equals("TitleBar"))
{
    if (acTitleBar.isMousePressed() == false) FirstDrag = true;
    else
    {
        int DeltaX = 0;
        int DeltaY = 0;
        if (FirstDrag == false)
        {
            DeltaX = acTitleBar.getMouseX() - XDrag;
            DeltaY = acTitleBar.getMouseY() - YDrag;
            XDrag = acTitleBar.getMouseX() - DeltaX;
            YDrag = acTitleBar.getMouseY() - DeltaY; //获取鼠标位置
            OrigineX = OrigineX + DeltaX;
            OrigineY = OrigineY + DeltaY;
            setLocation(OrigineX, OrigineY);
        }
    }
}

```

```
        else
        {
            FirstDrag = false;
            XDrag = acTitleBar.getMouseX();      //标题条位置
            YDrag = acTitleBar.getMouseY();
        }
    }
}
```

本 章 小 结

本章列举了几个综合性的案例，体现了Java知识在编程中的综合应用。Java是一种纯面向对象的程序设计语言，具有许多优良特性，如高性能、跨平台性、可移植性、动态性、稳定性、安全性、多线程等。Java本身博大精深，并不是几个例子能说清楚的。