Home

"Eddies," said Ford, "in the space-time continuum."
"Ah," nodded Arthur, "is he? Is he?"

Trippi 1.4.1 Released
Edwin Shin posted on Oct 17, 2008
Trippi has been updated to include the latest Mulgara release (2.0.6). Notably, this latest release of Mulgara includes updates to the storage layer (XA 1.1) which improves performance while using less space and fewer files. See the Mulgara release notes for more details.

I've been a bit remiss about actually creating file releases for Trippi on Sourceforge when new versions have been tagged, but this version has had a proper release.

Fedora trunk and the maintenance branch have also been updated, so 3.1 and 2.2.4 (due later next week) will both include Trippi 1.4.1 (and thereby Mulgara 2.0.6).

- mulgara
- trippi

funAPI 0.1 Released
Edwin Shin posted on Oct 10, 2008
I've released version 0.1 of the Fedora unAPI HTTP Service to Sourceforge.

No major changes since the original prototype, except for the addition of a FedoraPmhResolver. In contrast to the FedoraResolver, the FedoraPmhResolver relies on Fedora's OAI service to describe the available formats for Fedora objects. Depending on the desired use, one may be more suitable than another. The FedoraResolver is more flexible and can support arbitrary formats, but it requires Fedora 3.x. The FedoraPmhResolver should work with any version of Fedora that supports OAI-PMH (although I've only tested it with 3.0).

Similarly, I expect the DSpacePmhResolver should work with any version of DSpace that support OAI-PMH, but I've only tested it against 1.5.1.

Dissemination Architecture Updates
Edwin Shin posted on Sep 28, 2008
In previous versions of Fedora (including 3.0), datastream inputs to service methods were limited to datastreams that were members of a given data object. A typical scenario involved a dissemination that used XSL to transform one metadata format into another.

For example, in order to create a dissemination which used an XSL document to transform one metadata format into another, each object was required to have a member datastream that contained the XSL document, even though the same XSL document was used by each object.

One way to mitigate this design was to create the XSL datastream in one object (say, demo:foo) and have every other object reference demo:foo's XSL datastream. While this eliminated the need for multiple copies of the same XSL document, every data object still required an XSL datasteam of its own that redirected to demo:foo's XSL datastream.

In order to workaround the requirement for every object to have its own XSL datastream (inline or redirect), the WSDL binding in the service deployment could hardcode a reference to demo:foo's XSL datastream:

```
<wsdl:binding name="DC2MODS_http" type="tns:DC2MODSPortType">
  <http:binding verb="GET"/>
  <wsdl:operation name="transform">
  <http:operation location="SaxonServlet?source=(DC)&style=http://localhost:8080/fedora/get/demo:dc2mods.cmodel
/XSL"/>
```

Although this does the trick, it's a bit of a hack, and it's desirable to be able to describe the datastream binding in a more formal fashion. It's also fragile, because of the hardcoding of the host and port and it also doesn't support authentication (e.g. if authentication is required for API-A).

I came across this issue in the development of the unAPI HTTP service for Fedora. Rather than hardcode a datastream location in the WSDL binding, I extended the DSInputSpec schema to include an optional pid attribute. Absent the pid attribute, the datastream input is still assumed to belong to the data object. However, if the pid is specified, the binding occurs against that pid's datastream.

As shown in the diagram above, the data objects no longer include an XSL datastream. Instead, the XSL datastream is located in demo:cmodel and is referenced in the DSINPUTSPEC datastream of demo:sdep. One thing to note: although the XSL datastream is a part of the content model object, its presence isn't actually described by the content model. In this case, however, I don't think it's appropriate to extend the dsCompositeModel schema such that the dsTypeModel element take a pid attribute. As it stands, the content model object would itself have to have another content model in order to accomplish this.

Other changes include removing the now-obsolete bDefPID attribute from both the Service Deployment (sDep) Method Map and DSInputSpec schemas. To take advantage of these schema updates, the FORMAT_URIs for DSINPUTSPEC and METHODMAP datastreams should be updated to info:fedora/fedora-system:FedoraDSInputSpec-1.1 and info:fedora/fedora-system:FedoraSDepMethodMap-1.1, respectively.

This work is currently available in the FC-254 development branch and is planned for inclusion in the upcoming Fedora 3.1 release.
3 Comments ·

- cma

Fedora 2.2.4 will replace Kowari for Mulgara
Edwin Shin posted on Sep 28, 2008
The upcoming maintenance release of Fedora 2.2.4 replaces the aging Kowari triplestore with Mulgara.

