Using ImageMagick, Lasso and Passthru

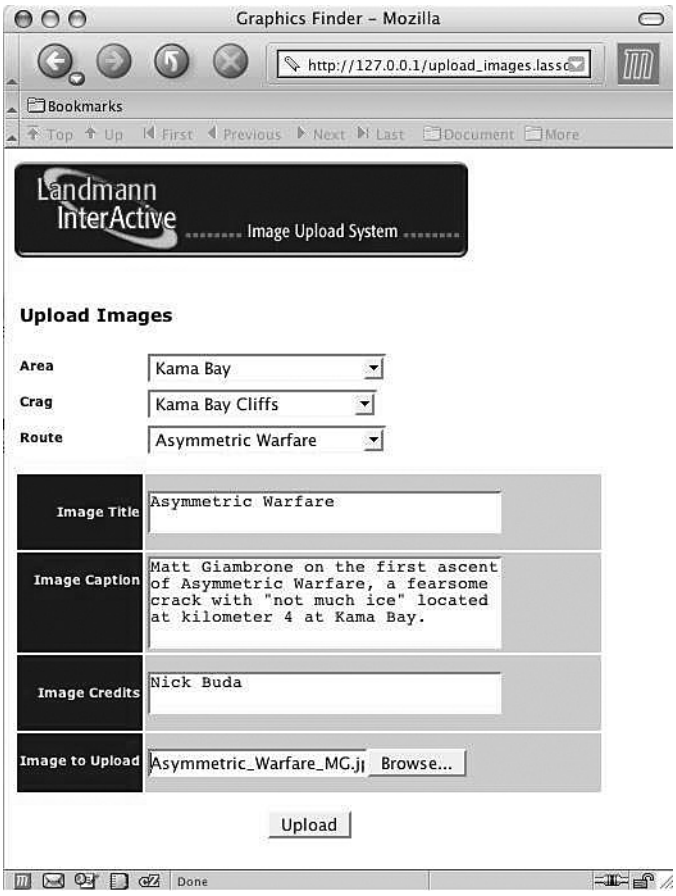
Eric Landmann

INTRODUCTION

Photo upload systems that "do it all" for the user are becoming increasingly common on websites. Systems that allow a user to upload a photo and capture some additional information are used in many types of applications. Some of these include an employee directory or staff listing, an image bank, a blog, a members area of a chat site, a used equipment site, or a real estate listings system.

The examples provided arefor a site providing information about climbing routes. The photo uploaded might show a picture of the climb which would be of interest to other climbers attempting the route.

This functionality would typically be found in an administrative or user area to restrict access from the general public. The example provided does not show any user authentication code or scheme; this would have to be added by the coder. Below is a screenshot of the application just before the user submits the form.



Why do it this way?

Lasso comes bundled with ImageMagick, a very useful suite of utilities that can manipulate images in many ways. Lasso ties into ImageMagick through the use of Lasso's Image tags.

Using Lasso's builtin Image tags, you can resize an image, get info on an image, crop images, sharpen, and do many other things. Our solution does not use all of this functionality. Instead, we use PassThru, a plugin that issues Terminalstyle commands, to pass commands directly to ImageMagick. It is a fair question why we would use this method instead of using the builtin Lasso image tags. Here are some reasons:

1. Speed — Working with ImageMagick directly is faster than using Lasso's tags
2. Error Reporting — We can capture more descriptive errors directly from ImageMagick
3. Flexibility

   — by using a separate install of ImageMagick, you can take advantage of new releases

   — You can tailor ImageMagick commands to specific image situations

   — Upload and image directories are easily tailorable

FEATURES OF THIS APPLICATION

Requires the user to make selections from dropdowns to be able to upload an image. This process identifies a record in the database to which the photo will be related. Cleans up troublesome filenames and creates a unique filename for the uploaded source image file.

Creates three versions of the image, with modifiable sizes.

Created images are placed in three separate folders, allowing one database entry for the filename.

Folder paths called in HTML determine which size photo is accessed; the filename is the same.

This version works with .jpg files, but is easily modifiable to include other file formats by calling the appropriate ImageMagick convert command.

