Development-Thumbnailing

Overview

Many people have asked how to create thumbnails of images using WebObjects. There is nothing specific to WebObjects about this problem, and there are quite a few options for solving it:

- On Mac OS X, you can use Runtime.exec(..) and call the commandline program "sips," which uses the ImageIO and CoreImage frameworks
- On Mac OS X, you can use Brendan Duddridge's ImageIO JNI Wrapper
- An all platforms, you can use Java2D with Java's ImageIO and BufferedImage
- On many platforms, you can run ImageMagick and use Runtime.exec to call the commandline "convert"
- On many platforms, you can build and run JMagick, a Java JNI wrapper around ImageMagick

ImageMagick

A utility class to call ImageMagick binaries as an external process with Runtime.exec, to either discover height/width or resize an image. It has a few default filepaths that correspond to what I need on my systems, you probably want to change them for your system(s).

Anjo Krank: I may be wrong, but I'm pretty sure that this code won't work, at least not reliably. There is no guarantee that the the contents of the supplied arrays are filled before the process exits. I had a lot of null-results when I tried this in Wonder. The only safe way I've seen so far is to actually wait until the stream is truly finished reading before accessing the result. And this can only be done by waiting on a sema. Take a look at ERXRuntimeUtilities for a version that does work.

```
/* ImageMagickUtil.java created by jrochkind on Thu 24-Apr-2003 */

import com.webobjects.foundation.*;
import com.webobjects.eocontrol.*;
import com.webobjects.eoaccess.*;
import com.webobjects.appserver.*;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.BufferedReader;

/* Utility methods that deal with images by calling the ImageMagick
software
as an external process */

/* Dealing with Runtime.exec in a thread-safe way is tricky! Sorry for
that.
I used the article at
http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html
as a guide for understanding how to do it right. */

public class ImageMagickUtil {
  //protected static final String imUtilLocationPath = "C:\\Program
Files\\ImageMagick-5.5.6-Q8\\";
  protected static final String imUtilLocationPath;// =
"/export/home/jar247/mtBin/";
  //The image util location path can be supplied as a Java property,
  //or we can try to guess it from the OS.
  static {
```

```
        String locProp = System.getProperty("imUtilPath");
        String osName = System.getProperty("os.name");
        if ( locProp != null ) {
            imUtilLocationPath = locProp;
        }
        else if ( osName.indexOf("Windows") != -1 ) {
            imUtilLocationPath = "C:\\Program Files\\ImageMagick-5.5.6-Q8\\";
        }
        else {
            //Assume our deployment machine, which is currently set up
            //to make this the location...
            imUtilLocationPath = "/export/home/webob/ImageMagick/bin/";
        }
    }

    protected static final String imIdentifyCommand = "identify";
    protected static final String imConvertCommand = "convert";

    public static ImageProperties getImageSize(String filePath) throws
IMException {
        return getImageSize( new java.io.File(filePath) );
    }

    public static ImageProperties getImageSize(java.io.File imageFile) throws
IMException {
        String filePathToImage = imageFile.getPath();

        String[] cmdArray = new String[] {
            imUtilLocationPath + imIdentifyCommand,
            "-format", "%w\n%h", // [width][newline][height]
            filePathToImage
        };

        NSMutableArray stdOutContents = new NSMutableArray();
        NSMutableArray stdErrContents = new NSMutableArray();
        int resultCode = -1;
        try {
            resultCode = exec(cmdArray, stdOutContents, stdErrContents);
        }
        catch (IOException ioE ) {
            //For some reason we couldn't exec the process! Convert it to an
IMException.
            //One reason this exception is thrown is if the path specified to
the im
            //executable isn't correct.
            throw new IMException("Could not exec imagemagick process: " +
ioE.getMessage(), cmdArray, null);
        }
        catch ( InterruptedException intE ) {
            //re-throw it as an IMException.
            //This exception should really never be thrown, as far as I know.
            throw new IMException("imagemagick process interrupted! " +
intE.getMessage(), cmdArray, null);
```

```
      }

      if ( resultCode != 0 ) {
          //The external process reports failure!
          IMException e = new IMException("Identify failed! ", cmdArray,
stdErrContents);
          e.exitValue = resultCode;
          throw e;
      }

      //Now we need to parse the result line for the height
      //and width information.
      if ( stdOutContents.count() >= 2 ) {
          //First line is width, second is height, because
          //we asked the imagemagick 'identify' utility to
          //output like that.
          String widthStr = (String) stdOutContents.objectAtIndex(0);
          String heightStr = (String) stdOutContents.objectAtIndex(1);
          Integer width = new Integer( widthStr );
          Integer height = new Integer( heightStr );

          ImageProperties p = new ImageProperties();
          p.width = width;
          p.height = height;
          return p;
      }
      else {
          //Umm? Error condition.
          throw new IMException("Unexpected output of imagemagick process",
cmdArray, stdErrContents);
      }
  }

  // An external image magick process will be run to resize the image. It's
reccomended you
  // check to make sure it's neccesary to resize the image first!
  // Beware, if the sizes you pass in are LARGER than the existing size,
the output image
  // WILL be BIGGER than the input---this isn't just for resizing downward.
  // Null outFilePath means to overwrite the source file path with the
resized image.
  // You can pass null for either maxWidth or maxHeight, but not both, that
would be silly!
  // [not implemented yet:] Returned is an object telling you the new
resized size of the output image.
  public static void resizeImage(String sourceFilePath, String outFilePath,
int maxWidth, int maxHeight)
  throws IMException {
      if ( outFilePath == null ) {
          //overwrite original file if neccesary
          outFilePath = sourceFilePath;
      }
      /*else if ( NSPathUtilities.pathExtension( outFilePath ) == null ) {
```

```
          //give the output file path the same extension as the in file
path.
          outFilePath = NSPathUtilities.stringByAppendingExtension(
outFilePath,

NSPathUtilities.pathExtension(sourceFilePath));
      }*/

      StringBuffer dimensionBuffer = new StringBuffer();
      if ( maxWidth != -1) {
          dimensionBuffer.append(maxWidth);
      }
      dimensionBuffer.append( "x" );
      if ( maxHeight != -1) {
          dimensionBuffer.append( maxHeight );
