Phased Migration to Drupal: The Nuts and Bolts

innovating
tomorrow's
government_

AQUILENT

Aquilent at a Glance

We move government ahead as the recognized leading provider of digital communications, cloud services and application development.

- Over 30 years of experience in DoD and Federal IT programs

More than 85 Federal Application Portals & Websites…

- AHRQ.gov
- USPS.com
- Usability.gov
- MyHealth.VA.gov
- Stopbullying.gov
- HHS.gov
- USA.gov
- FDA.gov
- CFTC.gov

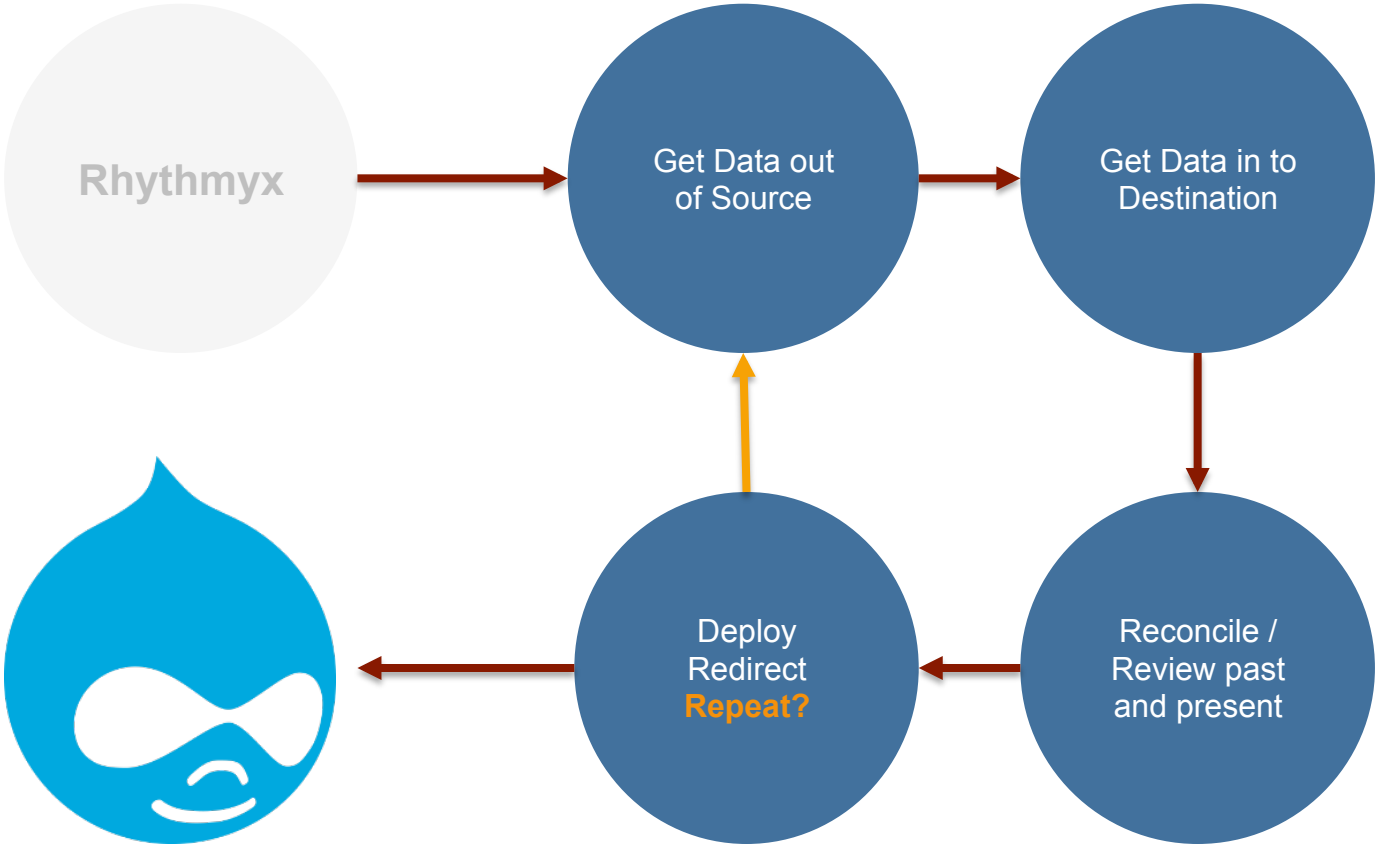
…and more than 50 Successful Cloud Projects

- Navy
- GSA
- HHS
- CMS
- USDA
- FDA
- Energy
- Library of Congress

Numerous Customer & Industry Awards

AQUILENT

Phased migration of Agency for Healthcare Research and Quality (ahrq.gov)



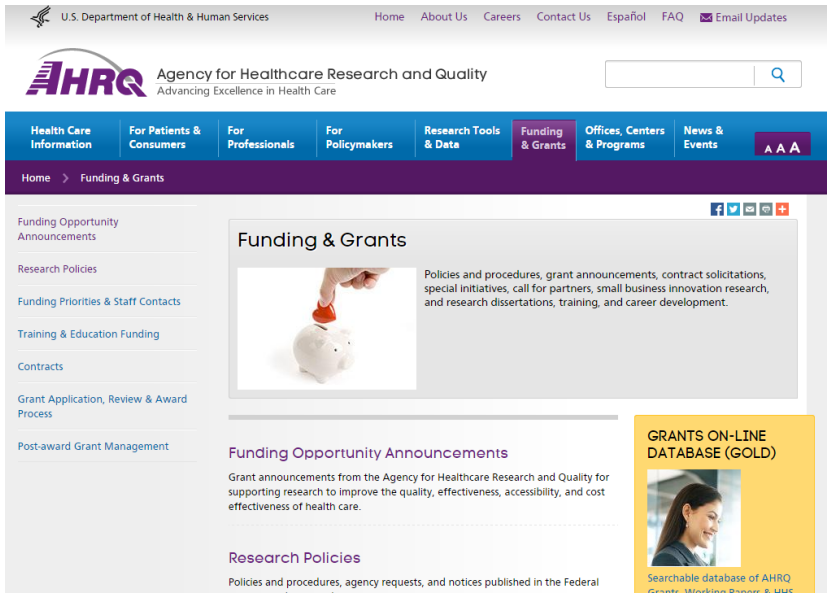
AQUILENT

Session Agenda

- **Bird's eye view of the process**
  - What triggered our migration
  - Migration strategy
    - BIG SWITCH vs PHASED
  - Technology enablers
    - API/Scraper, Migration Module, CDN
- **Deep dive of the Migrate module**
  - The various forms of data that Migrate module can consume.
  - Basic structure of a Migrate module script
  - Analysis of the many callbacks provided by Migrate to manipulate the migration process and how we used them.

AQUILENT

Why migrate?



