Contents

com.crsoft.controls

Note: To use the following controls, you must import the control package as in;
import com.crsoft.controls.*;

Class Index

PropertySheet

public class PropertySheet
extends Panel

Overview

The PropertySheet control enables you to use tab controls in you Applets or applications.
See the example at the end of this page for further discussion on how to use it.

Constructor

- public PropertySheet(Frame frm, String title, boolean modal)

Constructs a propertysheet.
Arguments:
Frame = the frame in your application or applet
title = the title of the propertysheet
modal = whether to show the propertysheet modal or non-modal

Variables

- OKButton

public Button OKButton

- CancelButton

public Button CancelButton

- ApplyButton

public Button ApplyButton

- HelpButton

public Button HelpButton

Methods

- public void add(PropertyPage ppg)

Adds a PropertyPage to the PropertySheet.

- public void show()

Shows the PropertySheet

- public void setCloseAction(int action)

Sets the close action on the propertysheet when the system menu 'close' item is clicked.
Where action can be one of the following;

1. *static int PropertySheet.DESTROY (default)*
2. *static int PropertySheet.HIDE*

where;

PropertySheet.HIDE = Hide propertysheet only, do not destroy. Showing and hiding the propertysheet is much faster than re-creating it. For example, you might create the propertysheet in the init() method of your applet and have a button that shows and hides the Propertysheet.

PropertySheet.DESTROY = Destroy and remove propertysheet. If you use this option you must re-create the propertysheet each time you show it. This is the default.

Example;

```
setCloseAction( PropertySheet.DESTROY )
```

Example

There are four basic steps to create a propertysheet.

1. Import the control Package

```
Import com.crsoft.controls.*;
```

2. Define the Panels

Note: You should at least resize one panel so the PropertySheet(shown below) can resize itself.

```
// First Tab
class Tab1_Panel extends Panel {
    public Tab1_Panel() {
        super();
        resize( 250, 200 );
        add( new Label( "First Name:" ) );
        add( new TextField( 10 ) );
        add( new Label( "Last Name: " ) );
        add( new TextField( 20 ) );
        Choice ddlb = new Choice();
        ddlb.addItem( "one" );
        ddlb.addItem( "two" );
        ddlb.addItem( "three" );
        add( ddlb );
        setFont( new Font( "TimesRoman",Font.PLAIN, 10 ) );
  }
```

```
// Handle all events on this panel
      public boolean handleEvent( Event evt ) {
            if( evt.target instanceof TextField ) {
                  System.out.println( "EmployeeTextField" );
            }
            return( super.handleEvent( evt ) );
      }
}

// Second Tab
class Tab2_Panel extends Panel {

      public Tab2_Panel() {
            super();

            setLayout( new BorderLayout() );
            add( "North", new Button("North" ) );
            add( "South", new Button( "South" ) );
            add( "East", new Button("East" ) );
            add( "West", new Button( "West" ) );
            add( "Center", new Button( "Center" ) );
      }

// Handle all events on this panel
      public boolean handleEvent( Event evt ) {
            if( evt.target instanceof Choice ) {
                  System.out.println( "Choice" );
            }
            return( super.handleEvent( evt ) );
      }

}
```

3. Define the PropertySheet and add the PropertyPages

```
    class myPropertySheet extends PropertySheet {

// Methods
      // Constructor
      public myPropertySheet( Frame frm, String strLabel, boolean bMode ) {
            super( frm, strLabel, bMode );

      // Create PropertyPages and Panels
            add( new PropertyPage( "Tab 1", new Tab1_Panel() ) );
            add( new PropertyPage( "Tab 2", new Tab2_Panel() ) );

      // Set Font on all tabs...
            setFont( new Font( "Dialog", Font.ITALIC + Font.BOLD, 13 ) );

    // Set the closeaction to destroy
            setCloseAction( PropertySheet.DESTROY );    // This is actually the default

    // Override button defaults
```

```
            OKButton.setFont( new Font( "Dialog", Font.BOLD, 12 ) );
            OKButton.setLabel( "&OK" );     // change the label if you wish
            ApplyButton.hide();
        }

        // handle button events such as OK, Cancel, Apply and Help
        public boolean handleEvent( Event evt ) {

           // This works because these are the only buttons on the PropertySheet
           // The other events are handled at the panel level
              if( evt.target instanceof Button ) {
                    if( evt.arg.equals( "Help" ) ) {
                          System.out.println( "Help hit" );
                    }
                    if( evt.arg.equals( "Cancel" ) ) {
                          System.out.println( "Cancel hit" );
                    }
                    if( evt.arg.equals( "&OK" ) ) {
                          System.out.println( "OK hit" );
                    }

              }

             return super.handleEvent( evt );
        }
}
```

4. Create and display the PropertySheet

```
    // For this example, we could call the code below inside a button or another component to
    // create and display the PropertySheet.

     myPropertySheet myPPS = new myPropertySheet( new Frame(), "Tab Control Example", true );
     myPPS.show();

   // Or you could alternatively call,
      ( new myPropertySheet( new Frame(), "Tab Control Example", true ) ).show();
```

That's all there is to it!!!

