Readux Documentation

Release 1.8.1

Emory University Libraries

Apr 11, 2017

Contents

Contents:

CHAPTER 1

Installation

Instructions to install required software and systems, configure the application, and run various scripts to load initial data.

Software Dependencies

We recommend the use of pip and virtualenv for environment and dependency management in this and other Python projects. If you don't have them installed, you can get them with `sudo easy_install pip` and then `sudo pip install virtualenv`.

CHAPTER 2

Bootstrapping a development environment

- Copy `readux/localsettings.py.dist` to `readux/localsettings.py` and configure any local settings: **DATABASES**, **SECRET_KEY**, **SOLR_**, **FEDORA_**, customize **LOGGING**, etc.
- Create a new virtualenv and activate it.
- Install fabric: `pip install fabric`
- Use fabric to run a local build, which will install python dependencies in your virtualenv, run unit tests, and build sphinx documentation: `fab build`

Deploy to QA and Production should be done using `fab deploy`.

After configuring your database, run syncdb:

    python manage.py syncdb

Use eulindexer to index Repository content into the configured Solr instance.

For annotated edition export functionality, the digital edition jekyll theme automatically handled as a python dependency but the teifacsimile-to-jekyll Ruby gem must currently be installed manually.

CHAPTER 3

Initial QA/production deploy

- Use fabric deploy method
- Configure localsettings (particularly **DATABASES**, **SOLR**, **FEDORA**, **MEDIA_ROOT** and **UPLOAD_TO** settings)
  - Note that Fedora access requires a non-privileged guest account, in order to acess an API-M method for information about a datastream, used for PDF download view, etc.
- Run `python manage.py syncdb`
- Run `python manage.py migrate`
- Configure the site to run under apache (see `deploy/apache/readux.conf` for a sample apache configuration)
- Use Django admin interface to configure the site domain name (used to generate absolute urls to full-text content for use with Voyant)
- Use Django console to update LSDI collection objects with an owner value that can be used for xacml policies and filtering:

```
python manage.py shell
>>> from readux.fedora import ManagementRepository
>>> from readux.collection.models import Collection
>>> repo = ManagementRepository()
>>> colls = repo.get_objects_with_cmodel(Collection.COLLECTION_CONTENT_MODEL,␣
↪type=Collection)
>>> for c in colls:
...     if 'LSDI' in c.label:
...         c.owner = 'LSDI-project'
...         c.save('set owner to "LSDI-project" for xacml policy')
...
```

- Configure eulindexer to point to the new site url, restart eulindexer service, and reindex the site
- Update/deploy xacml to allow API-A access to LSDI collections

- Run a manage script to populate initial collection descriptions:

```
python manage.py collection_descriptions
```

CHAPTER 4

Upgrade Notes

Patch affects all releases until further notice

Readux is currently affected by a bug in Django's debug logic; a patch is available but not yet included in any official Django releases. To apply this patch to the locally installed copy of Django, use the patch file included in the deploy directory. From the top level of your virtualenv directory, run:

```
patch -p0 < django-views-debug.patch
```

Release 1.8

- In `localsettings.py`, replace:

  ```
  django.utils.log.NullHandler
  ```

  with:

  ```
  logging.NullHandler
  ```

  See here for details: *django deprecation - NullHandler <https://www.monotalk.xyz/blog/no-module-named-djangoutilslognullhandler-djangoutilslog-is-not-a-package/>*

- In `localsettings.py`, add Python Social Auth credentials for corresponding environments. There are currently two environments, one being the `testreadux.ecds.emory.edu` and the other being `readux.ecds.emory.edu`. These accounts could be found on social media sites with the login `eulsystems@gmail.com`. Or alternatively, the social auth keys and secrets could also be found on `emory-libraries/settings` GitHub repository under Readux. The social media accounts in this release includes: Facebook, GitHub, Google, Twitter, and Zotero.
- Please read `https://www.pivotaltracker.com/story/show/141019047` for detailed Daphne upgrade details, which include a lot of changes made accommodate breaking changes that were introduced since the ugprade.

Release 1.7

- Run migrations for database updates:

```
python manage.py migrate
```

- Configure Amazon S3 settings for temporary storage of background export for downloaded zip files. See the required fields in `localsettings.py.dist`.
- This release makes use of Channels. See the channels deploy documentation and configure **CHANNEL_LAYERS** in `localsettings.py`.
- **JEKYLLIMPORT_TEI_SCRIPT** can now be specified in `localsettings.py` if the exact path needs to be specified.
- New configuration in `localsettings.py` to configure Fedora Collections to be listed based on owner attribute. For Emory, this should be set to the value **LSDI-project**. See `localsettings.py.dist` for example and more details.
- Requires an updated version of the teifacsimile-to-jekyll gem (0.7).
- Requires using channels-0.17.3 and daphne-0.14.3 as for now to avoid Django channel issues. Newer daphne and channels have been released but we need to update our code in order to adapt to them.

Release 1.6

- Run migrations for database updates:

```
python manage.py migrate
```

Release 1.5

- Zotero should be configured as a social authentication backend to support citation lookups. Add OAuth keys based on the example configuration in `localsettings.py.dist`

Release 1.4

- Basic export functionality requires the teifacsimile-to-jekyll gem version 0.5 be installed (available from 0.5 release on GitHub).

  The application expects the jekyll and jekyllimport_teifacsimile commands to be available in the configured environment path. One way to do this is by creating a file to be sourced when users login and by /etc/sysconfig/httpd. Example environment file:

```
source /opt/rh/rh-ruby22/enable
source /opt/rh/python27/enable
export PATH=$PATH:/opt/rh/rh-ruby22/root/usr/local/bin
```

- The GitHub export uses new **GIT_AUTHOR_EMAIL** and **GIT_AUTHOR_NAME** configurations; defaults are included in `settings.py`, but can be overridden in `localsettings.py`.

Release 1.3

- Some page images in Fedora have a generic mimetype, which Loris can't handle for recognizing and generating images. Before switching to the new version, these should be cleaned up in the python console:

```
from readux.fedora import ManagementRepository
from readux.books.models import PageV1_0
repo = ManagementRepository()
query = '''select ?pid
where {
  ?pid <fedora-model:hasModel> <info:fedora/emory-control:ScannedPage-1.0> .
  ?pid <fedora-view:disseminates> ?ds .
  ?ds <fedora-view:mimeType> 'application/octet-stream'
}'''
results = repo.risearch.find_statements(query, language='sparql', type='tuples')
for n in results:
  page = repo.get_object(n['pid'], type=PageV1_0)
  if page.image.mimetype == 'application/octet-stream':
     page.image.mimetype = 'image/jp2'
     page.save('Updating image mimetype')
     print 'Updated %s' % n['pid']
```

- The new IIIF-based image handling requires new configurations be added to `localsettings.py`: **IIIF_API_ENDPOINT** and **IIIF_ID_PREFIX** (prefix is optional, depending on configuration). See `localsettings.py.dist` for an example.
- Run migrations for database updates:

```
python manage.py migrate
```

- If using MySQL, make sure the database has time zone data loaded: http://dev.mysql.com/doc/refman/5.7/en/mysql-tzinfo-to-sql.html
- The URL format for pages has changed; update page ARK records by running a script:

```
python manage.py update_page_arks
```

- Generate TEI for all volumes with pages loaded:

  python manage.py add_pagetei –all

- The dependency on `eullocal` has been removed, so if you are using an existing virtualenv, eullocal can be uninstalled after this upgrade.

