$\mathbf{APP}_{NAME}$

Documentation

Release 0.5

me

August 25, 2011

CONTENTS

SuperTagging is an auto-tagging app using OpenCalais, based off of Django Tagging

Contents:

CHAPTER

ONE

INSTALLATION

1.1 Download SuperTagging

There are a couple ways for you to get Django-SuperTagging,

1. Clone the git repository from GitHub
2. Use pip to install it from PyPI

```
pip install supertagging
```

1.2 Dependencies

- simplejson *(Required)*
- freebase *(Optional)*

1.3 Add SuperTagging to your project

Add to INSTALLED_APPS

```
INSTALLED_APPS = (
    ...
    supertagging,
    ...
)
```

Run syncdb:

```
>>> ./manage.py syncdb
```

CHAPTER

TWO

GETTING STARTED

If you have not installed SuperTagging yet, go to the *Installation* page.

2.1 Create basic settings

In your `settings.py` file:

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': True,
    'DEBUG': True,
}
```

2.2 Setting up OpenCalais API

Go to OpenCalais's website and register for an api key, and in your `settings.py` file, alter `SUPERTAGGING_SETTINGS`:

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': True,
    'DEBUG': True,
    'OPEN_CALAIS': {
        'API_KEY': 'YOUR_API_KEY',
    }
}
```

2.3 Setting up models to be tagged

You will need to decide which models and which fields in those models you will want SuperTagging to mark for tagging:

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': True,
    'DEBUG': True,
    'OPEN_CALAIS': {
        'API_KEY': 'YOUR_API_KEY',
    },
    'WATCHED_FIELDS': {
        'stories.story': {
```

```
            'fields':[
                {'name': 'body'},
            ],
        },
    },
}
```

The code above tells SuperTagging to tag the **body** field of model **stories.story**. We can specify any number of fields and models as well.

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': True,
    'DEBUG': True,
    'OPEN_CALAIS': {
        'API_KEY': 'YOUR_API_KEY',
    },
    'WATCHED_FIELDS': {
        'stories.story': {
            'fields':[
                    {'name': 'body'},
                    {'name': 'tease'},
                    {'name': 'kicker'},
            ],
        },
        'media.image': {
            'fields':[
                    {'name': 'caption'},
                    {'name': 'description'},
            ],
        }
    },
}
```

View *WATCHED_FIELDS* for more information.

2.4 Set up automatic processing

Finally, add:

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': True,
    'DEBUG': True,
    'OPEN_CALAIS': {
        'API_KEY': 'YOUR_API_KEY',
    },
    'WATCHED_FIELDS': {
        'stories.story': {
            'fields':[
                {'name': 'body'},
            ],
        },
    },
    'AUTO_PROCESS': True,
}
```

Post save and post delete signals will be connected to the models specified in *WATCHED_FIELDS*. Visit *Settings* to

View the complete list of *Settings*

2.5 Conclusion

That is all that is needed to get SuperTagging to start tagging your data. Upon saving a instance of one of the models specified in *SUPERTAGGING_MODULES*, the field(s) data will be sent to OpenCalais for processing.

Next step: View the *Real World Example* section of how The Washington Times has SuperTagging setup.

CHAPTER

THREE

REAL WORLD EXAMPLE

SuperTagging came about when The Washington Times was looking for a cheap alternative method of tagging its contents. The previous method was the process of sending the content to a similar, but paid service, and that service would return keywords and the content in its marked up state. The content then was saved into our story body field with the links in place. These links went to a section of our site we then called "Themes".

The first thing we wanted to get rid of was the monthly fee we had to paid for the service, we ended up finding OpenCalais, which gave us a free way to tag our content and also be able to easily markup our content with the links as before.

We now have a similar section on our site called Topics which is powered by SuperTagging, and most of our stories have the links to this section as before.

This section of the documentation, we will go through all the pieces of SuperTagging we use here at The Washington Times.

3.1 Our current SuperTagging settings

This first group of settings are the general settings.

```
SUPERTAGGING_DEBUG = False
SUPERTAGGING_ENABLED = True
SUPERTAGGING_AUTO_PROCESS = True
SUPERTAGGING_CALAIS_API_KEY = '...'
SUPERTAGGING_USE_QUEUE = True
```

Explanation:

We have USE_QUEUE set to *True*. Together with AUTO_PROCESS set to *True*, the objects are saved into the *SuperTagQueue* model for later processing. We run the management command provided by SuperTagging every 5 minutes.

This next group is what we call the processing settings for SuperTagging.

