Exporting Images

1. Encoding

jCharts exports to the following formats: **PNG**, **JPEG**, and **SVG**, via encoder objects located in the `org.krysalis.jcharts.encoders` package. Each of these encoders has a method to encode your charts:

`encode( Chart chart, OutputStream outputStream )` - a generic version so you can pass any implementation of OutputStream through this method, such as FileOutputStream.

PNG and **JPEG** support is provided through the JSDK 1.4. In other words, you need the JSDK 1.4 to use the provided jCharts image encoders. However, there is now a `JPEGEncoder13` class for legacy support of older JDK's.

SVG support is provided via the Apache XML project: **Batik**. Please visit: http://xml.apache.org/batik/

2. Images From A Servlet

There has been a helper Class, `org.krysalis.jcharts.encoders.ServletEncoderHelper`, added for exporting charts from Servlets and JSP's. There are methods on this class to export to any of the jCharts supported formats, with the added benefit of automatically setting the **MIME** type of the chart for your browser so that the browser knows how to render the image correctly.

I don't know about you, but I always had trouble remembering the **MIME** types for different file formats.

Note:

Some users attempt to write a chart image to disk then stream it back to the browser. This is not necessary! You can simply stream the image back to the browser by using the affore mentioned helper class, which will avoid expensive file io.

Note:

This helper Class was created so as to avoid a compile time dependency on the J2EE jar. If you simply overload the `encode` method on the xxxEncoder Class, the compiler will try to load all argument Classes for each overloaded signature.

3. Images With No X Server

The JSDK 1.4+ allows jCharts to run on a headless *nix box without a virtual frame buffer! Set the follow property may be specified at the java command line: `-Djava.awt.headless=true` or a less flexible placement in your code: `System.setProperty("java.awt.headless","true");`

Otherwise, if you are running pre JDK 1.4, you will have to use a virtual frame buffer, like: xvfb.

4. Charts In Swing Apps

You can also use jcharts inside a Swing application. Below is the `SwingDemo` Class from the `org.krysalis.jcharts.demo.swing` package. Here, you simply set the `Graphics` Object from your Swing Component, into the Chart and call: `render()`

```
import org.krysalis.jcharts.chartData.ChartDataException;
import org.krysalis.jcharts.chartData.PieChartDataSet;
import org.krysalis.jcharts.properties.PropertyException;
import org.krysalis.jcharts.properties.PieChart2DProperties;
import org.krysalis.jcharts.properties.ChartProperties;
import org.krysalis.jcharts.properties.LegendProperties;
import org.krysalis.jcharts.nonAxisChart.PieChart2D;

import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowEvent;


public class SwingDemo extends JFrame
{
  private JPanel panel;


  public SwingDemo() throws ChartDataException, PropertyException
  {
    initComponents();
  }


  private void initComponents() throws ChartDataException,PropertyException
  {
    this.setSize( 500, 500 );
    this.panel = new JPanel( true );
    this.panel.setSize( 500, 500 );
    this.getContentPane().add( this.panel );

    this.pieChart2DProperties = new PieChart2DProperties();
    this.legendProperties= new LegendProperties();
```

```
    this.chartProperties= new ChartProperties();

    this.setVisible( true );


    addWindowListener( new java.awt.event.WindowAdapter()
    {
      public void windowClosing( WindowEvent windowEvent )
      {
        exitForm( windowEvent );
      }
    }
    );
  }


  /**************************************************************************
        *
        * @param graphics
        *************************************************************************/
  public void paint( Graphics graphics )
  {
    try {
      String[] labels = {"BMW", "Audi", "Lexus"};
      String title = "Cars that Own";
      Paint[] paints = {Color.blue, Color.gray, Color.red};
      double[] data = {50d, 30d, 20d};
      PieChartDataSet pieChartDataSet = new PieChartDataSet( title, data,
                                labels, paints, this.pieChart2DProperties );
      Dimension dimension= this.panel.getSize();
      PieChart2D pieChart2D = new PieChart2D( pieChartDataSet,
                                              this.legendProperties,
                                              this.chartProperties,
                                              (int) dimension.getWidth(),
                                              (int) dimension.getHeight() );

       //***** BEGIN SWING SPECIFIC CODE ******************************
       pieChart2D.setGraphics2D( (Graphics2D) this.panel.getGraphics() );
       pieChart2D.render();
       //***** END SWING SPECIFIC CODE ********************************
    }
    catch( ChartDataException chartDataException ) {
      chartDataException.printStackTrace();
    }
    catch( PropertyException propertyException ) {
      propertyException.printStackTrace();
    }
  }


  private void exitForm( WindowEvent windowEvent )
  {
    System.exit( 0 );
  }
```

```
  public static void main( String args[] ) throws ChartDataException,
                                                   PropertyException
  {
    new SwingDemo();
  }
}
```