

Technical Notes:

Migrating a Complex Corporate WordPress Site to Drupal 8

Howard Thompson
January 10, 2021

Note: Due to Non-Disclosure Agreement requirements, the documented name of the corporation has been changed and any identifying information modified.

I tried different off the shelf migration methods

Documentation for WordPress Migrate: <https://www.drupal.org/node/1593370>
<https://www.drupal.org/docs/7/migrating-to-drupal-7/migrating-from-wordpress>
https://www.drupal.org/project/wordpress_migrate

** As of Jan 9, 2021, the Word Press Migrate Module is in Alpha4 with 67 Open Issues.

I generated a WXR file (XML) from WordPress and installed Drupal “WordPress Migrate” / “Migrate” module (now in Core). I went to “WordPress Migration” and chose “Import from a File”. That failed because the generated XML file wasn’t valid XML. I manually edited the XML file to make it valid XML. The import “worked” but the Categories were long lines of corrupted text. Tags looked odd too; some combined. Feature Images didn’t make it. The HTML was a straight copy, so everything still pointed to the old WordPress site... for images, links, blog posts, projects, users, ... everything. Drupal by default didn’t handle the Caption short-codes like WordPress... so, they didn’t work right. Then there were the non-standard content types that “WordPress Migration” knew nothing about and obviously didn’t migrate. The Authors/Users were copied over properly.

In all cases, my future would be fixing those HTML Content Bodies.

My conclusion was that the WXR migration might have been suited for more simple content. This was a complex site with lots of plugins and the data that went with them.

Reality

The reality is you are building a whole new website with content from the old Word Press site. You have to define all your content types (Articles, Projects, People, etc.), pages, and views. Then somehow import all the data from the old WordPress site, in a format needed to recreate that Content Data inside Drupal.

The only thing I kept from the WXR import was the Authors/Users.

Inspiration: <https://www.previousnext.com.au/blog/migrating-wordpress-blog-drupal-whilst-importing-embedded-images-and-files-use-media-module>

The content migration project plan looked like this:

- Move image files into a flat import folder
- Extract the Source (content data) into CSV files
- Process the Source CSV files into Media Library Import CSV files
- Import the Media Library CSV files into the Media Library
- Process the Source CSV files into Content-Type CSV files
- Fix the Content-Type html bodies
- Import the Content-Type CSV files
- Process the Source CSV files to validate each link and image

Move image files into a flat import folder

Both WordPress and Drupal put newly imported media into folders based on the name of the current month, year, etc. So, you can't just use the names of the image files because there may be many "interior.jpg" files. And they will all end up in the same Month/Year folder in Drupal; if imported within the same month.

The solution is to move the WordPress names of the folders to be part of the Media file name. Rather than copy the image files, links are made back to the original file but using the new file name.

output/uploads-2012-12-detail.jpg will be linked to *uploads/2020/12/detail.jpg*

Cd to wp/wp-content

Create a sub-folder "output"

You should have an "uploads" sub-folder

```
find uploads -type f -exec sh -c 'new=output/${echo "{}" | tr "/" "-" | tr " " "_"}; ln "{}" "$new" \;
```

Based on:

<https://gist.github.com/dale-c-anderson/e77979c8ddb45811a2bd74a6e6258c3f>

* In our case, we also had an additional folder for pdf files.

Extract the Source (content data) into CSV files

We had five different content types. The example below is just "Articles".

You are going to have to dig through your WordPress database and figure out how all the data is stored. This can get more than a bit strange depending on the plugins being used.

MySQL SQL

final-articles.sql

```

WITH terms AS (
SELECT wp_posts.ID as 'ID', wp_posts.post_title AS duplicate_title,
      wp_terms.name AS Category,
      wp_posts.post_date AS duplicate_date,
      wp_posts.post_name
FROM wp_posts
INNER JOIN wp_term_relationships
ON wp_posts.ID = wp_term_relationships.object_id
INNER JOIN wp_term_taxonomy
ON wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id
INNER JOIN wp_terms
ON wp_term_taxonomy.term_id = wp_terms.term_id
WHERE wp_posts.post_status = 'publish'
AND wp_posts.post_type = 'post'
AND wp_term_taxonomy.taxonomy = 'category'
ORDER BY wp_posts.post_date DESC
),
AltText AS (
SELECT post_id AS alt_meta_post_id, meta_value AS alt_meta_value
FROM wp_postmeta
WHERE wp_postmeta.meta_key='_wp_attachment_image_alt'
),
categories AS (
SELECT ID as cat_post_id, duplicate_title, duplicate_date,
      GROUP_CONCAT(DISTINCT Category
      ORDER BY Category SEPARATOR '|') AS category_list
FROM terms GROUP BY ID
ORDER BY duplicate_date),
HasImages AS
(
SELECT
  p1.*,
  wm2.meta_value AS post_image,
  wm1.meta_value AS image_post_id
FROM
  wp_posts p1
LEFT JOIN
  wp_postmeta wm1
ON (
  wm1.post_id = p1.id
  AND wm1.meta_value IS NOT NULL
  AND wm1.meta_key = "_thumbnail_id"
)
LEFT JOIN
  wp_postmeta wm2
ON (
  wm1.meta_value = wm2.post_id
  AND wm2.meta_key = "_wp_attached_file"

```

```

        AND wm2.meta_value IS NOT NULL
    )
WHERE
    p1.post_status="publish"
    AND p1.post_type="post"
ORDER BY
    p1.post_date DESC
),
Tags AS (
SELECT wp_posts.ID as tag_post_id, wp_posts.post_title AS Title,
wp_terms.term_id as 'Term_ID',
    wp_terms.name AS Tag,
    wp_posts.post_date AS duplicate_date
FROM wp_posts
INNER JOIN wp_term_relationships
ON wp_posts.ID = wp_term_relationships.object_id
INNER JOIN wp_terms
ON wp_term_relationships.term_taxonomy_id = wp_terms.term_id
INNER JOIN wp_term_taxonomy
ON wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id
WHERE wp_posts.post_status = 'publish'
AND wp_posts.post_type = 'post'
AND wp_term_taxonomy.taxonomy = 'post_tag'
ORDER BY wp_posts.post_date DESC
)
SELECT *, CONCAT("http://192.168.0.9/test-wp/wp/wp-content/output/uploads-",
REPLACE(post_image,"/","-")) AS mod_post_image,
CONCAT("uploads-", REPLACE(post_image,"/","-")) AS alt_post_image,
SUBSTRING_INDEX(category_list, '|', 1) AS mod_category_first
FROM (
SELECT *
    FROM ( SELECT * FROM HasImages LEFT JOIN categories
ON HasImages.id = categories.cat_post_id ) AS A
LEFT JOIN ( SELECT
    tag_post_id,
    GROUP_CONCAT(DISTINCT Tag
ORDER BY Tag SEPARATOR ',') AS tag_list
FROM tags GROUP BY Tag_Post_ID
ORDER BY duplicate_date ) AS B
    ON (A.ID=B.Tag_Post_ID)
) AS Q
LEFT JOIN ( SELECT wp_drupal_users.ID as wp_user_id, wp_drupal_users.user_email,
wp_drupal_users.drupal_user_id FROM wp_drupal_users) AS C
ON (Q.post_author=C.WP_User_ID)
LEFT JOIN AltText altdata ON (altdata.alt_meta_post_id = image_post_id)
ORDER BY ID DESC

```

CSV File Sample

```
ID,post_author,post_date,post_date_gmt,post_content,post_title,post_excerpt,post_status,comment_status,ping_status,post_password,post_name,to_ping,pinged,post_modified,post_modified_gmt,post_content_filtered,post_parent,guid,menu_order,post_type,post_mime_type,comment_count,post_image,image_post_id,cat_post_id,duplicate_title,duplicate_date,category_list,tag_post_id,tag_list,wp_user_id,user_email,drupal_user_id,alt_meta_post_id,alt_meta_value,mod_post_image,alt_post_image,mod_category_first
16362,36,2020-03-18 09:26:59,2020-03-18 13:26:59,"<img class=""aligncenter size-full wp-image-16363"" src=""http://www.test123.ca/wp/wp-content/uploads/2020/03/abc123.jpg"" alt=""COVID-19 and Test 123: Our Work Plan"" width=""2133"" height=""1200"" />
```

Dear Clients,

Given the spread of COVID-19, the Province of Ontario has called a State of Emergency. Test 123 wishes to provide guidance for the safe use of community garden plots to prevent the spread of Covid-19. Garden use is now to be limited to only essential service activities - growing produce.

