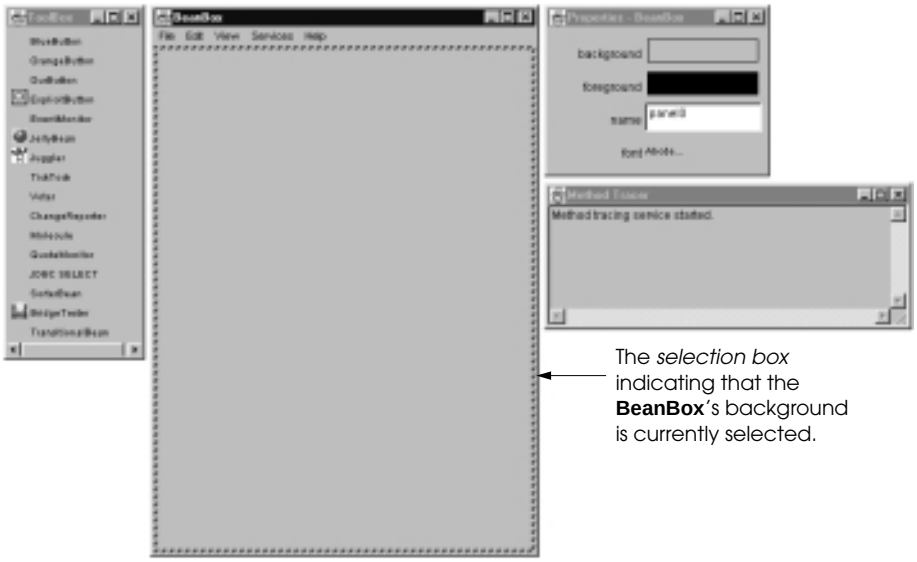
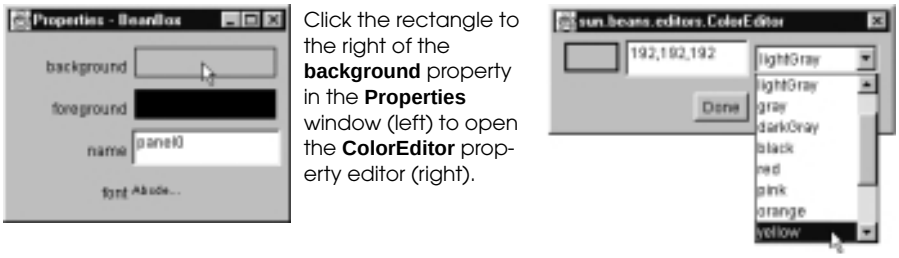


Fig. 25.1 The **ToolBox**, **BeanBox**, **Properties** and **Method Tracer** windows of the **BeanBox**.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company



Click the rectangle to the right of the **background** property in the **Properties** window (left) to open the **ColorEditor** property editor (right).

Fig. 25.2 Changing the **background** property for the **BeanBox**.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

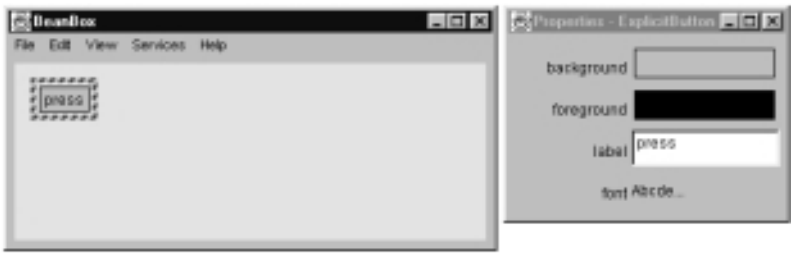


Fig. 25.3 The **BeanBox** window with the **`ExplicitButton`** bean selected.

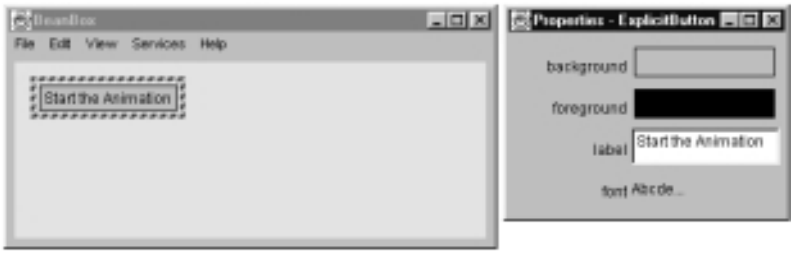


Fig. 25.4 The **`ExplicitButton`** with its new label.

