DRUPALCON

DENVER

MARCH 19-23, 2012

Collaborative Publishing for Every Device

Getting It Into Drupal With Migrate

Presented by drewish (Andrew Morton)
drewish@katherinehouse.com

So You've Got This Old Website...

Let's do some shows of hands on what people are doing right now. Feeds? Custom scripts? Intern data entry? How many just know you're going to have to in the next 6-months.

My Goals

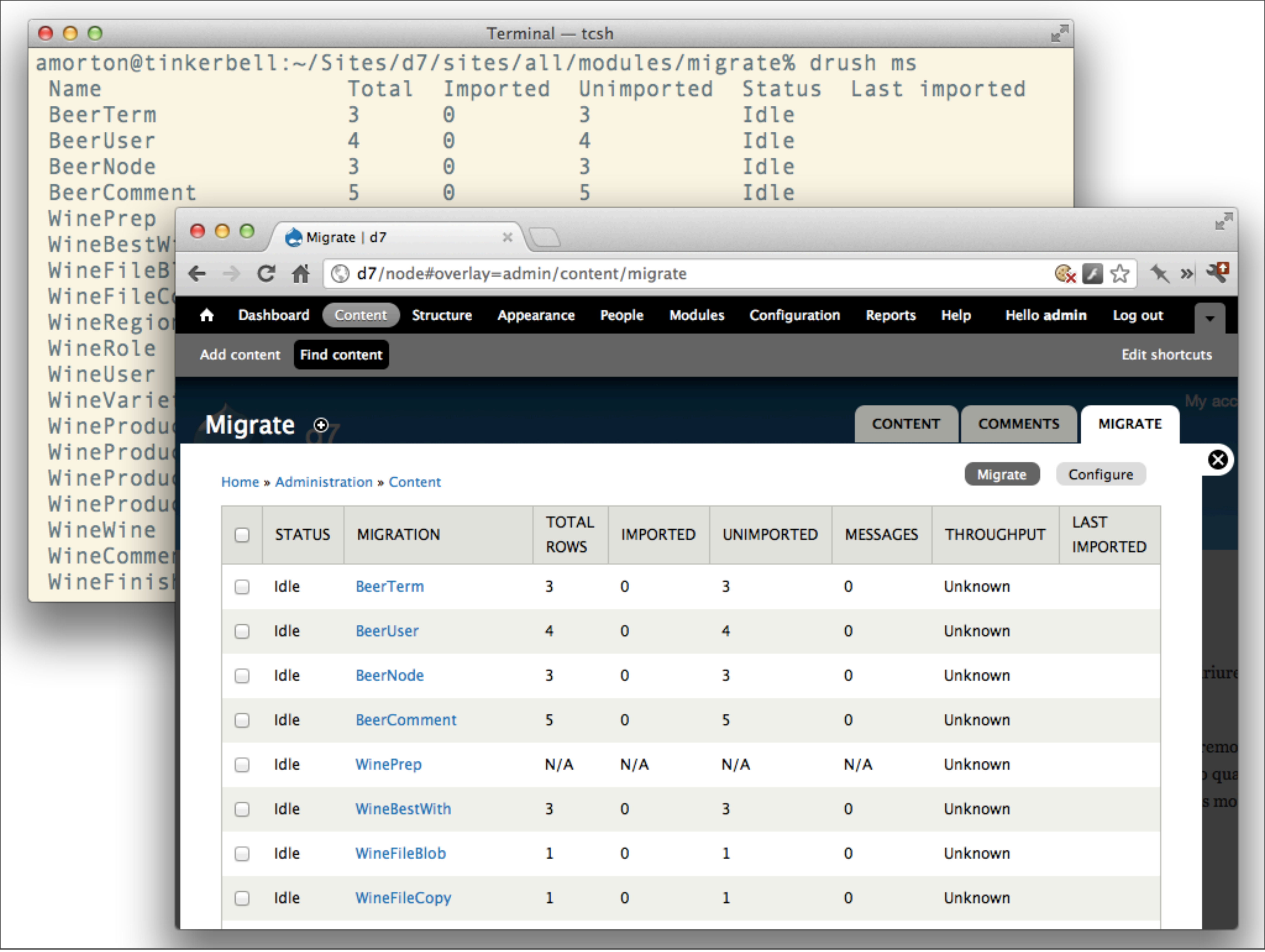
- Give you mental framework to understand Migrate.
- Step through the basics of building a migration.
- Convince to use Migrate rather than custom one-off scripts and/or add Migrate support to your modules.

Put another tool in your toolbox... and just talk you into using feeds.
Migrate has lots of pieces that are badly named making it harder to understand.

Migrate 2.0

- Powerful object oriented framework for moving content into Drupal.
- Minimal UI, primarily code based.
- Steep learning curve.

I’ve been using Migrate for almost a year and was annoying enough in the issue queue that the just made me a maintainer.
<See if Mike and Moshe are in the crowd and introduce them>
Explain the differences between v2 and v1.
aka migraine module, but hopefully this talk will help.



```
amorton@tinkerbell:~/Sites/d7/sites/all/modules/migrate% drush ms
 Name                    Total  Imported  Unimported  Status  Last imported
 BeerTerm                3      0         3           Idle
 BeerUser                4      0         4           Idle
 BeerNode                3      0         3           Idle
 BeerComment             5      0         5           Idle
 WinePrep
 WineBestW
 WineFileB
 WineFileC
 WineRegio
 WineRole
 WineUser
 WineVarie
 WineProdu
 WineProdu
 WineProdu
 WineProdu
 WineWine
 WineComme
 WineFinis
```

Home » Administration » Content

□	STATUS	MIGRATION	TOTAL ROWS	IMPORTED	UNIMPORTED	MESSAGES	THROUGHPUT	LAST IMPORTED
□	Idle	BeerTerm	3	0	3	0	Unknown	
□	Idle	BeerUser	4	0	4	0	Unknown	
□	Idle	BeerNode	3	0	3	0	Unknown	
□	Idle	BeerComment	5	0	5	0	Unknown	
□	Idle	WinePrep	N/A	N/A	N/A	N/A	Unknown	
□	Idle	WineBestWith	3	0	3	0	Unknown	
□	Idle	WineFileBlob	1	0	1	0	Unknown	
□	Idle	WineFileCopy	1	0	1	0	Unknown	

There’s a basic UI and great drush support. I’m going to ignore them because of time constraints.

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 most comprehensive and up-to-date documentation is the beer.inc and wine.inc examples.

Well these slides are really the best... but the might get out of date.

