Adding a Legend for a Dynamic Layer to a TOC in a MapObjects Java Application

By Rex Hansen, ESRI Developer Support Group

This article will discuss different methods for creating and adding a legend for a dynamic layer to a TOC component using MapObjects—Java Edition (MOJ). These methods differ depending on the amount of control the developer needs over the look and feel of the legend.

When using MOJ, graphic features can be created and displayed dynamically in a Map component (e.g., acetate layer). To manage the display of a dynamic layer, a developer may want to create a Legend to represent the layer in a TOC. Techniques for creating a dynamic layer legend vary in difficulty and functionality, depending on the method used to create the layer. Three primary techniques are available.

- Using a FeatureLayer and letting MOJ manage the Legend
- Adding a GraphicsLayer as a LayerEvent
- Reconstructing the TOC

Using a FeatureLayer and Letting MOJ Manage the Legend

When creating a dynamic layer as a FeatureLayer (e.g., BaseFeatureLayer) and adding it to a Map component, a new Legend will automatically be added to the Map TOC. A dynamic FeatureLayer will have the same properties as a FeatureLayer using data located on disk (i.e., shapefile, ArcSDE layer) so that a legend can be managed in the same fashion. This technique is the easiest method for creating a legend for a dynamic layer.

Adding a GraphicsLayer Using a LayerEvent

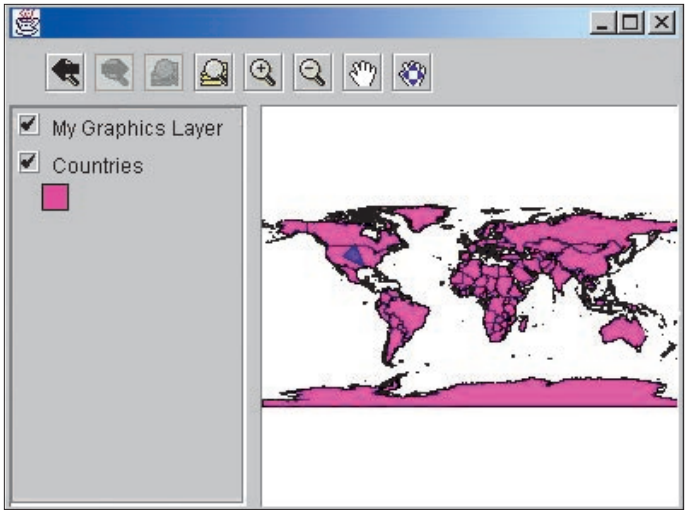
When creating a dynamic layer as a GraphicsLayer, it is usually associated with an AcetateLayer object before being added to a Map. By default, no legend is created for a GraphicsLayer. To add a legend for a GraphicsLayer, a new LayerEvent can be created and processed by a Map component. A LayerEvent informs the Map component LayerListener if a layer has been added, moved, or deleted from a Map. A legend for a GraphicsLayer can be added to a TOC component using the code shown in Listing 1.

The legend will not contain a symbol swatch, and a Map must have a registered TOC before the LayerEvent can be processed. In addition, a TocListener must be added to the TOC to listen for events on the dynamic layer legend (e.g., set layer visibility depending on the state of the check box). This technique can be used if a legend does not need a symbol swatch or the dynamic layer is a GraphicsLayer and the developer does not want to create a TOC from scratch (see the next section, Reconstructing a TOC).

Reconstructing a TOC

The TOC component must be reconstructed to add a legend entry if a dynamic layer created from an AcetateLayer is used or if the legend for a GraphicsLayer needs to display a symbol swatch. The TOC is essentially a JScrollPane containing a parent JPanel consisting of one or more child JPanels (Legends). The TOC and all of its components (e.g., legends, listeners) can be created from scratch, or the existing TOC component can be included within the new parent JPanel. To include the existing TOC components, retrieve the current TOC component, create a new parent Jpanel, and create a new JPanel to represent each layer including the dynamic layer. This process is shown in Listings 2 through 4.

Creating a new parent Jpanel is shown in Listing 3. This JPanel will contain multiple child JPanels as dynamic layer legends as well as existing TOC Legends. Ultimately, the content of the existing TOC component will be set to the new parent JPanel.



An example of a MapObjects Java application interface after a GraphicsLayer has been added using a LayerEvent.

Listing 4 demonstrates how to create a new JPanel to represent each layer including the dynamic layer. The developer can choose the type of check box, title, and symbol used. If an AcetateLayer is used, its visibility is set by adding and removing it from the Map component. The visibility of a GraphicsLayer can be set using the setVisible() method. Add this JPanel as a child to the JPanel created in the step shown in Listing 3. Also, be sure to add the appropriate listeners as needed (e.g., add an ActionListener to determine if the check box is checked). The code in Listing 4 illustrates how to create a JPanel for an AcetateLayer. This technique can be used if an AcetateLayer needs a legend entry or the content of a TOC needs to be completely customized.

