django-avatar Documentation

Release 2.0

django-avatar developers

Feb 23, 2023

Contents

1 Installation 3

2 Usage 5

3 Template tags and filter 7

4 Global Settings 9

5 Management Commands 13

Index 15

Django-avatar is a reusable application for handling user avatars. It has the ability to default to avatars provided by third party services (like Gravatar or Facebook) if no avatar is found for a certain user. Django-avatar automatically generates thumbnails and stores them to your default file storage backend for retrieval later.

CHAPTER 1

Installation

If you have pip installed, you can simply run the following command to install django-avatar:

```
pip install django-avatar
```

Included with this application is a file named `setup.py`. It's possible to use this file to install this application to your system, by invoking the following command:

```
python setup.py install
```

Once that's done, you should be able to begin using django-avatar at will.

CHAPTER 2

Usage

To integrate `django-avatar` with your site, there are relatively few things that are required. A minimal integration can work like this:

1. List this application in the `INSTALLED_APPS` portion of your settings file. Your settings file will look something like:

```
INSTALLED_APPS = (
    # ...
    'avatar',
)
```

2. Migrate your database:

```
python manage.py migrate
```

3. Add the avatar urls to the end of your root urlconf. Your urlconf will look something like:

```
urlpatterns = [
    # ...
    path('avatar/', include('avatar.urls')),
]
```

4. Somewhere in your template navigation scheme, link to the change avatar page:

```
<a href="{% url 'avatar_change' %}">Change your avatar</a>
```

5. Wherever you want to display an avatar for a user, first load the avatar template tags:

```
{% load avatar_tags %}
```

   Then, use the `avatar` tag to display an avatar of a default size:

```
{% avatar user %}
```

   Or specify a size (in pixels) explicitly:

```
{% avatar user 65 %}
```

Or specify a width and height (in pixels) explicitly:

```
{% avatar user 65 50 %}
```

Example for customize the attribute of the HTML `img` tag:

```
{% avatar user 65 class="img-circle img-responsive" id="user_avatar" %}
```

CHAPTER 3

Template tags and filter

To begin using these template tags, you must first load the tags into the template rendering system:

```
{% load avatar_tags %}
```

{% avatar_url user [size in pixels] %} Renders the URL of the avatar for the given user. User can be either a `django.contrib.auth.models.User` object instance or a username.

{% avatar user [size in pixels] **kwargs %} Renders an HTML `img` tag for the given user for the specified size. User can be either a `django.contrib.auth.models.User` object instance or a username. The (key, value) pairs in kwargs will be added to `img` tag as its attributes.

{% render_avatar avatar [size in pixels] %} Given an actual `avatar.models.Avatar` object instance, renders an HTML `img` tag to represent that avatar at the requested size.

{{ request.user|has_avatar }} Given a user object returns a boolean if the user has an avatar.

CHAPTER 4

Global Settings

There are a number of settings available to easily customize the avatars that appear on the site. Listed below are those settings:

AVATAR_AUTO_GENERATE_SIZES
An iterable of integers and/or sequences in the format `(width, height)` representing the sizes of avatars to generate on upload. This can save rendering time later on if you pre-generate the resized versions. Defaults to `(80,)`.

AVATAR_CACHE_ENABLED
Set to `False` if you completely disable avatar caching. Defaults to `True`.

AVATAR_DEFAULT_URL
The default URL to default to if the *GravatarAvatarProvider* is not used and there is no `Avatar` instance found in the system for the given user.

AVATAR_EXPOSE_USERNAMES
Puts the User's username field in the URL path when `True`. Set to `False` to use the User's primary key instead, preventing their email from being searchable on the web. Defaults to `False`.

AVATAR_FACEBOOK_GET_ID
A callable or string path to a callable that will return the user's Facebook ID. The callable should take a `User` object and return a string. If you want to use this then make sure you included *FacebookAvatarProvider* in *AVATAR_PROVIDERS*.

AVATAR_GRAVATAR_DEFAULT
A string determining the default Gravatar. Can be a URL to a custom image or a style of Gravatar. Ex. *retro*. All Available options listed in the Gravatar documentation. Defaults to `None`.

AVATAR_GRAVATAR_FORCEDEFAULT
A bool indicating whether or not to always use the default Gravitar. More details can be found in the Gravatar documentation. Defaults to `False`.

AVATAR_GRAVATAR_FIELD
The name of the user's field containing the gravatar email. For example, if you set this to `gravatar` then django-avatar will get the user's gravatar in `user.gravatar`. Defaults to `email`.

AVATAR_MAX_SIZE
File size limit for avatar upload. Default is `1024 * 1024` (1 MB). gravatar in `user.gravatar`.

AVATAR_MAX_AVATARS_PER_USER
The maximum number of avatars each user can have. Default is `42`.

AVATAR_PATH_HANDLER
Path to a method for avatar file path handling. Default is `avatar.models.avatar_path_handler`.