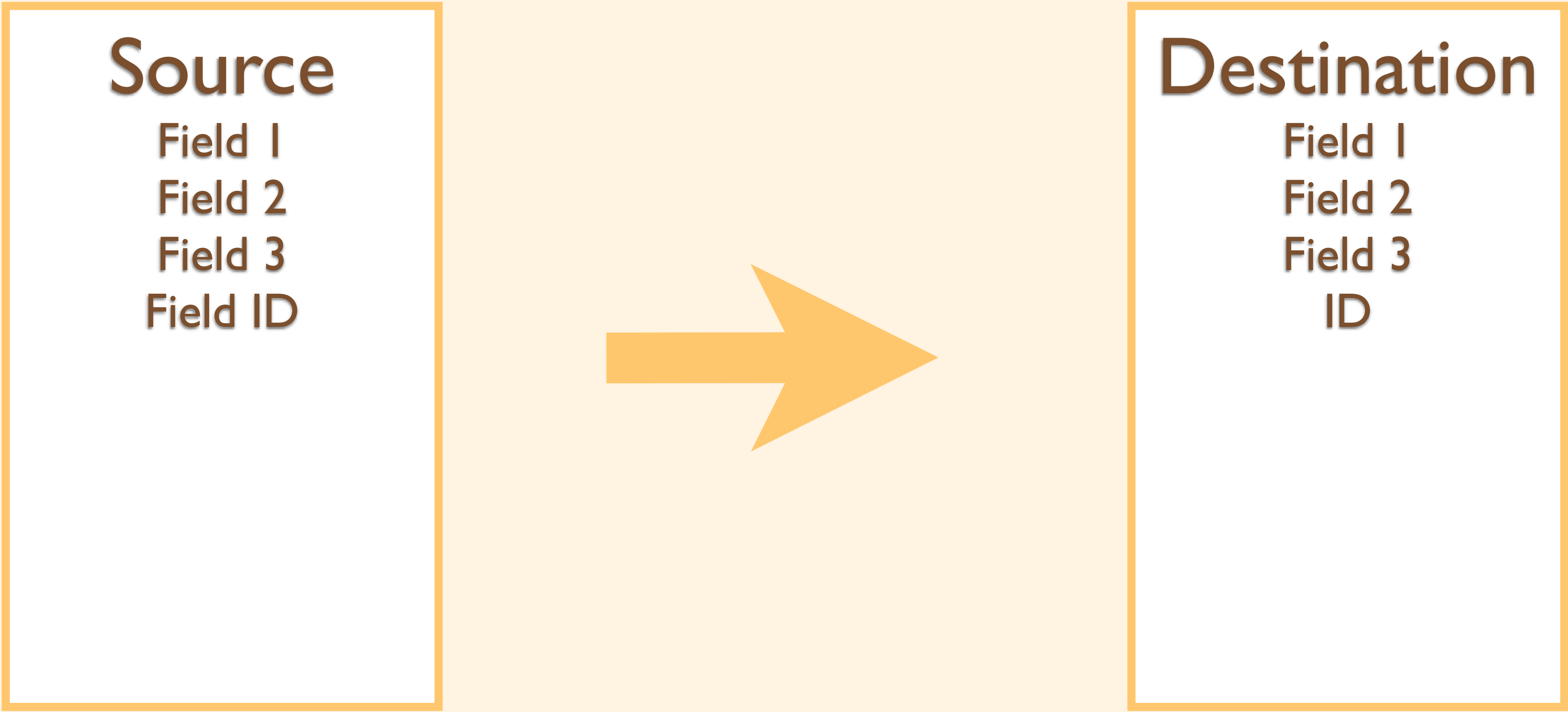
Why not just use Feeds?

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

Get asked this a lot. I’m pragmatic, use the best tool.

Theory

Source & Destination



Source
Field 1
Field 2
Field 3
Field ID

Destination
Field 1
Field 2
Field 3
ID

Source

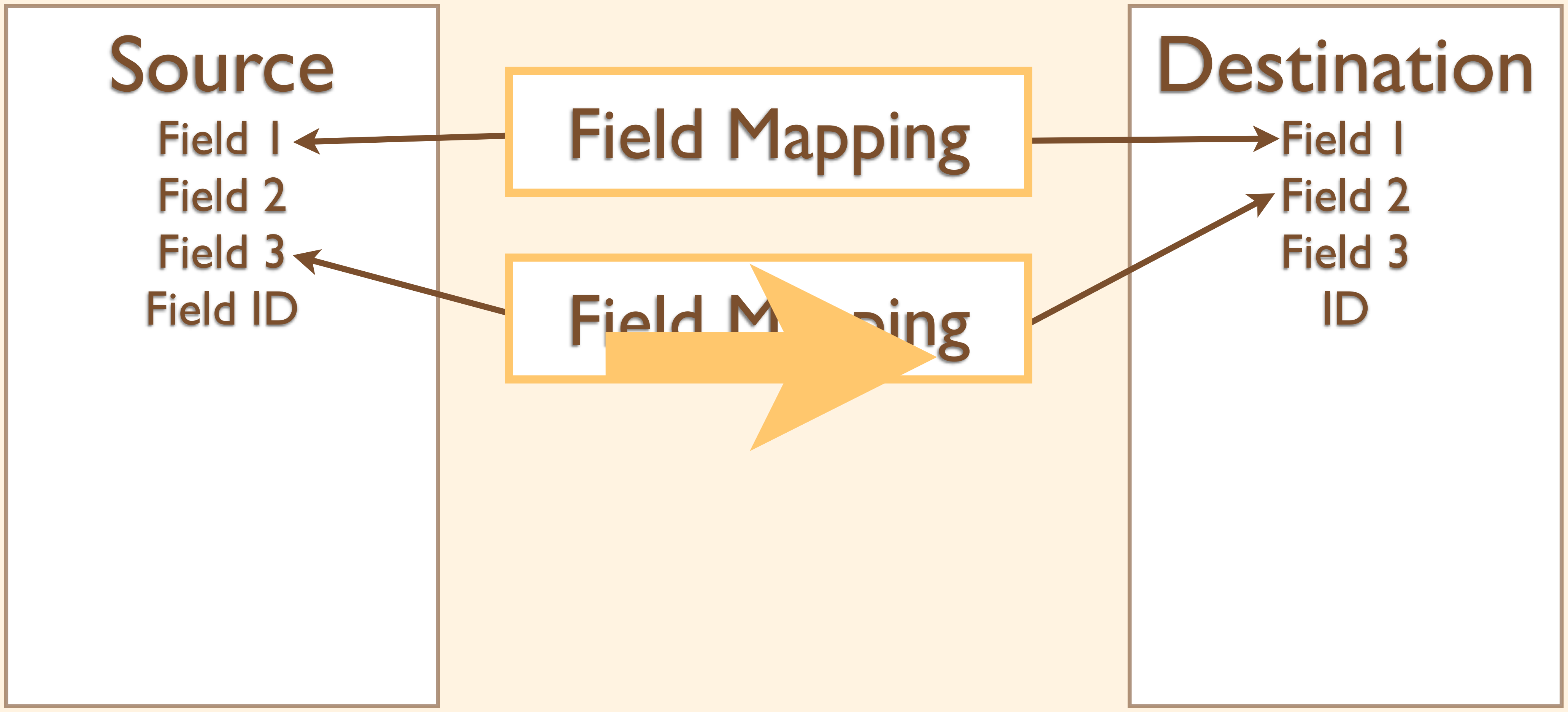
- Interface to your existing data (SQL, CSV, XML, JSON).
- Provides a list of fields and descriptions.
- Responsible for iterating over rows.