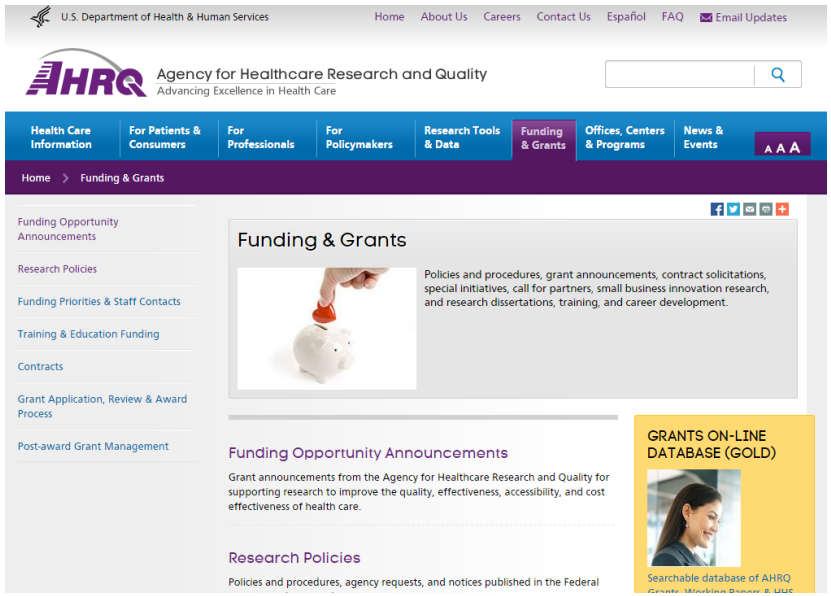
Technology obsolescence.

Difficulty of maintenance.

Speed of development.

AQUILENT

The Old System

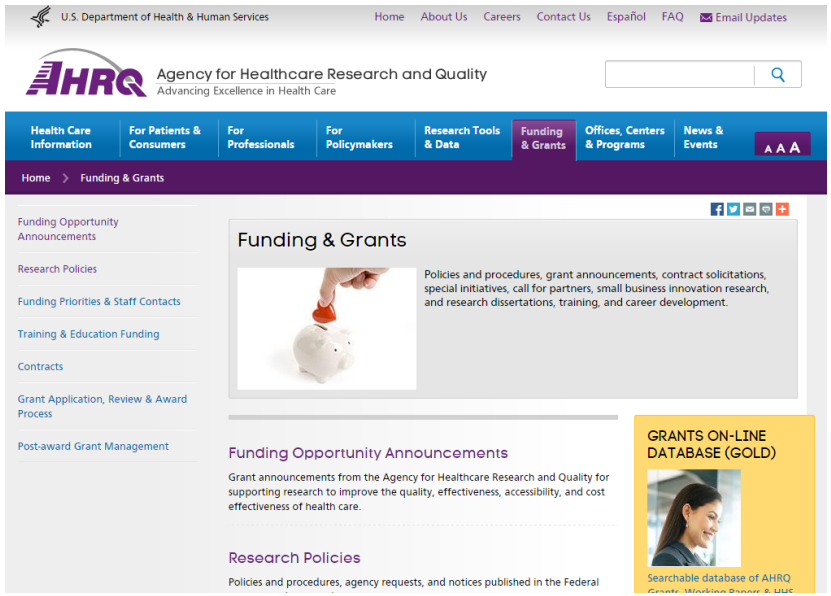


~80 “content types” or “templates”

~100,000 artifacts before vetting / archiving

AQUILENT

The New System



Content types: Generic, Landing Page, Publications, Case Studies ….

~10 Drupal Content Types

~10s of thousands artifacts archived

Major development is wrapped up.

Several sections have gone live.

AQUILENT

Why phased migration?

Big switch vs Phased approach

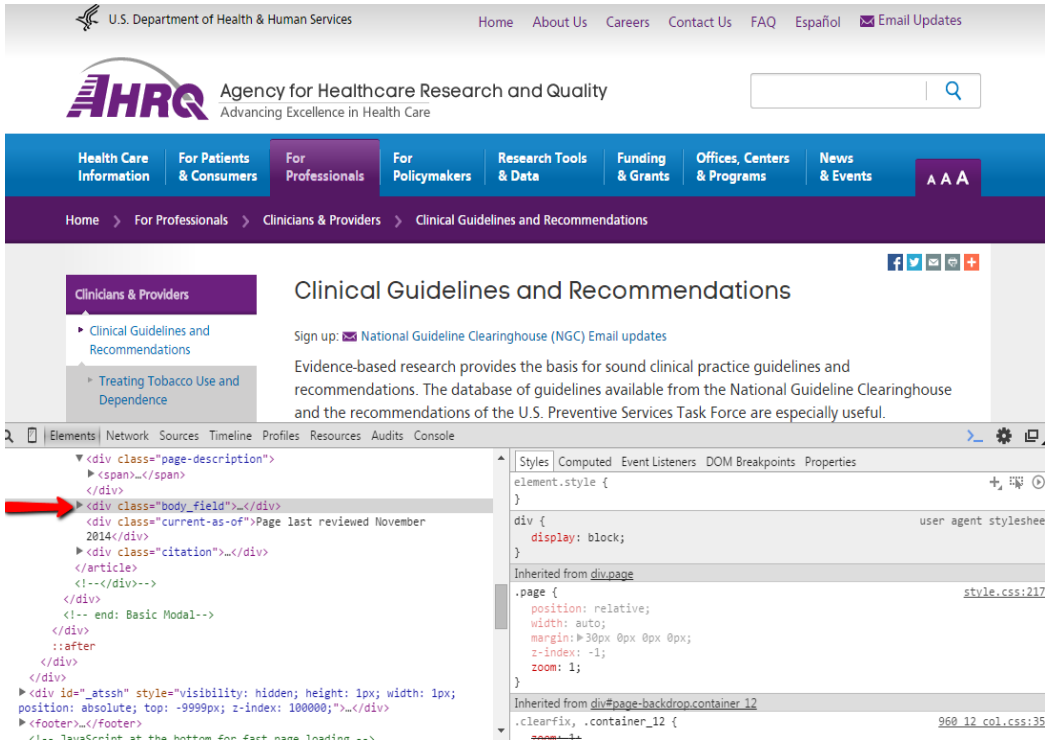
How do you freeze a live site with over 50,000 "active" artifacts? What happens on D-Day?

Phased migration reduces the content vetting process to manageable chunks.

On a large live site that's constantly changing, a prolonged content vetting process is not realistic.

AQUILENT

Challenge: getting the data out of Rhythmyx?



API vs Scraping

Depends on how much control do we have over the Source?

