The Glue

Introduction

This page describes the interaction between the functional and technical design. For every action in the functional design is described what happens 'under the hood' and the called functions are listed.

Analyzing/reverse engineering the code is quite intensive, but many parts are quite straight forward. For this reason only the most important and complicated processes are described here; the uploading of a file with and without google gears, the downloading and the handling of vouchers.

List files

- user navigates to myfiles
- *init()* is loaded in myfiles.mxml
- flex requests file list from fs_main.php, which will call *getUserFiles* in Functions.php
- *getUserFiles* will extract uid from current session
- *getUserFiles* will query database and return list of files belonging to uid

New upload

With gears

On the client:

- User navigates to New Upload
- Upload process is initialized by init() in upload_gears.mxml
- All callbacks are attached to function *returnStatus* in upload_gears.mxml
- The upload button is pressed by the user
- *doGearsUpload()* in upload_gears.mxml is called
- The email address is checked on validity
- Check number of emails recipients (defined in config.php)
- Check if AuP is selected, if not give alert
- Call javasript function *upload(filevoucheruid)* defined in fs_gears.js
- Split the file to upload in chunks, and call *sendChunk()* with each chunk
- Start POST connection to fs_gears_upload.php with arguments n = filename, b = start, vid = voucherid and total = total, put chunk in post
- callback function *returnStatus* is called with progress information and screen is updated
- When complete callback is received the upload is complete, if an error occurs or the upload is canceled it is handled here also

On the server:

- receives POST connection to *fs_gears_upload.php*

- Receives file data and stores this in a temp file in location defined by *site_temp_filestore* in config.php
- TODO: can't find where the file is moved to the final location

Without gears

On the client:

- User navigates to New Upload
- The upload button is pressed by the user
- *doupload()* in upload_std.mxml is called
- Check file size, if not 0 or too big (defined in config.php)
- Check number of emails recipients (defined in config.php)
- Get *SessionID* of current user TODO: HOW
- Open the file
- Start POST connection to fs_uploadit.php with argument *S* set to the *SessionID*
- Stream the file to server
- The screen is displaying a progress bar which is updated by the *progressHandler* callback function
- When the fileupload is completed the callback function *completeHandler* is called by flash.
- fs_main.php is called with the *closeVoucher* call, which indicates the file is completed. A list of JSON encoded properties is added to the request.
- This will display a confirmation or error, depending on if the upload was completed, canceled or an error occurred.
- fs_main.php is called with the *insertFile* call

on the server:

- In this case all interaction with the user is handled by fs_main.php
- clients initiates a POST connection to fs_uploadit.php
- HTTPS connection is forced
- authentication is checked
- The file is received, and moved to the designated file storage, by move_uploaded_file
- When the closeVoucher call at fs_main.php is received the database record for the file is updated and the filestatus is set to 'closed'
- When the insertFile call is received a database record for that file is inserted in the table 'files' and filled with the data supplied with a JSON encoded list of file properties
- The insertFile function will also sent an email to the recipients

File Download

- A user will press a download link in his email program
- A browser opens and will open download.php and will extract voucher ID from arguments (vid)
- The file name is extracted from the vid
- The location of the file is verified, otherwise file not found error
- then *readfile_chunked()* in *download.php* is called, which will read the file in chunks
- The file content is returned to the client in chunks with ob_start() and ob_flush()

List vouchers

- user navigates to Vouchers
- flex requests file list from Technical design - PHP#fs_main.php, which will call *getVouchers* in Technical design - PHP#Functions.php
- *getVouchers* will extract uid from current seesion
- *getVouchers* will query database and return list of files belonging to uid with status set to 'voucher'

Create voucher

- user navigates to Vouchers
- user presses create voucher button
- *createVouchers()* in vouchers.mxml is called
- parameters are extracted and checked
- *saveVoucher()* in vouchers.mxml is called with parameters
- Technical design - PHP#fs_main.php with *insertFile* call, which will create the voucher in the files table

Use voucher

- A user will press a voucher link in his email program
- A browser opens and will open filesender
- From there the sequence is identical to New upload