Do not visit the garden if you are ill.

Avoid touching your eyes, nose or mouth.

Maintain physical distancing.

Gates are being left open to avoid additional handling.

Sanitization protocols are in effect for surfaces (faucets, gates, hand washing stations, etc).

Gardeners are to bring their own tools and not share them.

If you have any questions or comments, please don't hesitate to contact us.

Best regards,

```
Test 123",To our clients during the COVID-19 pandemic,,publish,open,open,,to-our-clients-during-the-covid-19-pandemic,,,2020-03-20 11:11:11,2020-03-20 11:11:11,,0,http://www.test123.ca/?p=16111,0,post,,0,2020/03/abc_123.jpg,16363,16362,To our clients during the COVID-19 pandemic,2020-03-20 09:11:11,News & Events,,,36,howard@test123.ca,81,16363,COVID-19 and Test 123: Our Plan,http://192.168.1.9/test-123-wp/wp/wp-content/output/uploads-2020-03-COVID_test123.jpg,uploads-2020-03-COVID_test123.jpg,News & Events
```

Why the “duplicate_title” and “duplicate_date”?

There were times I needed Title and Date fields formatted differently. I thought I had to have two copies of the fields to use Tamper in Feeds, to get the different formatting. As noted later in this document, under Tips, I didn't need to be doing this. Instead, I should have just used the same field within Feeds Mapping and then used different Tamper configurations for each Field/Target pair.

Processing CSV Files

It all boils down to Dart code and Drupal Feeds import:

1. Process the Source CSV files into Media Library Import CSV files
2. Import the Media Library CSV files into Media Library
3. Process the Source CSV files into Content-Type CSV files
4. Fix the Content-Type html bodies
5. Import the Content-Type CSV files
6. Process the Source CSV files to validate each link and image

Dart Code

The “Generic” Dart code (for Articles or Projects) reads in the CSV file from an Article or Project SQL script CSV file. For our example, let’s say a set of Articles. The Dart code fixes the Image and Anchor/Link URLs. It also fixes paths to other Articles, Projects, and People. There is a lot to fix. It all results in a semi-fixed new Article CSV file. Semi-fixed because there are some invalid URLs that we don’t know what to do with. These URLs are left in the HTML code of the Article. But we generate a “reject” CSV file and a “validate” CSV file. The Reject has URLs that are known to be bad and are on a hardcoded reject list in the Dart code. The Validate CSV has URLs and the results from doing an HTTP Get on the URL. All non-Drupal site URLs are tested. The Drupal URLs are a different matter.

The “Media” (for Images and PDFs) Dart code reads in the CSV from an Article or Project SQL script CSV file. It outputs a CSV that either contains Media entries or File entries; Media for a set of Image types and File for PDFs. The URLs are set to the known location of the corresponding Media file. Again there is a reject list and URLs are validated resulting in a Validate CSV output.

The added step, not mentioned above, is that the Media CSVs are loaded into Drupal using Feeds FIRST, BEFORE the “Generic” Dart code is run. The “Generic” Dart code does a lookup in the Drupal database for using the Media/File URL (really just the title) and gets the UUID for the Media/File. Then the “Generic” Dart code inserts the UUID as an attribute of the drupal-media tag (to display the Media/PDF).

One more step. Args to the “Generic” Dart code include the Drupal ViewMode to be assigned as an attribute of the drupal-media tag (“data-view-mode”). This allows all the Articles to have the same ViewMode assigned to their images. Then later, when there is a need to change the size of these images, you would only have to change the ViewMode configuration and not go back and modify each Article!

Drupal Feeds

Feeds is an add-on module. <https://www.drupal.org/project/feeds> Yes, it is “not covered by Drupal’s security advisory policy.” but it is used by 95,285 sites. In our case, it allows us to import a row of data from a CSV file, create a Content Item (Article, Project, Person, etc) and then map the CSV columns to Fields in the Content Item. Each CSV file column assignment can be altered slightly using some “Tamper” options.

There are “Feed Types” which define the mappings and tampers used with a content type. There are “Feeds” which define a CSV file input, the Feeds Type to use with it and how frequently it should be run; in our case the “Import period” should be “off” (manual).

Why choose the Dart language?

The code was written in Dart. Python, Ruby, PHP or even JavaScript might have been used instead. I chose Dart because I had written a fair amount of it, it could run as a standalone console app, there were good libraries for working with CSV files, and DOM code to make changes to HTML.

Visual Studio Code (VSCode) was the IDE of choice.

This is throw-away code

Once the data is loaded onto the site, there will be no more need for this software. It has not been written for maintainability, efficiency, ... it has been written quickly. As an example, in some cases it references globals rather than passing parameters; it was faster to code to the globals. Functions should have been written instead of the sections of code that were copy-pasted.

I am using Visual Studio Code as the IDE with the Dart language and compiler configured.

Configuration json file

The “Args” parameter being passed includes:

- Current Drupal server media folder path – When processing the CSV for media, it is the current year and month
- Input file path
- The media file extensions being processed from the input CSV file
- Output added file name text
- Hardcoded path to locate site blogs or projects (needed if http get URL validation is desired; optional)

Note: In the second, commented out case, only PDF files are being processed and the blog location isn’t needed.

```
launch.json
{
```

```

"version": "0.2.0",
"configurations": [
  {
    "name": "Dart",
    "program": "gg-media.dart",
    "request": "launch",
    "type": "dart",
    "args": ["2020-11/", "../input-output-files/Articles-from-SQL-002a.csv", "jpg, jpeg, png, tif, svg,
gif", "image-only.out", "http://192.168.1.9/test123/blog/"]
    // "args": ["2020-11/", "../input-output-files/Articles-from-SQL-002a.csv", "pdf", "file-only.out",
""]
  }
]
}

```

media.dart

```

import 'dart:io';
import 'dart:core';
import 'package:universal_html/driver.dart';
import 'package:universal_html/prefer_universal/html.dart' as html;
import '../test-lib/test-common.dart';

// *****
// This program extracts Media Library Images from the html-body field of the passed
// csv file. The output is another csv file with the Images to be
// loaded onto the target Drupal server.
//
// Note: Because Media Images now have Captions as part of their Content Type definition.
// This code processes Caption shortcodes, associating them to be part of an image where
// appropriate.
//
// Args:
// 0 - Current Drupal server media folder path – When processing the CSV for media, it is the current
// year and month.
// 1 - Input file path.
// 2 - The media file extensions being processed from the input CSV file.
// 3 - Output added file name text.
// 4 - Hardcoded path to locate site blogs
//
// Output 1 : Media : same name as input only with Arg[3] and out.csv
// Ourput 2 : Rejects : same name as input only with Arg[3] and out.reject.csv
//
// Note: Rejects are Media that match a known reject list.
//
// *****

```

```

void main(List<String> args) async {
  // "args": ["2020-11/", "../input-output-files/Articles-from-SQL-002a.csv", "jpg, jpeg, png, tif, svg, gif", "image-only.out", "http://192.168.1.9/test123/blog/"]
  curSubDir = args.elementAt(0);
  // Read in the CSV file into Lists.
  myCSVListOfLists = await readListOfLists(args.elementAt(1));
  var validExtensions = args.elementAt(2);
  var outputFileName = args.elementAt(3);
  baseForLinks = args.elementAt(4);

  // Update rejectListOfList
  rejectListOfList.insert(0, new List<String>());
  rejectListOfList[0].insert(rejectListOfList[0].length, "index");
  rejectIndexPos = rejectListOfList[0].length - 1;
  rejectListOfList[0].insert(rejectListOfList[0].length, "status");
  rejectStatusPos = rejectListOfList[0].length - 1;
  rejectListOfList[0].insert(rejectListOfList[0].length, "url");
  rejectUrlPos = rejectListOfList[0].length - 1;
  rejectListOfList[0].insert(rejectListOfList[0].length, "path");
  rejectPathPos = rejectListOfList[0].length - 1;
  rejectListOfList[0].insert(rejectListOfList[0].length, "exception");
  rejectExceptionPos = rejectListOfList[0].length - 1;
  rejectLoopIndex = 1;

  // Create our Media Lists for CSV output.
  myCSVImageListOfLists.insert(0, new List<String>());
  myCSVImageListOfLists[0].insert(imagePathPos, "image_path");
  myCSVImageListOfLists[0].insert(imageAltPos, "image_alt");
  myCSVImageListOfLists[0].insert(imageLinkPos, "image_link");
  myCSVImageListOfLists[0].insert(imageTitlePos, "image_title");
  myCSVImageListOfLists[0].insert(imageLinkComboPos, "image_link_combo");
  myCSVImageListOfLists[0].insert(imageFileNamePos, "image_file_name");
  myCSVImageListOfLists[0].insert(imagePostPos, "image_post");

  // Find a series of CSV column indexes that could be in different positions depending on the content
  type.
  for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'post_content') {
      htmlPos = i;
      break;
    }
  }

  for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'post_title') {
      titlePos = i;
      break;
    }
  }
}

