User Guide

Table of contents

User Guide

User Guide

System Requirements

requires the following software to be already installed in your system:

- 1.3 or later.

Install Binary

- Download a binary-distribution
- Unzip
- Change to the -folder
- double click
- use (for memory use java -Xmx512m -jar launcher.jar)

You can find DXF samples in the -folder.

Building from Source

For building Kabeja you need Ant 1.6 (or later). For the build use the following steps:

- Download a source-distribution
- Unzip
- If you want to build the Cocoon-block, copy 'blocks.properties' to 'local.blocks.properties' and edit the path to the Cocoon-libraries.
- ant

User Interface

With the Kabeja GUI you can select files (or Drag and Drop to the main view) for processing and start different procssing pipelines. You can load your own processing configuration or use the default processing configuration.

The SVG blocks provides a DXF2SVGViewer for viewing the result of the converted SVG. Please note this is not a direct DXFViewer and will consume a lot of resources. With the JSScriptShell you can manipulate the parsed draft at runtime. The draft is available as variable 'dxf". If you use the reload button the changes will visible in other views of the Kabeja GUI (DXFSVGViewer).

You can switch off the different views for processing if you have problems with memory or don't need the views.

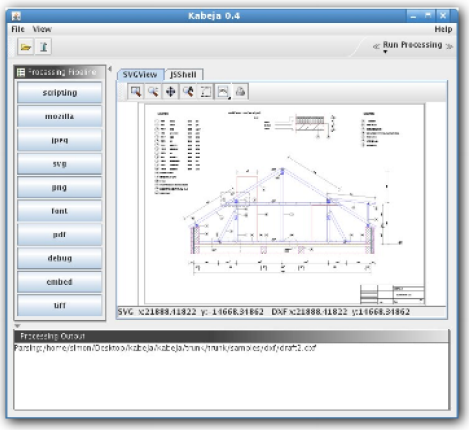
Starting:

double click the Kabeja.exe in the folder

In the of the 'kabeja'-folder try:

```
java -jar launcher.jar
```

```
java -Xmx512m -jar launcher.jar
```



Commandline Interface (CLI)

In the 'Kabeja'-folder you can start the cli processing with:

```
java -jar launcher.jar -nogui -pipeline svg myfile.dxf result.svg
```

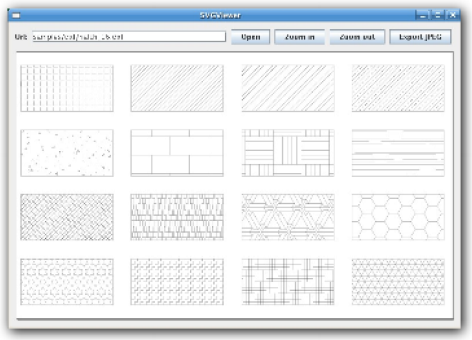
Directory-Mode: Converts all dxf files from a directory to svg. You can replace here the svg with other pipelines like pdf/jpeg/tiff.

```
java -jar launcher.jar -nogui -pipeline svg    C:\mydir
```

Show help:

```
java -jar launcher.jar  --help
```

Viewer



There is a simple Viewer available,in the 'Kabeja/lib'-folder. You can open DXF or SVG files. Try:

```
java -jar kabeja-svg-x.x.jar
```

Note: The Viewer converts first the DXF to SVG and renders then the SVG document (using Batik). So this can take time.

The viewer also renders SVG files.

Processing Overview

Kabeja provides a small processing system, where you can setup the different steps of the DXF conversion. The processing system uses a pipeline concept and is borrow from the XML web development framework Cocoon. A pipeline is splitted into "post processing" -> "XML-SAX generation" -> "XML SAX Filtering" -> "XML SAX Serialization". You are able to setup different pipelines for you need, e.g. one pipeline for DXF->SVG->PDF or a second for DXF->SVG->XSLT->JPEG.

will parse your CAD data. At the moment the Kabeja project provides only a DXF parser.

a PostProcessor will work direct with the parsed CAD data and can modify,delete or add data.

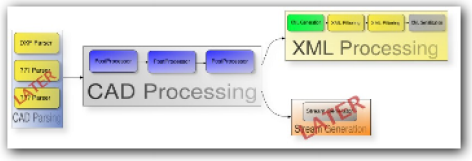
will generate an XML Stream (SAX events here) from the CAD data. The Kabeja project provides at the moment only an SVG generator. The dxf2calc project provides an statistics generator, which collects the bounds and length information of all containing geometries.

will filter or transform the XML data of the pipeline. We have here a XSLTFilter, which will apply your xslt-stylesheets. For embeding images into the SVG we also provide a special filter.

will output the XML format to any other format. The Kabeja project provides XML serializer, which will simple output the XML. By using the Batik project we can provide SVG2PDF, SVG2PNG, SVG2JPEG, SVG2TIFF and more serializers.

will direct output the parsed CAD data to any format. At the moment the Kabeja project does not provide a Stream generator. In one of the next releases we will add an Hatchpattern extractor, for extraction the used hatch pattern of a CAD drawing.

