cleverd

swing组件介绍

标签： swing import string object integer class

2007-01-22 00:45 9656人阅读 评论

分类： GUI (7)

学习swing组件,主要有三个内容
一是组件的显示,二是对组件支持的事件进行侦听,三是是自定义组件
1.JFrame
JFrame是主窗口,它和JDialog,JApplet的地位并列.但是,一个JFrame可以添加JDialog和JApplet进去它的内容面板,而反过来就不行
下面来看JFrame的例子
==========

```
package blog.swing;
import javax.swing.*;
import java.awt.event.*;

class  JFrameDemo
{
 JFrame mainFrame;
 public JFrameDemo() {
  mainFrame = new JFrame ( "JFrameDemo Title" ); //创建一个JFrame
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );//设置关闭动作
  mainFrame.setSize( 300,300 );//设置窗口大小
  mainFrame.setLocationRelativeTo(null);//使窗口显示在屏幕中央

  mainFrame.addWindowListener( new WindowListener(){
   public void windowOpened( WindowEvent e ){ System.out.println( "window opened" ); }
   public void windowClosing( WindowEvent e ){ System.out.println( "window closing" ); }
   public void windowClosed( WindowEvent e ){ System.out.println( "window closed" ); }
   public void windowIconified( WindowEvent e ){ System.out.println( "window iconified" ); }
   public void windowDeiconified( WindowEvent e ){ System.out.println( "window deiconified" ); }
   public void windowActivated( WindowEvent e ){ System.out.println( "window activated" ); }
   public void windowDeactivated( WindowEvent e ){ System.out.println( "window deactivated" ); }
  });
  mainFrame.addWindowFocusListener( new WindowFocusListener(){
   public void windowGainedFocus( WindowEvent e ){ System.out.println( "gained focus" ); }
   public void windowLostFocus( WindowEvent e ){ System.out.println( "lost focus" ); }
  });
  mainFrame.addWindowStateListener( new WindowStateListener(){
   public void windowStateChanged( WindowEvent e ){ System.out.println( "state changed" ); }
  });
```

Python打开和关闭文件 (15356)
Python类的__getitem__ (14661)
Python的reduce (11841)
Python读写文件及文件指 (10710)

评论排行
VC DLL基础教程 (9)
swing组件介绍 (5)
java里的dnd (4)
Python的字典的items(), l (4)
API挂接 (3)
使用工具提示控件 (2)
Python的reduce (2)
Python的类变量和实例变 (2)
Python统计字符串里某个 (2)
简单文件IO (2)

推荐文章
* CSDN日报20170622——《程序 Dog 的大梦想》
* 【Java高级开发工程师】近一个月的面试总结
* 一个文科生的工程师之路
* JavaWeb 与 MySQL 人鬼情未了
* PermissionsDispatcher、RxPermissions和easypermissions的使用和对比
* 每周荐书：架构、Scratch、增长黑客（评论送书）

最新评论
Python的字典的items(), keys(), v 圈奶特: 为什么 这三个函数总是先输出 中间的 ？不是无序的吗？他是 随机 输出吗？
Python的类变量和实例变量 wangzhen199009: 类变了self.__class__.***来获取；实例变量通过self.***直接获得
Python的字典的items(), keys(), v eifivewu: @igurong:字典是无序列表，list里的元素与dict是不会对应的
Python的字典的items(), keys(), v 唐小萌: 用的是哪个版本？python3.2返回的是view，而不是list。例如，dict.values()...
Python的reduce qiuliangpei: from operator import addfrom functools import redu...
Python的类变量和实例变量 itfanr: 请问类变量可以不初始化吗？
使用工具提示控件 a188037049: @a188037049:搞错了，是可以收到TTN_GETDISPINFO消息的～～【昏
使用工具提示控件 a188037049: 楼主你好，请问是不是在"ti.uFlags = TTF_TRACK | TTF_ABSOLUTE"的...
Python统计字符串里某个字符出: liaofuyan: good
java里的dnd gj447404500: 多谢分享。。。

```
 mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JFrameDemo();
 }
}
```

==========

这里出现了三个不同的窗口事件侦听器,并实现了它们所有的方法

WindowListener和WindowFocusListener都可以对窗口失去,获得焦点进行侦听,不同的是,非帧窗口和对话框窗口不能接收WindowListener的windowActivated和windodwDeactivated事件

虽然可以用WindowListener对窗口的一些状态进行侦听,但是WindowStateListener提供了更多的支持

如,WindowStateListener可以处理还原窗口的事件,可以判断一个窗口是不是在垂直和水平两个方向都可以最大化(命令提示符窗口只可以在垂直方向上最大化),而这些都是WindowListener都无能为力

2.JLabel

JLabel是一标签.在它的文本里嵌入html标签,可以简单实现一个超链接组件

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
class  JLabelDemo
{
 JFrame mainFrame;
 JLabel simpleLabel;
 public JLabelDemo() {
  mainFrame = new JFrame ( "JLabelDemo" );
  simpleLabel = new JLabel ("<html><a href=aaa>百度搜索</a></html>");//嵌入了html标签
  simpleLabel.addMouseListener( new MouseAdapter(){//添加鼠标事件侦听器,当单击标签时,打开网页
   public void mouseClicked( MouseEvent e ){
    try{
     Runtime.getRuntime().exec("cmd /c start http://www.baidu.com");
    }catch( IOException ee ){
     ee.printStackTrace();
    }
   }
  });
  simpleLabel.setCursor( new Cursor(Cursor.HAND_CURSOR) );//设置手形鼠标
  mainFrame.getContentPane().add( simpleLabel );//将标签添加到窗口
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();//使窗口自动根据添加了的组件调整大小
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JLabelDemo();
 }
}
```

3.JButton

JButton是一个按钮.它和JLabel一样的简单

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;

class JButtonDemo
{
 JFrame mainFrame;
 JButton simpleButton;
 public JButtonDemo() {
  mainFrame = new JFrame ( "JButtonDemo" );
  simpleButton = new JButton("百度搜索");
  mainFrame.getContentPane().add( simpleButton );
  simpleButton.addActionListener( new ActionListener(){//添加动作侦听器,当按钮被按下时执行这里的代码以
打开网页
   public void actionPerformed( ActionEvent e){
    try{
     Runtime.getRuntime().exec("cmd /c start http://www.baidu.com");
    }catch( IOException ee ){
     ee.printStackTrace();
    }
   }
  });
  simpleButton.setCursor( new Cursor(Cursor.HAND_CURSOR) );
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JButtonDemo();
 }
}
```

4.JTextField

一个文本框

```
package blog.swing;
import javax.swing.*;
import java.awt.event.*;

class JTextFieldDemo
{
 JFrame mainFrame;
 JTextField simpleTextField;
 public JTextFieldDemo() {
  mainFrame = new JFrame ( "JTextFieldDemo" );
  simpleTextField = new JTextField(20);//构造宽度为20个字符的文本框
  mainFrame.getContentPane().add( simpleTextField );
```

关闭

```
 simpleTextField.addActionListener( new ActionListener(){//在输入字符后按回车执行行代码,在标准输出窗口输出它的内容
  public void actionPerformed( ActionEvent e){
   System.out.println( simpleTextField.getText() );
  }
 });
 mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
 mainFrame.pack();
 mainFrame.setLocationRelativeTo(null);
 mainFrame.setVisible( true );
}
public static void main(String[] args)
{
 new JTextFieldDemo();
}
}
```

5.JTextArea

文本区域,与文本框不同的是它是多行的

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class JTextAreaDemo
{
 JFrame mainFrame;
 JTextArea simpleTextArea;
 JButton appendButton;
 public JTextAreaDemo() {
  mainFrame = new JFrame ( "JTextAreaDemo" );
  simpleTextArea = new JTextArea(10,20);//创建一个显示10行20列的文本域
  simpleTextArea.setLineWrap(true);//设置它自动换行
  simpleTextArea.setWrapStyleWord(true);//设置它自动换行时根据单词换行,而不是根据字符
  appendButton = new JButton ("append text to the text area");
  mainFrame.getContentPane().add( simpleTextArea,BorderLayout.PAGE_START );
  mainFrame.getContentPane().add( appendButton,BorderLayout.PAGE_END );
  appendButton.addActionListener( new ActionListener(){
   public void actionPerformed( ActionEvent e){
    simpleTextArea.append("button appended text here");
    System.out.println( simpleTextArea.getText() );
   }
  });
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JTextAreaDemo();
 }
```

关闭

```
}
```

6.JPasswordField

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class JPasswordFieldDemo
{
 JFrame mainFrame;
 JPasswordField simplePasswordField;
 public JPasswordFieldDemo() {
  mainFrame = new JFrame ( "JPasswordFieldDemo" );
  simplePasswordField = new JPasswordField(10);
  simplePasswordField.setEchoChar('*');//设定要显示的字符
  mainFrame.getContentPane().add( simplePasswordField );
  simplePasswordField.addActionListener( new ActionListener(){//回车时执行的动作
   public void actionPerformed( ActionEvent e){
    char[] input = simplePasswordField.getPassword();
    for( char c : input )
     System.out.print( c );
   }
  });
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JPasswordFieldDemo();
 }
}
```

7.JPanel

一个面板.一般用作控制组件的布局.

```
package blog.swing;
import javax.swing.*;

class JPanelDemo
{
 JFrame mainFrame;
 JPanel simplePanel;
 JButton simpleButton;
 JLabel simpleLabel;
 public JPanelDemo() {
  mainFrame = new JFrame ( "JPanelDemo" );
  simplePanel = new JPanel();
  simpleButton = new JButton ("button");
  simpleLabel = new JLabel ("label");
  simplePanel.add( simpleLabel );
```

```
 simplePanel.add( simpleButton );
 mainFrame.getContentPane().add( simplePanel );
 mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
 mainFrame.pack();
 mainFrame.setLocationRelativeTo(null);
 mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JPanelDemo();
 }
}
```

8.JCheckBox

复选框

```
package blog.swing;
import javax.swing.*;
import java.awt.event.*;

class JCheckBoxDemo implements ItemListener
{
 JFrame mainFrame;
 JPanel mainPanel;
 JCheckBox simpleCheckBox1;
 JCheckBox simpleCheckBox2;
 public JCheckBoxDemo() {
  mainFrame = new JFrame ( "JCheckBoxDemo" );
  mainPanel = new JPanel ();
  simpleCheckBox1 = new JCheckBox("checkbox1");
  simpleCheckBox1.setMnemonic('1');
  simpleCheckBox1.addItemListener(this);
  simpleCheckBox2 = new JCheckBox("checkbox2");
  simpleCheckBox2.setMnemonic('2');
  simpleCheckBox2.addItemListener(this);
  mainPanel.add(simpleCheckBox1);
  mainPanel.add(simpleCheckBox2);
  mainFrame.getContentPane().add( mainPanel );
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public void itemStateChanged( ItemEvent e ){
  JCheckBox cb = (JCheckBox)e.getSource();
  if( cb== simpleCheckBox1 )
   System.out.println( "simpleCheckBox1" );
  else
   System.out.println( "simpleCheckBox2" );
 }
 public static void main(String[] args)
 {
  new JCheckBoxDemo();
```

