Tapestry5HowToCreateASimpleGraphComponent

This is an example of a simple Pie chart component(tested with 5.0.5) using JFreeChart.

(This is similar to Tapestry5HowToCreatePieChartsInAPage)

To use it, add this to your page template:

```
<t:chart width="200" height="150" context="chart1" popup="popupSize"/>
<t:chart width="400" height="300" context="chart2" popup="popupSize"/>
```

add this to your page class:

```
public String[] getChart1(){
    return new String[]{"aa","22","bb","5"};
}

public String[] getChart2(){
    return new String[]{"aa","29","bb","30","cc","10"};
}
public int[] getpopupSize(){
    return new int[]{800,600};
}
```

add this to your pom.xml under "dependencies":

```
        <dependency>
                <groupId>jfree</groupId>
                <artifactId>jfreechart</artifactId>
                <version>1.0.5</version>
        </dependency>
```

Source:

(put this component class into yourApp.components package)

```
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import org.apache.tapestry.ComponentResources;
import org.apache.tapestry.Link;
import org.apache.tapestry.MarkupWriter;
import org.apache.tapestry.StreamResponse;
import org.apache.tapestry.annotations.Inject;
import org.apache.tapestry.annotations.Parameter;
import org.apache.tapestry.dom.Element;
import org.apache.tapestry.ioc.services.TypeCoercer;
import org.apache.tapestry.services.Response;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PiePlot3D;
import org.jfree.data.DefaultKeyedValues;
import org.jfree.data.general.DefaultPieDataset;
import org.jfree.data.general.PieDataset;

public class Chart{

    /**list(array) of paired values(label & value): [String,Number,String,Number,...]*/
    @Parameter(required=true)
    private List<Object> _context;

    @Parameter(required=true)
```

```
    private int _width;

    @Parameter(required=true)
    private int _height;

    /** width and height of the popup chart, if omitted, javascript for popup chart is omitted from output*/
    @Parameter
    private int[] _popup;

    @Inject
    private ComponentResources _resources;

    @Inject
    private TypeCoercer _coercer;

    @SuppressWarnings("unchecked")
    void beginRender(MarkupWriter writer)
    {
        //add width and height to begining of parameters
        Object[] contextArray = _context == null ? new Object[0] : _context.toArray();
        Object[] params = new Object[contextArray.length+2];
        System.arraycopy(contextArray, 0, params, 2, contextArray.length);
        params[0] = new Integer(_width);
        params[1] = new Integer(_height);

        //generate action link
        Link link = _resources.createActionLink("chart", false, params);
        Element img = writer.element("img", "src", link);

        //add javascript for popup
        if(_popup != null && _popup.length > 1){
            params[0] = _popup[0];
            params[1] = _popup[1];
            link = _resources.createActionLink("chart", false, params);
            img.attribute("onclick", "window.open('"+link+"','_blank','width="+(_popup[0]+24)+", height="+
(_popup[1]+24)+"')");
            img.attribute("style", "cursor:pointer");
        }

        _resources.renderInformalParameters(writer);
    }

    void afterRender(MarkupWriter writer)
    {
        writer.end();
    }

    public StreamResponse onChart(final int width, final int height, Object ...rest){
        DefaultKeyedValues values = new DefaultKeyedValues();
        for (int i = 3; i < rest.length; i+=2){
            values.addValue(rest[i-1].toString(), _coercer.coerce(rest[i], Number.class));
        }
        PieDataset pieDataset = new DefaultPieDataset(values);

        PiePlot3D plot = new PiePlot3D(pieDataset);
        plot.setForegroundAlpha(0.5f);
        plot.setDepthFactor(0.1);
        plot.setCircular(true);

        final JFreeChart chart = new JFreeChart(plot);

        return new StreamResponse(){
            public String getContentType(){
                return "image/jpeg";
            }
            public InputStream getStream() throws IOException {
                BufferedImage image  = chart.createBufferedImage(width, height);
                ByteArrayOutputStream byteArray = new ByteArrayOutputStream() ;
                ChartUtilities.writeBufferedImageAsJPEG(byteArray, image) ;
                return new ByteArrayInputStream(byteArray.toByteArray());
            }
```

```
            public void prepareResponse(Response response){}
        };
    }
}
```

This component has a limited usability since all data is sent via URL, and chart options are hardcoded. A better solution would be a component that takes a JFreeChart object as parameter and persits it for later viewing...

If you have more than few similar charts with not too much data, using modified version of this component can be a good enough solution.