P1.23 SGT BEANS: AN EASY-TO-USE GRAPHICAL DEVELOPMENT TOOL

Donald W. Denbo*
Joint Institute for the Study of Ocean and Atmosphere, University of Washington, Seattle, WA

1. INTRODUCTION

SGT Beans enables beginning and advanced developers to create custom graphical displays for their data in a point-and-click environment. A tutorial has been developed that demonstrates using SGT Beans and the JavaNetCDF library from Unidata to read time series and gridded data to create a sophisticated graphical display of the data. The full source code for the tutorial and all necessary support files are available at the SGT website at http://www.epic.noaa.gov/java/sgt.

2. SCIENTIFIC GRAPHICS TOOLKIT

The Scientific Graphics Toolkit (SGT) (Denbo, 2003; Denbo, 2001; Denbo, 2000; Denbo, 1999; Denbo, 1997) is a library of java graphics classes which facilitates the development of platform independent Java applications and web applets to produce highly interactive, flexible, publication quality, object-oriented graphics of scientific data. Plot types supported include contour, line, point, and vector, plot features include user settable or automatically scaled axes, sophisticated, automatically self-scaling time axes, mouse-movable objects (labels, line keys, icons), customizable objects, automatic generation of legends to explain the data being displayed. Although using the core SGT requires detailed expertise in object oriented java programming, it has proven popular with specialists in developing graphics applications. Since March 2000, there have been over 17,500 downloads of SGT source, class, and Javadoc jar files (jar files are Java archives) by 6,016 unique sites in 70 countries.

Version 3.0 of SGT has been made significantly more accessible to a far wider community of non-expert users by extending core SGT with easy-to-use SGT Beans. SGT Beans are easier for an expert programmer to utilize, and moreover, it gives beginner and intermediate Java developers access to these powerful scientific graphics in their Java applets, applications, and servlets, without the steep learning-curve required to use core SGT directly. SGT Beans includes a wizard that enables a developer to easily create fully populated instances of the PanelModel class and create a text file representing the objects state that can be used to initialize a PanelModel at runtime. (NOTE: Java classes are indicated by a serif font.) See http://www.epic.noaa.gov/java/sgt for more information and download links.

* *Corresponding author address:* Donald W. Denbo, NOAA/PMEL/OCRD, 7600 Sand Point Way NE, Seattle, WA 98115; e-mail: Donald.W.Denbo@noaa.gov

3. DESIGN GOALS

SGT Beans is designed to provide a simplified interface to SGT for non-expert users. JavaBeans, also know simply as Beans, were used to create reusable, platform-independent components. Using JavaBeans-compliant application-builder tools, you can combine these components into applets, applications, or composite components.

SGT Beans, unlike core SGT, handles more of the machinery required for an interactive graphics application. For example, SGT Beans handles user initiated zoom and zoom-reset events on a per DataGroup basis. Each DataGroup can be set to ignore or respond to zoom events and scale independently of other DataGroups. SGT Beans have also simplified and expanded how the user controls the placement and scaling of a graphic on a printer page.

In creating SGT Beans, much complexity has been placed into the PanelModel Bean. A “wizard” was developed to provide users a graphical tool for creating a PanelModel.

4. ARCHITECTURE

SGT Beans consists of three primary JavaBeans (Figure 1), the Page, DataModel, and PanelModel and many support classes. The DataModel and PanelModel beans provide the “state” information necessary for Page to create a graphic display.

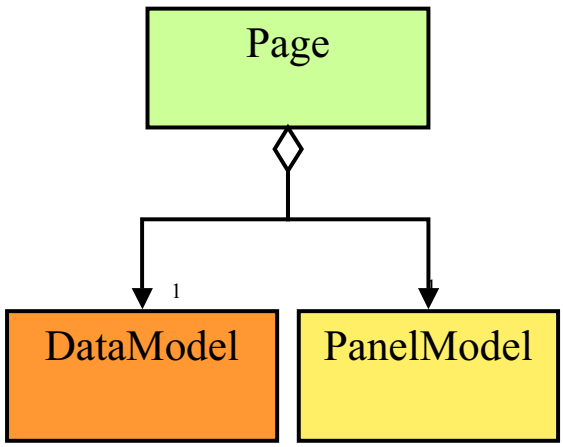

Figure 1. UML diagram showing the relationship between the three SGT Beans, Page, DataModel, and PanelModel.

4.1 *Page*

Page is a visual JavaBean that extends javax.swing.JComponent and can be added to a swing container, for example javax.swing.JPanel. Page can have any number of Panels placed inside it. Each Panel can contain DataGroups, which provides an independent graphical display of data. The location and size of the Panels and DataGroups are specified in the PanelModel (Figure 2).

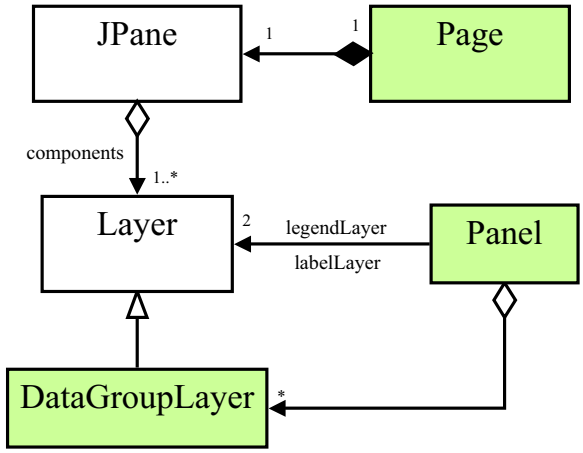

Figure 2. Page JavaBeans relationship to JPane and other components.

4.2 *PanelModel*

The PanelModel (Figure 3) is a non-visual JavaBean. It doesn't have a graphical presence, but development environments can interact with and invoke the "wizard" of non-visual Beans. The PanelModel is responsible for maintaining:

- The location and size of each Panel on the Page.
- AxisGroup state and position on a Panel.
- Legend state and position on a Panel.
- Label state and position on a Panel.

The PanelModel and the classes that it aggregates have been designed to be serialized to both a binary representation, using java.io.ObjectOutputStream, and textual representation, using java.beans.XMLEncoder. PanelModel has a static method *loadFromXML*(InputStream is) to simplify creation of a PanelModel from a textual serialized instance of PanelModel. XML serialization was chosen for the wizard because it is resistant to the versioning problems that binary serialization are often prone.

