Catalist Documentation

Release 0.0

Rachel Wu, Tony Zhang

January 24, 2016

Contents

Contents:

CHAPTER 1

catalist package

1.1 Submodules

1.2 catalist.api module

exception catalist.api.**InvalidAPIUsage**(*message*, *status_code=None*, *payload=None*)
Bases: exceptions.Exception

A class for exceptions to raise in invalid API usage. Shamelessly pillaged from Flask's documentation

status_code = 400

to_dict()

catalist.api.**add_to_my_lists**()
Add a specified list to "My Lists". POST {

listid: <listid>

}

catalist.api.**autocomplete**()
completes a word fragment with a possible list type usage: POST to this route with {"fragment": myfragment}, response is the list of possible completions of *myfragment* drawn from *autocomplete_dict*

catalist.api.**autocomplete_user**()

catalist.api.**create_list**()
Create a new, empty list and return the assigned listid

catalist.api.**entry_delete**()
Delete an entry from a Catalist. Requires at least edit permission.

usage: POST a JSON associative array as follows: {

listid: <the id of the Catalist>, entryind: <the index of the entry to remove>

}

catalist.api.**entry_title_save**()
AJAXily save the title of an entry. Requires at least edit permission

usage: POST a JS associative array (basically a dict) like so: {

listid: <the list id>, entryind: <index of entry w.r.t. list (0-indexing)>, newvalue: <new entry title>

}

catalist.api.**get_pref**(**args*, ***kwargs*)
Get the preferred theme for the user POST: {

uid: <uid>,

} returns: {

theme: <preferred theme [0, 1, ..]>

}

catalist.api.**handle_invalid_usage**(*error*)

catalist.api.**key_save**()
Save a key. Requires at least edit permission.

POST a JS associative array (basically a dict) like so: {

listid: <the list id>, entryind: <index of entry w.r.t. list (0-indexing)>, index: <index of key-val pair w.r.t. entry>, newvalue: <new value of key>

}

catalist.api.**key_val_save**(*req_form*, *key_or_val*)
Save a key or value in a KVP. Auxillary method for */api/savekey* and */api/savevalue* – captures repetitive code.

catalist.api.**kvp_delete**()
Delete a key-value pair from a Catalist entry. Requires at least edit permission.

usage: POST a JSON associative array as follows: {

listid: <the id of the Catalist>, entryind: <the index of the entry to remove>, index: <the index of the kvp within the entry>

}

catalist.api.**list_delete**()
Delete a Catalist. Requires at least own permission

usage: POST a JSON associative array as follows: {

listid: <the id of the list to be deleted>

}

catalist.api.**list_save**()
Save an entire list. If listid is provided, the list is written onto the referenced list. Otherwise, a new list is created. In both cases the listid to which we saved the list is returned.

usage: {

title: <thetitle>, contents: [

[title, [[attrname, attrval], ...]

] (optionally) , listid: <the listid to save to>

}

Returns: the given or assigned listid

catalist.api.**list_title_save**()
AJAXily save the title of a Catalist ^_^ Requires at least edit permission.

usage: POST a JS assoc array like so: {

listid: <the list id>, newvalue: <our new title>

}

catalist.api.**logged_in**()
Used for .js to call

catalist.api.**make_list**()
Upon making the first edit, an empty list will be created for the insertion of more data

catalist.api.**my_lists_interact**(*listid*, *addQ*)
Add or remove a list with specified listid from "My Lists".

Parameters

- **listid** – the listid of the list
- **addQ** – an integer specifying whether to add or remove: 1 to add, -1 to remove

catalist.api.**permissions_forfeit**()
Forfeit permissions to a list. Effectively sets permission to Catalist.public_level.

POST: { listid: <listid>

}

catalist.api.**permissions_get**()
Get the permission level a user has for a particular list.

usage: POST the following: {

listid: <the listid>

}

returns: {

permission: <the current permission>

}

catalist.api.**permissions_set**()

{ Set permissions for a user. Requires at least own permission

listid: <listid>, target: <username of user to set perms with>, permission: {none | view | edit | own | admin}

}

catalist.api.**public_level_set**()
Set the permission level for a list for the public at-large. Requires own permission.

POST: { listid: <the listid>, permission: {none | view | edit | own | admin}

}

catalist.api.**remove_from_my_lists**()
Remove a specified list from "My Lists". POST {

listid: <listid>

}

catalist.api.**value_save**()
Save the value in a particular key-value pair. Requires at least edit permission.

The API is virtually identical the that of key_save()

catalist.api.**vote**()

Two options: 1. Update the database to incorporate a user's vote on an entry. 2. Find the user's current vote and the current score of the entry. Requires at least view permission

usage: POST the following {

listid: <listid>, entryind: <entryind>, vote: {1 (upvote) | 0 (no vote) |

-1 (downvote) | 100 (get the current vote)}

}

Returns a response with the following forms:

if vote == 100 { current_vote: <the user's current vote>, score: <the entry's current score>

} if vote != 100 {

current_vote: <the vote just made>, score: <the entry's new score>

}

1.3 catalist.database module

class catalist.database.**Catalist**(**args*, ***values*)

Bases: flask_mongoengine.Document

A class for our lists (Catalists :P)

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception Catalist.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

Catalist.**contents**

A ListField designed specially to hold a list of embedded documents to provide additional query helpers.

