The Digital Mayflower

Data Pilgrimage with the Migrate Module

Erich Beyrent

Working with Drupal since 2004!

Drupal: http://drupal.org/u/ebeyrent

Twitter: http://twitter.com/ebeyrent

Github: http://github.com/ebeyrent

Agenda

Agenda

- Prerequisites

Agenda

- Prerequisites
- Migrate Concepts

Agenda

- Prerequisites
- Migrate Concepts
- Implementation

Agenda

- Prerequisites
- Migrate Concepts
- Implementation
- Classes & Organization

Agenda

- Prerequisites
- Migrate Concepts
- Implementation
- Classes & Organization
- Execution

Agenda

- Prerequisites
- Migrate Concepts
- Implementation
- Classes & Organization
- Execution
- Performance

Agenda

- Prerequisites
- Migrate Concepts
- Implementation
- Classes & Organization
- Execution
- Performance
- Q & A


PROPRIETARY


J.L.G. Ferris

Fractured Fairy Tales

Prerequisites

- A working Drupal 7 / Drupal 8 installation
- Basic understanding of PHP OO
- Understanding of Drupal modules, hooks, and the API
- Understanding of how to write SQL and the QueryInterface


J.L.G. FERRIS ©

Migrate Goals

- Accurate
- Repeatable
- Performant
- Safe

Concepts

Source

The source is the interface to the data that is being migrated.