Conclusion

The easiest method for creating a dynamic layer legend is using a FeatureLayer and allowing the TOC component to handle the legend properties. If a dynamic layer based on a GraphicsLayer is used, a LayerEvent can be created to add a legend with a check box and label to an existing TOC. If a dynamic layer based on an AcetateLayer is used, reconstructing the TOC and legend components from scratch is the only option. Ultimately, a MOJ developer should consider how the legend of a dynamic layer is to be represented in the TOC before deciding on the type of dynamic layer to be created.

LISTING 1

```
{CODE}
AcetateLayer acetateLayer = new AcetateLayer(new graphicsLayer());
acetateLayer.setName("My Graphics Layer");
aMap.add(acetateLayer);
LayerEvent event = new LayerEvent(acetateLayer.getLayer(), LayerEvent.ADD);
aMap.getLayerset().processLayerEvent(event);
{/CODE}
```

LISTING 2: Retrieve the current Toc component

```
{CODE}
Component component = aToc.getViewport().getView();
{/CODE}
```

LISTING 3

```
{CODE}
JPanel jpp = new JPanel();
jpp.setLayout(new BorderLayout());
jpp.add(component,BorderLayout.NORTH);
{/CODE}
```

LISTING 4

```
{CODE}

  public class dynamic_layer extends JPanel {

    customMouseAdapter myMouseAdapter = new customMouseAdapter();
    com.esri.mo.ui.bean.AcetateLayer m_acetate;
    com.esri.mo.ui.bean.Map m_map;

    public dynamic_layer(com.esri.mo.ui.bean.AcetateLayer layer, com.esri.mo.ui.bean.Map map) {

      m_acetate = layer;
      m_map = map;

      /* Set the layout manager for the JPanel.
      FlowLayout fl = new FlowLayout();
      this.setLayout(fl);
      fl.setAlignment(FlowLayout.LEFT);
      fl.setHgap(3);
      fl.setVgap(1);

      //Add a check box with an ActionListener to set the visibility of the AcetateLayer
      final JCheckBox jcb = new JCheckBox();
      jcb.setSelected(true);

      jcb.addActionListener(new ActionListener(){

        public void actionPerformed(ActionEvent ae){

          if (jcb.isSelected()) {
            m_map.add(m_acetate);
            m_map.redraw();
          } else {
                m_map.remove((Component) m_acetate);
         // After removing the AcetateLayer, the Map needs to be repainted
         m_map.repaint();
            m_map.redraw();
          }

        }

      });

      // Construct the legend label
      JTextField jtf = new JTextField("My Dynamic Layer");
      jtf.setEditable(false);
```

Continued on page 28

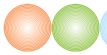
Adding a Legend for a Dynamic Layer to a TOC in a MapObjects Java Application

Continued from page 27

```
        // Add the components, set the color and size of the legend
        this.add(jcb);
        this.add(jtf);
        setBackground(Color.lightGray);
        setPreferredSize(new Dimension(140, 50));

        /* Add a MouseListener to listen for mouse interaction with the legend JPanel.
         * For example, selecting a legend to make it active.
         */
        addMouseListener(myMouseAdapter);



        }

        public void paintComponent(Graphics g){
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g;

            // Add the symbol swatch to the legend component
            java.awt.geom.Area e = new java.awt.geom.Area(new Rectangle2D.Double(17,25,15,15));
            g2d.setColor(new Color(0,20,250));
            g2d.fill(e);
            g2d.setColor(new Color(0, 0, 0));
            g2d.draw(e);

         }
    }
```

4) Once the new JPanel for the dynamic layer is created, add it to the "parent" JPanel, then associate the "parent" JPanel with the existing Toc.

```
{CODE}
dynamic_layer dyn_layer = new dynamic_layer(myAcetateLayer, map1);
jpp.add(dyn_layer, BorderLayout.CENTER);
aToc.setViewportView(jpp);
{/CODE}
```



Spatial Editing Shortcuts

Use the keyboard shortcuts for various editing tools and commands to make edits more quickly and efficiently.

Key	Function	Tool
T	Radius displays snapping tolerance	Sketch tool
V	Displays all vertices of features	All tools in Editor toolbar
O	Offset	Trace
Z	Zoom in	Rotate tool, Edit tool
X	Zoom out	Sketch tool
C	Pan	Split tool
R	Radius	End Point Arc, Distance-to-Distance
D	Diameter	Direction-Distance
Ctrl	Move selection anchor	End Point Arc, Distance-to-Distance, Direction-Distance
Shift + S	Snap anchor for rotate	Scale tool, Rotate tool