Athens University of Economics and Business

JphotoAlbum

Maintenance and Refactoring of Information Systems

Date: 1. 6. 2005
Name: Zbyněk Mužík
E-mail: zbynek.muzik@email.cz

0 Index

1 Introduction

I chose to participate in this particular project, because I spent lot of time looking for a free program, which would be suitable for creating virtual photo albums with subtitled pictures and this albums would be independent on the original program. As I haven’t found any really suitable, I decided to add this function to some of the programs that export photo albums into HTML. I found some projects on SourceForge.net and finally decided for JPhotoAlbum, as it was mature enough and very easy to use.
The project is written in Java. During working on it I was using Java SDK 1.4.2., NetBeans IDE, some tools from Cygwin and occasionally Linux for testing.

2 Program overview

2.1 General

JPhotoAlbum is a program written in pure Java, that allows browsing, commenting and organizing digital photos into albums, which are supposed to be published as web pages. Albums are saved in a special data format based on XML. These files are processed by XSLT templates to become HTML pages.
The project is has been developed by two Finnish programmers Jari Karjala and Tarja Hakala since 2004. The current release is 1.3.0. The project homepage is jphotoalbum.jpkware.com, but it is also hosted on sourceforge.net. The 1.3.0 version is the first one to be distributed under GNU General Public License Version 2.
The project doesn’t have a lot of documentation and javac comments are very scarce.
Besides the main package fi.iki.jka, JPhotoAlbum, it uses also 3 other packages:

- ✓ xerces – a powerful XML parser, of the Apache project.
- ✓ metadata-extractor-2.1 – library used for extracting the meta information from digital photos.
- ✓ castor – another package used for processing XML files.

The main package consists of 17 classes which make altogether slightly more than 5000 code lines.

2.2 Functionality

Creating albums

Albums can be created from a directory or separate pictures can be selected, drag and drop function is supported, or the photos to create the album can be submitted on the command line e.g. "java -jar JPhotoAlbum.jar /photos/*.jpg". The order of the photos can be, of course, changed and subtitles can be added to each picture.
Various page properties can be edited using the Page menu. These refer to the XSLT page templates. The watermark is written to the lower left corner of web images, it is useful for embedding copyright to published images. Albums can be very simply linked together.

Exporting HTML

The HTML albums will contain images resized to fit into a 640 times 640 pixel square, and optionally thumbnails are also generated. The HTML file is generated to the same directory as the saved album file. The scaled images are saved to pictures and thumbs subdirectories. The original pictures are not affected.

Exporting Subtitled Photos

The Export Subtitled Photos option can be used to export photos so that they fit in a 640 times 480 pixel rectangle, even if they are in portrait form factor. This can be also done from command line, e.g. "java -jar JPhotoAlbum.jar export *.jph".

Modifying HTML templates

The Export Template option can be used to save the default templates for generating the HTML pages to files. These templates can then be edited with any text editor to create custom look & feels for the albums.

Viewing slideshow

The currently open photo album can be viewed as a slide show by selecting View/Slideshow. The program will show the photos one after another in full screen mode, with description shown in the bottom as sub-titling. Each photo is shown for 5 seconds. Up and down arrows can be used to return to previous photos or skip to next photo. Esc will cancel the slide show.
Slideshow can be started start from command line with command "JPhotoShow album.jph".

Viewing EXIF information

The EXIF information is created by the digital camera when the picture is taken. It includes exposure information like aperture and exposure time as well other information. This information can be viewed by selecting the View/EXIF information menu option.