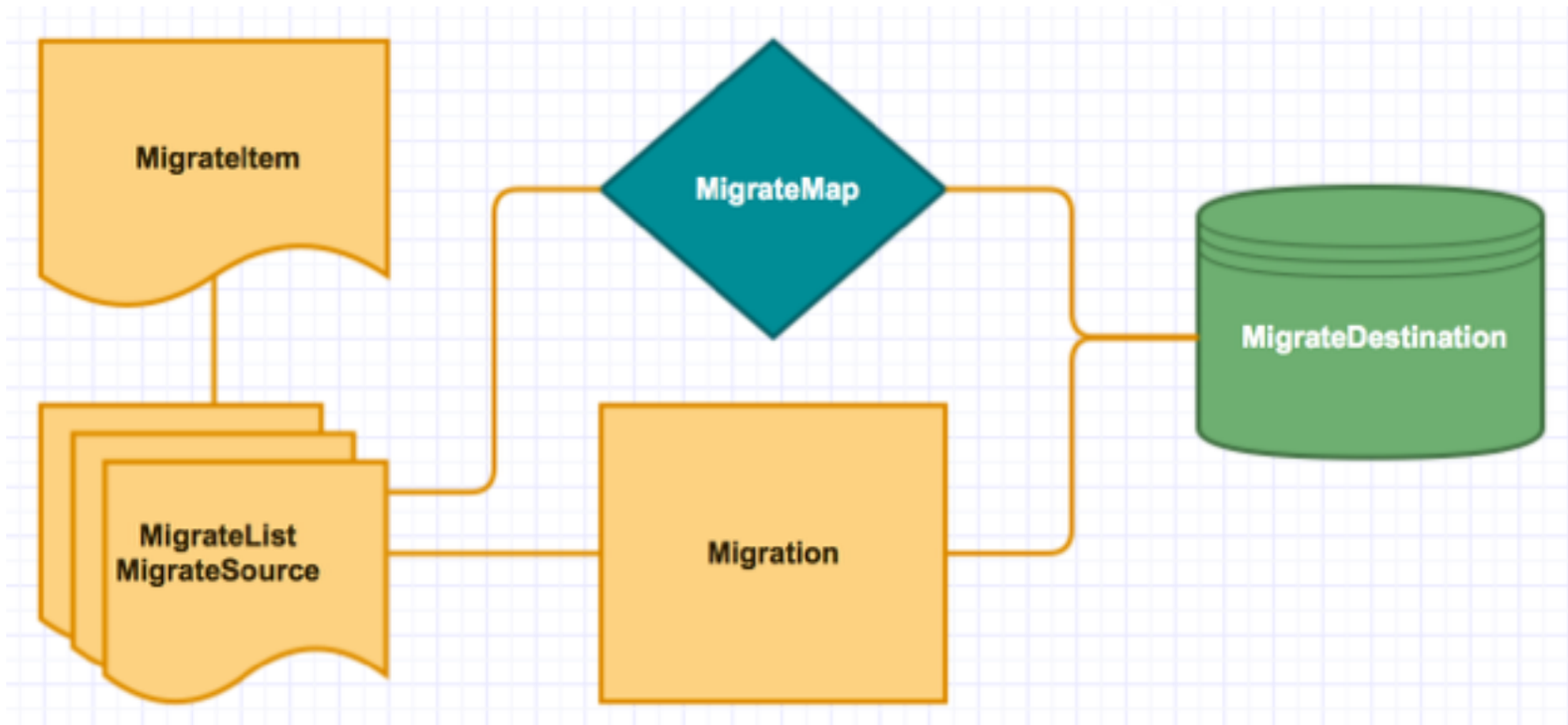
Destination

Destinations are responsible for saving data in Drupal.

Concepts

Map

Connects the source to the destination



Implementation

The Module

Implements hook_migrate_api()
Registers classes in the .info file

```
; **
; MY_MODULE.info
; **
dependencies[] = migrate
files[] = migrate_newsletters.module
files[] = migrate_newsletters.migrate.inc
files[] = base.inc
files[] = newsletters.inc
files[] = subscribers.inc
files[] = sources.inc
files[] = subscriptions.inc
```

```
/**
 * Implements hook_migrate_api().
 */
function MY_MODULE_migrate_api() {
  $api = array(
    'api' => 2,
    'groups' => array(
      'newsletters' => array(
        'title' => t('Newsletter'),
      ),
    ),
    'field handlers' => array(
      'DateMigrateFieldHandler',
    ),
    'migrations' => array(
      'Newsletters' => array(
        'class_name' => 'NewslettersMigration',
        'group_name' => 'newsletters',
      ),
      'Sources' => array(
        'class_name' => 'SourcesMigration',
        'group_name' => 'newsletters',
      ),
    ),
  );
  return $api;
}
```

```
/**
 * Implements hook_migrate_api().
 */
function MY_MODULE_migrate_api() {
  $api = array(
    'api' => 2,
    'groups' => array(
      'newsletters' => array(
        'title' => t('Newsletter'),
      ),
    ),
    'field handlers' => array(
      'DateMigrateFieldHandler',
    ),
    'migrations' => array(
      'Newsletters' => array(
        'class_name' => 'NewslettersMigration',
        'group_name' => 'newsletters',
      ),
      'Sources' => array(
        'class_name' => 'SourcesMigration',
        'group_name' => 'newsletters',
      ),
    ),
  );
  return $api;
}
```

```
/**
 * Implements hook_migrate_api().
 */
function MY_MODULE_migrate_api() {
  $api = array(
    'api' => 2,
    'groups' => array(
      'newsletters' => array(
        'title' => t('Newsletter'),
      ),
    ),
    'field handlers' => array(
      'DateMigrateFieldHandler',
    ),
    'migrations' => array(
      'Newsletters' => array(
        'class_name' => 'NewslettersMigration',
        'group_name' => 'newsletters',
      ),
      'Sources' => array(
        'class_name' => 'SourcesMigration',
        'group_name' => 'newsletters',
      ),
    ),
  );
  return $api;
}
```

```
/**
 * Implements hook_migrate_api().
 */
function MY_MODULE_migrate_api() {
  $api = array(
    'api' => 2,
    'groups' => array(
      'newsletters' => array(
        'title' => t('Newsletter'),
      ),
    ),
    'field handlers' => array(
      'DateMigrateFieldHandler',
    ),
    'migrations' => array(
      'Newsletters' => array(
        'class_name' => 'NewslettersMigration',
        'group_name' => 'newsletters',
      ),
      'Sources' => array(
        'class_name' => 'SourcesMigration',
        'group_name' => 'newsletters',
      ),
    ),
  );
  return $api;
}
```

```
/**
 * Implements hook_migrate_api().
 */
function MY_MODULE_migrate_api() {
  $api = array(
    'api' => 2,
    'groups' => array(
      'newsletters' => array(
        'title' => t('Newsletter'),
      ),
    ),
    'field handlers' => array(
      'DateMigrateFieldHandler',
    ),
    'migrations' => array(
      'Newsletters' => array(
        'class_name' => 'NewslettersMigration',
        'group_name' => 'newsletters',
      ),
      'Sources' => array(
        'class_name' => 'SourcesMigration',
        'group_name' => 'newsletters',
      ),
    ),
  );
  return $api;
}
```

