

Creating LOUD with Metafacture

Pascal Christoph & Tobias Bülte
Metadateninfrastruktur, Hochschulbibliothekszentrum NRW (hbz)

Workshop

SWIB 23

Berlin, 11. September 2023

<https://slides.lobid.org/2023-09-metafacture-workshop-swib/> : (PDF)



Who are we ?

Who are we ?

We are Pascal and Tobias.

Who are we ?

We are Pascal and Tobias.

Pascal is a software engineer. Tobias is a librarian.

Who are we ?

We are Pascal and Tobias.

Pascal is a software engineer. Tobias is a librarian.

We are both part of the Metadateninfrastruktur-Team at
hbz. fka. lobid, offene Infrastruktur.

Who are we 2/2 ?

Who are we 2/2 ?

Metadateninfrastruktur is a mixed team of librarians and developers.

Who are we 2/2 ?

Metadateninfrastruktur is a mixed team of librarians and developers.

We engage in everything open:

Who are we 2/2 ?

Metadateninfrastruktur is a mixed team of librarians and developers.

We engage in everything open:
open source

Who are we 2/2 ?

Metadateninfrastruktur is a mixed team of librarians and developers.

We engage in everything open:

open source

open data

Who are we 2/2 ?

Metadateninfrastruktur is a mixed team of librarians and developers.

We engage in everything open:

open source

open data

open standards

Who are you?

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

1. Tell your name and a little bit about your job

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

1. Tell your name and a little bit about your job
2. Do you have any knowledge in data transformation?

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

1. Tell your name and a little bit about your job
2. Do you have any knowledge in data transformation?
3. In Linked Open Data ?

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

1. Tell your name and a little bit about your job
2. Do you have any knowledge in data transformation?
3. In Linked Open Data ?
4. What do you expect from this course?

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

1. Tell your name and a little bit about your job
2. Do you have any knowledge in data transformation?
3. In Linked Open Data ?
4. What do you expect from this course?
5. Optional: fun fact about you

Who are you?

Let's team up in pairs and introduce yourself in 5 minutes.
Ask your interview partner e.g.:

1. Tell your name and a little bit about your job
2. Do you have any knowledge in data transformation?
3. In Linked Open Data ?
4. What do you expect from this course?
5. Optional: fun fact about you
6. After the 5 minutes: introduce to us your interview partner

Agenda

Part I: Metafacture and Motivation (10m)

Agenda

Part I: Metafacture and Motivation (10m)

Part II: Getting to know MF and FLUX (30m)
- 15m break -

Agenda

Part I: Metafactory and Motivation (10m)

Part II: Getting to know MF and FLUX (30m)
- 15m break -

Part III: More practice and MF FIX (40m)
- 30m break -

Agenda

Part I: Metafactory and Motivation (10m)

Part II: Getting to know MF and FLUX (30m)
- 15m break -

Part III: More practice and MF FIX (40m)
- 30m break -

Part IV: LOUD (2h)

Part I: What is Metafacture?

Part I: What is Metafactory?

Metafactory is a toolkit for processing semi-structured data with a focus on library metadata.

Part I: What is Metafactory?

Metafactory is a toolkit for processing semi-structured data with a focus on library metadata.

Metafactory can be used as a stand-alone CLI application or as a Java/JVM library in other applications.

Part I: What is Metafactory?

Metafactory is a toolkit for processing semi-structured data with a focus on library metadata.

Metafactory can be used as a stand-alone CLI application or as a Java/JVM library in other applications.

Allows Batch- and on the fly-processing

Part I: What is Metafactory?

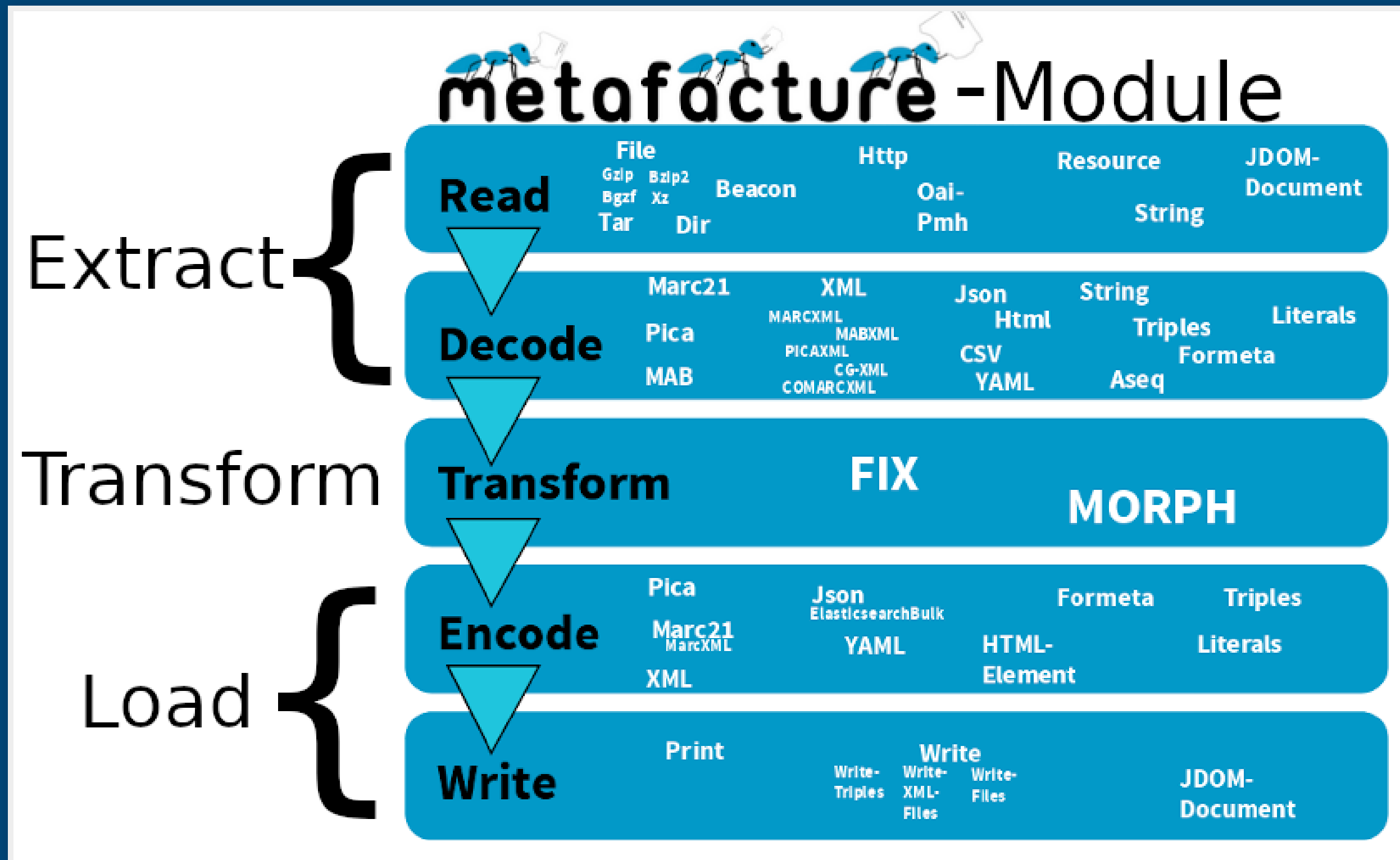
Metafactory is a toolkit for processing semi-structured data with a focus on library metadata.

Metafactory can be used as a stand-alone CLI application or as a Java/JVM library in other applications.

Allows Batch- and on the fly-processing

It is also an open framework. (You can use modules on their own. It's Open Source so you can develop your own modules.)

Metafactory-Module



The Workshop: Why are we doing what we are doing?

The Workshop: Why are we doing what we are doing?

Metafactory helps us to transform e.g. MARC21-XML, CSV and PICA to JSON-LD to index that into elasticsearch (JSON search engine based on lucene) for lobid to publish it as LOUD via our web API, properly documented.

The Workshop: Why are we doing what we are doing?

