

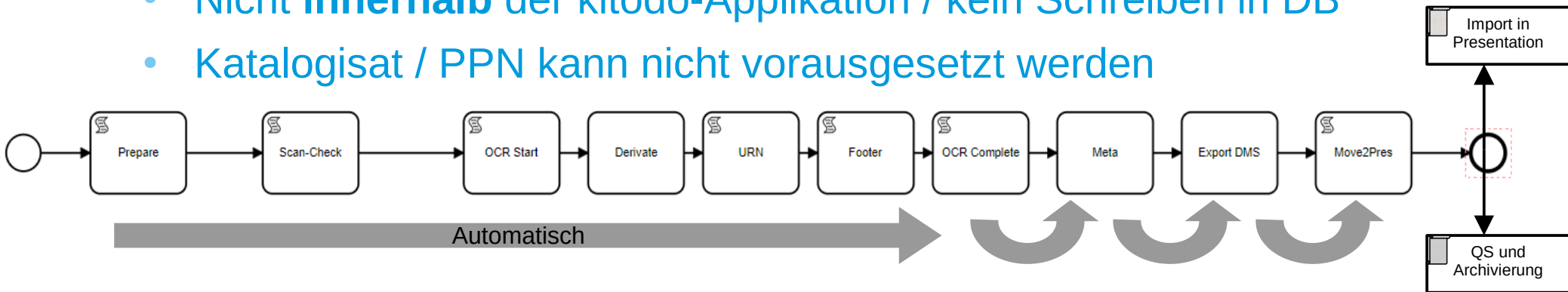
Realisierung eines kitodo-production-Workflows auf Basis einfacher Shellscripte

ULB Darmstadt / Voraussetzungen

- Seit 2008 Digitalisierung in der ULB Darmstadt
- Plattform: dwork (UB Heidelberg)
- Ca. 15.000 Objekte aller Art
- Strategische Entscheidung für kitodo
 - Gründe u.a.: Community, Lizenz, Support, Entwicklung – Features (?)
- Kaum valide Titelaufnahmen
- Enge personelle Kapazitäten
- Hessische Parlamentsprotokolle (MDL, Teil 2) als Pilot

Infrastruktur

- kitodo.production:
 - nur „Derivate“, „Meta“ und „Export DMS“
 - → andere „Funktionsgruppen“  müssen selbst realisiert werden
- Aufgrund unserer Voraussetzungen:
 - **Einfach**, auf Basis von ShellscripTEN
 - Nicht **innerhalb** der kitodo-Applikation / kein Schreiben in DB
 - Katalogisat / PPN kann nicht vorausgesetzt werden



Funktionsgruppen

- Prepare
- Scan-Check (Audit)
- OCR (Start und Complete)
- URN (und Meta)
- Footer
- Move to Presentation
- Import in Presentation
- QS und Archivierung

Realisierung 1

- Anbindung HEBIS-Verbund
- Kleine Funktionsbibliothek, einfache Konfigurationsdatei(en)
- Immer identischer Aufbau der bash-Scripts
- Anpassung an Struktur von kitodo.production
 - Projekte (1 Projekt → 1 Config-Datei)
 - Shellscripate als TaskScripts im Workflow von kitodo
 - Scripte schreiben in / lesen aus meta.xml
- Tickets zur Zustandskontrolle der Tasks / Workflow-Phasen
- Funktionen, die nicht in kitodo-GUI möglich sind → **kitotick**

