

Metadatenworkflows mit Metafacture erstellen und verwalten

Pascal Christoph & Tobias Bülte
Offene Infrastruktur, Hochschulbibliothekszentrum NRW (hbz)

Hands-on Lab

111. BiblioCon
Hannover, 25. Mai 2023

<https://slides.lobid.org/2023-05-metafacture-workshop/> : (PDF)



Kurze Vorstellungsrunde

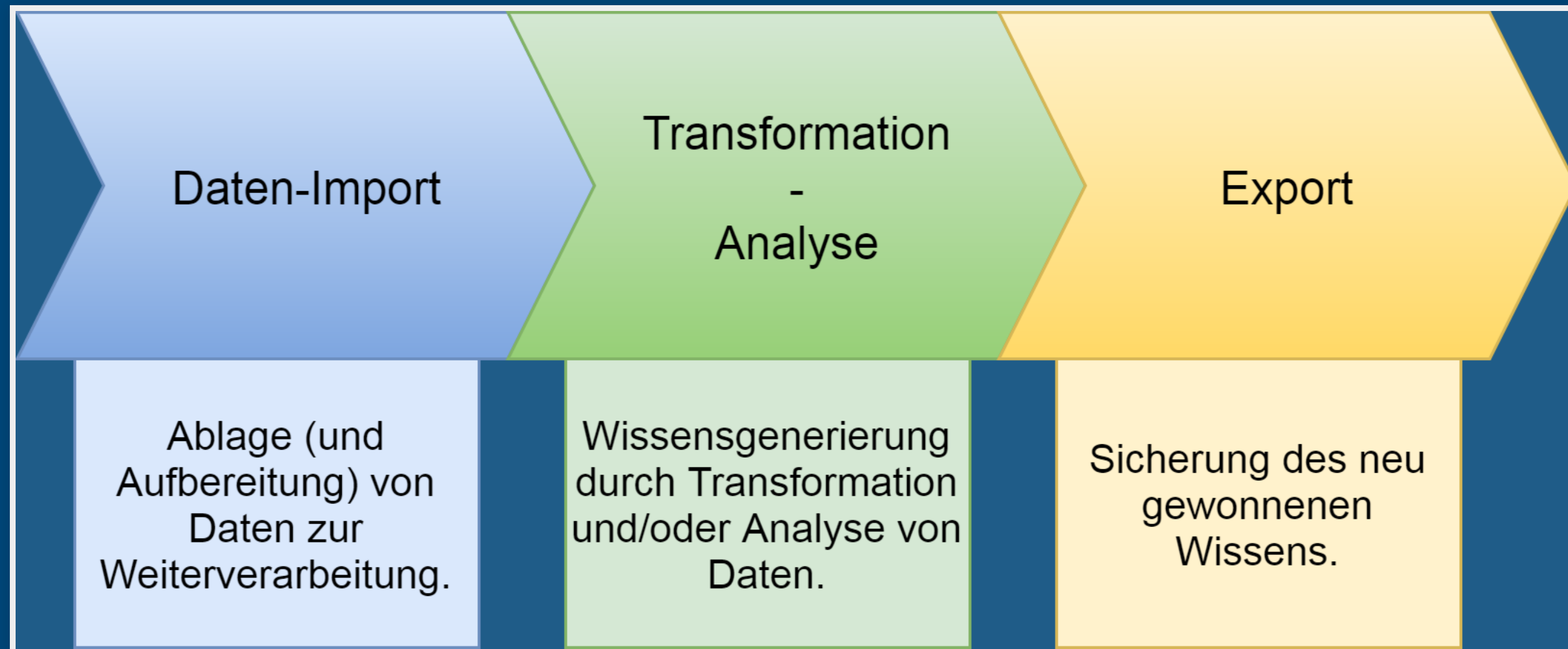
1. Wer (Name)
2. Wo (an welcher Institution beschäftigt)
3. Was (Hintergrund: Technik oder Bibliothekar:in oder ...)

Agenda

1. Einführung & Kontext
2. Workflows
3. Transformieren
4. Analysieren
5. Komplexere Beispiele
6. Ausblick, Fragen & Diskussion

1. Kontext: Datentransformation

ETL: Extract, Transform, Load



Aus: culturegraph, <https://github.com/culturegraph/culturegraph-workflow/blob/master/docs/src/docs/asciidoc/manual.adoc>

Anwendungsszenarien

Anwendungsszenarien

Datenanalyse, z.B. Feldabdeckung im Katalog

Anwendungsszenarien

Datenanalyse, z.B. Feldabdeckung im Katalog

Datenaufbereitung, z.B. zur Visualisierung mit Kibana

Anwendungsszenarien

Datenanalyse, z.B. Feldabdeckung im Katalog

Datenaufbereitung, z.B. zur Visualisierung mit Kibana

Datenanreicherung, z.B. Ergänzung von Daten aus Wikidata

Anwendungsszenarien

Datenanalyse, z.B. Feldabdeckung im Katalog

Datenaufbereitung, z.B. zur Visualisierung mit Kibana

Datenanreicherung, z.B. Ergänzung von Daten aus Wikidata

Datenaggregation aus unterschiedlichen Quellen, z.B. **OERSI**

Anwendungsszenarien

Datenanalyse, z.B. Feldabdeckung im Katalog

Datenaufbereitung, z.B. zur Visualisierung mit Kibana

Datenanreicherung, z.B. Ergänzung von Daten aus Wikidata

Datenaggregation aus unterschiedlichen Quellen, z.B. **OERSI**

Systemmigration, z.B. nach Alma oder Folio

Wo wir Metafacture nutzen

Wo wir Metafastructure nutzen

Transformation der Daten des Verbundkatalogs für die
Indexierung (lobid.org)

Wo wir Metafacture nutzen

Transformation der Daten des Verbundkatalogs für die
Indexierung (lobid.org)

Metadaten aus verschiedenen Quellen im OER Suchindex
aggregieren ([OERSI](https://oersi.org))

Wo wir Metafacture nutzen

Transformation der Daten des Verbundkatalogs für die
Indexierung (lobid.org)

Metadaten aus verschiedenen Quellen im OER Suchindex
aggregieren ([OERSI](https://oersi.org))

Transformation der Daten der Rheinland-Pfälzischen
Bibliographie

Nur ein kleiner Teil

Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,
Formaten, Modulen

Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,
Formaten, Modulen

Wir nutzen davon nur einen kleinen Teil, eben für unsere
Formate und Anwendungsfälle

Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,
Formaten, Modulen

Wir nutzen davon nur einen kleinen Teil, eben für unsere
Formate und Anwendungsfälle

Wir versuchen euch hier einen möglichst breiten Überblick
zu geben über Konzepte und Möglichkeiten

Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,
Formaten, Modulen

Wir nutzen davon nur einen kleinen Teil, eben für unsere
Formate und Anwendungsfälle

Wir versuchen euch hier einen möglichst breiten Überblick
zu geben über Konzepte und Möglichkeiten

Quasi ein Teaser was grundsätzlich geht, das würde man je
nach Anwendungsfall ganz unterschiedlich vertiefen

Annahmen

Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Es gibt viele unterschiedliche Methoden, die meist Programmierkenntnisse voraussetzen.

Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Es gibt viele unterschiedliche Methoden, die meist Programmierkenntnisse voraussetzen.

Datentransformationen werden meist im Zusammenspiel von Fachabteilungen & IT umgesetzt, verbunden mit größerem Kommunikationsaufwand.

Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Es gibt viele unterschiedliche Methoden, die meist Programmierkenntnisse voraussetzen.

Datentransformationen werden meist im Zusammenspiel von Fachabteilungen & IT umgesetzt, verbunden mit größerem Kommunikationsaufwand.

Bereits existierende, von anderen entwickelte Transformationsprozesse können nur bedingt entdeckt und nachgenutzt werden.

Das heißt:

Das heißt:

Es gibt großes Potential, eine immer wiederkehrende Arbeit zugänglicher, kollaborativer und effizienter zu gestalten.

Übergeordnete Ziele

Übergeordnete Ziele

Ermächtigung der Fachebene zur Konfiguration von
Datentransformationen

Übergeordnete Ziele

Ermächtigung der Fachebene zur Konfiguration von
Datentransformationen

Förderung von Praktiken zum Teilen und Auffinden von
Transformationsprozessen

Was ist Metafacture?

Was ist Metafactory?

Ein vielseitiges ETL-Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten

Was ist Metafactory?

Ein vielseitiges ETL-Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten
nutzbar als Kommandozeilentool, als Java/JVM library,

Was ist Metafactory?

Ein vielseitiges ETL-Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten
nutzbar als Kommandozeilentool, als Java/JVM library,
für Batch-Verarbeitung oder on-the-fly

Was ist Metafactory?

Ein vielseitiges ETL-Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten

nutzbar als Kommandozeilentool, als Java/JVM library,

für Batch-Verarbeitung oder on-the-fly

offenes Framework: Weiterentwicklung, Wiederverwendung und Austausch (von einzelnen Modulen und ganzen Workflows)

Metafactory-Historie

Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph auf sourceforge; damals schon Austausch mit dem hbz

Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph auf sourceforge; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub

Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph auf sourceforge; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub

2019: Mit der Zeit immer weniger DNB-Ressourcen für Metafactory. hbz wird Maintainer

Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph auf sourceforge; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub

2019: Mit der Zeit immer weniger DNB-Ressourcen für Metafactory. hbz wird Maintainer

2019: Start von Metafactory Fix

Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph auf sourceforge; damals schon Austausch mit dem hbz

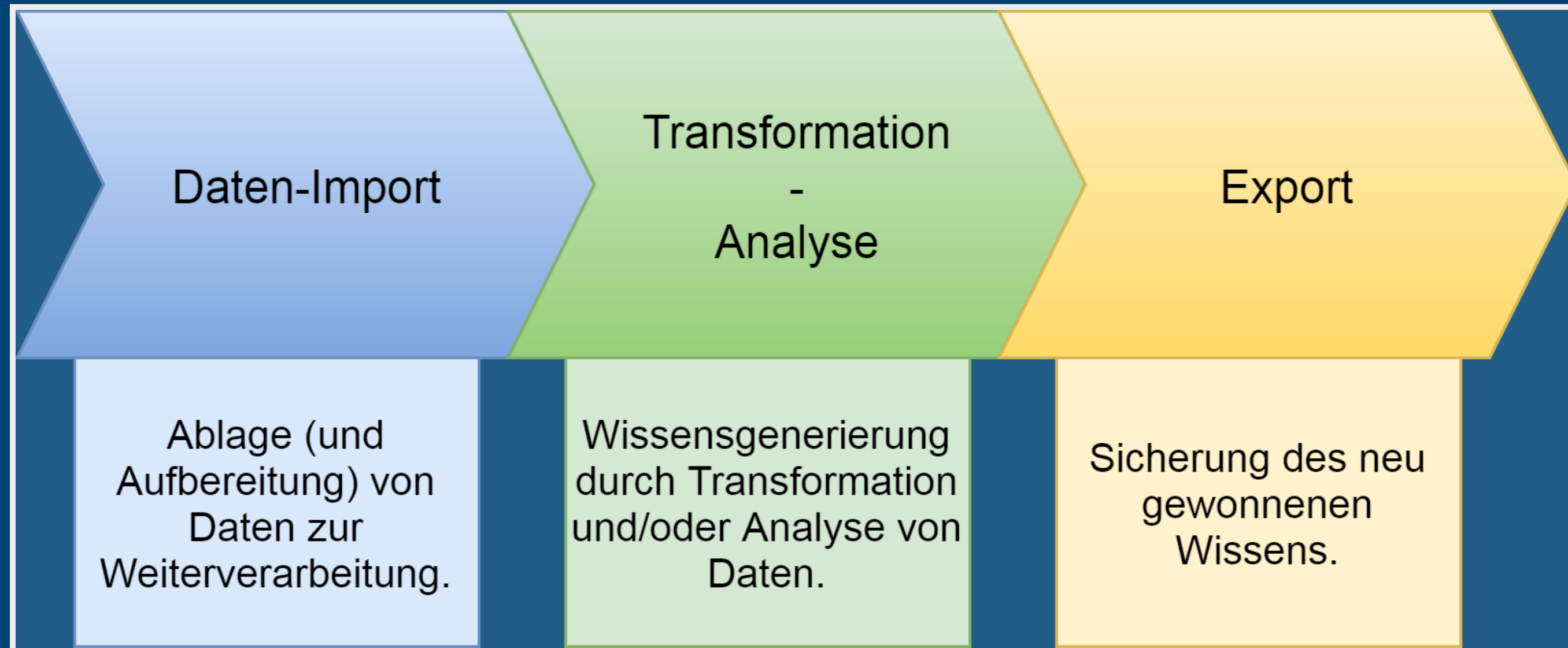
2013: Umzug auf GitHub

2019: Mit der Zeit immer weniger DNB-Ressourcen für Metafactory. hbz wird Maintainer

2019: Start von Metafactory Fix

2021: Start von Metafactory Playground

ETL



Aus: culturegraph, <https://github.com/culturegraph/culturegraph-workflow/blob/master/docs/src/docs/asciidoc/manual.adoc>

Wie Metafactory funktioniert

Wie Metafactory funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →

Wie Metafactory funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →

Jedes Modul erwartet Input eines bestimmten Typs und erzeugt Output eines bestimmten Typs

Wie Metafactory funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →

Jedes Modul erwartet Input eines bestimmten Typs und erzeugt Output eines bestimmten Typs

Verschiedene Formate werden unterstützt (z.B. PICA, MARC), erweiterbares Framework für eigene Formate

Wie Metafactory funktioniert

Grundidee: Daten fließen durch mehrere Module:

