

Community models and authorities for Hyrax applications

Julie Allinson, Lead Developer, CoSector (University of London)

Julie Hardesty, Metadata Analyst, Indiana University

Samvera Connect 2017

Evanston, Illinois

A Problem:

Everyone has similar types of digital objects
(**PHOTOGRAPH, BOOK, JOURNAL ARTICLE**)
but **No One** describes them the same way

MY BOOK HAS AN **AUTHOR** (rel:aut)

MY BOOK HAS A **CREATOR** (dc11:creator)

A Problem:

Everyone has similar types of digital objects
(**PHOTOGRAPH, BOOK, JOURNAL ARTICLE**)
but **No One** describes them the same way

MY PHOTOGRAPH HAS A PHOTOGRAPHER (rel:pht)

MY BOOK HAS AN AUTHOR (rel:aut)

MY PHOTOGRAPH HAS AN ARTIST (rel:art)

MY BOOK HAS A CREATOR (dc11:creator)

A Problem:

Everyone has similar types of digital objects

(PHOTOGRAPH, BOOK, JOURNAL ARTICLE)

but **No One** describes them the same way

MY PHOTOGRAPH HAS A **PHOTOGRAPHER** (rel:pht)

MY BOOK HAS AN **AUTHOR** (rel:aut)

MY PHOTOGRAPH HAS AN **ARTIST** (rel:art)

MY BOOK HAS A **CREATOR** (dc11:creator)

A Problem:

Everyone has similar types of digital objects
(PHOTOGRAPH, BOOK, JOURNAL ARTICLE)
but **No One** describes them the same way

MY JOURNAL ARTICLES HAVE **SUBJECTS** (dcterms:subject)

MY JOURNAL ARTICLES HAVE **KEYWORDS** (schema:keywords)

MY PHOTOGRAPH HAS A **PHOTOGRAPHER** (rel:pht)

MY BOOK HAS AN **AUTHOR** (rel:aut)

MY PHOTOGRAPH HAS AN **ARTIST** (rel:art)

MY BOOK HAS A **CREATOR** (dc11:creator)

A Problem:

Everyone has similar types of digital objects

(PHOTOGRAPH, BOOK, JOURNAL ARTICLE)

but **No One** describes them the same way

MY JOURNAL ARTICLES HAVE **SUBJECTS** (dcterms:subject)

MY JOURNAL ARTICLES HAVE **KEYWORDS** (schema:keywords)

WE ALL HAVE DATES BUT THEY'RE DIFFERENT!!!

(dcterms:date, dcterms:created, dcterms:issued, schema:dateCreated)

Another Problem:

Adding object types into Hyrax takes a lot of customization and programming work

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the solr_doc as `contact_email_tesi` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

Another Problem:

Adding object types into Hyrax takes a lot of customization and programming work

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have input type=text (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting index.as :stored_searchable, values will be added to the solr_doc as contact_email_test indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for index.as

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors for this property:

- Can have one or more values assigned. NOTE: By default properties are multi-value. You can also explicitly state this by adding , multiple: true before do |index|
- The remaining basic behaviors are the same as for single-value properties. See more information under [Add the new single-value property to the model](#) Expected behaviors.

Her Problem:

Getting into Hyrax takes a lot of
and programming work

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have input type=text (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting index.as :stored_searchable, values will be added to the solr_doc as contact_email_test indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for index.as

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors:

- Can have explicit
- The relationship under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to that of the single and multi-value properties. We will add a single-value controlled vocabulary field here so that it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/departments"), multiple: false do |index|
  index.as :stored_searchable, :facetable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the property up to be single-value. If this were multi-value, it would follow the behaviors of a multi-value field.