Fig. 25.5 The move cursor.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

Fig. 25.6 The **`ExplicitButton`** after moving.

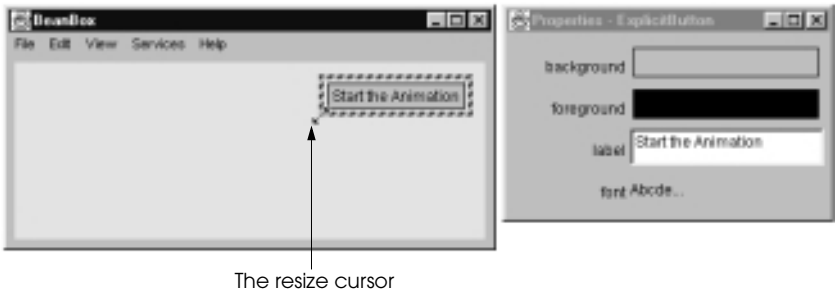


Fig. 25.7 The resize cursor.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

Fig. 25.8 The **`ExplicitButton`** after resizing.

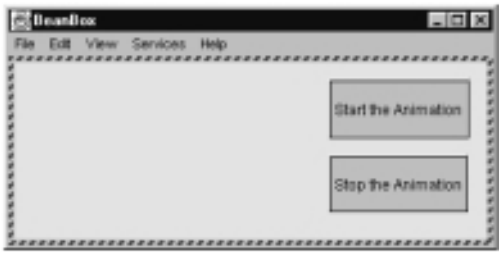


Fig. 25.9 The **BeanBox** window with two **`ExplicitButton`** beans.

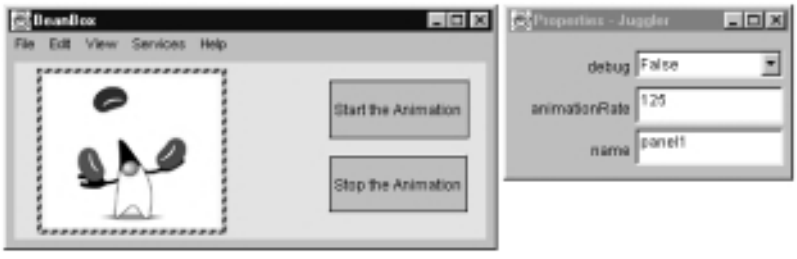


Fig. 25.10 The **BeanBox** window with the new **`Juggler`** bean.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

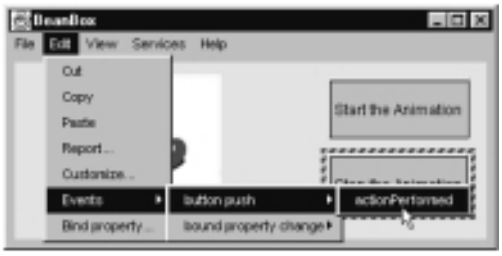


Fig. 25.11 Selecting the button-push event for an **`ExplicitButton`**.

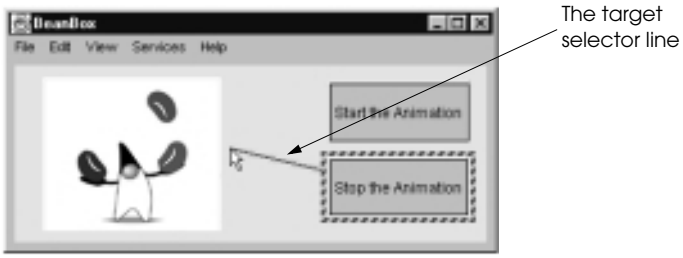


Fig. 25.12 The target selector line.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

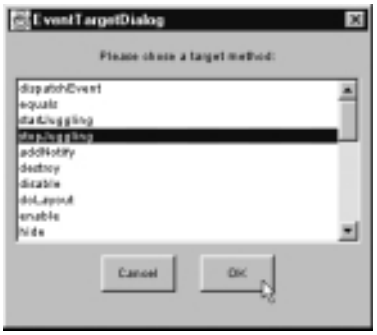


Fig. 25.13 The **EventTargetDialog** window displays the list of methods that can be called on the target of an event.