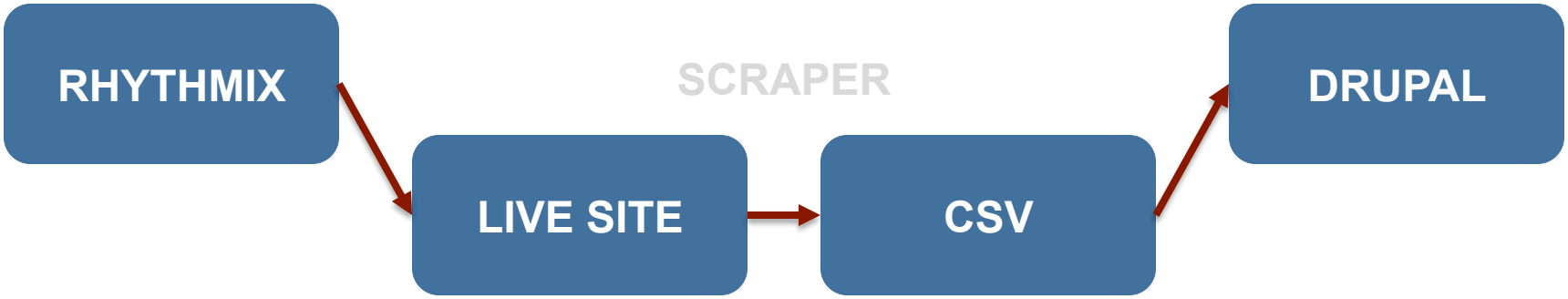
How robust is the API?

How flexible is the Scraper? It has to bend with the rules and whimsy of the contents.

Technology Enablers #1

AQUILENT

Source to Destination – via CSV



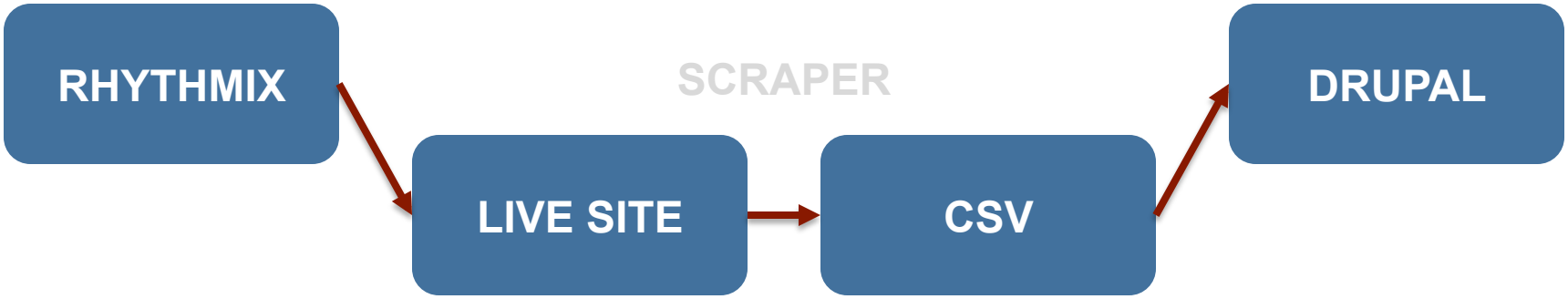
We chose to use CSVs generated with a custom scraper.

Scraper in .NET. It has powerful libraries for file system browsing, filtering etc. plus a very powerful, open source HTML DOM parser, and some powerful debugging tools.

CSV was like a half-way house, where we discovered Content Quirks, encoding issues. Also, allowed us to work in parallel.

AQUILENT

Source to Destination – via CSV



Other alternatives:

PHP Simple HTML DOM Parser: http://simplehtmldom.sourceforge.net/,

PHP's Dom module: DOMDocument();

