Simplest JavaBeans Bean and NetBeans, Part 2
Simplest but doing something, anything, please!

Carl W. David[1]
Department of Chemistry
University of Connecticut
Storrs, Connecticut 06269-3060
CarlDavid@uconn.edu

OK, we've made a bean which we can see. Now to make it do something. We look at he bean's properties, click on its list of events, and click on the top one, generating source code:

```
  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
       myBeanButton.setBackground(Color.BLUE);
    }
```

which, when the imports are fixed, works just fine. Try it, you'll like it. Note that, from the last article in this series, myBeanButton was defined in the bean's source, and created in the post-initialization code generation of the button by editing it's properties. There may be better ways of doing this, but I'm just a beginner, so slogging through is the best I can do.

Parenthetically, I need to note that in the initialization code for this bean, I've added the line:

```
this.setSize(600,100);//needed to get appearance in main
```

which allows the main code to use:

```
 initComponents();
```

1 As usual, all errors are mine. Please write me with corrections (suggestions, criticisms) at Carl.David@ucon.edu. Thanks.

```
        beaninput2 myBean = new beaninput2();
        this.add(myBean);//worked if sized in bean itself
```
instead of the more cumbersome code used before. Pretty cool, hunh?

Setters and Getters

Bean's properties are changed using setters and getters. Here is some typical code which shows how to do this:

```
public Color getColor(){// this is a getter method
        return color;
    }
public void setColor(Color newColor){// this is a setter method
        color = newColor;
    }
```

The calling program (Main, in our case) can have a statement such as

```
this.add(myBean);//worked if sized in bean itself
myBean.setbeaninput2Property("This was set from the main program!");
```
where the second line is the new one, corresponding to a bean setter method:

```
 public void setbeaninput2Property(String value){//setter
        String oldValue = beaninput2Property;
        beaninput2Property = value;
        beaninput2Support.firePropertyChange
                (beaninput2_Property,oldValue,beaninput2Property);
        System.out.println(value);
    }
```
When you insert the two (green) codes, the first in Main, the second in beaninput2, you will see that the setter works (the full code appears later in this article, if you're confused about incremental changes).

Events

We need to have the bean tell its parent things. Before proceeding, note that we've placed both Main and beaninput2 into the same package, so that we can step into the bean in debug mode and see what's going on.

I had a lot of trouble getting this to work, and the fault was partly mine and partly the way documentation is written. To give credit where credit is due, http://perso.wanadoo.fr/jm.doudoux/java/tutorial/ was the place where I finally found what I needed, a simple example (like this one). Once I'd seen that, I realized that Sun had virtually the same, hidden under flowers, in http://java.sun.com/docs/books/tutorial/uiswing/events/propertychangelistener.html . Here is a quote from the Sun site:

```
KeyboardFocusManager focusManager =
   KeyboardFocusManager.getCurrentKeyboardFocusManager();
focusManager.addPropertyChangeListener(new FocusManagerListener());
...
public FocusManagerListener() implements PropertyChangeListener {
    public void propertyChange(PropertyChangeEvent e) {
        String propertyName = e.getPropertyName();
        if ("focusOwner".equals(propertyName) {
            ...
        } else if ("focusedWindow".equals(propertyName) {
            ...
        }
    }
    ...
}
```

which I will show (below) is virtually what Jean Michel Doudoux wrote in his text (note, the text is in French, and when you use Google, for instance, to translate it, the code comes out funny, so you need to look at this material in two windows, the original French version, with the Java code in English, by the way) and the translated version in another window, where the code is goofed.