关闭

```
 }
}
```

9.JRadioButton

单选按钮.单选按钮要用到ButtonGroup.添加到同一个ButtonGroup的单选按钮表示在它们之间

ButtonGroup里的单选按钮相互之间的选择不受影响.

```
package blog.swing;
import javax.swing.*;
import java.awt.event.*;

class JRadioButtonDemo implements ItemListener
{
 JFrame mainFrame;
 JPanel mainPanel;
 ButtonGroup buttonGroup;
 JRadioButton simpleRadioButton1;
 JRadioButton simpleRadioButton2;
 public JRadioButtonDemo() {
  mainFrame = new JFrame ( "JRadioButtonDemo" );
  mainPanel = new JPanel ();
  simpleRadioButton1 = new JRadioButton("RadioButton1");
  simpleRadioButton1.setMnemonic('1');
  simpleRadioButton1.addItemListener(this);
  simpleRadioButton2 = new JRadioButton("RadioButton2");
  simpleRadioButton2.setMnemonic('2');
  simpleRadioButton2.addItemListener(this);
  buttonGroup = new ButtonGroup();
  buttonGroup.add(simpleRadioButton1);
  buttonGroup.add(simpleRadioButton2);
  mainPanel.add(simpleRadioButton1);
  mainPanel.add(simpleRadioButton2);
  mainFrame.getContentPane().add( mainPanel );
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public void itemStateChanged( ItemEvent e ){
  JRadioButton cb = (JRadioButton)e.getSource();
  if( cb== simpleRadioButton1 )
   System.out.println( "simpleRadioButton1" );
  else
   System.out.println( "simpleRadioButton2" );
 }
 public static void main(String[] args)
 {
  new JRadioButtonDemo();
 }
}
```

10.JScrollPane

JScrollPane由四个角,两个头部,和一个视口组成.四个角和两个头部都是由Component组成.四个角并不是总是显示

出来的.左上角只有当两个头部同时存在才显示,右下角只有两个滚动条同时出现才会出现.其他两个角可同理知道什么时候会出现.视口是显示内容的部分,由JViewport对象表示.而真正显示的内容,由JViewport的view表示

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class JScrollPaneDemo
{
 JFrame mainFrame;
 JScrollPane simpleScrollPane;
 JTextArea simpleTextArea;
 JButton changeViewport;
 public JScrollPaneDemo() {
  mainFrame = new JFrame ( "JScrollPaneDemo" );
  simpleTextArea = new JTextArea(10,20);
  simpleScrollPane = new JScrollPane( simpleTextArea );//创建一个滚动窗格,里面显示的内容是文本区域
  simpleScrollPane.setRowHeaderView( new JLabel ("this is a row header") );//设置行标题
  simpleScrollPane.setColumnHeaderView( new JLabel ("this is a column header") );//设置列标题
  simpleScrollPane.setCorner( JScrollPane.LOWER_RIGHT_CORNER,new JButton("corner") );//设置右下角
  simpleScrollPane.setCorner( JScrollPane.UPPER_LEFT_CORNER,new JButton("corner") );//设置左上角
  changeViewport = new JButton ("changeViewport");
  changeViewport.addActionListener( new ActionListener(){//当单击按钮时,滚动窗口显示的内容变为另一个文本区域
    public void actionPerformed( ActionEvent e){
     simpleScrollPane.getViewport().setView( new JTextArea("changeViewpot") );
    }
  });
  mainFrame.getContentPane().add( simpleScrollPane,BorderLayout.PAGE_START );
  mainFrame.getContentPane().add( changeViewport,BorderLayout.PAGE_END );
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JScrollPaneDemo();
 }
}
```

现在开始讲相对复杂一点的组件

11.JList

JList是一个列表.这里将首次引入Model这个概念,中文翻译是模型,不好理解,其实就是数据的意思.JList用ListModel保存它的数据.简单应用可以用实现了ListModel的AbstraceListModel的子类DefaultListModel来保存它的数据(尽管它的名字有Default,但是只有你使用了它创建JList,它才存在.即如果你没有用DefaultListModel创建JList,则你用getModel()返回的ListModel不能强制转换为DefaultListModel).

很多组件的Model都有这种结构,即 XXXModel--AbstractXXXModel--DefaultXXXModel

下面讲一下DefaultListModel,它的方法基本上和Vector<E>一样,另外添加了一个Object getElementAt(int index)

添加元素: void add(int index, Object element),void addElement(Object obj),void insertElementAt(Object obj, int index)

删除元素: Object remove(int index),boolean removeElement(Object obj),void removeElementAt(int index),

void removeAllElements(),void removeRange(int fromIndex, int toIndex),void clear()
查询元素: Object elementAt(int index),Object get(int index),Object getElementAt(int index), Object firstElement(),Object lastElement()
修改元素: Object set(int index, Object element),void setElementAt(Object obj, int index)
JList自身没有对单个元素处理的方法,只有void setListData(Object[] listData),void setListData(Vector listData),void setModel(ListModel model)
来设置全部元素的方法
当要对单个元素进行处理时,用ListModel getModel()返回Model后,再对Model操作
通过void addListDataListener(ListDataListener l)可以在ListModel被修改时被通知
和JList相关的还有ListSelectionModel,它管理JList的选择项.它没有AbstractListSelectionModel
一个DefaultListSelectionModel这个实现类,
JList默认也是使用的这个实现类. 和上面讲的DefaultListModel不同,这个Model不用你自己创建就已有
通过void addListSelectionListener(ListSelectionListener x)可以在选择改变时被通知.
void setSelectionMode(int selectionMode)可以设置多选(待续或非连续)或是单选
DefaultListSelectionModel没有返回选择了的元素的方法,它只负责去选择哪些项
修改选择项的方法:
void addSelectionInterval(int index0, int index1),
void removeSelectionInterval(int index0, int index1)
void setSelectionInterval(int index0, int index1)
void clearSelection()
下面这两个方法在ListModel被更改时更改选择项会更方便,因为不用根据ListModel的变动计算索引
void removeIndexInterval(int index0, int index1)
void insertIndexInterval(int index, int length, boolean before)
另外的一些方法
int getMaxSelectionIndex(),int getMinSelectionIndex()
boolean isSelectedIndex(int index),boolean isSelectionEmpty()
很多在DefaultListSelectionModel里的方法,在JList自身里也有.例如上面的XXXSelectionInnterval(...)和addListSelectionListener(...)
另外它还有:
int getSelectedIndex(),int[] getSelectedIndices()
Object getSelectedValue(), Object[] getSelectedValues()
void setSelectedIndex(int index),void setSelectedIndices(int[] indices)
void setSelectedValue(Object anObject, boolean shouldScroll)
最后一个内容是,自定义JList
默认JList显示的内容是String.在创建它的时候,如果你把其他非String的对象传给它要它显示,它会调用toString,然后显示返回的String.
通过void setCellRenderer(ListCellRenderer cellRenderer)可以自定义JList显示内容的类型.
参数是一个"渲染器".很多组件在自定义的时候都是用渲染器来实现的.它只一个方法:
Component getListCellRendererComponent(JList list, Object value, int index, boolean isSelected, boolean cellHasFocus)
下面举例说明一下它如何工作:假设你调用了setCellRenderer,而且把"label"这个String传给JList要它显示.那么"label"这个值就会传给上面这个方法的value这个值.
这时你可以用"label"构造一个组件,然后返回.例如,你可以JLabel label = new JLabel((String)value); return label.这样,在JList就会显示一个JLabel了.

```
package blog.swing;
import java.awt.*;
import javax.swing.event.*;
import javax.swing.*;
class  JListCustomDemo
{
 JFrame mainFrame;
 JList simpleList;
 public JListCustomDemo(){
  mainFrame = new JFrame ("JListCustomDemo");
```

关闭

```
 final DefaultListModel model = new DefaultListModel();
 model.addElement("button1");
 model.addElement("button2");
 simpleList = new JList(model);
 simpleList.setCellRenderer( new CustomListCellRenderer() );

 simpleList.addListSelectionListener( new ListSelectionListener(){
  public void valueChanged( ListSelectionEvent e){
   System.out.println( model.getElementAt( simpleList.getSelectedIndex() ) );
  }
 });

 mainFrame.add(simpleList);
 mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
 mainFrame.setLocationRelativeTo(null);
 mainFrame.pack();
 mainFrame.setVisible( true );
}
public static void main(String[] args)
{
 new JListCustomDemo();
}
}
class CustomListCellRenderer implements ListCellRenderer{
 public Component getListCellRendererComponent(
  JList list,
  Object value,
  int index,
  boolean isSelected,
  boolean cellHasFocus
  ){
  return new JButton( (String)value );
 }

}
```

```
package blog.swing;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class JListDemo
{
 JFrame mainFrame;
 JList simpleList;
 JButton changeSelections;
 public JListDemo() {
  mainFrame = new JFrame ( "JListDemo" );
/*  Vector<String> listData = new Vector<String>();
  listData.add("data1");
```

关闭

```
  listData.add("data2");
  listData.add("data3");
  listData.add("data4");
  simpleList  = new JList( listData );
*/
  DefaultListModel dlm = new DefaultListModel();
  dlm.addElement("data1");
  dlm.addElement("data2");
  dlm.addElement("data3");
  dlm.addElement("data4");
  simpleList = new JList( dlm );
  changeSelections = new JButton ("changeSelections");
  changeSelections.addActionListener( new ActionListener(){
   public void actionPerformed( ActionEvent e){
    DefaultListSelectionModel dlsm = (DefaultListSelectionModel)simpleList.getSelectionModel();
    //dlsm.addSelectionInterval(0,1);
    //dlsm.removeSelectionInterval(0,1);
    dlsm.setSelectionInterval(0,1);
/*    DefaultListModel dlm = (DefaultListModel)simpleList.getModel();
    dlm.remove(0);
    dlm.remove(1);
*/
   }
  });
  mainFrame.getContentPane().add( simpleList,BorderLayout.PAGE_START );
  mainFrame.getContentPane().add( changeSelections,BorderLayout.PAGE_END );
  mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  mainFrame.pack();
  mainFrame.setLocationRelativeTo(null);
  mainFrame.setVisible( true );
 }
 public static void main(String[] args)
 {
  new JListDemo();
 }
}
```

12.JComboBox

组合框和JList很相似,它们都是用ListModel来保存数据.默认它是使用实现了ListModel的子接口ComboBoxModel的DefaultComboBoxModel来保存数据的.与JList的情况不同,一个JComboBox总是有一个Model来保存数据的,而JList则不然.

DefaultComboBoxModel的方法:

添加元素:addElement(Object object),insertElementAt(Object object,int index)

删除元素:removeElement(Object object),removeElementAt(int index),removeAllElements()

获取元素:getElementAt(int index)

和选择有关的:getSelectedItem(),setSelectedItem(Object object)

此外还有getSize(),getIndexOf(Object object)

JComboBox自身也有一些处理项的方法:

void addItem(Object anObject),void insertItemAt(Object anObject, int index)

void removeItem(Object anObject),void removeItemAt(int anIndex),void removeAllItems()

Object getItemAt(int index)

int getItemCount()

以上基本上是把DefaultComboBoxModel里的方法的Element改为Item

int getSelectedIndex(),Object getSelectedItem(),Object[] getSelectedObjects()

关闭

通过在JComboBox上添加ActionListener,可以在选择改变了的时候作出响应.
最后是自定义组合框.通过调用和JList一样的void setRenderer(ListCellRenderer aRenderer) 就可以自定义组合框.
下面的例子示范了在JComboBox里显示图片

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.*;

class JComboBoxDemo
{
   JFrame mainFrame;
   JComboBox simpleComboBox;
   public JComboBoxDemo() {
      mainFrame = new JFrame ( "JComboBoxDemo" );
      Vector<String> cbData = new Vector<String>();
      cbData.add("images/Pig.gif");
      cbData.add("images/Bird.gif");
      cbData.add("images/Dog.gif");
      cbData.add("images/Cat.gif");
      simpleComboBox = new JComboBox( cbData);
      simpleComboBox.setPreferredSize( new Dimension(200,130) );
      simpleComboBox.setMaximumRowCount(2);
      simpleComboBox.setRenderer( new CustomComboBoxRenderer() );
      mainFrame.getContentPane().add( simpleComboBox );
      simpleComboBox.addActionListener( new ActionListener(){
         public void actionPerformed( ActionEvent e){
            System.out.println( "selection changed" );
            System.out.println( simpleComboBox.getSelectedItem() );
         }
      });
      simpleComboBox.setCursor( new Cursor(Cursor.HAND_CURSOR) );
      mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      mainFrame.pack();
      mainFrame.setLocationRelativeTo(null);
      mainFrame.setVisible( true );
   }

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

   class CustomComboBoxRenderer extends JLabel implements ListCellRenderer{
      CustomComboBoxRenderer(){
         setOpaque(true);
         setHorizontalAlignment(CENTER);
         setVerticalAlignment(CENTER);

      }
      public Component getListCellRendererComponent(
         JList list,
         Object value,
         int index,
```

```
            boolean isSelected,
            boolean cellHasFocus)
        {
            if (isSelected) {
                setBackground(list.getSelectionBackground());
                setForeground(list.getSelectionForeground());
            } else {
                setBackground(list.getBackground());
                setForeground(list.getForeground());
            }
            String imageFileName = (String)value;
            ImageIcon labelIcon = new ImageIcon( imageFileName );
            setText( imageFileName.substring(imageFileName.lastIndexOf('/')+1) );
            setIcon( labelIcon );
            return this;
        }
    }
}
```