Metafactory helps us to transform e.g. MARC21-XML, CSV and PICA to JSON-LD to index that into elasticsearch (JSON search engine based on lucene) for lobid to publish it as LOUD via our web API, properly documented.

The tools are very flexible so that we can easily and continuously develop our API and our LOUD in conversation with our consumers/customers.

The Workshop: Why are we doing what we are doing?

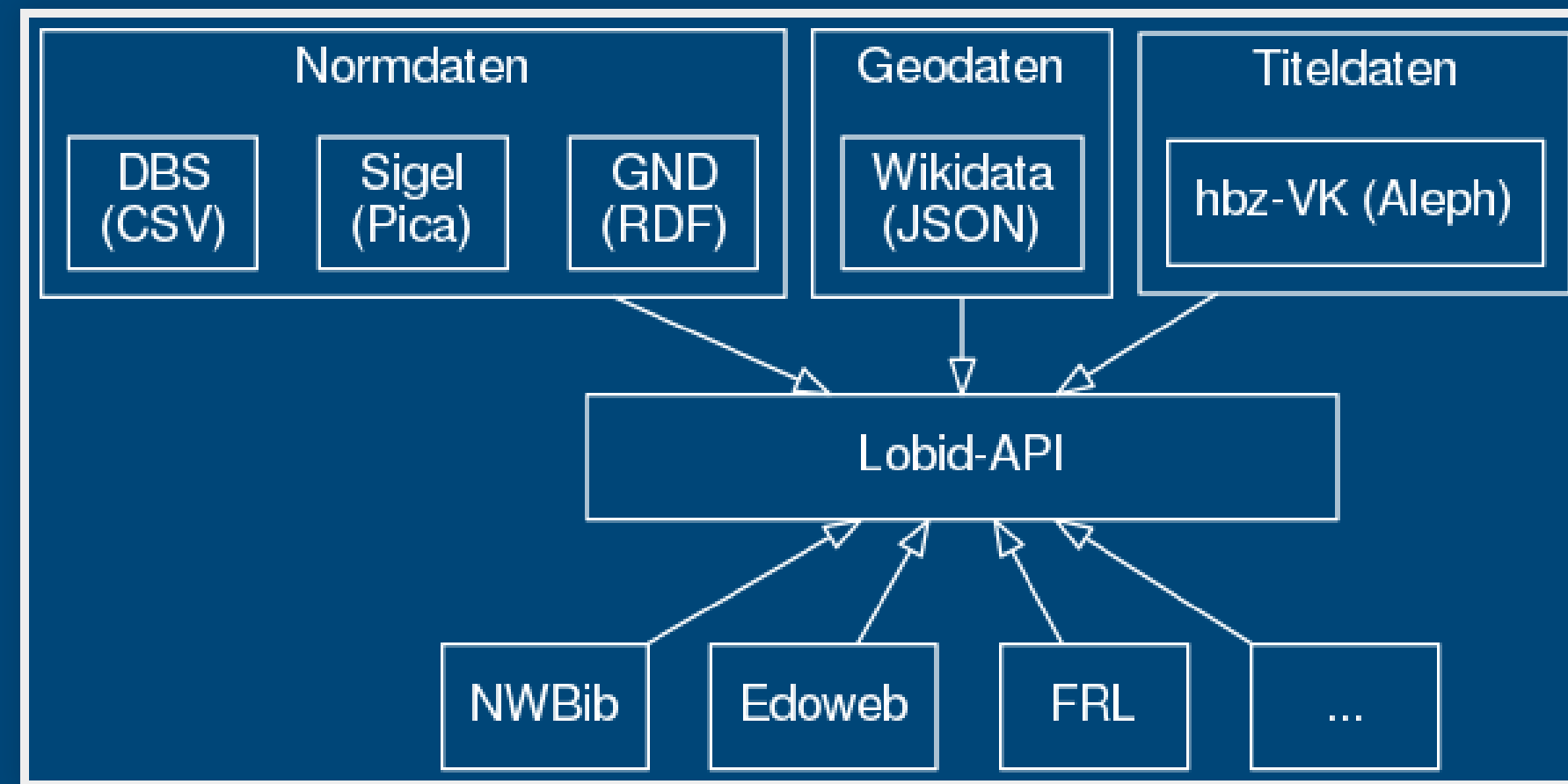
Metafactory helps us to transform e.g. MARC21-XML, CSV and PICA to JSON-LD to index that into elasticsearch (JSON search engine based on lucene) for lobid to publish it as LOUD via our web API, properly documented.

The tools are very flexible so that we can easily and continuously develop our API and our LOUD in conversation with our consumers/customers.

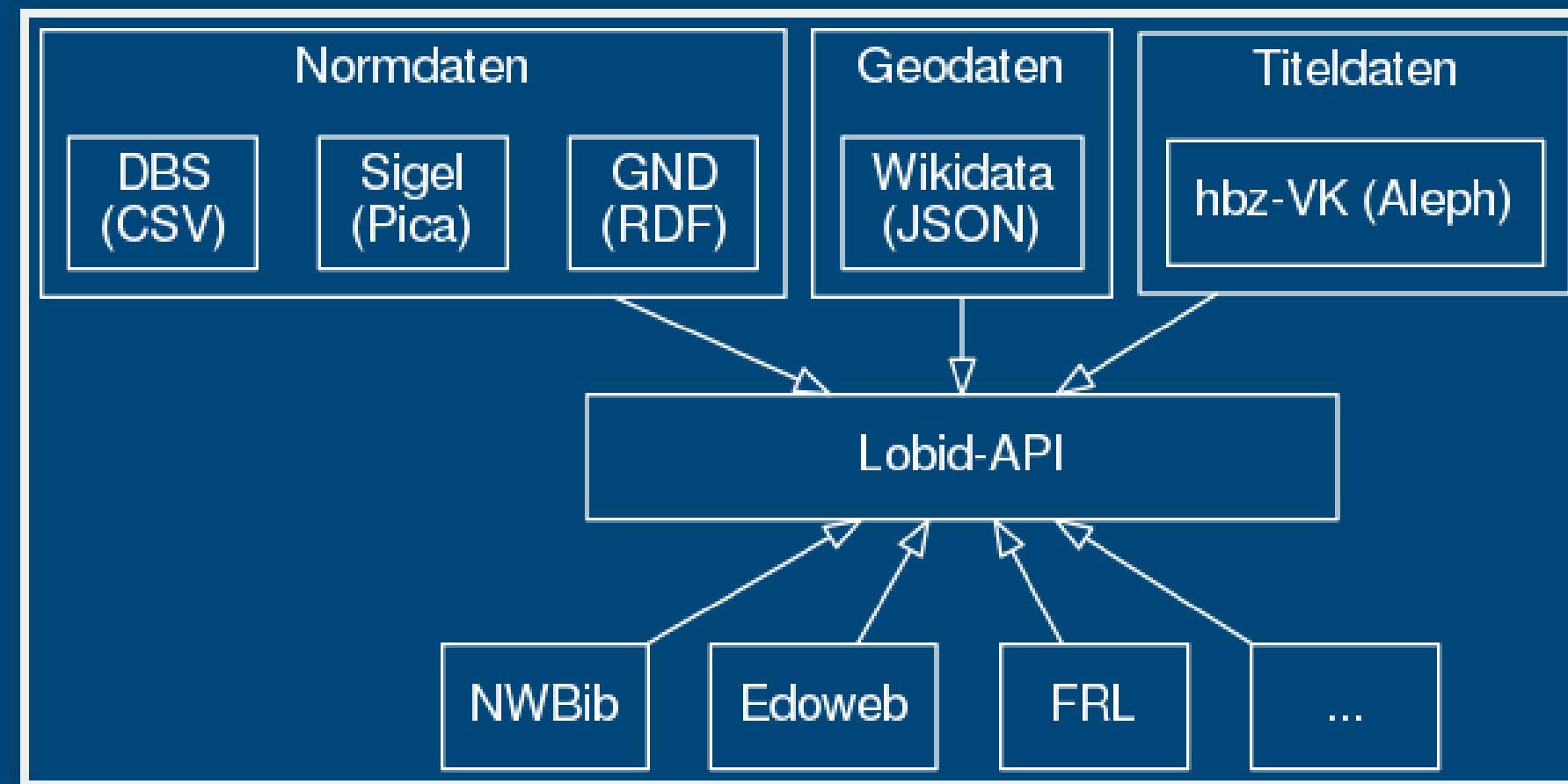
Metafactory is an open framework. (You can use modules on their own. It's Open Source so you can develop your own modules.)