```

```

for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'post_name') {
        postNamePos = i;
        break;
    }
}

int galleryListPos = -1;
for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'gallery_list') {
        galleryListPos = i;
        break;
    }
}
if (galleryListPos == -1) {
    print("No Gallery List detected");
}

int galleryAltFileNameListPos = -1;
for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'gallery_alt_file_name_list') {
        galleryAltFileNameListPos = i;
        break;
    }
}

int galleryAltTextListPos = -1;
for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'gallery_alt_text_list') {
        galleryAltTextListPos = i;
        break;
    }
}
// If we don't have an html content body... then this an error. Something is really wrong.
if (htmlPos == -1) {
    print(
        'A column with a label of post_content was not found. This column is needed for html to be
processed.');
```

processed.');

```

        exit(0);
    }

    // Walk through each CSV input row of data
    for (var i = 1; i < myCSVListOfLists.length; i++) {
        /*
        There are 6 different variations on shortcode Captions; with different image, anchor and text content.

        [caption id="attachment_15804" align="alignleft" width="3962"]<a
        href="http://www.test123.ca/wp/wp-content/uploads/2019/09/abc123-Toronto.png">
```

```

</a>
This map shows Toronto's ....[/caption]

```

The goal of the fixWPCaptionShortCodeToImageWithTitle function is to reformat the html as an IMG tag with the caption as a TITLE attribute. If an Anchor tag exists, then wrap the IMG with an ANCHOR.

```

    if (lnkStr == "") {
        print(
            '\n');
        return '';
    } else {
        print(
            '<a href="{lnkStr}"></a>\n');
        return '<a href="{lnkStr}"></a>';
    }
}

*/

print("processing [{myCSVListOfLists[i][titlePos]}]");
String str =
    fixWPCaptionShortCodeToImageWithTitle(myCSVListOfLists[i][htmlPos]);

// Parse the HTML Content Body into a DOM Document Fragment
String bodyStr = "<body>{str}</body>";
html.HtmlDocument doc = HtmlDriver.current.document;
final domParser = HtmlDriver.current.domParserDriver;
html.NodeValidator nodeValidator = testNodeValidatorInit();
final fragment = domParser.parseDocumentFragmentFromHtml(
    doc, bodyStr.trim(),
    validator: nodeValidator);

// <!-- wp:image {"id":16326,"sizeSlug":"large"} -->
// <figure class="size-large"></figure>
// <!-- /wp:image -->

// Figures with img tags processed to just be an img tag
await preProcessFigures(i, fragment);

// Note - Media URLs currently point to the old site or somewhere else

// <a tags with embdded img are changed to just be img tag with link attribute

```

```

await preProcessLinks(i, fragment);

// Some of the titles turn out to be numeric? Bug in CSV/database ...
var itPostTitle = myCSVListOfLists[i][titlePos].toString();

// Cycle through HTML <img> tags
// Remove Media Sizes, fix media paths
// If src media path is wp/wp-content/uploads/ then change to the /test-wp/wp/wp-content/output/
path
// (This requires converting the file paths into file names for the flat media folder)
// Handle special cases for pdf files and external path to import those.
// Handle 11 other special cases scenarios for the interpretation of media locations/paths.
// Check for base64 as invalid
// Fix the Title if it contains parts of the original media path, update it to the new path
// 31 odd cases hardcoded media path corrections. If X then Y.
// 65 hardcoded rejection paths. These are recorded in an output CSV but are left in the generic
// html output. They are recorded as rejected in media output but not output for Drupal Media
Import.

await processImages(validExtensions, i, fragment, itPostTitle.toString());

// Originally Gallery Media was part of the Article, Project, etc - Content type; loaded when the
// Content Type was loaded using Feeds. But now with Media
// Library, the media needs to be imported and then referenced as a list in the Content Type.
if (galleryListPos != -1) {
  await processGallery(validExtensions, i, myCSVListOfLists[i][galleryListPos],
myCSVListOfLists[i][galleryAltFileNameListPos] , myCSVListOfLists[i][galleryAltTextListPos] );
}

}
// Writes out data to be imported as Files or Media - CSV results.
String outputObject =
  args[1].split('.csv').first + '.' + outputFileName + '.csv';
await writeListofLists(myCSVImageListOfLists, outputObject);

// Writes out any rejection CSV results. (hardcoded rejection paths)
String rejectOutputObject =
  args[1].split('.csv').first + '.' + outputFileName + '.reject' + '.csv';
await writeListofLists(rejectListofList, rejectOutputObject);
}

```

generic.dart

```

import 'dart:io';
import 'dart:core';
import 'package:universal_html/driver.dart';

```

```

import 'package:universal_html/prefer_universal/html.dart' as html;
import '../test-lib/test-common.dart';
import 'package:mysql1/mysql1.dart';

// *****
// This program updates CSV columns, especially the HTML-Body Field.
// Captions, Anchor and Media that are being validated and transformed
// into what is needed for Drupal.
// - The Media is assumed to have been already imported to the Drupal site.
// - Drupal database lookups are made to get the UUID of the Media.
// - Links and Media in the HTML are updated to the new site.
// - Known mappings for Links and Media are updated.
// - Article (blog) and Project links are updated for the new site internal path.
//
// Note 1: Caption shortcodes are processed. Where associated with images, they are removed and
// added to the image content type as loaded separately with the media.dart code.
// Note 2: Reject links or Media paths are left in the HTML but documented in the Reject csv file.
//
// Args:
// 0 - Current Drupal server media folder path – When processing the CSV for media, it is the current
// year and month.
// 1 - Input file path.
// 2 - The media file extensions being processed from the input CSV file.
// 3 - Output added file name text.
// 4 - Hardcoded path to locate site blogs
//
// Output 1 : Article or Project Content Types : same name as input only with Arg[3] with out.csv
// Ourput 2 : Rejects : same name as input only with Arg[3] with out.reject.csv
//
// Note: Rejects are Media that match a known reject list.
//
// *****

```

```

void main(List<String> args) async {
  curSubDir = args[0];
  curViewMode = args[3];
  baseForLinks = args[2];

  // Drupal database connection
  var settings = new ConnectionSettings(
    host: '10.10.10.220',
    port: 3306,
    user: 'test',
    password: 'N3wsite',
    db: 'test123');

  // Update rejectListOfList in Common
  rejectListOfList.insert(0, new List<String>());
}

```

```

rejectListOfList[0].insert(rejectListOfList[0].length, "index");
rejectIndexPos = rejectListOfList[0].length - 1;
rejectListOfList[0].insert(rejectListOfList[0].length, "status");
rejectStatusPos = rejectListOfList[0].length - 1;
rejectListOfList[0].insert(rejectListOfList[0].length, "url");
rejectUrlPos = rejectListOfList[0].length - 1;
rejectListOfList[0].insert(rejectListOfList[0].length, "path");
rejectPathPos = rejectListOfList[0].length - 1;
rejectListOfList[0].insert(rejectListOfList[0].length, "exception");
rejectExceptionPos = rejectListOfList[0].length - 1;
rejectLoopIndex = 1;

```

```

// If validating images and links, then create a list of validation results
List<List<dynamic>> localListOfList = new List<List<dynamic>>();
localListOfList.insert(0, new List<String>());
localListOfList[0].insert(localListOfList[0].length, "index");
int validateIndexPos = localListOfList[0].length - 1;
localListOfList[0].insert(localListOfList[0].length, "status");
int validateStatusPos = localListOfList[0].length - 1;
localListOfList[0].insert(localListOfList[0].length, "url");
int validateUrlPos = localListOfList[0].length - 1;
localListOfList[0].insert(localListOfList[0].length, "path");
int validatePathPos = localListOfList[0].length - 1;
localListOfList[0].insert(localListOfList[0].length, "exception");
int validateExceptionPos = localListOfList[0].length - 1;
int loopIndex = 1;

