Getting Content into Drupal Using Migrate

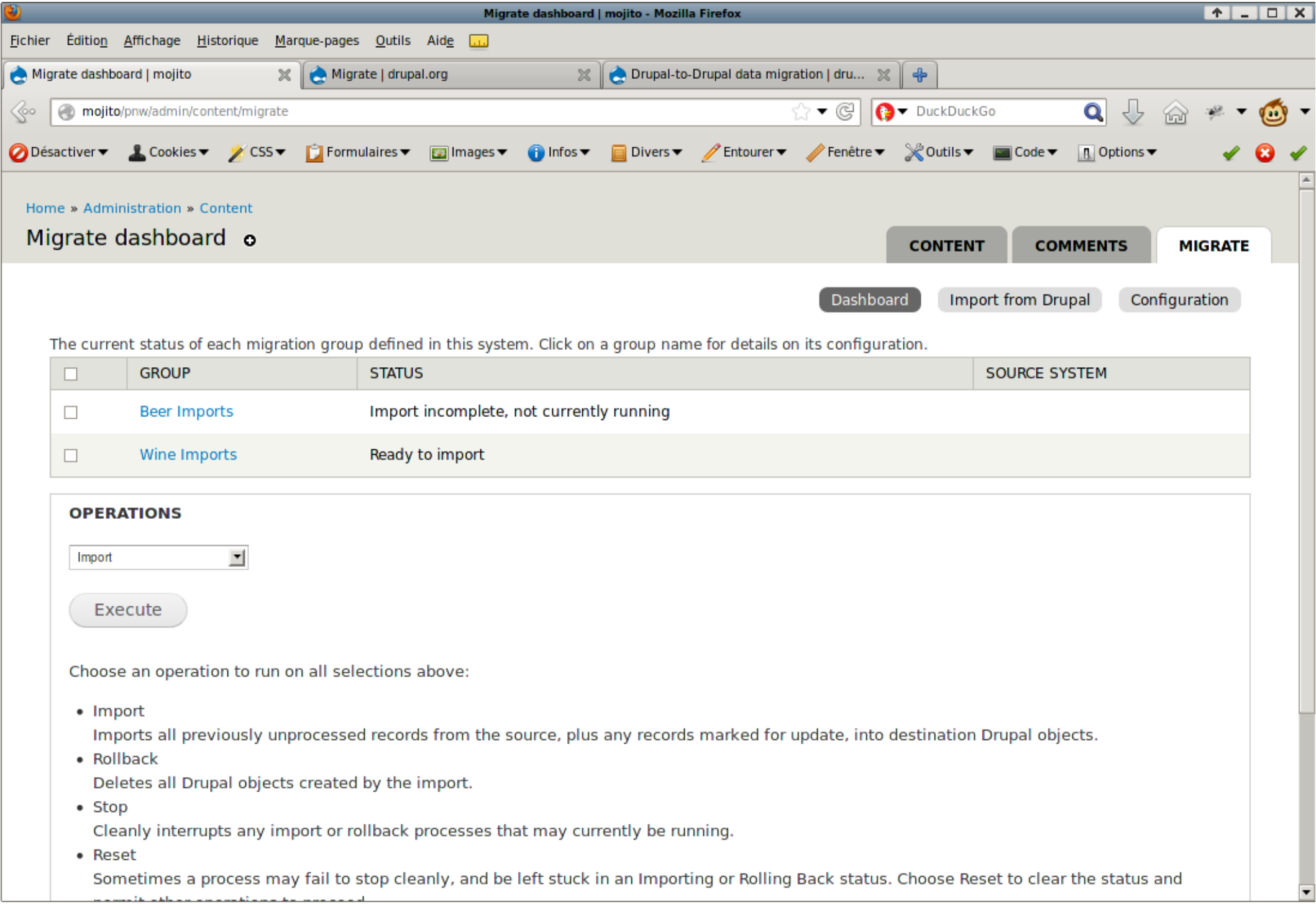
Version 0.3

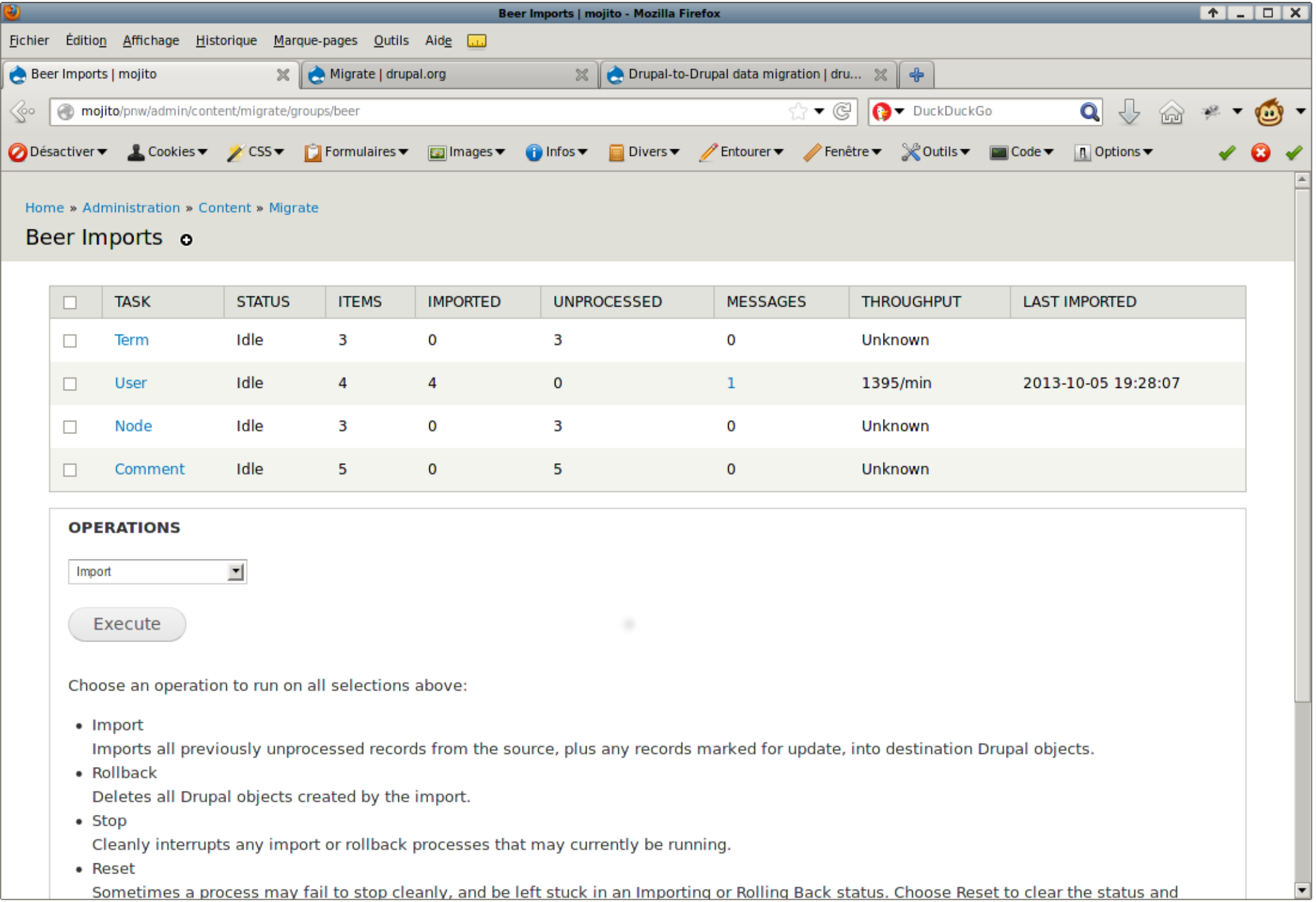
Ryan Weal
Kafei Interactive Inc.
Montréal QC