Destination

- Responsible for writing a specific type of data to Drupal, e.g. Node, User, or other Entity, but could be something like a SQL row.
- Creates one Destination record for each Source record.

Knows how to save it... user_save(), node_save(), entity_save(), db_insert(), etc.
So, if you're creating users and profiles, you'll need to do it in two migrations.

Map the Fields



Source

Field 1

Field 2

Field 3

Field ID

Field Mapping

Field Mapping

Destination

Field 1

Field 2

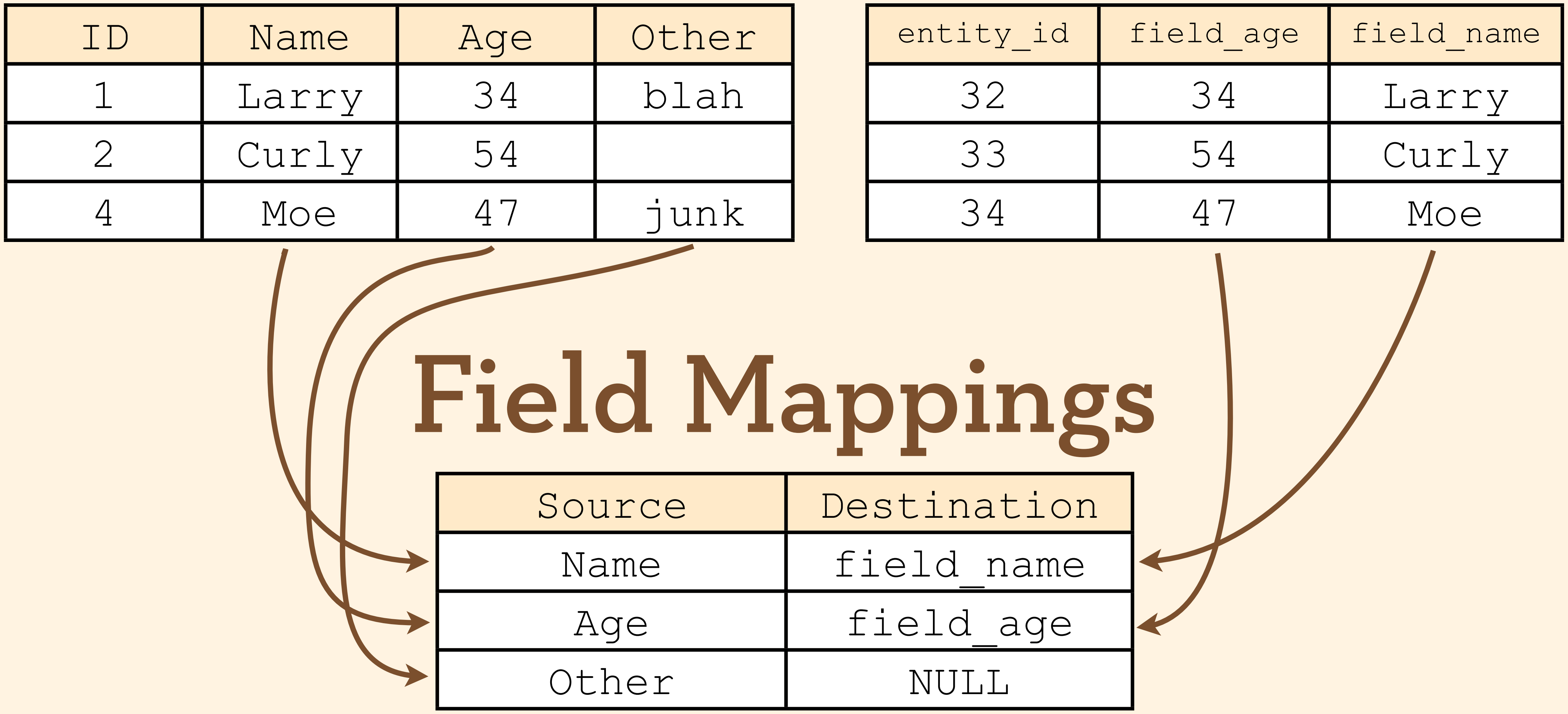
Field 3

ID

Wednesday, March 21, 12

With these pieces you’ve basically got feeds.

Field Mappings



Source

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

Destination

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

Field Mappings

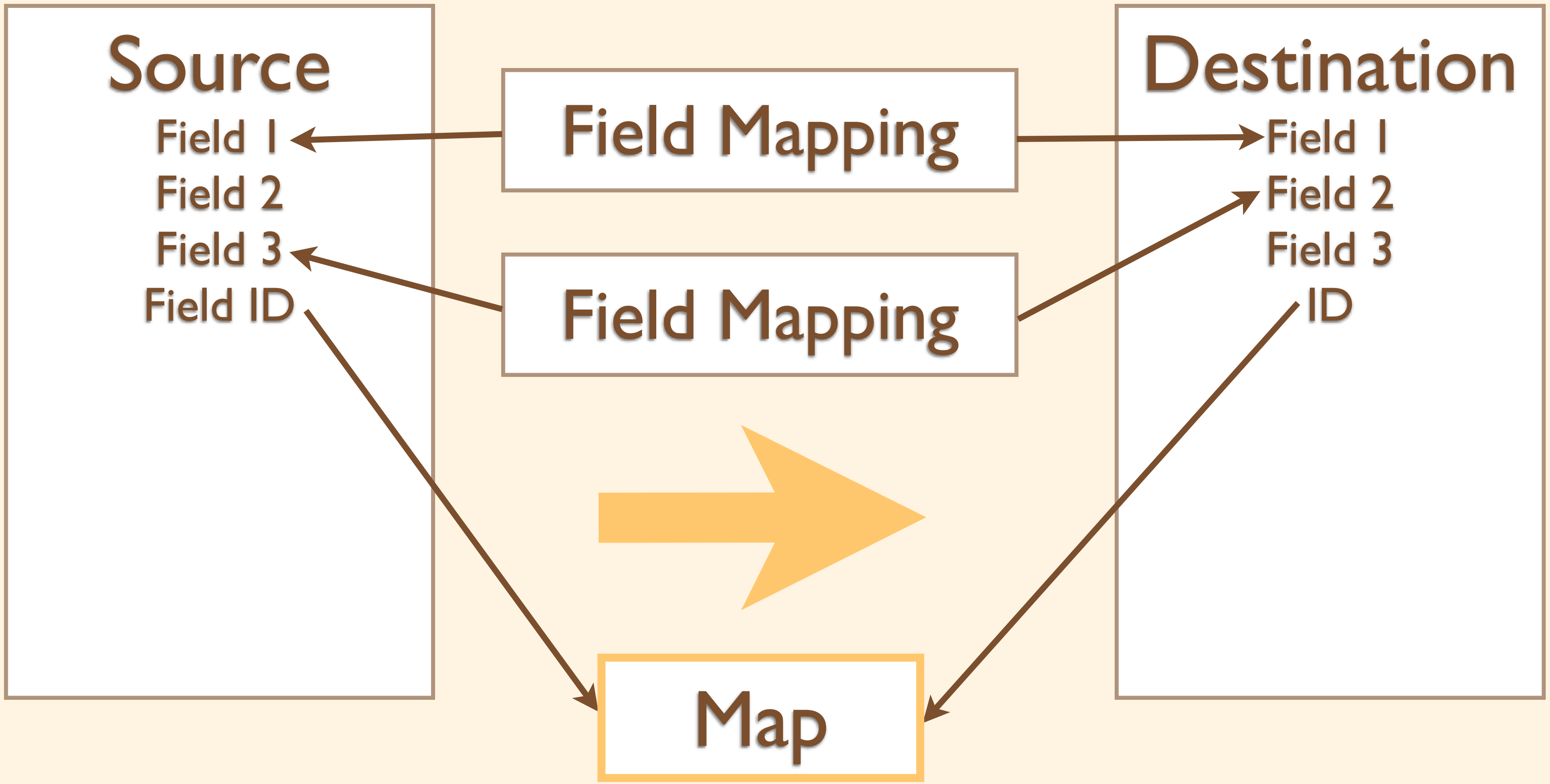
Source	Destination
Name	field_name
Age	field_age
Other	NULL

