Easy Social Feeds with the Migrate API

DrupalCampNJ, Feb. 3, 2018
Intros

Tom Mount
- Technology Lead, Eastern Standard
- Closet geek
- Hobbies include bass guitar and rec football year-round
- Email: tomm@easternstandard.com

Eastern Standard
- Philadelphia-based marketing and technology agency
- Collaborative dev team
- We’re hiring!
What do our customers need?
Penn Biden Center (https://pennbidencenter.global.upenn.edu) homepage showing single Twitter feed
Quick Demo

PBC Homepage Component

PBC content list showing created nodes

PBC content editing of individual node
Quick Demo

PBC list of Twitter migrations

PBC editing Twitter migration
What is the Migrate API?
What is the Migrate API?

- Provides a Drupal-specific implementation of the ETL (Extract, Transform, Load) process.
  - *Extract*: pull data into a system
  - *Transform*: manipulate the data, or use the data to manipulate some other data
  - *Load*: Save the manipulated data somewhere else for use later or by another system
- This is a very synchronous process - *transforming* doesn’t happen until data is *extracted*, and *loading* can’t happen until the *transform* phase has completed.
- This is also not a real-time process; data is periodically retrieved and cached for later.
- At its simplest, the Migrate API provides a way of importing structured data from some source, processing it, and saving it somewhere else.
What is the Migrate API?

- The API uses slightly different terms:

<table>
<thead>
<tr>
<th>ETL Term</th>
<th>Migrate API Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract</td>
<td>Source</td>
</tr>
<tr>
<td>Transform</td>
<td>Process</td>
</tr>
<tr>
<td>Load</td>
<td>Destination</td>
</tr>
</tbody>
</table>

- Each of these API terms matches a *plugin* for the API.
- The plugin configures the pipeline for each step of the process.
What is the Migrate API?

- Core Source Plugins
  - Embedded Data Source: data that is included in the YAML configuration file
  - SQL data source: pull information from a database (but you have to roll the plugin yourself)

And that’s it! There aren’t many options for configuring source data in the core module.
What is the Migrate API?

- Core Process Plugins (not a complete list*)
  - Concat: allows multiple pieces of source data to be concatenated into one string
  - Default Value: allows the use of static text
  - Entity Exists: looks up an entity based on source data and returns the entity ID
  - Format Date: uses Drupal’s `DateTimePlus` class to convert dates between formats
  - Static Map: converts incoming source data to a different value
  - Subprocess/Iterator: processes structured data through its own pipeline

**Important to note:** in most cases, multiple process plugins can be called on a single piece of source data. The output from one plugin is automatically piped to the input of the next plugin.

* There are plenty of additional process plugins available; see [https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins/list-of-core-process-plugins](https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins/list-of-core-process-plugins) for a complete list.
What is the Migrate API?

- Core Destination Plugins
  - Config: places data into YAML config files
  - Entity: stores data in entities (the type of entity can be configured)

As with source plugins, there aren’t many destination options defined in the core module.
What is the Migrate API?

With all the plugins available, what can we do with what we're given out-of-the-box?

- Specify structured data manually (in YAML format only!), or pull in SQL data if we have time to write a custom plugin for our specific situation.
- Manipulate that data a bunch of different ways.
- Store the results in nodes.

Based on what we know about how the API works, what kinds of things should we be able to do?

- It might be cool to grab structured XML or JSON data off the underlying filesystem.
- Maybe we could grab that kind of data from another website instead?
- What if we could do something really crazy, like consume a third-party REST API, maybe from some really complex data source like Facebook, manipulate that data, then store that content in nodes?
Extending the Migrate API
# Extending the Migrate API

The **migrate_plus** module contains a collection of additional plugins for the *source*, *process*, and *destination* phases of the API.

## Source Plugins
- URL: using Drupal’s [GuzzleHttp](https://guzzle.com/) client, allows the use of a URL as a data source
- File/HTTP Data Fetchers
- JSON/XML/SOAP Data Parsers
- Basic/Digest/OAuth2 Authentication

## Process Plugins
- Entity Lookup/Generate: finds (or creates) entities based on source data
- Merge: merge several source fields into one
- StrReplace: modifies strings
- Skip On Value: bypasses processing on certain values

## Destination Plugins
- Table: allows for storing data in any database table, even if it’s not registered with Drupal’s Schema API
Extending the Migrate API

As an added bonus, the *migrate_plus* module makes two key changes to the Migrate API:

- Migrations as entities: now migrations can be managed as Drupal entities, exported as YAML files, etc.
- Migration Groups:
  - Allows migrations to be batched together and run as a group.
  - Allows migrations to share a base configuration, which can be omitted or overridden in individual migrations.
Extending the Migrate API

