Flask-Avatars Documentation

Release 0.2.0

Grey Li

Jan 20, 2021

Contents

All avatar generators in one place.

CHAPTER 1

Installation

```
$ pip install flask-avatars
```

CHAPTER 2

Initialization

The extension needs to be initialized in the usual way before it can be used:

```
from flask_avatars import Avatars

app = Flask(__name__)
avatars = Avatars(app)
```

CHAPTER 3

Configuration

The configuration options available were listed below:

Configuration	Default Value	Description
AVATARS_GRAVATAR_DEFAULT	identicon	Gravatar default avatar type
AVATARS_SAVE_PATH	None	The path where avatar save
AVATARS_SIZE_TUPLE	(30, 60, 150)	The avatar size tuple in a format of `(small, medium, large)`, used when generate identicon avatar
AVATARS_IDENTICON_COLS	7	The cols of identicon avatar block
AVATARS_IDENTICON_ROWS	7	The ros of identicon avatar block
AVATARS_IDENTICON_BG	None	The back ground color of identicaon avatar, pass RGB tuple (for example `(125, 125, 125)` `). Default (``None`) to use random color
AVATARS_CROP_BASE_WIDTH	500	The display width of crop image
AVATARS_CROP_INIT_POS	(0, 0)	The initial position of cop box, a tuple of (x, y), default to left top corner
AVATARS_CROP_INIT_SIZE	None	The initial size of crop box, default to `AVATARS_SIZE_TUP LE[0]`
AVATARS_CROP_MIN_SIZE	None	The min size of crop box, default to no limit
AVATARS_CROP_PREVIEW_SIZE	None	The size of preview box, default to `AVATARS_SIZE_TUP LE[1]`
AVATARS_SERVE_LOCAL	False	Load Jcrop resources from local (built-in), default to use CDN

CHAPTER 4

Avatars

Flask-Avatars provide a `avatars` object in template context, you can use it to get avatar URL.

4.1 Gravatar

You can use `avatars.gravatar()` to get an avatar URL provided by Gravatar, pass the email hash:

```
<img src="{{ avatars.gravatar(email_hash) }}">
```

You can get email hash like this:

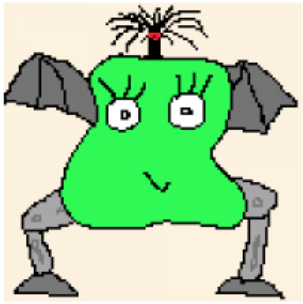
```
import hashlib

avatar_hash = hashlib.md5(my_email.lower().encode('utf-8')).hexdigest()
```

avatars.gravatar(hash)

- size: default to 100
- rating: default to 'g'
- default: default to 'identicon'

{{ avatars.gravatar(email_hash) }}: 

{{ avatars.gravatar(email_hash, size='200', default='monsterid') }}: 

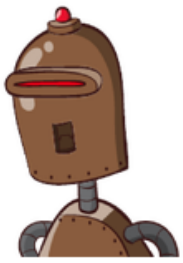
4.2 Robohash

Robohash provide random robot avatar, you can use `avatars.robohash()` to get the avatar URL, pass a random text:

```
<img src="{{ avatars.robohash(some_text) }}">
```

avatars.robohash(text)

- size: default to 200

{{ avatars.robohash('jack') }}: 

{{ avatars.robohash('peter', size='200') }}: 

4.3 Social Media Avatar by Avatars.io

Avatars.io let you use your social media's avatar (Twitter, Facebook or Instagram), you can use `avatars.social_media()` to get the avatar URL, pass your username on target social media:

```
<img src="{{ avatars.social_media(username) }}">
```

Default to use Twitter, use `platform` to change it:

```
<img src="{{ avatars.social_media(username, platform='facebook') }}">
```

avatars.social_media(username)

- platform: default to 'twitter', one of twitter, facebook, instagram, gravatar
- size: default to 'medium', one of 'small', 'medium', 'large'

{{ avatars.social_media('realDonaldTrump') }}: 

{{ avatars.social_media('stefsunyanzi', platform='instagram', size='large') }}:

4.4 Default Avatar

Flask-Avatars provide a default avatar with three size, use `avatars.default()` to get the URL:

```
<img src="{{ avatars.default() }}">
```

You can use `size` to change size (one of `s`, `m` and `l`), for example:

```
<img src="{{ avatars.default(size='s') }}">
```

avatars.default()

