Data Modeling 101

Getting Started with Data Modeling in GLAM

bit.ly/C4LDataModeling101
Your Fearless Facilitators

Hector Correa
Software developer, Brown University Library, @hectorjcorrea

Christina Harlow
Works with metadata somewhere in the world, @cm_harlow

Mark Matienzo
Collaboration & Interoperability Architect, Stanford @anarchivist

Steve Van Tuyl
Digital Repository Librarian at Oregon State, @badgerbouse

bit.ly/C4LDataModeling101
bit.ly/C4LDataModeling101

Link to Slides, Notes, Examples, Resources, and Other Workshop Materials
Communication Channels

- Alert a facilitator if you need help or have questions
- Feedback sticky notes:
  - Red sticky note = have question or need help
  - Sticky note feedback before break
- Code4Lib Slack: #c4l17-datamodeling channel
- Information about Code4Lib Slack: http://goo.gl/forms/p9Ayz93DgG
## Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00-9:15</td>
<td>Introduction (Mark Matienzo)</td>
</tr>
<tr>
<td>9:15-9:45</td>
<td>Introduction to Data Modeling (Hector Correa)</td>
</tr>
<tr>
<td>9:45-10:15</td>
<td>Data Modeling in RDF (Christina Harlow)</td>
</tr>
<tr>
<td>10:15-10:30</td>
<td>Break</td>
</tr>
<tr>
<td>10:30-11:00</td>
<td>Models in context (Steve Van Tuyl/Christina Harlow)</td>
</tr>
<tr>
<td>11:00-11:15</td>
<td>Full group: setup for breakout groups</td>
</tr>
<tr>
<td>11:15-11:45</td>
<td>Breakout groups: Data Modeling Examples</td>
</tr>
<tr>
<td>11:45-12:00</td>
<td>Full group: recap from breakouts and follow-up</td>
</tr>
</tbody>
</table>

bit.ly/C4LDataModeling101
Our Expectations of You

- Follow the Code4Lib Code of Conduct
- Follow the Recurse Center Social Rules (a.k.a. "Hacker School Rules")
- Be ready to learn about data modeling!

bit.ly/C4LDataModeling101
Code4Lib
Code of Conduct

http://2017.code4lib.org/conduct/

bit.ly/C4LDataModeling101
Recurse Center Social Rules
(a.k.a. Hacker School Rules)

https://www.recurse.com/manual#sub-sec-social-rules

- No feigning surprise
- No well-actually's
- No back-seat driving
- No subtle -isms
  - More info: https://www.recurse.com/blog/38-subtle-isms-at-hacker-school

bit.ly/C4LDataModeling101
Are you ready to data model?

bit.ly/C4LDataModeling101
Reminder

This is an informal workshop – ask questions and let facilitators know how we can help you.
This workshop is an attempt to

- Help build data modeling capacity in the library technology communities
- Help build best practices for data models

bit.ly/C4LDataModeling101
Our goals for this workshop

- Share motivations for data modeling as part of the application development process
- Equip you with knowledge needed to instigate modeling work at your institutions and participate in broader community discussions
- Demonstrate modeling practices and pitfalls
- Give context for data modeling, standards, and interoperability work in various LibTech communities

bit.ly/C4LDataModeling101
However, this workshop is not...

- A linked data or RDF workshop
- A metadata standards workshop
- A workshop on specific models

... but options exist for learning more about these topics!

bit.ly/C4LDataModeling101
Now: your goals for this workshop?

- Why are you attending this workshop?
- What are your goals - immediate or long-term?
- What's your level of comfort and experience with data modeling?