4.3 *DataModel*

The DataModel is also a non-visual JavaBean. The DataModel references information from the PanelModel to specify a connection between data/attribute and Panel/

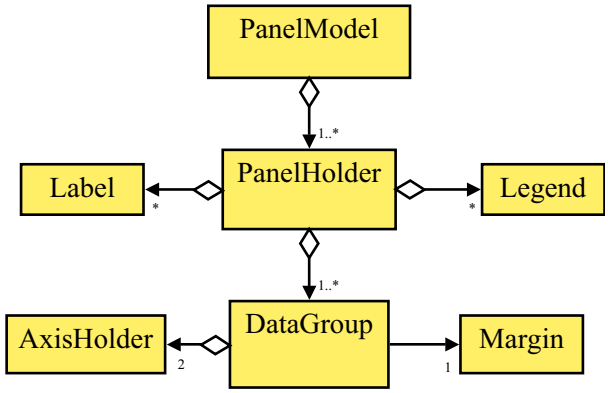

Figure 3. PanelModel and its important SGT Beans components.

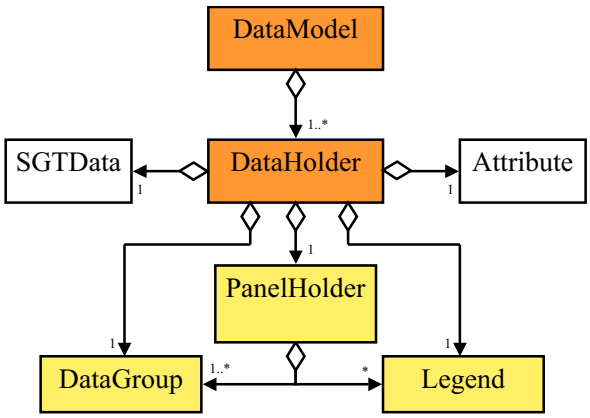

Figure 4. DataModel and its relationship to SGT and SGT Beans components (SGT Beans in orange and yellow).

4.4 *PanelModelEditor (wizard)*

The PanelModelEditor is designed to provide a very user friendly, graphical, and interactive way for a developer or user to create, modify, or save a PanelModel. The PanelModelEditor is used to position Panels, Labels, and Legends on a Page and to set their properties.

The PanelModelEditor can be run in several ways. It can be started as a "wizard" from within a development environment (e.g. JBuilder, NetBeans, etc.), started from a application to edit a PanelModel while the application is running, or run as a stand-alone application by executing the PanelModelEditor directly from the sgt.jar file (On many operating systems double clicking the sgt.jar file will launch the PanelModelEditor). The editor can open existing PanelModels serialized with java.beans.XMLEncoder and save the results of an editing session in the same format.

5. EXAMPLE

The following is an example of a multiple Panel SGT Beans graphics application. The top Panel contains a grid plot of temperature overlaid with velocity vectors. The bottom Panel contains a time series of vertical velocity. (More information on this and other SGT Beans examples can be found at http://www.epic.noaa.gov/talks/dwd/noaatech2004/SGTBeanTutorial.htm.)

The first step in creating a SGT Beans application is to create a serialized instantiation of PanelModel. The PanelModelEditor (Figure 5) is used to create, position, and

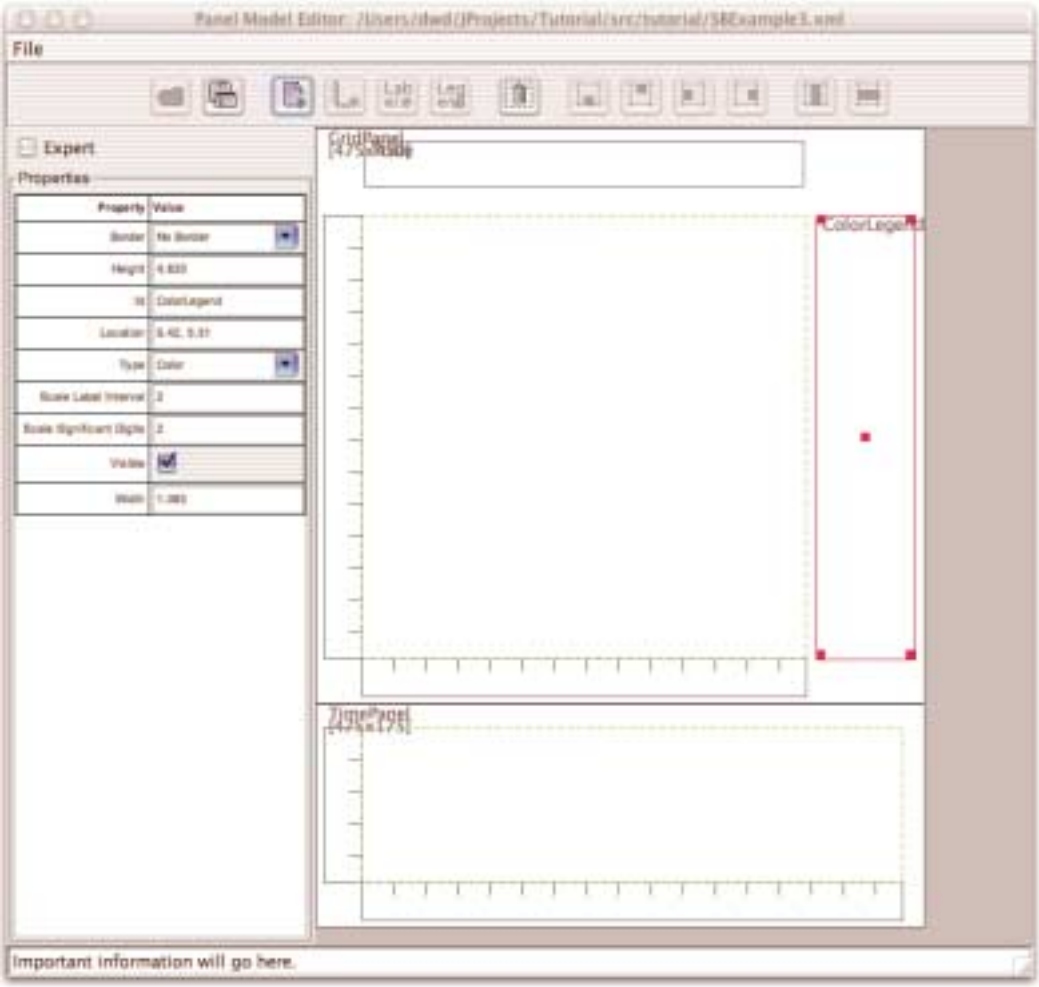

Figure 5. PanelModelEditor with Panel, Legend, and Label layout from the example.

name the Panel, Legend, and Label components. The components need to be named for later reference by the application. The top Panel is named “GridPanel”, the DataGroup named “GridData”, and the Legend named “ColorLegend”. The bottom Panel is named “TimePanel” and its DataGroup named “TimeSeries”.

The main application, SBExample3, creates a plot by instantiating Page (page_) and DataModel (dataModel_) objects with null constructors and the PanelModel (panelModel_) is instantiated from the serialized representation of a PanelModel create earlier using the PanelModelEditor (SBExample3.xml). The dataModel_ and panelModel_ objects are then passed to the page_ object.