- size: default to 'm', one of 's', 'm', 'l'

{{ avatars.default() }}: 

{{ avatars.default(size='l') }}: 

{{ avatars.default(size='s') }}: 

4.5 Identicon Generatation

Flask-Avatars provide a `Identicon` class to generate identicon avatar, most of the code was based on randomavatar. First, you need set configuration variable `AVATARS_SAVE_PATH` to tell Flask-Avatars the path to save generated avatars. Generally speaking, we will generate avavar when the user record was created, so the best place to generate avatar is in user database model class:

```
class User(db.Model):
    avatar_s = db.Column(db.String(64))
    avatar_m = db.Column(db.String(64))
    avatar_l = db.Column(db.String(64))

    def __init__():
        generate_avatar()

    def generate_avatar(self):
        avatar = Identicon()
        filenames = avatar.generate(text=self.username)
        self.avatar_s = filenames[0]
        self.avatar_m = filenames[1]
```

(continues on next page)

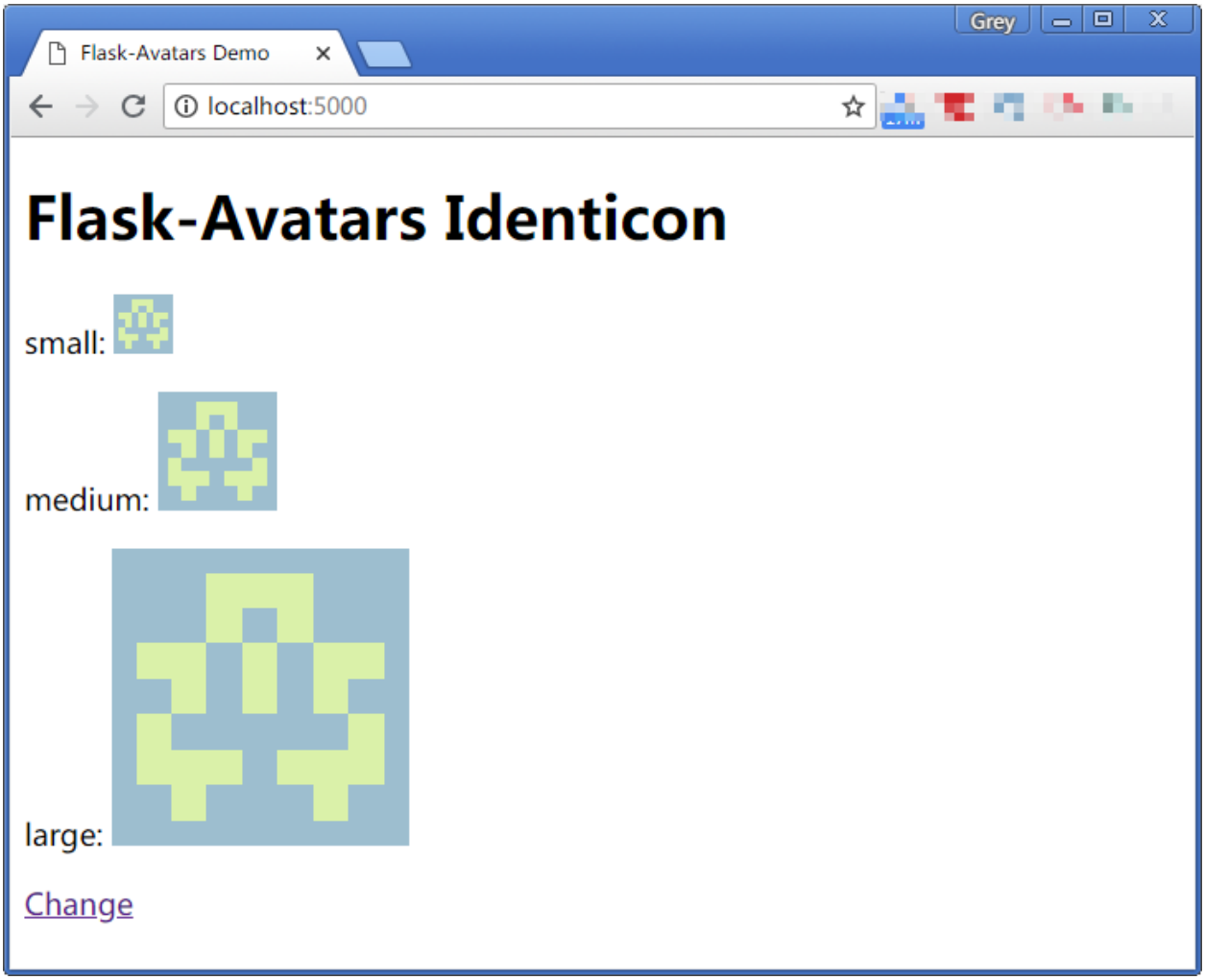
(continued from previous page)

```
        self.avatar_l = filenames[2]
        db.session.commit()
```

