

---

# **Readux Documentation**

***Release 0.6.0***

**Emory University Libraries**

**Nov 06, 2017**



---

## Contents

---

<b>1</b>	<b>Code Documentation</b>	<b>3</b>
<b>2</b>	<b>CHANGELOG</b>	<b>9</b>
<b>3</b>	<b>License</b>	<b>11</b>
<b>4</b>	<b>Custom Annotation Model</b>	<b>13</b>
<b>5</b>	<b>Unit Testing</b>	<b>15</b>
<b>6</b>	<b>Sphinx Documentation</b>	<b>17</b>
<b>7</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>



Contents:



## 1.1 Annotation

An implementation of the [Annotation store API](#) for [Annotator.js](#).

### 1.1.1 Models

**class** `annotator_store.models.AnnotationQuerySet` (*model=None, query=None, using=None, hints=None*)

Custom `QuerySet` for *Annotation*

**visible\_to** (*user*)

Return annotations the specified user is allowed to view. Objects are found based on `view_annotation` permission and

annotations; users can access only their own annotations or those where permissions have been granted to a group they belong to.

---

**Note:** Due to the use of `guardian.shortcuts.get_objects_for_user()`, it is recommended to use this method first; it does combine the existing `queryset` query, but it does not chain as `querysets` normally do.

---

**visible\_to\_group** (*group*)

Return annotations the specified group is allowed to view. Objects are found based on `view_annotation` permission and per-object permissions.

---

**Note:** Due to the use of `guardian.shortcuts.get_objects_for_user()`, it is recommended to use this method first; it does combine the existing `queryset` query, but it does not chain as `querysets` normally do.

---

**last\_created\_time()**

Creation time of the most recently created annotation. If queryset is empty, returns None.

**last\_updated\_time()**

Update time of the most recently created annotation. If queryset is empty, returns None.

**class** `annotator_store.models.AnnotationManager`

Custom Manager for *Annotation*. Returns *AnnotationQuerySet* as default queryset, and exposes *visible\_to()* for convenience.

**visible\_to(user)**

Convenience access to *AnnotationQuerySet.visible\_to()*

**visible\_to\_group(group)**

Convenience access to *AnnotationQuerySet.visible\_to\_group()*

**class** `annotator_store.models.BaseAnnotation(*args, **kwargs)`

Django database model to store Annotator.js annotation data, based on the [annotation format documentation](#).

**UUID\_REGEX = '[0-9a-f]{8}-[0-9a-f]{4}-[1-5][0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}'**

regex for recognizing valid UUID, for use in site urls

**schema\_version = 'v1.0'**

annotation schema version: default v1.0

**id**

unique id for the annotation; uses `uuid.uuid4()`

**created**

datetime annotation was created; automatically set when added

**updated**

datetime annotation was last updated; automatically updated on save

**text**

content of the annotation

**quote**

the annotated text

**uri**

URI of the annotated document

**user**

user who owns the annotation when serialized, id of annotation owner OR an object with an 'id' property

**extra\_data**

any additional data included in the annotation not parsed into specific model fields; this includes ranges, permissions, annotation data, etc

**common\_fields = ['text', 'quote', 'uri', 'user']**

fields in the db model that are provided by annotation json when creating or updating an annotation

**internal\_fields = ['updated', 'created', 'id', 'user', 'annotator\_schema\_version']**

internal fields that are not set from values provided by annotation json when creating or updating

**get\_absolute\_url()**

URL to view this annotation within the annotation API.

**text\_preview()**

Short preview of annotation text content

**uri\_link()**

URI as a clickable link



**related\_pages**

convenience access to list of related pages in extra data

**classmethod create\_from\_request** (*request*)

Initialize a new *Annotation* based on data from a `django.http.HttpRequest`.

Expects request body content to be JSON; sets annotation user based on the request user.

**update\_from\_request** (*request*)

Update attributes from data in a `django.http.HttpRequest`. Expects request body content to be JSON. Currently does *not* modify user.

**handle\_extra\_data** (*data*, *request*)

Handle any “extra” data that is not part of the stock annotation data model. Use this method to customize the logic for creating and updating annotations from request data.

NOTE: request is passed in to support permissions handling when object-level permissions are enabled.

**info** ()

Return a `collections.OrderedDict` of fields to be included in serialized JSON version of the current annotation.