Data is read from netCDF files using NcSimpleReader3 from a time series and grid dataset. We need to find the PanelHolder and DataGroup to which we want to add data. We can query panelModel_ to find the PanelHolder and DataGroup objects into which we want to add data. (It is also possible to query panelModel_ to determine what PanelHolders and DataGroups are available and then create a dialog that lets a user choose which PanelHolder, DataGroup, and Legend to use.)

The SGTData, Attribute, PanelHolder, DataGroup, and Legend are then added to dataModel_. The finished application is shown in Figure 6

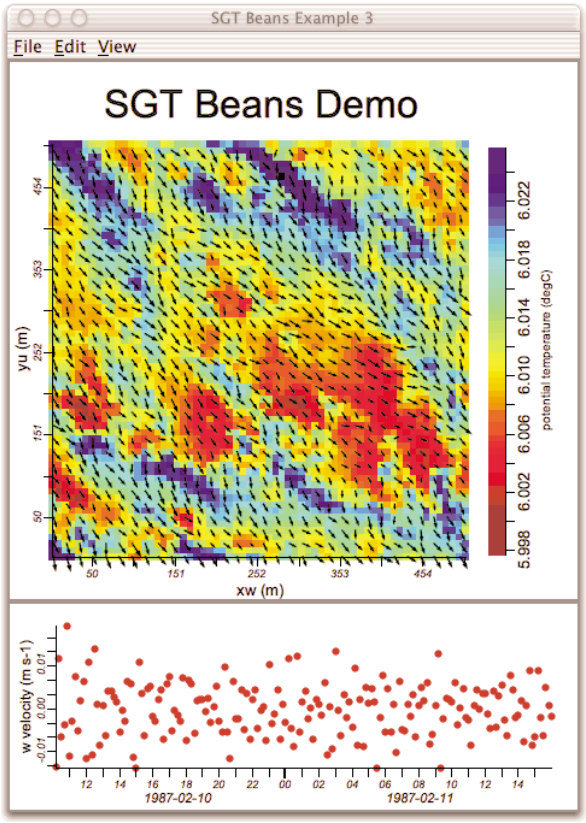

Figure 6. SGT Beans example.

(The code for SBExample3 and NcSimpleReader3, the application and netCDF reader, respectively, and SBExample3.xml can be found in the “APPENDIX” on page 4.)

6. FUTURE DIRECTIONS

Presently, the Panels are positioned on the Page without using a LayoutManager. This limitation requires the developer to manually resize and position the Panels if the Page is resized. We intend to add support for the SpringLayout and develop a GUI interface to adjust the SpringLayout properties. We will also add support for the new SGT Annotation objects. Annotations include line, oval, point, text, and rectangle objects.

Acknowledgment. This publication was supported by the Joint Institute for the Study of the Atmosphere and Ocean (JISAO) under NOAA Cooperative Agreement #NA17RJ1232, Contribution #1026. PMEL contribution 2638. The views expressed herein are those of the author(s) and do not necessarily reflect the views of NOAA or any of its subagencies. This work is funded by NOAA’s HPCC program.

7. REFERENCES

Denbo, D.W., 2003. Plug and Play Scientific Graphics with SGT. *19th International Conference on Interactive Information and Processing Systems*, AMS, 10-13 February 2003, Long Beach CA (on CD).

Denbo, D.W., 2001. Interactive graphics toolkit for Java applications and web applets. *17th Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, AMS, 14-19 January 2001, Albuquerque, NM, 372-375.

Denbo, D.W., 2000. The Scientific Graphics Toolkit. *Proceedings of Oceans '99 MTS/IEEE Conference*, 13-16 September, Seattle, WA (on CD).

Denbo, D.W., 1999. Using Java graphics to display ocean observations in NOAAServer. In *15th International Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, AMS, Dallas, TX, 10-15 January 1999, 442-444.

Denbo, D.W., 1997. NOAAServer Graphics Engine Architecture. Presented at *NOAA WebShop97*, October 22-23, 1997, Silver Spring, Maryland.

8. APPENDIX

Java code from the example. Available at http://www.epic.noaa.gov/java/sgt.