Then create a view to serve avatar image like this:

```
from flask import send_form_directory, current_app

@app.route('/avatars/<path:filename>')
def get_avatar(filename):
    return send_from_directory(current_app.config['AVATARS_SAVE_PATH'], filename)
```



CHAPTER 5

Avatar Crop

Flask-Avatars add support avatar crop based on Jcrop.

5.1 Step 1: Upload

The first step is to let user upload the raw image, so we need to create a form in HTML. **upload.html**

```
<form method="post" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit">
</form>
```

If you use Flask-WTF, you can create a form like this:

```
from flask_wtf.file import FileField, FileAllowed, FileRequired

class UploadAvatarForm(FlaskForm):
    image = FileField('Upload (<=3M)', validators=[
        FileRequired(),
        FileAllowed(['jpg', 'png'], 'The file format should be .jpg or .png.')
    ])
    submit = SubmitField()
```

When the user click the submit button, we save the file with `avatars.save_avatar()`:

```
app.config['AVATARS_SAVE_PATH'] = os.path.join(basedir, 'avatars')

# serve avatar image
@app.route('/avatars/<path:filename>')
def get_avatar(filename):
    return send_from_directory(app.config['AVATARS_SAVE_PATH'], filename)
```

(continues on next page)

(continued from previous page)

```
@app.route('/', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files.get('file')
        raw_filename = avatars.save_avatar(f)
        session['raw_filename'] = raw_filename  # you will need to store this␣
↪filename in database in reality
        return redirect(url_for('crop'))
    return render_template('upload.html')
```

5.2 Step 2: Crop

Now we create a crop route to render crop page:

```
@app.route('/crop', methods=['GET', 'POST'])
def crop():
    if request.method == 'POST':
        ...
    return render_template('crop.html')
```

Here is the content of crop.html:

```
<head>
    <meta charset="UTF-8">
    <title>Flask-Avatars Demo</title>
    {{ avatars.jcrop_css() }}  <!-- include jcrop css -->
    <style>
        <!-- some css to make a better preview window -->
    </style>
</head>
<body>
    <h1>Step 2: Crop</h1>
    {{ avatars.crop_box('get_avatar', session['raw_filename']) }}  <!-- crop window --
↪>
    {{ avatars.preview_box('get_avatar', session['raw_filename']) }}  <!-- preview␣
↪widow -->
    <form method="post">
        <input type="hidden" id="x" name="x">
        <input type="hidden" id="y" name="y">
        <input type="hidden" id="w" name="w">
        <input type="hidden" id="h" name="h">
        <input type="submit" value="Crop!">
    </form>
    {{ avatars.jcrop_js() }}  <!-- include jcrop javascript -->
    {{ avatars.init_jcrop() }}  <!-- init jcrop -->
</body>
```

Note the form we created to save crop position data, the four input's name and id must be `x`, `y`, `w`, `h`.