13.JFileChooser

JFileChooser代表一个打开/保存文件对话框

三个较简单的构造函数:

JFileChooser(),JFileChooser(File currentDirectory),JFileChooser(String currentDirectoryPath)

构造对象以后,调用int showOpenDialog(Component parent)或int showSaveDialog(Component parent)显示打开/保存对话框

调用int showDialog(Component parent, String approveButtonText) 显示自定义打开/保存按钮文字的对话框

三个方法的返回值都是整型.当按下打开/保存时,返回APPROVE_OPTION,否则返回CANCEL_OPTION

可以用javax.swing.filechooser.FileFilter来过滤不需要显示的文件.它只有两个方法

boolean accept(File f) 通过通过判断f的后缀来决定显示与否.要显示则返回true,否则返回false

String getDescription() 返回对这个过滤器的描述

和FileFilter有关的方法:

void setFileFilter(FileFilter filter),void addChoosableFileFilter(FileFilter filter)

当我们选择的文件改变(但是未按下打开或保存按钮)时,会有Java.beans.PropertyChangeEvent产生

通过void addPropertyChangeListener(PropertyChangeListener listener)可以对此事件进行侦听

PropertyChangeEvent的方法有:

Object getNewValue(), Object getOldValue(), String getPropertyName()

Object getPropagationId(),void setPropagationId(Object propagationId)

通过void setAccessory(JComponent c)可以向JFileChooser添加自定义的部分.

其他有用的方法:

File getSelectedFile(),File[] getSelectedFiles()

void setFileSelectionMode(int):FILES_ONLY,DIRECTORIES_ONLY,FILES_AND_DIRECTORIES

void setMultiSelectionEnabled(boolean),setAcceptAllFileFilterUsed(boolean)

void setCurrentDirectory(File) ,void setFileHidingEnabled(boolean)

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import java.io.File;
import java.beans.*;

class JFileChooserDemo
{
```

关闭

```
JFileChooser simpleFileChooser;
JScrollPane previewScrollPane;
JLabel previewLabel;
public JFileChooserDemo() {
    simpleFileChooser = new JFileChooser();
    previewLabel = new JLabel ();
    previewLabel.setHorizontalAlignment(SwingConstants.CENTER);
    previewScrollPane = new JScrollPane ( previewLabel );
    previewScrollPane.setPreferredSize(new Dimension(100,10));
    simpleFileChooser.setAccessory( previewScrollPane );
    simpleFileChooser.addChoosableFileFilter( new GifFileFilter() );
    simpleFileChooser.addChoosableFileFilter( new PngFileFilter() );
    simpleFileChooser.addChoosableFileFilter( new JpgFileFilter() );
    simpleFileChooser.addPropertyChangeListener( new PropertyChangeListener(){
        public void propertyChange( PropertyChangeEvent e ){
            if ( JFileChooser.SELECTED_FILE_CHANGED_PROPERTY.equals( e.getPropertyName() ) ){
                File newSelectedFile = (File)e.getNewValue();
                if( newSelectedFile != null){
                    ImageIcon icon = new ImageIcon( newSelectedFile.getPath() );
                    previewLabel.setIcon( icon );
                }
            }
        }
    });
    simpleFileChooser.showOpenDialog(null);
    //simpleFileChooser.showDialog(null,"自定义按钮文字");
}
class GifFileFilter extends FileFilter{
    public boolean accept( File f ){
        return f.getName().endsWith(".gif");
    }
    public String getDescription(){
        return "Gif files(.gif)";
    }
}
class PngFileFilter extends FileFilter{
    public boolean accept( File f ){
        return f.getName().endsWith(".png");
    }
    public String getDescription(){
        return "Png files(.png)";
    }
}
class JpgFileFilter extends FileFilter{
    public boolean accept( File f ){
        return f.getName().endsWith(".jpg");
    }
    public String getDescription(){
        return "Jpg files(.jpg)";
    }
}
public static void main(String[] args)
{
    new JFileChooserDemo();
```

关闭

```
    }
}
```

14.JColorChooser

一个颜色选择器.它的构造方法有:

JColorChooser(),JColorChooser(Color initialColor),JColorChooser(ColorSelectionModel mo

创建了对象以后,可以调用将它添加到JFrame,JPanel等其他容器里面.

也可调用它的静态方法static Color showDialog(Component component, String title, Color initi

模态的对话框.

它用ColorSelectionModel来管理选择的颜色.通过调用它的void addChangeListener(ChangeLi

在选择的颜色变化时作出反应.

调用void setSelectedColor(Color color)更改选择的颜色,Color getSelectedColor()取得选择的颜色

JColorChooser自身的获取颜色的方法是Color getColor().

JColorChooser由颜色选择面板和预览面板组成,所以自定义它,就是对这两部分作文章

void setPreviewPanel(JComponent) 可以设定预览面板.setPreviewPanel(new JPanel())可以移除

数设为null,即setPreviewPanel(null)可以将预览面板设回默认.

void setChooserPanels(AbstractColorChooserPanel[])和void addChooserPanel(AbstractColorChooserPanel)分别可以设置和添加颜色选择面板

void removeChooserPanel(AbstractColorChooserPanel)可以删除颜色选择面板

AbstractColorChooserPanel代表差一个颜色选择面板,它在javax.swing.colorchooser包里.它有五个抽象方法,而其中两个是目前不用的.所以只需定义下面三个:

String getDisplayName()面板显示的文本.

void buildChooser() 负责创建一个颜色选择面板

void updateChooser() 负责更新显示颜色面板

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;

class JColorChooserDemo
{
    JFrame mainFrame;
    JColorChooser simpleColorChooser;
    JLabel sampleLabel;
    public JColorChooserDemo() {
        mainFrame = new JFrame ( "JColorChooserDemo" );
        sampleLabel = new JLabel ("sample");
        simpleColorChooser = new JColorChooser();
        simpleColorChooser.getSelectionModel().addChangeListener( new ChangeListener(){
            public void stateChanged( ChangeEvent e ){
                sampleLabel.setForeground( simpleColorChooser.getColor() );
            }
        });
        mainFrame.getContentPane().add( simpleColorChooser,BorderLayout.PAGE_START );
        mainFrame.getContentPane().add( sampleLabel,BorderLayout.PAGE_END );
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    public static void main(String[] args)
```

关闭

```
    {
        new JColorChooserDemo();
    }
}
```

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
import javax.swing.colorchooser.*;
class JColorChooserCustomDemo
{
    JFrame mainFrame;
    JColorChooser simpleColorChooserCustom;
    JLabel sampleLabel;
    public JColorChooserCustomDemo() {
        mainFrame = new JFrame ( "JColorChooserCustomDemo" );
        sampleLabel = new JLabel ("sample");
        simpleColorChooserCustom = new JColorChooser();
        simpleColorChooserCustom.getSelectionModel().addChangeListener( new ChangeListener(){
            public void stateChanged( ChangeEvent e ){
                sampleLabel.setForeground( simpleColorChooserCustom.getColor() );
            }
        });
        AbstractColorChooserPanel accps[] = { new CustomColorChooserPanel(),
                new CustomColorChooserPanel()};
        simpleColorChooserCustom.setChooserPanels(accps);
        mainFrame.getContentPane().add( simpleColorChooserCustom,BorderLayout.PAGE_START );
        mainFrame.getContentPane().add( sampleLabel,BorderLayout.PAGE_END );
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    class CustomColorChooserPanel extends AbstractColorChooserPanel implements ActionListener{
        JButton redButton;
        JButton greenButton;
        JButton blueButton;
        public CustomColorChooserPanel(){
            this.redButton = new JButton("red");
            this.greenButton = new JButton("green");
            this.blueButton = new JButton("blue");
            redButton.addActionListener(this);
            greenButton.addActionListener(this);
            blueButton.addActionListener(this);
        }
        public void actionPerformed(ActionEvent ae) {
            if((JButton)ae.getSource() == redButton){
                getColorSelectionModel().setSelectedColor(Color.red);
            }else{
                if((JButton)ae.getSource() == greenButton){
                    getColorSelectionModel().setSelectedColor(Color.green);
```

关闭

```
            }
            else{
                getColorSelectionModel().setSelectedColor(Color.blue);
            }
        }
    }
    public void buildChooser(){
        add(redButton);
        add(greenButton);
        add(blueButton);
    }
    public void updateChooser(){}
    public String getDisplayName(){
        return "CustomPanel";
    }
    public Icon getSmallDisplayIcon() {
        return null;
    }
    public Icon getLargeDisplayIcon() {
        return null;
    }
}
public static void main(String[] args)
{
    new JColorChooserCustomDemo();
}
}
```

15.JSlider

JSlider是一个滑动条.其实它还是比较容易使用的

构造方法比较多:

JSlider(),JSlider(int orientation),JSlider(int min, int max)

JSlider(int min, int max, int value) ,JSlider(int orientation, int min, int max, int value)

通过void addChangeListener(ChangeListener l) 可以在它的值改变时作出反应

最需要操作的是它的刻度.下面是和刻度有关的方法:

void setMajorTickSpacing(int n),void setMinorTickSpacing(int n) ,void setLabelTable(Dictionary labels)

void setPaintLabels(boolean b),void setPaintTicks(boolean b)

另外一些常用方法:

setValue(int n),void setOrientation(int orientation),void setMinimum(int minimum),void setMaximum(int maximum)

int getValue() ,int getOrientation()