Implementation

The Classes

Migration classes extend the abstract Migration class

```php
abstract class NewslettersBaseMigration extends Migration {

  /**
   * @var array
   */
  protected $sourceConfiguration = array();

  /**
   * Class constructor.
   */
  public function __construct($group = NULL) {
    // Always call the parent constructor first for basic setup.
    parent::__construct($group);

    $this->team = array(
      new MigrateTeamMember('Joe Migrate', 'foo@bar.com', t('admin')),
    );

    $this->issuePattern = 'https://example.com/issues/:id:';

    $this->sourceConfiguration = array(
      'servername' => 'localhost',
      'username' => 'admin',
      'password' => 'bubbles',
      'database' => 'source_database',
    );
  }
}
```

```
abstract class NewslettersBaseMigration extends Migration {

  /**
   * @var array
   */
  protected $sourceConfiguration = array();

  /**
   * Class constructor.
   */
  public function __construct($group = NULL) {
    // Always call the parent constructor first for basic setup.
    parent::__construct($group);

    $this->team = array(
      new MigrateTeamMember('Joe Migrate', 'foo@bar.com', t('admin')),
    );

   $this->issuePattern = 'https://example.com/issues/:id:';

    $this->sourceConfiguration = array(
      'servername' => 'localhost',
      'username' => 'admin',
      'password' => 'bubbles',
      'database' => 'source_database',
    );
  }
}
```

```
abstract class NewslettersBaseMigration extends Migration {

  /**
   * @var array
   */
  protected $sourceConfiguration = array();

  /**
   * Class constructor.
   */
  public function __construct($group = NULL) {
    // Always call the parent constructor first for basic setup.
    parent::__construct($group);

    $this->team = array(
      new MigrateTeamMember('Joe Migrate', 'foo@bar.com', t('admin')),
    );

  $this->issuePattern = 'https://example.com/issues/:id:';

    $this->sourceConfiguration = array(
      'servername' => 'localhost',
      'username' => 'admin',
      'password' => 'bubbles',
      'database' => 'source_database',
    );
  }
}
```

```
abstract class NewslettersBaseMigration extends Migration {

  /**
   * @var array
   */
  protected $sourceConfiguration = array();

  /**
   * Class constructor.
   */
  public function __construct($group = NULL) {
    // Always call the parent constructor first for basic setup.
    parent::__construct($group);

    $this->team = array(
      new MigrateTeamMember('Joe Migrate', 'foo@bar.com', t('admin')),
    );

    $this->issuePattern = 'https://example.com/issues/:id:';

    $this->sourceConfiguration = array(
      'servername' => 'localhost',
      'username' => 'admin',
      'password' => 'bubbles',
      'database' => 'source_database',
    );
  }
}
```

Anatomy of a migration class

Migration classes typically contain four methods:

- __construct()
- prepare()
- prepareRow()
- complete()

```
class NewslettersMigration extends NewslettersBaseMigration {

  /**
   * Class constructor method.
   *
   * @param array $arguments
   */
  public function __construct(array $arguments = array()) {
    parent::__construct($arguments);
    $this->description = t('Newsletters');
    $options = array(
      'track_changes' => 1
    );
```

```
class NewslettersMigration extends NewslettersBaseMigration {

  /**
   * Class constructor method.
   *
   * @param array $arguments
   */
  public function __construct(array $arguments = array()) {
    parent::__construct($arguments);
    $this->description = t('Newsletters');
    $options = array(
      'track_changes' => 1
    );
```

```
// List of source fields to import.
$source_fields = array(
  ‘email’ => t(‘Subscriber email address’),
  // …
);

$select_query = Database::getConnection('default', ‘legacy')
  ->select('profile', 'p');
$select_query->fields('p', array());
$select_query->orderBy(‘email’);

$count_query = Database::getConnection('default', ‘legacy')
  ->select('profile', 'p');
$count_query->fields('p', array());
$count_query->orderBy(‘email’);
$count_query->addExpression(‘COUNT(email)’, ‘count’);

$this->source = new MigrateSourceSQL(
  $select_query,
  $source_fields,
  $count_query,
  $options
);
```

```
$this->destination = new MigrateDestinationTable(‘subscribers’);

/*
 * Other examples of destination:
 *
 * $this->destination = new MigrateDestinationNode(‘newsletter');
 *
 * $this->destination = new MigrateDestinationTerm(‘advertising_source');
 *
 */
```

```
// Source and destination relation for rollbacks.
$this->map = new MigrateSQLMap(
  $this->machineName,
  array(
   'id' => array(
     'type' => 'int',
     'not null' => TRUE,
     'description' => 'Newsletter ID.',
     'alias' => 'n',
   )
  ),
  MigrateDestinationNode::getKeySchema()
);
```

```
// Source and destination relation for rollbacks.
$this->map = new MigrateSQLMap(
  $this->machineName,
  array(
    'member_id' => array(
      'type' => 'varchar',
      'length' => 128,
      'not null' => TRUE,
      'description' => 'Subscriber Email ID.',
      'alias' => 'member_id',
    ),
  ),
  MigrateDestinationTable::getKeySchema('subscribers')
);
```

```
// Source and destination relation for rollbacks.
$this->map = new MigrateSQLMap(
  $this->machineName,
  array(
    'Source_ID' => array(
      'type' => 'int',
      'not null' => TRUE,
      'description' => 'Source ID.',
      'alias' => 't',
    ),
  ),
  MigrateDestinationTerm::getKeySchema()
);
```

```
// Basic field mapping.
$this->addFieldMapping('email', 'email')
  ->description(t('Map the subscriber email from the source’));

// More advanced field mapping.
$this->addFieldMapping('uuid', 'uuid')
  ->description(t('UUID for the subscriber’))
  ->defaultValue(uuid_generate())
  ->issueGroup(t('Client questions'))
  ->issueNumber(765736)
  ->issuePriority(MigrateFieldMapping::ISSUE_PRIORITY_OK);

// Getting fancy - convert string to an array.
$this->addFieldMapping(‘field_tags', ‘tags')
  ->separator(‘,’);

// Process the field with a callback.
$this->addFieldMapping(‘json_field’, ‘json’)
  ->callback(‘json_decode’);

// De-duplicate data.
$this->addFieldMapping('name', 'username')
  ->dedupe('users', 'name');
```

Map fields to previously migrated fields

```
// Map the field to a value from another migration.
$this->addFieldMapping('uid', 'author_id')
     ->sourceMigration('Users');
```

Skip source and destination fields

```
// Unmapped source fields.
$this->addUnmigratedSources(
  array(
    'first_name',
    'last_name',

  )
);

// Unmapped destination fields.
$this->addUnmigratedDestinations(
  array(
    'eid',
  )
);
```

Define the highwater field

```
// Define the highwater field.
$this->highwaterField = array(
  'name' => 'last_changed',
  'alias' => 'w',
  'type' => 'int',
);
```

Anatomy of a migration class

- prepare($entity, stdClass $row)

This method is called just before saving the entity.

- complete($entity, stdClass $row)

This method is called immediately after the entity is saved.