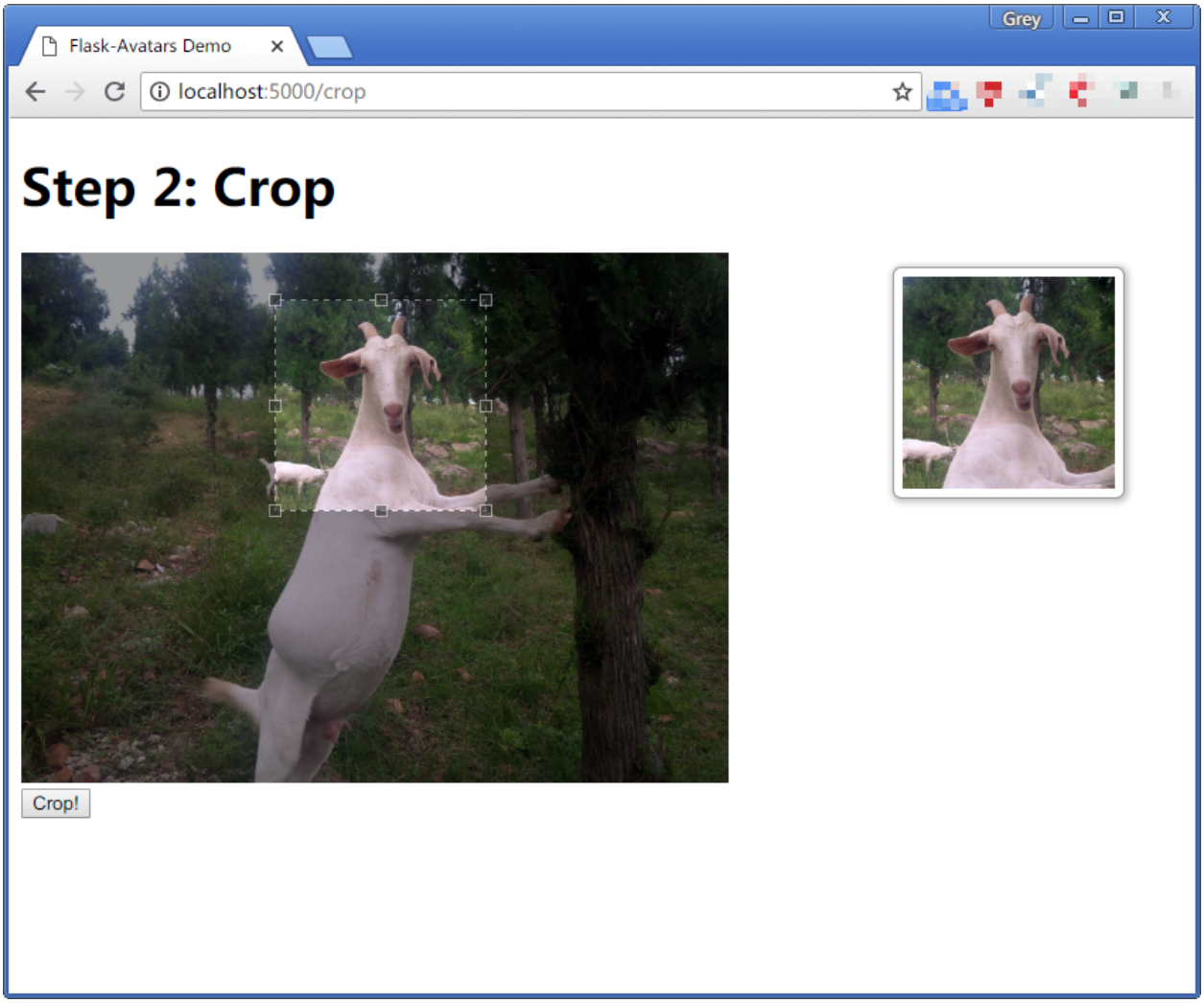
If you use Flask-WTF/WTForms, you can create a form class like this:

```
class CropAvatarForm(FlaskForm):
    x = HiddenField()
    y = HiddenField()
```

(continues on next page)

(continued from previous page)

```
    w = HiddenField()
    h = HiddenField()
    submit = SubmitField('Crop')
```



5.3 Step 3: Save

When the use click the crop button, we can handle the real crop work behind the screen:

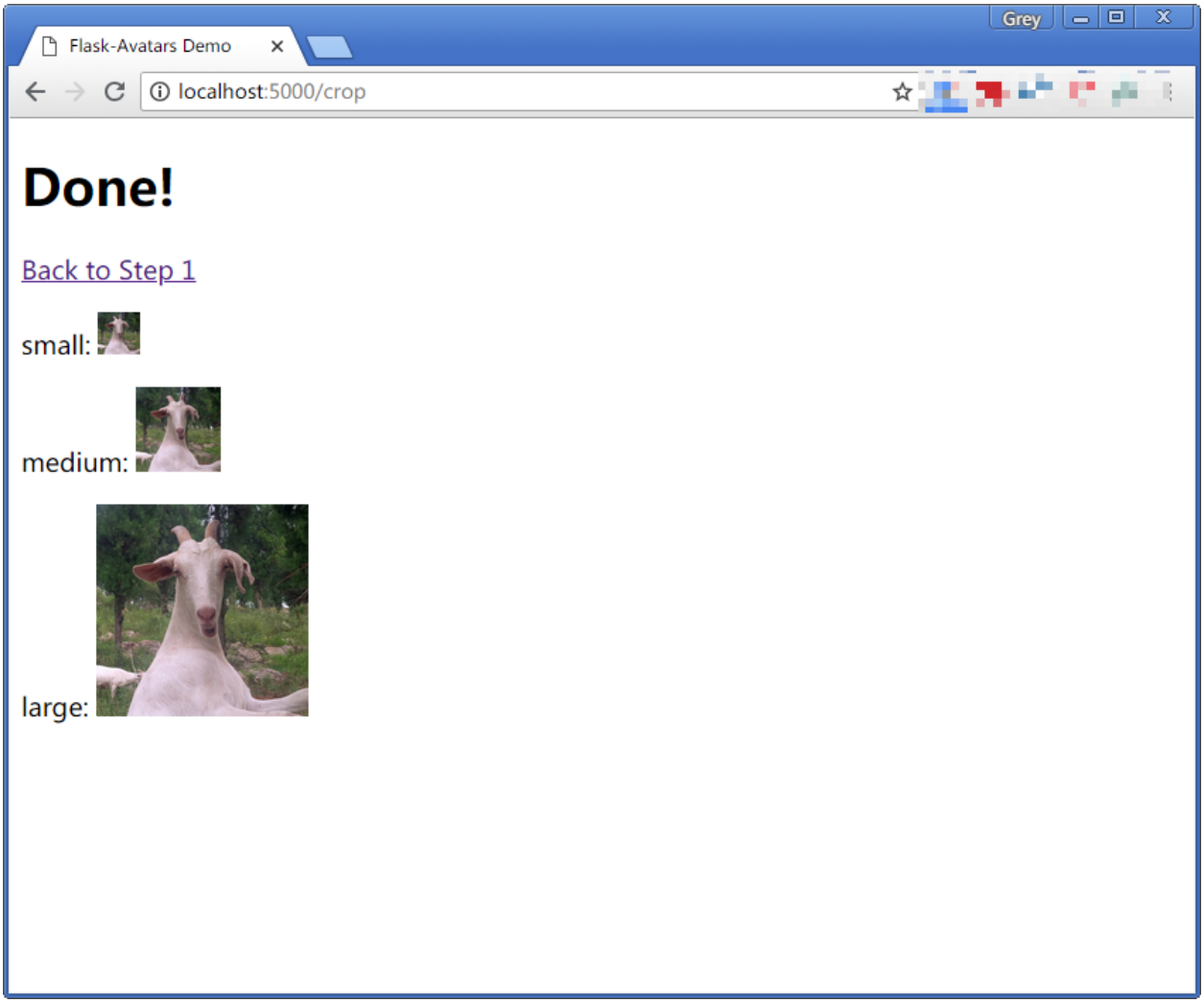
```
@app.route('/crop', methods=['GET', 'POST'])
def crop():
    if request.method == 'POST':
        x = request.form.get('x')
        y = request.form.get('y')
        w = request.form.get('w')
        h = request.form.get('h')
        filenames = avatars.crop_avatar(session['raw_filename'], x, y, w, h)
        url_s = url_for('get_avatar', filename=filenames[0])
        url_m = url_for('get_avatar', filename=filenames[1])
```

(continues on next page)

(continued from previous page)

```
        url_l = url_for('get_avatar', filename=filenames[2])
        return render_template('done.html', url_s=url_s, url_m=url_m, url_l=url_l)
    return render_template('crop.html')
```

`avatars.crop_avatar()` return the crop files name in a tuple `(filename_s, filename_m, filename_l)`, you may need to store it in database.



CHAPTER 6

Example Applications

Currently, we have three examples:

- examples/basic
- examples/identicon
- examples/crop

You can run the example applications in this way:

```
$ git clone https://github.com/greyli/flask-avatars.git
$ cd flask-avatars/examples
$ pip install flask flask-avatars
$ cd basic
$ flask run
```

CHAPTER 7

Development

We welcome all kinds of contributions. You can run test like this:

```
$ python setup.py test
```

CHAPTER 8

Authors

Maintainer: Grey Li

See also the list of contributors who participated in this project.

CHAPTER 9

License

CHAPTER 10

API

10.1 Avatars object in template

class flask_avatars.**_Avatars**

static crop_box(*endpoint=None*, *filename=None*)

Create a crop box.