We want to capture your goals, return to them throughout the workshop & going forward - feel free to add to responses to the shared notes.

bit.ly/C4LDataModeling101
Introduction to Data Modeling

What is it & why do it?

bit.ly/C4LDataModeling101
What is a model?
“A model is an abstract description that hides certain details while illuminating others"

- Semantic Web for the Working Ontologist: Effective Modeling In Rdfs and Owl.

bit.ly/C4LDataModeling101
“A model is a simplification of reality”
- The Unified Modeling Language User Guide
What Does a Repository Know?

bit.ly/C4LDataModeling101

https://flic.kr/p/78aoDi CC BY-NC 2.0
class Document
  attr_accessor: name, collection_id, derivatives, creator, visibility

  def save()
    ...
  end

  def create_derivatives()
    ...
  end
end
"What Do Systems Know?"
- Tom Johnson
Why do we model?
“Models are used to organize human thought in the form of explanations”

- Semantic Web for the Working Ontologist : Effective Modeling In Rdfs and Owl.

bit.ly/C4LDataModeling101
Models...
- Help people communicate
- Explain and make predictions
- Mediate among multiple viewpoints
- Models can be textual or visual.

- Semantic Web for the Working Ontologist: Effective Modeling In Rdfs and Owl.

Models are a great way to collaborate with others*
within your team and outside your team.

*others: people or machines

bit.ly/C4LDataModeling101
Who are you modeling for?

- Models for humans versus models for machines
  - Informal (human readable, rely on context)
  - Formal (machine readable, proofs)
- Levels of abstraction
  - Abstract, Conceptual, Physical models
- Static vs Dynamic parts of the system
  - e.g. Database tables vs classes

bit.ly/C4LDataModeling101
Where is the Knowledge?

- Domain standards (MARC, BIBFRAME, …)
- Design documents
- Database & System State
  - Long term storage, in-memory representations.
- Code

bit.ly/C4LDataModeling101
Defining “Data Model”
What is a Model?

“When we want to make resources and their metadata available in a structured manner on the web, we first need to decide what characteristics of theirs are the most important to be represented. By doing so, we make an abstraction of the reality through the development of a model.”

-  Linked Data for Libraries, Archives & Museums, p. 12

bit.ly/C4LDataModeling101
Conceptual Model

Captures the concepts and relationships of the data.

The Three-Schema Approach:

External ← Conceptual → Internal

Multiple external serializations
Multiple internal schemas

bit.ly/C4LDataModeling101
Physical Model

**Data Structure**

A representation of data in concrete form.

- A serialized document;
- A database schema;
- A data structure (array, hashmap, B-tree, etc...).

bit.ly/C4LDataModeling101
CREATE TABLE `blogs` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `title` varchar(255) NOT NULL,  
    `content` text,  
    `createdOn` datetime NOT NULL,  
    `updatedOn` datetime DEFAULT NULL,  
    `postedOn` datetime DEFAULT NULL,  
    PRIMARY KEY (`id`),  
    KEY `blogs_index_title` (`title`),  
    KEY `blogs_index_postedOn` (`postedOn`)  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8
Examples of Data Models We Know

Flat File Model - MARC?

Tabular Data Model - CSV, TSV, Excel

Relational Data Model - ER, RDBMS

Hierarchical Data Model - XML/JSON

Semantic Data Model - Graphs, RDF(S)

bit.ly/C4LDataModeling101
Tabular data
Each data item is structured as a line of field values. Fields are the same for all items; a header line can indicate their name.

Relational model
Data are structured as tables, each of which has its own set of attributes. Records in one table can relate to others by referencing their key column.

Meta-markup languages
XML documents have a hierarchical structure, which gives them a tree-like appearance. Each element can have one or more children; there is exactly one root element.

RDF
Each fact about a data item is expressed as a triple, which connects a subject to an object through a precise relationship. This leads to graph-structured data that can take any shape.

“Models have a profound influence on how a problem is tackled and a solution shaped”

- The Unified Modeling Language User Guide

bit.ly/C4LDataModeling101
Examples of Data Models We Know

**Semantic Modeling**

RDF - basic data model, 3-part statements (triples): <s> <o> <p>

Graphs of triples

RDF Schema (RDFS) - language for writing ontologies (knowledge representation, hierarchies, inferencing)

Ontologies describe static knowledge. Problem Solving Method (PSM) - modeling reasoning processes (out of scope)

bit.ly/C4LDataModeling101
Examples of Data Models We Know

**Semantic Modeling**

Big emphasis in machine readable and interoperability (cooperative information systems)

Schema becomes a bigger concern.