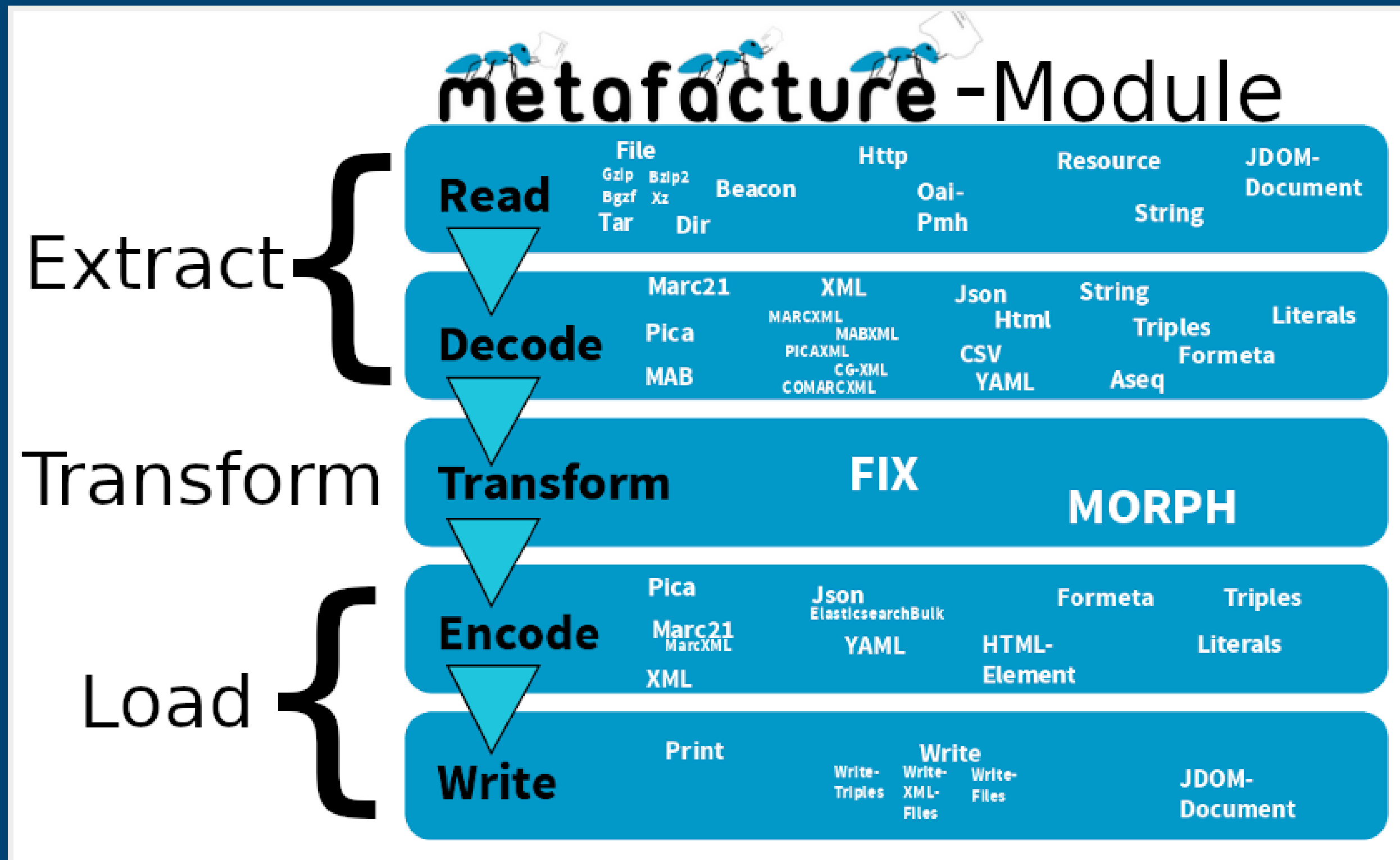
→ read → decode → transform → encode → write →

Jedes Modul erwartet Input eines bestimmten Typs und erzeugt Output eines bestimmten Typs

Verschiedene Formate werden unterstützt (z.B. PICA, MARC), erweiterbares Framework für eigene Formate

Mittels Kombination einzelner Module bauen wir einen Workflow, durch die unsere Daten fließen

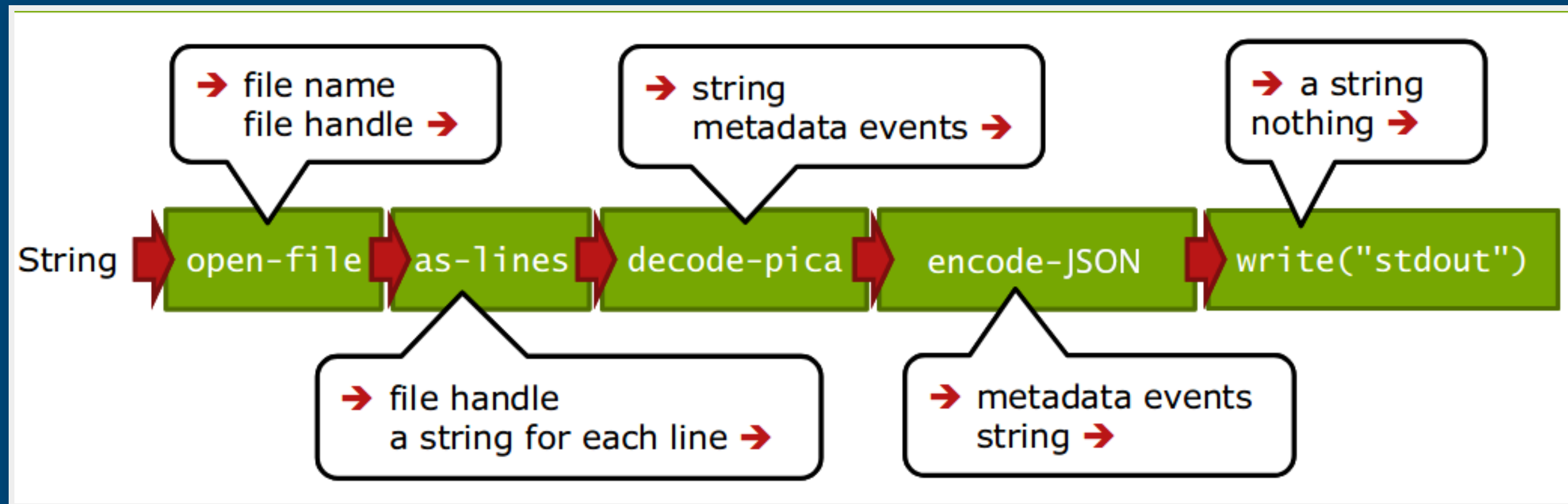
Metafactory-Module



2. Metafactory- Workflows

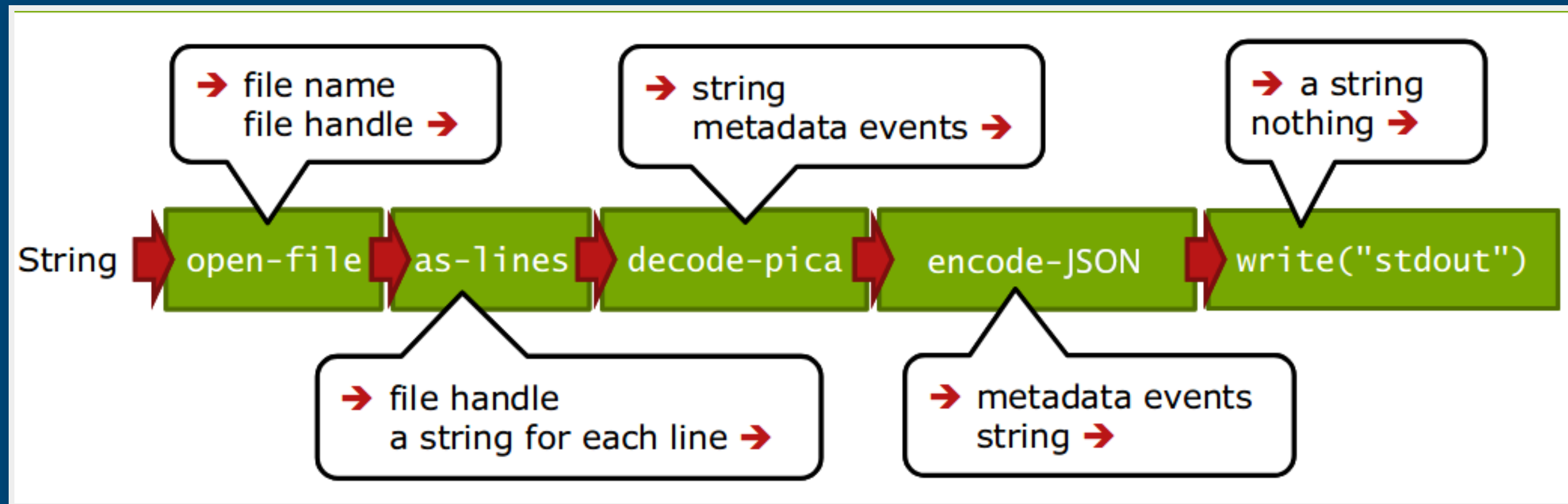
Ein Workflow

Ein Workflow



Basierend auf: Christoph Böhme, http://swib.org/swib13/slides/boehme_swib13_131.pdf