ryan@kafei.ca
Twitter : http://twitter.com/ryan_weal
Pump.io (open source, ftw!) :
http://comn.ca/ryanweal

What is Migrate?

- A pre-built framework for porting content to new Drupal sites
- A review structure that allows content administrators to see progress
- An alternative to upgrading old Drupal sites


Migrate dashboard | mojito - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
Migrate dashboard | mojito
Migrate | drupal.org
Drupal-to-Drupal data migration | dru...
mojito/pnw/admin/content/migrate
DuckDuckGo
Désactiver Cookies CSS Formulaires Images Infos Divers Entourer Fenêtre Outils Code Options
Home » Administration » Content
Migrate dashboard
CONTENT
COMMENTS
MIGRATE
Dashboard
Import from Drupal
Configuration
The current status of each migration group defined in this system. Click on a group name for details on its configuration.
GROUP
STATUS
SOURCE SYSTEM
Beer Imports
Import incomplete, not currently running
Wine Imports
Ready to import
OPERATIONS
Import
Execute
Choose an operation to run on all selections above:
Import
Imports all previously unprocessed records from the source, plus any records marked for update, into destination Drupal objects.
Rollback
Deletes all Drupal objects created by the import.
Stop
Cleanly interrupts any import or rollback processes that may currently be running.
Reset
Sometimes a process may fail to stop cleanly, and be left stuck in an Importing or Rolling Back status. Choose Reset to clear the status and


Beer Imports | mojito - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
Beer Imports | mojito
Migrate | drupal.org
Drupal-to-Drupal data migration | dru...
mojito/pnw/admin/content/migrate/groups/beer
DuckDuckGo
Désactiver Cookies CSS Formulaires Images Infos Divers Entourer Fenêtre Outils Code Options
Home » Administration » Content » Migrate
Beer Imports
TASK STATUS ITEMS IMPORTED UNPROCESSED MESSAGES THROUGHPUT LAST IMPORTED
Term Idle 3 0 3 0 Unknown
User Idle 4 4 0 1 1395/min 2013-10-05 19:28:07
Node Idle 3 0 3 0 Unknown
Comment Idle 5 0 5 0 Unknown
OPERATIONS
Import
Execute
Choose an operation to run on all selections above:
Import
Imports all previously unprocessed records from the source, plus any records marked for update, into destination Drupal objects.
Rollback
Deletes all Drupal objects created by the import.
Stop
Cleanly interrupts any import or rollback processes that may currently be running.
Reset
Sometimes a process may fail to stop cleanly, and be left stuck in an Importing or Rolling Back status. Choose Reset to clear the status and

Thinking of Upgrading Drupal?

- A long process, involving taking backup copies and occasionally *rolling back the entire site*
- Disable all of the modules
- Removing the theme, set to Garland
- Swap out the filesystem for D7
- Run the upgrade.php script, truncate your sessions and clear the caches (cross fingers)
- Unlucky? Delete or fix problematic variables, one by one
- Now all you have is Drupal Core (field data still exists, but no UI)

Upgrading the rest of Drupal

- Get the CCK module, enable content_migrate
- Find replacements to your field modules
- Run update.php every time you get a module, then run content_migrate for each field
- Be prepared to run custom queries to cleanup after bugs
- Now all you have is an upgraded site that is two weeks old

Upgrading post-snapshot data

- Feeds
  - Rejects duplicates, FUN!
  - Can mutate input with Feeds Tamper, with only 2700 clicks!
  - No way to easily capture errors
- Add XML : now you have two problems!
  - How about we use Regular Expressions to fix the XML?
  - Did anybody write down all the regexes we did?
  - Click-based mappings
- Temptation of regexes and the lost steps
- Oops, we screwed up. Let's use node_convert
  - does it support $x field?

How do we keep track of all of that?

- Upgrade core
- Upgrading all the fields
- New modules and themes
- Fixing bugs
- Extra cruft in the database
- Dealing with feature changes between versions

Still want to Upgrade, with Migrate?