2.3 Class overview

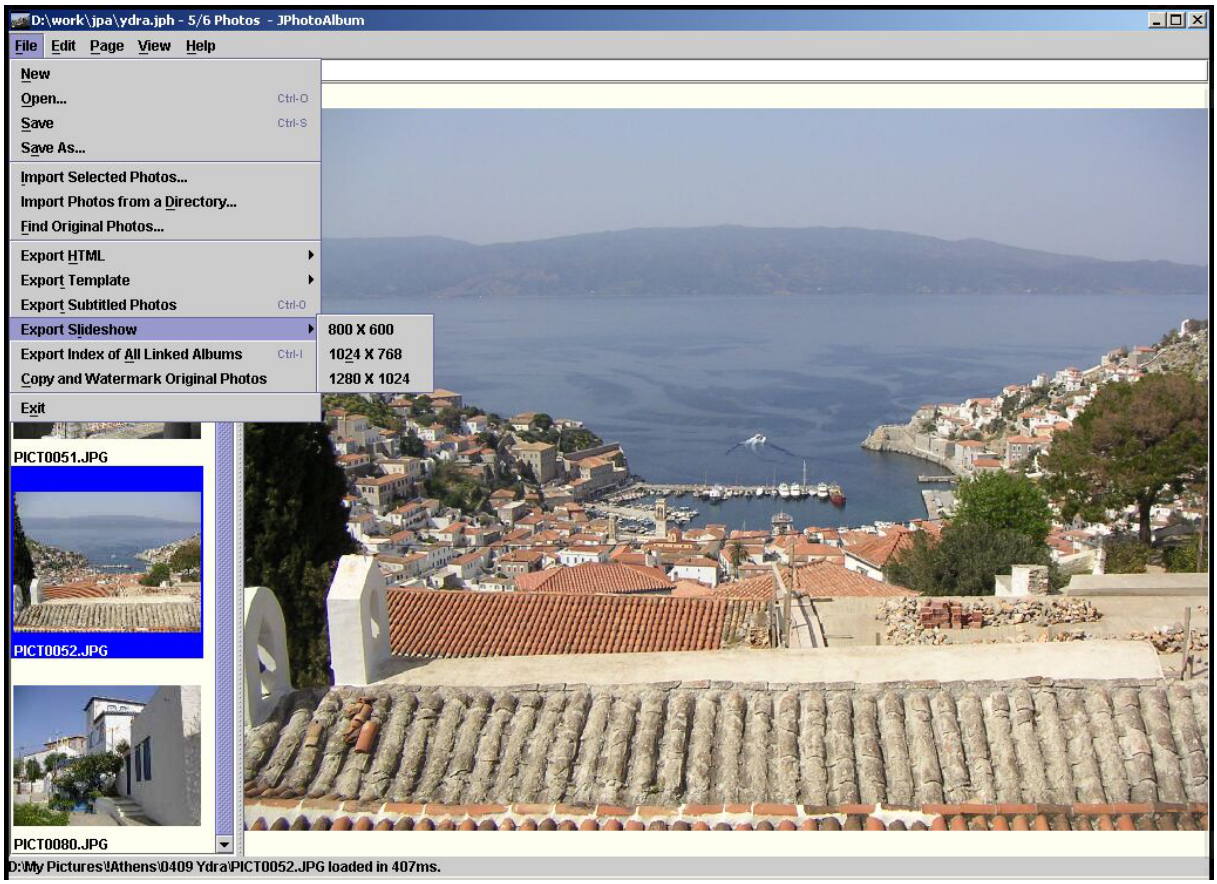
Class	Description	Lines
JPhoto	Container for a single photo. Handles basic actions with the picture. Getting the picture from the file, resizing, making thumbnails, saving the picture.	714
JPhotoAction	Inherits AbstractAction, used by the JPhotoMenu for as an action container.	39
JPhotoAlbumLink	Container for a link to another album. Extends JPhoto. Allows photos from different galleries to be included in a gallery.	109
JPhotoBrowser	Browser for viewing and selecting pictures.	258
JPhotoCollection	Container for collection (gallery, page) of pictures. Allows various ways of exporting into a set of web pages.	977
JPhotoDirectory	Extension of JPhoto which allows to include not only real photo, but also just a text element. Used by JPhotoCollection when searches for files and the certain file is a directory.	81
JPhotoExif	Handles the EXIF information of the picture.	78
JPhotoExifDialog	JDialog that provides EXIF information	85
JphotoFrame	Main class of the program. Draws the frame, handles the actions performed.	992
JPhotoList	List of JPhotos. extends JList	193
JPhotoMenu	Handles the main menu of the program	174
JPhotoPageInfo	Container for data about the web pages. Colors, title, keywords, ...	78
JPhotoPanel	Resizes and displays a single photo either in full screen or in the browser.	272
JPhotoShow	Slideshow that can be either run from the main program, or as a standalone program. But in this case requires many of the other classes.	148
JPhotoStatus	Class informing user about the status – providing progress bars and status messages	117
JPhotoTransferHandler	Allows copy, paste,...	170
Utils	Various utilities	550

