Data migration to Plone 5.2 and Volto

Rodrigo Ferreira de Souza

October, 2019

Where we are

Options to Migrate

- Plone 4.3 → Plone 5+
  - Collective Transmogrifier

- From Plone 4.3 we just have the option to use transmogrifier
- For Plone 5.1 you might consider use database upgrade
  - what means update pickle structure of ZODB to Python3 data type structure
  - to do it you need to: run the script in Python 2; don't start the instance (important); update buildout; run tests; start instance
  - during the process you might have problems with dependencies to fix
- The term was coined by Bill Waterson of Calvin and Hobbes fame

Options to Migrate

- Plone 4.3 → Plone 5+
  - Collective Transmogrifier
- Plone 5.1 → Plone 5.2+
  - Migrate a ZODB from Python 2.7 to Python 3

- From Plone 4.3 we just have the option to use transmogrifier
- For Plone 5.1 you might consider use database upgrade
  - what means update pickle structure of ZODB to Python3 data type structure
  - to do it you need to: run the script in Python 2; don't start the instance (important); update buildout; run tests; start instance
  - during the process you might have problems with dependencies to fix
- The term was coined by Bill Waterson of Calvin and Hobbes fame

Options to Migrate

- Plone 4.3 → Plone 5+
  - Collective Transmogrifier
- Plone 5.1 → Plone 5.2+
  - Migrate a ZODB from Python 2.7 to Python 3
  - Collective Transmogrifier

- From Plone 4.3 we just have the option to use transmogrifier
- For Plone 5.1 you might consider use database upgrade
  - what means update pickle structure of ZODB to Python3 data type structure
  - to do it you need to: run the script in Python 2; don't start the instance (important); update buildout; run tests; start instance
  - during the process you might have problems with dependencies to fix
- The term was coined by Bill Waterson of Calvin and Hobbes fame

Options to Migrate

- Plone 4.3 → Plone 5+
  - Collective Transmogrifier
- Plone 5.1 → Plone 5.2+
  - Migrate a ZODB from Python 2.7 to Python 3
  - Collective Transmogrifier

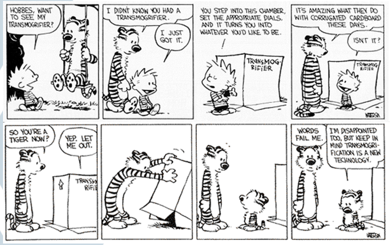

HOBBES, WANT TO SEE MY TRANSMOGRIFIER?
I DIDN'T KNOW YOU HAD A TRANSMOGRIFIER.
I JUST GOT IT.
YOU STEP INTO THIS CHAMBER, SET THE APPROPRIATE DIALS, AND IT TURNS YOU INTO WHATEVER YOU'D LIKE TO BE.
TRANSMOG-RIFIER
IT'S AMAZING WHAT THEY DO WITH CORRUGATED CARDBOARD THESE DAYS.
ISN'T IT?
SO YOU'RE A TIGER NOW?
YEP. LET ME OUT.
WORDS FAIL ME.
I'M DISAPPOINTED TOO, BUT KEEP IN MIND TRANSMOGRI-FICATION IS A NEW TECHNOLOGY.

Figure: A transmogrifier is fictional device used for transforming one object into another object.

- From Plone 4.3 we just have the option to use transmogrifier
- For Plone 5.1 you might consider use database upgrade
  - what means update pickle structure of ZODB to Python3 data type structure
  - to do it you need to: run the script in Python 2; don't start the instance (important); update buildout; run tests; start instance
  - during the process you might have problems with dependencies to fix
- The term was coined by Bill Waterson of Calvin and Hobbes fame

Why we use Transmogrifier?

- Have many generic Pipelines available for common cases

How transmogrifier works:

- each pipeline takes the preview data items
- modify;
- and yield data to next pipeline.

Why we use Transmogrifier?

- Have many generic Pipelines available for common cases
- Flexibility to deal with different use cases

How transmogrifier works:

- each pipeline takes the preview data items
- modify;
- and yield data to next pipeline.