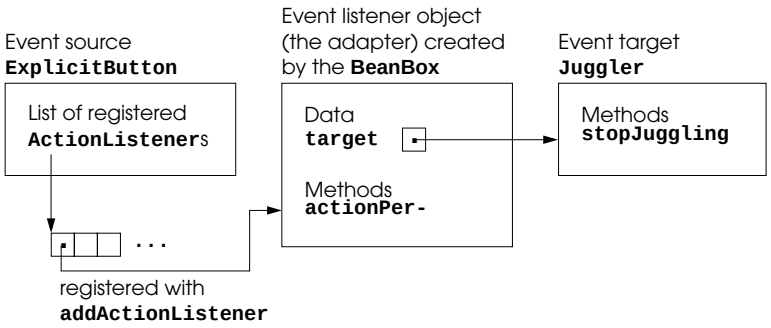


Fig. 25.14 The interaction between the **`ExplicitButton`** and the **`Juggler`**.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

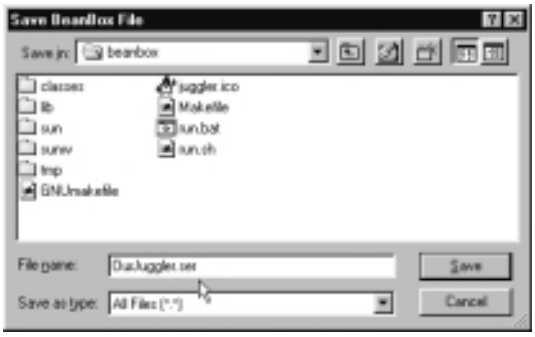


Fig. 25.15 The **Save BeanBox File** dialog.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

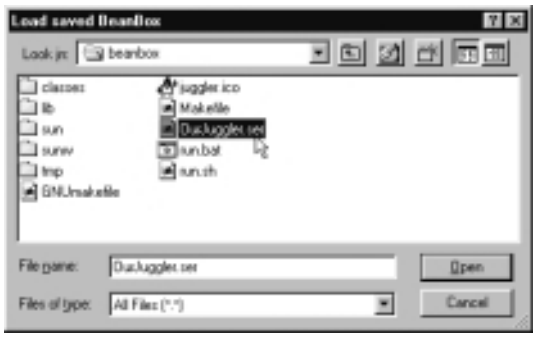


Fig. 25.16 The **Load saved BeanBox** dialog.

Fig. 25.17 The **Make an Applet** dialog.

Fig. 25.18 The **Choose JAR File** dialog.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

Fig. 25.19 The **Make an Applet** dialog after changing the default file name.

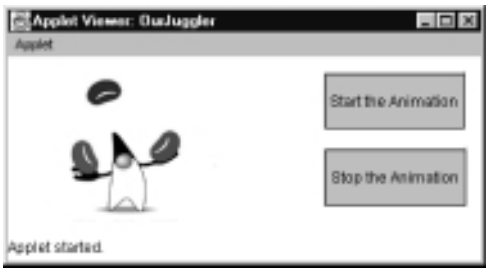


Fig. 25.20 The `OurJuggler` applet running in `appletviewer`.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
<html>
<head>
<title>Test page for OurJuggler as an APPLET</Title>
</head>
<body>
<h1>Test for OurJuggler as an APPLET</h1>
This is an example of the use of the generated
OurJuggler applet.  Notice the Applet tag requires several
archives, one per JAR used in building the Applet
<p>
<applet
    archive="./OurJuggler.jar,./support.jar
        ,./juggler.jar
        ,./buttons.jar
    "
    code="OurJuggler"
    width=382
    height=150
>
Trouble instantiating applet OurJuggler!!
</applet>
```

Fig. 25.21 The **OurJuggler.html** file generated by the **BeanBox**.

