Multi-valued Fields in Hydra

Purpose

The primary objective for this call would be to share what Hydra developers have been doing in their apps to work with multi-valued fields:

- OM Terminology
- Javascript
- MVC code

Hopefully we can find some common ground to recommend a best practices for multi-valued fields in hydra at a minimum. If possible it would be great if we could come up with recommendations on changes to components in the stack (I'm looking at you OM and AF) to facilitate handling some of these use cases.

The Problem

AF terms, by default, are arrays:

```
> obj.foo_term = "bar value"
> obj.save
> obj.foo_term
=> ["bar_value"]
> obj.foo_term = ["bar", "baz"]
> obj.save
> obj.foo_term
=> ["bar", "baz"]
```

If you use the :unique => true option in your delegates, then the term is a single string, and they're aren't any issues. However, if you want multiple values for your terms, you have to do some extra codework in your views to get the forms to work. The common strategy seems to be looping through the array and generating the html code for each term. This sort of action isn't available with the standard rails form_for helpers because they expect single values. Penn State's Scholarsphere loops through the terms at the view level. I use this same approach at the Rockhall.

Adding new fields can be accomplished in two ways:

1. Allow OM to insert in the new term and xml
2. Insert the xml first, then have OM updated the term

Option one works when the terms simple and don't involve complex, nested nodes. OM can deal with nested xml and will insert parent nodes if they don't exist, in which case you can use option 1 in your views. Using the first option involves writing some Javascript that inserts the extra field directly into the html form. This is how Scholarsphere does it (I believe). When the form is submitted back to the rails app, AF sees the additional term in the array, or the new term if it didn't exist before, and simply updates the document with the new term. OM generates the xml as needed... most of the time, it's correct too! If not, then, it's on to option 2.

In complex situations, where the xml has lots of nested nodes, specific attributes, etc. you have to take option 2. In that case, adding additional fields is a two-step process. A blank xml template for the term is inserted into the AF document, then the document is saved, the form is refreshed with the new html and then the term can be entered into the form and the document saved with the new term. I do it this way for the Rockhall's Hydra head using ajax calls to insert new nodes. It's a bit clunky, but it works.

Agenda

1. Review the current strategies at what we like/dislike about them
   a. Stanford
   b. Penn State
   c. Rockhall
2. Discuss options for solutions
   a. XML-only problem?
   b. How does ActiveRecord do it?
3. Implementing a solution
   a. OM versus AF versus Nom
4. Divvying up tasks

Meeting Notes

The issues can grouped into two areas:

1. "Backend" issues with OM and ActiveFedora
2. UI interface issues
Issues with UI tend to involve lots of Javascript and trying to leverage AJAX calls in an effective way. Stanford’s Hydrus application has the problem of keeping indexes of multivalued fields current as users add or delete different nodes. The problem is, if a user deletes field index #1, and this is removed from the xml in Fedora, the index number of the fields in the html must be refreshed so that they match those in the updated Fedora xml descMetadata datastream. The best solution seems to be simply reloading the entire section of the document, making a new call to ActiveFedora, and retrieving a complete array of the all terms with their new indexes.

While everyone seems to have slightly differing strategies for dealing with UI issues, there is a general consensus on approaching issues on the backend when using OM and ActiveFedora. Generally, most folks are solving these problems by using an xml template that inserts a term into an existing xml document. After the term is inserted, the value of the xml can be updated. For example, Stanford does this:


As well as the Rockhall:

https://github.com/awead/hydra-pbcore/blob/master/lib/hydra-pbcore/methods.rb

OM already provides functionality for creating template registries, which Hydrus is using, but the Rockhall is not. Because there is already a lot of overlap with the use of templates for managing multivalued fields, attacking the problem from here seems to be a good starting point.

Results

Adam Wead will write up a document outlining usage examples. These examples will be used to make tests for additional code that can be added to ActiveFedora. Hydra developers congress in December is the targeted time to try and get these changes mapped into ActiveFedora. The usage document will be drafted over the course of the coming weeks and other committers can add/edit as needed to reflect the needs of the community.