      }
      String dimensionDirective = dimensionBuffer.toString();

      //We include the ' +profile "*" ' argument to remove
      //all profiles from the output. Not sure exactly what this means...
      //but before we were doing this, we wound up with JPGs that
      //caused problems for IE, for reasons I do not understand.
      String[] cmdArray = new String[] {
          imUtilLocationPath + imConvertCommand,
          "-size", dimensionDirective,
          sourceFilePath,
          "-resize", dimensionDirective,
          "+profile", "*",
          outFilePath
      };

      NSMutableArray stdErrContents = new NSMutableArray();
      NSMutableArray stdOutContents = new NSMutableArray();
      int resultCode;
      try {
          resultCode = exec( cmdArray, stdOutContents, stdErrContents );
      }
      catch (IOException ioE ) {
          //For some reason we couldn't exec the process! Convert it to an
IMException.
          //One reason this exception is thrown is if the path specified to
the im
          //executable isn't correct.
          throw new IMException("Could not exec imagemagick process: " +
ioE.getMessage(), cmdArray, null);
      }
      catch ( InterruptedException intE ) {
          //re-throw it as an IMException.
          //This exception should really never be thrown, as far as I know.
          throw new IMException("imagemagick process interrupted! " +
intE.getMessage(), cmdArray, null);
      }
      if ( resultCode != 0 ) {
```

```
            //The external process reports failure!
            IMException e = new IMException("Conversion failed! ", cmdArray,
stdErrContents);
            e.exitValue = resultCode;
            throw e;
        }
    }

    //Invokes the external process. Puts standard out and standard error into
the
    //arrays given in arguments (they can be null, in which case the err/out
stream
    //is just thrown out.
    //Throws IOException if the exec of the external process doesn't work,
for instance
    //because the path to the command is no good.
    //Throws the InterruptedException... not sure when, if ever. But it means
that the exec
    //didn't work completely.
    public static int exec(String[] cmdArray, NSMutableArray stdOut,
NSMutableArray stdErr) throws IOException, InterruptedException {
        Process process = Runtime.getRuntime().exec( cmdArray );

        //We are interested in what that process writes to standard
        //output and standard error.
        //Grab the contents of those in their own seperate threads!
        //To avoid deadlock on the external process thread!

        StreamGrabber errGrabber = new StreamGrabber(
process.getErrorStream(), stdErr );
        StreamGrabber outGrabber = new StreamGrabber(
process.getInputStream(), stdOut );

        errGrabber.start();
        outGrabber.start();

        return process.waitFor();
    }


    //Will launch a NEW THREAD and put the contents of the given stream into
    //the NSMutableArray. Don't be accessing the array in another thread
before
    //we're done!
    //If array is null, StreamGrabber reads the input stream to completion,
    //but doesn't store results anywhere.
    static class StreamGrabber extends Thread
    {
        InputStream inputStream;
        NSMutableArray array;
        boolean done = false;
        Exception exceptionEncountered;
```

```
      StreamGrabber(InputStream is, NSMutableArray a)
      {
          super("StreamGrabber");
          this.inputStream = is;
          this.array = a;
      }

      public void run()
      {
          try {
          InputStreamReader isr = new InputStreamReader(inputStream);
          BufferedReader br = new BufferedReader(isr);
          String line=null;
          while ( (line = br.readLine()) != null)
          {
              if ( array != null ) {
                  array.addObject(line);
              }
          }
          }
          catch ( java.io.IOException e) {
              //hmm, what should we do?!?
              setExceptionEncountered( e );
          }
          setDone( true );
      }

      public synchronized void setDone(boolean v) {
          done = v;
      }
      //Can be used by parent thread to see if we're done yet.
      public synchronized boolean done() {
          return done;
      }
      public synchronized Exception exceptionEncountered() {
          return exceptionEncountered;
      }
      public synchronized void setExceptionEncountered(Exception e) {
          exceptionEncountered = e;
      }
      public boolean didEncounterException() {
          return exceptionEncountered() != null;
      }
  }

  //Exception thrown by utilty methods when the call to an external image
magick
  //process failed.
  public static class IMException extends Exception {
      protected int exitValue;
      protected String processErrorMessage;
      protected String invocationLine;
```

```
      protected String message;

      public IMException() {
          super();
      }
      public IMException(String s) {
          super();
          message = s;
      }
      //Constructs a long message from all these parts
      public IMException(String messagePrefix, String[] cmdArray,
NSMutableArray stdErr) {
          super();
          if ( cmdArray != null ) {
              invocationLine = new NSArray( cmdArray
).componentsJoinedByString(" ");
          }
          if ( stdErr != null ) {
              processErrorMessage = stdErr.componentsJoinedByString("; ");
          }

          StringBuffer b = new StringBuffer();
          b.append( messagePrefix );
          b.append(". invocation line: ");
          b.append( invocationLine );
          b.append(". error output: " );
          b.append( processErrorMessage );
          message = b.toString();
      }

      //the return code from the image magick external invocation.
      //I think it's probably always 1 in an error condition, so not so
useful.
      public int exitValue() {
          return exitValue;
      }
      //The error message reported by image magick.
      public String processErrorMessage() {
          return processErrorMessage;
      }
      //The command line used to invoke the external im process that
      //resulted in an error.
      public String invocationLine() {
          return invocationLine;
      }
      public void setInvocationLine( String[] cmdArray ) {
          invocationLine = new NSArray(cmdArray).componentsJoinedByString("
");
      }
      //over-riding
      public String getMessage() {
          return message;
      }
```

```
}

//Object that encapsulates data returned by an image operation
public static class ImageProperties extends Object {
    protected Integer height;
    protected Integer width;

    public ImageProperties() {
        super();
    }
    public Integer height() {
        return height;
    }
    public Integer width() {
        return width;
    }
```

```
  }
}
```