Why we use Transmogrifier?

- Have many generic Pipelines available for common cases
- Flexibility to deal with different use cases
- Briliant way to use Iterator Design Pattern!

How transmogrifier works:

- each pipeline takes the preview data items
- modify;
- and yield data to next pipeline.

Why we use Transmogrifier?

- Have many generic Pipelines available for common cases
- Flexibility to deal with different use cases
- Briliant way to use Iterator Design Pattern!

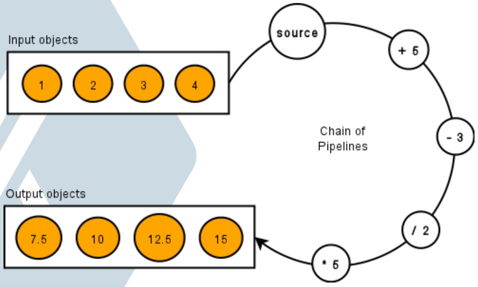


Figure: Transmogrify Diagram

How transmogrifier works:

- each pipeline takes the preview data items
- modify;
- and yield data to next pipeline.

Why we use Transmogrifier?

- Have many generic Pipelines available for common cases
- Flexibility to deal with different use cases
- Briliant way to use Iterator Design Pattern!

Figure: Modern Times – Production line

- In other words, Transmogrifier allow us to create something like a production line
- we have an object that is modified in each pipeline
- until it get ready in the end.

Where we are

Large University

Figure: Large University client website

- Not so big database;
- many custom packages;
- a frankenstein buildout.

High-profile government client

Figure: High-profile government client website

- Many data;
- Migration takes around 4 hours to run;
- some addons;
- custom report object type;
- was 1 archetypes, becomes 10 dexterity.

One of the largest research institutions in Germany

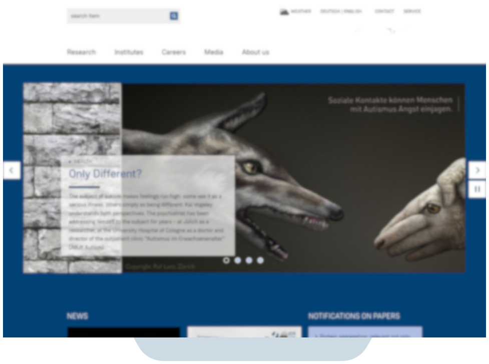


Figure: Large research institution client website

- Intranet;
- Had a third part system integration that needs to be imported in new website.

Where we are

The challendge

- From:
  - Python 2.x

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x
  - Archetypes or Dexterity

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x
  - Archetypes or Dexterity
  - Old Products

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x
  - Archetypes or Dexterity
  - Old Products
  - Sometimes other systems

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x
  - Archetypes or Dexterity
  - Old Products
  - Sometimes other systems
- To:
  - Python 3

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x
  - Archetypes or Dexterity
  - Old Products
  - Sometimes other systems
- To:
  - Python 3
  - Plone 5.2

So.. to sum up, those are the things that we need to do

The challendge

- From:
  - Python 2.x
  - Plone 4.3.x or 5.0.x or 5.1.x
  - Archetypes or Dexterity
  - Old Products
  - Sometimes other systems
- To:
  - Python 3
  - Plone 5.2
  - Volto

So.. to sum up, those are the things that we need to do

Advantages for the clients

- They spare a migration from Plone 5 to Plone 6

For the clients they can spare part of the Plone 5 to Plone 6 migration

Advantages for the clients

- They spare a migration from Plone 5 to Plone 6
- At least part of it

For the clients they can spare part of the Plone 5 to Plone 6 migration

Advantages for Plone solutions providers

- A way to sell clients the Python 3 upgrade

For the soluctions providers we can sell together Python 3 with Volto

Advantages for Plone solutions providers

- A way to sell clients the Python 3 upgrade
- Which is costly but does not gain the client anything in terms of functionality