Field Mappings

- Links a source field to a destination field.
- Has some basic functions to transform values splitting on separators, etc.
- Holds additional arguments to pass on to the destination field.

Might skip over the source id lookup since we haven’t really talked about migration maps yet.

Migration Map



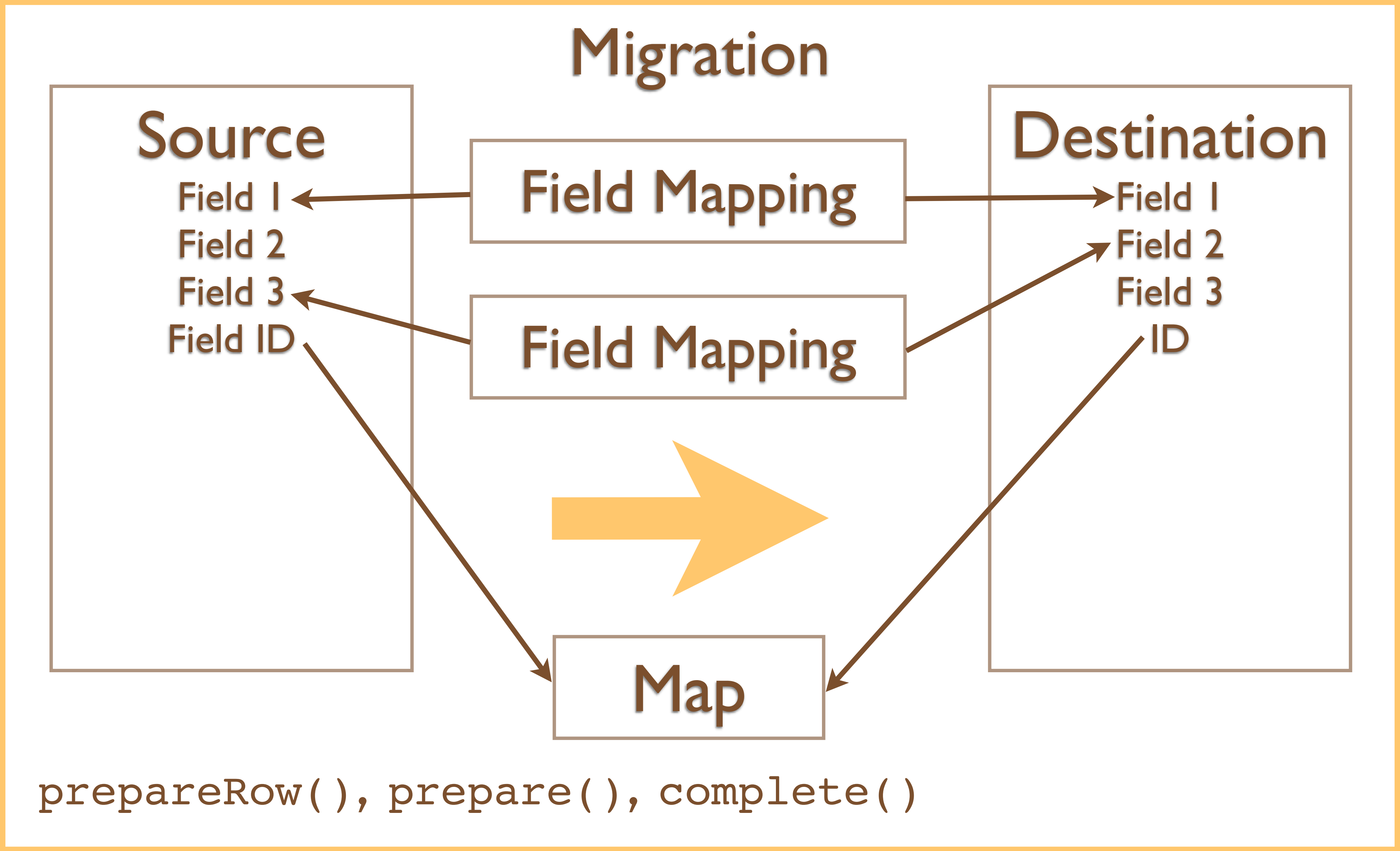
Now we need to do a little more than what Feeds provides. Lets track IDs so when we import users and articles we can attribute the articles to the correct user.

Map

- Connects the Source and Destination IDs allowing translation between them.
- Tracks the keys' schema format.
- Allows a migration to re-run and update existing records.
- Allows imported records to be deleted during a rollback.

Also provides some other benefits.
Composite keys are supported.

Migration



prepareRow(), prepare(), complete()