```

```

// Update the Image list in Common
myCSVImageListOfLists.insert(0, new List<String>());
myCSVImageListOfLists[0].insert(imagePathPos, "image_path");
myCSVImageListOfLists[0].insert(imageAltPos, "image_alt");
myCSVImageListOfLists[0].insert(imageLinkPos, "image_link");
myCSVImageListOfLists[0].insert(imageTitlePos, "image_title");
myCSVImageListOfLists[0].insert(imageLinkComboPos, "image_link_combo");
myCSVImageListOfLists[0].insert(imageFileNamePos, "image_file_name");

```

```

// Setup the database connection
var conn = await MySqlConnection.connect(settings);

```

```

// Read in the CSV file.
myCSVListOfLists = await readListOfLists(args[1]);

```

```

// Find CSV column indexes that could be different between input files.
for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'post_content') {
        htmlPos = i;
        break;
    }
}

```

```

for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'post_title') {
        titlePos = i;
        break;
    }
}

for (int i = 1; i < myCSVListOfLists[0].length; i++) {
    if (myCSVListOfLists[0][i] == 'post_name') {
        postNamePos = i;
        break;
    }
}

// If we don't have an html content body... then this an error. Something is really wrong.
if (htmlPos == -1) {
    print(
        'A column with a label of post_content was not found. This column is needed for html to be
processed.');
```

processed.');

```

    exit(0);
}

// For the Generic code, we are taking the read in CSV columns and expanding them
myCSVListOfLists[0].insert(myCSVListOfLists[0].length, "extra_image_list");
int extraImageListPos = myCSVListOfLists[0].length - 1;
myCSVListOfLists[0].insert(myCSVListOfLists[0].length, "extra_title_list");
int extraTitleListPos = myCSVListOfLists[0].length - 1;
myCSVListOfLists[0].insert(myCSVListOfLists[0].length, "extra_alt_text_list");
int extraAltTextListPos = myCSVListOfLists[0].length - 1;
myCSVListOfLists[0]
    .insert(myCSVListOfLists[0].length, "extra_image_link_list");
int extraImageLinkListPos = myCSVListOfLists[0].length - 1;
myCSVListOfLists[0]
    .insert(myCSVListOfLists[0].length, "extra_link_image_title_list");
int extraLinkImageTitleListPos = myCSVListOfLists[0].length - 1;
myCSVListOfLists[0]
    .insert(myCSVListOfLists[0].length, "extra_file_name_list");
int extraFileNameListPos = myCSVListOfLists[0].length - 1;
myCSVListOfLists[0]
    .insert(myCSVListOfLists[0].length, "duplicate_content_body");
int dupContentPos = myCSVListOfLists[0].length - 1;

for (var i = 1; i < myCSVListOfLists.length; i++) {
    myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
    myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
    myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
    myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
    myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
}
```

```

myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
myCSVListOfLists[i].insert(myCSVListOfLists[i].length, "");
}

```

```

// Cycle through rows of CSV data
for (var i = 1; i < myCSVListOfLists.length; i++) {

```

```

/*

```

There are 6 different variations on shortcode Captions; with different image, anchor and text content.

```

[caption id="attachment_15804" align="alignleft" width="3962"]<a
href="http://www.test123.ca/wp/wp-content/uploads/2019/09/abc123-Toronto.png">
</a>
This map shows Toronto's ....[/caption]

```

The goal of the fixWPCaptionShortCodeToImageWithTitle function is to reformat the html as an IMG tag with the caption as a TITLE attribute. If an Anchor tag exists, then wrap the IMG with an ANCHOR.

```

if (lnkStr == "") {
    print(
        '\n');
    return '';
} else {
    print(
        '<a href="{lnkStr}"></a>\n');
    return '<a href="{lnkStr}"></a>';
}
*/

```

```

print("[${i}]processing[${myCSVListOfLists[i][titlePos]}]");
String str =
    fixWPCaptionShortCodeToImageWithTitle(myCSVListOfLists[i][htmlPos]);

```

```

// Initialize COMMON variables
extraImageListStr = "";
extraTitleListStr = "";
extraAltTextListStr = "";
extraImageLinkListStr = "";
extraLinkImageTitleListStr = "";
extraFileNameListStr = "";
listCount = 0;

```

```

// Parse the HTML Content Body into a DOM Document Fragment

```

```

String bodyStr = "<body>${str}</body>";
html.HtmlDocument doc = HtmlDriver.current.document;
final domParser = HtmlDriver.current.domParserDriver;
html.NodeValidator nodeValidator = testNodeValidatorInit();
final fragment = domParser.parseDocumentFragmentFromHtml(
    doc, bodyStr.trim(),
    validator: nodeValidator);

// <!-- wp:image {"id":16326,"sizeSlug":"large"} -->
// <figure class="size-large"></figure>
// <!-- /wp:image -->

// Figures with img tags processed to just be an img tag
preProcessFigures(i, fragment);
print("preProcessFigures done");

// Cycling through <a> anchors. We are removing the link and image elements and
// replacing with a div, anchor/link and image. Why? Because Drupal 8 needs the div
// for the anchor to work.
preProcessLinksAddDivs(i, fragment);
print("preProcessLinksAddDivs done");

// Somewhere, there is bad non-string (numeric) entry. We need to catch it to not throw an error.
var itPostTitle = myCSVListOfLists[i][title_pos].toString();

// <a tags with embdded img are changed to just be img tag with link attributes
// preProcessLinks(i, fragment);
// print("preProcessLinks done");

// Change all img tags, which have media images already on the drupal system, to Drupal-Media
tags.
// Add the looked up image UUID.
// Add all images to separate image lists.

// Cycle through HTML <img> tags
// Remove Media Sizes, fix media paths
// If src media path is wp/wp-content/uploads/ then change to the /test-wp/wp/wp-content/output/
path
// (This requires converting the file paths into file names for the flat media folder)
// Handle special cases for pdf files and external path to import those.
// Handle 11 other special cases scenarios for the interpretation of media locations/paths.
// Check for base64 as invalid
// Fix the Title if it contains parts of the original media path, update it to the new path
// 31 odd cases hardcoded media path corrections. If X then Y.
// 65 hardcoded rejection paths. These are recorded in an output CSV but are left in the generic

