

Re: cobbling a ComboBox

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.gui/2008-06/msg00137.html>

- *From:* Daniele Futturovic <da.futt.news@xxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 26 Jun 2008 04:05:55 +0200
-

On 2008-06-25 13:07 +0100, Albretch Mueller allegedly wrote:

Hi Daniele et al,

I have been trying a number of things and I have started to think it may not be possible in the way we are trying it. Maybe we will have to go down to the abstrack buttons of the internal JList the combo box has.
~ As you can see I got, in a sense, the opposite of what I need; which is for the already selected item on the top on the combobox to have a dark color background and a white foreground (as it happens in most window bars). I am getting what I want but in the internal popup window the combo box has:

Do you use an IDE? If you do (and you probably should), I strongly suggest you have a look at the library source code (of Swing, in this case). With Netbeans, for instance, I right-click on any method or field name and select "Go To" -> "Source" and it opens the relevant class. Why am I telling you this? Because for your current problem, I simply opened the source code for `javax.swing.plaf.basic.BasicComboBoxUI` and `javax.swing.plaf.metal.MetalComboBoxUI`, looked how they did things, and thence knew which parts I could and should interfere with. I suggest you do the same. The library source code is, in my humble opinion, the most valuable well of insight.

Here, I simply copy/pasted the method `BasicComboBoxUI#paintCurrentValue`, and made a few modifications (no other way to do what you want to achieve, I'm afraid).

```
<sscce>
package scratch;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.plaf.*;
import javax.swing.plaf.basic.*;
import javax.swing.plaf.metal.*;

public class Test {
```

Re: cobbling a ComboBox

```
public static void main(String[] ss) throws Exception {

    UIManager.put("ComboBoxUI", "scratch.Test$CustomMetalComboBoxUI");

    UIManager.put("ComboBox.editorBackground", new ColorUIResource(0xffff0c0c));
    UIManager.put("ComboBox.editorForeground", new ColorUIResource(0));

    EventQueue.invokeLater( new Runnable(){
    public void run(){
    JComboBox jcb = new JComboBox(new Object[]{"One", "Two", "Three", "Four", "Five", "Six"});

    System.out.println(jcb.getUI().getClass().toString());

    jcb.setBorder( BorderFactory.createEmptyBorder(40, 60, 40, 60) );

    JFrame f = new JFrame();
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    f.getContentPane().add(jcb, BorderLayout.CENTER);

    f.pack();
    f.setLocationRelativeTo(null);

    f.setVisible(true);
    }
    });
}

public static class CustomMetalComboBoxUI
extends MetalComboBoxUI
{
    public static ComponentUI createUI(JComponent c) {
    return new CustomMetalComboBoxUI();
    }

    @Override
    public void paintCurrentValue(Graphics g,Rectangle bounds,boolean hasFocus) {
    ListCellRenderer renderer = comboBox.getRenderer();
    Component c;

    if ( hasFocus && !isPopupVisible(comboBox) ) {
    c = renderer.getListCellRendererComponent( listBox,

    comboBox.getSelectedItem(),
    -1,
    true,
    false );
    }
    else {
```

Re: cobbling a ComboBox

```
c = renderer.getListCellRendererComponent( listBox,
comboBox.getSelectedItem(),
-1,
false,
false );
c.setBackground(UIManager.getColor("ComboBox.background"));
}
c.setFont(comboBox.getFont());
if ( hasFocus && !isPopupVisible(comboBox) ) {
// c.setForeground(listBox.getSelectionForeground());
// c.setBackground(listBox.getSelectionBackground());
c.setForeground( UIManager.getColor("ComboBox.editorForeground") );
c.setBackground( UIManager.getColor("ComboBox.editorBackground") );
}
else {
if ( comboBox.isEnabled() ) {
c.setForeground(comboBox.getForeground());
c.setBackground(comboBox.getBackground());
}
else {
c.setForeground(sun.swing.DefaultLookup.getColor(
comboBox, this, "ComboBox.disabledForeground", null));
c.setBackground(sun.swing.DefaultLookup.getColor(
comboBox, this, "ComboBox.disabledBackground", null));
}
}

// Fix for 4238829: should lay out the JPanel.
boolean shouldValidate = false;
if (c instanceof JPanel) {
shouldValidate = true;
}

currentValuePane.paintComponent(g,c,comboBox,bounds.x,bounds.y,
bounds.width,bounds.height, shouldValidate);
}
}
}
</sscce>
```

(Note: you'll get two warnings when trying to compile this, because of the two calls to sun.swing.DefaultLookup. I wouldn't bother about those for the time being, but keep them in mind, as they may create problems in future releases)

Mind the package directory structure.

You must also name the class as I did or make the according modifications to the code.

Re: cobbling a ComboBox

~ Also could you clarify to me why doesn't

```
package scratch;
```

```
... .
```

```
import scratch.*;  
import scratch.Test06.*;
```

```
... .
```

```
UIManager.put("ComboBoxUI", "CustomMetalComboBoxUI");
```

```
... .
```

```
work?
```

Because, as I told you, the UIManager loads the UI delegate **reflectively**. In other words, it makes a call to `Class.forName()` (essentially) with that String. So the String you put as the value must be on the classpath. If the class "CustomMetalComboBoxUI" were not an inner class, as I wrote it, but a class in the same package as my "Test" class, the String ought to be "scratch.CustomMetalComboBoxUI". If it were in another package, say package x.y.z, the String ought to be "x.y.z.CustomMetalComboBoxUI". And so on. Google for "Java swing custom UI delegate".

--

DF.

to reply privately, change the top-level domain
in the FROM address from "invalid" to "net"

.