You can also build your own plugins. I created two process plugins for my `social_migration` module:

- **Coalesce**: takes a list of inputs and returns the first non-empty value in the list.
- **Permalink**: creates a Twitter permalink given the account name and a tweet ID.
Case Study: Importing Facebook Content
Case Study: Importing Facebook Content

1. Create a Drupal content type to hold all the information you want to import.
2. Configure source to use url source plugin with http data fetcher and oauth2 authentication.
   a. Primary source is the Graph API url and includes the desired fields.
   b. oauth2 plugin must be configured with API key from Facebook Developer.
   c. oauth2 plugin handles getting the token and applying it to the main call to the Graph API.
3. List all of the necessary fields within the source configuration.
4. Identify and configure an ID field in the data source to assist in caching.
5. Configure process plugin to assign source keys to the field names used in the content type, manipulating the data where necessary (e.g., truncating message value to 255 characters for the title field, converting the date to Drupal’s required format, adding descriptions to image URLs, etc.). Use the default_value plugin to set the node type and publishing status.
6. Set the destination plugin to entity:node to save the content as a node.
Case Study: Importing Facebook Content

- `url` source plugin
- `http` data fetcher plugin
- `oauth2` authentication plugin
- Since this is a shared configuration, the actual api keys for individual migrations do NOT appear here. They are supplied in the individual migrations.
- Pipeline of plugins for the title field: `coalesce` with a default value, followed by `substr` to reduce the result to 254 characters.
Case Study: Importing Facebook Content

- Assign values to Drupal content fields.
- Use the `format_date` plugin to transform a source date value into a different format.
Case Study: Importing Facebook Content

- Pretty easy to create your own plugin.
- This one takes one input, property_name, combines it with data from the source feed (id), and returns a Twitter permalink.
Case Study: Importing Facebook Content

Some potential next steps:

1. If more than one Facebook account should be retrieved, create a migration group and use the `shared_configuration` section to store configuration that would otherwise be duplicated with each Facebook migration.

2. Add fields on the content type to store metadata about the migration process in each created node (e.g., which migration generated the node). This is a great place to use taxonomies!

3. Create a module that...
   a. runs migrations on a cron;
   b. allows content managers or site owners to add or remove migrations; or
   c. specifies different permission levels so that larger organizations can control who can modify settings.
Case Study: Importing Facebook Content

Benefits of going this route:

1. Easy integration with Views and any sort of headless Drupal implementation you want.
2. Takes full advantage of Drupal's ability to cache content.
3. Allows non-developers to add or modify social media platforms and properties.
4. By default, the Migrate API won't re-import content that has already been imported, meaning content can be curated after it’s been imported and those changes will persist.
5. Fully compatible with content management workflows (e.g., Workbench).
6. Few to no third-party library dependencies.
7. Configuration is 100% compatible with Features or Configuration Export workflows (just be careful not to commit API keys to version control).
Further Reading
Further Reading

- Drupal 8 Migrate API docs: [https://www.drupal.org/docs/8/api/migrate-api/migrate-api-overview](https://www.drupal.org/docs/8/api/migrate-api/migrate-api-overview)
- [migrate_plus](https://www.drupal.org/project/migrate_plus) project page: [https://www.drupal.org/project/migrate_plus](https://www.drupal.org/project/migrate_plus) (adds Drush commands to run, rollback, and reset migrations)
- [migrate_tools](https://www.drupal.org/project/migrate_tools) project page: [https://www.drupal.org/project/migrate_tools](https://www.drupal.org/project/migrate_tools) (adds Drush commands to run, rollback, and reset migrations)
- "Migrating XML in Drupal 8" by Kelsey Bentham: [https://www.palantir.net/blog/migrating-xml-drupal-8](https://www.palantir.net/blog/migrating-xml-drupal-8) (one of two articles I used to figure out how to get the [social_migration](https://www.drupal.org/project/social_migration) module working properly)
- "Stop Waiting for Feeds Module: How to Import RSS in Drupal 8" by Campbell Vertesi: [https://ohthehugemanatee.org/blog/2017/06/07/stop-waiting-for-feeds-module-how-to-import-remote-feeds-in-drupal-8](https://ohthehugemanatee.org/blog/2017/06/07/stop-waiting-for-feeds-module-how-to-import-remote-feeds-in-drupal-8) (the second, and probably most helpful, article as I was learning about the Migrate API)
- [Eastern Standard Concepts blog post](https://www.easternstandardconcepts.com/blog/2017/06/07/stop-waiting-for-feeds-module-how-to-import-remote-feeds-in-drupal-8) which contains some more background information I didn’t include in this session.
- The home page for the [social_migration module](https://www.drupal.org/project/social_migration), now publicly available!
Questions