```
SUPERTAGGING_PROCESS_RELATIONS = True
SUPERTAGGING_PROCESS_TOPICS = True
SUPERTAGGING_RESOLVE_PROPERTY_KEYS = True
SUPERTAGGING_ONLY_NON_TAGGED_OBJECTS = False

SUPERTAGGING_MIN_RELEVANCE = 200

SUPERTAGGING_USE_FREEBASE = True
```

Explanation:

We process pretty much all OpenCalais gives us, that includes the Events/Facts (relations) and Topics, which are just tags but with no meta data.

We try to convert Calais ID's to tag names (*SUPERTAGGING_RESOLVE_PROPERTY_KEYS* = *True*)

We tag all objects every time they are saved (*SUPERTAGGING_ONLY_NON_TAGGED_OBJECTS* = *False*). This is so we can efficiently markup up the content without worry of data being stale.

We only accept tags that have at least 200 relevance

We use Freebase to disambiguate the tag names.

Next group is the markup settings

```
SUPERTAGGING_MARKUP = True
SUPERTAGGING_MARKUP_EXCLUDES = ['his', 'her', 'he', 'she', 'him',]
SUPERTAGGING_MARKUP_CONTENT_CACHE_TIMEOUT = 3600
SUPERTAGGING_MARKUP_FIELD_SUFFIX = "tagged"
```

Explanation:

We markup all the objects that get processed by OpenCalais, below is an example of content after it has been marked up.

```
<p><a href="/topics/charles-b-rangel/">Mr. Rangel</a>, the longtime top Democrat on the House
<a href="/topics/ways-and-means-committee/">Ways and Means Committee</a> who stepped down under
pressure in March, has been under investigation by the panel for two years. At issue is a
plethora of subjects, including <a href="/topics/charles-b-rangel/">Mr. Rangel</a>'s ownership
of several rent-controlled apartments in New York; his failure to report $75,000 in earnings
on tax returns; and use of his official position to raise money for the
<a href="/topics/charles-b-rangel-center/">Charles B. Rangel Center</a> for {
<a href="/topics/public-service/">Public Service</a> at
<a href="/topics/city-college-of-new-york/">City College of New York</a>.</p>
```

We exclude some markup values OpenCalais returns. OpenCalais returns exactly where, in the content sent over, the reference to the tags is. This will not just be the exact tag name but also references to the tag such as his, her, she. We found that there was just too many links in the content and needed a way to limit it. So combinations of *SUPERTAGGING_MIN_RELEVANCE* setting and *SUPERTAGGING_MARKUP_EXCLUDES* provided just that.

One of the things we did not like from the old tagging service, was that the links were added to the content directly. With SuperTagging the content field of your instance is never changed when your using the markup feature. Instead, another attribute is added to your instance during processing. *SUPERTAGGING_MARKUP_FIELD_SUFFIX* tells what the prefix for that field will be. For example, if we wanted to markup a field named 'content', after *SuperTagging* is done processing, another attribute will be available in that model called 'tagged_content'. This attribute will contain the content in a marked up state. This way, if we ever decided to change the way we markup our data, or change the location of where the links in the content go, we don't need to change the instance directly.

Vist the *Markup* page for more information

This next set of settings we use is to exclude the types (entities) and relation types (events/facts)

```
SUPERTAGGING_TAG_TYPE_EXCLUSIONS = [
    'Anniversary',
    'City',
    #'Company',
    'Continent',
    #'Country',
    'Currency',
    'EmailAddress',
    'EntertainmentAwardEvent',
```

```
    'Facility',
    'FaxNumber',
    #'Holiday',
    'IndustryTerm',
    'MarketIndex',
    'MedicalCondition',
    'MedicalTreatment',
    'Movie',
    'MusicAlbum',
    'MusicGroup',
    'NaturalFeature',
    'OperatingSystem',
    #'Organization',
    #'Person',
    'PhoneNumber',
    'PoliticalEvent',
    'Position',
    'Product',
    'ProgrammingLanguage',
    'ProvinceOrState',
    'PublishedMedium',
    #'RadioProgram',
    'RadioStation',
    'Region',
    #'SportsEvent',
    #'SportsGame',
    #'SportsLeague',
    'Technology',
    #'TVShow',
    'TVStation',
    'URL',
]

SUPERTAGGING_REL_TYPE_EXCLUSIONS = [
   #'Acquisition',
   'Alliance',
   'AnalystEarningsEstimate',
   'AnalystRecommendation',
   #'Arrest',
   #'Bankruptcy',
   'BonusSharesIssuance',
   #'BusinessRelation',
   'Buybacks',
   'CompanyAccountingChange',
   #'CompanyAffiliates',
   'CompanyCompetitor',
   'CompanyCustomer',
   'CompanyEarningsAnnouncement',
   'CompanyEarningsGuidance',
   #'CompanyEmployeesNumber',
   'CompanyExpansion',
   'CompanyForceMajeure',
   #'CompanyFounded',
   'CompanyInvestment',
   'CompanyLaborIssues',
   'CompanyLayoffs',
   'CompanyLegalIssues',
   'CompanyListingChange',
```