Release 1.2.1

- The dependency on `eullocal` has been removed, so eullocal can be uninstalled after upgrading if re-using a pre-existing virtualenv.
- Update `localsettings.py` to set **DOWNTIME_ALLOWED_IPS** to any IP addresses that should be allowed to access the site during configured downtime periods.

Release 1.2

- This release includes an update to Django 1.7 and includes new database migrations. To update the database, run:

```
python manage.py migrate
```

- LDAP login is now handled by django-auth-ldap. LDAP configuration settings will need to be updated in `localsettings.py`; see example configuration in `localsettings.py.dist`.
- Configure new setting **TEI_DISTRIBUTOR** in `localsettings.py`. See example configuration in `localsettings.py.dist`.
- Readux now supports social authentication via Twitter, Google, GitHub, Facebook, etc. OAuth keys for each of the configured backends should be requested and configured in `localsettings.py`. The list of enabled authentication backends can also be overridden in localsettings, if needed.

Release 1.1

- Update Fedora XACML policies to include new variant content models (ScannedVolume-1.1 and ScannedPage-1.1) and reload policies so that newly ingested content will be accessible.
- Restart eulindexer so it will pick up the new content models to be indexed.
- Configure new setting **LARGE_PDF_THRESHOLD** in `localsettings.py`. See sample config and default value in `localsettings.py.dist`.

Release 1.0.2

- Run **syncrepo** manage script to ensure all Fedora content models are loaded in the configured repository:

```
python manage.py syncrepo
```

Release 1.0

- Run the manage script to import covers for all books:

```
python manage.py import_covers
```

  or by collection:

```
python manage.py import_covers -c emory-control:LSDI-Yellowbacks
```

Note: Ingesting page images requires access to the Digitization Workflow web application and file-level access to the content managed by the Digitization Workflow (e.g., /mnt/lsdi).

- Run the manage script to import pages for *selected* books by pid:

```
python manage.py import_covers pid1 pid2 pid3 ...
```

or by collection:

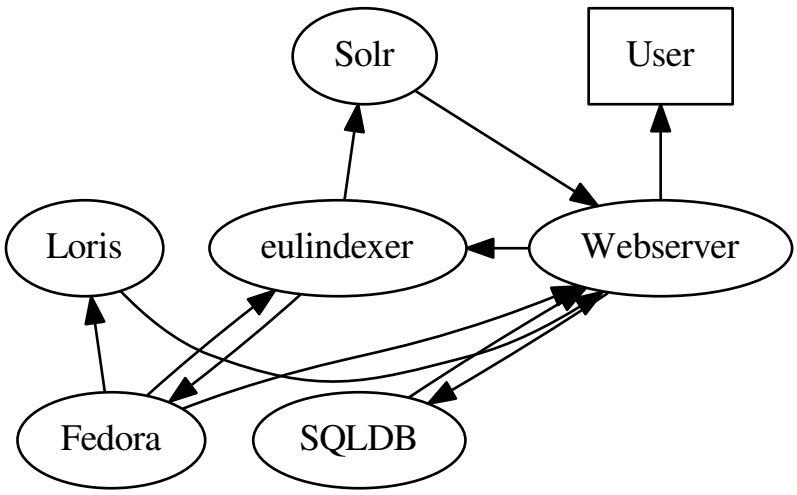
```
python manage.py import_pages -c emory-control:LSDI-Yellowbacks
```

CHAPTER 5

Architecture

High-Level Architecture

The following diagram provides a high-level view of the Readux system architecture. The direction of arrows indicates the flow of data.



Readux is a Django application running on a web server. It uses a SQL database for user accounts, collection banner images and annotations. Collection and digitized book content is stored in a Fedora Commons 3.x repository and

accessed using REST APIs with eulfedora. In normal operations, Readux does *not* ingest or modify content in Fedora (although the codebase does currently include scripts for ingesting contents).

Readux uses Loris IIIF Image Server to serve out page image in various sizes and support deep zoom. Loris is configured to pull images from Fedora by URL. Currently, Loris is not directly exposed to users; Loris IIIF outputs are mediated through the Readux application (although this is something that may be revisited). The page image urls do matter: Readux image annotations reference the image url, so changing urls will require updating existing annotations. Using more semantic image urls (thumbnail, page, full) within Readux *may* be a more durable choice than exposing specific IIIF URLs with sizes hard-coded.

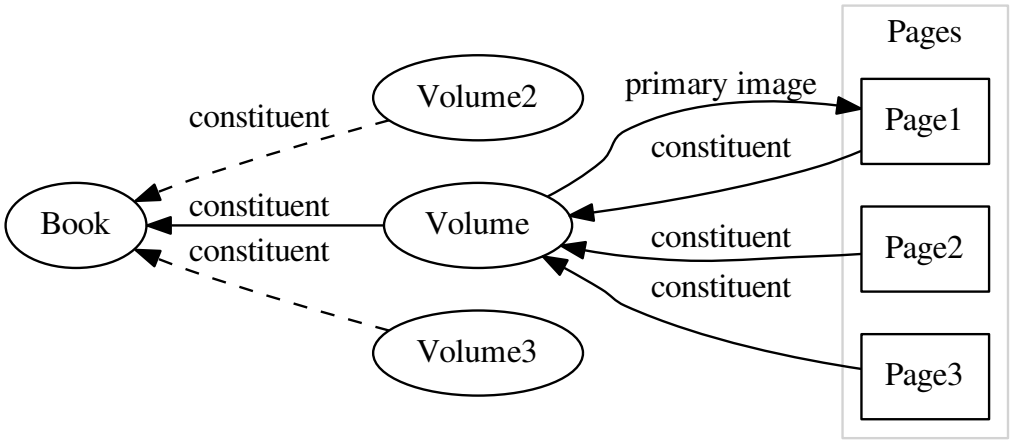
Readux uses Apache Solr for search and browse functionality. This includes:

- searching across all volumes (see `VolumeSearch()`)
- searching within a single volume (see `VolumeDetail()`)
- browsing volumes by collection (see `CollectionDetail()` or `CollectionCoverDetail()`)
- browsing pages within a volume (see `VolumePageList()`)

We use eulindexer to manage and update Fedora-based Solr indexes. `eulindexer` loads the Solr configuration from Readux and listens to the Fedora messaging queue for updates. When an update event occurs, `eulinedexer` queries Fedora to determine the content type (based on content model), and then queries the relevant application for the index data to be sent to Solr. Readux uses the `eulfedora.indexdata` views and extends the default `eulfedora.models.DigitalObject.index_data()` method to include additional fields needed for Readux-specific functionality; see the code for `readux.books.models.Volume.index_data()` and `readux.books.models.Page.index_data()` for specifics. The current Solr schema is included in the `deploy/solr` directory.