Wednesday, March 21, 12

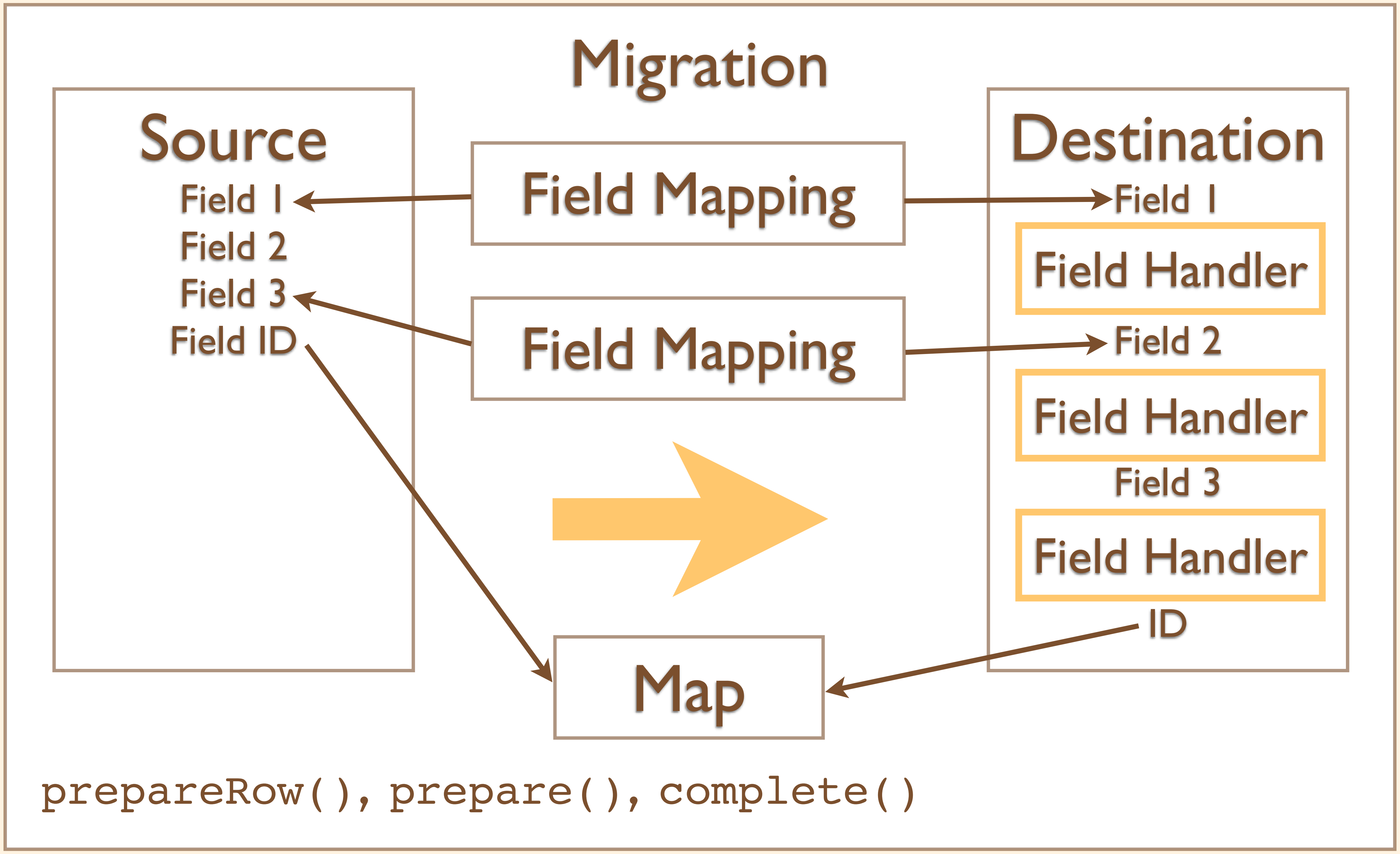
I’m going to ignore those three little functions for now but I just wanted to put them on here so you can visualize where they live.

Migration

- Sets up all the necessary pieces: Source, Destination, Map and Field Mappings.
- May provide logic for skipping over rows during the migration.
- May alter the entities during the migration.

Your class that glues all the parts together.

Field Handlers



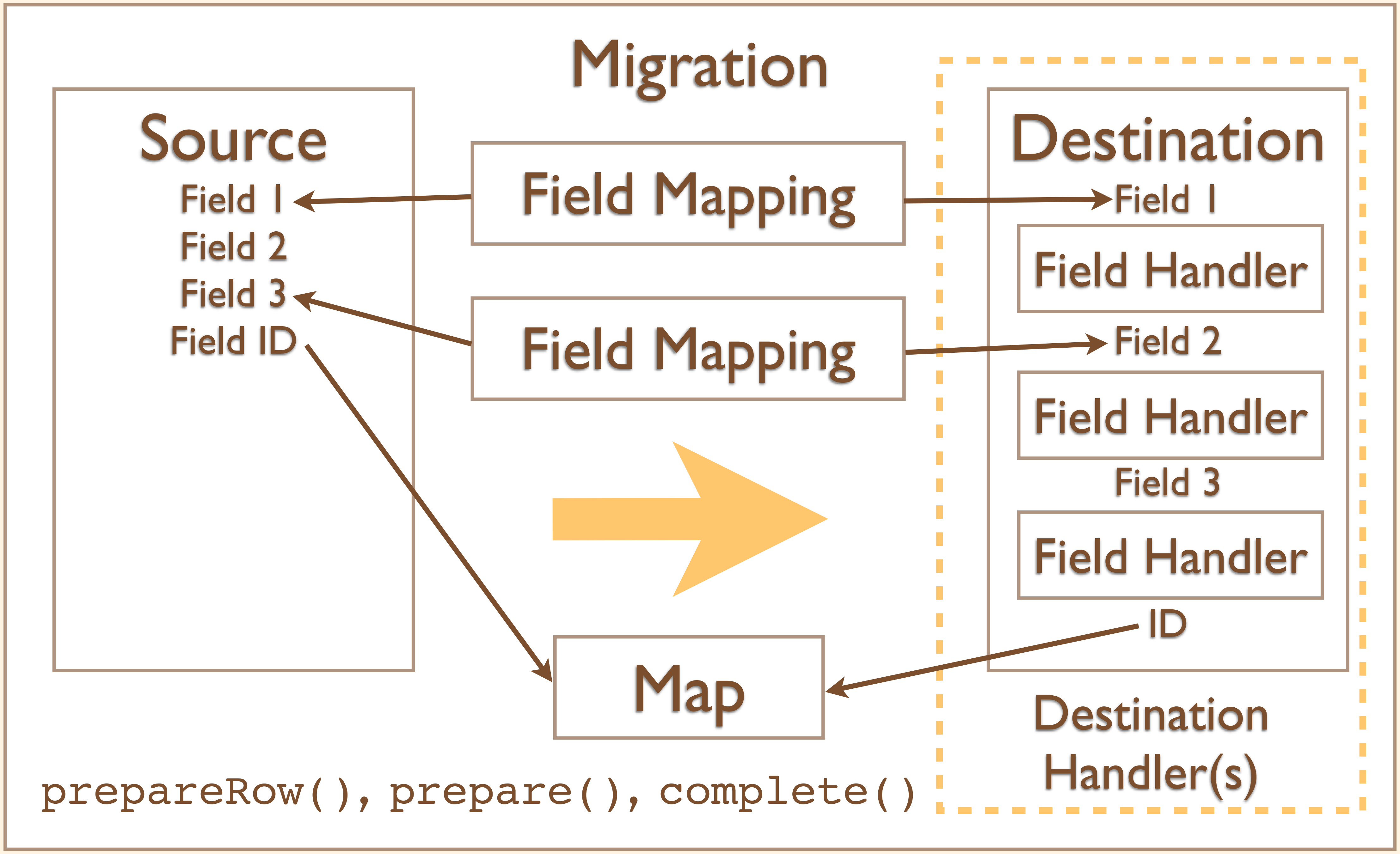
Wednesday, March 21, 12

If you think about the way the data would be coming out of the Source it'll be "flat" values like strings or integers. Drupal fields are typically nested arrays and something needs to know the format of those arrays, hence the field handler.

Field Handlers

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

Destination Handler



Destination Handler

- “Magically” extends an existing destination and adds functionality, e.g. `MigrateCommentNodeHandler` adds a comment status field to nodes.
- Contrib maintainers might need to create these.