- Reset your db counters on your tables (db-specific)
- Offset your migration to start at a certain number (highwater)

Skip upgrading, use Migrate!

- Migrate will do the dirty work
  - Developer will map the fields
  - Edge cases can be fixed in code
  - Migrate will do the rest
- New site development can start fresh (can use upgraded site)
- Rearchitect problematic areas from the old implementation
- Review your migrations with your team and content administrators

Migrate Benefits (1)

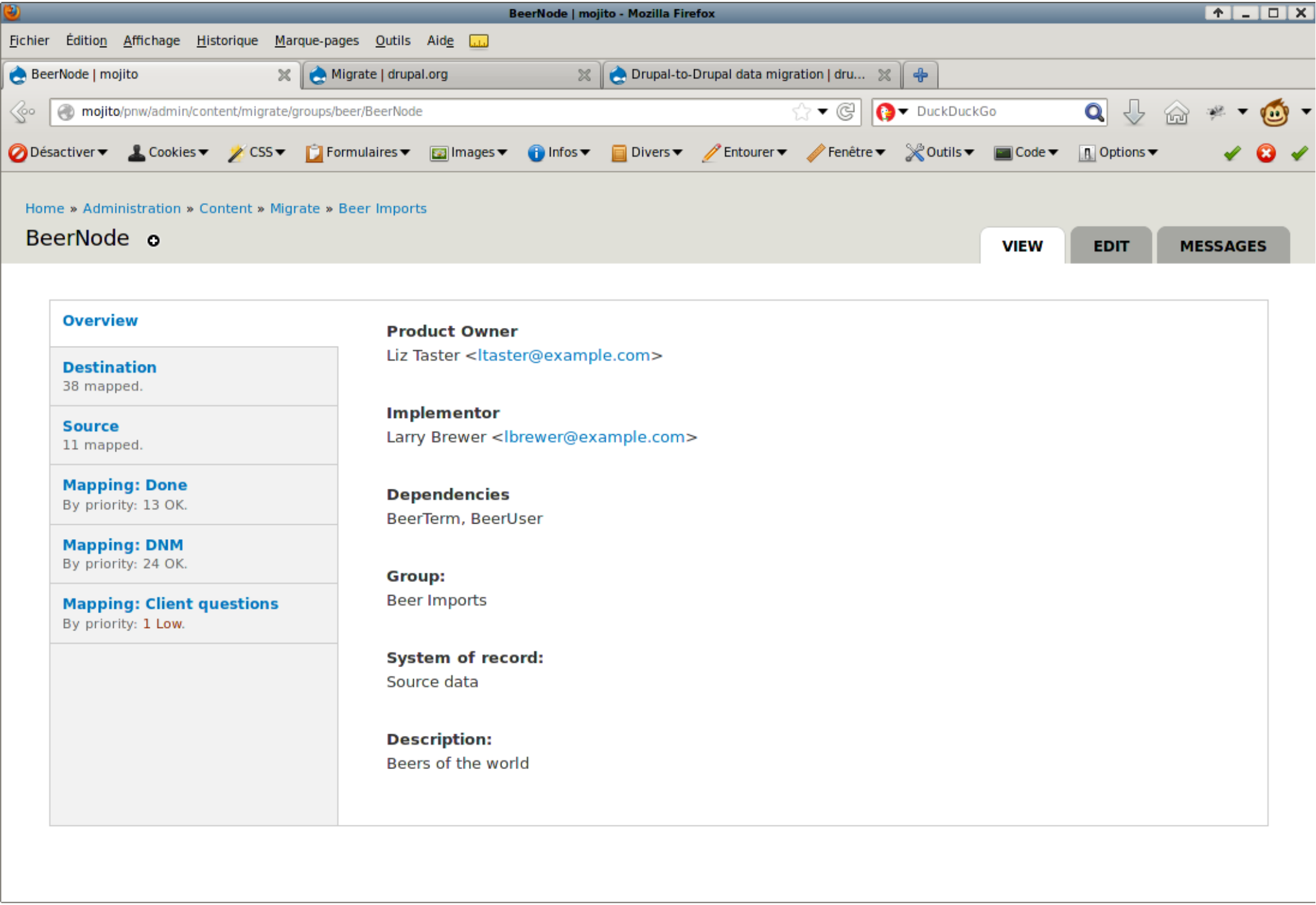
- Connects to almost anything :
  - MySQL / MariaDB
  - Postgres
  - MS SQL
  - Oracle
  - MongoDB, etc...
  - XML, CSV, etc...
    - Multiple-file imports (split files)