Book Content Models

The following diagram shows how Readux digitized book content is structured in Fedora.



For consistency between single and multi-volume works, every Volume is associated with a Book. The Book object contains the MARC XML metadata; each Volume includes a PDF, and may include OCR XML. For Volumes with

pages loaded, each page is stored as an individual object with a relation to the parent volume. Any volumes with a cover loaded will have at least one page, with the special hasPrimaryImage relationship to indicate which page image should be used as the cover or for thumbnails (this may or may not be the first page). The book-level object is not currently directly exposed in Readux, but it is used to associate volumes with collections, and volumes from the same book are linked as "related volumes" from each individual volume landing page.

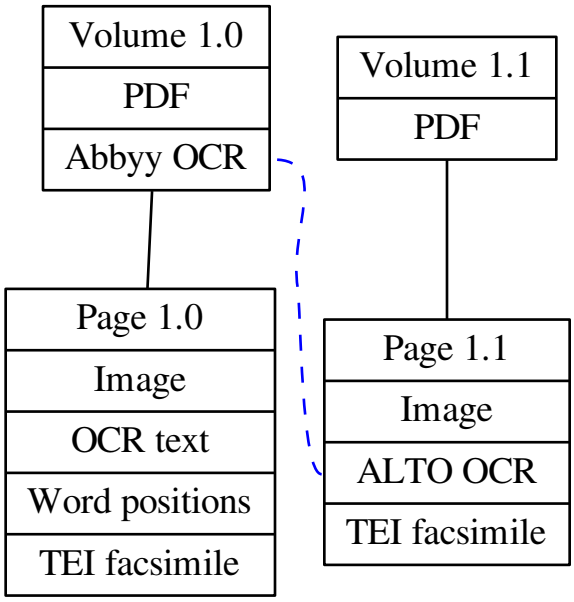
Pages are ordered within a volume using a pageOrder property set in the RELS-EXT of each page.

For implementation specifics, see code documentation for:

- `readux.books.models.Book`
- `readux.books.models.Volume`
- `readux.books.models.Page`

Volume and Page variants

Readux currently includes two different variants of `Volume` and `Page` objects. The primary difference is that the ScannedVolume-1.0 objects contain a single ABBYY OCR XML file with the OCR for the entire volume, where the ScannedVolume-1.1 objects have no volume-level OCR, but each page has a METS/ALTO OCR XML file, instead of the plain text OCR content present in the ScannedPage-1.0 objects.



Readux uses TEI facsimile to provide a consistent format for positioned OCR text data across these variations. Readux includes scripts and XSLT to generate TEI from the volume-level ABBYY OCR or the page-level ALTO, and adds the TEI to the page object in Fedora. In addition, Readux adds xml ids to the original OCR XML, which is carried through to the TEI and then the HTML displayed on the Readux site for annotation, in order to ensure durability and correlation of content with annotations.

Fedora pids

Readux is intended for display and access, and not as a management tool. However, for historical reasons it currently includes some scripts for importing covers and pages, and also a preliminary script for importing a new Volume-1.1 work with pages and metadata (see *import_volume*). Prior to Readux, existing Emory Libraries digitized book content in the repository only included Book and Volume records. There are manage commands to *import_covers* and *import_pages*, but the current implementation uses a legacy Digitization Workflow (`readux.books.digwf`).

Following our standard practice, any objects ingested via Readux have Archival Resource Keys (ARKs) generated via our PID manager application, which are then used as the basis for Fedora object pids. The ARK is stored in the object metadata and displayed on the website as a permalink.

CHAPTER 6

Code Documentation

Collection

Django app for collection-level content and access. Provides `eulfedora.models.DigitalObject` models for simple repository collection objects, Solr result object for collections, and a database model for associating thumbnail and banner images with collections, and views to browse available collections and volumes by collection.

Models

Views

Books

Contents

Django app for book, volume and page content and access. Provides `eulfedora.models.DigitalObject` models for books and two variants of volume and page objects.

For more information on the Fedora content modeling and object relations, see *Book Content Models* and *Volume and Page variants*.

Models

Views

DigWF

Client and `XmlObject` classes to interact with the Digitization Workflow API in order to retrieve information about Large Scale Digitization Initiative (LSDI) content.

Note: DigWF is a legacy application (no longer in use) that contains data about existing Emory Readux volumes. The DigWF API was used to import cover images and selected page images.

---

class `readux.books.digwf.`**`Client`**(*url*)

A simple client to query the Digitization Workflow REST(ish) API for information about files associated with LSDI/DigWF items.

Parameters **`baseurl`** – base url of the api for the DigWF REST service., e.g. `http://my.domain.com/digwf_api/`

`get_items`(***kwargs*)

Query the DigWF API getItems method. If no search terms are specified, getItems returns any items that are in the **Ready for Repository** state. Any keyword arguments will be passed to getItems as query arguments. Currently supports:

•control_key (e.g., ocm or ocn number) - may match more than one item

•item_id - the item id for the record in the DigWF

•pid - the noid portion of the pid/ARK for the item

Returns *Items*

class `readux.books.digwf.`**`Item`**(*node=None*, *context=None*, ***kwargs*)

`XmlObject` to read Item information returned by the DigWF API.

(Not all fields provided by DigWF are mapped here; only those currently in use.)

`pid` **= <eulxml.xmlmap.fields.StringField>**

pid (noid portion of the ARK or Fedora pid)

item_id = <eulxml.xmlmap.fields.StringField>
item_id within the DigWF

control_key = <eulxml.xmlmap.fields.StringField>
control key (e.g., ocm or ocn number in euclid; unique per book, not per volume)

display_image_path = <eulxml.xmlmap.fields.StringField>
display image path

ocr_file_path = <eulxml.xmlmap.fields.StringField>
path to OCR files (text & word position)

pdf = <eulxml.xmlmap.fields.StringField>
path to PDF file

class readux.books.digwf.**Items**(*node=None*, *context=None*, ***kwargs*)
XmlObject for the response returned by getItems. Has a count of the number of items found, and a list of *Item* objects with details about each item.

count = <eulxml.xmlmap.fields.IntegerField>
number of items in the result

items = <eulxml.xmlmap.fields.NodeListField>
List of items as instances of *Item*

