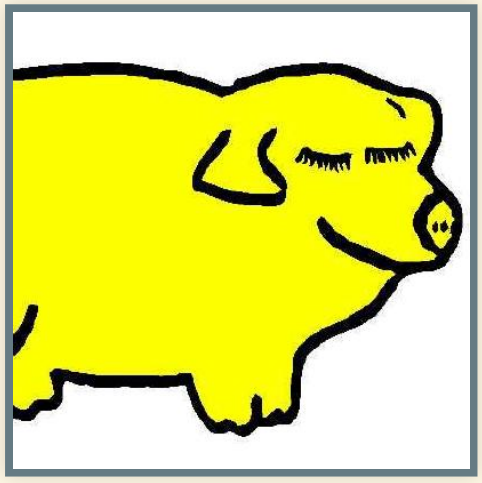
BUILD BACK BETTER - WITH THE MIGRATE API

Benji Fisher

May 27, 2022 - DrupalCamp NJ

INTRODUCTION

ABOUT ME



- Benji Fisher
- @benjifisher on d.o
- @benjifisher on GitHub
- @benjifisher on GitLab
- @benji17fisher on Twitter

Migration subsystem, Usability group, Security team (provisional member)

ABOUT FRUITION

FRUITION®

Build. Grow. Protect.

- Digital Marketing
- Website Design
- Development
- Security & Hosting

https://fruition.net/

FOLLOW ALONG

Find a link to this presentation on my GitLab Pages:

- https://slides.benjifisher.info/

OUTLINE

- Introduction
- Bringing Data Into Drupal
- Migrate API Introduction
- Examples
    1. Use Editor styles
    2. Structure unstructured content
    3. Update links in body text
    4. Update from Drupal 7 Media
- Conclusion

BRINGING DATA INTO DRUPAL

UPGRADING FROM DRUPAL 6 OR DRUPAL 7

I need to update my Drupal 6 site. Better late than never!

- Q: What tool will you use?
- A: The Migrate API

UPGRADING FROM OTHER SYSTEMS

I am getting tired of WordPress, but I have all these posts that I want to keep. How can I switch to Drupal?

- Q: What tool will you use?
- A: The Migrate API

RECURRING IMPORTS (FEEDS)

I need to create Drupal content every hour from an external Atom feed (or XML/JSON/SOAP/CSV)

- Q: What tool will you use?
- A: The Migrate API

Did anyone say “Feeds module”?

RESTRUCTURE A LIVE SITE

I need to change the structure of my live Drupal site: add/remove a field, move field data to linked Paragraphs, …

- Q: What tool will you use?
- A: It depends. Use the Migrate API if you
  - need to track old/new entity IDs
  - have complex dependencies
  - can use the tools it provides

MIGRATE API INTRODUCTION

ONE PROJECT, MANY MIGRATIONS

- Each migration has one source: SQL, CSV, XML, …
- Each migration creates one entity type:
  - node
  - media
  - taxonomy term
  - user
  - …
- A single XML file can be the source of several entity types, several migrations
- A site upgrade can have dozens of migrations

ONE MIGRATION, THREE STAGES

Three stages: Extract, Transform, Load (ETL)

- Extract (source plugin): one per migration
- Transform (process plugins): one or more per field/property
- Load (destination plugin): one per migration

Quiz: which stage is the most fun?

TRANSFORM/PROCESS: APPLY FILTERS

Filter pipelines:

- Bash: `git branch --merged | grep feature | xargs git branch -d`
- Twig: `list | map(item => item|lower) | join(', ')`

Each step gets its input from the previous one.

The Transform/Process stage of the Migrate API works the same way.

PROCESS PLUGINS

Drupal core and contrib modules provide many filters, or **process plugins**.

Most are configurable.

Learning to use them and combine them into pipelines takes some practice.

PROCESS PIPELINE (EXAMPLE)

The Migrate API uses YAML to describe pipelines.
(explanation of this example)

```yaml
process:
  field_formatted_text:
    - plugin: callback
      source: old_field
      callable: htmlentities
    - plugin: str_replace
      search: [' ', ' ']
      replace: ' '
    - plugin: callback
      callable: trim
```

DOM PROCESSING

Convert a text field (HTML string) to a DOMDocument object, process it, and save it as a string:

```
process:
  'body/value':
    - plugin: dom
      method: import
      source: 'body/0/value'
    # Other plugins do their work here.
    - plugin: dom
      method: export
```

The `body/0/value` bit is a short-cut. It is more complicated for multi-valued fields.