**class** `annotator_store.models.AnnotationWithPermissions` (*\*args*, *\*\*kwargs*)

Mix-in annotation class to provide object-level permissions handling via django-guardian

**permission\_to\_codename** = {'read': 'view\_annotation', 'admin': 'admin\_annotation', 'update': 'change\_annotation'}

map annotator permissions to django annotation permission codenames

**codename\_to\_permission** = {'change\_annotation': 'update', 'delete\_annotation': 'delete', 'view\_annotation': 'read'}

lookup annotation permission mode by django permission codename

**info** ()

Update default annotation info to include permissions

**save** (*\*args*, *\*\*kwargs*)

Extend default save method to ensure annotation user has access to edit and update their own annotation.

**handle\_extra\_data** (*data*, *request*)

Handle any “extra” data that is not part of the stock annotation data model. Use this method to customize the logic for updating an annotation from request data.

**user\_permissions** ()

Queryset of `guardian.model.UserObjectPermission` objects associated with this annotation.

**group\_permissions** ()

Queryset of `guardian.model.GroupObjectPermission` objects associated with this annotation.

**assign\_permission** (*permission*, *entity*)

Wrapper around `guardian.shortcuts.assign_perm()`. Gives the specified permission to the specified user or group on the current object.

**db\_permissions** (*permissions*)

Convert annotation permission data into actionable django permissions using guardian per-object permissions.

**permissions\_dict** ()

Convert stored guardian per-object permissions into annotation permission dictionary format

**user\_has\_perm** (*user*, *permission*)

Check if a user has a specific permission on this object.

**class** `annotator_store.models.AnnotationGroup` (*\*args*, *\*\*kwargs*)

Annotation Group; extends `django.contrib.auth.models.Group`.

Intended to facilitate group permissions on annotations.

**notes**

optional notes field

**created**

datetime annotation was created; automatically set when added

**updated**

datetime annotation was last updated; automatically updated on save

**class** `annotator_store.models.Annotation` (*id, created, updated, text, quote, uri, user, extra\_data*)

## 1.1.2 Views

**class** `annotator_store.views.AnnotationIndex` (*\*\*kwargs*)

Annotator store API index view, with information and links for API urls.

**class** `annotator_store.views.Annotations` (*\*\*kwargs*)

API annotations view.

On GET, lists annotations. On AJAX POST with json data in request body, creates a new annotation.

Users must be logged in to create new annotations, and can only view their own annotations.

**get** (*request*)

List viewable annotations as JSON.

**post** (*\*args, \*\*kwargs*)

Create a new annotation via AJAX.

**class** `annotator_store.views.AnnotationView` (*\*\*kwargs*)

Views for displaying, updating, and removing a single `Annotation`. All views require that the user be logged in and own the annotation being viewed, updated, or deleted.

**get** (*request, id*)

Display the JSON information for the requested annotation.

**put** (*request, id*)

Update the annotation via JSON data posted by AJAX.

**delete** (*request, id*)

Remove the annotation. On success, returns a 204 No Content response as per the annotator store API documentation.

**class** `annotator_store.views.AnnotationSearch` (*\*\*kwargs*)

Search annotations and display as JSON. Results are restricted to annotations the users has permission to view (currently only annotations owned by the user for everyone other than superusers).

**The following search fields are currently supported:**

- `uri` (exact match)
- `text` (case-insensitive partial match)
- `quote` (case-insensitive partial match)
- `user` (exact match on username)
- `keyword`: case-insensitive partial match on `text`, `quote`, or with extra data (e.g., to match tags)

Search results can be limited by specifying `limit` or `offset` parameters.

## 1.2 Utils

`annotator_store.utils.absolutize_url(local_url)`

Convert a local url to an absolute url, with scheme and server name, based on the current configured `Site`.

**Parameters** `local_url` – local url to be absolutized, e.g. something generated by `reverse()`

`annotator_store.utils.user_passes_test_ajax_403(test_func, login_url=None, redirect_field_name='next')`

Variation on `django.contrib.auth.decorators.user_passes_test()`. If the user is already logged in but has insufficient privileges, returns a 403 Forbidden response because redirecting to log in is not useful. If request is ajax user is not authenticated, returns a 401 because redirecting an ajax request to login is also not useful.

Partially adapted from/inspired by `eulcommon` implementation. <http://eulcommon.readthedocs.io/en/0.17.0/djangoextras.html#module-eulcommon.djangoextras.auth>

`annotator_store.utils.permission_required(perm, login_url=None)`

Version of `django.contrib.auth.decorators.permission_required()` that uses `user_passes_test_with_ajax_403()` for better http response codes and ajax handling.