```

```
// html output. They are recorded as rejected in media output but not output for Drupal Media
Import.
```

```
// Because a database connection is being passed (conn), it is assumed that the media has already
// been loaded on the Drupal site and we are to look up the UUID and put it in the HTML.
```

```
// The "forExtra" part is Gallery images in Project Content Types. These are stored in fields delimited
// by an "|". Yes, the media.dart file now stores these a separate media entries for Media Library
// import but here we still need this data to be able to reference those media entities.
await processImagesForExtra("jpg, jpeg, png, tif, svg, pdf, gif", i, conn, fragment,
itPostTitle.toString());
print("processImagesForExtra done");
```

```
// Cycle through HTML <a> anchors
// Fix anchor link paths
// For Anchors with text that roughly matches the link, change the text to match the updated link,
// map anchors to blogs, projects and people to the updated Drupal paths on the new site.
// (This includes such database lookup uglies as http://test.ca/author/mradmin to
// http://test.ca/person/howard-thompson)
await processLinks(i, conn, fragment);
print("processLinks done");
```

```
// Assign that COMMON Gallery Data that was updated in function calls, above
myCSVListOfLists[i][extraImageListPos] = extraImageListStr;
myCSVListOfLists[i][extraTitleListPos] = extraTitleListStr;
myCSVListOfLists[i][extraAltTextListPos] = extraAltTextListStr;
myCSVListOfLists[i][extraImageLinkListPos] = extraImageLinkListStr;
myCSVListOfLists[i][extraLinkImageTitleListPos] =
    extraLinkImageTitleListStr;
myCSVListOfLists[i][extraFileNameListPos] =
    extraFileNameListStr;
```

```
// So far we left our media as an <img> tag. But we are using Drupal Media Library, so it has to
change.
```

```
// The following function is doing a number of Replace/RegExp calls change <img> to
// <drupal-media>.
// Fixing the opening IMG tag
// String newString = parm.replaceAll(
//   RegExp(r'<img data-entity-uuid', '<drupal-media class="TEST-WP-Processed-Image" data-
entity-uuid');
// Fixing the closing IMG tag
// newString = newString.replaceAll(
//   RegExp(r'data-entity-type="media">'), 'data-entity-type="media"></drupal-media>');
```

```
// Why didn't we just use "drupal-media" from the start, as we processed the HTML
// Content Bodies? Why use <img> first and RegExp convert it? I think it came down to
// "new html.ImageElement();". I could create a new img element within the DOM easily but I
// couldn't create a drupal-media tag because I would have had to write that new code. And getting it
// to work in the context of that html package / the DOM, it might not have been so easy. The
```

```
// RegExp calls were easier.
```

```
str = fragment.innerHTML;  
str = changeIMGIntoMediaEntity(str);
```

```
myCSVListOfLists[i][htmlPos] = str;
```

```
// Remove all the html tags for the alternate (Summary) body  
html.NodeValidator nodeValidatorNoTags = testNodeValidatorNoTagsInit();  
final fragmentNoTags = domParser.parseDocumentFragmentFromHtml(  
    doc, str.trim(),  
    validator: nodeValidatorNoTags);  
myCSVListOfLists[i][dupContentPos] = fragmentNoTags.innerHTML;
```

```
// launch.json parameter - if "baseForLinks" is an empty string then don't create validation CSV file.
```

```
// A late feature is validating Image and Link URLs. This is simply loading the URL in HTTP and  
// recording and errors. (Like 404s!)  
// The results are stored in a CSV file with ".val." in the output file name.
```

```
// "baseForLinks" is an Arg from our launch.json set of parameters.  
// It provides are real http path to Articles or Projects at the target Drupal site.  
// It is used in the output CSV file to provide a link to the Content Item being processed.  
// If say an image could not be loaded, then the customer can see the failure and click on the  
// link to the Content Item that has the error, and fix it.
```

```
// If there is no baseForLinks then we aren't doing validation and this can be skipped.
```

```
if (baseForLinks != "") {  
    String url = "Title =" + myCSVListOfLists[i][titlePos].toString();  
    if (myCSVListOfLists[i][postNamePos] is String) {  
        url = myCSVListOfLists[i][postNamePos];  
        url = baseForLinks + url;  
    }  
}
```

```
// Cycles through all <a> tags and uses the "href" attribute to call  
// var response = await http.get(tmpPath);  
loopIndex = await validateLinks(loopIndex, localListOfList, validateIndexPos, validateStatusPos,  
validateUrlPos, validatePathPos, validateExceptionPos,  
url, i, conn, fragment);
```

```
// Cycles through all <img> tags and uses the "src" attribute to call  
// var response = await http.get(tmpPath);  
loopIndex = await validateImages(loopIndex, localListOfList, validateIndexPos, validateStatusPos,  
validateUrlPos, validatePathPos, validateExceptionPos, url, i, conn, fragment);  
}  
  
}
```

```

// Writes out any validation CSV results.
String localOut = args[1].split('.csv').first + '.val.csv';
await writeListofLists(localListOfList, localOut);

// Writes out the updated Article or Project CSV file.
String out = args[1].split('.csv').first + '.out.csv';
await writeListofLists(myCSVListOfLists, out);

// Writes out any rejection CSV results. (hardcoded rejection paths)
String rejectOutputObject =
  args[1].split('.csv').first + '.out.reject' + '.csv';
await writeListofLists(rejectListOfList, rejectOutputObject);

print("CSV Just Written!!");
await conn.close();
}

```

common.dart

This is mostly String/HTML/DOM/Document/Fragment processing (validating html, adding and removing DOM document elements). There are hardcoded rejection lists and mappings of invalid paths to good paths; all specific to the customer's world. You can learn about most of this somewhere else. Instead I'm going to pick and choose things that I think you need to know. Or that I didn't find easy.

Packages

You need to process CSV files, parse HTML, get data from a MYSQL database, and make HTTP get calls to validate that URLs exist.

```

import 'package:csv/csv.dart';
import 'package:universal_html/prefer_universal/html.dart' as html;
import 'dart:async';
import 'package:mysql1/mysql1.dart';
import 'package:http/http.dart' as http;

```

Don't forget your pubspec.yaml file, make the changes to be able to use the packages.

```

name: my_app
dependencies:
  csv: ^4.0.2
  lists: ^0.1.6
  http:
  universal_html:
  mysql1:

```

Read and Write Lists of Lists to CSV files

```
// Read "fname" CSV file. Returns a "List of List" object
Future<List<List<dynamic>>> readListOfLists(String fname) async {

  final input = new File(fname).openRead();

  return await input

    .transform(utf8.decoder)

    .transform(new CsvToListConverter())

    .toList();
}

// Write out the "parm" list to "fname" file in CSV format
writeListOfLists(List<List> parm, String fname) async {

  print("[Add ${csvImageCount}] vs Written ${parm.length}");

  File file = await File(fname);

  String csv = const ListToCsvConverter().convert(parm);

  await file.writeAsString('$csv');
}
```

The Drupal Database lookup for the UUID using the Image Title

```
Future<String> getUUIDAndNameFromTitleInDB(int index, MySqlConnection conn, String title)
async {

  var results = await conn.query(

    'SELECT uuid, name FROM media_field_data INNER JOIN media ON
media_field_data.mid=media.mid WHERE name=\'${title}\'';

  ...

  if (results.length > 1) {
    print("[${index}]Warning: Multiple (${results.length}) Media Match ${title}");
    //must be a nicer way to do this.
    for (var row in results) {
      print("[${index}] (${row[0].toString()}) Drupal Media Match ${title}");
      return row[0]+ "," + row[1]; // uuid , name
    }
  }
}
```

Darn, Author links in WordPress use email address name vs Person links in Drupal use their full name

So, in WordPress, you have “<http://test.ca/author/mradmin/>”. That’s mradmin@test.ca
But in Drupal you need to update the link to be <http://test.ca/person/howard-thompson>

* You will need to call this as you are processing your HTML Content body <a> Anchor links.

```
Future<String> getPathToPersonFromEmailInDB(int index, MySqlConnection conn, String email)
async {
```

```
String fullEmail = email.replaceAll("/", "") + "@test.ca";
```

```
/* Cut paste from dbeaver
```

```
SELECT field_last_name_value, field_first_name_value FROM
  node__field_email_address INNER JOIN node__field_first_name ON
  node__field_email_address.entity_id=node__field_first_name.entity_id
  INNER JOIN node__field_last_name ON
  node__field_email_address.entity_id=node__field_last_name.entity_id
  WHERE field_email_address_value="mradmin@test.ca"
*/
```

```
var results = await conn.query(
  'SELECT field_first_name_value, field_last_name_value FROM node__field_email_address INNER
  JOIN node__field_first_name ON
  node__field_email_address.entity_id=node__field_first_name.entity_id INNER JOIN
  node__field_last_name ON node__field_email_address.entity_id=node__field_last_name.entity_id
  WHERE field_email_address_value="\${fullEmail}\"');
```

```
if (results.length == 0) {
  print("[${index}] Drupal Person Email Match Failed ${fullEmail}");
  return "/person/" + email + "/";
}
if (results.length > 1)
  print("[${index}]Warning: Multiple (${results.length}) Email Address Match ${fullEmail}");
//must be a nicer way to do this.
```

```
for (var row in results) {
  print("[${index}] (${row[0].toString()}) Drupal Email Match ${fullEmail}");
```

```
String tmp = row[0] + "-" + row[1];
tmp = tmp.toLowerCase();
```

```
  return "/person/" + tmp + "/";
  // return row[0]+ "," + row[1]; // firstname-lastname
```

```
}
print("[${index}] NO Drupal Email Match ${fullEmail}");
```

```
return "/person/" + email + "/";  
}
```

Our Dart code had problems with the <drupal-media> tag

There is a function called “changeIMGIntoMediaEntity” that is called at the end of our HTML Content Body processing. We used an tag throughout our DOM/Document processing and in the very end, used Replace/RegEx to replace our tag with <drupal-media>.

Why didn't we just use "drupal-media" from the start, as we processed the HTML Content Bodies? It came down to “new html.ImageElement();”. I could create a new img element within the DOM easily but I couldn't create a drupal-media tag because I would have had to write that new code. And getting it to work in the context of that html package / the DOM. It might not have been so easy. The RegExp calls were easier.