For the soluctions providers we can sell together Python 3 with Volto

Where we are

Packages

- kitconcept Content Creator

k.migrator lives inside the Migration Package

Packages

- kitconcept Content Creator
- kitconcept Migrator

k.migrator lives inside the Migration Package

Packages

- kitconcept Content Creator
- kitconcept Migrator
- Migration Plone 5

k.migrator lives inside the Migration Package

Commander Utility

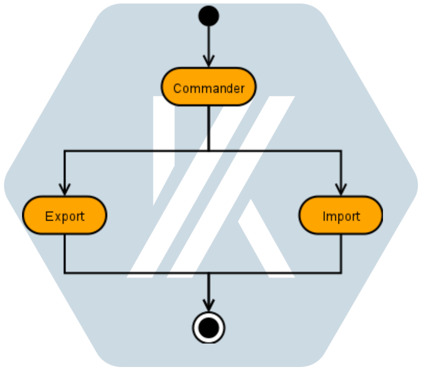


Figure: Commander Utility

We created an utility to help orchestrate the migration process.

Jenkins

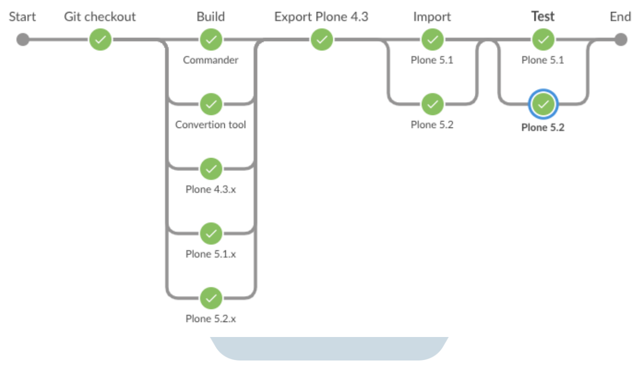
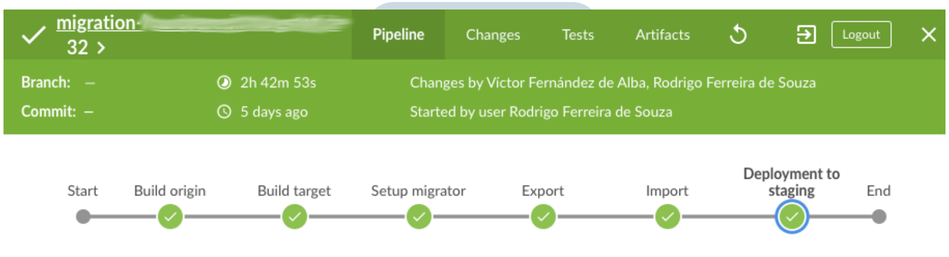


Figure: Jenkins

Each commit runs all those steps to make tests (smoke tests).

Migration Server



migration-[redacted]
32 >

Branch: –
Commit: –
2h 42m 53s
5 days ago
Changes by Víctor Fernández de Alba, Rodrigo Ferreira de Souza
Started by user Rodrigo Ferreira de Souza

Figure: Migration Server

Also we have migration Jenkins node, that we push the button to start the migration (what takes about 4 hours to finish)

Where we are

General Backend

- ATTopics → Collection

- For ATTopics our strategy was to fix during export phase
  - we basically run the default Plone upgrade step on the query,
  - and make the changes while still at Plone 4.3
  - and make the exported data ready for dexterity collections
- For tinymce html to volto,
  - we created a simple nodejs code that convert to DraftJS data,
  - so for each object we call this utility with our HTML,
  - and set the result in tile behaviour attribute.
- Portlets are migrated, but not showed in Volto;
  - we have one case with a special portlet that have a download URL that points to other part of the website;
  - so we need to keep track of this data.
- Postmigration are some "not migrated" data (c.cover for example)
  - that we should replace for something else after migration,
  - so we automated it creating some data manually
  - and saving the JSON extracted by restapi from this content.