Practice

Basic Migrate Module

- Create a new module to so you can disable it once the site launches.
- Implement `hook_migrate_api()` in the .module file.
- Create a class that extends Migration and in the constructor setup the Source, Destination, Map and Field Mappings.
- Register the class's file in the .info file.

If I'm being lazy and it's only a single migration class I'll just throw it in the .module file.

Migration Class

- Setup the Source, Destination, Map and Field Mappings in the constructor:

```
class BasicExample extends Migration {
  public function __construct() {
    parent::__construct();
    $this->source = ???;
    $this->destination = ???;
    $this->map = ???;
    $this->addFieldMapping(???, ???);
  }
}
```

You’ll probably have more than one Field Mapping.

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);
```

Able to extract the field names from the query. Descriptions are optional.

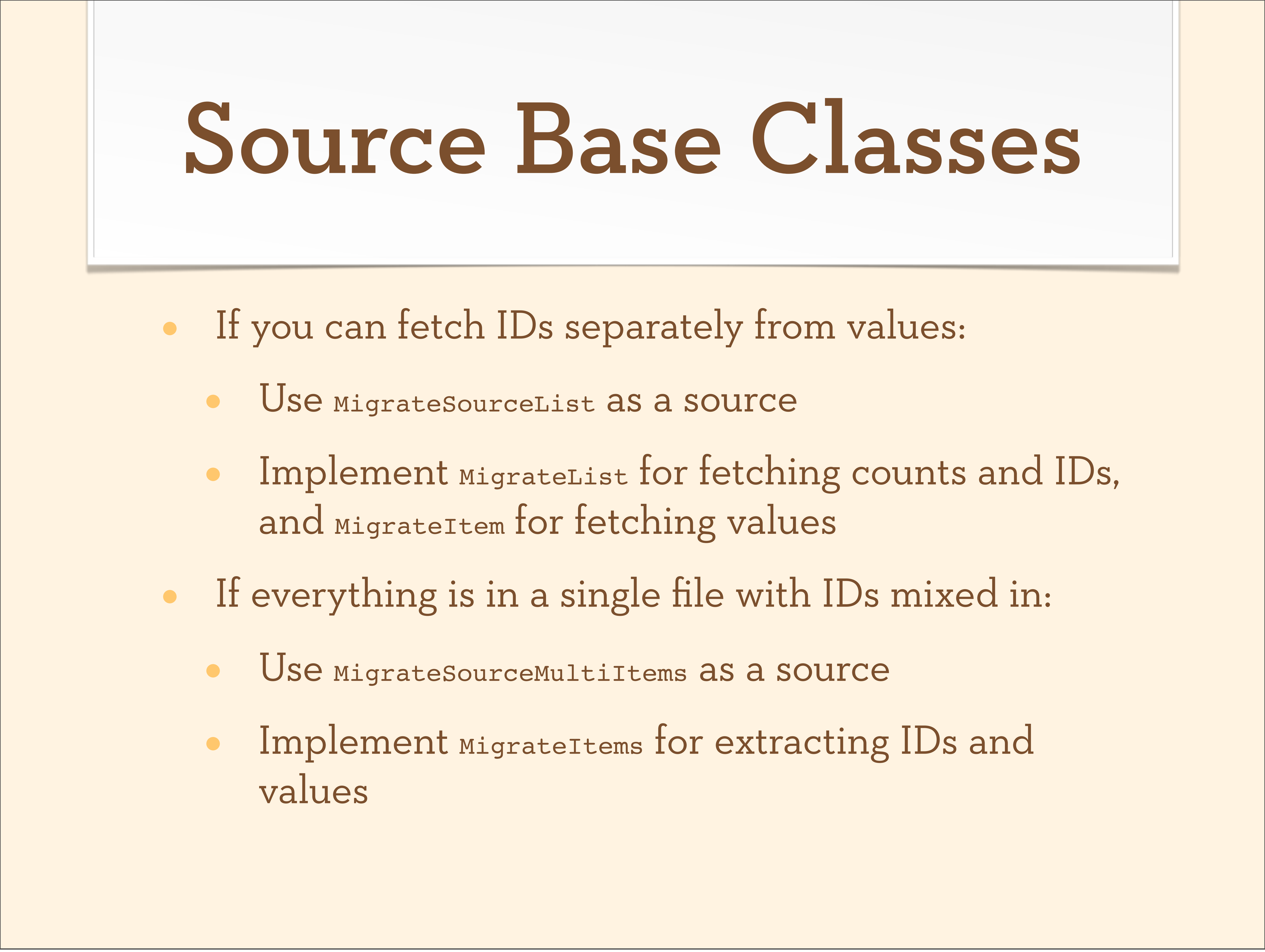
Or a CSV Source

```
// 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);
```

The CSV source can actually use the header row as column names but I’m not going to demonstrate that.

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

Wednesday, March 21, 12

I don’t really want to get too deep into this. I’m mentioning them so you’ll know they’re here when you discover you need them.

Migration Map

```
// inside __construct()

$this->map = new MigrateSQLMap($this->machineName,
  // You’ve got to describe your id’s schema.
  array(
    'style' => array(
      'type' => 'varchar',
      'length' => 255,
      'not null' => TRUE,
    )
  ),
  // Most Destinations provide a helper function:
  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();
```

Field Mappings

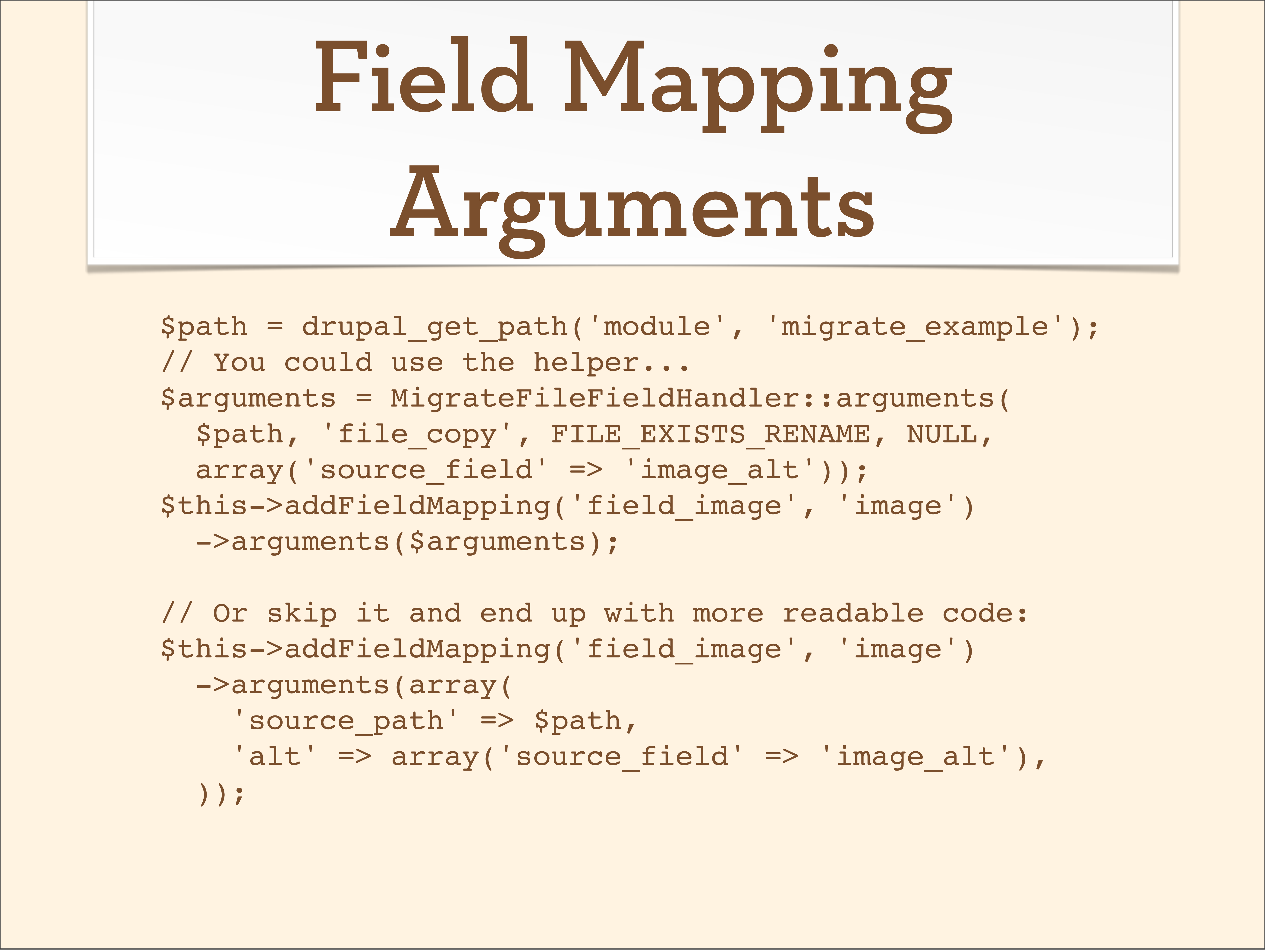
```
// inside __construct()