Employs a siteconfig file, which has sitewide settings and makes moving code from one site to another relatively easy.

Original file can be archived if desired (set in the siteconfig file).

The upload page is selfposting (less code maintenance).

SOFTWARE USED

Lasso — To communicate with the webserver, databases, and do various utility tasks ImageMagick — To analyze images and create derivative images. We are using the Lasso installed version in this example for ease of use, but an external installation certainly could easily be used.

PassThru — Aplugin that communicates between Lasso and the Terminal to pass commands through to and receive message from the operating system. It is a commercial addon. See the "References" section for where to get PassThru.

MySQL — To store the record information for the routes and images.

OPERATING PRINCIPLES

Here is how this application operates. The user is presented with a form with a dropdown dialog. When an "Area" is selected, the form selfposts, a database lookup is performed, and a "Crag"

dropdown list is presented. When a crag is selected, the form selfposts again, and a list of climbs are presented. When a climb is selected, the form selfposts again, and presents a form to fill out an image title, caption, credits, and a browse button to select the photo to upload. After the photo is selected and submit, the form selfposts one final time.

If all the form submissions pass the tests, the process_uploads.inc file does all the heavy lifting. This include file contains all the code to clean up the submitted filename of improper characters, create a unique new filename for the uploaded file, create the three sizes of thumbnails (called large, small, and thumb images) stored in their respective folders, make the database entry, and take care of file maintenance. Here is a code snippet that produces the large image (line 128 of process_uploads.inc). It also checks whether the image is wider or taller, and issues the appropriate ImageMagick command:

```
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
//
Make Large version
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
If:(#Height < #largeheight) || (#Width < #largewidth);
Local('ResizeTest') = 'Image Height or Width LESS THAN large image, make large
version, no resize<br>\r';
Local:'MakeLarge' = ($svPathToIM 'convert -density 72x72 -colorspace RGB "'
($svWebserverRoot) (#ThisFilePath) '" "' (#ULPathImageLargeOUT) '"');
Else;
Local('ResizeTest') = 'Image Height or Width GREATER THAN large image, make
large version, resize 600x600<br>\r';
Local:'MakeLarge' = ($svPathToIM 'convert -density 72x72 -colorspace RGB
-resize ' (#largewidth) 'x' (#largeHeight) ' "' ($svWebserverRoot) (#ThisFilePath)
'" "' (#ULPathImageLargeOUT) '"');
/If;
```

Immediately after this, and sprinkled throughout the code are various places that display information when debugging is on. This particular bit of code displays the value of the variables ResizeTest and MakeLarge:

```
If: $svDebug == 'Y';
'<p class="debug">\n';
'141: ResizeTest = ' (#ResizeTest) '<br>\r';
'141: MakeLarge = ' (#MakeLarge) '<br>\r';
'</p>\n';
/If;
```

Immediately after this (at line 146), the PassThru command is run, and the output of this command is captured in the variable "converting":

// Run the PassThru command to convert the file
Local('converting') = PassThru(#MakeLarge, -username=$svPassThruUsername, -password=
$svPassThruPassword);

At this point, the command has been passed to ImageMagick, and it should be doing its thing. If you are watching the server when executing this, you should see the newlycreated files appear in the image folders under /site/images.

FILE STRUCTURE

SQL Files

A filecalled beta_content_demo.sql contains the structure and data of the tables used in this application. To use this, create a table in MySQL called "beta_content_demo", and run the beta_content_demo.sql file. It will create the four tables listed below and populate them with test data.