Anatomy of a migration class

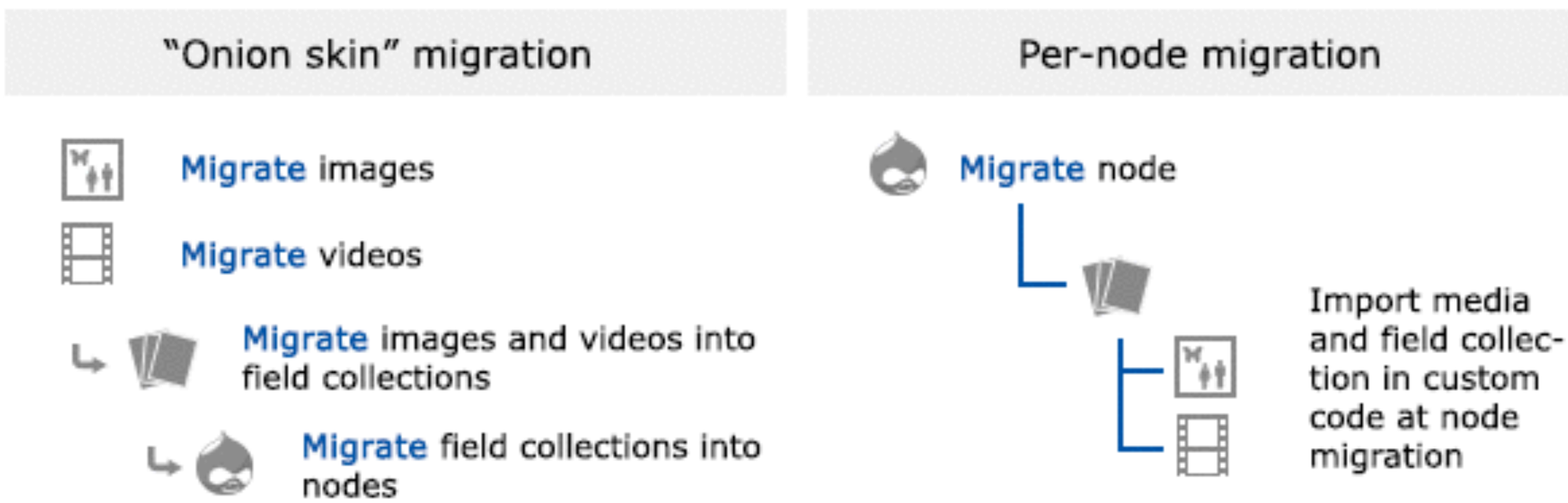
- prepareRow(stdClass $row)

This method is for conditionally skipping rows, and for modifying fields before other handlers

- createStub($migration, array $id)

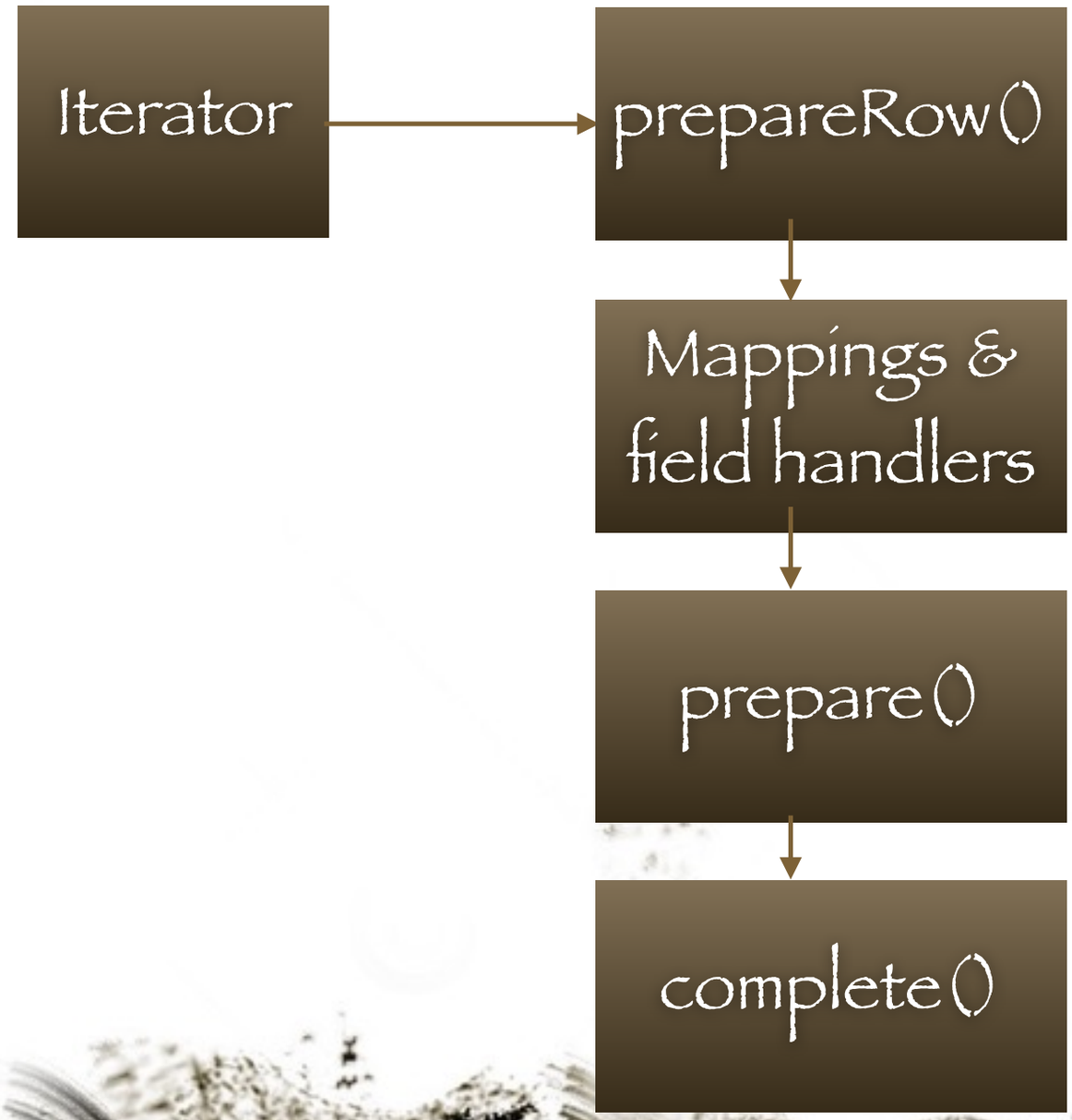
This method is for creating stub objects to be referenced by the current row