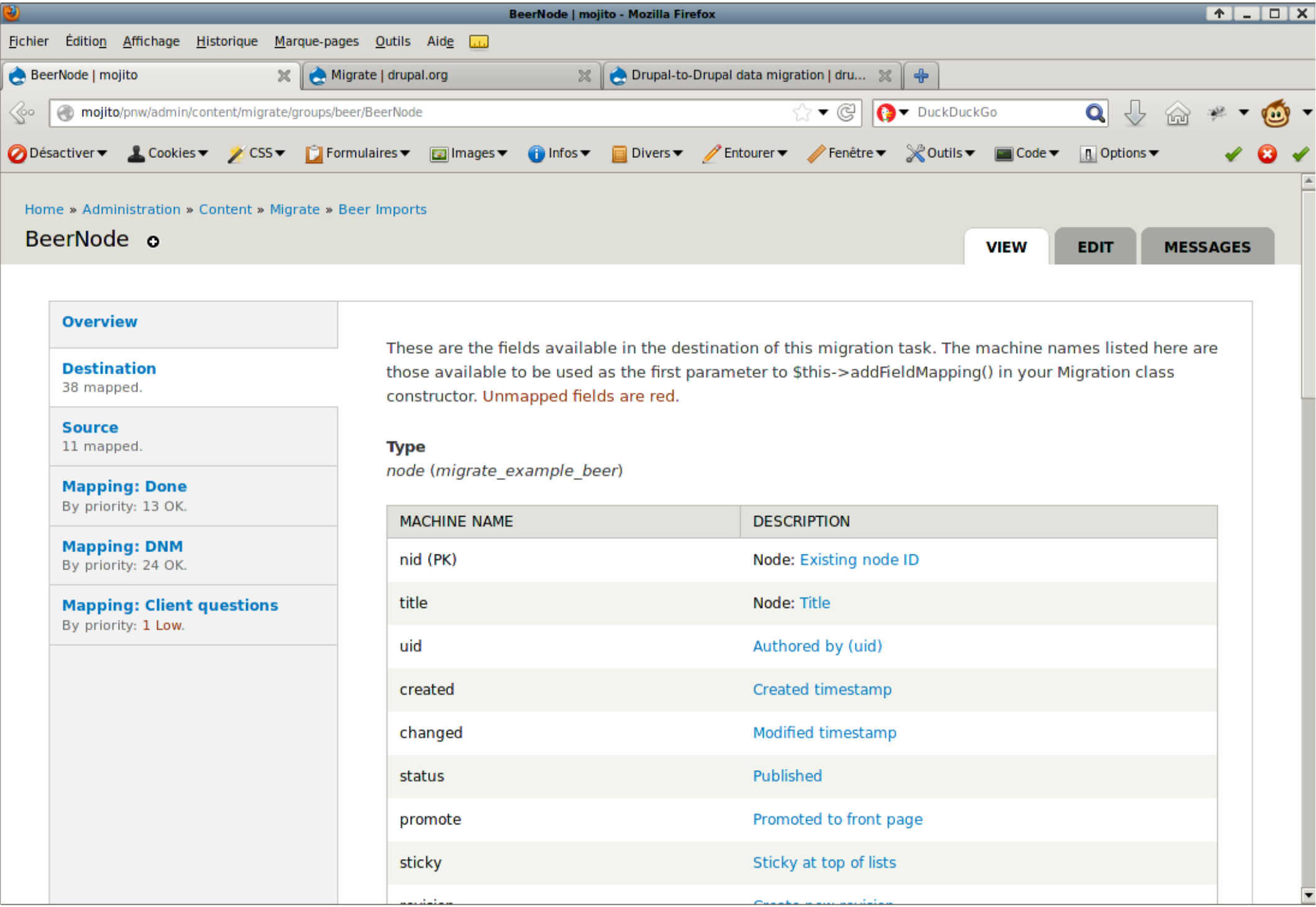
Migrate Benefits (2)

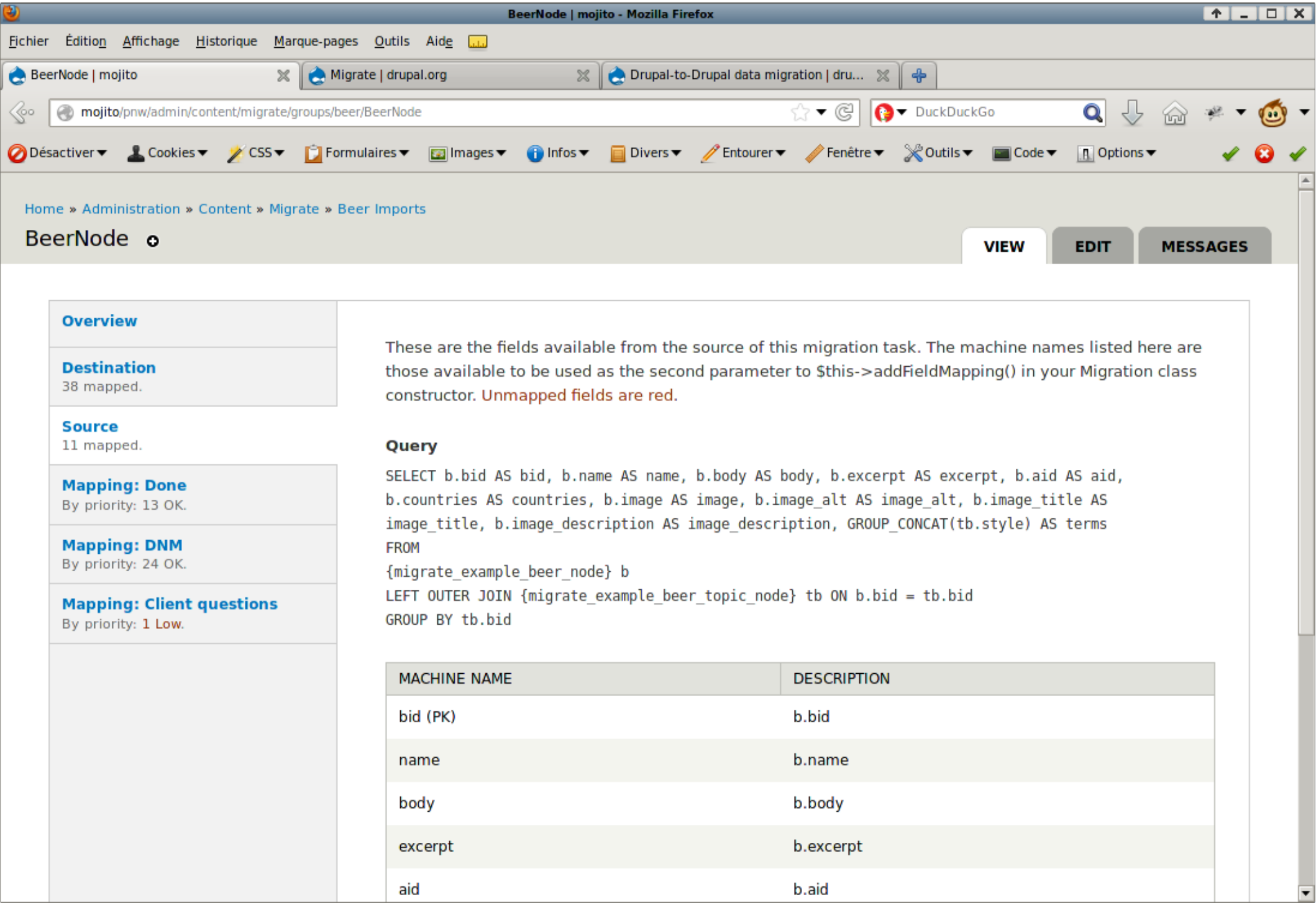
- Managed relationship between old and new content
  - Keeps track of NIDs, TIDs, UIDs, etc. between old and new systems
  - YES, you can preserve NIDs and UIDs!!!!! OMG YAY
- Runtime features :
  - Rollbacks
  - Incremental
  - Offsets / start points*
  - Memory management
  - Timeout prevention