TEI

class readux.books.tei.**TeiBase**(*node=None*, *context=None*, ***kwargs*)
Base class for all TEI objects, with all namespaces

class readux.books.tei.**Graphic**(*node=None*, *context=None*, ***kwargs*)
TEI Graphic

url = <eulxml.xmlmap.fields.StringField>
url

rend = <eulxml.xmlmap.fields.StringField>
rend

class readux.books.tei.**Zone**(*node=None*, *context=None*, ***kwargs*)
XmlObject for a zone in a TEI facsimile document

id = <eulxml.xmlmap.fields.StringField>
xml id

n = <eulxml.xmlmap.fields.StringField>
n attribute

type = <eulxml.xmlmap.fields.StringField>
type attribute

ulx = <eulxml.xmlmap.fields.FloatField>
upper left x coord

uly = <eulxml.xmlmap.fields.FloatField>
upper left y coord

lrx = <eulxml.xmlmap.fields.FloatField>
lower right x coord

lry = <eulxml.xmlmap.fields.FloatField>
lower right y coord

href = <eulxml.xmlmap.fields.StringField>
xlink href

text = <eulxml.xmlmap.fields.StringField>
text content

word_zones = <eulxml.xmlmap.fields.NodeListField>
list of word zones contained in this zone (e.g., within a textLine zone)

preceding = <eulxml.xmlmap.fields.NodeField>
nearest preceding sibling word zone (e.g., previous word in this line), if any)

parent = <eulxml.xmlmap.fields.NodeField>
nearest ancestor zone

page = <eulxml.xmlmap.fields.NodeField>
containing page

graphics = <eulxml.xmlmap.fields.NodeListField>
list of graphic elements (i.e. page images)

full_image = <eulxml.xmlmap.fields.NodeField>
full size image (tei:graphic with type "full")

page_image = <eulxml.xmlmap.fields.NodeField>
page size image (tei:graphic with type "page")

thumbnail = <eulxml.xmlmap.fields.NodeField>
thumbnail image (tei:graphic with type "thumbnail")

small_thumbnail = <eulxml.xmlmap.fields.NodeField>
small thumbnail image (tei:graphic with type "small-thumbnail")

image_info = <eulxml.xmlmap.fields.NodeField>
image info as provided by IIIF (tei:graphic with type "info")

width
zone width

height
zone height

avg_height
Calculated average height of word zones in the current zone (i.e. in a text line)

class readux.books.tei.**Ref**(*node=None*, *context=None*, ***kwargs*)
Tei reference

target = <eulxml.xmlmap.fields.StringField>
target

type = <eulxml.xmlmap.fields.StringField>
type

text = <eulxml.xmlmap.fields.StringField>
text

class readux.books.tei.**BiblStruct**(*node=None*, *context=None*, ***kwargs*)
Structured Bibliographic citation

id = <eulxml.xmlmap.fields.StringField>
xml id

corresp = <eulxml.xmlmap.fields.StringField>
corresp

type = <eulxml.xmlmap.fields.StringField>
type

class readux.books.tei.**Note**(*node=None*, *context=None*, ***kwargs*)
Tei Note, used here to contain an annotation

id = <eulxml.xmlmap.fields.StringField>
xml id

resp = <eulxml.xmlmap.fields.StringField>
responsibility

target = <eulxml.xmlmap.fields.StringField>
target

type = <eulxml.xmlmap.fields.StringField>
type

ana = <eulxml.xmlmap.fields.StringField>
ana attribute, e.g. for tag identifiers

href = <eulxml.xmlmap.fields.StringField>
xlink href

paragraphs = <eulxml.xmlmap.fields.StringListField>
list of paragraphs as strings

markdown = <eulxml.xmlmap.fields.StringField>
code for the markdown used in the original annotation

related_pages = <eulxml.xmlmap.fields.NodeListField>
links to related pages

citations = <eulxml.xmlmap.fields.NodeListField>
list of bibliographic citations/works cited

class readux.books.tei.**Bibl**(*node=None*, *context=None*, ***kwargs*)
TEI Bibl, with mappings for digital edition and pdf urls

type = <eulxml.xmlmap.fields.StringField>
type

title = <eulxml.xmlmap.fields.StringField>
title

authors = <eulxml.xmlmap.fields.StringListField>
author

date = <eulxml.xmlmap.fields.StringField>
date

url = <eulxml.xmlmap.fields.StringField>
url to digital edition

pdf_url = <eulxml.xmlmap.fields.StringField>
url to pdf of digital edition

class readux.books.tei.**PublicationStatement**(*node=None*, *context=None*, ***kwargs*)
Publication statement, with mapping for readux distributor

desc = <eulxml.xmlmap.fields.StringField>
descriptive statement (paragraph)

date = <eulxml.xmlmap.fields.DateField>
date in human-readable display format

date_normal = <eulxml.xmlmap.fields.DateField>
normalized date

distributor_readux = <eulxml.xmlmap.fields.StringField>
readux distributor reference (includes ref with target of readux.library.emory.edu)

class readux.books.tei.**Facsimile**(*node=None*, *context=None*, ***kwargs*)
Extension of eulxml.xmlmap.teimap.TEI to provide access to TEI facsimile elements

XSD_SCHEMA = 'file:///home/docs/checkouts/readthedocs.org/user_builds/readux/checkouts/develop/readux/books/schema
local xsd schema

page = <eulxml.xmlmap.fields.NodeField>
surface with type page, as *Zone*

page_list = <eulxml.xmlmap.fields.NodeListField>
list of pages (surface with type page)

lines = <eulxml.xmlmap.fields.NodeListField>
list of zones with type textLine or line as *Zone*

word_zones = <eulxml.xmlmap.fields.NodeListField>
list of word zones (type string) as *Zone*

distributor = <eulxml.xmlmap.fields.StringField>
publication statment distributor

pubstmt = <eulxml.xmlmap.fields.NodeField>
publication statmnt as *PublicationStatement*

encoding_desc = <eulxml.xmlmap.fields.NodeField>
encoding description

original_source = <eulxml.xmlmap.fields.NodeField>
source description for the original volume

digital_source = <eulxml.xmlmap.fields.NodeField>
source description for the readux digital edition

class readux.books.tei.**Name**(*node=None*, *context=None*, ***kwargs*)
Tei NAME, with id attribute and value

id = <eulxml.xmlmap.fields.StringField>
xml id

value = <eulxml.xmlmap.fields.StringField>
full name

class readux.books.tei.**AnnotatedFacsimile**(*node=None*, *context=None*, ***kwargs*)
Annotated Tei facsimile, with mappings needed to generate TEI with annotations.

main_title = <eulxml.xmlmap.fields.StringField>
main tei title

subtitle = <eulxml.xmlmap.fields.StringField>
tei subtitle (e.g., annotated edition)

responsibility = <eulxml.xmlmap.fields.StringField>
responsibility statement text

responsible_names = <eulxml.xmlmap.fields.NodeListField>
responsibility statement names

annotations = <eulxml.xmlmap.fields.NodeListField>
list of annotations at body/div[@type="annotations"]/note[@type="annotation"], as *Note*

citations = <eulxml.xmlmap.fields.NodeListField>
list of bibliographic citations/works cited

citation_ids = <eulxml.xmlmap.fields.StringListField>
list of bibliographic citation ids

tags = <eulxml.xmlmap.fields.NodeField>
annotation tags, as `TeiInterpGroup`

class `readux.books.tei.`**Anchor**(*node=None*, *context=None*, ***kwargs*)
TEI Anchor, for marking start and end of text annotation highlights

id = <eulxml.xmlmap.fields.StringField>
xml id

type = <eulxml.xmlmap.fields.StringField>
type

next = <eulxml.xmlmap.fields.StringField>
next attribute

Annotate

Markdown to TEI

`readux.books.markdown_tei.`**convert**(*text*)
Render markdown text as simple TEI. Does not include namespaces or wrapping elements; assumes that the rendered markdown will be inserted into a TEI document as text content, and that it is not intended to be an entire, valid document on its own.

class `readux.books.markdown_tei.`**TeiMarkdownRenderer**(***kwargs*)
TEI Markdown renderer for use with `mistune` markdown parsing and rendering library. Renderer is based on the built-in mistune HTML renderer.

audiovideo_ext_mimetype = {'.aac': 'aac', '.webm': 'webm', '.mpeg': 'mpeg', '.m4a': 'mp4', '.mpg': 'mpeg', '.wav
common html5 audio file extensions and corresponding mimetypes; used to infer audio mimetype when it is not specified

classmethod preprocess(*text*)
Method to preprocess text to make sure it is converted properly. Currently, adds whitespace to ensure that any audio tags will be processed as an html block.

block_code(*code*, *lang=None*)
Rendering block level code.