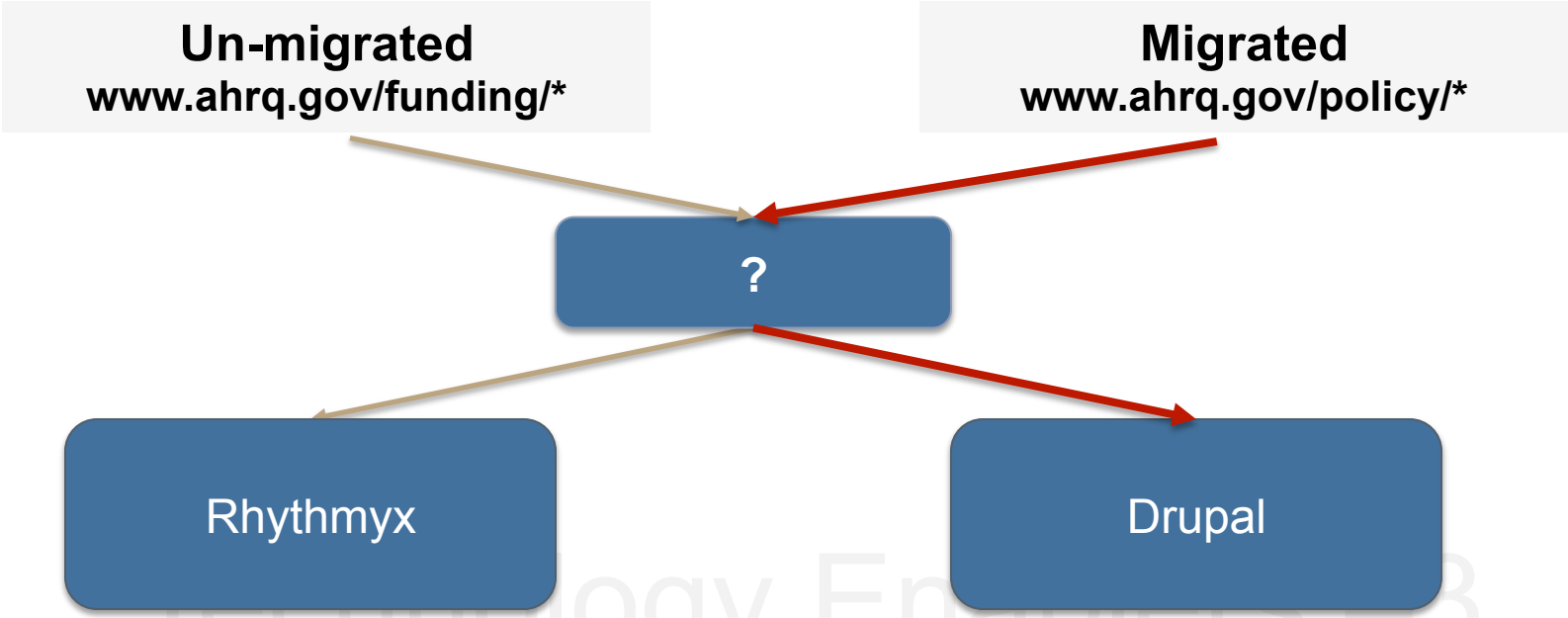
Scrapy: http://scrapy.org/

PHP Crawler: http://sourceforge.net/projects/php-crawler/

AQUILENT

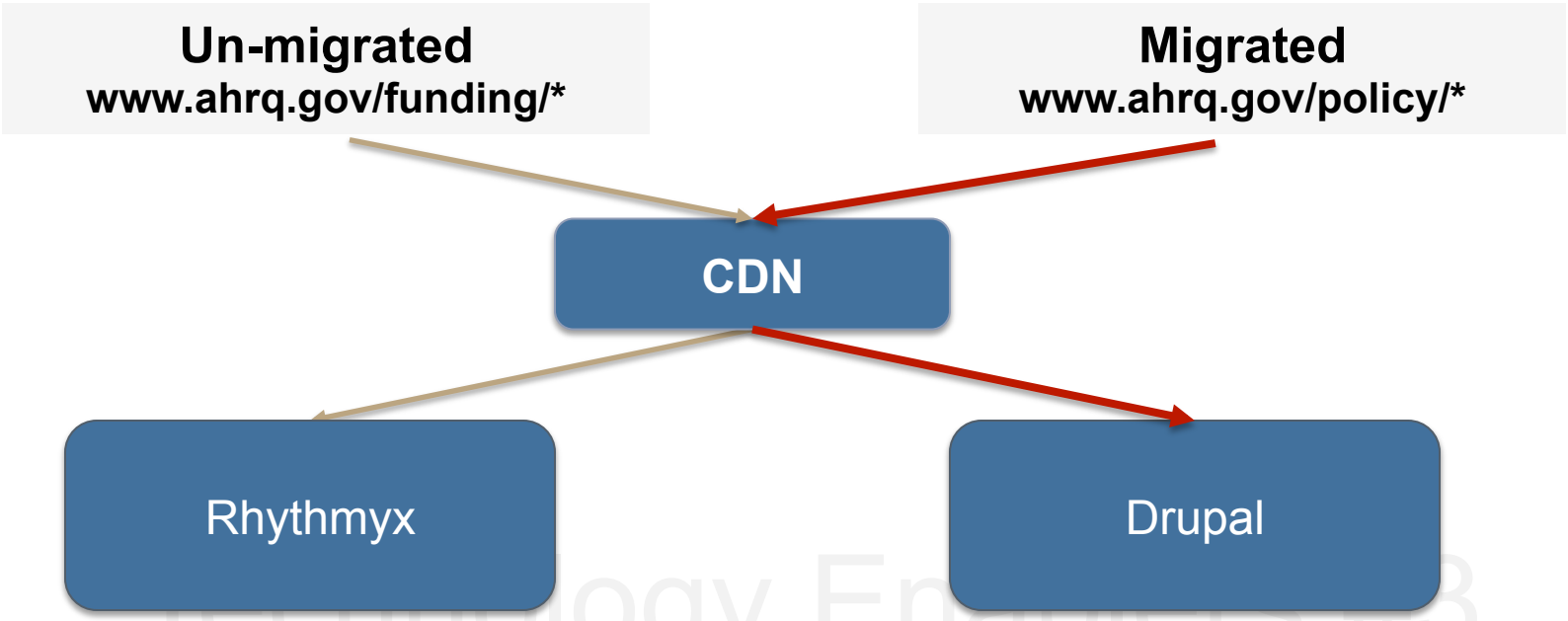
Challenge: making it transparent to user?

In a phased migration, some of the live site's content will be on the old CMS and some will be on the new CMS. The typical user must not be aware of it.

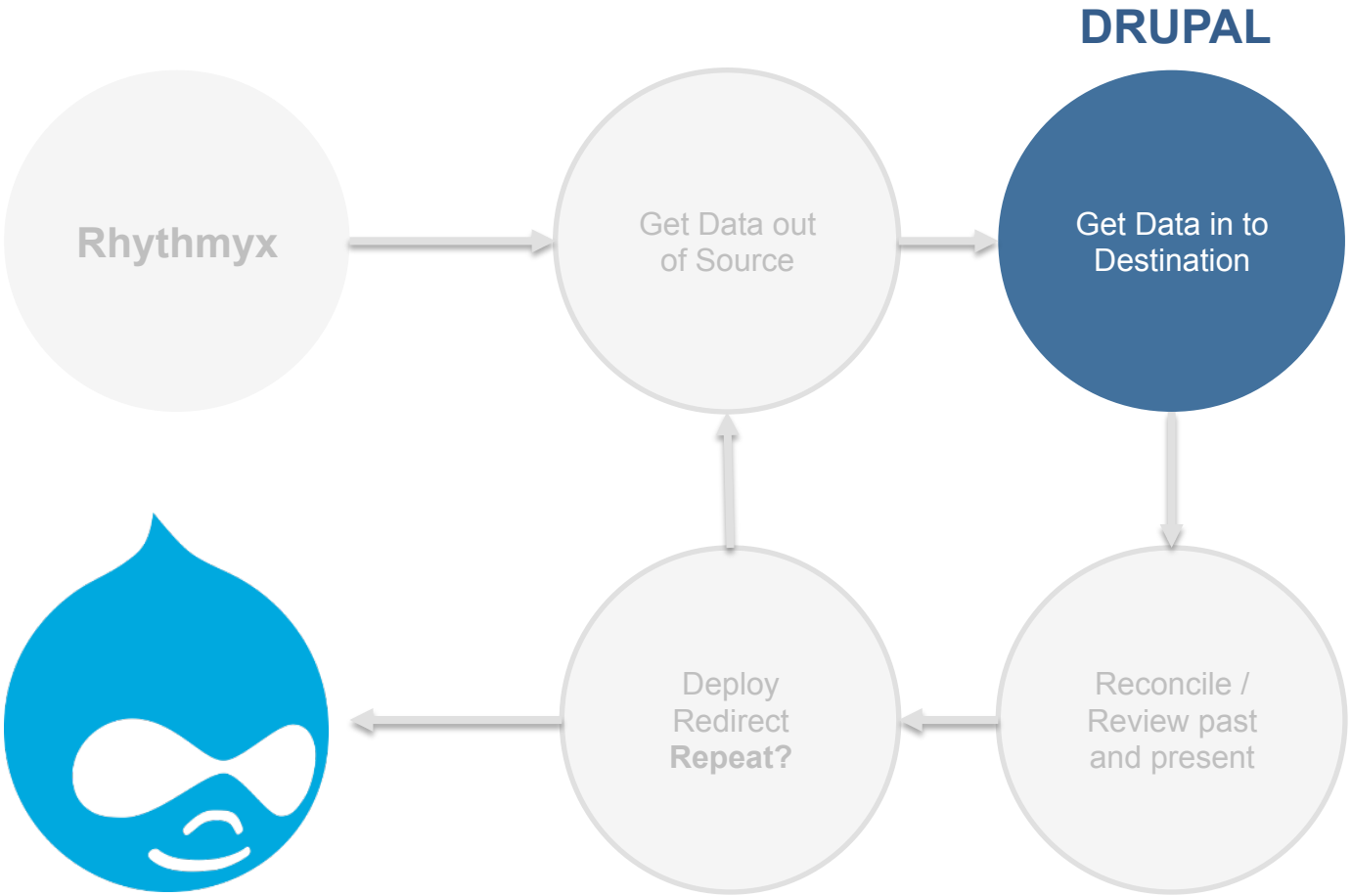


Challenge: making it transparent to user?

