Flask-Avatars

`Release 0.2.0`
`Grey Li`

安装

```
pip install flask-avatars
```

扩展在使用前需要以通常的方式初始化

```
from flask_avatars import Avatars

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

下面列出了可用的配置选项

配置	默认值	描述
AVATARS_GRAVATAR_DEFAULT	identicon	默认头像类型
AVATARS_SAVE_PATH	None	头像保存的路径
AVATARS_SIZE_TUPLE	(30,60,150)	头像大小元组，格式为(small, medium, large)，用于生成相同的头像
AVATARS_IDENTICON_COLS	7	identicon头像块的列
AVATARS_IDENTICON_ROWS	7	identicon头像块的行
AVATARS_IDENTICON_BG	N_B	identicaon头像的背景颜色，传递RGB元组(例如(125,125,125)' ').默认(' ' None)使用随机颜色
AVATARS_CROP_BASE_WIDTH	00	裁剪图像的显示宽度
AVATARS_CROP_INIT_POS	(0, 0)	cop框的初始位置，一个(x, y)元组，默认为左上角(0,0)
AVATARS_CROP_INIT_SIZE	None	剪裁框的初始大小，默认为AVATARS_SIZE_TUP LE[0]
AVATARS_CROP_MI_SIZE	None	裁剪框的最小尺寸，默认为无限制
AVATARS_CROP_PREVIEW_SIZE	None	预览框的大小，默认为AVATARS_SIZE_TUP LE[1]
AVATARS_SERVE_LOCAL	False	从本地(内置)加载Jcrop资源，默认使用CDN

`Flask-Avatars`在模板上下文中提供了一个`avatars`对象, 你可以使用它来获取`avatar URL`

Gravatar

可以使用`avatar.Gravatar()`获取由`Gravatar`提供的`avatar URL`, 传递电子邮件散列

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

你可以得到这样的邮件散列:

```
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') }}: 

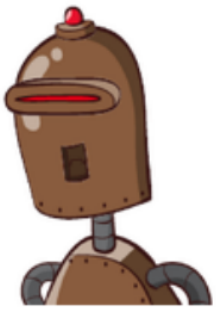
Robohash

`Robohash`提供随机的机器人头像, 你可以使用`avatars.robohash()`获取头像的URL, 传递一个随机的文本:

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

avatars.robohash(text)

- size: default to 200

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

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

Avatars.io的《社交媒体头像》(Social Media Avatar)

`Avatars.io`让你使用你的社交媒体的头像(Twitter、Facebook或Instagram), 你可以使用`avatars.social_media()`获取头像URL,在目标社交媒体上传递你的用户名:

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

默认使用Twitter, 可以使用`platform`更改:

```
1 <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') }}:

默认头像

`flask-avatar`提供了一个三种尺寸的默认头像, 使用`avatar.default()`获取URL:

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

你可以用`size`来改变头像大小 (s, m, l中的一个),例如:

```
1 <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') }}: 