```
// Fig. 25.22: LogoAnimator.java
// Animation bean
package jhtp3beans;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

```

Fig. 25.22 The **LogoAnimator** as a stand-alone application (part 1 of 3).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
11  public class LogoAnimator extends JPanel
12        implements ActionListener, Serializable {
13     protected ImageIcon images[];
14     protected int totalImages = 30,
15                   currentImage = 0,
16                   animationDelay = 50; // 50 millisecond delay
17     protected Timer animationTimer;
18
19     public LogoAnimator()
20     {
21        setSize( getPreferredSize() );
22
23        images = new ImageIcon[ totalImages ];
24
25        URL url;
26
27        for ( int i = 0; i < images.length; ++i ) {
28           url = getClass().getResource(
29                    "deitel" + i + ".gif" );
30           images[ i ] = new ImageIcon( url );
31        }
32
33        startAnimation();
34     }
35
36     public void paintComponent( Graphics g )
37     {
38        super.paintComponent( g );
39
40        if ( images[ currentImage ].getImageLoadStatus() ==
41             MediaTracker.COMPLETE ) {
42           g.setColor( getBackground() );
43           g.drawRect(
44              0, 0, getSize().width, getSize().height );
45           images[ currentImage ].paintIcon( this, g, 0, 0 );
46           currentImage = ( currentImage + 1 ) % totalImages;
47        }
48     }
49
50     public void actionPerformed( ActionEvent e )
51     {
52        repaint();
53     }
54
55     public void startAnimation()
56     {
57        if ( animationTimer == null ) {
58           currentImage = 0;
59           animationTimer = new Timer( animationDelay, this );
60           animationTimer.start();
61        }
```

Fig. 25.22 The **`LogoAnimator`** as a stand-alone application (part 2 of 3).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
62          else  // continue from last image displayed
63             if ( ! animationTimer.isRunning() )
64                animationTimer.restart();
65       }
66
67       public void stopAnimation()
68       {
69          animationTimer.stop();
70       }
71
72       public Dimension getMinimumSize()
73       {
74          return getPreferredSize();
75       }
76
77       public Dimension getPreferredSize()
78       {
79          return new Dimension( 160, 80 );
80       }
81
82       public static void main( String args[] )
83       {
84          LogoAnimator anim = new LogoAnimator();
85
86          JFrame app = new JFrame( "Animator test" );
87          app.getContentPane().add( anim, BorderLayout.CENTER );
88
89          app.addWindowListener(
90             new WindowAdapter() {
91                public void windowClosing( WindowEvent e )
92                {
93                   System.exit( 0 );
94                }
95             }
96          );
97
98          app.setSize( anim.getPreferredSize().width + 10,
99                       anim.getPreferredSize().height + 30 );
100         app.show();
101      }
102   }
```

Fig. 25.22 The **`LogoAnimator`** as a stand-alone application (part 3 of 3).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
1  Main-Class: jhtp3beans.LogoAnimator
2
3  Name: jhtp3beans/LogoAnimator.class
4  Java-Bean: True
```

