Migrate

Getting it into Drupal

andrew morton
drewish@zivtech.com

Zivtech
ILLUMINATING TECHNOLOGY

Migrate 2.0

- Powerful object oriented framework for moving content into Drupal.
- Minimal UI, primarily code based.
- Steep learning curve (aka migraine module) but hopefully this talk will help.

Migrate 2.0

- Drupal 6 requires autoload and dbtng modules. So the code is very similar in 6 and 7.
- Migrate Extras provides support for many contrib modules.
- The best documentation is in the wine.inc and beer.inc example code.

Why not just use Feeds?

- If it does what you need, use it. It's much easier to setup.
- Migrate is faster, and more flexible but you need to write code to map fields.
- Feeds doesn't work well if you've got different content types that need to reference each other.

The Concepts

- Source
- Destination
- Map
- Field Mapping
- Field Handlers
- Migration

Source

- Interface to your existing data (SQL, CSV, XML, JSON)
- Provides a list of fields and descriptions
- Iterates (reset, next) over rows

Destination

- Interface for writing data to Drupal—typically to a Entity.
- Creates one record for each record in the source. If you're creating users and profiles, you'll need two migrations.

Map

Source

SQL, XML, JSON, CSV, etc

ID	Name	Age
1	Larry	34
2	Curly	54
4	Moe	47

Destination

Node, User, Term, etc

entity_id	field_name	field_age
32	Larry	34
33	Curly	54
34	Moe	47

Map

Source ID	Dest ID
1	32
2	33
4	34

Map

- Connects the source's ID to the destination's ID.
- Provides lookup facilities.
- Allows created items to be deleted as part of a rollback.

Field Mappings

Source

SQL, XML, JSON, CSV, etc

ID	Name	Age	Junk
1	Larry	34	blah
2	Curly	54	
4	Moe	47	Spam

Destination

Node, User, Term, etc

entity_id	field_name	field_age
32	Larry	34
33	Curly	54
34	Moe	47

Field Mappings

Source	Destination
Name	field_name
Age	field_age
Junk	NULL

Field Mappings

- Links a source field to a destination field.
- Lets you look up IDs from the other migration's maps with sourceMigration().
- If the destination has a complex structure (e.g. Address or file field) then additional data is passed in via arguments().

Migration

- Code that glues the parts together.
- In your constructor you setup: source, destination, map and field mappings.
- Allows customization at several points: prepareRow(), prepare(), complete().

Field Handlers

- Handles the details of converting the field into the structure that Drupal understands.
- Turns `$entity->field_bar = “foo”` into `$entity->field_bar[‘und’][0][‘value’] = “foo”`
- Might pull additional data out of arguments.

Destination Handler

- Provides additional functionality to a destination, e.g. comment adding a field to nodes for comment status.
- Destination delegates calls to fields(), prepare(), complete() to the destination handlers.
- You probably won't need to write one.

UI Demo

- Lets go look at the UI.

Drush Commands

- migrate-status (ms) - List all migrations and display their current status.
- migrate-import (mi) - Start a migration and create/update destination objects.
- migrate-reset-status (mrs) - Reset a migration.
- migrate-rollback (mr) - Delete a migration's destination objects.

Basic Module

- Create a new module
- Implement hook_migrate_api()
- Create a class that extends Migration and setup the source, destination, map and field mappings in the constructor
- Register the class in the .info file

SQL Source

```
// inside __construct()

$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);
```

Or a CSV Source

```
// The definition of the columns. Keys are integers,
// values are an array of field name then description.
$columns = array(
  0 => array('cvs_uid', 'Id'),
  1 => array('email', 'Email'),
  2 => array('name', 'Name'),
  3 => array('date', 'Date'),
);

// Instantiate the class using the path to the CSV
// file and the columns.
$path = 'path/relative/to/drupal/root/your_file.csv';
$this->source = new MigrateSourceCSV($path, $columns);
```

Other Sources

- There are also classes for importing content from XML and JSON.
- Lots of variation among sources so expect to do some tweaking.

Source Base Classes

- If you can fetch IDs separately from values:
  - Use MigrateSourceList as a source
  - Implement MigrateList for fetching counts and IDs, and MigrateItem for fetching values
- If everything is in a single file with IDs mixed in:
  - Use MigrateSourceMultiItems as a source
  - Implement MigrateItems for extracting IDs and values

Migration Map

```
// inside __construct()

$this->map = new MigrateSQLMap($this->machineName,
  array(
    'style' => array(
      'type' => 'varchar',
      'length' => 255,
      'not null' => TRUE,
      'description' => 'Topic ID',
    )
  ),
  MigrateDestinationTerm::getKeySchema()
);
```

Destinations

```
// inside __construct()

// Create terms...
$this->destination = new
  MigrateDestinationTerm('example_beer_styles');

// ...or nodes...
$this->destination = new
  MigrateDestinationNode('article');

// ...or
$this->destination = new
  MigrateDestinationUser();
```

Creating Destinations

- Hopefully you won't need to.
- If you're working with entities created by the Entity API make sure you look at: http://drupal.org/node/1168196

Field Mappings

```
// inside __construct()

// Can be as simple as this...
$this->addFieldMapping('name', 'style');

// ...or more complicated.
$this->addFieldMapping(NULL, 'region')
  ->description('This is broken')
  ->issueGroup(t('Client Issues'))
  ->issuePriority(
    MigrateFieldMapping::ISSUE_PRIORITY_MEDIUM)
  ->issueNumber(770064);
```

Field Mapping Arguments

- Generally used as a hack to pass multiple source fields into a single destination field.
- Use this magic syntax to have arguments replaced by values from the source row:

```
$this->addFieldMapping('D', 'S1')->arguments(
  array('A1' => array('source_field' => 'S2')),
  array('A2' => array('source_field' => 'S3'))
);
```

Field Mapping Arguments

```
// Files have so many arguments there's a helper
// to build the array:
$arguments = MigrateFileFieldHandler::arguments(
  drupal_get_path('module', 'migrate_example'),
  'file_copy', FILE_EXISTS_RENAME, NULL,
  array('source_field' => 'image_alt'),
  array('source_field' => 'image_description'));
$this->addFieldMapping('field_image', 'image')
  ->arguments($arguments);
```

Field Mapping Source Migrations

- When you have an ID value from the old system and need to look up the new ID from the Migration Map:

```
$this->addFieldMapping('uid', 'author_id')
  ->sourceMigration('BeerUser')
```

- Add a dependency to make sure the other migration runs first:

```
$this->dependencies = array('BeerUser');
```

-

Circular Dependencies

- Break them by using stubs.
- Implement createStub($migration), create a dummy record and return the new id.
- Specify a sourceMigration on the field mapping.

prepareRow($row)

- Passes in the source row as an object so you can make modifications.
- Add or change field values by modifying the properties:

```
$row->name = $row->first . " " . $row->last;
$row->created = strtotime($row->access);
```

- Return FALSE to indicate that rows should be skipped over during an import.

prepare($entity, $row)

- Passes in the entity object with properties populated by field mappings, and the source row.
- Last chance to make changes before the entity is saved.
- If you have an unsupported field type you can manually populate it here:

complete($ent, $row)

- Passes in the saved entity (with any ID values from auto increment fields) and the source row.
- This is the place if you need to update other records to reference the new entity.
- If you're doing a node_save($ent) in here, you're doing it wrong.

Writing a Field Handler

- Hopefully you won't need to.
- Register compatible field type in constructor.
- Handle conversion in prepare().
- Optionally, use complete() for follow tasks.

Thanks!
Any Questions?

Zivtech

ILLUMINATING TECHNOLOGY