Identicon生成器

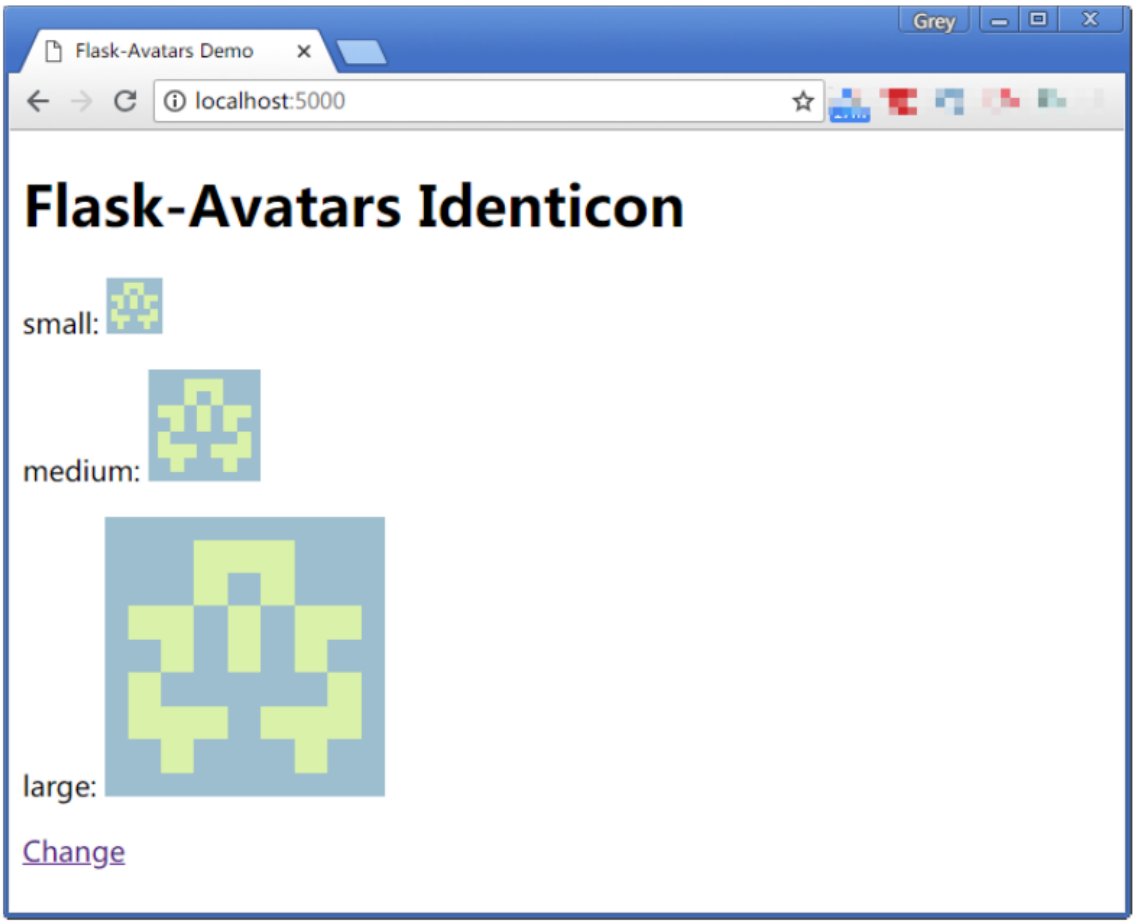
`flask-avatar`提供了一个`Identicon`类来生成`Identicon avatar`,大部分代码基于randomavatar. 首先,需要设置配置变量`AVATARS_SAVE_PATH`来告诉`flask-avatar`保存生成的avatar的路径. 一般来说,我们会在创建用户记录时生成avavar, 因此生成`avatar`的最佳位置是在用户数据库模型类中.

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

然后创建一个视图来服务头像, 如下图所示:

```
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)
```



Avatar Crop

第一步:Upload

第一步是让用户上传原始图像, 所以我们需要在HTML中创建一个表单`upload.html`

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