Fig. 25.23 The `manifest.tmp` file for the `LogoAnimator` bean.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
   0 Sun Mar 14 11:36:16 EST 1999 META-INF/
 163 Sun Mar 14 11:36:16 EST 1999 META-INF/MANIFEST.MF
4727 Thu Feb 15 00:37:04 EST 1996 jhtp3beans/deitel0.gif
4858 Thu Feb 15 00:39:32 EST 1996 jhtp3beans/deitel1.gif
4374 Thu Feb 15 00:55:46 EST 1996 jhtp3beans/deitel10.gif
4634 Thu Feb 15 00:56:52 EST 1996 jhtp3beans/deitel11.gif
4852 Thu Feb 15 00:58:00 EST 1996 jhtp3beans/deitel12.gif
4877 Thu Feb 15 00:59:10 EST 1996 jhtp3beans/deitel13.gif
4926 Thu Feb 15 01:00:20 EST 1996 jhtp3beans/deitel14.gif
4765 Thu Feb 15 01:01:32 EST 1996 jhtp3beans/deitel15.gif
4886 Thu Feb 15 01:05:16 EST 1996 jhtp3beans/deitel16.gif
4873 Thu Feb 15 01:06:12 EST 1996 jhtp3beans/deitel17.gif
4739 Thu Feb 15 01:07:18 EST 1996 jhtp3beans/deitel18.gif
4566 Thu Feb 15 01:08:24 EST 1996 jhtp3beans/deitel19.gif
4819 Thu Feb 15 00:41:06 EST 1996 jhtp3beans/deitel2.gif
4313 Thu Feb 15 01:09:48 EST 1996 jhtp3beans/deitel20.gif
3910 Thu Feb 15 01:10:46 EST 1996 jhtp3beans/deitel21.gif
3076 Thu Feb 15 01:12:02 EST 1996 jhtp3beans/deitel22.gif
3408 Thu Feb 15 01:13:16 EST 1996 jhtp3beans/deitel23.gif
4039 Thu Feb 15 01:14:06 EST 1996 jhtp3beans/deitel24.gif
4393 Thu Feb 15 01:15:02 EST 1996 jhtp3beans/deitel25.gif
4626 Thu Feb 15 01:16:06 EST 1996 jhtp3beans/deitel26.gif
4852 Thu Feb 15 01:17:18 EST 1996 jhtp3beans/deitel27.gif
4929 Thu Feb 15 01:18:18 EST 1996 jhtp3beans/deitel28.gif
4914 Thu Feb 15 01:19:16 EST 1996 jhtp3beans/deitel29.gif
4769 Thu Feb 15 00:42:52 EST 1996 jhtp3beans/deitel3.gif
4617 Thu Feb 15 00:43:54 EST 1996 jhtp3beans/deitel4.gif
4335 Thu Feb 15 00:47:14 EST 1996 jhtp3beans/deitel5.gif
3967 Thu Feb 15 00:49:40 EST 1996 jhtp3beans/deitel6.gif
3200 Thu Feb 15 00:50:58 EST 1996 jhtp3beans/deitel7.gif
3393 Thu Feb 15 00:52:32 EST 1996 jhtp3beans/deitel8.gif
4006 Thu Feb 15 00:53:48 EST 1996 jhtp3beans/deitel9.gif
 420 Sun Mar 14 11:36:16 EST 1999 jhtp3beans/LogoAnima-
tor$1.class
3338 Sun Mar 14 11:36:16 EST 1999 jhtp3beans/LogoAnima-
tor.class
```

Fig. 25.24 The contents of **`LogoAnimator.jar`**.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

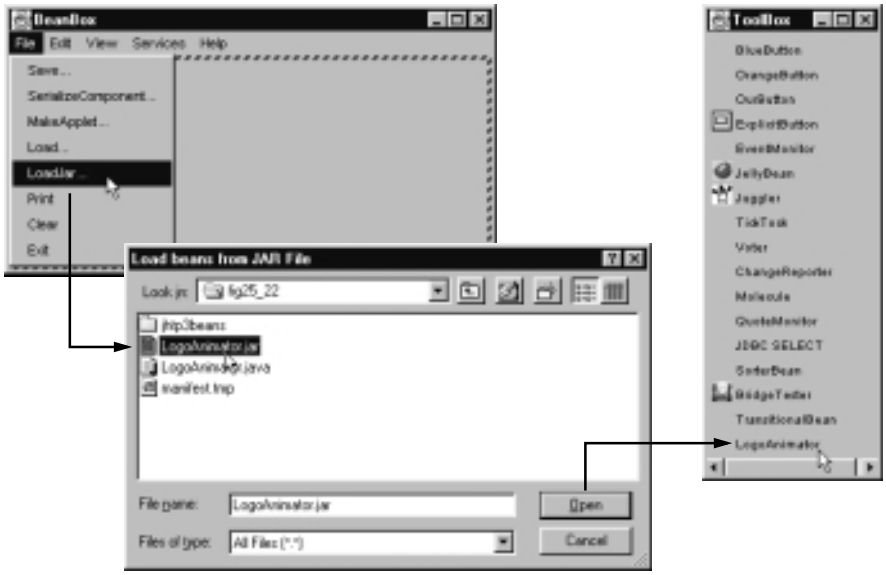


Fig. 25.25 Loading a bean into the **BeanBox**'s **Toolbox**.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

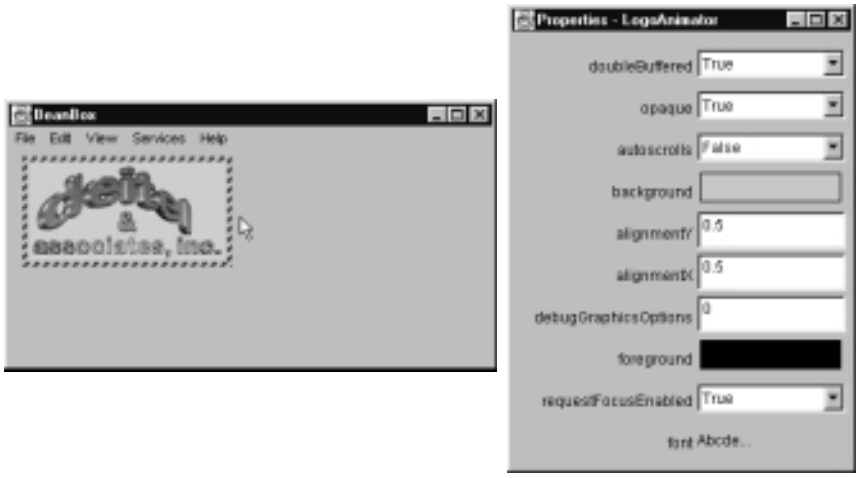


Fig. 25.26 The **`LogoAnimator`** on the **BeanBox** design area.

Fig. 25.27 Changing the background color of the **`LogoAnimator`**.

Fig. 25.28 Two **`ExplicitButton`**s to start and stop the animation.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

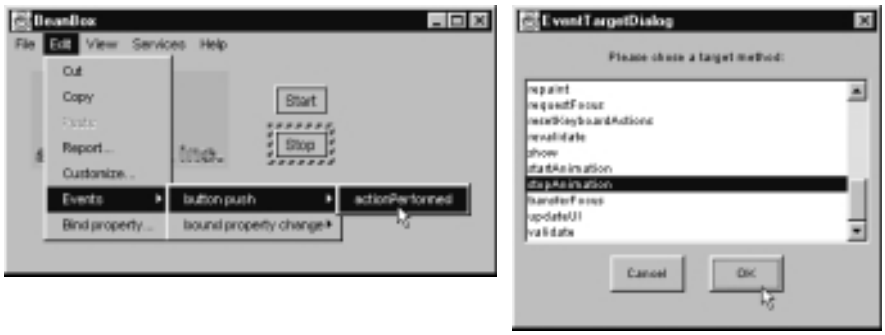


Fig. 25.29 Setting up the button-push event for the **Stop** button.

