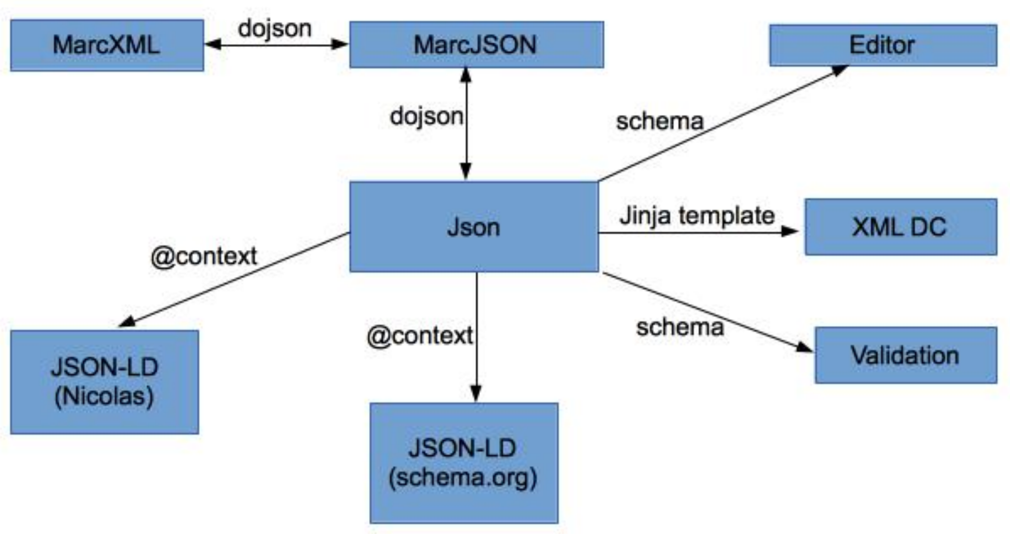
From MarcXML to JSON-LD

A first study at RERO

A new approach

- MarcXML is not **central** anymore => becomes an import/export format
- master format should be **human** readable and easy to use
- master format should be automagically validated => **JSON schema**
- an editor should be created from the cataloging rules => based on schema
- Linked data export should be generated by adding a simple context => the master format is not modified

Ideal workflow



dojson

New Decorator: to solve ISBN

- problem: 2 ISBN formats: isbn10 and isbn13 with **two corresponding ontologies** (bibo:isbn10, bibo:isbn13)
- pull request with a new decorator `@utils.ignore_value` (thanks to Jiri)
    - we do not know in advance which ISBN we have (same Marc field)
    - depends on a regular expression
    - returning `None` value removes the corresponding property => avoid `{"isbn10": null}`

JSON Editor

- JS code (https://github.com/jdorn/json-editor) + **JSON schema**
- need **specific** properties: `propertyOrder`, `watch`, etc.
- demo

JSON-LD = JSON + @context

Take a JSON, add an **header** (`context`) to generate triplets.

A good introduction: http://www.dataversity.net/smartdata-webinar-slides-json-ld/

LD Part

- **subject–predicate–object**
- **subject** entity is identified by a unique URI
- **predicate** is specified by a unique URI defining a property within an ontology
- **object** entity is identified by a unique URI or a literal (text)
- JSON-LD makes our data **understandable** for the rest of the world
- RERO LD data model by **Nicolas Prongué**

Used Ontologies

```
"@context": {
     "dcmitype": "http://purl.org/dc/dcmitype/",
     "bibo": "http://purl.org/ontology/bibo/",
     "dc": "http://purl.org/dc/elements/1.1/",
     "dct": "http://purl.org/dc/terms/",
     "edm": "http://www.europeana.eu/schemas/edm/",
     "foaf": "http://xmlns.com/foaf/0.1/",
     "rdau": "http://rdaregistry.info/Elements/u/",
     "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns# ",
     "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
},
"dc:title": "mytitle"
```

Alias

```
"@context": {
    "dc": "http://purl.org/dc/elements/1.1/",
    "title": "dc:title"
},
"title": "mytitle"
```

Record ID

```
"@context": {
    "@base": "http://doc.rero.ch/record/",
    "recid": "@id"
 },
 "recid": "1234"
```

Type

```
  "@context": {
     "dct": "http://purl.org/dc/terms/",
     "dcmitype": "http://purl.org/dc/dcmitype/",
     "type": "@type",
     "book": "dcmitype:Text",
     "bibrec": "dct:BibliographicResource",
  },
  "type": ["book", "bibrec"]
```

RERO ID as URI

```
  "@context": {
      "rero_id": {
         "@id": "dct:hasFormat",
         "@type": "@id"
      }
 },
  "rero_id": "http://data.rero.ch/01-R84732"
```

Literals

```
  "@context": {
     "dc": "http://purl.org/dc/elements/1.1/",
     "title": "dc:title",
     "full": "@value",
     "lang": "@language",
  },
  "title": {
      "main": "My Main Title",
      "sub": "My Subtitle",
      "full": "My Main Title: My Subtitle",
      "lang": "eng"
  }
```

Media Type an URI

```
 "media_type": {
     "@id": "rdau:mediaType",
     "@type": "@id"
 }
```

Invenio > 2.2 Preparation

- write mappings rules for **all** document types
- write **tests** for all rules using `pytest` for JSON validation
- check MarcXML -> JSON -> MarcXML for **all** RERO records
- configure the editor with **autocompletion** (ORCID, UDC, etc.)
- liked data **validation** by hand
- check **Elasticsearch** compatibility (mapping + anaysis)
- write an other `@context` for **schema.org**
- do all the stuff in **parallel** to have a nice data model