HTML/DOM/Document/Fragment manipulation

```
// We are removing the link and image elements, and replacing them with  
// a div, link and image.  
// Couldn't we just insert the div... ah maybe, I'm more sure on replacing complete elements.  
var div = new html.DivElement();  
div.className = "Coonelli-wp-image-migration";  
var link = new html.AnchorElement();  
var img = new html.ImageElement();  
  
img.src = imagePath;  
img.alt = altText;  
img.title = title;  
img.setAttribute("imagelink", imageLink);  
link.href = imageLink;  
link.append(img);  
div.append(link);  
elem.replaceWith(div);
```

Additional Feeds Runs

“Select related content” lists are applied to Articles and Projects through Feeds after the original Content Items are created. (These could not be added at creation time because the referenced related content might have not been created yet. It had to be done later in the process.)

Sometimes when you import data comes down to a “chicken or egg” question. This one is very much the case:

The Project Feature Images are created in the Media Library using Feeds AFTER the Media feeds are imported but BEFORE the import of the CSV file for Project Content Items. The Project Feature Image Feed uses the “Generic” output CSV file for Projects as an input because

it has the data in it for the creation of the Feature Image Media in the Media Library. Yes, this is the same CSV output that would be used to create the Projects Content Items.

****** The Project Feature Images are displayed as Media Library Images as part of the Project and not from within HTML, so there is no need to look up UUID or anything like that. The images just need to be loaded in the Media Library BEFORE the Projects are created.

Tips and Say what?

Chosen Database utility

Dbeaver worked great for running SQL and generating CSV data. (Yes, there is a free community version) <https://dbeaver.io/>

Linking Posts to Drupal User Authors

We manually put the Drupal User Ids into a newly created field in the WordPress database. These IDs were passed with Content Types (Articles and Projects) indicating their author.

Base64 image encoding

We did find Base64 images in the WordPress database. It was easier to convert these to jpg files. We loaded the image into a browser and saved it as a jpg, copying it into the image import flat folder. The WordPress database was changed to reference the jpg file.

(Why someone would Base64 encode images in the database? There did not appear to be a reason).

Feeds Tamper or not

There are times where you might try to modify a field's data before it becomes part of a Content Item during import. Often the answer is using Feeds Tamper. It can do basic string manipulation. Failing that, the solution is to fix the text using the Dart code or going back to the SQL which retrieved the data from the WordPress database.

Content Summary without HTML formatting

I tried to use Tamper to remove HTML tags to generate Summary text (without images, etc). The problem was that didn't always remove some tags. The Dart HTML NodeValidator was fairly easy to setup to remove all tags.

```
html.NodeValidator nodeValidatorNoTags = testNodeValidatorNoTagsInit();  
  
final fragmentNoTags = domParser.parseDocumentFragmentFromHtml(  
    doc, str.trim(),  
    validator: nodeValidatorNoTags);  
  
myCSVListOfLists[i][dupContentPos] = fragmentNoTags.innerHTML;
```

```
// *****

html.NodeValidator testNodeValidatorNoTagsInit() {

    return new html.NodeValidatorBuilder()

        ..allowTextElements();

}
```

Splitting a string field on a comma delimitation

I tried to use RegEx in Tamper to get everything before and after a comma delimiter. The RegEx was correct and validated online but it didn't work in Tamper. Given the columns weren't going to be modified in Dart as part of current processing, the decision was to separate the columns in the SQL. See the stored function section for SPLIT_STR.

Feeds Discoveries

1. You have to feed Feeds in chunks. 350 seemed to be our CVS row limit before things started to fail. So, we had to load our CSVs into Excel or whatever, and chunk them up for import into Drupal.
2. You MUST go under "Feed Type" Edit/Settings/Import period and set it to "Off"!!!! Feeds is designed to keep importing data on a cycle (every hour, week, etc). In our case, we wanted to just import manually whenever we ran it. If you don't turn this "Off"... wow... you will find you have a lot more data imported than you should have. And they will all be duplicates.
3. You can't have PathAuto turned on when importing using Feeds (trying to assign path alias' in Feeds). If you do have PathAuto On then Feeds will not be able to set the path alias.
4. Sometimes you do something a harder way until you finally notice how it should be done. I needed to apply different "Tamper" settings to a CSV column for two different targets. I just duplicated the field in SQL. I found out later that you can just use the same column name in Feeds Mapping to different targets; they will show up on different lines in Tamper, where you can modify them to be different.
5. If Feeds dies partway through importing a series of Content Items, you won't necessarily get an error message saying the Feed is "locked" but it could be. If you can't seem to get Feeds to run properly, look around for a "Locked State" added field with an "Unlock" button; it does exist.

Thumbnails for PDFs

"Media Thumbnails" and "Media Thumbnails PDF" modules do work but use ImageMagick locally. Not all shared server space allows for it to be installed and there were issues with the version of it that would work with Drupal 8. But if you are using Feeds, and in pre-production, you'll be adding and removing data from those CSV files a lot. If using these Thumbnail modules, there is some kind of bug where Feeds won't allow you to delete items; you select "Delete Items" and it fails with an error. You are left with a choice of either using Feeds or Thumbnails. Feeds wins; we didn't get to use thumbnails.

****I did create a module that would allow you to create thumbnails from other images already loaded in Drupal. So, if you had pineapple.jpg, then it could be used to generate a thumbnail for pineapple.pdf.**
<https://github.com/thomphow/drupal-thumbnail-mapper-module>

These are the instructions that worked to add ImageMagick with Drupal 8; several others didn't work.
<https://www.itzgeek.com/post/how-to-install-imagemagick-and-php-imagick-on-centos-8-rhel-8/>

Getting things done in Drupal 8

Sub-Theme ... Console errorstips

<https://www.drupal.org/docs/theming-drupal/creating-sub-themes>

Just load a few pages in Chrome and check the console debug screen. If you see some errors, part of it may be a couple of missing files in your sub-theme. Our theme was a sub-theme of the Stack Theme. I clued into errors related to files that I'd assumed would be "magically" picked up from the parent theme. I was wrong. I needed to copy over the files. Like what? The "fonts" folder with "iconsmind.ttf" and "iconsmind.woff" files. Stack's "js" folder. A sub-section of the Stack "css" folder content.

Some Modules we used

Display Suite – We didn't use Layout Builder. And we only used Display Suite to place Views inside the Display of Content Types as if the view was a field within the Content-Type.

<https://www.drupal.org/project/ds>

Better Exposed Filters – If you need to place a menu or set of radio buttons that allow the user to change the filter criteria for a View, then Better Exposed Filters is for you. But there is one small problem. Retaining the last filter value or selecting to not retain the last filter value, neither worked right for me. It wasn't random data. It was a previous filter value. Just not often the last one. But for some Views, it did keep the last one, correctly. Sigh. In the end, we couldn't auto-apply the filter because ... we ended up with an inconsistent UI experience. The closest we came to something "sane" was enabling an "Apply" button that the customer had to press after doing the radio button selection. Getting the user in the habit of "Choosing a value with a radio button and then clicking Apply" meant they were less likely to see that the last filter wasn't working right. I suspect this is only an issue for Radio Buttons but I haven't fully investigated. I'm praying someone fixes this.

Smart Trim – There were times we needed to trim content body text in slightly more complicated ways. This module was helpful.

Media – Parent Entity Link – See Media Library getting links to work right. (Say a gallery of images that you wanted to open as larger images, etc. By default those images would open to be admin edited.)

Metatag – Search Engine Optimization tags.

Exclude Node Title – There are situations where you don't want to see a content type's Title.

Media Delete media – Convenient button to delete Media Library items.

Media Bulk Upload – This is helpful but there is a problem. If I bulk upload xyz.jpg, the file name is xyz.jpg but it gets assigned a name of xyz as Media. If I upload xyz.jpg using the normal Media Library interface, then the file name is xyz.jpg and its name as Media is xyz.jpg.

This may not seem like a big issue but our module to map images to pdf files, generating thumbnails, relied on the name of the media being xyz.jpg and our pdf being xyz.pdf. So, bulk loading our images for our pdfs didn't work.

Automatic Entity Label – It was convenient to auto-generate a “Person” content-type “Title” using “[node:field_first_name] [node:field_last_name]”. This meant not having to have a “Person” “Title” field and also a “First Name” and “Last Name” field. (And having to explain why someone had to duplicate the data they were entering.)

