Beyond the Tutorial: Complex Content Models in Fedora 3
Presentation Proposal for the Open Repositories Conference 2008
Fedora Users Group
from
Peter Gorman, University of Wisconsin Digital Collections Center
Scott Prater, University of Wisconsin – Madison, Division of Information Technology
March 6, 2009

Abstract

The University of Wisconsin Digital Collections Center recently began a pilot project to create a digital collection of learning objects stored in a Fedora 3.1 repository. This pilot project is the proof-of-concept of many ideas and discussions, extending back for over five years, concerning the problem of storing, searching, and retrieving heterogeneous objects and object types, linked together in complex relations, in a way that is loosely coupled with front-end user display applications. During the course of this presentation, we will describe the issues we've confronted implementing increasingly rich digital collections over the past decade (including some real-life examples of complex digital objects), models we have developed with to resolve many of those issues, and how we have started implementing those models in Fedora 3.1, using the new Content Model functionality.

The Problem

The University of Wisconsin Digital Collections[1] has grown from a small number of HTML-based book projects in the late 1990s to a fully-developed digital library comprising nearly two million pages of text and over 60,000 other images, audio, and video resources, as well as over 4,000 archival finding aids. As with most other digital libraries, the infrastructure supporting these collections has grown organically over the years, gradually upgrading technologies, adding new user services, and supporting new content types as requested by librarians and faculty. It is not surprising, therefore, that over the years content has come to be sequestered within technological 'silos' accommodating the needs of particular services or types of content.

In addressing the balkanization of content, the UW Digital Collections Center recognized the need to develop a new infrastructure which could address three major goals:

1. integrate resource discovery and display across content types,
2. move away from collection-based resource organization,
3. integrate resource management in order to prepare for a true digital preservation platform.

Fedora Commons was chosen as the best platform on which to rebuild the UWDC infrastructure to meet these needs. It also appeared to provide the best means to preserve intact the rich array of complex object types the UWDC has created over the years.

After several years of preliminary discussions and planning, the UWDC began working with Fedora 3.0 in Fall 2008. We soon realized that the very first implementation would have to be able to support complex multimedia objects of several types, so we began by defining abstract object models for several existing multimedia objects:

[1] http://uwdc.library.wisc.edu/

1. A folk tune consisting of an audio stream, a sequence of JPEG sheet music page images, and a PDF of the sheet music[2]
2. A decorative trade binding, with multiple discrete views of the book, each available in several resolutions[3]

From working with these and similar objects, it became clear that a single 'logical object' could contain any number of content streams of varying types, and these high-level objects could not be easily categorized by the number or type of their content streams. There were, however, discrete classes of content objects: single images, simple image sequences, images with associated detail images, simple audio streams, etc.

While we were modeling the objects' internal characteristics, we also realized that any single object could also exist in one or more *contexts:* a single image could simultaneously be a single member of an image gallery (with its own descriptive metadata), but also be a page in the middle of a book. Objects would have to be defined in such a way that they could participate in multiple contexts (some of which might be volatile) without extensive refactoring.

The Model

Early on, we discovered the need to describe our objects both from a structural perspective (simple, composite objects) and from a functional perspective (first class parent objects, second class child objects):

	simple	**composite**
firstClass	citation	piece of music
secondClass	audio stream	image sequence

The Fedora 3.0 Content Model Architecture gave us the ability to define models that could describe an object from both perspectives. In this regard, the "mix-in" approach that Fedora implements for its content models (i.e., each content model is a discrete, stand-alone specification of content and disseminations, atomic and unrelated to other content models) allowed us to have our cake and eat it too: objects can belong to different content models and exhibit different behaviors depending on the context in which they are called. The mix-in approach also allows us to simulate multiple inheritance: an object can have several "parents", each parent an instance of a different model.

The core of our model is the idea of a first class object. The first class objects are strictly logical objects, carrying descriptive and some structural metadata, and point to zero or more children that contain discrete digital content. These first class objects hold the metadata that will be indexed; search results will display the information in these objects, and display of results will

[2] http://digital.library.wisc.edu/1711.dl/SSRecIDSearch?repl1=WiscFolkSong&repl2=WiscFolkSong.000089.bib

[3] http://digital.library.wisc.edu/1711.dl/SSRecIDSearch?repl1=PBO&repl2=_PBO.pbl00093.bib

be keyed off these objects. In a sense, they are the "virtual", or wrapper, object that contains no physical manifestations of the object in itself but, rather, pointers to other objects that are described by this wrapper. In RESTful terms, these first class objects are the persistent Resources; the child object datastreams and disseminations they point to are the Representations of the Resource. We implemented this type of object in Fedora with a CModelFirstClassObject content model.