You may minimally use a configurable load balancer, or, a CDN that supports multiple origins via path rules.

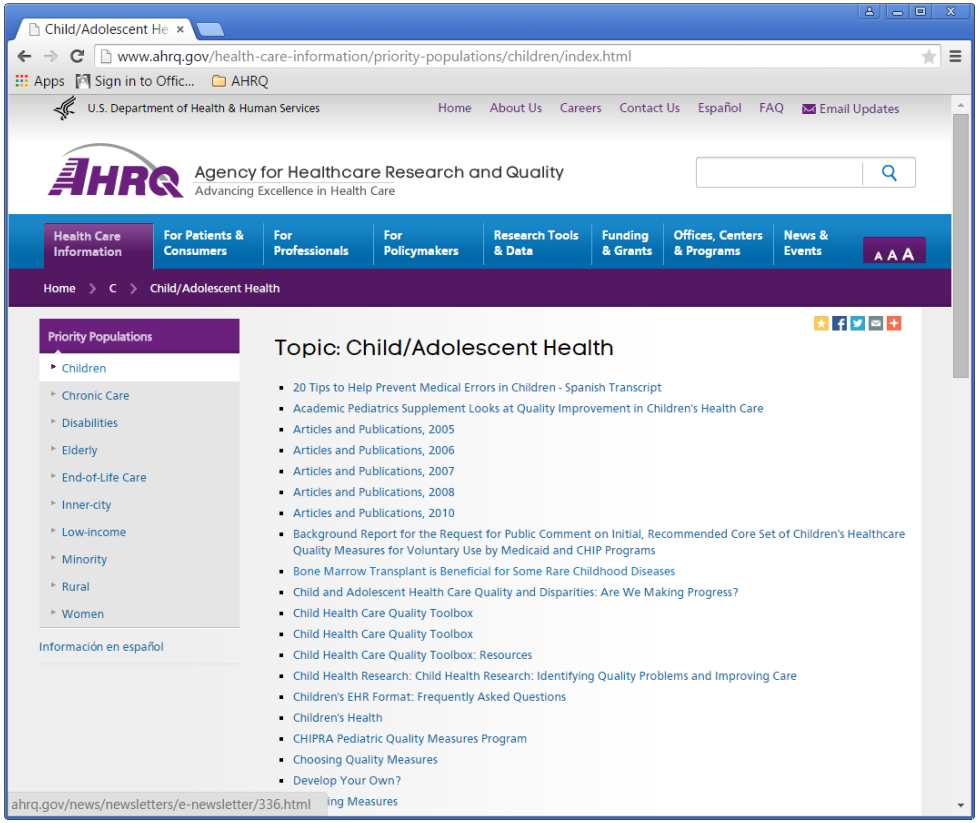


Deep dive: Migrate Module



AQUILENT

Challenge: getting the data into Drupal?



A Project Specific Challenge: Taxonomy

The Topics section pages are lists of pages from across the entire site. Categorized by topic.

This section needed to go online early in the migration. Even though much of it's referenced content would be imported later.

So the initial migration had to include the Topics taxonomy along with every page in the site including...

Page title, Topic term assignments, URL

Follow up migrations would then import the remaining content. Replacing generic nodes with more specific content types as necessary.

Technology Enablers #2

AQUILENT

Challenge: getting the data into Drupal?

There are basically three ways.

1. **Feeds Module**

2. **Custom Script**

3. **Migrate Module**

Technology Enablers #2

AQUILENT

Challenge: getting the data into Drupal?

There are basically three ways.

1. **Feeds Module:** Good for simple migrations. Driven from a UI, but it is not as powerful as Migrate module.
2. **Custom Script**
3. **Migrate Module**

Technology Enablers #2

AQUILENT

Challenge: getting the data into Drupal?

There are basically three ways.

1. **Feeds Module**

2. **Custom Script:** The sky's the limit but this can quickly become unmanageably complicated.

3. **Migrate Module**

Technology Enablers #2

AQUILENT

Challenge: getting the data into Drupal?

There are basically three ways.

1. **Feeds Module**

2. **Custom Script**

3. **Migrate Module:** Full featured and powerful. Almost as powerful as a custom script. Migrate has been included in Drupal 8 core and will be the de-facto methodology for data migrations and major version upgrades.
   www.drupal.org/project/migrate

Technology Enablers #2

AQUILENT

Basic Migrate Module Flow

1. **Read in source data**
2. **Pre-process source data** (optional)
3. **Map the data to Drupal**
4. **Pre-process Drupal data** (optional)
5. **Save to Drupal**
6. **Post-process Drupal data** (optional)

AQUILENT

Source

Migrate can consume source data in many ways.

Built-in classes include…

- **SQL** (Migrating from a Drupal site)
- **CSV**
- **XML**
- **JSON**
- **Two Part Lists** (Primary key list, content data list)
- **MultiItems** (Single source file with both list and items)
- **MS SQL**
- **Oracle DB**
- **MongoDB**
- **HTML** (Worst case scenario. Migration equivalent of a Hail Mary Pass)
- **Custom Source Class**

AQUILENT

Source

Migrate can consume source data in many ways.

Built-in classes include…

- **SQL** (Migrating from a Drupal site)
- **CSV**
- **XML**
- **JSON**
- **Two Part Lists** (Primary key list, content data list)
- **MultiItems** (Single source file with both list and items)
- **MSSQL**
- **Oracle**
- **MongoDB**
- **HTML** (Worst case scenario. Migration equivalent of a Hail Mary Pass)
- **Custom Source Class**

AQUILENT

Migration Module Basics - Destinations

Migrate provides a full assortment of destination classes

- Role
- User
- Term
- Node
- Comment
- File
- Revision
- Menus
- Table (Catch all class for the scenarios when the other classes don't apply)

AQUILENT

Migration Module Basics - Destinations

Migrate provides a full assortment of destination classes

- Role
- User
- Term
- Node
- Comment
- File
- Revision
- Menus
- Table (Catch all class for the scenarios when the other classes don't apply)