Database and Table Descriptions

beta_content contains information for the climbs

beta_crag contains information about the crags (climbing areas)

beta_errors contains information accessed by the Show_Error tag

beta_images contains information about the uploaded images

Code Files

Files of note are as follows:

upload_images.lasso — The entry point for this application

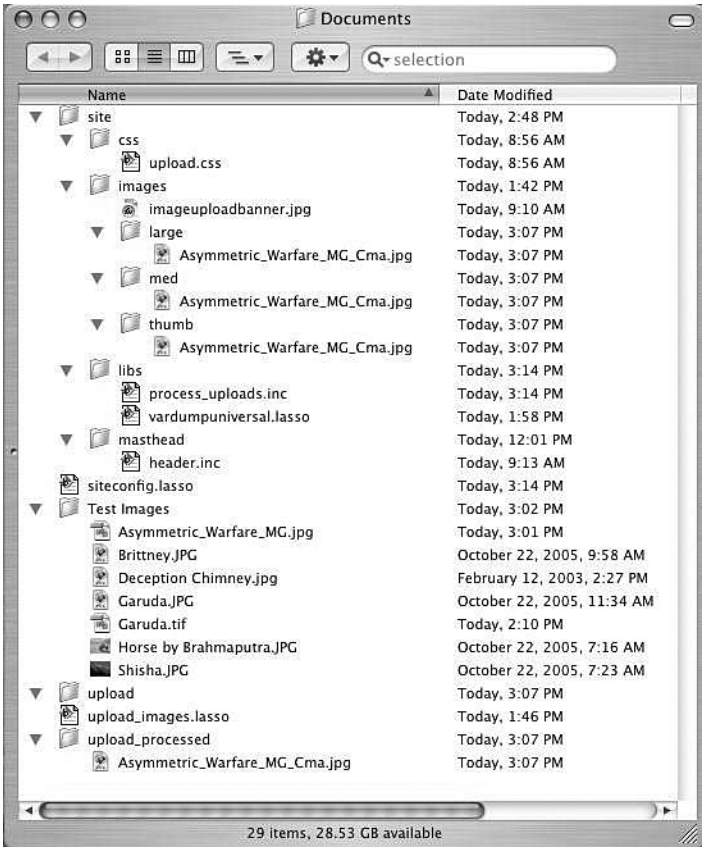
siteconfig.lasso — controls site and serverspecific specific information

process_uploads.inc — does the actual image file creation, communicating with ImageMagick via PassThru; also creates the database entry and handles error and message generation

vardumpuniversal.lasso — A handydandy variable dump utility that you can put anywhere in a page to display some information about the current file and the value of variables. It is written to not display Lasso "reserved" variables (those starting with "_") or siteconfig variables (those starting with "sv"). Used in debugging and site development.

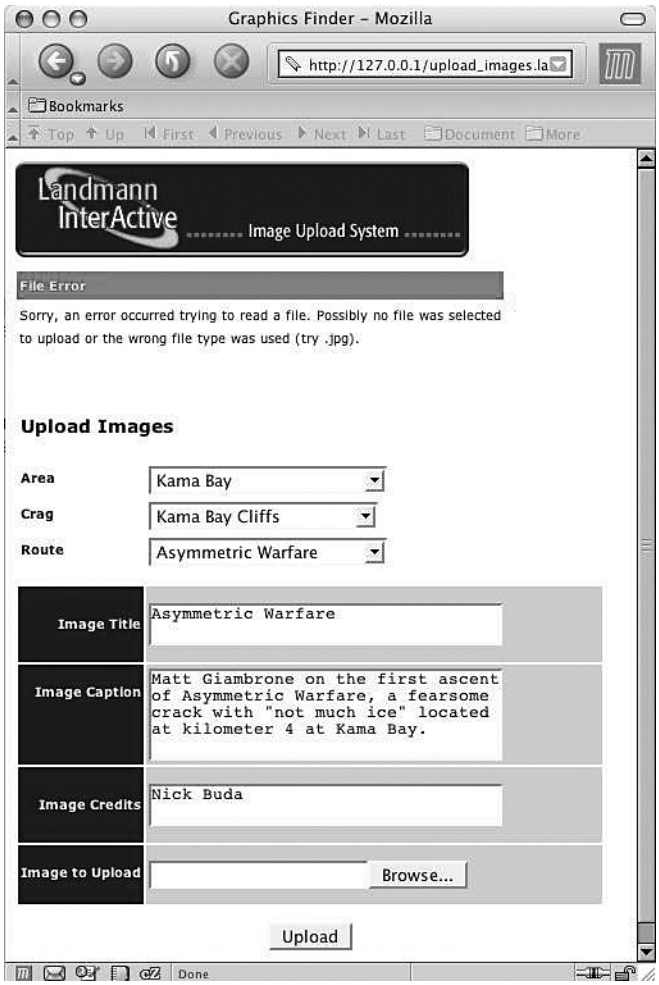
Directory Display

The list below shows a typical image upload system directory after one image (named "Asymmetric_Warfare_MG.jpg") has been uploaded. The three generated image files (a large, medium, and thumb version) are each in their own subfolders of /site/images. The original file has been renamed and moved to upload_processed.