Problem:

Getting into Hyrax takes a lot of programming work

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the `solr_doc` as `contact_email_test` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

Adding the properties to the work-type's new/edit form

Now we want to update `GenericWorkForm` to include each of the new properties. Edit `app/forms/hyrax/generic_work_form.rb` and modify `self.terms` to include all the new properties on the new/edit form. See [Defining Metadata in the Model](#) in section [The modified model](#) to see which properties were added as part of this tutorial.

```
self.terms += [:resource_type, :contact_email, :contact_phone, :department]
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

```
self.required_fields += [:department, :contact_email]
```

Optionally, you can also remove one of the basic properties defined by Hyrax from the set of required fields. See [Other Metadata Customizations](#) in section [Remove a default property from the set of required fields](#) for an example of removing a basic metadata property from the set of required properties.

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors:

- Can have explicit
- The relationship under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to that of the single and multi-value properties. We will add a single-value controlled vocabulary field here so that it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/departments"), multiple: false do |index|
  index.as :stored_searchable, :facetable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the property up to be single-value. If this were multi-value, it would follow the behaviors of a multi-value field.

Problem:

Getting into Hyrax takes a lot of programming work

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the `solr_doc` as `contact_email_text` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors:

- Can have explicit
- The relationship under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to that of the single and multi-value properties. We will add a single-value controlled vocabulary field here so that it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/departments"), multiple: false do |index|
  index.as :stored_searchable, :facetable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the property up to be single-value. If this were multi-value, it would follow the behaviors of a multi-value field.

Adding the properties to the work-type's new/edit form

Now we want to update `GenericWorkForm` to include each of the new properties. Edit `app/forms/hyrax/generic_work_form.rb` and modify `self.terms` to include all the new properties on the new/edit form. See [Defining Metadata in the Model](#) in section [The modified model](#) to see which properties were added as part of this tutorial.

```
self.terms += [:resource_type, :contact_email, :contact_phone, :department]
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

```
self
```

Optionally, fields. See required field properties.

Customizing the form field

To customize a form field, you create a partial with the property name under `app/views/records/edit_fields`. Add form code to display the form as desired. If this is the first form field customization you have made, you will need to create the `records/edit_fields` directories under `app/views`.

You can see [more examples](#) by exploring those created for the default fields in `Sufia`.

For a single-value field (optional)

Use something similar to...

```
<% # app/views/records/edit_fields/_contact_email.html.erb %>
<%= f.input :contact_email, as: :email, required: f.object.required?(key),
  input_html: { class: 'form-control', multiple: false } %>
```

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the solr_doc as `contact_email_text` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors:

- Can have explicit
- The result is under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to that of the single and multi-value properties. We will add a single-value controlled vocabulary field here so that it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/departments"), multiple: false do |index|
  index.as :stored_searchable, :facetable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the property up to be single-value. If this were multi-value, it would follow the behaviors of a multi-value field.

Adding the properties to the work-type's new/edit form

Now we want to update GenericWorkForm to include each of the new properties. Edit `app/forms/hyrax/generic_work_form.rb` and modify `self.terms` to include all the new properties on the new/edit form. See [Defining Metadata in the Model](#) in section [The modified model](#) to see which properties were added as part of this tutorial.

```
self.terms += [:resource_type, :contact_email, :contact_phone, :department]
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

```
self
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

Customizing the form field

To customize a form field, you create a partial with the property name under `app/views/records/edit_fields`. Add form code to display the form as desired. If this is the first form field customization you have made, you will need to create the `records/edit_fields` directories under `app/views`.

You can see [more examples](#) by exploring those created for the default fields in Sufia.

For a single-value

Use something similar

```
<% # app/views/records/edit_fields/_contact_email.html.erb %>
<%= f.input :contact_email, :label => t('hyrax.work.show.contact_email'), :input_html => { class: 'form-control' } %>
```

Create a custom presenter class.

To add your custom metadata to the show page, first you have to create a custom presenter class. NOTE: This class is NOT created when you generate the work type.