Migrate Benefits (3)

- Review interface for content architect
  - Detailed listing of what old fields are mapped (or not!) to new fields
  - Statistics on how many items are to be migrated
  - Average time spent migrating each item (good for estimating large migrations)
  - Error reports for each item in the migration


BeerNode | mojito - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
BeerNode | mojito
Migrate | drupal.org
Drupal-to-Drupal data migration | dru...
mojito/pnw/admin/content/migrate/groups/beer/BeerNode
DuckDuckGo
Désactiver Cookies CSS Formulaires Images Infos Divers Entourer Fenêtre Outils Code Options
Home » Administration » Content » Migrate » Beer Imports
BeerNode
VIEW
EDIT
MESSAGES
Overview
Destination
38 mapped.
Source
11 mapped.
Mapping: Done
By priority: 13 OK.
Mapping: DNM
By priority: 24 OK.
Mapping: Client questions
By priority: 1 Low.
Product Owner
Liz Taster <ltaster@example.com>
Implementor
Larry Brewer <lbrewer@example.com>
Dependencies
BeerTerm, BeerUser
Group:
Beer Imports
System of record:
Source data
Description:
Beers of the world


BeerNode | mojito - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
BeerNode | mojito
Migrate | drupal.org
Drupal-to-Drupal data migration | dru...
mojito/pnw/admin/content/migrate/groups/beer/BeerNode
DuckDuckGo
Désactiver Cookies CSS Formulaires Images Infos Divers Entourer Fenêtre Outils Code Options
Home » Administration » Content » Migrate » Beer Imports
BeerNode
VIEW
EDIT
MESSAGES
Overview
Destination
38 mapped.
Source
11 mapped.
Mapping: Done
By priority: 13 OK.
Mapping: DNM
By priority: 24 OK.
Mapping: Client questions
By priority: 1 Low.
These are the fields available in the destination of this migration task. The machine names listed here are those available to be used as the first parameter to $this->addFieldMapping() in your Migration class constructor. Unmapped fields are red.
Type
node (migrate_example_beer)
MACHINE NAME
DESCRIPTION
nid (PK)
Node: Existing node ID
title
Node: Title
uid
Authored by (uid)
created
Created timestamp
changed
Modified timestamp
status
Published
promote
Promoted to front page
sticky
Sticky at top of lists


BeerNode | mojito - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
BeerNode | mojito
Migrate | drupal.org
Drupal-to-Drupal data migration | dru...
mojito/pnw/admin/content/migrate/groups/beer/BeerNode
DuckDuckGo
Désactiver Cookies CSS Formulaires Images Infos Divers Entourer Fenêtre Outils Code Options
Home » Administration » Content » Migrate » Beer Imports
BeerNode
VIEW
EDIT
MESSAGES
Overview
Destination
38 mapped.
Source
11 mapped.
Mapping: Done
By priority: 13 OK.
Mapping: DNM
By priority: 24 OK.
Mapping: Client questions
By priority: 1 Low.
These are the fields available from the source of this migration task. The machine names listed here are those available to be used as the second parameter to $this->addFieldMapping() in your Migration class constructor. Unmapped fields are red.
Query
SELECT b.bid AS bid, b.name AS name, b.body AS body, b.excerpt AS excerpt, b.aid AS aid,
b.countries AS countries, b.image AS image, b.image_alt AS image_alt, b.image_title AS
image_title, b.image_description AS image_description, GROUP_CONCAT(tb.style) AS terms
FROM
{migrate_example_beer_node} b
LEFT OUTER JOIN {migrate_example_beer_topic_node} tb ON b.bid = tb.bid
GROUP BY tb.bid
MACHINE NAME
DESCRIPTION
bid (PK)
b.bid
name
b.name
body
b.body
excerpt
b.excerpt
aid
b.aid

Migrate Benefits (4)