The following figure tries to give you an overview of the processing system (as PDF).



Processing Usage

The processing system will setup in a XML file. Take a look at "conf/process.xml". To use the processing system go into the "Kabeja"-folder and type:

```
java -jar launcher.jar  -nogui -pp conf/process.xml   -pipeline svg  my.dxf my.svg
```

to invoke the pipeline "svg" ("-pipeline" switch) from the "conf/process.xml" ("-pp" switch) configuration file.

You can also process a complete directory, e.g. you have a pipeline for PDF-generation then use:

```
java -jar launcher.jar -nogui -pp conf/process.xml   -pipeline pdf   C:\my_dxf_folder
```

to convert the complete directory into pdf files.

Example pipelines

The following table explains the example processing, which comes with the Kabeja release.

Pipeline name	Comment
svg	Converts the file/directory to SVG
jpeg	Converts the file/directory to SVG -> JPEG
png	Converts the file/directory to SVG -> PNG

tiff	Converts the file/directory to SVG -> TIFF
pdf	Converts the file/directory to SVG -> PDF
debug	Converts the file/directory to SVG and mark the entities and bounds, which enlarge the draft.
mozilla	Converts the file/directory to SVG and apply a filter, which set fixed values for SVG-attribute stroke-width and font-size, otherwise Mozilla/Gecko based have problems to show the SVG.
embed	Embeds images and fonts (if they are setup) into the SVG-output.

Browser SVG Support

More and more browser comes with native SVG support.

Browser Support Table

Browser	Native SVG	Comment
Firefox 1.5.x	yes	Crashes with default Kabeja-output, use the processing system with the pipeline "mozilla", this will generate a usable SVG output.
Firefox 2.x	yes	works out-of-box
Internet Explorer 5/6/7	no	You have to use a plugin.
Konquerer 3.5.x	yes	works out-of-box
Opera 9.0	yes	works out-of-box

OpenOffice dxf2calc

The dxf2calc project uses the DXF parsing capabilities of kabeja to provide access to the geometry data of DXF drafts for OpenOffice Calc 2.0

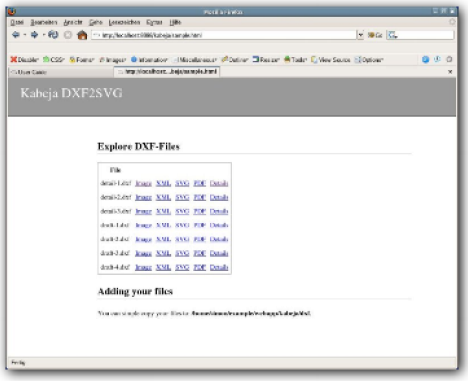
This project was initiated and is supported by Lars Brandi Jensen.

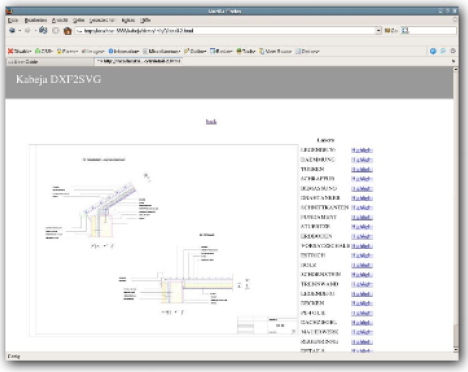
Visit the dxf2calc project site to find out more.

Cocoon Integration Overview

You can use Kabeja in the Web Development Framework Cocoon 2.x.

Example Cocoon Webapplication





The complete example is included in the source-dist (copy the folder blocks/cocoon/subsitemap-demo/kabeja to your cocoon webapp).

Cocoon Installation

- Copy the 'kabeja.jar' and 'kabeja-dxf2svg-cocoon.jar' to your WEB-INF/lib-folder of your Cocoon-Webapplication.
- Setup Kabeja as Generator in your sitemap.xmap or subsitemap.xmap (see below)
- Restart Cocoon

Cocoon Configuration

```
snippet:
--------

<map:components>
    .....
 <map:generators default="file">

  <map:generator name="dxf2svg" src="org.kabeja.cocoon.generation.DXF2SVGGenerator"/>

 </map:generators>


....
 <map:pipelines>

  <map:pipeline>

    <map:match pattern="dxf/*.svg">
      <map:generate type="dxf2svg" src="dxf/{1}.dxf"/>
        ...
      <!-- transform things you need -->
      <map:transform src="my.xsl"/>

        <!-- serialize (like svg2png,svg2jpeg) -->
      <map:serialize type="xml"/>
    </map:match>

   </map:pipeline>

  ....


  </map:pipelines>
```