```
    'CompanyLocation',
    'CompanyMeeting',
    'CompanyNameChange',
    'CompanyProduct',
    'CompanyReorganization',
    'CompanyRestatement',
    'CompanyTechnology',
    #'CompanyTicker',
    'CompanyUsingProduct',
    'ConferenceCall',
    #'Conviction',
    'CreditRating',
    'DebtFinancing',
    'DelayedFiling',
    'DiplomaticRelations',
    'Dividend',
    'EmploymentChange',
    #'EmploymentRelation',
    'EnvironmentalIssue',
    'Extinction',
    #'FamilyRelation',
    'FDAPhase',
    'Indictment',
    'IPO',
    'JointVenture',
    'ManMadeDisaster',
    'Merger',
    'MovieRelease',
    'MusicAlbumRelease',
    'NaturalDisaster',
    'PatentFiling',
    'PatentIssuance',
    #'PersonAttributes',
    #'PersonCareer',
    'PersonCommunication',
    #'PersonEducation',
    'PersonEmailAddress',
    'PersonRelation',
    'PersonTravel',
    'PoliticalEndorsement',
    #'PoliticalRelationship',
    'PollsResult',
    'ProductIssues',
    'ProductRecall',
    'ProductRelease',
    #'Quotation',
    'SecondaryIssuance',
    'StockSplit',
    'Trial',
    'VotingResult',
]
```

The ones that are commented out are the ones we do want. So so make sense of these to lists we only want the following:

- **Tag Types (Entities)**
  - Company

  - Country
  - Holiday
  - Organization
  - Person
  - RadioProgram
  - SportsEvent
  - SportsGame
  - SportsLeague
  - TVShow
- **Relation Types (Events/Facts)**
  - Acquisition
  - Arrest
  - Bankruptcy
  - BusinessRelation
  - CompanyAffiliates
  - CompanyEmployeesNumber
  - CompanyFounded
  - CompanyTicker
  - Conviction
  - EmploymentRelation
  - FamilyRelation
  - PersonAttributes
  - PersonCareer
  - PersonEducation
  - PoliticalRelationship
  - Quotation

This last section is the models we tag.

```
SUPERTAGGING_MODULES = {
   'stories.story':
        {'fields':(
            {'name': 'body',
             'process_type':'TEXT/HTML'},),
        },
    'massmedia.image':
        {'fields':(
            {'name': 'caption'},)
        },
}
```

Explanation:

This we very basic, since we only tag one field for both our stories and images. One thing that will change in the future for The Washington Times, will be that we will only tag stories with particular set of origins. For example, we currently have the following origins for our stories:

```
STORY_ORIGIN_CHOICES = (
    (0, 'Unknown'),
    (1, 'Admin'),
    (2, 'SaxoTech Editorial'),
    (3, 'SaxoTech Online'),
    (4, 'AP News'),
    (5, 'Bernini'),
    (6, 'AP NetNews'),
)
```

Some of these origins are not used anymore, and we will want to limit which ones we do tag, so eventually the *SUPERTAGGING_MODULES* will look like this:

```
SUPERTAGGING_MODULES = {
   'stories.story':
        {'fields':(
            {'name': 'body',
             'process_type':'TEXT/HTML'},),
         'match_kwargs': {'origin__in': [1,2,3,5]}
        },
    'massmedia.image':
        {'fields':(
            {'name': 'caption'},)
        },
}
```

If this was our current setup, we would not tag stories that had an origin of (0) Unknown, (4) AP News or (6) AP NetNews.

3.2 Showcasing what SuperTagging does

TODO

3.3 Running the process

Last thing is to run the cron job every 5 minutes to process the Queued objects.