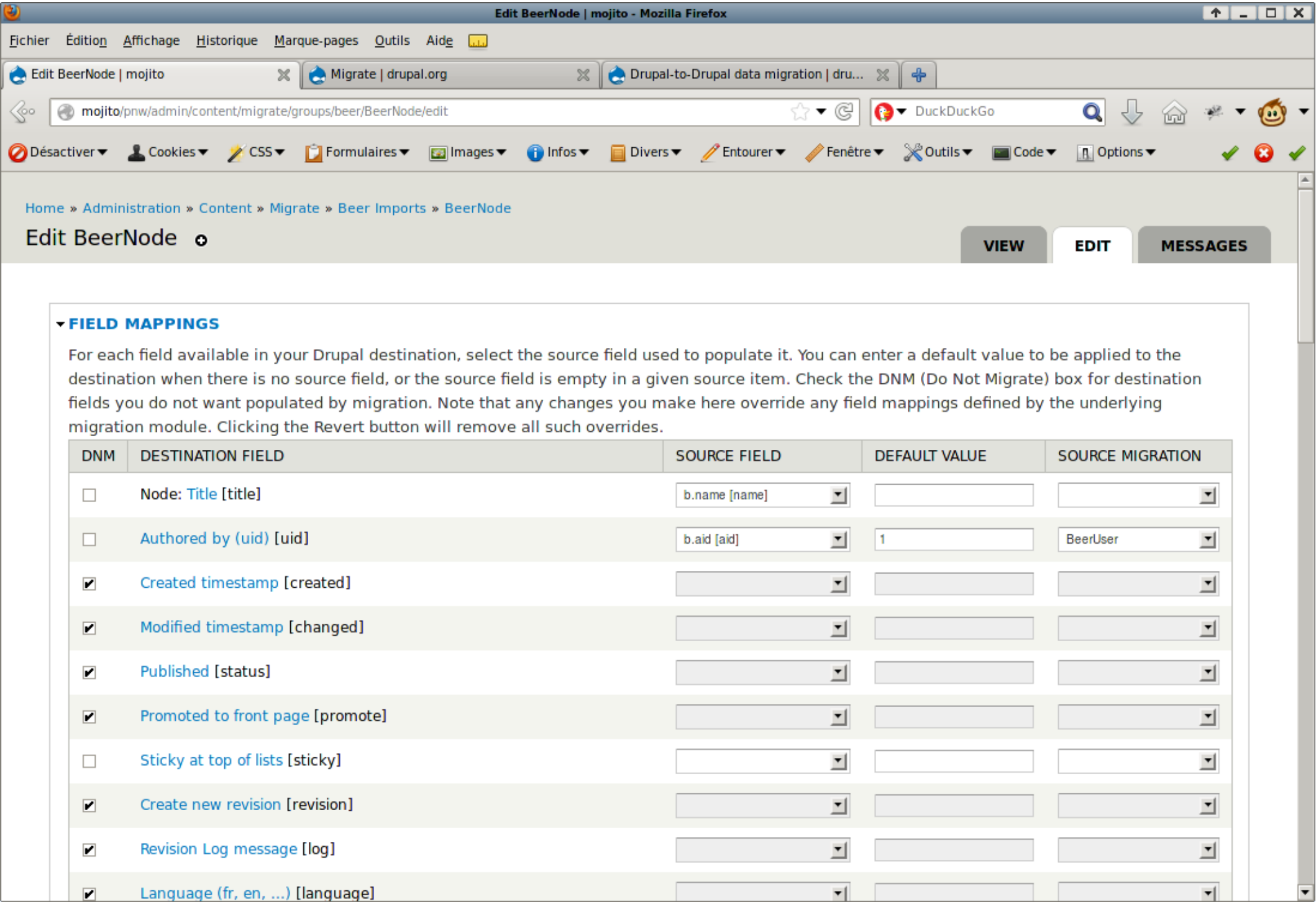
- Automate like crazy using Drush
  - Grouped migrations
  - Ordered migrations
- Deploy fresh content on a schedule
- Build around real data, demonstrate real sites
- Delete the migrate modules when you are done (not dependant)

Migrate Benefits (5)

- The future of Drupal
  - The migrate_d2d package was created as a proof of concept for importing from old drupal sites
  - Was flagged for D8, then D9, and now back to D8!
  - Generally works better than upgrading, more reporting data, less error prone, etc.

Drupal 2 Drupal migration!

- Collection of templates for Drupal Core, versions 5, 6 and 7
- Nodes, taxonomies, users, roles, comments, etc.
  - Basic settings already mapped
  - Published, sticky, etc...
- The dev version offers a UI based approach!
  - You can edit custom coded modules also!


Edit BeerNode | mojito - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
Edit BeerNode | mojito
Migrate | drupal.org
Drupal-to-Drupal data migration | dru...
mojito/pnw/admin/content/migrate/groups/beer/BeerNode/edit
DuckDuckGo
Désactiver Cookies CSS Formulaires Images Infos Divers Entourer Fenêtre Outils Code Options
Home » Administration » Content » Migrate » Beer Imports » BeerNode
Edit BeerNode
VIEW
EDIT
MESSAGES
FIELD MAPPINGS
For each field available in your Drupal destination, select the source field used to populate it. You can enter a default value to be applied to the destination when there is no source field, or the source field is empty in a given source item. Check the DNM (Do Not Migrate) box for destination fields you do not want populated by migration. Note that any changes you make here override any field mappings defined by the underlying migration module. Clicking the Revert button will remove all such overrides.
DNM
DESTINATION FIELD
SOURCE FIELD
DEFAULT VALUE
SOURCE MIGRATION
Node: Title [title]
b.name [name]
Authored by (uid) [uid]
b.aid [aid]
1
BeerUser
Created timestamp [created]
Modified timestamp [changed]
Published [status]
Promoted to front page [promote]
Sticky at top of lists [sticky]
Create new revision [revision]
Revision Log message [log]
Language (fr, en, ...) [language]

Migration in a Nutshell

- The process is much simplified compared to an upgrade :
  - Install Drupal
  - Add the migrate module
  - Add modules that contain base classes you want
  - Create your custom-coded migration classes to extend the templates
  - Connect to your old database(s)
  - Run the migration templates in bulk
- Roll up the entire process as an install profile?

How Do We Do This?

- Get the migrate module
- If upgrading Drupal, get migrate_d2d also
  - If you want UI, use 2.6x branch of migrate.
  - The Commerce Übercart Migration currently supports only 2.4x and 2.5x of migrate.
- Create a new custom module
- Put your OOP hat on
- Copy source DB and files to the same host as D7

Binding to Source

- Preferred route is secondary database entry in settings.php
- UI-based migrations will have a configuration page
  - migrate_d2d 2.6 UI database settings cannot be edited yet
    - You can truncate migrate_status table or overwrite DB settings to retain mappings
- XML and other sources can be bound later in one of our classes

Your Custom Module

- kafei_migrate.info
  - List out all of your files[] here
- kafei_migrate.migration.inc
  - Register your groups and classes here
- kafei_migrate.module
  - Helper functions (no migrate logic)
- user.inc, node.inc
  - Mapping of old fields to new fields
  - Each node type will probably get a separate class
- file.inc, menu_links.inc
  - More mappings, if needing customization (d2d mostly automatic)
  - Different handlers for public/private

Class Registration – How (1)

