

---

# Catalist Documentation

*Release 0.0*

**Rachel Wu, Tony Zhang**

January 24, 2016



<b>1</b>	<b>catalist package</b>	<b>3</b>
1.1	Submodules . . . . .	3
1.2	catalist.api module . . . . .	3
1.3	catalist.database module . . . . .	6
1.4	catalist.errorviews module . . . . .	10
1.5	catalist.permissions module . . . . .	11
1.6	catalist.views module . . . . .	11
1.7	Module contents . . . . .	12
<b>2</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



Contents:



---

## 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.

#### **kvp\_id**

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



---

## Indices and tables

---

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



## C

- `catalist`, [12](#)
- `catalist.api`, [3](#)
- `catalist.database`, [6](#)
- `catalist.errorviews`, [10](#)
- `catalist.permissions`, [11](#)
- `catalist.views`, [11](#)



## A

about() (in module catalyst.views), 11  
acquaintances (catalist.database.User attribute), 9  
active (catalist.database.User attribute), 9  
add\_to\_my\_lists() (in module catalyst.api), 3  
admin\_unames (in module catalyst.permissions), 11  
anti\_my\_lists (catalist.database.User attribute), 9  
autocomplete() (in module catalyst.api), 3  
autocomplete\_user() (in module catalyst.api), 3

## C

Catalist (class in catalyst.database), 6  
catalist (module), 12  
catalist.api (module), 3  
catalist.database (module), 6  
Catalist.DoesNotExist, 6  
catalist.errorviews (module), 10  
Catalist.MultipleObjectsReturned, 6  
catalist.permissions (module), 11  
catalist.views (module), 11  
CatalistEntry (class in catalyst.database), 7  
CatalistKVP (class in catalyst.database), 8  
cmp\_permission() (in module catalyst.permissions), 11  
confirmed\_at (catalist.database.User attribute), 9  
contents (catalist.database.Catalist attribute), 6  
contents (catalist.database.CatalistEntry attribute), 7  
create\_list() (in module catalyst.api), 3  
created (catalist.database.Catalist attribute), 6  
creator (catalist.database.Catalist attribute), 6

## D

description (catalist.database.Role attribute), 8  
downvoters (catalist.database.CatalistEntry attribute), 8

## E

editors (catalist.database.Catalist attribute), 6  
email (catalist.database.User attribute), 9  
entry\_delete() (in module catalyst.api), 3  
entry\_title\_save() (in module catalyst.api), 3

## F

firstname (catalist.database.User attribute), 9  
forbidden() (in module catalyst.errorviews), 10

## G

get\_id() (in module catalyst.views), 11  
get\_pref() (in module catalyst.api), 3  
getlist() (in module catalyst.views), 11

## H

handle\_invalid\_usage() (in module catalyst.api), 4  
human\_readable\_time\_since() (in module catalyst.views),  
11

## I

id (catalist.database.Catalist attribute), 7  
id (catalist.database.Role attribute), 8  
id (catalist.database.User attribute), 9  
index() (in module catalyst.views), 11  
internal\_server\_error() (in module catalyst.errorviews), 10  
InvalidAPIUsage, 3

## K

key (catalist.database.CatalistKVP attribute), 8  
key\_save() (in module catalyst.api), 4  
key\_val\_save() (in module catalyst.api), 4  
kvp\_delete() (in module catalyst.api), 4  
kvpid (catalist.database.CatalistKVP attribute), 8

## L

last\_active (catalist.database.User attribute), 9  
last\_visited (catalist.database.Catalist attribute), 7  
lastname (catalist.database.User attribute), 10  
list\_delete() (in module catalyst.api), 4  
list\_save() (in module catalyst.api), 4  
list\_title\_save() (in module catalyst.api), 4  
listid (catalist.database.Catalist attribute), 7  
logged\_in() (in module catalyst.api), 5  
login() (in module catalyst.views), 11  
logout() (in module catalyst.views), 11

## M

`make_list()` (in module `catalist.api`), 5  
`method_not_allowed()` (in module `catalist.errorviews`), 10  
`my_custom_lists` (`catalist.database.User` attribute), 10  
`my_lists_interact()` (in module `catalist.api`), 5  
`mylisters` (`catalist.database.Catalist` attribute), 7

## N

`name` (`catalist.database.Role` attribute), 9

## O

`objects` (`catalist.database.Catalist` attribute), 7  
`objects` (`catalist.database.Role` attribute), 9  
`objects` (`catalist.database.User` attribute), 10  
`owners` (`catalist.database.Catalist` attribute), 7

## P

`page_gone()` (in module `catalist.errorviews`), 10  
`page_not_found()` (in module `catalist.errorviews`), 10  
`password` (`catalist.database.User` attribute), 10  
`perm_list` (in module `catalist.permissions`), 11  
`permissions_forfeit()` (in module `catalist.api`), 5  
`permissions_get()` (in module `catalist.api`), 5  
`permissions_set()` (in module `catalist.api`), 5  
`preferred_theme` (`catalist.database.User` attribute), 10  
`preview_list()` (in module `catalist.views`), 11  
`public_level` (`catalist.database.Catalist` attribute), 7  
`public_level_set()` (in module `catalist.api`), 5

## Q

`query_cur_perm()` (in module `catalist.permissions`), 11  
`query_permission()` (in module `catalist.permissions`), 11

## R

`register()` (in module `catalist.views`), 11  
`remove_from_my_lists()` (in module `catalist.api`), 5  
`Role` (class in `catalist.database`), 8  
`Role.DoesNotExist`, 8  
`Role.MultipleObjectsReturned`, 8  
`roles` (`catalist.database.User` attribute), 10

## S

`score` (`catalist.database.CatalistEntry` attribute), 8  
`signin()` (in module `catalist.views`), 11  
`signup()` (in module `catalist.views`), 11  
`status_code` (`catalist.api.InvalidAPIUsage` attribute), 3

## T

`title` (`catalist.database.Catalist` attribute), 7  
`title` (`catalist.database.CatalistEntry` attribute), 8  
`to_dict()` (`catalist.api.InvalidAPIUsage` method), 3

## U

`uid` (`catalist.database.User` attribute), 10  
`upvoters` (`catalist.database.CatalistEntry` attribute), 8  
`User` (class in `catalist.database`), 9  
`User.DoesNotExist`, 9  
`User.MultipleObjectsReturned`, 9  
`userlists()` (in module `catalist.views`), 12

## V

`value` (`catalist.database.CatalistKVP` attribute), 8  
`value_save()` (in module `catalist.api`), 5  
`viewers` (`catalist.database.Catalist` attribute), 7  
`vote()` (in module `catalist.api`), 5