```
*/5 *   *   *   *    /path/to/virtualenv/bin/python /path/to/project/manage.py st_process_queue>/dev/
```

CHAPTER

FOUR

API REFERENCE

Contents

4.1 SuperTag

4.1.1 Fields

- **calais_id - Contains the OpenCalais entity ID**
  - CharField
  - Length: 255
  - Unique
- **substitute - Substitute tags in order to have better disambiguation.**
  - ForeignKey to self
  - null=True, blank=True

- **name - The tag name.**
  - CharField
  - Length: 150
- **slug - Slugified name**
  - SlugField
  - Length: 150
- **stype - Tag type as returned by OpenCalais**
  - CharField
  - Length: 100
- **properties - Tag properties as returned by OpenCalais**
  - PickledObjectField
  - null=True, blank=True
- **enabled - Weather or not this tag is used.**
  - BooleanField
  - Default: True

4.1.2 Optional Fields

If *INCLUDE_DISPLAY_FIELDS* is True, these fields will be included with the model.

- **display_name - Name used for display purposes. Since all SuperTag name are lowered when returned from calais, we can**
  - CharField
  - Length: 150
  - null=True, blank=True
- **description - Description of the tag**
  - TextField
  - null=True, blank=True
- **icon - Image field for the tag**
  - ImageField
  - null=True, blank=True
- **related - Manually relating tags**
  - ManyToManyField to self
  - null=True, blank=True

4.1.3 Methods

get_name

Gets the name of the tag, this will try to retrieve the display name first if the display fields are available, if the display fields are not available the normal name will be returned.

has_display_fields

Returns True or False, if the display fields are available.

render

Renders the instance, view *Rendering Items* for more information.

4.2 SuperTagRelation

4.2.1 Fields

- **tag - The associated tag**
  - ForeignKey to *SuperTag*
- **stype - The type of relation**
  - CharField
  - Length: 100
- **name - Name of the relation**
  - CharField
  - Length: 150
- **properties - Relation properties returned by OpenCalais**
  - PickledObjectField
  - null=True, blank=True

4.2.2 Methods

render

Renders the instance, view *Rendering Items* for more information.

4.3 SuperTaggedItem

Generic relation to a *SuperTag*

4.3.1 Fields

- **tag - The associated tag**
  - ForeignKey to *SuperTag*
- **content_type - Content type of an object**
  - ForeignKey to *django.contrib.contenttypes.models.ContentType*
- **object_id - Instance primary key**
  - PositiveIntegerField
- **content_object - Gernric relation**
  - GenericForeignKey to content_type and object_id
- **field - The name of the field this instance refers to**
  - CharField
  - Length: 100
- **process_type - The type used to process the data, "TEXT/HTML", "TEXT/RAW" or "TEXT/XML"**
  - CharField
  - Length: 10
  - null=True, blank=True
- **relevance - The relevance score**
  - IntegerField
  - null=True, blank=True
- **instances - Contains a list of all the tags found in the content.**
  - PickledObjectField
  - null=True, blank=True
- **item_date - Date of the object**
  - DateTimeField
  - null=True, blank=True

4.3.2 Methods

render

Renders the instance, view *Rendering Items* for more information.

4.4 SuperTaggedRelationItem

4.4.1 Fields

- **relation - Associated relation**
  - ForeignKey to *SuperTagRelation*

- **content_type - Content type of an object**
  - ForeignKey to *django.contrib.contenttypes.models.ContentType*
- **object_id - Instance primary key**
  - PositiveIntegerField
- **content_object - Gernric relation**
  - GenericForeignKey to content_type and object_id
- **field - The name of the field this instance refers to**
  - CharField
  - Length: 100
- **process_type - The type used to process the data, "TEXT/HTML", "TEXT/RAW" or "TEXT/XML"**
  - CharField
  - Length: 10
  - null=True, blank=True
- **instances - Contains a list of all the tags found in the content.**
  - PickledObjectField
  - null=True, blank=True
- **item_date - Date of the object**
  - DateTimeField
  - null=True, blank=True

4.4.2 Methods

render

Renders the instance, view *Rendering Items* for more information.

4.5 SuperTagProcessQueue

Holds a generic relation to an object to be processed at a later time, this model is only used when *USE_QUEUE* is set to *True*

4.5.1 Fields

- **content_type - Content type of an object**
  - ForeignKey to *django.contrib.contenttypes.models.ContentType*
- **object_id - Instance primary key**
  - PositiveIntegerField
- **content_object - Gernric relation**

- **locked - Weather the object is being processed**
  - BooleanField
  - Default: False

4.6 Rendering Items

SuperTag, *SuperTaggedItem*, *SuperTagRelation* and *SuperTaggedRelationItem* have a *render* method in order to correctly display its contents.

4.6.1 Template Locations

Default location for these templates are in *supertagging/templates/render*. For each model there is an additional folder:

- SuperTag: "tags/"
- SuperTaggedItem: "tagged_items/"
- SuperTagRelation: "relations/"
- SuperTaggedRelationItem: "tagged_relations/"

For example the default template for a SuperTaggedItem would be "supertagging/templates/render/tagged_items/default.html"

This default template is the last resort, below is a detail list of template paths that will be checked first

1. template argument - this is a full path starting in your templates dir
2. **template_path + *stype* + *app* + *model* + *suffix* - for *SuperTag* and *SuperTagRelation* a type, model, app and suffix will be ad**
   - supertagging/render/tags/<stype>/<app>__<model>__<suffix>.html
   - supertagging/render/tags/people/stories__story__custom.html
3. **template_path + *stype* + *app* + *model* - Same as above but without the suffix**
   - supertagging/render/tags/people/stories__story.html
4. **template_path + *stype* + default + *suffix* - Same as #2 except not *app* and *model***
   - supertagging/render/tags/people/default__custom.html
5. **template_path + *stype* + default - Same as #4 except without the suffix**
   - supertagging/render/tags/people/default.html
6. **template_path + default - the last possible path to look for the template**
   - supertagging/render/tags/default.html

Note: As stated in #2 of the list above, *stype* only applies to *SuperTag* and *SuperTagRelation* since *SuperTaggedItem* and *SuperTaggedRelationItem* doesn't contain the *stype* field. It will simply not be part of the path.

4.6.2 Template Context

SuperTag and *SuperTagRelation* has only it self returned in the context

- **obj** - self

SuperTaggedItem and *SuperTaggedRelationItem* has 2 conext variables

- **obj** - the generic related item
- **content** - self

CHAPTER

FIVE

MARKUP

This is a way to populate your content with extra content in relation to the tags. The most common way would be to replace where the tags are located with links to another section of your site with more information.

5.1 Setup

In the settings you will need to have

