Von Typo3 zu Plone

Ein Migrationsbericht - Die Inhalte einer Webseite dürfen bei einem Wechsel auf einen anderes CMS nicht einfach verloren gehen. Da jedes CMS seine Daten etwas anders speichert ist Daten-Migration aber eine oft komplexe und aufwändige Aufgabe. Der Vortrag berichtet von einer konkreten Migration von Typo3 zu Plone, bei der Inhalte mittels Transmogrifier automatisch transferiert und angepasst wurden.

Andreas Schiweck, catWorkX GmbH

Ausgangssituation I

- **Kunde mit erhöhtem Sicherheitsbedürfnis...**
- **mit Typo3 Installation...**
- **und schlechten Erfahrungen!**

Ausgangssituation 2

- **Stabilitäts-/Performance-Probleme: Virtualisierung + hohe IO Last**
- **Ungewöhnlich große Datenbank**
- **Altlasten: keine sprechenden URLs, Index-Probleme**

Ausgangssituation 3

- Website ist Kommunikations-Platform: Abbildung einer bundesweiten Verbandstruktur mit Landes- und Bezirksverbänden
- Mehr als 2000 Website-Benutzer (Frontend) in Typo3, ca. 100 Redakteure (Typo3-Backend-Benutzer)
- Ausreichend Inhalte für eine automatisierte Migration: Seiten, tt_news, tt_cal,...

Der Wunsch: Ersatz des CMS durch Plone

- **Inkl. Migration aller Inhalte**
- **Konsolidierte Benutzerprofile**
- **Schulung der Redakteure über verbandseigene Multiplikatoren**

Anforderungen an Plone

- **Vollständige IE6 Kompatibilität**
- **Moderierbare Kommentare: plone.app.discussion**
- **Fotoalben, RSS-Feed-Integration, Videos, Veranstaltungskalender, Youtube-Integration,...**
- **Nachbildung des bisherigen Registrierungsworkflows**
- **Konsolidierte Benutzerquelle: LDAP (für OpenExchange)**

Keine Herausforderung! Aber wie migrieren?

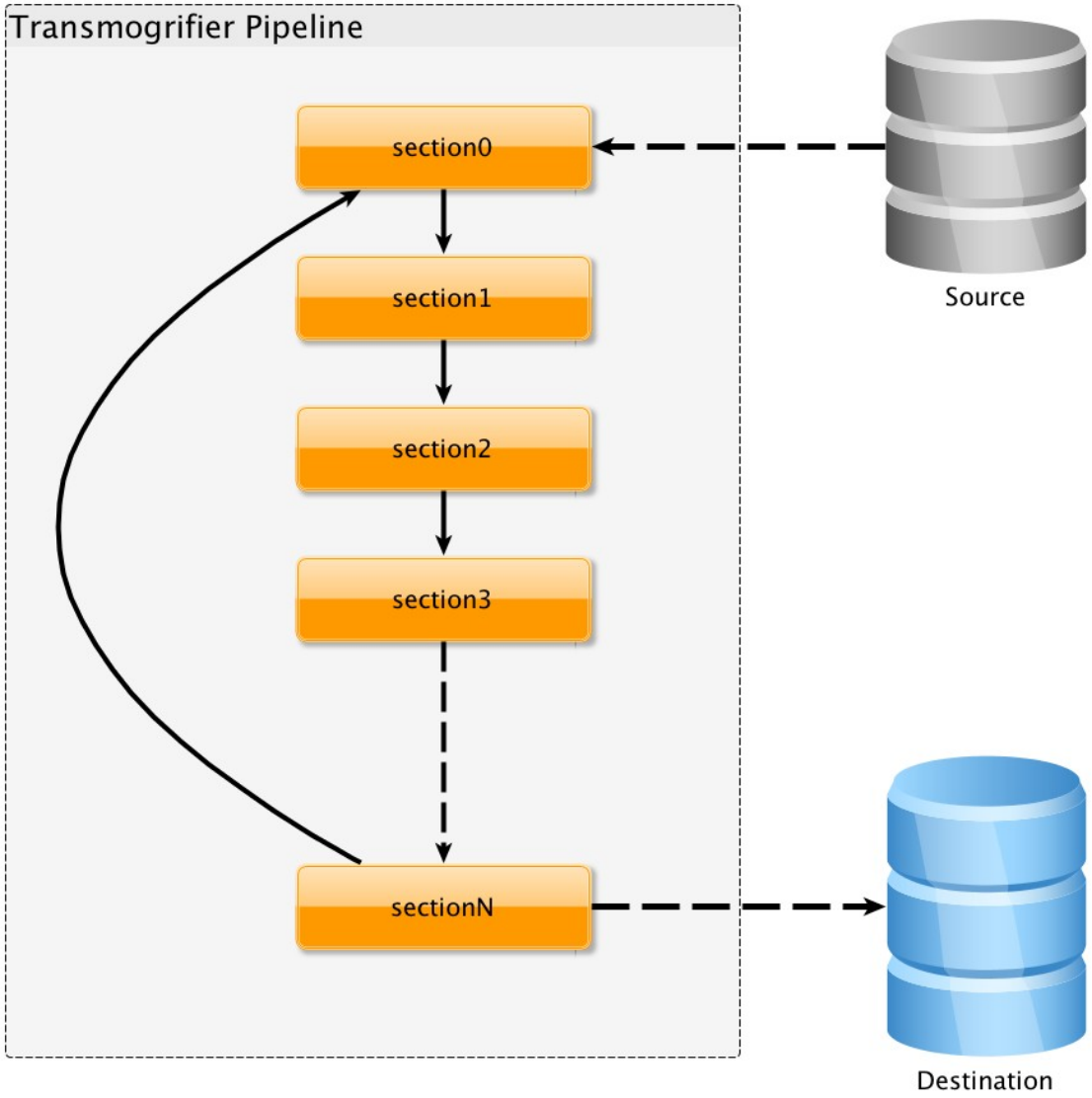
- **Das *n-te* Skript, Webcrawling, BeautifulSoup,... Nein!**
- **Lennart Regebro versprach kurz zuvor in Budapest: „Migrating to Plone with less Pain“: mit Transmogrifier.**
- **„Wer nicht wagt...“: das versuchen wir mal!**