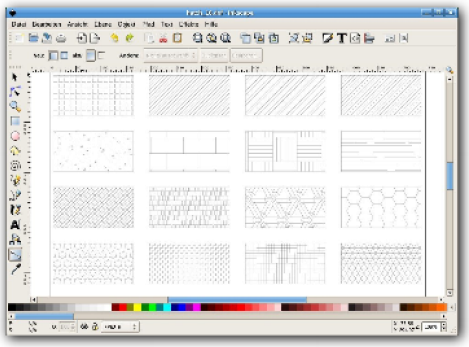
Notes

DXF-drafts may often large, so the SVGDocument will consume a lot of memory. The Generator is Cacheable so the first run will take more time.

Inkscape DXF Importfilter

You can use Kabeja as importfilter for the vector drawing program Inkscape. The Kabeja importfilter will coexist with an existing dxf2svg importfilter.

The Kabeja extension will be accessable as "Kabeja-AutoCAD DXF (*.dxf)" import type selection in the "File-> Open" dialog.



Installation Java extension (the default)

Copy the content of the "Kabeja/inkscape-extension" into "Extensions" folder of Inkscape (e.g. for Unix "$HOME/.inkscape/extensions" and Windows "C:\Program Files\Inkscape\extensions").

Installation native extension

The extension will be installed in the extensions folder of Inkscape (e.g. for Linux "$HOME/.inkscape/extensions").

- Download the Kabeja source-distribution
- Unpack and change into the directory
- Copy "blocks.properties" file to "local.blocks.properties" and change the line "inkscape.importfilter.type=java" to "inkscape.importfilter.type=native" there.
- Build and install by:

```
ant block -Dblock.name=inkscape -Dblock.target=install
```

Installation .NET extension

The .NET extension is available as separate package. The package contains all necessary libraries to run out-of-box on Mono and Windows .NET.

- Download the .NET Kabeja Inkscape Extension
- Unpack and copy all files to the Inkscape extension directory (e.g. for Linux "$HOME/.inkscape/extensions", Windows "C:\Programes\Inkscape\share\extensions")..

Fonts Overview

Kabeja does not support the shx fontfiles. To use shx fonts you could convert these fonts to SVG fonts. Then you can

setup Kabeja to use these fonts instead of the shx fonts.

If you use AutoCAD from Autodesk it is quite easy to setup the fonts. On the install-cd of AutoCAD you will find all shx fonts as TTF fonts and these fonts can be converted to SVG fonts by using the ttf2svg converter of the Batik-project.

There are some programs, which can convert shx fonts to ttf (google: shx2ttf). So you can use other fonts too.

Font Setup

You must have all TTF fonts inside a directory, be sure that the filenames of the shx and ttf files are the same. Go to the Kabje root directory and use the converting tool from ant:

```
ant -f font.xml -Dfont.source=C:\path\to\the\ttffonts
```

or with without ant. Go to the Kabeja directory, create a directory "fonts" and use:

```
java -cp kabeja-svgview.jar;lib\batik.jar org.kabeja.batik.tools.FontImport C:\path\to\ttffont    fonts  conf\font.properties
```

All TTF fonts will now converted and stored to "Kabeja\fonts" directory of your Kabeja installation. The converter will create a mapping file "conf/fonts.properties". This mapping file is important, Kabeja will lookup here. So you need the referenced fonts of your DXF drawing as SVG-font and setup inside the mapping file, otherwise Kabeja ignores the font.

Embedding Font

All fonts are referenced to fonts files of the "fonts" directory, but you can embed the SVG fonts inside your SVG file. See the Processing page (pipeline embed).

Default Font

If no font is set by the drawing, then Kabeja will ignore the default font. This can be done by a XSL-Stylesheet.

Using Kabeja API

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

```
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.

```
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
  }
 }
}
```

Develop Kabeja Components

Kabeja is splitt into different types of components. Examples for components are the PostProcessors, SAXGenerators, SAXFilters, SAXSerializers. This separation should help you to add more functionality to Kabeja.

The Kabeja project started as dxf2svg conversation tool, this explains the used name "DXFDocument" below. In a later release we will change this to a more generic name. We adapeted the pipeline processing from the Web Development Framework Cocoon. So you will find similarities here.

Parser

A Parser will create the data (DXFDocument) for the next step of processing. The Kabeja project provides at the moment only a DXF parser.

See Parser

PostProcessor

A PostProcessor will process on the (DXFDocument) and can change, add or remove data from the CAD data.

See PostProcessor.

SAXGenerator

A SAXGenerator converts the given CAD data (DXFDocument) to XML (SAX events).

See SAXGenerator.

SAXFilter

A SAXFilter will work on the XML SAX events and can change,add or remove XML SAX events.

See SAXFilter.

SAXSerializer

A SAXSerializer will output the XML SAX events to a given stream.

See SAXSerializer.

StreamGenerator

A StreamGenerator will output the CAD data (DXFDocument) direct to a given stream.

At the moment this kind of component is not supported in the processing system. This will be added in the next release.

See StreamGenerator.

Component Configuration

You can register your component in the processing configuration file ("conf/process.xml"). Add your component in the right section and put your component.jar into the "Kabeja/lib" directory of kabeja. Then you only need a processing pipeline, where you use the component.