如果你使用`Flask-WTF`, 你可以创建一个这样的表单:

```
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()
```

当用户点击提交按钮时，我们用`avatar.save_avatar()`保存文件：

```
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)



@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')
```

第二步:Crop

现在我们创建一个Crop路由渲染Crop页面：

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

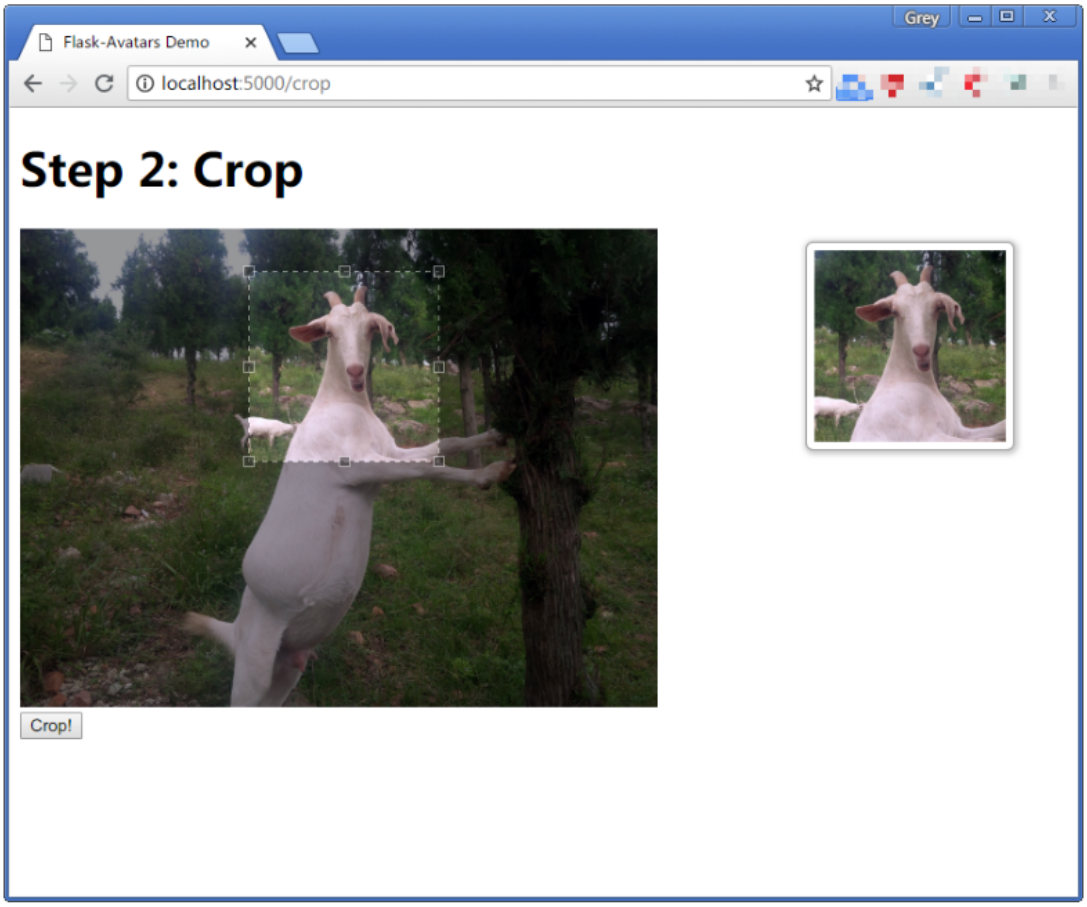
以下是crop.html的内容：

```
<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>
```