JAI example

An example how to resize an image with Java Advanced Imaging (http://java.sun.com/products/java-media/jai/). The jar files jai-codec.jar and jac-core.jar are in the NEXT_ROOT/Library/Java/Extensions folder and referenced in the classpath. This example uses logging capability from project wonder.

```
/* ImageResizer.java */

import java.awt.image.renderable.ParameterBlock;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import javax.media.jai.InterpolationNearest;
import javax.media.jai.JAI;
import javax.media.jai.OpImage;
import javax.media.jai.RenderedOp;
import com.sun.media.jai.codec.ByteArraySeekableStream;
import com.webobjects.foundation.NSData;
import er.extensions.ERXLogger;

public class ImageResizer {

  private static final ERXLogger log =
ERXLogger.getERXLogger(ImageResizer.class);

  /**
   * utility function to resize an image to either maxWidth or maxHeight
with jai
   * example:  logo = new NSData(aFileContents1);
   *           logo = ImageResizer.resizeImage(logo, 128, 42);
   * @param data image content in NSData array
   * @param maxWidth maxWidth in pixels picture
   * @param maxHeight maxHeight in pixels of picutre
   * @return resized array in JPG format if succesfull, null otherwise
   */
  static public NSData resizeImage(NSData data, int maxWidth, int
maxHeight) {
      try {
          ByteArraySeekableStream s = new ByteArraySeekableStream(data
                  .bytes());

          RenderedOp objImage = JAI.create("stream", s);
          ((OpImage) objImage.getRendering()).setTileCache(null);

          if (objImage.getWidth() == 0 || objImage.getHeight() == 0) {
              log.error("graphic size is zero");
              return null;
          }
```

```
        float xScale = (float) (maxWidth * 1.0) / objImage.getWidth();
        float yScale = (float) (maxHeight * 1.0) / objImage.getHeight();
        float scale = xScale;
        if (xScale > yScale) {
            scale = yScale;
        }

        ParameterBlock pb = new ParameterBlock();
        pb.addSource(objImage); // The source image
        pb.add(scale); // The xScale
        pb.add(scale); // The yScale
        pb.add(0.0F); // The x translation
        pb.add(0.0F); // The y translation
        pb.add(new InterpolationNearest()); // The interpolation

        objImage = JAI.create("scale", pb, null);

        ByteArrayOutputStream out = new java.io.ByteArrayOutputStream();
        JAI.create("encode", objImage, out, "JPEG");
        return new NSData(out.toByteArray());
    } catch (IOException e) {
        log.error("io exception " + e);
    } catch (RuntimeException e) {
        log.error("runtime exception "+e);
    }

    return null;
```

```
    }
  }
```