ImageProcessorS18 Documentation

Release v1.0.0

Harvey Shi, Edward Liang, Michelle Wei

May 01, 2018

Contents:

1 actions module 1

2 database module 3

3 images module 7

4 main module 9

5 segment module 11

6 Indices and tables 13

Python Module Index 15

CHAPTER 1

actions module

actions.**act_download**(*request*)
Handles download request for images

Parameters **request** – json request from client

Returns b64 image string of processed image

actions.**act_list**(*username*)
Lists the original and processed images for a user

Parameters **username** – client username

Returns json including uuid's of original and processed images

actions.**act_login**(*request*)
Authenticates login with email and password

Parameters **request** – json request from client

Returns json of jwt

actions.**act_process**(*request*)
Processes the original image that has been uploaded

Parameters **request** – request from client

Returns uuid of processed image

actions.**act_upload**(*request*)
Uploads original user image

Parameters **request** – request from client

Returns uuid of uploaded image

CHAPTER 2

database module

class database.**User**(**args*, ***kwargs*)

Bases: pymodm.base.models.MongoModel

exception DoesNotExist

Bases: pymodm.errors.DoesNotExist

exception MultipleObjectsReturned

Bases: pymodm.errors.MultipleObjectsReturned

objects

The default manager used for MongoModel instances.

This implementation of BaseManager uses QuerySet as its QuerySet class.

This Manager class (accessed via the objects attribute on a MongoModel) is used by default for all MongoModel classes, unless another Manager instance is supplied as an attribute within the MongoModel definition.

Managers have two primary functions:

1. Construct QuerySet instances for use when querying or working with MongoModel instances in bulk.
2. Define collection-level functionality that can be reused across different MongoModel types.

If you created a custom QuerySet that makes certain queries easier, for example, you will need to create a custom Manager type that returns this queryset using the from_queryset() method:

```
class UserQuerySet(QuerySet):
    def active(self):
        '''Return only active users.'''
        return self.raw({"active": True})

class User(MongoModel):
    active = fields.BooleanField()
    # Add our custom Manager.
    users = Manager.from_queryset(UserQuerySet)
```

In the above example, we added a *users* attribute on *User* so that we can use the *active* method on our new QuerySet type:

```
active_users = User.users.active()
```

If we wanted every method on the QuerySet to examine active users *only*, we can do that by customizing the Manager itself:

```
class UserManager(Manager):
    def get_queryset(self):
        # Override get_queryset, so that every QuerySet created will
        # have this filter applied.
        return super(UserManager, self).get_queryset().raw(
            {"active": True})

class User(MongoModel):
    active = fields.BooleanField()
    users = UserManager()

active_users = User.users.all()
```

original_image
A field that stores unicode strings.

password
A field that stores unicode strings.

processed_image
A field that stores unicode strings.

username
A field that stores email addresses.

database.**add_user**(*username*, *password*)
Creates new user if user does not exist in the mongo database

Parameters

- **username** – user email as string type which serves as user id
- **password** – user password as string type

Returns updates user information in mongo database

database.**delete_image**(*name*)
Deletes image stored in server :param name: name (uuid) of an image stored in the VM server

database.**delete_user**(*username*)
Deletes user from mongo database :param username: user email as string type which serves as user id

database.**get_original_image**(*username*)
Gets the original image uuid for a user :param username: user email as string type which serves as user id :returns: uuid of user's original image as a string

database.**get_processed_image**(*username*)
Gets the processed image uuid for a user :param username: user email as string type which serves as user id :returns: uuid (UUID4) of user's processed image as a string

database.**get_user**(*username*)
Gets user by unique username :param username: user email as string type which serves as user id :returns: user information

database.**login_user**(*username*, *password*)

Returns true if user exists and has the correct password :param username: user email as string type which serves as user id :param password: user password as string type :returns: True if password is correct, False if incorrect

database.**remove_images**(*username*)

Removes all images associated with a user :param username: user email as string type which serves as user id

database.**save_original_image_uuid**(*username*, *uuid*)

Updates existing user by adding the uuid of a user-uploaded image :param username: user email as string type which serves as user id :param uuid: UUID4 of user-uploaded image :returns: adds uuid of user-uploaded image to mongo database

database.**save_processed_image_uuid**(*username*, *uuid*)

Updates existing user by adding the uuid of the processed image :param username: user email as string type which serves as user id :param uuid: UUID4 of processed image :returns: adds uuid of processed image to mongo database

CHAPTER 3

images module

images.**get_image_as_b64**(*uuid*, *filetype='png'*)
Gets b64 image string by uuid

Parameters

- **uuid** – uuid of image
- **filetype** – file type to output, options are jpeg, png, or gif

Returns b64 string of image

images.**get_image_by_uuid**(*uuid*)
Retrieves uint array of image by its uuid

Parameters **uuid** – UUID of image as string

Returns grayscale image array

images.**save_image**(*img_str*)
Converts image string to binary and saves onto drive

Parameters **img_str** – base64 image string

Returns uuid of image

images.**save_image_from_arr**(*img_arr*)
Converts uint array to png file (intermediary format stored on server)

Parameters **img_arr** – uint array of image

Returns uuid of image

CHAPTER 4

main module

CHAPTER 5

segment module

segment.**segment**(*uuid*)

Segments image with input uuid, saves processed image to server and returns its uuid

param uuid uuid of original image

returns uuid of processed image, saves b64 string of image on server

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

actions, 1

d

database, 3

i

images, 7

s

segment, 11

Index

A

act_download() (in module actions), 1
act_list() (in module actions), 1
act_login() (in module actions), 1
act_process() (in module actions), 1
act_upload() (in module actions), 1
actions (module), 1
add_user() (in module database), 4

D

database (module), 3
delete_image() (in module database), 4
delete_user() (in module database), 4

G

get_image_as_b64() (in module images), 7
get_image_by_uuid() (in module images), 7
get_original_image() (in module database), 4
get_processed_image() (in module database), 4
get_user() (in module database), 4

I

images (module), 7

L

login_user() (in module database), 4

O

objects (database.User attribute), 3
original_image (database.User attribute), 4

P

password (database.User attribute), 4
processed_image (database.User attribute), 4

R

remove_images() (in module database), 5

S

save_image() (in module images), 7
save_image_from_arr() (in module images), 7
save_original_image_uuid() (in module database), 5
save_processed_image_uuid() (in module database), 5
segment (module), 11
segment() (in module segment), 11

U

User (class in database), 3
User.DoesNotExist, 3
User.MultipleObjectsReturned, 3
username (database.User attribute), 4