- THREE different ways to do this (two are legit) :
  - Best way is to put everything into kafei_migrate.migration.inc (2.6)
    - hook_migrate_api()
    - Find content > migrate > config > register
    - Protip : call register function in hook_install
  - Second way is to instantiate a new instance of a class (2.5)
    - Find content > migrate > config > register
    - Clear all caches (need custom hook for this from d2d docs)
  - Third way is depreciated and should be avoided at all costs. Involves $migrations[] array from hell. Only works in 2.4 and 2.5.
    - Do a rain dance
    - Goat sacrifice needed

Class Registration - Tips (2)

- Groups need to be instantiated first
- Individual classes for each type of data you wish to move across
  - Requires LOTS of parameters
  - Look at DDO migrate_d2d docs, even if you do not use d2d! It has all the needed params.
  - Adapt the parameters to work with the kafei_migrate.migration.inc structure oulined below :

Class Registration – Code (3)

```
/**
 * Class registration!
 */
$api = array(
  'api' => 2,
  'groups' => array(
    'kafei' => array(
      'title' => t('Kafei Migrations'),
    ),
  ),
  'migrations' => array(
    'KafeiUser' => $user_arguments,
    'KafeiFile' => $public_file_arguments,
    'KafeiMenu' => $menu_arguments,
    'KafeiMenuLinks' => $menu_link_arguments,
  ),
);
```

Class Registration – Arguments (4)

```
$common_arguments = array(
  'source_connection' => 'd6', // name of secondary DB in settings.php
  'source_version'=> 6,
  'group_name' => 'kafei',
  'paths' => $paths, // CUSTOM (only gets run once)
  'format_mappings' => array(
    '1' => 'filtered_html', // site manager
  ),
);
$user_arguments = $common_arguments + array(
  'machine_name' => 'KafeiUser',
  'description' => t('Import Drupal 6 users'),
  'class_name' => 'KafeiUser',
  //'role_migration' => 'KafeiRole',
);
```

Custom Classes – Basics (1)

- Put your OOP hat on!
- A class will define one set of data. Usually one for each type of node, one for each entity type
- *Extend* an existing class
  - Migration class, if using just the base module
  - DrupalUser6Migration, if using D2D with a D6 site
- Where are my arguments???
  - In the __*construct* function within your class extension
  - Inherit your arguments from the class you extended

Custom Classes - Construct (2)

```
class KafeiUser extends DrupalUser6Migration {
  public function __construct(array $arguments) {
    parent::__construct($arguments); // make use of extending that class!
  }
}
```

Field Mappings - Info (1)

- Field mappings are done within the construct function
- You need to define an association between any unmapped fields
- Extending a class may have done some fields already for you
- Click the name of the class in the UI to see what is mapped
  - Review these pages with your clients. Red means bad!
  - Once you are sure you do not need a source/dest field, you can set it to DNM (do not map)

Field Mappings - Samples (2)

$this->addFieldMapping($DESTINATION, $SOURCE); // source optional!

$this->addFieldMapping(NULL, $SOURCE); // source only needed as a temporary var

$this->addFieldMapping($DESTINATION, 'variable_defined_later'); // we haven't created this yet, we will in prepareRow

```
// LDAP mappings (note : not queried yet)
$this->addFieldMapping('ldap_user_puid_sid', 'ldap_sid');
$this->addFieldMapping('ldap_user_puid_sid:language')
  ->defaultvalue('UND'); // subfield

$this->addFieldMapping('order_number')
     ->xpath('ORDER_ID'); // XML source format
```

Field Mappings – Use the FOO! (3)

```
// Keep old UIDs. Same syntax for node NIDs.

// Do not do this since we do not login as user 1!

$this->addFieldMapping('uid', 'uid');

$this->removeFieldMapping('is_new');

$this->addFieldMapping('is_new')->defaultvalue(TRUE); // set to false to overwrite
```

Mapping & Binding - Overview (1)

- This section not really needed for migrate_d2d
  - Because settings.php
  - Because UI
  - Because extending classes where done already
- Map class manages relationship between old & new
  - Integer by default!
  - XML import needs to be varchar
  - Registered as integer?
    - Re-register or drop 2 tables with this specific class name (Map & Messages)

Mapping & Binding – Map Table (2)

```
$this->map = new MigrateSQLMap($this->machineName,
    array(
      'order_id' => array(
        'type' => 'varchar',
        'length' => 255,
        'not null' => TRUE,
        'description' => "Order ID",
      ),
    ),   MigrateDestinationEntityAPI::getKeySchema('commerce_order', 'commerce_order')
  );
```

Mapping & Binding - Sources (3)

- If you are not using D2D, you will also need to bind to a source and destination.
- You can find the supported handlers on DrupalContrib for both source and destination
- File-based imports (XML, CSV) have support for *multiple* input files so you can batch things. ^_^

Mapping & Binding – SQL Query (4)

```
$query = db_select('migrate_example_beer_topic', 'met')
        ->fields('met', array('style', 'details', 'style_parent', 'region',
'hoppiness'))
        ->orderBy('style_parent', 'ASC');
$this->source = new MigrateSourceSQL($query);
```

Mapping & Binding – XML Query (5)

```
// Set up our destination - terms in the migrate_example_beer_styles vocabulary
$this->destination = new MigrateDestinationTerm('migrate_example_beer_styles');

$xml_folder = '/mnt/tmp/transactions/';
$item_url = $xml_folder . 'transactions.xml';
$item_xpath = '/Transactions/Order'; // relative to doc
$item_id_xpath = 'ORDER_ID'; // relative to item_xpath
$this->source = new MigrateSourceList(new MigrateListXML($list_url),
  new MigrateItemXML($item_url), $fields);
$this->source =  new MigrateSourceXML($item_url, $item_xpath, $item_id_xpath, $fields);
$this->destination = new MigrateDestinationEntityAPI('commerce_order', 'commerce_order');
```

Queries – Extending Base SQL Query (1)

- A base query should be defined within your construct
- You can extend the fields caputred by the query in your construct
- Eventually adding things your query will blow up MySQL
  - Kill it fast, you're going to be pinned to 100% CPU!

