Source for java.beans.PropertyEditorSupport

```
/* java.beans.PropertyEditorSupport
   Copyright (C) 1998, 2004 Free Software Foundation, Inc.

This file is part of GNU Classpath.

GNU Classpath is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2, or (at your option)
any later version.

GNU Classpath is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with GNU Classpath; see the file COPYING.  If not, write to the
Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301 USA.

Linking this library statically or dynamically with other modules is
making a combined work based on this library.  Thus, the terms and
conditions of the GNU General Public License cover the whole
combination.

As a special exception, the copyright holders of this library give you
permission to link this library with independent modules to produce an
executable, regardless of the license terms of these independent
modules, and to copy and distribute the resulting executable under
terms of your choice, provided that you also meet, for each linked
independent module, the terms and conditions of the license of that
module.  An independent module is a module which is not derived from
or based on this library.  If you modify this library, you may extend
this exception to your version of the library, but you are not
```

```
obligated to do so.  If you do not wish to do so, delete this
exception statement from your version. */


package java.beans;


/**
 * PropertyEditorSupport helps with PropertyEditors,
 * implementing base functionality that they usually must
 * have but which is a pain to implement.  You may extend
 * from this class or use it as a standalone.<P>
 *
 * This class does not do any painting or actual editing.
 * For that, you must use or extend it.  See the
 * PropertyEditor class for better descriptions of what
* the various methods do.
 *
 * @author John Keiser
 * @author Robert Schuster
 * @since 1.1
 * @status updated to 1.5
 */
public class PropertyEditorSupport implements PropertyEditor
{
   Object eventSource;
  Object value;
   PropertyChangeSupport pSupport;

   /** Call this constructor when you are deriving from
    * PropertyEditorSupport.
    *
    * Using this constructor the event source is this PropertyEditorSupport
    * instance itself.
    *
    * @since 1.5
   * @specnote this was <code>protected</code> prior to 1.5
    */
   public PropertyEditorSupport()
   {
     eventSource = this;
     pSupport = new PropertyChangeSupport(this);
   }

   /** Call this constructor when you are using
    * PropertyEditorSupport as a helper object.
   *
    * This constructor throws a NullPointerException when <code>source</code> is <code>null</code>,
    * for compatibility reasons with J2SDK 1.5.0 .
    *
    * @param source The source to use when firing
    * property change events.
    * @since 1.5
    * @specnote this was <code>protected</code> prior to 1.5
    */
   public PropertyEditorSupport(Object source)
   {
     // note: constructor rejects source being null for the sake of compatibility
     // with official 1.5.0 implementation
     if (source == null)
       throw new NullPointerException("Event source must not be null.");

     eventSource = source;
     pSupport = new PropertyChangeSupport(eventSource);
   }

 /** Sets the current value of the property and a property change
    * event is fired to all registered PropertyChangeListener instances.
    *
    * @param newValue The new value for the property.
    */
   public void setValue(Object newValue)
   {
     value = newValue;

    // specification in java.beans.PropertyChangeEvent says
    // that without a property name (first argument) the
    // new and the old value should always be null
    pSupport.firePropertyChange(null, null, null);
 }

 /** Gets the current value of the property.
   *
   * @return the current value of the property.
   */
   public Object getValue()
   {
     return value;
   }

   /** Gets whether this object is paintable or not.
    *
    * @return <CODE>false</CODE>
    */
   public boolean isPaintable()
   {
    return false;
   }

   /** Paints this object.  This class does nothing in
    * this method.
    */
   public void paintValue(java.awt.Graphics g, java.awt.Rectangle r)
   {
   }

 /** Gets the Java initialization String for the current
    * value of the Object.  This class returns gibberish or
    * null (though the spec does not say which).<P>
    * <STRONG>Implementation Note:</STRONG> This class
    * returns the string "@$#^" to make sure the code will
    * be broken, so that you will know to override it when
    * you create your own property editor.
    *
    * @return the Java initialization string.
    */
 public String getJavaInitializationString()
   {
     return "@$#^";
   }

   /** Gets the value as text.
    * In this class, you cannot count on getAsText() doing
    * anything useful, although in this implementation I
    * do toString().
    *
   * @return the value as text.
    */
   public String getAsText()
   {
     return value != null ? value.toString() : "null";
   }

   /** Sets the value as text.
    * In this class, you cannot count on setAsText() doing
    * anything useful across implementations.
   * <STRONG>Implementation Note:</STRONG> In this
    * implementation it checks if the String is "null", and
    * if it is, sets the value to null, otherwise it throws
    * an IllegalArgumentException.
    *
    * @param s the text to convert to a new value.
    * @exception IllegalArgumentException if the text is
    * malformed.
    */
   public void setAsText(String s) throws IllegalArgumentException
  {
     if (s.equals("null"))
       setValue(null);
     else
       throw new IllegalArgumentException();
   }

   /** Returns a list of possible choices for the value.
    *
    * @return <CODE>null</CODE>
   */
   public String[] getTags()
   {
     return null;
   }

   /** Returns a custom component to edit the value.
    *
    * @return <CODE>null</CODE> in this class.
    */
   public java.awt.Component getCustomEditor()
   {
     return null;
   }

   /** Finds out whether this property editor supports a
```

```
    * custom component to edit its value.
    *
    * @return <CODE>false</CODE> in this class.
    */
  public boolean supportsCustomEditor()
  {
    return false;
  }

  /** Adds a property change listener to this property editor.
   *
   * @param l the listener to add.
   */
  public void addPropertyChangeListener(PropertyChangeListener l)
  {
    pSupport.addPropertyChangeListener(l);
  }

  /** Removes a property change listener from this property editor.
   *
   * @param l the listener to remove.
   */
  public void removePropertyChangeListener(PropertyChangeListener l)
  {
    pSupport.removePropertyChangeListener(l);
  }

  /** Notifies people that we've changed, although we don't
   * tell them just how.
   */
  public void firePropertyChange()
  {
    pSupport.firePropertyChange(null, null, null);
  }

  /** Returns the bean that is used as the source of events.
   *
   * @return The event source object
   * @since 1.5
   */
  public Object getSource()
  {
    return eventSource;
  }

  /** Sets the bean that is used as the source of events
   * when property changes occur.
   *
   * The event source bean is for informational purposes only
   * and should not be changed by the <code>PropertyEditor</code>.
   *
   * @param source
   * @since 1.5
   */
  public void setSource(Object source)
  {
    eventSource = source;
  }
}
```