Error Checking and Display

A custom tag called "Show_Error" is defined in the siteconfig.lasso file. It takes several parameters and returns a formatted table with a message. It could be an error or it could be a success message, depending upon the code that is passed to the tag. The code that is passed to the tag is typically a fourdigit number, like "5061" which means "Upload Successful." Below is a typical error message displayed at the top of the page:



This tag can easily be used with a redirect by passing a parameter "error" through the URL.

For example:

```
'upload_images.lasso?Error='$vEr
ror'&Option='$vOption;
```

The error is then displayed by calling the Show_Error tag as follows:

```
Show_Error: -ErrNum=(Var:'vError'), -Option=(Var:'vOption'),
-PosColor='0099FF', -NegColor='FF0000', -BgColor='CCCCCC;
```

See the tag definition in the siteconfig for info on what the parameters mean.

FILE AND FOLDER PERMISSIONS

Lasso is very persnickety about permissions to perform file manipulations. Rather than repeat the documentation about how to do it, check the Lasso documentation regarding file tags and folder permissions. The example files make use of a user ID and password for this purpose, set in the siteconfig. One commonly overlooked aspect is setting folder permissions for the enclosing folder where the graphics are stored. See screenshot below for example folder permissions.

DEBUGGING

This can be controlled either globally through a variable declaration at the top of the siteconfig (by setting svDebug = Y), or can be turned on for any page or code block. Turning debug on will allow you to see all the commands sent to ImageMagick, all the database calls and variable states.

Debugging also calls the vardumpuniversal.lasso file. This file displays some page information, action_params that have been submit to that page, and the variable values. It is typically used at the bottom of the page. A sample of debugging output is below:

```
PAGE INFO
Response_Filepath = /upload_images.lasso
Referrer_URL = http://127.0.0.1/upload_images.lasso

ACTION_PARAMS DUMP
Area = Orient Bay
Crag = Orient Bay
RouteID = 6

VARIABLE DUMP
vImageTitle =
vCrag = Orient Bay
vOption =
vImageCredits =
SQLSearchAreas = SELECT DISTINCT Crag_Area FROM beta_crag ORDER BY Crag_Area
vRouteID = 6
SQLSearchCrags = SELECT DISTINCT Crag_Crag FROM beta_crag WHERE Crag_Area = "Orient Bay"
newparams =
vAreaSelect = Thunder Bay
SQLSearchRoutes = SELECT DISTINCT Route_Name,ID FROM beta_content WHERE Crag="Orient
Bay" ORDER BY Route_Name
vError =
vArea = Orient Bay
vCaption =
vProcess =

vardumpuniversal.lasso loaded
```

REFERENCES

The Lasso documentation provides examples of how to set up the file tags to allow manipulations of the sort needed here. See Chapter 29 of the "Lasso 8 Language Guide."

PassThru

PassThru has its own reference manual (included with the software).

http://www.execuchoice.net/

ImageMagick Examples

http://www.cit.gu.edu.au/~anthony/graphics/imagick/

Lasso

http://www.omnipilot.com/

LassoTalk Archives

http://www.listsearch.com/

CREDITS

Thanks to the following Lasso coders for putting up with my learning curve, and contributing code bits to the community and to me directly:

— Pier Kuipers for the Create Unique ID tag
— Keith Schuster for the original idea about the error tag
— Greg Willits and Bil Corry for bits of the upload code
— Steffan Cline for help with PassThru
— Chris Corwin for help with ImageMagick
— the Lasso community, for being there