Queries – Extending Base SQL Query (2)

```
$query->leftJoin('content_field_location', 'l', 'n.nid = l.nid');
$query->fields('l', array('field_location_value'));
```

```
$this->addFieldMapping('field_location', 'field_location_value')->defaultValue(0);
$this->addFieldMapping('field_location:language')
  ->defaultvalue('UND');
```

Queries – Plan B (3)

- We can define any query we want in the later stages of processing.
- The standard Drupal way of doing things applies :
  - db_set_active('d6'); // d6 is name of second DB in settings.php
  - Remember to switch back to D7 database when you have your $result : db_set_active();
- We'are FINALLY done creating our basic class and construct !

function PrepareRow (1)

- Preprocessor function that allows you to modify the content that is being migrated
- Loads each row individually to conserve memory
- Allows you to run DB queries your secondary database, etc.
- Allows you to reject specific nodes/users/etc.
- Does NOT run for XML because would force file read to pause (impossible?) … probably a bad bad bad idea (depending on OS/filesystem)

function PrepareRow (2)

- Uses field handlers so the arrays are not expanded
- $row->body should have real data (not an array)
- Try just throwing data at it and see if it works!
  - $row->thing = thing;
- Look at the row here and reject it if you want
  - if ($row->title == 'test') {
  - return FALSE;
  - } // goodbye test node!
- Provide the source argument to one of your previous addFieldMapping sources :
  - $row->variable_defined_later = 'combine' . 'some' . 'things';
- Get stuff ready for the next stage
  - $row->tmp_junk = 'random things';

function prepare

- Fully expanded Drupal objects that we know and love
  - Deal with all your unsupported fields here
  - Date field not working? Who cares?! I can deal with an expanded Drupal array!
- Do all your XML or CSV based mangling here

function complete

- Usually unnecessary
- Dirty post-processing work
  - calculating commerce order totals from the migrated items in the order
- Affecting unrelated things in the system, live dangerously!
  - Commerce needs this to calculate order* total after items*.
  - *orders and items are separate classes

function rollback

- Usually unnecessary
- Runs globally by default, not per-row

Running the migration

- Use drush, apache/nginx gets in the way, can lose connection
  - Won't stop running? Lost connection. Use Reset!
- Migrate manages memory and long running processes automatically
- Show what is running with devel's dpm() and/or drush_print_r();

Interesting Bits - Minding the map (1)

- You can override your source queries to set a *count* query if you want solid numbers in the UI
- Might need to explicitly prevent joining across different databases...

Interesting Bits – Secondary DBs (2)

- Using db_set_active('d6'); can often cause errors :
  - Base table does not exist?
    - There is an error in your query syntax! Error thrown on D7 db because the D6 bind failed.
  - Remember to exit that DB!
    - db_set_active();

Interesting Bits – Migrate Extras (3)

- This module has some field handlers that you might want to make use of
- Just install it and things magically work (allowing you to work with things in prepareRow rather than prepare)
- This module will eventually go away

Interesting Bits – Webform (4)

- Webforms can be migrated, but just the submissions via migrate_extras
- Node revisions are possible at https://drupal.org/node/1298724
- Dealing with files : https://drupal.org/node/1540106

Interesting Bits – Simplehtmldom (5)

- If you need to edit HTML on the fly, do it in a jQuery-like object-oriented way after installing simplehtmldom module :

```
// Load and parse string into objects
$html = str_get_html($text);
// Apply new path pattern to any links
foreach($html->find('a') as $element) {
  // remap private file paths
  $element->href = str_replace('system/private_files', 'system/files', $element->href);
}

// Send the updated HTML text back
return $html->outertext;
```

Homework

- *Beer*. Shows you the basics of writing a migration. (migrate_example module)
- *Wine*. More advanced demo using multiple XML files with Xpath parsing (migrate_example module).
- Migrate_d2d has an example module – this one is GOOD (for reading)
- The examples module (always a good ressource).

More example migrations

- Übercart → Commerce (only 2.4 & 2.5 support... not 2.6)
- Wordpress to content type
  - Called a « dynamic » migration
  - Uses the WP backup file
  - Remap image URLs? :(
- Maybe your Typo3 site was a mistake? ;)

Planning to rearchitect D6 → D7?

- Content
  - Multilingual : content_translation (EN node + FR node) or entity_translation (single node)
  - References : node_reference (1-way) or entity reference + cnr (2-way)?
  - Custom fields into entities?
  - Move field_group « multigroup » into field_collections
  - Addressfield... did not exist before!
- Users
  - Profile : from profile module to core profile? Or add fields to user entity?

Migrate resources

- The migrate example module has Beer
- When you are done with Beer, then try Wine
- Documentation on drupal.orghttps://drupal.org/node/415260
  - Print version grabs sub-pages ;)
- D2D documentation https://drupal.org/node/1813498
- Review the Doxygen summaries (find source/dest connectors) : http://drupalcontrib.org/api/search/7/migrate
- Migrate_extras adds field support in some cases https://drupal.org/project/migrate_extras
- My blog post! Contains drush commands and example code : http://www.verbosity.ca/working-drupals-migrate-module

Bonus Slides

The following slides might be helpful for planning out your next migration project...

Planning for the migration (1)

- What are the content types and do they have equivalents in the new site?
  - Do some content types (or fields) merge together or split apart?
  - Is some data no longer necessary?
  - Can you use generic fields like addressfield rather than indivdual fields?

Planning for the migration (2)

- What migrate classes will you need to build?
  - One for each target entity
    - Every node type
    - Each vocabulary
    - Products?
  - Users may rely on Roles, etc.
- In what order do the classes need to be run?
- Can you extend a base class to get started?
- Each migration class requires time and budget

Planning for the migration (3)

- Include stakeholder content review in the project plan
- Let the client see the migrate pages and explain what is going on
- Have the client approve any « do not map » fields