8.1 *SBExample3*

```
/*
 * $Id: SBExample3.java,v 1.5 2003/10/10 18:28:26 dwd Exp $
 *
 * This software is provided by NOAA for full, free and open release.  It is
 * understood by the recipient/user that NOAA assumes no liability for any
 * errors contained in the code.  Although this software is released without
 * conditions or restrictions in its use, it is expected that appropriate
 * credit be given to its author and to the National Oceanic and Atmospheric
 * Administration should the software be included by the recipient as an
 * element in other product development.
 */

package tutorial;
```

```
import java.awt.BorderLayout;
import java.awt.AWTEvent;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Toolkit;

import java.awt.event.WindowEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.awt.print.PrinterJob;
import java.awt.print.PageFormat;
import java.awt.print.PrinterException;

import java.io.IOException;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JSeparator;
import javax.swing.RepaintManager;

import gov.noaa.pmel.sgt.LineAttribute;
import gov.noaa.pmel.sgt.GridAttribute;
import gov.noaa.pmel.sgt.VectorAttribute;
import gov.noaa.pmel.sgt.ColorMap;
import gov.noaa.pmel.sgt.IndexedColorMap;
import gov.noaa.pmel.sgt.LinearTransform;
import gov.noaa.pmel.sgt.JPane;

import gov.noaa.pmel.sgt.beans.Page;
import gov.noaa.pmel.sgt.beans.PanelModel;
import gov.noaa.pmel.sgt.beans.DataModel;
import gov.noaa.pmel.sgt.beans.PanelHolder;
import gov.noaa.pmel.sgt.beans.DataGroup;
import gov.noaa.pmel.sgt.beans.Legend;
import gov.noaa.pmel.sgt.beans.PanelModelEditor;

import gov.noaa.pmel.sgt.dm.SGTData;

import gov.noaa.pmel.sgt.swing.JClassTree;

/**
 * SGT Bean Example3.  Uses the SGT Bean classes to create a simple
 * time series plot of data from a netCDF file.
 *
 * @author Donald Denbo
 * @version 1.0
 */
public class SBExample3 extends JFrame implements ActionListener {
  /**
   * Instantiate the three primary SGT Bean objects.
   * Page -       Main SGT JavaBean used with PanelModel and DataModel to
   *              create a graphic.
   * PanelModel - A model that supports the Panel structure of a plot.
   * DataModel -  A model that supplies the relationship between SGTData objects,
   *              its Attribute and the Panel and DataGroup in which it is
   *              displayed and the Legend.
   */
  private Page page_ = new Page();
  private PanelModel panelModel_ = new PanelModel();
  private DataModel dataModel_ = new DataModel();
  /**
   * The JPanel that will be used to contain the graphic.
   */
  private JPanel graphicPanel_ = new JPanel(new BorderLayout());

  private JMenuItem printMenuItem_;
  private JMenuItem exitMenuItem_;
```

```
private JMenuItem resetZoomMenuItem_;
private JMenuItem classTreeMenuItem_;
private JMenuItem filePageMenuItem_;
private JMenuItem panelModelMenuItem_;
private PageFormat pageFormat_ = PrinterJob.getPrinterJob().defaultPage();

/**
 * SGT Bean Example3 constructor.
 */
public SBExample3() {
  setTitle("SGT Beans Example 3");
  setJMenuBar(makeMenuBar());
  /**
   * Enable WindowEvents.  Set the layout of the content pane to a
   * BorderLayout.
   */
  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  getContentPane().setLayout(new BorderLayout());
  /**
   * Create the graphic and add it to the content pane of the JFrame.
   */
  createGraphic();
  getContentPane().add(graphicPanel_, BorderLayout.CENTER);
}

/**
 * Main method for SGT Bean Example 3.
 * @param args Program arguments (unused)
 */
public static void main(String[] args) {
  /**
   * Create an instance of SBExample3 and pack() the frame.
   */
  SBExample3 frame = new SBExample3();
  frame.pack();
  //Center the window
  Dimension sSize = Toolkit.getDefaultToolkit().getScreenSize();
  Dimension fSize = frame.getSize();
  if (fSize.height > sSize.height) {
    fSize.height = sSize.height;
  }
  if (fSize.width > sSize.width) {
    fSize.width = sSize.width;
  }
  frame.setLocation((sSize.width - fSize.width)/2, (sSize.height - fSize.height)/2);
  /**
   * Since the current version of SGT Beans doesn't directly support
   * resizing disable resizing and make the frame visible.
   */
  frame.setResizable(false);
  frame.setVisible(true);
}

/**
 * Create the graphic to insert into graphicPanel.
 */
private void createGraphic() {
  NcSimpleReader3 timeSeriesReader = null;
  NcSimpleReader3 gridReader = null;
  /**
   * Instantiate a NcSimpleReader3 with a time series data file and a grid
   * data file.  Exit on error.
   */
  try {
    timeSeriesReader = new NcSimpleReader3("data/OLEM_tutorial_ts.nc");
    gridReader = new NcSimpleReader3("data/OLEM_tutorial_grid.nc");
  } catch (IOException ioe) {
    ioe.printStackTrace();
    System.exit(1);
  }
  /**
   * Add the page object to graphicPanel and set dataModel.
```

```
 */
graphicPanel_.add(page_, BorderLayout.CENTER);
page_.setDataModel(dataModel_);
/**
 * Create panelModel by de-serializing an existing PanelModel.  The
 * file SBExample3.xml, was created using
 * gov.noaa.pmel.sgt.beans.PanelModelEditor.  Exit on error.
 */
try {
 panelModel_=PanelModel.loadFromXML(getClass().getResource("SBExample3.xml").openStream());
} catch (Exception e) {
  e.printStackTrace();
  System.exit(1);
}
/**
 * Set panelModel.
 */
page_.setPanelModel(panelModel_);
/**
 * To add data to dataModel, we first need to find the PanelHolder and
 * DataGroup to which we want to add data.  The SBExample3.xml panelModel
 * was created with a Panel named "TimePanel" and a DataGroup named
 * "TimeSeries".  It is possible to query panelModel to determine what
 * PanelHolders and DataGroups are available.
 */
PanelHolder timePanel = panelModel_.findPanelHolder("TimePanel");
DataGroup timeData = timePanel.findDataGroup("TimeSeries");
/**
 * Get a time series from the netCDF file.
 */
SGTData data = timeSeriesReader.getTimeSeries("w", 1);
/**
 * Create a line attribute for dark red circular marks.
 */
LineAttribute lAttr = new LineAttribute(LineAttribute.MARK,51,Color.red.darker());
/**
 * Add data to PanelHolder, time, and DataGroup, timeData.
 */
dataModel_.addData(data, lAttr, timePanel, timeData, null);
/**
 * SBExample3.xml also has a Panel named "GridPanel" and a DataGroup named
 * "GridData".  We will use these with a GridAttribute object to create
 * a grid plot.
 */
PanelHolder gridPanel = panelModel_.findPanelHolder("GridPanel");
DataGroup gridData = gridPanel.findDataGroup("GridData");
Legend gridLegend = gridPanel.findLegend("ColorLegend");
/**
 * Get a grid from the netCDF file.
 */
SGTData grid = gridReader.getGrid("t", 1);
/**
 * Create a GridAttribute.
 */
GridAttribute gAttr = createGridAttribute();
/**
 * Add data to PanelHolder, gridPanel, and DataGroup, gridData.
 */
dataModel_.addData(grid, gAttr, gridPanel, gridData, gridLegend);
/**
 * Get a vector from the netCDF file with a stride of 2.
 */
SGTData vector = gridReader.getVectorGrid(2);
/**
 * Create a VectorAttribute
 */
Color blackish = Color.black;
VectorAttribute vAttr = new VectorAttribute(VectorAttribute.SCALED_HEAD,
                                            1.0, blackish, 0.3);
vAttr.setWidth(1.5f);
/**
```

```
   * Add vector to PanelHolder, gridPanel, and DataGroup, gridData
   */
  dataModel_.addData(vector, vAttr, gridPanel, gridData, null);
}

/**
 * Process WindowEvents, stop program if WINDOW_CLOSING event is caught.
 * @param e WindowEvent
 */
protected void processWindowEvent(WindowEvent e) {
  super.processWindowEvent(e);
  if(e.getID() == WindowEvent.WINDOW_CLOSING) {
    exitMenuItem_actionPerformed(null);
  }
}

/**
 * Create a GridAttribute using a "rainbow" color map.
 *
 * @return GridAttribute
 */
private GridAttribute createGridAttribute() {
  ColorMap cmap;
  /**
   * Define red, green, and blue.
   */
  int[] red =
  {    0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,
       0,  7, 23, 39, 55, 71, 87,103,
     119,135,151,167,183,199,215,231,
     247,255,255,255,255,255,255,255,
     255,255,255,255,255,255,255,255,
     255,246,228,211,193,175,158,140};
  int[] green =
  {    0,  0,  0,  0,  0,  0,  0,  0,
       0, 11, 27, 43, 59, 75, 91,107,
     123,139,155,171,187,203,219,235,
     251,255,255,255,255,255,255,255,
     255,255,255,255,255,255,255,255,
     255,247,231,215,199,183,167,151,
     135,119,103, 87, 71, 55, 39, 23,
       7,  0,  0,  0,  0,  0,  0,  0};
  int[] blue =
  {    0,143,159,175,191,207,223,239,
     255,255,255,255,255,255,255,255,
     255,255,255,255,255,255,255,255,
     255,247,231,215,199,183,167,151,
     135,119,103, 87, 71, 55, 39, 23,
       7,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0};
  /**
   * Create an IndexedColorMap from the red, green, blue components.  These
   * colors  will be opaque (alpha=255).  Use a LinearTransform  to map the
   * data value to a color.
   */
  cmap = new IndexedColorMap(red, green, blue);
  LinearTransform ctrans =
    new LinearTransform(0.0, (double)red.length, 0.0, 1.0);
  ((IndexedColorMap)cmap).setTransform(ctrans);
  /**
   * Return a new RASTER grid attribute.
   */
  return new GridAttribute(GridAttribute.RASTER, cmap);
}
/**
 * Create application menubar
 */
JMenuBar makeMenuBar() {
  JMenuBar menuBar = new JMenuBar();
```

```
  JSeparator separator;

  /*
   * File Menu
   *   Print...
   *   Exit
   */
  JMenu fileMenu = new JMenu();
  fileMenu.setText("File");
  fileMenu.setMnemonic((int)'F');
  menuBar.add(fileMenu);
  filePageMenuItem_ = new JMenuItem();
  filePageMenuItem_.setText("Page Layout...");
  filePageMenuItem_.setMnemonic((int)'y');
  fileMenu.add(filePageMenuItem_);
  printMenuItem_ = new JMenuItem();
  printMenuItem_.setText("Print...");
  printMenuItem_.setMnemonic((int)'P');
  fileMenu.add(printMenuItem_);
  separator = new JSeparator();
  fileMenu.add(separator);
  exitMenuItem_ = new JMenuItem();
  exitMenuItem_.setText("Exit");
  exitMenuItem_.setMnemonic((int)'X');
  fileMenu.add(exitMenuItem_);

  /*
   * Edit Menu
   *   PanelModel...
   */
  JMenu editMenu = new JMenu();
  editMenu.setText("Edit");
  editMenu.setMnemonic((int)'E');
  menuBar.add(editMenu);
  panelModelMenuItem_ = new JMenuItem();
  panelModelMenuItem_.setText("PanelModel...");
  panelModelMenuItem_.setMnemonic((int)'M');
  editMenu.add(panelModelMenuItem_);
  /*
   * View Menu
   *   Reset Zoom
   *   Class Tree...
   */
  JMenu viewMenu = new JMenu();
  viewMenu.setText("View");
  viewMenu.setMnemonic((int)'V');
  menuBar.add(viewMenu);
  resetZoomMenuItem_ = new JMenuItem();
  resetZoomMenuItem_.setText("Reset Zoom");
  resetZoomMenuItem_.setMnemonic((int)'Z');
  viewMenu.add(resetZoomMenuItem_);
  separator = new JSeparator();
  viewMenu.add(separator);
  classTreeMenuItem_ = new JMenuItem();
  classTreeMenuItem_.setText("Class Tree...");
  classTreeMenuItem_.setMnemonic((int)'C');
  viewMenu.add(classTreeMenuItem_);

  /*
   * add listeners for menu items
   */
  exitMenuItem_.addActionListener(this);
  printMenuItem_.addActionListener(this);
  resetZoomMenuItem_.addActionListener(this);
  classTreeMenuItem_.addActionListener(this);
  filePageMenuItem_.addActionListener(this);
  panelModelMenuItem_.addActionListener(this);

  return menuBar;
}

/**
```

```
 * Listen for events from menu items
 */
public void actionPerformed(ActionEvent event) {
  Object object = event.getSource();
  if(object == printMenuItem_)
    printMenuItem_actionPerformed(event);
  else if(object == exitMenuItem_)
    exitMenuItem_actionPerformed(event);
  else if(object == resetZoomMenuItem_)
    resetZoomMenuItem_actionPerformed(event);
  else if(object == classTreeMenuItem_)
    classTreeMenuItem_actionPerformed(event);
  else if(object == filePageMenuItem_)
    filePageMenuItem_actionPerformed(event);
  else if(object == panelModelMenuItem_)
    panelModelMenuItem_actionPerformed(event);
}

/**
 * Close application, dispose of resources, and exit.
 */
void exitMenuItem_actionPerformed(ActionEvent event) {
  /**
   * Clean up windows and exit.
   */
  setVisible(false);
  dispose();
  System.exit(0);
}

/**
 * Print Page.
 */
void printMenuItem_actionPerformed(ActionEvent event) {
  Color saveColor;
  JPane pane = page_.getJPane();

  PrinterJob printJob = PrinterJob.getPrinterJob();
  printJob.setPrintable(page_, pageFormat_);
  printJob.setJobName("BeanDemo");
  if(printJob.printDialog()) {
    try {
      RepaintManager currentManager = RepaintManager.currentManager(pane);
      currentManager.setDoubleBufferingEnabled(false);
      printJob.print();
      currentManager.setDoubleBufferingEnabled(true);
    } catch (PrinterException pe) {
      System.out.println("Error printing: " + pe);
    }
  }
}

/**
 * Reset the zoom for all windows.
 */
void resetZoomMenuItem_actionPerformed(ActionEvent event) {
  page_.resetZoom();
}

/**
 * Open the JClassTree with the page_ JPane.  The JClassTree provides
 * access to many dialogs that can be used for inspecting/editing
 * sgt classes.
 */
void classTreeMenuItem_actionPerformed(ActionEvent event) {
  JClassTree ct = new JClassTree();
  ct.setModal(false);
  ct.setJPane(page_.getJPane());
  ct.show();
}

/**
```

```
   * Edit the printer page layout.
   */
  void filePageMenuItem_actionPerformed(ActionEvent e) {
    PrinterJob pj = PrinterJob.getPrinterJob();
    pageFormat_ = pj.pageDialog(pageFormat_);
  }

  /**
   * Edit/Modify the PanelModel.
   */
  void panelModelMenuItem_actionPerformed(ActionEvent e) {
    PanelModelEditor pme = new PanelModelEditor(panelModel_);
    pme.setVisible(true);

  }
}
```