ImageButton

public class ImageButton
extends Panel

Overview

The ImageButton control allows you to display graphical images along with text in a button. In addition you can play audio files (applicable to Applets only) when clicking or moving the mouse over the control. See the example at the bottom of this page.

Constructors

- ImageButton()

Constructs a button with no image or label

- ImageButton(String image)

Constructs a button with an image when using Java applications.
Example: ImageButton ibtn = new ImageButton("t1.gif")

- ImageButton(Applet applet, URL url, String image)

Constructs a button with an image when using Java applets.
Example: ImageButton ibtn = new ImageButton(this, this.getDocumentBase(), "images/t1.gif")

Methods

- setImage(String image)

Associates an image with the button when using Java applications.

- setImage(Applet applet, URL url, String image)

Associates an image with the button when using Java applets.
Example: setImage(this, this.getDocumentBase(), "images/t1.gif")

- adjustToImage(boolean adjust)

Call this method if you wish the button to adjust itself to the size of the image. For example; ibtn.adjustToImage(true). The default is false.

- setLabel(String label)

Associates a label with the button. The label appears below and center of the image.

- setFont(Font buttonFont)

Sets the type of font associated with the button label. This method is ignored if a label is not set for the button.

- setColorText(Color color)

Sets the color associated with the button label. This method is ignored if a label is not set for the button.

- setAudio(Applet applet, URL url, String strAudoFile)

Call this method if you wish to associate an audio with your button in an applet.
Example: ibtn.setAudio(this, this.getDocumentBase(), "audio/t2.au");

- playOnMouseDown(boolean play)

Call this method to play the associated audio when the user clicks the button. This method is ignored if setAudio has not been called.

- playOnMouseOver(boolean play)

Call this method to play the associated audio when the user moves the mouse over the button. This method is ignored if setAudio has not been called.

- disable()

Call this method to disable the button.

Example

Import com.crsoft.controls.*; // Don't forget to import the control package

Note: the 'this' in this example refers to the applet in which the code resides.

```
ImageButton btn1 = new ImageButton( this, this.getDocumentBase(), "images/t1.gif" );
btn1.setAudio( this, this.getDocumentBase(), "audio/t2.au" );
btn1.playOnMouseDown(true)      // Play when the mouse is clicked
btn1.resize( 100,100 );
btn1.adjustToImage( true );
btn1.setLabel( "OK" );
btn1.setFont( new Font( "Helvetica", Font.BOLD + Font.ITALIC, 10 ) );
btn1.setTextColor( Color.yellow );

ImageButton btn2 = new ImageButton( this, this.getDocumentBase(), "images/t2.gif" );
btn2.resize( 200,75 );
btn2.adjustToImage( false );
btn2.setLabel( "Java is Fun!" );
```

```
btn2.setFont( new Font( "Helvetica", Font.BOLD, 10 ) );

add( btn1 );
add( btn2 );

btn1.disable();       // if you wish to disable the button
```

TimerControl

public class TimerControl
extends Frame

Overview

The TimerControl is a control that displays a message at a preset interval. The control will automatically center itself in the parent window. If you've seen the houseBlend demo, you've already seen an example of this control!

Constructor

- public TimerControl(String title)

Constructs the control with a title.

Methods

- public void setMessage(String message)

Set the message that will be displayed in the control.

- public void setTextColor (Color forecolor)

Set the color of the message text.

- public void setBackColor(Color backcolor)

Set the background color of the message.

- public void setTimer(int seconds)

Set the timer to display the control when seconds has passed. The default is 15 seconds

- public void resize(int width, int height)

Resize the control using a width and height.

- public void start()

Start the timer.

Example

```
TimerControl tc = new TimerControl("This is the title");
tc.setMessage( new String( "This is a message that will appear every 15 seconds" ) );
tc.resize( 300,300 );
tc.setTextColor( Color.yellow );
tc.setBackColor( Color.black );
tc.setTimer( 15 );
tc.start();
```

MaskField

public abstract class MaskField
extends TextField

Overview

The MaskField component is an abstract class that cannot be instantiated. This is the parent class of all edit masks in houseBlend. You can derive from this class to create your own edit masks.

