bibliotek-o : an overview

bibliotek-o defined

bibliotek-o is a framework and an ontology for modeling bibliographic metadata as linked data. Reusing BIBFRAME as its core, bibliotek-o is not implementable without BIBFRAME; however, bibliotek-o reuses properties and classes from already established ontologies, in addition to minting its own terms where the developers determined a lack of adequate terms for reuse. In addition to reusing individual properties and classes, bibliotek-o extends and offers alternative modeling for discrete areas of bibliographic description; these models are discussed in the "bibliotek-o modeling patterns" section below.

bibliotek-o is a joint effort of the Andrew W. Mellon Foundation funded Linked Data for Libraries Labs (LD4L Labs) and Linked Data for Production (LD4P) projects; this work represents significant effort by a large group of colleagues from Columbia, Cornell, Harvard, Princeton and Stanford universities as well as the Library of Congress (hereafter “Ontology Group”). LD4L Labs and LD4P are complementary efforts in support of tool development, RDF metadata production, community engagement and ontology development in the library realm.

bibliotek-o's motivation

The primary motivation behind bibliotek-o's development is to demonstrate potential modeling improvements to BIBFRAME for community consideration. Through publication of the ontology in the bibliotek-o GitHub repository and implementation of the framework in a MARC-2-RDF converter and RDF instance editor, we aim to facilitate comparisons between BIBFRAME and bibliotek-o modeling.

Importantly, bibliotek-o is not designed as a competitor to BIBFRAME; instead, we believe that BIBFRAME development could benefit from further community engagement. By demonstrating extended and alternative patterns, we aim to offer potential solutions to areas the Ontology Group believes that could benefit BIBFRAME modeling. As the community decides on model implementations, we aim to deprecate bibliotek-o patterns.

We firmly believe that decisions as important as the widespread adoption of bibliographic data models should be community-driven based on extensive consideration and experimentation. bibliotek-o represents LD4's experimentation in this space.

bibliotek-o ontology modeling patterns : comparisons with BIBFRAME

Activities

BIBFRAME makes a distinction between Provision and Contribution activities through distinct class assertions, the former applying to Instances and the latter to Works. While the Ontology Group understands the reasoning behind this distinction, we believe that because bf:Works, bf:Instance and bf:Item are not disjoint and an activity subclass could potentially apply to more than one resource type, the distinction between Instance- and Work-related activities is unnecessary and overly complicates query patterns.

bibliotek-o defines a general Activity pattern that facilitates explicit roles through subclassing of the bib:Activity class, which links Agents to BIBFRAME core classes (Works, Instances and Items). This pattern eliminates the distinction between provisions and contributions, thus simplifying both querying and potential extension for additional relationships between agents and bibliographic resources. This design pattern is adopted in other ontologies, such as the Schema.org Action class and the CIDOC CRM Activity class.

Gains: Simplified/flexible model, more closely aligned with Activity ontology design patterns. Straightforward extension to Item-related activities, such as those modeled in the rare materials and art ontology extensions.

Example: Without the bf:Contribution/Provision divide, a bf:Work, bf:Instance, and/or bf:Item can have a bib:SculptorActivity without questioning whether the Activity is a bf:Contribution or bf:Provision.

For the full Activities recommendation document, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/83237322/bibliotek-o_pattern_activities_201612.pdf

Content Accessibility

In February 2017, BIBFRAME changed bf:contentAccessibility from a datatype property into an object property, following the Content Accessibility recommendation finalized by the Ontology Group in December 2016.

bibliotek-o extends this revised bf:contentAccessibility/bf:ContentAccessibility model by making a distinction between Accessibility Hazards and Accessibility Features; bib:AccessibilityHazard pertains to aspects of the resource that cause issues with accessibility (e.g., flashing lights) whereas bib:AccessibilityFeature pertains to aspects of the resource that facilitate accessibility (e.g., Braille). Further, bibliotek-o mints specific subClasses of bib:AccessibilityHazard and bib:AccessibilityFeature.This model closely aligns with the schema.org model for content accessibility and better facilitates usage of library resources than having a general class without differentiation between hazards and features; schema.org's model was not directly reusable due to the lack of classes for these concepts.

For the full content accessibility recommendation document, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/83237324/bibliotek-o_pattern_content_accessibility_201612.pdf

Content Type, Carrier Type, Media Type

BIBFRAME asserts content types, carrier types and media types by establishing bf:content/bf:Content, bf:carrier/bf:Carrier, bf:media/bf:Media patterns. This modeling creates two potential means for stating the same thing: through the above property/class patterns and also through subclassing bf:Work, bf:Instance and bf:Item. Explicitly creating multiple patterns for the same concept fails to capture semantic commonalities and complicates queries; further, it diverges from the standard linked data practice of using rdf:type to declare that a resource is a particular kind of thing.

Rather than using the BIBFRAME multi-path pattern, bibliotek-o commits to modeling content types, carrier types and media types through subclasses of Work, Instance, and Item, alongside type assertions on individual resources. This pattern can be interpreted as RDA implementation because it expresses content/carrier/media information about library resources and thus supports cataloging according to the RDA content standard.

*Gains: S*implified/flexible model, more closely aligned with established classing of objects, ability to reuse existing classes.

Example: For CDs, one could reuse the Music Ontology 's class mo:cd. The closest analogous terms in the MARC carriers or RDA carriers are skos: Concepts for "audio disc" or "computer disc".