bit.ly/C4LDataModeling101
Data Models or Ontologies

Similar in concept but with (waaaaaaay) different scope

Data Models limited to a single system/organization

Ontologies are meant to span across organizations (sometimes even domains)

Ontologies describe static knowledge
Problem-Solving Methods adds reasoning (e.g. inferencing)

bit.ly/C4LDataModeling101
Semantic Modeling
(i.e. Modeling using RDF)
Our Semantic Modeling Bias: Other Approaches

- Hierarchical
- Network (see: Programmer as Navigator, Bachmann)
- Relational
- Object Oriented

More abstractly:

- Entity-Relationship
- Object-Role
- Definitional Theory

bit.ly/C4LDataModeling101
Our Focus on RDF

“By adopting an extremely simple data model consisting of triples, data represented in Resource Description Framework (RDF) become schema-neutral. ... This allows for maximum flexibility. Any resource in the world (the subject) can have a specific relationship (the predicate) to any resource in the world (the object). There is no limit on what can be connected to what.”

- Linked Data for Libraries, Archives & Museums, p. 44
Our Semantic Modeling Bias: What Gives?

- Avoid “Database Orientation”
  - or language specific “data structure orientation”.
  - or “document model orientation”.

- Focus on Interoperability
  - Semantic Modeling = data is self-describing in a known semantics.

- Simplicity
  - model reduces to a single kind of object and binary relations.
  - “Simple” not to be confused with “easy”.

bit.ly/C4LDataModeling101
Example 1 of Extensibility
Example 2 of Extensibility

aat:1000462296-de rdf:type skos:Label .
aat:1000462296-de gvp:term "hinterglasmalerei"@de .
aat:1000462296-de gvp:qualifier "visual works"@de .
aat:1000462296-de gvp:displayOrder "13"^^xs:positiveInteger .
aat:1000462296-de gvp:termType gvp:UsedForTerm .
...

Example of (1 Approach to) Ordering
Data Model? Ontology? Namespace?
Model/Meta-Model/Ontology

Low-Level Model / Upper Ontology

A Model that can be used to describe other, higher level models.

An Ontology that presents a view of the world common to all Ontologies.

Let’s use RDFS, SKOS, PCDM as examples of this...

bit.ly/C4LDataModeling101
Objects & Relations

- One kind of “Object” (Entity, Resource)
  - Objects are distinct from their names & from their records.

- Binary Relations between objects
  - Interpreted as sentences: (person isAuthorOf book)
  - And as facts.

- Group objects into Classes (or Entity Types)
  - (person isA Person)

bit.ly/C4LDataModeling101
Types of Relations

- Instances of a Relation Class
- Domain & Range
  - Not a constraint!
- Common relation types:
  - Partitive \((\text{chapter isPartOf book}), (\text{book hasPart chapter})\)
  - Membership \((\text{book isMemberOf collection})\)
  - Materialization \((\text{book isPortrayalOf work})\)
  - Role
    - Type \((\text{person isA author})\)
    - Surrogate \((\text{author isRoleOf person}, \text{book hasAuthor author})\)

bit.ly/C4LDataModeling101
### RDF & RDFS

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntactic form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong> (a class)</td>
<td>C rdf:type rdfs:Class</td>
<td>C (a resource) is an RDF class</td>
</tr>
<tr>
<td><strong>Property</strong> (a class)</td>
<td>P rdf:type rdf:Property</td>
<td>P (a resource) is an RDF property</td>
</tr>
<tr>
<td><strong>type</strong> (a property)</td>
<td>I rdf:type C</td>
<td>I (a resource) is an instance of C (a class)</td>
</tr>
<tr>
<td><strong>subClassOf</strong> (a property)</td>
<td>C1 rdfs:subClassOf C2</td>
<td>C1 (a class) is a subclass of C2 (a class)</td>
</tr>
<tr>
<td><strong>subPropertyOf</strong> (a property)</td>
<td>P1 rdfs:subPropertyOf P2</td>
<td>P1 (a property) is a sub-property of P2 (a property)</td>
</tr>
<tr>
<td><strong>domain</strong> (a property)</td>
<td>P rdfs:domain C</td>
<td>domain of P (a property) is C (a class)</td>
</tr>
<tr>
<td><strong>range</strong> (a property)</td>
<td>P rdfs:range C</td>
<td>range of P (a property) is C (a class)</td>
</tr>
</tbody>
</table>
A set of concepts, optionally including statements about semantic relationships between those concepts.

