

## Re: efficiency of JList setElementAt()

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.gui/2005-12/msg00312.html>

---

- *From:* Thomas Hawtin <[usenet@xxxxxxxxxxxxxxxxxxxxx](mailto:usenet@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 22 Dec 2005 01:54:59 +0000
- 

Raymond Cruz wrote:

Thanks Tom. I agree with your results when I run your program so that made me immediately consider something that I hadn't considered important before. My list objects are HTML strings solely for the purpose of highlighting a few words in a different color. This appears to make the list update about an order of magnitude more costly! If you modify your program to produce strings about 3 times as long, make the strings HTML with a font color tag, and increase the list size to about 130, I think you'll get the kind of results I cited in my first post. (Note: by virtue of the fact that your program did not compile with my 1.4.2 JDK, I suspect you are using an earlier Java version and I know that HTML support has been introduced over time so I'm not sure if you'll be able to use HTML strings).

So I have a new question -- is there a way to produce different font colors that is much more efficient than using HTML strings in Java?

It occurred to me that you can keep your HTML and make it go at a reasonable speed (unless you run low on memory). The Swing cell renderer design is based on assumptions that construction is expensive and updating values is cheap. Clearly, even for the humble JLabel, the second is not correct when the value is HTML. So let's cache.

The approach I have taken is to have a renderer lazily create a conventional renderer for each index. The renderers are softly referenced, in a naive manner. There are lots of variations with different trade-offs. This seems to work quite well for me. It is worth noting that the classical approach is probably going to be faster to show the initial screen. And my class name is too cute for production use.

The only necessary addition to original program is:

```
list.setCellRenderer(new DaisyListCellRenderer());
```

## Re: efficiency of JList setElementAt()

I think I shall add an improved version to my weblog.

Tom Hawtin

(Usual disclaimer.)

```
import java.lang.ref.Reference;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import javax.swing.*.*;

class DaisyListCellRenderer implements ListCellRenderer {
    private final Map<
        JList, List<Reference<ListCellRenderer>>
    > caches =
        new java.util.WeakHashMap<
            JList, List<Reference<ListCellRenderer>>
        >();
    public java.awt.Component getListCellRendererComponent(
        javax.swing.JList list,
        Object value,
        int index,
        boolean isSelected,
        boolean cellHasFocus
    ) {
        // Find cache for this list.
        List<Reference<ListCellRenderer>> cache = caches.get(list);
        if (cache == null) {
            cache =
                new java.util.ArrayList<Reference<ListCellRenderer>>();
            caches.put(list, cache);
        }

        // Cache list may need extending.
        int shortfall = index+1 - cache.size();
        if (shortfall > 0) {
            assert cache.size()+shortfall == index+1;
            cache.addAll(
                Collections.<Reference<ListCellRenderer>>nCopies(
                    shortfall, null
                )
            );
        }
    }
}
```

## Re: efficiency of JList setElementAt()

```
// Find renderer.
Reference<ListCellRenderer> rendererRef = cache.get(index);
ListCellRenderer renderer =
    rendererRef==null ? null : rendererRef.get();
if (renderer == null) {
    renderer = createRenderer(/*...*/);
    cache.set(
        index,
        new java.lang.ref.SoftReference<ListCellRenderer>(
            renderer
        )
    );
}

// Forward.
return renderer.getListCellRendererComponent(
    list, value, index, isSelected, cellHasFocus
);
}
private ListCellRenderer createRenderer(/*...*/) {
    return new javax.swing.DefaultListCellRenderer();
}
}
class LighstSpeed {
    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                swing();
            }
        });
    }
    private static final java.util.Random random =
        new java.util.Random();
    private static void swing() {
        assert java.awt.EventQueue.isDispatchThread();
        JFrame frame = new JFrame("LighstSpeed");
        final DefaultListModel model = new DefaultListModel();
        for (int ct=0; ct<50; ++ct) {
            model.addElement("<html>some data "+new java.util.Date());
        }
        final JList list = new JList(model);
        list.setCellRenderer(new DaisyListCellRenderer());
        frame.add(list); //new JScrollPane(list); //
        frame.pack();
        frame.setVisible(true);
        new javax.swing.Timer(1000, new ActionListener() {
            private boolean set;
            public void actionPerformed(ActionEvent event) {
                model.setElementAt(
                    "<html>new data "+new java.util.Date(),
                    random.nextInt(model.size())
                );
            }
        }).start();
    }
}
}
--
```

Unemployed English Java programmer  
<http://jroller.com/page/tackline/>

.