We presume you are here because you want to expose your data in a useful way. Like we at lobid:

We presume you are here because you want to expose your data in a useful way. Like we at lobid:



We presume you are here because you want to expose your data in a useful way. Like we at lobid:



LOUD Web APIs!

The Workshop: Why are we doing what we are doing? 2/2

The Workshop: Why are we doing what we are doing? 2/2

In our Workshop we want to introduce you to the tool Metafactory and show how you can transform MARC-XML to JSON-LD.

The Workshop: Why are we doing what we are doing? 2/2

In our Workshop we want to introduce you to the tool Metafacture and show how you can transform MARC-XML to JSON-LD.

We are using the new Metafacture Playground and rely on Live Coding.

The Workshop: Why are we doing what we are doing? 2/2

In our Workshop we want to introduce you to the tool Metafactory and show how you can transform MARC-XML to JSON-LD.

We are using the new Metafactory Playground and rely on Live Coding.

Especially the DSL FLUX and FIX do not need any special Java knowledge, so that open-minded librarians can be included in the development and maintenance of an ETL process.

Part II: Getting to know MF and FLUX

Instead of a formal
introduction to MF-FLUX lets
get straight to the Playground

Exercise 1: first steps: how to use the Playground

Exercise 2: how to open a file and read the data from it

Exercise 3: interpret a process

How does MF work?

How does MF work?

Data runs through multiple modules

How does MF work?

Data runs through multiple modules

roughly like this:

→ read → decode → transform → encode → write →

How does MF work?

Data runs through multiple modules

roughly like this:

→ read → decode → transform → encode → write →

Each module expects certain input and creates specific output ("signature") => the output of a certain module has to match the input of the following module

How does MF work?

Data runs through multiple modules

roughly like this:

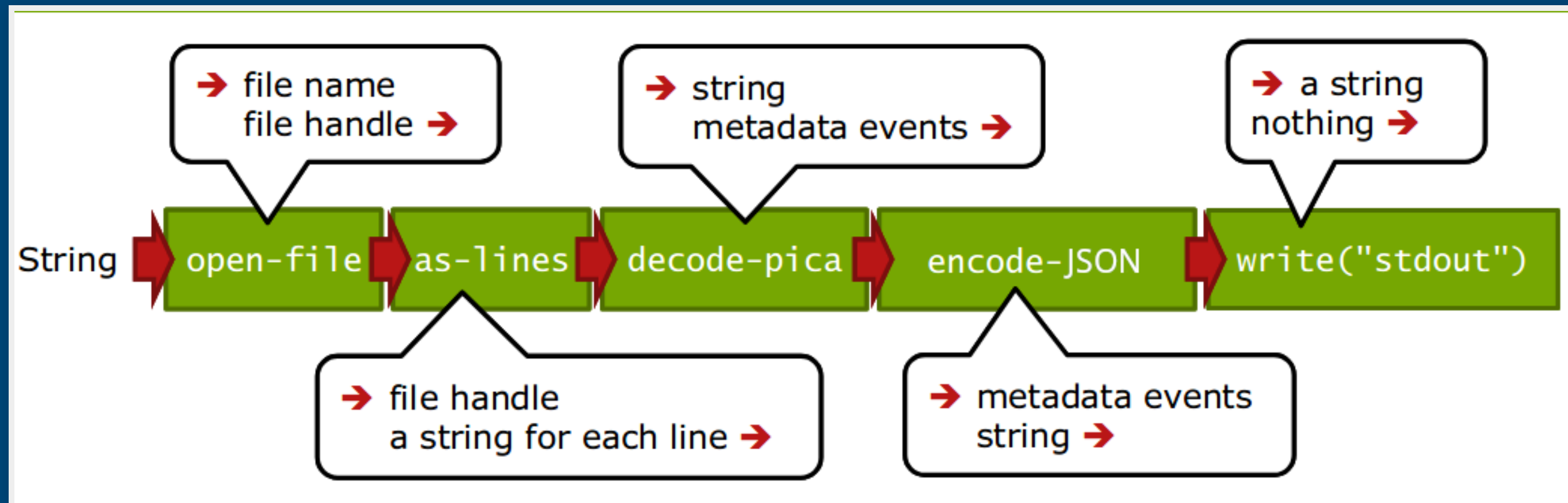
→ read → decode → transform → encode → write →

Each module expects certain input and creates specific output ("signature") => the output of a certain modul has to match the input of the following modul

Through the combination of different modules we create a MF workflow

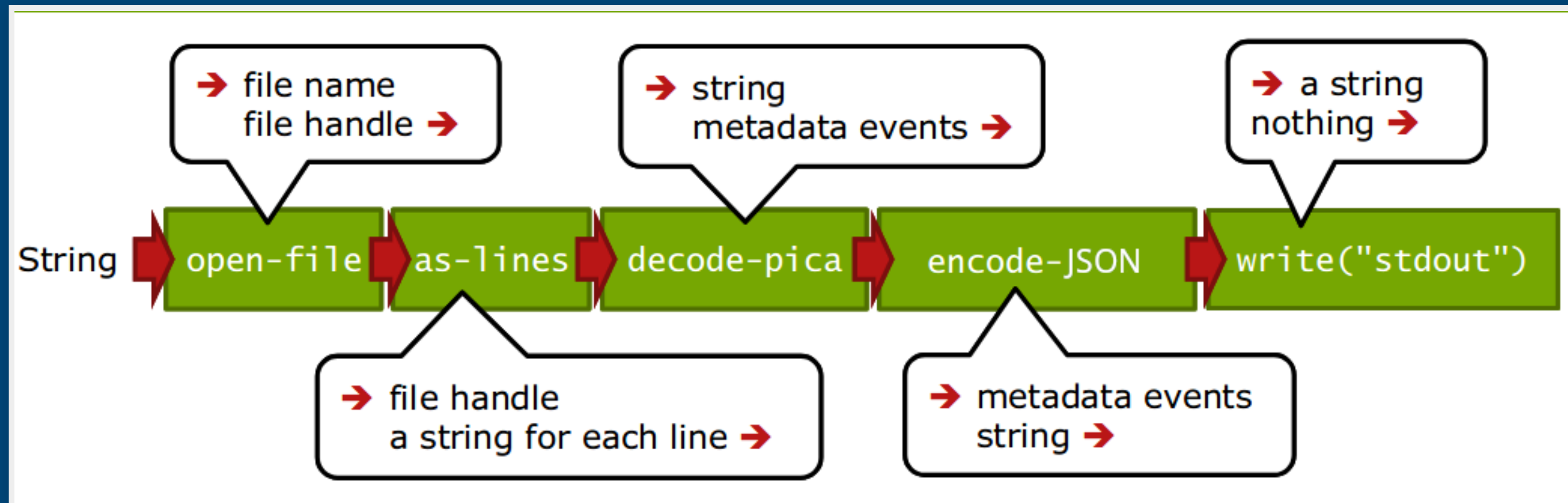
Example of a MF Workflow

Example of a MF Workflow



(from Christoph Böhme, http://swib.org/swib13/slides/boehme_swib13_131.pdf)