Realisierung 2

- Start 2021
- 3 github-Repos (Taskscripts, Config-Dateien, kitotick)
- Ca. 50 Scripts mit mehr als 6000 Zeilen Code
- Bisher ca. 120 Bände Hessische Parlamentsprotokolle verarbeitet
- Mittlerweile v 3.4 von kitodo.production
- kitodo.presentation mit Support durch Dienstleister
- ...

```
1 # Add footer with logo and text to XSL for a given ProcessID
2
3 # NAME: ProcessID (StepID)
4 # PARAMS: ProcessID (StepID)
5 # RETURN: 0 or 1
6
7 #!_ 2021-04-20
8 # 2021-04-20 erweitert um die Option, XSLTFOOD-Footer zu breiten oder zentrieren; kein Footer aus meta.xml auslesen
9 # 2021-04-20 um XSLTFOOD-Footer hinzugefügt; kein ProcessID und ein StepID zum XSLTFOOD-Footer
10 # 2021-04-20 erweitert um Schrittbeschreibung, z.B. um die ID des SchrittID, zum Typen im XSLTFOOD-Footer nach dem Schema der XSLTFOOD mit ein
11 # Content: https://github.com/kitodo/kitodo-production/blob/master/src/main/resources/xsl/fo/kitodo-fo.xsl#L100
12 # Content: https://github.com/kitodo/kitodo-production/blob/master/src/main/resources/xsl/fo/kitodo-fo.xsl#L100
13 # @author: dlabach@kitodo.org, SC2009, SC2009 (source: Publiclib, path, name to print: normal string, backslash)
14
15 #!_
16
17 #!_
18
19 #!_
20
21 #!_
22
23 #!_
24
25 #!_
26
27 #!_
28
29 #!_
30
31 #!_
32
33 #!_
34
35 #!_
36
37 #!_
38
39 #!_
40
41 #!_
42
43 #!_
44
45 #!_
46
47 #!_
48
49 #!_
50
51 #!_
52
53 #!_
54
55 #!_
56
57 #!_
58
59 #!_
60
61 #!_
62
63 #!_
64
65 #!_
66
67 #!_
68
69 #!_
70
71 #!_
72
73 #!_
74
75 #!_
76
77 #!_
78
79 #!_
80
81 #!_
82
83 #!_
84
85 #!_
86
87 #!_
88
89 #!_
90
91 #!_
92
93 #!_
94
95 #!_
96
97 #!_
98
99 #!_
100
```

Todos / Lessons learned / Caveats

- Performanz und Stabilität bisher positiv zu beurteilen – und wird immer besser :-)
- Support von Community ist gut, aber nicht so belastbar, dass man als Neuling ohne "bezahlte" Hilfestellung zum Erfolg kommt; daher Support von Dienstleistern
- es wird erhebliches Knowhow benötigt, um produktiv zu werden:
 - Systemadministration
 - METS/MODS
 - XML/XSLT/XPATH etc.
 - Scripting / Coding
 - Bibliothekarische Formate (Mapping Pica -> kitodo-METS/MODS)
 - Workflow-Modellierung
 - github / agiles Entwickeln
- Den Einstieg erleichtern würde eine schnell und stabil einsetzbare default-Konfiguration der rulesets mit den wichtigsten Publikationstypen
- Desiderate
 - IIF-Unterstützung, PDF-Export (on-the-fly), parallele Tasks, u.a.

Links und Quellen

Kitodo-Installationen in Darmstadt

- <https://kitoprod.ulb.tu-darmstadt.de/>
- <https://kitoshow.ulb.tu-darmstadt.de/>

3 Repositories mit

- <https://gitlab.ulb.tu-darmstadt.de/rothst/kitoprod-taskscripts>
- <https://gitlab.ulb.tu-darmstadt.de/rothst/kitoprod-taskscriptsconfig>
- <https://gitlab.ulb.tu-darmstadt.de/rothst/kitotick>

Die Scripts benötigen in der bash u.a.:

- Imagemagick, hashdeep, multital, jhove, saxon oder xmlstarlet, rsync, grep, sed, awk...

Realisierung eines kitodo-production-Workflows auf Basis einfacher Shellscripte

17.10.2022

Roland Roth-Steiner, ULB Darmstadt

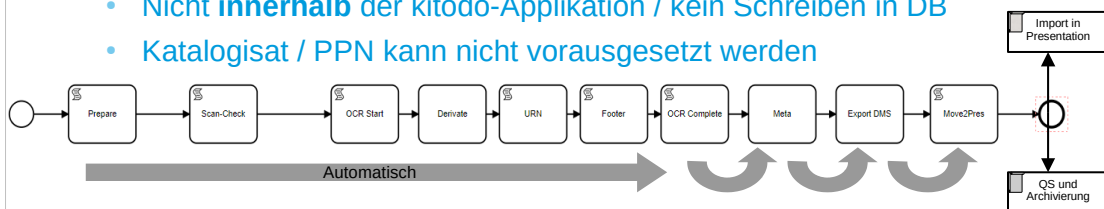
1

ULB Darmstadt / Voraussetzungen

- Seit 2008 Digitalisierung in der ULB Darmstadt
- Plattform: dwork (UB Heidelberg)
- Ca. 15.000 Objekte aller Art
- Strategische Entscheidung für kitodo
 - Gründe u.a.: Community, Lizenz, Support, Entwicklung – Features (?)
- Kaum valide Titelaufnahmen
- Enge personelle Kapazitäten
- Hessische Parlamentsprotokolle (MDL, Teil 2) als Pilot