We start with the main routine, which, sets up the bean, sets up the Listener, and then sits back and waits for the bean to do something.

```
 * Main.java
 *
 * Created on November 9, 2005, 11:12 AM
 */

package input2;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

public class Main extends javax.swing.JFrame {

   /** Creates new form Main */
   public Main(){
      initComponents();
      myBean = new beaninput2();
      /* this gives another window for this, just as good, we'll see.*/
      /*
      JFrame myFrame = new JFrame();
      myFrame.add(myBean);//this worked
      //myFrame.getContentPane().add(myBean);//this worked too
      myFrame.setSize(600,100); need to set here as well if this method is used
      myFrame.setVisible(true);
       */
      this.add(myBean);//worked if sized in bean itself
      myBean.addPropertyChangeListener(new PropertyChangeListener(){
        public void propertyChange(PropertyChangeEvent event){
            if(debug)System.out.println("[In Main] PropertyChange  "+
event.getPropertyName());
         }
      });
             myBean.setProperty("called from Main, myBean.setProperty!");
   }

   /** This method is called from within the constructor to
    * initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is
    * always regenerated by the Form Editor.
    */
   // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
   // </editor-fold>

   private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
      System.exit(0);
   }

   /**
    * @param args the command line arguments
    */
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new Main().setVisible(true);
         }
      });
   }

   // Variables declaration - do not modify
   // End of variables declaration

   private  beaninput2 myBean;
   private boolean debug = true;
}
```

The purple backgrounded text, above, is virtually line for line identical with the Sun text quoted above. Somehow or other, the Sun text didn't penetrate my (pathetically small) brain, and Doudoux's version did. Thank goodness.

Here's the bean, with two setter and two getter methods implemented.

```
/*
 * beaniput2.java
 *
 * Created on November 8, 2005, 11:00 AM
 */

package input2;

import java.awt.Color;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import javax.swing.JButton;
import java.io.*;
import javax.swing.JLabel;




public class beaninput2 extends javax.swing.JPanel implements Serializable{

   /** Creates new form BeanForm */
   public beaninput2() {
      initComponents();
      this.setSize(600,100);//needed to get appearance in main
      pcs = new PropertyChangeSupport(this);
      setVisible(true);
   }

   public String getProperty(){//getter
      return Property;
   }
   public String getLabel(){//getter
      return label;
   }

   public void setProperty(String value){//setter
      String oldValue = Property;
      Property = value;
      pcs.firePropertyChange("setProperty called",0,1);

      myBeanButton.setBackground(color_green);
      if(debug)System.out.println("[IN BEAN.setProperty]the bean passed the
value:"+value);
   }

   public void setLabel(String newLabel){
      String oldLabel = label;
      label = newLabel;
      if(debug)System.out.println("[IN BEAN.setLabel]set label called and ready to
fire");
```

```
        pcs.firePropertyChange("setLabelProperty",oldLabel,newLabel);
        myLabel.setText(newLabel);
    }

    public synchronized void addPropertyChangeListener(PropertyChangeListener pcl) {
        try {
            if(debug)System.out.println("[IN BEAN.addPropertyChangeListener]adding pcl
in subroutine");
            pcs.addPropertyChangeListener("setLabelProperty",pcl);
            pcs.addPropertyChangeListener("setProperty called",pcl);
        } finally {
            if(debug)System.out.println("[IN BEAN.addPropertyChangeListener]finally,
caught");
        }
    }

    public synchronized void removePropertyChangeListener(PropertyChangeListener
pcl) {
        pcs.removePropertyChangeListener(pcl);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    // </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        if(debug)System.out.println("[IN BEAN.jButton1ActionPerformed]color changed
to green, ready to fire Property Change");
        if(debug)System.out.println("[IN BEAN.jButton1ActionPerformed]ready to
setLabel");
        setLabel("I've changed");
        if(debug)System.out.println("[IN BEAN.jButton1ActionPerformed]setLabel done");
        setProperty("[IN BEAN.jButton1ActionPerformed]done with buttonPerfomed
ActionEvent");
        System.out.println("getProperty()"+getProperty());
        setColor(color_blue);
        myBeanButton.setBackground(color);
    }
    public Color getColor(){// this is a getter method
        return color;
    }
    public void setColor(Color newColor){
        color = newColor;
    }

    // Variables declaration - do not modify
    // End of variables declaration
    private Color color_green = Color.green;
    private Color color = Color.CYAN;//this is the starting color
    private Color color_red = Color.red;
    private Color color_blue = Color.blue;
    public static JButton myBeanButton;
    public  String Property = "intial property";
    private PropertyChangeSupport pcs;
    public JLabel myLabel;
    public String label;
    private boolean debug = true;
}
```

In the above code, the Design initiated coding has been hidden, so you can not see that we have exactly the same bean as before, one button, one label.