Kurze Zeit später wissen wir:

- **Transmogrifier bietet eine konfigurierbare Pipeline für Transformationen an.**
- **Es gibt schon eine große Anzahl an Transmogrifiern, die viele nötige Aufgaben übernehmen können: urlnormalizer, pageupdater, constructor, uidupdater, atschemaupdater, workflowupdater, reindexobject, savepoint**

Energize Transmogrifier, Mr. Crusher!

Transmogrifier?



customer.import

customer.import/customer/import/pipeline0.cfg:

```
[transmogrifier]
pipeline =
        section0
        section1
        savepoint

[section0]
blueprint = some.blueprint

[section1]
blueprint = some.blueprint

[savepoint]
blueprint = collective.transmogrifier.sections.savepoint
every = 50
```

customer.import/customer/import/configure.zcml:

```
<configure
  [...]

      xmlns:transmogrifier="http://namespaces.plone.org/tr
      ansmogrifier"
  [...]

 <transmogrifier:registerConfig
   name="customer.import.pipeline0"
   title="customer.import.pipeline0"
   description="pipeline0"
   configuration="pipeline0.cfg"
   />

</configure>
```

customer.import/customer/import/setuphandler.py:

```
from collective.transmogrifier.transmogrifier import
      Transmogrifier

def setupVarious(context):
   """
   """
   transmogrifier = Transmogrifier(context.getSite())
   # run the following pipelines
   transmogrifier("customer.import.pipeline0")
```

Typo3: Was brauchen wir jetzt noch?

- **Sources: TTNewsSource, PageSource, CalSource, StructureSource**
- **Constructors: TTNews2NewsItem, Page2Folder, Page2Document, Cal2Event**
- **Fixes: FixEncodings, FixLinks**
- **Integrators: SetRelatedItems, AttachCategories, AttachDiscussions, DiscussionsUpdater, AttachImage**

Blueprint?

Generator + ISection + Dictionary

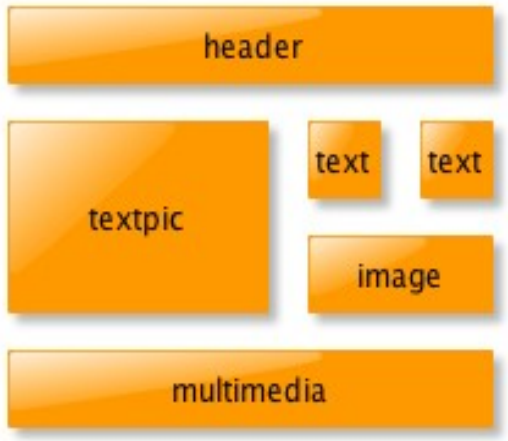
„item“!

```
>>> type(item)
<type 'dict'>
>>> item['title'] = "Lorem Ipsum"
>>> item['_path'] = "/foo/bar"
```

```
from collective.transmogrifier.interfaces import ISection
from collective.transmogrifier.interfaces import IsectionBlueprint

class Blueprint0(object):
    """
    """
    classProvides(ISectionBlueprint)
    implements(ISection)

    def __init__(self, transmogrifier, name, options, previous):
        """
        """
        self.previous = previous
        self.context = transmogrifier.context

    def __iter__(self):
        """
        """
        for item in self.previous:

            # Do whatever you like to item
            item['Creator'] = 'Bob'

            yield item
```