Parameters

- **code** – text content of the code block.
- **lang** – language of the given code.

block_quote(*text*)
Rendering <quote> with the given text.

Parameters **text** – text content of the blockquote.

block_html(*html*)
Rendering block level pure html content. Currently only supports html5 audio tags.

Parameters **html** – text content of the html snippet.

header(*text*, *level*, *raw=None*)
Rendering header/heading.

Parameters

- **text** – rendered text content for the header.
- **level** – a number for the header level, for example: 1.
- **raw** – raw text content of the header.

hrule()
Rendering method for horizontal rule.

list(*body*, *ordered=True*)
Rendering list tags.

Parameters

- **body** – body contents of the list.
- **ordered** – whether this list is ordered or not.

list_item(*text*)
Rendering list item.

paragraph(*text*)
Rendering paragraph tags. Like <p>.

table(*header*, *body*)
Rendering table element. Wrap header and body in it.

Parameters

- **header** – header part of the table.
- **body** – body part of the table.

table_row(*content*)
Rendering a table row.

Parameters **content** – content of current table row.

table_cell(*content*, ***flags*)
Rendering a table cell.

Parameters

- **content** – content of current table cell.
- **header** – whether this is header or not.
- **align** – align of current table cell.

double_emphasis(*text*)
Rendering **strong** text.

Parameters **text** – text content for emphasis.

emphasis(*text*)
Rendering *emphasis* text.

Parameters **text** – text content for emphasis.

codespan(*text*)
Rendering inline *code* text.

Parameters **text** – text content for inline code.

linebreak()
Rendering line break.

strikethrough(*text*)
Rendering ~~strikethrough~~ text.

Parameters **text** – text content for strikethrough.

text(*text*)
Rendering unformatted text.

Parameters **text** – text content.

autolink(*link*, *is_email=False*)
Rendering a given link or email address.

Parameters

- **link** – link content or email address.
- **is_email** – whether this is an email or not.

link(*link*, *title*, *text*)
Rendering a given link with content and title.

Parameters

- **link** – href link for `<a>` tag.
- **title** – title content for *title* attribute.
- **text** – text content for description.

image(*src*, *title*, *text*)
Rendering a image with title and text.

Parameters

- **src** – source link of the image.
- **title** – title text of the image.
- **text** – alt text of the image.

inline_html(*html*)
Rendering span level pure html content.

Parameters **html** – text content of the html snippet.

newline()
Rendering newline element.

footnote_ref(*key*, *index*)
Rendering the ref anchor of a footnote.

Parameters

- **key** – identity key for the footnote.

- **index** – the index count of current footnote.

footnote_item(*key*, *text*)
Rendering a footnote item.

Parameters

- **key** – identity key for the footnote.
- **text** – text content of the footnote.

footnotes(*text*)
Wrapper for all footnotes.

Parameters **text** – contents of all footnotes.

Export

GitHub

Consumers

Custom manage commands

The following management commands are available. For more details, use `manage.py help <command>`. As much as possible, all custom commands honor the built-in django verbosity options.

import_covers

import_pages

update_arks

import_volume

Annotation

An implementation of the Annotation store API for Annotator.js.

Models

Views

Fedora

Utilities

Consumers

CHAPTER 7

CHANGELOG

Release 1.8.1 - Bug Fixes * Annotation counts do not appear in Volume list * Option to export personal versus group annotations is missing

Release 1.8 - WebSocket 1.x Upgrade, Move to ECDS VM

- Chore: Move Readux to New Prod server
- Chore: As a Readux developer, I would like to replace the Social Auth OAuth Key and Secret with Emory University Library accounts.
- As a Readux admin, I would like to have a management command to make post-1922 Emory Yearbooks not visible in Readux
- Upgrade Daphne and related components from 0.x versions to 1.x fully stable versions
- Chore: Research related to beginning Readux 2.x development and setting up a new AWS testing environment
- Bugfix: Volume landing page does not display
- Chore: Add lxml rebuild to the fabfile for deploy
- Chore: Add missing tei page to emory:dzj14 and emory:cwv8v on Readux production site and create KB article for this issue
- Chore: Make a digitaledition_jekylltheme 0.7 release that is dependent on by the Readux 1.7 Release
- Chore: Fix pip install on Jenkins asking for user interaction
- Bugfix: Export issue: Pages with no XML in volume “Original sacred harp” are excluded from export

Release 1.7 - Export Notifications, Export Search Improvements, Bug Triage

- As a site administrator, I want to embed collections into editable pages that can be ordered randomly or alphabetically so that I can introduce users to the site
- Bugfix: Display issue in Readux export: Annotation highlighting cut off on oblong pages
- Bugfix: Display issue in Readux export: Right side of pages cut off on oblong pages
- Bugfix: Display issue in Readux export: Vertical page inelegantly displays in box for vertical pages in oblong books
- Bugfix: Image thumbnails show as broken links on "Annotations tagged with..." pages in exported editions
- Bugfix: Deep Zoom images are not loaded in the recent exports, even when images are hosted independently and deep zoom is included in the export
- As a scholarly edition author I can use a dynamic form that breaks the export process up into logical steps so that I understand the choices I am making when exporting a volume and avoid making incompatible choices
- Bugfix: Jekyll site website packages downloaded in the testreadux environment do not display page image thumbnails when uploaded to GitHub
- Bugfix: Image thumbnails on the "browse pages" page and images for individual pages served from Readux show as broken links in exported editions
- Bugfix: Top level navigation should include "Collections", "About", "Annotation", "Export", and "Credits."
- As a scholarly edition author I can start the generation of my edition on my computer, run in the background, and receive notification when it is ready, so that I can do other things while it is generating and downloading
- As a scholarly edition author I want the option to exclude deep zoom from my website package so that I can display my edition without having to connect to Readux
- Bugfix: clicking on export button causes error when logged in via LDAP
- As a scholarly edition author I want to the option to include deep zoom images in my website package so that I can display my edition without having to connect to Readux
- Bugfix: Fix CSS/Jquery issue in GitHub export
- Bugfix: No padding on simple pages
- Bugfix: Editable pages cannot include span tags (for COinS) and html source cannot be edited
- As a site administrator, I want the Readux home page to be editable and configurable so that I can display custom-designed text and collections to introduce users to the site
- As a scholarly edition author I want the option to download page images with my website package so that I can display my edition without having to connect to Readux
- As a scholarly edition author I want users to to see an index based on annotation tags that includes a count of annotation and page numbers so that they can see how tags are used across the volume
- As a user I want to see the option of downloading the TEI in the main export form so that I can choose between all export options on the same webpage
- As an annotated edition author I want users to be able to keep track of applied filters and search keywords in the URL so that each search could be referenced in the future or shared with another person
- As an annotated edition author I want users to be able to facet annotation searches by tags so that they can more easily identify relevant content