```
package blog.swing;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
import java.util.*;

class JSliderDemo
{
    JFrame mainFrame;
    JSlider simpleSlider;
    public JSliderDemo() {
```

关闭

```
        mainFrame = new JFrame ( "JSliderDemo" );
        simpleSlider = new JSlider(SwingConstants.VERTICAL);

        Hashtable sliderLabelHashTable = simpleSlider.createStandardLabels(10);
        for(int i=0; i<sliderLabelHashTable.size()*10; i+=10){
            sliderLabelHashTable.put(new Integer(i),new JLabel("label " + i));
        }
        simpleSlider.setLabelTable(sliderLabelHashTable);
        simpleSlider.setPaintLabels(true);
        simpleSlider.setMinorTickSpacing(5);
        simpleSlider.setMajorTickSpacing(10);
        simpleSlider.setPaintTicks(true);

        mainFrame.getContentPane().add( simpleSlider );
        simpleSlider.addChangeListener( new ChangeListener(){
            public void stateChanged( ChangeEvent e){
                System.out.println( simpleSlider.getValue() );
            }
        });
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    public static void main(String[] args)
    {
        new JSliderDemo();
    }
}
```

16.JLayeredPane

JFrame,JApplet,JDialog,JInternalFrame其实是由好几部分组成的

JFrame,JApplet,JDialog,JInternalFrame

|__JRootPane:根层

|__GlassPane(Component):GlassPane是用组件实现的,没有JGlassPane

|__JLayeredPane:分层.在这里可以定义组件的叠放次序

|__ContentPane:ContentPane和GlassPane一样,只一个抽象层,没有对应的类.在它们上面可以放组件

|__JMenuBar

但是,我们一般不直接使用JRootPane的JLayeredPane,而是自己定义一个.

它只有一个构造方法,无参的JLayeredPane()

用Component add(Component comp, int index)将组件添加到其上并指定层级,层级大的组件显示在小的上面.

以后可以动态改变所在层:

void moveToBack(Component c),void moveToFront(Component c):这两个方法改变的是层内的位置,而不是层间的位置

void setPosition(Component c, int position):设置层内的位置.0表示最上面,-1表示最下面

void setLayer(Component c, int layer)

void setLayer(Component c, int layer, int position) 设置组件的层级,position是指在层内的位置.

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
```

关闭

```
class JLayeredPaneDemo
{
    JFrame mainFrame;
    JLayeredPane layeredPane;
    JLabel blackLabel;
    JComboBox layerList;
    public JLayeredPaneDemo() {
        mainFrame = new JFrame ( "JLayeredPaneDemo" );
        layeredPane = new JLayeredPane();
        layeredPane.setPreferredSize( new Dimension(200,300) );
        Color[] colors = { Color.red, Color.green, Color.yellow, Color.blue };
        for( int i=0; i<4; i++){
            JLabel label = createLabel(i,colors[i]);
            layeredPane.add( label, new Integer(i) );
        }
        blackLabel = new JLabel ( "lll" );
        blackLabel.setBounds( 15,40,120,120);
        blackLabel.setOpaque( true );
        blackLabel.setBackground( Color.black );
        layeredPane.add( blackLabel, new Integer(1), 0 );
        layeredPane.addMouseMotionListener( new MouseInputAdapter(){
            public void mouseMoved( MouseEvent e ){
                blackLabel.setBounds( e.getX(),e.getY(),120,120 );
            }
        } );

        String layerListItem[] = { "PUT THE BLACK LABEL AT LAYER 0",
            "PUT THE BLACK LABEL AT LAYER 1","PUT THE BLACK LABEL AT LAYER 2",
            "PUT THE BLACK LABEL AT LAYER 3","PUT THE BLACK LABEL AT LAYER 4" };
        layerList = new JComboBox( layerListItem );
        layerList.addActionListener( new ActionListener(){
            public void actionPerformed( ActionEvent e ){
                layeredPane.setLayer( blackLabel,layerList.getSelectedIndex() );
            }
        } );
        mainFrame.getContentPane().add( layerList ,BorderLayout.PAGE_END);
        mainFrame.getContentPane().add( layeredPane, BorderLayout.PAGE_START);
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.setSize( 400,400 );
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    JLabel createLabel(int i,Color color){
        JLabel label = new JLabel ( ""+i);
        label.setOpaque( true );
        label.setBounds( i*60,i*60,140,140 );
        label.setBackground( color );
        return label;
    }
    public static void main(String[] args)
    {
        new JLayeredPaneDemo();
```

关闭

```
    }
}
```

17.JInternalFrame

JFrame不能添加JFrame到自己的内容面板.那么,如何实现多文档程序呢?用JInternalFrame可以实现

一般的做法是,把JInternalFrame添加到JDesktopPane,然后把JDesktopPane作为JFrame的内容

(ContentPane)

JInternalFrame(String title[, boolean resizable[, boolean closable[, boolean maximizable[, bo

[]里面的表示可以省略

```
package blog.swing;
import java.awt.*;
import javax.swing.*;

class  JInternalFrameDemo
{
    JFrame mainFrame;
    JDesktopPane desktop;
    public JInternalFrameDemo(){
        mainFrame = new JFrame ("JInternalFrame");
        desktop = new JDesktopPane();

        for(int i=0; i<4; i++){
            JInternalFrame internalFrame = new JInternalFrame();
            internalFrame.setVisible( true );
            internalFrame.setLocation(i*40,i*40);
            internalFrame.getContentPane().add( new JButton ("button") );
            internalFrame.pack();
            desktop.add(internalFrame);
        }
//        desktop.setDragMode(JDesktopPane.LIVE_DRAG_MODE);
        desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
        mainFrame.setContentPane(desktop);
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.setSize(400,400);
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }

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

18.GlassPane

GlassPane可以用来截获输入事件(键盘和鼠标).没有JGlassPane

可以调用JFrame的void setGlassPane(Component glassPane)来设置GlassPane

默认GlassPane是不可见的,要调用getGlassPane().setVisible(true)使其可见

```
package blog.swing;
import java.awt.*;
```

关闭

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

class GlassPaneDemo
{
   JFrame mainFrame;
   JPanel mainPanel;
   JButton button;
   public GlassPaneDemo() {
      mainFrame = new JFrame (  );
      mainPanel = new JPanel ();
      button = new JButton ("button");
      //mainFrame.setGlassPane( mainPanel );
      mainPanel.add( button );
      mainFrame.getContentPane().add( mainPanel );
      mainFrame.setGlassPane( new MyGlassPane() );
      mainFrame.getGlassPane().setVisible( true );
      mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      mainFrame.setSize( 300,400 );
      mainFrame.setLocationRelativeTo(null);
      mainFrame.setVisible( true );
   }
   private class MyGlassPane extends JComponent {
      Point point = new Point(10,10);
      public MyGlassPane(){
         addMouseListener( new MouseAdapter(){
            public void mouseClicked( MouseEvent e ){
               point = new Point( e.getX(),e.getY() );
               repaint();
            }
         } );
      }
      public void paintComponent( Graphics g ){
         g.setColor( Color.red );
         g.fillOval( point.x,point.y,20,20 );
      }
   }
   public static void main(String[] args)
   {
      SwingUtilities.invokeLater( new Runnable(){
         public void run(){
            new GlassPaneDemo();
         }
      });
   }
}
```

19.JProgressBar

进度条.当一个任务要较长时间来完成时,我们可以用一个进度条来表示任务的完成进度.

在讲进度条的用法之前,我们先来看javax.swing.SwingWorker类的用法.我们将用这个类来模拟我们的"较长的任务".

在java中,组件是在一个用户界面线各里绘制的.如果我们把一个用时较长的任务放到这个线程来实现(例如我们把一个用时较长的任务放到一个按钮的

actionPerformed(...)),那么用户界面将会僵死(例如包含那个按钮的窗口的菜单将暂不可用,而要等actionPerform完成返回后才可用).

通过SwingWorker,我们可以把这个较长的任务放到另外一个线程来实现,这样用户界面就不会僵死了.

关闭

这个SwingWorker是jdk1.6才引进的,.之前也有一个SwingWorker.但是它们有所不同:旧的SwingWorker是可重用的,而新的不能;另外它们的方法的名字也不一样.
SwingWorker主要有六个方法doInBackground,get,done,publish,process,execute
SwingWorker是一个泛型类,有两个类型参数.第一类型参数就是doInBackground和get的返回值的类型,而第二个类型参数是publish的形参类型.....
我们的较长任务是在doInBackground里完成的,doInBackground的返回值可以用get取得.get有
参数代表的是等待doInBackground完成的时间,无参表示直到doInBackground完成,get才返回.
get要等到doInBackground完成才知道任务完成情况.怎么了解任务的执行过程呢?publish可以
参数个数是任意的,但是,每一个参数的类型都必须是SwingWorker的第二个类型参数指定的类型
我们用publish向外界发布任务执行的情况,而用process来收集这些情况.process是在事件分发
执行之前,SwingWorker的publish可能已经执行多次,所以process的参数是一个List,这样就可以
情况.
done是在doInBackground执行完成之后执行的.
execute是使doInBackground开始执行.
以上的方法只有doInBackground是必须自己实现的,其他都是可选的.
下面是一个例子.在这个例子中有两个按钮.第一个按钮使SwingWorker开始工作,第二个按钮调用get方法取得doInBackground的返回值.
在SwingWorker开始工作以后但是还没有结束前按下第二个按钮,可以看到界面僵死了,这是因为我们在按钮的actionPerformed(在事件分发线程里调用)里调用了get,而无参的get在doInBackground返回前是不会返回的.
在doInBackground完成之后,我们再按下第一个按钮,程序并没有变化.这是因为SwingWorker是不可重用的.所以我们用匿名内部类来实现我们的SwingWorker.
在程序中我们还用到了publish和process.在process中,我们输出publish的结果.按下第二个按钮之前,process每次只输出一个值,而在doInBackground返回之前按下第二个按钮,因为process是在事件分发线程里执行的,而get阻塞了事件分发线程,所以process不再输出了,而是等到最后连续输出数个值.

```
import javax.swing.*;
import java.awt.event.*;
import java.util.*;
import java.util.concurrent.ExecutionException;

class SwingWorkerTest
{
    JFrame mainFrame;
    JPanel mainPanel;
    JButton button;
    JButton getButton;
    public SwingWorkerTest() {
        mainFrame = new JFrame (  );
        mainPanel = new JPanel ();
        final javax.swing.SwingWorker<Integer,Integer> worker =
                    new javax.swing.SwingWorker<Integer,Integer>(){
            public Integer doInBackground(){
                int coutn = 0;
                while( (coutn++)<10 ){
                    try{
                        System.out.println( "doInBackground() is doing a long job" );
                        Thread.sleep(1000);
                        publish( new Integer( (int)(Math.random()*1000) ) );
                    }catch( InterruptedException e ){
                        e.printStackTrace();
                    }
                }
                return new Integer(3);
```