2.4 Class Hierarchy

- class java.lang.Object
  - class javax.swing.AbstractListModel
    - class fi.iki.jka.JPhotoCollection
  - class java.awt.Component
    - class java.awt.Container
      - class javax.swing.JComponent
        - class javax.swing.JList
          - class fi.iki.jka.JPhotoList
        - class javax.swing.JPanel
          - class fi.iki.jka.JPhotoPanel
      - class java.awt.Window
        - class java.awt.Dialog
          - class javax.swing.JDialog
            - class fi.iki.jka.JPhotoBrowser
        - class java.awt.Frame
          - class javax.swing.JFrame
            - class fi.iki.jka.JPhotoFrame
            - class fi.iki.jka.JPhotoShow
  - class fi.iki.jka.JPhotoExif
  - class fi.iki.jka.JPhotoMenu
  - class fi.iki.jka.JPhotoPageInfo
  - class fi.iki.jka.JPhotoStatus
  - class java.util.Observable
    - class fi.iki.jka.JPhoto
      - class fi.iki.jka.JPhotoAlbumLink
      - class fi.iki.jka.JPhotoDirectory
  - class java.lang.Thread
    - class fi.iki.jka.JPhoto.ThumbLoader
    - class fi.iki.jka.JPhotoCollection.CopyThread
    - class fi.iki.jka.JPhotoCollection.ExportPhotosThread
  - class javax.swing.TransferHandler
    - class fi.iki.jka.JPhotoTransferHandler
  - class fi.iki.jka.Utils

3 Changes made

The functionality added by me is export of subtitled galleries into self-executable slideshow, which means standalone jar archive, which contains pictures to be viewed, their subtitles and a separate picture browser, independent from the original program. The following screenshot shows the new option in the program menu. There is a choice of three different resolutions to choose from.



I added 3 more classes and in order to integrate them into the project, I changed 4 of the original classes.

Class name	Lines added	Changed / New
JPhoto	59	changed
JphotoCollection	12	changed
JphotoFrame	47	changed
JphotoMenu	10	changed
JPhotoSlideShow	174	new
JPhotoSlideShowFrame	104	new
JPhotoSlideShowMaker	202	new

The jar file with the gallery has following structure:

```
/fi/iki/jka/JPhotoSlideShow.class
/fi/iki/jka/JPhotoSlideShowFrame$1.class
/fi/iki/jka/JPhotoSlideShowFrame$ShowAction.class
/fi/iki/jka/JPhotoSlideShowFrame.class
/gallery/0000.jpg
/gallery/0001.gif
...
/gallery/photos.lst
/Meta-inf/manifest.mf
```

The photos.lst file is a text file which contains path and comment for each picture on separate lines.

3.1 Class JPhotoSlideShowFrame

This class acts like the main program of the standalone viewer. Some of the functionality is taken from JPhotoShow (keyboard handling), but I couldn’t use JPhotoShow for this purpose, because it is dependent on the other classes of the project and containing them in the output jar archive would make it intolerably large, for example for sending by email.
From the programmer’s point of view, there is nothing very interesting in this class.