8.2 *NcSimpleReader3*

```
/*
 * $Id: NcSimpleReader3.java,v 1.4 2003/10/10 18:28:26 dwd Exp $
 *
 * This software is provided by NOAA for full, free and open release.  It is
 * understood by the recipient/user that NOAA assumes no liability for any
 * errors contained in the code.  Although this software is released without
 * conditions or restrictions in its use, it is expected that appropriate
 * credit be given to its author and to the National Oceanic and Atmospheric
 * Administration should the software be included by the recipient as an
 * element in other product development.
 */

package tutorial;

import java.io.IOException;

import java.util.Iterator;

import ucar.nc2.NetcdfFile;
import ucar.nc2.Variable;

import ucar.ma2.Array;
import ucar.ma2.InvalidRangeException;
import ucar.ma2.Range;

import gov.noaa.pmel.sgt.dm.SGTData;
import gov.noaa.pmel.sgt.dm.SGTGrid;
import gov.noaa.pmel.sgt.dm.SGTVector;
import gov.noaa.pmel.sgt.dm.SGTMetaData;
import gov.noaa.pmel.sgt.dm.SimpleLine;
import gov.noaa.pmel.sgt.dm.SimpleGrid;

import gov.noaa.pmel.util.GeoDateArray;
import gov.noaa.pmel.util.GeoDate;
import gov.noaa.pmel.util.IllegalTimeValue;

/**
 * NcSimpleReader3 demonstrates how a time series can be read from a netCDF
 * file and then encapsulated in a SGTData object.  NOTE: This example makes
 * several assumptions about the netCDF file.  The variables are assumed to
 * be float, the time axis int, and the time units of the form
 * "seconds since 1989-10-11 12:34 GMT".
 *
 * @author Donald Denbo
 * @version 1.0
 */
public class NcSimpleReader3 {
  private NetcdfFile ncFile_;
  /**
   * Construct the netCDF file reader.
   *
   * @param file NetCDF file name
```

```
 * @throws IOException if error occurs while openning the file.
 */
public NcSimpleReader3(String file) throws IOException {
  /*
  * Open a netCDF file.  Any exceptions are handled by the caller.
  */
  ncFile_ = new NetcdfFile(file);
}

/**
 * Reads 'u' and 'v' and combines into a vector.
 * @param step stride between elements
 * @return SGTData (SGTVector)
 */
public SGTData getVectorGrid(int step) {
  SGTGrid uGrid = (SGTGrid)getGrid("u", step);
  SGTGrid vGrid = (SGTGrid)getGrid("v", step);
  return new SGTVector(uGrid, vGrid);
}

/**
 * Reads a horizontal (x,y) grid from the netCDF data file.  Assumes that the
 * x axis name and y axis name contains "x" and "y", respectively.
 * Always returns a grid with the origin at {0, 0, ... 0} and the shape = 1,
 * except for the x and y axes.
 *
 * @param varName Variable name to be accessed as a time series
 * @param step stride between elements
 * @return SGTData (SimpleGrid).
 */
public SGTData getGrid(String varName, int step) {
  SimpleGrid grid = null;
  SGTMetaData xMeta = null;
  SGTMetaData yMeta = null;
  SGTMetaData zMeta = null;
  int[] dimShape = new int[1];
  int[] dimOrigin = new int[1];
  int xIndex = -1;
  int yIndex = -1;
  double[] xArray = null;
  double[] yArray = null;
  double[] zArray = null;

  /**
   * Get the requested grid variable and the x and y dimension variables. If
   * any variable doesn't exist return null.
   */
  Variable var = ncFile_.findVariable(varName);
  Variable xDim = null;
  Variable yDim = null;
  Iterator varIter = ncFile_.getVariableIterator();

  Variable dim = null;
  while(varIter.hasNext()) {
    dim = (Variable)varIter.next();
    if(dim.isCoordinateVariable()) {
      if(dim.getName().indexOf("x") != -1) {
        xDim = dim;
      } else if(dim.getName().indexOf("y") != -1) {
        yDim = dim;
      }
    }
  }
  if(var == null || xDim == null || yDim == null) return null;

  /**
   * Get the rank and shape of the grid.  Set the origin[] = {0, 0, ... }.
   * Find which are the x and y dimensions.  If the grid doesn't have either
   * x or y dimensions return null.
   */
  int rank = var.getRank();
  int[] shape = var.getShape();
```

```
  int[] origin = new int[rank];
  for(int i=0; i < rank; i++) { origin[i] = 0; };
  for(int i=0; i < rank; i++) {
    if(var.getDimension(i).getName().indexOf("x") != -1) xIndex = i;
    if(var.getDimension(i).getName().indexOf("y") != -1) yIndex = i;
  }
  if(xIndex == -1 || yIndex == -1) return null;

  /**
   * The coordinate variables are one-dimensional. Use the shape from the
   * grid variable to read the x and y arrays from the netCDF file.
   */
  dimOrigin[0] = 0;
  try {
    dimShape[0] = shape[xIndex];
    xArray = getDoubleArray(xDim, dimOrigin, dimShape, step);
    dimShape[0] = shape[yIndex];
    yArray = getDoubleArray(yDim, dimOrigin, dimShape, step);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  } catch (InvalidRangeException ire) {
    ire.printStackTrace();
  }
  /**
   * Set shape = 1 for all dimensions not x or y.
   */
  for(int i=0; i < rank; i++) {
    if(i != xIndex && i != yIndex) shape[i] = 1;
  }
  /**
   * Read the grid data from the netCDF file.
   */
  try {
    zArray = getDoubleArray(var, origin, shape, step);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  } catch (InvalidRangeException ire) {
    ire.printStackTrace();
  }
  /**
   * Create the x and y axis metadata and the z grid metadata.
   */
  xMeta = new SGTMetaData(xDim.getName(),
                          xDim.findAttribute("units").getStringValue());
  yMeta = new SGTMetaData(yDim.getName(),
                          yDim.findAttribute("units").getStringValue());
  zMeta = new SGTMetaData(var.findAttribute("long_name").getStringValue(),
                          var.findAttribute("units").getStringValue());
  /**
   * Create a SimpleGrid from the axes coordinate variables and the grid
   * variable.  Set the metadata.
   */
  grid = new SimpleGrid(zArray, xArray, yArray, varName + " Grid");
  grid.setXMetaData(xMeta);
  grid.setYMetaData(yMeta);
  grid.setZMetaData(zMeta);

  return grid;
}

/**
 * Reads a times series from the netCDF data file.  Assumes that time is in
 * a coordinate variable named "time".  Returns a line with the
 * origin at {0, 0, ... 0} and shape = 1, except for the time extent.
 *
 * @param varName Variable name to be accessed as a time series
 * @param step stride between elements
 * @return SGTData (SimpleLine).
 */
public SGTData getTimeSeries(String varName, int step) {
  SimpleLine line = null;
  SGTMetaData xMeta = null;
```

```
  SGTMetaData yMeta = null;
  /**
   * Get the requested variable and the time dimension variable. If either
   * variable doesn't exist or time isn't a coordinate variable return null.
   */
  Variable var = ncFile_.findVariable(varName);
  Variable timeDim = ncFile_.findVariable("time");
  int time = -1;
  if(var == null || timeDim == null || !timeDim.isCoordinateVariable()) return null;
  /**
   * Get the rank and shape of var.  Set the origin[] = {0, 0, ... }.
   * Find which dimension is time.  If this variable doesn't have a time
   * dimension return null.
   */
  int rank = var.getRank();
  int[] shape = var.getShape();
  int[] origin = new int[rank];
  for(int i=0; i < rank; i++) { origin[i] = 0;};
  for(int i=0; i < rank; i++) {
    if(var.getDimension(i).getName().equals("time")) {
      time = i;
      break;
    }
  }
  if(time < 0) return null;
  /**
   * Set shape = 1 for all non-time dimensions.
   */
  for(int i=0; i < rank; i++) {
    if(i != time) shape[i] = 1;
  }
  /**
   * Read the variable and time coordinate variable from the netCDF file.
   */
  Array dimData = null;
  double[] dataD = null;
  GeoDateArray gArray = null;
  try {
    dataD = getDoubleArray(var, origin, shape, step);
    gArray = getGeoDateArray(timeDim, step);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  } catch (InvalidRangeException ire) {
    ire.printStackTrace();
  }
  /**
   * Create the x and y axis metadata.  The xMeta object is not used to
   * label the graph because a TimeAxis is labelled using the time information.
   */
  xMeta = new SGTMetaData("X axis", "units");
  yMeta = new SGTMetaData(var.findAttribute("long_name").getStringValue(),
                          var.findAttribute("units").getStringValue());

  /**
   * Create a SimpleLine from the time coordinate and data variable.  Set the
   * metadata.
   */
  line = new SimpleLine(gArray, dataD, varName + " Time Series");
  line.setXMetaData(xMeta);
  line.setYMetaData(yMeta);

  return line;
}

/**
 * Read a variable from the netCDF file.
 *
 * @param var netCDF Variable
 * @param origin starting index
 * @param shape extent in each dimension.
 * @param step stride between elements
 * @return data array
```

```
 * @throws IOException
 * @throws InvalidRangeException
 */
double[] getDoubleArray(Variable var, int[] origin, int[] shape, int step)
  throws IOException, InvalidRangeException {
  /**
   * Construct a Range object for subsampling the data.
   */
  Range[] range = new Range[var.getRank()];
  for(int i=0; i < var.getRank(); i++) {
    range[i] = new Range(origin[i], origin[i] + shape[i] - 1, step);
  }
  /**
   * Read the variable from the netCDF file.
   */
  Array varData = null;
  varData = var.read(origin, shape).sectionNoReduce(range);
  /**
   * Extract the variable MultiArray data into a 1 dimensional java array
   * and convert to double[].  Assumes that the netCDF variable is float.
   */
  float[] data = (float[])varData.copyTo1DJavaArray();
  double[] dataD = new double[data.length];
  for(int i=0; i < data.length; i++) {
    dataD[i] = data[i];
  }
  return dataD;
}

/**
 * Read the time coordinate variable from the netCDF file.
 * @param timeDim netCDF time coordinate variable
 * @return GeoDateArray containing time axis data
 * @throws IOException
 * @throws InvalidRangeException
 */
GeoDateArray getGeoDateArray(Variable timeDim, int step)
    throws IOException, InvalidRangeException {
  Array dimData = null;
  /**
   * Construct a Range object for subsampling the data.
   */
  Range[] range = new Range[1];
  int len = (int)timeDim.getSize();
  range[0] = new Range(0, len-1,step);
  /**
   * Read the time coordinate variable from the netCDF file.
   */
  dimData = timeDim.read().sectionNoReduce(range);
  /**
   * Extract the time coordinate variable MultiArray data into a 1 dimensional
   * java array and convert to a GeoDateArray object.  Assumes the time
   * data is int and the units are well behaved.
   */
  int[] axis = (int[])dimData.copyTo1DJavaArray();
  long[] axisL = new long[axis.length];
  /**
   * Get the time units and parse to obtain the scaling factor and offset
   */
  String timeU = timeDim.findAttribute("units").getStringValue();
  int space = timeU.indexOf(" ");
  int since = timeU.indexOf("since");
  String timeUnits = timeU.substring(0,space);
  String timeOffset = timeU.substring(since+6);
  long factor = 1;
  if(timeUnits.startsWith("second")) {
    factor = 1000;
  } else if(timeUnits.startsWith("min")) {
    factor = 60000;
  } else if(timeUnits.startsWith("hour")) {
    factor = 3600000;
  } else if(timeUnits.startsWith("day")) {
```

```
      factor = 86400000;
    }
    long offset = 0;
    try {
      /*
       * Using GeoDate to parse the offset.  GeoDate forces a GMT calendar.
       */
      offset = new GeoDate(timeOffset, "yyyy-MM-dd HH:mm").getTime();
    } catch (IllegalTimeValue itv) { }
    /**
     * Apply the scaling and offset, then instantiate a GeoDateArray containing
     * the time axis values.
     */
    for(int i=0; i < axis.length; i++) {
      axisL[i] = axis[i]*factor + offset;
    }
    return new GeoDateArray(axisL);
  }
}
```