Ein Workflow



Basierend auf: Christoph Böhme, http://swib.org/swib13/slides/boehme_swib13_131.pdf

```
"dateiname" | open-file | as-lines | decode-pica | encode-  
json | write("stdout");
```

(Beispiel-Inhalt einer "flux" Datei)

Workflows konfigurieren und ausführen

Workflows konfigurieren und ausführen

Workflows können in Flux (einer speziellen Konfigurationssprache) oder mit Java (typischer über Java Generics) bearbeitet werden

Workflows konfigurieren und ausführen

Workflows können in Flux (einer speziellen Konfigurationssprache) oder mit Java (typischer über Java Generics) bearbeitet werden

Flux-Workflows können in einem Texteditor editiert und auf der Kommandozeile ausgeführt werden; Java-Workflows funktionieren wie andere Java-Komponenten

Workflows konfigurieren und ausführen

Workflows können in Flux (einer speziellen Konfigurationssprache) oder mit Java (typischer über Java Generics) bearbeitet werden

Flux-Workflows können in einem Texteditor editiert und auf der Kommandozeile ausgeführt werden; Java-Workflows funktionieren wie andere Java-Komponenten

Der Workshop führt in die Nutzung der Flux-Workflows ein, zum Ausführen verwenden wir den Metafactory Playground

Flux-Dokumentation

Flux-Dokumentation

Was für Module gibt es? Was machen die?

Flux-Dokumentation

Was für Module gibt es? Was machen die?

<https://github.com/metafactory/metafactory-documentation/blob/master/flux-commands.md>

Flux-Dokumentation

Was für Module gibt es? Was machen die?

<https://github.com/metafactory/metafactory-documentation/blob/master/flux-commands.md>

decode-json

- description: Decodes JSON to metadata events. The 'recordPath' option can be used to set a JsonPath to extract a path as JSON - or to split the data into multiple JSON documents.
- options: recordid (String), recordcount (int), booleanmarker (String), arraymarker (String), arrayname (String), recordpath (String), allowcomments (boolean), numbermarker (String)
- signature: String -> StreamReceiver
- java class: org.metafactory.json.JsonDecoder

Metafactory Playground

Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und
Austauschen von Workflows

Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und
Austauschen von Workflows

Ziel: Einstiegshürde für Metafactory senken, unserer
Erfahrung nach ein zentrales Problem bei der Metafactory-
Nutzung

Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und
Austauschen von Workflows

Ziel: Einstiegshürde für Metafactory senken, unserer
Erfahrung nach ein zentrales Problem bei der Metafactory-
Nutzung

Für Entwicklung, Dokumentation, Tutorials, Workshops

Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und
Austauschen von Workflows

Ziel: Einstiegshürde für Metafactory senken, unserer
Erfahrung nach ein zentrales Problem bei der Metafactory-
Nutzung

Für Entwicklung, Dokumentation, Tutorials, Workshops

<https://metafactory.org/playground>

Wie funktioniert das in der Praxis? Lasst uns das gemeinsam ausprobieren.

(die folgenden Screenshots verlinken die Beispiele zum Ausprobieren im Playground)



Metafactory Playground

metafactory-framework 5.5.0

metafix 0.5.1

[Load Examples](#) [Clear](#) [Process](#) [Share](#) [Import Workflow](#) [Export Workflow](#)

Data ^

1

Flux ^

```
1 "Hello, friend. I'am Metafactory!"
2 |print
3 ;
```

Fix **Morph** ^

1

Ändere den Input in eine Variable. Ändere das Resultat zu "Hello World, I am". Lasst uns das gemeinsam ausprobieren.

(die folgenden Screenshots verlinken die Beispiele zum Ausprobieren im Playground)

Load Examples Clear Process Share Import Workflow Export Workflow

Data

1

Flux

```
1 INPUT="Hello, friend. I'am Metafacture!";
2
3 INPUT
4 |print
5 ;
```

Fix Morph

1

Result

```
1 Hello, friend. I'am Metafacture!
2
```

Verschiebe den Input in den Playground

Nutze die Variable PG_DATA und dem Data-Fenster

(die folgenden Screenshots verlinken die Beispiele zum Ausprobieren im Playground)

Load Examples Clear Process Share Import Workflow Export Workflow

Data

```
1 Hello, friend. I'am Metafacture!
```

Flux

```
1 PG_DATA
2 | as-lines
3 | print
4 ;
```

Fix Morph

```
1
```

Dann lasst uns doch mal PICA-Daten zu JSON verarbeiten

(die folgenden Screenshots verlinken die Beispiele zum Ausprobieren im Playground)

Data

```
1 001@ a5 01-2 001A 01100:15-10-94 001B 09999:12-06-06 t16:10:17.000 001D 09999:99-99-99 001U 0utf8 001X 00 00
2 001@ 01 a5 001A 01140:08-12-99 001B 09999:05-01-08 t22:57:29.000 001D 09999:99-99-99 001U 0utf8 001X 00 002@
3 001@ a5 01 001A 01140:19-02-03 001B 09999:19-06-11 t01:20:13.000 001D 09999:26-04-03 001U 0utf8 001X 00 002@
4 001@ a5 01-3 001A 01240:01-08-95 001B 09999:24-09-10 t17:42:20.000 001D 09999:99-99-99 001U 0utf8 001X 00 00
5 001@ 01-2 a5 001A 01239:18-08-11 001B 09999:05-09-11 t23:31:44.000 001D 01240:30-08-11 001U 0utf8 001X 00 00
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | encode-json
5 | print
6 ;
```

Fix

Morph

```
1
```

Übung: Formatierung & Optionen

Playground-Beispiel anpassen:

```
PG_DATA
| as-lines
| decode-pica
| encode-json(prettyPrinting = "true")
| print
;
```

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aa003@ 095809056
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U Outf8001X 00002@ 0Aal003@ 036180957
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U Outf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U Outf8001X 00002@ 0Af003@ 01014417
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | encode-json(prettyPrinting = "true")
5 | print
6 ;
```

Fix Morph

Result

```
{
  "001@":{
    "a":"5",
    "O":"1-2"
  },
  "001A":{
    "O":"1100:15-10-94"
  },
}
```

Übung:

Marc21-Daten aus dem Web öffnen und zu YAML transformieren

```
"https://raw.githubusercontent.com/metafactory/metafactory-  
core/master/metafactory-  
runner/src/main/dist/examples/read/marc21/10.marc21"  
| open-http  
  