```
SUPERTAGGING_SETTINGS = {

    # ... Other settings

    'MARKUP': {
        'ENABLED': True,
    }
}
```

5.2 How It Works

When SuperTagging loads up and markup is enabled, it will add an additional attribute for every field specified in *WATCHED_FIELDS*.

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': True,
    'WATCHED_FIELDS': {
        'stories.story':
            {'fields':[
                {'name': 'body',
                 'markup_handler': 'MyCustomHandler'}]},
        'media.image':
            {'fields':[
                {'name': 'caption'}]},
        'blog.entry':
            {'fields':[
                {'name': 'content'},
                {'name': 'tease',
                 'markup': False}]}
    },

    # ... Other settings
```

```
    'MARKUP': {
        'ENABLED': True,
        'FIELD_SUFFIX': "tagged",
    },
}
```

Lets take the code sample above as an example. We notice that markup is enabled and the prefix for the markup fields is *tagged*. The first module is a **story** model, and the field named **body** is marked to be tagged. It also specifies a custom markup handler, which we'll get to a bit later. The next model is a **image** model and the **caption** field is marked for tagging. The last model is an **entry** model and it has 2 fields marked for tagging, **content** and **tease**, but tease is not to be marked up.

After *SuperTagging* is done loading you will end up with three additional attributes for the three different models.

- **Story model:** `body__tagged`
- **Image model:** `caption__tagged`
- **Entry model:** `content__tagged`

Notice that the a `tease__tagged` does not exist for the **Entry** model because the markup flag for that field is `False`.

5.3 Markup handler

Each field will be assigned a *MarkupHandler* object, which can be found in *supertagging/markup.py* file. This module does all the markup processing for you on the fly. If an error occurs, since the original content is never touched, the original content is returned.

You can create your own custom handler as well.

```
from supertagging.markup import MarkupHandler

class MyCustomHandler(MarkupHandler):
    def handle(self, instance):
        # DO YOUR CUSTOM MARKUP HERE
        return "MARKED UP CONTENT"
```

The `handle` method needs to return a string of the marked up content.

If you want a create a custom handler but use the default markup method, your code might look something like this:

```
from supertagging.markup import MarkupHandler, markup_content

class MyCustomHandler(MarkupHandler):
    def handle(self, instance):
        # DO SOMETHING HERE
        return markup_content(instance, self.field)
```

5.4 Markup Template

markup.html

This template is used to render the tags in a marked up state. Below is the default html rendered.

```
<a href="#">{{ actual_value }}</a>
```

Context

- actual_value - the value of the tag, this might be the same as the tag name or a reference to the tag, IE: 'his', 'her' etc.
- tag - a *SuperTag* instance

5.5 Caching

There is a build-in cache for the markup, since every time we call this new attribute, a couple database calls need to happen to retrieve all the tags and its meta data for an instance.

You can change the default timeout for this cache by changing the following setting

```
SUPERTAGGING_MARKUP_CONTENT_CACHE_TIMEOUT = 3600
```

5.6 Gotchas

In some cases, after enabling markup and successfully tagging an instance the markup does not show up. Two things might cause this, 1 is the cache has not expired and 2 the markup did not validate.

Markup validation happens when the markup field is called and the data retrieved does not match what the instance has stored. This usually means that the instance was edited and the field that gets tagged was changed and it has not been re-processed by OpenCalais.