// Can be as simple as this…
$this->addFieldMapping('dest_name', 'source_name');

// …or you can chain options onto it…
$this->addFieldMapping('dest_name')
  ->defaultValue(1);

// Lets pretend we already setup $arguments.
$this->addFieldMapping('field_favbeers', 'beers')
  ->separator('|')
  ->arguments($arguments);
```

Field Mapping Arguments

- Often used to pass multiple source fields into a single destination field.
- First argument will be the `teaser` field, the second argument will always be one:

```
$this->addFieldMapping('body', 'body')
  ->arguments(array(
   'summary' => array('source_field' => 'teaser'),
   'format' => 1
  ));
```

By default the values passed into the arguments will be used as literals. If you use that wacky array syntax it'll replace it with the value from that field on the current row.



Field Mapping Arguments

```
$path = drupal_get_path('module', 'migrate_example');
// You could use the helper...
$arguments = MigrateFileFieldHandler::arguments(
  $path, 'file_copy', FILE_EXISTS_RENAME, NULL,
  array('source_field' => 'image_alt'));
$this->addFieldMapping('field_image', 'image')
  ->arguments($arguments);

// Or skip it and end up with more readable code:
$this->addFieldMapping('field_image', 'image')
  ->arguments(array(
    'source_path' => $path,
    'alt' => array('source_field' => 'image_alt'),
  ));
```

So two versions that do the same thing. I’m showing you this to demonstrate that some of the helpers actually obscure the functionality with out adding much value.

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');
```

But Wait, There's More!

- You can implement methods to customize the process: `prepareRow()`, `prepare()`, `complete()`.
- Yeah, it might be hard to remember which prepare does what.

prepareRow($row)

- Passes in the source row as an object so you can make modifications.
- Can indicate that a row should be skipped over during the import by returning `FALSE`.
- Add or change field values by modifying the properties:

```
$row->first = drupal_strtoupper($row->first);
$row->created = strtotime($row->access);
```

Need to mention that if you add a field in prepareRow() you should really add it to the Source’s field list.

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:

```
$entity->field_foo[‘und’][0][‘foo’] = $row->foo;
```

complete($ent, $row)

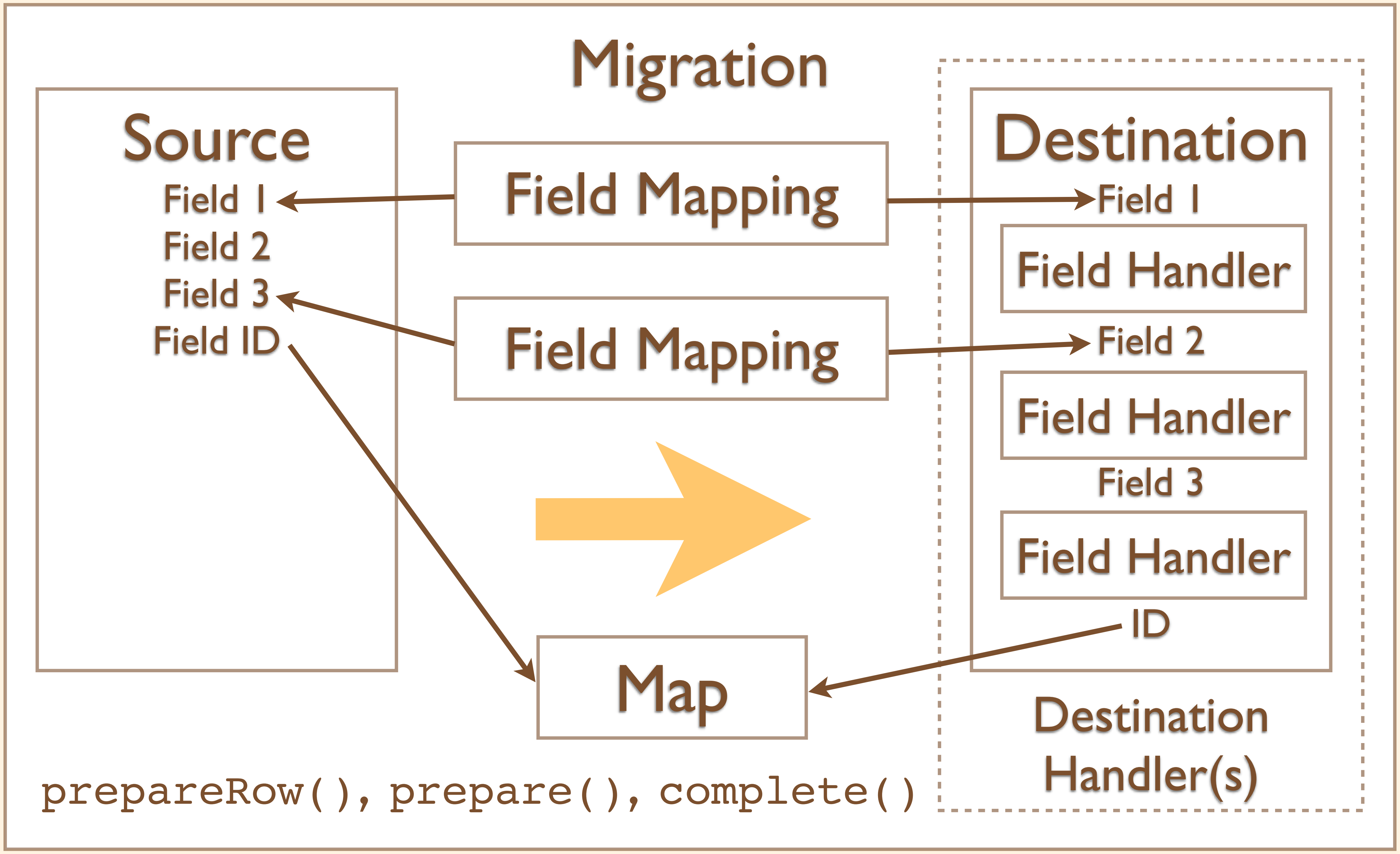
- Passes in the saved entity (with any ID values from auto increment fields) and the source row.
- This is the place if you need to update other records to reference the new entity.
- If you’re calling `node_save()` or `entity_save()` on the record you just imported, you’re probably doing something wrong.