```
1  // Fig. 25.30: LogoAnimator2.java
2  // Animation bean with animationDelay property
3  package jhtp3beans;
4
5  import java.awt.*;
6  import java.awt.event.*;
7  import javax.swing.*;
8
9  public class LogoAnimator2 extends LogoAnimator {
10    // the following two methods are
11    // for the animationDelay property
12    public void setAnimationDelay( int value )
13    {
14       animationDelay = value;
```

Fig. 25.30 **`LogoAnimator2`** with property **`animationDelay`** (part 1 of 2).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
15          animationTimer.setDelay( animationDelay );
16       }
17
18       public int getAnimationDelay()
19       {
20          return animationTimer.getDelay();
21       }
22
23       public static void main( String args[] )
24       {
25          LogoAnimator2 anim = new LogoAnimator2();
26
27          JFrame app = new JFrame( "Animator test" );
28          app.getContentPane().add( anim, BorderLayout.CENTER );
29
30          app.addWindowListener(
31             new WindowAdapter() {
32                public void windowClosing( WindowEvent e )
33                {
34                   System.exit( 0 );
35                }
36             }
37          );
38
39          app.setSize( anim.getPreferredSize().width + 10,
40                       anim.getPreferredSize().height + 30 );
41          app.show();
42       }
43    }
```