...  
;
```

`open-http` ruft Daten aus dem Web ab.

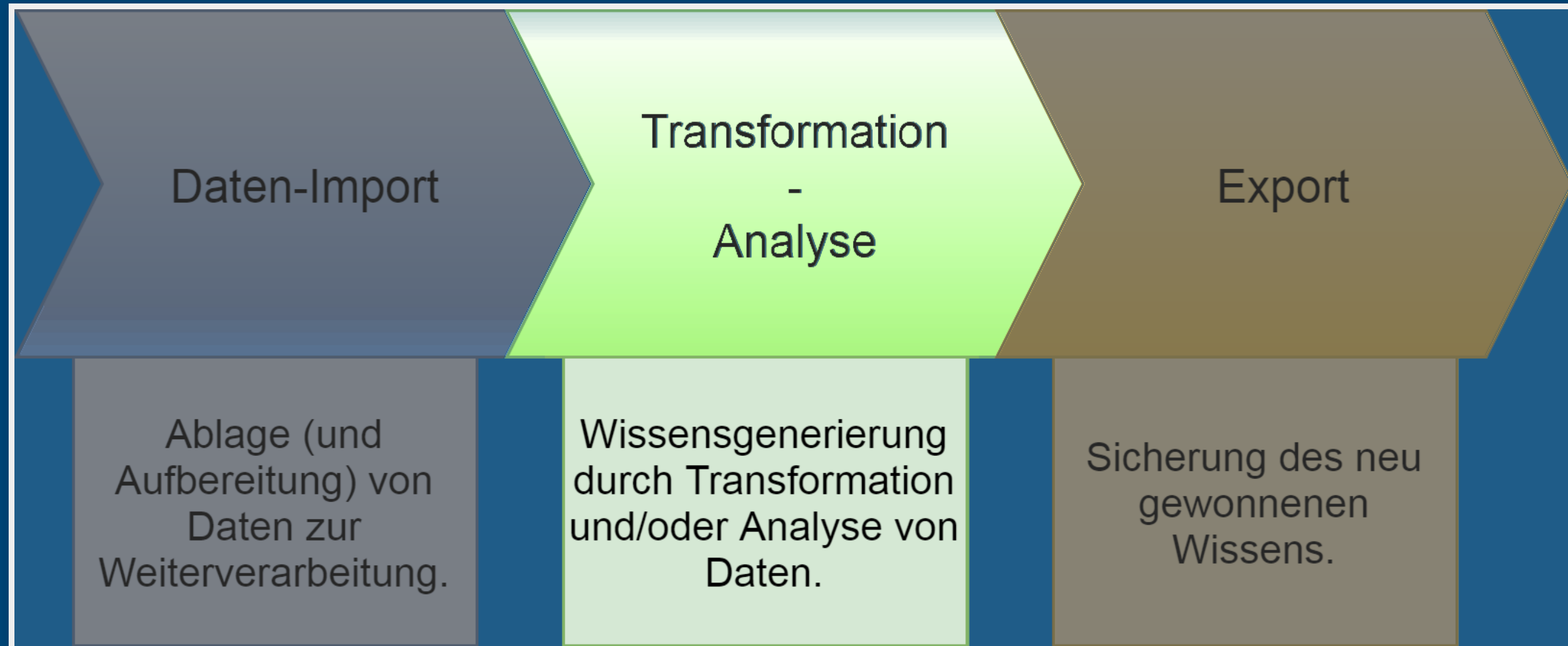
Schaut in die [Doku](#), wie ihr MARC21 decodiert und YAML encodiert

```
Flux
1  "https://raw.githubusercontent.com/metafactory/metafactory-core/master/metafactory-runner/src/main/dist/examples/read/r
2  | open-http
3  | as-lines
4  | decode-marc21
5  | encode-yaml
6  | print
7  ;
```

sample1_Marc21_to_YAML

3. Transformieren

ETL: Transformation



Transformieren

Transformieren

Manipulation von Feldnamen und -werten; filtern, kombinieren, trennen, normalisieren etc.

Transformieren

Manipulation von Feldnamen und -werten; filtern, kombinieren, trennen, normalisieren etc.

Änderung der Struktur und Hierarchie eines Records etc.

Transformieren

Manipulation von Feldnamen und -werten; filtern, kombinieren, trennen, normalisieren etc.

Änderung der Struktur und Hierarchie eines Records etc.

Feldwerte aus Lookup-Tabellen in externen Dateien (z.B. Freitextfelder -> kontrollierte Vokabulare)

Transformationsmodule

Transformationsmodule

Fix: eigene, Catmandu-Fix-artige Sprache, Record-basiert

Transformationsmodule

Fix: eigene, Catmandu-Fix-artige Sprache, Record-basiert

Morph: XML-basiert, Feld- / Metadaten-Event-Ebene

Fix Doku und Konzepte

Fix Doku und Konzepte

Funktionen ("Simple Fixes")

Fix Doku und Konzepte

Funktionen ("Simple Fixes")

```
add_field("hello", "world")  
remove_field("my.deep.nested.junk")  
replace_all("creator", "Dr.", "")
```

Fix Doku und Konzepte

Funktionen ("Simple Fixes")

```
add_field("hello", "world")  
remove_field("my.deep.nested.junk")  
replace_all("creator", "Dr.", "")
```

Conditionals (Bedingungen)

Fix Doku und Konzepte

Funktionen ("Simple Fixes")

```
add_field("hello", "world")  
remove_field("my.deep.nested.junk")  
replace_all("creator", "Dr.", "")
```

Conditionals (Bedingungen)

```
if exists("id")  
  set_field("is_valid", "yes")  
else  
  set_field("is_valid", "no")  
end
```

Fix Doku und Konzepte

Funktionen ("Simple Fixes")

```
add_field("hello", "world")
remove_field("my.deep.nested.junk")
replace_all("creator", "Dr.", "")
```

Conditionals (Bedingungen)

```
if exists("id")
  set_field("is_valid", "yes")
else
  set_field("is_valid", "no")
end
```

Binds/Loops

Fix Doku und Konzepte

Funktionen ("Simple Fixes")

```
add_field("hello", "world")
remove_field("my.deep.nested.junk")
replace_all("creator", "Dr.", "")
```

Conditionals (Bedingungen)

```
if exists("id")
  set_field("is_valid", "yes")
else
  set_field("is_valid", "no")
end
```

Binds/Loops

```
do list(path:"arrayPath[]", "var": "$i")
  add_field("$i.foo", "bar")
end
```

Metafactory Fix: Ziele

Metafactory Fix: Ziele

Erleichterung der Transformationskonfiguration

Metafactory Fix: Ziele

Erleichterung der Transformationskonfiguration

Anknüpfung an existierende Konfigurationssprache aus

Catmandu (mittelfristiges Ziel: Standardisierung, s.
<https://github.com/elag/FIG>)

Metafactory Fix: Ziele

Erleichterung der Transformationskonfiguration

Anknüpfung an existierende Konfigurationssprache aus **Catmandu** (mittelfristiges Ziel: Standardisierung, s. <https://github.com/elag/FIG>)

Vergrößerung der Zielgruppe um Bibliothekar:innen und andere Metadatenfachleute (bei uns z.B. in OERSI, erster Anwendungsfall und Entwicklungsbegleitung zu Fix)

Kurzer Exkurs: Metadatenstrukturen

```
title: Ordinary Vices  
author: Judith Shklar  
publisher: Belknap Press  
date: 1984
```

Kurzer Exkurs: Metadatenstrukturen