关闭

```
        }
        @Override
        public void process(List<Integer> integers){
            int i = 0;
            Iterator iterator = integers.iterator();
            while( iterator.hasNext() ){
                i++;
                Integer integer = (Integer)iterator.next();
                System.out.println( "在process输出publish的值"+i+"   "+integer );
            }
        }
    };
    button = new JButton ("start");
    button.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent e){
            worker.execute();
        }
    });
    getButton = new JButton ("Get");
    getButton.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent e){
            try{
                System.out.println( "doInBackground的返回值: "+worker.get() );
            }catch( InterruptedException ie ){
                ie.printStackTrace();
            }catch( ExecutionException ee ){
                ee.printStackTrace();
            }
        }
    });
    mainPanel.add(button);
    mainPanel.add(getButton);
    mainFrame.getContentPane().add( mainPanel );
    mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    mainFrame.pack();
    mainFrame.setLocationRelativeTo(null);
    mainFrame.setVisible( true );
  }
  public static void main(String[] args)
  {
    new SwingWorkerTest();
  }
}
```

除了用JProgressBar来显示进度,我们还可以用ProgressMonitor来实现.
ProgressMonitor提供了创建进度条的简便方法,它显示的进度条出现在一个对话框里
它只有一个构造方法:ProgressMonitor(Component parentComponent, Object message, String note, int min, int max)
message和note参数都是和进度条一起显示在对话框里,不同的是,note是可变的,而message不可以.min和max是进度的最小值和最大值
这个对话框并不是在任务一开始就显示出来的,而是等500个百万分之一秒再出来,这个"500"可以用void setMillisToPopup(int millisToPopup)来设定,参数的单位是百万分之一秒;而且,如果它计算得知这个任务用时不超过2000个百万分之一秒,那么这个对话框就永远不会出来.这个"2000",可以用void setMillisToDecideToPopup(int)来设定,参数的单位也是百万分之一秒
它其他的方法有:

int getMillisToPopup()

void setMinimum(int m),void setMaximum(int m),void setNote(String note),void setProgress(int nv)

int getMinimum() , int getMaximum() ,String getNote() ,没有int getProgress()

boolean isCanceled()

这里再介绍SwingWorker的几个方法:

setProgress:设置任务的进度

getProgress:得到任务的进度

可以用addPropertyChangeListener(PropertyChangeLitener)对上面两个方法的调用作出响应

cancel(boolean):取消任务

isCancelled():判断任务是否已被取消

下面是一个例子.

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.Random;
import java.beans.*;

class ProgressMonitorDemo
{
    JFrame mainFrame;
    ProgressMonitor simpleProgressMonitor;
    JButton startButton;
    Worker worker;
    public ProgressMonitorDemo() {
        mainFrame = new JFrame ( "ProgressMonitorDemo" );
        startButton = new JButton ("Start");
        startButton.addActionListener( new ActionListener(){
            public void actionPerformed( ActionEvent e){
                simpleProgressMonitor = new ProgressMonitor(mainFrame,"正在执行任务","",0,100);
                simpleProgressMonitor.setMillisToDecideToPopup(0);
                simpleProgressMonitor.setMillisToPopup(0);
                worker = new Worker();
                worker.addPropertyChangeListener( new PropertyChangeListener(){
                    public void propertyChange( PropertyChangeEvent e ){
                        if( "progress".equals( e.getPropertyName() )){
                            int progress = (Integer)e.getNewValue();
                            simpleProgressMonitor.setProgress( progress );
                            String message = String.format("%d%% completed",progress);
                            simpleProgressMonitor.setNote(message);
                        }
                    }
                });
                worker.execute();
                startButton.setEnabled(false);
            }
        } );
        mainFrame.getContentPane().add( startButton,BorderLayout.LINE_START );
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
```

关闭

```
    }

    public static void main(String[] args)
    {
        new ProgressMonitorDemo();
    }
    class Worker extends javax.swing.SwingWorker<Void,Void>{
        public Void doInBackground(){
            int progress = 0;
            Random r = new Random();
            while( progress<=100 && !isCancelled() ){
                progress += r.nextInt(10);
                setProgress( Math.min(progress,100) );
                try{
                    Thread.sleep( r.nextInt(1000) );
                }catch( InterruptedException e ){
                    e.printStackTrace();
                }
            }
            return null;
        }
        public void done(){
            startButton.setEnabled(true);
        }
    }
}
```

20.JTabbedPane

选项卡.

构造方法:JTabbedPane() ,JTabbedPane(int tabPlacement) ,JTabbedPane(int tabPlacement, int tabLayoutPolicy)

添加选项卡:

void addTab(String title, Component component)

void addTab(String title, Icon icon, Component component)

void addTab(String title, Icon icon, Component component, String tip)

void insertTab(String title, Icon icon, Component component, String tip, int index)

删除选项卡:

void remove(int index)

void removeAll()

void removeTabAt(int index)

修改选项卡上显示的组件:

void setComponentAt(int index, Component component)

Component getComponentAt(int index)

设置外观:

void setTabPlacement(int tabPlacement):JTabbedPane.TOP, JTabbedPane.BOTTOM

,JTabbedPane.LEFT,JTabbedPane.RIGHT

void setTabLayoutPolicy(int tabLayoutPolicy) :JTabbedPane.WRAP_TAB_LAYOUT

,JTabbedPane.SCROLL_TAB_LAYOUT

void setTitleAt(int index, String title)

void setToolTipTextAt(int index, String toolTipText)

void setIconAt(int index, Icon icon)

void setBackgroundAt(int index, Color background)

void setForegroundAt(int index, Color foreground)

查找选项卡:

int indexAtLocation(int x, int y)

int indexOfComponent(Component component)

关闭

int indexOfTab(Icon icon)

int indexOfTab(String title)

和选择有关的:

int getSelectedIndex()

void setSelectedIndex(int index)

Component getSelectedComponent()

void setSelectedComponent(Component c)

自定义标签上的组件:

void setTabComponentAt(int index, Component c);

Component getTabComponentAt(int index);

```
package blog.swing;
import javax.swing.*;
import java.awt.Color;
class JTabbedPaneDemo
{
   JFrame mainFrame;
   JTabbedPane simpleTabbedPane;
   public JTabbedPaneDemo() {
      mainFrame = new JFrame ( "JTabbedPaneDemo" );
      simpleTabbedPane = new JTabbedPane();
      simpleTabbedPane.setTabLayoutPolicy( JTabbedPane.SCROLL_TAB_LAYOUT );
      simpleTabbedPane.addTab("Tab1",new JLabel ("Component1"));
      simpleTabbedPane.addTab("Tab2",new JLabel ("Component2"));
      simpleTabbedPane.addTab("Tab3",new JLabel ("Component3"));
      simpleTabbedPane.addTab("Tab4",new JLabel ("Component4"));
      for(int i=0; i<4; i++){
         simpleTabbedPane.setTabComponentAt( i,new JButton (""+i));
         simpleTabbedPane.setBackgroundAt(i,Color.white);
      }
      mainFrame.getContentPane().add( simpleTabbedPane );
      mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      mainFrame.pack();
      mainFrame.setLocationRelativeTo(null);
      mainFrame.setVisible( true );
   }
   public static void main(String[] args)
   {
      new JTabbedPaneDemo();
   }
}
```

21.JFormattedTextField

在讲JFormattedTextField之前,先讲用于指定格式的类:

Locale,NumberFormat,DecimalFormat,DecimalFormatSymbols,DateFormat,SimpleDateFormat,DateFormatSymbol

根据地区/语言(Locale)的不同,各种数字,日期的格式会有所不同.例如902333这个数字在德国会写作902.333,而在美国写作902,333

创建Locale可以使用它的构造方法,也可以使用它的一些常量.例如下面两个语句是等价的:

Locale locale1 = new Locale("zh","CN");

Locale locale2 = Locale.CHINA;

上面用到的"zh"(小写)和"CN"(大写)分别遵循着一定的规定,在下面的链接可以找到这些搭配:

http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

关闭

你可以用任意的"xx"和"XX"搭配来创建Locale,但是,并不是所有都是有意义的,即Locale不一定可被上面的XXXFormat使用.
使用下面的程序可以得到DateFormat可以使用的组合:

```
package blog.swing;
import java.util.Locale;
import java.text.DateFormat;
class AvailableLocale
{
   public static void main(String[] args)
   {
      Locale[] locales = DateFormat.getAvailableLocales();
      for( Locale locale : locales ){
         System.out.println( locale.toString());
         //System.out.println( locale.getDisplayName() );
      }
   }
}
```

如果你不设定Locale,XXXFormat将使用默认的Locale.这个默认的Locale是和你所用的系统有关的
用Locale.getDefault()可以得到默认的Locale
NumberFormat可以用于数字,货币和百分数的格式化(根据不同的Locale).对于数字,货币和百分数,分别调用静态方法getNumberInstanc(Locale),
getCurrencyInstance(Locale),getPercentInstance(Locale)来取得实例,再用String format(double)来返回格式化后的字符串.
DecimalFormat是NumberFormat的子类,它对格式提供了更多的控制.在构造它的时候可以指定数字显示格式.它不可以直接指定Locale.要指定Locale的时候,可以把一个NumberFormat强制转换为DecimalFormat,再调用applyPattern(String pattern)来指定数字格式.
同样它用String format(double)来返回格式化后的字符串.
可以用DecimalFormatSymbols来指定数字里面的各个符号,例如小数点.在DecimalFormat的构造方法里传入DecimalFormatSymbols就可以了.DecimalFormatSymbols还可以指定Locale,所以用了DecimalFormatSymbols就不用将一个NumberFormat转换为Decimalformat以指定Locale了

```
package blog.swing;
import java.util.Locale;
import java.text.NumberFormat;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
class NumberFormatDemo{
   public static void main( String[] args ){
      double number = 96356.127;
      double currency = 56832.523;
      double percent = 0.72;
      String out;
      Locale locales[] = { Locale.CHINA,Locale.US,Locale.GERMANY };
      NumberFormat formats = NumberFormat.getNumberInstance();
      for( int i=0; i<3; i++){
         formats = NumberFormat.getNumberInstance( locales[i] );
         out = formats.format( number );
         System.out.println( out+"  "+locales[i].getDisplayName() );
         formats = NumberFormat.getCurrencyInstance( locales[i] );
         out = formats.format( currency );
```

关闭

```
            System.out.println( out+"  "+locales[i].getDisplayName() );
            formats = NumberFormat.getPercentInstance( locales[i] );
            out = formats.format( percent );
            System.out.println( out+"  "+locales[i].getDisplayName() );
        }
        DecimalFormat df = new DecimalFormat();
        String pattern = "@#,###.##";
        df.applyPattern( pattern );
        out = df.format(number);
        System.out.println( out );
        pattern = "#@###.####";
        df.applyPattern( pattern );
        out = df.format(number);
        System.out.println( out );
        df = (DecimalFormat)formats;
        df.applyPattern("#,###.##");
        out = df.format(number);
        System.out.println( out );
        DecimalFormatSymbols dfss = new DecimalFormatSymbols(Locale.GERMANY);
        dfss.setDecimalSeparator('^');
        df.setDecimalFormatSymbols( dfss );
        df.applyPattern("00,000.000");
        out = df.format(number);
        System.out.println( out );
    }
}
```

pattern的格式应满足:

pattern := subpattern{;subpattern}

subpattern := {prefix}integer{.fraction}{suffix}

prefix := '//u0000'..'//uFFFD' - specialCharacters

suffix := '//u0000'..'//uFFFD' - specialCharacters

integer := '#'* '0'* '0'

fraction := '0'* '#'*

上面讲的都是和数字有关的,下面讲的是和日期和时间有关的

和日期,时间有关的格式用DateFormat,它的用法和NumberFormat差不多,也是调用静态方法来取得实例,再调用String format(Date)来返回格式化后的字符串

这些静态方法有:

DateFormat getDateInstance(),DateFormat getDateInstance(int style),DateFormat getDateInstance(int style, Locale aLocale)

DateFormat getTimeInstance(),DateFormat getTimeInstance(int style),DateFormat getTimeInstance(int style, Locale aLocale)

DateFormat getDateTimeInstance(),DateFormat getDateTimeInstance(int style),DateFormat getDateTimeInstance(int style, Locale aLocale)

第一个参数指定显示的风格,根据第二个参数的不同,这些风格也有所不同.可以取的值有:

DEFAULT,LONG,MEDIUM,SHORT,FULL,它们都是DateFormat的静态常量.

对应于数字的DecimalFormat,在日期时间方面,有一个SimpleDateFormat

与DecimalFormat不同的是,SimpleDateFormat在构造的时个就可指定Locale了.

与数字的DeciamlFormatSymblos对应,在日期时间方面,有一个DateFormatSymbols.