```
<?xml version="1.0" encoding="UTF-8"?>
<processing xmlns="http://kabeja.org/processing/1.0"
            xmlns:xi="http://www.w3.org/2001/XInclude">
 <!--+
     | The processing configuration
     +-->
  <configuration>
  <!--+
      |not yet implemented, but will follow: the parser configuration
      |<xi:include href="parser.xml"/>
      +-->

   <postprocessors>
    <postprocessor class="org.kabeja.processing.BoundsDebugger" name="bounds.debugger"/>
    ...
   </postprocessors>

   <filters>
    <filter class="org.kabeja.batik.tools.ImageBase64Encoder" name="image"/>
    ...
```

```
</filters>

<serializers>
 <serializer class="org.kabeja.xml.SAXPrettyOutputter" name="svg"/>
 ...
</serializers>

<generators>
 <generator class="org.kabeja.svg.SVGGenerator" name="svg"/>
</generators>
```

Problem: Coordinate range - Java VM Crash

DXF uses double-precision floating point 64bit as coordinate range and SVG uses single-precision floating point (32bit). The Kabeja SVGGenerator lets the coordinates untouched, so you can work with the original coordinates from the CAD draft. But if you render a SVG with a larger coordinate range as 32bit on java based renderer (like batik) you can run in trouble. This happens on curved pathes with a dash pattern and will result in a complete VM Crash. Other SVG viewer/renderer like Firefox/Inkscape/Adobe have no problems.

We will provide a solution for this problem later, but you can use the following workarounds.

- use other SVG to image renderer
- switch of the line type parsing

You can check the Crash.svg whith the DXF/SVG viewer of Kabeja and see how your Java VM will handle this.

Problem:Unsupported Entities

The SVGGenerator of Kabeja does not suppoert all entities (geometries) yet, but if you can export your draft to DXF version 12.0 you may have more luck.

State of DXF ENTITES

The following tables gives an overview of the DXF entities state, the state of the other DXF Sections (Tables,Blocks,Classes,Objects) is not shown here.

Note that the SVG output only work on 2D drafts.

DXF R12

DXF entity	Parsed	SVG	Comment
3DFACE	yes	yes	
ARC	yes	yes	
ATTRIB	yes	yes	
CIRCLE	yes	yes	

DIMENSION	yes	yes	SVG output only if there is a complete DIMENSION description (DIMENSION block)
INSERT	yes	yes	
LINE	yes	yes	
POINT	yes	yes	
POLYLINE	yes	yes	PolyMeshes and Splines are supported too.
SECTION	no	no	
SHAPE	yes	no	
SOLID	yes	yes	
TEXT	yes	yes	
TRACE	yes	yes	
VIEWPORT	yes	no	

DXF R13

DXF entity	Parsed	SVG	Comment
3DSOLID	yes	no	ACIS data
BODY	yes	no	ACIS data
ELLIPSE	yes	yes	Elliptical arcs are supported too.
LEADER	yes	yes	
MLINE	yes	yes	since 0.4
MTEXT	yes	yes	
OLEFRAME	no	never	
RAY	yes	yes	
REGION	yes	no	ACIS data
SPLINE	yes	yes	since 0.4
TOLERANCE	yes	no	
XLINE	yes	yes	

DXF R14

DXF entity	Parsed	SVG	Comment
IMAGE	yes	yes	clipping support will follow
HATCH	yes	yes	since 0.3
LWPOLYLINE	yes	yes	
OLE2FRAME	no	never	

DXF 2000

DXF entity	Parsed	SVG	Comment
RTEXT	no	no	
WIPEOUT	no	no	later

DXF 2002

RTEXT removed from DXF

DXF 2004

no new entities

DXF 2005

DXF entity	Parsed	SVG	Comment
TABLE	no	no	

DXF 2006

no new entities

DXF 2007

DXF entity	Parsed	SVG	Comment
HELIX	no	no	
LIGHT	no	no	
SUN	no	no	

SURFACE	no	no	

Kabeja-License

Kabeja is lisenced under the Apache Software License 2.0

and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.