The custom presenter class

Create the following as a starting point for the custom presenter class.

```
# app/presenters/generic_work_presenter.rb
class GenericWorkPresenter < Hyrax::WorkShowPresenter
end
```

Assign the presenter class in the generated controller. Edit `app/controllers/hyrax/generic_work_controller.rb` and add the following line under the `curation_concern_type` assignment.

```
self.show_presenter = GenericWorkPresenter
```


Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the `solr_doc` as `contact_email_text` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors

- Can have explicit
- The result under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to that of the single and multi-value properties. We will add a single-value controlled vocabulary field here so that it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/departments"), multiple: false do |index|
  index.as :stored_searchable, :facettable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the property up to be single-value. If this were multi-value, it would follow the behaviors of a multi-value field.

Adding the properties to the work-type's new/edit form

Now we want to update GenericWorkForm to include each of the new properties. Edit `app/forms/hyrax/generic_work_form.rb` and modify `self.terms` to include all the new properties on the new/edit form. See [Defining Metadata in the Model](#) in section [The modified model](#) to see which properties were added as part of this tutorial.

```
self.terms += [:resource_type, :contact_email, :contact_phone, :department]
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

```
self
```

Optionally, you can add properties to the set of required fields. See [required fields](#) for more information on properties.

Customizing the form field

To customize a form field, you create a partial with the property name under `app/views/records/edit_fields`. Add form code to display the form as desired. If this is the first form field customization you have made, you will need to create the `records/edit_fields` directories under `app/views`.

You can see [more examples](#) by exploring those created for the default fields in Sufia.

For a single-value

Use something similar

```
<% # app/v
<%= f.input
input_html
%>
```

Create a custom presenter class.

To add your custom metadata to the show page, first you have to create a custom presenter class.

Get the value from Solr

Delegate retrieval to solr_document

Edit the custom presenter class (e.g. `app/presenters/generic_work_presenter.rb`) and delegate the retrieval of properties to `solr_document` for each of the properties to be displayed.

```
delegate :contact_email, :contact_phone, :department, to: :solr_document
```

The modified presenter class

The full custom presenter class now looks like...

```
# app/presenters/generic_work_presenter.rb
class GenericWorkPresenter < Hyrax::WorkShowPresenter
  delegate :contact_email, :contact_phone, :department, to: :solr_document
end
```

Create methods to retrieve properties from solr

Edit `app/models/solr_document.rb` and add a method to retrieve each property's value from the solr doc. NOTE: Use `Solrizer.solr_name` to generate the solr field name for each property.

```
def contact_email
  self[Solrizer.solr_name('contact_email')]
end
```

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the `solr_doc` as `contact_email_text` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors:

- Can have explicit
- The result is under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to the single and multi-value properties. We will add a single-value controlled vocabulary field here so it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/departments"),
multiple: false do |index|
  index.as :stored_searchable, :facetable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the property up to be single-value. If this were multi-value, it would follow the behaviors of a multi-value field.

Adding the properties to the work-type's new/edit form

Now we want to update GenericWorkForm to include each of the new properties. Edit `app/forms/hyrax/generic_work_form.rb` and modify `self.terms` to include all the new properties on the new/edit form. See [Defining Metadata in the Model](#) in section [The modified model](#) to see which properties were added as part of this tutorial.

```
self.terms += [:resource_type, :contact_email, :contact_phone, :department]
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

self

Optionally, fields. See required field properties.

Customizing the form field

To customize a form field, you create a partial with the property name under `app/views/records/edit_fields`. Add form code to display the form as desired. If this is the first form field customization you have made, you will need to create the `records/edit_fields` directories under `app/views`.

You can see [more examples](#) by exploring those created for the default fields in Sufia.

For a single-value

Use something similar

```
<% # app/v
<%= f.input
input_html
```

Create a custom presenter class.

To add your custom metadata to the show page, first you have to create a custom presenter class.

Get the value from Solr

Delegate retrieval to solr_document

Adding the properties to the show page

Add to set of attributes to display

If this is the first custom property added to the show page, you will need to copy [Hyrax's app/views/hyrax/base/_attribute_rows.html.erb](#) to the same directory structure in your app. NOTE: The link goes to master. Make sure you copy from the release/branch of Hyrax that your app has installed.