8.3 *SBExample3.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1_01" class="java.beans.XMLDecoder">
 <object class="gov.noaa.pmel.sgt.beans.PanelModel">
  <void property="pageSize">
   <object class="java.awt.Dimension">
    <int>475</int>
    <int>625</int>
   </object>
  </void>
  <void property="panelList">
   <void method="put">
    <string>TimePanel</string>
    <object class="gov.noaa.pmel.sgt.beans.PanelHolder">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>0</int>
       <int>450</int>
       <int>475</int>
       <int>175</int>
      </object>
     </void>
     <void property="dataGroups">
      <void method="put">
       <string>TimeSeries</string>
       <object class="gov.noaa.pmel.sgt.beans.DataGroup">
        <void property="XAxisHolder">
         <void property="axisType">
          <int>3</int>
         </void>
         <void property="boundsP">
          <void property="height">
           <double>0.4166666567325592</double>
          </void>
          <void property="width">
           <double>5.847222328186035</double>
          </void>
          <void property="x">
           <double>0.5</double>
          </void>
          <void property="y">
           <double>0.0833333358168602</double>
          </void>
         </void>
         <void property="transformGroup">
          <string>TimeSeries</string>
         </void>
         <void property="userRange">
          <object class="gov.noaa.pmel.util.SoTRange$Time">
           <void id="SoTValue$Time0" property="delta">
```