AQUILENT

Migration Module Basics - Field Mapping

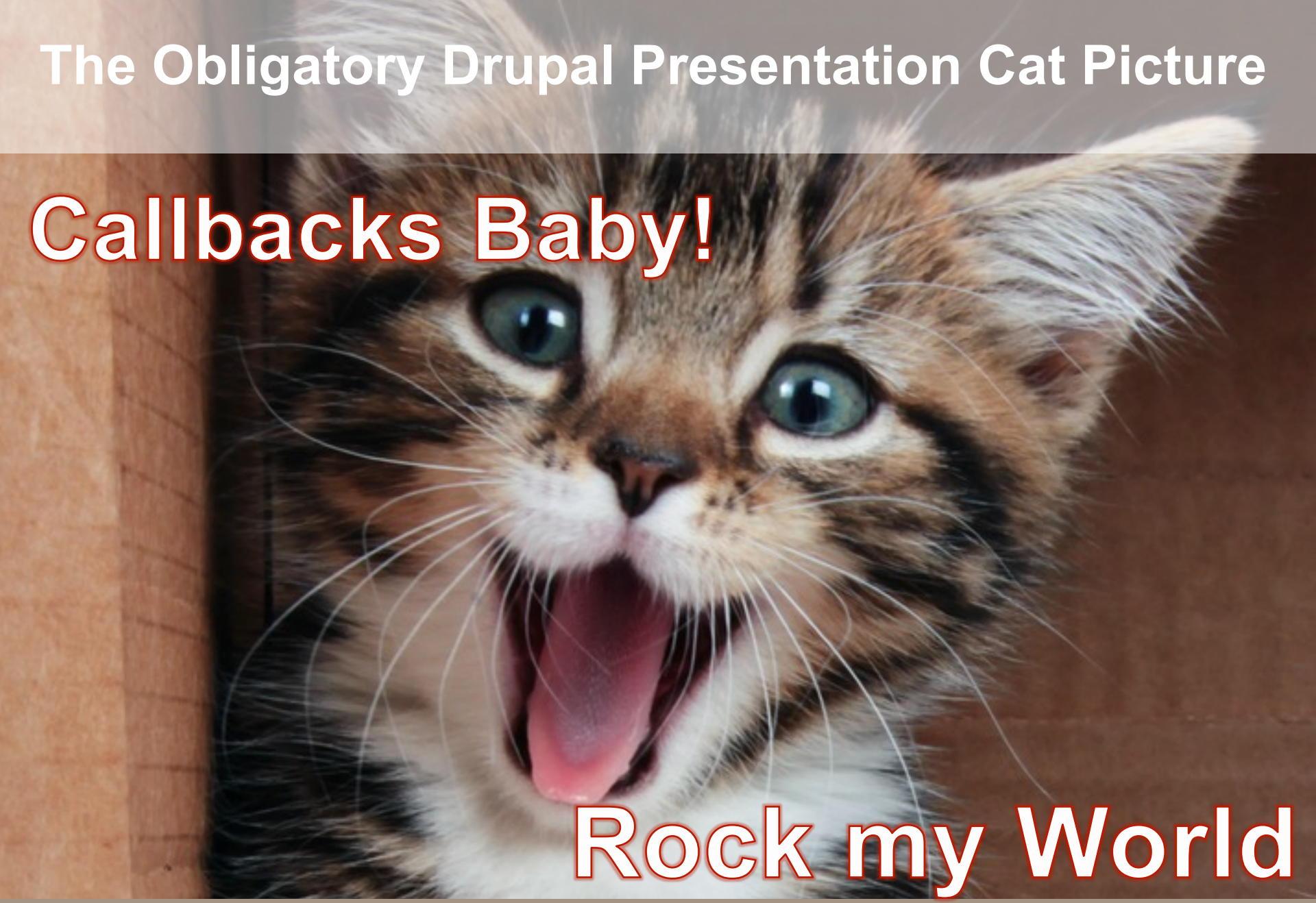
Migrate makes it easy to map source data to destination fields.

It's also provides special handling for a variety of processes directly in each field mapping

- **Arguments (field properties)**
- **Default values**
- **Multiple value processing**
- **Sub fields**
- **Deduping**
- **Callbacks**

AQUILENT

The Obligatory Drupal Presentation Cat Picture



AQUILENT

Migration Module Basics - Callbacks

- **preImport():** Called once just as the migration begins.
- **prepareRow():** After the source data has been read but before it's been processed.
  - Data validation
  - Data scrubbing
  - Swapping nodes between content types
- **Mapping callbacks:** Process source data per field just before it's mapped to its destination field
  - Data scrubbing
  - Node ownership assignment
- **prepare():** After the Drupal object has been built, but before it's saved.
  - Swapping nodes between content types
  - Housekeeping
- **complete():** After the Drupal object has been saved.
  - Swapping nodes between content types
- **postImport():** Called once at the end of the migration
  - Housekeeping
- **createStub():** A special function that Migrate calls when a reference is made to an record that hasn't been imported yet.

AQUILENT

So How Did We Use All of This?

We had 5 different types of Migrations

1. **Users**
2. **Menu**
3. **Topics (Taxonomy)**
4. **Generic nodes**
5. **Specialty nodes**

AQUILENT

So How Did We Use All of This?

We had 5 different types of Migrations

1. Users
2. Menu
3. Topics (Taxonomy)
4. **Generic nodes**
5. **Specialty nodes**

We are focusing on the last two.

AQUILENT

Some Real World Examples

U.S. Department of Health & Human Services | Home | About Us | Careers | Contact Us | Español | FAQ | Email Updates

AHRQ Agency for Healthcare Research and Quality
Advancing Excellence in Health Care

Health Care Information | For Patients & Consumers | For Professionals | For Policymakers | Research Tools & Data | Funding & Grants | Offices, Centers & Programs | News & Events | A A A

Home > Policies > Freedom of Information Act (FOIA)

- AHRQ Equal Employment Opportunity Policy
- Guideline User Policies for Electronic Versions
- Freedom of Information Act (FOIA)
- Social Media Standards and Policies

Freedom of Information Act (FOIA)

Information about the Freedom of Information Act (FOIA) as pertains to AHRQ.

Welcome to the Agency for Healthcare Research and Quality (AHRQ) Freedom of Information Act page. We hope you find that it facilitates your efforts to obtain public information about AHRQ programs.

Information Resources

- Please check to see if the information you are looking for is already available on AHRQ's Web Site before submitting a FOIA request. Start with our Frequently Asked Questions page. You can also use our Search and Browse features or our Publications & Products page to help you find what you are looking for.
- For current program information and to facilitate your efforts to obtain public information about the Department of Health and Human Services (HHS) programs, go to the HHS FOIA Home Page.
- For a general overview of the Freedom of Information Act, read the United States Department of Justice Guide to the Freedom of Information Act.