Example of a MF Workflow

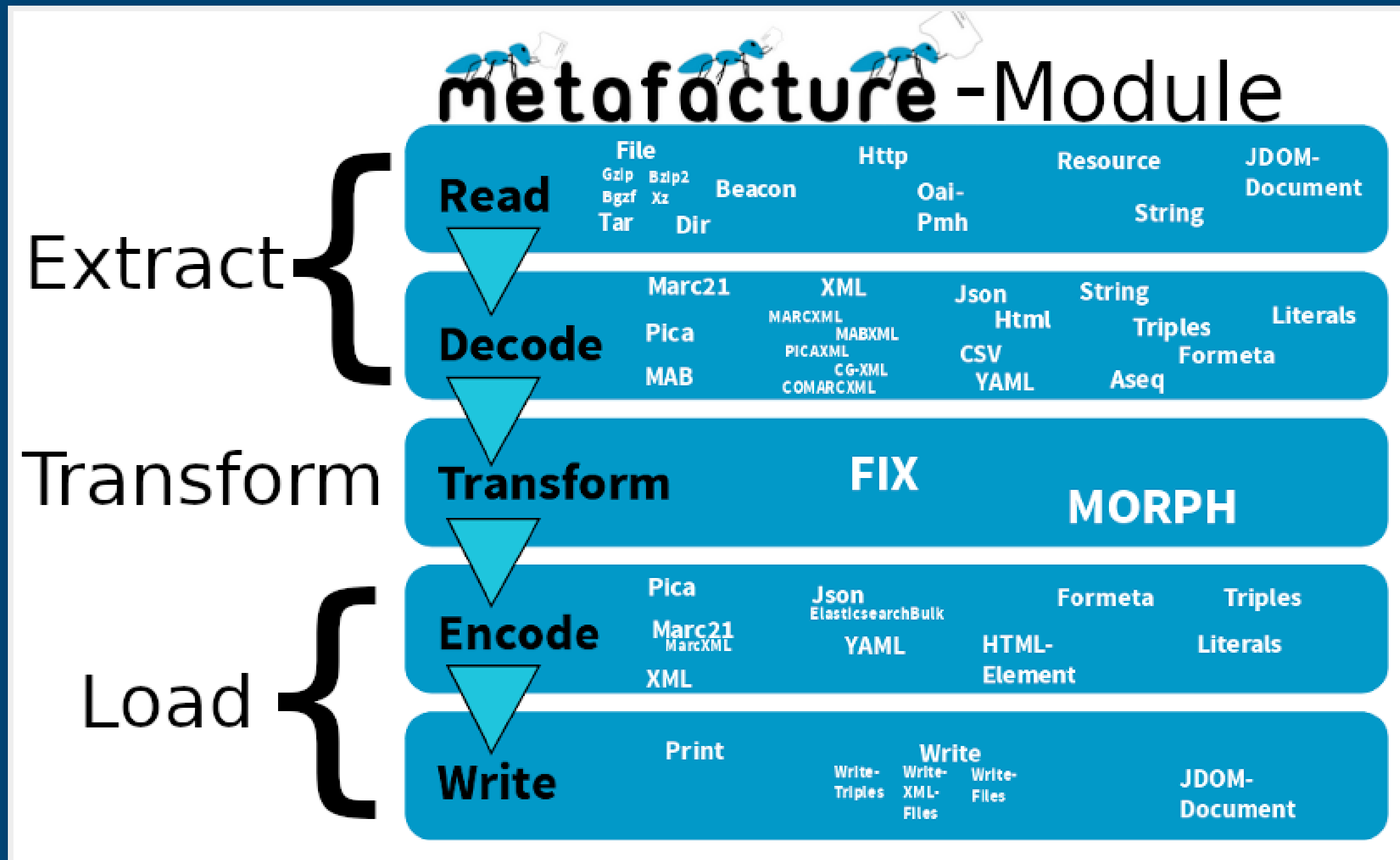


(from Christoph Böhme, http://swib.org/swib13/slides/boehme_swib13_131.pdf)

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

(example of a "flux" file)

Metafactory-Module



MF workflows

Workflows can be configured by using:

MF workflows

Workflows can be configured by using:

1. **FLUX**(the scripting language we already used in the Playground)
2. **Java by building a Flow**(typesafe as Java Generics)

MF workflows

Workflows can be configured by using:

1. **FLUX**(the scripting language we already used in the Playground)
2. **Java by building a Flow**(typesafe as Java Generics)

Workflows can be executed:

1. **with the CLI**
2. **invoking the FLUX from source code**
3. **starting the Flow**

MF workflows

Workflows can be configured by using:

1. **FLUX**(the scripting language we already used in the Playground)
2. **Java by building a Flow**(typesafe as Java Generics)

Workflows can be executed:

1. **with the CLI**
2. **invoking the FLUX from source code**
3. **starting the Flow**

For getting to know MF you also can use the Playground.

MF Modules

MF Modules

which Modules exists?

MF Modules

which Modules exists?

what do they do?

MF Modules

which Modules exists?

what do they do?

what options do they have?

MF Modules

which Modules exists?

what do they do?

what options do they have?

what is the signature (i.e. expected input and created output)

MF Modules

which Modules exists?

what do they do?

what options do they have?

what is the signature (i.e. expected input and created output)

what is the name of the Java class?

MF Modules

which Modules exists?

what do they do?

what options do they have?

what is the signature (i.e. expected input and created output)

what is the name of the Java class?

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

Exercise 4: Pica to YAML

Exercise 5: Collect MARC-XML from the WEB and transform them to JSON

15m break ☕

Part III: More practice and FIX

Part III: More practice and FIX

Transforming Metadata from one format to another is not enough

Part III: More practice and FIX

Transforming Metadata from one format to another is not enough

Moving MARC to JSON and keeping the structure is NO linked data

Part III: More practice and FIX

Transforming Metadata from one format to another is not enough

Moving MARC to JSON and keeping the structure is NO linked data

We also have to transform elements, structure and values by manipulating the records and elements

Part III: More practice and FIX

Transforming Metadata from one format to another is not enough

Moving MARC to JSON and keeping the structure is NO linked data

We also have to transform elements, structure and values by manipulating the records and elements

e.g. add fields, lookup values, clean data etc.

Part III: More practice and FIX

Transforming Metadata from one format to another is not enough

Moving MARC to JSON and keeping the structure is NO linked data

We also have to transform elements, structure and values by manipulating the records and elements

e.g. add fields, lookup values, clean data etc.

For that we need FIX - the transformation module of Metafactory (in contrast to FLUX - which configures the broader workflow)

Part III: More practice and FIX

Transforming Metadata from one format to another is not enough

Moving MARC to JSON and keeping the structure is NO linked data

We also have to transform elements, structure and values by manipulating the records and elements

e.g. add fields, lookup values, clean data etc.

For that we need FIX - the transformation module of Metafactory (in contrast to FLUX - which configures the broader workflow)

FIX helps us to manipulate the data on the record level

How to FIX

How to FIX

Before we start creating real LOUD metadata lets play around with FIX

How to FIX

Before we start creating real LOUD metadata lets play around with FIX

Exercise 6: **Manipulating text with FIX**

What is FIX

What is FIX

simple scripting language

What is FIX

simple scripting language

inspired by **Catmandu** Fix

What is FIX

simple scripting language

inspired by **Catmandu** Fix

it allows you to to manipulate records and metadata elements, e.g.:

What is FIX

simple scripting language

inspired by **Catmandu** Fix

it allows you to to manipulate records and metadata elements, e.g.:

1. fields can be changed, removed, filterd, separated or added

What is FIX

simple scripting language

inspired by **Catmandu** Fix

it allows you to to manipulate records and metadata elements, e.g.:

1. fields can be changed, removed, filterd, separated or added
2. structure and hierachy can be changed

What is FIX

simple scripting language

inspired by **Catmandu** Fix

it allows you to to manipulate records and metadata elements, e.g.:

1. fields can be changed, removed, filtered, separated or added
2. structure and hierarchy can be changed
3. values can be looked up in external files

FIX concepts

```
# Simple Fix functions
add_field("hello", "world")
remove_field("my.deep.nested.junk")
copy_field("stats", "output.$append")

# Conditionals:
if exists("error")
  set_field("is_valid", "no")
else
  set_field("is_valid", "yes")
end

# Binds - Loops:
do list(path: "foo", "var": "$i")
  add_field("$i.bar", "baz")
end
```

FIX Docu

FIX User Guide
FIX Functions and Cookbook

Exercise 7:

doing some cleanup with FIX

FIX Path: how to reference fields

FIX Path: how to reference fields

If you want to manipulate specific elements or create a new element you need to know the FIX path to the element.

FIX Path: how to reference fields

If you want to manipulate specific elements or create a new element you need to know the FIX path to the element.