```
    <void property="value">
     <long>172800000</long>
    </void>
   </void>
   <void property="delta">
    <object idref="SoTValue$Time0"/>
   </void>
   <void id="SoTValue$Time1" property="end">
    <void property="value">
     <long>978307200000</long>
    </void>
   </void>
   <void property="end">
    <object idref="SoTValue$Time1"/>
   </void>
   <void id="SoTValue$Time2" property="start">
    <void property="value">
     <long>946684800000</long>
    </void>
   </void>
   <void property="start">
    <object idref="SoTValue$Time2"/>
   </void>
  </object>
 </void>
</void>
<void property="YAxisHolder">
 <void property="boundsP">
  <void property="height">
   <double>1.6805555820465088</double>
  </void>
  <void property="width">
   <double>0.4166666567325592</double>
  </void>
  <void property="x">
   <double>0.0833333358168602</double>
  </void>
  <void property="y">
   <double>0.5</double>
  </void>
 </void>
 <void property="labelInterval">
  <int>3</int>
 </void>
 <void property="title">
  <void property="moveable">
   <boolean>false</boolean>
  </void>
 </void>
 <void property="transformGroup">
  <string>TimeSeries</string>
 </void>
 <void property="userRange">
  <void id="SoTValue$Double0" property="delta"/>
  <void property="delta">
   <object idref="SoTValue$Double0"/>
  </void>
  <void id="SoTValue$Double1" property="end"/>
  <void property="end">
   <object idref="SoTValue$Double1"/>
  </void>
  <void id="SoTValue$Double2" property="start"/>
  <void property="start">
   <object idref="SoTValue$Double2"/>
  </void>
 </void>
</void>
<void property="ZRangeU">
 <void id="SoTValue$Double3" property="delta"/>
 <void property="delta">
  <object idref="SoTValue$Double3"/>
 </void>
```