Cron stopped working: Feeds and Tables.php in Core problem

Cron hadn't run for over a month on our site. So, none of our searches worked because none of the content had been indexed. After all, Cron hadn't run.

The problem was related to <https://www.drupal.org/project/feeds/issues/3074699>

Broadly reading around this ... in theory... the Tables.php was possibly missing an added check to see if \$field_storage exists. In the end, I modified the Core Tables.php code adding the check.

Cron now runs successfully but we'll see if there are other issues long term. Why hasn't this been fixed in Core? There was a “patch” but I don't think there ever was a real fix to a release. Then things come down to “Automatically closed – issue fixed for 2 weeks with no activity.”

Original Tables.php, getColumns eventually calls addField()

```
// Is this a field column?

    $columns = $field_storage->getColumns();
    if (isset($columns[$next]) || in_array($next, $table_mapping->getReservedColumns())) {
        // Use it.
        $sql_column = $table_mapping->getFieldColumnName($field_storage, $next);
        // Do not process it again.
        $key++;
    }
```

Updated Core Tables.php, wrapping the above code in “if (\$field_storage) {”

```
// Is this a field column?

// HT Added check after cron failure
if ($field_storage) {
    $columns = $field_storage->getColumns();
    if (isset($columns[$next]) || in_array($next, $table_mapping->getReservedColumns())) {
        // Use it.
        $sql_column = $table_mapping->getFieldColumnName($field_storage, $next);
        // Do not process it again.
```

```
$key++;  
}  
}
```

Search bar, options and not

I didn't like the default search bar button and the lack of ability to change the search hint text. I tried using https://www.drupal.org/project/simple_search_form instead. I wanted the search bar up with the main navigation menu, like the WordPress site. It worked for a while. Resizing the screen still always moved things so the search ended up in the middle of menu items as the screen was reduced. Then Simple Search Form stopped working. Looking back, I don't have notes on the exact failure. In the end, we went back to the default Search.

Drush – yes, you still need it.

I used Drush 10.2.2. Drush can be installed locally or globally. Our Drupal server was running on Centos 8 - <https://www.linode.com/docs/guides/how-to-install-drush-on-centos-8/>

Note: Drush no longer has an interface for updates, Composer is the “Official” method.

There are still some times where the only answer seems to be Drush

I remember we somehow got “stuck” in a mid-Drupal 8 upgrade with failures (maybe 8.9). We needed to flush our caches but no admin pages would load.

At a database level... we got strange errors.

```
MariaDB [test]> TRUNCATE cache_data;  
ERROR 1932 (42S02): Table 'test.cache_data' doesn't exist in engine  
MariaDB [test]> select * from cache_data;  
ERROR 1932 (42S02): Table 'test.cache_data' doesn't exist in engine  
MariaDB [test]>
```

Trying to clear the cache using Drush also failed.

After research, it came down to installing Drush locally.

Composer, Modules and Themes

I really hate to say that I used Composer but found myself doing manual updates too many times.

There was a really weird point where Navigation and footer regions, in block layout, stopped working. Reinstalling the Stack theme resolved the problem; our custom sub-theme is based on the Stack theme.

Media Library Discoveries

You need full HTML captions for your Media...oh dear

In the CKEditor, clicking on a piece of media in the rendered HTML... you'll see an "Edit Media" popup and a "Caption" toggle button. This inserts 'data-caption=" "' into the HTML (An empty Caption). You can edit this Attribute in the HTML Source and put in the desired text but it CANNOT contain HTML tags. (Yes, it would make more sense if the CKEditor allowed you to edit this without having to look at the HTML... and that you could add HTML tags! But no.)

```
<drupal-media data-align="center" data-entity-type="media" data-entity-uuid="90aeef27-98df-42fc-8553-8163fa4799ee" data-caption="" data-view-mode="test_article_default_display_mode"></drupal-media>
```

So, what do you do to get full HTML captions? I went to [admin/structure/media/manage/image/fields](#) and added a "Full HTML Caption" field to the Image Content-Type. Then under [/admin/structure/media/manage/image/display](#) I configured it to be displayed in the desired modes. Then from Feeds, when the Image was being created, I put in the Caption. Not ideal; it would be nicer if the Caption was in the HTML but this seemed to be my only option.

Turning an image into an Anchor Link is messed up in Drupal 8

This is a simple Media Library image.

```
<drupal-media class="Test-WP-Processed-Image" data-align="center" data-entity-type="media" data-entity-uuid="1e1fc527-ba85-45f8-8e0c-dc03f348f7ee" data-view-mode="test_project_default_display_mode"></drupal-media>
```

This is a Media Library image with an Anchor tag wrapping. It should work but it doesn't work in Drupal 8. Why? Ah ... https://www.drupal.org/project/anchor_link/issues/3065627 ... it doesn't work.

```
<a href="/test123/sites/default/files/2020-11/uploads-2010-07-img_0123.jpg">
<drupal-media class="Test-WP-Processed-Image" data-align="center" data-entity-type="media" data-entity-uuid="b4911806-9c70-48a1-bdd7-d1e015a4a4ee" data-view-mode="test_article_default_display_mode"></drupal-media>
</a>
```

Add a DIV tag around it and then it works. Why? I have no idea. It works.

```
<div class="test-wp-image-migration"><a href="/test123/sites/default/files/2020-11/uploads-2010-07-img_0123.jpg">
<drupal-media class="Test-WP-Processed-Image" data-align="center" data-entity-type="media" data-entity-uuid="b4911806-9c70-48a1-bdd7-d1e015a4a4ee" data-view-mode="test_article_default_display_mode"></drupal-media>
</a></div>
```

How to make adding that DIV less of a pain in CKEditor

I added a DIV icon to wrap the Anchor. A real solution would be to re-write the Anchor code integrating the DIV invisibly but I looked at the code and it wasn't going to be simple.

https://www.drupal.org/project/ckeditor_div_manager

Tip: If your DIV icon isn't showing up in CKEditor, debugged by looking at the Chrome/Console. Something could be in the wrong folder somewhere.

But adding a DIV tag wrapper also means that you have to remove the DIV to edit the Anchor! Ahhh!!! Drupal 8 is not a perfect world.

Thinking differently about Content-Type Fields and Media Library

Our Project Content-Type has a Gallery of Media; images and a few pdf files. These were originally a list of Images, imported using Feeds, directly into each Project Content Type item.

With Media Library, things changed. First, we had to import our Images and Files into the Media Library. They had to be two separate Feeds operations; one for Files and one for Images.

The Project Content Type Gallery was now a list of Media Library references ("Entity Reference"). Great. But I couldn't display them from within the Content-Type anymore; it just didn't work right. Instead, I had to create a View; displaying only the gallery list of the current Project based on URL. And then put the view in a Display Suite Field. And then place the Display Suite Field within Project Content-Type / Manage Display / desired Display mode.

So, working with Media Library and getting things to display properly in a list is more complicated.

The footer images in a Block didn't like CSS

I tried using the drupal-media class (test-footer-image) and the div class that wrapped it (test-footer-image). In the end, I had to put everything in the HTML.

```
<div class="column"><a class="test-footer-image" href="http://somesite.com/">
<drupal-media class="test-footer-image" data-align="center" data-entity-type="media" data-entity-
uuid="9afdb72e-76d9-47b5-8f05-dbcda5e3d3ee" data-view-
mode="test_footer_image_default_view_mode" onmouseout="this.style.opacity=0.6"
onmouseover="this.style.opacity=1.0" style="opacity:0.6"></drupal-media>
</a></div>
```

Sometimes inline CSS styles was what worked

```
<div class="column" style="margin-right: auto; ">
<div><a href="https://goo.gl/maps/RkCix8q643f8LjCMA">
<drupal-media data-align="center" data-entity-embed-
display="view_mode:node.test_contact_view_mode" data-entity-type="media" data-entity-
uuid="a06401ee-8e59-4d06-b866-456f59dc7fee"></drupal-media>
</a></div>
</div>
</div>
```

You might want to modify the Media Library Content search to include UUID

The problem is the HTML for an Image only references the UUID for the image in the Media Library. If you wanted to say replace it with a new image then you'd find the UUID of the new image and update the HTML. But then you'd need to find the old image to delete it. That's where it would be nice to search by UUID. It's simple to do. Just go to the Media Library Content Search Page (admin/content/media) and if logged in as an admin, find the edit button for the page (top right). Then find the pencil edit icon at the end of the Media horizontal list of fields; where we want to add UUID. Edit the view. Under "Fields", add the UUID.

