OhioLINK Disseminators

There is http__--dltj.org-2006-05-fedora-disseminators- some background material leading up to this perspective that may be useful to review. (It also has a general discussion about what disseminators are and why they are important.) Keep in mind that OhioLINK's Fedora repository vision doesn't expect to have one front end; rather we anticipate getting to the repository data from a number of genre-, topic-, or technology-specific interfaces. In doing so, a lot of the intelligence about how to handle media types needs to go into the disseminators. With this perspective, one thinks about how an object can present itself in generic ways to a wide variety of interfaces.

The name/label of a disseminator in the repository has three parts:

- action
- presentation
- optional sizing parameters

An action can be one of:

- "get" - raw stream of bits from the datastream
- "view" - HTML-wrapped version of the stream of bits plus activities that can be applied to the datastream intended for access by a GUI or to be transformed via XSLT

I tried to combine the GUI and XSLT actions into "view" on the theory that the HTML wrapper would have sufficient CSS "id" and "class" values to make it possible to style it with CSS or transform it with XSLT. This may not be a practical theory once we get to implementation.

A presentation can be one of:

- "preview" - a small/short version of the datastream returned in the datastream's original format
- "screen" - a roughly GUI-screen-sized version of the datastream returned in the datastream's original format
- "thumb" - a small, static image derivative of the datastream
- "audio" - an auditory derivative of the datastream
- "description" - a Dublin Core description of the item marked up in an HTML table
- "record" - HTML markup of Thumb plus Description (suitable, for instance, as a representation of the object in a browse list)

The final piece of the name is "Sized" which can be used to pass parameters that override the dimensions of the "preview" and "thumb" presentations.

So these would get put together like this (with examples based on still images):

- "getPreview" - return an x-by-y derivative of the datastream
- "getThumb" - in the case of still images, same as "getPreview"
- "viewThumb" - the same derivative as "getThumb" wrapped in an HTML div such as:

```
      <div class="viewThumb" id="viewThumb[PID][DS]">
{panel}
        <div class="getThumb" id="getThumb[PID][DS]">
          <img class="getThumbImg" id="getThumb[PID][DS]Img" alt="Thumbnail of [DS]" src="..."  />
        </div>
        <div class="getThumbOptions" id="getThumbOptions[PID][DS]">
          <span class="getThumbOptionScreen" id="getThumbOptionScreen[PID][DS]">
            <a href="[URL to getScreen]">View Screen-sized</a>
          </span>
          <span class="getThumbOptionDescription"
      id="getThumbOptionDescription[PID][DS]">
            <a href="[URL to getDescription">View Description</a>
          </span>
          ....
        </div>
      </div>
{panel}
```

(where PID is the Fedora PID and DS is the datastream label)

For non-static images, it gets a little more interesting because:

- "getPreview" of a video would return a short video segment defined as the 'preview' of the larger video where as "getThumb" of that same video datastream would return just a single frame from the video.
- "getPreview" of a journal article could return a block of text that is the abstract of the article while "getThumb" of that same journal article could return an image rendering of the first page of the article
- "getScreen" of a journal article could return an HTML fragment of the article itself while "getAudio" might return a prerecorded or computer-synthesized rendition of the article