CHAPTER

SIX

REFERENCE

6.1 Settings

Contents

The default SuperTagging settings are:

```
SUPERTAGGING_SETTINGS = {
    'ENABLED': False,
    'DEBUG': False,
    'WATCHED_FIELDS': {},
    'AUTO_PROCESS': False,
    'ONLY_NON_TAGGED_OBJECTS': False,
    'CONTENTTYPE_NAME_MAPPING': {},
    'INCLUDE_DISPLAY_FIELDS': True,
    'REGISTER_MODELS': True,
    'REMOVE_REL_ON_DISABLE': True,
    'RESOLVE_PROPERTY_KEYS': True,
    'SUBSTITUTE_TAG_UPDATE': True,
```

```
    'USE_QUEUE': False,
    'FILE_STORAGE': 'django.core.files.storage.FileSystemStorage',
    'EXCLUSIONS': {
        'MIN_RELEVANCE': 0,
        'REL_TYPE_EXCLUSIONS': [],
        'TAG_TYPE_EXCLUSIONS': [],
        'TAG_TYPE_QUERY_EXCLUSIONS': []},
    'FREEBASE': {
        'DESCRIPTION_URL': 'http://www.freebase.com/api/trans/raw',
        'ENABLED': False,
        'RETRIEVE_DESCRIPTIONS': False,
        'TYPE_MAPPINGS': {}},
    'MARKUP': {
        'CONTENT_CACHE_TIMEOUT': 3600,
        'ENABLED': False,
        'EXCLUDE': [],
        'FIELD_SUFFIX': 'tagged',
        'MIN_RELEVANCE': 0},
    'OPEN_CALAIS': {
        'API_KEY': '',
        'DEFAULT_PROCESS_TYPE': 'TEXT/RAW',
        'PROCESSING_DIRECTIVES': {
            'calculateRelevanceScore': True,
            'contentType': 'TEXT/RAW',
            'docRDFaccessible': True,
            'enableMetadataType': '',
            'outputFormat': 'application/json',
            'reltagBaseURL': ''},
        'PROCESS_RELATIONS': True,
        'PROCESS_SOCIALTAGS': True,
        'PROCESS_TOPICS': True,
        'USER_DIRECTIVES': {
            'allowDistribution': False,
            'allowSearch': False,
            'externalID': '',
            'submitter': 'python-calais client v.1.5'}},
}
```

6.1.1 ENABLED

Default: `False`

Whether or not SuperTagging is enabled. Will not process any objects if `False`. This allows starting and stopping tag processing while preserving the value of *AUTO_PROCESS*.

6.1.2 DEBUG

Default: `False`

If `True`, errors will fail loudly in order to debug the code.

6.1.3 WATCHED_FIELDS

Default: `{}`

This settings is a dictionary that specifies all the models, fields and options.

The keys of the dictionary are strings in the format `app_name.model_name`. The value of each key is a dictionary, where the fields and other options are specified.

- **fields** - *(Required)* List of dictionaries that specify field names and its options
  - **name** - *(Required)* `String` The name of the field
  - **process_type** - *(Optional)* `String` The process type that OpenCalais should use when tagging the data, possible values are `TEXT/RAW`, `TEXT/HTML`, `TEXT/HTMLRAW`, or `TEXT/XML`. Default is the value of *DEFAULT_PROCESS_TYPE*.
  - **markup** - *(Optional)* `bool` Should SuperTagging automatically markup this field? Default is `False`.
  - **combine_fields** - *(Optional)* `list` A list of two or more fields on the model to combine into one submission to OpenCalais for processing. Markup is not available for these combined fields.
- **match_kwargs** - *(Optional)* `dict` A dictionary of extra query parameters to check when processing instances of the model. Performs an extra `.get(**kwargs)` on the instance to ensure it validates against the extra query parameters.
- **date_field** - *(Optional)* `String` The name of the field to retrieve the instance date. If this is not specified, supertagging will try to retrieve the data from the instance `_meta.get_latest_by` or `_meta.ordering`. This field is saved into `SuperTaggedItem` to allow easy sorting of the items by date.

Here is a complete example:

```
SUPERTAGGING_MODULES = {
    'stories.story': {
        'fields': [{
                'name': 'body',
                'process_type': 'TEXT/HTML',
                'markup': True
            }, {
                'name': 'tease'
            }, {
                'name': 'kicker',
                'markup': True
            }],
        'match_kwargs': {
            'status__in': [1,2,3,],
            'published_date__isnull': False},
        'date_field': 'published_date'
    },
    'media.image': {
        'fields': [{'name': 'caption',
                    'process_type': 'TEXT/HTML',
                    'markup': True}],
        'date_field': 'creation_date'
    }
}
```

6.1.4 INCLUDE_DISPLAY_FIELDS

Default: `True`

Should SuperTagging include three extra fields for display purposes:

- **description** - a text field

- **icon** - a image field
- **related** - a many2many field to 'self' (SuperTag)

6.1.5 AUTO_PROCESS

Default: `False`

If True, will set up post_save and post_delete signals to process the data.