- As an annotated edition author I want users to be able to facet searches by annotation and page content so that they can more easily identify relevant content
- As an annotated edition author I want users to be able to search my edition partial word matching so that they can more easily identify relevant content
- As a scholarly edition author I can refresh my website if I make changes to the annotations on Readux, without overwriting my original local customizations, so that I can create an updated copy of my content
- As an annotated edition author I want my site to include a sitemap for volume pages, annotations and other content so that my site will be findable by search engines
- New **IIIF_ID_SUFFIX** configuration option for IIIF image server (#4 via @ghukill)
- OCR to TEI facsimile now supports output from ABBYY Recognition Server (#4 via @ghukill)

Release 1.6.1

- Require eulfedora 1.6 or greater for debug filter and connection retries

Release 1.6 - Group Annotation

- As a site administrator I want to create and manage annotation groups of existing users so that I can support group annotation projects.
- As a logged in user I want to see annotations shared with groups I belong to so that I can collaborate with other members of those groups.
- As a logged in user when I am making an annotation I want to grant annotation groups access to read, edit, or delete my annotation so that I can collaborate with group members.
- As a logged in user, I want to see an indication when an annotation is shared with a group.
- As a logged in user, I want to see who authored an annotation so that I can easily distinguish my annotations from those shared with groups I belong to.
- As a logged in user, I can only update annotation permissions if I have the admin annotation permission, so that full access to editing and deleting annotations can be controlled.
- As a logged in user when I export a volume I want to choose between exporting only my annotations or all annotations in a group I belong to so that I can export an individual or collaborative project.
- Now using django-guardian for per-object annotation permissions.
- Includes a new annotator permissions Javasscript module (included in readux codebase for now).
- Data migrations to clean up redundant data in annotation extra data JSON field and grant default annotation author permissions.

Release 1.5.1

- Reference final released versions of annotator-meltdown and annotator-meltdown-zotero

Release 1.5 - Enhanced Annotation

- As a researcher I want to make internal references to other pages in order to show connections to other parts of the same work.
- As a researcher I want to include audio in my annotations so I can demonstrate audible differences in the content.
- As a researcher I want to include video in my annotations so I can associate enriched media with volume content.
- As a researcher I want to link my Zotero account with my Readux login so that I can add Zotero citations to my annotations.
- As a researcher I want to look up Zotero citations and add them to my annotations in order to show my sources.
- As researcher I want to search the text of my annotations for the volume I am working on in order to find specific notes or content.
- As a site user I want to login in with Emory credentials so that I can easily start making annotations.
- As a user, I can find readux volume pages through a search engine, so I can easily find relevant content.
- TEI export now includes an encoding description in the TEI header.
- bugfix: Annotation window sometimes pops up in the top right of the screen, should hover near highlighted text/image. (Actual fix in annotator-marginalia)
- bugfix: Exported site browse annotations by tag never displays more than one annotation. (Actual fix in digitaledition-jekylltheme)
- Project documentation now includes technical descriptions and diagrams of Fedora object models and readux processes.

Release 1.4.1

- As a Readux admin, I want a record when the export feature is used so that I can find out who is creating exported editions.

Release 1.4 - Basic Export

This release adds the capability to export a single Readux volume with annotations to create a standalone annotated website edition, using Jekyll and with optional GitHub / GitHub Pages integration.

Export functionality

- As an annotated edition author I want to export an edition that has TEI with text, image references, and annotations so that I can have a durable format copy of my edition with my annotation content.
- As an annotated edition author, I want to generate a web site package with volume content and annotations so that I can publish my digital edition.
- As an annotated edition author I want to generate a website package that can be modified so that I can customize my edition.
- As an annotated edition author, I want a website package that allows me to browse pages by thumbnail so that site visitor can easily select a page of interest.

- As an annotated edition author, I want my website edition to include annotation counts for each page so that my site visitors know which pages have annotations.
- As an annotated edition author, I want my website edition to include tags in the annotation display so that my site visitors can see my categorization.
- As an annotated edition author, I want my website edition to support keyword searching so that my site visitors can find content of interest.
- As an annotated edition author, I want to be able to customize my website edition's page urls to match the number in the source text so that my site visitors experience an intuitive navigation of the edition.
- As an annotated edition author, I want the option of creating a new GitHub repository with my exported website edition, so that I can version my data and publish it on GitHub Pages.
- As an annotated edition author, I want my website edition to include citation information so that my site visitors can reference it properly.
- As an annotated edition author, I want to have a copy of the exported TEI in the website bundle so that I can see the data used to generate the web edition.
- As an annotated edition author, I want my website edition to include social media integration so that my site visitors can share content.
- As an annotated edition author, I want my website edition to be viewable on tablets so that my site visitors can view it on multiple devices.
- As an annotated edition author I want my website edition to include individual annotation pages so that users can more easily view and cite long form and multimedia annotation content.

Other updates

- As a site user, I want to link my social login accounts so that I can access annotations from any of my accounts.
- As an annotated edition author, I want to see an error message in the event that I log out while trying to export my edition so that I know I need to be logged in to complete the export.
- As a site user I want to see a permanent url on the pages for volume and single-page so that I can make stable references.
- Update latest 3.x Bootstrap and django-eultheme 1.2.1

Release 1.3.7

- As a site administrator I want to include video content in site pages so that I can share dynamic content like screencasts.

Release 1.3.6

- Improved regenerate-id logic for OCR, use a readux image url when generating page TEI.

Release 1.3.5

- Proxy all IIIF image requests through the application, to handle IIIF server that is not externally accessible.

Release 1.3.4

- bugfix: collection detail pagination navigation
- bugfix: id generation error in OCR/TEI xml
- Improved page mismatch detection when generating TEI from OCR
- Revised placeholder page images for covers and volume browse
- Modify update_page_arks manage command to handle the large number of page arks in production

Release 1.3.3

- bugfix: collection detail pagination display
- bugfix: correct page absolute url, esp. for use in annotation uris

Release 1.3 - Simple Annotation

TEI Facsimile

- As a system administrator, I want to run a script to generate TEI facsimile for volumes that have pages loaded, so that I can work with OCR content in a standardized format.
- As a user I would like to view the TEI underlying the page view and annotation, so that I can understand more about how it works, and to understand how to use facsimile data.
- As a researcher I want to see a view of the TEI underlying the page view and annotation that excludes OCR for barcodes so that I can focus on facsimile data of scholarly importance.