```
title: Ordinary Vices  
author: Judith Shklar  
publisher: Belknap Press  
date: 1984
```

Wir sehen...

Kurzer Exkurs: Metadatenstrukturen

```
title: Ordinary Vices  
author: Judith Shklar  
publisher: Belknap Press  
date: 1984
```

Wir sehen...

...**Metadaten** für ein Buch

Kurzer Exkurs: Metadatenstrukturen

```
title: Ordinary Vices  
author: Judith Shklar  
publisher: Belknap Press  
date: 1984
```

Wir sehen...

...**Metadaten** für ein Buch

...einen **Record/Datensatz** eine Sammlung von Metadaten zu einem Gegenstand (z.B. Buch)

Kurzer Exkurs: Metadatenstrukturen

```
title: Ordinary Vices  
author: Judith Shklar  
publisher: Belknap Press  
date: 1984
```

Wir sehen...

...**Metadaten** für ein Buch

...einen **Record/Datensatz** eine Sammlung von Metadaten zu einem Gegenstand (z.B. Buch)

...**Elemente/Felder(/Properties/Attribute/Keys)**

Kurzer Exkurs: Metadatenstrukturen

```
title: Ordinary Vices  
author: Judith Shklar  
publisher: Belknap Press  
date: 1984
```

Wir sehen...

...**Metadaten** für ein Buch

...einen **Record/Datensatz** eine Sammlung von Metadaten zu einem Gegenstand (z.B. Buch)

...**Elemente/Felder(/Properties/Attribute/Keys)**

...**Werte/Values**

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Unterfeld/Unterelement und Objekte

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Unterfeld/Unterelement und Objekte

```
author:  
  firstName: Judith  
  lastName: Shklar
```

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Unterfeld/Unterelement und Objekte

```
author:  
  firstName: Judith  
  lastName: Shklar
```

Attribute (im Kontext XML)

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Unterfeld/Unterelement und Objekte

```
author:  
  firstName: Judith  
  lastName: Shklar
```

Attribute (im Kontext XML)

```
<creator type="author">Judith Shklar</creator>
```

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Unterfeld/Unterelement und Objekte

```
author:  
  firstName: Judith  
  lastName: Shklar
```

Attribute (im Kontext XML)

```
<creator type="author">Judith Shklar</creator>
```

Array/Liste (von einfachen Werten oder Objekten)

Kurzer Exkurs: Metadatenstrukturen- Besonderheiten

Unterfeld/Unterelement und Objekte

```
author:  
  firstName: Judith  
  lastName: Shklar
```

Attribute (im Kontext XML)

```
<creator type="author">Judith Shklar</creator>
```

Array/Liste (von einfachen Werten oder Objekten)

```
keywords:  
- Politische Theorie  
- Philosophie  
- Grausamkeit
```

Felder anwählen: Pfade

```
id: 978-3-518-10639-6
title:
  mainTitle: Öffentlichkeit und Erfahrung
  subTitle: Zur Organisationsanalyse von bürgerlicher und proletarischer Öffentlichkeit
creator:
  firstName: Alexander
  lastName: Kluge
creator:
  firstName: Oskar
  lastName: Negt
```


Felder anwählen: Pfade

```
id: 978-3-518-10639-6
title:
  mainTitle: Öffentlichkeit und Erfahrung
  subTitle: Zur Organisationsanalyse von bürgerlicher und proletarischer Öffentlichkeit
creator:
  firstName: Alexander
  lastName: Kluge
creator:
  firstName: Oskar
  lastName: Negt
```

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre **Pfade** angeben

Felder anwählen: Pfade

```
id: 978-3-518-10639-6
title:
  mainTitle: Öffentlichkeit und Erfahrung
  subTitle: Zur Organisationsanalyse von bürgerlicher und proletarischer Öffentlichkeit
creator:
  firstName: Alexander
  lastName: Kluge
creator:
  firstName: Oskar
  lastName: Negt
```

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre **Pfade** angeben

Einfache Elemente der obersten Ebene / Felder: z.B. `id`

Felder anwählen: Pfade

```
id: 978-3-518-10639-6
title:
  mainTitle: Öffentlichkeit und Erfahrung
  subTitle: Zur Organisationsanalyse von bürgerlicher und proletarischer Öffentlichkeit
creator:
  firstName: Alexander
  lastName: Kluge
creator:
  firstName: Oskar
  lastName: Negt
```

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre **Pfade** angeben

Einfache Elemente der obersten Ebene / Felder: z.B. `id`

Elemente auf einer unteren Ebene / Unterfelder: z.B. `title.mainTitle`

Felder anwählen: Pfade

```
id: 978-3-518-10639-6
title:
  mainTitle: Öffentlichkeit und Erfahrung
  subTitle: Zur Organisationsanalyse von bürgerlicher und proletarischer Öffentlichkeit
creator:
  firstName: Alexander
  lastName: Kluge
creator:
  firstName: Oskar
  lastName: Negt
```

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre **Pfade** angeben

Einfache Elemente der obersten Ebene / Felder: z.B. `id`

Elemente auf einer unteren Ebene / Unterfelder: z.B. `title.mainTitle`

Wiederholte Felder/Arrays werden als Listen mit Index-Nummer angegeben: z.B. `creator.2.firstName`

Felder anwählen: Besonderheiten

Felder anwählen: Besonderheiten

JSON-Arrays bekommen in der FIX am Ende des Namens
einen Arraymarker: "array[]"

Felder anwählen: Besonderheiten

JSON-Arrays bekommen in der FIX am Ende des Namens
einen Arraymarker: "array[]"

Pfad-Wildcards: u.a. ? und * für Feldnamen

Felder anwählen: Besonderheiten

JSON-Arrays bekommen in der FIX am Ende des Namens
einen Arraymarker: "array[]"

Pfad-Wildcards: u.a. ? und * für Feldnamen

Array-Wildcards: u.a. * (alle), \$append (neues anhängen),
\$last (nur letzter)

Feld-Pfad-Übersicht

Feld-Pfad-Übersicht

Bevor man anfängt mit Feldern zu arbeiten ist es nützlich eine Übersicht zu bekommen

Feld-Pfad-Übersicht

Bevor man anfängt mit Feldern zu arbeiten ist es nützlich
eine Übersicht zu bekommen

Dazu wollen wir die verfügbaren Feld-Pfade und ihre Werte
ausgeben

Feld-Pfad-Übersicht

Bevor man anfängt mit Feldern zu arbeiten ist es nützlich eine Übersicht zu bekommen

Dazu wollen wir die verfügbaren Feld-Pfade und ihre Werte ausgeben

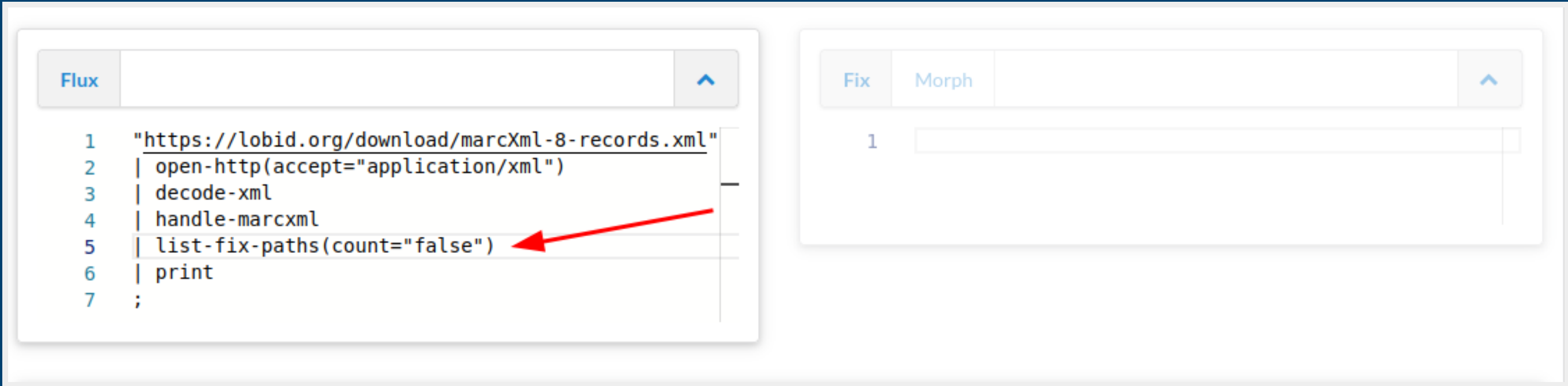
Dabei hilft uns das neue Flux-Commando ``list-fix-paths``

Flux

