Using Library

Table of contents

Using Library

Overview

You can use Kabeja as library in your application if you need to parse DXF and generate SVG output.

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.xml.sax.ContentHandler;

import org.kabeja.dxf.DXFDocument;
import org.kabeja.parser.DXFParseException;
import org.kabeja.parser.Parser;
import org.kabeja.parser.ParserBuilder;
import org.kabeja.svg.SVGGenerator;
import org.kabeja.xml.SAXGenerator;
 public class MyClass{

  public MyClass(){
          ...
  }

  public void parseFile(String sourceFile) {

    Parser parser = ParserBuilder.createDefaultParser();

    try {
        parser.parse(new FileInputStream(sourceFile));

        DXFDocument doc = parser.getDocument();

        //the SVG will be emitted as SAX-Events
        //see org.xml.sax.ContentHandler for more information

        ContentHandler myhandler = new ContentHandlerImpl();

        //the output - create first a SAXGenerator (SVG here)
        SAXGenerator generator = new SVGGenerator();

        //setup properties
        generator.setProperties(new HashMap());

        //start the output
        generator.generate(doc,myhandler);


       } catch (DXFParseException e) {
           e.printStackTrace();
       } catch (IOException ioe) {
           ioe.printStackTrace();
       }
  }
}
```

Get data from the DXFDocument

The main goal is the conversion to SVG, but you can query most data from the DXFDocument and work with the data inside your application.

The following example shows how to extract a layer and polyline of a draft.

Using Library

```
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.kabeja.dxf.DXFConstants;
import org.kabeja.dxf.DXFDocument;
import org.kabeja.dxf.DXFLayer;
import org.kabeja.dxf.DXFLine;
import org.kabeja.dxf.DXFPolyline;
import org.kabeja.dxf.DXFVertex;
import org.kabeja.dxf.DXFConstants;
import org.kabeja.dxf.helpers.Point;
import org.kabeja.parser.DXFParseException;
import org.kabeja.parser.Parser;
import org.kabeja.parser.DXFParser;
import org.kabeja.parser.ParserBuilder;

public class DXFExtractor{

 public void  read(InputStream in, String layerid) {

  Parser parser = ParserBuilder.createDefaultParser();
  try {

      //parse
      parser.parse(in, DXFParser.DEFAULT_ENCODING);

      //get the documnet and the layer
      DXFDocument doc = parser.getDocument();
      DXFLayer layer = doc.getDXFLayer(layerid);

      //get all polylines from the layer
      List plines = layer.getDXFEntities(DXFConstants.ENTITY_TYPE_POLYLINE);

      //work with the first polyline
      doSomething((DXFPolyline) plines.get(0));

     } catch (DXFParseException e) {
       e.printStackTrace();
     }
 }

 public void doSomething(DXFPolyline pline) {

   //iterate over all vertex of the polyline
   for (int i = 0; i < pline.getVertexCount(); i++) {

    DXFVertex vertex = pline.getVertex(i);

    //do something like collect the data and
    //build a mesh for a FEM system
   }
 }
}
```