3.2 Class JPhotoSlideShow

This class contains actually all the functionality of the separate browser. First it loads the picture list (/gallery/photos.lst) file.

```
void loadPictureList(String fileName) {
    // initialization of variables
    picList = new ArrayList();
    // loads all the file into ArrayList called picList
    try {
        BufferedReader vstup =
                new                                          BufferedReader(new
InputStreamReader(this.getClass().getResourceAsStream(fileName)));
        String s;
        while ((s = vstup.readLine()) != null) {
            picList.add(s);
        }
        vstup.close();
    }
    catch (IOException e){
        System.out.println ("I/O Error while reading the picture list...");
    }
    nPics = picList.size()/2; //actual number of pictures in the gallery
}
```

As the file is not a real file, but just an item of jar archive, it’s InputStreamReader cannot be initiated just by filename, but we have to get it’s URL:

```
this.getClass().getResourceAsStream(fileName))
```

In the following step, we load the image and preload the following one.

Than, we have to rescale the image, to fill the screen.

```
void reScaleImg() {
    screeny = getHeight();
    screenx = getWidth();
    picy = curImg.getHeight(this);
    picx = curImg.getWidth(this);
    float picRatio = (float) picx/picy;
    float screenRatio = (float) screenx/screeny;
    if (picRatio<screenRatio) {
        scaledy = screeny;
        scaledx = Math.round(screeny * picRatio);
    }
    else {
        scaledx = screenx;
        scaledy = Math.round(screenx / picRatio);
    }
}
```

Following step is drawing the picture on the screen.

```
g.drawImage(curImg,   (screenx-scaledx)/2,   (screeny-scaledy)/2,   scaledx,
scaledy, this);
```

And finally writing the subtitle in white color with black border. This is why we have to write it five times.

```
g.setColor(Color.BLACK);
g.setFont(new Font("SansSerif", Font.BOLD, 16));
int commentMove = g.getFontMetrics().stringWidth(comment)/2;
g.drawString(comment,(screenx/2) - commentMove-1,screeny - 5-1);
g.drawString(comment,(screenx/2) - commentMove+1,screeny - 5+1);
g.drawString(comment,(screenx/2) - commentMove-1,screeny - 5+1);
g.drawString(comment,(screenx/2) - commentMove+1,screeny - 5-1);
g.setColor(Color.WHITE);
g.drawString(comment,(screenx/2) - commentMove,screeny - 5);
```

To switch to the next picture, the main class (JPhotoSlideShowFrame) has to call the following function, which gets the preloaded image and depicts it on the screen and preloads another image to be shown.

```
void selectNext() {
    if (activePic < (nPics - 1)) {
        prevImg = curImg;
        curImg = nextImg;
        activePic++;
        comment = getComment(activePic);
        repaint();
        if (activePic < (nPics - 1)) {
            nextImg = getPic(activePic + 1);
        }
    }
}
```

For switching to previous picture, there’s a similar procedure.

Besides the picture which is currently shown on the screen, the program keeps the previous and following picture preloaded in the memory, as it is very slow process to decompress the file from the archive, decompress the jpg, gif, or any other file and draw it on the screen.

3.3 Class JPhotoSlideShowMaker

This class is to export the gallery into the jar archive. All the necessary classes for working with jar archives are included in the package java.util.jar. This class runs as a separate thread and while it is exporting, a progress bar is shown on the screen.
It uses I/O streams to write directly to the new .jar file. No temporary files are used. The parameters of the constructor

```
public JPhotoSlideShowMaker(String targetFile, ArrayList pl, int resol)
```

are target jar file, list of pictures and their comments and a number that identifies the resolution of the future gallery. 1 stands for 800x600, 2 means 1024x768 and 3 menas 1280x1024 pixels.
The manifest of the jar file has to be created manually, and written into the file. The following function returns the manifest for the jar file:

```
Manifest getManifest() {
    String [] manLines = {
        "Manifest-Version: 1.0\n",
        "X-COMMENT: Main-Class will be added automatically by build\n",
        "Created-By: 1.4.2_07-b05 (Sun Microsystems Inc.)\n",
        "Ant-Version: Apache Ant 1.6.2\n",
        "Main-Class: fi.iki.jka.JPhotoSlideShowFrame\n"};
    Manifest man = new Manifest();
    try {
        for (int i=0; i < manLines.length; i++) {
            ByteArrayInputStream            ba            =            new
ArrayInputStream(manLines[i].getBytes());
            man.read(ba);
        }
    }
    catch (IOException e3) {
        System.out.println("error while creating the manifest...");
    }
    return man;
}
```