Add the properties to the local copy of `app/views/curation_concerns/base/_attribute_rows.html.erb`

```
<%= presenter.attribute_to_html(:contact_email) %>
<%= presenter.attribute_to_html(:contact_phone) %>
<%= presenter.attribute_to_html(:department) %>
```

solr doc. NOTE: Use `Solrizer.solr_name` to generate the solr field name for each property.

```
def contact_email
  self[Solrizer.solr_name('contact_email')]
end
```

Extend the model

To add a new single-value property

To define a property that has a single text value, add the following to the GenericWork model.

```
property :contact_email, predicate: ::RDF::Vocab::VCARD.hasEmail, multiple: false
do |index|
  index.as :stored_searchable
end
```

- It will be limited to a single value (set multiple: true or leave off for multi-value, which is the default behavior)
- If included in the new/edit form, it will have `input type=text` (There is a bit more configuration under section [Add the new single-value property to the new/edit form](#) to have this included in the form.)
- By setting `index.as :stored_searchable`, values will be added to the `solr_doc` as `contact_email_text` indicating this field is English text (te), stored (s), indexed (i)
 - See [Solr Schema](#) documentation for more information on dynamic solr field postfixes.
 - See [Solrizer::DefaultDescriptors](#) documentation for more information on values for `index.as`.

To add a new multi-value property

To define a property that has multiple text values, add the following to the GenericWork model.

```
property :contact_phone, predicate: ::RDF::Vocab::VCARD.hasTelephone do |index|
  index.as :stored_searchable
end
```

Expected behaviors:

- Can have explicit
- The result is under

To add a new controlled vocabulary property

The process for adding a property whose value comes from a controlled vocabulary is identical to the single and multi-value properties. We will add a single-value controlled vocabulary field here so it is available for use in later examples.

```
property :department, predicate: ::RDF::URI.new("http://lib.my.edu/"), multiple: false do |index|
  index.as :stored_searchable, :facetable
end
```

Expected behaviors for this property:

- The behaviors are the same as for single-value properties because we set the single-value. If this were multi-value, it would follow the behaviors of a multi-value.

Adding the properties to the work-type's new/edit form

Now we want to update GenericWorkForm to include each of the new properties. Edit `app/forms/hyrax/generic_work_form.rb` and modify `self.terms` to include all the new properties on the new/edit form. See [Defining Metadata in the Model](#) in section [The modified model](#) to see which properties were added as part of this tutorial.

```
self.terms += [:resource_type, :contact_email, :contact_phone, :department]
```

Optionally, you can add properties to the set of required fields. In this example, we will require the department and contact email.

self

Optionally, you can add properties to the set of required fields. See [required fields](#) properties.

Customizing the form field

To customize a form field, you create a partial with the property name under `app/views/records/edit_fields`. Add form code to display the form as desired. If this is the first form field customization you have made, you will need to create the `records/edit_fields` directories under `app/views`.

You can see [more examples](#) by exploring those created for the default fields in Sufia.

For a single-value

Use something similar

```
<% # app/v
<%= f.inpu
input_html
```

Create a custom presenter class.

To add your custom metadata to the show page, first you have to create a custom presenter class.

Get the value from Solr

Delegate retrieval to solr_document

delegate

Add the properties to the show page

Add to set of attributes to display

If this is the first custom property added to the show page, you will need to copy [Hyrax's app/views/hyrax/base/_attribute_rows.html.erb](#) to the same directory structure in your app. NOTE: The link goes to master. Make sure you copy from the release/branch of Hyrax that your app has installed.

Add the properties to the local copy of `app/views/curation_concerns/base/_attribute_rows.html.erb`

Configure Blacklight to show the property in search results

Optionally, you can configure a property to be shown in the search results for a work.