XPATH EXAMPLES

Use an XPath `selector` to identify one or more elements in a DOMDocument object:

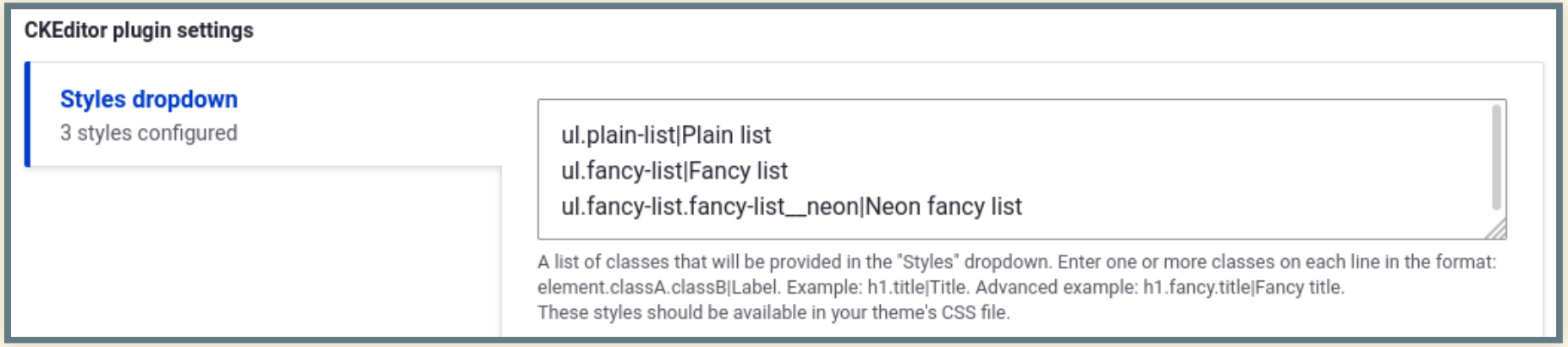
`selector`	Matches
`//a`	all <a> elements
`//a[class="external"]`	all <a> elements with `class="external"`
`//li[class="nav"]/a`	all <a> elements direct children of <li `class="nav">`

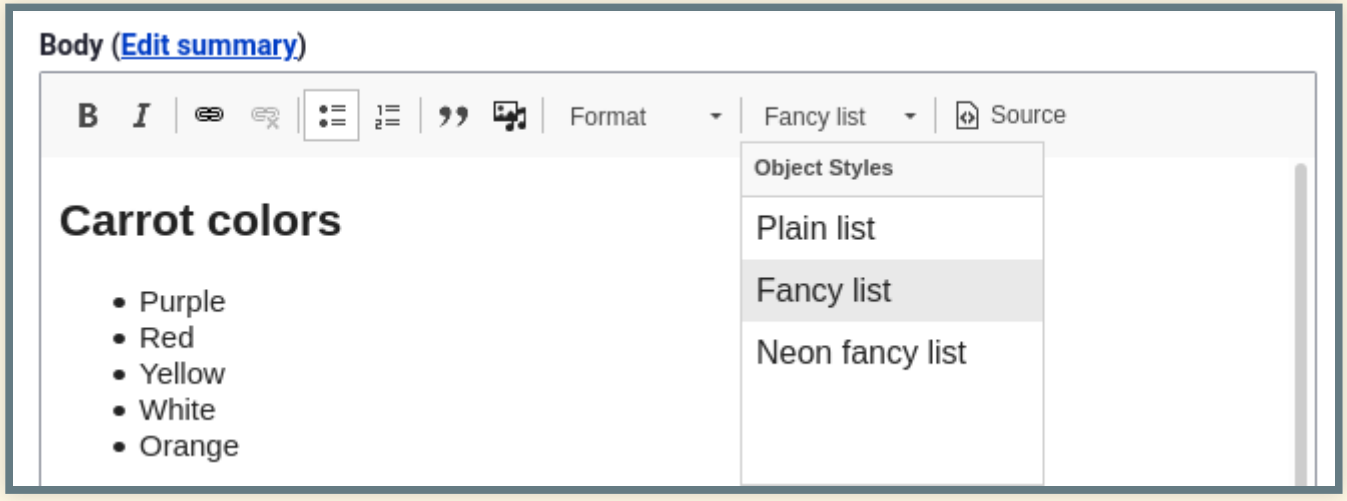
EXAMPLE: USE EDITOR STYLES

THE CHALLENGE

I have to import documentation pages from an external system. The documentation is formatted as HTML, but it does not have the magic CSS classes that my theme uses. How can I make it match the site style guide?

EDITOR STYLES





APPLY STYLES BASED ON XPATH

Let's hope the source HTML has some consistency. Then we can identify elements we want to style with an XPath expression and apply configured styles:

```
process:
  body/value:
    - plugin: dom
      method: import
      source: 'body/0/value'
    - plugin: dom_apply_styles
      format: basic_html
      rules:
        - xpath: '//ul'
          style: Fancy list
    - plugin: dom
      method: export
```

EXAMPLE: STRUCTURE UNSTRUCTURED CONTENT

THE CHALLENGE

Every Person page starts with a job title in an <h4> tag and a photo. How can I move those into separate fields, and keep the rest in the Body field?