Anatomy of a migration class



source: http://alexrayu.com/blog/drupal-migration-onion-skin-or-node

Migration flow



Execution

Use drush, not the web interface

https://www.drupal.org/node/1806824

Run a single migration:

drush migrate-import Subscribers

Run all the migrations in a migration group:

drush migrate-import --group newsletters

Performance

Benchmarking

```
$ drush mi Migration --instrument [timer, memory, all]

$ drush mi Newsletters \
   --limit="100 items" \
   --instrument="all" \
   --feedback="30 seconds"
```

Performance

Database indexes add a lot of overhead on writes, especially with multiple multi-column indexes

```
db_drop_index($table, $index)
db_add_index($table, $index, array $fields)
```

```
class MyMigration extends MigrationBase {

  /**
   * Code to execute before first article row is imported.
   */
  public function preImport() {
    parent::preImport();
    $this->dropIndexes();
  }

  /**
   * Code to execute after the last article row has been imported.
   */
  public function postImport() {
    parent::postImport();
    $this->restoreIndexes();
  }

  protected function dropIndexes() {
    db_drop_index('my_table', 'name_of_index');
    return $this;
  }

  protected function restoreIndexes() {
    db_add_index('my_table', 'name_of_index', array('name_of_field'));
    return $this;
  }
}
```

Performance

MySQL Tuning

innodb_flush_log_at_trx_commit = 0
innodb_doublewrite = 0
innodb_support_xa = 0

Performance

MySQL Tuning

key_buffer_size = 128M
max_allowed_packet = 20M
query_cache_size = 128M
table_open_cache = 64
read_buffer_size = 2M
read_rnd_buffer_size = 8M
myisam_sort_buffer_size = 16M
join_buffer_size = 4M
tmp_table_size = 92M
max_heap_table_size = 92M
sort_buffer_size = 4M
innodb_additional_mem_pool_size = 8M

Performance

MySQL Tuning

innodb_file_per_table

Resources

Project page - http://drupal.org/project/migrate
Migrate Handbook - http://drupal.org/node/415260
Drush migrate commands - http://drupal.org/node/1561820
Using stubs - http://drupal.org/node/1013506
Database Tuning - http://www.drupal.org/node/1994584

Questions & Answers