Contrary to normal practice in these pages, I'm including the entire source code of the bean, including some added JDialog materials to help the reader follow the interplay between the bean and its calling program Here's the full bean:

```
/*
 * beaniput2.java
 *
 * Created on November 8, 2005, 11:00 AM
 */

package input2;

import java.awt.Color;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import javax.swing.JButton;
import java.io.*;
import javax.swing.JLabel;
import javax.swing.JOptionPane;




public class beaninput2 extends javax.swing.JPanel implements
Serializable{

    /** Creates new form BeanForm */
    public beaninput2() {
        initComponents();
        this.setSize(600,100);//needed to get appearance in main
        pcs = new PropertyChangeSupport(this);
        setVisible(true);
    }

    public String getProperty(){//getter
        return Property;
    }
    public String getLabel(){//getter
        return label;
    }

    public void setProperty(String value){//setter
        String oldValue = Property;
        Property = value;
        pcs.firePropertyChange("setProperty called",0,1);//this one
isn't caught
        int answer = JOptionPane.showConfirmDialog(this, "We fired the
Property Change from the setProperty code with the 'value' = "+value);
        myBeanButton.setBackground(color_green);
        if(debug)System.out.println("[IN BEAN.setProperty]the bean
passed the value:"+value);
    }

    public void setLabel(String newLabel){
        String oldLabel = label;
        label = newLabel;
        if(debug)System.out.println("[IN BEAN.setLabel]set label called
and ready to fire");
        pcs.firePropertyChange("setLabelProperty",oldLabel,newLabel);
        int answer = JOptionPane.showConfirmDialog(this,"We've just
fired the setLabel property change!");
        myLabel.setText(newLabel);
    }

    public synchronized void addPropertyChangeListener
(PropertyChangeListener pcl) {
        try {
            if(debug)System.out.println("[IN
BEAN.addPropertyChangeListener]adding pcl in subroutine");
            pcs.addPropertyChangeListener("setLabelProperty",pcl);
            pcs.addPropertyChangeListener("setProperty called",pcl);
        } finally {
            if(debug)System.out.println("[IN
BEAN.addPropertyChangeListener]finally, caught");
        }
    }

    public synchronized void removePropertyChangeListener
(PropertyChangeListener pcl) {
        pcs.removePropertyChangeListener(pcl);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code
">//GEN-BEGIN:initComponents
    private void initComponents() {
        javax.swing.JButton jButton1;
        javax.swing.JLabel jLabel1;

        jButton1 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();

        jButton1.setText("Bean Button");
        myBeanButton=jButton1;
        jButton1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                jButton1ActionPerformed(evt);
            }
        });

        add(jButton1);

        jLabel1.setText("Label original");
        myLabel = jLabel1;
        label = "Label original";
        add(jLabel1);

    }
    // </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton1ActionPerformed
// TODO add your handling code here:
        if(debug)System.out.println("[IN BEAM.jButton1ActionPerformed]
color changed to green, ready to fire Property Change");
        if(debug)System.out.println("[IN BEAM.jButton1ActionPerformed]
ready to setLabel");
        setLabel("I've changed");
        if(debug)System.out.println("[IN BEAM.jButton1ActionPerformed]
setLabel done");
        if(debug)System.out.println("");
        setProperty("[IN BEAM.jButton1ActionPerformed]done with
buttonPerfomed ActionEvent");
        System.out.println("getProperty()"+getProperty());
        setColor(color_blue);
        myBeanButton.setBackground(color);
    }//GEN-LAST:event_jButton1ActionPerformed
    public Color getColor(){// this is a getter method
        return color;
    }
    public void setColor(Color newColor){
        color = newColor;
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
    private Color color_green = Color.green;
    private Color color = Color.CYAN;//this is the starting color
    private Color color_red = Color.red;
    private Color color_blue = Color.blue;
    public static JButton myBeanButton;
    public  String Property = "initial property";
    private PropertyChangeSupport pcs;
    public JLabel myLabel;
    public String label;
    private boolean debug = true;
```

```
}
```

