Concept - Migration M5

Your Rating: Results: 111 rates

- Goal
- A bit of History
  - Migration Module 1.1.x
    - Key features
    - Steps to run
    - Issues faced
  - Migration Module 1.2.x
    - Improvements
    - Key Features
    - Step to run
    - Important improvements
- M5 migration
  - Scenarios
  - Big pictures
  - Applications
    - Report App
    - M5 Migration App
  - Tasks
    - Report App
    - M5 Migration app:
- Already implemented migration task (M5)
  - DAM migration
  - Node
  - Dialog
  - Data Module
  - MultiSite
- Modules not planned to be migrated to M5
  - Workflow

Goal

The goal is to perform the migrations needed in order to update from

- Magnolia 4.4.6 to 5.x
- Magnolia 4.5.x to 5.x

In case of a fresh installation of Magnolia 5.x, no migration tool is needed.

A bit of History

Let first introduce the migration modules used for 4.5.x.

Migration Module 1.1.x

Used to **migrate** from **Magnolia 4.4.6 to 4.5.0 until 4.5.6**

Key features

- Migration task configured and run by calling Groovy scripts
- Migration tasks written in Groovy
- Migration run manually once the Magnolia was updated from 4.4.6 to 4.5.x.
  This migration had to
    - Migrate Magnolia Modules (SKT, FORM, ...)
    - Custom modules

Steps to run

1. Install the new version of Magnolia (4.5.x)
2. From the Admin Central Tools menu, configure the Groovy Script.
    a. Set modules to migrate (Installed Magnolia Modules, Customs modules, Template migration,...)
    b. Set references properties (Path to templates, path to custom/magnolia site definition, ...)
3. From the same place, run the Migration by calling a Groovy Script
4. Restart Magnolia in order to complete the migration.

Issues faced

Hard to configured
Hard to isolate issues (All migration was performed at once)
Hard to view the result of the migration.

Migration Module 1.2.x

Used to **migrate** from **Magnolia 4.4.6 to 4.5.7 or higher**

Improvements

Migrate to Java (All groovy tasks are now translated to Java tasks)
Add a new Reporting mechanism
Isolate tasks in order to be able to run a migration for single module

Each Magnolia module is now responsible for his own migration. The end user has only to take car for his migration.

Key Features

- Creation of 4 main migration tasks and 22 sub migration tasks
    - Main **configuration migration tasks**
        - Migrate Content : Site migration
        - Migrate simple template : Migrate configuration for modules using only the Magnolia template module
        - Migrate STK base template: Migrate configuration for modules using/extending STK for templating.
    - Main **template migration task**
        - Migrate Templates: Perform FTL's/JSP's migration.
    - Sub tasks are used by the main tasks and perform logical steps like changing node type of content, review/introduce extends mechanism...
- Creation of a reporting tool allowing to generate better report in an HTML / Text form.
- Enforce the Exception/Session handling.

Step to run

1. Create a new version handler for custom module (using one of the 3 main **configuration migration task**)

2. Install the new version of Magnolia (4.5.x)
   At this point all magnolia modules are migrated without a specific user action.
3. Configure the Groovy script responsible to run the Java Template migration task
4. Run the **Template migration task**
5. Create the report.

Important improvements

- Creation of a reporting service
- Creation of an ID map service (Storing key/pair values into JCR)

M5 migration

Used to **migrate** from

- **Magnolia 4.4.6 to 5.x or higher**
- **Magnolia 4.5.x to 5.x or higher**

Like for the migration module 1.2, **modules will be responsible to perform the appropriate migrations** (DAM module will provide migration task in order to move content from DMS to DAM workspace, Handle content,...).