Constructor

- public MaskField(int columns, String mask)

columns = The length of the field in columns.
mask = The mask format. For example, ###.## . This is dependant on the child component.

Variables

- protected String _strMask

Contains the mask for the control.

- protected String _strKeyBuffer

Contains the contents of this control.

- protected int _selStart

Contains the current start selection. This field is populated via setSelection()

- protected int _selEnd

Contains the current end selection. This field is populated via setSelection()

Methods

- public void setSelection( )

Assigns the attributes _selStart and _selEnd to the current selection.

- public int getSelectionCount()

Returns the number of characters selected in the control

- public void incrementSelection()

Increment the selection (_selStart and _selEnd).

- public void decrementSelection()

Decrement the selection (_selStart and _selEnd).

- public void cursorCurrent()

Set the cursor to the current selection as indicated by (_selStart and _selEnd). This will place the cursor (or caret) in the field and select the chars between _selStart and _selEnd inclusive.

- public void cursorRight()

Move the cursor(or caret) to the right. This method does not change _selStart or _selEnd.

- public void cursorEnd()

Move the cursor(or caret) to the end. This method does not change _selStart or _selEnd.

- public void cursorHome()

Move the cursor(or caret) to the beginning. This method does not change _selStart or _selEnd.

- public String getField()

Get the buffer contents (_strKeyBuffer) and convert it to a string.

- public void setField(String contents)

Set the contents in the component. This function obeys any formatting characters present in the control. For example, setField("221992321") in a control that has been masked as "### - ## - ####" , will show 221-99-2321 .

- public String getUnformattedField()

Returns the contents of the buffer without the formatting characters.

- public void clearField()

Clears the contents of the buffer and initializes the display.

- public static boolean isChar(char c)

Determine if 'c' is a character, returns true or false.

NumberField

public class NumberField
extends MaskField

Overview

The NumberField component allows you to create a field that will accept only numbers such as integers or decimals up to the length indicated by the format mask. See the constructor for a description of the format mask.

Constructor

- public NumberField(int columns, String mask, boolean allowNegatives)

Arguments:
columns = The length of the field in columns.
mask = The mask format.
allowNegatives = A value of false will not allow negative numbers. A value of true will allow negative numbers.

Examples for mask;
= an integer with up to 5 digits.
###.## = a decimal with 3 digits to the left and two digits to the right.

NOTE: Commas and dollar signs are not yet supported. Look for these features in upcoming releases.

- public NumberField(int columns, String mask)

See description above. The only difference with this constructor and the one above is that this constructor defaults to allow negative numbers, so you do not have to pass it as a parameter.

Methods

- public void allowNegative(boolean neg)

A value of true will allow negative numbers. A value of false will NOT allow negative numbers.

- public Integer getInteger()

Return the contents as an Integer class.

- public Double getDouble()

Return the contents as a Double class.

- public Float getFloat()

Return the contents as a Float class.

- public Long getLong()

Return the contents as a Long class.

Example

```
NumberField nbf = new NumberField( 10, "#.##" );
```

will only allow one digit to the left and two digits to the right.

StringField

public class StringField
extends MaskField

Overview

The StringField component allows you to create a field that will accept integers and/or characters combined with a formatting mask. This field is useful for phone numbers, SSN etc. See the constructor for a description of the format mask.

Constructor

- public StringField(int columns, String mask)

Arguments:
columns = The length of the field in columns.
mask = The mask format.

Mask Formatting Characters:

#	= Number
X	= character
!	= uppercase character
^	= lowercase character
x	= any character

NOTE: Any character entered in the format mask other than what is described above (such as slashes, dashes, parenthesis etc.) may be used as formatting characters. However, any number or letter entered in the format mask will not be treated as a format character but as a default that may be overridden by the user.

Examples for mask;

(###) ### - ####	= A phone number
(214) ### - ####	= A phone number with a default area code
### - ## - ####	= A Social security Number
!^^^^ !^^^^^	= Two words each letter uppercase and the rest lowercase
XXXXXXX	= Up to seven characters (no numbers allowed).
xxxxx	= Any character up to five in length

Using the constructor it would be;

```
StringField sf = new StringField( 20, "(###) ### - ####" );
```

.

com.crsoft.tools

Note: To use the following tools, you must import the package as in;
import com.crsoft.tools.*;

Class Index

Tools

public class Tools
extends Object

Overview

The class currently supports different types of formatting options for the System.out.println function, particurlarly in the area decimals, integers etc. It accomplishes this by using a trace function. In addition, you may turn all tracing ON or OFF by using one function. see below.

Methods

- public static void traceOn( )

Turns tracing on. This is the default.

