

Nodes & Explorer Views

Geertjan Wielenga
Sun Microsystems

Agenda

- Problem Statement & Solution
- Nodes
- Explorer Views
- Q/A

Problem Statement

```
public class JListDemo extends JFrame {
    public static void main(String args[]) {
        final JList jList1 = new JList();
        jList1.setModel(new AbstractListModel() {
            String[] strings = {"Tom", "Dick", "Harry "};
            @Override
            public int getSize() {
                return strings.length;
            }
            @Override
            public Object getElementAt(int i) {
                return strings[i];
            }
        });
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                JListDemo list = new JListDemo();
                list.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
                list.setTitle("JList Demo");
                list.setSize(new Dimension(200, 200));
                list.add(jList1);
                list.setVisible(true);
            }
        });
    }
}
```

Problem Statement

```
public class JTreeDemo extends JFrame {

    public static void main(String args[]) {

        final JTree jTree1 = new JTree();
        DefaultMutableTreeNode treeNode1 = new DefaultMutableTreeNode("Names");
        DefaultMutableTreeNode treeNode2 = new DefaultMutableTreeNode("Tom");
        treeNode1.add(treeNode2);
        treeNode2 = new DefaultMutableTreeNode("Dick");
        treeNode1.add(treeNode2);
        treeNode2 = new DefaultMutableTreeNode("Harry");
        treeNode1.add(treeNode2);
        jTree1.setModel(new DefaultTreeModel(treeNode1));

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                JTreeDemo list = new JTreeDemo();
                list.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
                list.setTitle("JTree Demo");
                list.setSize(new Dimension(200, 200));
                list.add(jTree1);
                list.setVisible(true);
            }
        });
    }
}
```

Problem Statement

- When presenting data models to user...
 - > Need to program many low level details (e.g., cell renderers)
 - > Switching from one control (e.g., Jlist) to another control (JTree) is hard
 - > Context sensitivity
- A lot of plumbing code is needed!

Solution

- Nodes API:
 - > Easy to create a tree-like model
- Explorer API:
 - > Many UI components that can render the model

Solution

```
class NamesNode extends Children.Keys<String> {  
  
    @Override  
    protected void addNotify() {  
        String[] strings = {"Tom", "Dick", "Harry "};  
        setKeys(strings);  
    }  
  
    @Override  
    protected Node[] createNodes(String key) {  
        return new Node[]{new PropNode(key)};  
    }  
  
    private class PropNode extends AbstractNode {  
  
        private PropNode(String key) {  
            super(Children.LEAF, Lookups.fixed());  
            setDisplayName(key);  
        }  
  
    }  
  
}
```

What is a Node?

- Nodes are a *presentation layer*
- Nodes are *hierarchical*
 - > They have child nodes that can have their own child nodes.

DEMO

How to Create a Node

Presentation Layer

- Nodes take a random object and provide human-friendly features
 - > Actions
 - > Display name
 - > Description
 - > Icon
 - > Properties (can be shown/edited in property sheet)
 - > Clipboard operations

Nodes API

```
import org.openide.nodes.AbstractNode;  
import org.openide.nodes.Children;  
class MyNode extends AbstractNode {  
    public MyNode() {  
        super(new MyChildren());  
    }  
}  
class MyChildren extends Children.Keys<String> {  
    protected void addNotify() {  
        String[] names = {"Tom", "Dick", "Harry "};  
        setKeys(names);  
    }  
    protected Node[] createNodes(String key) {  
        MyNode n = new MyNode();  
        n.setName(key);  
        return new Node[] { n };  
    }  
}
```

What is an Explorer View?

- An “explorer” component is a Swing component
- It can show a Node and its children
- Many different components
 - > Trees, Lists, Combo Boxes, Tree Tables, Property Sheet
 - > all in `org.openide.explorer.view`
- Nodes provide a universal tree-model for presenting data
- Explorer views are components to show that data to the user

DEMO

How to Add Explorer Views to the Palette

Views

- ExplorerManager
 - > root context
 - > explored context
 - > selected nodes (vetoable)
- General Model behind Swing
 - > BeanTreeView, ContextTreeView
 - > ListView
 - > PropertySheet
 - > TableTreeView

Using Explorer Views

```
class MyPanel extends JPanel implements
    ExplorerManager.Provider {

    public MyPanel() {
        myManager = new ExplorerManager();
        add(new BeanTreeView());
        add(new PropertySheetView());
        myManager.setRootContext(myNode);
    }

    public ExplorerManager getExplorerManager() {
        return myManager;
    }
}
```

DEMO

How to Use Explorer Views

DEMO

How to Port to the NetBeans Platform

Nodes and Selection

- Each window component has a Lookup
- Nodes have Lookups
- You can easily have the window component's Lookup proxy whatever the Lookup(s) of the selected Node(s) in an explorer view

Conclusion

- Nodes are typed JavaBeans
- Hierarchy
- Extensible
- Rich Set of Views
- Standalone

Q&A

- <http://bits.netbeans.org/dev/javadoc/org-openide-nodes/>
- <http://bits.netbeans.org/dev/javadoc/org-openide-explorer/>