Edit `app/controllers/catalog_controller.rb` and look for the section including `add_index_field` statements. Add the following:

```
config.add_index_field solr_name("contact_email", :stored_searchable), label: "Contact Email"
```


Things that could help

THING 1:

Hyrax Basic Metadata

Property	Predicate	Multiple
label	ActiveFedora::RDF::Fcrepo::Model.downloadFilename	FALSE
relative_path	::RDF::URI.new('http://scholarsphere.psu.edu/ns#relativePath')	FALSE
import_url	::RDF::URI.new('http://scholarsphere.psu.edu/ns#importUrl')	FALSE
part_of	::RDF::Vocab::DC.isPartOf	TRUE
resource_type	::RDF::Vocab::DC.type	TRUE
creator	::RDF::Vocab::DC11.creator	TRUE
contributor	::RDF::Vocab::DC11.contributor	TRUE
description	::RDF::Vocab::DC11.description	TRUE
keyword	::RDF::Vocab::DC11.relation	TRUE
rights	::RDF::Vocab::DC.rights	TRUE
rights_statement	::RDF::Vocab::EDM.rights	TRUE
publisher	::RDF::Vocab::DC11.publisher	TRUE
date_created	::RDF::Vocab::DC.created	TRUE
subject	::RDF::Vocab::DC11.subject	TRUE
language	::RDF::Vocab::DC11.language	TRUE
identifier	::RDF::Vocab::DC.identifier	TRUE
based_near	::RDF::Vocab::FOAF.based_near	TRUE
related_url	::RDF::RDFS.seeAlso	TRUE
bibliographic_citation	::RDF::Vocab::DC.bibliographicCitation	TRUE
source	::RDF::Vocab::DC.source	TRUE

THING 2:

Predicate Decision Tree

from the
Samvera Metadata Interest Group (SMIG)

Predicate Decision Tree

October 3, 2016, version 1; updated: February 19, 2017, August 3, 2017

Predicate needs for RDF statements arise for various situations such as application development and metadata mapping for migration. Both developers and librarians might find themselves in situations that require looking for a predicate to use or deciding if a new predicate needs to be created. The purpose of this document is to help provide a review process of existing predicates and their application. This is not intended to provide specific recommendations for a given field.

1. Is the predicate for technical metadata?
 - a. Basic technical properties:
<https://wiki.duraspace.org/display/samvera/Technical+Metadata+Application+Profile>
 - b. See also for additional technical properties - [EBUCore](#)
2. Is the predicate for rights metadata?
 - a. <http://wiki.duraspace.org/display/samvera/Rights+Metadata+Recommendation>
3. Is the predicate describing structure?
 - a. PCDM (<https://github.com/duraspace/pcdm/wiki>)
4. Is the predicate for geographic resources?
 - a. [Samvera Geospatial Interest Group](#)
 - b. General spatial characteristics of a resource - [DC.spatial](#)
 - c. Latitude/Longitude - [EXIF](#) (gpsLatitude and gpsLongitude)
5. Is the predicate for preservation events or provenance?
 - a. [PROV-O](#)
 - b. [Premis](#)
6. Converting from MODS?
 - a. <https://wiki.duraspace.org/display/samvera/MODS+and+RDF+Descriptive+Metadata+Subgroup>
 - b. Look at [DPLA Metadata Application Profile](#)
7. None of the above? Search for Existing Predicates
 - a. Prefer common ontologies:
 - i. [Dublin Core](#) (DC)
 - ii. [SKOS](#)
 - iii. [MARC Relators](#) (Creators/Photographers/Agents/Other Publishers)
 - iv. [VRA](#)
 - v. [Darwin Core](#)
 - vi. [Schema.org](#)
 - vii. [Europeana Data Model](#) (EDM)
 - viii. [BIBFRAME](#)
 - ix. [EBUCore](#)

THING 3:

SMIG MODS and RDF Mapping Recommendations

MODS elements as RDF Collaboration Documents (Detailed page: [Collaboration Documents](#))

1. [MODS title](#) (two tabs)
2. [MODS name](#) (two tabs)
3. [MODS typeOfResource](#)
4. [MODS genre](#)
5. [MODS originInfo](#) (three tabs)
6. [MODS language](#)
7. [MODS physicalDescription](#)
8. [MODS abstract](#)
9. [MODS tableOfContents](#)
10. [MODS targetAudience](#) (only BPL and UNC-CH mapped this)
11. [MODS note](#) (two supported options likely)
12. [MODS subject](#)
13. [MODS classification](#) (only BPL and Columbia mapped this)
14. [MODS identifier](#)
15. [MODS physicalLocation](#)
16. [MODS accessCondition](#)
17. [MODS recordInfo](#)
18. [MODS series and collections](#)