Now, when we have the manifest, we can initialize a new jar file

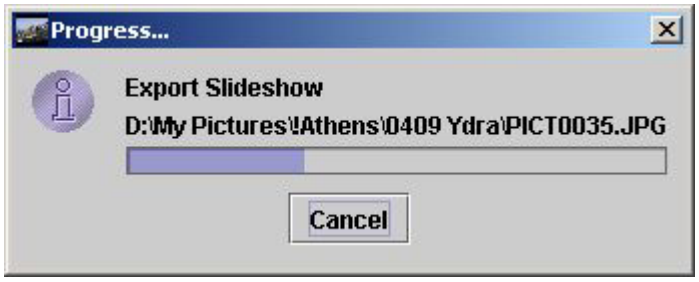
```
FileOutputStream f = null;
JarOutputStream out = null;
JPhotoStatus.showProgress(JPhotoMenu.A_EXPORT_SLIDESHOW_INDEX,    "Preparing
export...", 0);
try {
    f = new FileOutputStream(targArchName);
    out = new JarOutputStream(new OutputStream(f),getManifest());
```

and we can start writing the pictures into the archive:

```
    // writing the pictures
    for(int i = 0; i < nPics; i++) {
        progress = Math.round(i*100 / (float)(nPics));
        String src = picSet[i];
        String                              dest                              =
getJarPath(makePicName(i)+getFileExtension(picSet[i]));

JPhotoStatus.showProgress(JPhotoMenu.A_EXPORT_SLIDESHOW_INDEX,          src,
progress);
        JPhoto jp = new JPhoto(new File(src));
        BufferedInputStream              in              =              new
BufferedInputStream(jp.getSlideStream(resolution));
        JarEntry entry = new JarEntry(dest);
        byte[] file = new byte[in.available()];
        in.read(file, 0, file.length);
        out.putNextEntry(entry);
        out.write(file, 0, file.length);
    }
}
catch(Exception e1) {
    e1.printStackTrace();
    System.out.println("Error occured while writing the pictures into the
jar archive...");
}
```

Whereas the progress bar is shown.



With a similar procedure, we also input all the necessary classes.

3.4 Class JPhoto

As there are three choices for the resolution of exported pictures, I had to add a method to this class that would export the pictures in this resolutions.
First I added the necessary dimensions:

```
protected static Dimension slideLimits1 = new Dimension(800, 600);
protected static Dimension slideLimits2 = new Dimension(1024, 768);
protected static Dimension slideLimits3 = new Dimension(1280, 1024);
```

then a method, that returns BufferedImage of the slide:

```
public BufferedImage generateSlide(BufferedImage img, Dimension
slideLimits) {
    Dimension                                dim                                =
Utils.getScaledSize(slideLimits,img.getWidth(),img.getHeight());
    BufferedImage sldImg = new BufferedImage(dim.width, dim.height,
BufferedImage.TYPE_INT_RGB);
    Image scaledImg = img.getScaledInstance(dim.width, dim.height,
Image.SCALE_SMOOTH);
    Graphics2D g2 = sldImg.createGraphics();
    g2.drawImage(scaledImg, null, null);
    g2.dispose();
    return sldImg;
}
```