Fig. 25.30 **`LogoAnimator2`** with property **`animationDelay`** (part 2 of 2).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
1  Main-Class: jhtp3beans.LogoAnimator2
2
3  Name: jhtp3beans/LogoAnimator2.class
4  Java-Bean: True
```

Fig. 25.31 The manifest file for the **`LogoAnimator2`** bean.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

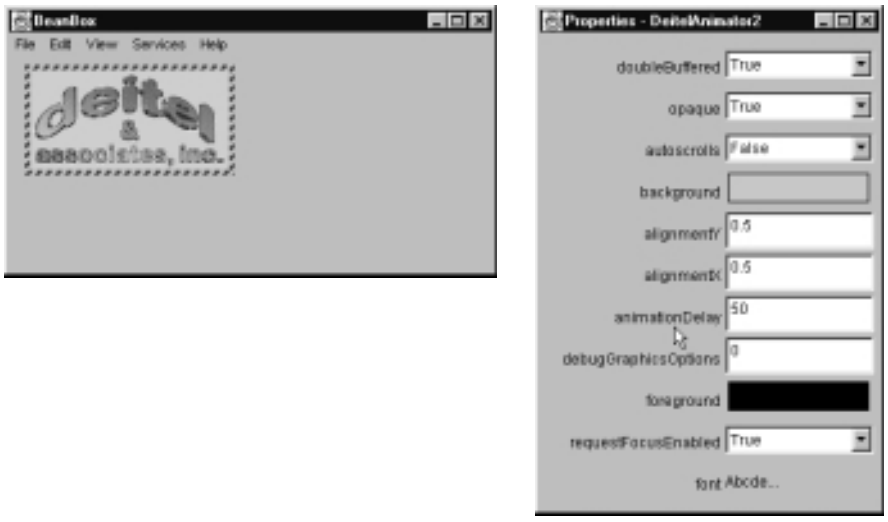


Fig. 25.32 The **`LogoAnimator2`** bean with property **`animationDelay`** exposed in the property sheet.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
1  // Fig. 25.33: SliderFieldPanel.java
2  // A subclass of JPanel containing a JSlider and a JTextField
3  package jhtp3beans;
4
5  import javax.swing.*;
6  import javax.swing.event.*;
7  import java.io.*;
8  import java.awt.*;
9  import java.awt.event.*;
10 import java.beans.*;
11
12 public class SliderFieldPanel extends JPanel
13                               implements Serializable {
14    private JSlider slider;
15    private JTextField field;
16    private Box boxContainer;
17    private int currentValue;
18
19    // object to support bound property changes
20    private PropertyChangeSupport changeSupport;
21
22    public SliderFieldPanel()
23    {
24       // create PropertyChangeSupport for bound properties
25       changeSupport = new PropertyChangeSupport( this );
26
27       slider =
28          new JSlider( SwingConstants.HORIZONTAL, 1, 100, 1 );
29       field = new JTextField(
30                  String.valueOf( slider.getValue() ), 5 );
31
32       boxContainer = new Box( BoxLayout.X_AXIS );
33       boxContainer.add( slider );
34       boxContainer.add( Box.createHorizontalStrut( 5 ) );
35       boxContainer.add( field );
36
37       setLayout( new BorderLayout() );
38       add( boxContainer );
39
40       slider.addChangeListener(
41          new ChangeListener() {
42             public void stateChanged( ChangeEvent e )
43             {
44                setCurrentValue( slider.getValue() );
45             }
46          }
47       );
48
```

Fig. 25.33 Class **SliderFieldPanel** definition (part 1 of 3).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
49          field.addActionListener(
50             new ActionListener() {
51                public void actionPerformed( ActionEvent e )
52                {
53                   setCurrentValue(
54                      Integer.parseInt( field.getText() ) );
55                }
56             }
57          );
58       }
59
60       // methods for adding and removing PropertyChangeListeners
61       public void addPropertyChangeListener(
62          PropertyChangeListener listener )
63       {
64          changeSupport.addPropertyChangeListener( listener );
65       }
66
67       public void removePropertyChangeListener(
68          PropertyChangeListener listener )
69       {
70          changeSupport.removePropertyChangeListener( listener );
71       }
72
73       // property minimumValue
74       public void setMinimumValue( int min )
75       {
76          slider.setMinimum( min );
77
78          if ( slider.getValue() < slider.getMinimum() ) {
79             slider.setValue( slider.getMinimum() );
80             field.setText( String.valueOf( slider.getValue() ) );
81          }
82       }
83
84       public int getMinimumValue()
85       {
86          return slider.getMinimum();
87       }
88
89       // property maximumValue
90       public void setMaximumValue( int max )
91       {
92          slider.setMaximum( max );
93
94          if ( slider.getValue() > slider.getMaximum() ) {
95             slider.setValue( slider.getMaximum() );
96             field.setText( String.valueOf( slider.getValue() ) );
97          }
98       }
99
```

Fig. 25.33 Class **SliderFieldPanel** definition (part 2 of 3).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
100    public int getMaximumValue()
101    {
102       return slider.getMaximum();
103    }
104
105    // property currentValue
106    public void setCurrentValue( int current )
107    {
108       int oldValue = currentValue;
109       currentValue = current;
110       slider.setValue( currentValue );
111       field.setText( String.valueOf( currentValue ) );
112       changeSupport.firePropertyChange(
113          "currentValue", new Integer( oldValue ),
114          new Integer( currentValue ) );
115    }
116
117    public int getCurrentValue()
118    {
119       return slider.getValue();
120    }
121
122    // property fieldWidth
123    public void setFieldWidth( int cols )
124    {
125       field.setColumns( cols );
126       boxContainer.validate();
127    }
128
129    public int getFieldWidth()
130    {
131       return field.getColumns();
132    }
133
134    public Dimension getMinimumSize()
135    {
136       return boxContainer.getMinimumSize();
137    }
138
139    public Dimension getPreferredSize()
140    {
141       return boxContainer.getPreferredSize();
142    }
143 }
```