Example:

```
<h4>Chief Assistant to the Assistant Chief</h4>
<img src="..." alt="...">
<p>Alfred E. Newman has been with Mad Magazine since ...</p>
```

STRUCTURE UNSTRUCTURED CONTENT: WHY?

- Q: Why is it better to have the job title and image in separate fields?
- Q: Were you planning to hide parts of the Body field with CSS?
- Q: Are People pages coming from a Drupal 7 site, WordPress, or an XML feed?

JOB TITLE IN A SEPARATE FIELD

When processing a Person page, use the `dom_select` plugin:

```
process:
  field_job_title:
    - plugin: dom
      source: body/0/value
      method: import
    - plugin: dom_select
      selector: '//h4'
      limit: 1
    - plugin: extract
      index:
        - 0
```

PHOTO IN A SEPARATE FIELD

Getting the photo is similar:

- Use `dom_select` with `selector: //img/@src`.
- Once you have the image URL, copy the file and make a File entity.
- Do all that in a separate migration.
- Use `migration_lookup` to get the File ID in the Person migration.

REMOVE ELEMENTS FROM BODY FIELD

Once the job title and photo are in separate fields, remove them from the Body field:

```
process:
  'body/value':
    - plugin: dom
      method: import
      source: 'body/0/value'
    - plugin: dom_remove
      selector: '//h4'
      limit: 1
    - plugin: dom_remove
      selector: '//img'
      limit: 1
    - plugin: dom
      method: export
```

EXAMPLE: UPDATE LINKS IN BODY TEXT

THE CHALLENGE

In my Drupal 7 site, "About us" was `/node/6`, but in the new site it is `/node/136`. A lot of Body fields have `<a href="/node/6">About Us</a>`. What can I do?

This is why Marco Villegas (`@marvil07`) and I wrote the DOM process plugins. Thanks to Isovera and Pega Systems for letting us donate the code to the Migrate Plus module.

UPDATE LINKS: LOOKUP

The Migrate API keeps track of source and destination IDs. Use `migration_lookup` to handle entity-reference fields:

```
process:
  field_related_content:
    - plugin: migration_lookup
      source: field_related_content
      migration:
        - this_migration
        - that_migration
```

Pause and reflect.

UPDATE LINKS: BODY FIELD

Use `dom_migration_lookup` to handle text fields:

```
process:
  body/value:
    - plugin: dom
      method: import
      source: 'body/0/value'
    - plugin: dom_migration_lookup
      mode: attribute
      xpath: '//a'
      attribute_options:
        name: href
      search: '@/node/(\d+)@'
      replace: '/node/[mapped-id]'
      migrations:
        - article
        - page
```

EXAMPLE: UPDATE FROM DRUPAL 7 MEDIA

THE CHALLENGE

I have a Drupal 7 site that uses the Media module. How do I migrate to Drupal 9?

CREATE FILES, THEN MEDIA

Standard migration: migrate files to files.

Custom migration: migrate files to media.

- Source (Extract): same as the standard migration
- Process (Transform): use `migration_lookup` to find file ID from first step
- Destination (Load): create Media, not Files

This works great for structured data (File fields).

MEDIA TOKENS

Media tokens in text fields (Media and WYSIWYG modules)

```
Look at my kitten photo:
[[{"type": "media", "fid": 1909, ... }]]
```

There's a module for that: Media Migration (alpha)

IMAGE TAGS

```
My kitten is even cuter!
<img src="/images/kitten.jpg" alt="cutest!" />
```

There's a module for that, too: Migrate Media Handler

```
process:
  'body/value':
    - plugin: dom
      method: import
      source: 'body/0/value'
    - plugin: dom_inline_doc_handler
    - plugin: dom_inline_image_handler
    - plugin: dom
      method: export
```

CONCLUSION

SUMMARY

- Introduction
- Bringing Data Into Drupal
- Migrate API Introduction
- Examples
    1. Use Editor styles
    2. Structure unstructured content
    3. Update links in body text
    4. Update from Drupal 7 Media
- Conclusion

WHAT'S NEXT

- Alternatives to `DOMDocument`
- Source plugin for JSON:API
- Source plugin for Drupal 7 field
- More granular processing of DOM nodes

REFERENCES

- Benji's slide decks
- Migrate API documentation on drupal.org
- Migrate Plus module home page
- Change record describing the DOMDocument-based plugins
- XPath documentation on MDN
- Process Pipelines on drupal.org
- Media Migration module
- Migrate Media Handler module

QUESTIONS

COPYLEFT