```
       <void id="SoTValue$Double4" property="end"/>
       <void property="end">
        <object idref="SoTValue$Double4"/>
       </void>
       <void id="SoTValue$Double5" property="start"/>
       <void property="start">
        <object idref="SoTValue$Double5"/>
       </void>
      </void>
      <void property="id">
       <string>TimeSeries</string>
      </void>
      <void id="Margin0" property="margin"/>
      <void property="margin">
       <object idref="Margin0"/>
      </void>
     </object>
    </void>
   </void>
   <void property="id">
    <string>TimePanel</string>
   </void>
  </object>
 </void>
 <void method="put">
  <string>GridPanel</string>
  <object class="gov.noaa.pmel.sgt.beans.PanelHolder">
   <void property="bounds">
    <object class="java.awt.Rectangle">
     <int>0</int>
     <int>0</int>
     <int>475</int>
     <int>450</int>
    </object>
   </void>
   <void property="dataGroups">
    <void method="put">
     <string>GridData</string>
     <object class="gov.noaa.pmel.sgt.beans.DataGroup">
      <void property="XAxisHolder">
       <void property="boundsP">
        <void property="height">
         <double>0.4166666567325592</double>
        </void>
        <void property="width">
         <double>4.80555534362793</double>
        </void>
        <void property="x">
         <double>0.5</double>
        </void>
        <void property="y">
         <double>0.0833333358168602</double>
        </void>
       </void>
       <void property="title">
        <void property="moveable">
         <boolean>false</boolean>
        </void>
       </void>
       <void property="transformGroup">
        <string>GridData</string>
       </void>
       <void property="userRange">
        <void id="SoTValue$Double6" property="delta"/>
        <void property="delta">
         <object idref="SoTValue$Double6"/>
        </void>
        <void id="SoTValue$Double7" property="end"/>
        <void property="end">
         <object idref="SoTValue$Double7"/>
        </void>
        <void id="SoTValue$Double8" property="start"/>
```

```
   <void property="start">
    <object idref="SoTValue$Double8"/>
   </void>
  </void>
 </void>
 <void property="YAxisHolder">
  <void property="boundsP">
   <void property="height">
    <double>4.80555534362793</double>
   </void>
   <void property="width">
    <double>0.4166666567325592</double>
   </void>
   <void property="x">
    <double>0.0833333358168602</double>
   </void>
   <void property="y">
    <double>0.5</double>
   </void>
  </void>
  <void property="title">
   <void property="moveable">
    <boolean>false</boolean>
   </void>
  </void>
  <void property="transformGroup">
   <string>GridData</string>
  </void>
  <void property="userRange">
   <void id="SoTValue$Double9" property="delta"/>
   <void property="delta">
    <object idref="SoTValue$Double9"/>
   </void>
   <void id="SoTValue$Double10" property="end"/>
   <void property="end">
    <object idref="SoTValue$Double10"/>
   </void>
   <void id="SoTValue$Double11" property="start"/>
   <void property="start">
    <object idref="SoTValue$Double11"/>
   </void>
  </void>
 </void>
 <void property="ZRangeU">
  <void id="SoTValue$Double12" property="delta"/>
  <void property="delta">
   <object idref="SoTValue$Double12"/>
  </void>
  <void id="SoTValue$Double13" property="end"/>
  <void property="end">
   <object idref="SoTValue$Double13"/>
  </void>
  <void id="SoTValue$Double14" property="start"/>
  <void property="start">
   <object idref="SoTValue$Double14"/>
  </void>
 </void>
 <void property="id">
  <string>GridData</string>
 </void>
 <void id="Margin1" property="margin">
  <void property="right">
   <float>1.2916666</float>
  </void>
  <void property="top">
   <float>0.9444444</float>
  </void>
 </void>
 <void property="margin">
  <object idref="Margin1"/>
 </void>
</object>
```

```
      </void>
     </void>
     <void property="id">
      <string>GridPanel</string>
     </void>
     <void property="labels">
      <void method="put">
       <string>Title</string>
       <object class="gov.noaa.pmel.sgt.beans.Label">
        <void property="boundsP">
         <object class="gov.noaa.pmel.util.Rectangle2D$Double">
          <void property="height">
           <double>0.5</double>
          </void>
          <void property="width">
           <double>4.763888835906982</double>
          </void>
          <void property="x">
           <double>0.5277777910232544</double>
          </void>
          <void property="y">
           <double>5.611111164093018</double>
          </void>
         </object>
        </void>
        <void property="id">
         <string>Ttitle</string>
        </void>
        <void property="justification">
         <int>1</int>
        </void>
        <void property="text">
         <string>SGT Beans Demo</string>
        </void>
       </object>
      </void>
     </void>
     <void property="legends">
      <void method="put">
       <string>ColorLegend</string>
       <object class="gov.noaa.pmel.sgt.beans.Legend">
        <void property="boundsP">
         <object class="gov.noaa.pmel.util.Rectangle2D$Double">
          <void property="height">
           <double>4.833333492279053</double>
          </void>
          <void property="width">
           <double>1.0833333730697632</double>
          </void>
          <void property="x">
           <double>5.416666507720947</double>
          </void>
          <void property="y">
           <double>0.4722222089767456</double>
          </void>
         </object>
        </void>
        <void property="id">
         <string>ColorLegend</string>
        </void>
        <void property="type">
         <int>1</int>
        </void>
       </object>
      </void>
     </void>
    </object>
   </void>
  </void>
 </object>
</java>
```