6.1.6 ONLY_NON_TAGGED_OBJECTS

Default: `False`

Used with *AUTO_PROCESS*. If `True`, will only process objects that have not been tagged before. Objects that have tags but need re-processing must be added to the queue manually.

If `False`, process all objects.

6.1.7 RESOLVE_PROPERTY_KEYS

Default: `True`

If `True`, SuperTagging will try resolve the Calais ID to a tag name.

6.1.8 REGISTER_MODELS

Default: `False`

If `True`, an additional attribute will be avilable in a model's instance for easy query related access to SuperTagging.

6.1.9 SUBSTITUTE_TAG_UPDATE

Default: `False`

When `True`, and a substitute is specified in *SuperTag* all associated *SuperTaggedItem* and *SuperTagRelation* will be updated with the new tag.

6.1.10 REMOVE_REL_ON_UPDATE

Default: `False`

If `True`, all content related to a tag is removed (items from models *SuperTaggedItem* and *SuperTaggedRelationItem*.

6.1.11 FILE_STORAGE

Default: `settings.DEFAULT_FILE_STORAGE`

Default file storage used for the icon display field.

6.1.12 USE_QUEUE

Default: `False`

If `True`, use the queuing system. When a object is saved, it will be saved to a queue for later processing. A management command is included for you to process the queue.

If `False`, process the object on save.

6.1.13 CONTENTTYPE_NAME_MAPPING

Default: `{}`

A dict of mapped content type ids to names, used for the views

```
{
    34: 'stories',
    83: 'images',
}
```

Where the key is the content type id and the value is the string used in the url:

This:

/supertagging/tag/barack_obama/**stories/**

/supertagging/tag/barack_obama/**images/**

instead of this:

/supertagging/tag/barack_obama/**34/**

/supertagging/tag/barack_obama/**83/**

This was done in order to make readable urls.

6.1.14 OPEN_CALAIS

DEFAULT_PROCESS_TYPE

Default: `TEXT/RAW`

Tells the default process type for OpenCalais to process the data.

There are four options that can be supplied.

- `TEXT/RAW`
- `TEXT/HTML`
- `TEXT/HTMLRAW`
- `TEXT/XML`

API_KEY

Default: `''`

Your OpenCalais API Key

These next two settings are options for open calais.

USER_DIRECTIVES

Default:

```
{
    "allowDistribution": False,
    "allowSearch": False,
    "externalID": '',
    "submitter": "python-calais client v.1.5",
}
```

View Input Parameters on OpenCalais.com for more information.

PROCESSING_DIRECTIVES

Default:

```
{
    "contentType": "TEXT/RAW",
    "outputFormat": "application/json",
    "reltagBaseURL": '',
    "calculateRelevanceScore": True,
    "enableMetadataType": '',
    "docRDFaccessible": True,
}
```

View Input Parameters on OpenCalais.com for more information.

PROCESS_RELATIONS

Default: `False`

If `True`, save the tag relations (Events/Facts) returned by OpenCalais

PROCESS_TOPICS

Default: `False`

If `True`, save the topics returned by OpenCalais. These will simply be added as tags, but will not include all tag details.

PROCESS_SOCIALTAGS

Default: `False`

If `True`, save the social tags returned by OpenCalais. These will simply be added as tags, but will not include all tag details.

6.1.15 EXCLUSIONS

TAG_TYPE_EXCLUSIONS

Default: `[]`

Tag types as strings to exclude from being added. These tags should be all the "Entities" listed on the following link.

OpenCalais Entities, Events and Facts

REL_TYPE_EXCLUSIONS

Default: `[]`

Same as above but these are the relations and are shown on the following link as "Events and Facts"

OpenCalais Entities, Events and Facts

TAG_TYPE_QUERY_EXCLUSIONS

NOT IMPLEMENTED (YET)

Tags will be saved, but not returned in the queries

MIN_RELEVANCE

Default: `0`

Integer between 0 and 1000, will only save tags that have a higher relevance that this setting.

6.1.16 FREEBASE

ENABLED

Default: `False`

Use Freebase to disambiguate the tags?

TYPE_MAPPINGS

Default: `{}`

For better disambiguation, use this setting to map Calais types to freebase types.

RETRIEVE_DESCRIPTIONS

Default: `False`

If the display fields are enabled, you can have freebase retrieve the description for the tags.

DESCRIPTION_URL

Default: `"http://www.freebase.com/api/trans/raw"`

The first part of the url from where to retrieve the descriptions.

6.1.17 MARKUP

ENABLED

Default: `False`

Is automatic markup of content enabled?

MIN_RELEVANCE

Default: `0`

Integer between 0 and 1000, tells SuperTagging the minimum relevance to use when marking up the content.