```
package blog.swing;
import java.util.Locale;
import java.util.Calendar;
```

关闭

```
import java.util.GregorianCalendar;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.text.DateFormatSymbols;
class  DateTimeFormatDemo
{
   public static void main(String[] args)
   {
      Calendar calendar = new GregorianCalendar();
      Date date = calendar.getTime();
      Locale[]  locales = { Locale.CHINA,Locale.US,Locale.GERMANY };
      String[] patterns = { "yy-MM-dd","E yyyy/MM/dd","yy.MM.dd.hh.mm.ss" };
      DateFormat formats;
      SimpleDateFormat sdf;
      String out;
      for( int i=0; i<3; i++ ){
         formats = DateFormat.getDateInstance(DateFormat.DEFAULT,locales[i]);
         out = formats.format( date );
         System.out.println( out );
         formats = DateFormat.getTimeInstance(DateFormat.LONG,locales[i]);
         out = formats.format( date );
         System.out.println( out );
         formats = DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL,locales[i]);
         out = formats.format( date );
         System.out.println( out );
         sdf = new SimpleDateFormat(patterns[i],locales[i]);
         out = sdf.format( date );
         System.out.println( out+"  "+patterns[i] );
         System.out.println( "======================" );
      }
      DateFormatSymbols dfss = new DateFormatSymbols(Locale.CHINA);
      sdf = new SimpleDateFormat();
      String[] capitalDays = {
               "","SUN-星期日", "MON-星期一", "TUE-星期二", "WED-星期三",
                  "THU-星期四", "FRI-星期五", "SAT-星期六"};
      dfss.setShortWeekdays(capitalDays);
      sdf.applyPattern("E");
      sdf.setDateFormatSymbols( dfss );
      out = sdf.format(date);
      System.out.println( out );
   }
}
```

下面是一个JFormattedTextField的例子.在这个例子中的三个JFormattedTextField分别使用了NumberFormat的三个静态方法取得的NumberFormat,当它们失去焦点时,它们显示的文本就会被格式化再重新显示,如果输入的是无效的文本,则文本被重设为上次的有效文本.

```
package blog.swing;
import java.awt.GridLayout;
import java.awt.Container;
import javax.swing.BorderFactory;
import javax.swing.*;
```

关闭

```
import java.text.NumberFormat;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeEvent;

class JFormattedTextFieldDemo implements PropertyChangeListener
{
   JFrame mainFrame;
   JPanel mainPanel;
   JFormattedTextField priceFormattedTextField;
   JFormattedTextField discountFormattedTextField;
   JFormattedTextField paymentFormattedTextField;
   JLabel priceLabel;
   JLabel discountLabel;
   JLabel paymentLabel;
   NumberFormat priceFormat;
   NumberFormat discountFormat;
   NumberFormat paymentFormat;
   public JFormattedTextFieldDemo() {
      mainFrame = new JFrame ( "JFormattedTextFieldDemo" );
      mainPanel = new JPanel ( new GridLayout(3,2) );
      mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

      priceFormat = NumberFormat.getNumberInstance();
      priceFormattedTextField = new JFormattedTextField(priceFormat);
      priceFormattedTextField.setValue(72.023);
      priceFormattedTextField.addPropertyChangeListener("value",this);
      priceLabel = new JLabel ("Price");
      priceLabel.setLabelFor( priceFormattedTextField );
      mainPanel.add( priceLabel );
      mainPanel.add( priceFormattedTextField );

      discountFormat = NumberFormat.getPercentInstance();
      discountFormattedTextField = new JFormattedTextField(discountFormat);
      discountFormattedTextField.setValue(0.75);
      discountFormattedTextField.addPropertyChangeListener("value",this);
      discountLabel = new JLabel ("Discount");
      discountLabel.setLabelFor( discountFormattedTextField );
      mainPanel.add( discountLabel );
      mainPanel.add( discountFormattedTextField );

      paymentFormat = NumberFormat.getCurrencyInstance();
      paymentFormattedTextField = new JFormattedTextField(paymentFormat);
      paymentFormattedTextField.setEditable( false );
      paymentFormattedTextField.addPropertyChangeListener("value",this);
      paymentLabel = new JLabel ("Payment");
      paymentLabel.setLabelFor( paymentFormattedTextField );
      mainPanel.add( paymentLabel );
      mainPanel.add( paymentFormattedTextField );

      mainFrame.getContentPane().add( mainPanel );
      mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      mainFrame.pack();
      mainFrame.setLocationRelativeTo(null);
      mainFrame.setVisible( true );
```

关闭

```
    }
    public void propertyChange( PropertyChangeEvent e ){
        double price = ((Number)priceFormattedTextField.getValue()).doubleValue();
        double discount = ((Number)discountFormattedTextField.getValue()).doubleValue();
        paymentFormattedTextField.setValue( price*discount );
    }
    public static void main(String[] args)
    {
        new JFormattedTextFieldDemo();
    }
}
```

除了使用上面讲到的java.text包中的各种formatter,还可以使用javax.swing.text.MaskFormatter来限制用户可以输入的字符

```
package blog.swing;
import javax.swing.*;
import javax.swing.text.MaskFormatter;
import java.text.ParseException;
class  MaskFormatterDemo
{
    JFrame mainFrame;
    JFormattedTextField simpleFormattedTextField;
    MaskFormatter mask;
    public MaskFormatterDemo(){
        mainFrame = new JFrame ( "MaskFormatterDemo" );
        try{
            mask = new MaskFormatter("####");
            simpleFormattedTextField = new JFormattedTextField( mask );
            mainFrame.getContentPane().add( simpleFormattedTextField );
        }catch( ParseException e ){
            e.printStackTrace();
        }
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    public static void main(String[] args)
    {
        new MaskFormatterDemo();
    }
}
```

22.JSpinner

微调组件.

微调组件由Editor,微调按钮,和它的Model组成.

在构造JSpinner时,可以指定它的Model.

swing提供了三个Model:

SpinnerListModel,SpinnerNumberModel,SpinnerDateModel

结构是SpinnerModel

|_AbstractSpinnerModel

|_SpinnerListModel,SpinnerNumberModel,SpinnerDateModel

对应有三个Editor:

JSpinner.ListEditor,JSpinner.NumberEditor,JSpinner.DateEditor,三个都是JSpinner.DefaultEditor的子类

JSpinner.DefaultEditor

|_JSpinner.ListEditor,JSpinner.NumberEditor,JSpinner.DateEditor

可以看到,有很大的空间可以自定义一个JSpinner

当你需要自定义它的Editor时,你可以用void setEditor(JComponent editor),也可以用

JSpinner.DefaultEditor.getTextField()来取得JFormattedTextField,然后对这个JFormattedTextI

可以通过addChangeListener对JSpinner值的改变作出反应.

```
package blog.swing;
import java.awt.GridLayout;
import java.awt.Color;
import javax.swing.*;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.util.Date;
import java.util.Calendar;
import java.text.DecimalFormat;

class JSpinnerDemo implements ChangeListener
{
   JFrame mainFrame;
   JPanel mainPanel;
   JSpinner listSpinner;
   JSpinner numberSpinner;
   JSpinner dateSpinner;
   public JSpinnerDemo() {
      mainFrame = new JFrame ( "JSpinnerDemo" );
      mainPanel = new JPanel ( new GridLayout(3,1) );

      String[] listData = { "SpinnerListModel","SpinnerNumberModel","SpinnerDateModel" };
      //使用自定义的Model来实现cycle
      SpinnerModel listModel = new CustomSpinnerListModel(listData);
      listSpinner = new JSpinner( listModel );
      listSpinner.addChangeListener( this );
      mainPanel.add(listSpinner);

      SpinnerModel numberModel = new SpinnerNumberModel(1.0,0.0,2.0,0.1);
      numberSpinner = new JSpinner( numberModel );
      numberSpinner.addChangeListener( this );
      //通过取得JFormattedTextField来自定义Editor
      JSpinner.DefaultEditor editor = (JSpinner.DefaultEditor)numberSpinner.getEditor();
      JFormattedTextField ftf = editor.getTextField();
      ftf.setForeground( Color.red );
      mainPanel.add( numberSpinner );

      Calendar calendar = Calendar.getInstance();
      Date initDate = calendar.getTime();
      calendar.add(Calendar.YEAR,-100);
      Date earliestDate = calendar.getTime();
      calendar.add(Calendar.YEAR,200);
      Date latestDate = calendar.getTime();
      SpinnerModel dateModel = new SpinnerDateModel(initDate,earliestDate,latestDate,Calendar.YEAR);
      dateSpinner = new JSpinner( dateModel );
```

关闭

