Installation

IRun-J is a java application. In order to run IRun-J you need to have at least J2RE1.4 (Java2 runtime) installed on your system. Extract the irunjxxx.zip file in to an empty folder and set JAVA_HOME variable in the irun.bat (irun.sh in linux) to point to the correct java home before running IRun-J.

The following folders will be found under the IRun root:
/lib: contains necessary jar files, and irun.jar itself
/src: java sources
/doc: javadoc files

Running

In order to run IRun-J converter from a user interface, just execute the *irun.bat* (or *irun.sh* in linux). If the JAVA_HOME is set correctly then converter UI should show up.

You can select a source file by typing the path, or by selecting from the file system. You have to give the destination location and name also.

The default conversion type is "HTML with CSS", you can choose other options (XML and plain HTML) from the "type" part.

There are some options to choose when especially converting to HTML (with or without CSS). Here are the meanings of conversion options:

Add results : Fields, symbols and some other RTF special tags have "result"s, which are ordinary RTF texts. Instead of trying to evaluate the field, IRun reads results. Otherwise results are discarded, and since most of the time evaluation of fields is not supported, the fields are completelely discarded. It is suggested to choose this option.

No header : You may want to add your headers to the converted output, or insert output to another document. At those times you may choose this option; IRun will only add the tags between

elements discarding the header tags.

Convert WMF to PNG : If this option is selected, IRun will convert WMF and EMF pictures (and also objects with WMF and EMF images) to PNG format. (Not yet implemented)

Convert WMF to SVG : If selected IRun will convert WMF and EMF pictures (and also objects with WMF and EMF images) to SVG format. SVG is a vector graphic format that requires a SVG enabled browser to show.

Embed base64 images : This option makes IRun embed images rather than create separate files for images. Generated output is a single HTML or XML. The images are encoded in base64 binary encoding style. Not all browsers support base64 encoded images.

Tag section boundaries : This option tells IRun to put tags at section boundaries. is not a valid tag. You may use it just for the sake of information.

Display header/footer content : This option tells IRun to use and convert header/footer data. H/F is one of many differences between a web page and a document. The default option is to discard H/F data.

After selecting the options, and determining source and destination files, just click the "Convert" button. This will create the destination file (or overwrite it if already exists), if everything works fine. You may use your browser to see the created file.

Using as a Component

In order to convert RTF from buffer to buffer, write batch conversion scripts or add conversion functionality to server side components you may use the conversion API.

com.pilot.irun.RtfConverter is the only necessary class to import and use. *RtfConverter* has different convert functions for file-to-file and buffer-to-buffer conversion. One has to set the options prior to calling the convert methods.

For restricting and/or allowing debug-error log you can set two static fields in *com.pilot.irun.Util* class

Sample Code

```
//Example1 : convert and RTF file to HTML with CSS.
import com.pilot.irun.RtfConverter;
import com.pilot.irun.Util;

public class Test1
{
 public static void main(String[] args)
 {
  Util.debug=false;
  Util.error=true;
  RtfConverter conv=new RtfConverter();

  conv.setBgcolor("ffffff");
  conv.setDislayHeaderFooterData(false);
  conv.setEncodeBase64(true);
  conv.setNoHeader(false);
  conv.setTagSections(true);
  conv.setTitle("Welcome to the world of IRun-J");
  conv.setWmfAction(RtfConverter.CONVERT_TO_SVG);
  conv.convert(RtfConverter.HTML_WITH_CSS, "c:\\temp\\Sample.rtf", "c:\\temp\\Sample.htm

 }
}


//Example2 : convert and RTF file to HTML with CSS + no base64 encoding
import com.pilot.irun.RtfConverter;
import com.pilot.irun.Util;

public class Test2
{
 public static void main(String[] args)
 {
  Util.debug=false;
  Util.error=true;
  RtfConverter conv=new RtfConverter();

  conv.setBgcolor("ff00ff");
  conv.setDislayHeaderFooterData(false);
  conv.setEncodeBase64(false);
  conv.setNoHeader(false);
```

```
  conv.setTagSections(true);
  conv.setTitle("Welcome to the world of IRun-J");
  conv.setWmfAction(RtfConverter.CONVERT_TO_SVG);
  conv.convert(RtfConverter.HTML_WITH_CSS, "c:\\temp\\Sample.rtf", "c:\\temp\\Sample.htm

 }
}

//Example3 : convert and RTF file to XML
import com.pilot.irun.RtfConverter;
import com.pilot.irun.Util;

public class Test3
{
 public static void main(String[] args)
 {
  Util.debug=true;
  Util.error=true;
  RtfConverter conv=new RtfConverter();

  conv.setDislayHeaderFooterData(false);
  conv.setEncodeBase64(true);
  conv.convert(RtfConverter.XML, "c:\\temp\\Sample.rtf", "c:\\temp\\Sample.xml");
 }
}
```

WMF/EMF Sample

IRun-J can also parse, render and convert WMF and EMF metafiles.This functionality can be accessed seperately through the *com.pilot.irun.wmf.SvgConverter* class.

Here is a WMF/EMF to SVG conversion example:

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import com.pilot.irun.wmf.Metafile;
import com.pilot.irun.wmf.SvgConverter;
import com.pilot.irun.wmf.WMFParser;

public class TestWmf
{
 public static void main(String[] args)
 {
  Metafile mf=null;
  try
  {
   FileInputStream fis=new FileInputStream("c:\\temp\\wmftest\\graph.wmf");
   FileOutputStream fout=new FileOutputStream("c:\\temp\\wmftest\\graph.svg");
   WMFParser parser=new WMFParser();
   mf=parser.parse(fis);
   SvgConverter converter=new SvgConverter();
   converter.convert(mf, fout);
   fout.flush();
   fout.close();
  }
  catch (FileNotFoundException e)
  catch (IOException e)
   e.printStackTrace();
```

```
 }
}
```

Custom Parser

One can also write a custom RTF converter by implementing the *com.pilot.irun.RtfListener* interface. You can register your listener to the *com.pilot.irun.RtfParser* , which is responsible for parsing the RTF and informing listeners.
RtfListener interface:

```
public interface RtfListener
{
 public void handleDocumentStart(RtfInfo info);
 public void handleParagraph(RtfInfo info,Par par);
 public void handleRow(RtfInfo info,Row row);
 public void handleSectionStart(RtfInfo info,Sect sect);
 public void handleSectionEnd(RtfInfo info,Sect sect);
 public void handleDestinationStart(RtfInfo info,String dest);
 public void handleDestinationEnd(RtfInfo info,String dest);
 public void handleDocumentEnd(RtfInfo info,Doc doc);
}
```

RtfInfo, *Par*, *Sect*, *Doc* are RTF structures (ie. simple java objects) containing RTF properties and data. Suppose *TestListener* is an *RtfListener* implementation:

```
package com.pilot.irun.test;
import com.pilot.irun.*;
import java.io.*;

public class TestReader
{
 public static void main(String[] args)
 {
  RtfParser  reader=new RtfParser();
  FileInputStream fis=null;
  ByteArrayOutputStream out=new ByteArrayOutputStream();
  try
  {
   fis=new FileInputStream("c:\\temp\\Sample.rtf");
   fous=new FileOutputStream("c:\\temp\\Sample.html");
   reader.registerListener(new TestListener());
   reader.read(fis,true);
  }
  catch(Exception e)
   System.out.println("TestReader::main:exception:"+e);
 }
}
```

Last modified on 2013-05-29 by admin