Entity Authority Tool Set Documentation

Release 3.0

Jamie Norrish

Jun 26, 2018

Contents

EATS is a web application for recording, editing, using and displaying authority information about entities. It is designed to allow for multiple authorities to each maintain their own independent data, while operating on a common base that means information about the same entity is all in one place.

EATS also comes with client tools for automatically looking up entities by name and adding TEI markup.

A couple of papers, based on an old version of EATS but still largely applicable, are: An introduction to EATS and Topic Maps and Entity Authority Records: an Effective Cyber Infrastructure for Digital Humanities.

CHAPTER 1

Contents

1.1 Installation and Setup

1.1.1 Prerequisites

EATS is a Django application, and depends on other Django applications, django-tmapi, django-selectable, ddh_django_utils, and the Python XML library lxml. It also uses django-webtest for its view tests.

EATS works with Django versions 1.8 and later, and requires Python 3.4 or later.

1.1.2 Project settings

EATS uses Django's built-in sites framework as the source for the URLs it associates with the entities it creates. Set the domain name appropriately.

The URL for the Topic Map that underpins EATS must be set in the Django project settings as EATS_TOPIC_MAP.

The number of search results per page can be specified as EATS_RESULTS_PER_PAGE.

The number of extra forms to supply for each property assertion on edit entity pages may be customised using the following settings: EATS_EXTRA_EXISTENCE_FORMS, EATS_EXTRA_ENTITY_TYPE_FORMS, EATS_EXTRA_NAME_FORMS, EATS_EXTRA_NAME_PART_FORMS, EATS_EXTRA_ENTITY_RELATIONSHIP_FORMS, EATS_EXTRA_NOTE_FORMS, and EATS_EXTRA_SUBJECT_IDENTIFIER_FORMS.

1.2 Getting Started

1.2.1 Administration

After installing and setting up EATS, there are four steps to getting everything in place to create entities. These are all done via the administration interface at `/administer/` (this is *not* the Django admin interface).

- Create the Topic Map that will hold your EATS data.

- Add whatever languages, scripts, name types, entity types, etc, that you wish to use.
- Add one or more authorities, and specify which language, scripts, etc, are available to them.
- Create EATS users (these are existing Django users), and if they are to be able to create and edit entities, add them to each authority for which they are to be an editor.

1.2.2 Creating and editing entities

To add new entities, either use the form at `/entity/add/` or use one of the client tools to generate them from the names in a source text. An entity may be viewed at `/entity/<id>/` and edited at `/entity/<id>/edit/`.

1.3 EATSML: Importing and Exporting

EATS uses an XML format called EATSML for serialisation of its data. Existing data can be exported from EATS, and new data imported into EATS using it.

1.3.1 Export

There are several different exports available, depending on what data is wanted:

- Base export, that exports the infrastructural data (authorities, languages, name types, etc) but no entities. Typically the infrastructural data exported is limited to that relating to the authorities the user is an editor for.

  This is available at `/export/eatsml/base/`.
- Entity export, that exports the entities, along with the infrastructural data that is referenced by those entities.

  This is available at `/export/eatsml/entities/`.
- Full export, that exports all infrastructure data and all entities.

  This is available at `/export/eatsml/full/`.

The EATSML of an export specifies the identifier of a piece of data in the EATS database, in the `eats_id` attribute. This is needed when performing an import that adds new data that reference existing information, such as the authority for a new existence property assertion.

1.3.2 Import

The import of an EATSML document is available at `/import/`.

When importing EATSML that has been exported from a different EATS system, the `eats_id` attributes must be either removed (if the identified data does not exist in the new system), or changed to match the identifiers used in the new system. Otherwise the import will fail because it cannot find the referenced data - or worse, succeed but associate the imported information with the wrong data that just happens to share the same id!

After making an import, the imported EATSML can be viewed either in the form it was imported, or with the appropriate `eats_id` attributes added.

The import process automatically prunes the EATSML of any material that is neither to be added, nor referenced by data that is to be added. Therefore, the EATSML that is displayed for an import may not exactly match the EATSML that was actually sent to the server. This is done to make it easier to see what is added in an import.

1.4 Client programs

EATS comes with a couple of client programs designed to add entity references to TEI XML markup.

1.4.1 Banquet

Banquet is a tool for performing bulk keying of name markup in TEI XML documents. Its user interface presents all of the textual content of name elements in groups, allowing the user to perform a single lookup against an EATS instance and add an entity identifier to all of the selected elements.

Banquet is a Python 2 project, and requires PyGTK and lxml.

1.4.2 oXygen lookup plugin

The lookup client is a plugin for the oXygen XML editor that allows a user to select some text, perform a lookup against an EATS instance using that text, and add in appropriate TEI XML markup reference the selected entity. It handles creating new name markup as well as modifying existing name markup.

Most of the JARs that this plugin requires come bundled with it. The exception is jsoup, available at https://jsoup.org/. The JAR filename specified in plugin.xml must match the actual jsoup JAR filename; this file should be placed in the EATS plugin directory.

1.5 Entity merging

If two entities are found to be duplicates, they may be merged together. This is preferable to simply deleting one, for two reasons:

1. If the entities (may) have both been referenced, then deleting one breaks those references.
2. Any information on the entity to be deleted that is not shared with the other entity must be manually added.

Merging one entity into another avoids both of these problems. The URLs for the merged entity are associated with the other entity, and all property assertions for that entity are merged in with those of the other entity.

The merging of property assertions is simple. If the identical information is present on both entities, and the property assertion is not a name or entity relationship, the duplicate will be discarded; otherwise, it is effectively copied over. This may mean (such as with identical names) that there are duplicates that must be manually deleted, but this is easily done in the editing interface.

Merging is a one way process that cannot be undone. To merge two entities together, use the link at the bottom of the edit page for the entity that will be merged *into*.