```
        dateSpinner.addChangeListener( this );
        //通过setEditor来自定义Editor
        dateSpinner.setEditor( new JSpinner.DateEditor(dateSpinner,"yyyy-MM-dd") );
        mainPanel.add(dateSpinner);

        mainPanel.setBorder( BorderFactory.createEmptyBorder(10,10,10,10) );
        mainFrame.getContentPane().add( mainPanel );
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    public void stateChanged( ChangeEvent e ){
        JSpinner spinner = (JSpinner)e.getSource();
        System.out.println( spinner.getValue() );
    }
    //自定义的Model,实现Cycle.
    class CustomSpinnerListModel extends SpinnerListModel{
        Object[] values;
        CustomSpinnerListModel( Object[] values ){
            super(values);
            this.values = values;
        }
        public Object getPreviousValue(){
            Object value = super.getPreviousValue();
            return value != null ? value : values[values.length-1];
        }
        public Object getNextValue(){
            Object value = super.getNextValue();
            return value != null ? value : values[0];
        }
    }
    public static void main(String[] args)
    {
        new JSpinnerDemo();
    }
}
```

23.JTree

这个组件太复杂了,只能很简单很简单地介绍一下.

一树由根节点和子节点组成.它们都是由DefaultMutableTreeNode表示

根节点是必须的,子节点可有可无.

传给DefaultMutableTreeNode的构造方法的是一个Object.在构造JTree的时候,会调用这个Object的toString()取得显示在JTree上的节点的文本.

调用void add(MutableTreeNode newChild)来增加一个子节点.

在构造JTree时,将用DefaultMutableTreeNode表示的根传入构造方法JTree(TreeNode root) ,这样就可以构造一棵树.

JTree用TreeSelectionModel来维护它的选择项,默认使用的是它的实现类DefaultTreeSelectionModel.

通过它的void addTreeSelectionListener(TreeSelectionListener x)可以对选择作出反应.

自定义JTree的外观

void setRootVisible(boolean rootVisible) 可以设置是否隐藏根节点

void setShowsRootHandles(boolean newValue) 设置是否显示节点前面的加号

void putClientProperty(Object key,Object value)设置节点之间的连线的样式

要自定义节点的图标,可以使用DefaultTreeCellRenderer,它是JLabel的一个字类

void setClosedIcon(Icon icon)设置非展开时的图标

void setOpenIcon(Icon newIcon) 设置节点展开时的图标
void setLeafIcon(Icon newIcon) 设置叶节点的图标
它也有一般Renderer类有的方法:
Component getTreeCellRendererComponent(JTree tree, Object value, boolean sel, boolean expanded, boolean leaf, int row, boolean hasFocus)
通过这个方法可以定义每个节点所显示的组件
JTree的void setEditable(boolean)可以设置是否可以就地编辑该树
通过JTree的数据Model TreeModel的void addTreeModelListener(TreeModelListener l),我们可
候作出反应.
简单的应用可以使用TreeModel的实现类DefaultTreeModel.它的构造和JTree一样,也是把树的根
去,DefaultTreeModel(TreeNode root).
创建了它以后,就可以用JTree的另外一个构造方法JTree(TreeModel newModel)来构造一棵树.

```
package blog.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
import javax.swing.tree.*;
class JTreeDemo
{
   JFrame mainFrame;
   JScrollPane scrollPane;
   JTree simpleTree;
   JButton addButton;
   JButton removeButton;
   JTextField insertField;
   JPanel panel;
   public JTreeDemo() {
      mainFrame = new JFrame ( "JTreeDemo" );
      DefaultMutableTreeNode swing = new DefaultMutableTreeNode("Swing");
      buildTree(swing);
      simpleTree = new JTree(swing);
      simpleTree.getSelectionModel().addTreeSelectionListener(new TreeSelectionListener(){
         public void valueChanged( TreeSelectionEvent e ){
            System.out.println( "selection changed" );
         }
      });
      simpleTree.setRootVisible(false);
      simpleTree.setShowsRootHandles(false);
      simpleTree.putClientProperty("JTree.lineStyle","Horizontal");
      simpleTree.putClientProperty("JTree.lineStyle","None");
      simpleTree.setCellRenderer( new CustomTreeCellRenderer() );
      simpleTree.setEditable( true );
      simpleTree.getModel().addTreeModelListener(new TreeModelListener(){
            public void treeNodesChanged(TreeModelEvent e) {
               System.out.println("node changed");
            }
            public void treeNodesInserted(TreeModelEvent e) {
               System.out.println( "node inserted" );
            }
            public void treeNodesRemoved(TreeModelEvent e) {
               System.out.println("node removed");
```

关闭

```
            }
            public void treeStructureChanged(TreeModelEvent e) {
                System.out.println( "strutrued changed" );
            }
        });
    scrollPane = new JScrollPane( simpleTree );

    addButton = new JButton ("add");
    addButton.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent e){
            TreePath parentPath = simpleTree.getSelectionPath();
            if( parentPath != null ){
                DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode)parentPath.getLastPathCo
mponent();
                DefaultTreeModel model = (DefaultTreeModel)simpleTree.getModel();
                DefaultMutableTreeNode child = new DefaultMutableTreeNode( insertField.getText() );
                model.insertNodeInto( child , parentNode,0 );
                simpleTree.scrollPathToVisible( new TreePath( child.getPath() ) );
            }
        }
    });
    removeButton = new JButton ("remove");
    removeButton.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent e){
            TreePath path = simpleTree.getSelectionPath();
            if( path != null ){
                DefaultMutableTreeNode removeNode = (DefaultMutableTreeNode)path.getLastPathCompo
nent();
                DefaultTreeModel model = (DefaultTreeModel)simpleTree.getModel();
                model.removeNodeFromParent( removeNode );
            }
        }
    });
    insertField = new JTextField(20);
    panel = new JPanel ( new GridLayout(15,1) );
    panel.add(insertField);
    panel.add( addButton );
    panel.add( removeButton );

    mainFrame.getContentPane().add( panel,BorderLayout.LINE_START );
    mainFrame.getContentPane().add( scrollPane,BorderLayout.LINE_END  );
    mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    mainFrame.pack();
    mainFrame.setLocationRelativeTo(null);
    mainFrame.setVisible( true );
  }
  private void buildTree(DefaultMutableTreeNode root){
    DefaultMutableTreeNode parent;
    DefaultMutableTreeNode child;
    parent = new DefaultMutableTreeNode("Containers");
    child = new DefaultMutableTreeNode("JFrame");
    parent.add(child);
    child = new DefaultMutableTreeNode("JPanel");
    parent.add(child);
```

关闭

```
        child = new DefaultMutableTreeNode("JDialog");
        parent.add(child);
        root.add(parent);
        parent = new DefaultMutableTreeNode("Components");
        child = new DefaultMutableTreeNode("JButton");
        parent.add(child);
        child = new DefaultMutableTreeNode("JLabel");
        parent.add(child);
        child = new DefaultMutableTreeNode("JList");
        parent.add(child);
        root.add(parent);
    }
    private class CustomTreeCellRenderer extends DefaultTreeCellRenderer{
        public CustomTreeCellRenderer(){
            /*setLeafIcon( new ImageIcon("images/leaf.gif") );
            setOpenIcon( new ImageIcon("images/expand.gif") );
            setClosedIcon( new ImageIcon("images/unexpand.gif") );*/
        }
        public Component getTreeCellRendererComponent(
            JTree tree, Object value,
            boolean sel, boolean expanded,
            boolean leaf, int row, boolean hasFocus){
            JButton button = new JButton ( value.toString() );
            if( leaf )
                button.setIcon(new ImageIcon("images/leaf.gif") );
            else{
                if( expanded )
                    button.setIcon(new ImageIcon("images/expand.gif") );
                else
                    button.setIcon(new ImageIcon("images/unexpand.gif") );
            }
            return button;
        }
    }
    public static void main(String[] args)
    {
        new JTreeDemo();
    }
}
```

24.Jtable 表格组件

应网友要求,勉强加上JTable的用法讲述.写得不好,望不见怪

没有比这个组件更复杂的了

它由标题头和单元格组成.而单元格又分为编辑状态和非编辑状态.自定义JTable主要是对单元格作文章.默认的单元格是一个label,和JTree一样,如果你在创建JTable的时候传递的是其他对象而不是String对象,则该对象的toString方法将被调用.它返回的字符串就会显示在单元格的label里.同以往一样,可以通过"渲染器"将其他组件显示在单元格里;而编辑状态下的单元格默认是一个文本框.可以通过指定CellEditor来指定其他的组件作为编辑状态下单元格的组件(简称编辑器).你可以为某一种类型的数据指定一种编辑器,也可以为一整列的数据指定特定的编辑器.编辑器可以用作验证用户输入的合法性.对于渲染器和编辑器的概念,是和JTree里的相似的

这个组件一般放在一个滚动窗格里.你可以把一个表格作为滚动窗格的viewport,然后再把滚动窗格添加到主框架的内容窗格(content pane)里.如果你不这样做,那么你必须分别把表格头和表格添加到框架窗口的content pane里.

这个组件也是用所谓的模型来保存它的数据的.TableModel就是用来保存数据的.而它和JList一样,用ListSelectionModel来保存选择了的项.另外它还有TableColumnModel来保存关于列的数据.下面几幅图总结了各部Model和Renderer的父子关系.下面是例子程序

```
/*本程序演示了如何自定义自己的渲染器,自己的编辑器,和自己的数据模型
其中渲染器使得数据是Color类型时显示一个带颜色的JLabel
当数据是Color类型时编辑器是一个按钮,用以选择颜色*/



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

class JTableTest
{
    JFrame mainFrame;
    JPanel mainPanel;
    JTable table;
    JScrollPane scrollPane;
    public JTableTest() {

        mainFrame = new JFrame ( "JTableTest" );
        mainPanel = new JPanel ( new BorderLayout() );

        String[] columnName = { "Name","Age","Favorite Color","Career","Married" };
        Object[][] data = { { "kate", new Integer(34),new Color(255,0,0),"engineer", new Boolean(true) },
                            { "jim", new Integer(56),Color.white,"manager", new Boolean(false) },
                           { "roy", new Integer(23),Color.black,"driver", new Boolean(false) },
                           { "paul", new Integer(33),Color.blue,"teacher", new Boolean(true) }};

        CustomTableModel customModel = new CustomTableModel( data,columnName );//自定义的数据模型
        table = new JTable( customModel ){//在构造方法里传递数据模型作为参数
            public String getToolTipText( MouseEvent e){//设定工具提示
                int row = rowAtPoint( e.getPoint() );
                int column = columnAtPoint( e.getPoint() );
                int realColumn = convertColumnIndexToModel( column );
                if( realColumn == 3 ){
                    TableModel model = getModel();
                    String name = (String)model.getValueAt( row, 0 );
                    String career = (String)model.getValueAt( row, realColumn );
                    return name + "'s" + "career is " + career;
                }
                return null;
            }

            public TableCellEditor getCellEditor(int row, int column) {
                if ((row == 1) && (column == 2)) {
                    return new CustomCellEditor();//好像没起作用?
                }
                return super.getCellEditor(row, column);
            }
        };
        table.getTableHeader().setToolTipText("click here can't sort the contents");//设置标题头的工具提示.排序功能
```

关闭

```
并未实现
        ListSelectionModel rowSm = table.getSelectionModel();
        rowSm.addListSelectionListener( new ListSelectionListener(){
            public void valueChanged( ListSelectionEvent e ){
                ListSelectionModel lsm = (ListSelectionModel)e.getSource();
                System.out.println( lsm.getMaxSelectionIndex() );
            }
        } );

        TableColumn column = table.getColumnModel().getColumn(3);
        JComboBox combox = new JComboBox();
        combox.addItem("engineer");
        combox.addItem("students");
        combox.addItem("driver");
        combox.addItem("teacher");
        column.setCellEditor( new DefaultCellEditor( combox ) );//为一整列指定编辑器

        table.setDefaultEditor( Color.class , new CustomCellEditor() );//为特定的数据类型指定编辑器
        table.setDefaultRenderer( Color.class , new CustomRenderer() );



        TableModel tableModel = table.getModel();
        tableModel.addTableModelListener( new TableModelListener(){
            public void tableChanged( TableModelEvent e ){
                TableModel model = (TableModel)e.getSource();
                int row = e.getFirstRow();
                int column = e.getColumn();