FIX-Path is a close relative of JSON-Path.

Excursion: Metadata structures

Excursion: Metadata structures

Metadata example:

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

Excursion: Metadata structures

Metadata example:

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

We see:

Excursion: Metadata structures

Metadata example:

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

We see:

... a record /data set (a collection of metadata of that book)

Excursion: Metadata structures

Metadata example:

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

We see:

... a record /data set (a collection of metadata of that book)

... elements/fields/properties/attributes/keys

Excursion: Metadata structures

Metadata example:

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

We see:

... a record /data set (a collection of metadata of that book)

... elements/fields/properties/attributes/keys

... values of these elements

Excursion: Metadata structures

Excursion: Metadata structures

The structure is often not just flat "key:value"

Excursion: Metadata structures

The structure is often not just flat "key:value"

Subfields and Objects

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

Excursion: Metadata structures

The structure is often not just flat "key:value"

Subfields and Objects

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

Attributes in context of XML

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

Excursion: Metadata structures

The structure is often not just flat "key:value"

Subfields and Objects

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

Attributes in context of XML

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

Array/List (of strings or objects)

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

Selecting elements with FIX path

Selecting elements with FIX path

Metadata example:

```
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
```


Selecting elements with FIX path

Metadata example:

```
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
```

Select simple top level element: **id**

Selecting elements with FIX path

Metadata example:

```
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
```

Select simple top level element: **id**

sublevel element: **title.subTitle** (separate with **.**)

Selecting elements with FIX path

Metadata example:

```
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
```

Select simple top level element: **id**

sublevel element: **title.subTitle** (separate with **.**)

repeated elements/arrays: **creator.2.firstName** (using index number)

FIX Path: selecting specialities

FIX Path: selecting specialities

JSON arrays have a array marker `[]`, due to the JSON encoder/decoder, e.g. `creator[]`

FIX Path: selecting specialities

JSON arrays have a array marker `[]`, due to the JSON encoder/decoder, e.g. `creator[]`

Path wildcards: `?` and `*` in element names

FIX Path: selecting specialities

JSON arrays have a array marker `[]`, due to the JSON encoder/decoder, e.g. `creator[]`

Path wildcards: `?` and `*` in element names

Array-Wildcards:

FIX Path: selecting specialities

JSON arrays have a array marker `[]`, due to the JSON encoder/decoder, e.g. `creator[]`

Path wildcards: `?` and `*` in element names

Array-Wildcards:

`*` (all instances)

FIX Path: selecting specialities

JSON arrays have a array marker `[]`, due to the JSON encoder/decoder, e.g. `creator[]`

Path wildcards: `?` and `*` in element names

Array-Wildcards:

`*` (all instances)

`$append` for a new instance

FIX Path: selecting specialities

JSON arrays have a array marker `[]`, due to the JSON encoder/decoder, e.g. `creator[]`

Path wildcards: `?` and `*` in element names

Array-Wildcards:

`*` (all instances)

`$append` for a new instance

`$last` for the last instance

Know your paths

Know your paths

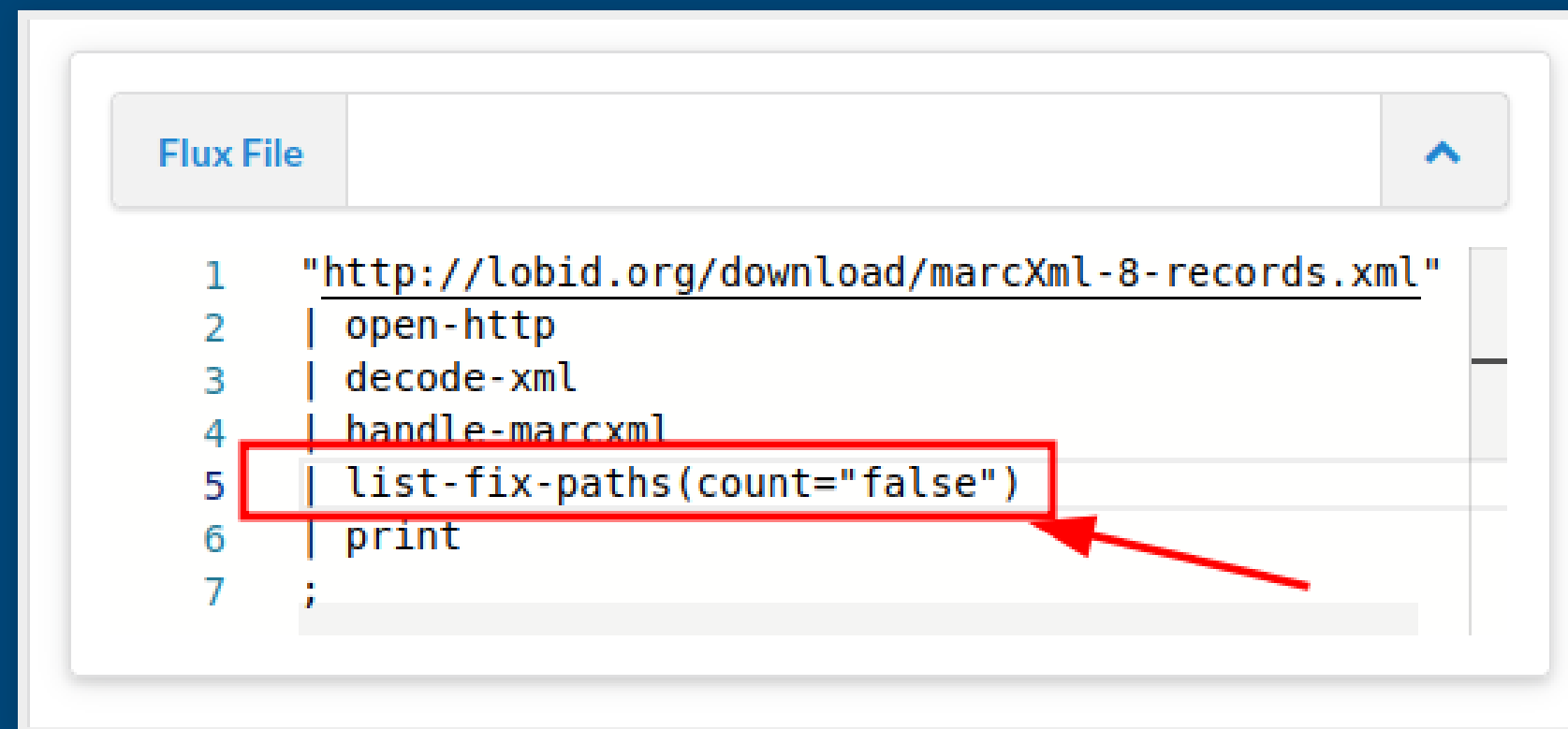
use the flux command **list-fix-paths**

Know your paths

use the flux command **list-fix-paths**
(it's useful to get an overview of your paths)

Know your paths

use the flux command **list-fix-paths**
(it's useful to get an overview of your paths)



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

link

Selecting an element

Selecting an element

Discard any data but keep all title information from element

245:

Selecting an element

Discard any data but keep all title information from element

245:

use "any indicator" => **245??**

Selecting an element

Discard any data but keep all title information from element

245:

use "any indicator" => **245??**

plus use **retain** to discard all other data => **retain('245??')**

Selecting an element

Discard any data but keep all title information from element

245:

use "any indicator" => **245??**

plus use **retain** to discard all other data => **retain('245??')**

```
Flux File ^
1  "http://lobid.org/download/marcXml-8-records.xml"
2  | open-http
3  | decode-xml
4  | handle-marcxml
5  | fix("retain('245??')")
6  | encode-json
7  | print
8  ;
```