The vast majority of our objects consist of a single representation (an single image and its metadata). It would have been simpler to attach those binary datastreams directly to their corresponding firstClassObject, but we decided to keep a strict boundary between the Resource and its Representation. This allows us to maintain consistency in structural implementation (all firstClassObjects have the same structure). More importantly, it gives an object room to grow and evolve over the course of its life: adding, changing, removing representations of an object (datastreams, disseminations, even hierarchies of child objects) requires minimal updating of the parent firstClassObject, if any, and no retooling of the applications that consume the data.

The firstClassObject also gives us a way to create new contexts (as defined above), either by adding more descriptive metadata datastreams to an existing firstClassObject, or by simply defining a new firstClassObject, and linking the relevant child objects to it, thereby promoting object reuse in multiple contexts.

Second class objects are the simpler case: they are the things themselves. However, there are various kinds of "things": some of the more complicated objects are images with accompanying close-ups of details, or page scans of a book. To handle these cases, we developed models that contain structural metadata to link related second class objects together.

Implementation

We created the following content model objects in Fedora 3 to implement the object models described above:

CModelUWDCObject: The parent object for all UWDC objects. This Content Model is a purely an abstraction convenience right now, a way for us to distinguish all our digital objects from "meta-objects" (i.e., cmodel objects, sdefine objects, sdeploy objects, objects that contain implementation data such as XSD schemas, etc.) All digital objects belong to this content model.

CModelFirstClassObject: The model for the logical object described above. Objects belonging to this model contain the descriptive metadata about the object. This model's objects must contain one or more BIB*n* XML datastreams with MODS descriptive data. Except in the case of citation objects (in which the citation is encoded in the BIB*n* datastreams), every firstClassObject will contain pointers to one or more children in its RELS-EXT datastream. This content model implements the viewMETS() disseminator, described below.

CModelCompositeObject: This is a model for objects that have children. It is used strictly to represent the structural relationships of the children in a given object. Objects belonging to this model must contain a STRUCT XML datastream with a METS structmap. The RELS-EXT datastreams of objects belonging to this model are presumed to contain RDF pointers to constituent objects.

CModelImageWithDetail: This is a special content model to relate image objects to child image objects (in this case, details of the image stored in the parent object). Objects belonging

to this model will have image bitstreams and disseminations and a GEOMETRY XML datastream describing the position of each child detail image in the parent image. The RELS-EXT datastream of objects of this type will contain pointers to child image objects. Objects belonging to this model are presumed also to belong to *CModelCompositeObject* (a case where inheritance would be useful), and any of the static image content models (to support display of the main image).

CModelAudioStream, CModelImageWithDefaultRes, etc.: These are the models for leaf-node objects, the digital object physical datastreams (jpeg, jpeg2000, mp3, etc.) with no subsidiary structure. They do not contain descriptive metadata, only bitstreams, and pointers to their immediate parents (CModelFirstClassObject/CModelCompositeObject logical objects). The primary purpose of these content models are to provide media-dependent disseminators for the bitstreams.

Since Fedora content models do not support inheritance, we simulate inheritance (and multiple inheritance) using a content model "mix-in" strategy: objects may belong to as many content models as makes sense to adequately describe their contents and purpose in the virtual hierarchy. Thus, an object containing an image of a chair, linked to accompanying detail images of the chair, may belong to *CModelUWDCObject, CModelCompositeObject, CModelImageWithDetail*, and *CModelImageWithDefaultRes.*

Once the content model objects were created, we then created a disseminator to return a firstClassObject's structure: the viewMETS() disseminator. This disseminator takes a firstClassObject PID as an argument, and returns a METS XML document with the firstClassObject's metadata, then links to all the descendant objects (arbitrarily deep) and their physical datastream disseminations (getIcon(), getAudioStream(), getZoom(), etc.) This allows a consuming application to be able to retrieve and display information and datastreams embodied in rich, complex object hierarchies in one API call to the repository. Examples of this disseminator in action will be shown.