"Link to Parent" is the magic phrase

So, you have a Drupal View of Articles with feature images that you want to be links to take you to the Article. By default, you are taken to the page to edit the Media Library Image! Wow. Not what you want. And off the shelf, there is no way to configure what you want. Really? You need this module. https://www.drupal.org/project/media_parent_entity_link

Gallery of Media Library Items, click to a larger view

You want the Image Thumbnails to open a larger view of the image; ideally being able to cycle through the images. For PDF file icons, you want to pop up a view of the PDF file.

By default with Media Library items, clicking an image icon puts you into editing the Image!

The latest Colorbox module seems to do the trick; Colorbox 8.x-1.4, Colorbox Field Formatter, and Colorbox Inline. <https://www.drupal.org/project/colorbox>

To change the Gallery image formatting, view a Content-Type with a gallery of images and click the edit icon of an image, and select "Manage Display" for the image. Select a Format of "Colorbox" and then adjust the settings using the config icon just to the right.

Note 1: Yes, you can do video media icons that click through to playing the video. The sizing of the icon and playing of the video is a bit of a challenge but it is possible.

Note 2: No, you can only cycle through images. PDFs and videos will not be in the cycle of images.

Stored Functions for the database

These are just a few stored procedures we used in our SQL scripting and references to their origins.

ExtractNumber : extract numbers out of an arbitrary string

<https://stackoverflow.com/questions/37268248/how-to-get-only-digits-from-string-in-mysql/37269038>

```
CREATE DEFINER=`WPtest`@`%` FUNCTION `test`.`ExtractNumber`(in_string VARCHAR(50))
RETURNS int(11)
    NO SQL
BEGIN
    DECLARE ctrNumber VARCHAR(50);
    DECLARE finNumber VARCHAR(50) DEFAULT "";
    DECLARE sChar VARCHAR(1);
    DECLARE inti INTEGER DEFAULT 1;

    IF LENGTH(in_string) > 0 THEN
        WHILE(inti <= LENGTH(in_string)) DO
            SET sChar = SUBSTRING(in_string, inti, 1);
            SET ctrNumber = FIND_IN_SET(sChar, '0,1,2,3,4,5,6,7,8,9');
            IF ctrNumber > 0 THEN
                SET finNumber = CONCAT(finNumber, sChar);
            END IF;
            SET inti = inti + 1;
        END WHILE;
        RETURN CAST(finNumber AS UNSIGNED);
    ELSE
        RETURN 0;
    END IF;
END
```

reg_replace: Replace through a regular expression.

* Note: If using MySQL 8.0+ you can use the built in REGEXP_REPLACE, which is what was done in the end.

<https://stackoverflow.com/questions/986826/how-to-do-a-regular-expression-replace-in-mysql/986870#986870>

```
CREATE DEFINER=`WPtest`@`%` FUNCTION `test`.`reg_replace`(subject VARCHAR(21845),
pattern VARCHAR(21845),
replacement VARCHAR(21845), greedy BOOLEAN, minMatchLen INT, maxMatchLen INT)
RETURNS varchar(21845) CHARSET latin1
    DETERMINISTIC
BEGIN
```

```

DECLARE result, subStr, usePattern VARCHAR(21845);
DECLARE startPos, prevStartPos, startInc, len, lenInc INT;
IF subject REGEXP pattern THEN
  SET result = "";
  -- Sanitize input parameter values
  SET minMatchLen = IF(minMatchLen < 1, 1, minMatchLen);
  SET maxMatchLen = IF(maxMatchLen < 1 OR maxMatchLen > CHAR_LENGTH(subject),
    CHAR_LENGTH(subject), maxMatchLen);
  -- Set the pattern to use to match an entire string rather than part of a string
  SET usePattern = IF (LEFT(pattern, 1) = '^', pattern, CONCAT('^', pattern));
  SET usePattern = IF (RIGHT(pattern, 1) = '$', usePattern, CONCAT(usePattern, '$'));
  -- Set start position to 1 if pattern starts with ^ or doesn't end with $.
  IF LEFT(pattern, 1) = '^' OR RIGHT(pattern, 1) <> '$' THEN
    SET startPos = 1, startInc = 1;
  -- Otherwise (i.e. pattern ends with $ but doesn't start with ^): Set start pos
  -- to the min or max match length from the end (depending on "greedy" flag).
  ELSEIF greedy THEN
    SET startPos = CHAR_LENGTH(subject) - maxMatchLen + 1, startInc = 1;
  ELSE
    SET startPos = CHAR_LENGTH(subject) - minMatchLen + 1, startInc = -1;
  END IF;
  WHILE startPos >= 1 AND startPos <= CHAR_LENGTH(subject)
    AND startPos + minMatchLen - 1 <= CHAR_LENGTH(subject)
    AND !(LEFT(pattern, 1) = '^' AND startPos <> 1)
    AND !(RIGHT(pattern, 1) = '$'
      AND startPos + maxMatchLen - 1 < CHAR_LENGTH(subject)) DO
    -- Set start length to maximum if matching greedily or pattern ends with $.
    -- Otherwise set starting length to the minimum match length.
    IF greedy OR RIGHT(pattern, 1) = '$' THEN
      SET len = LEAST(CHAR_LENGTH(subject) - startPos + 1, maxMatchLen), lenInc = -1;
    ELSE
      SET len = minMatchLen, lenInc = 1;
    END IF;
    SET prevStartPos = startPos;
    lenLoop: WHILE len >= 1 AND len <= maxMatchLen
      AND startPos + len - 1 <= CHAR_LENGTH(subject)
      AND !(RIGHT(pattern, 1) = '$'
        AND startPos + len - 1 <> CHAR_LENGTH(subject)) DO
      SET subStr = SUBSTRING(subject, startPos, len);
      IF subStr REGEXP usePattern THEN
        SET result = IF(startInc = 1,
          CONCAT(result, replacement), CONCAT(replacement, result));
        SET startPos = startPos + startInc * len;
        LEAVE lenLoop;
      END IF;
      SET len = len + lenInc;
    END WHILE;
    IF (startPos = prevStartPos) THEN
      SET result = IF(startInc = 1, CONCAT(result, SUBSTRING(subject, startPos, 1)),

```

```

        CONCAT(SUBSTRING(subject, startPos, 1), result));
    SET startPos = startPos + startInc;
END IF;
END WHILE;
IF startInc = 1 AND startPos <= CHAR_LENGTH(subject) THEN
    SET result = CONCAT(result, RIGHT(subject, CHAR_LENGTH(subject) + 1 - startPos));
ELSEIF startInc = -1 AND startPos >= 1 THEN
    SET result = CONCAT(LEFT(subject, startPos), result);
END IF;
ELSE
    SET result = subject;
END IF;
RETURN result;
END

```

SPLIT_STR : Splits a string by the delimiter, returning the desired position (before or after the delimiter).

<https://blog.fedecarg.com/2009/02/22/mysql-split-string-function/>

```

CREATE DEFINER='WPtest'@`%` FUNCTION `test`.`SPLIT_STR`(
  x VARCHAR(255),
  delim VARCHAR(12),
  pos INT
) RETURNS varchar(255) CHARSET latin1
  DETERMINISTIC
BEGIN
  RETURN REPLACE(SUBSTRING(SUBSTRING_INDEX(x, delim, pos),
    LENGTH(SUBSTRING_INDEX(x, delim, pos - 1)) + 1),
    delim, "");
END

```

Deleting Files on Drupal

When you delete a file in Drupal it still exists until the next garbage collection cycle; that could be hours away. If you want to delete the files immediately, you can run the following php code.

Remember to flush caches before and after.

```
>drush php-script del-unused.php
```

```

<?php
$result = db_query("SELECT fm.*
FROM {file_managed} AS fm
LEFT OUTER JOIN {file_usage} AS fu ON ( fm.fid = fu.fid )
LEFT OUTER JOIN {node} AS n ON ( fu.id = n.nid )
WHERE (fu.type = 'node' OR fu.type IS NULL) AND n.nid IS NULL
ORDER BY `fm`.`fid` DESC");

```

```

//Delete file & database entry
foreach ($result as $delta => $record) {

```

```
file_delete($record->fid);  
}  
?>
```

<https://www.drupal.org/forum/support/post-installation/2010-03-05/delete-unused-images>