SubmissionSystem

Thoughts regarding a new submission system architecture for DSpace:

Separation of Submission Elements

At the moment the submission flow is very linear, and due to the way that form validation is performed, there is some frustration to be caused during continual authoring of a document. Instead perhaps a base page (which I refer to as the Workspace Item Page) which has buttons with options like:
Item Information ||
The Metadata collection pages for the workspace item (see Customisable Metadata below for more info)
File Management ||
Upload, Edit, emove, Describe etc of files associated with the item
Licencing ||
Licence information for the submitter, and facilities to accept licences in readiness for submission
Notes ||
Annotations that go with the item. Particularly useful for collaborative authoring (see Notes System below for more info)
Submit ||
Move onto the verify stage and subsequent injection into the workflow system (see WorkflowSystem)
emove ||
emove the workspace item
So the submitter creates an item and then is provided with the option to open each of these sections in turn and perform whatever operations are required. From there, the Submit option could perform all the validation required and flag locations that the user has to go to to complete the item for submission. There is no need for this method to mean that the submission system cannot flow, as there is no reason not to have navigational buttons inside each of the sub elements which would allow the user to jump from location to location with great ease.
Parallel to this, I would suggest that the underlying code reflect something similar, and that we have classes for metadata management, file managemet, and licencing - these may then turn out to be useful in other contexts such as item record display.
For our software I've made a page which contains a whole bunch of options for the workspace item, including Edit, View, Notes and emove. This would probably be a perfectly good launch pad for the other functionality too ichardJones

---

Licencing System

There needs to be greater flexibility in the licencing system. At Edinburgh we are going to need to apply at least two licences to each item: the Site Licence and the Use Licence.
The Site Licence is the one which provides us as the institution the right to preserve, display, migrate and so forth the item, while the Use Licence tells visitors to the site what they may or may not do with the item once they can see it. The contents of both of these licences should, obviously, be up to the institution.
We anticipate that the final structure of the licence will be something like this (I'm working on this just now, so I'll keep you up to date):
{{

```
|+----------------------------------+| Use Licence for all other times |+----------------------------------+| Site Licence |+----------------------------------+}}
```

}
This can then be stored in licence.txt and is a one stop shop for all the licencing for the item for its entire life. We include the time dependent section because for E-Theses we anticipate some items which will be withheld (due to patents pending etc.) for up to 2 years after submission.
I've done some work on this, and have produced a Licence.java file which is basically a method library for doing some basic licence like operations. I'm certain that this could form a good basis for a general system of this nature ichardJones

---

Customisable Metadata

It should be relatively straightforward to define schemas for metadata at submission. I imagine something like the following set of options:
Field ID ||
Unique ID of the field (for the database)
Schema ID ||
ID that links together all the fields in a schema (for the database)
Element ID ||
ID of the metadata element (e.g. the ID in DCegistry)
equired ||
(true/false) Is the field required before submission is allowed
epeatable ||
(true/false) Is the field allowed to be repeatable
Type ||
What sort of field is it (e.g. at the moment we have text and name and so forth). There is an argument, I feel, for removing the idea of typing the fields all together and having everything of one uniform text type,
Authority Controlled ||
Is the item authority controlled (see explanation below)
Free Entry ||
Is the author allowed to modify this field (see explanation below)
Section Name ||
The heading under which this field will be displayed. This can be viewed as a sort of sub-grouping within the metadata schema, so perhaps there's a more formal way of representing this.
In Submission ||
Is the field displayed in the submission flow
In Workflow ||
Is the field displayed in the workflow
Position ||
What position is the field displayed in on the screen. How would this work with Section Name I wonder.
Name ||
The human readable name of the field
Description ||
The human readable description of the field
Example ||
The human readable example of how to fill in the field
Authority Controlled and Free Entry are to cover the possibility that you may want metadata to be obtained from the Item Template, but which is not subsequently editable by the submitter (for example, E-Theses are published by "University of Edinburgh", and the submitter may not change this value); Free Entry, then, suggests that the submitter may edit the value. Interesting results arise if both AC and FE are selected: here we would expect X non-editable fields followed by as many free-text fields as are required by the submitter. So, for example, an E-Thesis has been published by "University of Edinburgh", and there is nothing the submitter can do to change this, but it is feasable that the thesis has also been published by another body, and the option is then available for the submitter to record who that body is at this stage using the free text field beneath the authority controlled field.
The authority controlled functionality has already been applied in the EUL-DSpace Add-On if you want to have a look at it in action.
Anyway, the great thing then is that we can choose what things define the submission system. For example, a table called collection2submission would allow it to be customisable by collection, or you could have decisions based on item types and so forth.
In addition we could have a schema import and export facility with turns the structure of the metadata into an XML document. This can then be distributed as the submission system for purpose Y. For example, I would write a schema which contained the DSpace metadata set up for E-Theses, or we might make one which makes the metadata for EPrints, or one for audio files and so forth. These can then be loaded by anyone that wants to use them, and some administrative tools could be developed to allow for attaching schemas to different actions (e.g. a schema is selected at the collection creation page) or for creation of new schemas.
I figure that there would probably be a class like org.dspace.submit.Metadata that did all the magic here, and that an array of such could be passed from the logic layer to the display layer, which would build the interactive front end based on the values in the Metadata object.
It would also be necessary, at point of creation to tie the item to the metadata schema explicitly, so that subsequent resumption of editing, or workflow editing would access the correct schema. It might even be worth storing which schema was used after archive so that subsequent edits that may or may not be allowed in later version can use the same one. This creates some complications over what happens when you want to remove or edit schemas, though, so it's not necessarily good.

Notes System

For the Edinburgh University Library system, we have implemented a very basic annotation tool which could potentially be of use in other contexts. Primarily our interest has been due to the fact that we are also implementing collaborative or supervised authoring of documents, and a notes sytems makes communication in a recordable way between parties easier.
There are a couple of questions hanging over this system:
1. Are the content of the notes of archival interest?
1. If so, what should be done with them (metadata or written to bitstream)?
1. Are there complex functional requirements for the notes (e.g. should it look more like an email system?), or the simpler the better?
Obviously I have implemented this and it is part of the package that we have produced, but it's not necessarily something that the community at large would be interested in. Comments welcome ichardJones

---

More to be added...