FOIA CONTENT

- Submit a FOIA Request
- FOIA Fees
- Electronic Library
- Annual Reports
- Privacy Act
- Reference Materials
- Contacts

HHS FOIA Requester Service Center

- AoA - Administration on Aging
- ACF - Administration for Children and Families
- AHRQ - Agency for Healthcare Research and Quality
- CDC - Centers for Disease Control and

th & Human Services | Home | About Us | Careers | Contact Us | Español | FAQ | Email Updates

gency for Healthcare Research and Quality
vancing Excellence in Health Care

nts & ers | For Professionals | For Policymakers | Research Tools & Data | Funding & Grants | Offices, Centers & Programs | News & Events | A A A

Research Policies > Funding Opportunity Announcement (FOA) Guidance

- Federal Regulations & Authorities
- Federal Register Notices
- Public Access to Federally Funded Research
- Funding Priorities & Staff Contacts
- Training & Education Funding

Funding Opportunity Announcement (FOA) Guidance

This page contains AHRQ policy information and guidance about procedures related to grants.

AHRQ Grants Policy and Guidance
AHRQ Research Priorities for R01, R18, and R03 Applications

AHRQ Grants Policy and Guidance

Human Subjects Protection

Federal regulations at 45 CFR Part 46 require that applications and proposals involving human subjects research must be evaluated in accordance with those regulations, with reference to the risks to the subjects, the adequacy of protection against these risks, the potential benefits of the research to the subjects and others, and the importance of the knowledge gained or to be gained.

Standards for Privacy of Individually Identifiable Health Information

The HIPAA Privacy Rule, 45 CFR Parts 160 and 164, is administered and enforced by the HHS Office for Civil Rights (OCR). The OCR Web site provides information on the Privacy Rule, including a complete Regulation Text and a set of decision tools that may be used to determine whether a researcher is a staff member of a covered entity.

Access to Research Data Through the Freedom of Information Act

The Generic Content Type

AQUILENT

Some Real World Examples

U.S. Department of Health & Human Services

Home | About Us | Careers | Contact Us | Español | FAQ | Email Updates

AHRQ Agency for Healthcare Research and Quality
Advancing Excellence in Health Care

Health Care Information | For Patients & Consumers | For Professionals | For Policymakers | Research Tools & Data | Funding & Grants | Offices, Centers & Programs | News & Events | A A A

Home > Funding & Grants

- Funding Opportunity Announcements
- Research Policies
- Funding Priorities & Staff Contacts
- Training & Education Funding
- Contracts
- Grant Application, Review & Award Process
- Post-award Grant Management

Funding & Grants

Policies and procedures, grant announcements, contract solicitations, special initiatives, call for partners, small business innovation research, and research dissertations, training, and career development.

Funding Opportunity Announcements

Grant announcements from the Agency for Healthcare Research and Quality for supporting research to improve the quality, effectiveness, accessibility, and cost effectiveness of health care.

Research Policies

Policies and procedures, agency requests, and notices published in the Federal

GRANTS ON-LINE DATABASE (GOLD)

Searchable database of AHRQ

The Landing Page

AQUILENT

The Basic Files Involved

Migrations are always built as custom modules. The required files are

1. **mymigration.module**
2. **mymigration.info**
   - These are required by Drupal to recognize the module. Migrate doesn't even use the first one. It's completely empty.

3. **mymigration.migrate.inc**
   - This is where you declare the migration.
4. **mymigration.inc**
   - This is where all your code goes, beginning with the constructor method.

AQUILENT

The Basic Files Involved

Migrations are always built as custom modules. The required files are

1. **mymigration.module**
2. **mymigration.info**
   - These are required by Drupal to recognize the module. Migrate doesn't even use the first one. It's completely empty.

3. **mymigration.migrate.inc**
   - This is where you declare the migration.
4. **mymigration.inc**
   - This is where all your code goes, beginning with the constructor method.

AQUILENT

Migration Core Code

mymigration.migrate.inc

```
function ahrq_migrate_generic_migrate_api() {
  $api = array(
    'api' => 2,
    'migrations' => array(
      'AhrqGeneric' => array('class_name' => 'AhrqGeneric'),
    ),
  );
  return $api;
}
```


AQUILENT

The Basic Files Involved

Migrations are always built as custom modules. The required files are

1. **mymigration.module**
2. **mymigration.info**
   - These are required by Drupal to recognize the module. Migrate doesn't even use the first one. It's completely empty.
3. **mymigration.migrate.inc**
   - This is where you declare the migration.
4. **mymigration.inc**
   - This is where all your code goes, beginning with the constructor method.

AQUILENT

Migration Core Code

mymigration.inc

Constructor method including…

1. **Primary key definition**
   - Migrate creates mapping tables for each migration that records the relationship between the source data's primary key and the primary key of the matching Drupal object. At the very least this allows Migrate to perform updates and rollbacks, but we'll tap into this data for our own purposes.
2. **Source definition**
3. **Destination definition**
4. **Field mappings.**
5. **Optional callback functions (exist outside the contructor method)**

AQUILENT

Migration Core Code

Constructor Method

```
class AhrqGeneric extends DynamicMigration {
    public function __construct() {
        parent::__construct();
        $this->map = new MigrateSQLMap(
            $this->machineName, array(
                'source_id' => array(
                    'type' => 'int',
                    'not null' => TRUE,
                    'alias' => 'import'
                )
            ), MigrateDestinationNode::getKeySchema()
        );
        // Source declaration here
        // Destination declaration here
        // Field mappings here
   }
   // The rest of your migration code will follow here. Callbacks, custom functions, etc.
}
```


AQUILENT

Migration Core Code

mymigration.inc

Constructor method including…

1. **Primary key definition**
   - Migrate creates mapping tables for each migration that records the relationship between the source data's primary key and the primary key of the matching Drupal object. At the very least this allows Migrate to perform updates and rollbacks, but we'll tap into this data for our own purposes
2. **Source definition**
3. **Destination definition**
4. **Field mappings.**
5. **Optional callback functions**

AQUILENT

Migration Core Code

Source declaration

```
$this->source = new MigrateSourceCSV(
    dirname(__FILE__) . "/mymigration.csv",
    $this->csvcolumns(), // Function that maps csv columns to machine names (NOT field names)
    array('header_rows' => 1)
);

public function csvcolumns() {
    $columns = array(
         0 => array('ahrq_rythmx_id', 'DCTERMS.identifier'),
         1 => array('path', 'Path'),
         2 => array('dcterms_title', 'DCTERMS.title'),
         And so on...
    );
    return $columns;
}
```

Destination declaration

```
$this->destination = new MigrateDestinationNode('content_type_name');
```


AQUILENT

Migration Core Code

mymigration.inc

Constructor method including…

1. **Primary key definition**
   - Migrate creates mapping tables for each migration that records the relationship between the source data's primary key and the primary key of the matching Drupal object. At the very least this allows Migrate to perform updates and rollbacks, but we'll tap into this data for our own purposes
2. **Source definition**
3. **Destination definition**
4. **Field mappings.**
5. **Optional callback functions**

AQUILENT

Migration Core Code

Field Mapping (Examples)

* A basic mapping

```
$this->addFieldMapping('path', 'path_src');
```

* Setting a default value

```
$this->addFieldMapping('status')
    ->defaultValue(1);
$this->addFieldMapping(field_contact_email‘, ‘contact_email’)
    ->defaultValue(‘info@ahrq.gov’);
```