THING 4: Institutional MAPs

University of York/University of London data model for Thesis / Dissertation

Model for Thesis / Dissertation

Type:

<http://purl.org/ontology/bibo/Thesis>

Property (Hydra)	Predicate	Type	Expected Object	Single Multi	Solr	Usage
abstract	dc:abstract	property	Text	M	stored_searchable	0..n
date_accepted	dc:dateAccepted	property	Date	M	stored_searchable facettable	0..1
advisor_resource	mrel:ths	habm	Agent	M	_ssim preflabel of Agent indexed in solr as advisor_value_* stored_searchable facettable	0..n Object-based
advisor	uketd:advisor	property	String	M	stored_searchable Additionally indexed in solr as advisor_value_* stored_searchable facettable	0..n Used for advisors that are Strings rather than related objects. Mixed in with _value to provide a full index of advisors.
department_resource	uketd:department	habm	Agent	M	_ssim preflabel of Agent indexed in solr as department_value_* stored_searchable	0..n object-based
awarding_institution_resource	bf:dissertationInstitution	habm	Agent	M	_ssim preflabel of Agent indexed in solr as *_value_* stored_searchable	0..n object-based
qualification_level	uketd:qualificationLevel	property	String	M	stored_searchable	1 file_based
qualification_name	uketd:qualificationName	property	String	M	stored_searchable	1 file_based

And one other **THING**...

Dog Biscuits

Dog Biscuits

Models, vocabularies and behaviours for Hyrax applications 🐶 🍪

Pre-defined Models

- Published Work
- Thesis
- Journal Article
- Exam Paper
- Conference Item
- Dataset



Journal Article

An Article about Mona Lisa - Mozilla Firefox

An Article about Mona Lis x +

localhost:3000/concern/journal_articles/jh343s28d?locale=en

Hyrax

English jhardes@iu.edu

My Dashboard / Your Works / An Article about Mona Lisa

An Article about Mona Lisa

Public Deposited

Relationships


In Administrative Set: Default Admin Set

Descriptions

Attribute Name	Values
Creator	da Vinci, Leonardo
Date created	2017
Date published	2017
Issue number	1
Journal	Journal of Art Images on My OS
Keyword	art
Resource type	Article
Rights statement	http://rightsstatements.org/vocab/NoC-US/1.0/
Volume number	2

Edit Delete

Last modified: 11/02/2017



Download image

f t g+ t

Citations:
[EndNote](#) | [Zotero](#) | [Mendeley](#)

Demo Time

How it works

- Build a Hyrax application
- Add the dog_biscuits gem && bundle install
- Run the dog_biscuits install
- Run the dog_biscuits works generator

What it does

Firstly it runs the Hyrax Work generator

Then it replaces the model, indexer, form, actor and presenter with biscuit-ified ones

It also adds in a replacement catalog_controller

Then it updates the catalog_controller, hyrax locale, schema_org metadata and views (_attribute_rows.html.erb) with the things defined in the configuration

Yeah, but I don't want that ...

- Configuration options
 - Facet_properties
 - Index_properties
 - Singular_properties
 - `#{model}_properties`
 - `#{model}_properties_required`
 - Properly mappings - `help_text`, `labels`, `renderers`, `helpers`
- Add new properties locally
 - Add to model
 - Add to solr document
 - Add configurations ^^

Then re-run the work generator (with the `--skip_model` flag)

Code: https://github.com/ULCC/dog_biscuits (Hyrax2 branch)

Current status: hyrax2 branch will become master once hyrax2 is released;
more work to do on authorities and autosuggest.

Wiki: https://github.com/ULCC/dog_biscuits/wiki

Thank you!