Display improvements

- As a user, I want to navigate from page view to page view without having to scroll down to each page view, so that I have a better reading experience.
- As a user, I can see the thumbnail for landscape pages when browsing volumes, so I can better select appropriate pages.

Annotation

- As a researcher, I want to select the OCR text on a page in order to copy or annotate content.
- As a site user I want to filter volumes by whether or not they have page-level access so that I know which volumes I can read online and annotate.
- As a researcher I can log in to readux using social media credentials, so that I do not need a separate account to create annotations.
- As a researcher I want to annotate part of the text on a page in order to provide additional information about the text.
- As a researcher I want to annotate an image or part of an image in order to provide additional information about the image.

- As a researcher I want to include simple formatting in my notes to make them more readable.
- As a researcher I want to include images in my annotations so that users can see important visual materials.
- As a researcher I want to tag annotations so that I can indicate connections among related content.
- As a researcher I want to edit and delete my annotations, in order to make changes or remove notes I no longer want.
- As a user I can see my annotations in the margin of the page, so that I can read all of the annotations conveniently.
- As a researcher I want to see which volumes I have annotated when I am browsing or searching so that I can easily resume annotating.
- As a researcher I want to see which pages I have annotated so that I can assess the status of my digital edition.
- As a researcher I want to make annotations anchored to stable identifiers that are unique across an entire volume so that I can maintain consistency and generate valid exports in my digital editions.
- As a user I want to see a created or last modified timestamp on annotations so that I know when they were last updated.
- As a user I want to see only the date created or last modified on annotations that are more than a week old so that I know a rough estimate of when they were last updated.

Annotation Administration

- As a site administrator I want to see which user authored an annotation so that I can respond to the correct user in reference to an annotation.
- As a site administrator, I want to view, edit, and delete annotations in the Django admin site so that I can manage annotations to remove spam or update the annotation owner.
- As a site administrator I want to click on the URI link for an annotation in the admin and see the annotated page in a separate window so that I can verify its display.

Additional Administration functionality

- As a site administrator I want to create and edit html pages so that I can add content explaining the site to users.

Release 1.2.2

- Require eulfedora 1.2 for auto-checksum on ingest against Fedora 3.8

Release 1.2.1

- Update required version of django-downtime and eultheme.

Release 1.2 - Fedora 3.8 migration support

- As a site user I will see a Site Down page when maintenance is being performed on the site or or other circumstances that will cause the site to be temporarily unavailable so that I will have an general idea of when I can use the site again.

- As a site user I will see a banner that displays an informative message on every page of the site so that I can be informed about future site maintenance or other events and know an approximate amount of any scheduled downtime.
- As an application administrator, I want to generate a list of pids for testing so that I can verify the application works with real data.
- Any new Fedora objects will be created with Managed datastreams instead of Inline for RELS-EXT and Dublin Core.
- Upgraded to Django 1.7
- Now using *django-auth-ldap <https://pythonhosted.org/django-auth-ldap/>* for LDAP login instead of eullocal.

Release 1.1.2

- Fix last-modified method for search results to work in cover mode.

Release 1.1.1

- Fix volume sitemaps to include both Volume 1.0 and 1.1 content models.

Release 1.1 - Import

- As an administrative user, I want to run a script to import a volume and its associated metadata into the repository so that I can add new content to readux.
- As a user, I want to browse newly imported content and previously digitized content together, so that I can access newly added content in the same way as previously digitized materials.
- As a user I can opt to sort items on a collection browse page by date added, in order to see the newest material at the top of the list, so that I can see what is new in a collection.
- As a user, I want the option to view or download a PDF, with a warning for large files, so that I can choose how best to view the content.
- As an administrative user, I want to be able to run a script to load missing pages for a volume so that I can load all pages when the initial page load was interrupted.

Release 1.0.2

- As a user, I want the website to support caching so I don't have to re-download content that hasn't changed and the site will be faster.
- bugfix: fix indexing error for items with multiple titles
- error-handling & logging for volumes with incomplete or invalid OCR XML
- adjust models to allow eulfedora syncrepo to create needed content model objects

Release 1.0.1

- Include *.TIF in image file patterns searched when attempting to identify page images in **import_covers** and **import_pages** scripts
- Additional documentation and tips for running **import_covers** and **import_pages** scripts
- Bugfix: workaround for pdfminer maximum recursion error being triggered by outline detection for some PDF documents
- Enable custom 404, 403, and 500 error pages based on eultheme styles

Release 1.0 - Page-Level Access

Cover images and list view improvements

- As a researcher, when I'm viewing a list of titles, I want the option to toggle to a cover view as an alternate way to view the content.
- As a user, when I toggle between cover and list views I want to be able to reload or go back in history without needing to reselect the mode I was last viewing, so that the site doesn't disrupt my browsing experience.
- As a user, when I page through a collection or search results, I expect the display to stay in the mode that I've selected (covers or list view), so that I don't have to reselect it each time.

Volume landing page and Voyant improvements

- As a user when I select a title in the list view, I first see an information page about the item, including pdf and page view selections, so that I know more about the item before I access it.
- As a user, I want to be able to see the full title of a book without longer titles overwhelming the page, so I can get to the information I want efficiently.
- As a researcher, I want to pass a text to Voyant for analysis in a way that takes advantage of caching, so that if the text has already been loaded in Voyant I won't have to wait as long.
- As a reseacher, I can easily read a page of text in Voyant, because the text is neatly formatted, so that I can more readily comprehend the text.
- As a user, I can see how large a pdf is before downloading it so that I can make appropriate choices about where and how to view pdfs.
- As a user, when I load a pdf I want to see the first page with content rather than a blank page, so that I have easier access with less confusion.

Page-level access / read online

- As a researcher, I can page through a book viewing a single page at a time in order to allow viewing the details or bookmarking individual pages.
- As a user, when I'm browsing a collection or viewing search results, I can select an option to read the book online if pages are available, so that I can quickly access the content.
- As a researcher, I want the option to view pages as thumbnails to enhance navigation.

- As a researcher, when I'm browsing page image thumbnails I want to see an indicator when there's an error loading an image so that I don't mistake errors for blank pages.
- As a researcher, I want to be able to toggle to a mode where I can zoom in on an image so that I can inspect the features of a page.
- As a user, I want to be able to distinguish when I can and cannot use the zoom function, so I can tell when the feature is unavailable (e.g., due to image load error).
- As a researcher, I want to search within a single book so that I can find specific pages that contain terms relevant to my research.

Navigation improvements

- As a user, I want to see a label or source information for the collection banner image so that I know where the image comes from.
- As a user, I want to be able to identify a resource in an open tab by title, so I can quickly select the correct tab when using multiple tabs.
- As a user, when paging through a collection list or viewing a set of pages in the reading view, I can find the web page navigation at the top or bottom of the page, so that I do not have to scroll far to click to go to another web page in the series.