and finally, a method that returns an input stream of the picture, as we want to include it into the archive directly.

```
public ByteArrayInputStream getSlideStream(int resol) {
    Dimension d = new Dimension(slideLimits1);
    if (resol == 2) d = slideLimits2;
    if (resol == 3) d = slideLimits3;

    BufferedImage bi = generateSlide(this.getFullImage(),d);
    ByteArrayInputStream slide = null;
    ByteArrayOutputStream out = null;
    try {
        out = new ByteArrayOutputStream();
        JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
        JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bi);
         param.setQuality(0.8f,false);
         encoder.setJPEGEncodeParam(param);
         encoder.encode(bi);
         bi.flush();
         slide = new ByteArrayInputStream(out.toByteArray());
         return slide;
     }
     catch (Exception e) {
         JPhotoStatus.showStatus(null, "getSlideStream error " + e);
         return null;
     }
     finally {
         try {
             out.close();
         }
         catch (Exception e) {
             JPhotoStatus.showStatus(null, "getSlideStream error " + e);
         }
     }
}
```

3.5 Class JphotoCollection

Here, I added one method for exporting the collection. Actually, it just creates a new instance of JPhotoSlideShowMaker and runs the thread.

```
public boolean exportSlideshow(String targetFile, int resol) {
    JPhotoSlideShowMaker ssm = new JPhotoSlideShowMaker(targetFile, photos,
resol);
    ssm.start(); //runs as a separate thread
    return(true);
}
```

3.6 Classes JphotoFrame and JphotoMenu

There’s some minor contribution to these classes also, but it is only adding menu items and calling appropriate methods after a mouse or keyboard action is performed.

4 Communication with the developers

```
to: jari@jpkware.com
subject: JPhotoAlbum
from: zbynek.muzik@email.cz

Hi Jari,

My name is Zbynek Muzik and I'm from Czech Republic. At the moment
I'm an exchange student at Athens University of Economics and
Business and one of my classes is Maintenance and Refactoring of
Information Systems which deals with open source software.
All the students have to contribute to an open source project and I
chose your JPhotoAlbum. I added one feature to it - export of the
photo collections into one self executable .jar file. As I cannot
find your email address on this page, I cannot send you the sources,
but if you are interested in my extension, feel free to contact me.
Best Regards
Zbynek Muzik
```

```
to: zbynek.muzik@email.cz
subject: Re: JPhotoAlbum
from: jari@jpkware.com

It is nice to hear that people are using JPhotoAlbum, and better
yet, extending it! You can reply to this message with the source
attached in a zip file.
Please indicate clearly which parts you have changed.
```

```
to: jari@jpkware.com
subject: Re: JPhotoAlbum
from: zbynek.muzik@email.cz

Ok, hereby, i'm sending you the sources of the new classes and
diff's of those modified. Good luck.
```

```
to: zbynek.muzik@email.cz
subject: Re: JPhotoAlbum
from: jari@jpkware.com

Thanks for the sources, this is very useful functionality, I now
incorporated it into the source in CVS. I also added it to help,
and gave you credit in the copyright statement.
It seems that your slideshow viewer class has somewhat different
functionality than the built-in slideshow; you don't have timer
or wrap-around from last to first. Is this intentional? I guess
it could make sense to try to combine both slideshows to a single
one with exact same functionality.
I also somewhat question the decision to scale the 800x600 images
to fit the screen, if the screen is much larger, the resulting
image is not very pretty.
```

```
to: jari@jpkware.com
subject: Re: JPhotoAlbum
from: zbynek.muzik@email.cz

Hi
The wrap-around from the first to the last one could be useful. Also
the 800x600 pics look pretty ugly, that's true, I'll fix this things
and send you the sources next month, as i'm pretty busy now. If you
have time, feel free to fix it. The reason why i didn't use the
built-in slideshow is, that there are some dependences on the other
classes and including them and the other libraries into the jar
archive would make the file too large for e.g. sending by email etc.
```

5 Conclusions

Participating on an open source project improved my code reading and programming skills very much. Now I also have much better clue about how bigger programs work and how to design them. Understanding a program written by someone else is not very easy from the very beginning, but after having learned how to use some software tools, it is much more comfortable. I found, that the open source community is open minded to various ideas of other people, eager for cooperation, helpful and friendly.