And here's the Main program:

```
/*
 * Main.java
 *
 * Created on November 9, 2005, 11:12 AM
 */

package input2;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import javax.swing.JOptionPane;

public class Main extends javax.swing.JFrame {

    /** Creates new form Main */
    public Main(){
        initComponents();
        myBean = new beaninput2();
        this.add(myBean);//worked if sized in bean itself
        myBean.addPropertyChangeListener(new PropertyChangeListener(){
           public void propertyChange(PropertyChangeEvent event){
               if(debug)System.out.println("[In Main] PropertyChange
"+ event.getPropertyName());
           }
        });
                myBean.setProperty("called from Main,
myBean.setProperty!");
                int answer = JOptionPane.showConfirmDialog(this, "Did
you see that we sent a message to setProperty, i.e.," +
                        "printed (in the OUTPUT window) from the bean's
code!");
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code
">//GEN-BEGIN:initComponents
    private void initComponents() {
        javax.swing.JMenuItem aboutMenuItem;
        javax.swing.JMenuItem contentsMenuItem;
        javax.swing.JMenuItem copyMenuItem;
        javax.swing.JMenuItem cutMenuItem;
        javax.swing.JMenuItem deleteMenuItem;
        javax.swing.JMenu editMenu;
        javax.swing.JMenuItem exitMenuItem;
        javax.swing.JMenu fileMenu;
        javax.swing.JMenu helpMenu;
        javax.swing.JMenuBar menuBar;
        javax.swing.JMenuItem openMenuItem;
        javax.swing.JMenuItem pasteMenuItem;
        javax.swing.JMenuItem saveAsMenuItem;
        javax.swing.JMenuItem saveMenuItem;

        menuBar = new javax.swing.JMenuBar();
        fileMenu = new javax.swing.JMenu();
        openMenuItem = new javax.swing.JMenuItem();
        saveMenuItem = new javax.swing.JMenuItem();
        saveAsMenuItem = new javax.swing.JMenuItem();
        exitMenuItem = new javax.swing.JMenuItem();
        editMenu = new javax.swing.JMenu();
        cutMenuItem = new javax.swing.JMenuItem();
        copyMenuItem = new javax.swing.JMenuItem();
        pasteMenuItem = new javax.swing.JMenuItem();
        deleteMenuItem = new javax.swing.JMenuItem();
        helpMenu = new javax.swing.JMenu();
        contentsMenuItem = new javax.swing.JMenuItem();
        aboutMenuItem = new javax.swing.JMenuItem();

        setDefaultCloseOperation
(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        fileMenu.setText("File");
        openMenuItem.setText("Open");
        fileMenu.add(openMenuItem);

        saveMenuItem.setText("Save");
        fileMenu.add(saveMenuItem);

        saveAsMenuItem.setText("Save As ...");
        fileMenu.add(saveAsMenuItem);

        exitMenuItem.setText("Exit");
        exitMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                exitMenuItemActionPerformed(evt);
            }
        });

        fileMenu.add(exitMenuItem);

        menuBar.add(fileMenu);

        editMenu.setText("Edit");
        cutMenuItem.setText("Cut");
        editMenu.add(cutMenuItem);

        copyMenuItem.setText("Copy");
        editMenu.add(copyMenuItem);

        pasteMenuItem.setText("Paste");
        editMenu.add(pasteMenuItem);

        deleteMenuItem.setText("Delete");
        editMenu.add(deleteMenuItem);

        menuBar.add(editMenu);

        helpMenu.setText("Help");
        contentsMenuItem.setText("Contents");
        helpMenu.add(contentsMenuItem);
        aboutMenuItem.setText("About");
        helpMenu.add(aboutMenuItem);

        menuBar.add(helpMenu);

        setJMenuBar(menuBar);

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
            .add(0, 400, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
            .add(0, 279, Short.MAX_VALUE)
        );
        pack();
    }
    // </editor-fold>//GEN-END:initComponents

    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_exitMenuItemActionPerformed
        System.exit(0);
    }//GEN-LAST:event_exitMenuItemActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Main().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables

    private  beaninput2 myBean;
   // private PropertyChangeEvent evt;
    private boolean debug = true;
}
```