Dealing With Circular Dependencies

- Break them by using stubs.
- Specify a `sourceMigration('BeerUser')` on the ID's field mapping.
- Add `createStub($migration, $source_key)` to `BeerUser` which creates an empty record and returns the new id.
- When `BeerUser` runs it'll update the stub and fill it with the real values.

Basically one migration asks another to look up the ID mapping, if it doesn't exists the second migration will create an empty record to get an ID so the first migration can complete. Then when the second migration runs it will update the dummy record that was created earlier.

Import Flow


Migration
Source
Field 1
Field 2
Field 3
Field ID
Field Mapping
Field Mapping
Map
Destination
Field 1
Field Handler
Field 2
Field Handler
Field 3
Field Handler
ID
Destination Handler(s)
prepareRow(), prepare(), complete()

Import Flow

- Source iterates until it locates an appropriate record, then calls the Migration's `prepareRow($row)` letting you modify or reject the data in `$row`.
- Migration applies the Field Mappings and the Field Handlers to convert the data in `$row` into `$entity`.
- Migration calls `prepare($entity, $row)` letting you modify the final entity using data from the Source's row.
- Destination saves the entity.
- Migration records the IDs into the Map then calls `complete()` so you can see the entity's final ID.

Supporting Migrate in Contrib

- If you're creating new objects (e.g. Webform Submissions) write a Destination.
- If you're writing field modules write a Field Handler.

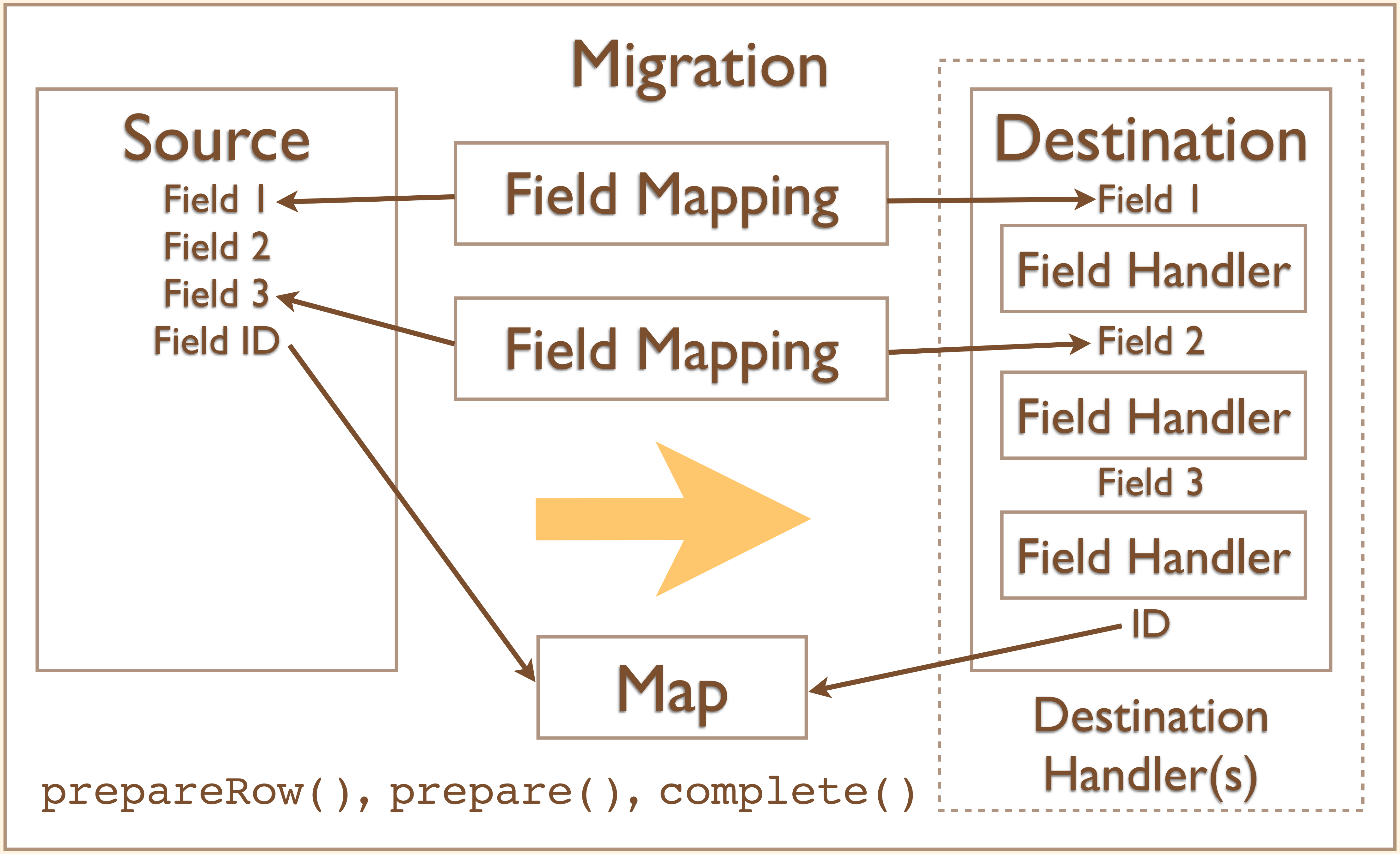
Creating Destinations

- 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

Creating Field Handlers

- Look at extending `MigrateSimpleFieldHandler` for single value fields.
- In the constructor register your field types.
- In `prepare()` convert the value into the format your field requires.
- Optionally, in `complete()` handle any followup tasks that require the entity's ID.

Questions?


Migration
Source
Field 1
Field 2
Field 3
Field ID
Field Mapping
Field Mapping
Map
Destination
Field 1
Field Handler
Field 2
Field Handler
Field 3
Field Handler
ID
Destination Handler(s)
prepareRow(), prepare(), complete()

DRUPALCON

DENVER

What did you think?

Locate this session on the DrupalCon Denver website

http://denver2012.drupal.org/program

Click the “Take the Survey” link.

Thank You!