A concept scheme may be defined to include concepts from different sources.

Thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', and other types of controlled vocabulary are all examples of concept schemes.
Vocabularies Are Perspectives

1. DC Terms (generalisms)
2. FOAF & BIBO (concretizations)
3. SKOS (nominalisms (bridged with foaf:focus))
4. FRBR (ideal forms)
5. PROV (entities, activities, agents; qualified roles)
6. OpenAnnotation (descriptions of descriptions)
7. BibFrame (nominal things)
8. Schema.org (a potpurri of perspectives)
10 Minute Break
Reconvene at 10:40 AM

bit.ly/C4LDataModeling101
Modeling in (a) Context

What does Data Modeling as described look like?

bit.ly/C4LDataModeling101
Communication & Data Modeling


bit.ly/C4LDataModeling101
Communication of Needs: An Example

Our repository contains digital collections. Digital collections contain objects that are both born-digital and digital surrogates of analog objects. Objects can be a book, journal, image, artwork, or other. Objects can contain multiple parts. Each object or part can have a digital representation. It may have multiple files for different types of representation.

Digital collections should have curators, titles, identifiers. Objects could have titles, subjects, identifiers, creators, languages, and relationships to other bibliographic resources.

Some digital collections and some objects can be viewed by only a subset of users. All objects can be edited only by a editors.

bit.ly/C4LDataModeling101
When Starting Towards Modeling...

1. What concepts exist?
   a. Entities?
   b. Relationships?
   c. Types?
   d. Operations?

2. What information structure do you want to preserve?

3. Any existing models or patterns you could use?

bit.ly/C4LDataModeling101
Communication of Needs: An Example

Our repository contains digital collections. Digital collections contain objects that are both born-digital and digital surrogates of analog objects. Objects can be a book, journal, image, artwork, or other. Objects can contain multiple parts. Each object or part can have a digital representation. It may have multiple files for different types of representation.

Digital collections should have curators, titles, identifiers. Objects could have titles, subjects, identifiers, creators, languages, and relationships to other bibliographic resources.

Some digital collections and some objects can be viewed by only a subset of users. All objects can be edited only by editors.

bit.ly/C4LDataModeling101
Communication of Needs: An Example

Our repository contains digital collections. Digital collections contain objects that are both born-digital and digital surrogates of analog objects. Objects can be a book, journal, image, artwork, or other. Objects can contain multiple parts. Each object or part can have a digital representation. It may have multiple files for different types of representation.

Digital collections should have curators, titles, identifiers. Objects could have titles, subjects, identifiers, creators, languages, and relationships to other bibliographic resources.

Some digital collections and some objects can be viewed by only a subset of users. All objects can be edited only by a editors.

bit.ly/C4LDataModeling101
Communication of Needs: An Example

**Domain**: Digital Repository

- **Classes**: digital collections, (digital) objects, analog/physical objects, parts, representation, formats, user, editor, ...
- **Attributes**: titles, subjects, identifiers, creators, languages, ...
- **Relationships**:
  - Collection contains Object ; Object contains Part
  - Object or Part has Representation
  - Object is a digital Surrogate of Analog/Physical
- **Operations**:
  - Subset of Users can view Objects
  - Editors (Type of Users) can edit Objects
- **Types/Enumerations**: (format) book, journal, image, artwork, ...

bit.ly/C4LDataModeling101
One Possible View...
Another Possible View...
Data Models & Application Independence