注意我们创建的保存作物位置数据的表单，四个输入的名称和id必须是`x`、`y`、`w`、`h`
如果你使用`Flask-WTF`/`WTForms`，你可以创建一个这样的表单类：

```
class CropAvatarForm(FlaskForm):
    x = HiddenField()
    y = HiddenField()
    w = HiddenField()
    h = HiddenField()
    submit = SubmitField('Crop')
```



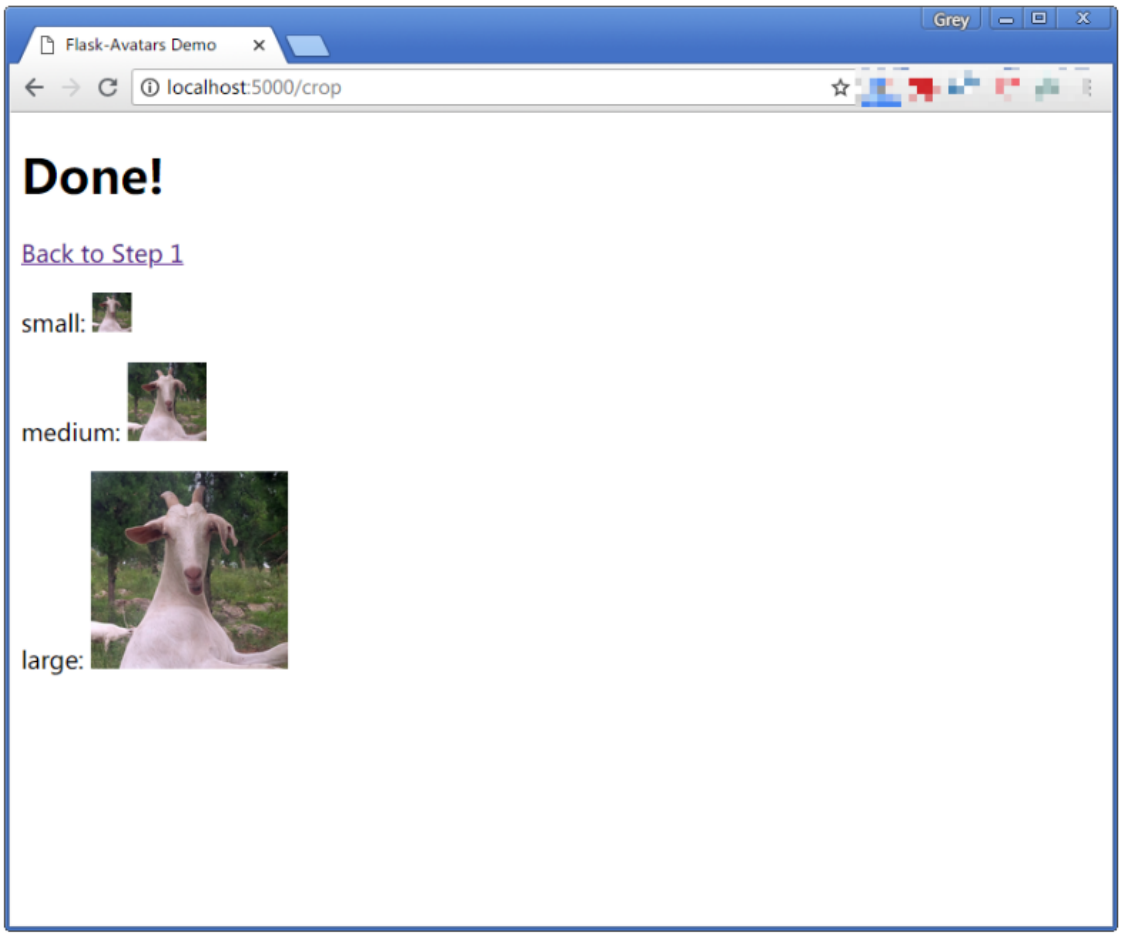
第三步：保存

当使用点击裁剪按钮时，我们可以在屏幕后面处理真正的裁剪工作：

```
@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])
        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')
```

`crop_avatar()`返回一个元组(`filename_s`、`filename_m`、`filename_l`)中的裁剪文件名,您可能需要将它存储在数据库中



目前我们有三个例子:

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

你可以这样运行示例应用程序:

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

API

Avatars object in template

`class flask_avatars._Avatars`

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

功能:创建一个裁剪框
参数:

- `endpoint`:服务头像图像文件的视图函数的端点
- `filename`:需要进行裁剪的映像的文件名

`static default(size='m')`

功能:返回内置默认头像
参数:
* `size`:avatar的大小, s m l中的一个
返回:默认的 avatar URL

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

功能:通过电子邮件散列, 返回Gravatar URL. 你可以像下面这样得到这样的邮件散列:

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

参数:

- `hash`:用于生成头像URL的电子邮件散列.
- `size`:头像的大小，默认为100像素.
- `rating`:avatar的等级，default to g
- `default`:默认的avatar类型, default to identicon.
- `include_extension`:在URL的末尾追加一个'.jpg '扩展名, default to False.
- `force_default`:强制使用默认头像, default to False.

`static init_jcrop(min_size=None)`

功能:初始化jcrop
参数:
* `min_size`:crop面积的最小值

`static jcrop_css(css_url=None)`

功能:加载 jcrop css文件
参数:
`css_url`:自定义CSS URL

`static jcrop_js(js_url=None, with_jquery=True)`

功能:加载 jcrop Javascrip文件
参数:

* `js_url`:自定义的js URL
* `with_jquery`:是否包含jquery,默认为`True`

`static preview_box(endpoint=None, filename=None)`

功能:创建一个预览框
参数:

- `endpoint`:服务头像图像文件的视图函数的端点
- `filename`:需要进行裁剪的映像的文件名

`static robohash(text, size=200)`

功能:传递文本, 返回Robohash-style avatar(robot).更多信息请访问https://robohash.org/
参数:
* `text`:用于生成头像的文本
* `size`:头像大小, 默认为200像素

`static social_media(username, platform='twitter', size='medium')`

功能:在社交媒体上返回头像URL. 访问https://avatars.io获取更多信息
参数:

- `username`:社交媒体的用户名
- `platform`:facebook, instagram, twitter, gravatar其中之一
- `size`: avatar的尺寸, `small`, `medium`, `large` 其中之一

Avatars object in Python

`class flask_avatars.Avatars(app=None)`

`crop_avatar(filename, x, y, w, h)`

功能:裁剪指定大小的头像, 返回文件名列表:`[filename_s, filename_m, filename_l]`
参数:
* `filename`:原始图像的文件名
* `x`:开始剪裁的x坐标
* `y`:开始剪裁的y坐标
* `w`:剪裁的宽度
* `h`:剪裁的高度

`resize_avatar(img, base_width)`

功能:重新定义头像大小
参数:
* `img`:需要被resize的图片
* `base_width`:输出图像的宽度

`save_avatar(image)`

功能:保存头像图片, 返回一个新文件名
参数:
`image`:需要被保存的图像

Identicon

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

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

功能:生成identicon图像
参数:
* `rows`:头像宽度
* `columns`:头像高度
* `bg_color`:背景色, 通过RGB元组, 例如:(125,125,125). 将其设置为`None`以使用随机颜色

`generate(text)`

功能:生成并保存头像, 返回文件名列表:`[filename_s, filename_m, filename_l]`
参数:
* `text`:用于生成图像的文本