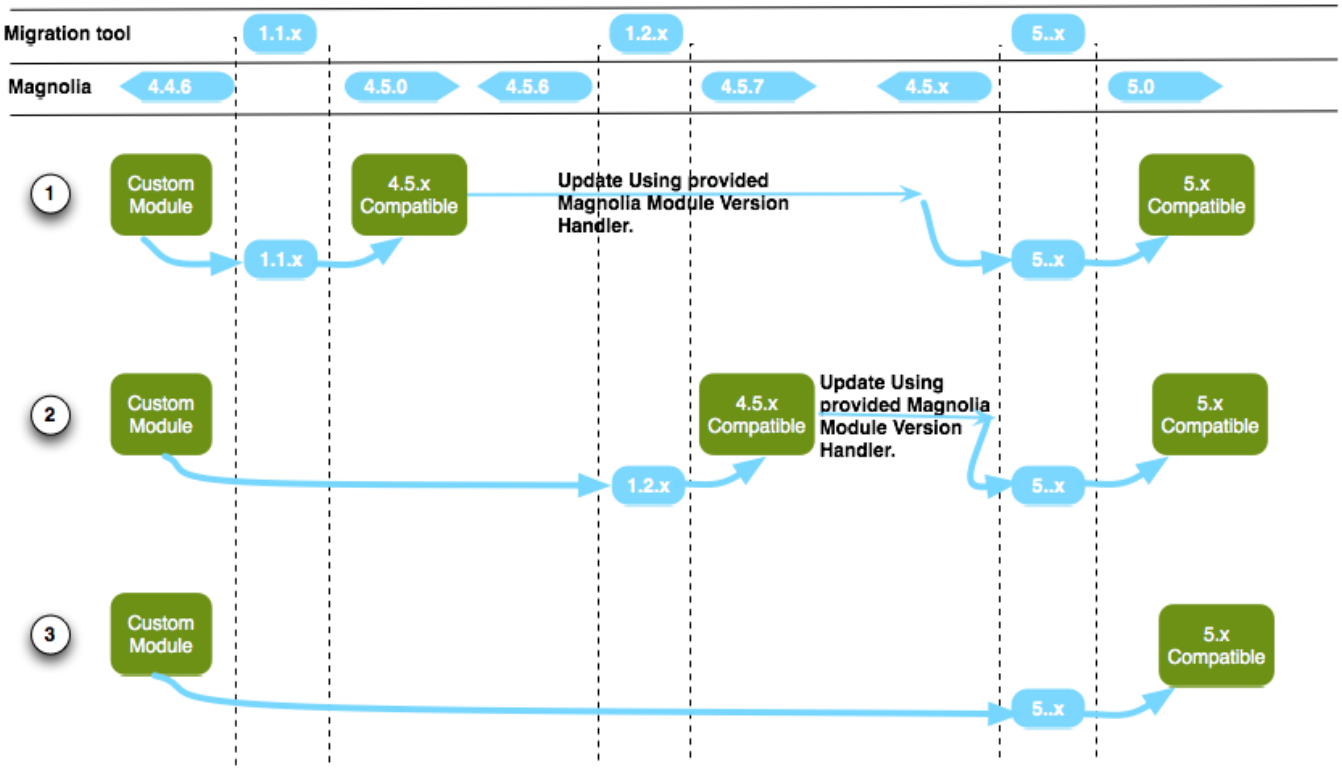
The big change is that we will no more use a Migration module, but App's

- Report App
- Template/Configuration migration App

Scenarios

M5 migration will have to support the following scenarios:

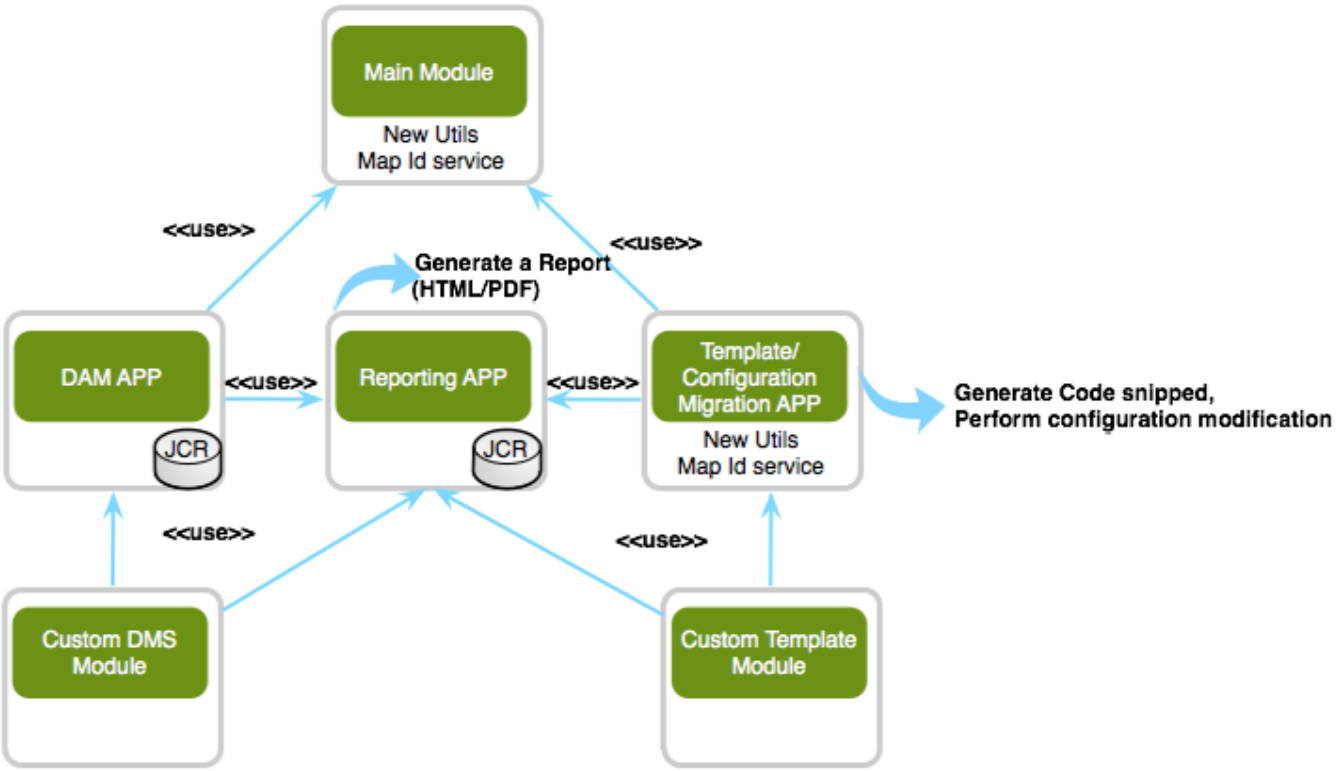
1. Previous Magnolia Instance is 4.4.6 and Magnolia (and custom modules) was already migrated to 4.5.1 .
   a. The update from 4.4.6 to 4.5.1 was performed using the 1.1.x migration tool.
   b. Update from 4.5.1 to 4.5.x is normally done with the module's version handler (Normal magnolia life cycle).
   c. Update from 4.5.x to 5.x will be performed using **M5 migration modules**
2. A Magnolia instance (with custom module) is updated from 4.4.6 to 4.5.7 .
   a. The Update from 4.4.6 to 4.5.7 is performed using the 1.2.x migration tool.
   b. Update from 4.5.x to 5.x is done by using **M5 migration modules**
3. Magnolia update from 4.4.6 to 5.x with custom module.
   a. Update from 4.4.6 to 5.x is done by using **M5 migration modules**
4. A fresh installation of Magnolia 4.5.x was performed. (Equivalent to 1.c.).