W3C WebAccessControl Model

Our Model

WebAC Implementations
Preservation of Information via Data Models

- If, or rather, as applications evolve/migrate, one can still decode the information context from the Model
- Break reliance of understanding information on 1 person, process, database structure, or application
- Capture Information Context for future re-use vis-a-vis the Conceptual Model

bit.ly/C4LDataModeling101
Data Models & Database Structures

- Internal in Internal/Conceptual/External Approach
- Alternatively, Database Structures are Subset of Data Models (thinking Entity-Relationship Models primarily)
  - Logical Structure of Database
  - Physical Structure of Database Instance
  - In RDF & Graph Stores, Logical == Physical
- Database Structures detail how Data in Domain Data Model is Stored & Managed

bit.ly/C4LDataModeling101
Data Models & Database Structures in RDF

@prefix ex: <http://example.edu/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/terms/> .
ex:Object a rdfs:Class ;
    rdfs:label "Object" ;
    rdfs:comment "Digital Object held by a repository." .
ex:DigitalCollection a rdfs:Class ;
    rdfs:label "Digital Collection" ;
    rdfs:comment "Aggregation of Digital Objects in a repository." .
ex:contains a rdfs:Property ;
    rdfs:label "surrogate of" ;
    rdfs:domain ex:DigitalCollection ;
...

ex:DigitalCollection

ex:contains

ex:Object
dc:title: rdfs:Literal
dc:subject: skos:Concept
dc:identifier: rdfs:Literal
dc:creator: foaf:Agent
dc:language: lexvo:Language
dc:type ex:Type

1...n
ex:surrogate

ex:curator: foaf:Agent
dc:title: rdfs:Literal
dc:identifier: rdfs:Literal
Data Models & Serializations

- Converts Data Structures (from Data Models) into Format
- Format is Translation of Data Structure used for Transfer, Storage, Distribution among Systems
- Serialization Contains All Needed Information to (re-)Construct In-Memory Model
  - So Serializations are Logically Equivalent
- Examples:
  - Tabular Data -> CSV, TSV
  - RDF Data -> Turtle, N-Triples, RDF/XML, JSON-LD

bit.ly/C4LDataModeling101
Data Models & Serializations in RDF

@prefix: <http://example.edu/> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/> .
:MyCollection a :DigitalCollection ;
  dc:identifier "ex12345" ;
  dc:title "Title Value" ;
  :curator :JaneSmith .
:JaneSmith a foaf:Agent ;
  foaf:name "Jane Smith" ;
  ...

bit.ly/C4LDataModeling101
Data Models & Data Exchange Protocols

- Protocol State Machines in Olivé is Broader Idea
- Rules (Standards) Outlining Syntax, Semantics & Synchronization of Data or Information across Systems
- Protocols: SOAP; Z39.50; SRU; OAI-PMH; SPARQL
  ○ Also, HTTP == Hypertext Transfer Protocol, on which we build the Web & Leverage often for RDF transfer
- Data Models can Surface Needs, Expectations of as well as Data Instances & Formats used in Exchange Protocols

bit.ly/C4LDataModeling101
... Now Data Models & Data Protocols in RDF

Post
/test?graph-uri=http%3A%2F%2Fexample.edu%2F HTTP/1.1
Host: example.edu/sparql
Accept: */*
Content-Type: application/sparql-update
Content-Length: 136

PREFIX ex: <http://example.edu/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/terms/>
INSERT { ?digitalCollection ex:curator ex:JaneSmith }
WHERE { ?digitalCollection dc:identifier “ex12345” }

bit.ly/C4LDataModeling101
Application Logic & Data Models

• Application Logic == Logic Specific to an Application
• Fundamentally different from Domain & Infrastructure Logic due to Target & Re-usability:
  ■ Domain Logic should be reusable in systems or applications addressing the same domain
  ■ Infrastructure is generic technical capabilities
• Application Logic should leverage the Data Model, may serialize Data into Application-Language Objects

bit.ly/C4LDataModeling101
Application Logic & Data Models with RDF

```ruby
module Hydra::Works
  # This module provides all the Behaviors of a Hydra::Works::GenericFile
  #
  # behavior:
  # 1) Hydra::Works::FileSet can contain (pcdm:hasFile) Hydra::PCDM::File (inherits from Hydra::PCDM::Object)
  # 2) Hydra::Works::FileSet can contain (pcdm:hasRelatedFile) Hydra::PCDM::File (inherits from Hydra::PCDM::Object)
  # 3) Hydra::Works::FileSet can aggregate (pcdm:hasMember) Hydra::Works::FileSet
  # 4) Hydra::Works::FileSet can NOT aggregate anything other than Hydra::Works::FileSets
  # 5) Hydra::Works::FileSet can have descriptive metadata
  # 6) Hydra::Works::FileSet can have access metadata

  module FileSetBehavior
    include ActiveSupport::Concern

    included do
      def self.type_validator
        Hydra::PCDM::Validators::CompositeValidator.new(
          Hydra::Works::NotCollectionValidator,
          super
        )
      end
    end

  end
end
```