AVATAR_PROVIDERS
Tuple of classes that are tried in the given order for returning avatar URLs. Defaults to:

```
(
    'avatar.providers.PrimaryAvatarProvider',
    'avatar.providers.LibRAvatarProvider',
    'avatar.providers.GravatarAvatarProvider',
    'avatar.providers.DefaultAvatarProvider',
)
```

If you want to implement your own provider, it must provide a class method `get_avatar_url(user, width, height)`.

class avatar.providers.**PrimaryAvatarProvider**
Returns the primary avatar stored for the given user.

class avatar.providers.**GravatarAvatarProvider**
Adds support for the Gravatar service and will always return an avatar URL. If the user has no avatar registered with Gravatar a default will be used (see *AVATAR_GRAVATAR_DEFAULT*).

class avatar.providers.**FacebookAvatarProvider**
Add this provider to *AVATAR_PROVIDERS* in order to add support for profile images from Facebook. Note that you also need to set the *AVATAR_FACEBOOK_GET_ID* setting.

class avatar.providers.**DefaultAvatarProvider**
Provides a fallback avatar defined in *AVATAR_DEFAULT_URL*.

AVATAR_RESIZE_METHOD
The method to use when resizing images, based on the options available in Pillow. Defaults to `Image.LANCZOS`.

AVATAR_STORAGE_DIR
The directory under `MEDIA_ROOT` to store the images. If using a non-filesystem storage device, this will simply be appended to the beginning of the file name. Defaults to `avatars`. Pillow. Defaults to `Image.ANTIALIAS`.

AVATAR_THUMB_FORMAT
The file format of thumbnails, based on the options available in Pillow. Defaults to *PNG*.

AVATAR_THUMB_QUALITY
The quality of thumbnails, between 0 (worst) to 95 (best) or the string "keep" (only JPEG) as provided by Pillow. Defaults to *85*.

AVATAR_THUMB_MODES
A sequence of acceptable modes for thumbnails as provided by Pillow. If the mode of the image is not in the list, the thumbnail will be converted to the first mode in the list. Defaults to *('RGB', 'RGBA')*.

AVATAR_CLEANUP_DELETED
`True` if the avatar image files should be deleted when an avatar is deleted from the database. Defaults to `True`.

AVATAR_ADD_TEMPLATE
Path to the Django template to use for adding a new avatar. Defaults to `avatar/add.html`.

AVATAR_CHANGE_TEMPLATE
Path to the Django template to use for changing a user's avatar. Defaults to `avatar/change.html`.

AVATAR_DELETE_TEMPLATE
Path to the Django template to use for confirming a delete of a user's avatar. Defaults to `avatar/avatar/confirm_delete.html`.

AVATAR_ALLOWED_MIMETYPES
Limit allowed avatar image uploads by their actual content payload and what image codecs we wish to support. This limits website user content site attack vectors against image codec buffer overflow and similar bugs. You must have python-imaging library installed. Suggested safe setting: `("image/png", "image/gif", "image/jpeg")`. When enabled you'll get the following error on the form upload *File content is invalid. Detected: image/tiff Allowed content types are: image/png, image/gif, image/jpg*.

CHAPTER 5

Management Commands

This application does include one management command: `rebuild_avatars`. It takes no arguments and, when run, re-renders all of the thumbnails for all of the avatars for the pixel sizes specified in the *AVATAR_AUTO_GENERATE_SIZES* setting.

Index

A

avatar.providers.DefaultAvatarProvider (*built-in class*), 10
avatar.providers.FacebookAvatarProvider (*built-in class*), 10
avatar.providers.GravatarAvatarProvider (*built-in class*), 10
avatar.providers.PrimaryAvatarProvider (*built-in class*), 10
AVATAR_ADD_TEMPLATE (*built-in variable*), 10
AVATAR_ALLOWED_MIMETYPES (*built-in variable*), 11
AVATAR_AUTO_GENERATE_SIZES (*built-in variable*), 9
AVATAR_CACHE_ENABLED (*built-in variable*), 9
AVATAR_CHANGE_TEMPLATE (*built-in variable*), 10
AVATAR_CLEANUP_DELETED (*built-in variable*), 10
AVATAR_DEFAULT_URL (*built-in variable*), 9
AVATAR_DELETE_TEMPLATE (*built-in variable*), 11
AVATAR_EXPOSE_USERNAMES (*built-in variable*), 9
AVATAR_FACEBOOK_GET_ID (*built-in variable*), 9
AVATAR_GRAVATAR_DEFAULT (*built-in variable*), 9
AVATAR_GRAVATAR_FIELD (*built-in variable*), 9
AVATAR_GRAVATAR_FORCEDEFAULT (*built-in variable*), 9
AVATAR_MAX_AVATARS_PER_USER (*built-in variable*), 10
AVATAR_MAX_SIZE (*built-in variable*), 9
AVATAR_PATH_HANDLER (*built-in variable*), 10
AVATAR_PROVIDERS (*built-in variable*), 10
AVATAR_RESIZE_METHOD (*built-in variable*), 10
AVATAR_STORAGE_DIR (*built-in variable*), 10
AVATAR_THUMB_FORMAT (*built-in variable*), 10
AVATAR_THUMB_MODES (*built-in variable*), 10
AVATAR_THUMB_QUALITY (*built-in variable*), 10