So we clearly see that the new M5 migration tool has to be able to handle:

- Migration form 4.5.x to 5.0
- Migration from 4.4.6 to 5.0.

Big pictures



The current migration module will become a M5 App (Template/Configuration Migration App).
This App will have:

- User interface allowing to configure and run the Template migration.
- All main and sub configuration migration task in order to handle the migration from 4.4.6 to 4.5.x.

The Reporting mechanism currently defined into the migration tool, will become an own M5 App (Reporting App)
This App will have:

- User interface allowing to configure and run the Report generation.

Current utility defined in the migration module will be migrated to the main module:

- Service map ID
- Some utilities

Applications

Report App

Requirements:

The generated report should look as designed (SCRUM Ticket)

User interface should have the following availabilities:

- "Generate report" button + selects to choose the report format (single HTML / ZIP) and desired level of messages included to the report (error /warning/info/system/debug);
- "Delete button" to allow delete old report records;
- Show the date and time of the last added record and the last report generation -> i.e. whether there are new messages since the last time report has been generated;
- The Single-HTML report should allow "folding" of tasks/modules/scripts, i.e. when task/script is "folded", then only its main data (name, result, timestamp, and general info) are displayed, not the messages; similarly for the module;
- The Single-HTML report should allow to show only errors, user-action requests and/or code snippets, i.e. to hide general "info" messages; (I am not sure we can make it reasonably using the CSS+Javascript, perhaps this should be an option for the report generation);

M5 Migration App

Requirements

User interface should have the following availabilities:

- The configuration performed via Groovy script should be moved into a configuration screen/dialog.
- The Template migration should be rerun as often as needed
- Implement/migrate these requirements (SCRUM Ticket)

Tasks

Report App

- Create a new Report App
  - Adapt the code from the current Migration Report page
  - Add the "timestamp" information (last added message, last report generated)
  - Add the Delete button
- Add styles and javascript to the generated report to allow "folding"
- Explore the possibility to hide/show different kind of messages using CSS/Javascript

M5 Migration app:

- Create a new M5 App
  - Create a sub app for Template migration configuration and execution
- Extract current Code from the Migration project to the new App
  - For Template migration
  - For Configuration migration
  - Review the Migration class hierarchy and implementation (SCRUM Ticket)

- Extract all relevant code to the main module
    - Id service
    - Generic JCR utility methods
- Use the new Report app to log user information
- Adapt code of migrated modules to point now to the new Configuration migration tasks (STK/Form/PUR/....)
- Review the implemented M5 migration tasks (DAM/Node)

Already implemented migration task (M5)

DAM migration

DAM migration tasks

TODO ehe: Review Reference migration for Bookmark & Images links.

Node

MetaData as mixin

Dialog

TODO ehe: Add Doc

Data Module

TODO ehe: Add Doc

MultiSite

MultiSite has a migration task which moves old ETK site configs

`/modules/extended-templating-kit/config/sites`

to the new config node of MultiSite

`/modules/multisite/config/sites.`

`See: info.magnolia.multisite.setup.MigrateETKSitesTask`

TODO: Whenever Migration from 4.5 to 5.x is available this has to be tested.

Modules not planned to be migrated to M5

Workflow

Workflow will not be migrated. As OpenWFE is not supported anymore, we encourage costumers to migrate their workflows to our new JBPM based workflow.

There have been ideas how to make this step easier for customers, but so far no definite decision has been made:

- Support them actively by actually doing it for them.
- Write a legacy OpenWFE based workflow module.
- ... ?