Catalist Documentation

Release 0.0

Rachel Wu, Tony Zhang

January 18, 2016

CONTENTS

Contents:

class routes.**Catalist**(**args*, ***values*)
A class for our lists (Catalists :P)

class routes.**CatalistEntry**(**args*, ***kwargs*)
A class for the entries in our Catalists

class routes.**CatalistKVP**(**args*, ***kwargs*)
A class for individual key-value pairs in our Catalist entries

exception routes.**InvalidAPIUsage**(*message*, *status_code=None*, *payload=None*)
A class for exceptions to raise in invalid API usage. Shamelessly pillaged from Flask's documentation

class routes.**Role**(**args*, ***values*)
A class for user roles (e.g. user, admin, ...)

class routes.**User**(**args*, ***values*)
A class for users. Can have any/none of these attributes.

routes.**about**()
About us.

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

routes.**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*

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

routes.**create_list**()
Create a new list and return the assigned listid

Returns: the assigned listid

routes.**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>

}

routes.**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>

}

routes.**get_id**()
Return name of current user

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

routes.**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.

routes.**index**()
Our homepage!

routes.**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>

}

routes.**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>

}

routes.**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>

}

routes.**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

routes.**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>

}

routes.**login**()
Page for user login

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

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

routes.**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

routes.**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>

}

routes.**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}

}

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

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

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

routes.**register**()
Page for user registration

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

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

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

routes.**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()

routes.**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>, userid: <userid>, 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>

}

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

r