RendererLite

Introduction

LiteRenderer is the simpler and smaller one. It is simple and lean, with low memory overheads since it works in full streaming mode, that is, it loads and processes only one feature at a time from the persistent storage. This is also its Achille's heel, since it has to load everything from the data store every time it renders the map. As such, it is better suited for stateless rendering, where the MapContext is displayed just one time, and when the input data is so big that it cannot conceivably fit into the main memory.

In the meantime this renderer supports on-the-fly reprojection (coordinate reference system changes).

Example

```
//Create the feature collection
        // Request extent
        Envelope ex = new Envelope(5, 15, 5,
15);

        AttributeType[] types = new
AttributeType[2];

        GeometryFactory geomFac = new
GeometryFactory();

        //simpleline
        Coordinate[] linestringCoordinates =
new Coordinate[7];
        linestringCoordinates[0] = new
Coordinate(5.0d,5.0d);
        linestringCoordinates[1] = new
Coordinate(6.0d,5.0d);
        linestringCoordinates[2] = new
Coordinate(6.0d,6.0d);
        linestringCoordinates[3] = new
```

```
Coordinate(7.0d,6.0d);
        linestringCoordinates[4] = new
Coordinate(7.0d,7.0d);
        linestringCoordinates[5] = new
Coordinate(8.0d,7.0d);
        linestringCoordinates[6] = new
Coordinate(8.0d,8.0d);
        LineString line =
geomFac.createLineString(linestringCoordinat
es);

        types[0] =
AttributeTypeFactory.newAttributeType("cente
rline", line.getClass());
        types[1] =
AttributeTypeFactory.newAttributeType("name"
, String.class);
        FeatureType lineType =
FeatureTypeFactory.newFeatureType(types,
"linefeature");
        Feature lineFeature =
lineType.create(new Object[]{line,
"centerline"});

        //simple polygon
        Coordinate[] polygonCoordinates =
new Coordinate[10];
        polygonCoordinates[0] = new
Coordinate(7,7);
        polygonCoordinates[1] = new
Coordinate(6,9);
```

```
        polygonCoordinates[2] = new
Coordinate(6,11);
        polygonCoordinates[3] = new
Coordinate(7,12);
        polygonCoordinates[4] = new
Coordinate(9,11);
        polygonCoordinates[5] = new
Coordinate(11,12);
        polygonCoordinates[6] = new
Coordinate(13,11);
        polygonCoordinates[7] = new
Coordinate(13,9);
        polygonCoordinates[8] = new
Coordinate(11,7);
        polygonCoordinates[9] = new
Coordinate(7,7);
        LinearRing ring =
geomFac.createLinearRing(polygonCoordinates)
;
        Polygon polygon =
geomFac.createPolygon(ring,null);

        types[0] =
AttributeTypeFactory.newAttributeType("edge"
, polygon.getClass());
        types[1] =
AttributeTypeFactory.newAttributeType("name"
, String.class);
        FeatureType polygonType =
FeatureTypeFactory.newFeatureType(types,"pol
ygonfeature");
```

```
        Feature polygonFeature =
polygonType.create(new Object[]{polygon,
"edge"});

        //simple point
        Coordinate c = new
Coordinate(14.0d,14.0d);
        Point point =
geomFac.createPoint(c);
        types[0] =
AttributeTypeFactory.newAttributeType("centr
e", point.getClass());
        types[1] =
AttributeTypeFactory.newAttributeType("name"
, String.class);
        FeatureType pointType =
FeatureTypeFactory.newFeatureType(types,"poi
ntfeature");

        Feature pointFeature =
pointType.create(new Object[]{point,
"centre"});

        //simple linear ring
        LinearRing ring2 =
geomFac.createLinearRing(polygonCoordinates)
;
        Polygon polygon2 =
geomFac.createPolygon(ring2,null);
        // build Shifted Geometry
```

```
        Polygon clone = (Polygon)
polygon2.clone();
        Coordinate[] coords =
clone.getCoordinates();
        for(int i = 0; i < coords.length;
i++) {
            Coordinate coord = coords[i];