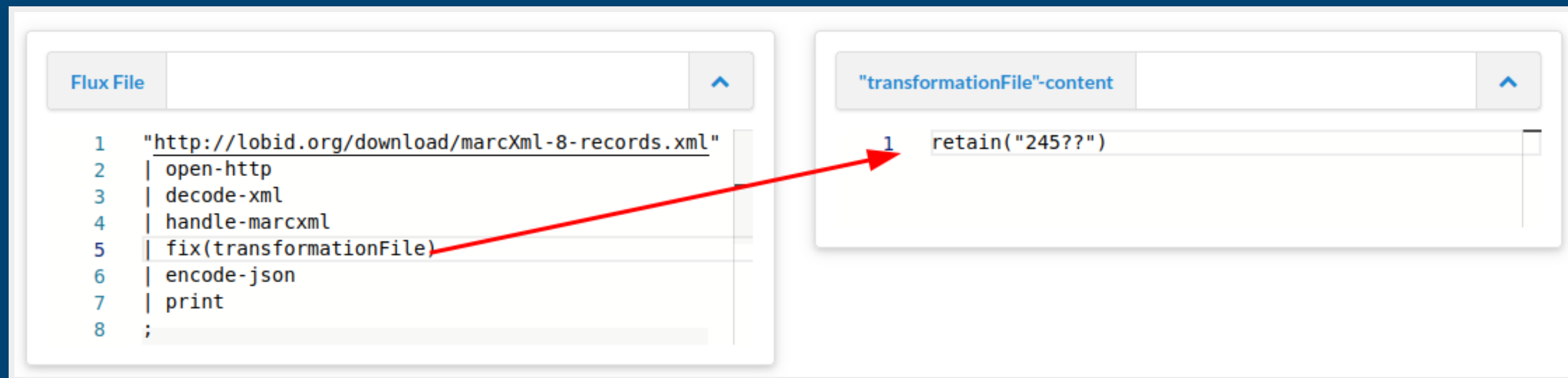
link

Outsource the FIX

FIX may get really big. You can separate it from the FLUX.

Outsource the FIX

FIX may get really big. You can separate it from the FLUX.



The image shows two panels from a Flux editor. The left panel, titled 'Flux File', contains a list of steps:

```
1 "http://lobid.org/download/marcXml-8-records.xml"  
2 | open-http  
3 | decode-xml  
4 | handle-marcxml  
5 | fix(transformationFile)  
6 | encode-json  
7 | print  
8 ;
```

The right panel, titled '"transformationFile"-content', contains a single step:

```
1 retain("245??")
```

A red arrow points from the 'fix(transformationFile)' step in the left panel to the '1 retain("245??")' step in the right panel, indicating a link between the two.

link

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, '~from', 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, '~from', 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, '~from', c)
```

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

[example in Playground](#)

Practice!

Practice!

Let's transform MARC-XML in simple steps

Pica example

Pica example

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 0utf8001X 00002@ 0Aa003@ 0958090564
2 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U 0utf8001X 00002@ 0Aa1003@ 036180954
3 001@ 02a5001A 01200:24-11-77001B 09999:16-01-08t01:01:54.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aan003@ 078051014
4 001@ a501-2001A 01140:28-06-99001B 09999:17-04-11t14:40:50.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 09567459
5 001@ 01a5001A 09999:08-03-02001B 09999:12-03-04t11:47:27.000001D 09999:08-03-02001U 0utf8001X 00002@ 0Aa1003@ 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"
}
```

Lookups using external tables

Lookups using external tables

Map bibliographic level from MARC leader to human readable representation by doing a lookup on a tabulator separated value file (tsv) residing at the web

RDF web lookups

RDF web lookups

In Alma Marc XML we get only the identifier of "DNB Sachgruppen":

```
<datafield tag="084" ind1=" " ind2=" ">  
  <subfield code="a">340</subfield>  
  <subfield code="q">DE-101</subfield>  
  <subfield code="2">sdnb</subfield>  
</datafield>
```

Lookup "DNB Sachgruppen" to enrich with skos:notation and skos:prefLabel

Usage of wildcards and binds

Usage of wildcards and binds

Collect all contributors from MARC21 and map those to
dc:contributor

Analysis of element values

Analysis of element values

with flux command: **list-fix-values**

That's it

That's it

This was our crash course "Metafactory". In Part IV we will create LOUD JSON-LD records with Metafactory. We only showed you a part of the metafactory tool box.

That's it

This was our crash course "Metafactory". In Part IV we will create LOUD JSON-LD records with Metafactory. We only showed you a part of the metafactory tool box.

We did not talk about:

That's it

This was our crash course "Metafactory". In Part IV we will create LOUD JSON-LD records with Metafactory. We only showed you a part of the metafactory tool box.

We did not talk about:

very complex transformations

That's it

This was our crash course "Metafactory". In Part IV we will create LOUD JSON-LD records with Metafactory. We only showed you a part of the metafactory tool box.

We did not talk about:

very complex transformations

complex analytics., e.g. counting patterns

That's it

This was our crash course "Metafactory". In Part IV we will create LOUD JSON-LD records with Metafactory. We only showed you a part of the metafactory tool box.

We did not talk about:

very complex transformations

complex analytics., e.g. counting patterns

merging different records from different sources with

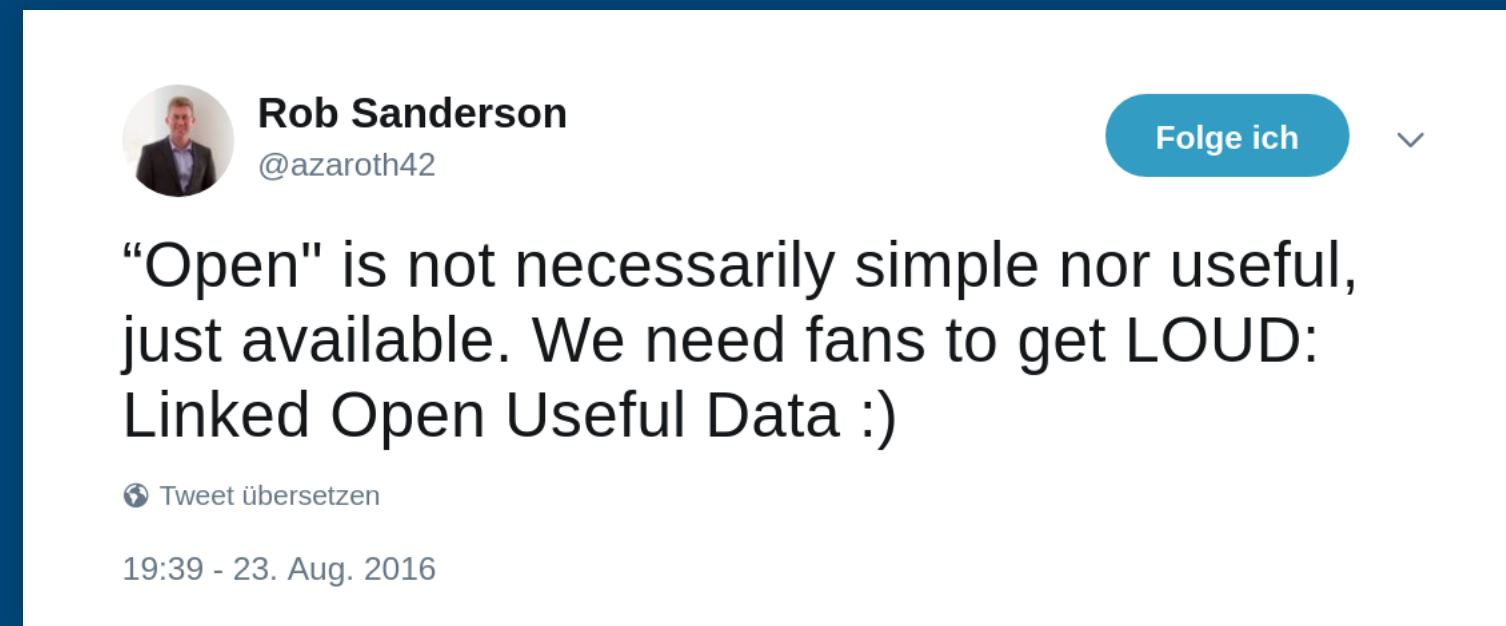
collect-triples

30m break

Part IV (2h with a break)

LOUD introduction

What is this LOUD?