* A multi-value field, for a term, followed by a flag to create new terms if necessary.

```
$this->addFieldMapping('topics_taxonomy', 'topics_src')
    ->separator('|');
$this->addFieldMapping('topics_taxonomy:create_term')
    ->defaultValue(TRUE);
```


AQUILENT

Migration Core Code

Field Mapping (Examples)

Setting a field property, in this example, a text format.

```
$this->addFieldMapping('generic_body', 'src_body')
    ->arguments(array('format' => 'full_html'));
```

Using Dedupe.

```
$this->addFieldMapping('name', 'src_username')
    ->dedupe ('users' => 'name');
```

Some callback examples.

```
$this->addFieldMapping('date_created', 'src_created')
    ->callbacks(array($this, 'dateToTimestamp'));
$this->addFieldMapping('uid', 'src_author')
    ->callbacks(array($this, 'assignAuthor'));
```


AQUILENT

Migration Core Code

Field Mapping (Some field callback functions).

```
protected function decodeString($value) {
    return html_entity_decode($value, ENT_QUOTES, 'UTF-8');
}

protected function dateToTimestamp($value) {
    return strtotime($value);
}

protected function assignAuthor($email_address) {
    $uid = db_query("SELECT uid FROM {users} WHERE mail = :mail",
                    array(':mail' => $email_address))->fetchCol(0);
    if ($uid[0]) {
        return $uid[0];
    } else {
        return 1;
    }
}
```


AQUILENT

Migration Core Code

mymigration.inc

Constructor method including…

1. **Primary key definition**
   - Migrate creates mapping tables for each migration that records the relationship between the source data's primary key and the primary key of the matching Drupal object. At the very least this allows Migrate to perform updates and rollbacks, but we'll tap into this data for our own purposes
2. **Source definition**
3. **Destination definition**
4. **Field mappings.**
5. **Callback functions**

AQUILENT

Callback Generic Page: prepareRow()

The source data has been read into memory

```
public function prepareRow($row) {
    // Ignore specific records that will be migrated manually
    $ignoreSrcRecords = array('82008','84001');
    if (in_array($row->source_id, $ ignoreSrcRecords)) {
        // $this->GenericMapsToDelete is a class variable
        // available to every function
        $this->GenericMapsToDelete[] = $row->source_id;
        watchdog(
            'migrate_generic',
            'The Source record @rid was ignored on migration because
             it will be migrated manually.',
             array('@rid' => $row->source_id), WATCHDOG_NOTICE);
        return FALSE;
    }
    return TRUE;
}
```


AQUILENT

Callback Generic Page: prepareRow(), cont.

The source data has been read into memory

```
public function prepareRow($row) {
  // Seach all non-generic mapping tables, looking for this record. If it exists SKIP this record.
  $migrateMaps = array('migrate_map_landing', 'migrate_map_publication');
  foreach($migrateMaps as $map) {
    $is_table = db_query("SHOW TABLES LIKE :table", array(':table' => $map))->fetchCol(0);
      if ($is_table[0]) {
        $result = db_query("SELECT `destid1` FROM {" . $map . "} WHERE sourceid1 = :sid",
                           array(':sid' => $row->source_id))->fetchCol(0);
        if ($result) {
          // $ this->GenericMapsToDelete is a class variable available to every function
          $this->GenericMapsToDelete[] = $row->source_id;
          watchdog(
            'migrate_generic',
            'The Souce record @rid was ignored on migration because it has been imported
            as another content type.', array('@rid' => $row->source_id), WATCHDOG_NOTICE);
          return FALSE;
        }
      }
    }
  return TRUE;
}
```


AQUILENT

Callback Generic Page: prepare()

The Drupal node is created, but has not been saved.

```
function prepare($entity, stdClass $row) {
    $entity->menu['enabled'] = TRUE;
    $txtFields = array('generic_body','field_catch_all');
    foreach($txtFields as $f) {
        if (isset($entity->{$f}[LANGUAGE_NONE][0]) &&
            $entity->{$f}[LANGUAGE_NONE][0]['value'] == '') {
                unset($entity->{$f}[LANGUAGE_NONE]);
        }
    }
}
```


AQUILENT

Callback Landing Page: prepare()

The Drupal node is created, but has not been saved.

```
public function prepare($entity, stdClass $row) {
    $this->mlid = FALSE; // Class variable, available to every function.
    $entity->menu['enabled'] = TRUE;
    // Find the placeholder node, if any.
    $results = db_query("SELECT * FROM {migrate_map_generic} WHERE sourceid1 = :sid",
                    array(':sid' => $row->source_id));
    foreach ($results as $result) {
        if ($result->destid1) {
            // Temporarily re-assign the menu item to the home page
            $this->mlid = db_query("SELECT mlid FROM {menu_links} WHERE link_path = :path",
                            array(':path' => 'node/' . $result->destid1))->fetchCol(0);
            if (isset($this->mlid[0])) {
                db_update('menu_links')->fields(array(
                        'link_path' => '<front>', 'router_path' => '',
                    ))->condition('mlid', $this->mlid[0])->execute();
            }
            // Delete the placeholder node if any and its Migrate map.
            node_delete($result->destid1);
            db_delete('migrate_map_generic')->condition('sourceid1', $result->sourceid1)->execute();
        }
    }
}
```


AQUILENT

Callback Landing Page: complete()

The Drupal node has been saved.

```
public function complete($entity, stdClass $row) {
    // $this->mild is a class variable that was populated
    // in the prepare() function.
    if (isset($this->mlid[0])) {
        db_update('menu_links')
            ->fields(array(
                'link_path' => 'node/' . $entity->nid,
                'router_path' => 'node/%',
            ))
            ->condition('mlid', $this->mlid[0])
            ->execute();
    }
}
```


AQUILENT

Callback Landing Page: postImport()

postImport(): Runs once at the end of the script

```
public function postImport() {
    parent::postImport();
    // After the last row has been imported, Delete any mappings for un-imported records
    // $this->LandingMapsToDelete is a class variable
    // populated during validation in prepareRow().
    if (count($this->LandingMapsToDelete)) {
        foreach($this-> LandingMapsToDelete as $sid) {
            $map_deleted = db_delete('migrate_map_landing')
                ->condition('sourceid1', $sid)
                ->execute();
        }
        watchdog(
            'migrate_landing',
            '@count mapping records were deleted from the table
             "migrate_map_landing" because they were ignored by the migration.',
            array('@count' => count($this-> LandingMapsToDelete )), WATCHDOG_NOTICE);
    }
}
```


AQUILENT

Thank You

Chris Desautels

Lead Technical Consultant

chris.desautels@aquilent.com

Adil Faisal

Lead Technical Consultant

adil.faisal@aquilent.com

AQUILENT

Thank You

Questions?

Don’t be shy.

AQUILENT