unAPI and Zotero, meet Fedora and DSpace

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.