Source: [Rob Sanderson on Twitter](#)

See also Rob Sanderson's Europeanatech 2018 Keynote ([Slides Video](#))

Not just LOD



Not just LOD

... which could look like this RDF (ntriples):



```
<http://lobid.org/resource/99371981001306441#  
<http://purl.org/dc/terms/title>  
"Smart grids and sustainable energy" .
```

Not just LOD

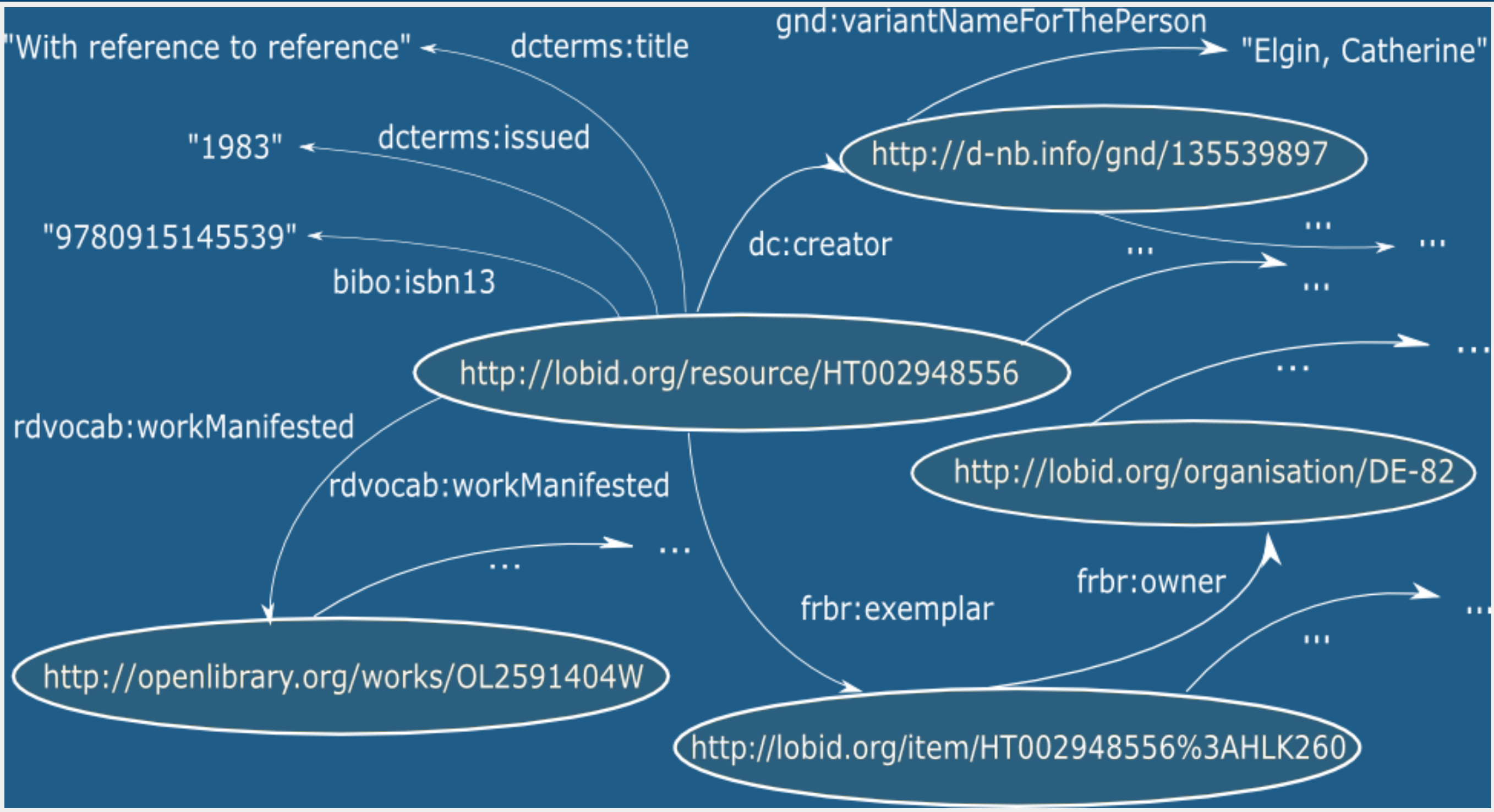
... which could look like this RDF (ntriples):



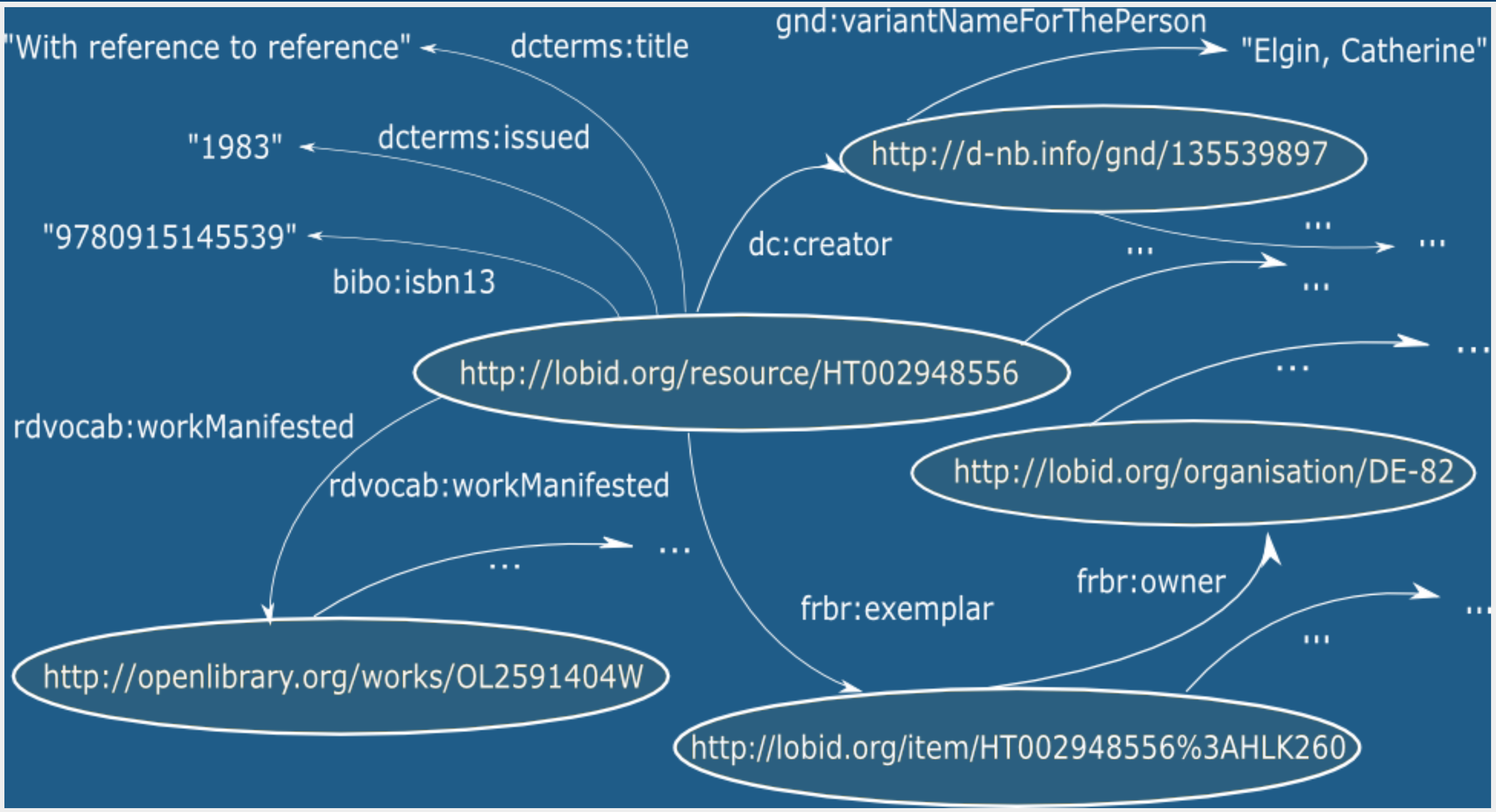
```
<http://lobid.org/resource/99371981001306441#  
<http://purl.org/dc/terms/title>  
"Smart grids and sustainable energy" .
```

Which doesn't look so bad - but "blank nodes" etc ... you would need a TripleStore.