General Backend

- ATTopics → Collection
- RichText → Volto Blocks

- For ATTopics our strategy was to fix during export phase
  - we basically run the default Plone upgrade step on the query,
  - and make the changes while still at Plone 4.3
  - and make the exported data ready for dexterity collections
- For tinymce html to volto,
  - we created a simple nodejs code that convert to DraftJS data,
  - so for each object we call this utility with our HTML,
  - and set the result in tile behaviour attribute.
- Portlets are migrated, but not showed in Volto;
  - we have one case with a special portlet that have a download URL that points to other part of the website;
  - so we need to keep track of this data.
- Postmigration are some "not migrated" data (c.cover for example)
  - that we should replace for something else after migration,
  - so we automated it creating some data manually
  - and saving the JSON extracted by restapi from this content.

General Backend

- ATTopics → Collection
- RichText → Volto Blocks
- Portlets

- For ATTopics our strategy was to fix during export phase
  - we basically run the default Plone upgrade step on the query,
  - and make the changes while still at Plone 4.3
  - and make the exported data ready for dexterity collections
- For tinymce html to volto,
  - we created a simple nodejs code that convert to DraftJS data,
  - so for each object we call this utility with our HTML,
  - and set the result in tile behaviour attribute.
- Portlets are migrated, but not showed in Volto;
  - we have one case with a special portlet that have a download URL that points to other part of the website;
  - so we need to keep track of this data.
- Postmigration are some "not migrated" data (c.cover for example)
  - that we should replace for something else after migration,
  - so we automated it creating some data manually
  - and saving the JSON extracted by restapi from this content.

General Backend

- ATTopics → Collection
- RichText → Volto Blocks
- Portlets
- Postmigration

- For ATTopics our strategy was to fix during export phase
  - we basically run the default Plone upgrade step on the query,
  - and make the changes while still at Plone 4.3
  - and make the exported data ready for dexterity collections
- For tinymce html to volto,
  - we created a simple nodejs code that convert to DraftJS data,
  - so for each object we call this utility with our HTML,
  - and set the result in tile behaviour attribute.
- Portlets are migrated, but not showed in Volto;
  - we have one case with a special portlet that have a download URL that points to other part of the website;
  - so we need to keep track of this data.
- Postmigration are some "not migrated" data (c.cover for example)
  - that we should replace for something else after migration,
  - so we automated it creating some data manually
  - and saving the JSON extracted by restapi from this content.

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types
  - Deal with default pages

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types
  - Deal with default pages
  - Convert RichText HTML to Volto DraftJS (node utility)

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types
  - Deal with default pages
  - Convert RichText HTML to Volto DraftJS (node utility)
  - Easily point to old website when content not imported

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types
  - Deal with default pages
  - Convert RichText HTML to Volto DraftJS (node utility)
  - Easily point to old website when content not imported
  - Fix URLs (planned resolveuid)

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types
  - Deal with default pages
  - Convert RichText HTML to Volto DraftJS (node utility)
  - Easily point to old website when content not imported
  - Fix URLs (planned resolveuid)
  - Simple Folders → Document with Collection Block (planned)

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

The Volto part

- What we Polish to enter in Volto land:
  - Use Collective Folderish Types
  - Deal with default pages
  - Convert RichText HTML to Volto DraftJS (node utility)
  - Easily point to old website when content not imported
  - Fix URLs (planned resolveuid)
  - Simple Folders → Document with Collection Block (planned)
  - Simple Collection → Document with Collection Block (planned)

- We use c.folderishtypes because in volto we don't have the concept of default pages, so for migration we need to make almost everything folderish to keep in sync the old and new website URLs
- Some content types of old website is NOT going to be imported, in this case we write some code to show a link to point to the old website
- Well we still don't have resolveuid concept in volto.. we plan to use something similar to what Plone does for tinymce, but with restapi and volto
- Folders will be as simple document with listing block showing the folder content
- Collections becomes simple cocument with listing block running the collection query

Questions?

THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
MERCI
BOLZÏN
BIYAN
DANKSCHEEN
GRAZIE
MEHRBANI
TASHAKKUR ATU
SUKSAMA
JUSPAXAR
GOZAIMASHITA
EFCHARISTO
KOMAPSUMNIDA