Earlier last week, I committed the Mulgara-related updates to the maintenance-2.2 branch (r7733). Pending the backport of the fix for FCREPO-239, we should be ready for release.

The replacement of Kowari 1.0.5 with Mulgara 2.0.5 (the current stable release at this writing) brings a number of bug fixes, performance and stability improvements, and new features. A full list of modifications since the Kowari fork is available in the Mulgara distribution in the directory KOWARI-MODIFICATIONS.

Notably, with this update, the Resource Index search interface (risearch) no longer supports RDQL. However, TQL continues to be supported and SPARQL support has been introduced.

- resourceindex
- kowari
- mulgara

unAPI and Zotero, meet Fedora and DSpace
Edwin Shin posted on Sep 23, 2008

Background

unAPI is an HTTP API for the few basic operations necessary to copy discrete, identified content from any kind of web application. Although there's general utility in having Fedora support unAPI, the real motivation is enabling the automatic capture of citation information in Zotero.

There are three components to an unAPI implementation:

1. an identifier microformat
2. an autodiscovery link pointing to an unAPI service
3. an unAPI HTTP service

The unAPI HTTP service interface defines three methods:

1. listFormats (e.g.: http://example.org/unapi/fedora)
2. listFormats for a given identifier (e.g. http://example.org/unapi/fedora?identifier=info:fedora/demo:1)
3. getObject for a given identifier and format (e.g. http://example.org/unapi/fedora?identifer=info:fedora/demo:1)

The response format for 1 & 2 is an XML document that includes the name of the format, its mime-type, and an optional description of the format. For example:

```
<formats id="info:fedora/demo:1">
  <format name="oai_dc" type="text/xml" docs="http://www.openarchives.org/OAI/2.0/oai_dc.xsd" />
  <format name="mods" type="text/xml" docs="http://www.loc.gov/standards/mods/v3/mods-3-2.xsd" />
</formats>
```

Fedora Implementation

Implementing the first two components of unAPI is quite straightforward. The identifier microformat for a Fedora resource would look like:

```
<abbr class="unapi-id" title="info:fedora/demo:1" />
```

The autodiscovery link might look like:

```
<link rel="unapi-server" type="application/xml" title="unAPI" href="http://example.org:8080/unapi/fedora" />
```

For the moment, I've just modified the viewObjectProfile.xslt (located in $FEDORA_HOME/server/access/) to include these two elements. As a result, the rendering of say, http://www.example.org:8080/fedora/get/demo:1http://example.org/fedora/get/demo:1, would now include those elements.

There are a number of ways to implement the unAPI HTTP service. One approach might take a Fedora digital object (e.g. demo:1), and consider each of its component datastreams as a format. Imagine demo:1 represents a journal article and contains four datastreams, DC, RELS-EXT, XML, and IMAGE.

However, I find this approach unsatisfactory because it doesn't allow for the author's notion of the different formats of the object. For example, the IMAGE datastream might just be a component in an HTML rendering of the journal article and oughtn't be considered a format on its own. Moreover, this object might be bound to services that can generate different representations of the object which wouldn't be captured at all by this approach.

What's called for is a programmatic means of indicating which representations of an object should be considered formats, at least as far as unAPI is concerned. A content model savvy approach might enable a dissemination that returned the unAPI formats appropriate for a given content model instance.

In my prototype, I've added an XML datastream, UNAPI-FORMATS, to a content model object. This datastream describes the disseminations that should be considered an unAPI format. Originally, I intended to describe the formats in RDF, but at least for purposes of the prototype, I'm using JSON (wrapped with <json> tags so that I could have an inline XML datastream). For example:

```
<json>
  [["info:fedora/*/DC","oai_dc","text/xml","http://www.openarchives.org/OAI/2.0/oai_dc.xsd"],
  ["info:fedora/*/sdef:md/get?format=mods","mods","text/xml","http://www.loc.gov/standards/mods/v3/mods-3-2.
xsd"]]
</json>
```

This is an array of arrays. Those familiar with the Resource Index circa Fedora 2.x might be familiar with the first (inner) array element, which we called a dissemination type. It is simply a dissemination URI where the PID of the object is replaced with a "*". The remaining elements correspond directly to the unAPI format elements. Again, this particular implementation was an expedient. A more Fedora-esque solution might employ an sDef & sDep to bind against a service that generated the JSON (or RDF) array.