Integrations with external services

- As a twitter user, when I tweet a link to a readux collection, book, or page image, I want a preview displayed on twitter so that my followers can see something of the content without clicking through.
- As a facebook user, when I share a link to a readux collection, book, or page image, I want a preview displayed on facebook so that my friends can see something of the content without clicking through.
- A search engine crawling the readux site will be able to obtain basic semantic data about collections and books on the site so the search engine's results can be improved.
- A search engine can harvest information about volume content via site maps in order to index the content and make it more discoverable.

Release 0.9 - PDF Access

- As a researcher, I want to browse a list of collections in order to select a subset of items to browse.
- As a researcher, I want to browse through a paginated list of all the books in a single collection in order to see the range of materials that are present.
- As a researcher, when looking at a list of books in a collection, I can view a PDF using my native PDF browser in order to view the contents of the book.
- As a researcher, I can search by simple keyword or phrase in order to find books that fit my interests.
- A search engine can harvest information about site content via site maps in order to index the content and make it more discoverable.
- As a researcher, I can select a text and pass it to Voyant to do text analysis for the purposes of my research.
- As a researcher, I want to be able to harvest contents into my Zotero library in order to facilitate research.

- As a researcher browsing a list of titles in a collection or search results, I want to see the author name and the year of publication so that if I am looking for a particular title or edition I have more information to identify it quickly without opening the pdf.
- As a researcher viewing keyword search results, I want to see titles or authors with matching terms higher in the list so that if I am searching for a title or author by keyword the item appears on the first or second page of results, and I don't have to page through all the results to find what I am looking for.
- As a user I can see a logo for the site, so I visually recognize that I am in a coherent site whenever I see it.
- As a user I see university branding on the site, so that I know that it is an Emory University resource.
- As a user I want to read a brief description of the content of a collection on the collection list page and individual collection pages, so that I can determine my level of interest in it.
- As an admin user, I want to be able to login with my Emory LDAP account so that I can re-use my existing credentials.
- As a user I can view a list of collections on the landing page by thumbnail image so that I can select an area of interest from visual cues.
- As a user, when viewing a single collection, I can see a visual cue of the collection's content, so that I can connect the item I see on the list view to the page I am viewing.
- As a researcher I can filter search results by collection facets, in order to see the material most relevant to my interests.
- As an admin, I can upload images and associate them with collections, so that I can manage thumbnail and splash images displayed on collection browse and display pages.

CHAPTER 8

Developer Notes

Export Dependencies

The digitaledition-jekylltheme dependency is setup so as a minimal python module so it can be installed as a python dependency. It is currently included in *requirements/minimum.txt*, which can be modified to require a branch or tagged version, and the appropriate version is automatically installed as part of the fab deploy process.

However, note that the appropriate version of the teifacsimile-to-jekyll Ruby gem must currently be installed manually.

Useful Queries

Find Volume objects in Fedora that do *not* have a cover image:

```
SELECT ?pid
WHERE {
    ?pid <fedora-model:hasModel> <info:fedora/emory-control:ScannedVolume-1.0> .
    OPTIONAL {
        ?pid <http://pid.emory.edu/ns/2011/repo-management/#hasPrimaryImage> ?img
    }
    FILTER (!BOUND(?img))
}
```

Note: Because Fedora's RISearch currently only supports SPARQL 1.0, this query requires the optional/not bound filter rather than the more straightforward NOT EXISTS that is supported in SPARQL 1.1+.

Other tasks

Sync volume content and annotations between instances

How to sync a volume with all its related content (relevant book and page objects in Fedora) along with a set of annotations between two environments, e.g. between production and QA.

In the source installation of Readux (e.g. production), use the `find_test_pids` script to get a list of volume, book, and page pids. This reequires an input file, so create a file with a list of pids, one per line (only one line if you are only synchronizing one volume):

```
echo "pid" > /tmp/vol_pid.txt
python manage.py find_test_pids /tmp/vol_pid.txt > /tmp/vol_all_pids.txt
```

Use the eulfedora repo-cp script to sync the data between Fedora repositories using the list of pids you generated:

```
repo-cp prod qa --file /tmp/vol_all_pids.txt
```

Use the annotations api from your source installation to find the annotations associated with your volume, e.g.:

> http://readux.library.emory.edu/annotations/api/search?user=<username>

or search by volume uri (available as of readux 1.4):

```
http://readux.library.emory.edu/annotations/api/search?volume_uri=http://readux.
↪library.emory.edu/books/pid:###/
```

Save annotations as JSON and edit to replace the base source urls with destination site urls (e.g., readux.library to testreadux.library; but note that this *must* match the url configured in your Django sites, so if you are using a proxy use that url). Then import the annotations into your destination readux instance:

```
python manage.py import_annotations my_annotations.json
```

Note that this requires equivalent user accounts to exist in both instances (and if a different user happened to have the same username in the second location, you have just given them access to another person's annotations).

Remove post-1922 yearbooks from the Solr index

For historical reasons, the Emory repository includes digitized yearbooks that are not public domain but are inaccurately described as public domain in the metadata. For now, these must be suppressed from discovery within Readux.

Start up a django python shell (`python manage.py shell`) and do the following:

```
from readux.utils import solr_interface
from readux.books.models import VolumeV1_0, SolrVolume
solr = solr_interface()
vols = solr.query(content_model=VolumeV1_0.VOLUME_CONTENT_MODEL,
   collection_id='emory-control:LSDI-EmoryYearbooks').results_as(SolrVolume)
solr.delete([{'pid': vol['pid']} for vol in vols if int(vol.volume[:4]) >= 1923])
solr.query(content_model=VolumeV1_0.VOLUME_CONTENT_MODEL,
   collection_id='emory-control:LSDI-EmoryYearbooks').count()
```

CHAPTER 9

Readux

documentation

code

Readux is a repository-based Django web application for access to digitized books. Readux runs on Fedora Commons 3.8 and Django 1.8. Readux provides search and browse access to all digitized books, citations that can be harvested with Zotero, the ability to send text to Voyant for analysis. For volumes with pages loaded, OCR text is provided as an invisible overlay on page images for selection and search. Logged in users can annotate text and image selections of page images, and can export an annotated volume as TEI or a standalone Jekyll website.

Documentation available at readux.readthedocs.org.

License

This software is distributed under the Apache 2.0 License.

Dependencies

Services

- SQL database for administration, user accounts, collection banner images, and annotation storage
- Fedora repository for access to digitized book content
- Apache Solr for browse by collection, search across all volumes, and search within a single volume
- IIIF image server for page image content; e.g. Loris

Software Dependencies

- Python Social Auth for social authentication, and access to GitHub accounts for annotated volume export
- eulfedora
- django-eultheme
- Annotator.js for annotation
- annotator plugins for marginal display, formatting, and image selection
- teifacsimile-to-jekyll and digitaledition-jekylltheme for exporting annotated editions

Components

`readux.collection` Models and views for access to collections, which are used to group digitized book content

`readux.books` Models and views for access to digizited book content; includes management commands for loading book covers and page content and generating page-level TEI facsimile

`readux.annotations` Django db models and views to provide an annotator-store backend; implements the Annotator Core Storage api.

`readux.pages` Functionality for site content such as an about page, annotation and export documentation, credits. Content management based on FeinCMS.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

U

V

W

X

Z