```
1 "https://lobid.org/download/marcXml-8-records.xml"
2 | open-http(accept="application/xml")
3 | decode-xml
4 | handle-marcxml
5 | list-fix-paths(count="false")
6 | print
7 ;
```

Fix Morph

1



Feld auswählen

Feld auswählen

Auf Basis dieser Übersicht ein Feld auswählen, z.B. 021A
(Titel)

Feld auswählen

Auf Basis dieser Übersicht ein Feld auswählen, z.B. 021A
(Titel)

alles andere weglassen: retain, s. Dokumentation

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aa003@ 095809056
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U Outf8001X 00002@ 0Aal003@ 036180957
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U Outf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U Outf8001X 00002@ 0Af003@ 0101441
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix("retain('021A')")
5 | encode-json(prettyPrinting = "true")
6 | print;
```

Fix Morph

```
1
```

Result

```
{
  "021A":{
    "a": "Die @Berufsfreiheit der Arbeitnehmer und ihre Ausgestaltung in völkerrechtlichen Verträgen",
    "d": "Eine Grundrechtsbetrachtg"
  }
}
```

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 0958090564
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U 0utf8001X 00002@ 0Aal003@ 036180957
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U 0utf8001X 00002@ 0Af003@ 01014417
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | encode-json(prettyPrinting = "true")
6 | print;
```

Fix Morph

```
1 retain(['021A'])
```

Result

```
{
  "021A": {
    "a": "Die @Berufsfreiheit der Arbeitnehmer und ihre Ausgestaltung in völkerrechtlichen Verträgen",
    "d": "Eine Grundrechtsbetrachtg"
  }
}
```

move_field, paste, retain

move_field, paste, retain

```
{"a": "Faust", "b": {"n": "Goethe", "v": "JW"}, "c": "Weimar"}  
{"a": "Räuber", "b": {"n": "Schiller", "v": "F"}, "c":  
"Weimar"}
```

move_field, paste, retain

```
{"a": "Faust", "b": {"n": "Goethe", "v": "JW"}, "c": "Weimar"}  
{"a": "Räuber", "b": {"n": "Schiller", "v": "F"}, "c":  
"Weimar"}
```

```
move_field(a, title)
```

move_field, paste, retain

```
{"a": "Faust", "b": {"n": "Goethe", "v": "JW"}, "c": "Weimar"}  
{"a": "Räuber", "b": {"n": "Schiller", "v": "F"}, "c":  
"Weimar"}
```

```
move_field(a, title)
```

```
paste(author, b.v, b.n, '~aus', c)
```

move_field, paste, retain

```
{"a": "Faust", "b": {"n": "Goethe", "v": "JW"}, "c": "Weimar"}  
{"a": "Räuber", "b": {"n": "Schiller", "v": "F"}, "c":  
"Weimar"}
```

```
move_field(a, title)
```

```
paste(author, b.v, b.n, '~aus', c)
```

```
retain(title, author)
```

move_field, paste, retain

```
{"a": "Faust", "b": {"n": "Goethe", "v": "JW"}, "c": "Weimar"}  
{"a": "Räuber", "b": {"n": "Schiller", "v": "F"}, "c":  
"Weimar"}
```

```
move_field(a, title)
```

```
paste(author, b.v, b.n, '~aus', c)
```

```
retain(title, author)
```

[Beispiel im Playground](#)

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 {"a": "Faust", "b": {"n": "Goethe", "v": "JW"}, "c": "Weimar"}
2 {"a": "Räuber", "b": {"n": "Schiller", "v": "F"}, "c": "Weimar"}
```

Flux

```
1 PG_DATA
2 |as-lines
3 |decode-json
4 |fix
5 |encode-xml(rootTag="collection")
6 |print
7 ;
```

Fix

Morph

```
1 move_field("_id", "id")
2 move_field("a", "title")
3 paste("creator", "b.v", "b.n", "~aus", "c")
4 retain("id", "title", "creator")
```

Result

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <collection>
3
4   <record>
5     <id>1</id>
6     <title>Faust</title>
7     <creator>JW Goethe aus Weimar</creator>
8   </record>
9
```

Übung

Übung

Aufgaben für MARC-Transformation in kleinen Schritten

Load Examples
Clear
Process
Share
Import Workflow
Export Workflow

Data

```

1 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aa003@ 0958090564
2 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U Outf8001X 00002@ 0Aal003@ 036180954
3 001@ 02a5001A 01200:24-11-77001B 09999:16-01-08t01:01:54.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aan003@ 078051014
4 001@ a501-2001A 01140:28-06-99001B 09999:17-04-11t14:40:50.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aa003@ 09567459
5 001@ 01a5001A 09999:08-03-02001B 09999:12-03-04t11:47:27.000001D 09999:08-03-02001U Outf8001X 00002@ 0Aal003@ 057612144

```

Flux

```

1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | encode-json(prettyPrinting = "true")
6 | print
7 ;

```

Fix Morph

```

1 move_field('021A.a', 'Title')
2 move_field('011@a', Year)
3 paste('Publisher', '033A.n', '~: ', '033A.p')
4 retain(Title, Year, Publisher)

```

Result

```

{
  "Title": "Zukunft Bildung",
  "Year": "1999",
  "Publisher": "Polit. Akad.: Wien"
}

```

Lookup

Lookup

"Feldwerte aus Lookup-Tabellen in externen Dateien (z.B. Freitextfelder -> kontrollierte Vokabulare)"

Übung

Übung

Feld 002@.0 → dcterms:format

Übung

Feld 002@.0 → dcterms:format

In 002@.0, Position 1, A: print, B: audiovisual, O: online

Übung

Feld 002@.0 → dcterms:format

In 002@.0, Position 1, A: print, B: audiovisual, O: online

z.B. mit `copy_field`, `substring`, `lookup`, `retain`, `s`.

Dokumentation

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | encode-json(prettyPrinting = "true")
6 | print;
```

Fix

Morph

```
1 copy_field('002@.0', 'dcterms:format')
2 substring('dcterms:format', '0', '1')
3 lookup('dcterms:format', A: print, B: audiovisual, O: online)
4 retain('002@', 'dcterms:format')
```