                System.out.println( model.getValueAt( row, column ) );
                System.out.println( model.getColumnName( column ) );
            }
        } );

        mainPanel.add(table.getTableHeader(),BorderLayout.PAGE_START);
        mainPanel.add(table,BorderLayout.CENTER);
        scrollPane = new JScrollPane (mainPanel);//这样,表头会自动添加到滚动空格里;否则,则要使用下面三行将标
题头也添加到框架窗口
/*        mainFrame.getContentPane().setLayout(new BorderLayout());
        mainFrame.getContentPane().add(table.getTableHeader(), BorderLayout.PAGE_START);
        mainFrame.getContentPane().add(table, BorderLayout.CENTER);*/



        mainFrame.getContentPane().add( scrollPane );
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    private class CustomTableModel extends AbstractTableModel{//自定义的数据类型
        String[] columnName;
        Object[][] data;
        CustomTableModel(Object[][] data,String[] columnName){
            this.columnName = columnName;
            this.data = data;
```

关闭

```
        }
        public Object getValueAt( int rowIndex, int columnIndex ){
            return data[rowIndex][columnIndex];
        }
        public int getColumnCount(){
            return data[0].length;
        }
        public int getRowCount(){
            return data.length;
        }
        public String getColumnName( int columnIndex ){
            return columnName[columnIndex];
        }
        public Class getColumnClass( int columnIndex ){
            return getValueAt(0,columnIndex).getClass();

        }
        public void setValueAt( Object value, int row, int column ){
            data[row][column] = value;
            fireTableCellUpdated(row,column);
        }
        public boolean isCellEditable(int row ,int column){
            return true;
        }
    }
    //自定义的编辑器
    private class CustomCellEditor extends AbstractCellEditor implements TableCellEditor,ActionListener{
        Color currentColor;
        JButton button;
        JColorChooser chooser;
        JDialog dialog;
        CustomCellEditor(){
            button = new JButton ();
            button.addActionListener( this );
            button.setActionCommand("editor");
            chooser = new JColorChooser();
            dialog = JColorChooser.createDialog( button,"pick a color",true,chooser,this,null);
        }
        public void actionPerformed( ActionEvent e ){
            if( e.getActionCommand().equals( "editor" ) ){
                button.setBackground( currentColor );
                chooser.setColor(currentColor);
                dialog.setVisible( true );
                fireEditingStopped();
            }else{

                currentColor = chooser.getColor();
            }
        }
        public Object getCellEditorValue(){
            return currentColor;
        }
        public Component getTableCellEditorComponent( JTable table,Object value,boolean isSelected,int row,int colu
mn){
```

关闭

```
            currentColor = (Color)value;
            return button;
        }
    }
    //自定义的渲染器
    private class CustomRenderer extends JLabel implements TableCellRenderer{
        CustomRenderer(){
            setOpaque(true);
        }
        public Component getTableCellRendererComponent(
            JTable table,Object value,boolean isSelected,
            boolean hasFocus,int row,int column){
                setBackground( (Color)value );
                setToolTipText( "RGB Value: "+((Color)value).getRed()+" "
                    +((Color)value).getGreen()+" "+((Color)value).getBlue() );
                return this;
        }

    }
    public static void main(String[] args)
    {
        SwingUtilities.invokeLater( new Runnable(){
            public void run(){
                new JTableTest();
            }
        });
    }
}
```

24.Jtable 表格组件

没有比这个组件更复杂的了

它由标题头和单元格组成.而单元格又分为编辑状态和非编辑状态.自定义JTable主要是对单元格作文章.默认的单元格是一个label,和JTree一样,如果你在创建JTable的时候传递的是其他对象而不是String对象,则该对象的toString方法将被调用.它返回的字符串就会显示在单元格的label里.同以往一样,可以通过"渲染器"将其他组件显示在单元格里;而编辑状态下的单元格默认是一个文本框.可以通过指定CellEditor来指定其他的组件作为编辑状态下单元格的组件(简称编辑器).你可以为某一种类型的数据指定一种编辑器,也可以为一整列的数据指定特定的编辑器.编辑器可以用作验证用户输入的合法性.对于渲染器和编辑器的概念,是和JTree里的相似的

这个组件一般放在一个滚动窗格里.你可以把一个表格作为滚动窗格的viewport,然后再把滚动窗格添加到主框架的内容窗格(content pane)里.如果你不这样做,那么你必须分别把表格头和表格添加到框架窗口的content pane里.

这个组件也是用所谓的模型来保存它的数据的.TableModel就是用来保存数据的.而它和JList一样,用ListSelectionModel来保存选择了的项.另外它还有TableColumnModel来保存关于列的数据.下面几幅图总结了各部Model和Renderer的父子关系.

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

class JTableTest
{
    JFrame mainFrame;
    JPanel mainPanel;
    JTable table;
```

关闭

```
JScrollPane scrollPane;
public JTableTest() {

    mainFrame = new JFrame ( "JTableTest" );
    mainPanel = new JPanel ( new BorderLayout() );

    String[] columnName = { "Name","Age","Favorite Color","Career","Married" };
    Object[][] data = { { "kate", new Integer(34),new Color(255,0,0),"engineer", new Boolean(true)
                        { "jim", new Integer(56),Color.white,"manager", new Boolean(false) },
                       { "roy", new Integer(23),Color.black,"driver", new Boolean(false) },
                       { "paul", new Integer(33),Color.blue,"teacher", new Boolean(true) }};

    CustomTableModel customModel = new CustomTableModel( data,columnName );//自定义的数据模型
    table = new JTable( customModel ){//在构造方法里传递数据模型作为参数
      public String getToolTipText( MouseEvent e){//设定工具提示
        int row = rowAtPoint( e.getPoint() );
        int column = columnAtPoint( e.getPoint() );
        int realColumn = convertColumnIndexToModel( column );
        if( realColumn == 3 ){
          TableModel model = getModel();
          String name = (String)model.getValueAt( row, 0 );
          String career = (String)model.getValueAt( row, realColumn );
          return name + "'s" + "career is " + career;
        }
        return null;
      }

      public TableCellEditor getCellEditor(int row, int column) {
        if ((row == 1) && (column == 2)) {
          return new CustomCellEditor();//好像没起作用?
        }
        return super.getCellEditor(row, column);
      }
    };
    table.getTableHeader().setToolTipText("click here can't sort the contents");//设置标题头的工具提示.排序功能
并未实现
    ListSelectionModel rowSm = table.getSelectionModel();
    rowSm.addListSelectionListener( new ListSelectionListener(){
      public void valueChanged( ListSelectionEvent e ){
        ListSelectionModel lsm = (ListSelectionModel)e.getSource();
        System.out.println( lsm.getMaxSelectionIndex() );
      }
    } );

    TableColumn column = table.getColumnModel().getColumn(3);
    JComboBox combox = new JComboBox();
    combox.addItem("engineer");
    combox.addItem("students");
    combox.addItem("driver");
    combox.addItem("teacher");
    column.setCellEditor( new DefaultCellEditor( combox ) );//为一整列指定编辑器

    table.setDefaultEditor( Color.class , new CustomCellEditor() );//为特定的数据类型指定编辑器
    table.setDefaultRenderer( Color.class , new CustomRenderer() );
```

关闭

```
        TableModel tableModel = table.getModel();
        tableModel.addTableModelListener( new TableModelListener(){
            public void tableChanged( TableModelEvent e ){
                TableModel model = (TableModel)e.getSource();
                int row = e.getFirstRow();
                int column = e.getColumn();

                System.out.println( model.getValueAt( row, column ) );
                System.out.println( model.getColumnName( column ) );
            }
        } );

        mainPanel.add(table.getTableHeader(),BorderLayout.PAGE_START);
        mainPanel.add(table,BorderLayout.CENTER);
        scrollPane = new JScrollPane (mainPanel);//这样,表头会自动添加到滚动空格里;否则,则要使用下面三行将标题头也添加到框架窗口
/*        mainFrame.getContentPane().setLayout(new BorderLayout());
        mainFrame.getContentPane().add(table.getTableHeader(), BorderLayout.PAGE_START);
        mainFrame.getContentPane().add(table, BorderLayout.CENTER);*/


        mainFrame.getContentPane().add( scrollPane );
        mainFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        mainFrame.pack();
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setVisible( true );
    }
    private class CustomTableModel extends AbstractTableModel{//自定义的数据类型
        String[] columnName;
        Object[][] data;
        CustomTableModel(Object[][] data,String[] columnName){
            this.columnName = columnName;
            this.data = data;
        }
        public Object getValueAt( int rowIndex, int columnIndex ){
            return data[rowIndex][columnIndex];
        }
        public int getColumnCount(){
            return data[0].length;
        }
        public int getRowCount(){
            return data.length;
        }
        public String getColumnName( int columnIndex ){
            return columnName[columnIndex];
        }
        public Class getColumnClass( int columnIndex ){
            return getValueAt(0,columnIndex).getClass();

        }
        public void setValueAt( Object value, int row, int column ){
            data[row][column] = value;
```

关闭

```
            fireTableCellUpdated(row,column);
        }
        public boolean isCellEditable(int row ,int column){
            return true;
        }
    }
    //自定义的编辑器
    private class CustomCellEditor extends AbstractCellEditor implements TableCellEditor,ActionLis
        Color currentColor;
        JButton button;
        JColorChooser chooser;
        JDialog dialog;
        CustomCellEditor(){
            button = new JButton ();
            button.addActionListener( this );
            button.setActionCommand("editor");
            chooser = new JColorChooser();
            dialog = JColorChooser.createDialog( button,"pick a color",true,chooser,this,null);
        }
        public void actionPerformed( ActionEvent e ){
            if( e.getActionCommand().equals( "editor" ) ){
                button.setBackground( currentColor );
                chooser.setColor(currentColor);
                dialog.setVisible( true );
                fireEditingStopped();
            }else{

                currentColor = chooser.getColor();
            }
        }
        public Object getCellEditorValue(){
            return currentColor;
        }
        public Component getTableCellEditorComponent( JTable table,Object value,boolean isSelected,int row,int colu
mn){
            currentColor = (Color)value;
            return button;
        }
    }
    //自定义的渲染器
    private class CustomRenderer extends JLabel implements TableCellRenderer{
        CustomRenderer(){
            setOpaque(true);
        }
        public Component getTableCellRendererComponent(
            JTable table,Object value,boolean isSelected,
            boolean hasFocus,int row,int column){
                setBackground( (Color)value );
                setToolTipText( "RGB Value: "+((Color)value).getRed()+" "
                    +((Color)value).getGreen()+" "+((Color)value).getBlue() );
                return this;
        }

    }
```

关闭

```
    public static void main(String[] args)
    {
        SwingUtilities.invokeLater( new Runnable(){
            public void run(){
                new JTableTest();
            }
        });
    }
}
```


上一篇 java线程简单介绍
下一篇 系统托盘功能

查看评论

3楼 dzl84394 2009-12-14 10:35发表

复选框JCheckBoxDemo 我执行有一点问题，
1、两个一样的对象一样的（simpleCheckBox1、cb 我debug看了是一样的）但用"=="号一直不相等，
addItemListener里面用this一直报错

2楼 kissfanqie 2007-05-31 10:32发表

等着看下面的哟!

Re: xxkkff 2007-06-01 00:04发表

好好~再等几天吧

Re: xxkkff 2007-06-02 16:18发表

本来还有几张图片的.但是上传不了.这个blog太难操作了

1楼 zengxiaohui 2007-04-30 20:03发表

非常感谢你啊，能加你QQ吗？我的是369749456

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

关闭