wp2octopress.js

wp2octopress.js v0.0.1

https://github.com/geneticgrabbag/wp2octopress

Required Libraries

```
var util       = require('util'),
    fs         = require('fs'),
    wrench     = require('wrench')
    async      = require('async'),
    path       = require('path'),
    csv        = require('ya-csv'),
    toMarkdown = require('to-markdown').toMarkdown,
    slugs      = require('slugs')
    _          = require('underscore');
```

Source (WordPress) Data Mappings

Each source has the name of the file containing its WordPress content and a function that returns a hash of the desired output (Octopress) values.

```
var source = {
    posts: {
        filename: 'wp_posts.csv',
        parser:   function(data) {
            return {
                id:      data.ID,
                guid:    data.guid,
                title:   data.post_title,
                slug:    slugs(data.post_title),
                date:    data.post_date,
                content: toMarkdown(data.post_content),

                status:  data.post_status
            }
        }
    },
    users: {
        filename: 'wp_users.csv',
        parser:   function(data) {
            return {
                login:    data.user_login,
                friendly: data.display_name
```

Validate Command-Line Arguments

¶ Must pass exactly one argument, the name of a directory containing source (WordPress) CSV files.

```
                friendly: data.display_name
            }
        }
    },
    terms: {
        filename: 'wp_terms.csv',
        parser:   function(data) {
            return {
                id:   data.term_id,
                name: data.name,
                slug: data.slug
            }
        }
    },
    term_taxonomy: {
        filename: 'wp_term_taxonomy.csv',
        parser:   function(data) {
            return {
                tax_id:          data.term_taxonomy_id,
                term_id:         data.term_id,
                tax_type:        data.taxonomy,
                tax_description: data.description,
                tax_parent:      data.parent
            }
        }
    },
    term_relationships: {
        filename: 'wp_term_relationships.csv',
        parser:   function(data) {

            return {
                object_id: data.object_id,
                tax_id:    data.term_taxonomy_id
            }
        }
    }
};


var argv = process.argv;
var args = argv.slice(2)


if (args.length != 1) {
    console.error('Usage: node ' + path.basename(argv[1]) + " [directory]");
    process.exit(1);
}
var stats = fs.statSync(args[0]);
var dir_name = args[0];
if (! stats.isDirectory()) {
```

Read and Parse Source Data

Count of parsers awaiting to complete

Loop through each data source, creating a CSV reader and callbacks to add the source's values transformed into what's required for Octopress posts.

Hang around until all readers have parsed their data.

Callback to generate posts after all source data has been parsed.

Create a subdirectory to store generated posts.

Loops through posts, ignoring all but published ones.

```
    console.error("Path exists but is not a directory.");
    process.exit(2);
}


var keys = ["posts", "users", "terms", "term_taxonomy", "term_relationships"];
var readers = {};
var sourceValues = {};


var nWaiting = keys.length;


keys.forEach(function(key, iter) {
    var source = source[key];
    var reader = readers[key] = csv.createCsvFileReader(
        "./" + path.join('.', dir_name, source.filename),
        { columnsFromHeader: true }
        );
    var values = sourceValues[key] = [];
    reader.addListener('data', function(data) {
        values.push(source.parser(data));
        });
    reader.addListener('end', function() {
        nWaiting--;
        });
    });


async.whilst(
    function()    { return nWaiting > 0;   },
    function(cb)  { setTimeout(cb, 50);    },
    function(err) { onValuesGathered(err); }
    );


var onValuesGathered = function(err) {


    var export_dir = path.join(dir_name, '_posts');
    if (fs.existsSync(export_dir))
        wrench.rmdirSyncRecursive(export_dir);
    fs.mkdirSync(export_dir);


    _.chain(sourceValues.posts)
        .filter(function(post) { return post.status == 'publish' } )
        .each(function(post) {
```

Calculate an Octopress-compatible output file name.

```
        var ofname = path.join(
            export_dir,
            post.date.slice(0, 10) + '-' + post.slug + '.markdown'
            );
```

Find the terms associated with this post to use as categories.

```
        var tax_ids = _.chain(sourceValues.term_relationships)
            .filter(function(r) { return r.object_id == post.id })
            .pluck('tax_id')
            .value();
        var term_ids = _.chain(sourceValues.term_taxonomy)
            .filter(function(r) { return _.indexOf(tax_ids, r.tax_id) >= 0; })
            .pluck('term_id')
            .value();
        var categories = _.chain(sourceValues.terms)
            .filter(function(t) { return _.indexOf(term_ids, t.id) >= 0; })
            .pluck('slug')
            .value();
```

Output frontmatter.

```
        fs.appendFileSync(ofname, "---\n");
        fs.appendFileSync(ofname, "layout: post\n");
        fs.appendFileSync(ofname, "title: \"" + post.title + "\"\n");
        fs.appendFileSync(ofname, "date: " + post.date + "\n");
        fs.appendFileSync(ofname, "categories:\n");
        _.chain(categories)
```

Ignore uncategorized (not a real category!)

```
            .filter(function(category) { category != 'uncategorized' })
            .each(function(category) {
                fs.appendFileSync(ofname, "- " + category + "\n");
            });
        fs.appendFileSync(ofname, "---\n");
```

Output content (HTML converted to Markdown).

```
        fs.appendFileSync(ofname, post.content + "\n");
    });
};
```