Use the TTNewsSource, Luke!

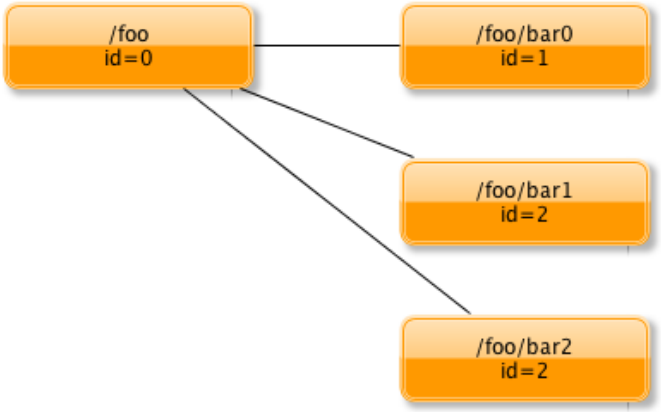
```
import MySQLdb

query = """select uid, pid, title, short,
      bodytext, author, links, image, endtime,
      type, page, ext_url,
      news_files as attachments, crdate as creation_date,
      datetime as modification_date
      from tt_news where deleted = 0 and archivedate = 0"""

ql = "select uid, pid, title, sorting from pages where uid
      = %s order by sorting"

self.query = """select * from pages where deleted =
      0""" + limit

query = """select header, image,
        imagewidth from tt_content
        where uid = %s"""

query = """select header, header_layout,
        media, imagecaption
        from tt_content where uid = %s"""
```



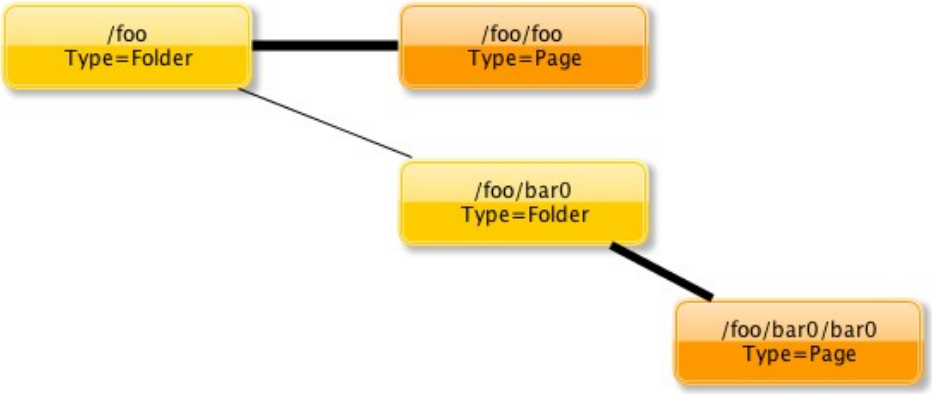
Reverse Engeneering, Versuch, Irtum, Änderung, Re-Import!

StructureSource?

Typo3



Plone



Flexibilität für den Kunden:

- Objekte verschieben (item['_path'])
- Modifikationen des Titels
- Kodierungsprobleme
- Doch keine Keyword-Migration
- ...

Die (Ab-)Rechnung bitte!

- 4 Pipelines: structure, pages, news, events
- Je Pipeline: 10-15 Sections
- 15 generische Typo3-Blueprints, 4 kundenspezifische Updates
- Gefühlte 15 Mio. unkritische Re-Imports nach Anpassungen

Unser Fazit

- **Transmogrifier can do the job!**
- **Migrationen mit Transmogrifiern sind flexibel, wiederverwendbar und zu warten!**
- **Aber: Aufwand beim ersten Einsatz nicht zu unterschätzen!**

Danke!

email: andreas.schiweck (at) catworkx.de

Twitter: @_rainrider

IRC: rainrider