Result

```
{
  "002@":{
    "0": "Aa"
  },
  "dcterms:format": "print"
}
{
  "002@":{
    "0": "Aa"
  },
  "dcterms:format": "print"
}
```


Analysieren

Beispiel: Feldwerte zählen

Beispiel: Feldwerte zählen

Anzahl unterschiedlicher Werte im Edition-Feld (032@a)

Beispiel: Feldwerte zählen

Anzahl unterschiedlicher Werte im Edition-Feld (032@.a)

```
PG_DATA
| as-lines
| decode-pica
| list-fix-values("032@a")
| print
;
```


Data

```

1  9118700561 7Tp1 Vpiz Agnd 0118700561 E1900 B1997 aHager dWerner 033A pLeipzig 034D a82 S. 034I a4 037A aAuch in Bu
2  sisch! dEine Orestie mit glücl. Ausgang hWalter Mehring (1896 - 1981). Mit e. Nachw. hrsg. von Didier Plassard. Univ.
3  er Knabenzeit hvon Franziskus Nagler 028A 9116880430 7Tp1 Vpiz Agnd 0116880430 E1873 B1957 aNagler dFranciscus 032@
4

```

Flux

```

1  PG_DATA
2  | as-lines
3  | decode-pica
4  | list-fix-values("032@a")
5  | print
6  ;

```

Fix Morph

```

1

```

Result

```

1  1 | (2. Aufl.)
2  1 | 3. Aufl.
3

```

sample7_Count_values

Lasst uns jetzt mal PICA-Daten aus dem Web abrufen und
ansonsten den Workflow übernehmen. Von der Quelle:
[https://github.com/hbz/metafactory-flux-
examples/blob/master/sample4/bib-data-1k.pica?raw=true](https://github.com/hbz/metafactory-flux-examples/blob/master/sample4/bib-data-1k.pica?raw=true)

Flux

```
1 "https://github.com/hbz/metafactory-flux-examples/blob/master/sample4/bib-data-1k.pica?raw=true"  
2 | open-http  
3 | as-lines  
4 | decode-pica  
5 | list-fix-values("032@a")  
6 | print  
7 ;
```

6. Komplexere Beispiele

Was u.a. nicht vorkam

komplexe Transformation

Was u.a. nicht vorkam

komplexe Transformation

Komplexere Analyse, z.B. Muster zählen

Was u.a. nicht vorkam

komplexe Transformation

Komplexere Analyse, z.B. Muster zählen

Daten aus verschiedenen Quellen sammeln und zu Records
zusammensetzen: `collect-triples`

Fix: Beispiele & Dokumentation

Fix: Beispiele & Dokumentation

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),
für ALMA **hbz-Verbundkatalog** (ALMA MARC → JSON) und
Rheinland-Pfälzische Bibliographie (Allegro → JSON)

Fix: Beispiele & Dokumentation

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),
für ALMA **hbz-Verbundkatalog** (ALMA MARC → JSON) und
Rheinland-Pfälzische Bibliographie (Allegro → JSON)

Integration-Tests: Input, Flux, Fix, Output als Dateien wie bei
Real-World-Setup

Fix: Beispiele & Dokumentation

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),
für ALMA **hbz-Verbundkatalog** (ALMA MARC → JSON) und
Rheinland-Pfälzische Bibliographie (Allegro → JSON)

Integration-Tests: Input, Flux, Fix, Output als Dateien wie bei
Real-World-Setup

Aktuelle Dokumentation, GitHub-Repo, Playground

7. Ausblick, Fragen & Diskussion

Ausblick

Fix & Playground weiterentwickeln

Fix & Playground weiterentwickeln

Fix-Funktionalität erweitern, Fehler beheben, Catmandu-
Kompatibilität erhöhen

Fix & Playground weiterentwickeln

Fix-Funktionalität erweitern, Fehler beheben, Catmandu-Kompatibilität erhöhen

Playground weiter verbessern, z.B. mehr Hinweise im Editor, integrierte Dokumentation (was gibt es für Module, wie kann ich sie kombinieren)

Standards nutzen und aufbauen

Standards nutzen und aufbauen

SKOS Lookups (zum Andocken an [SkoHub Vocab](#)s)

Standards nutzen und aufbauen

SKOS Lookups (zum Andocken an [SkoHub Vocab](#)s)

[Entity Reconciliation](#) mit [OpenRefine](#)-kompatiblen Diensten

Standards nutzen und aufbauen

SKOS Lookups (zum Andocken an [SkoHub Vocabs](#))

Entity Reconciliation mit [OpenRefine](#)-kompatiblen Diensten

Fix-Standardisierung, s. <https://github.com/elag/FIG>

ETL Hub

ETL: Extract, Transform, Load

ETL Hub

ETL: Extract, Transform, Load

mehr Kollaboration, Teilen & Auffinden von Workflows
ermöglichen (nicht nur für Metafactory)

ETL Hub

ETL: Extract, Transform, Load

mehr Kollaboration, Teilen & Auffinden von Workflows
ermöglichen (nicht nur für Metafactory)

Entwicklung von Best Practices zur Paketierung und
Beschreibung von ETL-Konfigurationen

ETL Hub

ETL: Extract, Transform, Load

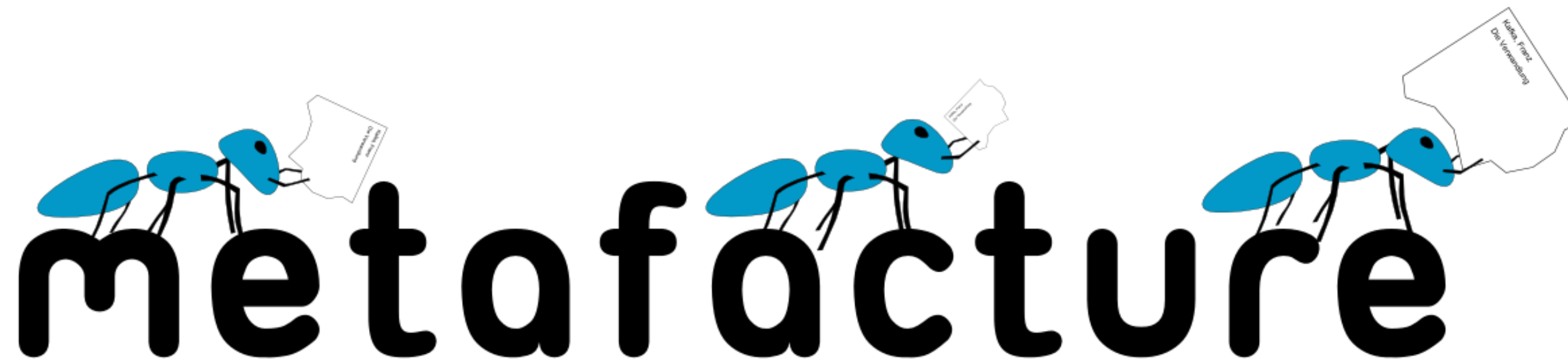
mehr Kollaboration, Teilen & Auffinden von Workflows
ermöglichen (nicht nur für Metafactory)

Entwicklung von Best Practices zur Paketierung und
Beschreibung von ETL-Konfigurationen

Aufbau eines ETL Hubs zum Entdecken existierender ETL-
Prozesse für die einfache Nachnutzung und Anpassung

Fragen und Diskussion

... gerne auch gleich am Stand der Verbände (14:00 - 15:00)



metafacture

<https://metafacture.org>