### 2.1 Release 0.6

- Creating new annotations via annotator API now requires the user to have the `annotator_store.add_annotation` permission. (Previously any logged in user could create annotations.)
- Any annotation edits (create, update, delete) made via annotator API now generate Django admin log entries.

### 2.2 Release 0.5

Initial release of `annotator_store` as a stand-alone Django application, refactored out of the `Readux` codebase.

- Support for custom Annotation model via `ANNOTATOR_ANNOTATION_MODEL` setting and `annotator_store.models.BaseAnnotation` abstract model
- Configurable support for normal Django permissions *or* per-item permissions via **django-guardian** (*NOTE* that per-item permissions functionality is not fully tested or supported and should be considered alpha quality)
- Annotation text and quote fields marked as optional for use in Django admin
- Supports both Python 3.x and Python 2.7
- Includes example templates with code for initializing an `annotator.js` instance and connecting it to `annotator_store` as a backend.

**annotator\_store** is a `Django` application meant for use within a Django project as an `annotator.js` 2.x annotation store backend, and implements the `Annotator Storage API`.

**annotator\_store** was originally develop as a component of `Readux`.



This software is distributed under the Apache 2.0 License.

### 3.1 Installation

Use pip to install:

```
pip install annotator_store
```

You can also install from GitHub. Use branch or tag name, e.g. @develop or @1.0, to install a specific tagged release or branch:

```
pip install git+https://github.com/Princeton-CDH/django-annotator-store.git@develop  
↪#egg=annotator_store
```

### 3.2 Configuration

Add *annotator\_store* to installed applications and make sure that other required components are enabled:

```
INSTALLED_APPS = (  
    ...  
    'django.contrib.auth',  
    'django.contrib.admin',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
    'annotator_store',  
    ...  
)
```

Include the annotation storage API urls at the desired base url with the namespace:

```
from annotator_store import views as annotator_views

urlpatterns = [
    # annotations
    url(r'^annotations/api/', include('annotator_store.urls', namespace='annotation-
↪api')),
    # annotatorjs doesn't handle trailing slash in api prefix url
    url(r'^annotations/api', annotator_views.AnnotationIndex.as_view(), name=
↪'annotation-api-prefix'),
]
```

Run migrations to create annotation database tables:

```
python manage.py migrate
```

---

**Note:** If you want per-object permissions on individual annotations (rather than the standard django type-based permissions), you must also install *django-guardian* and include *guardian* in your **INSTALLED\_APPS**. Per-object permissions must be turned on in Django settings by setting **ANNOTATION\_OBJECT\_PERMISSIONS** to True.

---



---

## Custom Annotation Model

---

This module is designed to allow the use of a custom Annotation model, in order to add functionality or relationships to other models within an application. To take advantage of this feature, you should extend the abstract model *annotator\_store.models.BaseAnnotation* and configure your model in Django settings, e.g.:

```
ANNOTATOR_ANNOTATION_MODEL = 'myapp.LocalAnnotation'
```

If you want per-object permissions on your annotation model, you should extend *annotator\_store.models.AnnotationWithPermissions* rather than the base annotation class.

---

**Note:** Per-object permissions require that a [permissions plugin](#) be included when you initialize your *annotator.js* Annotator object. That code is currently available as a plugin in the [Readux codebase](#)

---

### 4.1 Development instructions

This git repository uses [git flow](#) branching conventions.

Initial setup and installation:

- recommended: create and activate a python virtualenv:

```
virtualenv annotator-store  
source annotator-store/bin/activate
```

- pip install the package with its python dependencies:

```
pip install -e .
```



Unit tests are run with `py.test` but use Django test classes for convenience and compatibility with django test suites. Running the tests requires a minimal settings file for Django required configurations.

- Copy sample test settings and add a **SECRET\_KEY**:

```
cp ci/testsettings.py testsettings.py
```

- To run the tests, either use the configured `setup.py test` command:

```
python setup.py test
```

- Or install test requirements and use `py.test` directly:

```
pip install -e '.[test]'  
py.test
```



## CHAPTER 6

---

### Sphinx Documentation

---

- To work with the sphinx documentation, install *sphinx* directly via pip or via:

```
pip install -e '.[docs]'
```

- Documentation can be built in the *docs* directory using:

```
make html
```



## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





### **a**

`annotator_store`, 3  
`annotator_store.models`, 3  
`annotator_store.utils`, 7  
`annotator_store.views`, 6