Graphs - good for triple stores



Graphs - good for triple stores



but the most of us don't use SPARQL and triple stores ...



Ah - ok - JSON then! Trees to the rescue:

Ah - ok - JSON then! Trees to the rescue: It could look like this JSON/RDF result:

```
{ "head": { "vars": [ "s", "p", "o" ] },
  "results": {
    "bindings": [ {
      "o": {
        "type": "literal",
        "value": "Smart grids and sustainable energy"
      },
      "p": {
        "type": "uri",
        "value": "http://purl.org/dc/terms/title"
      },
      "s": {
        "type": "uri",
        "value": "http://lobid.org/resource/99371981001306441#!"
      }
    }
  ]
}
```

(example of an unLOUD JSON representation of LOD - a "flat" tree)

Turning it LOUDer: What you see is what you can query

```
"contribution": [  
  {  
    "agent": {  
      "gndIdentifier": "135539897",  
      "id": "https://d-nb.info/gnd/135539897",  
      "label": "Elgin, Catherine Z.",  
      "type": [  
        "Person"  
      ],  
      "dateOfBirth": "1948",  
      "altLabel": [  
        "Elgin, Catherine"  
      ]  
    },  
    "role": {  
      "id": "http://id.loc.gov/vocabulary/relators/aut",  
      "label": "Autor/in"  
    },  
    "type": [  
      "Contribution"  
    ]  
  }  
],
```


Turning it LOUDer: What you see is what you can query

```
"contribution": [  
  {  
    "agent": {  
      "gndIdentifier": "135539897",  
      "id": "https://d-nb.info/gnd/135539897",  
      "label": "Elgin, Catherine Z.",  
      "type": [  
        "Person"  
      ],  
      "dateOfBirth": "1948",  
      "altLabel": [  
        "Elgin, Catherine"  
      ]  
    },  
    "role": {  
      "id": "http://id.loc.gov/vocabulary/relators/aut",  
      "label": "Autor/in"  
    },  
    "type": [  
      "Contribution"  
    ]  
  }  
],
```

(Query example: `contribution.agent.label:"Elgin, Catherine" AND NOT contribution.role.id:"http://id.loc.gov/vocabulary/relators/aut"`)

Turning it LOUDer: What you see is what you can query

```
"contribution": [  
  {  
    "agent": {  
      "gndIdentifier": "135539897",  
      "id": "https://d-nb.info/gnd/135539897",  
      "label": "Elgin, Catherine Z.",  
      "type": [  
        "Person"  
      ],  
      "dateOfBirth": "1948",  
      "altLabel": [  
        "Elgin, Catherine"  
      ]  
    },  
    "role": {  
      "id": "http://id.loc.gov/vocabulary/relators/aut",  
      "label": "Autor/in"  
    },  
    "type": [  
      "Contribution"  
    ]  
  }  
],
```

(Query example: `contribution.agent.label:"Elgin, Catherine" AND NOT contribution.role.id:"http://id.loc.gov/vocabulary/relators/aut"`)

But wait - this is not LOD !

Machine readable semantics are missing! Let's add that with a "context":

Machine readable semantics are missing! Let's add that with a "context":

```
{
  "@context": "http://lobid.org/resources/context.jsonld",
  "contribution": [
    {
      "agent": {
        "gndIdentifier": "135539897",
        "id": "https://d-nb.info/gnd/135539897",
        "label": "Elgin, Catherine Z.",
        [...]
      }
    }
  ]
}
```

Machine readable semantics are missing! Let's add that with a "context":

```
{
  "@context": "http://lobid.org/resources/context.jsonld",
  "contribution": [
    {
      "agent": {
        "gndIdentifier": "135539897",
        "id": "https://d-nb.info/gnd/135539897",
        "label": "Elgin, Catherine Z.",
        [...]
      }
    }
  ]
}
```

JSON + context => JSON-LD ("context" here not embedded but as a reference)

JSON-LD

JSON-LD

"designed to be usable directly as JSON, with no knowledge of RDF" — it's real JSON!

JSON-LD

"designed to be usable directly as JSON, with no knowledge of RDF" — it's real JSON!

"also designed to be usable as RDF"

JSON-LD

"designed to be usable directly as JSON, with no knowledge of RDF" — it's real JSON!

"also designed to be usable as RDF"

<https://www.w3.org/TR/json-ld/>

Let's have a look into the referenced "context" at
<http://lobid.org/resources/context.jsonld>:

Let's have a look into the referenced "context" at
<http://lobid.org/resources/context.jsonld>:

```
{
  "@context": {
    "contribution": {
      "@id":
"http://id.loc.gov/ontologies/bibframe/contribution",
      "@container": "@list"
    },
    "agent": {
      "@id": "http://id.loc.gov/ontologies/bibframe/agent"
    },
    "label": {
      "@id": "http://www.w3.org/2000/01/rdf-schema#label"
    },
    [...]
  }
}
```

Five Stars of Linked Open Usable Data

- ☆ right Abstraction for the audience
- ☆ few Barriers to entry
- ☆ Comprehensible by introspection
- ☆ Documentation with working examples
- ☆ few Exceptions, many consistent patterns



So let's do LOUD now!

So let's do LOUD now!

Main thing to learn is there is not *the one* solution of the transformation. You have to map, you have to transform you have to try and test.

So let's do LOUD now!

Main thing to learn is there is not *the one* solution of the transformation. You have to map, you have to transform you have to try and test.

Iterate.

So let's do LOUD now!

Main thing to learn is there is not *the one* solution of the transformation. You have to map, you have to transform you have to try and test.

Iterate.

Improve step by step.

So let's do LOUD now!

Main thing to learn is there is not *the one* solution of the transformation. You have to map, you have to transform you have to try and test.

Iterate.

Improve step by step.

This is true for all work.

So let's do LOUD now!

Main thing to learn is there is not *the one* solution of the transformation. You have to map, you have to transform you have to try and test.

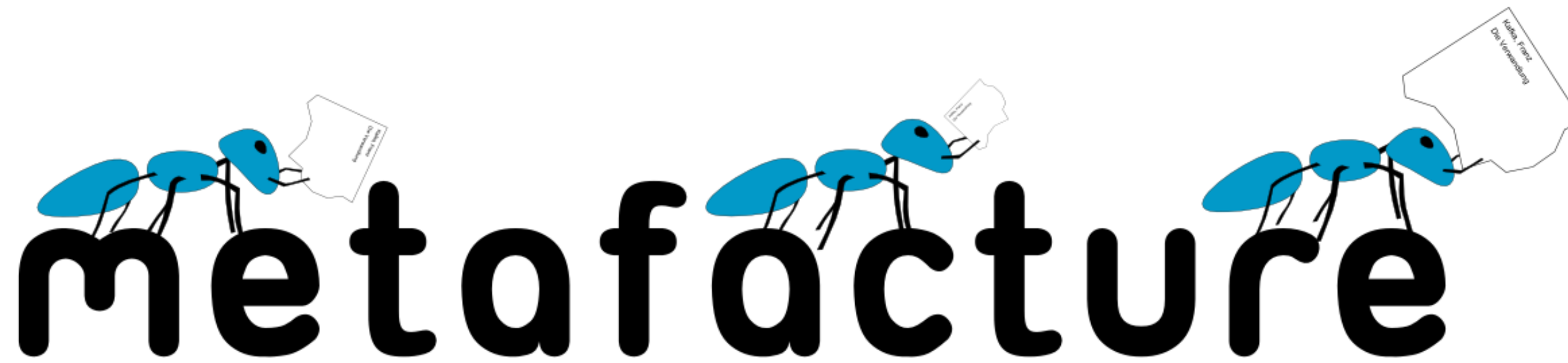
Iterate.

Improve step by step.

This is true for all work.

Let's go to

<https://pad.lobid.org/g4oXCsn2TbSZb5F4rSm1Nw#>



metafacture

<https://metafacture.org>