- public static void traceOff( )

Turns tracing off. Note: This is helpful in that you may have many trace functions throughout your program, that you may not necessarily want to execute in a release version. By executing traceOff() in the beginning of your program, no trace function will execute. It is also helpful for debugging purposes.

- public static void setMarker(String marker)

Set the marker used by the trace function to determine where to insert Number classes.
The default is "{}". See example for more info.

- public static void trace(String strMessage)

Display a message.

- public static void trace(String strMessage, Number1, Number2,...Number6)

Display a message formatted with the Number values passed.
Arguments:
strMessage = The string message that must include marker positions for the Number values.
For example, "Cost={} Sales={} " The markers in this case are "{}" which is the default.
Number1..6 = Any Number class such as Double, Integer, Float and Long. Up to six values may be passed.

Examples

Example 1:

```
Tools.trace( "Java is great fun!" );
```

Example 2:

```
Double cost = new Double( "1.29" );
Double price= new Double( "2.10" );
Integer units= new Integer( "3" );
Tools.trace( "We purchased at {}, sold at {} for {} units. ",   cost, price, units );
```

would display, "We purchased at 1.29, sold at 2.10 for 3 units."

Example 3:

```
Tools.setMarker( "xx" );
Tools.trace( "We purchased at xx, sold at xx for xx units. ",   cost, price, units );
```

would display, "We purchased at 1.29, sold at 2.10 for 3 units."

Example 4:

```
Tools.traceOff();
```

this turns all tracing off.. which means that all tracing functions, although they may exist in the program, will not execute. see Tools.traceOn().

Setup instructions

Below are the setup instructions for houseBlend 1.0

1. Ensure that the following directories are created on your hard drive;

com\crsoft\controls
com\crsoft\tools
com\crsoft\draw

Note: These directories can exist anywhere on your hard drive. They do not have to be created at the root level.

2. Ensure that the file classes.zip that ships with houseBlend is located on your classpath. For example, CLASSPATH=c:\com\crsoft\classes.zip.

3. Ensure that you include the parent directory of the com\crsoft\controls and com\crsoft\draw in your classpath. For example, if you created the following directories off the "c:\Java\" directory as in ;

c:\Java\com\crsoft\controls
c:\Java\com\crsoft\tools
c:\Java\com\crsoft\draw

then, "c:\Java\" would be the parent directory. Your classpath would be; CLASSPATH=c:\Java\com\crsoft\classes.zip;c:\Java\;

The "c:\Java\"; in the CLASSPATH instructs the compiler to search for packages from within this parent directory.

NOTE: If you created the com\crsoft directories off the root directory, the parent directory would be "c:\;".

houseBlend 1.0 - How to order

houseBlend 1.0 costs $15.00 for one application. See registration form for multiple license.

Mail

To register by mail select the jump topic below and print the registration form.

Registration Form

CompuServe

To register through CompuServe, you need to have a CompuServe account.

- Log on to CompuServe
- "GO SWREG" and select registration ID #11280
- The registration fee will be billed to your CompuServe account. Upon registering, we will recieve an email notifying us that you have registered houseBlend. The latest version of houseBlend will be sent to you.

THE ABOVE NUMBERS ARE FOR ORDERS ONLY. Any questions about the status of the shipment of the order, registration options, product details, technical support, dealer pricing, site licenses, etc. must be directed to CreativeSoft at CIS Email 75207,2664 or the address below.

CreativeSoft
P.O. Box 1713
Dallas, TX 75221
USA

houseBlend 1.0 Registration Form

Name:

Company:

Address1:

Address2:

City: *State:*
Zip:

Country:

Tel: *Fax:*

Email:

Single Application License

1 Application ($15.00) .....
$_____

Multiple Application License

2 to 5 applications: _____ applications at $10.00 each.....$_____

> 5 applications: _____ applications at $8.00 each.....
$_____

Note: This product ships on 3.5 inch disks

Send completed form with Check or Money Order to

CreativeSoft

P.O. Box 1713
Dallas, TX 75221
USA

Where did you hear about houseBlend ?

What additional controls would you like to see ?

30 Day Money Back guarantee if your not completely satisfied with the product

Index

PropertyPage

public class PropertyPage
extends Panel

Overview

The PropertyPage control is used in conjunction with the PropertySheet control and should not be used by itself. You associate a panel(containing your controls, ie. TextFields etc) with each PropertyPage and add the PropertyPage to the PropertySheet..
See the PropertySheet example for further information.

Constructor

- public PropertyPage(String tabTitle, Panel pnl)

Constructs a PropertyPage.
Arguments:
tabTitle = The title that will appear on the tab control for this propertypage.
Panel = The panel that contains all the controls you wish to display on this propertypage.