The unAPI HTTP service is implemented as a separate web app. It's intended to be a general purpose service, not bound to Fedora. The Fedora-specific implementation discussed above is provided by an implementation of an ObjectResolver interface. As a proof of concept, I also implemented an OAI-PMH resolver, designed to provide unAPI services for any application that exposes OAI-PMH.

Zotero Integration

Enabling automatic citation capture in Zotero typically involves the creation of a site translator. However, translators depend on regular expression matching against a site's URL, which doesn't work for the general case of supporting any Fedora based repository. Another approach would be to embed Zotero-supported metadata in disseminations, e.g. embedding COinS in HTML renderings of Fedora objects.

By far the most flexible approach with all-around utility appears to be exposing unAPI in Fedora.

One noteworthy detail: Zotero doesn't appear to parse any other format than mods, at least when using unAPI. In my server access logs, after the request for the object (e.g. http://example.org/fedora/get/demo:1), the next request is for the mods format (e.g., http://example.org/unapi/fedora?id=info:fedora/demo:1). I don't see any requests for the various formats that might be available for that resource (e.g. http://example.org/unapi/fedora?id=info:fedora/demo:1). This is using Zotero 1.0.7, I haven't tried the 1.5 preview release.

Once I added a MODS datastream to my Fedora objects, browsing to an object's profile view yielded the little blue icon in my browser's location bar, indicating that Zotero could grok my Fedora object.

DSpace Integration

As mentioned above, I also implemented an OAI-PMH resolver for the unAPI HTTP service. As DSpace provides OAI-PMH services, providing Zotero support only involved the additional steps of enabling the MODS crosswalk and adding the unAPI microformat identifier and unAPI service link to display-item.jsp.

Although the OAI-PMH resolver would also work with Fedora's OAI provider, the Fedora resolver allows for more flexibility. The Fedora resolver is not limited to the formats exposed by OAI. If Zotero integration is the only use case, this additional flexibility isn't important. It shouldn't be difficult to write a DSpace-specific resolver as well. I'm just not familiar enough with DSpace's API to do it myself.

Future Work

This work is part of a general Zotero integration effort. Making Fedora content Zotero-friendly is just one side of the coin. On the reverse is using Zotero as a client to Fedora. If people are already using Firefox & Zotero to browse, cite and archive resources, it's just a small leap to imagine Zotero as a client that enables users to save those resources (with proper metadata, no less) to a Fedora-based repository, and to share, preserve and re-use that content in the context of a Fedora repository. And of course others will come and use Zotero to cite, annotate and share this new Fedora resource anew.

The Zotero 1.5 preview release showcases initial support for saving content to a remote server (in contrast to the local SQLite database used in 1.0.x). Unfortunately (from my perspective), it looks like Zotero's remote sync feature requires WebDAV support and I haven't heard of any plans to support other protocols (AtomPub, anyone?). I'm wary of the effort that implementing WebDAV in Fedora would require.

- zotero
- unapi
- oai

Bug Parade - DC, Checksums, et al.

Edwin Shin posted on Sep 19, 2008

Bugs fixed in dev-150796:

1. FCREPO-215 (Error validating checksum when adding/updating M datastreams)
2. FCREPO-223 (xml:lang attribute dropped from DC datastream)
3. FCREPO-236 (RI does not delete all triples)

The retainPIDs configuration parameter narrowly missed being purged entirely from the repository code base. For the time being, it lives on, but it is now entirely optional, defaulting to "*", or, allow every PID namespace. Although I can conceive of cases where this parameter would be used, it hardly seems worth the confusion it causes among new users. So it will carry forward into the next release in a diminished state, and perhaps disappear entirely the release after that.

3rd-party library updates as well, including the latest MySQL and Postgres JDBC drivers, JUnit, and XMLUnit.

Trippi 1.4 has also been tagged and is now included in both dev-150796 and dev-2075361.

- trippi
- mulgara
- resourceindex

SPARQL Support

Edwin Shin posted on Sep 16, 2008

I've updated Trippi to expose SPARQL in Mulgara. This was easier to implement than I had originally thought. I had thought that exposing Mulgara's SPARQL support in Trippi would require the use of Mulgara's new Connection API, but that was not the case. I've also updated Trippi with the latest Mulgara release (2.0.5).

Pending review of a patch submitted by Morten Grouleff for an unrelated Trippi bug, I'll tag a new 1.4 release for Trippi. In the meantime, dev-150796 and dev-2075351 (branches for Fedora 3.0 and Fedora 2.2.x, respectively) contain the latest Trippi builds with SPARQL support.

- trippi
- mulgara
- sparql
- resourceindex