Parameters

- **endpoint** – The endpoint of view function that serve avatar image file.
- **filename** – The filename of the image that need to be crop.

static default(*size='m'*)

Return built-in default avatar.

Parameters **size** – The size of avatar, one of s, m, l.

Returns Default avatar URL

static gravatar(*hash*, *size=100*, *rating='g'*, *default='identicon'*, *include_extension=False*, *force_default=False*)

Pass email hash, return Gravatar URL. You can get email hash like this:

```
import hashlib
avatar_hash = hashlib.md5(email.lower().encode('utf-8')).hexdigest()
```

Visit https://en.gravatar.com/site/implement/images/ for more information.

Parameters

- **hash** – The email hash used to generate avatar URL.
- **size** – The size of the avatar, default to 100 pixel.
- **rating** – The rating of the avatar, default to `g`

- **default** – The type of default avatar, default to `identicon`.
- **include_extension** – Append a '.jpg' extension at the end of URL, default to `False`.
- **force_default** – Force to use default avatar, default to `False`.

static init_jcrop(*min_size=None*)
Initialize jcrop.

Parameters **min_size** – The minimal size of crop area.

static jcrop_css(*css_url=None*)
Load jcrop css file.

Parameters **css_url** – The custom CSS URL.

static jcrop_js(*js_url=None*, *with_jquery=True*)
Load jcrop Javascript file.

Parameters

- **js_url** – The custom JavaScript URL.
- **with_jquery** – Include jQuery or not, default to `True`.

static preview_box(*endpoint=None*, *filename=None*)
Create a preview box.

Parameters

- **endpoint** – The endpoint of view function that serve avatar image file.
- **filename** – The filename of the image that need to be crop.

static robohash(*text*, *size=200*)
Pass text, return Robohash-style avatar (robot). Visit https://robohash.org/ for more information.

Parameters

- **text** – The text used to generate avatar.
- **size** – The size of the avatar, default to 200 pixel.

static social_media(*username*, *platform='twitter'*, *size='medium'*)
Return avatar URL at social media. Visit https://avatars.io for more information.

Parameters

- **username** – The username of the social media.
- **platform** – One of facebook, instagram, twitter, gravatar.
- **size** – The size of avatar, one of small, medium and large.

10.2 Avatars object in Python

class `flask_avatars`.**Avatars**(*app=None*)

crop_avatar(*filename*, *x*, *y*, *w*, *h*, *uuid_filename=True*)
Crop avatar with given size, return a list of file name: [filename_s, filename_m, filename_l].

Parameters

- **filename** – The raw image's filename.
- **x** – The x-pos to start crop.
- **y** – The y-pos to start crop.
- **w** – The crop width.
- **h** – The crop height.

resize_avatar(*img*, *base_width*)

Resize an avatar.

Parameters

- **img** – The image that needs to be resize.
- **base_width** – The width of output image.

save_avatar(*image*)

Save an avatar as raw image, return new filename.

Parameters **image** – The image that needs to be saved.

10.3 Identicon

class flask_avatars.identicon.**Identicon**(*rows=None*, *cols=None*, *bg_color=None*)

__init__(*rows=None*, *cols=None*, *bg_color=None*)

Generate identicon image.

Parameters

- **rows** – The row of pixels in avatar.
- **columns** – The column of pixels in avatar.
- **bg_color** – Backgroud color, pass RGB tuple, for example: (125, 125, 125). Set it to `None` to use random color.

generate(*text*)

Generate and save avatars, return a list of file name: [filename_s, filename_m, filename_l].

Parameters **text** – The text used to generate image.

CHAPTER 11

ChangeLog

11.1 0.2.2

Release date: 2019/3/4

- Fix `TypeError` when crop avatar without upload the raw image (use default avatar).

11.2 0.2.1

Release date: 2018/8/10

- Add a proper documentation.
- Built-in resources behaviour will not based on `FLASK_ENV`.

11.3 0.2.0

Release date: 2018/7/21

- Add three example applications.
- `avatars.jcrop_js()` now default to include jQuery (`with_jquery=True`).
- Built-in resources will be used when `FLASK_ENV` set to `development`.

11.4 0.1.0

Release date: 2018/6/19

Initialize release.

Python Module Index

f

Index

Symbols

A

C

D

F

G

I

J

P

R

S