Note: The only valid list values are subclasses of EmbeddedDocument.

New in version 0.9.

Catalist.**created**

A datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

Catalist.**creator**

A unicode string field.

Catalist.**editors**

A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

Catalist.**id**
A field wrapper around MongoDB's ObjectIds.

Catalist.**last_visited**
A datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use `ComplexDateTimeField` if you need accurate microsecond support.

Catalist.**listid**
A unicode string field.

Catalist.**mylisters**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

Catalist.**objects = [<Catalist: Catalist object>, <Catalist: Catalist object>, <Catalist: Catalist object>, <Catalist: C**

Catalist.**owners**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

Catalist.**public_level**
A unicode string field.

Catalist.**title**
A unicode string field.

Catalist.**viewers**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

class catalist.database.**CatalistEntry**(**args*, ***kwargs*)
Bases: mongoengine.document.EmbeddedDocument

A class for the entries in our Catalists

contents
A ListField designed specially to hold a list of embedded documents to provide additional query helpers.

Note: The only valid list values are subclasses of EmbeddedDocument.

New in version 0.9.

downvoters
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

score
An 32-bit integer field.

title
A unicode string field.

upvoters
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

class catalist.database.**CatalistKVP**(**args*, ***kwargs*)
Bases: mongoengine.document.EmbeddedDocument

A class for individual key-value pairs in our Catalist entries

key
A unicode string field.

kvpid
A unicode string field.

value
A unicode string field.

class catalist.database.**Role**(**args*, ***values*)
Bases: flask_mongoengine.Document, flask_security.core.RoleMixin

A class for user roles (e.g. user, admin, ...)

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception Role.**MultipleObjectsReturned**
Bases: mongoengine.errors.MultipleObjectsReturned

Role.**description**
A unicode string field.

Role.**id**
A field wrapper around MongoDB's ObjectIds.

Role.**name**
A unicode string field.

Role.**objects** = []

class catalist.database.**User**(**args*, ***values*)
Bases: flask_mongoengine.Document, flask_security.core.UserMixin

A class for users. Can have any/none of these attributes.

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception User.**MultipleObjectsReturned**
Bases: mongoengine.errors.MultipleObjectsReturned

User.**acquaintances**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

User.**active**
A boolean field type.

New in version 0.1.2.

User.**anti_my_lists**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

User.**confirmed_at**
A datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

User.**email**
A unicode string field.

User.**firstname**
A unicode string field.

User.**id**
A field wrapper around MongoDB's ObjectIds.

User.**last_active**
A datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use `ComplexDateTimeField` if you need accurate microsecond support.

User.**lastname**
A unicode string field.

User.**my_custom_lists**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

User.**objects = [<User: User object>, <User: User object>, <User: User object>, <User: User object>, <User: User ob**

User.**password**
A unicode string field.

User.**preferred_theme**
An 32-bit integer field.

User.**roles**
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

User.**uid**
A unicode string field.

1.4 catalist.errorviews module

catalist.errorviews.**forbidden**(*e*)

catalist.errorviews.**internal_server_error**(*e*)

catalist.errorviews.**method_not_allowed**(*e*)

catalist.errorviews.**page_gone**(*e*)

catalist.errorviews.**page_not_found**(*e*)

1.5 catalist.permissions module

catalist.permissions.**admin_unames = ['rmwu', 'txz']**
Currently admins are determined by residency on this list. Hacky, I know. .__.

catalist.permissions.**cmp_permission**(*perm1*, *perm2*)
Return a positive/0/negative integer when perm1 >/=/< perm2.

catalist.permissions.**perm_list = ['none', 'view', 'edit', 'own', 'admin']**
A list of all permission levels, from lowest to highest. The levels:

1.none – no permission

2.view – permission to view a list

3.edit – permission to edit a list

4.own – permission to change permission for a list

5.admin – can do anything

catalist.permissions.**query_cur_perm**(*catalist*)
Finds the permission the current user has for list *catalist*

catalist.permissions.**query_permission**(*user*, *catalist*)
Gives the permission level a user has for a list. "None" represents an anonymous user.

1.6 catalist.views module

catalist.views.**about**()
About us.

catalist.views.**get_id**()
Return name of current user

catalist.views.**getlist**(*listid*)
Display a list with given listid from our database.

catalist.views.**human_readable_time_since**(*tiem*)
Give a human-readable representation of time elapsed since a given time

Parameters **tiem** – a `datetime` object representing the given time.

catalist.views.**index**()
Our homepage!

catalist.views.**login**()
Page for user login

catalist.views.**logout**()
Log out the current user, clearing the Remember Me cookie

catalist.views.**preview_list**(*listid*)
Fetch the list with given listid from our database, display with template

catalist.views.**register**()
Page for user registration

catalist.views.**signin**()
Sign the user in, given valid credentials.

catalist.views.**signup**()
Sign the user up, given valid credentials and a username the doesn't already exist in our database.

catalist.views.**userlists**(**args*, ***kwargs*)
A page displaying all lists belonging to the user.

1.7 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

C

M

N

O

P

Q

R

S

T

U

V