github.com/projecthydra/hydra-works/blob/master/lib/hydra/works/models/concerns/file_set_behavior.rb
Data Models & Metadata Standards

- Rules that Establish Common Semantics for Particular Domain of Data
- In turn supports “Correct” Use & Understanding of Data by Users vis-a-vis Commonly-Defined Attributes
- Data Models leverage Metadata Standards to define their own Semantics & Requirements
- Data Models can reuse, extend multiple Metadata Standards
- Standards can be built off of a common Model

bit.ly/C4LDataModeling101
@prefix: <http://example.edu/> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/> .
:MyCollection a :DigitalCollection ;
  dc:identifier "ex12345" ;
  dc:title "Title Value" ;
  :curator :JaneSmith .
:JaneSmith a foaf:Agent ;
  foaf:name "Jane Smith" ;
...
Expressing Data Models
Quick Review of Modeling Languages

How models get expressed - a limited review of some conceptual modeling languages & methods:

- UML*
- XML*
- Java (Ruby, Python, ... Programming Language) Classes
- RDF
- Diagramming

bit.ly/C4LDataModeling101
Universal Modeling Language (UML)

- Standardized Modeling Language
- Complex but General Purpose
- Different Diagram Groups
- User for Software Engineering
- Visualizes designs of Systems from a few approaches:
  - Use Cases
  - Static or Class Diagrams
  - State changes
  - ...

bit.ly/C4LDataModeling101
Unified Modeling Language (UML)
Simple Class Diagram
XML

- Document-focused model, with tree structure containing nodes in hierarchies - elements, attributes, values
- Languages that support models in XML:
  - XML Schema
  - RELAX NG
  - Schematron

bit.ly/C4LDataModeling101
**XML data model:**

**Portion of XML Schema for MODS 3.6**

```xml
<xs:element name="location" type="locationDefinition"/>

<xs:complexType name="locationDefinition">
  <xs:sequence>
    <xs:element ref="physicalLocation" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="shelfLocator" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="url" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="holdingSimple" minOccurs="0"/>
    <xs:element ref="holdingExternal" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="languageAttributeGroup"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="altRepGroup" type="xs:string"/>
</xs:complexType>

******** Subordinate Elements for <location>

******** physicalLocation ********

<xs:element name="physicalLocation" type="physicalLocationDefinition"/>

<xs:complexType name="physicalLocationDefinition">
  <xs:simpleContent>
    <xs:extension base="stringPlusLanguagePlusAuthority">
      <xs:attributeGroup ref="xlink:simpleLink"/>
      <xs:attribute name="displayLabel" type="xs:string"/>
      <xs:attribute name="type" type="xs:string"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

******** altRepGroup ********

<xs:element name="shelfLocator" type="stringPlusLanguage"/>
```

www.loc.gov/standards/mods/v3/mods-3-6.xsd
Java Classes

- **Object-Oriented Programming Language:**
  - Objects aggregate Data, Attributes & Methods
  - Objects communicate with each other by invoking methods
  - Basic Programming Unit is a Class
  - Object == Instance of a Class

- In Java, all Classes except Object inherit information defined in another Class
  - Java’s Object class defines Common Behaviors
  - Java Classes define Data Representation for Objects

- Class declares variables for storing values of Primitive Types & references to Objects

bit.ly/C4LDataModeling101
@Scope("request")
@Path("/{path: .*}")

public class FedoraLdp extends ContentExposingResource {

    private static final Logger LOGGER = getLogger(FedoraLdp.class);