For the full Content/Carrier/Media Type recommendation, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/83237325/bibliotek-o_pattern_content_carrier_media_201612.pdf

Notes & Annotations

BIBFRAME prefers the use of specified properties with expected domains of BIBFRAME core class (Work, Instance, Item) for many note types. Rather than relating a note directly to a BIBFRAME core class, bibliotek-o uses the Web Annotation (OA) model for many of these note types (e.g.: bf:credits, bf:custodialHistory, bf:historyOfWork, bf:natureOfContent, bf: preferredCitation , bf:summary, bf:review, bf:systemRequirements, and bf: tableOfContents).

This deviation stems from a concern about directly attributing these data to the bibliographic resource; generally, these notes types are not necessarily intrinsically related to the resource but are asserted by the cataloger. By using the OA model, one can provide a target (i.e., the resource being described) and a motivation (e.g., summarizing) for the note, meanwhile providing an intermediate node to separate the note from data intrinsically connected to the resource. Rather than simple text strings, the OA model provides richer expressivity, such as annotation bodies that are resources, along with specification of type and format; making multiple atomic but related annotations on a specific resource; and ascribing purposes and motivations to annotations.

Gains: Single pattern for notes, greater expressivity to describe the note.

For the full Notes & Annotations recommendation, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/83237329/bibliotek-o_pattern_notes_annotations_201612.pdf

Relations

Rather than reusing existing relationship properties from other ontologies to relate bibliographic entities, BIBFRAME defines its own properties.

bibliotek-o uses a combination of BIBFRAME and RDA Unconstrained (RDAU) relationship properties. As a framework with BIBFRAME as its core, bibliotek-o retains many BIBFRAME properties (e.g.: bf:relatedTo subproperties: bf:expressionOf, bf:hasExpression, bf:otherEdition, bf: referencedBy, and bf:references); however, bibliotek-o preferences RDAU properties in place of many BIBFRAME properties (e.g.: bf: hasEquivalent, bf:hasDerivative, bf:derivativeOf, bf:hasReproduction, bf:originalVersion, bf:originalVersionOf, bf:otherPhysicalFormat, bf: reproductionOf, bf:translation and bf:translationOf); further, bibliotek-o uses RDAU properties for derivative relationships (rdau:P60250, its inverse and subproperties).

The use of RDAU properties is primarily due to their more granular nature than what is afforded in BIBFRAME, as well as being a well established descriptive standard in the library community. Please note that biblotek-o does not currently address part-whole relationships, accompanying material or sequential relationships as part of the framework (as of December 2016).

Gains: Greater expressivity of relationships between bf:Works, Instances, Items; explicit reuse/alignment with RDA.

Example: Currently in BIBFRAME there is no way to say a work is "based on" (rdau: P60305).

For the full Relations recommendation, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/83237330/bibliotek-o_pattern_relations_201612.pdf

Titles

With the addition of several constructs and patterns to the BIBFRAME model, bibliotek-o introduces a more expressive and accurate title model, including the representation of a main or primary title; modeling title parts as objects rather than literals in order to express nuanced relationships among these elements; and defining a controlled vocabulary of title origins such as binding, cover, spine, and so on, rather than using free-form literals.

Gains: Consistent recording of origins, ordering of title parts, designation of preferred titles.

Example: Titles with multiple subtitles can be captured separately, in the proper order, and with a language tag; can confidently identify Instances with better control over Title origins.

For the full Titles recommendation, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/83237331/bibliotek-o_pattern_titles_201612.pdf

Legacy Literals

Broadly, bibliotek-o defines 'legacy literals' to mean any existing textual metadata being migrated into RDF that does not conform to the desired model and/or application profile, and represents data as unstructured, unparsed, and unnormalized string literals.

The authors of BIBFRAME are legitimately concerned with preserving these legacy literals, and have defined a broad range of datatype properties to do so. Given bibliotek-o's preference for structured, machine actionable data over note-like strings, it instead defines object properties and classes where appropriate, to provide semantically rich models and promote the future creation of structured metadata. In order to preserve legacy metadata, it defines a custom datatype legacySourceData to flag this data for future cleanup, thus achieveing both goals of an expressive data model and preservation of legacy literals.

Gains: The model can support use cases and ontology design best practices without being limited by how legacy metadata is formulated.

Example:

```
508##$aPhotographer, Richard Beymer.
```

This 508 MARC note could convert to the following RDF, which can later be enhanced to assert the more specific bib:PhotographerActivity type and the foaf:name cleaned up and parsed.

```
ex:activity1 a bib:Activity ;
    bib:hasAgent ex:agent1 .
ex:agent1 foaf:name "Photographer, Richard Beymer"^^http://bibliotek-o.org/datatypes/legacySourceData .
```

For the full Legacy Literal recommendation, see: https://wiki.duraspace.org/display/LD4P/bibliotek-o?preview=/79795231/87468650/bibliotek-o_pattern_legacy_literals_201612.pdf

bibliotek-o framework : ontologies used

bibliotek-o uses properties and classes from the following ontologies:

- BIBFRAME (core ontology)
- bibliotek-o
- CIDOC-CRM
- DCMI Terms
- FOAF
- Lingvo
- Prov-o
- RDAU
- Schema
- SKOS
- VIVO
- Web Annotation

For a full list of terms used from these ontologies, click "Ontology Modules" on the top navigation bar of: https://bibliotek-o.org/overview/overview.html (note : cannot link directly).