            coord.x += 0;
            coord.y += 100;
        }
        Polygon polyg = clone;
        LinearRing ring3 = (LinearRing)
polyg.getExteriorRing();
        types[0] =
AttributeTypeFactory.newAttributeType("cente
rline", line.getClass());
        types[1] =
AttributeTypeFactory.newAttributeType("name"
, String.class);
        FeatureType lrType =
FeatureTypeFactory.newFeatureType(types,"rin
gfeature");
        Feature ringFeature =
lrType.create(new Object[]{ring3,
"centerline"});

        //memory store
        MemoryDataStore data = new
MemoryDataStore();
        data.addFeature(lineFeature);
        data.addFeature(polygonFeature);
```

```
        data.addFeature(pointFeature);
        data.addFeature(ringFeature);

        String typeName =
data.getTypeNames()[0];
     FeatureCollection ft =
data.getFeatureSource( typeName
).getFeatures().collection();

//Create style
     StyleFactory sFac =
StyleFactory.createStyleFactory();
        //The following is complex, and
should be built from
        //an SLD document and not by hand
        PointSymbolizer pointsym =
sFac.createPointSymbolizer();

pointsym.setGraphic(sFac.getDefaultGraphic()
);

        LineSymbolizer linesym =
sFac.createLineSymbolizer();
        Stroke myStroke =
sFac.getDefaultStroke();
        FilterFactory filterFactory =
FilterFactory.createFilterFactory();

myStroke.setColor(filterFactory.createLitera
lExpression("#0000ff"));
```

```
myStroke.setWidth(filterFactory.createLitera
lExpression(new Integer(5)));
        linesym.setStroke(myStroke);

        PolygonSymbolizer polysym =
sFac.createPolygonSymbolizer();
        Fill myFill = sFac.getDefaultFill();

myFill.setColor(filterFactory.createLiteralE
xpression("#ff0000"));
        polysym.setFill(myFill);

polysym.setStroke(sFac.getDefaultStroke());
        Rule rule = sFac.createRule();
        rule.setSymbolizers(new
Symbolizer[]{polysym});
        FeatureTypeStyle fts =
sFac.createFeatureTypeStyle(new
Rule[]{rule});

fts.setFeatureTypeName("polygonfeature");

        Rule rule2 = sFac.createRule();
        rule2.setSymbolizers(new
Symbolizer[]{linesym});
        FeatureTypeStyle fts2 =
sFac.createFeatureTypeStyle();
        fts2.setRules(new Rule[]{rule2});

fts2.setFeatureTypeName("linefeature");
```

```
        Rule rule3 = sFac.createRule();
        rule3.setSymbolizers(new
Symbolizer[]{pointsym});
        FeatureTypeStyle fts3 =
sFac.createFeatureTypeStyle();
        fts3.setRules(new Rule[]{rule3});

fts3.setFeatureTypeName("pointfeature");

        Rule rule4 = sFac.createRule();
        rule4.setSymbolizers(new
Symbolizer[]{linesym});
        FeatureTypeStyle fts4 =
sFac.createFeatureTypeStyle();
        fts4.setRules(new Rule[]{rule4});

fts4.setFeatureTypeName("ringFeature");

        Style style = sFac.createStyle();
        style.setFeatureTypeStyles(new
FeatureTypeStyle[]{fts, fts2, fts3, fts4});

 MapContext map = new DefaultMapContext();
        map.addLayer(ft,style);

 //Showing render
        Frame frame = new Frame();
        frame.addWindowListener(new
WindowAdapter() {
            public void
windowClosing(WindowEvent e)
```

```
{e.getWindow().dispose(); }
        });
        Panel p = new Panel();
        frame.add(p);
        frame.setSize(300,300);
        frame.setVisible(true);

        LiteRenderer renderer = new
LiteRenderer(map);
        renderer.paint((Graphics2D)
p.getGraphics(),p.getBounds(), new
AffineTransform());
        int w = 300, h = 600;
        BufferedImage image = new
BufferedImage(w,h,BufferedImage.TYPE_INT_RGB
);
        Graphics g = image.getGraphics();
        g.setColor(Color.white);
        g.fillRect(0,0,w,h);
        renderer.paint((Graphics2D) g,new
Rectangle(0,0,w,h), new AffineTransform());
```

```
Thread.sleep(1000);
frame.dispose();
```