    /**
     * Retrieve the node profile
     * @param rangeValue the range value
     * @return a binary or the triples for the specified node
     * @throws IOException if IO exception occurred
     */
    @GET
    @Produces({"TURTLE + ;qs=1.0", "JSON_LD + ;qs=0.8",
               "N3, N3_ALT2, RDF_XML, NTRIPLES, APPLICATION_XML, TEXT_PLAIN, TURTLE_X,
               TEXT_HTML, APPLICATION_XHTML_XML"})
    public Response getResource(@HeaderParam("Range") final String rangeValue) throws IOException {
        checkCacheControlHeaders(request, servletResponse, resource(), session);

        LOGGER.info("GET resource '{externalPath}'");

        [...]

module DPLA::MAP
  class Aggregation < ActiveTriples::Resource
    configure :type => RDF::ORE.Aggregation

    validates_presence_of :sourceResource, :originalRecord, :isShownAt, :object, :provider

    property :sourceResource, :predicate => RDF::EDM.aggregatedCHO, :class_name => 'DPLA::MAP::SourceResource'
    property :dataProvider, :predicate => RDF::EDM.dataProvider, :class_name => 'DPLA::MAP::Agent'
    property :originalRecord, :predicate => RDF::DPLA.originalRecord
    property :hasView, :predicate => RDF::EDM.hasView, :class_name => 'DPLA::MAP::WebResource'
    property :intermediateProvider, :predicate => RDF::DPLA.intermediateProvider, :class_name => 'DPLA::MAP::Agent'
    property :isShownAt, :predicate => RDF::EDM.isShownAt, :class_name => 'DPLA::MAP::WebResource'
    property :object, :predicate => RDF::EDM.object, :class_name => 'DPLA::MAP::WebResource'
    property :preview, :predicate => RDF::EDM.preview, :class_name => 'DPLA::MAP::WebResource'
    property :provider, :predicate => RDF::EDM.provider, :class_name => 'DPLA::MAP::Agent'
    property :rightsStatement, :predicate => RDF::EDM.rights, :class_name => 'DPLA::MAP::RightsStatement'

    def jsonld_context
      DPLA::MAP::CONTEXT[@context]
    end

    def to_jsonld
      JSON::LD::API.frame(JSON.parse(dump(:jsonld)), DPLA::MAP::FRAME)
    end
  end
end

https://github.com/dpla/dpla_map/blob/map-4.0/lib/dpla/map/aggregation.rb
RDF, RDFS, OWL, RDF/XML - here, FOAF

```
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Organization" rdfs:label="Organization"
    rdfs:comment="An organization." vs:term_status="stable">
    <rdfs:subClassOf rdf:resource="http://xmlns.com/wordnet/1.6/Organization"/>
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
    <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1"/>
    <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  </rdfs:Class>

  <rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Group" vs:term_status="stable" rdfs:label="Group"
    rdfs:comment="A class of Agents.">
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Group"/>
  </rdfs:Class>

  <rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Agent" vs:term_status="stable" rdfs:label="Agent"
    rdfs:comment="An agent (eg. person, group, software or physical artifact).">
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Group"/>
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
  </rdfs:Class>
</rdf:RDF>
```

bit.ly/C4LDataModeling101  
http://xmlns.com/foaf/spec/
RDF Diagram (Mostly) Using Graffoo
Breakouts
4 Data Models Breakout Groups Options

1. BIBFRAME 2.0
2. Portland Common Data Model (PCDM)
3. Europeana Data Model (EDM) / Digital Public Library of America (DPLA) Metadata Application Profile
4. International Image Interoperability Framework (IIIF) Presentation API

If possible, pick model you’re least familiar with

bit.ly/C4LDataModeling101
BIBFRAME 2.0

- Conceptual/practical **model** for bibliographic resources & other other cultural materials
- BIBFRAME “Core” breaks with FRBR & RDA, using simplified Work - Instance - Item
- BIBFRAME v.“2.0” released in 2016 & is still being developed
  - Differs from BIBFRAME v.“1.0” in some significant ways
  - Analysis of the BIBFRAME Ontology / Rob Sanderson, 2015 (bit.ly/bibframe-analysis) followed in many regards
BIBFRAME 2.0

“Core”
Portland Common Data Model (PCDM)