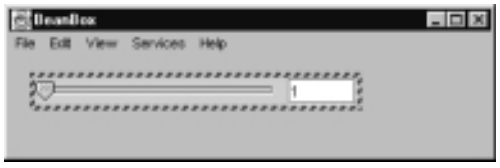


Fig. 25.33 Class **`SliderFieldPanel`** definition (part 3 of 3).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

Fig. 25.35 Selecting the bound property from the **PropertyNameDialog**.

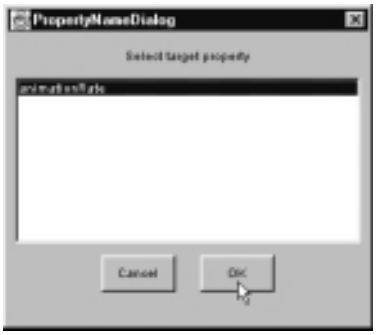


Fig. 25.36 Selecting the target property from the **PropertyNameDialog**.

Fig. 25.37 The **BeanBox** window with the **`SliderFieldPanel`** and Juggler beans.

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
1  // Fig. 25.38: SliderFieldPanelBeanInfo.java
2  // The BeanInfo class for SliderFieldPanel
3  package jhtp3beans;
4
5  import java.beans.*;
6
7  public class SliderFieldPanelBeanInfo extends SimpleBeanInfo {
8     public final static Class beanClass =
9        SliderFieldPanel.class;
10
11    public PropertyDescriptor[] getPropertyDescriptors()
12    {
13       try {
14          PropertyDescriptor fieldWidth =
15             new PropertyDescriptor( "fieldWidth", beanClass );
16          PropertyDescriptor currentValue =
17             new PropertyDescriptor(
18                "currentValue", beanClass );
19          PropertyDescriptor maximumValue =
20             new PropertyDescriptor(
21                "maximumValue", beanClass );
22          PropertyDescriptor minimumValue =
23            new PropertyDescriptor( "minimumValue", beanClass );
24
25          // ensure PropertyChangeEvent occurs for this property
26          currentValue.setBound( true );
27
28          PropertyDescriptor descriptors[] = { fieldWidth,
29             currentValue, maximumValue, minimumValue };
30
31          return descriptors;
32       }
33       catch ( IntrospectionException ie ) {
34          throw new RuntimeException( ie.toString() );
35       }
36    }
37
```

Fig. 25.38 Demonstrating class **`SliderFieldPanelBeanInfo`** in the **BeanBox** (part 1 of 2).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company

```
38      // the index for the currentValue property
39      public int getDefaultPropertyIndex()
40      {
41         return 1;
42      }
43
44      public EventSetDescriptor[] getEventSetDescriptors() {
45         try {
46            EventSetDescriptor changed =
47               new EventSetDescriptor( beanClass,
48               "propertyChange",
49               java.beans.PropertyChangeListener.class,
50               "propertyChange");
51
52            changed.setDisplayName(
53               "SliderFieldPanel value changed" );
54
55            EventSetDescriptor[] descriptors = { changed };
56
57            return descriptors;
58         }
59         catch (IntrospectionException e) {
60            throw new RuntimeException(e.toString());
61         }
62      }
63   }
```

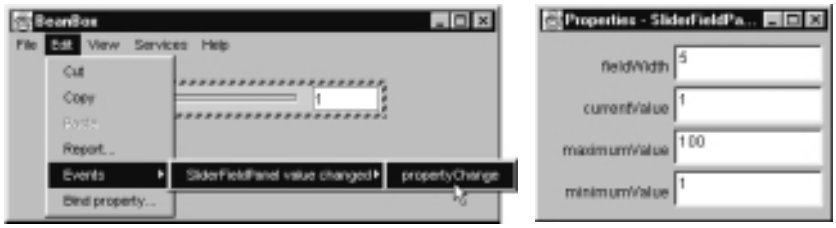


Fig. 25.38 Demonstrating class **`SliderFieldPanelBeanInfo`** in the **BeanBox** (part 2 of 2).

from Java: How To Program, 3e
by Deitel & Deitel
©2000 Prentice Hall, Inc.
A Pearson Company