django-avatar Documentation

Release 2.0

django-avatar developers

Jul 05, 2017

Contents

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 = [
    # ...
    (r'^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 %}
```

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 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 `True`.

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).

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.GravatarAvatarProvider',
    'avatar.providers.DefaultAvatarProvider',
)
```

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

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 PIL. Defaults to `Image.ANTIALIAS`.

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`.

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`.

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