## A

absolutize\_url() (in module annotator\_store.utils), 7  
 Annotation (class in annotator\_store.models), 6  
 AnnotationGroup (class in annotator\_store.models), 5  
 AnnotationIndex (class in annotator\_store.views), 6  
 AnnotationManager (class in annotator\_store.models), 4  
 AnnotationQuerySet (class in annotator\_store.models), 3  
 Annotations (class in annotator\_store.views), 6  
 AnnotationSearch (class in annotator\_store.views), 6  
 AnnotationView (class in annotator\_store.views), 6  
 AnnotationWithPermissions (class in annotator\_store.models), 5  
 annotator\_store (module), 3  
 annotator\_store.models (module), 3  
 annotator\_store.utils (module), 7  
 annotator\_store.views (module), 6  
 assign\_permission() (annotator\_store.models.AnnotationWithPermissions method), 5

## B

BaseAnnotation (class in annotator\_store.models), 4

## C

codename\_to\_permission (annotator\_store.models.AnnotationWithPermissions attribute), 5  
 common\_fields (annotator\_store.models.BaseAnnotation attribute), 4  
 create\_from\_request() (annotator\_store.models.BaseAnnotation class method), 5  
 created (annotator\_store.models.AnnotationGroup attribute), 6  
 created (annotator\_store.models.BaseAnnotation attribute), 4

## D

db\_permissions() (annotator\_store.models.AnnotationWithPermissions

method), 5

delete() (annotator\_store.views.AnnotationView method), 6

## E

extra\_data (annotator\_store.models.BaseAnnotation attribute), 4

## G

get() (annotator\_store.views.Annotations method), 6  
 get() (annotator\_store.views.AnnotationView method), 6  
 get\_absolute\_url() (annotator\_store.models.BaseAnnotation method), 4  
 group\_permissions() (annotator\_store.models.AnnotationWithPermissions method), 5

## H

handle\_extra\_data() (annotator\_store.models.AnnotationWithPermissions method), 5  
 handle\_extra\_data() (annotator\_store.models.BaseAnnotation method), 5

## I

id (annotator\_store.models.BaseAnnotation attribute), 4  
 info() (annotator\_store.models.AnnotationWithPermissions method), 5  
 info() (annotator\_store.models.BaseAnnotation method), 5  
 internal\_fields (annotator\_store.models.BaseAnnotation attribute), 4

## L

last\_created\_time() (annotator\_store.models.AnnotationQuerySet method), 3

last\_updated\_time() (annotator\_store.models.AnnotationQuerySet method), 4

## N

notes (annotator\_store.models.AnnotationGroup attribute), 6

## P

permission\_required() (in module annotator\_store.utils), 7

permission\_to\_codename (annotator\_store.models.AnnotationWithPermissions attribute), 5

permissions\_dict() (annotator\_store.models.AnnotationWithPermissions method), 5

post() (annotator\_store.views.Annotations method), 6

put() (annotator\_store.views.AnnotationView method), 6

## Q

quote (annotator\_store.models.BaseAnnotation attribute), 4

## R

related\_pages (annotator\_store.models.BaseAnnotation attribute), 4

## S

save() (annotator\_store.models.AnnotationWithPermissions method), 5

schema\_version (annotator\_store.models.BaseAnnotation attribute), 4

## T

text (annotator\_store.models.BaseAnnotation attribute), 4

text\_preview() (annotator\_store.models.BaseAnnotation method), 4

## U

update\_from\_request() (annotator\_store.models.BaseAnnotation method), 5

updated (annotator\_store.models.AnnotationGroup attribute), 6

updated (annotator\_store.models.BaseAnnotation attribute), 4

uri (annotator\_store.models.BaseAnnotation attribute), 4

uri\_link() (annotator\_store.models.BaseAnnotation method), 4

user (annotator\_store.models.BaseAnnotation attribute), 4

user\_has\_perm() (annotator\_store.models.AnnotationWithPermissions method), 5

user\_passes\_test\_ajax\_403() (in module annotator\_store.utils), 7

user\_permissions() (annotator\_store.models.AnnotationWithPermissions method), 5

UUID\_REGEX (annotator\_store.models.BaseAnnotation attribute), 4

## V

visible\_to() (annotator\_store.models.AnnotationManager method), 4

visible\_to() (annotator\_store.models.AnnotationQuerySet method), 3

visible\_to\_group() (annotator\_store.models.AnnotationManager method), 4

visible\_to\_group() (annotator\_store.models.AnnotationQuerySet method), 3