Infrastruktur

- kitodo.production:
 - nur „Derivate“, „Meta“ und „Export DMS“
 - → andere „Funktionsgruppen“ müssen selbst realisiert werden
- Aufgrund unserer Voraussetzungen:
 - **Einfach**, auf Basis von Shellscripten
 - Nicht **innerhalb** der kitodo-Applikation / kein Schreiben in DB
 - Katalogisat / PPN kann nicht vorausgesetzt werden



Funktionsgruppen

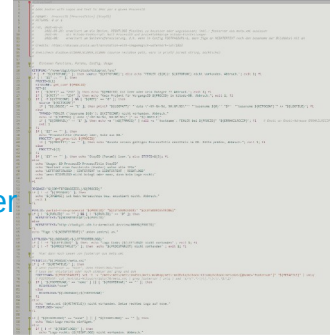
- Prepare
- Scan-Check (Audit)
- OCR (Start und Complete)
- URN (und Meta)
- Footer
- Move to Presentation
- Import in Presentation
- QS und Archivierung

Realisierung 1

- Anbindung HEBIS-Verbund
- Kleine Funktionsbibliothek, einfache Konfigurationsdatei(en)
- Immer identischer Aufbau der bash-Scripts
- Anpassung an Struktur von kitodo.production
 - Projekte (1 Projekt → 1 Config-Datei)
 - Shellscripte als TaskScripts im Workflow von kitodo
 - Scripte schreiben in / lesen aus meta.xml
- Tickets zur Zustandskontrolle der Tasks / Workflow-Phasen
- Funktionen, die nicht in kitodo-GUI möglich sind → **kitotick**

Realisierung 2

- Start 2021
- 3 github-Repos (Taskscripts, Config-Dateien, kitotick)
- Ca. 50 Scripts mit mehr als 6000 Zeilen Code
- Bisher ca. 120 Bände Hessische Parlamentsprotokolle verarbeitet
- Mittlerweile v 3.4 von kitodo.production
- kitodo.presentation mit Support durch Dienstleister
- ...

A screenshot of a code editor window displaying a large amount of code. The code is organized into several sections with headings, and it appears to be a complex script or configuration file. The text is small and dense, typical of a code editor view.

Todos / Lessons learned / Caveats

- Performanz und Stabilität bisher positiv zu beurteilen – und wird immer besser :-)
- Support von Community ist gut, aber nicht so belastbar, dass man als Neuling ohne "bezahlte" Hilfestellung zum Erfolg kommt; daher Support von Dienstleistern
- es wird erhebliches Knowhow benötigt, um produktiv zu werden:
 - Systemadministration
 - METS/MODS
 - XML/XSLT/XPATH etc.
 - Scripting / Coding
 - Bibliothekarische Formate (Mapping Pica -> kitodo-METS/MODS)
 - Workflow-Modellierung
 - github / agiles Entwickeln
- Den Einstieg erleichtern würde eine schnell und stabil einsetzbare default-Konfiguration der rulesets mit den wichtigsten Publikationstypen
- Desiderate
 - IIIF-Unterstützung, PDF-Export (on-the-fly), parallele Tasks, u.a.

17.10.2022

Roland Roth-Steiner, ULB Darmstadt

7

Links und Quellen

Kitodo-Installationen in Darmstadt

- <https://kitoprod.ulb.tu-darmstadt.de/>
- <https://kitoshow.ulb.tu-darmstadt.de/>

3 Repositories mit

- <https://gitlab.ulb.tu-darmstadt.de/rothst/kitoprod-taskscripsts>
- <https://gitlab.ulb.tu-darmstadt.de/rothst/kitoprod-taskscripstsconfig>
- <https://gitlab.ulb.tu-darmstadt.de/rothst/kitotick>

Die Scripts benötigen in der bash u.a.:

- Imagemagick, hashdeep, multital, jhove, saxon oder xmlstarlet, rsync, grep, sed, awk...