FIELD_SUFFIX

Default: `"tagged"`

If markup is enabled, SuperTagging will add a field to the instance with the marked up content, this setting specifies the suffix.

For example: if `'body'` field is marked for tagging, by default a field called `'body__tagged'` will be available in the instance that contains the content with marked up content.

EXCLUDES

Default: `[]`

List of strings of values to exclude from being marked up. For example, OpenCalais returns 'his', 'her', 'him' etc. in reference to a tag.

CONTENT_CACHE_TIMEOUT

Default: `3600`

Cache timeout for the markup content in seconds.

6.2 Template Tags

Contents

Here is the list of current template tags, most of these are tags from Django Tagging with some addtions

---

Note: Tag names have changed slightly, the biggest difference is that now there is "super" prepended to them. This is so we don't clash with a project that uses Django-Tagging and SuperTagging together.

The following "Tags from Django-Tagging" section are a modified version of Django-Tagging template tag documentation

---

6.2.1 Tags from Django-Tagging

The `supertagging.templatetags.supertagging_tags` module defines a number of template tags which may be used to work with tags.

Tag reference

supertags_for_model

Retrieves a list of `SuperTag` objects associated with a given model and stores them in a context variable.

Usage

```
{% supertags_for_model [model] as [varname] %}
```

The model is specified in `[appname].[modelname]` format.

Extended usage

```
{% supertags_for_model [model] as [varname] with counts %}
```

If specified - by providing extra `with counts` arguments - adds a `count` attribute to each tag containing the number of instances of the given model which have been tagged with it.

Examples

```
{% supertags_for_model products.Widget as widget_tags %}
{% supertags_for_model products.Widget as widget_tags with counts %}
```

supertag_cloud_for_model

Retrieves a list of `SuperTag` objects for a given model, with tag cloud attributes set, and stores them in a context variable.

Usage

```
{% supertag_cloud_for_model [model] as [varname] %}
```

The model is specified in `[appname].[modelname]` format.

Extended usage

```
{% supertag_cloud_for_model [model] as [varname] with [options] %}
```

Extra options can be provided after an optional `with` argument, with each option being specified in `[name]=[value]` format. Valid extra options are:

steps Integer. Defines the range of font sizes.

min_count Integer. Defines the minimum number of times a tag must have been used to appear in the cloud.

distribution One of `linear` or `log`. Defines the font-size distribution algorithm to use when generating the tag cloud.

Examples

```
{% supertag_cloud_for_model products.Widget as widget_tags %}
{% supertag_cloud_for_model products.Widget as widget_tags with steps=9 min_count=3 distribution=log
```

supertags_for_object

Retrieves a list of `SuperTag` objects associated with an object and stores them in a context variable.

Usage

```
{% supertags_for_object [object] as [varname] %}
```

Example

```
{% supertags_for_object foo_object as tag_list %}
```

supertagged_objects

Retrieves a list of instances of a given model which are tagged with a given `SuperTag` and stores them in a context variable.

Usage

```
{% supertagged_objects [tag] in [model] as [varname] %}
```

The model is specified in `[appname].[modelname]` format.

The tag must be an instance of a `SuperTag`, not the name of a tag.

Example

```
{% supertagged_objects comedy_tag in tv.Show as comedies %}
```

6.2.2 New Tags for SuperTagging

Below is a list of the new tags that can be used with SuperTagging

Tag reference

relations_for_supertag

Usage

```
{% relations_for_supertag [tag] as [varname] %}
{% relations_for_supertag [tag] as [varname] with type=[TYPE] %}
```

The tag must of an instance of a `SuperTag`, not the name of a tag.

Example

```
{% relations_for_supertag state_tag as relations %}
{% relations_for_supertag state_tag as relations with type=Quotation %}
```

relations_for_object

Useage

```
{% relations_for_object [object] as [varname] %}
{% relations_for_object [object] as [varname] with [type=TYPE]}
```

Example

```
{% relations_for_object story as story_relations %}
{% relations_for_object story as story_relations with type=Quotation %}
```

relations_for

Returns a list of *SuperTagRelation* objects for a tag within a given object.

Useage

```
{% relations_for [tag] in [object] as [varname] %}
```

Example

```
{% relations_for state_tag in obj as obj_relations %}
```

supertag_render

Useage

```
{% supertag_render [SuperTag or SuperTaggedItem or SuperTagRelation or SuperTaggedRelationItem] [with
```

Example

```
{% supertag_render tag %}
{% supertag_render tagged_item with suffix=custom %}
{% supertag_render rel_item with template=myvustomtemplates/supertags/custom.html %}
```

Only suffix OR template can be specified, but not both.

View *Rendering Items* for more information about rendering.

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*