- Started Officially in 2015
- Community Effort for Interoperable Modeling of Digital Repository Objects
- PCDM extends Object Reuse & Exchange (ORE) Data Model
- Interacts with Other Specifications (like LDP), Platforms (like Fedora 4), but Modeled to be Neutral
- Hydra works with PCDM Data from Datastore (Fedora 4) by Translating into Ruby Objects leveraging ActiveFedora
PCDM "Core"
Europeana Data Model (EDM) / Digital Public Library of America (DPLA) MAP

- RDF-based, Descriptive Models
- Focus is Cultural Heritage Resource Aggregation
- Extends CIDOC-CRM as well as Other Models, Vocabularies
- DPLA is Subset of EDM
- Basis for Profiles, i.e. not Designed to be Exhaustive
- More Concerned with Resource Description than System Functions
Fragment of Europeana Data Model Diagram

ore:Aggregation

edm:ProvidedCHO

edm:aggregatedCHO

edm:hasView

edm:WebResource

edm:rights

http://creativecommons.org/licenses/by-nc-sa/3.0/

edm:WebResource

edm:rights

http://creativecommons.org/licenses/by-nc-sa/3.0/

edm:rights

http://creativecommons.org/licenses/by-nc-sa/3.0/
DPLA v.4
Model Diagram
International Image Interoperability Framework (IIIF)

- Set of 4 APIs & Models Working Together to Share Images:
  - Presentation (the focus of the break-out)
  - Image
  - Authentication
  - Search

- Growing Range of IIIF-Compatible Image Viewer Systems
  - Mirador
  - Universal Viewer
  - ...
IIIF
Presentation
API Model
Diagram

* Also available via “members”
N.B. These properties will be deprecated in 3.0
Breakouts’ Shared Example

Digital Surrogate of “A. Danforth's Great Vegetable Pain Destroyer [inscribed]” from UCLA Digital Collections


Data Models Breakouts Activity

Learn about Your Data Model & Discuss:

1. Entity Types;
2. Relationships Types;
3. Attributes.
4. Contextualize within Domain, Applications, Standards;
5. Diagram an Instance of the Data Model with a Shared Example.

* Please Capture Work & Diagrams in the Shared Notes *
bit.ly/C4LDataModeling101
Breakouts until 11:50

BIBFRAME 2 - Christina
PCDM - Steve
IIIF - Hector
EDM/DPLA - Mark

Use the Shared Drive for Notes

bit.ly/C4LDataModeling101
Breakouts Recap
Volunteer from each Breakout to Report Back on Your Group’s Modeling Work

bit.ly/C4LDataModeling101
Different Models, Different Data?

- How do our breakout group responses, ideas, approaches compare?
- How do we use modeling to build interoperability + shared understanding?
- Where do we see differences in what data modeling accomplishes across these breakouts?

bit.ly/C4LDataModeling101
Building Data Modeling in the Code4lib Community/Beyond
Maintaining Data Modeling Momentum

- Code4lib Slack channels that discuss modeling:
  - #c4l17-datamodeling -> #datamodeling
- Hydra Metadata Interest Group
  - #metadata on project-hydra Slack
- Others…?
- Notes/Shared Resources from Workshop - Keep Adding to this!

bit.ly/C4LDataModeling101
Broader or Related Communities Working on Modeling

- PCDM
- IIIF
- Fedora 4 (& Fedora broadly)
- Islandora
  - CLAW / Islandora & Fedora 4 Architecture
  - Islandora Interest Groups (Includes Metadata)
- Europeana Data Modeling Work
- W3C Efforts: LDP, Shapes, ...
- DCMI Work

bit.ly/C4LDataModeling101
Data Modeling Tools & Resources

- **Data Modeling Resources Going Forward**
  - Starter List of Tools
  - Some Links to Modeling Work in Cultural Heritage Institutions

- **Data Modeling Needs in Code4Lib & Related Communities:**
  - Location for Open Discussions & Issues

bit.ly/C4LDataModeling101
Your ideas to continue support for data modeling in communities?

bit.ly/C4LDataModeling101
Thank you!

bit.ly/C4LDataModeling101